```
**********************************************************
   28039 Wed Aug 13 19:51:23 2014
new/exception_lists/packaging
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 #
  23 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
  25 # Copyright 2012 OmniTI Computer Consulting, Inc.  All rights reserved.
  26 #

  28 #
  29 # Exception List for validate_pkg
  30 #

  32 #
  33 # The following entries are built in the /proto area
  34 # but not included in any packages - this is intentional.
  35 #
  36 usr/include/auth_list.h
  37 usr/include/bsm/audit_door_infc.h
  38 usr/include/bsm/audit_private.h
  39 usr/include/bsm/devalloc.h
  40 usr/include/getxby_door.h
  41 usr/include/passwdutil.h
  42 usr/include/priv_utils.h
  43 usr/include/rpcsvc/daemon_utils.h
  44 usr/include/rpcsvc/svc_dg_priv.h
  45 usr/include/security/pam_impl.h
  46 usr/include/sys/clock_impl.h
  47 usr/include/sys/ieeefp.h
  48 usr/include/sys/winlockio.h
  49 usr/include/scsi/plugins/ses/vendor/sun_impl.h
  50 #
  51 # Private/Internal libraries of the Cryptographic Framework.
  52 #
  53 lib/libkcfd.so
  54 lib/llib-lelfsign
  55 lib/llib-lelfsign.ln
  56 lib/llib-lkcfd
  57 lib/llib-lkcfd.ln
  58 usr/include/libelfsign.h
  59 usr/lib/llib-lsoftcrypto
  60 usr/lib/llib-lsoftcrypto.ln
  61 usr/lib/amd64/llib-lsoftcrypto.ln       i386
```



```
  62 usr/lib/sparcv9/llib-lsoftcrypto.ln     sparc

  64 #
  65 # The following files are used by the DHCP service, the
  66 # standalone's DHCP implementation, and the kernel (nfs_dlboot).
  67 # They contain interfaces which are currently private.
  68 #
  69 usr/include/dhcp_svc_confkey.h
  70 usr/include/dhcp_svc_confopt.h
  71 usr/include/dhcp_svc_private.h
  72 usr/include/dhcp_symbol.h
  73 usr/include/sys/sunos_dhcp_class.h
  74 usr/lib/libdhcpsvc.so
  75 usr/lib/llib-ldhcpsvc
  76 usr/lib/llib-ldhcpsvc.ln
  77 #
  78 # Private MAC driver header files
  79 #
  80 usr/include/inet/iptun.h
  81 usr/include/sys/aggr_impl.h
  82 usr/include/sys/aggr.h
  83 usr/include/sys/dld_impl.h
  84 usr/include/sys/dld_ioc.h
  85 usr/include/sys/dls_impl.h
  86 usr/include/sys/dls.h
  87 usr/include/sys/mac_client_impl.h
  88 usr/include/sys/mac_client.h
  89 usr/include/sys/mac_flow_impl.h
  90 usr/include/sys/mac_impl.h
  91 usr/include/sys/mac_soft_ring.h
  92 usr/include/sys/mac_stat.h
  93 #
  94 # Private GLDv3 userland libraries and headers
  95 #
  96 usr/include/libdladm_impl.h
  97 usr/include/libdlaggr.h
  98 usr/include/libdlether.h
  99 usr/include/libdlflow_impl.h
 100 usr/include/libdlflow.h
 101 usr/include/libdliptun.h
 102 usr/include/libdlmgmt.h
 103 usr/include/libdlsim.h
 104 usr/include/libdlstat.h
 105 usr/include/libdlvnic.h
 106 usr/include/libdlwlan_impl.h
 107 usr/include/libdlwlan.h
 108 #
 109 # Virtual Network Interface Card (VNIC)
 110 #
 111 usr/include/sys/vnic.h
 112 usr/include/sys/vnic_impl.h
 113 #
 114 # Private libipadm lint library and header files
 115 #
 116 usr/include/ipadm_ipmgmt.h
 117 usr/include/ipadm_ndpd.h
 118 usr/include/libipadm.h
 119 lib/llib-lipadm
 120 lib/llib-lipadm.ln
 121 lib/libipadm.so
 122 #
 123 # Private libsocket header file
 124 #
 125 usr/include/libsocket_priv.h
 126 #
 127 # IKE and IPsec support library exceptions.  The IKE support
```

```
# library contains exclusively private interfaces, as does
# libipsecutil.  My apologies for the glut of header files here.
#
usr/include/errfp.h
usr/include/ikedoor.h
usr/include/ipsec_util.h
usr/lib/libike.so
usr/lib/amd64/libike.so                 i386
usr/lib/sparcv9/libike.so               sparc
usr/lib/libipsecutil.so
usr/lib/amd64/libipsecutil.so           i386
usr/lib/sparcv9/libipsecutil.so         sparc
usr/lib/llib-like
usr/lib/llib-like.ln
usr/lib/amd64/llib-like.ln              i386
usr/lib/sparcv9/llib-like.ln            sparc
usr/lib/llib-lipsecutil
usr/lib/llib-lipsecutil.ln
usr/lib/amd64/llib-lipsecutil.ln        i386
usr/lib/sparcv9/llib-lipsecutil.ln      sparc
#
usr/include/inet/ip_impl.h
usr/include/inet/ip_ndp.h
usr/include/inet/ip2mac_impl.h
usr/include/inet/ip2mac.h
usr/include/inet/rawip_impl.h
usr/include/inet/tcp_impl.h
usr/include/inet/udp_impl.h
usr/include/libmail.h
usr/include/libnwam_priv.h
usr/include/protocols/ripngd.h
usr/include/s_string.h
usr/include/sys/logindmux_impl.h
usr/include/sys/vgareg.h
#
# Some IPsec headers can't be shipped lest we hit export controls...
#
usr/include/inet/ipsec_impl.h
usr/include/inet/ipsec_info.h
usr/include/inet/ipsecah.h
usr/include/inet/ipsecesp.h
usr/include/inet/keysock.h
usr/include/inet/sadb.h
usr/include/sys/sha1_consts.h
usr/include/sys/sha2_consts.h
#
#
# Filtering out directories not shipped
#
usr/4lib                                i386
#
# These files contain definitions shared privately between the kernel
# and libc.  There is no reason for them to be part of a package that
# a customer should ever see.  They are installed in the proto area by
# the uts build because libc and and other components, like truss, are
# dependent upon their contents and should not have their own copies.
#
usr/include/sys/libc_kernel.h
usr/include/sys/synch32.h
#
# These files are installed in the proto area by the build of libproc for
# the benefit of the builds of cmd/truss, cmd/gcore and cmd/ptools, which
# use libproc as their common process-control library.  These are not
# interfaces for customer use, so the files are excluded from packaging.
#
lib/llib-lproc
```



```
lib/llib-lproc.ln
lib/amd64/llib-lproc.ln         i386
lib/sparcv9/llib-lproc.ln       sparc
usr/include/libproc.h
#
# Private interfaces for libdisasm
#
usr/include/libdisasm.h
usr/lib/llib-ldisasm
usr/lib/llib-ldisasm.ln
usr/lib/amd64/llib-ldisasm.ln           i386
usr/lib/sparcv9/llib-ldisasm.ln         sparc
#
# Private interfaces for libraidcfg
#
usr/include/raidcfg_spi.h
usr/include/raidcfg.h
usr/lib/libraidcfg.so
usr/lib/amd64/libraidcfg.so             i386
usr/lib/sparcv9/libraidcfg.so           sparc
usr/lib/llib-lraidcfg
usr/lib/llib-lraidcfg.ln
usr/lib/amd64/llib-lraidcfg.ln          i386
usr/lib/sparcv9/llib-lraidcfg.ln        sparc
#
# This file is used for private communication between mdb, drv/kmdb, and
# misc/kmdb.  The interfaces described herein are not intended for customer
# use, and are thus excluded from packaging.
#
usr/include/sys/kmdb.h
#
# These files are installed in the proto area by the build of libdhcpagent
# and libdhcputil for the benefit of DHCP-related networking commands such
# as dhcpagent, dhcpinfo, ifconfig, and netstat.  These are not interfaces
# for customer use, so the files are excluded from packaging.
#
lib/libdhcpagent.so
lib/libdhcputil.so
lib/amd64/libdhcputil.so                i386
lib/sparcv9/libdhcputil.so              sparc
lib/llib-ldhcpagent
lib/llib-ldhcpagent.ln
lib/llib-ldhcputil
lib/llib-ldhcputil.ln
lib/amd64/llib-ldhcputil.ln             i386
lib/sparcv9/llib-ldhcputil.ln           sparc
usr/include/dhcp_hostconf.h
usr/include/dhcp_impl.h
usr/include/dhcp_inittab.h
usr/include/dhcp_stable.h
usr/include/dhcp_symbol_common.h
usr/include/dhcpagent_ipc.h
usr/include/dhcpagent_util.h
usr/include/dhcpmsg.h
usr/lib/libdhcpagent.so
usr/lib/libdhcputil.so
usr/lib/amd64/libdhcputil.so            i386
usr/lib/sparcv9/libdhcputil.so          sparc
usr/lib/llib-ldhcpagent
usr/lib/llib-ldhcpagent.ln
usr/lib/llib-ldhcputil
usr/lib/llib-ldhcputil.ln
usr/lib/amd64/llib-ldhcputil.ln         i386
usr/lib/sparcv9/llib-ldhcputil.ln       sparc
#
# These files are installed in the proto area by the build of libinstzones
```

```
# and libpkg
#
usr/lib/llib-linstzones
usr/lib/llib-linstzones.ln
usr/lib/amd64/llib-linstzones.ln        i386
usr/lib/sparcv9/llib-linstzones.ln      sparc
usr/lib/llib-lpkg
usr/lib/llib-lpkg.ln
#
# Don’t ship header files private to libipmp and in.mpathd
#
usr/include/ipmp_query_impl.h
#
# These files are installed in the proto area by the build of libinetsvc,
# an inetd-specific library shared by inetd, inetadm and inetconv. Only
# the shared object is shipped.
#
usr/include/inetsvc.h
usr/lib/libinetsvc.so
usr/lib/llib-linetsvc
usr/lib/llib-linetsvc.ln
#
# These files are installed in the proto area by the build of libinetutil,
# a general purpose library for the benefit of internet utilities. Only
# the shared object is shipped.
#
lib/libinetutil.so
lib/amd64/libinetutil.so                i386
lib/sparcv9/libinetutil.so              sparc
lib/llib-linetutil
lib/llib-linetutil.ln
lib/amd64/llib-linetutil.ln             i386
lib/sparcv9/llib-linetutil.ln           sparc
usr/include/libinetutil.h
usr/include/netinet/inetutil.h
usr/include/ofmt.h
usr/lib/libinetutil.so
usr/lib/amd64/libinetutil.so            i386
usr/lib/sparcv9/libinetutil.so          sparc
usr/lib/llib-linetutil
usr/lib/llib-linetutil.ln
usr/lib/amd64/llib-linetutil.ln         i386
usr/lib/sparcv9/llib-linetutil.ln       sparc
#
# Miscellaneous kernel interfaces or kernel<->user interfaces that are
# consolidation private and we do not want to export at this time.
#
usr/include/sys/cryptmod.h
usr/include/sys/dumpadm.h
usr/include/sys/ontrap.h
usr/include/sys/sysmsg_impl.h
usr/include/sys/vlan.h
#
# These files are installed in the proto area so lvm can use
# them during the build process.
#
lib/llib-lmeta
lib/llib-lmeta.ln
usr/include/sdssc.h
usr/lib/llib-lmeta
usr/lib/llib-lmeta.ln
#
# non-public pci header
#
usr/include/sys/pci_impl.h
usr/include/sys/pci_tools.h
```



```
#
# Exception list for RCM project, included by librcm and rcm_daemon
#
usr/include/librcm_event.h
usr/include/librcm_impl.h
#
# MDB deliverables that are not yet public
#
usr/lib/mdb/proc/mdb_test.so
usr/lib/mdb/proc/sparcv9/mdb_test.so    sparc
#
# SNCA project exception list
#
usr/include/inet/kssl/kssl.h
usr/include/inet/kssl/ksslimpl.h
usr/include/inet/kssl/ksslproto.h
usr/include/inet/nca
#
# these are "removed" from the source product build because the only
# packages that currently deliver them are removed.
# they really should’t be in here.
#
etc/sfw
#
# Entries for the libmech_krb5 symlink, which has been included
# for build purposes only, not delivered to customers.
#
usr/lib/gss/libmech_krb5.so
usr/lib/amd64/gss/libmech_krb5.so       i386
usr/lib/sparcv9/gss/libmech_krb5.so     sparc
usr/lib/libmech_krb5.so
usr/lib/amd64/libmech_krb5.so           i386
usr/lib/sparcv9/libmech_krb5.so         sparc
#
# Entries for headers from efcode project which user does not need to see
#
usr/platform/sun4u/include/sys/fc_plat.h                        sparc
usr/platform/sun4u/include/sys/fcode.h                          sparc
#
# Private net80211 headers
#
usr/include/sys/net80211_crypto.h
usr/include/sys/net80211_ht.h
usr/include/sys/net80211_proto.h
usr/include/sys/net80211.h
#
usr/include/net/wpa.h
#
# PPPoE files not delivered to customers.
#
usr/include/net/pppoe.h
usr/include/net/sppptun.h
#
# Simnet
#
usr/include/net/simnet.h
#
# Bridging internal data structures
#
usr/include/net/bridge_impl.h
#
# User<->kernel interface used by cfgadm/USB only
#
usr/include/sys/usb/hubd/hubd_impl.h
#
# User<->kernel interface used by cfgadm/SATA only
```

```
392 #
393 usr/include/sys/sata/sata_cfgadm.h                      i386
394 #
395 # Private ucred kernel header
396 #
397 usr/include/sys/ucred.h
398 #
399 # Private and/or platform-specific smf(5) files
400 #
401 lib/librestart.so
402 lib/llib-lrestart
403 lib/llib-lrestart.ln
404 lib/amd64/llib-lrestart.ln                              i386
405 lib/sparcv9/llib-lrestart.ln                            sparc
406 usr/include/libcontract_priv.h
407 usr/include/librestart_priv.h
408 usr/include/librestart.h
409 usr/lib/librestart.so
410 usr/lib/sparcv9/librestart.so                           sparc
411 lib/svc/manifest/platform/sun4u                         i386
412 lib/svc/manifest/platform/sun4v                         i386
413 var/svc/manifest/platform/sun4u                         i386
414 var/svc/manifest/platform/sun4v                         i386
415 etc/svc/profile/platform_sun4v.xml                      i386
416 etc/svc/profile/platform_SUNW,SPARC-Enterprise.xml      i386
417 etc/svc/profile/platform_SUNW,Sun-Fire-15000.xml        i386
418 etc/svc/profile/platform_SUNW,Sun-Fire-880.xml         i386
419 etc/svc/profile/platform_SUNW,Sun-Fire-V890.xml        i386
420 etc/svc/profile/platform_SUNW,Sun-Fire.xml            i386
421 etc/svc/profile/platform_SUNW,Ultra-Enterprise-10000.xml  i386
422 etc/svc/profile/platform_SUNW,UltraSPARC-IIe-NetraCT-40.xml  i386
423 etc/svc/profile/platform_SUNW,UltraSPARC-IIe-NetraCT-60.xml  i386
424 etc/svc/profile/platform_SUNW,UltraSPARC-IIi-Netract.xml  i386
425 #
426 # Private libuutil files
427 #
428 lib/libuutil.so
429 lib/llib-luutil
430 lib/llib-luutil.ln
431 lib/sparcv9/llib-luutil.ln              sparc
432 usr/include/libuutil_impl.h
433 usr/lib/libuutil.so
434 usr/lib/sparcv9/libuutil.so             sparc
435 #
436 # Private Multidata file.
437 #
438 usr/include/sys/multidata_impl.h
439 #
440 # The following files are used by wanboot.
441 # They contain interfaces which are currently private.
442 #
443 usr/include/sys/wanboot_impl.h
444 usr/include/wanboot
445 usr/include/wanbootutil.h
446 #
447 # Even though all the objects built under usr/src/stand are later glommed
448 # together into a couple of second-stage boot loaders, we dump the static
449 # archives and lint libraries into $(ROOT)/stand for intermediate use
450 # (e.g., for lint, linking the second-stage boot loaders, ...).  Since
451 # these are merely intermediate objects, they do not need to be packaged.
452 #
453 stand                                   sparc
454 #
455 # Private KCF header files
456 #
457 usr/include/sys/crypto/elfsign.h
```



```
458 usr/include/sys/crypto/impl.h
459 usr/include/sys/crypto/ops_impl.h
460 usr/include/sys/crypto/sched_impl.h
461 #
462 # The following files are installed in the proto area
463 # by the build of libavl (AVL Tree Interface Library).
464 # libavl contains interfaces which are all private interfaces.
465 #
466 lib/libavl.so
467 lib/amd64/libavl.so             i386
468 lib/sparcv9/libavl.so           sparc
469 lib/llib-lavl
470 lib/llib-lavl.ln
471 lib/amd64/llib-lavl.ln          i386
472 lib/sparcv9/llib-lavl.ln        sparc
473 usr/lib/libavl.so
474 usr/lib/amd64/libavl.so         i386
475 usr/lib/sparcv9/libavl.so       sparc
476 usr/lib/llib-lavl
477 usr/lib/llib-lavl.ln
478 usr/lib/amd64/llib-lavl.ln      i386
479 usr/lib/sparcv9/llib-lavl.ln    sparc
480 #
481 # The following files are installed in the proto area
482 # by the build of libcmdutils (Command Utilities Library).
483 # libcmdutils contains interfaces which are all private interfaces.
484 #
485 lib/libcmdutils.so
486 lib/amd64/libcmdutils.so                i386
487 lib/sparcv9/libcmdutils.so              sparc
488 lib/llib-lcmdutils
489 lib/llib-lcmdutils.ln
490 lib/amd64/llib-lcmdutils.ln             i386
491 lib/sparcv9/llib-lcmdutils.ln           sparc
492 usr/include/libcmdutils.h
493 usr/lib/libcmdutils.so
494 usr/lib/amd64/libcmdutils.so            i386
495 usr/lib/sparcv9/libcmdutils.so          sparc
496 usr/lib/llib-lcmdutils
497 usr/lib/llib-lcmdutils.ln
498 usr/lib/amd64/llib-lcmdutils.ln         i386
499 usr/lib/sparcv9/llib-lcmdutils.ln       sparc
500 #
501 # Private interfaces in libsec
502 #
503 usr/include/aclutils.h
504 #
505 # USB skeleton driver stays in sync with the rest of USB but doesn’t ship.
506 #
507 kernel/drv/usbskel              i386
508 kernel/drv/amd64/usbskel        i386
509 kernel/drv/sparcv9/usbskel      sparc
510 kernel/drv/usbskel.conf
511 #
512 # Consolidation and Sun private libdevid interfaces
513 # Public libdevid interfaces provided by devid.h
514 #
515 usr/include/sys/libdevid.h
516 #
517 # The following files are installed in the proto area by the build of
518 # libprtdiag.  libprtdiag contains interfaces which are all private.
519 # Only the shared object is shipped.
520 #
521 usr/platform/sun4u/lib/llib-lprtdiag                            sparc
522 usr/platform/sun4u/lib/llib-lprtdiag.ln                         sparc
523 usr/platform/sun4v/lib/llib-lprtdiag.ln                         sparc
```

```
524 #
525 # The following files are installed in the proto area by the build of
526 # mdesc driver in sun4v. These header files are used on in the build
527 # and do not need to be shipped to customers.
528 #
529 usr/include/sys/mdesc.h                                 sparc
530 usr/include/sys/mdesc_impl.h                            sparc
531 usr/platform/sun4v/include/sys/mach_descrip.h           sparc
532 #
533 # The following files are installed in the proto area by the build of
534 # libpcp.  libpcp contains interfaces which are all private.
535 # Only the shared object is shipped.
536 #
537 usr/platform/sun4v/lib/llib-lpcp.ln                             sparc
538 usr/platform/SUNW,Netra-CP3060/lib/llib-lpcp.ln                 sparc
539 usr/platform/SUNW,Netra-CP3260/lib/llib-lpcp.ln                 sparc
540 usr/platform/SUNW,Netra-T5220/lib/llib-lpcp.ln                  sparc
541 usr/platform/SUNW,Netra-T5440/lib/llib-lpcp.ln                  sparc
542 usr/platform/SUNW,SPARC-Enterprise-T5120/lib/llib-lpcp.ln       sparc
543 usr/platform/SUNW,Sun-Blade-T6300/lib/llib-lpcp.ln            sparc
544 usr/platform/SUNW,Sun-Blade-T6320/lib/llib-lpcp.ln            sparc
545 usr/platform/SUNW,Sun-Fire-T200/lib/llib-lpcp.ln              sparc
546 usr/platform/SUNW,T5140/lib/llib-lpcp.ln                      sparc
547 usr/platform/SUNW,USBRDT-5240/lib/llib-lpcp.ln                sparc
548 #
549 # ZFS internal tools and lint libraries
550 #
551 usr/lib/llib-lzfs_jni
552 usr/lib/llib-lzfs_jni.ln
553 usr/lib/amd64/llib-lzfs_jni.ln          i386
554 usr/lib/sparcv9/llib-lzfs_jni.ln        sparc
555 usr/lib/llib-lzpool
556 usr/lib/llib-lzpool.ln                  i386
557 usr/lib/amd64/llib-lzpool.ln            i386
558 usr/lib/sparcv9/llib-lzpool.ln          sparc
559 #
560 # ZFS JNI headers
561 #
562 usr/include/libzfs_jni_dataset.h
563 usr/include/libzfs_jni_disk.h
564 usr/include/libzfs_jni_diskmgt.h
565 usr/include/libzfs_jni_ipool.h
566 usr/include/libzfs_jni_main.h
567 usr/include/libzfs_jni_pool.h
568 usr/include/libzfs_jni_property.h
569 usr/include/libzfs_jni_util.h
570 #
571 # These files are installed in the proto area for Solaris scsi_vhci driver
572 # (for MPAPI support) and should not be shipped
573 #
574 usr/include/sys/scsi/adapters/mpapi_impl.h
575 usr/include/sys/scsi/adapters/mpapi_scsi_vhci.h
576 #
577 # This library is installed in the proto area by the build of libdisasm, and is
578 # only used when building the KMDB disasm module.
579 #
580 usr/lib/libstanddisasm.so
581 usr/lib/amd64/libstanddisasm.so         i386
582 usr/lib/sparcv9/libstanddisasm.so       sparc
583 #
584 # TSol: tsol doesn't ship lint source, and tsnet isn't for customers at all.
585 #
586 lib/libtsnet.so
587 usr/lib/llib-ltsnet
588 usr/lib/llib-ltsol
589 #
```



```
590 # nss interfaces shared between libnsl and other ON libraries.
591 #
592 usr/include/nss.h
593 #
594 # AT&T AST (ksh93) files which are currently needed only to build OS/Net
595 # (msgcc&co.)
596 # libast
597 usr/lib/libast.so
598 usr/lib/amd64/libast.so                 i386
599 usr/lib/sparcv9/libast.so               sparc
600 usr/lib/llib-last
601 usr/lib/llib-last.ln
602 usr/lib/amd64/llib-last.ln              i386
603 usr/lib/sparcv9/llib-last.ln            sparc
604 # libcmd
605 usr/lib/llib-lcmd
606 usr/lib/llib-lcmd.ln
607 usr/lib/amd64/llib-lcmd.ln              i386
608 usr/lib/sparcv9/llib-lcmd.ln            sparc
609 # libdll
610 usr/lib/libdll.so
611 usr/lib/amd64/libdll.so                 i386
612 usr/lib/sparcv9/libdll.so               sparc
613 usr/lib/llib-ldll
614 usr/lib/llib-ldll.ln
615 usr/lib/amd64/llib-ldll.ln              i386
616 usr/lib/sparcv9/llib-ldll.ln            sparc
617 # libpp (a helper library needed by AST's msgcc)
618 usr/lib/libpp.so
619 usr/lib/llib-lpp
620 usr/lib/llib-lpp.ln
621 usr/lib/locale/C/LC_MESSAGES/libpp
622 # libshell
623 usr/lib/libshell.so
624 usr/lib/amd64/libshell.so               i386
625 usr/lib/sparcv9/libshell.so             sparc
626 usr/lib/llib-lshell
627 usr/lib/llib-lshell.ln
628 usr/lib/amd64/llib-lshell.ln            i386
629 usr/lib/sparcv9/llib-lshell.ln          sparc
630 # libsum
631 usr/lib/libsum.so
632 usr/lib/amd64/libsum.so                 i386
633 usr/lib/sparcv9/libsum.so               sparc
634 usr/lib/llib-lsum
635 usr/lib/llib-lsum.ln
636 usr/lib/amd64/llib-lsum.ln              i386
637 usr/lib/sparcv9/llib-lsum.ln            sparc
638 #
639 # This file is used in ON to build DSCP clients.  It is not for customers.
640 #
641 usr/include/libdscp.h           sparc
642 #
643 # These files are used by the iSCSI Target and the iSCSI Initiator
644 #
645 usr/include/sys/iscsi_protocol.h
646 usr/include/sys/iscsi_authclient.h
647 usr/include/sys/iscsi_authclientglue.h
648 #
649 # These files are used by the COMSTAR iSCSI target port provider
650 #
651 usr/include/sys/idm
652 usr/include/sys/iscsit/chap.h
653 usr/include/sys/iscsit/iscsi_if.h
654 usr/include/sys/iscsit/isns_protocol.h
655 usr/include/sys/iscsit/radius_packet.h
```

```
usr/include/sys/iscsit/radius_protocol.h
#
# libshare is private and the 64-bit sharemgr is not delivered.
#
usr/lib/libshare.so
usr/lib/amd64/libshare.so               i386
usr/lib/sparcv9/libshare.so             sparc
usr/lib/fs/autofs/libshare_autofs.so
usr/lib/fs/autofs/amd64/libshare_autofs.so              i386
usr/lib/fs/autofs/sparcv9/libshare_autofs.so            sparc
usr/lib/fs/nfs/libshare_nfs.so
usr/lib/fs/nfs/amd64/libshare_nfs.so            i386
usr/lib/fs/nfs/sparcv9/libshare_nfs.so          sparc
usr/lib/fs/smb/libshare_smb.so
usr/lib/fs/smb/amd64/libshare_smb.so            i386
usr/lib/fs/smb/sparcv9/libshare_smb.so          sparc
usr/lib/fs/smbfs/libshare_smbfs.so
usr/lib/fs/smbfs/amd64/libshare_smbfs.so        i386
usr/lib/fs/smbfs/sparcv9/libshare_smbfs.so      sparc
usr/include/libshare_impl.h
usr/include/scfutil.h
#
# These files are installed in the proto area by the build of libpri for
# the benefit of the builds of FMA libldom, Zeus, picld plugins, and/or
# other libpri consumers. However, the libpri interfaces are private to
# Sun (Consolidation Private) and not intended for customer use. So these
# files (the symlink and the lint library) are excluded from packaging.
#
usr/lib/libpri.so               sparc
usr/lib/llib-lpri               sparc
usr/lib/llib-lpri.ln            sparc
usr/lib/sparcv9/libpri.so       sparc
usr/lib/sparcv9/llib-lpri.ln    sparc
#
# These files are installed in the proto area by the build of libds for
# the benefit of the builds of sun4v IO FMA and/or other libds
# consumers.  However, the libds interfaces are private to Sun
# (Consolidation Private) and not intended for customer use. So these
# files (the symlink and the lint library) are excluded from packaging.
#
usr/lib/libds.so                        sparc
usr/lib/sparcv9/libds.so                sparc
usr/lib/llib-lds                        sparc
usr/lib/llib-lds.ln                     sparc
usr/lib/sparcv9/llib-lds.ln             sparc
usr/lib/libdscfg.so
usr/lib/llib-ldscfg.ln
usr/platform/sun4v/include/sys/libds.h  sparc
usr/platform/sun4v/include/sys/vlds.h   sparc
#
# Private/Internal u8_textprep header file. Do not ship.
#
usr/include/sys/u8_textprep_data.h
#
# SQLite is private, used by SMF (svc.configd), idmapd and libsmb.
#
usr/include/sqlite
usr/lib/libsqlite-native.o
usr/lib/libsqlite.o
usr/lib/llib-lsqlite.ln
usr/lib/smbsrv/libsqlite.so
#
# Private/Internal kiconv header files. Do not ship.
#
usr/include/sys/kiconv_big5_utf8.h
usr/include/sys/kiconv_cck_common.h
```



```
usr/include/sys/kiconv_cp950hkscs_utf8.h
usr/include/sys/kiconv_emea1.h
usr/include/sys/kiconv_emea2.h
usr/include/sys/kiconv_euckr_utf8.h
usr/include/sys/kiconv_euctw_utf8.h
usr/include/sys/kiconv_gb18030_utf8.h
usr/include/sys/kiconv_gb2312_utf8.h
usr/include/sys/kiconv_hkscs_utf8.h
usr/include/sys/kiconv_ja_jis_to_unicode.h
usr/include/sys/kiconv_ja_unicode_to_jis.h
usr/include/sys/kiconv_ja.h
usr/include/sys/kiconv_ko.h
usr/include/sys/kiconv_latin1.h
usr/include/sys/kiconv_sc.h
usr/include/sys/kiconv_tc.h
usr/include/sys/kiconv_uhc_utf8.h
usr/include/sys/kiconv_utf8_big5.h
usr/include/sys/kiconv_utf8_cp950hkscs.h
usr/include/sys/kiconv_utf8_euckr.h
usr/include/sys/kiconv_utf8_euctw.h
usr/include/sys/kiconv_utf8_gb18030.h
usr/include/sys/kiconv_utf8_gb2312.h
usr/include/sys/kiconv_utf8_hkscs.h
usr/include/sys/kiconv_utf8_uhc.h
#
# At this time, the ttydefs.cleanup file is only useful on sun4u systems
#
etc/flash/postdeployment/ttydefs.cleanup        i386
#
# This header file is shared only between the power commands and
# ppm/srn modules # and should not be in any package
#
usr/include/sys/srn.h
#
# Private/Internal header files of smbsrv. Do not ship.
#
usr/include/smb
usr/include/smbsrv
#
# Private/Internal dtrace scripts of smbsrv. Do not ship.
#
usr/lib/smbsrv/dtrace
#
# Private/Internal (lint) libraries of smbsrv. Do not ship.
#
usr/lib/reparse/llib-lreparse_smb
usr/lib/reparse/llib-lreparse_smb.ln
usr/lib/smbsrv/llib-lmlrpc
usr/lib/smbsrv/llib-lmlrpc.ln
usr/lib/smbsrv/llib-lmlsvc
usr/lib/smbsrv/llib-lmlsvc.ln
usr/lib/smbsrv/llib-lsmb
usr/lib/smbsrv/llib-lsmb.ln
usr/lib/smbsrv/llib-lsmbns
usr/lib/smbsrv/llib-lsmbns.ln
#
#
# Private/Internal 64-bit libraries of smbsrv. Do not ship.
#
usr/lib/smbsrv/amd64                            i386
usr/lib/smbsrv/sparcv9                          sparc

usr/lib/reparse/amd64/libreparse_smb.so         i386
usr/lib/reparse/amd64/libreparse_smb.so.1       i386
usr/lib/reparse/amd64/llib-lreparse_smb.ln      i386
usr/lib/reparse/sparcv9/libreparse_smb.so       sparc
```

```
usr/lib/reparse/sparcv9/libreparse_smb.so.1     sparc
usr/lib/reparse/sparcv9/llib-lreparse_smb.ln    sparc
#
# Private dirent, extended to include flags, for use by SMB server
#
usr/include/sys/extdirent.h
#
# Private header files for vscan service
#
usr/include/libvscan.h
usr/include/sys/vscan.h
#
# libvscan is private
#
usr/lib/vscan/llib-lvscan
usr/lib/vscan/llib-lvscan.ln
#
# i86hvm is not a full platform.  It is just a home for paravirtualized
# drivers.  There is no usr/ component to this sub-platform, but the
# directory is created in the proto area to keep other tools happy.
#
usr/platform/i86hvm                                             i386
#
# Private sdcard framework headers
#
usr/include/sys/sdcard
#
# libsmbfs is private
#
usr/include/netsmb
usr/lib/libsmbfs.so
usr/lib/amd64/libsmbfs.so               i386
usr/lib/sparcv9/libsmbfs.so             sparc
usr/lib/llib-lsmbfs
usr/lib/llib-lsmbfs.ln
usr/lib/amd64/llib-lsmbfs.ln            i386
usr/lib/sparcv9/llib-lsmbfs.ln          sparc
#
# demo & test program for smbfs (private) ACL support
#
usr/lib/fs/smbfs/chacl
usr/lib/fs/smbfs/lsacl
usr/lib/fs/smbfs/testnp
#
# FC related files
kernel/kmdb/fcip                i386
kernel/kmdb/amd64/fcip          i386
kernel/kmdb/sparcv9/fcip        sparc
kernel/kmdb/fcp                 i386
kernel/kmdb/amd64/fcp           i386
kernel/kmdb/sparcv9/fcp         sparc
kernel/kmdb/fctl                i386
kernel/kmdb/amd64/fctl          i386
kernel/kmdb/sparcv9/fctl        sparc
kernel/kmdb/qlc                 i386
kernel/kmdb/amd64/qlc           i386
kernel/kmdb/sparcv9/qlc         sparc
lib/llib-la5k                   sparc
lib/llib-la5k.ln                sparc
lib/sparcv9/llib-la5k.ln        sparc
lib/llib-lg_fc                  sparc
lib/llib-lg_fc.ln               sparc
lib/sparcv9/llib-lg_fc.ln       sparc
usr/include/a_state.h           sparc
usr/include/a5k.h               sparc
usr/include/exec.h              sparc
```



```
usr/include/g_scsi.h            sparc
usr/include/g_state.h           sparc
usr/include/gfc.h               sparc
usr/include/l_common.h          sparc
usr/include/l_error.h           sparc
usr/include/rom.h               sparc
usr/include/stgcom.h            sparc
usr/include/sys/fibre-channel
usr/lib/llib-lHBAAPI
usr/lib/llib-lHBAAPI.ln
usr/lib/amd64/llib-lHBAAPI.ln   i386
usr/lib/sparcv9/llib-lHBAAPI.ln sparc
#
usr/bin/dscfgcli
usr/bin/sd_diag
usr/bin/sd_stats
usr/include/nsctl.h
usr/include/sys/ncall
usr/include/sys/nsc_ddi.h
usr/include/sys/nsc_thread.h
usr/include/sys/nsctl
usr/include/sys/nskernd.h
usr/include/sys/unistat
usr/lib/libnsctl.so
usr/lib/librdc.so
usr/lib/libunistat.so
usr/lib/llib-lnsctl.ln
usr/lib/llib-lrdc.ln
usr/lib/llib-lunistat.ln
#
# These files are used by the iSCSI initiator only.
# No reason to ship them.
#
usr/include/sys/scsi/adapters/iscsi_door.h
usr/include/sys/scsi/adapters/iscsi_if.h
#
# sbd ioctl hdr
#
usr/include/sys/stmf_sbd_ioctl.h
#
# proxy port provider interface
#
usr/include/sys/pppt_ic_if.h
usr/include/sys/pppt_ioctl.h
#
# proxy daemon lint library
#
usr/lib/llib-lstmfproxy
usr/lib/llib-lstmfproxy.ln
usr/lib/amd64/llib-lstmfproxy.ln        i386
usr/lib/sparcv9/llib-lstmfproxy.ln      sparc
#
# portable object file and dictionary used by libfmd_msg test
#
usr/lib/fm/dict/TEST.dict
usr/lib/locale/C/LC_MESSAGES/TEST.mo
usr/lib/locale/C/LC_MESSAGES/TEST.po
#
# Private idmap RPC protocol
#
usr/include/rpcsvc/idmap_prot.h
usr/include/rpcsvc/idmap_prot.x
#
# Private idmap directory API
#
usr/include/directory.h
```

```
920 #
921 # librstp is private for bridging
922 #
923 usr/include/stp_bpdu.h
924 usr/include/stp_in.h
925 usr/include/stp_vectors.h
926 usr/lib/librstp.so
927 usr/lib/llib-lrstp
928 usr/lib/llib-lrstp.ln
929 #
930 # Private nvfru API
931 #
932 usr/include/nvfru.h
933 #
934 # vrrp
935 #
936 usr/include/libvrrpadm.h
937 usr/lib/libvrrpadm.so
938 usr/lib/amd64/libvrrpadm.so             i386
939 usr/lib/sparcv9/libvrrpadm.so           sparc
940 usr/lib/llib-lvrrpadm
941 usr/lib/llib-lvrrpadm.ln
942 usr/lib/amd64/llib-lvrrpadm.ln          i386
943 usr/lib/sparcv9/llib-lvrrpadm.ln        sparc
944 #
945 # This is only used during the -t tools build
946 #
947 opt/onbld/bin/i386/elfsign      i386
948 opt/onbld/bin/sparc/elfsign     sparc

950 #
951 # Private libdwarf
952 #
953 opt/onbld/lib/i386/libdwarf.so  i386
954 opt/onbld/lib/sparc/libdwarf.so sparc

956 #
957 # Private socket filter API
958 #
959 usr/include/sys/sockfilter.h
960 #
961 # We don't actually validate license action payloads, and the license
962 # staging area is provided as a separate basedir for package
963 # publication.  The net result is that everything therein should be
964 # ignored for packaging validation.
965 #
966 licenses
967 #
968 # Libbe is private
969 #
970 usr/include/libbe_priv.h
971 #
972 # ipmi is at present only useful on i386, but for historical reasons is
973 # delivered on SPARC and used by the build.
974 #
975 usr/include/sys/ipmi.h  sparc

977 #
978 # libsaveargs is private
979 #
980 usr/include/saveargs.h                  i386
981 usr/lib/amd64/libsaveargs.so            i386
982 usr/lib/amd64/libstandsaveargs.so       i386
983 usr/lib/amd64/llib-lsaveargs.ln         i386

985 #
```



```
986 # libpcidb is private
987 #
988 usr/include/pcidb.h
989 usr/lib/amd64/libpcidb.so               i386
990 usr/lib/amd64/llib-lpcidb.ln            i386
991 usr/lib/sparcv9/libpcidb.so             sparc
992 usr/lib/sparcv9/llib-lpcidb.ln          sparc
993 usr/lib/libpcidb.so
994 usr/lib/llib-lpcidb
995 usr/lib/llib-lpcidb.ln

997 #
998 # Private OpenSSL library
999 #
1000 usr/include/openssl
1001 usr/lib/amd64/libsunw_crypto.so         i386
1002 usr/lib/amd64/libsunw_ssl.so           i386
1003 usr/lib/libsunw_crypto.so
1004 usr/lib/libsunw_ssl.so
1005 #endif /* ! codereview */
```

```
**********************************************************
    1278 Wed Aug 13 19:51:24 2014
new/usr/src/cmd/cmd-inet/usr.lib/wanboot/p12split/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

include ../Makefile.com

PROG=           p12split
LDLIBS +=       -lwanboot -linetutil -lwanbootutil
LDLIBS +=       -lwanboot -linetutil -lwanbootutil -lcrypto
CPPFLAGS +=     -I$(CMNCRYPTDIR)

# libsunw_crypto has no lint library, so we can only include this while building
$(PROG) := LDLIBS += -lsunw_crypto

LINTFLAGS +=    -erroff=E_NAME_USED_NOT_DEF2
#endif /* ! codereview */

all:            $(PROG)

install:        all $(ROOTCMD)

clean:

lint:           lint_PROG

include ../../../../Makefile.targ
```

```
**********************************************************
   15471 Wed Aug 13 19:51:24 2014
new/usr/src/cmd/cmd-inet/usr.lib/wanboot/p12split/p12split.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2002-2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include <stdio.h>
#include <libintl.h>
#include <locale.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/wanboot_impl.h>
#include <unistd.h>
#include <string.h>
#include <libinetutil.h>
#include <wanbootutil.h>

#include <openssl/crypto.h>
#include <openssl/buffer.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/pkcs12.h>
#include <openssl/evp.h>
#include <p12aux.h>

static boolean_t verbose = B_FALSE;     /* When nonzero, do in verbose mode */

/* The following match/cert values require PKCS12 */
static int  matchty;            /* Type of matching do to on input */
static char *k_matchval;        /* localkeyid value to match */
static uint_t k_len;            /* length of k_matchval */

#define IO_KEYFILE      1       /* Have a separate key file or data */
#define IO_CERTFILE     2       /* Have a separate cert file or data */
#define IO_TRUSTFILE    4       /* Have a separate trustanchor file */

static char *input = NULL;      /* Consolidated input file */
```



```
static char *key_out = NULL;    /* Key file to be output */
static char *cert_out = NULL;   /* Cert file to be output */
static char *trust_out = NULL;  /* Trust anchor file to be output */
static uint_t outfiles;         /* What files are there for output */
static char *progname;

/* Returns from time_check */
typedef enum {
        CHK_TIME_OK = 0,                /* Cert in effect and not expired */
        CHK_TIME_BEFORE_BAD,            /* not_before field is invalid */
        CHK_TIME_AFTER_BAD,             /* not_after field is invalid */
        CHK_TIME_IS_BEFORE,             /* Cert not yet in force */
        CHK_TIME_HAS_EXPIRED            /* Cert has expired */
} time_errs_t;
______unchanged_portion_omitted_

static int
do_certs(void)
{
        char *bufp;
        STACK_OF(X509) *ta_in = NULL;
        EVP_PKEY *pkey_in = NULL;
        X509 *xcert_in = NULL;

        sunw_crypto_init();

        if (read_files(&ta_in, &xcert_in, &pkey_in) < 0)
                return (-1);

        if (verbose) {
                if (xcert_in != NULL) {
                        (void) printf(gettext("\nMain cert:\n"));

                        /*
                         * sunw_subject_attrs() returns a pointer to
                         * memory allocated on our behalf. The same
                         * behavior is exhibited by sunw_issuer_attrs().
                         */
                        bufp = sunw_subject_attrs(xcert_in, NULL, 0);
                        if (bufp != NULL) {
                                (void) printf(gettext("  Subject: %s\n"),
                                    bufp);
                                OPENSSL_free(bufp);
                        }

                        bufp = sunw_issuer_attrs(xcert_in, NULL, 0);
                        if (bufp != NULL) {
                                (void) printf(gettext("  Issuer: %s\n"), bufp);
                                OPENSSL_free(bufp);
                        }

                        (void) sunw_print_times(stdout, PRNT_BOTH, NULL,
                            xcert_in);
                }

                if (ta_in != NULL) {
                        X509 *x;
                        int i;

                        for (i = 0; i < sk_X509_num(ta_in); i++) {
                                /* LINTED */
                                x = sk_X509_value(ta_in, i);
                                (void) printf(
                                    gettext("\nTrust Anchor cert %d:\n"), i);

                                /*
```

```
268                                 * sunw_subject_attrs() returns a pointer to
269                                 * memory allocated on our behalf. We get the
270                                 * same behavior from sunw_issuer_attrs().
271                                 */
272                                bufp = sunw_subject_attrs(x, NULL, 0);
273                                if (bufp != NULL) {
274                                        (void) printf(
275                                            gettext("  Subject: %s\n"), bufp);
276                                        OPENSSL_free(bufp);
277                                }

279                                bufp = sunw_issuer_attrs(x, NULL, 0);
280                                if (bufp != NULL) {
281                                        (void) printf(
282                                            gettext("  Issuer: %s\n"), bufp);
283                                        OPENSSL_free(bufp);
284                                }

286                                (void) sunw_print_times(stdout, PRNT_BOTH,
287                                        NULL, x);
288                        }
289                }
290        }

292        check_certs(ta_in, &xcert_in);
293        if (xcert_in != NULL && pkey_in != NULL) {
294                if (sunw_check_keys(xcert_in, pkey_in) == 0) {
295                        wbku_printerr("warning: key and certificate do "
296                            "not match\n");
297                }
298        }

300        return (write_files(ta_in, xcert_in, pkey_in));
301 }
```

*unchanged_portion_omitted_*

```
354 static void
355 check_certs(STACK_OF(X509) *ta_in, X509 **c_in)
356 {
357        X509 *curr;
358        time_errs_t ret;
359        int i;
360        int del_expired = (outfiles != 0);

362        if (c_in != NULL && *c_in != NULL) {
363                ret = time_check_print(*c_in);
364                if ((ret != CHK_TIME_OK && ret != CHK_TIME_IS_BEFORE) &&
365                    del_expired) {
366                        (void) fprintf(stderr, gettext("  Removing cert\n"));
367                        X509_free(*c_in);
368                        *c_in = NULL;
369                }
370        }

372        if (ta_in == NULL)
373                return;

375        for (i = 0; i < sk_X509_num(ta_in); ) {
379                /* LINTED */
376                curr = sk_X509_value(ta_in, i);
377                ret = time_check_print(curr);
378                if ((ret != CHK_TIME_OK && ret != CHK_TIME_IS_BEFORE) &&
379                    del_expired) {
380                        (void) fprintf(stderr, gettext("  Removing cert\n"));
385                        /* LINTED */
381                        curr = sk_X509_delete(ta_in, i);
```



```
382                        X509_free(curr);
383                        continue;
384                }
385                i++;
386        }
387 }
```

*unchanged_portion_omitted_*

```
521 static int
522 do_ofile(char *name, EVP_PKEY *pkey, X509 *cert, STACK_OF(X509) *ta)
523 {
524        STACK_OF(EVP_PKEY) *klist = NULL;
525        STACK_OF(X509)  *clist = NULL;
526        PKCS12          *p12 = NULL;
527        int             ret = 0;
528        FILE            *fp;
529        struct stat     sbuf;

531        if (stat(name, &sbuf) == 0 && !S_ISREG(sbuf.st_mode)) {
532                wbku_printerr("%s is not a regular file\n", name);
533                return (-1);
534        }

536        if ((fp = fopen(name, "w")) == NULL) {
537                wbku_printerr("cannot open output file %s", name);
538                return (-1);
539        }

541        if ((clist = sk_X509_new_null()) == NULL ||
542            (klist = sk_EVP_PKEY_new_null()) == NULL) {
543                wbku_printerr("out of memory\n");
544                ret = -1;
545                goto cleanup;
546        }

548        if (cert != NULL && sk_X509_push(clist, cert) == 0) {
549                wbku_printerr("out of memory\n");
550                ret = -1;
551                goto cleanup;
552        }

554        if (pkey != NULL && sk_EVP_PKEY_push(klist, pkey) == 0) {
555                wbku_printerr("out of memory\n");
556                ret = -1;
557                goto cleanup;
558        }

560        p12 = sunw_PKCS12_create(WANBOOT_PASSPHRASE, klist, clist, ta);
561        if (p12 == NULL) {
562                wbku_printerr("cannot create %s: %s\n", name, cryptoerr());
563                ret = -1;
564                goto cleanup;
565        }

567        if (i2d_PKCS12_fp(fp, p12) == 0) {
568                wbku_printerr("cannot write %s: %s\n", name, cryptoerr());
569                ret = -1;
570                goto cleanup;
571        }

573 cleanup:
574        (void) fclose(fp);
575        if (p12 != NULL)
576                PKCS12_free(p12);
577        /*
578         * Put the cert and pkey off of the stack so that they won't
```

```
579          * be freed two times.  (If they get left in the stack then
580          * they will be freed with the stack.)
581          */
582         if (clist != NULL) {
583                 if (cert != NULL && sk_X509_num(clist) == 1) {
589                         /* LINTED */
584                         (void) sk_X509_delete(clist, 0);
585                 }
586                 sk_X509_pop_free(clist, X509_free);
587         }
588         if (klist != NULL) {
589                 if (pkey != NULL && sk_EVP_PKEY_num(klist) == 1) {
596                         /* LINTED */
590                         (void) sk_EVP_PKEY_delete(klist, 0);
591                 }
592                 sk_EVP_PKEY_pop_free(klist, sunw_evp_pkey_free);
593         }

595         return (ret);
596 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    1289 Wed Aug 13 19:51:24 2014
new/usr/src/cmd/cmd-inet/usr.lib/wanboot/wanboot-cgi/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

include ../Makefile.com

PROG =          wanboot-cgi
LDLIBS +=       -lgen -lnsl -lwanbootutil -lnvpair -lwanboot
LDLIBS +=       -lgen -lnsl -lwanbootutil -lnvpair -lwanboot -lcrypto
CPPFLAGS +=     -I$(CMNCRYPTDIR)

# libsunw_crypto has no lint library, so we can only include this while building
$(PROG) := LDLIBS += -lsunw_crypto

LINTFLAGS +=    -erroff=E_NAME_USED_NOT_DEF2
#endif /* ! codereview */

all:            $(PROG)

install:        all $(ROOTCMD)

clean:

lint:           lint_PROG

include ../../../../Makefile.targ
```

```
**********************************************************
   45017 Wed Aug 13 19:51:25 2014
new/usr/src/cmd/cmd-inet/usr.lib/wanboot/wanboot-cgi/wanboot-cgi.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
______unchanged_portion_omitted_

 773 /*
 774  * Add the certs found in the trustfile found in path (a trust store) to
 775  * the file found at bootfs_dir/truststore.  If necessary, create the
 776  * output file.
 777  */
 778 static int
 779 build_trustfile(const char *path, void *truststorepath)
 780 {
 781         int             ret = WBCGI_FTW_CBERR;
 782         STACK_OF(X509)  *i_anchors = NULL;
 783         STACK_OF(X509)  *o_anchors = NULL;
 784         char            message[WBCGI_MAXBUF];
 785         PKCS12          *p12 = NULL;
 786         FILE            *rfp = NULL;
 787         FILE            *wfp = NULL;
 788         struct stat     i_st;
 789         struct stat     o_st;
 790         X509            *x = NULL;
 791         int             errtype = 0;
 792         int             wfd = -1;
 793         int             chars;
 794         int             i;

 796         if (!WBCGI_FILE_EXISTS(path, i_st)) {
 797                 goto cleanup;
 798         }

 800         if (WBCGI_FILE_EXISTS((char *)truststorepath, o_st)) {
 801                 /*
 802                  * If we are inadvertantly writing to the input file.
 803                  * return success.
 804                  * XXX Pete: how can this happen, and why success?
 805                  */
 806                 if (i_st.st_ino == o_st.st_ino) {
 807                         ret = WBCGI_FTW_CBCONT;
 808                         goto cleanup;
 809                 }
 810                 if ((wfp = fopen((char *)truststorepath, "r+")) == NULL) {
 811                         goto cleanup;
 812                 }
 813                 /*
 814                  * Read what's already there, so that new information
 815                  * can be added.
 816                  */
 817                 if ((p12 = d2i_PKCS12_fp(wfp, NULL)) == NULL) {
 818                         errtype = 1;
 819                         goto cleanup;
 820                 }
 821                 i = sunw_PKCS12_parse(p12, WANBOOT_PASSPHRASE, DO_NONE, NULL,
 822                     0, NULL, NULL, NULL, &o_anchors);
 823                 if (i <= 0) {
 824                         errtype = 1;
 825                         goto cleanup;
 826                 }

 828                 PKCS12_free(p12);
 829                 p12 = NULL;
 830         } else {
 831                 if (errno != ENOENT) {
```



```
 832                         chars = snprintf(message, sizeof (message),
 833                             "(error accessing file %s, error %s)",
 834                             path, strerror(errno));
 835                         if (chars > 0 && chars < sizeof (message))
 836                                 print_status(500, message);
 837                         else
 838                                 print_status(500, NULL);
 839                         return (WBCGI_FTW_CBERR);
 840                 }

 842                 /*
 843                  * Note: We could copy the file to the new trustfile, but
 844                  * we can't verify the password that way.  Therefore, copy
 845                  * it by reading it.
 846                  */
 847                 if ((wfd = open((char *)truststorepath,
 848                     O_CREAT|O_EXCL|O_RDWR, 0700)) < 0) {
 849                         goto cleanup;
 850                 }
 851                 if ((wfp = fdopen(wfd, "w+")) == NULL) {
 852                         goto cleanup;
 853                 }
 854                 o_anchors = sk_X509_new_null();
 855                 if (o_anchors == NULL) {
 856                         goto cleanup;
 857                 }
 858         }

 860         if ((rfp = fopen(path, "r")) == NULL) {
 861                 goto cleanup;
 862         }
 863         if ((p12 = d2i_PKCS12_fp(rfp, NULL)) == NULL) {
 864                 errtype = 1;
 865                 goto cleanup;
 866         }
 867         i = sunw_PKCS12_parse(p12, WANBOOT_PASSPHRASE, DO_NONE, NULL, 0, NULL,
 868             NULL, NULL, &i_anchors);
 869         if (i <= 0) {
 870                 errtype = 1;
 871                 goto cleanup;
 872         }
 873         PKCS12_free(p12);
 874         p12 = NULL;

 876         /*
 877          * Merge the two stacks of pkcs12 certs.
 878          */
 879         for (i = 0; i < sk_X509_num(i_anchors); i++) {
 880                 /* LINTED */
 880                 x = sk_X509_delete(i_anchors, i);
 881                 (void) sk_X509_push(o_anchors, x);
 882         }

 884         /*
 885          * Create the pkcs12 structure from the modified input stack and
 886          * then write out that structure.
 887          */
 888         p12 = sunw_PKCS12_create((const char *)WANBOOT_PASSPHRASE, NULL, NULL,
 889             o_anchors);
 890         if (p12 == NULL) {
 891                 goto cleanup;
 892         }
 893         rewind(wfp);
 894         if (i2d_PKCS12_fp(wfp, p12) == 0) {
 895                 goto cleanup;
 896         }
```

```
 898         ret = WBCGI_FTW_CBCONT;
 899 cleanup:
 900         if (ret == WBCGI_FTW_CBERR) {
 901                 if (errtype == 1) {
 902                         chars = snprintf(message, sizeof (message),
 903                             "(internal PKCS12 error while copying %s to %s)",
 904                             path, (char *)truststorepath);
 905                 } else {
 906                         chars = snprintf(message, sizeof (message),
 907                             "(error copying %s to %s)",
 908                             path, (char *)truststorepath);
 909                 }
 910                 if (chars > 0 && chars <= sizeof (message)) {
 911                         print_status(500, message);
 912                 } else {
 913                         print_status(500, NULL);
 914                 }
 915         }
 916         if (rfp != NULL) {
 917                 (void) fclose(rfp);
 918         }
 919         if (wfp != NULL) {
 920                 /* Will also close wfd */
 921                 (void) fclose(wfp);
 922         }
 923         if (p12 != NULL) {
 924                 PKCS12_free(p12);
 925         }
 926         if (i_anchors != NULL) {
 927                 sk_X509_pop_free(i_anchors, X509_free);
 928         }
 929         if (o_anchors != NULL) {
 930                 sk_X509_pop_free(o_anchors, X509_free);
 931         }

 933         return (ret);
 934 }
______unchanged_portion_omitted_

1056 /*
1057  * Loop through the certificates in a file
1058  */
1059 static int
1060 get_hostnames(const char *path, void *nvl)
1061 {
1062         int             ret = WBCGI_FTW_CBERR;
1063         STACK_OF(X509)  *certs = NULL;
1064         PKCS12          *p12 = NULL;
1065         char            message[WBCGI_MAXBUF];
1066         char            buf[WBCGI_MAXBUF + 1];
1067         FILE            *rfp = NULL;
1068         X509            *x = NULL;
1069         int             errtype = 0;
1070         int             chars;
1071         int             i;

1073         if ((rfp = fopen(path, "r")) == NULL) {
1074                 goto cleanup;
1075         }

1077         if ((p12 = d2i_PKCS12_fp(rfp, NULL)) == NULL) {
1078                 errtype = 1;
1079                 goto cleanup;
1080         }
1081         i = sunw_PKCS12_parse(p12, WANBOOT_PASSPHRASE, DO_NONE, NULL, 0, NULL,
```



```
1082             NULL, NULL, &certs);
1083         if (i <= 0) {
1084                 errtype = 1;
1085                 goto cleanup;
1086         }

1088         PKCS12_free(p12);
1089         p12 = NULL;

1091         for (i = 0; i < sk_X509_num(certs); i++) {
1093                 /* LINTED */
1092                 x = sk_X509_value(certs, i);
1093                 if (!one_name(sunw_issuer_attrs(x, buf, sizeof (buf) - 1),
1094                     nvl)) {
1095                         goto cleanup;
1096                 }
1097         }

1099         ret = WBCGI_FTW_CBCONT;
1100 cleanup:
1101         if (ret == WBCGI_FTW_CBERR) {
1102                 if (errtype == 1) {
1103                         chars = snprintf(message, sizeof (message),
1104                             "(internal PKCS12 error reading %s)", path);
1105                 } else {
1106                         chars = snprintf(message, sizeof (message),
1107                             "error reading %s", path);
1108                 }
1109                 if (chars > 0 && chars <= sizeof (message)) {
1110                         print_status(500, message);
1111                 } else {
1112                         print_status(500, NULL);
1113                 }
1114         }
1115         if (rfp != NULL) {
1116                 (void) fclose(rfp);
1117         }
1118         if (p12 != NULL) {
1119                 PKCS12_free(p12);
1120         }
1121         if (certs != NULL) {
1122                 sk_X509_pop_free(certs, X509_free);
1123         }

1125         return (ret);
1126 }
______unchanged_portion_omitted_
```

```
**********************************************************
    1459 Wed Aug 13 19:51:25 2014
new/usr/src/cmd/cmd-inet/usr.lib/wpad/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 # Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  22 # Use is subject to license terms.
  23 #

  25 PROG =          wpad
  26 MANIFEST =      wpa.xml
  27 OBJS =          wpa_supplicant.o wpa.o wpa_enc.o eloop.o \
  28                 driver_wifi.o l2_packet.o
  29 SRCS =          $(OBJS:%.o=%.c)

  31 include ../../../Makefile.cmd

  33 ROOTMANIFESTDIR = $(ROOTSVCNETWORK)

  35 LDLIBS    +=    -ldladm -ldlpi
  36 $(PROG) := LDLIBS += -lsunw_crypto
  36 all install := LDLIBS += -lcrypto

  38 LINTFLAGS +=    -u

  40 .KEEP_STATE:

  42 all:            $(PROG)

  44 $(PROG):        $(OBJS)
  45                 $(LINK.c) $(OBJS) -o $@ $(LDLIBS)
  46                 $(POST_PROCESS)

  48 include ../Makefile.lib

  50 install:        all $(ROOTLIBINETPROG) $(ROOTMANIFEST)

  52 check:          $(CHKMANIFEST)

  54 clean:
  55                 $(RM) $(OBJS)

  57 lint:           lint_SRCS

  59 include ../../../Makefile.targ
```

```
**********************************************************
    2418 Wed Aug 13 19:51:26 2014
new/usr/src/cmd/sendmail/src/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

# Copyright (c) 2011 Gary Mills

#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/sendmail/src/Makefile
#

PROG=   sendmail

include         ../../Makefile.cmd
include         ../Makefile.cmd

OBJS= alias.o arpadate.o bf.o collect.o conf.o control.o convtime.o daemon.o \
        deliver.o domain.o envelope.o err.o headers.o macro.o main.o map.o \
        mci.o milter.o mime.o parseaddr.o queue.o ratectrl.o readcf.o \
        recipient.o sasl.o savemail.o sfsasl.o sm_resolve.o srvrsmtp.o stab.o \
        stats.o sysexits.o tls.o trace.o udb.o usersmtp.o util.o version.o

SRCS=   $(OBJS:%.o=%.c)

MAPFILES =      $(MAPFILE.INT) $(MAPFILE.NGB)
LDFLAGS +=      $(MAPFILES:%=-M%)

LDLIBS +=       ../libsmutil/libsmutil.a ../libsm/libsm.a -lresolv -lsocket \
                -lnsl ../db/libdb.a -lldap -lsldap -lwrap -lumem \
                -lsunw_ssl -lsunw_crypto -lsasl
                -lssl -lcrypto -lsasl

INCPATH=        -I. -I../include -I../db

ENVDEF=         -DNETINET6 -DTCPWRAPPERS -DSTARTTLS -DSASL=20115
SUNENVDEF=      -DSUN_EXTENSIONS -DVENDOR_DEFAULT=VENDOR_SUN \
                -DSUN_INIT_DOMAIN -DSUN_SIMPLIFIED_LDAP -D_FFR_LOCAL_DAEMON \
                -D_FFR_MAIL_MACRO

CPPFLAGS =      $(INCPATH) $(ENVDEF) $(SUNENVDEF) $(DBMDEF) $(CPPFLAGS.sm)

FILEMODE=       2555
```



```
ROOTSYMLINKS=   $(ROOTUSRSBIN)/newaliases $(ROOTUSRSBIN)/sendmail

# build rule
#

.KEEP_STATE:
all:            $(PROG)

.PARALLEL:      $(OBJS)

$(PROG):        $(OBJS) $(MAPFILES) \
        ../libsmutil/libsmutil.a ../libsm/libsm.a ../db/libdb.a
        $(LINK.c) -o $@ $(OBJS) $(LDLIBS)
        $(POST_PROCESS)

install:        $(ROOTLIBPROG) $(ROOTSYMLINKS)

$(ROOTSYMLINKS):
        $(RM) $@; $(SYMLINK) ../lib/sendmail $@

clean:
        $(RM) $(PROG) $(OBJS)

lint:   lint_SRCS

include         ../../Makefile.targ
```

```
**********************************************************
    2444 Wed Aug 13 19:51:26 2014
new/usr/src/cmd/ssh/libssh/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBRARY =       libssh.a
VERS =          .1

OBJECTS =       \
                addrmatch.o \
                atomicio.o \
                authfd.o \
                authfile.o \
                bufaux.o \
                buffer.o \
                canohost.o \
                channels.o \
                cipher.o \
                cipher-ctr.o \
                compat.o \
                compress.o \
                crc32.o \
                deattack.o \
                dh.o \
                dispatch.o \
                engine.o \
                entropy.o \
                fatal.o \
                g11n.o \
                hostfile.o \
                key.o \
                kex.o \
                kexdh.o \
                kexdhc.o \
                kexdhs.o \
                kexgex.o \
                kexgexc.o \
                kexgexs.o \
                kexgssc.o \
                kexgsss.o \
                log.o \
                mac.o \
```



```
                match.o \
                misc.o \
                mpaux.o \
                msg.o \
                nchan.o \
                packet.o \
                progressmeter.o \
                proxy-io.o \
                radix.o \
                readconf.o \
                readpass.o \
                rsa.o \
                sftp-common.o \
                ssh-dss.o \
                ssh-gss.o \
                ssh-rsa.o \
                tildexpand.o \
                ttymodes.o \
                uidswap.o \
                uuencode.o \
                xlist.o \
                xmalloc.o

include $(SRC)/lib/Makefile.lib

BUILD.AR =       $(RM) $@ ; $(AR) $(ARFLAGS) $@ $(AROBJS)

SRCDIR =        ../common
SRCS =          $(OBJECTS:%.o=../common/%.c)

LIBS =          $(LIBRARY) $(LINTLIB)

# Define LDLIBS conditionally for lintcheck, rather than in general, since
# we're building an archive library which itself links to nothing, we just
# want lint to know about these libraries.
lintcheck  :=   LDLIBS += -lz -lsocket -lnsl -lc
lintcheck  :=   LDLIBS += -lcrypto -lz -lsocket -lnsl -lc
$(LINTLIB) :=   SRCS = $(SRCDIR)/$(LINTSRC)

POFILE_DIR =    ../..

.KEEP_STATE:

all:            $(LIBS)

# lint requires the (not installed) lint library
lint:           $(LINTLIB) .WAIT lintcheck

include $(SRC)/lib/Makefile.targ

objs/%.o:       $(SRCDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

include ../../Makefile.ssh-common
include ../../Makefile.msg.targ
```

```
**********************************************************
    1604 Wed Aug 13 19:51:26 2014
new/usr/src/cmd/ssh/sftp-server/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  22 # Use is subject to license terms.
  23 #
  24 # cmd/ssh/sftp-server/Makefile

  26 PROG =          sftp-server

  28 OBJS =          sftp-server.o sftp-server-main.o
  29 SRCS =          $(OBJS:.o=.c)

  31 include ../../Makefile.cmd
  32 include ../Makefile.ssh-common

  34 LDLIBS +=       $(SSH_COMMON_LDLIBS) -lsocket

  36 # libsunw_crypto has no lint library, so we can only use it when building
  37 $(PROG) := LDLIBS += -lsunw_crypto
  34 LDLIBS +=       $(SSH_COMMON_LDLIBS) -lsocket -lcrypto

  39 POFILE_DIR =    ..

  41 .KEEP_STATE:

  43 .PARALLEL:      $(OBJS)

  45 all:            $(PROG)

  47 $(PROG):        $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/
  48         $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
  49         $(POST_PROCESS)

  51 install:        all $(ROOTLIBSSHPROG) $(ROOTLIBSSH)

  53 clean:
  54         $(RM) -f $(OBJS) $(PROG)

  56 lint:           lint_SRCS

  58 include ../Makefile.msg.targ
  59 include ../../Makefile.targ
```

```
**********************************************************
    1590 Wed Aug 13 19:51:27 2014
new/usr/src/cmd/ssh/sftp/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  22 # Use is subject to license terms.
  23 #
  24 # cmd/ssh/sftp/Makefile

  26 PROG =          sftp

  28 OBJS =          \
  29                 sftp.o \
  30                 sftp-client.o \
  31                 sftp-glob.o

  33 SRCS =          $(OBJS:.o=.c)

  35 include ../../Makefile.cmd
  36 include ../Makefile.ssh-common

  38 LDLIBS +=       $(SSH_COMMON_LDLIBS) -lsocket -ltecla

  40 # libsunw_crypto has no lint library, so we can only use it when building
  41 $(PROG) := LDLIBS += -lsunw_crypto
  38 LDLIBS +=       $(SSH_COMMON_LDLIBS) -lsocket -lcrypto -ltecla

  43 POFILE_DIR =    ..

  45 .KEEP_STATE:

  47 .PARALLEL:      $(OBJS)

  49 all: $(PROG)

  51 $(PROG):        $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/
  52         $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
  53         $(POST_PROCESS)

  55 install:        all $(ROOTPROG)

  57 clean:
  58         $(RM) -f $(OBJS) $(PROG)

  60 lint:           lint_SRCS
```



```
  62 include ../Makefile.msg.targ
  63 include ../../Makefile.targ
```

```
**********************************************************
    1552 Wed Aug 13 19:51:27 2014
new/usr/src/cmd/ssh/ssh-add/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/ssh/ssh-add/Makefile

PROG= ssh-add

OBJS    =       \
        ssh-add.o
SRCS    = $(OBJS:.o=.c)

include ../../Makefile.cmd
include ../Makefile.ssh-common

LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket

# libsunw_crypto has no lint library, so we can only use it when building
$(PROG) := LDLIBS += -lsunw_crypto
LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket -lcrypto

POFILE_DIR= ..

.KEEP_STATE:

.PARALLEL: $(OBJS)

all: $(PROG)

$(PROG): $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/libopen
        $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
        $(POST_PROCESS)

install: all $(ROOTPROG)

clean:
        $(RM) -f $(OBJS) $(PROG)

lint:   lint_SRCS

include ../Makefile.msg.targ
include ../../Makefile.targ
```

```
**********************************************************
    1558 Wed Aug 13 19:51:27 2014
new/usr/src/cmd/ssh/ssh-agent/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/ssh/ssh-agent/Makefile

PROG= ssh-agent

OBJS    =       \
        ssh-agent.o
SRCS    = $(OBJS:.o=.c)

include ../../Makefile.cmd
include ../Makefile.ssh-common

LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket

# libsunw_crypto has no lint library, so we can only use it when building
$(PROG) := LDLIBS += -lsunw_crypto
LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket -lcrypto

POFILE_DIR= ..

.KEEP_STATE:

.PARALLEL: $(OBJS)

all: $(PROG)

$(PROG): $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/libopen
        $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
        $(POST_PROCESS)

install: all $(ROOTPROG)

clean:
        $(RM) -f $(OBJS) $(PROG)

lint:   lint_SRCS

include ../Makefile.msg.targ
include ../../Makefile.targ
```

```
**********************************************************
    1561 Wed Aug 13 19:51:27 2014
new/usr/src/cmd/ssh/ssh-keygen/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 # Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  22 # Use is subject to license terms.
  23 #
  24 # cmd/ssh/ssh-keygen/Makefile

  26 PROG= ssh-keygen

  28 OBJS    =       \
  29         ssh-keygen.o
  30 SRCS    = $(OBJS:.o=.c)

  32 include ../../Makefile.cmd
  33 include ../Makefile.ssh-common

  35 LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket

  37 # libsunw_crypto has no lint library, so we can only use it when building
  38 $(PROG) := LDLIBS += -lsunw_crypto
  35 LDLIBS += $(SSH_COMMON_LDLIBS) -lcrypto -lsocket

  40 POFILE_DIR= ..

  42 .KEEP_STATE:

  44 .PARALLEL: $(OBJS)

  46 all: $(PROG)

  48 $(PROG): $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/libopen
  49         $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
  50         $(POST_PROCESS)

  52 install: all $(ROOTPROG)

  54 clean:
  55         $(RM) -f $(OBJS) $(PROG)

  57 lint:   lint_SRCS

  59 include ../Makefile.msg.targ
  60 include ../../Makefile.targ
```

```
**********************************************************
    1574 Wed Aug 13 19:51:28 2014
new/usr/src/cmd/ssh/ssh-keyscan/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 # Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
  22 # Use is subject to license terms.
  23 #
  24 # cmd/ssh/ssh-keyscan/Makefile

  26 PROG= ssh-keyscan

  28 OBJS    =       \
  29         ssh-keyscan.o
  30 SRCS    = $(OBJS:.o=.c)

  32 include ../../Makefile.cmd
  33 include ../Makefile.ssh-common

  35 LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket -lnsl -lz

  37 # libsunw_crypto has no lint library, so we can only use it when building
  38 $(PROG) := LDLIBS += -lsunw_crypto
  35 LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket -lnsl -lz -lcrypto

  40 POFILE_DIR= ..

  42 .KEEP_STATE:

  44 .PARALLEL: $(OBJS)

  46 all: $(PROG)

  48 $(PROG): $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/libopen
  49         $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
  50         $(POST_PROCESS)

  52 clean:
  53         $(RM) -f $(OBJS) $(PROG)

  55 lint:   lint_SRCS

  57 include ../Makefile.msg.targ
  58 include ../../Makefile.targ

  60 install: all $(ROOTPROG)
```

```
**********************************************************
    1695 Wed Aug 13 19:51:28 2014
new/usr/src/cmd/ssh/ssh-keysign/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/ssh/ssh-keysign/Makefile

PROG= ssh-keysign

DIRS= $(ROOTLIBSSH)


OBJS    = ssh-keysign.o
SRCS    = $(OBJS:.o=.c)

include ../../Makefile.cmd
include ../Makefile.ssh-common

FILEMODE= 04555

LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket -lnsl -lz

# libsunw_crypto has no lint library, so we can only use it when building
$(PROG) := LDLIBS += -lsunw_crypto
LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket -lnsl -lz -lcrypto

POFILE_DIR= ..

.KEEP_STATE:

.PARALLEL: $(OBJS)

all: $(PROG)

$(PROG): $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/libopen
        $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
        $(POST_PROCESS)

clean:
        $(RM) -f $(OBJS) $(PROG)

lint:   lint_SRCS
```



```
include ../Makefile.msg.targ
include ../../Makefile.targ

install: all $(DIRS) $(ROOTLIBSSHPROG) $(ROOTLIBSSH)


$(ROOTLIBSSHPROG)/%: %
        $(INS.file)

$(DIRS):
        $(INS.dir)
```

```
**********************************************************
    1691 Wed Aug 13 19:51:28 2014
new/usr/src/cmd/ssh/ssh/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/ssh/ssh/Makefile

PROG= ssh

OBJS    = ssh.o \
          sshconnect.o \
          sshconnect1.o \
          sshconnect2.o \
          sshtty.o \
          clientloop.o \
          gss-clnt.o
SRCS    = $(OBJS:.o=.c)

include ../../Makefile.cmd
include ../Makefile.ssh-common

LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket \
        -lnsl \
        -lz \
        -lcrypto \
        -lgss

# libsunw_crypto has no lint library, so we can only use it when building
$(PROG) := LDLIBS += -lsunw_crypto
#endif /* ! codereview */

POFILE_DIR= ..

.KEEP_STATE:

.PARALLEL: $(OBJS)

all: $(PROG)

$(PROG): $(OBJS) ../libssh/$(MACH)/libssh.a ../libopenbsd-compat/$(MACH)/libopen
        $(LINK.c) $(OBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
        $(POST_PROCESS)
```



```
install: all $(ROOTPROG)

clean:
        $(RM) -f $(OBJS) $(PROG)

lint:   lint_SRCS

include ../Makefile.msg.targ

XGETFLAGS += --keyword=log
include ../../Makefile.targ
```

```
**********************************************************
    2513 Wed Aug 13 19:51:29 2014
new/usr/src/cmd/ssh/sshd/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# cmd/ssh/sshd/Makefile

PROG= sshd

DIRS= $(ROOTLIBSSH)

OBJS    = sshd.o \
          altprivsep.o \
          auth.o \
          auth1.o \
          auth2.o \
          auth-options.o \
          auth2-chall.o \
          auth2-gss.o \
          auth2-hostbased.o \
          auth2-kbdint.o \
          auth2-none.o \
          auth2-passwd.o \
          auth2-pam.o \
          auth2-pubkey.o \
          auth-bsdauth.o \
          auth-chall.o \
          auth-rhosts.o \
          auth-krb4.o \
          auth-krb5.o \
          auth-pam.o \
          auth-passwd.o \
          auth-rsa.o \
          auth-rh-rsa.o \
          auth-sia.o \
          auth-skey.o \
          bsmaudit.o \
          groupaccess.o \
          gss-serv.o \
          loginrec.o \
          servconf.o \
          serverloop.o \
          session.o \
```



```
          sshlogin.o \
          sshpty.o

EXTOBJS = sftp-server.o

SRCS    = $(OBJS:.o=.c) ../sftp-server/sftp-server.c

include ../../Makefile.cmd
include ../Makefile.ssh-common

LDLIBS += $(SSH_COMMON_LDLIBS) -lsocket \
        -lnsl \
        -lz \
        -lpam \
        -lbsm \
        -lwrap \
        -lcrypto \
        -lgss \
        -lcontract
MAPFILES = $(MAPFILE.INT) $(MAPFILE.NGB)
LDFLAGS += $(MAPFILES:%=-M%)

# libsunw_crypto has no lint library, so we can only use it when building
$(PROG) := LDLIBS += -lsunw_crypto
#endif /* ! codereview */

POFILE_DIR= ..

.KEEP_STATE:

.PARALLEL: $(OBJS)

all: $(PROG)

$(PROG): $(OBJS) $(EXTOBJS) $(MAPFILES) ../libssh/$(MACH)/libssh.a \
        ../libopenbsd-compat/$(MACH)/libopenbsd-compat.a
        $(LINK.c) $(OBJS) $(EXTOBJS) -o $@ $(LDLIBS) $(DYNFLAGS)
        $(POST_PROCESS)

%.o : ../sftp-server/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

install: all $(DIRS) $(ROOTLIBSSHPROG) $(ROOTLIBSSH)


$(ROOTLIBSSHPROG)/%: %
        $(INS.file)

$(DIRS):
        $(INS.dir)

clean:
        $(RM) $(OBJS) $(EXTOBJS)

lint:   lint_SRCS

include ../Makefile.msg.targ
include ../../Makefile.targ
```

```
**********************************************************
    1303 Wed Aug 13 19:51:29 2014
new/usr/src/cmd/svr4pkg/pkgadd/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

PROG=           pkgadd

OBJS=           check.o         \
                main.o          \
                quit.o

ROOTLINKS=   $(ROOTUSRSBIN)/pkgask

include $(SRC)/cmd/svr4pkg/Makefile.svr4pkg

LDLIBS  +=      -lpkg -linstzones -ladm
LDLIBS  +=      -lsunw_crypto -lwanboot
LDLIBS  +=      -lcrypto -lwanboot

.KEEP_STATE:

all: $(PROG)

install: all $(ROOTUSRSBINPROG) $(ROOTLINKS)

$(ROOTLINKS): $(ROOTUSRSBINPROG)
        $(RM) $@
        $(LN) $(ROOTUSRSBINPROG) $@

include $(SRC)/cmd/svr4pkg/Makefile.svr4pkg.targ
```

```
**********************************************************
    1223 Wed Aug 13 19:51:29 2014
new/usr/src/cmd/svr4pkg/pkgadm/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#

#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

PROG=           pkgadm

OBJS=           addcert.o       \
                certs.o         \
                listcert.o      \
                lock.o          \
                main.o          \
                removecert.o

include $(SRC)/cmd/svr4pkg/Makefile.svr4pkg

LDLIBS  +=      -lpkg -ladm -lsunw_crypto -lgen
LDLIBS  +=      -lpkg -ladm -lcrypto -lgen

.KEEP_STATE:
all:            $(PROG)

install:        all $(ROOTPROG)

include $(SRC)/cmd/svr4pkg/Makefile.svr4pkg.targ
```

****
72083 Wed Aug 13 19:51:30 2014
new/usr/src/common/net/wanboot/boot_http.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****
_____***unchanged_portion_omitted_***

```
2230 /*
2231  * print_ciphers - Print the list of ciphers for debugging.
2232  *
2233  *       print_ciphers(ssl);
2234  *
2235  * Arguments:
2236  *     ssl      - SSL connection.
2237  *
2238  * Returns:
2239  *     none
2240  */
2241 static void
2242 print_ciphers(SSL *ssl)
2243 {
2244         SSL_CIPHER      *c;
2245         STACK_OF(SSL_CIPHER)    *sk;
2246         int     i;
2247         const char      *name;

2249         if (ssl == NULL)
2250                 return;

2252         sk = SSL_get_ciphers(ssl);
2253         if (sk == NULL)
2254                 return;

2256         for (i = 0; i < sk_SSL_CIPHER_num(sk); i++) {
2257                 /* LINTED */
2257                 c = sk_SSL_CIPHER_value(sk, i);
2258                 libbootlog(BOOTLOG_VERBOSE, "%08lx %s", c->id, c->name);
2259         }
2260         name = SSL_get_cipher_name(ssl);
2261         if (name == NULL)
2262                 name = "";
2263         libbootlog(BOOTLOG_VERBOSE, "Current cipher = %s", name);
2264 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   13543 Wed Aug 13 19:51:30 2014
new/usr/src/lib/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 # Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
  23 # Copyright (c) 1989, 2010, Oracle and/or its affiliates. All rights reserved.
  24 # Copyright (c) 2012 by Delphix. All rights reserved.
  25 # Copyright (c) 2012, Joyent, Inc. All rights reserved.
  26 # Copyright (c) 2013 Gary Mills

  28 include ../Makefile.master

  30 #       Note that libcurses installs commands along with its library.
  31 #       This is a minor bug which probably should be fixed.
  32 #       Note also that a few extra libraries are kept in cmd source.
  33 #
  34 # Certain libraries are linked with, hence depend on, other libraries.
  35 #
  36 # Although we have historically used .WAIT to express dependencies, it
  37 # reduces the amount of parallelism and thus lengthens the time it
  38 # takes to build the libraries.  Thus, we now require that any new
  39 # libraries explicitly call out their dependencies.  Eventually, all
  40 # the library dependencies will be called out explicitly.  See
  41 # "Library interdependencies" near the end of this file.
  42 #
  43 # Aside from explicit dependencies (and legacy .WAITs), all libraries
  44 # are built in parallel.
  45 #
  46 .PARALLEL:

  48 SUBDIRS= \
  49         common                  .WAIT   \
  50         ../cmd/sgs/libconv              \
  51         ../cmd/sgs/libdl        .WAIT

  53 SUBDIRS += \
  54         libc                    .WAIT   \
  55         ../cmd/sgs/libelf       .WAIT   \
  56         c_synonyms                      \
  57         libmd                           \
  58         libmd5                          \
  59         librsm                          \
  60         libmp                   .WAIT   \
  61         libnsl                  \
```



```
  62         libsecdb                .WAIT   \
  63         librpcsvc                       \
  64         libsocket               .WAIT   \
  65         libsctp                 \
  66         libsip                  \
  67         libcommputil            \
  68         libresolv                       \
  69         libresolv2              .WAIT   \
  70         libw                    .WAIT   \
  71         libintl                 .WAIT   \
  72         ../cmd/sgs/librtld_db   \
  73         libaio                  \
  74         libast                  \
  75         libdll                  \
  76         libcmd                  \
  77         libshell                \
  78         libsum                  \
  79         librt                   \
  80         libadm                  \
  81         libctf                  \
  82         libdtrace               \
  83         libdtrace_jni           \
  84         libcurses               \
  85         libtermcap              \
  86         libgen                  \
  87         libgss                  \
  88         libpam                  \
  89         libuuid                 \
  90         libthread               \
  91         libpthread      .WAIT   \
  92         libslp          \
  93         libbsdmalloc    \
  94         libdoor         \
  95         libdevinfo      \
  96         libdladm        \
  97         libdlpi         \
  98         libeti          \
  99         libcrypt        \
 100         libdns_sd       \
 101         libefi          \
 102         libfstyp        \
 103         libwanboot      \
 104         libwanbootutil  \
 105         libcryptoutil   \
 106         libinetutil     \
 107         libipadm        \
 108         libipd          \
 109         libipmp         \
 110         libiscsit       \
 111         libkmf          \
 112         libkstat        \
 113         libkvm          \
 114         liblm           \
 115         libmalloc       \
 116         libmapmalloc    \
 117         libmtmalloc     \
 118         libnls          \
 119         libnwam         \
 120         libsmbios       \
 121         libtecla        \
 122         libumem         \
 123         libnvpair       .WAIT   \
 124         libexacct       \
 125         libsasl         \
 126         libldap5        \
 127         libsldap        .WAIT   \
```

```
128         libbsm          \
129         libsys          \
130         libsysevent     \
131         libnisdb        \
132         libpool         \
133         libpp           \
134         libproc         \
135         libproject      \
136         libsendfile     \
137         nametoaddr      \
138         ncad_addr       \
139         hbaapi          \
140         smhba           \
141         sun_fc          \
142         sun_sas         \
143         gss_mechs/mech_krb5     .WAIT   \
144         libkrb5 .WAIT   \
145         krb5    .WAIT   \
146         libsmbfs        \
147         libfcoe         \
148         libsrpt         \
149         libstmf         \
150         libstmfproxy    \
151         libnsctl        \
152         libunistat      \
153         libdscfg        \
154         librdc          \
155         libinstzones    \
156         libpkg          \
157         libpcidb

159 SUBDIRS += \
160         passwdutil      \
161         pam_modules     \
162         crypt_modules   \
163         libadt_jni      \
164         abi             \
165         auditd_plugins  \
166         libvolmgt       \
167         libdevice       \
168         libdevid        \
169         libdhcpsvc      \
170         libc_db         \
171         libndmp         \
172         libsec          \
173         libtnfprobe     \
174         libtnf          \
175         libtnfctl       \
176         libdhcpagent    \
177         libdhcpdu       \
178         libdhcputil     \
179         libxnet         \
180         libipsecutil    \
181         openssl         \
182 #endif /* ! codereview */
183         nsswitch        \
184         print           \
185         libuutil        \
186         libscf          \
187         libinetsvc      \
188         librestart      \
189         libsched        \
190         libelfsign      \
191         pkcs11          .WAIT   \
192         libpctx         .WAIT   \
193         libcpc          \
```



```
194         getloginx       \
195         watchmalloc     \
196         extendedFILE    \
197         madv            \
198         mpss            \
199         libdisasm       \
200         libwrap         \
201         libxcurses      \
202         libxcurses2     \
203         libbrand        .WAIT   \
204         libzonecfg      \
205         libzoneinfo     \
206         libzonestat     \
207         libtsnet        \
208         libtsol         \
209         gss_mechs/mech_spnego   \
210         gss_mechs/mech_dummy    \
211         gss_mechs/mech_dh       \
212         rpcsec_gss      \
213         libraidcfg      .WAIT   \
214         librcm          .WAIT   \
215         libcfgadm       .WAIT   \
216         libpicl         .WAIT   \
217         libpicltree     .WAIT \
218         raidcfg_plugins \
219         cfgadm_plugins  \
220         libmail         \
221         lvm             \
222         libsmedia       \
223         libipp          \
224         libdiskmgt      \
225         liblgrp         \
226         libfsmgt        \
227         fm              \
228         libavl          \
229         libcmdutils     \
230         libcontract     \
231         ../cmd/sendmail/libmilter       \
232         sasl_plugins    \
233         udapl           \
234         libzpool        \
235         libzfs_core     \
236         libzfs          \
237         libbe           \
238         pylibbe         \
239         libzfs_jni      \
240         pyzfs           \
241         pysolaris       \
242         libmapid        \
243         brand           \
244         policykit       \
245         hal             \
246         libshare        \
247         libsqlite       \
248         libidmap        \
249         libadutils      \
250         libipmi         \
251         libexacct/demo  \
252         libvrrpadm      \
253         libvscan        \
254         libgrubmgmt     \
255         smbsrv          \
256         libilb          \
257         scsi            \
258         libima          \
259         libsun_ima      \
```

```
260         mpapi           \
261         librstp         \
262         libreparse      \
263         libhotplug      \
264         libfruutils     .WAIT   \
265         libfru          \
266         $($(MACH)_SUBDIRS)

268 i386_SUBDIRS=           \
269         libntfs         \
270         libparted       \
271         libfdisk        \
272         libsaveargs

274 sparc_SUBDIRS= .WAIT    \
275         efcode          \
276         libds           \
277         libdscp         \
278         libprtdiag      .WAIT   \
279         libprtdiag_psr  \
280         libpri          \
281         librsc          \
282         storage         \
283         libpcp          \
284         libtsalarm      \
285         libv12n

287 FM_sparc_DEPLIBS= libpri

289 fm:                     \
290         libexacct       \
291         libipmi         \
292         libzfs          \
293         scsi            \
294         $(FM_$(MACH)_DEPLIBS)

296 #
297 # Create a special version of $(SUBDIRS) with no .WAIT's, for use with the
298 # clean and clobber targets (for more information, see those targets, below).
299 #
300 NOWAIT_SUBDIRS= $(SUBDIRS:.WAIT=)

302 DCSUBDIRS =             \
303         lvm

305 MSGSUBDIRS=             \
306         abi             \
307         auditd_plugins  \
308         brand           \
309         cfgadm_plugins  \
310         gss_mechs/mech_dh       \
311         gss_mechs/mech_krb5     \
312         krb5            \
313         libast          \
314         libbsm          \
315         libc            \
316         libcfgadm       \
317         libcmd          \
318         libcontract     \
319         libcurses       \
320         libdhcpsvc      \
321         libdhcputil     \
322         libipsecutil    \
323         libdiskmgt      \
324         libdladm        \
325         libdll          \
```



```
326         libgrubmgmt     \
327         libgss          \
328         libidmap        \
329         libipmp         \
330         libilb          \
331         libinetutil     \
332         libinstzones    \
333         libipadm        \
334         libnsl          \
335         libnwam         \
336         libpam          \
337         libpicl         \
338         libpool         \
339         libpkg          \
340         libpp           \
341         libscf          \
342         libsasl         \
343         libldap5        \
344         libsecdb        \
345         libshare        \
346         libshell        \
347         libsldap        \
348         libslp          \
349         libsmbfs        \
350         libsmedia       \
351         libsum          \
352         libtsol         \
353         libuutil        \
354         libvrrpadm      \
355         libvscan        \
356         libwanboot      \
357         libwanbootutil  \
358         libzfs          \
359         libzonecfg      \
360         lvm             \
361         madv            \
362         mpss            \
363         pam_modules     \
364         pyzfs           \
365         pysolaris       \
366         rpcsec_gss      \
367         libreparse
368 MSGSUBDIRS += \
369         $($(MACH)_MSGSUBDIRS)

371 sparc_MSGSUBDIRS=       \
372         libprtdiag      \
373         libprtdiag_psr

375 i386_MSGSUBDIRS= libfdisk

377 HDRSUBDIRS=             \
378         auditd_plugins  \
379         libast          \
380         libbrand        \
381         libbsm          \
382         libc            \
383         libcmd          \
384         libcmdutils     \
385         libcommputil    \
386         libcontract     \
387         libcpc          \
388         libctf          \
389         libcurses       \
390         libtermcap      \
391         libcryptoutil   \
```

```
392         libdevice       \
393         libdevid        \
394         libdevinfo      \
395         libdiskmgt      \
396         libdladm        \
397         libdll          \
398         libdlpi         \
399         libdhcpagent    \
400         libdhcpsvc      \
401         libdhcputil     \
402         libdisasm       \
403         libdns_sd       \
404         libdscfg        \
405         libdtrace       \
406         libdtrace_jni   \
407         libelfsign      \
408         libeti          \
409         libfru          \
410         libfstyp        \
411         libgen          \
412         libipadm        \
413         libipd          \
414         libipsecutil    \
415         libinetsvc      \
416         libinetutil     \
417         libinstzones    \
418         libipmi         \
419         libipmp         \
420         libipp          \
421         libiscsit       \
422         libkstat        \
423         libkvm          \
424         libmail         \
425         libmd           \
426         libmtmalloc     \
427         libndmp         \
428         libnvpair       \
429         libnsctl        \
430         libnsl          \
431         libnwam         \
432         libpam          \
433         libpcidb        \
434         libpctx         \
435         libpicl         \
436         libpicltree     \
437         libpool         \
438         libpp           \
439         libproc         \
440         libraidcfg      \
441         librcm          \
442         librdc          \
443         libscf          \
444         libsip          \
445         libsmbios       \
446         librestart      \
447         librpcsvc       \
448         librsm          \
449         librstp         \
450         libsasl         \
451         libsec          \
452         libshell        \
453         libslp          \
454         libsmedia       \
455         libsocket       \
456         libsqlite       \
457         libfcoe         \
```



```
458         libsrpt         \
459         libstmf         \
460         libstmfproxy    \
461         libsum          \
462         libsysevent     \
463         libtecla        \
464         libtnf          \
465         libtnfctl       \
466         libtnfprobe     \
467         libtsnet        \
468         libtsol         \
469         libvrrpadm      \
470         libvolmgt       \
471         libumem         \
472         libunistat      \
473         libuutil        \
474         libwanboot      \
475         libwanbootutil  \
476         libwrap         \
477         libxcurses2     \
478         libzfs          \
479         libzfs_core     \
480         libzfs_jni      \
481         libzoneinfo     \
482         libzonestat     \
483         hal             \
484         policykit       \
485         lvm             \
486         pkcs11          \
487         passwdutil      \
488         ../cmd/sendmail/libmilter       \
489         fm              \
490         udapl           \
491         libmapid        \
492         libkrb5         \
493         libsmbfs        \
494         libshare        \
495         libidmap        \
496         libvscan        \
497         libgrubmgmt     \
498         smbsrv          \
499         libilb          \
500         scsi            \
501         hbaapi          \
502         smhba           \
503         libima          \
504         libsun_ima      \
505         mpapi           \
506         libreparse      \
507         $($(MACH)_HDRSUBDIRS)

509 i386_HDRSUBDIRS=       \
510         libparted       \
511         libfdisk        \
512         libsaveargs

514 sparc_HDRSUBDIRS=       \
515         libds           \
516         libdscp         \
517         libpri          \
518         libv12n         \
519         storage

521 all :=          TARGET= all
522 check :=        TARGET= check
523 clean :=        TARGET= clean
```

```
524 clobber :=      TARGET= clobber
525 install :=      TARGET= install
526 install_h :=    TARGET= install_h
527 lint :=        TARGET= lint
528 _dc :=         TARGET= _dc
529 _msg :=        TARGET= _msg

531 .KEEP_STATE:

533 #
534 # For the all and install targets, we clearly must respect library
535 # dependencies so that the libraries link correctly.  However, for
536 # the remaining targets (check, clean, clobber, install_h, lint, _dc
537 # and _msg), libraries do not have any dependencies on one another
538 # and thus respecting dependencies just slows down the build.
539 # As such, for these rules, we use pattern replacement to explicitly
540 # avoid triggering the dependency information.  Note that for clean,
541 # clobber and lint, we must use $(NOWAIT_SUBDIRS) rather than
542 # $(SUBDIRS), to prevent '.WAIT' from expanding to '.WAIT-nodepend'.
543 #

545 all:                    $(SUBDIRS)

547 install:                $(SUBDIRS) .WAIT install_extra

549 # extra libraries kept in other source areas
550 install_extra:
551         @cd ../cmd/sgs; pwd; $(MAKE) install_lib
552         @pwd

554 clean clobber lint:     $(NOWAIT_SUBDIRS:%=%-nodepend)

556 install_h check:        $(HDRSUBDIRS:%=%-nodepend)

558 _msg:                   $(MSGSUBDIRS:%=%-nodepend) .WAIT _dc

560 _dc:                    $(DCSUBDIRS:%=%-nodepend)

562 #
563 # Library interdependencies are called out explicitly here
564 #
565 auditd_plugins: libbsm libnsl libsecdb
566 gss_mechs/mech_krb5:    libgss libnsl libsocket libresolv pkcs11
567 krb5:           openssl
568 #endif /* ! codereview */
569 libadt_jni:     libbsm
570 libast:         libsocket
571 libadutils:     libldap5 libresolv libsocket libnsl
572 nsswitch:       libadutils libidmap
573 libbe:          libzfs
574 libbsm:         libtsol
575 libcmd:         libsum libast libsocket libnsl
576 libcmdutils:    libavl
577 libcontract:    libnvpair
578 libdevid:       libdevinfo
579 libdevinfo:     libnvpair libsec
580 libdhcpagent:   libsocket libdhcputil libuuid libdlpi libcontract
581 libdhcpsvc:     libinetutil
582 libdhcputil:    libnsl libgen libinetutil libdlpi
583 libdladm:       libdevinfo libinetutil libsocket libscf librcm libnvpair \
584                 libexacct libnsl libkstat libcurses
585 libdll:         libast
586 libdlpi:        libinetutil libdladm
587 libds:          libsysevent
588 libdscfg:       libnsctl libunistat libsocket libnsl
589 libdtrace:      libproc libgen libctf
```



```
590 libdtrace_jni:  libuutil libdtrace
591 libefi:         libuuid
592 libfstyp:       libnvpair
593 libelfsign:     libcryptoutil libkmf
594 libidmap:       libadutils libldap5 libavl libsldap libuutil
595 libipadm:       libnsl libinetutil libsocket libdlpi libnvpair libdhcpagent \
596                 libdladm libsecdb
597 libiscsit:      libc libnvpair libstmf libuuid libnsl
598 libkmf:         libcryptoutil pkcs11 openssl
181 libkmf:         libcryptoutil pkcs11
599 libnsl:         libmd5
600 libmapid:       libresolv
601 librdc:         libsocket libnsl libnsctl libunistat libdscfg
602 libuuid:        libdlpi
603 libinetutil:    libsocket
604 libipsecutil:   libtecla libsocket
605 libinstzones:   libzonecfg libcontract
606 libpkg:         libwanboot libscf libadm openssl
189 libpkg:         libwanboot libscf libadm
607 libnwam:        libscf
608 libsecdb:       libnsl
609 libsasl:        libgss libsocket pkcs11 libmd
610 sasl_plugins:   pkcs11 libgss libsocket libsasl
611 libsctp:        libsocket
612 libshell:       libast libcmd libdll libsocket libsecdb
613 libsip:        libmd5
614 libsmbfs:      libcmdutils libsocket libnsl libkrb5
615 libsocket:     libnsl
616 libstmfproxy:  libstmf libsocket libnsl libpthread
617 libsum:        libast
618 libsysevent:   libsecdb
619 libldap5:      libsasl libsocket libnsl libmd
620 libsldap:      libldap5 libtsol libnsl libc libscf libresolv
621 libpool:       libnvpair libexacct
622 libpp:         libast
623 libzonecfg:    libc libsocket libnsl libuuid libnvpair libsysevent libsec \
624                 libbrand libpool libscf
625 libproc:       ../cmd/sgs/librtld_db ../cmd/sgs/libelf libctf libsaveargs
626 libproject:    libpool libproc libsecdb
627 libtermcap:    libcurses
628 libtsnet:      libnsl libtsol libsecdb
629 libwrap:       libnsl libsocket
630 libwanboot:    libnvpair libresolv libnsl libsocket libdevinfo libinetutil \
631                 libdhcputil openssl
214                 libdhcputil
632 libwanbootutil: libnsl
633 pam_modules:    libproject passwdutil smbsrv
634 libscf:         libuutil libmd libgen libsmbios libnsl
635 libinetsvc:     libscf
636 librestart:     libuutil libscf
637 libsaveargs:    libdisasm
638 ../cmd/sgs/libdl:       ../cmd/sgs/libconv
639 ../cmd/sgs/libelf:      ../cmd/sgs/libconv
640 pkcs11:         libcryptoutil openssl
223 pkcs11:         libcryptoutil
641 print:          libldap5
642 udapl/udapl_tavor:      udapl/libdat
643 libzfs:         libdevid libgen libnvpair libuutil \
644                 libadm libavl libefi libidmap libmd libzfs_core
645 libzfs_core:    libnvpair
646 libzfs_jni:     libdiskmgt libnvpair libzfs
647 libzpool:       libavl libumem libnvpair libcmdutils
648 libsec:         libavl libidmap
649 brand:          libc libsocket
650 libshare:       libscf libzfs libuuid libfsmgt libsecdb libumem libsmbfs
651 libexacct/demo: libexacct libproject libsocket libnsl
```

```
652 libtsalarm:     libpcp
653 smbsrv:         libsocket libnsl libmd libxnet libpthread librt \
654                 libshare libidmap pkcs11 libsqlite libcryptoutil \
655                 libreparse libcmdutils
656 libv12n:        libds libuuid
657 libvrrpadm:     libsocket libdladm libscf
658 libvscan:       libscf
659 libfru:         libfruutils
660 scsi:           libnvpair libfru
661 mpapi:          libpthread libdevinfo libsysevent libnvpair
662 sun_fc:         libdevinfo libsysevent libnvpair
663 libsun_ima:     libdevinfo libsysevent libnsl
664 sun_sas:        libdevinfo libsysevent libnvpair libkstat libdevid
665 libgrubmgmt:    libdevinfo libzfs libfstyp
666 pylibbe:        libbe libzfs
667 pyzfs:          libnvpair libzfs
668 pysolaris:      libsec libidmap
669 libreparse:     libnvpair
670 libhotplug:     libnvpair
671 cfgadm_plugins: libhotplug
672 libilb:         libsocket
673 openssl:        libnsl libc libsocket
674 #endif /* ! codereview */
675 $(INTEL_BUILD)libdiskmgt:libfdisk

677 #
678 # The reason this rule checks for the existence of the
679 # Makefile is that some of the directories do not exist
680 # in certain situations (e.g., exportable source builds,
681 # OpenSolaris).
682 #
683 $(SUBDIRS): FRC
684         @if [ -f $@/Makefile  ]; then \
685                 cd $@; pwd; $(MAKE) $(TARGET); \
686         else \
687                 true; \
688         fi

690 $(SUBDIRS:%=%-nodepend):
691         @if [ -f $(@:%-nodepend=%)/Makefile  ]; then \
692                 cd $(@:%-nodepend=%); pwd; $(MAKE) $(TARGET); \
693         else \
694                 true; \
695         fi

697 FRC:
```

```
**********************************************************
    2278 Wed Aug 13 19:51:31 2014
new/usr/src/lib/krb5/plugins/preauth/pkinit/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#

LIBRARY= pkinit.a
VERS= .1

PKINIT_OBJS= \
        pkinit_accessor.o \
        pkinit_clnt.o \
        pkinit_crypto_openssl.o \
        pkinit_identity.o \
        pkinit_lib.o \
        pkinit_matching.o \
        pkinit_profile.o \
        pkinit_srv.o


OBJECTS= $(PKINIT_OBJS)

# include library definitions
include $(SRC)/lib/krb5/Makefile.lib

SRCS= $(PKINIT_OBJS:%.o=../%.c)

LIBS=           $(DYNLIB)

include $(SRC)/lib/gss_mechs/mech_krb5/Makefile.mech_krb5

POFILE = $(LIBRARY:%.a=%.po)
POFILES = generic.po

#override liblink
INS.liblink=    -$(RM) $@; $(SYMLINK) $(LIBLINKS)$(VERS) $@


CPPFLAGS +=     -I$(SRC)/lib/krb5 \
                -I$(SRC)/lib/krb5/kdb \
                -I$(SRC)/lib/gss_mechs/mech_krb5/include \
```



```
                -I$(SRC)/lib/gss_mechs/mech_krb5/krb5/os \
                -I$(SRC)/lib/gss_mechs/mech_krb5/include/krb5 \
                -I$(SRC)/uts/common/gssapi/include/ \
                -I$(SRC)/uts/common/gssapi/mechs/krb5/include \
                -I$(SRC)

CERRWARN        += -_gcc=-Wno-uninitialized
CERRWARN        += -_gcc=-Wno-unused-function

CFLAGS +=       $(CCVERBOSE) -I..
DYNFLAGS +=     $(KRUNPATH) $(KMECHLIB) -znodelete
LDLIBS +=       -L $(ROOTLIBDIR) -lsunw_crypto -lc
LDLIBS +=       -L $(ROOTLIBDIR) -lcrypto -lc

ROOTLIBDIR= $(ROOT)/usr/lib/krb5/plugins/preauth

$(ROOTLIBDIR):
        $(INS.dir)

.KEEP_STATE:

all:    $(LIBS)

lint:   lintcheck

# include library targets
include $(SRC)/lib/krb5/Makefile.targ

FRC:

generic.po: FRC
        $(RM) messages.po
        $(XGETTEXT) $(XGETFLAGS) `$(GREP) -l gettext ../*.[ch]`
        $(SED) "/^domain/d" messages.po > $@
        $(RM) messages.po
```

```
**********************************************************
    1949 Wed Aug 13 19:51:31 2014
new/usr/src/lib/libkmf/plugins/kmf_openssl/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  22 # Use is subject to license terms.
  23 #
  24 # Makefile for KMF Plugins
  25 #

  27 LIBRARY=        kmf_openssl.a
  28 VERS=           .1

  30 OBJECTS=        openssl_spi.o

  32 include $(SRC)/lib/Makefile.lib

  34 LIBLINKS=       $(DYNLIB:.so.1=.so)
  35 KMFINC=         -I../../../include -I../../../ber_der/inc

  37 BERLIB=         -lkmf -lkmfberder
  38 BERLIB64=       $(BERLIB)

  40 OPENSSLLIBS=    $(BERLIB) -lsunw_crypto -lcryptoutil -lc
  41 OPENSSLLIBS64=  $(BERLIB64) -lsunw_crypto -lcryptoutil -lc
  40 OPENSSLLIBS=    $(BERLIB) -lcrypto -lcryptoutil -lc
  41 OPENSSLLIBS64=  $(BERLIB64) -lcrypto -lcryptoutil -lc

  43 LINTSSLLIBS     = $(BERLIB) -lcryptoutil -lc
  44 LINTSSLLIBS64   = $(BERLIB64) -lcryptoutil -lc
  43 LINTSSLLIBS     = $(BERLIB) -lcrypto -lcryptoutil -lc
  44 LINTSSLLIBS64   = $(BERLIB64) -lcrypto -lcryptoutil -lc

  46 SRCDIR=         ../common
  47 INCDIR=         ../../include

  49 CFLAGS          +=      $(CCVERBOSE)
  50 CPPFLAGS        +=      -D_REENTRANT $(KMFINC) \
  51                         -I$(INCDIR) -I$(ADJUNCT_PROTO)/usr/include/libxml2

  53 CERRWARN        +=      -_gcc=-Wno-unused-label
  54 CERRWARN        +=      -_gcc=-Wno-unused-value
  55 CERRWARN        +=      -_gcc=-Wno-uninitialized

  57 PICS=   $(OBJECTS:%=pics/%)
```



```
  59 lint:=  OPENSSLLIBS=    $(LINTSSLLIBS)
  60 lint:=  OPENSSLLIBS64=  $(LINTSSLLIBS64)

  62 LDLIBS32        +=      $(OPENSSLLIBS)

  64 ROOTLIBDIR=     $(ROOTFS_LIBDIR)/crypto
  65 ROOTLIBDIR64=   $(ROOTFS_LIBDIR)/crypto/$(MACH64)

  67 .KEEP_STATE:

  69 LIBS    =       $(DYNLIB)
  70 all:    $(DYNLIB) $(LINTLIB)

  72 lint: lintcheck

  74 FRC:

  76 include $(SRC)/lib/Makefile.targ
```

```
**********************************************************
  133380 Wed Aug 13 19:51:31 2014
new/usr/src/lib/libkmf/plugins/kmf_openssl/common/openssl_spi.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
______unchanged_portion_omitted_

2483 /* ocsp_find_signer_sk() is copied from openssl source */
2484 static X509 *ocsp_find_signer_sk(STACK_OF(X509) *certs, OCSP_RESPID *id)
2485 {
2486         int i;
2487         unsigned char tmphash[SHA_DIGEST_LENGTH], *keyhash;

2489         /* Easy if lookup by name */
2490         if (id->type == V_OCSP_RESPID_NAME)
2491                 return (X509_find_by_subject(certs, id->value.byName));

2493         /* Lookup by key hash */

2495         /* If key hash isn't SHA1 length then forget it */
2496         if (id->value.byKey->length != SHA_DIGEST_LENGTH)
2497                 return (NULL);

2499         keyhash = id->value.byKey->data;
2500         /* Calculate hash of each key and compare */
2501         for (i = 0; i < sk_X509_num(certs); i++) {
2502                 /* LINTED E_BAD_PTR_CAST_ALIGN */
2502                 X509 *x = sk_X509_value(certs, i);
2503                 /* Use pubkey_digest to get the key ID value */
2504                 (void) X509_pubkey_digest(x, EVP_sha1(), tmphash, NULL);
2505                 if (!memcmp(keyhash, tmphash, SHA_DIGEST_LENGTH))
2506                         return (x);
2507         }
2508         return (NULL);
2509 }
______unchanged_portion_omitted_

3595 /*
3596  * Helper function to extract keys and certificates from
3597  * a single PEM file.  Typically the file should contain a
3598  * private key and an associated public key wrapped in an x509 cert.
3599  * However, the file may be just a list of X509 certs with no keys.
3600  */
3601 static KMF_RETURN
3602 extract_pem(KMF_HANDLE *kmfh,
3603         char *issuer, char *subject, KMF_BIGINT *serial,
3604         char *filename, CK_UTF8CHAR *pin,
3605         CK_ULONG pinlen, EVP_PKEY **priv_key, KMF_DATA **certs,
3606         int *numcerts)
3607 /* ARGSUSED6 */
3608 {
3609         KMF_RETURN rv = KMF_OK;
3610         FILE *fp;
3611         STACK_OF(X509_INFO) *x509_info_stack = NULL;
3612         int i, ncerts = 0, matchcerts = 0;
3613         EVP_PKEY *pkey = NULL;
3614         X509_INFO *info;
3615         X509 *x;
3616         X509_INFO **cert_infos = NULL;
3617         KMF_DATA *certlist = NULL;

3619         if (priv_key)
3620                 *priv_key = NULL;
3621         if (certs)
3622                 *certs = NULL;
3623         fp = fopen(filename, "r");
```



```
3624         if (fp == NULL)
3625                 return (KMF_ERR_OPEN_FILE);

3627         x509_info_stack = PEM_X509_INFO_read(fp, NULL, NULL, pin);
3628         if (x509_info_stack == NULL) {
3629                 (void) fclose(fp);
3630                 return (KMF_ERR_ENCODING);
3631         }
3632         cert_infos = (X509_INFO **)malloc(sk_X509_INFO_num(x509_info_stack) *
3633             sizeof (X509_INFO *));
3634         if (cert_infos == NULL) {
3635                 (void) fclose(fp);
3636                 rv = KMF_ERR_MEMORY;
3637                 goto err;
3638         }

3640         for (i = 0; i < sk_X509_INFO_num(x509_info_stack); i++) {
3642                 /* LINTED E_BAD_PTR_CAST_ALIGN */
3641                 cert_infos[ncerts] = sk_X509_INFO_value(x509_info_stack, i);
3642                 ncerts++;
3643         }

3645         if (ncerts == 0) {
3646                 (void) fclose(fp);
3647                 rv = KMF_ERR_CERT_NOT_FOUND;
3648                 goto err;
3649         }

3651         if (priv_key != NULL) {
3652                 rewind(fp);
3653                 pkey = PEM_read_PrivateKey(fp, NULL, NULL, pin);
3654         }
3655         (void) fclose(fp);

3657         x = cert_infos[ncerts - 1]->x509;
3658         /*
3659          * Make sure the private key matchs the last cert in the file.
3660          */
3661         if (pkey != NULL && !X509_check_private_key(x, pkey)) {
3662                 EVP_PKEY_free(pkey);
3663                 rv = KMF_ERR_KEY_MISMATCH;
3664                 goto err;
3665         }

3667         certlist = (KMF_DATA *)calloc(ncerts, sizeof (KMF_DATA));
3668         if (certlist == NULL) {
3669                 if (pkey != NULL)
3670                         EVP_PKEY_free(pkey);
3671                 rv = KMF_ERR_MEMORY;
3672                 goto err;
3673         }

3675         /*
3676          * Convert all of the certs to DER format.
3677          */
3678         matchcerts = 0;
3679         for (i = 0; rv == KMF_OK && certs != NULL && i < ncerts; i++) {
3680                 boolean_t match = FALSE;
3681                 info =  cert_infos[ncerts - 1 - i];

3683                 rv = check_cert(info->x509, issuer, subject, serial, &match);
3684                 if (rv != KMF_OK || match != TRUE) {
3685                         rv = KMF_OK;
3686                         continue;
3687                 }
```

```
3689                 rv = ssl_cert2KMFDATA(kmfh, info->x509,
3690                         &certlist[matchcerts++]);

3692                 if (rv != KMF_OK) {
3693                         int j;
3694                         for (j = 0; j < matchcerts; j++)
3695                                 kmf_free_data(&certlist[j]);
3696                         free(certlist);
3697                         certlist = NULL;
3698                         ncerts = matchcerts = 0;
3699                 }
3700         }

3702         if (numcerts != NULL)
3703                 *numcerts = matchcerts;

3705         if (certs != NULL)
3706                 *certs = certlist;
3707         else if (certlist != NULL) {
3708                 for (i = 0; i < ncerts; i++)
3709                         kmf_free_data(&certlist[i]);
3710                 free(certlist);
3711                 certlist = NULL;
3712         }

3714         if (priv_key == NULL && pkey != NULL)
3715                 EVP_PKEY_free(pkey);
3716         else if (priv_key != NULL && pkey != NULL)
3717                 *priv_key = pkey;

3719 err:
3720         /* Cleanup the stack of X509 info records */
3721         for (i = 0; i < sk_X509_INFO_num(x509_info_stack); i++) {
3724                 /* LINTED E_BAD_PTR_CAST_ALIGN */
3722                 info = (X509_INFO *)sk_X509_INFO_value(x509_info_stack, i);
3723                 X509_INFO_free(info);
3724         }
3725         if (x509_info_stack)
3726                 sk_X509_INFO_free(x509_info_stack);

3728         if (cert_infos != NULL)
3729                 free(cert_infos);

3731         return (rv);
3732 }

3734 static KMF_RETURN
3735 openssl_parse_bags(STACK_OF(PKCS12_SAFEBAG) *bags, char *pin,
3736         STACK_OF(EVP_PKEY) *keys, STACK_OF(X509) *certs)
3737 {
3738         KMF_RETURN ret;
3739         int i;

3741         for (i = 0; i < sk_PKCS12_SAFEBAG_num(bags); i++) {
3745                 /* LINTED E_BAD_PTR_CAST_ALIGN */
3742                 PKCS12_SAFEBAG *bag = sk_PKCS12_SAFEBAG_value(bags, i);
3743                 ret = openssl_parse_bag(bag, pin, (pin ? strlen(pin) : 0),
3744                     keys, certs);

3746                 if (ret != KMF_OK)
3747                         return (ret);
3748         }

3750         return (ret);
3751 }
```



```
3753 static KMF_RETURN
3754 set_pkey_attrib(EVP_PKEY *pkey, ASN1_TYPE *attrib, int nid)
3755 {
3756         X509_ATTRIBUTE *attr = NULL;

3758         if (pkey == NULL || attrib == NULL)
3759                 return (KMF_ERR_BAD_PARAMETER);

3761         if (pkey->attributes == NULL) {
3762                 pkey->attributes = sk_X509_ATTRIBUTE_new_null();
3763                 if (pkey->attributes == NULL)
3764                         return (KMF_ERR_MEMORY);
3765         }
3766         attr = X509_ATTRIBUTE_create(nid, attrib->type, attrib->value.ptr);
3767         if (attr != NULL) {
3768                 int i;
3769                 X509_ATTRIBUTE *a;
3770                 for (i = 0;
3771                     i < sk_X509_ATTRIBUTE_num(pkey->attributes); i++) {
3776                         /* LINTED E_BAD_PTR_CASE_ALIGN */
3772                         a = sk_X509_ATTRIBUTE_value(pkey->attributes, i);
3773                         if (OBJ_obj2nid(a->object) == nid) {
3774                                 X509_ATTRIBUTE_free(a);
3780                                 /* LINTED E_BAD_PTR_CAST_ALIGN */
3775                                 sk_X509_ATTRIBUTE_set(pkey->attributes,
3776                                     i, attr);
3777                                 return (KMF_OK);
3778                         }
3779                 }
3780                 if (sk_X509_ATTRIBUTE_push(pkey->attributes, attr) == NULL) {
3781                         X509_ATTRIBUTE_free(attr);
3782                         return (KMF_ERR_MEMORY);
3783                 }
3784         } else {
3785                 return (KMF_ERR_MEMORY);
3786         }

3788         return (KMF_OK);
3789 }
```

unchanged_portion_omitted_

```
3919 static KMF_RETURN
3920 openssl_pkcs12_parse(PKCS12 *p12, char *pin,
3921         STACK_OF(EVP_PKEY) *keys,
3922         STACK_OF(X509) *certs,
3923         STACK_OF(X509) *ca)
3924 /* ARGSUSED3 */
3925 {
3926         KMF_RETURN ret = KMF_OK;
3927         STACK_OF(PKCS7) *asafes = NULL;
3928         STACK_OF(PKCS12_SAFEBAG) *bags = NULL;
3929         int i, bagnid;
3930         PKCS7 *p7;

3932         if (p12 == NULL || (keys == NULL && certs == NULL))
3933                 return (KMF_ERR_BAD_PARAMETER);

3935         if (pin == NULL || *pin == NULL) {
3936                 if (PKCS12_verify_mac(p12, NULL, 0)) {
3937                         pin = NULL;
3938                 } else if (PKCS12_verify_mac(p12, "", 0)) {
3939                         pin = "";
3940                 } else {
3941                         return (KMF_ERR_AUTH_FAILED);
3942                 }
3943         } else if (!PKCS12_verify_mac(p12, pin, -1)) {
```

```
3944                    return (KMF_ERR_AUTH_FAILED);
3945            }

3947            if ((asafes = PKCS12_unpack_authsafes(p12)) == NULL)
3948                    return (KMF_ERR_PKCS12_FORMAT);

3950            for (i = 0; ret == KMF_OK && i < sk_PKCS7_num(asafes); i++) {
3951                    bags = NULL;
3958                    /* LINTED E_BAD_PTR_CAST_ALIGN */
3952                    p7 = sk_PKCS7_value(asafes, i);
3953                    bagnid = OBJ_obj2nid(p7->type);

3955                    if (bagnid == NID_pkcs7_data) {
3956                            bags = PKCS12_unpack_p7data(p7);
3957                    } else if (bagnid == NID_pkcs7_encrypted) {
3958                            bags = PKCS12_unpack_p7encdata(p7, pin,
3959                                (pin ? strlen(pin) : 0));
3960                    } else {
3961                            continue;
3962                    }
3963                    if (bags == NULL) {
3964                            ret = KMF_ERR_PKCS12_FORMAT;
3965                            goto out;
3966                    }

3968                    if (openssl_parse_bags(bags, pin, keys, certs) != KMF_OK)
3969                            ret = KMF_ERR_PKCS12_FORMAT;

3971                    sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
3972            }
3973    out:
3974            if (asafes != NULL)
3975                    sk_PKCS7_pop_free(asafes, PKCS7_free);

3977            return (ret);
3978    }
```

*_____unchanged_portion_omitted_*

```
4219    static X509_ATTRIBUTE *
4220    find_attr(STACK_OF(X509_ATTRIBUTE) *attrs, int nid)
4221    {
4222            X509_ATTRIBUTE *a;
4223            int i;

4225            if (attrs == NULL)
4226                    return (NULL);

4228            for (i = 0; i < sk_X509_ATTRIBUTE_num(attrs); i++) {
4236                    /* LINTED E_BAD_PTR_CAST_ALIGN */
4229                    a = sk_X509_ATTRIBUTE_value(attrs, i);
4230                    if (OBJ_obj2nid(a->object) == nid)
4231                            return (a);
4232            }
4233            return (NULL);
4234    }

4236    static KMF_RETURN
4237    convertToRawKey(EVP_PKEY *pkey, KMF_RAW_KEY_DATA *key)
4238    {
4239            KMF_RETURN rv = KMF_OK;
4240            X509_ATTRIBUTE *attr;

4242            if (pkey == NULL || key == NULL)
4243                    return (KMF_ERR_BAD_PARAMETER);
4244            /* Convert SSL key to raw key */
4245            switch (pkey->type) {
```



```
4246                    case EVP_PKEY_RSA:
4247                            rv = exportRawRSAKey(EVP_PKEY_get1_RSA(pkey),
4248                                key);
4249                            if (rv != KMF_OK)
4250                                    return (rv);
4251                            break;
4252                    case EVP_PKEY_DSA:
4253                            rv = exportRawDSAKey(EVP_PKEY_get1_DSA(pkey),
4254                                key);
4255                            if (rv != KMF_OK)
4256                                    return (rv);
4257                            break;
4258                    default:
4259                            return (KMF_ERR_BAD_PARAMETER);
4260            }
4261            /*
4262             * If friendlyName, add it to record.
4263             */
4264            attr = find_attr(pkey->attributes, NID_friendlyName);
4265            if (attr != NULL) {
4266                    ASN1_TYPE *ty = NULL;
4267                    int numattr = sk_ASN1_TYPE_num(attr->value.set);
4268                    if (attr->single == 0 && numattr > 0) {
4277                            /* LINTED E_BAD_PTR_CAST_ALIGN */
4269                            ty = sk_ASN1_TYPE_value(attr->value.set, 0);
4270                    }
4271                    if (ty != NULL) {
4272    #if OPENSSL_VERSION_NUMBER < 0x10000000L
4273                            key->label = uni2asc(ty->value.bmpstring->data,
4274                                ty->value.bmpstring->length);
4275    #else
4276                            key->label = OPENSSL_uni2asc(ty->value.bmpstring->data,
4277                                ty->value.bmpstring->length);
4278    #endif
4279                    }
4280            } else {
4281                    key->label = NULL;
4282            }

4284            /*
4285             * If KeyID, add it to record as a KMF_DATA object.
4286             */
4287            attr = find_attr(pkey->attributes, NID_localKeyID);
4288            if (attr != NULL) {
4289                    ASN1_TYPE *ty = NULL;
4290                    int numattr = sk_ASN1_TYPE_num(attr->value.set);
4291                    if (attr->single == 0 && numattr > 0) {
4301                            /* LINTED E_BAD_PTR_CAST_ALIGN */
4292                            ty = sk_ASN1_TYPE_value(attr->value.set, 0);
4293                    }
4294                    key->id.Data = (uchar_t *)malloc(
4295                        ty->value.octet_string->length);
4296                    if (key->id.Data == NULL)
4297                            return (KMF_ERR_MEMORY);
4298                    (void) memcpy(key->id.Data, ty->value.octet_string->data,
4299                        ty->value.octet_string->length);
4300                    key->id.Length = ty->value.octet_string->length;
4301            } else {
4302                    (void) memset(&key->id, 0, sizeof (KMF_DATA));
4303            }

4305            return (rv);
4306    }

4308    static KMF_RETURN
4309    convertPK12Objects(
```

```
4310         KMF_HANDLE *kmfh,
4311         STACK_OF(EVP_PKEY) *sslkeys,
4312         STACK_OF(X509) *sslcert,
4313         STACK_OF(X509) *sslcacerts,
4314         KMF_RAW_KEY_DATA **keylist, int *nkeys,
4315         KMF_X509_DER_CERT **certlist, int *ncerts)
4316 {
4317         KMF_RETURN rv = KMF_OK;
4318         KMF_RAW_KEY_DATA key;
4319         int i;

4321         for (i = 0; sslkeys != NULL && i < sk_EVP_PKEY_num(sslkeys); i++) {
4332                 /* LINTED E_BAD_PTR_CAST_ALIGN */
4322                 EVP_PKEY *pkey = sk_EVP_PKEY_value(sslkeys, i);
4323                 rv = convertToRawKey(pkey, &key);
4324                 if (rv == KMF_OK)
4325                         rv = add_key_to_list(keylist, &key, nkeys);

4327                 if (rv != KMF_OK)
4328                         return (rv);
4329         }

4331         /* Now add the certificate to the certlist */
4332         for (i = 0; sslcert != NULL && i < sk_X509_num(sslcert); i++) {
4344                 /* LINTED E_BAD_PTR_CAST_ALIGN */
4333                 X509 *cert = sk_X509_value(sslcert, i);
4334                 rv = add_cert_to_list(kmfh, cert, certlist, ncerts);
4335                 if (rv != KMF_OK)
4336                         return (rv);
4337         }

4339         /* Also add any included CA certs to the list */
4340         for (i = 0; sslcacerts != NULL && i < sk_X509_num(sslcacerts); i++) {
4341                 X509 *c;
4342                 /*
4343                  * sk_X509_value() is macro that embeds a cast to (X509 *).
4344                  * Here it translates into ((X509 *)sk_value((ca), (i))).
4345                  * Lint is complaining about the embedded casting, and
4346                  * to fix it, you need to fix openssl header files.
4347                  */
4360                 /* LINTED E_BAD_PTR_CAST_ALIGN */
4348                 c = sk_X509_value(sslcacerts, i);

4350                 /* Now add the ca cert to the certlist */
4351                 rv = add_cert_to_list(kmfh, c, certlist, ncerts);
4352                 if (rv != KMF_OK)
4353                         return (rv);
4354         }
4355         return (rv);
4356 }
```

unchanged_portion_omitted_

```
5309 KMF_RETURN
5310 OpenSSL_FindCertInCRL(KMF_HANDLE_T handle, int numattr, KMF_ATTRIBUTE *attrlist)
5311 {
5312         KMF_RETURN ret = KMF_OK;
5313         KMF_HANDLE *kmfh = (KMF_HANDLE *)handle;
5314         KMF_ENCODE_FORMAT format;
5315         BIO *in = NULL;
5316         X509   *xcert = NULL;
5317         X509_CRL   *xcrl = NULL;
5318         STACK_OF(X509_REVOKED) *revoke_stack = NULL;
5319         X509_REVOKED *revoke;
5320         int i;
5321         char *crlfilename, *crlfile, *dirpath, *certfile;
```



```
5323         if (numattr == 0 || attrlist == NULL) {
5324                 return (KMF_ERR_BAD_PARAMETER);
5325         }

5327         crlfilename = kmf_get_attr_ptr(KMF_CRL_FILENAME_ATTR,
5328             attrlist, numattr);

5330         if (crlfilename == NULL)
5331                 return (KMF_ERR_BAD_CRLFILE);

5333         certfile = kmf_get_attr_ptr(KMF_CERT_FILENAME_ATTR, attrlist, numattr);
5334         if (certfile == NULL)
5335                 return (KMF_ERR_BAD_CRLFILE);

5337         dirpath = kmf_get_attr_ptr(KMF_DIRPATH_ATTR, attrlist, numattr);

5339         crlfile = get_fullpath(dirpath, crlfilename);

5341         if (crlfile == NULL)
5342                 return (KMF_ERR_BAD_CRLFILE);

5344         if (isdir(crlfile)) {
5345                 ret = KMF_ERR_BAD_CRLFILE;
5346                 goto end;
5347         }

5349         ret = kmf_is_crl_file(handle, crlfile, &format);
5350         if (ret != KMF_OK)
5351                 goto end;

5353         /* Read the CRL file and load it into a X509_CRL structure */
5354         in = BIO_new_file(crlfilename, "rb");
5355         if (in == NULL) {
5356                 SET_ERROR(kmfh, ERR_get_error());
5357                 ret = KMF_ERR_OPEN_FILE;
5358                 goto end;
5359         }

5361         if (format == KMF_FORMAT_ASN1) {
5362                 xcrl = d2i_X509_CRL_bio(in, NULL);
5363         } else if (format == KMF_FORMAT_PEM) {
5364                 xcrl = PEM_read_bio_X509_CRL(in, NULL, NULL, NULL);
5365         }

5367         if (xcrl == NULL) {
5368                 SET_ERROR(kmfh, ERR_get_error());
5369                 ret = KMF_ERR_BAD_CRLFILE;
5370                 goto end;
5371         }
5372         (void) BIO_free(in);

5374         /* Read the Certificate file and load it into a X509 structure */
5375         ret = kmf_is_cert_file(handle, certfile, &format);
5376         if (ret != KMF_OK)
5377                 goto end;

5379         in = BIO_new_file(certfile, "rb");
5380         if (in == NULL) {
5381                 SET_ERROR(kmfh, ERR_get_error());
5382                 ret = KMF_ERR_OPEN_FILE;
5383                 goto end;
5384         }

5386         if (format == KMF_FORMAT_ASN1) {
5387                 xcert = d2i_X509_bio(in, NULL);
5388         } else if (format == KMF_FORMAT_PEM) {
```

```
5389                    xcert = PEM_read_bio_X509(in, NULL, NULL, NULL);
5390            }

5392            if (xcert == NULL) {
5393                    SET_ERROR(kmfh, ERR_get_error());
5394                    ret = KMF_ERR_BAD_CERTFILE;
5395                    goto end;
5396            }

5398            /* Check if the certificate and the CRL have same issuer */
5399            if (X509_NAME_cmp(xcert->cert_info->issuer, xcrl->crl->issuer) != 0) {
5400                    ret = KMF_ERR_ISSUER;
5401                    goto end;
5402            }

5404            /* Check to see if the certificate serial number is revoked */
5405            revoke_stack = X509_CRL_get_REVOKED(xcrl);
5406            if (sk_X509_REVOKED_num(revoke_stack) <= 0) {
5407                    /* No revoked certificates in the CRL file */
5408                    SET_ERROR(kmfh, ERR_get_error());
5409                    ret = KMF_ERR_EMPTY_CRL;
5410                    goto end;
5411            }

5413            for (i = 0; i < sk_X509_REVOKED_num(revoke_stack); i++) {
5427                    /* LINTED E_BAD_PTR_CAST_ALIGN */
5414                    revoke = sk_X509_REVOKED_value(revoke_stack, i);
5415                    if (ASN1_INTEGER_cmp(xcert->cert_info->serialNumber,
5416                        revoke->serialNumber) == 0) {
5417                            break;
5418                    }
5419            }

5421            if (i < sk_X509_REVOKED_num(revoke_stack)) {
5422                    ret = KMF_OK;
5423            } else {
5424                    ret = KMF_ERR_NOT_REVOKED;
5425            }

5427 end:
5428            if (in != NULL)
5429                    (void) BIO_free(in);
5430            if (xcrl != NULL)
5431                    X509_CRL_free(xcrl);
5432            if (xcert != NULL)
5433                    X509_free(xcert);

5435            return (ret);
5436 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    2561 Wed Aug 13 19:51:32 2014
new/usr/src/lib/libpkg/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 #
  23 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  24 # Use is subject to license terms.
  25 #

  27 LIBRARY= libpkg.a
  28 VERS=   .1

  30 # include library definitions
  31 OBJECTS=        \
  32                 canonize.o   ckparam.o    ckvolseq.o \
  33                 devtype.o    dstream.o    gpkglist.o \
  34                 gpkgmap.o    isdir.o      logerr.o \
  35                 mappath.o    ncgrpw.o     nhash.o \
  36                 pkgexecl.o   pkgexecv.o   pkgmount.o \
  37                 pkgtrans.o   ppkgmap.o \
  38                 progerr.o    putcfile.o   rrmdir.o \
  39                 runcmd.o     srchcfile.o  tputcfent.o \
  40                 verify.o     security.o   pkgweb.o \
  41                 pkgerr.o     keystore.o   p12lib.o \
  42                 vfpops.o     fmkdir.o     pkgstr.o \
  43                 handlelocalfs.o pkgserv.o


  46 # include library definitions
  47 include $(SRC)/lib/Makefile.lib

  49 SRCDIR=         ../common

  51 POFILE =        libpkg.po
  52 MSGFILES =      $(OBJECTS:%.o=../common/%.i)
  53 CLEANFILES +=   $(MSGFILES)

  55 # This library is NOT lint clean

  57 # openssl forces us to ignore dubious pointer casts, thanks to its clever
  58 # use of macros for stack management.
  59 LINTFLAGS=      -umx -errtags \
  60                 -erroff=E_BAD_PTR_CAST_ALIGN,E_BAD_PTR_CAST
  61 $(LINTLIB):=    SRCS = $(SRCDIR)/$(LINTSRC)
```



```
  64 LIBS = $(DYNLIB) $(LINTLIB)


  67 LDLIBS +=       -lc -lwanboot -lscf -ladm

  69 # libsunw_crypto and libsunw_ssl have no lint library, and so can only be used w
  70 # building
  71 $(DYNLIB) := LDLIBS += -lsunw_crypto -lsunw_ssl
  67 LDLIBS +=       -lc -lssl -lwanboot -lcrypto -lscf -ladm

  73 CFLAGS +=       $(CCVERBOSE)
  74 CERRWARN +=     -_gcc=-Wno-unused-label
  75 CERRWARN +=     -_gcc=-Wno-parentheses
  76 CERRWARN +=     -_gcc=-Wno-uninitialized
  77 CERRWARN +=     -_gcc=-Wno-clobbered
  78 CERRWARN +=     -_gcc=-Wno-switch
  79 CERRWARN +=     -_gcc=-Wno-unused-value
  80 CPPFLAGS +=     -I$(SRCDIR) -D_FILE_OFFSET_BITS=64

  82 .KEEP_STATE:

  84 all:    $(LIBS)

  86 $(POFILE): $(MSGFILES)
  87         $(BUILDPO.msgfiles)

  89 _msg: $(MSGDOMAINPOFILE)

  91 lint: lintcheck

  93 # include library targets
  94 include $(SRC)/lib/Makefile.targ
  95 include $(SRC)/Makefile.msg.targ
```

```
**********************************************************
    6803 Wed Aug 13 19:51:32 2014
new/usr/src/lib/libpkg/common/security.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
______unchanged_portion_omitted_

 79 /*
 80  * get_cert_chain - Builds a chain of certificates, from a given
 81  * user certificate to a trusted certificate.
 82  *
 83  * Arguments:
 84  * err - Error object to add errors to
 85  * cert - User cert to start with
 86  * cas - Trusted certs to use as trust anchors
 87  * chain - The resulting chain of certs (in the form of an
 88  * ordered set) is placed here.
 89  *
 90  * Returns:
 91  *   0 - Success - chain is stored in 'chain'.
 92  * non-zero - Failure, errors recorded in err
 93  */
 94 int
 95 get_cert_chain(PKG_ERR *err, X509 *cert, STACK_OF(X509) *clcerts,
 96     STACK_OF(X509) *cas, STACK_OF(X509) **chain)
 97 {
 98         X509_STORE_CTX  *store_ctx = NULL;
 99         X509_STORE      *ca_store = NULL;
100         X509            *ca_cert = NULL;
101         int i;
102         int ret = 0;

104         if ((ca_store = X509_STORE_new()) == NULL) {
105                 pkgerr_add(err, PKGERR_NOMEM,
106                     gettext(ERR_MEM));
107                 ret = 1;
108                 goto cleanup;
109         }

111         /* add all ca certs into the store */
112         for (i = 0; i < sk_X509_num(cas); i++) {
113                 /* LINTED pointer cast may result in improper alignment */
113                 ca_cert = sk_X509_value(cas, i);
114                 if (X509_STORE_add_cert(ca_store, ca_cert) == 0) {
115                         pkgerr_add(err, PKGERR_NOMEM, gettext(ERR_MEM));
116                         ret = 1;
117                         goto cleanup;
118                 }
119         }

121         /* initialize context object used during the chain resolution */

123         if ((store_ctx = X509_STORE_CTX_new()) == NULL) {
124                 pkgerr_add(err, PKGERR_NOMEM, gettext(ERR_MEM));
125                 ret = 1;
126                 goto cleanup;
127         }

129         (void) X509_STORE_CTX_init(store_ctx, ca_store, cert, clcerts);
130         /* attempt to verify the cert, which builds the cert chain */
131         if (X509_verify_cert(store_ctx) <= 0) {
132                 pkgerr_add(err, PKGERR_CHAIN,
133                     gettext(ERR_CERTCHAIN),
134                     get_subject_display_name(cert),
135                     X509_verify_cert_error_string(store_ctx->error));
136                 ret = 1;
```



```
137                 goto cleanup;
138         }
139         *chain = X509_STORE_CTX_get1_chain(store_ctx);

141 cleanup:
142         if (ca_store != NULL)
143                 (void) X509_STORE_free(ca_store);
144         if (store_ctx != NULL) {
145                 (void) X509_STORE_CTX_cleanup(store_ctx);
146                 (void) X509_STORE_CTX_free(store_ctx);
147         }

149         return (ret);
150 }

152 /*
153  * Name:                get_subject_name
154  * Description: Retrieves a name used for identifying a certificate's subject.
155  *
156  * Arguments:   cert - The certificate to get the name from
157  *
158  * Returns :    A static buffer containing the common name (CN) of the
159  *              subject of the cert.
160  *
161  *              if the CN is not available, returns a string with the entire
162  * X509 distinguished name.
163  */
164 char
165 *get_subject_display_name(X509 *cert)
166 {

168         X509_NAME       *xname;
169         static char     sname[ATTR_MAX];

171         xname = X509_get_subject_name(cert);
172         if (X509_NAME_get_text_by_NID(xname,
173             NID_commonName, sname,
174             ATTR_MAX) <= 0) {
175                 (void) strncpy(sname,
176                     X509_NAME_oneline(xname, NULL, 0), ATTR_MAX);
177                     X509_NAME_oneline(xname,
178                         NULL, 0), ATTR_MAX);
177                 sname[ATTR_MAX - 1] = '\0';
178         }
179         return (sname);
180 }

182 /*
183  * Name:                get_display_name
184  * Description: Retrieves a name used for identifying a certificate's issuer.
185  *
186  * Arguments:   cert - The certificate to get the name from
187  *
188  * Returns :    A static buffer containing the common name (CN)
189  *              of the issuer of the cert.
190  *
191  *              if the CN is not available, returns a string with the entire
192  *              X509 distinguished name.
193  */
194 char
195 *get_issuer_display_name(X509 *cert)
196 {

198         X509_NAME       *xname;
199         static char     sname[ATTR_MAX];
```

```
201         xname = X509_get_issuer_name(cert);
202         if (X509_NAME_get_text_by_NID(xname,
203             NID_commonName, sname,
204             ATTR_MAX) <= 0) {
205                 (void) strncpy(sname,
206                     X509_NAME_oneline(xname, NULL, 0), ATTR_MAX);
208                     X509_NAME_oneline(xname,
209                         NULL, 0), ATTR_MAX);
207                 sname[ATTR_MAX - 1] = '\0';
208         }
209         return (sname);
210 }
```

*unchanged_portion_omitted_*

```
**********************************************************
    2781 Wed Aug 13 19:51:32 2014
new/usr/src/lib/libwanboot/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
# Copyright (c) 2012 by Delphix. All rights reserved.
#

LIBRARY =       libwanboot.a
VERS =          .1

# List of locally located modules.
LOC_DIR =       ../common
LOC_OBJS =      socket_inet.o bootinfo_aux.o
LOC_SRCS =      $(LOC_OBJS:%.o=$(LOC_DIR)/%.c)

# List of common wanboot objects.
COM_DIR =       ../../../common/net/wanboot
COM_OBJS =      auxutil.o \
                boot_http.o \
                bootconf.o \
                bootconf_errmsg.o \
                bootinfo.o \
                bootlog.o \
                http_errorstr.o \
                p12access.o \
                p12auxpars.o \
                p12auxutl.o \
                p12err.o \
                p12misc.o \
                parseURL.o
COM_SRCS =      $(COM_OBJS:%.o=$(COM_DIR)/%.c)

# List of common DHCP modules.
DHCP_DIR =      $(SRC)/common/net/dhcp
DHCP_OBJS =     dhcpinfo.o
DHCP_SRCS =     $(DHCP_OBJS:%.o=$(DHCP_DIR)/%.c)

OBJECTS =       $(LOC_OBJS) $(COM_OBJS) $(DHCP_OBJS)

include ../../Makefile.lib

LIBS +=         $(LINTLIB)
```



```
LDLIBS +=       -lnvpair -lresolv -lnsl -lsocket -ldevinfo -ldhcputil \
                -linetutil -lc

# libsunw_crypto and libsunw_ssl have no lint library, so we can only use it whe
# building
$(DYNLIB) := LDLIBS += -lsunw_crypto -lsunw_ssl

                -linetutil -lc -lcrypto -lssl
CPPFLAGS =      -I$(SRC)/common/net/wanboot/crypt $(CPPFLAGS.master)
CERRWARN +=     -_gcc=-Wno-switch
CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-unused-value
CERRWARN +=     -_gcc=-Wno-uninitialized

# Must override SRCS from Makefile.lib since sources have
# multiple source directories.
SRCS =          $(LOC_SRCS) $(COM_SRCS) $(DHCP_SRCS)

# Must define location of lint library source.
SRCDIR =        $(LOC_DIR)
$(LINTLIB) :=   SRCS = $(SRCDIR)/$(LINTSRC)

# OpenSSL requires us to turn this off
LINTFLAGS   +=    -erroff=E_BAD_PTR_CAST_ALIGN
LINTFLAGS64 +=    -erroff=E_BAD_PTR_CAST_ALIGN

CFLAGS +=       $(CCVERBOSE)
CPPFLAGS +=     -I$(LOC_DIR) -I$(COM_DIR) -I$(DHCP_DIR)

.KEEP_STATE:

all: $(LIBS)

lint: lintcheck

pics/%.o: $(COM_DIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o: $(DHCP_DIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

include ../../Makefile.targ
```

```
**********************************************************
    3013 Wed Aug 13 19:51:33 2014
new/usr/src/lib/openssl/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2014 Alexander Pyhalov
# Use is subject to license terms.
#
# lib/openssl/Makefile
#

include $(SRC)/lib/Makefile.lib

SUBDIRS = libsunw_crypto .WAIT libsunw_ssl

# conditional assignments
all :=          TARGET= all
install :=      TARGET= install
clean :=        TARGET= clean
clobber :=      TARGET= clobber
lint :=         TARGET= lint
_msg :=         TARGET= _msg

.KEEP_STATE:

HDRS=    aes.h \
         asn1.h \
         asn1_mac.h \
         asn1t.h \
         bio.h \
         blowfish.h \
         bn.h \
         buffer.h \
         camellia.h \
         cast.h \
         cmac.h \
         cms.h \
         comp.h \
         conf.h \
         conf_api.h \
         crypto.h \
         des.h \
         des_old.h \
         dh.h \
```



```
         dsa.h \
         dso.h \
         dtls1.h \
         e_os2.h \
         ebcdic.h \
         engine.h \
         err.h \
         evp.h \
         hmac.h \
         krb5_asn.h \
         kssl.h \
         lhash.h \
         md2.h \
         md4.h \
         md5.h \
         modes.h \
         obj_mac.h \
         objects.h \
         ocsp.h \
         opensslconf.h \
         opensslv.h \
         ossl_typ.h \
         pem.h \
         pem2.h \
         pkcs12.h \
         pkcs7.h \
         pqueue.h \
         rand.h \
         rc2.h \
         rc4.h \
         ripemd.h \
         rsa.h \
         safestack.h \
         sha.h \
         srp.h \
         srtp.h \
         ssl.h \
         ssl2.h \
         ssl23.h \
         ssl3.h \
         stack.h \
         sunw_prefix.h \
         symhacks.h \
         tls1.h \
         ts.h \
         txt_db.h \
         ui.h \
         ui_compat.h \
         x509.h \
         x509_vfy.h \
         x509v3.h

HDRDIR=         include/openssl
ROOTHDRDIR=     $(ROOT)/usr/include/openssl
ROOTHDRS=       $(HDRS:%=$(ROOTHDRDIR)/%)


all install clean clobber lint _msg: $(SUBDIRS)

$(ROOTHDRS):    $(ROOTHDRDIR)

$(ROOTHDRDIR):
        $(INS.dir)

$(ROOTHDRDIR)/%: $(HDRDIR)/%
        $(INS.file)
```

```
129 install install_h:      $(ROOTHDRS)

131 $(SUBDIRS): FRC
132         @cd $@; pwd; $(MAKE) $(TARGET)

134 check:
135         @cd libsunw_ssl; pwd; $(MAKE) $@
136         @cd libsunw_crypto; pwd; $(MAKE) $@

138 FRC:
139
140 #endif /* ! codereview */
```

```
**********************************************************
    1967 Wed Aug 13 19:51:33 2014
new/usr/src/lib/openssl/README
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
  1 This is a private copy of OpenSSL library, used by illumos.
  2 Libraries are renamed to libsunw_crypto.so.1 and libsunw_ssl.so.1.
  3 All global symbols are prefixed by "sunw_" to avoid conflicts
  4 with OpenSSL library provided by distributions.
  5 To use this library with your code you should ensure that it
  6 includes opensslconf.h so that global symbols are transparently
  7 renamed.

  9 Compared to the original OpenSSL version the code is
 10 organized in the following way.
 11 Only files which are used in illumos i386/amd64 build are present.
 12 All private include files are moved to usr/src/lib/openssl/include.
 13 All public include files are moved to usr/src/lib/openssl/include/openssl.
 14 C files from ssl/ are moved to usr/src/lib/openssl/libsunw_ssl,
 15 C files from crypto/ are moved to usr/src/lib/openssl/sunw_crypto,
 16 subdirectory structure  is preserved.
 17 All Perl files used to generate assembler are moved from crypto/perlasm
 18 and crypto/*/asm to usr/src/lib/openssl/libsunw_crypto/pl/.
 19 No plain asm files are used.
 20 Scripts form https://github.com/joyent/illumos-extra/tree/master/openssl1x/tools
 21 were used to generate initial sunw_prefix.h and mapfiles.

 23 The easiest way to update OpenSSL to the next minor version (i.e. form 1.0.1X to
 24 1.0.1X+1) would be just make diff between 1.0.1X and 1.0.1X+1 and manually apply
 25 it to usr/src/lib/openssl.
 26 While doing this, take into consideration the following:
 27 - only diff for files in ssl,crypto and include subdirectories is necessary;
 28 - some files are not present in this copy of OpenSSL, so remove from diff change
 29 to the files which are not present in usr/src/lib/openssl;
 30 - some header files in original OpenSSL version are present in both include
 31 directory and ssl or crypto directories, for such files only diff for files in
 32 include directory is necessary.

 34 You can use check_symbols target to list new unprefixed function symbols which
 35 could be introduced by update. You likely want to add them to mapfile-vers and
 36 sunw_prefix.h files
 37 #endif /* ! codereview */
```

```
**********************************************************
    6271 Wed Aug 13 19:51:33 2014
new/usr/src/lib/openssl/THIRDPARTYLICENSE
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
  2  LICENSE ISSUES
  3  ==============

  5  The OpenSSL toolkit stays under a dual license, i.e. both the conditions of
  6  the OpenSSL License and the original SSLeay license apply to the toolkit.
  7  See below for the actual license texts. Actually both licenses are BSD-style
  8  Open Source licenses. In case of any license issues related to OpenSSL
  9  please contact openssl-core@openssl.org.

 11  OpenSSL License
 12  ---------------

 14 /* ====================================================================
 15  * Copyright (c) 1998-2011 The OpenSSL Project.  All rights reserved.
 16  *
 17  * Redistribution and use in source and binary forms, with or without
 18  * modification, are permitted provided that the following conditions
 19  * are met:
 20  *
 21  * 1. Redistributions of source code must retain the above copyright
 22  *    notice, this list of conditions and the following disclaimer.
 23  *
 24  * 2. Redistributions in binary form must reproduce the above copyright
 25  *    notice, this list of conditions and the following disclaimer in
 26  *    the documentation and/or other materials provided with the
 27  *    distribution.
 28  *
 29  * 3. All advertising materials mentioning features or use of this
 30  *    software must display the following acknowledgment:
 31  *    "This product includes software developed by the OpenSSL Project
 32  *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 33  *
 34  * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 35  *    endorse or promote products derived from this software without
 36  *    prior written permission. For written permission, please contact
 37  *    openssl-core@openssl.org.
 38  *
 39  * 5. Products derived from this software may not be called "OpenSSL"
 40  *    nor may "OpenSSL" appear in their names without prior written
 41  *    permission of the OpenSSL Project.
 42  *
 43  * 6. Redistributions of any form whatsoever must retain the following
 44  *    acknowledgment:
 45  *    "This product includes software developed by the OpenSSL Project
 46  *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 47  *
 48  * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 49  * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 50  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 51  * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 52  * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 53  * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 54  * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 55  * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 56  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 57  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 58  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 59  * OF THE POSSIBILITY OF SUCH DAMAGE.
 60  * ====================================================================
 61  *
```



```
 62  * This product includes cryptographic software written by Eric Young
 63  * (eay@cryptsoft.com).  This product includes software written by Tim
 64  * Hudson (tjh@cryptsoft.com).
 65  *
 66  */

 68  Original SSLeay License
 69  -----------------------

 71 /* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 72  * All rights reserved.
 73  *
 74  * This package is an SSL implementation written
 75  * by Eric Young (eay@cryptsoft.com).
 76  * The implementation was written so as to conform with Netscapes SSL.
 77  *
 78  * This library is free for commercial and non-commercial use as long as
 79  * the following conditions are aheared to.  The following conditions
 80  * apply to all code found in this distribution, be it the RC4, RSA,
 81  * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 82  * included with this distribution is covered by the same copyright terms
 83  * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 84  *
 85  * Copyright remains Eric Young's, and as such any Copyright notices in
 86  * the code are not to be removed.
 87  * If this package is used in a product, Eric Young should be given attribution
 88  * as the author of the parts of the library used.
 89  * This can be in the form of a textual message at program startup or
 90  * in documentation (online or textual) provided with the package.
 91  *
 92  * Redistribution and use in source and binary forms, with or without
 93  * modification, are permitted provided that the following conditions
 94  * are met:
 95  * 1. Redistributions of source code must retain the copyright
 96  *    notice, this list of conditions and the following disclaimer.
 97  * 2. Redistributions in binary form must reproduce the above copyright
 98  *    notice, this list of conditions and the following disclaimer in the
 99  *    documentation and/or other materials provided with the distribution.
100  * 3. All advertising materials mentioning features or use of this software
101  *    must display the following acknowledgement:
102  *    "This product includes cryptographic software written by
103  *     Eric Young (eay@cryptsoft.com)"
104  *    The word 'cryptographic' can be left out if the rouines from the library
105  *    being used are not cryptographic related :-).
106  * 4. If you include any Windows specific code (or a derivative thereof) from
107  *    the apps directory (application code) you must include an acknowledgement:
108  *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
109  *
110  * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
111  * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
112  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
113  * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
114  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
115  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
116  * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
117  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
118  * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
119  * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
120  * SUCH DAMAGE.
121  *
122  * The licence and distribution terms for any publically available version or
123  * derivative of this code cannot be changed.  i.e. this code cannot simply be
124  * copied and put under another distribution licence
125  * [including the GNU Public Licence.]
126  */
127 #endif /* ! codereview */
```

```
*********************************************************
      16 Wed Aug 13 19:51:33 2014
new/usr/src/lib/openssl/THIRDPARTYLICENSE.descrip
4853 illumos-gate is not lint-clean when built with openssl 1.0
*********************************************************
```

```
   1 OpenSSL library
   2 #endif /* ! codereview */
```

```
**********************************************************
    3525 Wed Aug 13 19:51:33 2014
new/usr/src/lib/openssl/include/aes_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes.h -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#ifndef HEADER_AES_LOCL_H
#define HEADER_AES_LOCL_H

#include <openssl/e_os2.h>

#ifdef OPENSSL_NO_AES
#error AES is disabled.
#endif

#include <stdio.h>
```



```
#include <stdlib.h>
#include <string.h>

#if defined(_MSC_VER) && (defined(_M_IX86) || defined(_M_AMD64) || defined(_M_X6
# define SWAP(x) (_lrotl(x, 8) & 0x00ff00ff | _lrotr(x, 8) & 0xff00ff00)
# define GETU32(p) SWAP(*((u32 *)(p)))
# define PUTU32(ct, st) { *((u32 *)(ct)) = SWAP((st)); }
#else
# define GETU32(pt) (((u32)(pt)[0] << 24) ^ ((u32)(pt)[1] << 16) ^ ((u32)(pt)[2]
# define PUTU32(ct, st) { (ct)[0] = (u8)((st) >> 24); (ct)[1] = (u8)((st) >> 16)
#endif

#ifdef AES_LONG
typedef unsigned long u32;
#else
typedef unsigned int u32;
#endif
typedef unsigned short u16;
typedef unsigned char u8;

#define MAXKC   (256/32)
#define MAXKB   (256/8)
#define MAXNR   14

/* This controls loop-unrolling in aes_core.c */
#undef FULL_UNROLL

#endif /* !HEADER_AES_LOCL_H */
#endif /* ! codereview */
```

```
**********************************************************
    5470 Wed Aug 13 19:51:34 2014
new/usr/src/lib/openssl/include/asn1_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* asn1t.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* Internal ASN1 structures and functions: not for application use */

/* ASN1 print context structure */
```



```
struct asn1_pctx_st
        {
        unsigned long flags;
        unsigned long nm_flags;
        unsigned long cert_flags;
        unsigned long oid_flags;
        unsigned long str_flags;
        } /* ASN1_PCTX */;

/* ASN1 public key method structure */

struct evp_pkey_asn1_method_st
        {
        int pkey_id;
        int pkey_base_id;
        unsigned long pkey_flags;

        char *pem_str;
        char *info;

        int (*pub_decode)(EVP_PKEY *pk, X509_PUBKEY *pub);
        int (*pub_encode)(X509_PUBKEY *pub, const EVP_PKEY *pk);
        int (*pub_cmp)(const EVP_PKEY *a, const EVP_PKEY *b);
        int (*pub_print)(BIO *out, const EVP_PKEY *pkey, int indent,
                                                        ASN1_PCTX *pctx);

        int (*priv_decode)(EVP_PKEY *pk, PKCS8_PRIV_KEY_INFO *p8inf);
        int (*priv_encode)(PKCS8_PRIV_KEY_INFO *p8, const EVP_PKEY *pk);
        int (*priv_print)(BIO *out, const EVP_PKEY *pkey, int indent,
                                                        ASN1_PCTX *pctx);

        int (*pkey_size)(const EVP_PKEY *pk);
        int (*pkey_bits)(const EVP_PKEY *pk);

        int (*param_decode)(EVP_PKEY *pkey,
                                const unsigned char **pder, int derlen);
        int (*param_encode)(const EVP_PKEY *pkey, unsigned char **pder);
        int (*param_missing)(const EVP_PKEY *pk);
        int (*param_copy)(EVP_PKEY *to, const EVP_PKEY *from);
        int (*param_cmp)(const EVP_PKEY *a, const EVP_PKEY *b);
        int (*param_print)(BIO *out, const EVP_PKEY *pkey, int indent,
                                                        ASN1_PCTX *pctx);
        int (*sig_print)(BIO *out,
                        const X509_ALGOR *sigalg, const ASN1_STRING *sig,
                                        int indent, ASN1_PCTX *pctx);


        void (*pkey_free)(EVP_PKEY *pkey);
        int (*pkey_ctrl)(EVP_PKEY *pkey, int op, long arg1, void *arg2);

        /* Legacy functions for old PEM */

        int (*old_priv_decode)(EVP_PKEY *pkey,
                                const unsigned char **pder, int derlen);
        int (*old_priv_encode)(const EVP_PKEY *pkey, unsigned char **pder);
        /* Custom ASN1 signature verification */
        int (*item_verify)(EVP_MD_CTX *ctx, const ASN1_ITEM *it, void *asn,
                                X509_ALGOR *a, ASN1_BIT_STRING *sig,
                                EVP_PKEY *pkey);
        int (*item_sign)(EVP_MD_CTX *ctx, const ASN1_ITEM *it, void *asn,
                                X509_ALGOR *alg1, X509_ALGOR *alg2,
                                ASN1_BIT_STRING *sig);

        } /* EVP_PKEY_ASN1_METHOD */;
```

```
128 /* Method to handle CRL access.
129  * In general a CRL could be very large (several Mb) and can consume large
130  * amounts of resources if stored in memory by multiple processes.
131  * This method allows general CRL operations to be redirected to more
132  * efficient callbacks: for example a CRL entry database.
133  */

135 #define X509_CRL_METHOD_DYNAMIC         1

137 struct x509_crl_method_st
138         {
139         int flags;
140         int (*crl_init)(X509_CRL *crl);
141         int (*crl_free)(X509_CRL *crl);
142         int (*crl_lookup)(X509_CRL *crl, X509_REVOKED **ret,
143                           ASN1_INTEGER *ser, X509_NAME *issuer);
144         int (*crl_verify)(X509_CRL *crl, EVP_PKEY *pk);
145         };
146 #endif /* ! codereview */
```

```
**********************************************************
    8789 Wed Aug 13 19:51:34 2014
new/usr/src/lib/openssl/include/bf_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/bf_locl.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_BF_LOCL_H
#define HEADER_BF_LOCL_H
#include <openssl/opensslconf.h> /* BF_PTR, BF_PTR2 */
```



```
#undef c2l
#define c2l(c,l)        (l =((unsigned long)(*((c)++)))    , \
                         l|=((unsigned long)(*((c)++)))<< 8L, \
                         l|=((unsigned long)(*((c)++)))<<16L, \
                         l|=((unsigned long)(*((c)++)))<<24L)

/* NOTE - c is not incremented as per c2l */
#undef c2ln
#define c2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))<<24L; \
                        case 7: l2|=((unsigned long)(*(--(c))))<<16L; \
                        case 6: l2|=((unsigned long)(*(--(c))))<< 8L; \
                        case 5: l2|=((unsigned long)(*(--(c))));     \
                        case 4: l1 =((unsigned long)(*(--(c))))<<24L; \
                        case 3: l1|=((unsigned long)(*(--(c))))<<16L; \
                        case 2: l1|=((unsigned long)(*(--(c))))<< 8L; \
                        case 1: l1|=((unsigned long)(*(--(c))));     \
                                } \
                        }

#undef l2c
#define l2c(l,c)        (*((c)++)=(unsigned char)(((l)     )&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24L)&0xff))

/* NOTE - c is not incremented as per l2c */
#undef l2cn
#define l2cn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)>>24L)&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>>16L)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>> 8L)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)     )&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)>>24L)&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>>16L)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>> 8L)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)     )&0xff); \
                                } \
                        }

/* NOTE - c is not incremented as per n2l */
#define n2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))    ; \
                        case 7: l2|=((unsigned long)(*(--(c))))<< 8; \
                        case 6: l2|=((unsigned long)(*(--(c))))<<16; \
                        case 5: l2|=((unsigned long)(*(--(c))))<<24; \
                        case 4: l1 =((unsigned long)(*(--(c))))    ; \
                        case 3: l1|=((unsigned long)(*(--(c))))<< 8; \
                        case 2: l1|=((unsigned long)(*(--(c))))<<16; \
                        case 1: l1|=((unsigned long)(*(--(c))))<<24; \
                                } \
                        }

/* NOTE - c is not incremented as per l2n */
#define l2nn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
```

```
128                         case 8: *(--(c))=(unsigned char)((((l2)    )&0xff); \
129                         case 7: *(--(c))=(unsigned char)((((l2)>> 8)&0xff); \
130                         case 6: *(--(c))=(unsigned char)((((l2)>>16)&0xff); \
131                         case 5: *(--(c))=(unsigned char)((((l2)>>24)&0xff); \
132                         case 4: *(--(c))=(unsigned char)((((l1)    )&0xff); \
133                         case 3: *(--(c))=(unsigned char)((((l1)>> 8)&0xff); \
134                         case 2: *(--(c))=(unsigned char)((((l1)>>16)&0xff); \
135                         case 1: *(--(c))=(unsigned char)((((l1)>>24)&0xff); \
136                                 } \
137                         }

139 #undef n2l
140 #define n2l(c,l)        (l =((unsigned long)(*((c)++)))<<24L, \
141                          l|=((unsigned long)(*((c)++)))<<16L, \
142                          l|=((unsigned long)(*((c)++)))<< 8L, \
143                          l|=((unsigned long)(*((c)++))))

145 #undef l2n
146 #define l2n(l,c)        (*((c)++)=(unsigned char)((((l)>>24L)&0xff), \
147                          *((c)++)=(unsigned char)((((l)>>16L)&0xff), \
148                          *((c)++)=(unsigned char)((((l)>> 8L)&0xff), \
149                          *((c)++)=(unsigned char)((((l)     )&0xff))

151 /* This is actually a big endian algorithm, the most significant byte
152  * is used to lookup array 0 */

154 #if defined(BF_PTR2)

156 /*
157  * This is basically a special Intel version. Point is that Intel
158  * doesn't have many registers, but offers a reach choice of addressing
159  * modes. So we spare some registers by directly traversing BF_KEY
160  * structure and hiring the most decorated addressing mode. The code
161  * generated by EGCS is *perfectly* competitive with assembler
162  * implementation!
163  */
164 #define BF_ENC(LL,R,KEY,Pi) (\
165         LL^=KEY[Pi], \
166         t=  KEY[BF_ROUNDS+2 +   0 + ((R>>24)&0xFF)], \
167         t+= KEY[BF_ROUNDS+2 + 256 + ((R>>16)&0xFF)], \
168         t^= KEY[BF_ROUNDS+2 + 512 + ((R>>8 )&0xFF)], \
169         t+= KEY[BF_ROUNDS+2 + 768 + ((R    )&0xFF)], \
170         LL^=t \
171         )

173 #elif defined(BF_PTR)

175 #ifndef BF_LONG_LOG2
176 #define BF_LONG_LOG2  2       /* default to BF_LONG being 32 bits */
177 #endif
178 #define BF_M  (0xFF<<BF_LONG_LOG2)
179 #define BF_0  (24-BF_LONG_LOG2)
180 #define BF_1  (16-BF_LONG_LOG2)
181 #define BF_2  ( 8-BF_LONG_LOG2)
182 #define BF_3  BF_LONG_LOG2 /* left shift */

184 /*
185  * This is normally very good on RISC platforms where normally you
186  * have to explicitly "multiply" array index by sizeof(BF_LONG)
187  * in order to calculate the effective address. This implementation
188  * excuses CPU from this extra work. Power[PC] uses should have most
189  * fun as (R>>BF_i)&BF_M gets folded into a single instruction, namely
190  * rlwinm. So let'em double-check if their compiler does it.
191  */

193 #define BF_ENC(LL,R,S,P) ( \
```



```
194         LL^=P, \
195         LL^= (((*(BF_LONG *)((unsigned char *)&(S[  0])+((R>>BF_0)&BF_M))+ \
196                 *(BF_LONG *)((unsigned char *)&(S[256])+((R>>BF_1)&BF_M)))^ \
197                 *(BF_LONG *)((unsigned char *)&(S[512])+((R>>BF_2)&BF_M)))+ \
198                 *(BF_LONG *)((unsigned char *)&(S[768])+((R<<BF_3)&BF_M))) \
199         )
200 #else

202 /*
203  * This is a *generic* version. Seem to perform best on platforms that
204  * offer explicit support for extraction of 8-bit nibbles preferably
205  * complemented with "multiplying" of array index by sizeof(BF_LONG).
206  * For the moment of this writing the list comprises Alpha CPU featuring
207  * extbl and s[48]addq instructions.
208  */

210 #define BF_ENC(LL,R,S,P) ( \
211         LL^=P, \
212         LL^=((( S[       ((int)(R>>24)&0xff)] + \
213                 S[0x0100+((int)(R>>16)&0xff)])^ \
214                 S[0x0200+((int)(R>> 8)&0xff)])+ \
215                 S[0x0300+((int)(R    )&0xff)])&0xffffffffL \
216         )
217 #endif

219 #endif
220 #endif /* ! codereview */
```

```
**********************************************************
   17039 Wed Aug 13 19:51:34 2014
new/usr/src/lib/openssl/include/bf_pi.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/bf_pi.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

static const BF_KEY bf_init= {
        {
        0x243f6a88L, 0x85a308d3L, 0x13198a2eL, 0x03707344L,
```



```
        0xa4093822L, 0x299f31d0L, 0x082efa98L, 0xec4e6c89L,
        0x452821e6L, 0x38d01377L, 0xbe5466cfL, 0x34e90c6cL,
        0xc0ac29b7L, 0xc97c50ddL, 0x3f84d5b5L, 0xb5470917L,
        0x9216d5d9L, 0x8979fb1b
        },{
        0xd1310ba6L, 0x98dfb5acL, 0x2ffd72dbL, 0xd01adfb7L,
        0xb8e1afedL, 0x6a267e96L, 0xba7c9045L, 0xf12c7f99L,
        0x24a19947L, 0xb3916cf7L, 0x0801f2e2L, 0x858efc16L,
        0x636920d8L, 0x71574e69L, 0xa458fea3L, 0xf4933d7eL,
        0x0d95748fL, 0x728eb658L, 0x718bcd58L, 0x82154aeeL,
        0x7b54a41dL, 0xc25a59b5L, 0x9c30d539L, 0x2af26013L,
        0xc5d1b023L, 0x286085f0L, 0xca417918L, 0xb8db38efL,
        0x8e79dcb0L, 0x603a180eL, 0x6c9e0e8bL, 0xb01e8a3eL,
        0xd71577c1L, 0xbd314b27L, 0x78af2fdaL, 0x55605c60L,
        0xe65525f3L, 0xaa55ab94L, 0x57489862L, 0x63e81440L,
        0x55ca396aL, 0x2aab10b6L, 0xb4cc5c34L, 0x1141e8ceL,
        0xa15486afL, 0x7c72e993L, 0xb3ee1411L, 0x636fbc2aL,
        0x2ba9c55dL, 0x741831f6L, 0xce5c3e16L, 0x9b87931eL,
        0xafd6ba33L, 0x6c24cf5cL, 0x7a325381L, 0x28958677L,
        0x3b8f4898L, 0x6b4bb9afL, 0xc4bfe81bL, 0x66282193L,
        0x61d809ccL, 0xfb21a991L, 0x487cac60L, 0x5dec8032L,
        0xef845d5dL, 0xe98575b1L, 0xdc262302L, 0xeb651b88L,
        0x23893e81L, 0xd396acc5L, 0x0f6d6ff3L, 0x83f44239L,
        0x2e0b4482L, 0xa4842004L, 0x69c8f04aL, 0x9e1f9b5eL,
        0x21c66842L, 0xf6e96c9aL, 0x670c9c61L, 0xabd388f0L,
        0x6a51a0d2L, 0xd8542f68L, 0x960fa728L, 0xab5133a3L,
        0x6eef0b6cL, 0x137a3be4L, 0xba3bf050L, 0x7efb2a98L,
        0xa1f1651dL, 0x39af0176L, 0x66ca593eL, 0x82430e88L,
        0x8cee8619L, 0x456f9fb4L, 0x7d84a5c3L, 0x3b8b5ebeL,
        0xe06f75d8L, 0x85c12073L, 0x401a449fL, 0x56c16aa6L,
        0x4ed3aa62L, 0x363f7706L, 0x1bfedf72L, 0x429b023dL,
        0x37d0d724L, 0xd00a1248L, 0xdb0fead3L, 0x49f1c09bL,
        0x075372c9L, 0x80991b7bL, 0x25d479d8L, 0xf6e8def7L,
        0xe3fe501aL, 0xb6794c3bL, 0x976ce0bdL, 0x04c006baL,
        0xc1a94fb6L, 0x409f60c4L, 0x5e5c9ec2L, 0x196a2463L,
        0x68fb6fafL, 0x3e6c53b5L, 0x1339b2ebL, 0x3b52ec6fL,
        0x6dfc511fL, 0x9b30952cL, 0xcc814544L, 0xaf5ebd09L,
        0xbee3d004L, 0xde334afdL, 0x660f2807L, 0x192e4bb3L,
        0xc0cba857L, 0x45c8740fL, 0xd20b5f39L, 0xb9d3fbdbL,
        0x5579c0bdL, 0x1a60320aL, 0xd6a100c6L, 0x402c7279L,
        0x679f25feL, 0xfb1fa3ccL, 0x8ea5e9f8L, 0xdb3222f8L,
        0x3c7516dfL, 0xfd616b15L, 0x2f501ec8L, 0xad0552abL,
        0x323db5faL, 0xfd238760L, 0x53317b48L, 0x3e00df82L,
        0x9e5c57bbL, 0xca6f8ca0L, 0x1a87562eL, 0xdf1769dbL,
        0xd542a8f6L, 0x287effc3L, 0xac6732c6L, 0x8c4f5573L,
        0x695b27b0L, 0xbbca58c8L, 0xe1ffa35dL, 0xb8f011a0L,
        0x10fa3d98L, 0xfd2183b8L, 0x4afcb56cL, 0x2dd1d35bL,
        0x9a53e479L, 0xb6f84565L, 0xd28e49bcL, 0x4bfb9790L,
        0xe1ddf2daL, 0xa4cb7e33L, 0x62fb1341L, 0xcee4c6e8L,
        0xef20cadaL, 0x36774c01L, 0xd07e9efeL, 0x2bf11fb4L,
        0x95dbda4dL, 0xae909198L, 0xeaad8e71L, 0x6b93d5a0L,
        0xd08ed1d0L, 0xafc725e0L, 0x8e3c5b2fL, 0x8e7594b7L,
        0x8ff6e2fbL, 0xf2122b64L, 0x8888b812L, 0x900df01cL,
        0x4fad5ea0L, 0x688fc31cL, 0xd1cff191L, 0xb3a8c1adL,
        0x2f2f2218L, 0xbe0e1777L, 0xea752dfeL, 0x8b021fa1L,
        0xe5a0cc0fL, 0xb56f74e8L, 0x18acf3d6L, 0xce89e299L,
        0xb4a84fe0L, 0xfd13e0b7L, 0x7cc43b81L, 0xd2ada8d9L,
        0x165fa266L, 0x80957705L, 0x93cc7314L, 0x211a1477L,
        0xe6ad2065L, 0x77b5fa86L, 0xc75442f5L, 0xfb9d35cfL,
        0xebcdaf0cL, 0x7b3e89a0L, 0xd6411bd3L, 0xae1e7e49L,
        0x00250e2dL, 0x2071b35eL, 0x226800bbL, 0x57b8e0afL,
        0x2464369bL, 0xf009b91eL, 0x5563911dL, 0x59dfa6aaL,
        0x78c14389L, 0xd95a537fL, 0x207d5ba2L, 0x02e5b9c5L,
        0x83260376L, 0x6295cfa9L, 0x11c81968L, 0x4e734a41L,
        0xb3472dcaL, 0x7b14a94aL, 0x1b510052L, 0x9a532915L,
        0xd60f573fL, 0xbc9bc6e4L, 0x2b60a476L, 0x81e67400L,
```

```
128         0x08ba6fb5L, 0x571be91fL, 0xf296ec6bL, 0x2a0dd915L,
129         0xb6636521L, 0xe7b9f9b6L, 0xff34052eL, 0xc5855664L,
130         0x53b02d5dL, 0xa99f8fa1L, 0x08ba4799L, 0x6e85076aL,
131         0x4b7a70e9L, 0xb5b32944L, 0xdb75092eL, 0xc4192623L,
132         0xad6ea6b0L, 0x49a7df7dL, 0x9cee60b8L, 0x8fedb266L,
133         0xecaa8c71L, 0x699a17ffL, 0x5664526cL, 0xc2b19ee1L,
134         0x193602a5L, 0x75094c29L, 0xa0591340L, 0xe4183a3eL,
135         0x3f54989aL, 0x5b429d65L, 0x6b8fe4d6L, 0x99f73fd6L,
136         0xa1d29c07L, 0xefe830f5L, 0x4d2d38e6L, 0xf0255dc1L,
137         0x4cdd2086L, 0x8470eb26L, 0x6382e9c6L, 0x021ecc5eL,
138         0x09686b3fL, 0x3ebaefc9L, 0x3c971814L, 0x6b6a70a1L,
139         0x687f3584L, 0x52a0e286L, 0xb79c5305L, 0xaa500737L,
140         0x3e07841cL, 0x7fdeae5cL, 0x8e7d44ecL, 0x5716f2b8L,
141         0xb03ada37L, 0xf0500c0dL, 0xf01c1f04L, 0x0200b3ffL,
142         0xae0cf51aL, 0x3cb574b2L, 0x25837a58L, 0xdc0921bdL,
143         0xd19113f9L, 0x7ca92ff6L, 0x94324773L, 0x22f54701L,
144         0x3ae5e581L, 0x37c2dadcL, 0xc8b57634L, 0x9af3dda7L,
145         0xa9446146L, 0x0fd0030eL, 0xecc8c73eL, 0xa4751e41L,
146         0xe238cd99L, 0x3bea0e2fL, 0x3280bba1L, 0x183eb331L,
147         0x4e548b38L, 0x4f6db908L, 0x6f420d03L, 0xf60a04bfL,
148         0x2cb81290L, 0x24977c79L, 0x5679b072L, 0xbcaf89afL,
149         0xde9a771fL, 0xd9930810L, 0xb38bae12L, 0xdccf3f2eL,
150         0x5512721fL, 0x2e6b7124L, 0x501adde6L, 0x9f84cd87L,
151         0x7a584718L, 0x7408da17L, 0xbc9f9abcL, 0xe94b7d8cL,
152         0xec7aec3aL, 0xdb851dfaL, 0x63094366L, 0xc464c3d2L,
153         0xef1c1847L, 0x3215d908L, 0xdd433b37L, 0x24c2ba16L,
154         0x12a14d43L, 0x2a65c451L, 0x50940002L, 0x133ae4ddL,
155         0x71dff89eL, 0x10314e55L, 0x81ac77d6L, 0x5f11199bL,
156         0x043556f1L, 0xd7a3c76bL, 0x3c11183bL, 0x5924a509L,
157         0xf28fe6edL, 0x97f1fbfaL, 0x9ebabf2cL, 0x1e153c6eL,
158         0x86e34570L, 0xeae96fb1L, 0x860e5e0aL, 0x5a3e2ab3L,
159         0x771fe71cL, 0x4e3d06faL, 0x2965dcb9L, 0x99e71d0fL,
160         0x803e89d6L, 0x5266c825L, 0x2e4cc978L, 0x9c10b36aL,
161         0xc6150ebaL, 0x94e2ea78L, 0xa5fc3c53L, 0x1e0a2df4L,
162         0xf2f74ea7L, 0x361d2b3dL, 0x1939260fL, 0x19c27960L,
163         0x5223a708L, 0xf71312b6L, 0xebadfe6eL, 0xeac31f66L,
164         0xe3bc4595L, 0xa67bc883L, 0xb17f37d1L, 0x018cff28L,
165         0xc332ddefL, 0xbe6c5aa5L, 0x65582185L, 0x68ab9802L,
166         0xeecea50fL, 0xdb2f953bL, 0x2aef7dadL, 0x5b6e2f84L,
167         0x1521b628L, 0x29076170L, 0xecdd4775L, 0x619f1510L,
168         0x13cca830L, 0xeb61bd96L, 0x0334fe1eL, 0xaa0363cfL,
169         0xb5735c90L, 0x4c70a239L, 0xd59e9e0bL, 0xcbaade14L,
170         0xeecc86bcL, 0x60622ca7L, 0x9cab5cabL, 0xb2f3846eL,
171         0x648b1eafL, 0x19bdf0caL, 0xa02369b9L, 0x655abb50L,
172         0x40685a32L, 0x3c2ab4b3L, 0x319ee9d5L, 0xc021b8f7L,
173         0x9b540b19L, 0x875fa099L, 0x95f7997eL, 0x623d7da8L,
174         0xf837889aL, 0x97e32d77L, 0x11ed935fL, 0x16681281L,
175         0x0e358829L, 0xc7e61fd6L, 0x96dedfa1L, 0x7858ba99L,
176         0x57f584a5L, 0x1b227263L, 0x9b83c3ffL, 0x1ac24696L,
177         0xcdb30aebL, 0x532e3054L, 0x8fd948e4L, 0x6dbc3128L,
178         0x58ebf2efL, 0x34c6ffeaL, 0xfe28ed61L, 0xee7c3c73L,
179         0x5d4a14d9L, 0xe864b7e3L, 0x42105d14L, 0x203e13e0L,
180         0x45eee2b6L, 0xa3aaabeaL, 0xdb6c4f15L, 0xfacb4fd0L,
181         0xc742f442L, 0xef6abbb5L, 0x654f3b1dL, 0x41cd2105L,
182         0xd81e799eL, 0x86854dc7L, 0xe44b476aL, 0x3d816250L,
183         0xcf62a1f2L, 0x5b8d2646L, 0xfc8883a0L, 0xc1c7b6a3L,
184         0x7f1524c3L, 0x69cb7492L, 0x47848a0bL, 0x5692b285L,
185         0x095bbf00L, 0xad19489dL, 0x1462b174L, 0x23820e00L,
186         0x58428d2aL, 0x0c55f5eaL, 0x1dadf43eL, 0x233f7061L,
187         0x3372f092L, 0x8d937e41L, 0xd65fecf1L, 0x6c223bdbL,
188         0x7cde3759L, 0xcbee7460L, 0x4085f2a7L, 0xce77326eL,
189         0xa6078084L, 0x19f8509eL, 0xe8efd855L, 0x61d99735L,
190         0xa969a7aaL, 0xc50c06c2L, 0x5a04abfcL, 0x800bcadcL,
191         0x9e447a2eL, 0xc3453484L, 0xfdd56705L, 0x0e1e9ec9L,
192         0xdb73dbd3L, 0x105588cdL, 0x675fda79L, 0xe3674340L,
193         0xc5c43465L, 0x713e38d8L, 0x3d28f89eL, 0xf16dff20L,
```



```
194         0x153e21e7L, 0x8fb03d4aL, 0xe6e39f2bL, 0xdb83adf7L,
195         0xe93d5a68L, 0x948140f7L, 0xf64c261cL, 0x94692934L,
196         0x411520f7L, 0x7602d4f7L, 0xbcf46b2eL, 0xd4a20068L,
197         0xd4082471L, 0x3320f46aL, 0x43b7d4b7L, 0x500061afL,
198         0x1e39f62eL, 0x97244546L, 0x14214f74L, 0xbf8b8840L,
199         0x4d95fc1dL, 0x96b591afL, 0x70f4ddd3L, 0x66a02f45L,
200         0xbfbc09ecL, 0x03bd9785L, 0x7fac6dd0L, 0x31cb8504L,
201         0x96eb27b3L, 0x55fd3941L, 0xda2547e6L, 0xabca0a9aL,
202         0x28507825L, 0x530429f4L, 0x0a2c86daL, 0xe9b66dfbL,
203         0x68dc1462L, 0xd7486900L, 0x680ec0a4L, 0x27a18deeL,
204         0x4f3ffea2L, 0xe887ad8cL, 0xb58ce006L, 0x7af4d6b6L,
205         0xaace1e7cL, 0xd3375fecL, 0xce78a399L, 0x406b2a42L,
206         0x20fe9e35L, 0xd9f385b9L, 0xee39d7abL, 0x3b124e8bL,
207         0x1dc9faf7L, 0x4b6d1856L, 0x26a36631L, 0xeae397b2L,
208         0x3a6efa74L, 0xdd5b4332L, 0x6841e7f7L, 0xca7820fbL,
209         0xfb0af54eL, 0xd8feb397L, 0x454056acL, 0xba489527L,
210         0x55533a3aL, 0x20838d87L, 0xfe6ba9b7L, 0xd096954bL,
211         0x55a867bcL, 0xa1159a58L, 0xcca92963L, 0x99e1db33L,
212         0xa62a4a56L, 0x3f3125f9L, 0x5ef47e1cL, 0x9029317cL,
213         0xfdf8e802L, 0x04272f70L, 0x80bb155cL, 0x05282ce3L,
214         0x95c11548L, 0xe4c66d22L, 0x48c1133fL, 0xc70f86dcL,
215         0x07f9c9eeL, 0x41041f0fL, 0x404779a4L, 0x5d886e17L,
216         0x325f51ebL, 0xd59bc0d1L, 0xf2bcc18fL, 0x41113564L,
217         0x257b7834L, 0x602a9c60L, 0xdff8e8a3L, 0x1f636c1bL,
218         0x0e12b4c2L, 0x02e1329eL, 0xaf664fd1L, 0xcad18115L,
219         0x6b2395e0L, 0x333e92e1L, 0x3b240b62L, 0xeebeb922L,
220         0x85b2a20eL, 0xe6ba0d99L, 0xde720c8cL, 0x2da2f728L,
221         0xd0127845L, 0x95b794fdL, 0x647d0862L, 0xe7ccf5f0L,
222         0x5449a36fL, 0x877d48faL, 0xc39dfd27L, 0xf33e8d1eL,
223         0x0a476341L, 0x992eff74L, 0x3a6f6eabL, 0xf4f8fd37L,
224         0xa812dc60L, 0xa1ebddf8L, 0x991be14cL, 0xdb6e6b0dL,
225         0xc67b5510L, 0x6d672c37L, 0x2765d43bL, 0xdcd0e804L,
226         0xf1290dc7L, 0xcc00ffa3L, 0xb5390f92L, 0x690fed0bL,
227         0x667b9ffbL, 0xcedb7d9cL, 0xa091cf0bL, 0xd9155ea3L,
228         0xbb132f88L, 0x515bad24L, 0x7b9479bfL, 0x763bd6ebL,
229         0x37392eb3L, 0xcc115979L, 0x8026e297L, 0xf42e312dL,
230         0x6842ada7L, 0xc66a2b3bL, 0x12754cccL, 0x782ef11cL,
231         0x6a124237L, 0xb79251e7L, 0x06a1bbe6L, 0x4bfb6350L,
232         0x1a6b1018L, 0x11caedfaL, 0x3d25bdd8L, 0xe2e1c3c9L,
233         0x44421659L, 0x0a121386L, 0xd90cec6eL, 0xd5abea2aL,
234         0x64af674eL, 0xda86a85fL, 0xbebfe988L, 0x64e4c3feL,
235         0x9dbc8057L, 0xf0f7c086L, 0x60787bf8L, 0x6003604dL,
236         0xd1fd8346L, 0xf6381fb0L, 0x7745ae04L, 0xd736fcccL,
237         0x83426b33L, 0xf01eab71L, 0xb0804187L, 0x3c005e5fL,
238         0x77a057beL, 0xbde8ae24L, 0x55464299L, 0xbf582e61L,
239         0x4e58f48fL, 0xf2ddfda2L, 0xf474ef38L, 0x8789bdc2L,
240         0x5366f9c3L, 0xc8b38e74L, 0xb475f255L, 0x46fcd9b9L,
241         0x7aeb2661L, 0x8b1ddf84L, 0x846a0e79L, 0x915f95e2L,
242         0x466e598eL, 0x20b45770L, 0x8cd55591L, 0xc902de4cL,
243         0xb90bace1L, 0xbb8205d0L, 0x11a86248L, 0x7574a99eL,
244         0xb77f19b6L, 0xe0a9dc09L, 0x662d09a1L, 0xc4324633L,
245         0xe85a1f02L, 0x09f0be8cL, 0x4a99a025L, 0x1d6efe10L,
246         0x1ab93d1dL, 0x0ba5a4dfL, 0xa186f20fL, 0x2868f169L,
247         0xdcb7da83L, 0x573906feL, 0xa1e2ce9bL, 0x4fcd7f52L,
248         0x50115e01L, 0xa70683faL, 0xa002b5c4L, 0x0de6d027L,
249         0x9af88c27L, 0x773f8641L, 0xc3604c06L, 0x61a806b5L,
250         0xf0177a28L, 0xc0f586e0L, 0x006058aaL, 0x30dc7d62L,
251         0x11e69ed7L, 0x2338ea63L, 0x53c2dd94L, 0xc2c21634L,
252         0xbbcbee56L, 0x90bcb6deL, 0xebfc7da1L, 0xce591d76L,
253         0x6f05e409L, 0x4b7c0188L, 0x39720a3dL, 0x7c927c24L,
254         0x86e3725fL, 0x724d9db9L, 0x1ac15bb4L, 0xd39eb8fcL,
255         0xed545578L, 0x08fca5b5L, 0xd83d7cd3L, 0x4dad0fc4L,
256         0x1e50ef5eL, 0xb161e6f8L, 0xa28514d9L, 0x6c51133cL,
257         0x6fd5c7e7L, 0x56e14ec4L, 0x362abfceL, 0xddc6c837L,
258         0xd79a3234L, 0x92638212L, 0x670efa8eL, 0x406000e0L,
259         0x3a39ce37L, 0xd3faf5cfL, 0xabc27737L, 0x5ac52d1bL,
```

```
260         0x5cb0679eL, 0x4fa33742L, 0xd3822740L, 0x99bc9bbeL,
261         0xd5118e9dL, 0xbf0f7315L, 0xd62d1c7eL, 0xc700c47bL,
262         0xb78c1b6bL, 0x21a19045L, 0xb26eb1beL, 0x6a366eb4L,
263         0x5748ab2fL, 0xbc946e79L, 0xc6a376d2L, 0x6549c2c8L,
264         0x530ff8eeL, 0x468dde7dL, 0xd5730a1dL, 0x4cd04dc6L,
265         0x2939bbdbL, 0xa9ba4650L, 0xac9526e8L, 0xbe5ee304L,
266         0xa1fad5f0L, 0x6a2d519aL, 0x63ef8ce2L, 0x9a86ee22L,
267         0xc089c2b8L, 0x43242ef6L, 0xa51e03aaL, 0x9cf2d0a4L,
268         0x83c061baL, 0x9be96a4dL, 0x8fe51550L, 0xba645bd6L,
269         0x2826a2f9L, 0xa73a3ae1L, 0x4ba99586L, 0xef5562e9L,
270         0xc72fefd3L, 0xf752f7daL, 0x3f046f69L, 0x77fa0a59L,
271         0x80e4a915L, 0x87b08601L, 0x9b09e6adL, 0x3b3ee593L,
272         0xe990fd5aL, 0x9e34d797L, 0x2cf0b7d9L, 0x022b8b51L,
273         0x96d5ac3aL, 0x017da67dL, 0xd1cf3ed6L, 0x7c7d2d28L,
274         0x1f9f25cfL, 0xadf2b89bL, 0x5ad6b472L, 0x5a88f54cL,
275         0xe029ac71L, 0xe019a5e6L, 0x47b0acfdL, 0xed93fa9bL,
276         0xe8d3c48dL, 0x283b57ccL, 0xf8d56629L, 0x79132e28L,
277         0x785f0191L, 0xed756055L, 0xf7960e44L, 0xe3d35e8cL,
278         0x15056dd4L, 0x88f46dbaL, 0x03a16125L, 0x0564f0bdL,
279         0xc3eb9e15L, 0x3c9057a2L, 0x97271aecL, 0xa93a072aL,
280         0x1b3f6d9bL, 0x1e6321f5L, 0xf59c66fbL, 0x26dcf319L,
281         0x7533d928L, 0xb155fdf5L, 0x03563482L, 0x8aba3cbbL,
282         0x28517711L, 0xc20ad9f8L, 0xabcc5167L, 0xccad925fL,
283         0x4de81751L, 0x3830dc8eL, 0x379d5862L, 0x9320f991L,
284         0xea7a90c2L, 0xfb3e7bceL, 0x5121ce64L, 0x774fbe32L,
285         0xa8b6e37eL, 0xc3293d46L, 0x48de5369L, 0x6413e680L,
286         0xa2ae0810L, 0xdd6db224L, 0x69852dfdL, 0x09072166L,
287         0xb39a460aL, 0x6445c0ddL, 0x586cdecfL, 0x1c20c8aeL,
288         0x5bbef7ddL, 0x1b588d40L, 0xccd2017fL, 0x6bb4e3bbL,
289         0xdda26a7eL, 0x3a59ff45L, 0x3e350a44L, 0xbcb4cdd5L,
290         0x72eacea8L, 0xfa6484bbL, 0x8d6612aeL, 0xbf3c6f47L,
291         0xd29be463L, 0x542f5d9eL, 0xaec2771bL, 0xf64e6370L,
292         0x740e0d8dL, 0xe75b1357L, 0xf8721671L, 0xaf537d5dL,
293         0x4040cb08L, 0x4eb4e2ccL, 0x34d2466aL, 0x0115af84L,
294         0xe1b00428L, 0x95983a1dL, 0x06b89fb4L, 0xce6ea048L,
295         0x6f3f3b82L, 0x3520ab82L, 0x011a1d4bL, 0x277227f8L,
296         0x611560b1L, 0xe7933fdcL, 0xbb3a792bL, 0x344525bdL,
297         0xa08839e1L, 0x51ce794bL, 0x2f32c9b7L, 0xa01fbac9L,
298         0xe01cc87eL, 0xbcc7d1f6L, 0xcf0111c3L, 0xa1e8aac7L,
299         0x1a908749L, 0xd44fbd9aL, 0xd0dadecbL, 0xd50ada38L,
300         0x0339c32aL, 0xc6913667L, 0x8df9317cL, 0xe0b12b4fL,
301         0xf79e59b7L, 0x43f5bb3aL, 0xf2d519ffL, 0x27d9459cL,
302         0xbf97222cL, 0x15e6fc2aL, 0x0f91fc71L, 0x9b941525L,
303         0xfae59361L, 0xceb69cebL, 0xc2a86459L, 0x12baa8d1L,
304         0xb6c1075eL, 0xe3056a0cL, 0x10d25065L, 0xcb03a442L,
305         0xe0ec6e0eL, 0x1698db3bL, 0x4c98a0beL, 0x3278e964L,
306         0x9f1f9532L, 0xe0d392dfL, 0xd3a0342bL, 0x8971f21eL,
307         0x1b0a7441L, 0x4ba3348cL, 0xc5be7120L, 0xc37632d8L,
308         0xdf359f8dL, 0x9b992f2eL, 0xe60b6f47L, 0x0fe3f11dL,
309         0xe54cda54L, 0x1edad891L, 0xce6279cfL, 0xcd3e7e6fL,
310         0x1618b166L, 0xfd2c1d05L, 0x848fd2c5L, 0xf6fb2299L,
311         0xf523f357L, 0xa6327623L, 0x93a83531L, 0x56cccd02L,
312         0xacf08162L, 0x5a75ebb5L, 0x6e163697L, 0x88d273ccL,
313         0xde966292L, 0x81b949d0L, 0x4c50901bL, 0x71c65614L,
314         0xe6c6c7bdL, 0x327a140aL, 0x45e1d006L, 0xc3f27b9aL,
315         0xc9aa53fdL, 0x62a80f00L, 0xbb25bfe2L, 0x35bdd2f6L,
316         0x71126905L, 0xb2040222L, 0xb6cbcf7cL, 0xcd769c2bL,
317         0x53113ec0L, 0x1640e3d3L, 0x38abbd60L, 0x2547adf0L,
318         0xba38209cL, 0xf746ce76L, 0x77afa1c5L, 0x20756060L,
319         0x85cbfe4eL, 0x8ae88dd8L, 0x7aaaf9b0L, 0x4cf9aa7eL,
320         0x1948c25cL, 0x02fb8a8cL, 0x01c36ae4L, 0xd6ebe1f9L,
321         0x90d4f869L, 0xa65cdea0L, 0x3f09252dL, 0xc208e69fL,
322         0xb74e6132L, 0xce77e25bL, 0x578fdfe3L, 0x3ac372e6L,
323         }
324         };
325 #endif /* ! codereview */
```

****
771 Wed Aug 13 19:51:34 2014
new/usr/src/lib/openssl/include/bio_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
#include <openssl/bio.h>

#if BIO_FLAGS_UPLINK==0
/* Shortcut UPLINK calls on most platforms... */
#define UP_stdin        stdin
#define UP_stdout       stdout
#define UP_stderr       stderr
#define UP_fprintf      fprintf
#define UP_fgets        fgets
#define UP_fread        fread
#define UP_fwrite       fwrite
#undef  UP_fsetmod
#define UP_feof         feof
#define UP_fclose       fclose

#define UP_fopen        fopen
#define UP_fseek        fseek
#define UP_ftell        ftell
#define UP_fflush       fflush
#define UP_ferror       ferror
#ifdef _WIN32
#define UP_fileno       _fileno
#define UP_open         _open
#define UP_read         _read
#define UP_write        _write
#define UP_lseek        _lseek
#define UP_close        _close
#else
#define UP_fileno       fileno
#define UP_open         open
#define UP_read         read
#define UP_write        write
#define UP_lseek        lseek
#define UP_close        close
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   17173 Wed Aug 13 19:51:34 2014
new/usr/src/lib/openssl/include/bn_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/bn/bn_lcl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_BN_LCL_H
#define HEADER_BN_LCL_H

#include <openssl/bn.h>

#ifdef  __cplusplus
extern "C" {
#endif


/*
 * BN_window_bits_for_exponent_size -- macro for sliding window mod_exp function
 *
 *
 * For window size 'w' (w >= 2) and a random 'b' bits exponent,
 * the number of multiplications is a constant plus on average
```

```
 *
 *    2^(w-1) + (b-w)/(w+1);
 *
 * here  2^(w-1)  is for precomputing the table (we actually need
 * entries only for windows that have the lowest bit set), and
 * (b-w)/(w+1)  is an approximation for the expected number of
 * w-bit windows, not counting the first one.
 *
 * Thus we should use
 *
 *    w >= 6  if        b > 671
 *     w = 5  if  671 > b > 239
 *     w = 4  if  239 > b >  79
 *     w = 3  if   79 > b >  23
 *    w <= 2  if   23 > b
 *
 * (with draws in between).  Very small exponents are often selected
 * with low Hamming weight, so we use  w = 1  for b <= 23.
 */
#if 1
#define BN_window_bits_for_exponent_size(b) \
                ((b) > 671 ? 6 : \
                 (b) > 239 ? 5 : \
                 (b) >  79 ? 4 : \
                 (b) >  23 ? 3 : 1)
#else
/* Old SSLeay/OpenSSL table.
 * Maximum window size was 5, so this table differs for b==1024;
 * but it coincides for other interesting values (b==160, b==512).
 */
#define BN_window_bits_for_exponent_size(b) \
                ((b) > 255 ? 5 : \
                 (b) > 127 ? 4 : \
                 (b) >  17 ? 3 : 1)
#endif



/* BN_mod_exp_mont_conttime is based on the assumption that the
 * L1 data cache line width of the target processor is at least
 * the following value.
 */
#define MOD_EXP_CTIME_MIN_CACHE_LINE_WIDTH      ( 64 )
#define MOD_EXP_CTIME_MIN_CACHE_LINE_MASK       (MOD_EXP_CTIME_MIN_CACHE_LINE_WI

/* Window sizes optimized for fixed window size modular exponentiation
 * algorithm (BN_mod_exp_mont_consttime).
 *
 * To achieve the security goals of BN_mode_exp_mont_consttime, the
 * maximum size of the window must not exceed
 * log_2(MOD_EXP_CTIME_MIN_CACHE_LINE_WIDTH).
 *
 * Window size thresholds are defined for cache line sizes of 32 and 64,
 * cache line sizes where log_2(32)=5 and log_2(64)=6 respectively. A
 * window size of 7 should only be used on processors that have a 128
 * byte or greater cache line size.
 */
#if MOD_EXP_CTIME_MIN_CACHE_LINE_WIDTH == 64

#  define BN_window_bits_for_ctime_exponent_size(b) \
                ((b) > 937 ? 6 : \
                 (b) > 306 ? 5 : \
                 (b) >  89 ? 4 : \
                 (b) >  22 ? 3 : 1)
#  define BN_MAX_WINDOW_BITS_FOR_CTIME_EXPONENT_SIZE    (6)
```



```
#elif MOD_EXP_CTIME_MIN_CACHE_LINE_WIDTH == 32

#  define BN_window_bits_for_ctime_exponent_size(b) \
                ((b) > 306 ? 5 : \
                 (b) >  89 ? 4 : \
                 (b) >  22 ? 3 : 1)
#  define BN_MAX_WINDOW_BITS_FOR_CTIME_EXPONENT_SIZE    (5)

#endif


/* Pentium pro 16,16,16,32,64 */
/* Alpha       16,16,16,16.64 */
#define BN_MULL_SIZE_NORMAL                     (16) /* 32 */
#define BN_MUL_RECURSIVE_SIZE_NORMAL            (16) /* 32 less than */
#define BN_SQR_RECURSIVE_SIZE_NORMAL            (16) /* 32 */
#define BN_MUL_LOW_RECURSIVE_SIZE_NORMAL        (32) /* 32 */
#define BN_MONT_CTX_SET_SIZE_WORD               (64) /* 32 */

#if !defined(OPENSSL_NO_ASM) && !defined(OPENSSL_NO_INLINE_ASM) && !defined(PEDA
/*
 * BN_UMULT_HIGH section.
 *
 * No, I'm not trying to overwhelm you when stating that the
 * product of N-bit numbers is 2*N bits wide:-) No, I don't expect
 * you to be impressed when I say that if the compiler doesn't
 * support 2*N integer type, then you have to replace every N*N
 * multiplication with 4 (N/2)*(N/2) accompanied by some shifts
 * and additions which unavoidably results in severe performance
 * penalties. Of course provided that the hardware is capable of
 * producing 2*N result... That's when you normally start
 * considering assembler implementation. However! It should be
 * pointed out that some CPUs (most notably Alpha, PowerPC and
 * upcoming IA-64 family:-) provide *separate* instruction
 * calculating the upper half of the product placing the result
 * into a general purpose register. Now *if* the compiler supports
 * inline assembler, then it's not impossible to implement the
 * "bignum" routines (and have the compiler optimize 'em)
 * exhibiting "native" performance in C. That's what BN_UMULT_HIGH
 * macro is about:-)
 *
 *                                      <appro@fy.chalmers.se>
 */
# if defined(__alpha) && (defined(SIXTY_FOUR_BIT_LONG) || defined(SIXTY_FOUR_BIT
#  if defined(__DECC)
#   include <c_asm.h>
#   define BN_UMULT_HIGH(a,b)   (BN_ULONG)__asm__("umulh %a0,%a1,%v0",(a),(b))
#  elif defined(__GNUC__) && __GNUC__>=2
#   define BN_UMULT_HIGH(a,b)   ({      \
        register BN_ULONG ret;          \
        __asm__ ("umulh %1,%2,%0"       \
             : "=r"(ret)                \
             : "r"(a), "r"(b));         \
        ret;                    })
#  endif        /* compiler */
# elif defined(_ARCH_PPC) && defined(__64BIT__) && defined(SIXTY_FOUR_BIT_LONG)
#  if defined(__GNUC__) && __GNUC__>=2
#   define BN_UMULT_HIGH(a,b)   ({      \
        register BN_ULONG ret;          \
        __asm__ ("mulhdu        %0,%1,%2"       \
             : "=r"(ret)                \
             : "r"(a), "r"(b));         \
        ret;                    })
#  endif        /* compiler */
# elif (defined(__x86_64) || defined(__x86_64__)) && \
       (defined(SIXTY_FOUR_BIT_LONG) || defined(SIXTY_FOUR_BIT))
```

```
260 #  if defined(__GNUC__) && __GNUC__>=2
261 #   define BN_UMULT_HIGH(a,b)   ({      \
262         register BN_ULONG ret,discard;  \
263         __asm__ ("mulq  %3"             \
264              : "=a"(discard),"=d"(ret)  \
265              : "a"(a), "g"(b)           \
266              : "cc");                   \
267         ret;                    })
268 #   define BN_UMULT_LOHI(low,high,a,b)  \
269         __asm__ ("mulq  %3"             \
270                 : "=a"(low),"=d"(high)  \
271                 : "a"(a),"g"(b)         \
272                 : "cc");
273 #  endif
274 # elif (defined(_M_AMD64) || defined(_M_X64)) && defined(SIXTY_FOUR_BIT)
275 #  if defined(_MSC_VER) && _MSC_VER>=1400
276     unsigned __int64 __umulh    (unsigned __int64 a,unsigned __int64 b);
277     unsigned __int64 _umul128   (unsigned __int64 a,unsigned __int64 b,
278                                  unsigned __int64 *h);
279 #   pragma intrinsic(__umulh,_umul128)
280 #   define BN_UMULT_HIGH(a,b)           __umulh((a),(b))
281 #   define BN_UMULT_LOHI(low,high,a,b)  ((low)=_umul128((a),(b),&(high)))
282 #  endif
283 # elif defined(__mips) && (defined(SIXTY_FOUR_BIT) || defined(SIXTY_FOUR_BIT_LON
284 #  if defined(__GNUC__) && __GNUC__>=2
285 #   if __GNUC__>=4 && __GNUC_MINOR__>=4 /* "h" constraint is no more since 4.4 *
286 #     define BN_UMULT_HIGH(a,b)         (((__uint128_t)(a)*(b))>>64)
287 #     define BN_UMULT_LOHI(low,high,a,b) ({     \
288         __uint128_t ret=(__uint128_t)(a)*(b);   \
289         (high)=ret>>64; (low)=ret;       })
290 #   else
291 #     define BN_UMULT_HIGH(a,b) ({      \
292         register BN_ULONG ret;          \
293         __asm__ ("dmultu        %1,%2"          \
294              : "=h"(ret)                \
295              : "r"(a), "r"(b) : "l");   \
296         ret;                    })
297 #     define BN_UMULT_LOHI(low,high,a,b)\
298         __asm__ ("dmultu        %2,%3"          \
299              : "=l"(low),"=h"(high)     \
300              : "r"(a), "r"(b));
301 #    endif
302 #  endif
303 # endif         /* cpu */
304 #endif          /* OPENSSL_NO_ASM */

306 /*********************************************************
307  * Using the long long type
308  */
309 #define Lw(t)    (((BN_ULONG)(t))&BN_MASK2)
310 #define Hw(t)    (((BN_ULONG)((t)>>BN_BITS2))&BN_MASK2)

312 #ifdef BN_DEBUG_RAND
313 #define bn_clear_top2max(a) \
314         { \
315         int      ind = (a)->dmax - (a)->top; \
316         BN_ULONG *ftl = &(a)->d[(a)->top-1]; \
317         for (; ind != 0; ind--) \
318                 *(++ftl) = 0x0; \
319         }
320 #else
321 #define bn_clear_top2max(a)
322 #endif

324 #ifdef BN_LLONG
325 #define mul_add(r,a,w,c) { \
```



```
326         BN_ULLONG t; \
327         t=(BN_ULLONG)w * (a) + (r) + (c); \
328         (r)= Lw(t); \
329         (c)= Hw(t); \
330         }

332 #define mul(r,a,w,c) { \
333         BN_ULLONG t; \
334         t=(BN_ULLONG)w * (a) + (c); \
335         (r)= Lw(t); \
336         (c)= Hw(t); \
337         }

339 #define sqr(r0,r1,a) { \
340         BN_ULLONG t; \
341         t=(BN_ULLONG)(a)*(a); \
342         (r0)=Lw(t); \
343         (r1)=Hw(t); \
344         }

346 #elif defined(BN_UMULT_LOHI)
347 #define mul_add(r,a,w,c) {              \
348         BN_ULONG high,low,ret,tmp=(a);  \
349         ret =  (r);                     \
350         BN_UMULT_LOHI(low,high,w,tmp);  \
351         ret += (c);                     \
352         (c) =  (ret<(c))?1:0;           \
353         (c) += high;                    \
354         ret += low;                     \
355         (c) += (ret<low)?1:0;           \
356         (r) =  ret;                     \
357         }

359 #define mul(r,a,w,c)    {               \
360         BN_ULONG high,low,ret,ta=(a);   \
361         BN_UMULT_LOHI(low,high,w,ta);   \
362         ret =  low + (c);               \
363         (c) =  high;                    \
364         (c) += (ret<low)?1:0;           \
365         (r) =  ret;                     \
366         }

368 #define sqr(r0,r1,a)    {               \
369         BN_ULONG tmp=(a);               \
370         BN_UMULT_LOHI(r0,r1,tmp,tmp);   \
371         }

373 #elif defined(BN_UMULT_HIGH)
374 #define mul_add(r,a,w,c) {              \
375         BN_ULONG high,low,ret,tmp=(a);  \
376         ret =  (r);                     \
377         high=  BN_UMULT_HIGH(w,tmp);    \
378         ret += (c);                     \
379         low =  (w) * tmp;               \
380         (c) =  (ret<(c))?1:0;           \
381         (c) += high;                    \
382         ret += low;                     \
383         (c) += (ret<low)?1:0;           \
384         (r) =  ret;                     \
385         }

387 #define mul(r,a,w,c)    {               \
388         BN_ULONG high,low,ret,ta=(a);   \
389         low =  (w) * ta;                \
390         high=  BN_UMULT_HIGH(w,ta);     \
391         ret =  low + (c);               \
```

```
392         (c) =  high;                    \
393         (c) += (ret<low)?1:0;           \
394         (r) =  ret;                     \
395         }

397 #define sqr(r0,r1,a)    {               \
398         BN_ULONG tmp=(a);               \
399         (r0) = tmp * tmp;               \
400         (r1) = BN_UMULT_HIGH(tmp,tmp);  \
401         }

403 #else
404 /*************************************************************
405  * No long long type
406  */

408 #define LBITS(a)        ((a)&BN_MASK2l)
409 #define HBITS(a)        (((a)>>BN_BITS4)&BN_MASK2l)
410 #define L2HBITS(a)      (((a)<<BN_BITS4)&BN_MASK2)

412 #define LLBITS(a)       ((a)&BN_MASKl)
413 #define LHBITS(a)       (((a)>>BN_BITS2)&BN_MASKl)
414 #define LL2HBITS(a)     ((BN_ULLONG)((a)&BN_MASKl)<<BN_BITS2)

416 #define mul64(l,h,bl,bh) \
417         { \
418         BN_ULONG m,m1,lt,ht; \
419  \
420         lt=l; \
421         ht=h; \
422         m =(bh)*(lt); \
423         lt=(bl)*(lt); \
424         m1=(bl)*(ht); \
425         ht =(bh)*(ht); \
426         m=(m+m1)&BN_MASK2; if (m < m1) ht+=L2HBITS((BN_ULONG)1); \
427         ht+=HBITS(m); \
428         m1=L2HBITS(m); \
429         lt=(lt+m1)&BN_MASK2; if (lt < m1) ht++; \
430         (l)=lt; \
431         (h)=ht; \
432         }

434 #define sqr64(lo,ho,in) \
435         { \
436         BN_ULONG l,h,m; \
437  \
438         h=(in); \
439         l=LBITS(h); \
440         h=HBITS(h); \
441         m =(l)*(h); \
442         l*=l; \
443         h*=h; \
444         h+=(m&BN_MASK2h1)>>(BN_BITS4-1); \
445         m =(m&BN_MASK2l)<<(BN_BITS4+1); \
446         l=(l+m)&BN_MASK2; if (l < m) h++; \
447         (lo)=l; \
448         (ho)=h; \
449         }

451 #define mul_add(r,a,bl,bh,c) { \
452         BN_ULONG l,h; \
453  \
454         h= (a); \
455         l=LBITS(h); \
456         h=HBITS(h); \
457         mul64(l,h,(bl),(bh)); \
```



```
458  \
459         /* non-multiply part */ \
460         l=(l+(c))&BN_MASK2; if (l < (c)) h++; \
461         (c)=(r); \
462         l=(l+(c))&BN_MASK2; if (l < (c)) h++; \
463         (c)=h&BN_MASK2; \
464         (r)=l; \
465         }

467 #define mul(r,a,bl,bh,c) { \
468         BN_ULONG l,h; \
469  \
470         h= (a); \
471         l=LBITS(h); \
472         h=HBITS(h); \
473         mul64(l,h,(bl),(bh)); \
474  \
475         /* non-multiply part */ \
476         l+=(c); if ((l&BN_MASK2) < (c)) h++; \
477         (c)=h&BN_MASK2; \
478         (r)=l&BN_MASK2; \
479         }
480 #endif /* !BN_LLONG */

482 #if defined(OPENSSL_DOING_MAKEDEPEND) && defined(OPENSSL_FIPS)
483 #undef bn_div_words
484 #endif

486 void bn_mul_normal(BN_ULONG *r,BN_ULONG *a,int na,BN_ULONG *b,int nb);
487 void bn_mul_comba8(BN_ULONG *r,BN_ULONG *a,BN_ULONG *b);
488 void bn_mul_comba4(BN_ULONG *r,BN_ULONG *a,BN_ULONG *b);
489 void bn_sqr_normal(BN_ULONG *r, const BN_ULONG *a, int n, BN_ULONG *tmp);
490 void bn_sqr_comba8(BN_ULONG *r,const BN_ULONG *a);
491 void bn_sqr_comba4(BN_ULONG *r,const BN_ULONG *a);
492 int bn_cmp_words(const BN_ULONG *a,const BN_ULONG *b,int n);
493 int bn_cmp_part_words(const BN_ULONG *a, const BN_ULONG *b,
494         int cl, int dl);
495 void bn_mul_recursive(BN_ULONG *r,BN_ULONG *a,BN_ULONG *b,int n2,
496         int dna,int dnb,BN_ULONG *t);
497 void bn_mul_part_recursive(BN_ULONG *r,BN_ULONG *a,BN_ULONG *b,
498         int n,int tna,int tnb,BN_ULONG *t);
499 void bn_sqr_recursive(BN_ULONG *r,const BN_ULONG *a, int n2, BN_ULONG *t);
500 void bn_mul_low_normal(BN_ULONG *r,BN_ULONG *a,BN_ULONG *b, int n);
501 void bn_mul_low_recursive(BN_ULONG *r,BN_ULONG *a,BN_ULONG *b,int n2,
502         BN_ULONG *t);
503 void bn_mul_high(BN_ULONG *r,BN_ULONG *a,BN_ULONG *b,BN_ULONG *l,int n2,
504         BN_ULONG *t);
505 BN_ULONG bn_add_part_words(BN_ULONG *r, const BN_ULONG *a, const BN_ULONG *b,
506         int cl, int dl);
507 BN_ULONG bn_sub_part_words(BN_ULONG *r, const BN_ULONG *a, const BN_ULONG *b,
508         int cl, int dl);
509 int bn_mul_mont(BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp, const BN_U

511 #ifdef  __cplusplus
512 }
513 #endif

515 #endif
516 #endif /* ! codereview */
```

```
**********************************************************
   14983 Wed Aug 13 19:51:34 2014
new/usr/src/lib/openssl/include/bn_prime.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Auto generated by bn_prime.pl */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef EIGHT_BIT
#define NUMPRIMES 2048
typedef unsigned short prime_t;
```



```
#else
#define NUMPRIMES 54
typedef unsigned char prime_t;
#endif
static const prime_t primes[NUMPRIMES]=
        {
           2,   3,   5,   7,  11,  13,  17,  19,
          23,  29,  31,  37,  41,  43,  47,  53,
          59,  61,  67,  71,  73,  79,  83,  89,
          97, 101, 103, 107, 109, 113, 127, 131,
         137, 139, 149, 151, 157, 163, 167, 173,
         179, 181, 191, 193, 197, 199, 211, 223,
         227, 229, 233, 239, 241, 251,
#ifndef EIGHT_BIT
         257, 263,
         269, 271, 277, 281, 283, 293, 307, 311,
         313, 317, 331, 337, 347, 349, 353, 359,
         367, 373, 379, 383, 389, 397, 401, 409,
         419, 421, 431, 433, 439, 443, 449, 457,
         461, 463, 467, 479, 487, 491, 499, 503,
         509, 521, 523, 541, 547, 557, 563, 569,
         571, 577, 587, 593, 599, 601, 607, 613,
         617, 619, 631, 641, 643, 647, 653, 659,
         661, 673, 677, 683, 691, 701, 709, 719,
         727, 733, 739, 743, 751, 757, 761, 769,
         773, 787, 797, 809, 811, 821, 823, 827,
         829, 839, 853, 857, 859, 863, 877, 881,
         883, 887, 907, 911, 919, 929, 937, 941,
         947, 953, 967, 971, 977, 983, 991, 997,
        1009,1013,1019,1021,1031,1033,1039,1049,
        1051,1061,1063,1069,1087,1091,1093,1097,
        1103,1109,1117,1123,1129,1151,1153,1163,
        1171,1181,1187,1193,1201,1213,1217,1223,
        1229,1231,1237,1249,1259,1277,1279,1283,
        1289,1291,1297,1301,1303,1307,1319,1321,
        1327,1361,1367,1373,1381,1399,1409,1423,
        1427,1429,1433,1439,1447,1451,1453,1459,
        1471,1481,1483,1487,1489,1493,1499,1511,
        1523,1531,1543,1549,1553,1559,1567,1571,
        1579,1583,1597,1601,1607,1609,1613,1619,
        1621,1627,1637,1657,1663,1667,1669,1693,
        1697,1699,1709,1721,1723,1733,1741,1747,
        1753,1759,1777,1783,1787,1789,1801,1811,
        1823,1831,1847,1861,1867,1871,1873,1877,
        1879,1889,1901,1907,1913,1931,1933,1949,
        1951,1973,1979,1987,1993,1997,1999,2003,
        2011,2017,2027,2029,2039,2053,2063,2069,
        2081,2083,2087,2089,2099,2111,2113,2129,
        2131,2137,2141,2143,2153,2161,2179,2203,
        2207,2213,2221,2237,2239,2243,2251,2267,
        2269,2273,2281,2287,2293,2297,2309,2311,
        2333,2339,2341,2347,2351,2357,2371,2377,
        2381,2383,2389,2393,2399,2411,2417,2423,
        2437,2441,2447,2459,2467,2473,2477,2503,
        2521,2531,2539,2543,2549,2551,2557,2579,
        2591,2593,2609,2617,2621,2633,2647,2657,
        2659,2663,2671,2677,2683,2687,2689,2693,
        2699,2707,2711,2713,2719,2729,2731,2741,
        2749,2753,2767,2777,2789,2791,2797,2801,
        2803,2819,2833,2837,2843,2851,2857,2861,
        2879,2887,2897,2903,2909,2917,2927,2939,
        2953,2957,2963,2969,2971,2999,3001,3011,
        3019,3023,3037,3041,3049,3061,3067,3079,
        3083,3089,3109,3119,3121,3137,3163,3167,
        3169,3181,3187,3191,3203,3209,3217,3221,
        3229,3251,3253,3257,3259,3271,3299,3301,
```

```
128         3307,3313,3319,3323,3329,3331,3343,3347,
129         3359,3361,3371,3373,3389,3391,3407,3413,
130         3433,3449,3457,3461,3463,3467,3469,3491,
131         3499,3511,3517,3527,3529,3533,3539,3541,
132         3547,3557,3559,3571,3581,3583,3593,3607,
133         3613,3617,3623,3631,3637,3643,3659,3671,
134         3673,3677,3691,3697,3701,3709,3719,3727,
135         3733,3739,3761,3767,3769,3779,3793,3797,
136         3803,3821,3823,3833,3847,3851,3853,3863,
137         3877,3881,3889,3907,3911,3917,3919,3923,
138         3929,3931,3943,3947,3967,3989,4001,4003,
139         4007,4013,4019,4021,4027,4049,4051,4057,
140         4073,4079,4091,4093,4099,4111,4127,4129,
141         4133,4139,4153,4157,4159,4177,4201,4211,
142         4217,4219,4229,4231,4241,4243,4253,4259,
143         4261,4271,4273,4283,4289,4297,4327,4337,
144         4339,4349,4357,4363,4373,4391,4397,4409,
145         4421,4423,4441,4447,4451,4457,4463,4481,
146         4483,4493,4507,4513,4517,4519,4523,4547,
147         4549,4561,4567,4583,4591,4597,4603,4621,
148         4637,4639,4643,4649,4651,4657,4663,4673,
149         4679,4691,4703,4721,4723,4729,4733,4751,
150         4759,4783,4787,4789,4793,4799,4801,4813,
151         4817,4831,4861,4871,4877,4889,4903,4909,
152         4919,4931,4933,4937,4943,4951,4957,4967,
153         4969,4973,4987,4993,4999,5003,5009,5011,
154         5021,5023,5039,5051,5059,5077,5081,5087,
155         5099,5101,5107,5113,5119,5147,5153,5167,
156         5171,5179,5189,5197,5209,5227,5231,5233,
157         5237,5261,5273,5279,5281,5297,5303,5309,
158         5323,5333,5347,5351,5381,5387,5393,5399,
159         5407,5413,5417,5419,5431,5437,5441,5443,
160         5449,5471,5477,5479,5483,5501,5503,5507,
161         5519,5521,5527,5531,5557,5563,5569,5573,
162         5581,5591,5623,5639,5641,5647,5651,5653,
163         5657,5659,5669,5683,5689,5693,5701,5711,
164         5717,5737,5741,5743,5749,5779,5783,5791,
165         5801,5807,5813,5821,5827,5839,5843,5849,
166         5851,5857,5861,5867,5869,5879,5881,5897,
167         5903,5923,5927,5939,5953,5981,5987,6007,
168         6011,6029,6037,6043,6047,6053,6067,6073,
169         6079,6089,6091,6101,6113,6121,6131,6133,
170         6143,6151,6163,6173,6197,6199,6203,6211,
171         6217,6221,6229,6247,6257,6263,6269,6271,
172         6277,6287,6299,6301,6311,6317,6323,6329,
173         6337,6343,6353,6359,6361,6367,6373,6379,
174         6389,6397,6421,6427,6449,6451,6469,6473,
175         6481,6491,6521,6529,6547,6551,6553,6563,
176         6569,6571,6577,6581,6599,6607,6619,6637,
177         6653,6659,6661,6673,6679,6689,6691,6701,
178         6703,6709,6719,6733,6737,6761,6763,6779,
179         6781,6791,6793,6803,6823,6827,6829,6833,
180         6841,6857,6863,6869,6871,6883,6899,6907,
181         6911,6917,6947,6949,6959,6961,6967,6971,
182         6977,6983,6991,6997,7001,7013,7019,7027,
183         7039,7043,7057,7069,7079,7103,7109,7121,
184         7127,7129,7151,7159,7177,7187,7193,7207,
185         7211,7213,7219,7229,7237,7243,7247,7253,
186         7283,7297,7307,7309,7321,7331,7333,7349,
187         7351,7369,7393,7411,7417,7433,7451,7457,
188         7459,7477,7481,7487,7489,7499,7507,7517,
189         7523,7529,7537,7541,7547,7549,7559,7561,
190         7573,7577,7583,7589,7591,7603,7607,7621,
191         7639,7643,7649,7669,7673,7681,7687,7691,
192         7699,7703,7717,7723,7727,7741,7753,7757,
193         7759,7789,7793,7817,7823,7829,7841,7853,
```



```
194         7867,7873,7877,7879,7883,7901,7907,7919,
195         7927,7933,7937,7949,7951,7963,7993,8009,
196         8011,8017,8039,8053,8059,8069,8081,8087,
197         8089,8093,8101,8111,8117,8123,8147,8161,
198         8167,8171,8179,8191,8209,8219,8221,8231,
199         8233,8237,8243,8263,8269,8273,8287,8291,
200         8293,8297,8311,8317,8329,8353,8363,8369,
201         8377,8387,8389,8419,8423,8429,8431,8443,
202         8447,8461,8467,8501,8513,8521,8527,8537,
203         8539,8543,8563,8573,8581,8597,8599,8609,
204         8623,8627,8629,8641,8647,8663,8669,8677,
205         8681,8689,8693,8699,8707,8713,8719,8731,
206         8737,8741,8747,8753,8761,8779,8783,8803,
207         8807,8819,8821,8831,8837,8839,8849,8861,
208         8863,8867,8887,8893,8923,8929,8933,8941,
209         8951,8963,8969,8971,8999,9001,9007,9011,
210         9013,9029,9041,9043,9049,9059,9067,9091,
211         9103,9109,9127,9133,9137,9151,9157,9161,
212         9173,9181,9187,9199,9203,9209,9221,9227,
213         9239,9241,9257,9277,9281,9283,9293,9311,
214         9319,9323,9337,9341,9343,9349,9371,9377,
215         9391,9397,9403,9413,9419,9421,9431,9433,
216         9437,9439,9461,9463,9467,9473,9479,9491,
217         9497,9511,9521,9533,9539,9547,9551,9587,
218         9601,9613,9619,9623,9629,9631,9643,9649,
219         9661,9677,9679,9689,9697,9719,9721,9733,
220         9739,9743,9749,9767,9769,9781,9787,9791,
221         9803,9811,9817,9829,9833,9839,9851,9857,
222         9859,9871,9883,9887,9901,9907,9923,9929,
223         9931,9941,9949,9967,9973,10007,10009,10037,
224         10039,10061,10067,10069,10079,10091,10093,10099,
225         10103,10111,10133,10139,10141,10151,10159,10163,
226         10169,10177,10181,10193,10211,10223,10243,10247,
227         10253,10259,10267,10271,10273,10289,10301,10303,
228         10313,10321,10331,10333,10337,10343,10357,10369,
229         10391,10399,10427,10429,10433,10453,10457,10459,
230         10463,10477,10487,10499,10501,10513,10529,10531,
231         10559,10567,10589,10597,10601,10607,10613,10627,
232         10631,10639,10651,10657,10663,10667,10687,10691,
233         10709,10711,10723,10729,10733,10739,10753,10771,
234         10781,10789,10799,10831,10837,10847,10853,10859,
235         10861,10867,10883,10889,10891,10903,10909,10937,
236         10939,10949,10957,10973,10979,10987,10993,11003,
237         11027,11047,11057,11059,11069,11071,11083,11087,
238         11093,11113,11117,11119,11131,11149,11159,11161,
239         11171,11173,11177,11197,11213,11239,11243,11251,
240         11257,11261,11273,11279,11287,11299,11311,11317,
241         11321,11329,11351,11353,11369,11383,11393,11399,
242         11411,11423,11437,11443,11447,11467,11471,11483,
243         11489,11491,11497,11503,11519,11527,11549,11551,
244         11579,11587,11593,11597,11617,11621,11633,11657,
245         11677,11681,11689,11699,11701,11717,11719,11731,
246         11743,11777,11779,11783,11789,11801,11807,11813,
247         11821,11827,11831,11833,11839,11863,11867,11887,
248         11897,11903,11909,11923,11927,11933,11939,11941,
249         11953,11959,11969,11971,11981,11987,12007,12011,
250         12037,12041,12043,12049,12071,12073,12097,12101,
251         12107,12109,12113,12119,12143,12149,12157,12161,
252         12163,12197,12203,12211,12227,12239,12241,12251,
253         12253,12263,12269,12277,12281,12289,12301,12323,
254         12329,12343,12347,12373,12377,12379,12391,12401,
255         12409,12413,12421,12433,12437,12451,12457,12473,
256         12479,12487,12491,12497,12503,12511,12517,12527,
257         12539,12541,12547,12553,12569,12577,12583,12589,
258         12601,12611,12613,12619,12637,12641,12647,12653,
259         12659,12671,12689,12697,12703,12713,12721,12739,
```

```
260         12743,12757,12763,12781,12791,12799,12809,12821,
261         12823,12829,12841,12853,12889,12893,12899,12907,
262         12911,12917,12919,12923,12941,12953,12959,12967,
263         12973,12979,12983,13001,13003,13007,13009,13033,
264         13037,13043,13049,13063,13093,13099,13103,13109,
265         13121,13127,13147,13151,13159,13163,13171,13177,
266         13183,13187,13217,13219,13229,13241,13249,13259,
267         13267,13291,13297,13309,13313,13327,13331,13337,
268         13339,13367,13381,13397,13399,13411,13417,13421,
269         13441,13451,13457,13463,13469,13477,13487,13499,
270         13513,13523,13537,13553,13567,13577,13591,13597,
271         13613,13619,13627,13633,13649,13669,13679,13681,
272         13687,13691,13693,13697,13709,13711,13721,13723,
273         13729,13751,13757,13759,13763,13781,13789,13799,
274         13807,13829,13831,13841,13859,13873,13877,13879,
275         13883,13901,13903,13907,13913,13921,13931,13933,
276         13963,13967,13997,13999,14009,14011,14029,14033,
277         14051,14057,14071,14081,14083,14087,14107,14143,
278         14149,14153,14159,14173,14177,14197,14207,14221,
279         14243,14249,14251,14281,14293,14303,14321,14323,
280         14327,14341,14347,14369,14387,14389,14401,14407,
281         14411,14419,14423,14431,14437,14447,14449,14461,
282         14479,14489,14503,14519,14533,14537,14543,14549,
283         14551,14557,14561,14563,14591,14593,14621,14627,
284         14629,14633,14639,14653,14657,14669,14683,14699,
285         14713,14717,14723,14731,14737,14741,14747,14753,
286         14759,14767,14771,14779,14783,14797,14813,14821,
287         14827,14831,14843,14851,14867,14869,14879,14887,
288         14891,14897,14923,14929,14939,14947,14951,14957,
289         14969,14983,15013,15017,15031,15053,15061,15073,
290         15077,15083,15091,15101,15107,15121,15131,15137,
291         15139,15149,15161,15173,15187,15193,15199,15217,
292         15227,15233,15241,15259,15263,15269,15271,15277,
293         15287,15289,15299,15307,15313,15319,15329,15331,
294         15349,15359,15361,15373,15377,15383,15391,15401,
295         15413,15427,15439,15443,15451,15461,15467,15473,
296         15493,15497,15511,15527,15541,15551,15559,15569,
297         15581,15583,15601,15607,15619,15629,15641,15643,
298         15647,15649,15661,15667,15671,15679,15683,15727,
299         15731,15733,15737,15739,15749,15761,15767,15773,
300         15787,15791,15797,15803,15809,15817,15823,15859,
301         15877,15881,15887,15889,15901,15907,15913,15919,
302         15923,15937,15959,15971,15973,15991,16001,16007,
303         16033,16057,16061,16063,16067,16069,16073,16087,
304         16091,16097,16103,16111,16127,16139,16141,16183,
305         16187,16189,16193,16217,16223,16229,16231,16249,
306         16253,16267,16273,16301,16319,16333,16339,16349,
307         16361,16363,16369,16381,16411,16417,16421,16427,
308         16433,16447,16451,16453,16477,16481,16487,16493,
309         16519,16529,16547,16553,16561,16567,16573,16603,
310         16607,16619,16631,16633,16649,16651,16657,16661,
311         16673,16691,16693,16699,16703,16729,16741,16747,
312         16759,16763,16787,16811,16823,16829,16831,16843,
313         16871,16879,16883,16889,16901,16903,16921,16927,
314         16931,16937,16943,16963,16979,16981,16987,16993,
315         17011,17021,17027,17029,17033,17041,17047,17053,
316         17077,17093,17099,17107,17117,17123,17137,17159,
317         17167,17183,17189,17191,17203,17207,17209,17231,
318         17239,17257,17291,17293,17299,17317,17321,17327,
319         17333,17341,17351,17359,17377,17383,17387,17389,
320         17393,17401,17417,17419,17431,17443,17449,17467,
321         17471,17477,17483,17489,17491,17497,17509,17519,
322         17539,17551,17569,17573,17579,17581,17597,17599,
323         17609,17623,17627,17657,17659,17669,17681,17683,
324         17707,17713,17729,17737,17747,17749,17761,17783,
325         17789,17791,17807,17827,17837,17839,17851,17863,
```



```
326 #endif
327         };
328 #endif /* ! codereview */
```

```
**********************************************************
    8569 Wed Aug 13 19:51:35 2014
new/usr/src/lib/openssl/include/cast_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/cast_lcl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */


#include "e_os.h"
```



```
#ifdef OPENSSL_SYS_WIN32
#include <stdlib.h>
#endif


#undef c2l
#define c2l(c,l)        (l =((unsigned long)(*((c)++)))    , \
                         l|=((unsigned long)(*((c)++)))<< 8L, \
                         l|=((unsigned long)(*((c)++)))<<16L, \
                         l|=((unsigned long)(*((c)++)))<<24L)

/* NOTE - c is not incremented as per c2l */
#undef c2ln
#define c2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))<<24L; \
                        case 7: l2|=((unsigned long)(*(--(c))))<<16L; \
                        case 6: l2|=((unsigned long)(*(--(c))))<< 8L; \
                        case 5: l2|=((unsigned long)(*(--(c))));     \
                        case 4: l1 =((unsigned long)(*(--(c))))<<24L; \
                        case 3: l1|=((unsigned long)(*(--(c))))<<16L; \
                        case 2: l1|=((unsigned long)(*(--(c))))<< 8L; \
                        case 1: l1|=((unsigned long)(*(--(c))));     \
                                } \
                        }

#undef l2c
#define l2c(l,c)        (*((c)++)=(unsigned char)(((l)     )&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24L)&0xff))

/* NOTE - c is not incremented as per l2c */
#undef l2cn
#define l2cn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)>>24L)&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>>16L)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>> 8L)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)     )&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)>>24L)&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>>16L)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>> 8L)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)     )&0xff); \
                                } \
                        }

/* NOTE - c is not incremented as per n2l */
#define n2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))    ; \
                        case 7: l2|=((unsigned long)(*(--(c))))<< 8; \
                        case 6: l2|=((unsigned long)(*(--(c))))<<16; \
                        case 5: l2|=((unsigned long)(*(--(c))))<<24; \
                        case 4: l1 =((unsigned long)(*(--(c))))    ; \
                        case 3: l1|=((unsigned long)(*(--(c))))<< 8; \
                        case 2: l1|=((unsigned long)(*(--(c))))<<16; \
                        case 1: l1|=((unsigned long)(*(--(c))))<<24; \
                                } \
                        }
```

```
/* NOTE - c is not incremented as per l2n */
#define l2nn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)    )&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>> 8)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>>16)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)>>24)&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)    )&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>> 8)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>>16)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)>>24)&0xff); \
                                } \
                        }

#undef n2l
#define n2l(c,l)        (l =((unsigned long)(*((c)++)))<<24L, \
                         l|=((unsigned long)(*((c)++)))<<16L, \
                         l|=((unsigned long)(*((c)++)))<< 8L, \
                         l|=((unsigned long)(*((c)++))))

#undef l2n
#define l2n(l,c)        (*((c)++)=(unsigned char)(((l)>>24L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)     )&0xff))

#if defined(OPENSSL_SYS_WIN32) && defined(_MSC_VER)
#define ROTL(a,n)     (_lrotl(a,n))
#else
#define ROTL(a,n)     ((((a)<<(n))&0xffffffffL)|((a)>>(32-(n))))
#endif

#define C_M    0x3fc
#define C_0    22L
#define C_1    14L
#define C_2     6L
#define C_3     2L /* left shift */

/* The rotate has an extra 16 added to it to help the x86 asm */
#if defined(CAST_PTR)
#define E_CAST(n,key,L,R,OP1,OP2,OP3) \
        { \
        int i; \
        t=(key[n*2] OP1 R)&0xffffffffL; \
        i=key[n*2+1]; \
        t=ROTL(t,i); \
        L^= ((((*(CAST_LONG *)((unsigned char *) \
                        CAST_S_table0+((t>>C_2)&C_M)) OP2 \
                *(CAST_LONG *)((unsigned char *) \
                        CAST_S_table1+((t<<C_3)&C_M)))&0xffffffffL) OP3 \
                *(CAST_LONG *)((unsigned char *) \
                        CAST_S_table2+((t>>C_0)&C_M)))&0xffffffffL) OP1 \
                *(CAST_LONG *)((unsigned char *) \
                        CAST_S_table3+((t>>C_1)&C_M)))&0xffffffffL; \
        }
#elif defined(CAST_PTR2)
#define E_CAST(n,key,L,R,OP1,OP2,OP3) \
        { \
        int i; \
        CAST_LONG u,v,w; \
        w=(key[n*2] OP1 R)&0xffffffffL; \
        i=key[n*2+1]; \
        w=ROTL(w,i); \
        u=w>>C_2; \
        v=w<<C_3; \
```



```
        u&=C_M; \
        v&=C_M; \
        t= *(CAST_LONG *)((unsigned char *)CAST_S_table0+u); \
        u=w>>C_0; \
        t=(t OP2 *(CAST_LONG *)((unsigned char *)CAST_S_table1+v))&0xffffffffL;\
        v=w>>C_1; \
        u&=C_M; \
        v&=C_M; \
        t=(t OP3 *(CAST_LONG *)((unsigned char *)CAST_S_table2+u)&0xffffffffL);\
        t=(t OP1 *(CAST_LONG *)((unsigned char *)CAST_S_table3+v)&0xffffffffL);\
        L^=(t&0xffffffff); \
        }
#else
#define E_CAST(n,key,L,R,OP1,OP2,OP3) \
        { \
        CAST_LONG a,b,c,d; \
        t=(key[n*2] OP1 R)&0xffffffff; \
        t=ROTL(t,(key[n*2+1])); \
        a=CAST_S_table0[(t>> 8)&0xff]; \
        b=CAST_S_table1[(t    )&0xff]; \
        c=CAST_S_table2[(t>>24)&0xff]; \
        d=CAST_S_table3[(t>>16)&0xff]; \
        L^=(((((a OP2 b)&0xffffffffL) OP3 c)&0xffffffffL) OP1 d)&0xffffffffL; \
        }
#endif

extern const CAST_LONG CAST_S_table0[256];
extern const CAST_LONG CAST_S_table1[256];
extern const CAST_LONG CAST_S_table2[256];
extern const CAST_LONG CAST_S_table3[256];
extern const CAST_LONG CAST_S_table4[256];
extern const CAST_LONG CAST_S_table5[256];
extern const CAST_LONG CAST_S_table6[256];
extern const CAST_LONG CAST_S_table7[256];
#endif /* ! codereview */
```

```
**********************************************************
   27190 Wed Aug 13 19:51:35 2014
new/usr/src/lib/openssl/include/cast_s.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/cast_s.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
OPENSSL_GLOBAL const CAST_LONG CAST_S_table0[256]={
        0x30fb40d4,0x9fa0ff0b,0x6beccd2f,0x3f258c7a,
        0x1e213f2f,0x9c004dd3,0x6003e540,0xcf9fc949,
        0xbfd4af27,0x88bbbdb5,0xe2034090,0x98d09675,
```



```
        0x6e63a0e0,0x15c361d2,0xc2e7661d,0x22d4ff8e,
        0x28683b6f,0xc07fd059,0xff2379c8,0x775f50e2,
        0x43c340d3,0xdf2f8656,0x887ca41a,0xa2d2bd2d,
        0xa1c9e0d6,0x346c4819,0x61b76d87,0x22540f2f,
        0x2abe32e1,0xaa54166b,0x22568e3a,0xa2d341d0,
        0x66db40c8,0xa784392f,0x004dff2f,0x2db9d2de,
        0x97943fac,0x4a97c1d8,0x527644b7,0xb5f437a7,
        0xb82cbaef,0xd751d159,0x6ff7f0ed,0x5a097a1f,
        0x827b68d0,0x90ecf52e,0x22b0c054,0xbc8e5935,
        0x4b6d2f7f,0x50bb64a2,0xd2664910,0xbee5812d,
        0xb7332290,0xe93b159f,0xb48ee411,0x4bff345d,
        0xfd45c240,0xad31973f,0xc4f6d02e,0x55fc8165,
        0xd5b1caad,0xa1ac2dae,0xa2d4b76d,0xc19b0c50,
        0x882240f2,0x0c6e4f38,0xa4e4bfd7,0x4f5ba272,
        0x564c1d2f,0xc59c5319,0xb949e354,0xb04669fe,
        0xb1b6ab8a,0xc71358dd,0x6385c545,0x110f935d,
        0x57538ad5,0x6a390493,0xe63d37e0,0x2a54f6b3,
        0x3a787d5f,0x6276a0b5,0x19a6fcdf,0x7a42206a,
        0x29f9d4d5,0xf61b1891,0xbb72275e,0xaa508167,
        0x38901091,0xc6b505eb,0x84c7cb8c,0x2ad75a0f,
        0x874a1427,0xa2d1936b,0x2ad286af,0xaa56d291,
        0xd7894360,0x425c750d,0x93b39e26,0x187184c9,
        0x6c00b32d,0x73e2bb14,0xa0bebc3c,0x54623779,
        0x64459eab,0x3f328b82,0x7718cf82,0x59a2cea6,
        0x04ee002e,0x89fe78e6,0x3fab0950,0x325ff6c2,
        0x81383f05,0x6963c5c8,0x76cb5ad6,0xd49974c9,
        0xca180dcf,0x380782d5,0xc7fa5cf6,0x8ac31511,
        0x35e79e13,0x47da91d0,0xf40f9086,0xa7e2419e,
        0x31366241,0x051ef495,0xaa573b04,0x4a805d8d,
        0x548300d0,0x00322a3c,0xbf64cddf,0xba57a68e,
        0x75c6372b,0x50afd341,0xa7c13275,0x915a0bf5,
        0x6b54bfab,0x2b0b1426,0xab4cc9d7,0x449ccd82,
        0xf7fbf265,0xab85c5f3,0x1b55db94,0xaad4e324,
        0xcfa4bd3f,0x2deaa3e2,0x9e204d02,0xc8bd25ac,
        0xeadf55b3,0xd5bd9e98,0xe31231b2,0x2ad5ad6c,
        0x954329de,0xadbe4528,0xd8710f69,0xaa51c90f,
        0xaa786bf6,0x22513f1e,0xaa51a79b,0x2ad344cc,
        0x7b5a41f0,0xd37cfbad,0x1b069505,0x41ece491,
        0xb4c332e6,0x032268d4,0xc9600acc,0xce387e6d,
        0xbf6bb16c,0x6a70fb78,0x0d03d9c9,0xd4df39de,
        0xe01063da,0x4736f464,0x5ad328d8,0xb347cc96,
        0x75bb0fc3,0x98511bfb,0x4ffbcc35,0xb58bcf6a,
        0xe11f0abc,0xbfc5fe4a,0xa70aec10,0xac39570a,
        0x3f04442f,0x6188b153,0xe0397a2e,0x5727cb79,
        0x9ceb418f,0x1cacd68d,0x2ad37c96,0x0175cb9d,
        0xc69dff09,0xc75b65f0,0xd9db40d8,0xec0e7779,
        0x4744ead4,0xb11c3274,0xdd24cb9e,0x7e1c54bd,
        0xf01144f9,0xd2240eb1,0x9675b3fd,0xa3ac3755,
        0xd47c27af,0x51c85f4d,0x56907596,0xa5bb15e6,
        0x580304f0,0xca042cf1,0x011a37ea,0x8dbfaadb,
        0x35ba3e4a,0x3526ffa0,0xc37b4d09,0xbc306ed9,
        0x98a52666,0x5648f725,0xff5e569d,0x0ced63d0,
        0x7c63b2cf,0x700b45e1,0xd5ea50f1,0x85a92872,
        0xaf1fbda7,0xd4234870,0xa7870bf3,0x2d3b4d79,
        0x42e04198,0x0cd0ede7,0x26470db8,0xf881814c,
        0x474d6ad7,0x7c0c5e5c,0xd1231959,0x381b7298,
        0xf5d2f4db,0xab838653,0x6e2f1e23,0x83719c9e,
        0xbd91e046,0x9a56456e,0xdc39200c,0x20c8c571,
        0x962bda1c,0xe1e696ff,0xb141ab08,0x7cca89b9,
        0x1a69e783,0x02cc4843,0xa2f7c579,0x429ef47d,
        0x427b169c,0x5ac9f049,0xdd8f0f00,0x5c8165bf,
        };
OPENSSL_GLOBAL const CAST_LONG CAST_S_table1[256]={
        0x1f201094,0xef0ba75b,0x69e3cf7e,0x393f4380,
        0xfe61cf7a,0xeec5207a,0x55889c94,0x72fc0651,
        0xada7ef79,0x4e1d7235,0xd55a63ce,0xde0436ba,
```

```
        0x99c430ef,0x5f0c0794,0x18dcdb7d,0xa1d6eff3,
        0xa0b52f7b,0x59e83605,0xee15b094,0xe9ffd909,
        0xdc440086,0xef944459,0xba83ccb3,0xe0c3cdfb,
        0xd1da4181,0x3b092ab1,0xf997f1c1,0xa5e6cf7b,
        0x01420ddb,0xe4e7ef5b,0x25a1ff41,0xe180f806,
        0x1fc41080,0x179bee7a,0xd37ac6a9,0xfe5830a4,
        0x98de8b7f,0x77e83f4e,0x79929269,0x24fa9f7b,
        0xe113c85b,0xacc40083,0xd7503525,0xf7ea615f,
        0x62143154,0x0d554b63,0x5d681121,0xc866c359,
        0x3d63cf73,0xcee234c0,0xd4d87e87,0x5c672b21,
        0x071f6181,0x39f7627f,0x361e3084,0xe4eb573b,
        0x602f64a4,0xd63acd9c,0x1bbc4635,0x9e81032d,
        0x2701f50c,0x99847ab4,0xa0e3df79,0xba6cf38c,
        0x10843094,0x2537a95e,0xf46f6ffe,0xa1ff3b1f,
        0x208cfb6a,0x8f458c74,0xd9e0a227,0x4ec73a34,
        0xfc884f69,0x3e4de8df,0xef0e0088,0x3559648d,
        0x8a45388c,0x1d804366,0x721d9bfd,0xa58684bb,
        0xe8256333,0x844e8212,0x128d8098,0xfed33fb4,
        0xce280ae1,0x27e19ba5,0xd5a6c252,0xe49754bd,
        0xc5d655dd,0xeb667064,0x77840b4d,0xa1b6a801,
        0x84db26a9,0xe0b56714,0x21f043b7,0xe5d05860,
        0x54f03084,0x066ff472,0xa31aa153,0xdadc4755,
        0xb5625dbf,0x68561be6,0x83ca6b94,0x2d6ed23b,
        0xeccf01db,0xa6d3d0ba,0xb6803d5c,0xaf77a709,
        0x33b4a34c,0x397bc8d6,0x5ee22b95,0x5f0e5304,
        0x81ed6f61,0x20e74364,0xb45e1378,0xde18639b,
        0x881ca122,0xb96726d1,0x8049a7e8,0x22b7da7b,
        0x5e552d25,0x5272d237,0x79d2951c,0xc60d894c,
        0x488cb402,0x1ba4fe5b,0xa4b09f6b,0x1ca815cf,
        0xa20c3005,0x8871df63,0xb9de2fcb,0x0cc6c9e9,
        0x0beeff53,0xe3214517,0xb4542835,0x9f63293c,
        0xee41e729,0x6e1d2d7c,0x50045286,0x1e6685f3,
        0xf33401c6,0x30a22c95,0x31a70850,0x60930f13,
        0x73f98417,0xa1269859,0xec645c44,0x52c877a9,
        0xcdff33a6,0xa02b1741,0x7cbad9a2,0x2180036f,
        0x50d99c08,0xcb3f4861,0xc26bd765,0x64a3f6ab,
        0x80342676,0x25a75e7b,0xe4e6d1fc,0x20c710e6,
        0xcdf0b680,0x17844d3b,0x31eef84d,0x7e0824e4,
        0x2ccb49eb,0x846a3bae,0x8ff77888,0xee5d60f6,
        0x7af75673,0x2fdd5cdb,0xa11631c1,0x30f66f43,
        0xb3faec54,0x157fd7fa,0xef8579cc,0xd152de58,
        0xdb2ffd5e,0x8f32ce19,0x306af97a,0x02f03ef8,
        0x99319ad5,0xc242fa0f,0xa7e3ebb0,0xc68e4906,
        0xb8da230c,0x80823028,0xdcdef3c8,0xd35fb171,
        0x088a1bc8,0xbec0c560,0x61a3c9e8,0xbca8f54d,
        0xc72feffa,0x22822e99,0x82c570b4,0xd8d94e89,
        0x8b1c34bc,0x301e16e6,0x273be979,0xb0ffeaa6,
        0x61d9b8c6,0x00b24869,0xb7ffce3f,0x08dc283b,
        0x43daf65a,0xf7e19798,0x7619b72f,0x8f1c9ba4,
        0xdc8637a0,0x16a7d3b1,0x9fc393b7,0xa7136eeb,
        0xc6bcc63e,0x1a513742,0xef6828bc,0x520365d6,
        0x2d6a77ab,0x3527ed4b,0x821fd216,0x095c6e2e,
        0xdb92f2fb,0x5eea29cb,0x145892f5,0x91584f7f,
        0x5483697b,0x2667a8cc,0x85196048,0x8c4bacea,
        0x833860d4,0x0d23e0f9,0x6c387e8a,0x0ae6d249,
        0xb284600c,0xd835731d,0xdcb1c647,0xac4c56ea,
        0x3ebd81b3,0x230eabb0,0x6438bc87,0xf0b5b1fa,
        0x8f5ea2b3,0xfc184642,0x0a036b7a,0x4fb089bd,
        0x649da589,0xa345415e,0x5c038323,0x3e5d3bb9,
        0x43d79572,0x7e6dd07c,0x06dfdf1e,0x6c6cc4ef,
        0x7160a539,0x73bfbe70,0x83877605,0x4523ecf1,
        };
OPENSSL_GLOBAL const CAST_LONG CAST_S_table2[256]={
        0x8defc240,0x25fa5d9f,0xeb903dbf,0xe810c907,
        0x47607fff,0x369fe44b,0x8c1fc644,0xaececa90,
        0xbeb1f9bf,0xeefbcaea,0xe8cf1950,0x51df07ae,
```



```
        0x920e8806,0xf0ad0548,0xe13c8d83,0x927010d5,
        0x11107d9f,0x07647db9,0xb2e3e4d4,0x3d4f285e,
        0xb9afa820,0xfade82e0,0xa067268b,0x8272792e,
        0x553fb2c0,0x489ae22b,0xd4ef9794,0x125e3fbc,
        0x21fffcee,0x825b1bfd,0x9255c5ed,0x1257a240,
        0x4e1a8302,0xbae07fff,0x528246e7,0x8e57140e,
        0x3373f7bf,0x8c9f8188,0xa6fc4ee8,0xc982b5a5,
        0xa8c01db7,0x579fc264,0x67094f31,0xf2bd3f5f,
        0x40fff7c1,0x1fb78dfc,0x8e6bd2c1,0x437be59b,
        0x99b03dbf,0xb5dbc64b,0x638dc0e6,0x55819d99,
        0xa197c81c,0x4a012d6e,0xc5884a28,0xccc36f71,
        0xb843c213,0x6c0743f1,0x8309893c,0x0feddd5f,
        0x2f7fe850,0xd7c07f7e,0x02507fbf,0x5afb9a04,
        0xa747d2d0,0x1651192e,0xaf70bf3e,0x58c31380,
        0x5f98302e,0x727cc3c4,0x0a0fb402,0x0f7fef82,
        0x8c96fdad,0x5d2c2aae,0x8ee99a49,0x50da88b8,
        0x8427f4a0,0x1eac5790,0x796fb449,0x8252dc15,
        0xefbd7d9b,0xa672597d,0xada840d8,0x45f54504,
        0xfa5d7403,0xe83ec305,0x4f91751a,0x925669c2,
        0x23efe941,0xa903f12e,0x60270df2,0x0276e4b6,
        0x94fd6574,0x927985b2,0x8276dbcb,0x02778176,
        0xf8af918d,0x4e48f79e,0x8f616ddf,0xe29d840e,
        0x842f7d83,0x340ce5c8,0x96bbb682,0x93b4b148,
        0xef303cab,0x984faf28,0x779faf9b,0x92dc560d,
        0x224d1e20,0x8437aa88,0x7d29dc96,0x2756d3dc,
        0x8b907cee,0xb51fd240,0xe7c07ce3,0xe566b4a1,
        0xc3e9615e,0x3cf8209d,0x6094d1e3,0xcd9ca341,
        0x5c76460e,0x00ea983b,0xd4d67881,0xfd47572c,
        0xf76cedd9,0xbda8229c,0x127dadaa,0x438a074e,
        0x1f97c090,0x081bdb8a,0x93a07ebe,0xb938ca15,
        0x97b03cff,0x3dc2c0f8,0x8d1ab2ec,0x64380e51,
        0x68cc7bfb,0xd90f2788,0x12490181,0x5de5ffd4,
        0xdd7ef86a,0x76a2e214,0xb9a40368,0x925d958f,
        0x4b39fffa,0xba39aee9,0xa4ffd30b,0xfaf7933b,
        0x6d498623,0x193cbcfa,0x27627545,0x825cf47a,
        0x61bd8ba0,0xd11e42d1,0xcead04f4,0x127ea392,
        0x10428db7,0x8272a972,0x9270c4a8,0x127de50b,
        0x285ba1c8,0x3c62f44f,0x35c0eaa5,0xe805d231,
        0x428929fb,0xb4fcdf82,0x4fb66a53,0x0e7dc15b,
        0x1f081fab,0x108618ae,0xfcfd086d,0xf9ff2889,
        0x694bcc11,0x236a5cae,0x12deca4d,0x2c3f8cc5,
        0xd2d02dfe,0xf8ef5896,0xe4cf52da,0x95155b67,
        0x494a488c,0xb9b6a80c,0x5c8f82bc,0x89d36b45,
        0x3a609437,0xec00c9a9,0x44715253,0x0a874b49,
        0xd773bc40,0x7c34671c,0x02717ef6,0x4feb5536,
        0xa2d02fff,0xd2bf60c4,0xd43f03c0,0x50b4ef6d,
        0x07478cd1,0x006e1888,0xa2e53f55,0xb9e6d4bc,
        0xa2048016,0x97573833,0xd7207d67,0xde0f8f3d,
        0x72f87b33,0xabcc4f33,0x7688c55d,0x7b00a6b0,
        0x947b0001,0x570075d2,0xf9bb88f8,0x8942019e,
        0x4264a5ff,0x856302e0,0x72dbd92b,0xee971b69,
        0x6ea22fde,0x5f08ae2b,0xaf7a616d,0xe5c98767,
        0xcf1febd2,0x61efc8c2,0xf1ac2571,0xcc8239c2,
        0x67214cb8,0xb1e583d1,0xb7dc3e62,0x7f10bdce,
        0xf90a5c38,0x0ff0443d,0x606e6dc6,0x60543a49,
        0x5727c148,0x2be98a1d,0x8ab41738,0x20e1be24,
        0xaf96da0f,0x68458425,0x99833be5,0x600d457d,
        0x282f9350,0x8334b362,0xd91d1120,0x2b6d8da0,
        0x642b1e31,0x9c305a00,0x52bce688,0x1b03588a,
        0xf7baefd5,0x4142ed9c,0xa4315c11,0x83323ec5,
        0xdfef4636,0xa133c501,0xe9d3531c,0xee353783,
        };
OPENSSL_GLOBAL const CAST_LONG CAST_S_table3[256]={
        0x9db30420,0x1fb6e9de,0xa7be7bef,0xd273a298,
        0x4a4f7bdb,0x64ad8c57,0x85510443,0xfa020ed1,
        0x7e287aff,0xe60fb663,0x095f35a1,0x79ebf120,
```

```
260         0xfd059d43,0x6497b7b1,0xf3641f63,0x241e4adf,
261         0x28147f5f,0x4fa2b8cd,0xc9430040,0x0cc32220,
262         0xfdd30b30,0xc0a5374f,0x1d2d00d9,0x24147b15,
263         0xee4d111a,0x0fca5167,0x71ff904c,0x2d195ffe,
264         0x1a05645f,0x0c13fefe,0x081b08ca,0x05170121,
265         0x80530100,0xe83e5efe,0xac9af4f8,0x7fe72701,
266         0xd2b8ee5f,0x06df4261,0xbb9e9b8a,0x7293ea25,
267         0xce84ffdf,0xf5718801,0x3dd64b04,0xa26f263b,
268         0x7ed48400,0x547eebe6,0x446d4ca0,0x6cf3d6f5,
269         0x2649abdf,0xaea0c7f5,0x36338cc1,0x503f7e93,
270         0xd3772061,0x11b638e1,0x72500e03,0xf80eb2bb,
271         0xabe0502e,0xec8d77de,0x57971e81,0xe14f6746,
272         0xc9335400,0x6920318f,0x081dbb99,0xffc304a5,
273         0x4d351805,0x7f3d5ce3,0xa6c866c6,0x5d5bcca9,
274         0xdaec6fea,0x9f926f91,0x9f46222f,0x3991467d,
275         0xa5bf6d8e,0x1143c44f,0x43958302,0xd0214eeb,
276         0x022083b8,0x3fb6180c,0x18f8931e,0x281658e6,
277         0x26486e3e,0x8bd78a70,0x7477e4c1,0xb506e07c,
278         0xf32d0a25,0x79098b02,0xe4eabb81,0x28123b23,
279         0x69dead38,0x1574ca16,0xdf871b62,0x211c40b7,
280         0xa51a9ef9,0x0014377b,0x041e8ac8,0x09114003,
281         0xbd59e4d2,0xe3d156d5,0x4fe876d5,0x2f91a340,
282         0x557be8de,0x00eae4a7,0x0ce5c2ec,0x4db4bba6,
283         0xe756bdff,0xdd3369ac,0xec17b035,0x06572327,
284         0x99afc8b0,0x56c8c391,0x6b65811c,0x5e146119,
285         0x6e85cb75,0xbe07c002,0xc2325577,0x893ff4ec,
286         0x5bbfc92d,0xd0ec3b25,0xb7801ab7,0x8d6d3b24,
287         0x20c763ef,0xc366a5fc,0x9c382880,0x0ace3205,
288         0xaac9548a,0xeca1d7c7,0x041afa32,0x1d16625a,
289         0x6701902c,0x9b757a54,0x31d477f7,0x9126b031,
290         0x36cc6fdb,0xc70b8b46,0xd9e66a48,0x56e55a79,
291         0x026a4ceb,0x52437eff,0x2f8f76b4,0x0df980a5,
292         0x8674cde3,0xedda04eb,0x17a9be04,0x2c18f4df,
293         0xb7747f9d,0xab2af7b4,0xefc34d20,0x2e096b7c,
294         0x1741a254,0xe5b6a035,0x213d42f6,0x2c1c7c26,
295         0x61c2f50f,0x6552daf9,0xd2c231f8,0x25130f69,
296         0xd8167fa2,0x0418f2c8,0x001a96a6,0x0d1526ab,
297         0x63315c21,0x5e0a72ec,0x49bafefd,0x187908d9,
298         0x8d0dbd86,0x311170a7,0x3e9b640c,0xcc3e10d7,
299         0xd5cad3b6,0x0caec388,0xf73001e1,0x6c728aff,
300         0x71eae2a1,0x1f9af36e,0xcfcbd12f,0xc1de8417,
301         0xac07be6b,0xcb44a1d8,0x8b9b0f56,0x013988c3,
302         0xb1c52fca,0xb4be31cd,0xd8782806,0x12a3a4e2,
303         0x6f7de532,0x58fd7eb6,0xd01ee900,0x24adffc2,
304         0xf4990fc5,0x9711aac5,0x001d7b95,0x82e5e7d2,
305         0x109873f6,0x00613096,0xc32d9521,0xada121ff,
306         0x29908415,0x7fbb977f,0xaf9eb3db,0x29c9ed2a,
307         0x5ce2a465,0xa730f32c,0xd0aa3fe8,0x8a5cc091,
308         0xd49e2ce7,0x0ce454a9,0xd60acd86,0x015f1919,
309         0x77079103,0xdea03af6,0x78a8565e,0xdee356df,
310         0x21f05cbe,0x8b75e387,0xb3c50651,0xb8a5c3ef,
311         0xd8eeb6d2,0xe523be77,0xc2154529,0x2f69efdf,
312         0xafe67afb,0xf470c4b2,0xf3e0eb5b,0xd6cc9876,
313         0x39e4460c,0x1fda8538,0x1987832f,0xca007367,
314         0xa99144f8,0x296b299e,0x492fc295,0x9266beab,
315         0xb5676e69,0x9bd3ddda,0xdf7e052f,0xdb25701c,
316         0x1b5e51ee,0xf65324e6,0x6afce36c,0x0316cc04,
317         0x8644213e,0xb7dc59d0,0x7965291f,0xccd6fd43,
318         0x41823979,0x932bcdf6,0xb657c34d,0x4edfd282,
319         0x7ae5290c,0x3cb9536b,0x851e20fe,0x9833557e,
320         0x13ecf0b0,0xd3ffb372,0x3f85c5c1,0x0aef7ed2,
321         };
322 OPENSSL_GLOBAL const CAST_LONG CAST_S_table4[256]={
323         0x7ec90c04,0x2c6e74b9,0x9b0e66df,0xa6337911,
324         0xb86a7fff,0x1dd358f5,0x44dd9d44,0x1731167f,
325         0x08fbf1fa,0xe7f511cc,0xd2051b00,0x735aba00,
```



```
326         0x2ab722d8,0x386381cb,0xacf6243a,0x69befd7a,
327         0xe6a2e77f,0xf0c720cd,0xc4494816,0xccf5c180,
328         0x38851640,0x15b0a848,0xe68b18cb,0x4caadeff,
329         0x5f480a01,0x0412b2aa,0x259814fc,0x41d0efe2,
330         0x4e40b48d,0x248eb6fb,0x8dba1cfe,0x41a99b02,
331         0x1a550a04,0xba8f65cb,0x7251f4e7,0x95a51725,
332         0xc106ecd7,0x97a5980a,0xc539b9aa,0x4d79fe6a,
333         0xf2f3f763,0x68af8040,0xed0c9e56,0x11b4958b,
334         0xe1eb5a88,0x8709e6b0,0xd7e07156,0x4e29fea7,
335         0x6366e52d,0x02d1c000,0xc4ac8e05,0x9377f571,
336         0x0c05372a,0x578535f2,0x2261be02,0xd642a0c9,
337         0xdf13a280,0x74b55bd2,0x682199c0,0xd421e5ec,
338         0x53fb3ce8,0xc8adedb3,0x28a87fc9,0x3d959981,
339         0x5c1ff900,0xfe38d399,0x0c4eff0b,0x062407ea,
340         0xaa2f4fb1,0x4fb96976,0x90c79505,0xb0a8a774,
341         0xef55a1ff,0xe59ca2c2,0xa6b62d27,0xe66a4263,
342         0xdf65001f,0x0ec50966,0xdfdd55bc,0x29de0655,
343         0x911e739a,0x17af8975,0x32c7911c,0x89f89468,
344         0x0d01e980,0x524755f4,0x03b63cc9,0x0cc844b2,
345         0xbcf3f0aa,0x87ac36e9,0xe53a7426,0x01b3d82b,
346         0x1a9e7449,0x64ee2d7e,0xcddbb1da,0x01c94910,
347         0xb868bf80,0x0d26f3fd,0x9342ede7,0x04a5c284,
348         0x636737b6,0x50f5b616,0xf24766e3,0x8eca36c1,
349         0x136e05db,0xfef18391,0xfb887a37,0xd6e7f7d4,
350         0xc7fb7dc9,0x3063fcdf,0xb6f589de,0xec2941da,
351         0x26e46695,0xb7566419,0xf654efc5,0xd08d58b7,
352         0x48925401,0xc1bacb7f,0xe5ff550f,0xb6083049,
353         0x5bb5d0e8,0x87d72e5a,0xab6a6ee1,0x223a66ce,
354         0xc62bf3cd,0x9e0885f9,0x68cb3e47,0x086c010f,
355         0xa21de820,0xd18b69de,0xf3f65777,0xfa02c3f6,
356         0x407edac3,0xcbb3d550,0x1793084d,0xb0d70eba,
357         0x0ab378d5,0xd951fb0c,0xded7da56,0x4124bbe4,
358         0x94ca0b56,0x0f5755d1,0xe0e1e56e,0x6184b5be,
359         0x580a249f,0x94f74bc0,0xe327888e,0x9f7b5561,
360         0xc3dc0280,0x05687715,0x646c6bd7,0x44904db3,
361         0x66b4f0a3,0xc0f1648a,0x697ed5af,0x49e92ff6,
362         0x309e374f,0x2cb6356a,0x85808573,0x4991f840,
363         0x76f0ae02,0x083be84d,0x28421c9a,0x44489406,
364         0x736e4cb8,0xc1092910,0x8bc95fc6,0x7d869cf4,
365         0x134f616f,0x2e77118d,0xb31b2be1,0xaa90b472,
366         0x3ca5d717,0x7d161bba,0x9cad9010,0xaf462ba2,
367         0x9fe459d2,0x45d34559,0xd9f2da13,0xdbc65487,
368         0xf3e4f94e,0x176d486f,0x097c13ea,0x631da5c7,
369         0x445f7382,0x175683f4,0xcdc66a97,0x70be0288,
370         0xb3cdcf72,0x6e5dd2f3,0x20936079,0x459b80a5,
371         0xbe60e2db,0xa9c23101,0xeba5315c,0x224e42f2,
372         0x1c5c1572,0xf6721b2c,0x1ad2fff3,0x8c25404e,
373         0x324ed72f,0x4067b7fd,0x0523138e,0x5ca3bc78,
374         0xdc0fd66e,0x75922283,0x784d6b17,0x58ebb16e,
375         0x44094f85,0x3f481d87,0xfcfeae7b,0x77b5ff76,
376         0x8c2302bf,0xaaf47556,0x5f46b02a,0x2b092801,
377         0x3d38f5f7,0x0ca81f36,0x52af4a8a,0x66d5e7c0,
378         0xdf3b0874,0x95055110,0x1b5ad7a8,0xf61ed5ad,
379         0x6cf6e479,0x20758184,0xd0cefa65,0x88f7be58,
380         0x4a046826,0x0ff6f8f3,0xa09c7f70,0x5346aba0,
381         0x5ce96c28,0xe176eda3,0x6bac307f,0x376829d2,
382         0x85360fa9,0x17e3fe2a,0x24b79767,0xf5a96b20,
383         0xd6cd2595,0x68ff1ebf,0x7555442c,0xf19f06be,
384         0xf9e0659a,0xeeb9491d,0x34010718,0xbb30cab8,
385         0xe822fe15,0x88570983,0x750e6249,0xda627e55,
386         0x5e76ffa8,0xb1534546,0x6d47de08,0xefe9e7d4,
387         };
388 OPENSSL_GLOBAL const CAST_LONG CAST_S_table5[256]={
389         0xf6fa8f9d,0x2cac6ce1,0x4ca34867,0xe2337f7c,
390         0x95db08e7,0x016843b4,0xeced5cbc,0x325553ac,
391         0xbf9f0960,0xdfa1e2ed,0x83f0579d,0x63ed86b9,
```

```
392         0x1ab6a6b8,0xde5ebe39,0xf38ff732,0x8989b138,
393         0x33f14961,0xc01937bd,0xf506c6da,0xe4625e7e,
394         0xa308ea99,0x4e23e33c,0x79cbd7cc,0x48a14367,
395         0xa3149619,0xfec94bd5,0xa114174a,0xeaa01866,
396         0xa084db2d,0x09a8486f,0xa888614a,0x2900af98,
397         0x01665991,0xe1992863,0xc8f30c60,0x2e78ef3c,
398         0xd0d51932,0xcf0fec14,0xf7ca07d2,0xd0a82072,
399         0xfd41197e,0x9305a6b0,0xe86be3da,0x74bed3cd,
400         0x372da53c,0x4c7f4448,0xdab5d440,0x6dba0ec3,
401         0x083919a7,0x9fbaeed9,0x49dbcfb0,0x4e670c53,
402         0x5c3d9c01,0x64bdb941,0x2c0e636a,0xba7dd9cd,
403         0xea6f7388,0xe70bc762,0x35f29adb,0x5c4cdd8d,
404         0xf0d48d8c,0xb88153e2,0x08a19866,0x1ae2eac8,
405         0x284caf89,0xaa928223,0x9334be53,0x3b3a21bf,
406         0x16434be3,0x9aea3906,0xefe8c36e,0xf890cdd9,
407         0x80226dae,0xc340a4a3,0xdf7e9c09,0xa694a807,
408         0x5b7c5ecc,0x221db3a6,0x9a69a02f,0x68818a54,
409         0xceb2296f,0x53c0843a,0xfe893655,0x25bfe68a,
410         0xb4628abc,0xcf222ebf,0x25ac6f48,0xa9a99387,
411         0x53bddb65,0xe76ffbe7,0xe967fd78,0x0ba93563,
412         0x8e342bc1,0xe8a11be9,0x4980740d,0xc8087dfc,
413         0x8de4bf99,0xa11101a0,0x7fd37975,0xda5a26c0,
414         0xe81f994f,0x9528cd89,0xfd339fed,0xb87834bf,
415         0x5f04456d,0x22258698,0xc9c4c83b,0x2dc156be,
416         0x4f628daa,0x57f55ec5,0xe2220abe,0xd2916ebf,
417         0x4ec75b95,0x24f2c3c0,0x42d15d99,0xcd0d7fa0,
418         0x7b6e27ff,0xa8dc8af0,0x7345c106,0xf41e232f,
419         0x35162386,0xe6ea8926,0x3333b094,0x157ec6f2,
420         0x372b74af,0x692573e4,0xe9a9d848,0xf3160289,
421         0x3a62ef1d,0xa787e238,0xf3a5f676,0x74364853,
422         0x20951063,0x4576698d,0xb6fad407,0x592af950,
423         0x36f73523,0x4cfb6e87,0x7da4cec0,0x6c152daa,
424         0xcb0396a8,0xc50dfe5d,0xfcd707ab,0x0921c42f,
425         0x89dff0bb,0x5fe2be78,0x448f4f33,0x754613c9,
426         0x2b05d08d,0x48b9d585,0xdc049441,0xc8098f9b,
427         0x7dede786,0xc39a3373,0x42410005,0x6a091751,
428         0x0ef3c8a6,0x890072d6,0x28207682,0xa9a9f7be,
429         0xbf32679d,0xd45b5b75,0xb353fd00,0xcbb0e358,
430         0x830f220a,0x1f8fb214,0xd372cf08,0xcc3c4a13,
431         0x8cf63166,0x061c87be,0x88c98f88,0x6062e397,
432         0x47cf8e7a,0xb6c85283,0x3cc2acfb,0x3fc06976,
433         0x4e8f0252,0x64d8314d,0xda3870e3,0x1e665459,
434         0xc10908f0,0x513021a5,0x6c5b68b7,0x822f8aa0,
435         0x3007cd3e,0x74719eef,0xdc872681,0x073340d4,
436         0x7e432fd9,0x0c5ec241,0x8809286c,0xf592d891,
437         0x08a930f6,0x957ef305,0xb7fbffbd,0xc266e96f,
438         0x6fe4ac98,0xb173ecc0,0xbc60b42a,0x953498da,
439         0xfba1ae12,0x2d4bd736,0x0f25faab,0xa4f3fceb,
440         0xe2969123,0x257f0c3d,0x9348af49,0x361400bc,
441         0xe8816f4a,0x3814f200,0xa3f94043,0x9c7a54c2,
442         0xbc704f57,0xda41e7f9,0xc25ad33a,0x54f4a084,
443         0xb17f5505,0x59357cbe,0xedbd15c8,0x7f97c5ab,
444         0xba5ac7b5,0xb6f6deaf,0x3a479c3a,0x5302da25,
445         0x653d7e6a,0x54268d49,0x51a477ea,0x5017d55b,
446         0xd7d25d88,0x44136c76,0x0404a8c8,0xb8e5a121,
447         0xb81a928a,0x60ed5869,0x97c55b96,0xeaec991b,
448         0x29935913,0x01fdb7f1,0x088e8dfa,0x9ab6f6f5,
449         0x3b4cbf9f,0x4a5de3ab,0xe6051d35,0xa0e1d855,
450         0xd36b4cf1,0xf544edeb,0xb0e93524,0xbebb8fbd,
451         0xa2d762cf,0x49c92f54,0x38b5f331,0x7128a454,
452         0x48392905,0xa65b1db8,0x851c97bd,0xd675cf2f,
453         };
454 OPENSSL_GLOBAL const CAST_LONG CAST_S_table6[256]={
455         0x85e04019,0x332bf567,0x662dbfff,0xcfc65693,
456         0x2a8d7f6f,0xab9bc912,0xde6008a1,0x2028da1f,
457         0x0227bce7,0x4d642916,0x18fac300,0x50f18b82,
```



```
458         0x2cb2cb11,0xb232e75c,0x4b3695f2,0xb28707de,
459         0xa05fbcf6,0xcd4181e9,0xe150210c,0xe24ef1bd,
460         0xb168c381,0xfde4e789,0x5c79b0d8,0x1e8bfd43,
461         0x4d495001,0x38be4341,0x913cee1d,0x92a79c3f,
462         0x089766be,0xbaeeadf4,0x1286becf,0xb6eacb19,
463         0x2660c200,0x7565bde4,0x64241f7a,0x8248dca9,
464         0xc3b3ad66,0x28136086,0x0bd8dfa8,0x356d1cf2,
465         0x107789be,0xb3b2e9ce,0x0502aa8f,0x0bc0351e,
466         0x166bf52a,0xeb12ff82,0xe3486911,0xd34d7516,
467         0x4e7b3aff,0x5f43671b,0x9cf6e037,0x4981ac83,
468         0x334266ce,0x8c9341b7,0xd0d854c0,0xcb3a6c88,
469         0x47bc2829,0x4725ba37,0xa66ad22b,0x7ad61f1e,
470         0x0c5cbafa,0x4437f107,0xb6e79962,0x42d2d816,
471         0x0a961288,0xe1a5c06e,0x13749e67,0x72fc081a,
472         0xb1d139f7,0xf9583745,0xcf19df58,0xbec3f756,
473         0xc06eba30,0x07211b24,0x45c28829,0xc95e317f,
474         0xbc8ec511,0x38bc46e9,0xc6e6fa14,0xbae8584a,
475         0xad4ebc46,0x468f508b,0x7829435f,0xf124183b,
476         0x821dba9f,0xaff60ff4,0xea2c4e6d,0x16e39264,
477         0x92544a8b,0x009b4fc3,0xaba68ced,0x9ac96f78,
478         0x06a5b79a,0xb2856e6e,0x1aec3ca9,0xbe838688,
479         0x0e0804e9,0x55f1be56,0xe7e5363b,0xb3a1f25d,
480         0xf7debb85,0x61fe033c,0x16746233,0x3c034c28,
481         0xda6d0c74,0x79aac56c,0x3ce4e1ad,0x51f0c802,
482         0x98f8f35a,0x1626a49f,0xeed82b29,0x1d382fe3,
483         0x0c4fb99a,0xbb325778,0x3ec6d97b,0x6e77a6a9,
484         0xcb658b5c,0xd45230c7,0x2bd1408b,0x60c03eb7,
485         0xb9068d78,0xa33754f4,0xf430c87d,0xc8a71302,
486         0xb96d8c32,0xebd4e7be,0xbe8b9d2d,0x7979fb06,
487         0xe7225308,0x8b75cf77,0x11ef8da4,0xe083c858,
488         0x8d6b786f,0x5a6317a6,0xfa5cf7a0,0x5dda0033,
489         0xf28ebfb0,0xf5b9c310,0xa0eac280,0x08b9767a,
490         0xa3d9d2b0,0x79d34217,0x021a718d,0x9ac6336a,
491         0x2711fd60,0x438050e3,0x069908a8,0x3d7fedc4,
492         0x826d2bef,0x4eeb8476,0x488dcf25,0x36c9d566,
493         0x28e74e41,0xc2610aca,0x3d49a9cf,0xbae3b9df,
494         0xb65f8de6,0x92aeaf64,0x3ac7d5e6,0x9ea80509,
495         0xf22b017d,0xa4173f70,0xdd1e16c3,0x15e0d7f9,
496         0x50b1b887,0x2b9f4fd5,0x625aba82,0x6a017962,
497         0x2ec01b9c,0x15488aa9,0xd716e740,0x40055a2c,
498         0x93d29a22,0xe32dbf9a,0x058745b9,0x3453dc1e,
499         0xd699296e,0x496cff6f,0x1c9f4986,0xdfe2ed07,
500         0xb87242d1,0x19de7eae,0x053e561a,0x15ad6f8c,
501         0x66626c1c,0x7154c24c,0xea082b2a,0x93eb2939,
502         0x17dcb0f0,0x58d4f2ae,0x9ea294fb,0x52cf564c,
503         0x9883fe66,0x2ec40581,0x763953c3,0x01d6692e,
504         0xd3a0c108,0xa1e7160e,0xe4f2dfa6,0x693ed285,
505         0x74904698,0x4c2b0edd,0x4f757656,0x5d393378,
506         0xa132234f,0x3d321c5d,0xc3f5e194,0x4b269301,
507         0xc79f022f,0x3c997e7e,0x5e4f9504,0x3ffafbbd,
508         0x76f7ad0e,0x296693f4,0x3d1fce6f,0xc61e45be,
509         0xd3b5ab34,0xf72bf9b7,0x1b0434c0,0x4e72b567,
510         0x5592a33d,0xb5229301,0xcfd2a87f,0x60aeb767,
511         0x1814386b,0x30bcc33d,0x38a0c07d,0xfd1606f2,
512         0xc363519b,0x589dd390,0x5479f8e6,0x1cb8d647,
513         0x97fd61a9,0xea7759f4,0x2d57539d,0x569a58cf,
514         0xe84e63ad,0x462e1b78,0x6580f87e,0xf3817914,
515         0x91da55f4,0x40a230f3,0xd1988f35,0xb6e318d2,
516         0x3ffa50bc,0x3d40f021,0xc3c0bdae,0x4958c24c,
517         0x518f36b2,0x84b1d370,0x0fedce83,0x878ddada,
518         0xf2a279c7,0x94e01be8,0x90716f4b,0x954b8aa3,
519         };
520 OPENSSL_GLOBAL const CAST_LONG CAST_S_table7[256]={
521         0xe216300d,0xbbddfffc,0xa7ebdabd,0x35648095,
522         0x7789f8b7,0xe6c1121b,0x0e241600,0x052ce8b5,
523         0x11a9cfb0,0xe5952f11,0xece7990a,0x9386d174,
```

```
524         0x2a42931c,0x76e38111,0xb12def3a,0x37ddddfc,
525         0xde9adeb1,0x0a0cc32c,0xbe197029,0x84a00940,
526         0xbb243a0f,0xb4d137cf,0xb44e79f0,0x049eedfd,
527         0x0b15a15d,0x480d3168,0x8bbbde5a,0x669ded42,
528         0xc7ece831,0x3f8f95e7,0x72df191b,0x7580330d,
529         0x94074251,0x5c7dcdfa,0xabbe6d63,0xaa402164,
530         0xb301d40a,0x02e7d1ca,0x53571dae,0x7a3182a2,
531         0x12a8ddec,0xfdaa335d,0x176f43e8,0x71fb46d4,
532         0x38129022,0xce949ad4,0xb84769ad,0x965bd862,
533         0x82f3d055,0x66fb9767,0x15b80b4e,0x1d5b47a0,
534         0x4cfde06f,0xc28ec4b8,0x57e8726e,0x647a78fc,
535         0x99865d44,0x608bd593,0x6c200e03,0x39dc5ff6,
536         0x5d0b00a3,0xae63aff2,0x7e8bd632,0x70108c0c,
537         0xbbd35049,0x2998df04,0x980cf42a,0x9b6df491,
538         0x9e7edd53,0x06918548,0x58cb7e07,0x3b74ef2e,
539         0x522fffb1,0xd24708cc,0x1c7e27cd,0xa4eb215b,
540         0x3cf1d2e2,0x19b47a38,0x424f7618,0x35856039,
541         0x9d17dee7,0x27eb35e6,0xc9aff67b,0x36baf5b8,
542         0x09c467cd,0xc18910b1,0xe11dbf7b,0x06cd1af8,
543         0x7170c608,0x2d5e3354,0xd4de495a,0x64c6d006,
544         0xbcc0c62c,0x3dd00db3,0x708f8f34,0x77d51b42,
545         0x264f620f,0x24b8d2bf,0x15c1b79e,0x46a52564,
546         0xf8d7e54e,0x3e378160,0x7895cda5,0x859c15a5,
547         0xe6459788,0xc37bc75f,0xdb07ba0c,0x0676a3ab,
548         0x7f229b1e,0x31842e7b,0x24259fd7,0xf8bef472,
549         0x835ffcb8,0x6df4c1f2,0x96f5b195,0xfd0af0fc,
550         0xb0fe134c,0xe2506d3d,0x4f9b12ea,0xf215f225,
551         0xa223736f,0x9fb4c428,0x25d04979,0x34c713f8,
552         0xc4618187,0xea7a6e98,0x7cd16efc,0x1436876c,
553         0xf1544107,0xbedeee14,0x56e9af27,0xa04aa441,
554         0x3cf7c899,0x92ecbae6,0xdd67016d,0x151682eb,
555         0xa842eedf,0xfdba60b4,0xf1907b75,0x20e3030f,
556         0x24d8c29e,0xe139673b,0xefa63fb8,0x71873054,
557         0xb6f2cf3b,0x9f326442,0xcb15a4cc,0xb01a4504,
558         0xf1e47d8d,0x844a1be5,0xbae7dfdc,0x42cbda70,
559         0xcd7dae0a,0x57e85b7a,0xd53f5af6,0x20cf4d8c,
560         0xcea4d428,0x79d130a4,0x3486ebfb,0x33d3cddc,
561         0x77853b53,0x37effcb5,0xc5068778,0xe580b3e6,
562         0x4e68b8f4,0xc5c8b37e,0x0d809ea2,0x398feb7c,
563         0x132a4f94,0x43b7950e,0x2fee7d1c,0x223613bd,
564         0xdd06caa2,0x37df932b,0xc4248289,0xacf3ebc3,
565         0x5715f6b7,0xef3478dd,0xf267616f,0xc148cbe4,
566         0x9052815e,0x5e410fab,0xb48a2465,0x2eda7fa4,
567         0xe87b40e4,0xe98ea084,0x5889e9e1,0xefd390fc,
568         0xdd07d35b,0xdb485694,0x38d7e5b2,0x57720101,
569         0x730edebc,0x5b643113,0x94917e4f,0x503c2fba,
570         0x646f1282,0x7523d24a,0xe0779695,0xf9c17a8f,
571         0x7a5b2121,0xd187b896,0x29263a4d,0xba510cdf,
572         0x81f47c9f,0xad1163ed,0xea7b5965,0x1a00726e,
573         0x11403092,0x00da6d77,0x4a0cdd61,0xad1f4603,
574         0x605bdfb0,0x9eedc364,0x22ebe6a8,0xcee7d28a,
575         0xa0e736a0,0x5564a6b9,0x10853209,0xc7eb8f37,
576         0x2de705ca,0x8951570f,0xdf09822b,0xbd691a6c,
577         0xaa12e4f2,0x87451c0f,0xe0f6a27a,0x3ada4819,
578         0x4cf1764f,0x0d771c2b,0x67cdb156,0x350d8384,
579         0x5938fa0f,0x42399ef3,0x36997b07,0x0e84093d,
580         0x4aa93e61,0x8360d87b,0x1fa98b0c,0x1149382c,
581         0xe97625a5,0x0614d1b7,0x0e25244b,0x0c768347,
582         0x589e8d82,0x0d2059d1,0xa466bb1e,0xf8da0a82,
583         0x04f19130,0xba6e4ec0,0x99265164,0x1ee7230d,
584         0x50b2ad80,0xeaee6801,0x8db2a283,0xea8bf59e,
585         };
586 #endif /* ! codereview */
```

```
**********************************************************
     526 Wed Aug 13 19:51:35 2014
new/usr/src/lib/openssl/include/charmap.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* Auto generated with chartype.pl script.
 * Mask of various character properties
 */

static const unsigned char char_type[] = {
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
120, 0, 1,40, 0, 0, 0,16,16,16, 0,25,25,16,16,16,
16,16,16,16,16,16,16,16,16,16, 9, 9,16, 9,16,
 0,16,16,16,16,16,16,16,16,16,16,16,16,16,16,16,
16,16,16,16,16,16,16,16,16,16, 0, 1, 0, 0, 0,
 0,16,16,16,16,16,16,16,16,16,16,16,16,16,16,16,
16,16,16,16,16,16,16,16,16,16, 0, 0, 0, 0, 2
};
#endif /* ! codereview */
```

```
**********************************************************
    4021 Wed Aug 13 19:51:35 2014
new/usr/src/lib/openssl/include/cmll_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/camellia/camellia_locl.h -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright 2006 NTT (Nippon Telegraph and Telephone Corporation) .
 * ALL RIGHTS RESERVED.
 *
 * Intellectual Property information for Camellia:
 *     http://info.isl.ntt.co.jp/crypt/eng/info/chiteki.html
 *
 * News Release for Announcement of Camellia open source:
 *     http://www.ntt.co.jp/news/news06e/0604/060413a.html
 *
 * The Camellia Code included herein is developed by
 * NTT (Nippon Telegraph and Telephone Corporation), and is contributed
 * to the OpenSSL project.
 *
 * The Camellia Code is licensed pursuant to the OpenSSL open source
 * license provided below.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
```



```
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#ifndef HEADER_CAMELLIA_LOCL_H
#define HEADER_CAMELLIA_LOCL_H

typedef unsigned int  u32;
typedef unsigned char u8;

int Camellia_Ekeygen(int keyBitLength, const u8 *rawKey,
                     KEY_TABLE_TYPE keyTable);
void Camellia_EncryptBlock_Rounds(int grandRounds, const u8 plaintext[],
                const KEY_TABLE_TYPE keyTable, u8 ciphertext[]);
void Camellia_DecryptBlock_Rounds(int grandRounds, const u8 ciphertext[],
                const KEY_TABLE_TYPE keyTable, u8 plaintext[]);
void Camellia_EncryptBlock(int keyBitLength, const u8 plaintext[],
                const KEY_TABLE_TYPE keyTable, u8 ciphertext[]);
void Camellia_DecryptBlock(int keyBitLength, const u8 ciphertext[],
                const KEY_TABLE_TYPE keyTable, u8 plaintext[]);
int private_Camellia_set_key(const unsigned char *userKey, const int bits,
                             CAMELLIA_KEY *key);
#endif /* #ifndef HEADER_CAMELLIA_LOCL_H */
#endif /* ! codereview */
```

```
**********************************************************
   13357 Wed Aug 13 19:51:35 2014
new/usr/src/lib/openssl/include/cms_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_lcl.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#ifndef HEADER_CMS_LCL_H
#define HEADER_CMS_LCL_H

#ifdef __cplusplus
extern "C" {
#endif

#include <openssl/x509.h>
```



```
/* Cryptographic message syntax (CMS) structures: taken
 * from RFC3852
 */

/* Forward references */

typedef struct CMS_IssuerAndSerialNumber_st CMS_IssuerAndSerialNumber;
typedef struct CMS_EncapsulatedContentInfo_st CMS_EncapsulatedContentInfo;
typedef struct CMS_SignerIdentifier_st CMS_SignerIdentifier;
typedef struct CMS_SignedData_st CMS_SignedData;
typedef struct CMS_OtherRevocationInfoFormat_st CMS_OtherRevocationInfoFormat;
typedef struct CMS_OriginatorInfo_st CMS_OriginatorInfo;
typedef struct CMS_EncryptedContentInfo_st CMS_EncryptedContentInfo;
typedef struct CMS_EnvelopedData_st CMS_EnvelopedData;
typedef struct CMS_DigestedData_st CMS_DigestedData;
typedef struct CMS_EncryptedData_st CMS_EncryptedData;
typedef struct CMS_AuthenticatedData_st CMS_AuthenticatedData;
typedef struct CMS_CompressedData_st CMS_CompressedData;
typedef struct CMS_OtherCertificateFormat_st CMS_OtherCertificateFormat;
typedef struct CMS_KeyTransRecipientInfo_st CMS_KeyTransRecipientInfo;
typedef struct CMS_OriginatorPublicKey_st CMS_OriginatorPublicKey;
typedef struct CMS_OriginatorIdentifierOrKey_st CMS_OriginatorIdentifierOrKey;
typedef struct CMS_KeyAgreeRecipientInfo_st CMS_KeyAgreeRecipientInfo;
typedef struct CMS_OtherKeyAttribute_st CMS_OtherKeyAttribute;
typedef struct CMS_RecipientKeyIdentifier_st CMS_RecipientKeyIdentifier;
typedef struct CMS_KeyAgreeRecipientIdentifier_st CMS_KeyAgreeRecipientIdentifie
typedef struct CMS_RecipientEncryptedKey_st CMS_RecipientEncryptedKey;
typedef struct CMS_KEKIdentifier_st CMS_KEKIdentifier;
typedef struct CMS_KEKRecipientInfo_st CMS_KEKRecipientInfo;
typedef struct CMS_PasswordRecipientInfo_st CMS_PasswordRecipientInfo;
typedef struct CMS_OtherRecipientInfo_st CMS_OtherRecipientInfo;
typedef struct CMS_ReceiptsFrom_st CMS_ReceiptsFrom;

struct CMS_ContentInfo_st
        {
        ASN1_OBJECT *contentType;
        union   {
                ASN1_OCTET_STRING *data;
                CMS_SignedData *signedData;
                CMS_EnvelopedData *envelopedData;
                CMS_DigestedData *digestedData;
                CMS_EncryptedData *encryptedData;
                CMS_AuthenticatedData *authenticatedData;
                CMS_CompressedData *compressedData;
                ASN1_TYPE *other;
                /* Other types ... */
                void *otherData;
                } d;
        };

struct CMS_SignedData_st
        {
        long version;
        STACK_OF(X509_ALGOR) *digestAlgorithms;
        CMS_EncapsulatedContentInfo *encapContentInfo;
        STACK_OF(CMS_CertificateChoices) *certificates;
        STACK_OF(CMS_RevocationInfoChoice) *crls;
        STACK_OF(CMS_SignerInfo) *signerInfos;
        };

struct CMS_EncapsulatedContentInfo_st
        {
        ASN1_OBJECT *eContentType;
        ASN1_OCTET_STRING *eContent;
        /* Set to 1 if incomplete structure only part set up */
```

```
128         int partial;
129         };

131 struct CMS_SignerInfo_st
132         {
133         long version;
134         CMS_SignerIdentifier *sid;
135         X509_ALGOR *digestAlgorithm;
136         STACK_OF(X509_ATTRIBUTE) *signedAttrs;
137         X509_ALGOR *signatureAlgorithm;
138         ASN1_OCTET_STRING *signature;
139         STACK_OF(X509_ATTRIBUTE) *unsignedAttrs;
140         /* Signing certificate and key */
141         X509 *signer;
142         EVP_PKEY *pkey;
143         };

145 struct CMS_SignerIdentifier_st
146         {
147         int type;
148         union   {
149                 CMS_IssuerAndSerialNumber *issuerAndSerialNumber;
150                 ASN1_OCTET_STRING *subjectKeyIdentifier;
151                 } d;
152         };

154 struct CMS_EnvelopedData_st
155         {
156         long version;
157         CMS_OriginatorInfo *originatorInfo;
158         STACK_OF(CMS_RecipientInfo) *recipientInfos;
159         CMS_EncryptedContentInfo *encryptedContentInfo;
160         STACK_OF(X509_ATTRIBUTE) *unprotectedAttrs;
161         };

163 struct CMS_OriginatorInfo_st
164         {
165         STACK_OF(CMS_CertificateChoices) *certificates;
166         STACK_OF(CMS_RevocationInfoChoice) *crls;
167         };

169 struct CMS_EncryptedContentInfo_st
170         {
171         ASN1_OBJECT *contentType;
172         X509_ALGOR *contentEncryptionAlgorithm;
173         ASN1_OCTET_STRING *encryptedContent;
174         /* Content encryption algorithm and key */
175         const EVP_CIPHER *cipher;
176         unsigned char *key;
177         size_t keylen;
178         /* Set to 1 if we are debugging decrypt and don't fake keys for MMA */
179         int debug;
180         };

182 struct CMS_RecipientInfo_st
183         {
184         int type;
185         union   {
186                 CMS_KeyTransRecipientInfo *ktri;
187                 CMS_KeyAgreeRecipientInfo *kari;
188                 CMS_KEKRecipientInfo *kekri;
189                 CMS_PasswordRecipientInfo *pwri;
190                 CMS_OtherRecipientInfo *ori;
191                 } d;
192         };
```



```
194 typedef CMS_SignerIdentifier CMS_RecipientIdentifier;

196 struct CMS_KeyTransRecipientInfo_st
197         {
198         long version;
199         CMS_RecipientIdentifier *rid;
200         X509_ALGOR *keyEncryptionAlgorithm;
201         ASN1_OCTET_STRING *encryptedKey;
202         /* Recipient Key and cert */
203         X509 *recip;
204         EVP_PKEY *pkey;
205         };

207 struct CMS_KeyAgreeRecipientInfo_st
208         {
209         long version;
210         CMS_OriginatorIdentifierOrKey *originator;
211         ASN1_OCTET_STRING *ukm;
212         X509_ALGOR *keyEncryptionAlgorithm;
213         STACK_OF(CMS_RecipientEncryptedKey) *recipientEncryptedKeys;
214         };

216 struct CMS_OriginatorIdentifierOrKey_st
217         {
218         int type;
219         union   {
220                 CMS_IssuerAndSerialNumber *issuerAndSerialNumber;
221                 ASN1_OCTET_STRING *subjectKeyIdentifier;
222                 CMS_OriginatorPublicKey *originatorKey;
223                 } d;
224         };

226 struct CMS_OriginatorPublicKey_st
227         {
228         X509_ALGOR *algorithm;
229         ASN1_BIT_STRING *publicKey;
230         };

232 struct CMS_RecipientEncryptedKey_st
233         {
234         CMS_KeyAgreeRecipientIdentifier *rid;
235         ASN1_OCTET_STRING *encryptedKey;
236         };

238 struct CMS_KeyAgreeRecipientIdentifier_st
239         {
240         int type;
241         union   {
242                 CMS_IssuerAndSerialNumber *issuerAndSerialNumber;
243                 CMS_RecipientKeyIdentifier *rKeyId;
244                 } d;
245         };

247 struct CMS_RecipientKeyIdentifier_st
248         {
249         ASN1_OCTET_STRING *subjectKeyIdentifier;
250         ASN1_GENERALIZEDTIME *date;
251         CMS_OtherKeyAttribute *other;
252         };

254 struct CMS_KEKRecipientInfo_st
255         {
256         long version;
257         CMS_KEKIdentifier *kekid;
258         X509_ALGOR *keyEncryptionAlgorithm;
259         ASN1_OCTET_STRING *encryptedKey;
```

```
260         /* Extra info: symmetric key to use */
261         unsigned char *key;
262         size_t keylen;
263         };

265 struct CMS_KEKIdentifier_st
266         {
267         ASN1_OCTET_STRING *keyIdentifier;
268         ASN1_GENERALIZEDTIME *date;
269         CMS_OtherKeyAttribute *other;
270         };

272 struct CMS_PasswordRecipientInfo_st
273         {
274         long version;
275         X509_ALGOR *keyDerivationAlgorithm;
276         X509_ALGOR *keyEncryptionAlgorithm;
277         ASN1_OCTET_STRING *encryptedKey;
278         /* Extra info: password to use */
279         unsigned char *pass;
280         size_t passlen;
281         };

283 struct CMS_OtherRecipientInfo_st
284         {
285         ASN1_OBJECT *oriType;
286         ASN1_TYPE *oriValue;
287         };

289 struct CMS_DigestedData_st
290         {
291         long version;
292         X509_ALGOR *digestAlgorithm;
293         CMS_EncapsulatedContentInfo *encapContentInfo;
294         ASN1_OCTET_STRING *digest;
295         };

297 struct CMS_EncryptedData_st
298         {
299         long version;
300         CMS_EncryptedContentInfo *encryptedContentInfo;
301         STACK_OF(X509_ATTRIBUTE) *unprotectedAttrs;
302         };

304 struct CMS_AuthenticatedData_st
305         {
306         long version;
307         CMS_OriginatorInfo *originatorInfo;
308         STACK_OF(CMS_RecipientInfo) *recipientInfos;
309         X509_ALGOR *macAlgorithm;
310         X509_ALGOR *digestAlgorithm;
311         CMS_EncapsulatedContentInfo *encapContentInfo;
312         STACK_OF(X509_ATTRIBUTE) *authAttrs;
313         ASN1_OCTET_STRING *mac;
314         STACK_OF(X509_ATTRIBUTE) *unauthAttrs;
315         };

317 struct CMS_CompressedData_st
318         {
319         long version;
320         X509_ALGOR *compressionAlgorithm;
321         STACK_OF(CMS_RecipientInfo) *recipientInfos;
322         CMS_EncapsulatedContentInfo *encapContentInfo;
323         };

325 struct CMS_RevocationInfoChoice_st
```



```
326         {
327         int type;
328         union   {
329                 X509_CRL *crl;
330                 CMS_OtherRevocationInfoFormat *other;
331                 } d;
332         };

334 #define CMS_REVCHOICE_CRL               0
335 #define CMS_REVCHOICE_OTHER             1

337 struct CMS_OtherRevocationInfoFormat_st
338         {
339         ASN1_OBJECT *otherRevInfoFormat;
340         ASN1_TYPE *otherRevInfo;
341         };

343 struct CMS_CertificateChoices
344         {
345         int type;
346                 union {
347                 X509 *certificate;
348                 ASN1_STRING *extendedCertificate;       /* Obsolete */
349                 ASN1_STRING *v1AttrCert;        /* Left encoded for now */
350                 ASN1_STRING *v2AttrCert;        /* Left encoded for now */
351                 CMS_OtherCertificateFormat *other;
352                 } d;
353         };

355 #define CMS_CERTCHOICE_CERT             0
356 #define CMS_CERTCHOICE_EXCERT           1
357 #define CMS_CERTCHOICE_V1ACERT          2
358 #define CMS_CERTCHOICE_V2ACERT          3
359 #define CMS_CERTCHOICE_OTHER            4

361 struct CMS_OtherCertificateFormat_st
362         {
363         ASN1_OBJECT *otherCertFormat;
364         ASN1_TYPE *otherCert;
365         };

367 /* This is also defined in pkcs7.h but we duplicate it
368  * to allow the CMS code to be independent of PKCS#7
369  */

371 struct CMS_IssuerAndSerialNumber_st
372         {
373         X509_NAME *issuer;
374         ASN1_INTEGER *serialNumber;
375         };

377 struct CMS_OtherKeyAttribute_st
378         {
379         ASN1_OBJECT *keyAttrId;
380         ASN1_TYPE *keyAttr;
381         };

383 /* ESS structures */

385 #ifdef HEADER_X509V3_H

387 struct CMS_ReceiptRequest_st
388         {
389         ASN1_OCTET_STRING *signedContentIdentifier;
390         CMS_ReceiptsFrom *receiptsFrom;
391         STACK_OF(GENERAL_NAMES) *receiptsTo;
```

```
392         };


395 struct CMS_ReceiptsFrom_st
396         {
397         int type;
398         union
399                 {
400                 long allOrFirstTier;
401                 STACK_OF(GENERAL_NAMES) *receiptList;
402                 } d;
403         };
404 #endif

406 struct CMS_Receipt_st
407         {
408         long version;
409         ASN1_OBJECT *contentType;
410         ASN1_OCTET_STRING *signedContentIdentifier;
411         ASN1_OCTET_STRING *originatorSignatureValue;
412         };

414 DECLARE_ASN1_FUNCTIONS(CMS_ContentInfo)
415 DECLARE_ASN1_ITEM(CMS_SignerInfo)
416 DECLARE_ASN1_ITEM(CMS_IssuerAndSerialNumber)
417 DECLARE_ASN1_ITEM(CMS_Attributes_Sign)
418 DECLARE_ASN1_ITEM(CMS_Attributes_Verify)
419 DECLARE_ASN1_ITEM(CMS_RecipientInfo)
420 DECLARE_ASN1_ITEM(CMS_PasswordRecipientInfo)
421 DECLARE_ASN1_ALLOC_FUNCTIONS(CMS_IssuerAndSerialNumber)

423 #define CMS_SIGNERINFO_ISSUER_SERIAL    0
424 #define CMS_SIGNERINFO_KEYIDENTIFIER    1

426 #define CMS_RECIPINFO_ISSUER_SERIAL     0
427 #define CMS_RECIPINFO_KEYIDENTIFIER     1

429 BIO *cms_content_bio(CMS_ContentInfo *cms);

431 CMS_ContentInfo *cms_Data_create(void);

433 CMS_ContentInfo *cms_DigestedData_create(const EVP_MD *md);
434 BIO *cms_DigestedData_init_bio(CMS_ContentInfo *cms);
435 int cms_DigestedData_do_final(CMS_ContentInfo *cms, BIO *chain, int verify);

437 BIO *cms_SignedData_init_bio(CMS_ContentInfo *cms);
438 int cms_SignedData_final(CMS_ContentInfo *cms, BIO *chain);
439 int cms_set1_SignerIdentifier(CMS_SignerIdentifier *sid, X509 *cert, int type);
440 int cms_SignerIdentifier_get0_signer_id(CMS_SignerIdentifier *sid,
441                                         ASN1_OCTET_STRING **keyid,
442                                         X509_NAME **issuer, ASN1_INTEGER **sno);
443 int cms_SignerIdentifier_cert_cmp(CMS_SignerIdentifier *sid, X509 *cert);

445 CMS_ContentInfo *cms_CompressedData_create(int comp_nid);
446 BIO *cms_CompressedData_init_bio(CMS_ContentInfo *cms);

448 void cms_DigestAlgorithm_set(X509_ALGOR *alg, const EVP_MD *md);
449 BIO *cms_DigestAlgorithm_init_bio(X509_ALGOR *digestAlgorithm);
450 int cms_DigestAlgorithm_find_ctx(EVP_MD_CTX *mctx, BIO *chain,
451                                         X509_ALGOR *mdalg);

453 BIO *cms_EncryptedContent_init_bio(CMS_EncryptedContentInfo *ec);
454 BIO *cms_EncryptedData_init_bio(CMS_ContentInfo *cms);
455 int cms_EncryptedContent_init(CMS_EncryptedContentInfo *ec,
456                               const EVP_CIPHER *cipher,
457                               const unsigned char *key, size_t keylen);
```



```
459 int cms_Receipt_verify(CMS_ContentInfo *cms, CMS_ContentInfo *req_cms);
460 int cms_msgSigDigest_add1(CMS_SignerInfo *dest, CMS_SignerInfo *src);
461 ASN1_OCTET_STRING *cms_encode_Receipt(CMS_SignerInfo *si);

463 BIO *cms_EnvelopedData_init_bio(CMS_ContentInfo *cms);
464 CMS_EnvelopedData *cms_get0_enveloped(CMS_ContentInfo *cms);

466 /* PWRI routines */
467 int cms_RecipientInfo_pwri_crypt(CMS_ContentInfo *cms, CMS_RecipientInfo *ri,
468                                                         int en_de);

470 #ifdef  __cplusplus
471 }
472 #endif
473 #endif
474 #endif /* ! codereview */
```

```
**********************************************************
    9243 Wed Aug 13 19:51:35 2014
new/usr/src/lib/openssl/include/conf_def.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/conf/conf_def.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* THIS FILE WAS AUTOMAGICALLY GENERATED!
   Please modify and use keysets.pl to regenerate it. */
```



```
#define CONF_NUMBER             1
#define CONF_UPPER              2
#define CONF_LOWER              4
#define CONF_UNDER              256
#define CONF_PUNCTUATION        512
#define CONF_WS                 16
#define CONF_ESC                32
#define CONF_QUOTE              64
#define CONF_DQUOTE             1024
#define CONF_COMMENT            128
#define CONF_FCOMMENT           2048
#define CONF_EOF                8
#define CONF_HIGHBIT            4096
#define CONF_ALPHA              (CONF_UPPER|CONF_LOWER)
#define CONF_ALPHA_NUMERIC      (CONF_ALPHA|CONF_NUMBER|CONF_UNDER)
#define CONF_ALPHA_NUMERIC_PUNCT (CONF_ALPHA|CONF_NUMBER|CONF_UNDER| \
                                  CONF_PUNCTUATION)

#define KEYTYPES(c)             ((unsigned short *)((c)->meth_data))
#ifndef CHARSET_EBCDIC
#define IS_COMMENT(c,a)         (KEYTYPES(c)[(a)&0xff]&CONF_COMMENT)
#define IS_FCOMMENT(c,a)        (KEYTYPES(c)[(a)&0xff]&CONF_FCOMMENT)
#define IS_EOF(c,a)             (KEYTYPES(c)[(a)&0xff]&CONF_EOF)
#define IS_ESC(c,a)             (KEYTYPES(c)[(a)&0xff]&CONF_ESC)
#define IS_NUMBER(c,a)          (KEYTYPES(c)[(a)&0xff]&CONF_NUMBER)
#define IS_WS(c,a)              (KEYTYPES(c)[(a)&0xff]&CONF_WS)
#define IS_ALPHA_NUMERIC(c,a)   (KEYTYPES(c)[(a)&0xff]&CONF_ALPHA_NUMERIC)
#define IS_ALPHA_NUMERIC_PUNCT(c,a) \
                                (KEYTYPES(c)[(a)&0xff]&CONF_ALPHA_NUMERIC_PUNCT)
#define IS_QUOTE(c,a)           (KEYTYPES(c)[(a)&0xff]&CONF_QUOTE)
#define IS_DQUOTE(c,a)          (KEYTYPES(c)[(a)&0xff]&CONF_DQUOTE)
#define IS_HIGHBIT(c,a)         (KEYTYPES(c)[(a)&0xff]&CONF_HIGHBIT)

#else /*CHARSET_EBCDIC*/

#define IS_COMMENT(c,a)         (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_COMMENT)
#define IS_FCOMMENT(c,a)        (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_FCOMMENT)
#define IS_EOF(c,a)             (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_EOF)
#define IS_ESC(c,a)             (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_ESC)
#define IS_NUMBER(c,a)          (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_NUMBER)
#define IS_WS(c,a)              (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_WS)
#define IS_ALPHA_NUMERIC(c,a)   (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_ALPHA_NUME
#define IS_ALPHA_NUMERIC_PUNCT(c,a) \
                                (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_ALPHA_NUME
#define IS_QUOTE(c,a)           (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_QUOTE)
#define IS_DQUOTE(c,a)          (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_DQUOTE)
#define IS_HIGHBIT(c,a)         (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_HIGHBIT)
#endif /*CHARSET_EBCDIC*/

static unsigned short CONF_type_default[256]={
        0x0008,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0010,0x0010,0x0000,0x0000,0x0010,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
        0x0010,0x0200,0x0040,0x0080,0x0000,0x0200,0x0200,0x0040,
        0x0000,0x0000,0x0200,0x0200,0x0200,0x0200,0x0200,0x0200,
        0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
        0x0001,0x0001,0x0000,0x0200,0x0000,0x0000,0x0000,0x0200,
        0x0200,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,
        0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,
        0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,
        0x0002,0x0002,0x0002,0x0000,0x0020,0x0000,0x0200,0x0100,
        0x0040,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,
        0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,
        0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,
        0x0004,0x0004,0x0004,0x0000,0x0200,0x0000,0x0200,0x0000,
```

```
128         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
129         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
130         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
131         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
132         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
133         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
134         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
135         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
136         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
137         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
138         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
139         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
140         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
141         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
142         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
143         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
144         };

146 static unsigned short CONF_type_win32[256]={
147         0x0008,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
148         0x0000,0x0010,0x0010,0x0000,0x0000,0x0010,0x0000,0x0000,
149         0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
150         0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
151         0x0010,0x0200,0x0400,0x0000,0x0000,0x0200,0x0200,0x0000,
152         0x0000,0x0000,0x0200,0x0200,0x0200,0x0200,0x0200,0x0200,
153         0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,0x0001,
154         0x0001,0x0001,0x0000,0x0A00,0x0000,0x0000,0x0000,0x0200,
155         0x0200,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,
156         0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,
157         0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,0x0002,
158         0x0002,0x0002,0x0002,0x0000,0x0000,0x0000,0x0200,0x0100,
159         0x0000,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,
160         0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,
161         0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,0x0004,
162         0x0004,0x0004,0x0004,0x0000,0x0200,0x0000,0x0200,0x0000,
163         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
164         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
165         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
166         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
167         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
168         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
169         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
170         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
171         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
172         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
173         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
174         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
175         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
176         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
177         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
178         0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,0x1000,
179         };
180 #endif /* ! codereview */
```

```
**********************************************************
    4432 Wed Aug 13 19:51:36 2014
new/usr/src/lib/openssl/include/cryptlib.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cryptlib.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_CRYPTLIB_H
#define HEADER_CRYPTLIB_H
```



```
#include <stdlib.h>
#include <string.h>

#include "e_os.h"

#ifdef OPENSSL_USE_APPLINK
#define BIO_FLAGS_UPLINK 0x8000
#include "ms/uplink.h"
#endif

#include <openssl/crypto.h>
#include <openssl/buffer.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/opensslconf.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifndef OPENSSL_SYS_VMS
#define X509_CERT_AREA          OPENSSLDIR
#define X509_CERT_DIR           OPENSSLDIR "/certs"
#define X509_CERT_FILE          OPENSSLDIR "/cert.pem"
#define X509_PRIVATE_DIR        OPENSSLDIR "/private"
#else
#define X509_CERT_AREA          "SSLROOT:[000000]"
#define X509_CERT_DIR           "SSLCERTS:"
#define X509_CERT_FILE          "SSLCERTS:cert.pem"
#define X509_PRIVATE_DIR        "SSLPRIVATE:"
#endif

#define X509_CERT_DIR_EVP        "SSL_CERT_DIR"
#define X509_CERT_FILE_EVP       "SSL_CERT_FILE"

/* size of string representations */
#define DECIMAL_SIZE(type)      ((sizeof(type)*8+2)/3+1)
#define HEX_SIZE(type)          (sizeof(type)*2)

void OPENSSL_cpuid_setup(void);
extern unsigned int OPENSSL_ia32cap_P[];
void OPENSSL_showfatal(const char *fmta,...);
void *OPENSSL_stderr(void);
extern int OPENSSL_NONPIC_relocated;

void illumos_locking_setup();

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    2579 Wed Aug 13 19:51:36 2014
new/usr/src/lib/openssl/include/cryptoki.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2003 Sun Microsystems, Inc.   All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _CRYPTOKI_H
#define _CRYPTOKI_H

#pragma ident   "@(#)cryptoki.h 1.2     05/06/08 SMI"

#ifdef  __cplusplus
extern "C" {
#endif

#ifndef CK_PTR
#define CK_PTR *
#endif

#ifndef CK_DEFINE_FUNCTION
#define CK_DEFINE_FUNCTION(returnType, name) returnType name
#endif

#ifndef CK_DECLARE_FUNCTION
#define CK_DECLARE_FUNCTION(returnType, name) returnType name
#endif

#ifndef CK_DECLARE_FUNCTION_POINTER
#define CK_DECLARE_FUNCTION_POINTER(returnType, name) returnType (* name)
#endif

#ifndef CK_CALLBACK_FUNCTION
#define CK_CALLBACK_FUNCTION(returnType, name) returnType (* name)
#endif

#ifndef NULL_PTR
#include <unistd.h>     /* For NULL */
#define NULL_PTR NULL
#endif

/*
```



```
 * pkcs11t.h defines TRUE and FALSE in a way that upsets lint
 */
#ifndef CK_DISABLE_TRUE_FALSE
#define CK_DISABLE_TRUE_FALSE
#ifndef TRUE
#define TRUE    1
#endif /* TRUE */
#ifndef FALSE
#define FALSE   0
#endif /* FALSE */
#endif /* CK_DISABLE_TRUE_FALSE */

#undef CK_PKCS11_FUNCTION_INFO

#include "pkcs11.h"

/* Solaris specific functions */

#include <stdlib.h>

/*
 * SUNW_C_GetMechSession will initialize the framework and do all
 * the necessary PKCS#11 calls to create a session capable of
 * providing operations on the requested mechanism
 */
CK_RV SUNW_C_GetMechSession(CK_MECHANISM_TYPE mech,
    CK_SESSION_HANDLE_PTR hSession);

/*
 * SUNW_C_KeyToObject will create a secret key object for the given
 * mechanism from the rawkey data.
 */
CK_RV SUNW_C_KeyToObject(CK_SESSION_HANDLE hSession,
    CK_MECHANISM_TYPE mech, const void *rawkey, size_t rawkey_len,
    CK_OBJECT_HANDLE_PTR obj);


#ifdef  __cplusplus
}
#endif

#endif  /* _CRYPTOKI_H */
#endif /* ! codereview */
```

```
**********************************************************
   13700 Wed Aug 13 19:51:36 2014
new/usr/src/lib/openssl/include/des_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/des_locl.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_DES_LOCL_H
#define HEADER_DES_LOCL_H
```



```
#include <openssl/e_os2.h>

#if defined(OPENSSL_SYS_WIN32)
#ifndef OPENSSL_SYS_MSDOS
#define OPENSSL_SYS_MSDOS
#endif
#endif

#include <stdio.h>
#include <stdlib.h>

#ifndef OPENSSL_SYS_MSDOS
#if !defined(OPENSSL_SYS_VMS) || defined(__DECC)
#ifdef OPENSSL_UNISTD
# include OPENSSL_UNISTD
#else
# include <unistd.h>
#endif
#include <math.h>
#endif
#endif
#include <openssl/des.h>

#ifdef OPENSSL_SYS_MSDOS                /* Visual C++ 2.1 (Windows NT/95) */
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <io.h>
#endif

#if defined(__STDC__) || defined(OPENSSL_SYS_VMS) || defined(M_XENIX) || defined
#include <string.h>
#endif

#ifdef OPENSSL_BUILD_SHLIBCRYPTO
# undef OPENSSL_EXTERN
# define OPENSSL_EXTERN OPENSSL_EXPORT
#endif

#define ITERATIONS 16
#define HALF_ITERATIONS 8

/* used in des_read and des_write */
#define MAXWRITE        (1024*16)
#define BSIZE           (MAXWRITE+4)

#define c2l(c,l)        (l =((DES_LONG)(*((c)++)))    , \
                         l|=((DES_LONG)(*((c)++)))<< 8L, \
                         l|=((DES_LONG)(*((c)++)))<<16L, \
                         l|=((DES_LONG)(*((c)++)))<<24L)

/* NOTE - c is not incremented as per c2l */
#define c2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((DES_LONG)(*(--(c))))<<24L; \
                        case 7: l2|=((DES_LONG)(*(--(c))))<<16L; \
                        case 6: l2|=((DES_LONG)(*(--(c))))<< 8L; \
                        case 5: l2|=((DES_LONG)(*(--(c)))); \
                        case 4: l1 =((DES_LONG)(*(--(c))))<<24L; \
                        case 3: l1|=((DES_LONG)(*(--(c))))<<16L; \
                        case 2: l1|=((DES_LONG)(*(--(c))))<< 8L; \
                        case 1: l1|=((DES_LONG)(*(--(c)))); \
                                } \
                        }
```

```
129 #define l2c(l,c)        (*((c)++)=(unsigned char)(((l)     )&0xff), \
130                          *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
131                          *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
132                          *((c)++)=(unsigned char)(((l)>>24L)&0xff))

134 /* replacements for htonl and ntohl since I have no idea what to do
135  * when faced with machines with 8 byte longs. */
136 #define HDRSIZE 4

138 #define n2l(c,l)        (l =((DES_LONG)(*((c)++)))<<24L, \
139                          l|=((DES_LONG)(*((c)++)))<<16L, \
140                          l|=((DES_LONG)(*((c)++)))<< 8L, \
141                          l|=((DES_LONG)(*((c)++))))

143 #define l2n(l,c)        (*((c)++)=(unsigned char)(((l)>>24L)&0xff), \
144                          *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
145                          *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
146                          *((c)++)=(unsigned char)(((l)     )&0xff))

148 /* NOTE - c is not incremented as per l2c */
149 #define l2cn(l1,l2,c,n) { \
150                         c+=n; \
151                         switch (n) { \
152                         case 8: *(--(c))=(unsigned char)(((l2)>>24L)&0xff); \
153                         case 7: *(--(c))=(unsigned char)(((l2)>>16L)&0xff); \
154                         case 6: *(--(c))=(unsigned char)(((l2)>> 8L)&0xff); \
155                         case 5: *(--(c))=(unsigned char)(((l2)     )&0xff); \
156                         case 4: *(--(c))=(unsigned char)(((l1)>>24L)&0xff); \
157                         case 3: *(--(c))=(unsigned char)(((l1)>>16L)&0xff); \
158                         case 2: *(--(c))=(unsigned char)(((l1)>> 8L)&0xff); \
159                         case 1: *(--(c))=(unsigned char)(((l1)     )&0xff); \
160                                 } \
161                         }

163 #if (defined(OPENSSL_SYS_WIN32) && defined(_MSC_VER)) || defined(__ICC)
164 #define ROTATE(a,n)     (_lrotr(a,n))
165 #elif defined(__GNUC__) && __GNUC__>=2 && !defined(__STRICT_ANSI__) && !defined(
166 # if defined(__i386) || defined(__i386__) || defined(__x86_64) || defined(__x86_
167 #  define ROTATE(a,n)   ({ register unsigned int ret;   \
168                                 __asm__ ("rorl %1,%0"   \
169                                         : "=r"(ret)     \
170                                         : "I"(n),"0"(a) \
171                                         : "cc");        \
172                            ret;                         \
173                         })
174 # endif
175 #endif
176 #ifndef ROTATE
177 #define ROTATE(a,n)     (((a)>>(n))+((a)<<(32-(n))))
178 #endif

180 /* Don't worry about the LOAD_DATA() stuff, that is used by
181  * fcrypt() to add it's little bit to the front */

183 #ifdef DES_FCRYPT

185 #define LOAD_DATA_tmp(R,S,u,t,E0,E1) \
186         { DES_LONG tmp; LOAD_DATA(R,S,u,t,E0,E1,tmp); }

188 #define LOAD_DATA(R,S,u,t,E0,E1,tmp) \
189         t=R^(R>>16L); \
190         u=t&E0; t&=E1; \
191         tmp=(u<<16); u^=R^s[S  ]; u^=tmp; \
192         tmp=(t<<16); t^=R^s[S+1]; t^=tmp
193 #else
```



```
194 #define LOAD_DATA_tmp(a,b,c,d,e,f) LOAD_DATA(a,b,c,d,e,f,g)
195 #define LOAD_DATA(R,S,u,t,E0,E1,tmp) \
196         u=R^s[S  ]; \
197         t=R^s[S+1]
198 #endif

200 /* The changes to this macro may help or hinder, depending on the
201  * compiler and the architecture.  gcc2 always seems to do well :-).
202  * Inspired by Dana How <how@isl.stanford.edu>
203  * DO NOT use the alternative version on machines with 8 byte longs.
204  * It does not seem to work on the Alpha, even when DES_LONG is 4
205  * bytes, probably an issue of accessing non-word aligned objects :-( */
206 #ifdef DES_PTR

208 /* It recently occurred to me that 0^0^0^0^0^0^0 == 0, so there
209  * is no reason to not xor all the sub items together.  This potentially
210  * saves a register since things can be xored directly into L */

212 #if defined(DES_RISC1) || defined(DES_RISC2)
213 #ifdef DES_RISC1
214 #define D_ENCRYPT(LL,R,S) { \
215         unsigned int u1,u2,u3; \
216         LOAD_DATA(R,S,u,t,E0,E1,u1); \
217         u2=(int)u>>8L; \
218         u1=(int)u&0xfc; \
219         u2&=0xfc; \
220         t=ROTATE(t,4); \
221         u>>=16L; \
222         LL^= *(const DES_LONG *)(des_SP      +u1); \
223         LL^= *(const DES_LONG *)(des_SP+0x200+u2); \
224         u3=(int)(u>>8L); \
225         u1=(int)u&0xfc; \
226         u3&=0xfc; \
227         LL^= *(const DES_LONG *)(des_SP+0x400+u1); \
228         LL^= *(const DES_LONG *)(des_SP+0x600+u3); \
229         u2=(int)t>>8L; \
230         u1=(int)t&0xfc; \
231         u2&=0xfc; \
232         t>>=16L; \
233         LL^= *(const DES_LONG *)(des_SP+0x100+u1); \
234         LL^= *(const DES_LONG *)(des_SP+0x300+u2); \
235         u3=(int)t>>8L; \
236         u1=(int)t&0xfc; \
237         u3&=0xfc; \
238         LL^= *(const DES_LONG *)(des_SP+0x500+u1); \
239         LL^= *(const DES_LONG *)(des_SP+0x700+u3); }
240 #endif
241 #ifdef DES_RISC2
242 #define D_ENCRYPT(LL,R,S) { \
243         unsigned int u1,u2,s1,s2; \
244         LOAD_DATA(R,S,u,t,E0,E1,u1); \
245         u2=(int)u>>8L; \
246         u1=(int)u&0xfc; \
247         u2&=0xfc; \
248         t=ROTATE(t,4); \
249         LL^= *(const DES_LONG *)(des_SP      +u1); \
250         LL^= *(const DES_LONG *)(des_SP+0x200+u2); \
251         s1=(int)(u>>16L); \
252         s2=(int)(u>>24L); \
253         s1&=0xfc; \
254         s2&=0xfc; \
255         LL^= *(const DES_LONG *)(des_SP+0x400+s1); \
256         LL^= *(const DES_LONG *)(des_SP+0x600+s2); \
257         u2=(int)t>>8L; \
258         u1=(int)t&0xfc; \
259         u2&=0xfc; \
```

```
260         LL^= *(const DES_LONG *)(des_SP+0x100+u1); \
261         LL^= *(const DES_LONG *)(des_SP+0x300+u2); \
262         s1=(int)(t>>16L); \
263         s2=(int)(t>>24L); \
264         s1&=0xfc; \
265         s2&=0xfc; \
266         LL^= *(const DES_LONG *)(des_SP+0x500+s1); \
267         LL^= *(const DES_LONG *)(des_SP+0x700+s2); }
268 #endif
269 #else
270 #define D_ENCRYPT(LL,R,S) { \
271         LOAD_DATA_tmp(R,S,u,t,E0,E1); \
272         t=ROTATE(t,4); \
273         LL^= \
274         *(const DES_LONG *)(des_SP      +((u     )&0xfc))^ \
275         *(const DES_LONG *)(des_SP+0x200+((u>> 8L)&0xfc))^ \
276         *(const DES_LONG *)(des_SP+0x400+((u>>16L)&0xfc))^ \
277         *(const DES_LONG *)(des_SP+0x600+((u>>24L)&0xfc))^ \
278         *(const DES_LONG *)(des_SP+0x100+((t     )&0xfc))^ \
279         *(const DES_LONG *)(des_SP+0x300+((t>> 8L)&0xfc))^ \
280         *(const DES_LONG *)(des_SP+0x500+((t>>16L)&0xfc))^ \
281         *(const DES_LONG *)(des_SP+0x700+((t>>24L)&0xfc)); }
282 #endif

284 #else /* original version */

286 #if defined(DES_RISC1) || defined(DES_RISC2)
287 #ifdef DES_RISC1
288 #define D_ENCRYPT(LL,R,S) {\
289         unsigned int u1,u2,u3; \
290         LOAD_DATA(R,S,u,t,E0,E1,u1); \
291         u>>=2L; \
292         t=ROTATE(t,6); \
293         u2=(int)u>>8L; \
294         u1=(int)u&0x3f; \
295         u2&=0x3f; \
296         u>>=16L; \
297         LL^=DES_SPtrans[0][u1]; \
298         LL^=DES_SPtrans[2][u2]; \
299         u3=(int)u>>8L; \
300         u1=(int)u&0x3f; \
301         u3&=0x3f; \
302         LL^=DES_SPtrans[4][u1]; \
303         LL^=DES_SPtrans[6][u3]; \
304         u2=(int)t>>8L; \
305         u1=(int)t&0x3f; \
306         u2&=0x3f; \
307         t>>=16L; \
308         LL^=DES_SPtrans[1][u1]; \
309         LL^=DES_SPtrans[3][u2]; \
310         u3=(int)t>>8L; \
311         u1=(int)t&0x3f; \
312         u3&=0x3f; \
313         LL^=DES_SPtrans[5][u1]; \
314         LL^=DES_SPtrans[7][u3]; }
315 #endif
316 #ifdef DES_RISC2
317 #define D_ENCRYPT(LL,R,S) {\
318         unsigned int u1,u2,s1,s2; \
319         LOAD_DATA(R,S,u,t,E0,E1,u1); \
320         u>>=2L; \
321         t=ROTATE(t,6); \
322         u2=(int)u>>8L; \
323         u1=(int)u&0x3f; \
324         u2&=0x3f; \
325         LL^=DES_SPtrans[0][u1]; \
```



```
326         LL^=DES_SPtrans[2][u2]; \
327         s1=(int)u>>16L; \
328         s2=(int)u>>24L; \
329         s1&=0x3f; \
330         s2&=0x3f; \
331         LL^=DES_SPtrans[4][s1]; \
332         LL^=DES_SPtrans[6][s2]; \
333         u2=(int)t>>8L; \
334         u1=(int)t&0x3f; \
335         u2&=0x3f; \
336         LL^=DES_SPtrans[1][u1]; \
337         LL^=DES_SPtrans[3][u2]; \
338         s1=(int)t>>16; \
339         s2=(int)t>>24L; \
340         s1&=0x3f; \
341         s2&=0x3f; \
342         LL^=DES_SPtrans[5][s1]; \
343         LL^=DES_SPtrans[7][s2]; }
344 #endif

346 #else

348 #define D_ENCRYPT(LL,R,S) {\
349         LOAD_DATA_tmp(R,S,u,t,E0,E1); \
350         t=ROTATE(t,4); \
351         LL^=\
352                 DES_SPtrans[0][(u>> 2L)&0x3f]^ \
353                 DES_SPtrans[2][(u>>10L)&0x3f]^ \
354                 DES_SPtrans[4][(u>>18L)&0x3f]^ \
355                 DES_SPtrans[6][(u>>26L)&0x3f]^ \
356                 DES_SPtrans[1][(t>> 2L)&0x3f]^ \
357                 DES_SPtrans[3][(t>>10L)&0x3f]^ \
358                 DES_SPtrans[5][(t>>18L)&0x3f]^ \
359                 DES_SPtrans[7][(t>>26L)&0x3f]; }
360 #endif
361 #endif

363         /* IP and FP
364          * The problem is more of a geometric problem that random bit fiddling.
365          0  1  2  3  4  5  6  7      62 54 46 38 30 22 14  6
366          8  9 10 11 12 13 14 15      60 52 44 36 28 20 12  4
367         16 17 18 19 20 21 22 23      58 50 42 34 26 18 10  2
368         24 25 26 27 28 29 30 31  to  56 48 40 32 24 16  8  0

370         32 33 34 35 36 37 38 39      63 55 47 39 31 23 15  7
371         40 41 42 43 44 45 46 47      61 53 45 37 29 21 13  5
372         48 49 50 51 52 53 54 55      59 51 43 35 27 19 11  3
373         56 57 58 59 60 61 62 63      57 49 41 33 25 17  9  1

375         The output has been subject to swaps of the form
376         0 1 -> 3 1 but the odd and even bits have been put into
377         2 3    2 0
378         different words.  The main trick is to remember that
379         t=((l>>size)^r)&(mask);
380         r^=t;
381         l^=(t<<size);
382         can be used to swap and move bits between words.

384         So l =  0  1  2  3  r = 16 17 18 19
385                 4  5  6  7      20 21 22 23
386                 8  9 10 11      24 25 26 27
387                12 13 14 15      28 29 30 31
388         becomes (for size == 2 and mask == 0x3333)
389            t =   2^16  3^17 -- --   l =  0  1 16 17  r =  2  3 18 19
390                  6^20  7^21 -- --        4  5 20 21       6  7 22 23
391                 10^24 11^25 -- --        8  9 24 25      10 11 24 25
```

```
392                 14^28 15^29 -- --       12 13 28 29      14 15 28 29

394         Thanks for hints from Richard Outerbridge - he told me IP&FP
395         could be done in 15 xor, 10 shifts and 5 ands.
396         When I finally started to think of the problem in 2D
397         I first got ~42 operations without xors.  When I remembered
398         how to use xors :-) I got it to its final state.
399         */
400 #define PERM_OP(a,b,t,n,m) ((t)=((((a)>>(n))^(b))&(m)),\
401         (b)^=(t),\
402         (a)^=((t)<<(n)))

404 #define IP(l,r) \
405         { \
406         register DES_LONG tt; \
407         PERM_OP(r,l,tt, 4,0x0f0f0f0fL); \
408         PERM_OP(l,r,tt,16,0x0000ffffL); \
409         PERM_OP(r,l,tt, 2,0x33333333L); \
410         PERM_OP(l,r,tt, 8,0x00ff00ffL); \
411         PERM_OP(r,l,tt, 1,0x55555555L); \
412         }

414 #define FP(l,r) \
415         { \
416         register DES_LONG tt; \
417         PERM_OP(l,r,tt, 1,0x55555555L); \
418         PERM_OP(r,l,tt, 8,0x00ff00ffL); \
419         PERM_OP(l,r,tt, 2,0x33333333L); \
420         PERM_OP(r,l,tt,16,0x0000ffffL); \
421         PERM_OP(l,r,tt, 4,0x0f0f0f0fL); \
422         }

424 extern const DES_LONG DES_SPtrans[8][64];

426 void fcrypt_body(DES_LONG *out,DES_key_schedule *ks,
427                  DES_LONG Eswap0, DES_LONG Eswap1);

429 #ifdef OPENSSL_SMALL_FOOTPRINT
430 #undef DES_UNROLL
431 #endif
432 #endif
433 #endif /* ! codereview */
```

```
**********************************************************
    3637 Wed Aug 13 19:51:36 2014
new/usr/src/lib/openssl/include/des_ver.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/des_ver.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/e_os2.h>

#ifdef OPENSSL_BUILD_SHLIBCRYPTO
```



```
# undef OPENSSL_EXTERN
# define OPENSSL_EXTERN OPENSSL_EXPORT
#endif

/* The following macros make sure the names are different from libdes names */
#define DES_version OSSL_DES_version
#define libdes_version OSSL_libdes_version

OPENSSL_EXTERN const char OSSL_DES_version[];   /* SSLeay version string */
OPENSSL_EXTERN const char OSSL_libdes_version[];        /* old libdes version st
#endif /* ! codereview */
```

****
2868 Wed Aug 13 19:51:36 2014
new/usr/src/lib/openssl/include/dsa_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* ====================================================================
 * Copyright (c) 2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/dsa.h>

int dsa_builtin_paramgen(DSA *ret, size_t bits, size_t qbits,
        const EVP_MD *evpmd, const unsigned char *seed_in, size_t seed_len,
        unsigned char *seed_out,
        int *counter_ret, unsigned long *h_ret, BN_GENCB *cb);
#endif /* ! codereview */
```

```
**********************************************************
   23512 Wed Aug 13 19:51:36 2014
new/usr/src/lib/openssl/include/e_os.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* e_os.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_E_OS_H
#define HEADER_E_OS_H
```



```
#include <openssl/opensslconf.h>

#include <openssl/e_os2.h>
/* <openssl/e_os2.h> contains what we can justify to make visible
 * to the outside; this file e_os.h is not part of the exported
 * interface. */

#ifdef  __cplusplus
extern "C" {
#endif

/* Used to checking reference counts, most while doing perl5 stuff :-) */
#ifdef REF_PRINT
#undef REF_PRINT
#define REF_PRINT(a,b)  fprintf(stderr,"%08X:%4d:%s\n",(int)b,b->references,a)
#endif

#ifndef DEVRANDOM
/* set this to a comma-separated list of 'random' device files to try out.
 * My default, we will try to read at least one of these files */
#define DEVRANDOM "/dev/urandom","/dev/random","/dev/srandom"
#endif
#ifndef DEVRANDOM_EGD
/* set this to a comma-seperated list of 'egd' sockets to try out. These
 * sockets will be tried in the order listed in case accessing the device files
 * listed in DEVRANDOM did not return enough entropy. */
#define DEVRANDOM_EGD "/var/run/egd-pool","/dev/egd-pool","/etc/egd-pool","/etc/
#endif

#if defined(OPENSSL_SYS_VXWORKS)
#  define NO_SYS_PARAM_H
#  define NO_CHMOD
#  define NO_SYSLOG
#endif

#if defined(OPENSSL_SYS_MACINTOSH_CLASSIC)
# if macintosh==1
#  ifndef MAC_OS_GUSI_SOURCE
#    define MAC_OS_pre_X
#    define NO_SYS_TYPES_H
#  endif
#  define NO_SYS_PARAM_H
#  define NO_CHMOD
#  define NO_SYSLOG
#  undef  DEVRANDOM
#  define GETPID_IS_MEANINGLESS
# endif
#endif

/*************************************************************************
 The Microsoft section
 ************************************************************************/
/* The following is used because of the small stack in some
 * Microsoft operating systems */
#if defined(OPENSSL_SYS_MSDOS) && !defined(OPENSSL_SYSNAME_WIN32)
#  define MS_STATIC     static
#else
#  define MS_STATIC
#endif

#if defined(OPENSSL_SYS_WIN32) && !defined(WIN32)
#  define WIN32
#endif
#if defined(OPENSSL_SYS_WINDOWS) && !defined(WINDOWS)
#  define WINDOWS
#endif
```

```
#if defined(OPENSSL_SYS_MSDOS) && !defined(MSDOS)
#  define MSDOS
#endif

#if defined(MSDOS) && !defined(GETPID_IS_MEANINGLESS)
#  define GETPID_IS_MEANINGLESS
#endif

#ifdef WIN32
#define get_last_sys_error()    GetLastError()
#define clear_sys_error()       SetLastError(0)
#if !defined(WINNT)
#define WIN_CONSOLE_BUG
#endif
#else
#define get_last_sys_error()    errno
#define clear_sys_error()       errno=0
#endif

#if defined(WINDOWS)
#define get_last_socket_error() WSAGetLastError()
#define clear_socket_error()    WSASetLastError(0)
#define readsocket(s,b,n)       recv((s),(b),(n),0)
#define writesocket(s,b,n)      send((s),(b),(n),0)
#elif defined(__DJGPP__)
#define WATT32
#define get_last_socket_error() errno
#define clear_socket_error()    errno=0
#define closesocket(s)          close_s(s)
#define readsocket(s,b,n)       read_s(s,b,n)
#define writesocket(s,b,n)      send(s,b,n,0)
#elif defined(MAC_OS_pre_X)
#define get_last_socket_error() errno
#define clear_socket_error()    errno=0
#define closesocket(s)          MacSocket_close(s)
#define readsocket(s,b,n)       MacSocket_recv((s),(b),(n),true)
#define writesocket(s,b,n)      MacSocket_send((s),(b),(n))
#elif defined(OPENSSL_SYS_VMS)
#define get_last_socket_error() errno
#define clear_socket_error()    errno=0
#define ioctlsocket(a,b,c)      ioctl(a,b,c)
#define closesocket(s)          close(s)
#define readsocket(s,b,n)       recv((s),(b),(n),0)
#define writesocket(s,b,n)      send((s),(b),(n),0)
#elif defined(OPENSSL_SYS_VXWORKS)
#define get_last_socket_error() errno
#define clear_socket_error()    errno=0
#define ioctlsocket(a,b,c)          ioctl((a),(b),(int)(c))
#define closesocket(s)              close(s)
#define readsocket(s,b,n)           read((s),(b),(n))
#define writesocket(s,b,n)          write((s),(char *)(b),(n))
#elif defined(OPENSSL_SYS_BEOS_R5)
#define get_last_socket_error() errno
#define clear_socket_error()    errno=0
#define FIONBIO SO_NONBLOCK
#define ioctlsocket(a,b,c)              setsockopt((a),SOL_SOCKET,(b),(c),size
#define readsocket(s,b,n)       recv((s),(b),(n),0)
#define writesocket(s,b,n)      send((s),(b),(n),0)
#elif defined(OPENSSL_SYS_NETWARE)
#if defined(NETWARE_BSDSOCK)
#define get_last_socket_error() errno
#define clear_socket_error()    errno=0
#define closesocket(s)          close(s)
#define ioctlsocket(a,b,c)      ioctl(a,b,c)
#if defined(NETWARE_LIBC)
#define readsocket(s,b,n)       recv((s),(b),(n),0)
```



```
#define writesocket(s,b,n)      send((s),(b),(n),0)
#else
#define readsocket(s,b,n)       recv((s),(char*)(b),(n),0)
#define writesocket(s,b,n)      send((s),(char*)(b),(n),0)
#endif
#else
#define get_last_socket_error() WSAGetLastError()
#define clear_socket_error()    WSASetLastError(0)
#define readsocket(s,b,n)               recv((s),(b),(n),0)
#define writesocket(s,b,n)              send((s),(b),(n),0)
#endif
#else
#define get_last_socket_error() errno
#define clear_socket_error()    errno=0
#define ioctlsocket(a,b,c)      ioctl(a,b,c)
#define closesocket(s)          close(s)
#define readsocket(s,b,n)       read((s),(b),(n))
#define writesocket(s,b,n)      write((s),(b),(n))
#endif

#ifdef WIN16 /* never the case */
#  define MS_CALLBACK   _far _loadds
#  define MS_FAR        _far
#else
#  define MS_CALLBACK
#  define MS_FAR
#endif

#ifdef OPENSSL_NO_STDIO
#  undef OPENSSL_NO_FP_API
#  define OPENSSL_NO_FP_API
#endif

#if (defined(WINDOWS) || defined(MSDOS))

#  ifdef __DJGPP__
#    include <unistd.h>
#    include <sys/stat.h>
#    include <sys/socket.h>
#    include <tcp.h>
#    include <netdb.h>
#    define _setmode setmode
#    define _O_TEXT O_TEXT
#    define _O_BINARY O_BINARY
#    undef DEVRANDOM
#    define DEVRANDOM "/dev/urandom\x24"
#  endif /* __DJGPP__ */

#  ifndef S_IFDIR
#    define S_IFDIR     _S_IFDIR
#  endif

#  ifndef S_IFMT
#    define S_IFMT      _S_IFMT
#  endif

#  if !defined(WINNT) && !defined(__DJGPP__)
#    define NO_SYSLOG
#  endif
#  define NO_DIRENT

#  ifdef WINDOWS
#    if !defined(_WIN32_WCE) && !defined(_WIN32_WINNT)
       /*
        * Defining _WIN32_WINNT here in e_os.h implies certain "discipline."
        * Most notably we ought to check for availability of each specific
```

```
260         * routine with GetProcAddress() and/or guard NT-specific calls with
261         * GetVersion() < 0x80000000. One can argue that in latter "or" case
262         * we ought to /DELAYLOAD some .DLLs in order to protect ourselves
263         * against run-time link errors. This doesn't seem to be necessary,
264         * because it turned out that already Windows 95, first non-NT Win32
265         * implementation, is equipped with at least NT 3.51 stubs, dummy
266         * routines with same name, but which do nothing. Meaning that it's
267         * apparently sufficient to guard "vanilla" NT calls with GetVersion
268         * alone, while NT 4.0 and above interfaces ought to be linked with
269         * GetProcAddress at run-time.
270         */
271 #       define _WIN32_WINNT 0x0400
272 #     endif
273 #     if !defined(OPENSSL_NO_SOCK) && defined(_WIN32_WINNT)
274         /*
275         * Just like defining _WIN32_WINNT including winsock2.h implies
276         * certain "discipline" for maintaining [broad] binary compatibility.
277         * As long as structures are invariant among Winsock versions,
278         * it's sufficient to check for specific Winsock2 API availability
279         * at run-time [DSO_global_lookup is recommended]...
280         */
281 #       include <winsock2.h>
282 #       include <ws2tcpip.h>
283         /* yes, they have to be #included prior to <windows.h> */
284 #     endif
285 #     include <windows.h>
286 #     include <stdio.h>
287 #     include <stddef.h>
288 #     include <errno.h>
289 #     include <string.h>
290 #     ifdef _WIN64
291 #       define strlen(s) _strlen31(s)
292 /* cut strings to 2GB */
293 static unsigned int _strlen31(const char *str)
294         {
295         unsigned int len=0;
296         while (*str && len<0x80000000U) str++, len++;
297         return len&0x7FFFFFFF;
298         }
299 #     endif
300 #     include <malloc.h>
301 #     if defined(_MSC_VER) && _MSC_VER<=1200 && defined(_MT) && defined(isspace)
302         /* compensate for bug in VC6 ctype.h */
303 #       undef isspace
304 #       undef isdigit
305 #       undef isalnum
306 #       undef isupper
307 #       undef isxdigit
308 #     endif
309 #     if defined(_MSC_VER) && !defined(_DLL) && defined(stdin)
310 #       if _MSC_VER>=1300
311 #         undef stdin
312 #         undef stdout
313 #         undef stderr
314           FILE *__iob_func();
315 #         define stdin  (&__iob_func()[0])
316 #         define stdout (&__iob_func()[1])
317 #         define stderr (&__iob_func()[2])
318 #       elif defined(I_CAN_LIVE_WITH_LNK4049)
319 #         undef stdin
320 #         undef stdout
321 #         undef stderr
322           /* pre-1300 has __p__iob(), but it's available only in msvcrt.lib,
323            * or in other words with /MD. Declaring implicit import, i.e.
324            * with _imp_ prefix, works correctly with all compiler options,
325            * but without /MD results in LINK warning LNK4049:
```



```
326           * 'locally defined symbol "__iob" imported'.
327           */
328           extern FILE *_imp___iob;
329 #         define stdin  (&_imp___iob[0])
330 #         define stdout (&_imp___iob[1])
331 #         define stderr (&_imp___iob[2])
332 #       endif
333 #     endif
334 #   endif
335 #   include <io.h>
336 #   include <fcntl.h>

338 #   ifdef OPENSSL_SYS_WINCE
339 #     define OPENSSL_NO_POSIX_IO
340 #   endif

342 #   if defined (__BORLANDC__)
343 #     define _setmode setmode
344 #     define _O_TEXT O_TEXT
345 #     define _O_BINARY O_BINARY
346 #     define _int64 __int64
347 #     define _kbhit kbhit
348 #   endif

350 #   define EXIT(n) exit(n)
351 #   define LIST_SEPARATOR_CHAR ';'
352 #   ifndef X_OK
353 #     define X_OK        0
354 #   endif
355 #   ifndef W_OK
356 #     define W_OK        2
357 #   endif
358 #   ifndef R_OK
359 #     define R_OK        4
360 #   endif
361 #   define OPENSSL_CONF  "openssl.cnf"
362 #   define SSLEAY_CONF   OPENSSL_CONF
363 #   define NUL_DEV       "nul"
364 #   define RFILE         ".rnd"
365 #   ifdef OPENSSL_SYS_WINCE
366 #     define DEFAULT_HOME  ""
367 #   else
368 #     define DEFAULT_HOME  "C:"
369 #   endif

371 /* Avoid Windows 8 SDK GetVersion deprecated problems */
372 #if defined(_MSC_VER) && _MSC_VER>=1800
373 #  define check_winnt() (1)
374 #else
375 #  define check_winnt() (GetVersion() < 0x80000000)
376 #endif

378 #else /* The non-microsoft world */

380 #   ifdef OPENSSL_SYS_VMS
381 #     define VMS 1
382     /* some programs don't include stdlib, so exit() and others give implicit
383        function warnings */
384 #     include <stdlib.h>
385 #     if defined(__DECC)
386 #       include <unistd.h>
387 #     else
388 #       include <unixlib.h>
389 #     endif
390 #     define OPENSSL_CONF        "openssl.cnf"
391 #     define SSLEAY_CONF         OPENSSL_CONF
```

```
#    define RFILE              ".rnd"
#    define LIST_SEPARATOR_CHAR ','
#    define NUL_DEV            "NLA0:"
  /* We don't have any well-defined random devices on VMS, yet... */
#    undef DEVRANDOM
  /* We need to do this since VMS has the following coding on status codes:

     Bits 0-2: status type: 0 = warning, 1 = success, 2 = error, 3 = info ...
               The important thing to know is that odd numbers are considered
               good, while even ones are considered errors.
     Bits 3-15: actual status number
     Bits 16-27: facility number.  0 is considered "unknown"
     Bits 28-31: control bits.  If bit 28 is set, the shell won't try to
                 output the message (which, for random codes, just looks ugly)

     So, what we do here is to change 0 to 1 to get the default success status,
     and everything else is shifted up to fit into the status number field, and
     the status is tagged as an error, which I believe is what is wanted here.
     -- Richard Levitte
  */
#    define EXIT(n)            do { int __VMS_EXIT = n; \
                                    if (__VMS_EXIT == 0) \
                                      __VMS_EXIT = 1; \
                                    else \
                                      __VMS_EXIT = (n << 3) | 2; \
                                    __VMS_EXIT |= 0x10000000; \
                                    exit(__VMS_EXIT); } while(0)
#    define NO_SYS_PARAM_H

#  elif defined(OPENSSL_SYS_NETWARE)
#    include <fcntl.h>
#    include <unistd.h>
#    define NO_SYS_TYPES_H
#    undef  DEVRANDOM
#    ifdef NETWARE_CLIB
#      define getpid GetThreadID
#      extern int GetThreadID(void);
/* #      include <conio.h> */
#      extern int kbhit(void);
#    else
#      include <screen.h>
#    endif
#    define NO_SYSLOG
#    define _setmode setmode
#    define _kbhit kbhit
#    define _O_TEXT O_TEXT
#    define _O_BINARY O_BINARY
#    define OPENSSL_CONF   "openssl.cnf"
#    define SSLEAY_CONF    OPENSSL_CONF
#    define RFILE    ".rnd"
#    define LIST_SEPARATOR_CHAR ';'
#    define EXIT(n)  { if (n) printf("ERROR: %d\n", (int)n); exit(n); }

#  else
     /* !defined VMS */
#    ifdef OPENSSL_SYS_MPE
#      define NO_SYS_PARAM_H
#    endif
#    ifdef OPENSSL_UNISTD
#      include OPENSSL_UNISTD
#    else
#      include <unistd.h>
#    endif
#    ifndef NO_SYS_TYPES_H
#      include <sys/types.h>
#    endif
```



```
#    if defined(NeXT) || defined(OPENSSL_SYS_NEWS4)
#      define pid_t int /* pid_t is missing on NEXTSTEP/OPENSTEP
                         * (unless when compiling with -D_POSIX_SOURCE,
                         * which doesn't work for us) */
#    endif
#    ifdef OPENSSL_SYS_NEWS4 /* setvbuf is missing on mips-sony-bsd */
#      define setvbuf(a, b, c, d) setbuffer((a), (b), (d))
       typedef unsigned long clock_t;
#    endif
#    ifdef OPENSSL_SYS_WIN32_CYGWIN
#      include <io.h>
#      include <fcntl.h>
#    endif

#    define OPENSSL_CONF        "openssl.cnf"
#    define SSLEAY_CONF         OPENSSL_CONF
#    define RFILE               ".rnd"
#    define LIST_SEPARATOR_CHAR ':'
#    define NUL_DEV             "/dev/null"
#    define EXIT(n)             exit(n)
#  endif

#  define SSLeay_getpid()       getpid()

#endif


/*************/

#ifdef USE_SOCKETS
#  if defined(WINDOWS) || defined(MSDOS)
      /* windows world */

#    ifdef OPENSSL_NO_SOCK
#      define SSLeay_Write(a,b,c)       (-1)
#      define SSLeay_Read(a,b,c)        (-1)
#      define SHUTDOWN(fd)              close(fd)
#      define SHUTDOWN2(fd)             close(fd)
#    elif !defined(__DJGPP__)
#      if defined(_WIN32_WCE) && _WIN32_WCE<410
#        define getservbyname _masked_declaration_getservbyname
#      endif
#      if !defined(IPPROTO_IP)
         /* winsock[2].h was included already? */
#        include <winsock.h>
#      endif
#      ifdef getservbyname
#        undef getservbyname
         /* this is used to be wcecompat/include/winsock_extras.h */
         struct servent* PASCAL getservbyname(const char*,const char*);
#      endif

#      ifdef _WIN64
/*
 * Even though sizeof(SOCKET) is 8, it's safe to cast it to int, because
 * the value constitutes an index in per-process table of limited size
 * and not a real pointer.
 */
#        define socket(d,t,p)   ((int)socket(d,t,p))
#        define accept(s,f,l)   ((int)accept(s,f,l))
#      endif
#      define SSLeay_Write(a,b,c)       send((a),(b),(c),0)
#      define SSLeay_Read(a,b,c)        recv((a),(b),(c),0)
#      define SHUTDOWN(fd)              { shutdown((fd),0); closesocket(fd); }
#      define SHUTDOWN2(fd)             { shutdown((fd),2); closesocket(fd); }
#    else
```

```
#      define SSLeay_Write(a,b,c)       write_s(a,b,c,0)
#      define SSLeay_Read(a,b,c)        read_s(a,b,c)
#      define SHUTDOWN(fd)              close_s(fd)
#      define SHUTDOWN2(fd)             close_s(fd)
#    endif

#  elif defined(MAC_OS_pre_X)

#    include "MacSocket.h"
#    define SSLeay_Write(a,b,c)         MacSocket_send((a),(b),(c))
#    define SSLeay_Read(a,b,c)          MacSocket_recv((a),(b),(c),true)
#    define SHUTDOWN(fd)                MacSocket_close(fd)
#    define SHUTDOWN2(fd)               MacSocket_close(fd)

#  elif defined(OPENSSL_SYS_NETWARE)
         /* NetWare uses the WinSock2 interfaces by default, but can be configur
         */
#      if defined(NETWARE_BSDSOCK)
#        include <sys/socket.h>
#        include <netinet/in.h>
#        include <sys/time.h>
#        if defined(NETWARE_CLIB)
#          include <sys/bsdskt.h>
#        else
#          include <sys/select.h>
#        endif
#        define INVALID_SOCKET (int)(~0)
#      else
#        include <novsock2.h>
#      endif
#      define SSLeay_Write(a,b,c)   send((a),(b),(c),0)
#      define SSLeay_Read(a,b,c) recv((a),(b),(c),0)
#      define SHUTDOWN(fd)    { shutdown((fd),0); closesocket(fd); }
#      define SHUTDOWN2(fd)      { shutdown((fd),2); closesocket(fd); }

#  else

#    ifndef NO_SYS_PARAM_H
#      include <sys/param.h>
#    endif
#    ifdef OPENSSL_SYS_VXWORKS
#      include <time.h>
#    elif !defined(OPENSSL_SYS_MPE)
#      include <sys/time.h> /* Needed under linux for FD_XXX */
#    endif

#    include <netdb.h>
#    if defined(OPENSSL_SYS_VMS_NODECC)
#      include <socket.h>
#      include <in.h>
#      include <inet.h>
#    else
#      include <sys/socket.h>
#      ifdef FILIO_H
#        include <sys/filio.h> /* Added for FIONBIO under unixware */
#      endif
#      include <netinet/in.h>
#      if !defined(OPENSSL_SYS_BEOS_R5)
#      include <arpa/inet.h>
#    endif
#    endif

#    if defined(NeXT) || defined(_NEXT_SOURCE)
#      include <sys/fcntl.h>
#      include <sys/types.h>
#    endif
```



```
#    ifdef OPENSSL_SYS_AIX
#      include <sys/select.h>
#    endif

#    ifdef __QNX__
#      include <sys/select.h>
#    endif

#    if defined(sun)
#      include <sys/filio.h>
#    else
#      ifndef VMS
#        include <sys/ioctl.h>
#      else
         /* ioctl is only in VMS > 7.0 and when socketshr is not used */
#        if !defined(TCPIP_TYPE_SOCKETSHR) && defined(__VMS_VER) && (__VMS_VER >
#          include <sys/ioctl.h>
#        endif
#      endif
#    endif

#    ifdef VMS
#      include <unixio.h>
#      if defined(TCPIP_TYPE_SOCKETSHR)
#        include <socketshr.h>
#      endif
#    endif

#    define SSLeay_Read(a,b,c)     read((a),(b),(c))
#    define SSLeay_Write(a,b,c)    write((a),(b),(c))
#    define SHUTDOWN(fd)    { shutdown((fd),0); closesocket((fd)); }
#    define SHUTDOWN2(fd)   { shutdown((fd),2); closesocket((fd)); }
#    ifndef INVALID_SOCKET
#    define INVALID_SOCKET      (-1)
#    endif /* INVALID_SOCKET */
#  endif

/* Some IPv6 implementations are broken, disable them in known bad
 * versions.
 */
#  if !defined(OPENSSL_USE_IPV6)
#    if defined(AF_INET6) && !defined(OPENSSL_SYS_BEOS_BONE) && !defined(NETWARE
#      define OPENSSL_USE_IPV6 1
#    else
#      define OPENSSL_USE_IPV6 0
#    endif
#  endif

#endif

#if defined(sun) && !defined(__svr4__) && !defined(__SVR4)
  /* include headers first, so our defines don't break it */
#include <stdlib.h>
#include <string.h>
  /* bcopy can handle overlapping moves according to SunOS 4.1.4 manpage */
# define memmove(s1,s2,n) bcopy((s2),(s1),(n))
# define strtoul(s,e,b) ((unsigned long int)strtol((s),(e),(b)))
extern char *sys_errlist[]; extern int sys_nerr;
# define strerror(errnum) \
        (((errnum)<0 || (errnum)>=sys_nerr) ? NULL : sys_errlist[errnum])
  /* Being signed SunOS 4.x memcpy breaks ASN1_OBJECT table lookup */
#include "crypto/o_str.h"
# define memcmp OPENSSL_memcmp
#endif
```

```
656 #ifndef OPENSSL_EXIT
657 # if defined(MONOLITH) && !defined(OPENSSL_C)
658 #  define OPENSSL_EXIT(n) return(n)
659 # else
660 #  define OPENSSL_EXIT(n) do { EXIT(n); return(n); } while(0)
661 # endif
662 #endif

664 /********************************************/

666 #define DG_GCC_BUG      /* gcc < 2.6.3 on DGUX */

668 #ifdef sgi
669 #define IRIX_CC_BUG     /* all version of IRIX I've tested (4.* 5.*) */
670 #endif
671 #ifdef OPENSSL_SYS_SNI
672 #define IRIX_CC_BUG     /* CDS++ up to V2.0Bsomething suffered from the same bug
673 #endif

675 #if defined(OPENSSL_SYS_WINDOWS)
676 #  define strcasecmp _stricmp
677 #  define strncasecmp _strnicmp
678 #elif defined(OPENSSL_SYS_VMS)
679 /* VMS below version 7.0 doesn't have strcasecmp() */
680 #  include "o_str.h"
681 #  define strcasecmp OPENSSL_strcasecmp
682 #  define strncasecmp OPENSSL_strncasecmp
683 #  define OPENSSL_IMPLEMENTS_strncasecmp
684 #elif defined(OPENSSL_SYS_OS2) && defined(__EMX__)
685 #  define strcasecmp stricmp
686 #  define strncasecmp strnicmp
687 #elif defined(OPENSSL_SYS_NETWARE)
688 #  include <string.h>
689 #  if defined(NETWARE_CLIB)
690 #    define strcasecmp stricmp
691 #    define strncasecmp strnicmp
692 #  endif /* NETWARE_CLIB */
693 #endif

695 #if defined(OPENSSL_SYS_OS2) && defined(__EMX__)
696 # include <io.h>
697 # include <fcntl.h>
698 # define NO_SYSLOG
699 #endif

701 /* vxworks */
702 #if defined(OPENSSL_SYS_VXWORKS)
703 #include <ioLib.h>
704 #include <tickLib.h>
705 #include <sysLib.h>

707 #define TTY_STRUCT int

709 #define sleep(a) taskDelay((a) * sysClkRateGet())

711 #include <vxWorks.h>
712 #include <sockLib.h>
713 #include <taskLib.h>

715 #define getpid taskIdSelf

717 /* NOTE: these are implemented by helpers in database app!
718  * if the database is not linked, we need to implement them
719  * elswhere */
720 struct hostent *gethostbyname(const char *name);
721 struct hostent *gethostbyaddr(const char *addr, int length, int type);
```



```
722 struct servent *getservbyname(const char *name, const char *proto);

724 #endif
725 /* end vxworks */

727 /* beos */
728 #if defined(OPENSSL_SYS_BEOS_R5)
729 #define SO_ERROR 0
730 #define NO_SYS_UN
731 #define IPPROTO_IP 0
732 #include <OS.h>
733 #endif


736 #ifdef  __cplusplus
737 }
738 #endif

740 #endif
741 #endif /* ! codereview */
```

```
**********************************************************
   21909 Wed Aug 13 19:51:37 2014
new/usr/src/lib/openssl/include/ec_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ec/ec_lcl.h */
/*
 * Originally written by Bodo Moeller for the OpenSSL project.
 */
/* ====================================================================
 * Copyright (c) 1998-2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * Portions of the attached software ("Contribution") are developed by
```



```
 * SUN MICROSYSTEMS, INC., and are contributed to the OpenSSL project.
 *
 * The Contribution is licensed pursuant to the OpenSSL open source
 * license provided above.
 *
 * The elliptic curve binary polynomial software is originally written by
 * Sheueling Chang Shantz and Douglas Stebila of Sun Microsystems Laboratories.
 *
 */


#include <stdlib.h>

#include <openssl/obj_mac.h>
#include <openssl/ec.h>
#include <openssl/bn.h>

#if defined(__SUNPRO_C)
# if __SUNPRO_C >= 0x520
# pragma error_messages (off,E_ARRAY_OF_INCOMPLETE_NONAME,E_ARRAY_OF_INCOMPLETE)
# endif
#endif

/* Use default functions for poin2oct, oct2point and compressed coordinates */
#define EC_FLAGS_DEFAULT_OCT    0x1

/* Structure details are not part of the exported interface,
 * so all this may change in future versions. */

struct ec_method_st {
        /* Various method flags */
        int flags;
        /* used by EC_METHOD_get_field_type: */
        int field_type; /* a NID */

        /* used by EC_GROUP_new, EC_GROUP_free, EC_GROUP_clear_free, EC_GROUP_co
        int (*group_init)(EC_GROUP *);
        void (*group_finish)(EC_GROUP *);
        void (*group_clear_finish)(EC_GROUP *);
        int (*group_copy)(EC_GROUP *, const EC_GROUP *);

        /* used by EC_GROUP_set_curve_GFp, EC_GROUP_get_curve_GFp, */
        /* EC_GROUP_set_curve_GF2m, and EC_GROUP_get_curve_GF2m: */
        int (*group_set_curve)(EC_GROUP *, const BIGNUM *p, const BIGNUM *a, con
        int (*group_get_curve)(const EC_GROUP *, BIGNUM *p, BIGNUM *a, BIGNUM *b

        /* used by EC_GROUP_get_degree: */
        int (*group_get_degree)(const EC_GROUP *);

        /* used by EC_GROUP_check: */
        int (*group_check_discriminant)(const EC_GROUP *, BN_CTX *);

        /* used by EC_POINT_new, EC_POINT_free, EC_POINT_clear_free, EC_POINT_co
        int (*point_init)(EC_POINT *);
        void (*point_finish)(EC_POINT *);
        void (*point_clear_finish)(EC_POINT *);
        int (*point_copy)(EC_POINT *, const EC_POINT *);

        /* used by EC_POINT_set_to_infinity,
         * EC_POINT_set_Jprojective_coordinates_GFp,
         * EC_POINT_get_Jprojective_coordinates_GFp,
         * EC_POINT_set_affine_coordinates_GFp,     ..._GF2m,
         * EC_POINT_get_affine_coordinates_GFp,     ..._GF2m,
         * EC_POINT_set_compressed_coordinates_GFp, ..._GF2m:
         */
        int (*point_set_to_infinity)(const EC_GROUP *, EC_POINT *);
```

```
128         int (*point_set_Jprojective_coordinates_GFp)(const EC_GROUP *, EC_POINT
129                 const BIGNUM *x, const BIGNUM *y, const BIGNUM *z, BN_CTX *);
130         int (*point_get_Jprojective_coordinates_GFp)(const EC_GROUP *, const EC_
131                 BIGNUM *x, BIGNUM *y, BIGNUM *z, BN_CTX *);
132         int (*point_set_affine_coordinates)(const EC_GROUP *, EC_POINT *,
133                 const BIGNUM *x, const BIGNUM *y, BN_CTX *);
134         int (*point_get_affine_coordinates)(const EC_GROUP *, const EC_POINT *,
135                 BIGNUM *x, BIGNUM *y, BN_CTX *);
136         int (*point_set_compressed_coordinates)(const EC_GROUP *, EC_POINT *,
137                 const BIGNUM *x, int y_bit, BN_CTX *);

139         /* used by EC_POINT_point2oct, EC_POINT_oct2point: */
140         size_t (*point2oct)(const EC_GROUP *, const EC_POINT *, point_conversion
141                 unsigned char *buf, size_t len, BN_CTX *);
142         int (*oct2point)(const EC_GROUP *, EC_POINT *,
143                 const unsigned char *buf, size_t len, BN_CTX *);

145         /* used by EC_POINT_add, EC_POINT_dbl, ECP_POINT_invert: */
146         int (*add)(const EC_GROUP *, EC_POINT *r, const EC_POINT *a, const EC_PO
147         int (*dbl)(const EC_GROUP *, EC_POINT *r, const EC_POINT *a, BN_CTX *);
148         int (*invert)(const EC_GROUP *, EC_POINT *, BN_CTX *);

150         /* used by EC_POINT_is_at_infinity, EC_POINT_is_on_curve, EC_POINT_cmp:
151         int (*is_at_infinity)(const EC_GROUP *, const EC_POINT *);
152         int (*is_on_curve)(const EC_GROUP *, const EC_POINT *, BN_CTX *);
153         int (*point_cmp)(const EC_GROUP *, const EC_POINT *a, const EC_POINT *b,

155         /* used by EC_POINT_make_affine, EC_POINTs_make_affine: */
156         int (*make_affine)(const EC_GROUP *, EC_POINT *, BN_CTX *);
157         int (*points_make_affine)(const EC_GROUP *, size_t num, EC_POINT *[], BN

159         /* used by EC_POINTs_mul, EC_POINT_mul, EC_POINT_precompute_mult, EC_POI
160          * (default implementations are used if the 'mul' pointer is 0): */
161         int (*mul)(const EC_GROUP *group, EC_POINT *r, const BIGNUM *scalar,
162                 size_t num, const EC_POINT *points[], const BIGNUM *scalars[], B
163         int (*precompute_mult)(EC_GROUP *group, BN_CTX *);
164         int (*have_precompute_mult)(const EC_GROUP *group);


167         /* internal functions */

169         /* 'field_mul', 'field_sqr', and 'field_div' can be used by 'add' and 'd
170          * the same implementations of point operations can be used with differe
171          * optimized implementations of expensive field operations: */
172         int (*field_mul)(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, const BIG
173         int (*field_sqr)(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CTX *)
174         int (*field_div)(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, const BIG

176         int (*field_encode)(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CTX
177         int (*field_decode)(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CTX
178         int (*field_set_to_one)(const EC_GROUP *, BIGNUM *r, BN_CTX *);
179 } /* EC_METHOD */;

181 typedef struct ec_extra_data_st {
182         struct ec_extra_data_st *next;
183         void *data;
184         void *(*dup_func)(void *);
185         void (*free_func)(void *);
186         void (*clear_free_func)(void *);
187 } EC_EXTRA_DATA; /* used in EC_GROUP */

189 struct ec_group_st {
190         const EC_METHOD *meth;

192         EC_POINT *generator; /* optional */
193         BIGNUM order, cofactor;
```



```
195         int curve_name;/* optional NID for named curve */
196         int asn1_flag; /* flag to control the asn1 encoding */
197         point_conversion_form_t asn1_form;

199         unsigned char *seed; /* optional seed for parameters (appears in ASN1) *
200         size_t seed_len;

202         EC_EXTRA_DATA *extra_data; /* linked list */

204         /* The following members are handled by the method functions,
205          * even if they appear generic */

207         BIGNUM field; /* Field specification.
208                        * For curves over GF(p), this is the modulus;
209                        * for curves over GF(2^m), this is the
210                        * irreducible polynomial defining the field.
211                        */

213         int poly[6]; /* Field specification for curves over GF(2^m).
214                       * The irreducible f(t) is then of the form:
215                       *     t^poly[0] + t^poly[1] + ... + t^poly[k]
216                       * where m = poly[0] > poly[1] > ... > poly[k] = 0.
217                       * The array is terminated with poly[k+1]=-1.
218                       * All elliptic curve irreducibles have at most 5
219                       * non-zero terms.
220                       */

222         BIGNUM a, b; /* Curve coefficients.
223                       * (Here the assumption is that BIGNUMs can be used
224                       * or abused for all kinds of fields, not just GF(p).)
225                       * For characteristic  > 3,  the curve is defined
226                       * by a Weierstrass equation of the form
227                       *     y^2 = x^3 + a*x + b.
228                       * For characteristic  2,  the curve is defined by
229                       * an equation of the form
230                       *     y^2 + x*y = x^3 + a*x^2 + b.
231                       */

233         int a_is_minus3; /* enable optimized point arithmetics for special case

235         void *field_data1; /* method-specific (e.g., Montgomery structure) */
236         void *field_data2; /* method-specific */
237         int (*field_mod_func)(BIGNUM *, const BIGNUM *, const BIGNUM *, BN_CTX *
238 } /* EC_GROUP */;

240 struct ec_key_st {
241         int version;

243         EC_GROUP *group;

245         EC_POINT *pub_key;
246         BIGNUM   *priv_key;

248         unsigned int enc_flag;
249         point_conversion_form_t conv_form;

251         int     references;
252         int     flags;

254         EC_EXTRA_DATA *method_data;
255 } /* EC_KEY */;

257 /* Basically a 'mixin' for extra data, but available for EC_GROUPs/EC_KEYs only
258  * (with visibility limited to 'package' level for now).
259  * We use the function pointers as index for retrieval; this obviates
```

```
 * global ex_data-style index tables.
 */
int EC_EX_DATA_set_data(EC_EXTRA_DATA **, void *data,
        void *(*dup_func)(void *), void (*free_func)(void *), void (*clear_free_
void *EC_EX_DATA_get_data(const EC_EXTRA_DATA *,
        void *(*dup_func)(void *), void (*free_func)(void *), void (*clear_free_
void EC_EX_DATA_free_data(EC_EXTRA_DATA **,
        void *(*dup_func)(void *), void (*free_func)(void *), void (*clear_free_
void EC_EX_DATA_clear_free_data(EC_EXTRA_DATA **,
        void *(*dup_func)(void *), void (*free_func)(void *), void (*clear_free_
void EC_EX_DATA_free_all_data(EC_EXTRA_DATA **);
void EC_EX_DATA_clear_free_all_data(EC_EXTRA_DATA **);



struct ec_point_st {
        const EC_METHOD *meth;

        /* All members except 'meth' are handled by the method functions,
         * even if they appear generic */

        BIGNUM X;
        BIGNUM Y;
        BIGNUM Z; /* Jacobian projective coordinates:
                   * (X, Y, Z)  represents  (X/Z^2, Y/Z^3)  if  Z != 0 */
        int Z_is_one; /* enable optimized point arithmetics for special case */
} /* EC_POINT */;



/* method functions in ec_mult.c
 * (ec_lib.c uses these as defaults if group->method->mul is 0) */
int ec_wNAF_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *scalar,
        size_t num, const EC_POINT *points[], const BIGNUM *scalars[], BN_CTX *)
int ec_wNAF_precompute_mult(EC_GROUP *group, BN_CTX *);
int ec_wNAF_have_precompute_mult(const EC_GROUP *group);


/* method functions in ecp_smpl.c */
int ec_GFp_simple_group_init(EC_GROUP *);
void ec_GFp_simple_group_finish(EC_GROUP *);
void ec_GFp_simple_group_clear_finish(EC_GROUP *);
int ec_GFp_simple_group_copy(EC_GROUP *, const EC_GROUP *);
int ec_GFp_simple_group_set_curve(EC_GROUP *, const BIGNUM *p, const BIGNUM *a,
int ec_GFp_simple_group_get_curve(const EC_GROUP *, BIGNUM *p, BIGNUM *a, BIGNUM
int ec_GFp_simple_group_get_degree(const EC_GROUP *);
int ec_GFp_simple_group_check_discriminant(const EC_GROUP *, BN_CTX *);
int ec_GFp_simple_point_init(EC_POINT *);
void ec_GFp_simple_point_finish(EC_POINT *);
void ec_GFp_simple_point_clear_finish(EC_POINT *);
int ec_GFp_simple_point_copy(EC_POINT *, const EC_POINT *);
int ec_GFp_simple_point_set_to_infinity(const EC_GROUP *, EC_POINT *);
int ec_GFp_simple_set_Jprojective_coordinates_GFp(const EC_GROUP *, EC_POINT *,
        const BIGNUM *x, const BIGNUM *y, const BIGNUM *z, BN_CTX *);
int ec_GFp_simple_get_Jprojective_coordinates_GFp(const EC_GROUP *, const EC_POI
        BIGNUM *x, BIGNUM *y, BIGNUM *z, BN_CTX *);
int ec_GFp_simple_point_set_affine_coordinates(const EC_GROUP *, EC_POINT *,
        const BIGNUM *x, const BIGNUM *y, BN_CTX *);
int ec_GFp_simple_point_get_affine_coordinates(const EC_GROUP *, const EC_POINT
        BIGNUM *x, BIGNUM *y, BN_CTX *);
int ec_GFp_simple_set_compressed_coordinates(const EC_GROUP *, EC_POINT *,
        const BIGNUM *x, int y_bit, BN_CTX *);
size_t ec_GFp_simple_point2oct(const EC_GROUP *, const EC_POINT *, point_convers
        unsigned char *buf, size_t len, BN_CTX *);
int ec_GFp_simple_oct2point(const EC_GROUP *, EC_POINT *,
        const unsigned char *buf, size_t len, BN_CTX *);
```



```
int ec_GFp_simple_add(const EC_GROUP *, EC_POINT *r, const EC_POINT *a, const EC
int ec_GFp_simple_dbl(const EC_GROUP *, EC_POINT *r, const EC_POINT *a, BN_CTX *
int ec_GFp_simple_invert(const EC_GROUP *, EC_POINT *, BN_CTX *);
int ec_GFp_simple_is_at_infinity(const EC_GROUP *, const EC_POINT *);
int ec_GFp_simple_is_on_curve(const EC_GROUP *, const EC_POINT *, BN_CTX *);
int ec_GFp_simple_cmp(const EC_GROUP *, const EC_POINT *a, const EC_POINT *b, BN
int ec_GFp_simple_make_affine(const EC_GROUP *, EC_POINT *, BN_CTX *);
int ec_GFp_simple_points_make_affine(const EC_GROUP *, size_t num, EC_POINT *[],
int ec_GFp_simple_field_mul(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, const
int ec_GFp_simple_field_sqr(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CTX


/* method functions in ecp_mont.c */
int ec_GFp_mont_group_init(EC_GROUP *);
int ec_GFp_mont_group_set_curve(EC_GROUP *, const BIGNUM *p, const BIGNUM *a, co
void ec_GFp_mont_group_finish(EC_GROUP *);
void ec_GFp_mont_group_clear_finish(EC_GROUP *);
int ec_GFp_mont_group_copy(EC_GROUP *, const EC_GROUP *);
int ec_GFp_mont_field_mul(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, const BI
int ec_GFp_mont_field_sqr(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CTX *
int ec_GFp_mont_field_encode(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CT
int ec_GFp_mont_field_decode(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CT
int ec_GFp_mont_field_set_to_one(const EC_GROUP *, BIGNUM *r, BN_CTX *);


/* method functions in ecp_nist.c */
int ec_GFp_nist_group_copy(EC_GROUP *dest, const EC_GROUP *src);
int ec_GFp_nist_group_set_curve(EC_GROUP *, const BIGNUM *p, const BIGNUM *a, co
int ec_GFp_nist_field_mul(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, const BI
int ec_GFp_nist_field_sqr(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CTX *


/* method functions in ec2_smpl.c */
int ec_GF2m_simple_group_init(EC_GROUP *);
void ec_GF2m_simple_group_finish(EC_GROUP *);
void ec_GF2m_simple_group_clear_finish(EC_GROUP *);
int ec_GF2m_simple_group_copy(EC_GROUP *, const EC_GROUP *);
int ec_GF2m_simple_group_set_curve(EC_GROUP *, const BIGNUM *p, const BIGNUM *a,
int ec_GF2m_simple_group_get_curve(const EC_GROUP *, BIGNUM *p, BIGNUM *a, BIGNU
int ec_GF2m_simple_group_get_degree(const EC_GROUP *);
int ec_GF2m_simple_group_check_discriminant(const EC_GROUP *, BN_CTX *);
int ec_GF2m_simple_point_init(EC_POINT *);
void ec_GF2m_simple_point_finish(EC_POINT *);
void ec_GF2m_simple_point_clear_finish(EC_POINT *);
int ec_GF2m_simple_point_copy(EC_POINT *, const EC_POINT *);
int ec_GF2m_simple_point_set_to_infinity(const EC_GROUP *, EC_POINT *);
int ec_GF2m_simple_point_set_affine_coordinates(const EC_GROUP *, EC_POINT *,
        const BIGNUM *x, const BIGNUM *y, BN_CTX *);
int ec_GF2m_simple_point_get_affine_coordinates(const EC_GROUP *, const EC_POINT
        BIGNUM *x, BIGNUM *y, BN_CTX *);
int ec_GF2m_simple_set_compressed_coordinates(const EC_GROUP *, EC_POINT *,
        const BIGNUM *x, int y_bit, BN_CTX *);
size_t ec_GF2m_simple_point2oct(const EC_GROUP *, const EC_POINT *, point_conver
        unsigned char *buf, size_t len, BN_CTX *);
int ec_GF2m_simple_oct2point(const EC_GROUP *, EC_POINT *,
        const unsigned char *buf, size_t len, BN_CTX *);
int ec_GF2m_simple_add(const EC_GROUP *, EC_POINT *r, const EC_POINT *a, const E
int ec_GF2m_simple_dbl(const EC_GROUP *, EC_POINT *r, const EC_POINT *a, BN_CTX
int ec_GF2m_simple_invert(const EC_GROUP *, EC_POINT *, BN_CTX *);
int ec_GF2m_simple_is_at_infinity(const EC_GROUP *, const EC_POINT *);
int ec_GF2m_simple_is_on_curve(const EC_GROUP *, const EC_POINT *, BN_CTX *);
int ec_GF2m_simple_cmp(const EC_GROUP *, const EC_POINT *a, const EC_POINT *b, B
int ec_GF2m_simple_make_affine(const EC_GROUP *, EC_POINT *, BN_CTX *);
int ec_GF2m_simple_points_make_affine(const EC_GROUP *, size_t num, EC_POINT *[]
int ec_GF2m_simple_field_mul(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, const
int ec_GF2m_simple_field_sqr(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, BN_CT
```

```
392 int ec_GF2m_simple_field_div(const EC_GROUP *, BIGNUM *r, const BIGNUM *a, const


395 /* method functions in ec2_mult.c */
396 int ec_GF2m_simple_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *scalar,
397         size_t num, const EC_POINT *points[], const BIGNUM *scalars[], BN_CTX *)
398 int ec_GF2m_precompute_mult(EC_GROUP *group, BN_CTX *ctx);
399 int ec_GF2m_have_precompute_mult(const EC_GROUP *group);

401 /* method functions in ec2_mult.c */
402 int ec_GF2m_simple_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *scalar,
403         size_t num, const EC_POINT *points[], const BIGNUM *scalars[], BN_CTX *)
404 int ec_GF2m_precompute_mult(EC_GROUP *group, BN_CTX *ctx);
405 int ec_GF2m_have_precompute_mult(const EC_GROUP *group);

407 #ifndef OPENSSL_NO_EC_NISTP_64_GCC_128
408 /* method functions in ecp_nistp224.c */
409 int ec_GFp_nistp224_group_init(EC_GROUP *group);
410 int ec_GFp_nistp224_group_set_curve(EC_GROUP *group, const BIGNUM *p, const BIGN
411 int ec_GFp_nistp224_point_get_affine_coordinates(const EC_GROUP *group, const EC
412 int ec_GFp_nistp224_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *scalar
413 int ec_GFp_nistp224_points_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM
414 int ec_GFp_nistp224_precompute_mult(EC_GROUP *group, BN_CTX *ctx);
415 int ec_GFp_nistp224_have_precompute_mult(const EC_GROUP *group);

417 /* method functions in ecp_nistp256.c */
418 int ec_GFp_nistp256_group_init(EC_GROUP *group);
419 int ec_GFp_nistp256_group_set_curve(EC_GROUP *group, const BIGNUM *p, const BIGN
420 int ec_GFp_nistp256_point_get_affine_coordinates(const EC_GROUP *group, const EC
421 int ec_GFp_nistp256_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *scalar
422 int ec_GFp_nistp256_points_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM
423 int ec_GFp_nistp256_precompute_mult(EC_GROUP *group, BN_CTX *ctx);
424 int ec_GFp_nistp256_have_precompute_mult(const EC_GROUP *group);

426 /* method functions in ecp_nistp521.c */
427 int ec_GFp_nistp521_group_init(EC_GROUP *group);
428 int ec_GFp_nistp521_group_set_curve(EC_GROUP *group, const BIGNUM *p, const BIGN
429 int ec_GFp_nistp521_point_get_affine_coordinates(const EC_GROUP *group, const EC
430 int ec_GFp_nistp521_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM *scalar
431 int ec_GFp_nistp521_points_mul(const EC_GROUP *group, EC_POINT *r, const BIGNUM
432 int ec_GFp_nistp521_precompute_mult(EC_GROUP *group, BN_CTX *ctx);
433 int ec_GFp_nistp521_have_precompute_mult(const EC_GROUP *group);

435 /* utility functions in ecp_nistputil.c */
436 void ec_GFp_nistp_points_make_affine_internal(size_t num, void *point_array,
437         size_t felem_size, void *tmp_felems,
438         void (*felem_one)(void *out),
439         int (*felem_is_zero)(const void *in),
440         void (*felem_assign)(void *out, const void *in),
441         void (*felem_square)(void *out, const void *in),
442         void (*felem_mul)(void *out, const void *in1, const void *in2),
443         void (*felem_inv)(void *out, const void *in),
444         void (*felem_contract)(void *out, const void *in));
445 void ec_GFp_nistp_recode_scalar_bits(unsigned char *sign, unsigned char *digit,
446 #endif
447 #endif /* ! codereview */
```

```
**********************************************************
    3698 Wed Aug 13 19:51:37 2014
new/usr/src/lib/openssl/include/ech_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/ecdh/ech_locl.h */
/* ====================================================================
 * Copyright (c) 2000-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_ECH_LOCL_H
#define HEADER_ECH_LOCL_H

#include <openssl/ecdh.h>

#ifdef  __cplusplus
```



```
extern "C" {
#endif

struct ecdh_method
        {
        const char *name;
        int (*compute_key)(void *key, size_t outlen, const EC_POINT *pub_key, EC
                           void *(*KDF)(const void *in, size_t inlen, void *out,
#if 0
        int (*init)(EC_KEY *eckey);
        int (*finish)(EC_KEY *eckey);
#endif
        int flags;
        char *app_data;
        };

/* If this flag is set the ECDH method is FIPS compliant and can be used
 * in FIPS mode. This is set in the validated module method. If an
 * application sets this flag in its own methods it is its responsibility
 * to ensure the result is compliant.
 */

#define ECDH_FLAG_FIPS_METHOD   0x1

typedef struct ecdh_data_st {
        /* EC_KEY_METH_DATA part */
        int (*init)(EC_KEY *);
        /* method specific part */
        ENGINE  *engine;
        int     flags;
        const ECDH_METHOD *meth;
        CRYPTO_EX_DATA ex_data;
} ECDH_DATA;

ECDH_DATA *ecdh_check(EC_KEY *);

#ifdef  __cplusplus
}
#endif

#endif /* HEADER_ECH_LOCL_H */
#endif /* ! codereview */
```

```
**********************************************************
    4179 Wed Aug 13 19:51:37 2014
new/usr/src/lib/openssl/include/ecs_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ecdsa/ecs_locl.h */
/*
 * Written by Nils Larsch for the OpenSSL project
 */
/* ====================================================================
 * Copyright (c) 2000-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_ECS_LOCL_H
#define HEADER_ECS_LOCL_H
```



```
#include <openssl/ecdsa.h>

#ifdef  __cplusplus
extern "C" {
#endif

struct ecdsa_method
        {
        const char *name;
        ECDSA_SIG *(*ecdsa_do_sign)(const unsigned char *dgst, int dgst_len,
                        const BIGNUM *inv, const BIGNUM *rp, EC_KEY *eckey);
        int (*ecdsa_sign_setup)(EC_KEY *eckey, BN_CTX *ctx, BIGNUM **kinv,
                        BIGNUM **r);
        int (*ecdsa_do_verify)(const unsigned char *dgst, int dgst_len,
                        const ECDSA_SIG *sig, EC_KEY *eckey);
#if 0
        int (*init)(EC_KEY *eckey);
        int (*finish)(EC_KEY *eckey);
#endif
        int flags;
        char *app_data;
        };

/* If this flag is set the ECDSA method is FIPS compliant and can be used
 * in FIPS mode. This is set in the validated module method. If an
 * application sets this flag in its own methods it is its responsibility
 * to ensure the result is compliant.
 */

#define ECDSA_FLAG_FIPS_METHOD  0x1

typedef struct ecdsa_data_st {
        /* EC_KEY_METH_DATA part */
        int (*init)(EC_KEY *);
        /* method (ECDSA) specific part */
        ENGINE  *engine;
        int     flags;
        const ECDSA_METHOD *meth;
        CRYPTO_EX_DATA ex_data;
} ECDSA_DATA;

/** ecdsa_check
 * checks whether ECKEY->meth_data is a pointer to a ECDSA_DATA structure
 * and if not it removes the old meth_data and creates a ECDSA_DATA structure.
 * \param  eckey pointer to a EC_KEY object
 * \return pointer to a ECDSA_DATA structure
 */
ECDSA_DATA *ecdsa_check(EC_KEY *eckey);

#ifdef  __cplusplus
}
#endif

#endif /* HEADER_ECS_LOCL_H */
#endif /* ! codereview */
```

```
**********************************************************
    8293 Wed Aug 13 19:51:37 2014
new/usr/src/lib/openssl/include/eng_int.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/engine/eng_int.h */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
```



```
 */

#ifndef HEADER_ENGINE_INT_H
#define HEADER_ENGINE_INT_H

#include "cryptlib.h"
/* Take public definitions from engine.h */
#include <openssl/engine.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* If we compile with this symbol defined, then both reference counts in the
 * ENGINE structure will be monitored with a line of output on stderr for each
 * change. This prints the engine's pointer address (truncated to unsigned int),
 * "struct" or "funct" to indicate the reference type, the before and after
 * reference count, and the file:line-number pair. The "engine_ref_debug"
 * statements must come *after* the change. */
#ifdef ENGINE_REF_COUNT_DEBUG

#define engine_ref_debug(e, isfunct, diff) \
        fprintf(stderr, "engine: %08x %s from %d to %d (%s:%d)\n", \
                (unsigned int)(e), (isfunct ? "funct" : "struct"), \
                ((isfunct) ? ((e)->funct_ref - (diff)) : ((e)->struct_ref - (dif
                ((isfunct) ? (e)->funct_ref : (e)->struct_ref), \
                (__FILE__), (__LINE__));

#else

#define engine_ref_debug(e, isfunct, diff)

#endif

/* Any code that will need cleanup operations should use these functions to
 * register callbacks. ENGINE_cleanup() will call all registered callbacks in
 * order. NB: both the "add" functions assume CRYPTO_LOCK_ENGINE to already be
 * held (in "write" mode). */
typedef void (ENGINE_CLEANUP_CB)(void);
typedef struct st_engine_cleanup_item
        {
        ENGINE_CLEANUP_CB *cb;
        } ENGINE_CLEANUP_ITEM;
DECLARE_STACK_OF(ENGINE_CLEANUP_ITEM)
void engine_cleanup_add_first(ENGINE_CLEANUP_CB *cb);
void engine_cleanup_add_last(ENGINE_CLEANUP_CB *cb);

/* We need stacks of ENGINEs for use in eng_table.c */
DECLARE_STACK_OF(ENGINE)

/* If this symbol is defined then engine_table_select(), the function that is
 * used by RSA, DSA (etc) code to select registered ENGINEs, cache defaults and
 * functional references (etc), will display debugging summaries to stderr. */
/* #define ENGINE_TABLE_DEBUG */

/* This represents an implementation table. Dependent code should instantiate it
 * as a (ENGINE_TABLE *) pointer value set initially to NULL. */
typedef struct st_engine_table ENGINE_TABLE;
int engine_table_register(ENGINE_TABLE **table, ENGINE_CLEANUP_CB *cleanup,
                ENGINE *e, const int *nids, int num_nids, int setdefault);
void engine_table_unregister(ENGINE_TABLE **table, ENGINE *e);
void engine_table_cleanup(ENGINE_TABLE **table);
#ifndef ENGINE_TABLE_DEBUG
ENGINE *engine_table_select(ENGINE_TABLE **table, int nid);
#else
ENGINE *engine_table_select_tmp(ENGINE_TABLE **table, int nid, const char *f, in
```

```
128 #define engine_table_select(t,n) engine_table_select_tmp(t,n,__FILE__,__LINE__)
129 #endif
130 typedef void (engine_table_doall_cb)(int nid, STACK_OF(ENGINE) *sk, ENGINE *def,
131 void engine_table_doall(ENGINE_TABLE *table, engine_table_doall_cb *cb, void *ar

133 /* Internal versions of API functions that have control over locking. These are
134  * used between C files when functionality needs to be shared but the caller may
135  * already be controlling of the CRYPTO_LOCK_ENGINE lock. */
136 int engine_unlocked_init(ENGINE *e);
137 int engine_unlocked_finish(ENGINE *e, int unlock_for_handlers);
138 int engine_free_util(ENGINE *e, int locked);

140 /* This function will reset all "set"able values in an ENGINE to NULL. This
141  * won't touch reference counts or ex_data, but is equivalent to calling all the
142  * ENGINE_set_***() functions with a NULL value. */
143 void engine_set_all_null(ENGINE *e);

145 /* NB: Bitwise OR-able values for the "flags" variable in ENGINE are now exposed
146  * in engine.h. */

148 /* Free up dynamically allocated public key methods associated with ENGINE */

150 void engine_pkey_meths_free(ENGINE *e);
151 void engine_pkey_asn1_meths_free(ENGINE *e);

153 /* This is a structure for storing implementations of various crypto
154  * algorithms and functions. */
155 struct engine_st
156         {
157         const char *id;
158         const char *name;
159         const RSA_METHOD *rsa_meth;
160         const DSA_METHOD *dsa_meth;
161         const DH_METHOD *dh_meth;
162         const ECDH_METHOD *ecdh_meth;
163         const ECDSA_METHOD *ecdsa_meth;
164         const RAND_METHOD *rand_meth;
165         const STORE_METHOD *store_meth;
166         /* Cipher handling is via this callback */
167         ENGINE_CIPHERS_PTR ciphers;
168         /* Digest handling is via this callback */
169         ENGINE_DIGESTS_PTR digests;
170         /* Public key handling via this callback */
171         ENGINE_PKEY_METHS_PTR pkey_meths;
172         /* ASN1 public key handling via this callback */
173         ENGINE_PKEY_ASN1_METHS_PTR pkey_asn1_meths;

175         ENGINE_GEN_INT_FUNC_PTR destroy;

177         ENGINE_GEN_INT_FUNC_PTR init;
178         ENGINE_GEN_INT_FUNC_PTR finish;
179         ENGINE_CTRL_FUNC_PTR ctrl;
180         ENGINE_LOAD_KEY_PTR load_privkey;
181         ENGINE_LOAD_KEY_PTR load_pubkey;

183         ENGINE_SSL_CLIENT_CERT_PTR load_ssl_client_cert;

185         const ENGINE_CMD_DEFN *cmd_defns;
186         int flags;
187         /* reference count on the structure itself */
188         int struct_ref;
189         /* reference count on usability of the engine type. NB: This
190          * controls the loading and initialisation of any functionlity
191          * required by this engine, whereas the previous count is
192          * simply to cope with (de)allocation of this structure. Hence,
193          * running_ref <= struct_ref at all times. */
```



```
194         int funct_ref;
195         /* A place to store per-ENGINE data */
196         CRYPTO_EX_DATA ex_data;
197         /* Used to maintain the linked-list of engines. */
198         struct engine_st *prev;
199         struct engine_st *next;
200         };

202 #ifdef  __cplusplus
203 }
204 #endif

206 #endif /* HEADER_ENGINE_INT_H */
207 #endif /* ! codereview */
```

```
**********************************************************
   13923 Wed Aug 13 19:51:37 2014
new/usr/src/lib/openssl/include/evp_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* evp_locl.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* Macros to code block cipher wrappers */

/* Wrapper functions for each cipher mode */
```



```
#define BLOCK_CIPHER_ecb_loop() \
        size_t i, bl; \
        bl = ctx->cipher->block_size;\
        if(inl < bl) return 1;\
        inl -= bl; \
        for(i=0; i <= inl; i+=bl)

#define BLOCK_CIPHER_func_ecb(cname, cprefix, kstruct, ksched) \
static int cname##_ecb_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out, const uns
{\
        BLOCK_CIPHER_ecb_loop() \
                cprefix##_ecb_encrypt(in + i, out + i, &((kstruct *)ctx->cipher_
        return 1;\
}

#define EVP_MAXCHUNK ((size_t)1<<(sizeof(long)*8-2))

#define BLOCK_CIPHER_func_ofb(cname, cprefix, cbits, kstruct, ksched) \
static int cname##_ofb_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out, const uns
{\
        while(inl>=EVP_MAXCHUNK)\
            {\
            cprefix##_ofb##cbits##_encrypt(in, out, (long)EVP_MAXCHUNK, &((kstru
            inl-=EVP_MAXCHUNK;\
            in +=EVP_MAXCHUNK;\
            out+=EVP_MAXCHUNK;\
            }\
        if (inl)\
            cprefix##_ofb##cbits##_encrypt(in, out, (long)inl, &((kstruct *)ctx-
        return 1;\
}

#define BLOCK_CIPHER_func_cbc(cname, cprefix, kstruct, ksched) \
static int cname##_cbc_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out, const uns
{\
        while(inl>=EVP_MAXCHUNK) \
            {\
            cprefix##_cbc_encrypt(in, out, (long)EVP_MAXCHUNK, &((kstruct *)ctx-
            inl-=EVP_MAXCHUNK;\
            in +=EVP_MAXCHUNK;\
            out+=EVP_MAXCHUNK;\
            }\
        if (inl)\
            cprefix##_cbc_encrypt(in, out, (long)inl, &((kstruct *)ctx->cipher_d
        return 1;\
}

#define BLOCK_CIPHER_func_cfb(cname, cprefix, cbits, kstruct, ksched) \
static int cname##_cfb##cbits##_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
{\
        size_t chunk=EVP_MAXCHUNK;\
        if (cbits==1)  chunk>>=3;\
        if (inl<chunk) chunk=inl;\
        while(inl && inl>=chunk)\
            {\
            cprefix##_cfb##cbits##_encrypt(in, out, (long)((cbits==1) && !(ctx->
            inl-=chunk;\
            in +=chunk;\
            out+=chunk;\
            if(inl<chunk) chunk=inl;\
            }\
        return 1;\
}

#define BLOCK_CIPHER_all_funcs(cname, cprefix, cbits, kstruct, ksched) \
```

```
128         BLOCK_CIPHER_func_cbc(cname, cprefix, kstruct, ksched) \
129         BLOCK_CIPHER_func_cfb(cname, cprefix, cbits, kstruct, ksched) \
130         BLOCK_CIPHER_func_ecb(cname, cprefix, kstruct, ksched) \
131         BLOCK_CIPHER_func_ofb(cname, cprefix, cbits, kstruct, ksched)

133 #define BLOCK_CIPHER_def1(cname, nmode, mode, MODE, kstruct, nid, block_size, \
134                           key_len, iv_len, flags, init_key, cleanup, \
135                           set_asn1, get_asn1, ctrl) \
136 static const EVP_CIPHER cname##_##mode = { \
137         nid##_##nmode, block_size, key_len, iv_len, \
138         flags | EVP_CIPH_##MODE##_MODE, \
139         init_key, \
140         cname##_##mode##_cipher, \
141         cleanup, \
142         sizeof(kstruct), \
143         set_asn1, get_asn1,\
144         ctrl, \
145         NULL \
146 }; \
147 const EVP_CIPHER *EVP_##cname##_##mode(void) { return &cname##_##mode; }

149 #define BLOCK_CIPHER_def_cbc(cname, kstruct, nid, block_size, key_len, \
150                              iv_len, flags, init_key, cleanup, set_asn1, \
151                              get_asn1, ctrl) \
152 BLOCK_CIPHER_def1(cname, cbc, cbc, CBC, kstruct, nid, block_size, key_len, \
153                   iv_len, flags, init_key, cleanup, set_asn1, get_asn1, ctrl)

155 #define BLOCK_CIPHER_def_cfb(cname, kstruct, nid, key_len, \
156                              iv_len, cbits, flags, init_key, cleanup, \
157                              set_asn1, get_asn1, ctrl) \
158 BLOCK_CIPHER_def1(cname, cfb##cbits, cfb##cbits, CFB, kstruct, nid, 1, \
159                   key_len, iv_len, flags, init_key, cleanup, set_asn1, \
160                   get_asn1, ctrl)

162 #define BLOCK_CIPHER_def_ofb(cname, kstruct, nid, key_len, \
163                              iv_len, cbits, flags, init_key, cleanup, \
164                              set_asn1, get_asn1, ctrl) \
165 BLOCK_CIPHER_def1(cname, ofb##cbits, ofb, OFB, kstruct, nid, 1, \
166                   key_len, iv_len, flags, init_key, cleanup, set_asn1, \
167                   get_asn1, ctrl)

169 #define BLOCK_CIPHER_def_ecb(cname, kstruct, nid, block_size, key_len, \
170                              flags, init_key, cleanup, set_asn1, \
171                              get_asn1, ctrl) \
172 BLOCK_CIPHER_def1(cname, ecb, ecb, ECB, kstruct, nid, block_size, key_len, \
173                   0, flags, init_key, cleanup, set_asn1, get_asn1, ctrl)

175 #define BLOCK_CIPHER_defs(cname, kstruct, \
176                           nid, block_size, key_len, iv_len, cbits, flags, \
177                           init_key, cleanup, set_asn1, get_asn1, ctrl) \
178 BLOCK_CIPHER_def_cbc(cname, kstruct, nid, block_size, key_len, iv_len, flags, \
179                      init_key, cleanup, set_asn1, get_asn1, ctrl) \
180 BLOCK_CIPHER_def_cfb(cname, kstruct, nid, key_len, iv_len, cbits, \
181                      flags, init_key, cleanup, set_asn1, get_asn1, ctrl) \
182 BLOCK_CIPHER_def_ofb(cname, kstruct, nid, key_len, iv_len, cbits, \
183                      flags, init_key, cleanup, set_asn1, get_asn1, ctrl) \
184 BLOCK_CIPHER_def_ecb(cname, kstruct, nid, block_size, key_len, flags, \
185                      init_key, cleanup, set_asn1, get_asn1, ctrl)


188 /*
189 #define BLOCK_CIPHER_defs(cname, kstruct, \
190                               nid, block_size, key_len, iv_len, flags,\
191                               init_key, cleanup, set_asn1, get_asn1, ctrl)\
192 static const EVP_CIPHER cname##_cbc = {\
193         nid##_cbc, block_size, key_len, iv_len, \
```



```
194         flags | EVP_CIPH_CBC_MODE,\
195         init_key,\
196         cname##_cbc_cipher,\
197         cleanup,\
198         sizeof(EVP_CIPHER_CTX)-sizeof((((EVP_CIPHER_CTX *)NULL)->c))+\
199                 sizeof((((EVP_CIPHER_CTX *)NULL)->c.kstruct)),\
200         set_asn1, get_asn1,\
201         ctrl, \
202         NULL \
203 };\
204 const EVP_CIPHER *EVP_##cname##_cbc(void) { return &cname##_cbc; }\
205 static const EVP_CIPHER cname##_cfb = {\
206         nid##_cfb64, 1, key_len, iv_len, \
207         flags | EVP_CIPH_CFB_MODE,\
208         init_key,\
209         cname##_cfb_cipher,\
210         cleanup,\
211         sizeof(EVP_CIPHER_CTX)-sizeof((((EVP_CIPHER_CTX *)NULL)->c))+\
212                 sizeof((((EVP_CIPHER_CTX *)NULL)->c.kstruct)),\
213         set_asn1, get_asn1,\
214         ctrl,\
215         NULL \
216 };\
217 const EVP_CIPHER *EVP_##cname##_cfb(void) { return &cname##_cfb; }\
218 static const EVP_CIPHER cname##_ofb = {\
219         nid##_ofb64, 1, key_len, iv_len, \
220         flags | EVP_CIPH_OFB_MODE,\
221         init_key,\
222         cname##_ofb_cipher,\
223         cleanup,\
224         sizeof(EVP_CIPHER_CTX)-sizeof((((EVP_CIPHER_CTX *)NULL)->c))+\
225                 sizeof((((EVP_CIPHER_CTX *)NULL)->c.kstruct)),\
226         set_asn1, get_asn1,\
227         ctrl,\
228         NULL \
229 };\
230 const EVP_CIPHER *EVP_##cname##_ofb(void) { return &cname##_ofb; }\
231 static const EVP_CIPHER cname##_ecb = {\
232         nid##_ecb, block_size, key_len, iv_len, \
233         flags | EVP_CIPH_ECB_MODE,\
234         init_key,\
235         cname##_ecb_cipher,\
236         cleanup,\
237         sizeof(EVP_CIPHER_CTX)-sizeof((((EVP_CIPHER_CTX *)NULL)->c))+\
238                 sizeof((((EVP_CIPHER_CTX *)NULL)->c.kstruct)),\
239         set_asn1, get_asn1,\
240         ctrl,\
241         NULL \
242 };\
243 const EVP_CIPHER *EVP_##cname##_ecb(void) { return &cname##_ecb; }
244 */

246 #define IMPLEMENT_BLOCK_CIPHER(cname, ksched, cprefix, kstruct, nid, \
247                                block_size, key_len, iv_len, cbits, \
248                                flags, init_key, \
249                                cleanup, set_asn1, get_asn1, ctrl) \
250         BLOCK_CIPHER_all_funcs(cname, cprefix, cbits, kstruct, ksched) \
251         BLOCK_CIPHER_defs(cname, kstruct, nid, block_size, key_len, iv_len, \
252                           cbits, flags, init_key, cleanup, set_asn1, \
253                           get_asn1, ctrl)

255 #define EVP_C_DATA(kstruct, ctx)        ((kstruct *)(ctx)->cipher_data)

257 #define IMPLEMENT_CFBR(cipher,cprefix,kstruct,ksched,keysize,cbits,iv_len) \
258         BLOCK_CIPHER_func_cfb(cipher##_##keysize,cprefix,cbits,kstruct,ksched) \
259         BLOCK_CIPHER_def_cfb(cipher##_##keysize,kstruct, \
```

```
260                             NID_##cipher##_##keysize, keysize/8, iv_len, cbits,
261                             0, cipher##_init_key, NULL, \
262                             EVP_CIPHER_set_asn1_iv, \
263                             EVP_CIPHER_get_asn1_iv, \
264                             NULL)

266 struct evp_pkey_ctx_st
267         {
268         /* Method associated with this operation */
269         const EVP_PKEY_METHOD *pmeth;
270         /* Engine that implements this method or NULL if builtin */
271         ENGINE *engine;
272         /* Key: may be NULL */
273         EVP_PKEY *pkey;
274         /* Peer key for key agreement, may be NULL */
275         EVP_PKEY *peerkey;
276         /* Actual operation */
277         int operation;
278         /* Algorithm specific data */
279         void *data;
280         /* Application specific data */
281         void *app_data;
282         /* Keygen callback */
283         EVP_PKEY_gen_cb *pkey_gencb;
284         /* implementation specific keygen data */
285         int *keygen_info;
286         int keygen_info_count;
287         } /* EVP_PKEY_CTX */;

289 #define EVP_PKEY_FLAG_DYNAMIC   1

291 struct evp_pkey_method_st
292         {
293         int pkey_id;
294         int flags;

296         int (*init)(EVP_PKEY_CTX *ctx);
297         int (*copy)(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src);
298         void (*cleanup)(EVP_PKEY_CTX *ctx);

300         int (*paramgen_init)(EVP_PKEY_CTX *ctx);
301         int (*paramgen)(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey);

303         int (*keygen_init)(EVP_PKEY_CTX *ctx);
304         int (*keygen)(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey);

306         int (*sign_init)(EVP_PKEY_CTX *ctx);
307         int (*sign)(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
308                             const unsigned char *tbs, size_t tbslen);

310         int (*verify_init)(EVP_PKEY_CTX *ctx);
311         int (*verify)(EVP_PKEY_CTX *ctx,
312                             const unsigned char *sig, size_t siglen,
313                             const unsigned char *tbs, size_t tbslen);

315         int (*verify_recover_init)(EVP_PKEY_CTX *ctx);
316         int (*verify_recover)(EVP_PKEY_CTX *ctx,
317                             unsigned char *rout, size_t *routlen,
318                             const unsigned char *sig, size_t siglen);

320         int (*signctx_init)(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx);
321         int (*signctx)(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
322                                 EVP_MD_CTX *mctx);

324         int (*verifyctx_init)(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx);
325         int (*verifyctx)(EVP_PKEY_CTX *ctx, const unsigned char *sig,int siglen,
```



```
326                                 EVP_MD_CTX *mctx);

328         int (*encrypt_init)(EVP_PKEY_CTX *ctx);
329         int (*encrypt)(EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,
330                                 const unsigned char *in, size_t inlen);

332         int (*decrypt_init)(EVP_PKEY_CTX *ctx);
333         int (*decrypt)(EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,
334                                 const unsigned char *in, size_t inlen);

336         int (*derive_init)(EVP_PKEY_CTX *ctx);
337         int (*derive)(EVP_PKEY_CTX *ctx, unsigned char *key, size_t *keylen);

339         int (*ctrl)(EVP_PKEY_CTX *ctx, int type, int p1, void *p2);
340         int (*ctrl_str)(EVP_PKEY_CTX *ctx, const char *type, const char *value);


343         } /* EVP_PKEY_METHOD */;

345 void evp_pkey_set_cb_translate(BN_GENCB *cb, EVP_PKEY_CTX *ctx);

347 int PKCS5_v2_PBKDF2_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
348                              ASN1_TYPE *param,
349                              const EVP_CIPHER *c, const EVP_MD *md, int en_de);

351 #ifdef OPENSSL_FIPS

353 #ifdef OPENSSL_DOING_MAKEDEPEND
354 #undef SHA1_Init
355 #undef SHA1_Update
356 #undef SHA224_Init
357 #undef SHA256_Init
358 #undef SHA384_Init
359 #undef SHA512_Init
360 #undef DES_set_key_unchecked
361 #endif

363 #define RIPEMD160_Init  private_RIPEMD160_Init
364 #define WHIRLPOOL_Init  private_WHIRLPOOL_Init
365 #define MD5_Init        private_MD5_Init
366 #define MD4_Init        private_MD4_Init
367 #define MD2_Init        private_MD2_Init
368 #define MDC2_Init       private_MDC2_Init
369 #define SHA_Init        private_SHA_Init
370 #define SHA1_Init       private_SHA1_Init
371 #define SHA224_Init     private_SHA224_Init
372 #define SHA256_Init     private_SHA256_Init
373 #define SHA384_Init     private_SHA384_Init
374 #define SHA512_Init     private_SHA512_Init

376 #define BF_set_key      private_BF_set_key
377 #define CAST_set_key    private_CAST_set_key
378 #define idea_set_encrypt_key    private_idea_set_encrypt_key
379 #define SEED_set_key    private_SEED_set_key
380 #define RC2_set_key     private_RC2_set_key
381 #define RC4_set_key     private_RC4_set_key
382 #define DES_set_key_unchecked   private_DES_set_key_unchecked
383 #define Camellia_set_key        private_Camellia_set_key

385 #endif
386 #endif /* ! codereview */
```

```
**********************************************************
    4606 Wed Aug 13 19:51:37 2014
new/usr/src/lib/openssl/include/ext_dat.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ext_dat.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* This file contains a table of "standard" extensions */

extern X509V3_EXT_METHOD v3_bcons, v3_nscert, v3_key_usage, v3_ext_ku;
extern X509V3_EXT_METHOD v3_pkey_usage_period, v3_sxnet, v3_info, v3_sinfo;
```



```
extern X509V3_EXT_METHOD v3_ns_ia5_list[], v3_alt[], v3_skey_id, v3_akey_id;
extern X509V3_EXT_METHOD v3_crl_num, v3_crl_reason, v3_crl_invdate;
extern X509V3_EXT_METHOD v3_delta_crl, v3_cpols, v3_crld, v3_freshest_crl;
extern X509V3_EXT_METHOD v3_ocsp_nonce, v3_ocsp_accresp, v3_ocsp_acutoff;
extern X509V3_EXT_METHOD v3_ocsp_crlid, v3_ocsp_nocheck, v3_ocsp_serviceloc;
extern X509V3_EXT_METHOD v3_crl_hold, v3_pci;
extern X509V3_EXT_METHOD v3_policy_mappings, v3_policy_constraints;
extern X509V3_EXT_METHOD v3_name_constraints, v3_inhibit_anyp, v3_idp;
extern X509V3_EXT_METHOD v3_addr, v3_asid;

/* This table will be searched using OBJ_bsearch so it *must* kept in
 * order of the ext_nid values.
 */

static const X509V3_EXT_METHOD *standard_exts[] = {
&v3_nscert,
&v3_ns_ia5_list[0],
&v3_ns_ia5_list[1],
&v3_ns_ia5_list[2],
&v3_ns_ia5_list[3],
&v3_ns_ia5_list[4],
&v3_ns_ia5_list[5],
&v3_ns_ia5_list[6],
&v3_skey_id,
&v3_key_usage,
&v3_pkey_usage_period,
&v3_alt[0],
&v3_alt[1],
&v3_bcons,
&v3_crl_num,
&v3_cpols,
&v3_akey_id,
&v3_crld,
&v3_ext_ku,
&v3_delta_crl,
&v3_crl_reason,
#ifndef OPENSSL_NO_OCSP
&v3_crl_invdate,
#endif
&v3_sxnet,
&v3_info,
#ifndef OPENSSL_NO_RFC3779
&v3_addr,
&v3_asid,
#endif
#ifndef OPENSSL_NO_OCSP
&v3_ocsp_nonce,
&v3_ocsp_crlid,
&v3_ocsp_accresp,
&v3_ocsp_nocheck,
&v3_ocsp_acutoff,
&v3_ocsp_serviceloc,
#endif
&v3_sinfo,
&v3_policy_constraints,
#ifndef OPENSSL_NO_OCSP
&v3_crl_hold,
#endif
&v3_pci,
&v3_name_constraints,
&v3_policy_mappings,
&v3_inhibit_anyp,
&v3_idp,
&v3_alt[2],
&v3_freshest_crl,
};
```

```
129 /* Number of standard extensions */

131 #define STANDARD_EXTENSION_COUNT (sizeof(standard_exts)/sizeof(X509V3_EXT_METHOD
132 #endif /* ! codereview */
```

```
**********************************************************
   13663 Wed Aug 13 19:51:38 2014
new/usr/src/lib/openssl/include/hw_pk11_err.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* crypto/engine/hw_pk11_err.h */
/*
 * This product includes software developed by the OpenSSL Project for
 * use in the OpenSSL Toolkit (http://www.openssl.org/).
 *
 * This project also referenced hw_pkcs11-0.9.7b.patch written by
 * Afchine Madjlessi.
 */
/*
 * ====================================================================
 * Copyright (c) 2000-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
```



```
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HW_PK11_ERR_H
#define HW_PK11_ERR_H

void ERR_pk11_error(int function, int reason, char *file, int line);
void PK11err_add_data(int function, int reason, CK_RV rv);
#define PK11err(f, r)   ERR_pk11_error((f), (r), __FILE__, __LINE__)

/* Error codes for the PK11 functions. */

/* Function codes. */

#define PK11_F_INIT                             100
#define PK11_F_FINISH                           101
#define PK11_F_DESTROY                          102
#define PK11_F_CTRL                             103
#define PK11_F_RSA_INIT                         104
#define PK11_F_RSA_FINISH                       105
#define PK11_F_GET_PUB_RSA_KEY                  106
#define PK11_F_GET_PRIV_RSA_KEY                 107
#define PK11_F_RSA_GEN_KEY                      108
#define PK11_F_RSA_PUB_ENC                      109
#define PK11_F_RSA_PRIV_ENC                     110
#define PK11_F_RSA_PUB_DEC                      111
#define PK11_F_RSA_PRIV_DEC                     112
#define PK11_F_RSA_SIGN                         113
#define PK11_F_RSA_VERIFY                       114
#define PK11_F_RAND_ADD                         115
#define PK11_F_RAND_BYTES                       116
#define PK11_F_GET_SESSION                      117
#define PK11_F_FREE_SESSION                     118
#define PK11_F_LOAD_PUBKEY                      119
#define PK11_F_LOAD_PRIVKEY                     120
#define PK11_F_RSA_PUB_ENC_LOW                  121
#define PK11_F_RSA_PRIV_ENC_LOW                 122
#define PK11_F_RSA_PUB_DEC_LOW                  123
#define PK11_F_RSA_PRIV_DEC_LOW                 124
#define PK11_F_DSA_SIGN                         125
#define PK11_F_DSA_VERIFY                       126
#define PK11_F_DSA_INIT                         127
#define PK11_F_DSA_FINISH                       128
#define PK11_F_GET_PUB_DSA_KEY                  129
#define PK11_F_GET_PRIV_DSA_KEY                 130
#define PK11_F_DH_INIT                          131
#define PK11_F_DH_FINISH                        132
#define PK11_F_MOD_EXP_DH                       133
#define PK11_F_GET_DH_KEY                       134
#define PK11_F_FREE_ALL_SESSIONS                135
#define PK11_F_SETUP_SESSION                    136
#define PK11_F_DESTROY_OBJECT                   137
#define PK11_F_CIPHER_INIT                      138
#define PK11_F_CIPHER_DO_CIPHER                 139
#define PK11_F_GET_CIPHER_KEY                   140
#define PK11_F_DIGEST_INIT                      141
#define PK11_F_DIGEST_UPDATE                    142
#define PK11_F_DIGEST_FINAL                     143
#define PK11_F_CHOOSE_SLOT                      144
#define PK11_F_CIPHER_FINAL                     145
#define PK11_F_LIBRARY_INIT                     146
#define PK11_F_LOAD                             147
```

```
#define PK11_F_DH_GEN_KEY                       148
#define PK11_F_DH_COMP_KEY                      149
#define PK11_F_DIGEST_COPY                      150
#define PK11_F_CIPHER_CLEANUP                   151
#define PK11_F_ACTIVE_ADD                       152
#define PK11_F_ACTIVE_DELETE                    153
#define PK11_F_CHECK_HW_MECHANISMS              154
#define PK11_F_INIT_SYMMETRIC                   155
#define PK11_F_ADD_AES_CTR_NIDS                 156
#define PK11_F_INIT_ALL_LOCKS                   157
#define PK11_F_RETURN_SESSION                   158

/* Reason codes. */
#define PK11_R_ALREADY_LOADED                   100
#define PK11_R_DSO_FAILURE                      101
#define PK11_R_NOT_LOADED                       102
#define PK11_R_PASSED_NULL_PARAMETER            103
#define PK11_R_COMMAND_NOT_IMPLEMENTED          104
#define PK11_R_INITIALIZE                       105
#define PK11_R_FINALIZE                         106
#define PK11_R_GETINFO                          107
#define PK11_R_GETSLOTLIST                      108
#define PK11_R_NO_MODULUS_OR_NO_EXPONENT        109
#define PK11_R_ATTRIBUT_SENSITIVE_OR_INVALID    110
#define PK11_R_GETATTRIBUTVALUE                 111
#define PK11_R_NO_MODULUS                       112
#define PK11_R_NO_EXPONENT                      113
#define PK11_R_FINDOBJECTSINIT                  114
#define PK11_R_FINDOBJECTS                      115
#define PK11_R_FINDOBJECTSFINAL                 116
#define PK11_R_CREATEOBJECT                     118
#define PK11_R_DESTROYOBJECT                    119
#define PK11_R_OPENSESSION                      120
#define PK11_R_CLOSESESSION                     121
#define PK11_R_ENCRYPTINIT                      122
#define PK11_R_ENCRYPT                          123
#define PK11_R_SIGNINIT                         124
#define PK11_R_SIGN                             125
#define PK11_R_DECRYPTINIT                      126
#define PK11_R_DECRYPT                          127
#define PK11_R_VERIFYINIT                       128
#define PK11_R_VERIFY                           129
#define PK11_R_VERIFYRECOVERINIT                130
#define PK11_R_VERIFYRECOVER                    131
#define PK11_R_GEN_KEY                          132
#define PK11_R_SEEDRANDOM                       133
#define PK11_R_GENERATERANDOM                   134
#define PK11_R_INVALID_MESSAGE_LENGTH           135
#define PK11_R_UNKNOWN_ALGORITHM_TYPE           136
#define PK11_R_UNKNOWN_ASN1_OBJECT_ID           137
#define PK11_R_UNKNOWN_PADDING_TYPE             138
#define PK11_R_PADDING_CHECK_FAILED             139
#define PK11_R_DIGEST_TOO_BIG                   140
#define PK11_R_MALLOC_FAILURE                   141
#define PK11_R_CTRL_COMMAND_NOT_IMPLEMENTED     142
#define PK11_R_DATA_GREATER_THAN_MOD_LEN        143
#define PK11_R_DATA_TOO_LARGE_FOR_MODULUS       144
#define PK11_R_MISSING_KEY_COMPONENT            145
#define PK11_R_INVALID_SIGNATURE_LENGTH         146
#define PK11_R_INVALID_DSA_SIGNATURE_R          147
#define PK11_R_INVALID_DSA_SIGNATURE_S          148
#define PK11_R_INCONSISTENT_KEY                 149
#define PK11_R_ENCRYPTUPDATE                    150
#define PK11_R_DECRYPTUPDATE                    151
#define PK11_R_DIGESTINIT                       152
#define PK11_R_DIGESTUPDATE                     153
```



```
#define PK11_R_DIGESTFINAL                      154
#define PK11_R_ENCRYPTFINAL                     155
#define PK11_R_DECRYPTFINAL                     156
#define PK11_R_NO_PRNG_SUPPORT                  157
#define PK11_R_GETTOKENINFO                     158
#define PK11_R_DERIVEKEY                        159
#define PK11_R_GET_OPERATION_STATE              160
#define PK11_R_SET_OPERATION_STATE              161
#define PK11_R_INVALID_HANDLE                   162
#define PK11_R_KEY_OR_IV_LEN_PROBLEM            163
#define PK11_R_INVALID_OPERATION_TYPE           164
#define PK11_R_ADD_NID_FAILED                   165
#define PK11_R_ATFORK_FAILED                    166

/* max byte length of a symetric key we support */
#define PK11_KEY_LEN_MAX                        32

/*
 * This structure encapsulates all reusable information for a PKCS#11
 * session. A list of these objects is created on behalf of the
 * calling application using an on-demand method. Each operation
 * type (see PK11_OPTYPE below) has its own per-process list.
 * Each of the lists is basically a cache for faster PKCS#11 object
 * access to avoid expensive C_Find{,Init,Final}Object() calls.
 *
 * When a new request comes in, an object will be taken from the list
 * (if there is one) or a new one is created to handle the request
 * (if the list is empty). See pk11_get_session() on how it is done.
 */
typedef struct PK11_st_SESSION
        {
        struct PK11_st_SESSION  *next;
        CK_SESSION_HANDLE       session;        /* PK11 session handle */
        pid_t                   pid;            /* Current process ID */
        union
                {
#ifndef OPENSSL_NO_RSA
                struct
                        {
                        CK_OBJECT_HANDLE        rsa_pub_key; /* pub handle */
                        CK_OBJECT_HANDLE        rsa_priv_key; /* priv handle */
                        RSA                     *rsa_pub; /* pub key addr */
                        BIGNUM                  *rsa_n_num; /* pub modulus */
                        BIGNUM                  *rsa_e_num; /* pub exponent */
                        RSA                     *rsa_priv; /* priv key addr */
                        BIGNUM                  *rsa_d_num; /* priv exponent */
                        } u_RSA;
#endif /* OPENSSL_NO_RSA */
#ifndef OPENSSL_NO_DSA
                struct
                        {
                        CK_OBJECT_HANDLE        dsa_pub_key; /* pub handle */
                        CK_OBJECT_HANDLE        dsa_priv_key; /* priv handle */
                        DSA                     *dsa_pub; /* pub key addr */
                        BIGNUM                  *dsa_pub_num; /* pub key */
                        DSA                     *dsa_priv; /* priv key addr */
                        BIGNUM                  *dsa_priv_num; /* priv key */
                        } u_DSA;
#endif /* OPENSSL_NO_DSA */
#ifndef OPENSSL_NO_DH
                struct
                        {
                        CK_OBJECT_HANDLE        dh_key; /* key handle */
                        DH                      *dh; /* dh key addr */
                        BIGNUM                  *dh_priv_num; /* priv dh key */
                        } u_DH;
```

```
#endif /* OPENSSL_NO_DH */
                struct
                        {
                        CK_OBJECT_HANDLE        cipher_key; /* key handle */
                        unsigned char           key[PK11_KEY_LEN_MAX];
                        int                     key_len; /* priv key len */
                        int                     encrypt; /* 1/0 enc/decr */
                        } u_cipher;
                } opdata_u;
        } PK11_SESSION;

#define opdata_rsa_pub_key      opdata_u.u_RSA.rsa_pub_key
#define opdata_rsa_priv_key     opdata_u.u_RSA.rsa_priv_key
#define opdata_rsa_pub          opdata_u.u_RSA.rsa_pub
#define opdata_rsa_priv         opdata_u.u_RSA.rsa_priv
#define opdata_rsa_n_num        opdata_u.u_RSA.rsa_n_num
#define opdata_rsa_e_num        opdata_u.u_RSA.rsa_e_num
#define opdata_rsa_d_num        opdata_u.u_RSA.rsa_d_num
#define opdata_dsa_pub_key      opdata_u.u_DSA.dsa_pub_key
#define opdata_dsa_priv_key     opdata_u.u_DSA.dsa_priv_key
#define opdata_dsa_pub          opdata_u.u_DSA.dsa_pub
#define opdata_dsa_pub_num      opdata_u.u_DSA.dsa_pub_num
#define opdata_dsa_priv         opdata_u.u_DSA.dsa_priv
#define opdata_dsa_priv_num     opdata_u.u_DSA.dsa_priv_num
#define opdata_dh_key           opdata_u.u_DH.dh_key
#define opdata_dh               opdata_u.u_DH.dh
#define opdata_dh_priv_num      opdata_u.u_DH.dh_priv_num
#define opdata_cipher_key       opdata_u.u_cipher.cipher_key
#define opdata_key              opdata_u.u_cipher.key
#define opdata_key_len          opdata_u.u_cipher.key_len
#define opdata_encrypt          opdata_u.u_cipher.encrypt

/*
 * We have 3 different groups of operation types:
 *   1) asymmetric operations
 *   2) random operations
 *   3) symmetric and digest operations
 *
 * This division into groups stems from the fact that it's common that hardware
 * providers may support operations from one group only. For example, hardware
 * providers on UltraSPARC T2, n2rng(7d), ncp(7d), and n2cp(7d), each support
 * only a single group of operations.
 *
 * For every group a different slot can be chosen. That means that we must have
 * at least 3 different lists of cached PKCS#11 sessions since sessions from
 * different groups may be initialized in different slots.
 *
 * To provide locking granularity in multithreaded environment, the groups are
 * further splitted into types with each type having a separate session cache.
 */
typedef enum PK11_OPTYPE_ENUM
        {
        OP_RAND,
        OP_RSA,
        OP_DSA,
        OP_DH,
        OP_CIPHER,
        OP_DIGEST,
        OP_MAX
        } PK11_OPTYPE;

/*
 * This structure contains the heads of the lists forming the object caches
 * and locks associated with the lists.
 */
typedef struct PK11_st_CACHE
```



```
        {
        PK11_SESSION *head;
        pthread_mutex_t *lock;
        } PK11_CACHE;

/* structure for tracking handles of asymmetric key objects */
typedef struct PK11_active_st
        {
        CK_OBJECT_HANDLE h;
        unsigned int refcnt;
        struct PK11_active_st *prev;
        struct PK11_active_st *next;
        } PK11_active;

extern pthread_mutex_t *find_lock[];
extern PK11_active *active_list[];

#define LOCK_OBJSTORE(alg_type) \
        (void) pthread_mutex_lock(find_lock[alg_type])
#define UNLOCK_OBJSTORE(alg_type)       \
        (void) pthread_mutex_unlock(find_lock[alg_type])

extern PK11_SESSION *pk11_get_session(PK11_OPTYPE optype);
extern void pk11_return_session(PK11_SESSION *sp, PK11_OPTYPE optype);

#ifndef OPENSSL_NO_RSA
extern int pk11_destroy_rsa_key_objects(PK11_SESSION *session);
extern int pk11_destroy_rsa_object_pub(PK11_SESSION *sp, CK_BBOOL uselock);
extern int pk11_destroy_rsa_object_priv(PK11_SESSION *sp, CK_BBOOL uselock);
extern EVP_PKEY *pk11_load_privkey(ENGINE *e, const char *pubkey_file,
        UI_METHOD *ui_method, void *callback_data);
extern EVP_PKEY *pk11_load_pubkey(ENGINE *e, const char *pubkey_file,
        UI_METHOD *ui_method, void *callback_data);
extern RSA_METHOD *PK11_RSA(void);
#endif /* OPENSSL_NO_RSA */
#ifndef OPENSSL_NO_DSA
extern int pk11_destroy_dsa_key_objects(PK11_SESSION *session);
extern int pk11_destroy_dsa_object_pub(PK11_SESSION *sp, CK_BBOOL uselock);
extern int pk11_destroy_dsa_object_priv(PK11_SESSION *sp, CK_BBOOL uselock);
extern DSA_METHOD *PK11_DSA(void);
#endif /* OPENSSL_NO_DSA */
#ifndef OPENSSL_NO_DH
extern int pk11_destroy_dh_key_objects(PK11_SESSION *session);
extern int pk11_destroy_dh_object(PK11_SESSION *sp, CK_BBOOL uselock);
extern DH_METHOD *PK11_DH(void);
#endif /* OPENSSL_NO_DH */

extern CK_FUNCTION_LIST_PTR pFuncList;

#endif /* HW_PK11_ERR_H */
#endif /* ! codereview */
```

```
**********************************************************
    3438 Wed Aug 13 19:51:38 2014
new/usr/src/lib/openssl/include/kssl_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/kssl.h -*- mode: C; c-file-style: "eay" -*- */
/* Written by Vern Staats <staatsvr@asc.hpc.mil> for the OpenSSL project 2000.
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef KSSL_LCL_H
#define KSSL_LCL_H
```



```
#include <openssl/kssl.h>

#ifndef OPENSSL_NO_KRB5

#ifdef  __cplusplus
extern "C" {
#endif

/* Private (internal to OpenSSL) */
void print_krb5_data(char *label, krb5_data *kdata);
void print_krb5_authdata(char *label, krb5_authdata **adata);
void print_krb5_keyblock(char *label, krb5_keyblock *keyblk);

char *kstring(char *string);
char *knumber(int len, krb5_octet *contents);

const EVP_CIPHER *kssl_map_enc(krb5_enctype enctype);

int kssl_keytab_is_available(KSSL_CTX *kssl_ctx);
int kssl_tgt_is_available(KSSL_CTX *kssl_ctx);

#ifdef  __cplusplus
}
#endif
#endif  /* OPENSSL_NO_KRB5      */
#endif  /* KSSL_LCL_H   */
#endif /* ! codereview */
```

```
**********************************************************
   12520 Wed Aug 13 19:51:38 2014
new/usr/src/lib/openssl/include/md32_common.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md32_common.h */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

/*
 * This is a generic 32 bit "collector" for message digest algorithms.
 * Whenever needed it collects input character stream into chunks of
 * 32 bit values and invokes a block function that performs actual hash
 * calculations.
 *
 * Porting guide.
 *
 * Obligatory macros:
 *
```



```
 * DATA_ORDER_IS_BIG_ENDIAN or DATA_ORDER_IS_LITTLE_ENDIAN
 *      this macro defines byte order of input stream.
 * HASH_CBLOCK
 *      size of a unit chunk HASH_BLOCK operates on.
 * HASH_LONG
 *      has to be at lest 32 bit wide, if it's wider, then
 *      HASH_LONG_LOG2 *has to* be defined along
 * HASH_CTX
 *      context structure that at least contains following
 *      members:
 *              typedef struct {
 *                      ...
 *                      HASH_LONG       Nl,Nh;
 *                      either {
 *                      HASH_LONG       data[HASH_LBLOCK];
 *                      unsigned char   data[HASH_CBLOCK];
 *                      };
 *                      unsigned int    num;
 *                      ...
 *                      } HASH_CTX;
 *      data[] vector is expected to be zeroed upon first call to
 *      HASH_UPDATE.
 * HASH_UPDATE
 *      name of "Update" function, implemented here.
 * HASH_TRANSFORM
 *      name of "Transform" function, implemented here.
 * HASH_FINAL
 *      name of "Final" function, implemented here.
 * HASH_BLOCK_DATA_ORDER
 *      name of "block" function capable of treating *unaligned* input
 *      message in original (data) byte order, implemented externally.
 * HASH_MAKE_STRING
 *      macro convering context variables to an ASCII hash string.
 *
 * MD5 example:
 *
 *      #define DATA_ORDER_IS_LITTLE_ENDIAN
 *
 *      #define HASH_LONG               MD5_LONG
 *      #define HASH_LONG_LOG2          MD5_LONG_LOG2
 *      #define HASH_CTX                MD5_CTX
 *      #define HASH_CBLOCK             MD5_CBLOCK
 *      #define HASH_UPDATE             MD5_Update
 *      #define HASH_TRANSFORM          MD5_Transform
 *      #define HASH_FINAL              MD5_Final
 *      #define HASH_BLOCK_DATA_ORDER   md5_block_data_order
 *
 *                                      <appro@fy.chalmers.se>
 */

#if !defined(DATA_ORDER_IS_BIG_ENDIAN) && !defined(DATA_ORDER_IS_LITTLE_ENDIAN)
#error "DATA_ORDER must be defined!"
#endif

#ifndef HASH_CBLOCK
#error "HASH_CBLOCK must be defined!"
#endif
#ifndef HASH_LONG
#error "HASH_LONG must be defined!"
#endif
#ifndef HASH_CTX
#error "HASH_CTX must be defined!"
#endif

#ifndef HASH_UPDATE
#error "HASH_UPDATE must be defined!"
```

```
#endif
#ifndef HASH_TRANSFORM
#error "HASH_TRANSFORM must be defined!"
#endif
#ifndef HASH_FINAL
#error "HASH_FINAL must be defined!"
#endif

#ifndef HASH_BLOCK_DATA_ORDER
#error "HASH_BLOCK_DATA_ORDER must be defined!"
#endif

/*
 * Engage compiler specific rotate intrinsic function if available.
 */
#undef ROTATE
#ifndef PEDANTIC
# if defined(_MSC_VER) || defined(__ICC)
#  define ROTATE(a,n)   _lrotl(a,n)
# elif defined(__MWERKS__)
#  if defined(__POWERPC__)
#   define ROTATE(a,n)  __rlwinm(a,n,0,31)
#  elif defined(__MC68K__)
    /* Motorola specific tweak. <appro@fy.chalmers.se> */
#   define ROTATE(a,n)  ( n<24 ? __rol(a,n) : __ror(a,32-n) )
#  else
#   define ROTATE(a,n)  __rol(a,n)
#  endif
# elif defined(__GNUC__) && __GNUC__>=2 && !defined(OPENSSL_NO_ASM) && !defined(
  /*
   * Some GNU C inline assembler templates. Note that these are
   * rotates by *constant* number of bits! But that's exactly
   * what we need here...
   *                                    <appro@fy.chalmers.se>
   */
#  if defined(__i386) || defined(__i386__) || defined(__x86_64) || defined(__x86
#   define ROTATE(a,n)  ({ register unsigned int ret;   \
                                __asm__ (                       \
                                "roll %1,%0"            \
                                : "=r"(ret)             \
                                : "I"(n), "0"((unsigned int)(a))        \
                                : "cc");                \
                             ret;                       \
                          })
#  elif defined(_ARCH_PPC) || defined(_ARCH_PPC64) || \
        defined(__powerpc) || defined(__ppc__) || defined(__powerpc64__)
#   define ROTATE(a,n)  ({ register unsigned int ret;   \
                                __asm__ (                       \
                                "rlwinm %0,%1,%2,0,31"  \
                                : "=r"(ret)             \
                                : "r"(a), "I"(n));      \
                             ret;                       \
                          })
#  elif defined(__s390x__)
#   define ROTATE(a,n) ({ register unsigned int ret;    \
                                __asm__ ("rll %0,%1,%2" \
                                : "=r"(ret)             \
                                : "r"(a), "I"(n));      \
                             ret;                       \
                          })
#  endif
# endif
#endif /* PEDANTIC */

#ifndef ROTATE
#define ROTATE(a,n)     (((a)<<(n))|(((a)&0xffffffff)>>(32-(n))))
```



```
#endif

#if defined(DATA_ORDER_IS_BIG_ENDIAN)

#ifndef PEDANTIC
# if defined(__GNUC__) && __GNUC__>=2 && !defined(OPENSSL_NO_ASM) && !defined(OP
#  if ((defined(__i386) || defined(__i386__)) && !defined(I386_ONLY)) || \
      (defined(__x86_64) || defined(__x86_64__))
#   if !defined(B_ENDIAN)
    /*
     * This gives ~30-40% performance improvement in SHA-256 compiled
     * with gcc [on P4]. Well, first macro to be frank. We can pull
     * this trick on x86* platforms only, because these CPUs can fetch
     * unaligned data without raising an exception.
     */
#   define HOST_c2l(c,l)        ({ unsigned int r=*((const unsigned int *)(c));
                                   __asm__ ("bswapl %0":"=r"(r):"0"(r));
                                   (c)+=4; (l)=r;                       })
#   define HOST_l2c(l,c)        ({ unsigned int r=(l);                  \
                                   __asm__ ("bswapl %0":"=r"(r):"0"(r));
                                   *((unsigned int *)(c))=r; (c)+=4; r; })
#   endif
#  endif
# endif
#endif
#if defined(__s390__) || defined(__s390x__)
# define HOST_c2l(c,l) ((l)=*((const unsigned int *)(c)), (c)+=4, (l))
# define HOST_l2c(l,c) (*((unsigned int *)(c))=(l), (c)+=4, (l))
#endif

#ifndef HOST_c2l
#define HOST_c2l(c,l)   (l =(((unsigned long)(*((c)++)))<<24),          \
                         l|=(((unsigned long)(*((c)++)))<<16),          \
                         l|=(((unsigned long)(*((c)++)))<< 8),          \
                         l|=(((unsigned long)(*((c)++)))    ),          \
                         l)
#endif
#ifndef HOST_l2c
#define HOST_l2c(l,c)   (*((c)++)=(unsigned char)(((l)>>24)&0xff),      \
                         *((c)++)=(unsigned char)(((l)>>16)&0xff),      \
                         *((c)++)=(unsigned char)(((l)>> 8)&0xff),      \
                         *((c)++)=(unsigned char)(((l)    )&0xff),      \
                         l)
#endif

#elif defined(DATA_ORDER_IS_LITTLE_ENDIAN)

#ifndef PEDANTIC
# if defined(__GNUC__) && __GNUC__>=2 && !defined(OPENSSL_NO_ASM) && !defined(OP
#  if defined(__s390x__)
#   define HOST_c2l(c,l)        ({ __asm__ ("lrv        %0,%1"
                                   :"=d"(l) :"m"(*(const unsigned int *)(c)));\
                                   (c)+=4; (l);                         })
#   define HOST_l2c(l,c)        ({ __asm__ ("strv       %1,%0"
                                   :"=m"(*(unsigned int *)(c)) :"d"(l));\
                                   (c)+=4; (l);                         })
#  endif
# endif
#endif
#if defined(__i386) || defined(__i386__) || defined(__x86_64) || defined(__x86_6
# ifndef B_ENDIAN
   /* See comment in DATA_ORDER_IS_BIG_ENDIAN section. */
#  define HOST_c2l(c,l) ((l)=*((const unsigned int *)(c)), (c)+=4, l)
#  define HOST_l2c(l,c) (*((unsigned int *)(c))=(l), (c)+=4, l)
# endif
#endif
```

```
#ifndef HOST_c2l
#define HOST_c2l(c,l)   (l =(((unsigned long)(*((c)++)))    ),          \
                         l|=(((unsigned long)(*((c)++)))<< 8),          \
                         l|=(((unsigned long)(*((c)++)))<<16),          \
                         l|=(((unsigned long)(*((c)++)))<<24),          \
                         l)
#endif
#ifndef HOST_l2c
#define HOST_l2c(l,c)   (*((c)++)=(unsigned char)(((l)    )&0xff),      \
                         *((c)++)=(unsigned char)(((l)>> 8)&0xff),      \
                         *((c)++)=(unsigned char)(((l)>>16)&0xff),      \
                         *((c)++)=(unsigned char)(((l)>>24)&0xff),      \
                         l)
#endif

#endif

/*
 * Time for some action:-)
 */

int HASH_UPDATE (HASH_CTX *c, const void *data_, size_t len)
        {
        const unsigned char *data=data_;
        unsigned char *p;
        HASH_LONG l;
        size_t n;

        if (len==0) return 1;

        l=(c->Nl+(((HASH_LONG)len)<<3))&0xffffffffUL;
        /* 95-05-24 eay Fixed a bug with the overflow handling, thanks to
         * Wei Dai <weidai@eskimo.com> for pointing it out. */
        if (l < c->Nl) /* overflow */
                c->Nh++;
        c->Nh+=(HASH_LONG)(len>>29);    /* might cause compiler warning on 16-bi
        c->Nl=l;

        n = c->num;
        if (n != 0)
                {
                p=(unsigned char *)c->data;

                if (len >= HASH_CBLOCK || len+n >= HASH_CBLOCK)
                        {
                        memcpy (p+n,data,HASH_CBLOCK-n);
                        HASH_BLOCK_DATA_ORDER (c,p,1);
                        n      = HASH_CBLOCK-n;
                        data  += n;
                        len   -= n;
                        c->num = 0;
                        memset (p,0,HASH_CBLOCK);       /* keep it zeroed */
                        }
                else
                        {
                        memcpy (p+n,data,len);
                        c->num += (unsigned int)len;
                        return 1;
                        }
                }

        n = len/HASH_CBLOCK;
        if (n > 0)
                {
                HASH_BLOCK_DATA_ORDER (c,data,n);
```



```
                n    *= HASH_CBLOCK;
                data += n;
                len  -= n;
                }

        if (len != 0)
                {
                p = (unsigned char *)c->data;
                c->num = (unsigned int)len;
                memcpy (p,data,len);
                }
        return 1;
        }


void HASH_TRANSFORM (HASH_CTX *c, const unsigned char *data)
        {
        HASH_BLOCK_DATA_ORDER (c,data,1);
        }


int HASH_FINAL (unsigned char *md, HASH_CTX *c)
        {
        unsigned char *p = (unsigned char *)c->data;
        size_t n = c->num;

        p[n] = 0x80; /* there is always room for one */
        n++;

        if (n > (HASH_CBLOCK-8))
                {
                memset (p+n,0,HASH_CBLOCK-n);
                n=0;
                HASH_BLOCK_DATA_ORDER (c,p,1);
                }
        memset (p+n,0,HASH_CBLOCK-8-n);

        p += HASH_CBLOCK-8;
#if   defined(DATA_ORDER_IS_BIG_ENDIAN)
        (void)HOST_l2c(c->Nh,p);
        (void)HOST_l2c(c->Nl,p);
#elif defined(DATA_ORDER_IS_LITTLE_ENDIAN)
        (void)HOST_l2c(c->Nl,p);
        (void)HOST_l2c(c->Nh,p);
#endif
        p -= HASH_CBLOCK;
        HASH_BLOCK_DATA_ORDER (c,p,1);
        c->num=0;
        memset (p,0,HASH_CBLOCK);

#ifndef HASH_MAKE_STRING
#error "HASH_MAKE_STRING must be defined!"
#else
        HASH_MAKE_STRING(c,md);
#endif

        return 1;
        }

#ifndef MD32_REG_T
#if defined(__alpha) || defined(__sparcv9) || defined(__mips)
#define MD32_REG_T long
/*
 * This comment was originaly written for MD5, which is why it
 * discusses A-D. But it basically applies to all 32-bit digests,
 * which is why it was moved to common header file.
```

```
392  *
393  * In case you wonder why A-D are declared as long and not
394  * as MD5_LONG. Doing so results in slight performance
395  * boost on LP64 architectures. The catch is we don't
396  * really care if 32 MSBs of a 64-bit register get polluted
397  * with eventual overflows as we *save* only 32 LSBs in
398  * *either* case. Now declaring 'em long excuses the compiler
399  * from keeping 32 MSBs zeroed resulting in 13% performance
400  * improvement under SPARC Solaris7/64 and 5% under AlphaLinux.
401  * Well, to be honest it should say that this *prevents*
402  * performance degradation.
403  *                              <appro@fy.chalmers.se>
404  */
405 #else
406 /*
407  * Above is not absolute and there are LP64 compilers that
408  * generate better code if MD32_REG_T is defined int. The above
409  * pre-processor condition reflects the circumstances under which
410  * the conclusion was made and is subject to further extension.
411  *                              <appro@fy.chalmers.se>
412  */
413 #define MD32_REG_T int
414 #endif
415 #endif
416 #endif /* ! codereview */
```

```
**********************************************************
    4683 Wed Aug 13 19:51:38 2014
new/usr/src/lib/openssl/include/md4_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md4/md4_locl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdlib.h>
#include <string.h>
#include <openssl/opensslconf.h>
```



```
#include <openssl/md4.h>

#ifndef MD4_LONG_LOG2
#define MD4_LONG_LOG2 2 /* default to 32 bits */
#endif

void md4_block_data_order (MD4_CTX *c, const void *p,size_t num);

#define DATA_ORDER_IS_LITTLE_ENDIAN

#define HASH_LONG               MD4_LONG
#define HASH_CTX                MD4_CTX
#define HASH_CBLOCK             MD4_CBLOCK
#define HASH_UPDATE             MD4_Update
#define HASH_TRANSFORM          MD4_Transform
#define HASH_FINAL              MD4_Final
#define HASH_MAKE_STRING(c,s)   do {    \
        unsigned long ll;               \
        ll=(c)->A; (void)HOST_l2c(ll,(s));      \
        ll=(c)->B; (void)HOST_l2c(ll,(s));      \
        ll=(c)->C; (void)HOST_l2c(ll,(s));      \
        ll=(c)->D; (void)HOST_l2c(ll,(s));      \
        } while (0)
#define HASH_BLOCK_DATA_ORDER   md4_block_data_order

#include "md32_common.h"

/*
#define F(x,y,z)        (((x) & (y))  |  ((~(x)) & (z)))
#define G(x,y,z)        (((x) & (y))  |  ((x) & ((z))) | ((y) & ((z))))
*/

/* As pointed out by Wei Dai <weidai@eskimo.com>, the above can be
 * simplified to the code below.  Wei attributes these optimizations
 * to Peter Gutmann's SHS code, and he attributes it to Rich Schroeppel.
 */
#define F(b,c,d)        ((((c) ^ (d)) & (b)) ^ (d))
#define G(b,c,d)        (((b) & (c)) | ((b) & (d)) | ((c) & (d)))
#define H(b,c,d)        ((b) ^ (c) ^ (d))

#define R0(a,b,c,d,k,s,t) { \
        a+=((k)+(t)+F((b),(c),(d))); \
        a=ROTATE(a,s); };

#define R1(a,b,c,d,k,s,t) { \
        a+=((k)+(t)+G((b),(c),(d))); \
        a=ROTATE(a,s); };\

#define R2(a,b,c,d,k,s,t) { \
        a+=((k)+(t)+H((b),(c),(d))); \
        a=ROTATE(a,s); };
#endif /* ! codereview */
```

```
**********************************************************
    5188 Wed Aug 13 19:51:38 2014
new/usr/src/lib/openssl/include/md5_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md5/md5_locl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdlib.h>
#include <string.h>
#include <openssl/e_os2.h>
```



```
#include <openssl/md5.h>

#ifndef MD5_LONG_LOG2
#define MD5_LONG_LOG2 2 /* default to 32 bits */
#endif

#ifdef MD5_ASM
# if defined(__i386) || defined(__i386__) || defined(_M_IX86) || defined(__INTEL
      defined(__x86_64) || defined(__x86_64__) || defined(_M_AMD64) || defined(_M
#  define md5_block_data_order md5_block_asm_data_order
# elif defined(__ia64) || defined(__ia64__) || defined(_M_IA64)
#  define md5_block_data_order md5_block_asm_data_order
# endif
#endif

void md5_block_data_order (MD5_CTX *c, const void *p,size_t num);

#define DATA_ORDER_IS_LITTLE_ENDIAN

#define HASH_LONG               MD5_LONG
#define HASH_CTX                MD5_CTX
#define HASH_CBLOCK             MD5_CBLOCK
#define HASH_UPDATE             MD5_Update
#define HASH_TRANSFORM          MD5_Transform
#define HASH_FINAL              MD5_Final
#define HASH_MAKE_STRING(c,s)   do {    \
        unsigned long ll;               \
        ll=(c)->A; (void)HOST_l2c(ll,(s));      \
        ll=(c)->B; (void)HOST_l2c(ll,(s));      \
        ll=(c)->C; (void)HOST_l2c(ll,(s));      \
        ll=(c)->D; (void)HOST_l2c(ll,(s));      \
        } while (0)
#define HASH_BLOCK_DATA_ORDER   md5_block_data_order

#include "md32_common.h"

/*
#define F(x,y,z)        (((x) & (y))  |  ((~(x)) & (z)))
#define G(x,y,z)        (((x) & (z))  |  ((y) & (~(z))))
*/

/* As pointed out by Wei Dai <weidai@eskimo.com>, the above can be
 * simplified to the code below.  Wei attributes these optimizations
 * to Peter Gutmann's SHS code, and he attributes it to Rich Schroeppel.
 */
#define F(b,c,d)        ((((c) ^ (d)) & (b)) ^ (d))
#define G(b,c,d)        ((((b) ^ (c)) & (d)) ^ (c))
#define H(b,c,d)        ((b) ^ (c) ^ (d))
#define I(b,c,d)        (((~(d)) | (b)) ^ (c))

#define R0(a,b,c,d,k,s,t) { \
        a+=((k)+(t)+F((b),(c),(d))); \
        a=ROTATE(a,s); \
        a+=b; };\

#define R1(a,b,c,d,k,s,t) { \
        a+=((k)+(t)+G((b),(c),(d))); \
        a=ROTATE(a,s); \
        a+=b; };

#define R2(a,b,c,d,k,s,t) { \
        a+=((k)+(t)+H((b),(c),(d))); \
        a=ROTATE(a,s); \
        a+=b; };

#define R3(a,b,c,d,k,s,t) { \
```

```
128         a+=((k)+(t)+I((b),(c),(d))); \
129         a=ROTATE(a,s); \
130         a+=b; };
131 #endif /* ! codereview */
```

```
**********************************************************
    3636 Wed Aug 13 19:51:38 2014
new/usr/src/lib/openssl/include/modes_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ====================================================================
 * Copyright (c) 2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use is governed by OpenSSL license.
 * ====================================================================
 */

#include <openssl/modes.h>


#if (defined(_WIN32) || defined(_WIN64)) && !defined(__MINGW32__)
typedef __int64 i64;
typedef unsigned __int64 u64;
#define U64(C) C##UI64
#elif defined(__arch64__)
typedef long i64;
typedef unsigned long u64;
#define U64(C) C##UL
#else
typedef long long i64;
typedef unsigned long long u64;
#define U64(C) C##ULL
#endif

typedef unsigned int u32;
typedef unsigned char u8;

#define STRICT_ALIGNMENT 1
#if defined(__i386)     || defined(__i386__)    || \
    defined(__x86_64)   || defined(__x86_64__)  || \
    defined(_M_IX86)    || defined(_M_AMD64)    || defined(_M_X64) || \
    defined(__s390__)   || defined(__s390x__)
# undef STRICT_ALIGNMENT
#endif

#if !defined(PEDANTIC) && !defined(OPENSSL_NO_ASM) && !defined(OPENSSL_NO_INLINE
#if defined(__GNUC__) && __GNUC__>=2
# if defined(__x86_64) || defined(__x86_64__)
#  define BSWAP8(x) ({  u64 ret=(x);                    \
                        __asm__ ("bswapq %0"            \
                        : "+r"(ret));   ret;            })
#  define BSWAP4(x) ({  u32 ret=(x);                    \
                        __asm__ ("bswapl %0"            \
                        : "+r"(ret));   ret;            })
# elif (defined(__i386) || defined(__i386__)) && !defined(I386_ONLY)
#  define BSWAP8(x) ({  u32 lo=(u64)(x)>>32,hi=(x);     \
                        __asm__ ("bswapl %0; bswapl %1" \
                        : "+r"(hi),"+r"(lo));           \
                        (u64)hi<<32|lo;                 })
#  define BSWAP4(x) ({  u32 ret=(x);                    \
                        __asm__ ("bswapl %0"            \
                        : "+r"(ret));   ret;            })
# elif (defined(__arm__) || defined(__arm)) && !defined(STRICT_ALIGNMENT)
#  define BSWAP8(x) ({  u32 lo=(u64)(x)>>32,hi=(x);     \
                        __asm__ ("rev %0,%0; rev %1,%1" \
                        : "+r"(hi),"+r"(lo));           \
                        (u64)hi<<32|lo;                 })
#  define BSWAP4(x) ({  u32 ret;                        \
                        __asm__ ("rev %0,%1"            \
                        : "=r"(ret) : "r"((u32)(x)));   \
                        ret;                            })
```



```
# endif
#elif defined(_MSC_VER)
# if _MSC_VER>=1300
#  pragma intrinsic(_byteswap_uint64,_byteswap_ulong)
#  define BSWAP8(x)     _byteswap_uint64((u64)(x))
#  define BSWAP4(x)     _byteswap_ulong((u32)(x))
# elif defined(_M_IX86)
   __inline u32 _bswap4(u32 val) {
        _asm mov eax,val
        _asm bswap eax
   }
#  define BSWAP4(x)     _bswap4(x)
# endif
#endif
#endif

#if defined(BSWAP4) && !defined(STRICT_ALIGNMENT)
#define GETU32(p)       BSWAP4(*(const u32 *)(p))
#define PUTU32(p,v)     *(u32 *)(p) = BSWAP4(v)
#else
#define GETU32(p)       ((u32)(p)[0]<<24|(u32)(p)[1]<<16|(u32)(p)[2]<<8|(u32)(p)
#define PUTU32(p,v)     ((p)[0]=(u8)((v)>>24),(p)[1]=(u8)((v)>>16),(p)[2]=(u8)((
#endif

/* GCM definitions */

typedef struct { u64 hi,lo; } u128;

#ifdef  TABLE_BITS
#undef  TABLE_BITS
#endif
/*
 * Even though permitted values for TABLE_BITS are 8, 4 and 1, it should
 * never be set to 8 [or 1]. For further information see gcm128.c.
 */
#define TABLE_BITS 4

struct gcm128_context {
        /* Following 6 names follow names in GCM specification */
        union { u64 u[2]; u32 d[4]; u8 c[16]; size_t t[16/sizeof(size_t)]; }
          Yi,EKi,EK0,len,Xi,H;
        /* Relative position of Xi, H and pre-computed Htable is used
         * in some assembler modules, i.e. don't change the order! */
#if TABLE_BITS==8
        u128 Htable[256];
#else
        u128 Htable[16];
        void (*gmult)(u64 Xi[2],const u128 Htable[16]);
        void (*ghash)(u64 Xi[2],const u128 Htable[16],const u8 *inp,size_t len);
#endif
        unsigned int mres, ares;
        block128_f block;
        void *key;
};

struct xts128_context {
        void      *key1, *key2;
        block128_f block1,block2;
};

struct ccm128_context {
        union { u64 u[2]; u8 c[16]; } nonce, cmac;
        u64 blocks;
        block128_f block;
        void *key;
};
```

```
128 #endif /* ! codereview */
```

****
2111 Wed Aug 13 19:51:39 2014
new/usr/src/lib/openssl/include/o_dir.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* crypto/o_dir.h -*- mode:C; c-file-style: "eay" -*- */
/* Copied from Richard Levitte's (richard@levitte.org) LP library.  All
 * symbol names have been changed, with permission from the author.
 */

/* $LP: LPlib/source/LPdir.h,v 1.1 2004/06/14 08:56:04 _cvs_levitte Exp $ */
/*
 * Copyright (c) 2004, Richard Levitte <richard@levitte.org>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */


#ifndef O_DIR_H
#define O_DIR_H

#ifdef __cplusplus
extern "C" {
#endif

  typedef struct OPENSSL_dir_context_st OPENSSL_DIR_CTX;

  /* returns NULL on error or end-of-directory.
     If it is end-of-directory, errno will be zero */
  const char *OPENSSL_DIR_read(OPENSSL_DIR_CTX **ctx, const char *directory);
  /* returns 1 on success, 0 on error */
  int OPENSSL_DIR_end(OPENSSL_DIR_CTX **ctx);

#ifdef __cplusplus
}
#endif

#endif /* LPDIR_H */
#endif /* ! codereview */
```

```
**********************************************************
    3060 Wed Aug 13 19:51:39 2014
new/usr/src/lib/openssl/include/o_str.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/o_str.h -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2003.
 */
/* ====================================================================
 * Copyright (c) 2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_O_STR_H
#define HEADER_O_STR_H
```



```
#include <stddef.h>             /* to get size_t */

int OPENSSL_strcasecmp(const char *str1, const char *str2);
int OPENSSL_strncasecmp(const char *str1, const char *str2, size_t n);
int OPENSSL_memcmp(const void *p1,const void *p2,size_t n);

#endif
#endif /* ! codereview */
```

```
**********************************************************
    2988 Wed Aug 13 19:51:39 2014
new/usr/src/lib/openssl/include/o_time.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/o_time.h -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_O_TIME_H
#define HEADER_O_TIME_H
```



```
#include <time.h>

struct tm *OPENSSL_gmtime(const time_t *timer, struct tm *result);
int OPENSSL_gmtime_adj(struct tm *tm, int offset_day, long offset_sec);

#endif
#endif /* ! codereview */
```

```
**********************************************************
  255546 Wed Aug 13 19:51:39 2014
new/usr/src/lib/openssl/include/obj_dat.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/objects/obj_dat.h */

/* THIS FILE IS GENERATED FROM objects.h by obj_dat.pl via the
 * following command:
 * perl obj_dat.pl obj_mac.h obj_dat.h
 */

/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
```



```
 * [including the GNU Public Licence.]
 */

#define NUM_NID 920
#define NUM_SN 913
#define NUM_LN 913
#define NUM_OBJ 857

static const unsigned char lvalues[5974]={
0x2A,0x86,0x48,0x86,0xF7,0x0D,                  /* [  0] OBJ_rsadsi */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,             /* [  6] OBJ_pkcs */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x02,        /* [ 13] OBJ_md2 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x05,        /* [ 21] OBJ_md5 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x03,0x04,        /* [ 29] OBJ_rc4 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x01,/* [ 37] OBJ_rsaEncryption */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x02,/* [ 46] OBJ_md2WithRSAEncryption *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x04,/* [ 55] OBJ_md5WithRSAEncryption *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x01,/* [ 64] OBJ_pbeWithMD2AndDES_CBC *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x03,/* [ 73] OBJ_pbeWithMD5AndDES_CBC *
0x55,                                           /* [ 82] OBJ_X500 */
0x55,0x04,                                      /* [ 83] OBJ_X509 */
0x55,0x04,0x03,                                 /* [ 85] OBJ_commonName */
0x55,0x04,0x06,                                 /* [ 88] OBJ_countryName */
0x55,0x04,0x07,                                 /* [ 91] OBJ_localityName */
0x55,0x04,0x08,                                 /* [ 94] OBJ_stateOrProvinceName */
0x55,0x04,0x0A,                                 /* [ 97] OBJ_organizationName */
0x55,0x04,0x0B,                                 /* [100] OBJ_organizationalUnitName
0x55,0x08,0x01,0x01,                            /* [103] OBJ_rsa */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x07,        /* [107] OBJ_pkcs7 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x07,0x01,/* [115] OBJ_pkcs7_data */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x07,0x02,/* [124] OBJ_pkcs7_signed */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x07,0x03,/* [133] OBJ_pkcs7_enveloped */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x07,0x04,/* [142] OBJ_pkcs7_signedAndEnvelop
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x07,0x05,/* [151] OBJ_pkcs7_digest */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x07,0x06,/* [160] OBJ_pkcs7_encrypted */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x03,        /* [169] OBJ_pkcs3 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x03,0x01,/* [177] OBJ_dhKeyAgreement */
0x2B,0x0E,0x03,0x02,0x06,                       /* [186] OBJ_des_ecb */
0x2B,0x0E,0x03,0x02,0x09,                       /* [191] OBJ_des_cfb64 */
0x2B,0x0E,0x03,0x02,0x07,                       /* [196] OBJ_des_cbc */
0x2B,0x0E,0x03,0x02,0x11,                       /* [201] OBJ_des_ede_ecb */
0x2B,0x06,0x01,0x04,0x01,0x81,0x3C,0x07,0x01,0x01,0x02,/* [206] OBJ_idea_cbc */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x03,0x02,        /* [217] OBJ_rc2_cbc */
0x2B,0x0E,0x03,0x02,0x12,                       /* [225] OBJ_sha */
0x2B,0x0E,0x03,0x02,0x0F,                       /* [230] OBJ_shaWithRSAEncryption *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x03,0x07,        /* [235] OBJ_des_ede3_cbc */
0x2B,0x0E,0x03,0x02,0x08,                       /* [243] OBJ_des_ofb64 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,        /* [248] OBJ_pkcs9 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x01,/* [256] OBJ_pkcs9_emailAddress */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x02,/* [265] OBJ_pkcs9_unstructuredName
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x03,/* [274] OBJ_pkcs9_contentType */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x04,/* [283] OBJ_pkcs9_messageDigest */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x05,/* [292] OBJ_pkcs9_signingTime */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x06,/* [301] OBJ_pkcs9_countersignature
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x07,/* [310] OBJ_pkcs9_challengePasswor
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x08,/* [319] OBJ_pkcs9_unstructuredAddr
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x09,/* [328] OBJ_pkcs9_extCertAttribute
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,             /* [337] OBJ_netscape */
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,        /* [344] OBJ_netscape_cert_extensio
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x02,        /* [352] OBJ_netscape_data_type */
0x2B,0x0E,0x03,0x02,0x1A,                       /* [360] OBJ_sha1 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x05,/* [365] OBJ_sha1WithRSAEncryption
0x2B,0x0E,0x03,0x02,0x0D,                       /* [374] OBJ_dsaWithSHA */
0x2B,0x0E,0x03,0x02,0x0C,                       /* [379] OBJ_dsa_2 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x0B,/* [384] OBJ_pbeWithSHA1AndRC2_CBC
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x0C,/* [393] OBJ_id_pbkdf2 */
```

```
0x2B,0x0E,0x03,0x02,0x1B,                       /* [402] OBJ_dsaWithSHA1_2 */
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x01,/* [407] OBJ_netscape_cert_type */
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x02,/* [416] OBJ_netscape_base_url */
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x03,/* [425] OBJ_netscape_revocation_ur
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x04,/* [434] OBJ_netscape_ca_revocation
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x07,/* [443] OBJ_netscape_renewal_url *
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x08,/* [452] OBJ_netscape_ca_policy_url
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x0C,/* [461] OBJ_netscape_ssl_server_na
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x01,0x0D,/* [470] OBJ_netscape_comment */
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x02,0x05,/* [479] OBJ_netscape_cert_sequence
0x55,0x1D,                                      /* [488] OBJ_id_ce */
0x55,0x1D,0x0E,                                 /* [490] OBJ_subject_key_identifier
0x55,0x1D,0x0F,                                 /* [493] OBJ_key_usage */
0x55,0x1D,0x10,                                 /* [496] OBJ_private_key_usage_peri
0x55,0x1D,0x11,                                 /* [499] OBJ_subject_alt_name */
0x55,0x1D,0x12,                                 /* [502] OBJ_issuer_alt_name */
0x55,0x1D,0x13,                                 /* [505] OBJ_basic_constraints */
0x55,0x1D,0x14,                                 /* [508] OBJ_crl_number */
0x55,0x1D,0x20,                                 /* [511] OBJ_certificate_policies *
0x55,0x1D,0x23,                                 /* [514] OBJ_authority_key_identifi
0x2B,0x06,0x01,0x04,0x01,0x97,0x55,0x01,0x02,/* [517] OBJ_bf_cbc */
0x55,0x08,0x03,0x65,                            /* [526] OBJ_mdc2 */
0x55,0x08,0x03,0x64,                            /* [530] OBJ_mdc2WithRSA */
0x55,0x04,0x2A,                                 /* [534] OBJ_givenName */
0x55,0x04,0x04,                                 /* [537] OBJ_surname */
0x55,0x04,0x2B,                                 /* [540] OBJ_initials */
0x55,0x1D,0x1F,                                 /* [543] OBJ_crl_distribution_point
0x2B,0x0E,0x03,0x02,0x03,                       /* [546] OBJ_md5WithRSA */
0x55,0x04,0x05,                                 /* [551] OBJ_serialNumber */
0x55,0x04,0x0C,                                 /* [554] OBJ_title */
0x55,0x04,0x0D,                                 /* [557] OBJ_description */
0x2A,0x86,0x48,0x86,0xF6,0x7D,0x07,0x42,0x0A,/* [560] OBJ_cast5_cbc */
0x2A,0x86,0x48,0x86,0xF6,0x7D,0x07,0x42,0x0C,/* [569] OBJ_pbeWithMD5AndCast5_CBC
0x2A,0x86,0x48,0xCE,0x38,0x04,0x03,             /* [578] OBJ_dsaWithSHA1 */
0x2B,0x0E,0x03,0x02,0x1D,                       /* [585] OBJ_sha1WithRSA */
0x2A,0x86,0x48,0xCE,0x38,0x04,0x01,             /* [590] OBJ_dsa */
0x2B,0x24,0x03,0x02,0x01,                       /* [597] OBJ_ripemd160 */
0x2B,0x24,0x03,0x03,0x01,0x02,                  /* [602] OBJ_ripemd160WithRSA */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x03,0x08,        /* [608] OBJ_rc5_cbc */
0x29,0x01,0x01,0x85,0x1A,0x01,                  /* [616] OBJ_rle_compression */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x08,/* [622] OBJ_zlib_compres
0x55,0x1D,0x25,                                 /* [633] OBJ_ext_key_usage */
0x2B,0x06,0x01,0x05,0x05,0x07,                  /* [636] OBJ_id_pkix */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,             /* [642] OBJ_id_kp */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x01,        /* [649] OBJ_server_auth */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x02,        /* [657] OBJ_client_auth */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x03,        /* [665] OBJ_code_sign */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x04,        /* [673] OBJ_email_protect */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x08,        /* [681] OBJ_time_stamp */
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x02,0x01,0x15,/* [689] OBJ_ms_code_ind */
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x02,0x01,0x16,/* [699] OBJ_ms_code_com */
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x0A,0x03,0x01,/* [709] OBJ_ms_ctl_sign */
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x0A,0x03,0x03,/* [719] OBJ_ms_sgc */
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x0A,0x03,0x04,/* [729] OBJ_ms_efs */
0x60,0x86,0x48,0x01,0x86,0xF8,0x42,0x04,0x01,/* [739] OBJ_ns_sgc */
0x55,0x1D,0x1B,                                 /* [748] OBJ_delta_crl */
0x55,0x1D,0x15,                                 /* [751] OBJ_crl_reason */
0x55,0x1D,0x18,                                 /* [754] OBJ_invalidity_date */
0x2B,0x65,0x01,0x04,0x01,                       /* [757] OBJ_sxnet */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x01,0x01,/* [762] OBJ_pbe_WithSHA1And12
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x01,0x02,/* [772] OBJ_pbe_WithSHA1And40
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x01,0x03,/* [782] OBJ_pbe_WithSHA1And3_
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x01,0x04,/* [792] OBJ_pbe_WithSHA1And2_
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x01,0x05,/* [802] OBJ_pbe_WithSHA1And12
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x01,0x06,/* [812] OBJ_pbe_WithSHA1And40
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x0A,0x01,0x01,/* [822] OBJ_keyBag */
```



```
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x0A,0x01,0x02,/* [833] OBJ_pkcs8Shroude
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x0A,0x01,0x03,/* [844] OBJ_certBag */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x0A,0x01,0x04,/* [855] OBJ_crlBag */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x0A,0x01,0x05,/* [866] OBJ_secretBag */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x0C,0x0A,0x01,0x06,/* [877] OBJ_safeContents
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x14,/* [888] OBJ_friendlyName */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x15,/* [897] OBJ_localKeyID */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x16,0x01,/* [906] OBJ_x509Certificate *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x16,0x02,/* [916] OBJ_sdsiCertificate *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x17,0x01,/* [926] OBJ_x509Crl */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x0D,/* [936] OBJ_pbes2 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x0E,/* [945] OBJ_pbmac1 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x07,     /* [954] OBJ_hmacWithSHA1 */
0x2B,0x06,0x01,0x05,0x05,0x07,0x02,0x01,     /* [962] OBJ_id_qt_cps */
0x2B,0x06,0x01,0x05,0x05,0x07,0x02,0x02,     /* [970] OBJ_id_qt_unotice */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x0F,/* [978] OBJ_SMIMECapabilities */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x04,/* [987] OBJ_pbeWithMD2AndRC2_CBC *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x06,/* [996] OBJ_pbeWithMD5AndRC2_CBC *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,0x0A,/* [1005] OBJ_pbeWithSHA1AndDES_CBC
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x02,0x01,0x0E,/* [1014] OBJ_ms_ext_req */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x0E,/* [1024] OBJ_ext_req */
0x55,0x04,0x29,                              /* [1033] OBJ_name */
0x55,0x04,0x2E,                              /* [1036] OBJ_dnQualifier */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,          /* [1039] OBJ_id_pe */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,          /* [1046] OBJ_id_ad */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x01,     /* [1053] OBJ_info_access */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,     /* [1061] OBJ_ad_OCSP */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x02,     /* [1069] OBJ_ad_ca_issuers */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x09,     /* [1077] OBJ_OCSP_sign */
0x2A,                                        /* [1085] OBJ_member_body */
0x2A,0x86,0x48,                              /* [1086] OBJ_ISO_US */
0x2A,0x86,0x48,0xCE,0x38,                    /* [1089] OBJ_X9_57 */
0x2A,0x86,0x48,0xCE,0x38,0x04,               /* [1094] OBJ_X9cm */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,     /* [1100] OBJ_pkcs1 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x05,     /* [1108] OBJ_pkcs5 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,/* [1116] OBJ_SMIME */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,/* [1125] OBJ_id_smime_mod */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,/* [1135] OBJ_id_smime_ct */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,/* [1145] OBJ_id_smime_aa */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,/* [1155] OBJ_id_smime_alg */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x04,/* [1165] OBJ_id_smime_cd */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x05,/* [1175] OBJ_id_smime_spq */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x06,/* [1185] OBJ_id_smime_cti */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x01,/* [1195] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x02,/* [1206] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x03,/* [1217] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x04,/* [1228] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x05,/* [1239] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x06,/* [1250] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x07,/* [1261] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x00,0x08,/* [1272] OBJ_id_smime_mo
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x01,/* [1283] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x02,/* [1294] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x03,/* [1305] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x04,/* [1316] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x05,/* [1327] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x06,/* [1338] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x07,/* [1349] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x08,/* [1360] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x01,/* [1371] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x02,/* [1382] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x03,/* [1393] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x04,/* [1404] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x05,/* [1415] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x06,/* [1426] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x07,/* [1437] OBJ_id_smime_aa
```

```
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x08,/* [1448] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x09,/* [1459] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x0A,/* [1470] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x0B,/* [1481] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x0C,/* [1492] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x0D,/* [1503] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x0E,/* [1514] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x0F,/* [1525] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x10,/* [1536] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x11,/* [1547] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x12,/* [1558] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x13,/* [1569] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x14,/* [1580] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x15,/* [1591] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x16,/* [1602] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x17,/* [1613] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x18,/* [1624] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x19,/* [1635] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x1A,/* [1646] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x1B,/* [1657] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x1C,/* [1668] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x02,0x1D,/* [1679] OBJ_id_smime_aa
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x01,/* [1690] OBJ_id_smime_al
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x02,/* [1701] OBJ_id_smime_al
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x03,/* [1712] OBJ_id_smime_al
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x04,/* [1723] OBJ_id_smime_al
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x05,/* [1734] OBJ_id_smime_al
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x06,/* [1745] OBJ_id_smime_al
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x07,/* [1756] OBJ_id_smime_al
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x04,0x01,/* [1767] OBJ_id_smime_cd
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x05,0x01,/* [1778] OBJ_id_smime_sp
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x05,0x02,/* [1789] OBJ_id_smime_sp
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x06,0x01,/* [1800] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x06,0x02,/* [1811] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x06,0x03,/* [1822] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x06,0x04,/* [1833] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x06,0x05,/* [1844] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x06,0x06,/* [1855] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x04,     /* [1866] OBJ_md4 */
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,          /* [1874] OBJ_id_pkix_mod */
0x2B,0x06,0x01,0x05,0x05,0x07,0x02,          /* [1881] OBJ_id_qt */
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,          /* [1888] OBJ_id_it */
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,          /* [1895] OBJ_id_pkip */
0x2B,0x06,0x01,0x05,0x05,0x07,0x06,          /* [1902] OBJ_id_alg */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,          /* [1909] OBJ_id_cmc */
0x2B,0x06,0x01,0x05,0x05,0x07,0x08,          /* [1916] OBJ_id_on */
0x2B,0x06,0x01,0x05,0x05,0x07,0x09,          /* [1923] OBJ_id_pda */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0A,          /* [1930] OBJ_id_aca */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0B,          /* [1937] OBJ_id_qcs */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0C,          /* [1944] OBJ_id_cct */
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x01,     /* [1951] OBJ_id_pkix1_explicit_88
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x02,     /* [1959] OBJ_id_pkix1_implicit_88
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x03,     /* [1967] OBJ_id_pkix1_explicit_93
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x04,     /* [1975] OBJ_id_pkix1_implicit_93
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x05,     /* [1983] OBJ_id_mod_crmf */
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x06,     /* [1991] OBJ_id_mod_cmc */
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x07,     /* [1999] OBJ_id_mod_kea_profile_88
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x08,     /* [2007] OBJ_id_mod_kea_profile_93
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x09,     /* [2015] OBJ_id_mod_cmp */
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x0A,     /* [2023] OBJ_id_mod_qualified_cert
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x0B,     /* [2031] OBJ_id_mod_qualified_cert
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x0C,     /* [2039] OBJ_id_mod_attribute_cert
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x0D,     /* [2047] OBJ_id_mod_timestamp_prot
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x0E,     /* [2055] OBJ_id_mod_ocsp */
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x0F,     /* [2063] OBJ_id_mod_dvcs */
0x2B,0x06,0x01,0x05,0x05,0x07,0x00,0x10,     /* [2071] OBJ_id_mod_cmp2000 */
```



```
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x02,     /* [2079] OBJ_biometricInfo */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x03,     /* [2087] OBJ_qcStatements */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x04,     /* [2095] OBJ_ac_auditEntity */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x05,     /* [2103] OBJ_ac_targeting */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x06,     /* [2111] OBJ_aaControls */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x07,     /* [2119] OBJ_sbgp_ipAddrBlock */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x08,     /* [2127] OBJ_sbgp_autonomousSysNum
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x09,     /* [2135] OBJ_sbgp_routerIdentifier
0x2B,0x06,0x01,0x05,0x05,0x07,0x02,0x03,     /* [2143] OBJ_textNotice */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x05,     /* [2151] OBJ_ipsecEndSystem */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x06,     /* [2159] OBJ_ipsecTunnel */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x07,     /* [2167] OBJ_ipsecUser */
0x2B,0x06,0x01,0x05,0x05,0x07,0x03,0x0A,     /* [2175] OBJ_dvcs */
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x01,     /* [2183] OBJ_id_it_caProtEncCert *
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x02,     /* [2191] OBJ_id_it_signKeyPairType
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x03,     /* [2199] OBJ_id_it_encKeyPairTypes
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x04,     /* [2207] OBJ_id_it_preferredSymmAl
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x05,     /* [2215] OBJ_id_it_caKeyUpdateInfo
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x06,     /* [2223] OBJ_id_it_currentCRL */
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x07,     /* [2231] OBJ_id_it_unsupportedOIDs
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x08,     /* [2239] OBJ_id_it_subscriptionReq
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x09,     /* [2247] OBJ_id_it_subscriptionRes
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x0A,     /* [2255] OBJ_id_it_keyPairParamReq
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x0B,     /* [2263] OBJ_id_it_keyPairParamRep
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x0C,     /* [2271] OBJ_id_it_revPassphrase *
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x0D,     /* [2279] OBJ_id_it_implicitConfirm
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x0E,     /* [2287] OBJ_id_it_confirmWaitTime
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x0F,     /* [2295] OBJ_id_it_origPKIMessage
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x01,     /* [2303] OBJ_id_regCtrl */
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x02,     /* [2311] OBJ_id_regInfo */
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x01,0x01,/* [2319] OBJ_id_regCtrl_regToken *
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x01,0x02,/* [2328] OBJ_id_regCtrl_authentica
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x01,0x03,/* [2337] OBJ_id_regCtrl_pkiPublica
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x01,0x04,/* [2346] OBJ_id_regCtrl_pkiArchive
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x01,0x05,/* [2355] OBJ_id_regCtrl_oldCertID
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x01,0x06,/* [2364] OBJ_id_regCtrl_protocolEn
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x02,0x01,/* [2373] OBJ_id_regInfo_utf8Pairs
0x2B,0x06,0x01,0x05,0x05,0x07,0x05,0x02,0x02,/* [2382] OBJ_id_regInfo_certReq */
0x2B,0x06,0x01,0x05,0x05,0x07,0x06,0x01,     /* [2391] OBJ_id_alg_des40 */
0x2B,0x06,0x01,0x05,0x05,0x07,0x06,0x02,     /* [2399] OBJ_id_alg_noSignature */
0x2B,0x06,0x01,0x05,0x05,0x07,0x06,0x03,     /* [2407] OBJ_id_alg_dh_sig_hmac_sh
0x2B,0x06,0x01,0x05,0x05,0x07,0x06,0x04,     /* [2415] OBJ_id_alg_dh_pop */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x01,     /* [2423] OBJ_id_cmc_statusInfo */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x02,     /* [2431] OBJ_id_cmc_identification
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x03,     /* [2439] OBJ_id_cmc_identityProof
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x04,     /* [2447] OBJ_id_cmc_dataReturn */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x05,     /* [2455] OBJ_id_cmc_transactionId
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x06,     /* [2463] OBJ_id_cmc_senderNonce */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x07,     /* [2471] OBJ_id_cmc_recipientNonce
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x08,     /* [2479] OBJ_id_cmc_addExtensions
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x09,     /* [2487] OBJ_id_cmc_encryptedPOP *
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x0A,     /* [2495] OBJ_id_cmc_decryptedPOP *
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x0B,     /* [2503] OBJ_id_cmc_lraPOPWitness
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x0F,     /* [2511] OBJ_id_cmc_getCert */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x10,     /* [2519] OBJ_id_cmc_getCRL */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x11,     /* [2527] OBJ_id_cmc_revokeRequest
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x12,     /* [2535] OBJ_id_cmc_regInfo */
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x13,     /* [2543] OBJ_id_cmc_responseInfo *
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x15,     /* [2551] OBJ_id_cmc_queryPending *
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x16,     /* [2559] OBJ_id_cmc_popLinkRandom
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x17,     /* [2567] OBJ_id_cmc_popLinkWitness
0x2B,0x06,0x01,0x05,0x05,0x07,0x07,0x18,     /* [2575] OBJ_id_cmc_confirmCertAcc
0x2B,0x06,0x01,0x05,0x05,0x07,0x08,0x01,     /* [2583] OBJ_id_on_personalData */
0x2B,0x06,0x01,0x05,0x05,0x07,0x09,0x01,     /* [2591] OBJ_id_pda_dateOfBirth */
0x2B,0x06,0x01,0x05,0x05,0x07,0x09,0x02,     /* [2599] OBJ_id_pda_placeOfBirth *
0x2B,0x06,0x01,0x05,0x05,0x07,0x09,0x03,     /* [2607] OBJ_id_pda_gender */
```

```
0x2B,0x06,0x01,0x05,0x05,0x07,0x09,0x04,     /* [2615] OBJ_id_pda_countryOfCitiz
0x2B,0x06,0x01,0x05,0x05,0x07,0x09,0x05,     /* [2623] OBJ_id_pda_countryOfResid
0x2B,0x06,0x01,0x05,0x05,0x07,0x0A,0x01,     /* [2631] OBJ_id_aca_authentication
0x2B,0x06,0x01,0x05,0x05,0x07,0x0A,0x02,     /* [2639] OBJ_id_aca_accessIdentity
0x2B,0x06,0x01,0x05,0x05,0x07,0x0A,0x03,     /* [2647] OBJ_id_aca_chargingIdenti
0x2B,0x06,0x01,0x05,0x05,0x07,0x0A,0x04,     /* [2655] OBJ_id_aca_group */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0A,0x05,     /* [2663] OBJ_id_aca_role */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0B,0x01,     /* [2671] OBJ_id_qcs_pkixQCSyntax_v
0x2B,0x06,0x01,0x05,0x05,0x07,0x0C,0x01,     /* [2679] OBJ_id_cct_crs */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0C,0x02,     /* [2687] OBJ_id_cct_PKIData */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0C,0x03,     /* [2695] OBJ_id_cct_PKIResponse */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x03,     /* [2703] OBJ_ad_timeStamping */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x04,     /* [2711] OBJ_ad_dvcs */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x01,/* [2719] OBJ_id_pkix_OCSP_basic */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x02,/* [2728] OBJ_id_pkix_OCSP_Nonce */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x03,/* [2737] OBJ_id_pkix_OCSP_CrlID */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x04,/* [2746] OBJ_id_pkix_OCSP_acceptab
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x05,/* [2755] OBJ_id_pkix_OCSP_noCheck
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x06,/* [2764] OBJ_id_pkix_OCSP_archiveC
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x07,/* [2773] OBJ_id_pkix_OCSP_serviceL
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x08,/* [2782] OBJ_id_pkix_OCSP_extended
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x09,/* [2791] OBJ_id_pkix_OCSP_valid */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x0A,/* [2800] OBJ_id_pkix_OCSP_path */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x01,0x0B,/* [2809] OBJ_id_pkix_OCSP_trustRoo
0x2B,0x0E,0x03,0x02,                         /* [2818] OBJ_algorithm */
0x2B,0x0E,0x03,0x02,0x0B,                    /* [2822] OBJ_rsaSignature */
0x55,0x08,                                   /* [2827] OBJ_X500algorithms */
0x2B,                                        /* [2829] OBJ_org */
0x2B,0x06,                                   /* [2830] OBJ_dod */
0x2B,0x06,0x01,                              /* [2832] OBJ_iana */
0x2B,0x06,0x01,0x01,                         /* [2835] OBJ_Directory */
0x2B,0x06,0x01,0x02,                         /* [2839] OBJ_Management */
0x2B,0x06,0x01,0x03,                         /* [2843] OBJ_Experimental */
0x2B,0x06,0x01,0x04,                         /* [2847] OBJ_Private */
0x2B,0x06,0x01,0x05,                         /* [2851] OBJ_Security */
0x2B,0x06,0x01,0x06,                         /* [2855] OBJ_SNMPv2 */
0x2B,0x06,0x01,0x07,                         /* [2859] OBJ_Mail */
0x2B,0x06,0x01,0x04,0x01,                    /* [2863] OBJ_Enterprises */
0x2B,0x06,0x01,0x04,0x01,0x8B,0x3A,0x82,0x58,/* [2868] OBJ_dcObject */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x19,/* [2877] OBJ_domainComponent
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x0D,/* [2887] OBJ_Domain */
0x55,0x01,0x05,                              /* [2897] OBJ_selected_attribute_ty
0x55,0x01,0x05,0x37,                         /* [2900] OBJ_clearance */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x03,/* [2904] OBJ_md4WithRSAEncryption
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x0A,     /* [2913] OBJ_ac_proxying */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x0B,     /* [2921] OBJ_sinfo_access */
0x2B,0x06,0x01,0x05,0x05,0x07,0x0A,0x06,     /* [2929] OBJ_id_aca_encAttrs */
0x55,0x04,0x48,                              /* [2937] OBJ_role */
0x55,0x1D,0x24,                              /* [2940] OBJ_policy_constraints */
0x55,0x1D,0x37,                              /* [2943] OBJ_target_information */
0x55,0x1D,0x38,                              /* [2946] OBJ_no_rev_avail */
0x2A,0x86,0x48,0xCE,0x3D,                    /* [2949] OBJ_ansi_X9_62 */
0x2A,0x86,0x48,0xCE,0x3D,0x01,0x01,          /* [2954] OBJ_X9_62_prime_field */
0x2A,0x86,0x48,0xCE,0x3D,0x01,0x02,          /* [2961] OBJ_X9_62_characteristic_
0x2A,0x86,0x48,0xCE,0x3D,0x02,0x01,          /* [2968] OBJ_X9_62_id_ecPublicKey
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x01,0x01,     /* [2975] OBJ_X9_62_prime192v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x01,0x02,     /* [2983] OBJ_X9_62_prime192v2 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x01,0x03,     /* [2991] OBJ_X9_62_prime192v3 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x01,0x04,     /* [2999] OBJ_X9_62_prime239v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x01,0x05,     /* [3007] OBJ_X9_62_prime239v2 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x01,0x06,     /* [3015] OBJ_X9_62_prime239v3 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x01,0x07,     /* [3023] OBJ_X9_62_prime256v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x04,0x01,          /* [3031] OBJ_ecdsa_with_SHA1 */
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x11,0x01,/* [3038] OBJ_ms_csp_name */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x01,/* [3047] OBJ_aes_128_ecb */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x02,/* [3056] OBJ_aes_128_cbc */
```



```
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x03,/* [3065] OBJ_aes_128_ofb128 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x04,/* [3074] OBJ_aes_128_cfb128 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x15,/* [3083] OBJ_aes_192_ecb */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x16,/* [3092] OBJ_aes_192_cbc */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x17,/* [3101] OBJ_aes_192_ofb128 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x18,/* [3110] OBJ_aes_192_cfb128 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x29,/* [3119] OBJ_aes_256_ecb */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x2A,/* [3128] OBJ_aes_256_cbc */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x2B,/* [3137] OBJ_aes_256_ofb128 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x2C,/* [3146] OBJ_aes_256_cfb128 */
0x55,0x1D,0x17,                              /* [3155] OBJ_hold_instruction_code
0x2A,0x86,0x48,0xCE,0x38,0x02,0x01,          /* [3158] OBJ_hold_instruction_none
0x2A,0x86,0x48,0xCE,0x38,0x02,0x02,          /* [3165] OBJ_hold_instruction_call
0x2A,0x86,0x48,0xCE,0x38,0x02,0x03,          /* [3172] OBJ_hold_instruction_reje
0x09,                                        /* [3179] OBJ_data */
0x09,0x92,0x26,                              /* [3180] OBJ_pss */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,          /* [3183] OBJ_ucl */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,     /* [3190] OBJ_pilot */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,/* [3198] OBJ_pilotAttributeType */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x03,/* [3207] OBJ_pilotAttributeSyntax
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,/* [3216] OBJ_pilotObjectClass */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x0A,/* [3225] OBJ_pilotGroups */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x03,0x04,/* [3234] OBJ_iA5StringSyntax
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x03,0x05,/* [3244] OBJ_caseIgnoreIA5Str
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x03,/* [3254] OBJ_pilotObject */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x04,/* [3264] OBJ_pilotPerson */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x05,/* [3274] OBJ_account */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x06,/* [3284] OBJ_document */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x07,/* [3294] OBJ_room */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x09,/* [3304] OBJ_documentSeries *
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x0E,/* [3314] OBJ_rFC822localPart
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x0F,/* [3324] OBJ_dNSDomain */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x11,/* [3334] OBJ_domainRelatedObj
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x12,/* [3344] OBJ_friendlyCountry
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x13,/* [3354] OBJ_simpleSecurityOb
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x14,/* [3364] OBJ_pilotOrganizatio
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x15,/* [3374] OBJ_pilotDSA */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x04,0x16,/* [3384] OBJ_qualityLabelledD
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x01,/* [3394] OBJ_userId */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x02,/* [3404] OBJ_textEncodedORAdd
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x03,/* [3414] OBJ_rfc822Mailbox */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x04,/* [3424] OBJ_info */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x05,/* [3434] OBJ_favouriteDrink *
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x06,/* [3444] OBJ_roomNumber */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x07,/* [3454] OBJ_photo */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x08,/* [3464] OBJ_userClass */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x09,/* [3474] OBJ_host */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x0A,/* [3484] OBJ_manager */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x0B,/* [3494] OBJ_documentIdentifi
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x0C,/* [3504] OBJ_documentTitle */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x0D,/* [3514] OBJ_documentVersion
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x0E,/* [3524] OBJ_documentAuthor *
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x0F,/* [3534] OBJ_documentLocation
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x14,/* [3544] OBJ_homeTelephoneNum
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x15,/* [3554] OBJ_secretary */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x16,/* [3564] OBJ_otherMailbox */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x17,/* [3574] OBJ_lastModifiedTime
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x18,/* [3584] OBJ_lastModifiedBy *
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x1A,/* [3594] OBJ_aRecord */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x1B,/* [3604] OBJ_pilotAttributeTy
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x1C,/* [3614] OBJ_mXRecord */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x1D,/* [3624] OBJ_nSRecord */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x1E,/* [3634] OBJ_sOARecord */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x1F,/* [3644] OBJ_cNAMERecord */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x25,/* [3654] OBJ_associatedDomain
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x26,/* [3664] OBJ_associatedName *
```

```
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x27,/* [3674] OBJ_homePostalAddres
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x28,/* [3684] OBJ_personalTitle */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x29,/* [3694] OBJ_mobileTelephoneN
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x2A,/* [3704] OBJ_pagerTelephoneNu
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x2B,/* [3714] OBJ_friendlyCountryN
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x2D,/* [3724] OBJ_organizationalSt
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x2E,/* [3734] OBJ_janetMailbox */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x2F,/* [3744] OBJ_mailPreferenceOp
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x30,/* [3754] OBJ_buildingName */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x31,/* [3764] OBJ_dSAQuality */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x32,/* [3774] OBJ_singleLevelQuali
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x33,/* [3784] OBJ_subtreeMinimumQu
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x34,/* [3794] OBJ_subtreeMaximumQu
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x35,/* [3804] OBJ_personalSignatur
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x36,/* [3814] OBJ_dITRedirect */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x37,/* [3824] OBJ_audio */
0x09,0x92,0x26,0x89,0x93,0xF2,0x2C,0x64,0x01,0x38,/* [3834] OBJ_documentPublishe
0x55,0x04,0x2D,                         /* [3844] OBJ_x500UniqueIdentifier
0x2B,0x06,0x01,0x07,0x01,               /* [3847] OBJ_mime_mhs */
0x2B,0x06,0x01,0x07,0x01,0x01,          /* [3852] OBJ_mime_mhs_headings */
0x2B,0x06,0x01,0x07,0x01,0x02,          /* [3858] OBJ_mime_mhs_bodies */
0x2B,0x06,0x01,0x07,0x01,0x01,0x01,     /* [3864] OBJ_id_hex_partial_messag
0x2B,0x06,0x01,0x07,0x01,0x01,0x02,     /* [3871] OBJ_id_hex_multipart_mess
0x55,0x04,0x2C,                         /* [3878] OBJ_generationQualifier *
0x55,0x04,0x41,                         /* [3881] OBJ_pseudonym */
0x67,0x2A,                              /* [3884] OBJ_id_set */
0x67,0x2A,0x00,                         /* [3886] OBJ_set_ctype */
0x67,0x2A,0x01,                         /* [3889] OBJ_set_msgExt */
0x67,0x2A,0x03,                         /* [3892] OBJ_set_attr */
0x67,0x2A,0x05,                         /* [3895] OBJ_set_policy */
0x67,0x2A,0x07,                         /* [3898] OBJ_set_certExt */
0x67,0x2A,0x08,                         /* [3901] OBJ_set_brand */
0x67,0x2A,0x00,0x00,                    /* [3904] OBJ_setct_PANData */
0x67,0x2A,0x00,0x01,                    /* [3908] OBJ_setct_PANToken */
0x67,0x2A,0x00,0x02,                    /* [3912] OBJ_setct_PANOnly */
0x67,0x2A,0x00,0x03,                    /* [3916] OBJ_setct_OIData */
0x67,0x2A,0x00,0x04,                    /* [3920] OBJ_setct_PI */
0x67,0x2A,0x00,0x05,                    /* [3924] OBJ_setct_PIData */
0x67,0x2A,0x00,0x06,                    /* [3928] OBJ_setct_PIDataUnsigned
0x67,0x2A,0x00,0x07,                    /* [3932] OBJ_setct_HODInput */
0x67,0x2A,0x00,0x08,                    /* [3936] OBJ_setct_AuthResBaggage
0x67,0x2A,0x00,0x09,                    /* [3940] OBJ_setct_AuthRevReqBagga
0x67,0x2A,0x00,0x0A,                    /* [3944] OBJ_setct_AuthRevResBagga
0x67,0x2A,0x00,0x0B,                    /* [3948] OBJ_setct_CapTokenSeq */
0x67,0x2A,0x00,0x0C,                    /* [3952] OBJ_setct_PInitResData */
0x67,0x2A,0x00,0x0D,                    /* [3956] OBJ_setct_PI_TBS */
0x67,0x2A,0x00,0x0E,                    /* [3960] OBJ_setct_PResData */
0x67,0x2A,0x00,0x10,                    /* [3964] OBJ_setct_AuthReqTBS */
0x67,0x2A,0x00,0x11,                    /* [3968] OBJ_setct_AuthResTBS */
0x67,0x2A,0x00,0x12,                    /* [3972] OBJ_setct_AuthResTBSX */
0x67,0x2A,0x00,0x13,                    /* [3976] OBJ_setct_AuthTokenTBS */
0x67,0x2A,0x00,0x14,                    /* [3980] OBJ_setct_CapTokenData */
0x67,0x2A,0x00,0x15,                    /* [3984] OBJ_setct_CapTokenTBS */
0x67,0x2A,0x00,0x16,                    /* [3988] OBJ_setct_AcqCardCodeMsg
0x67,0x2A,0x00,0x17,                    /* [3992] OBJ_setct_AuthRevReqTBS *
0x67,0x2A,0x00,0x18,                    /* [3996] OBJ_setct_AuthRevResData
0x67,0x2A,0x00,0x19,                    /* [4000] OBJ_setct_AuthRevResTBS *
0x67,0x2A,0x00,0x1A,                    /* [4004] OBJ_setct_CapReqTBS */
0x67,0x2A,0x00,0x1B,                    /* [4008] OBJ_setct_CapReqTBSX */
0x67,0x2A,0x00,0x1C,                    /* [4012] OBJ_setct_CapResData */
0x67,0x2A,0x00,0x1D,                    /* [4016] OBJ_setct_CapRevReqTBS */
0x67,0x2A,0x00,0x1E,                    /* [4020] OBJ_setct_CapRevReqTBSX *
0x67,0x2A,0x00,0x1F,                    /* [4024] OBJ_setct_CapRevResData *
0x67,0x2A,0x00,0x20,                    /* [4028] OBJ_setct_CredReqTBS */
0x67,0x2A,0x00,0x21,                    /* [4032] OBJ_setct_CredReqTBSX */
0x67,0x2A,0x00,0x22,                    /* [4036] OBJ_setct_CredResData */
```



```
0x67,0x2A,0x00,0x23,                    /* [4040] OBJ_setct_CredRevReqTBS *
0x67,0x2A,0x00,0x24,                    /* [4044] OBJ_setct_CredRevReqTBSX
0x67,0x2A,0x00,0x25,                    /* [4048] OBJ_setct_CredRevResData
0x67,0x2A,0x00,0x26,                    /* [4052] OBJ_setct_PCertReqData */
0x67,0x2A,0x00,0x27,                    /* [4056] OBJ_setct_PCertResTBS */
0x67,0x2A,0x00,0x28,                    /* [4060] OBJ_setct_BatchAdminReqDa
0x67,0x2A,0x00,0x29,                    /* [4064] OBJ_setct_BatchAdminResDa
0x67,0x2A,0x00,0x2A,                    /* [4068] OBJ_setct_CardCInitResTBS
0x67,0x2A,0x00,0x2B,                    /* [4072] OBJ_setct_MeAqCInitResTBS
0x67,0x2A,0x00,0x2C,                    /* [4076] OBJ_setct_RegFormResTBS *
0x67,0x2A,0x00,0x2D,                    /* [4080] OBJ_setct_CertReqData */
0x67,0x2A,0x00,0x2E,                    /* [4084] OBJ_setct_CertReqTBS */
0x67,0x2A,0x00,0x2F,                    /* [4088] OBJ_setct_CertResData */
0x67,0x2A,0x00,0x30,                    /* [4092] OBJ_setct_CertInqReqTBS *
0x67,0x2A,0x00,0x31,                    /* [4096] OBJ_setct_ErrorTBS */
0x67,0x2A,0x00,0x32,                    /* [4100] OBJ_setct_PIDualSignedTBE
0x67,0x2A,0x00,0x33,                    /* [4104] OBJ_setct_PIUnsignedTBE *
0x67,0x2A,0x00,0x34,                    /* [4108] OBJ_setct_AuthReqTBE */
0x67,0x2A,0x00,0x35,                    /* [4112] OBJ_setct_AuthResTBE */
0x67,0x2A,0x00,0x36,                    /* [4116] OBJ_setct_AuthResTBEX */
0x67,0x2A,0x00,0x37,                    /* [4120] OBJ_setct_AuthTokenTBE */
0x67,0x2A,0x00,0x38,                    /* [4124] OBJ_setct_CapTokenTBE */
0x67,0x2A,0x00,0x39,                    /* [4128] OBJ_setct_CapTokenTBEX */
0x67,0x2A,0x00,0x3A,                    /* [4132] OBJ_setct_AcqCardCodeMsgT
0x67,0x2A,0x00,0x3B,                    /* [4136] OBJ_setct_AuthRevReqTBE *
0x67,0x2A,0x00,0x3C,                    /* [4140] OBJ_setct_AuthRevResTBE *
0x67,0x2A,0x00,0x3D,                    /* [4144] OBJ_setct_AuthRevResTBEB
0x67,0x2A,0x00,0x3E,                    /* [4148] OBJ_setct_CapReqTBE */
0x67,0x2A,0x00,0x3F,                    /* [4152] OBJ_setct_CapReqTBEX */
0x67,0x2A,0x00,0x40,                    /* [4156] OBJ_setct_CapResTBE */
0x67,0x2A,0x00,0x41,                    /* [4160] OBJ_setct_CapRevReqTBE */
0x67,0x2A,0x00,0x42,                    /* [4164] OBJ_setct_CapRevReqTBEX *
0x67,0x2A,0x00,0x43,                    /* [4168] OBJ_setct_CapRevResTBE */
0x67,0x2A,0x00,0x44,                    /* [4172] OBJ_setct_CredReqTBE */
0x67,0x2A,0x00,0x45,                    /* [4176] OBJ_setct_CredReqTBEX */
0x67,0x2A,0x00,0x46,                    /* [4180] OBJ_setct_CredResTBE */
0x67,0x2A,0x00,0x47,                    /* [4184] OBJ_setct_CredRevReqTBE *
0x67,0x2A,0x00,0x48,                    /* [4188] OBJ_setct_CredRevReqTBEX
0x67,0x2A,0x00,0x49,                    /* [4192] OBJ_setct_CredRevResTBE *
0x67,0x2A,0x00,0x4A,                    /* [4196] OBJ_setct_BatchAdminReqTB
0x67,0x2A,0x00,0x4B,                    /* [4200] OBJ_setct_BatchAdminResTB
0x67,0x2A,0x00,0x4C,                    /* [4204] OBJ_setct_RegFormReqTBE *
0x67,0x2A,0x00,0x4D,                    /* [4208] OBJ_setct_CertReqTBE */
0x67,0x2A,0x00,0x4E,                    /* [4212] OBJ_setct_CertReqTBEX */
0x67,0x2A,0x00,0x4F,                    /* [4216] OBJ_setct_CertResTBE */
0x67,0x2A,0x00,0x50,                    /* [4220] OBJ_setct_CRLNotification
0x67,0x2A,0x00,0x51,                    /* [4224] OBJ_setct_CRLNotification
0x67,0x2A,0x00,0x52,                    /* [4228] OBJ_setct_BCIDistribution
0x67,0x2A,0x01,0x01,                    /* [4232] OBJ_setext_genCrypt */
0x67,0x2A,0x01,0x03,                    /* [4236] OBJ_setext_miAuth */
0x67,0x2A,0x01,0x04,                    /* [4240] OBJ_setext_pinSecure */
0x67,0x2A,0x01,0x05,                    /* [4244] OBJ_setext_pinAny */
0x67,0x2A,0x01,0x07,                    /* [4248] OBJ_setext_track2 */
0x67,0x2A,0x01,0x08,                    /* [4252] OBJ_setext_cv */
0x67,0x2A,0x05,0x00,                    /* [4256] OBJ_set_policy_root */
0x67,0x2A,0x07,0x00,                    /* [4260] OBJ_setCext_hashedRoot */
0x67,0x2A,0x07,0x01,                    /* [4264] OBJ_setCext_certType */
0x67,0x2A,0x07,0x02,                    /* [4268] OBJ_setCext_merchData */
0x67,0x2A,0x07,0x03,                    /* [4272] OBJ_setCext_cCertRequired
0x67,0x2A,0x07,0x04,                    /* [4276] OBJ_setCext_tunneling */
0x67,0x2A,0x07,0x05,                    /* [4280] OBJ_setCext_setExt */
0x67,0x2A,0x07,0x06,                    /* [4284] OBJ_setCext_setQualf */
0x67,0x2A,0x07,0x07,                    /* [4288] OBJ_setCext_PGWYcapabilit
0x67,0x2A,0x07,0x08,                    /* [4292] OBJ_setCext_TokenIdentifi
0x67,0x2A,0x07,0x09,                    /* [4296] OBJ_setCext_Track2Data */
0x67,0x2A,0x07,0x0A,                    /* [4300] OBJ_setCext_TokenType */
```

```
0x67,0x2A,0x07,0x0B,                        /* [4304] OBJ_setCext_IssuerCapabil
0x67,0x2A,0x03,0x00,                        /* [4308] OBJ_setAttr_Cert */
0x67,0x2A,0x03,0x01,                        /* [4312] OBJ_setAttr_PGWYcap */
0x67,0x2A,0x03,0x02,                        /* [4316] OBJ_setAttr_TokenType */
0x67,0x2A,0x03,0x03,                        /* [4320] OBJ_setAttr_IssCap */
0x67,0x2A,0x03,0x00,0x00,                   /* [4324] OBJ_set_rootKeyThumb */
0x67,0x2A,0x03,0x00,0x01,                   /* [4329] OBJ_set_addPolicy */
0x67,0x2A,0x03,0x02,0x01,                   /* [4334] OBJ_setAttr_Token_EMV */
0x67,0x2A,0x03,0x02,0x02,                   /* [4339] OBJ_setAttr_Token_B0Prime
0x67,0x2A,0x03,0x03,0x03,                   /* [4344] OBJ_setAttr_IssCap_CVM */
0x67,0x2A,0x03,0x03,0x04,                   /* [4349] OBJ_setAttr_IssCap_T2 */
0x67,0x2A,0x03,0x03,0x05,                   /* [4354] OBJ_setAttr_IssCap_Sig */
0x67,0x2A,0x03,0x03,0x03,0x01,              /* [4359] OBJ_setAttr_GenCryptgrm *
0x67,0x2A,0x03,0x03,0x04,0x01,              /* [4365] OBJ_setAttr_T2Enc */
0x67,0x2A,0x03,0x03,0x04,0x02,              /* [4371] OBJ_setAttr_T2cleartxt */
0x67,0x2A,0x03,0x03,0x05,0x01,              /* [4377] OBJ_setAttr_TokICCsig */
0x67,0x2A,0x03,0x03,0x05,0x02,              /* [4383] OBJ_setAttr_SecDevSig */
0x67,0x2A,0x08,0x01,                        /* [4389] OBJ_set_brand_IATA_ATA */
0x67,0x2A,0x08,0x1E,                        /* [4393] OBJ_set_brand_Diners */
0x67,0x2A,0x08,0x22,                        /* [4397] OBJ_set_brand_AmericanExp
0x67,0x2A,0x08,0x23,                        /* [4401] OBJ_set_brand_JCB */
0x67,0x2A,0x08,0x04,                        /* [4405] OBJ_set_brand_Visa */
0x67,0x2A,0x08,0x05,                        /* [4409] OBJ_set_brand_MasterCard
0x67,0x2A,0x08,0xAE,0x7B,                   /* [4413] OBJ_set_brand_Novus */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x03,0x0A,    /* [4418] OBJ_des_cdmf */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x06,/* [4426] OBJ_rsaOAEPEncryptionSET
0x67,                                       /* [4435] OBJ_international_organiz
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x14,0x02,0x02,/* [4436] OBJ_ms_smartcard_log
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x14,0x02,0x03,/* [4446] OBJ_ms_upn */
0x55,0x04,0x09,                             /* [4456] OBJ_streetAddress */
0x55,0x04,0x11,                             /* [4459] OBJ_postalCode */
0x2B,0x06,0x01,0x05,0x05,0x07,0x15,         /* [4462] OBJ_id_ppl */
0x2B,0x06,0x01,0x05,0x05,0x07,0x01,0x0E,    /* [4469] OBJ_proxyCertInfo */
0x2B,0x06,0x01,0x05,0x05,0x07,0x15,0x00,    /* [4477] OBJ_id_ppl_anyLanguage */
0x2B,0x06,0x01,0x05,0x05,0x07,0x15,0x01,    /* [4485] OBJ_id_ppl_inheritAll */
0x55,0x1D,0x1E,                             /* [4493] OBJ_name_constraints */
0x2B,0x06,0x01,0x05,0x05,0x07,0x15,0x02,    /* [4496] OBJ_Independent */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x0B,/* [4504] OBJ_sha256WithRSAEncrypti
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x0C,/* [4513] OBJ_sha384WithRSAEncrypti
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x0D,/* [4522] OBJ_sha512WithRSAEncrypti
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x0E,/* [4531] OBJ_sha224WithRSAEncrypti
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x02,0x01,/* [4540] OBJ_sha256 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x02,0x02,/* [4549] OBJ_sha384 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x02,0x03,/* [4558] OBJ_sha512 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x02,0x04,/* [4567] OBJ_sha224 */
0x2B,                                       /* [4576] OBJ_identified_organizati
0x2B,0x81,0x04,                             /* [4577] OBJ_certicom_arc */
0x67,0x2B,                                  /* [4580] OBJ_wap */
0x67,0x2B,0x01,                             /* [4582] OBJ_wap_wsg */
0x2A,0x86,0x48,0xCE,0x3D,0x01,0x02,0x03,    /* [4585] OBJ_X9_62_id_characterist
0x2A,0x86,0x48,0xCE,0x3D,0x01,0x02,0x03,0x01,/* [4593] OBJ_X9_62_onBasis */
0x2A,0x86,0x48,0xCE,0x3D,0x01,0x02,0x03,0x02,/* [4602] OBJ_X9_62_tpBasis */
0x2A,0x86,0x48,0xCE,0x3D,0x01,0x02,0x03,0x03,/* [4611] OBJ_X9_62_ppBasis */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x01,    /* [4620] OBJ_X9_62_c2pnb163v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x02,    /* [4628] OBJ_X9_62_c2pnb163v2 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x03,    /* [4636] OBJ_X9_62_c2pnb163v3 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x04,    /* [4644] OBJ_X9_62_c2pnb176v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x05,    /* [4652] OBJ_X9_62_c2tnb191v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x06,    /* [4660] OBJ_X9_62_c2tnb191v2 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x07,    /* [4668] OBJ_X9_62_c2tnb191v3 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x08,    /* [4676] OBJ_X9_62_c2onb191v4 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x09,    /* [4684] OBJ_X9_62_c2onb191v5 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x0A,    /* [4692] OBJ_X9_62_c2pnb208w1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x0B,    /* [4700] OBJ_X9_62_c2tnb239v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x0C,    /* [4708] OBJ_X9_62_c2tnb239v2 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x0D,    /* [4716] OBJ_X9_62_c2tnb239v3 */
```



```
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x0E,    /* [4724] OBJ_X9_62_c2onb239v4 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x0F,    /* [4732] OBJ_X9_62_c2onb239v5 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x10,    /* [4740] OBJ_X9_62_c2pnb272w1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x11,    /* [4748] OBJ_X9_62_c2pnb304w1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x12,    /* [4756] OBJ_X9_62_c2tnb359v1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x13,    /* [4764] OBJ_X9_62_c2pnb368w1 */
0x2A,0x86,0x48,0xCE,0x3D,0x03,0x00,0x14,    /* [4772] OBJ_X9_62_c2tnb431r1 */
0x2B,0x81,0x04,0x00,0x06,                   /* [4780] OBJ_secp112r1 */
0x2B,0x81,0x04,0x00,0x07,                   /* [4785] OBJ_secp112r2 */
0x2B,0x81,0x04,0x00,0x1C,                   /* [4790] OBJ_secp128r1 */
0x2B,0x81,0x04,0x00,0x1D,                   /* [4795] OBJ_secp128r2 */
0x2B,0x81,0x04,0x00,0x09,                   /* [4800] OBJ_secp160k1 */
0x2B,0x81,0x04,0x00,0x08,                   /* [4805] OBJ_secp160r1 */
0x2B,0x81,0x04,0x00,0x1E,                   /* [4810] OBJ_secp160r2 */
0x2B,0x81,0x04,0x00,0x1F,                   /* [4815] OBJ_secp192k1 */
0x2B,0x81,0x04,0x00,0x20,                   /* [4820] OBJ_secp224k1 */
0x2B,0x81,0x04,0x00,0x21,                   /* [4825] OBJ_secp224r1 */
0x2B,0x81,0x04,0x00,0x0A,                   /* [4830] OBJ_secp256k1 */
0x2B,0x81,0x04,0x00,0x22,                   /* [4835] OBJ_secp384r1 */
0x2B,0x81,0x04,0x00,0x23,                   /* [4840] OBJ_secp521r1 */
0x2B,0x81,0x04,0x00,0x04,                   /* [4845] OBJ_sect113r1 */
0x2B,0x81,0x04,0x00,0x05,                   /* [4850] OBJ_sect113r2 */
0x2B,0x81,0x04,0x00,0x16,                   /* [4855] OBJ_sect131r1 */
0x2B,0x81,0x04,0x00,0x17,                   /* [4860] OBJ_sect131r2 */
0x2B,0x81,0x04,0x00,0x01,                   /* [4865] OBJ_sect163k1 */
0x2B,0x81,0x04,0x00,0x02,                   /* [4870] OBJ_sect163r1 */
0x2B,0x81,0x04,0x00,0x0F,                   /* [4875] OBJ_sect163r2 */
0x2B,0x81,0x04,0x00,0x18,                   /* [4880] OBJ_sect193r1 */
0x2B,0x81,0x04,0x00,0x19,                   /* [4885] OBJ_sect193r2 */
0x2B,0x81,0x04,0x00,0x1A,                   /* [4890] OBJ_sect233k1 */
0x2B,0x81,0x04,0x00,0x1B,                   /* [4895] OBJ_sect233r1 */
0x2B,0x81,0x04,0x00,0x03,                   /* [4900] OBJ_sect239k1 */
0x2B,0x81,0x04,0x00,0x10,                   /* [4905] OBJ_sect283k1 */
0x2B,0x81,0x04,0x00,0x11,                   /* [4910] OBJ_sect283r1 */
0x2B,0x81,0x04,0x00,0x24,                   /* [4915] OBJ_sect409k1 */
0x2B,0x81,0x04,0x00,0x25,                   /* [4920] OBJ_sect409r1 */
0x2B,0x81,0x04,0x00,0x26,                   /* [4925] OBJ_sect571k1 */
0x2B,0x81,0x04,0x00,0x27,                   /* [4930] OBJ_sect571r1 */
0x67,0x2B,0x01,0x04,0x01,                   /* [4935] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x03,                   /* [4940] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x04,                   /* [4945] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x05,                   /* [4950] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x06,                   /* [4955] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x07,                   /* [4960] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x08,                   /* [4965] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x09,                   /* [4970] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x0A,                   /* [4975] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x0B,                   /* [4980] OBJ_wap_wsg_idm_ecid_wtls
0x67,0x2B,0x01,0x04,0x0C,                   /* [4985] OBJ_wap_wsg_idm_ecid_wtls
0x55,0x1D,0x20,0x00,                        /* [4990] OBJ_any_policy */
0x55,0x1D,0x21,                             /* [4994] OBJ_policy_mappings */
0x55,0x1D,0x36,                             /* [4997] OBJ_inhibit_any_policy */
0x2A,0x83,0x08,0x8C,0x9A,0x4B,0x3D,0x01,0x01,0x01,0x02,/* [5000] OBJ_camellia_12
0x2A,0x83,0x08,0x8C,0x9A,0x4B,0x3D,0x01,0x01,0x01,0x03,/* [5011] OBJ_camellia_19
0x2A,0x83,0x08,0x8C,0x9A,0x4B,0x3D,0x01,0x01,0x01,0x04,/* [5022] OBJ_camellia_25
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x01,    /* [5033] OBJ_camellia_128_ecb */
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x15,    /* [5041] OBJ_camellia_192_ecb */
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x29,    /* [5049] OBJ_camellia_256_ecb */
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x04,    /* [5057] OBJ_camellia_128_cfb128 *
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x18,    /* [5065] OBJ_camellia_192_cfb128 *
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x2C,    /* [5073] OBJ_camellia_256_cfb128 *
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x03,    /* [5081] OBJ_camellia_128_ofb128 *
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x17,    /* [5089] OBJ_camellia_192_ofb128 *
0x03,0xA2,0x31,0x05,0x03,0x01,0x09,0x2B,    /* [5097] OBJ_camellia_256_ofb128 *
0x55,0x1D,0x09,                             /* [5105] OBJ_subject_directory_att
0x55,0x1D,0x1C,                             /* [5108] OBJ_issuing_distribution_
```

```
0x55,0x1D,0x1D,                         /* [5111] OBJ_certificate_issuer */
0x2A,0x83,0x1A,0x8C,0x9A,0x44,          /* [5114] OBJ_kisa */
0x2A,0x83,0x1A,0x8C,0x9A,0x44,0x01,0x03,     /* [5120] OBJ_seed_ecb */
0x2A,0x83,0x1A,0x8C,0x9A,0x44,0x01,0x04,     /* [5128] OBJ_seed_cbc */
0x2A,0x83,0x1A,0x8C,0x9A,0x44,0x01,0x06,     /* [5136] OBJ_seed_ofb128 */
0x2A,0x83,0x1A,0x8C,0x9A,0x44,0x01,0x05,     /* [5144] OBJ_seed_cfb128 */
0x2B,0x06,0x01,0x05,0x05,0x08,0x01,0x01,     /* [5152] OBJ_hmac_md5 */
0x2B,0x06,0x01,0x05,0x05,0x08,0x01,0x02,     /* [5160] OBJ_hmac_sha1 */
0x2A,0x86,0x48,0x86,0xF6,0x7D,0x07,0x42,0x0D,/* [5168] OBJ_id_PasswordBasedMAC *
0x2A,0x86,0x48,0x86,0xF6,0x7D,0x07,0x42,0x1E,/* [5177] OBJ_id_DHBasedMac */
0x2B,0x06,0x01,0x05,0x05,0x07,0x04,0x10,     /* [5186] OBJ_id_it_suppLangTags */
0x2B,0x06,0x01,0x05,0x05,0x07,0x30,0x05,     /* [5194] OBJ_caRepository */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x09,/* [5202] OBJ_id_smime_ct
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x01,0x1B,/* [5213] OBJ_id_ct_ascii
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x05,/* [5224] OBJ_id_aes128_wrap */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x19,/* [5233] OBJ_id_aes192_wrap */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x2D,/* [5242] OBJ_id_aes256_wrap */
0x2A,0x86,0x48,0xCE,0x3D,0x04,0x02,          /* [5251] OBJ_ecdsa_with_Recommende
0x2A,0x86,0x48,0xCE,0x3D,0x04,0x03,          /* [5258] OBJ_ecdsa_with_Specified
0x2A,0x86,0x48,0xCE,0x3D,0x04,0x03,0x01,     /* [5265] OBJ_ecdsa_with_SHA224 */
0x2A,0x86,0x48,0xCE,0x3D,0x04,0x03,0x02,     /* [5273] OBJ_ecdsa_with_SHA256 */
0x2A,0x86,0x48,0xCE,0x3D,0x04,0x03,0x03,     /* [5281] OBJ_ecdsa_with_SHA384 */
0x2A,0x86,0x48,0xCE,0x3D,0x04,0x03,0x04,     /* [5289] OBJ_ecdsa_with_SHA512 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x06,     /* [5297] OBJ_hmacWithMD5 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x08,     /* [5305] OBJ_hmacWithSHA224 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x09,     /* [5313] OBJ_hmacWithSHA256 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x0A,     /* [5321] OBJ_hmacWithSHA384 */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x02,0x0B,     /* [5329] OBJ_hmacWithSHA512 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x03,0x01,/* [5337] OBJ_dsa_with_SHA224 */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x03,0x02,/* [5346] OBJ_dsa_with_SHA256 */
0x28,0xCF,0x06,0x03,0x00,0x37,          /* [5355] OBJ_whirlpool */
0x2A,0x85,0x03,0x02,0x02,               /* [5361] OBJ_cryptopro */
0x2A,0x85,0x03,0x02,0x09,               /* [5366] OBJ_cryptocom */
0x2A,0x85,0x03,0x02,0x02,0x03,          /* [5371] OBJ_id_GostR3411_94_with_
0x2A,0x85,0x03,0x02,0x02,0x04,          /* [5377] OBJ_id_GostR3411_94_with_
0x2A,0x85,0x03,0x02,0x02,0x09,          /* [5383] OBJ_id_GostR3411_94 */
0x2A,0x85,0x03,0x02,0x02,0x0A,          /* [5389] OBJ_id_HMACGostR3411_94 *
0x2A,0x85,0x03,0x02,0x02,0x13,          /* [5395] OBJ_id_GostR3410_2001 */
0x2A,0x85,0x03,0x02,0x02,0x14,          /* [5401] OBJ_id_GostR3410_94 */
0x2A,0x85,0x03,0x02,0x02,0x15,          /* [5407] OBJ_id_Gost28147_89 */
0x2A,0x85,0x03,0x02,0x02,0x16,          /* [5413] OBJ_id_Gost28147_89_MAC *
0x2A,0x85,0x03,0x02,0x02,0x17,          /* [5419] OBJ_id_GostR3411_94_prf *
0x2A,0x85,0x03,0x02,0x02,0x62,          /* [5425] OBJ_id_GostR3410_2001DH *
0x2A,0x85,0x03,0x02,0x02,0x63,          /* [5431] OBJ_id_GostR3410_94DH */
0x2A,0x85,0x03,0x02,0x02,0x0E,0x01,     /* [5437] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x0E,0x00,     /* [5444] OBJ_id_Gost28147_89_None_
0x2A,0x85,0x03,0x02,0x02,0x1E,0x00,     /* [5451] OBJ_id_GostR3411_94_TestP
0x2A,0x85,0x03,0x02,0x02,0x1E,0x01,     /* [5458] OBJ_id_GostR3411_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x1F,0x00,     /* [5465] OBJ_id_Gost28147_89_TestP
0x2A,0x85,0x03,0x02,0x02,0x1F,0x01,     /* [5472] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x1F,0x02,     /* [5479] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x1F,0x03,     /* [5486] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x1F,0x04,     /* [5493] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x1F,0x05,     /* [5500] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x1F,0x06,     /* [5507] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x1F,0x07,     /* [5514] OBJ_id_Gost28147_89_Crypt
0x2A,0x85,0x03,0x02,0x02,0x20,0x00,     /* [5521] OBJ_id_GostR3410_94_TestP
0x2A,0x85,0x03,0x02,0x02,0x20,0x02,     /* [5528] OBJ_id_GostR3410_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x20,0x03,     /* [5535] OBJ_id_GostR3410_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x20,0x04,     /* [5542] OBJ_id_GostR3410_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x20,0x05,     /* [5549] OBJ_id_GostR3410_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x21,0x01,     /* [5556] OBJ_id_GostR3410_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x21,0x02,     /* [5563] OBJ_id_GostR3410_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x21,0x03,     /* [5570] OBJ_id_GostR3410_94_Crypt
0x2A,0x85,0x03,0x02,0x02,0x23,0x00,     /* [5577] OBJ_id_GostR3410_2001_Tes
0x2A,0x85,0x03,0x02,0x02,0x23,0x01,     /* [5584] OBJ_id_GostR3410_2001_Cry
```



```
0x2A,0x85,0x03,0x02,0x02,0x23,0x02,          /* [5591] OBJ_id_GostR3410_2001_Cry
0x2A,0x85,0x03,0x02,0x02,0x23,0x03,          /* [5598] OBJ_id_GostR3410_2001_Cry
0x2A,0x85,0x03,0x02,0x02,0x24,0x00,          /* [5605] OBJ_id_GostR3410_2001_Cry
0x2A,0x85,0x03,0x02,0x02,0x24,0x01,          /* [5612] OBJ_id_GostR3410_2001_Cry
0x2A,0x85,0x03,0x02,0x02,0x14,0x01,          /* [5619] OBJ_id_GostR3410_94_a */
0x2A,0x85,0x03,0x02,0x02,0x14,0x02,          /* [5626] OBJ_id_GostR3410_94_aBis
0x2A,0x85,0x03,0x02,0x02,0x14,0x03,          /* [5633] OBJ_id_GostR3410_94_b */
0x2A,0x85,0x03,0x02,0x02,0x14,0x04,          /* [5640] OBJ_id_GostR3410_94_bBis
0x2A,0x85,0x03,0x02,0x09,0x01,0x06,0x01,     /* [5647] OBJ_id_Gost28147_89_cc */
0x2A,0x85,0x03,0x02,0x09,0x01,0x05,0x03,     /* [5655] OBJ_id_GostR3410_94_cc */
0x2A,0x85,0x03,0x02,0x09,0x01,0x05,0x04,     /* [5663] OBJ_id_GostR3410_2001_cc
0x2A,0x85,0x03,0x02,0x09,0x01,0x03,0x03,     /* [5671] OBJ_id_GostR3411_94_with_
0x2A,0x85,0x03,0x02,0x09,0x01,0x03,0x04,     /* [5679] OBJ_id_GostR3411_94_with_
0x2A,0x85,0x03,0x02,0x09,0x01,0x08,0x01,     /* [5687] OBJ_id_GostR3410_2001_Par
0x2B,0x06,0x01,0x04,0x01,0x82,0x37,0x11,0x02,/* [5695] OBJ_LocalKeySet */
0x55,0x1D,0x2E,                              /* [5704] OBJ_freshest_crl */
0x2B,0x06,0x01,0x05,0x05,0x07,0x08,0x03,     /* [5707] OBJ_id_on_permanentIdenti
0x55,0x04,0x0E,                              /* [5715] OBJ_searchGuide */
0x55,0x04,0x0F,                              /* [5718] OBJ_businessCategory */
0x55,0x04,0x10,                              /* [5721] OBJ_postalAddress */
0x55,0x04,0x12,                              /* [5724] OBJ_postOfficeBox */
0x55,0x04,0x13,                              /* [5727] OBJ_physicalDeliveryOffic
0x55,0x04,0x14,                              /* [5730] OBJ_telephoneNumber */
0x55,0x04,0x15,                              /* [5733] OBJ_telexNumber */
0x55,0x04,0x16,                              /* [5736] OBJ_teletexTerminalIdenti
0x55,0x04,0x17,                              /* [5739] OBJ_facsimileTelephoneNum
0x55,0x04,0x18,                              /* [5742] OBJ_x121Address */
0x55,0x04,0x19,                              /* [5745] OBJ_internationaliSDNNumb
0x55,0x04,0x1A,                              /* [5748] OBJ_registeredAddress */
0x55,0x04,0x1B,                              /* [5751] OBJ_destinationIndicator
0x55,0x04,0x1C,                              /* [5754] OBJ_preferredDeliveryMeth
0x55,0x04,0x1D,                              /* [5757] OBJ_presentationAddress *
0x55,0x04,0x1E,                              /* [5760] OBJ_supportedApplicationC
0x55,0x04,0x1F,                              /* [5763] OBJ_member */
0x55,0x04,0x20,                              /* [5766] OBJ_owner */
0x55,0x04,0x21,                              /* [5769] OBJ_roleOccupant */
0x55,0x04,0x22,                              /* [5772] OBJ_seeAlso */
0x55,0x04,0x23,                              /* [5775] OBJ_userPassword */
0x55,0x04,0x24,                              /* [5778] OBJ_userCertificate */
0x55,0x04,0x25,                              /* [5781] OBJ_cACertificate */
0x55,0x04,0x26,                              /* [5784] OBJ_authorityRevocationLi
0x55,0x04,0x27,                              /* [5787] OBJ_certificateRevocation
0x55,0x04,0x28,                              /* [5790] OBJ_crossCertificatePair
0x55,0x04,0x2F,                              /* [5793] OBJ_enhancedSearchGuide *
0x55,0x04,0x30,                              /* [5796] OBJ_protocolInformation *
0x55,0x04,0x31,                              /* [5799] OBJ_distinguishedName */
0x55,0x04,0x32,                              /* [5802] OBJ_uniqueMember */
0x55,0x04,0x33,                              /* [5805] OBJ_houseIdentifier */
0x55,0x04,0x34,                              /* [5808] OBJ_supportedAlgorithms *
0x55,0x04,0x35,                              /* [5811] OBJ_deltaRevocationList *
0x55,0x04,0x36,                              /* [5814] OBJ_dmdName */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x09,0x10,0x03,0x09,/* [5817] OBJ_id_alg_PWRI
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x06,/* [5828] OBJ_aes_128_gcm */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x07,/* [5837] OBJ_aes_128_ccm */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x08,/* [5846] OBJ_id_aes128_wrap_pad */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x1A,/* [5855] OBJ_aes_192_gcm */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x1B,/* [5864] OBJ_aes_192_ccm */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x1C,/* [5873] OBJ_id_aes192_wrap_pad */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x2E,/* [5882] OBJ_aes_256_gcm */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x2F,/* [5891] OBJ_aes_256_ccm */
0x60,0x86,0x48,0x01,0x65,0x03,0x04,0x01,0x30,/* [5900] OBJ_id_aes256_wrap_pad */
0x2A,0x83,0x08,0x8C,0x9A,0x4B,0x3D,0x01,0x01,0x03,0x02,/* [5909] OBJ_id_camellia
0x2A,0x83,0x08,0x8C,0x9A,0x4B,0x3D,0x01,0x01,0x03,0x03,/* [5920] OBJ_id_camellia
0x2A,0x83,0x08,0x8C,0x9A,0x4B,0x3D,0x01,0x01,0x03,0x04,/* [5931] OBJ_id_camellia
0x55,0x1D,0x25,0x00,                         /* [5942] OBJ_anyExtendedKeyUsage *
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x08,/* [5946] OBJ_mgf1 */
```

```
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x0A,/* [5955] OBJ_rsassaPss */
0x2A,0x86,0x48,0x86,0xF7,0x0D,0x01,0x01,0x07,/* [5964] OBJ_rsaesOaep */
};

static const ASN1_OBJECT nid_objs[NUM_NID]={
{"UNDEF","undefined",NID_undef,0,NULL,0},
{"rsadsi","RSA Data Security, Inc.",NID_rsadsi,6,&(lvalues[0]),0},
{"pkcs","RSA Data Security, Inc. PKCS",NID_pkcs,7,&(lvalues[6]),0},
{"MD2","md2",NID_md2,8,&(lvalues[13]),0},
{"MD5","md5",NID_md5,8,&(lvalues[21]),0},
{"RC4","rc4",NID_rc4,8,&(lvalues[29]),0},
{"rsaEncryption","rsaEncryption",NID_rsaEncryption,9,&(lvalues[37]),0},
{"RSA-MD2","md2WithRSAEncryption",NID_md2WithRSAEncryption,9,
        &(lvalues[46]),0},
{"RSA-MD5","md5WithRSAEncryption",NID_md5WithRSAEncryption,9,
        &(lvalues[55]),0},
{"PBE-MD2-DES","pbeWithMD2AndDES-CBC",NID_pbeWithMD2AndDES_CBC,9,
        &(lvalues[64]),0},
{"PBE-MD5-DES","pbeWithMD5AndDES-CBC",NID_pbeWithMD5AndDES_CBC,9,
        &(lvalues[73]),0},
{"X500","directory services (X.500)",NID_X500,1,&(lvalues[82]),0},
{"X509","X509",NID_X509,2,&(lvalues[83]),0},
{"CN","commonName",NID_commonName,3,&(lvalues[85]),0},
{"C","countryName",NID_countryName,3,&(lvalues[88]),0},
{"L","localityName",NID_localityName,3,&(lvalues[91]),0},
{"ST","stateOrProvinceName",NID_stateOrProvinceName,3,&(lvalues[94]),0},
{"O","organizationName",NID_organizationName,3,&(lvalues[97]),0},
{"OU","organizationalUnitName",NID_organizationalUnitName,3,
        &(lvalues[100]),0},
{"RSA","rsa",NID_rsa,4,&(lvalues[103]),0},
{"pkcs7","pkcs7",NID_pkcs7,8,&(lvalues[107]),0},
{"pkcs7-data","pkcs7-data",NID_pkcs7_data,9,&(lvalues[115]),0},
{"pkcs7-signedData","pkcs7-signedData",NID_pkcs7_signed,9,
        &(lvalues[124]),0},
{"pkcs7-envelopedData","pkcs7-envelopedData",NID_pkcs7_enveloped,9,
        &(lvalues[133]),0},
{"pkcs7-signedAndEnvelopedData","pkcs7-signedAndEnvelopedData",
        NID_pkcs7_signedAndEnveloped,9,&(lvalues[142]),0},
{"pkcs7-digestData","pkcs7-digestData",NID_pkcs7_digest,9,
        &(lvalues[151]),0},
{"pkcs7-encryptedData","pkcs7-encryptedData",NID_pkcs7_encrypted,9,
        &(lvalues[160]),0},
{"pkcs3","pkcs3",NID_pkcs3,8,&(lvalues[169]),0},
{"dhKeyAgreement","dhKeyAgreement",NID_dhKeyAgreement,9,
        &(lvalues[177]),0},
{"DES-ECB","des-ecb",NID_des_ecb,5,&(lvalues[186]),0},
{"DES-CFB","des-cfb",NID_des_cfb64,5,&(lvalues[191]),0},
{"DES-CBC","des-cbc",NID_des_cbc,5,&(lvalues[196]),0},
{"DES-EDE","des-ede",NID_des_ede_ecb,5,&(lvalues[201]),0},
{"DES-EDE3","des-ede3",NID_des_ede3_ecb,0,NULL,0},
{"IDEA-CBC","idea-cbc",NID_idea_cbc,11,&(lvalues[206]),0},
{"IDEA-CFB","idea-cfb",NID_idea_cfb64,0,NULL,0},
{"IDEA-ECB","idea-ecb",NID_idea_ecb,0,NULL,0},
{"RC2-CBC","rc2-cbc",NID_rc2_cbc,8,&(lvalues[217]),0},
{"RC2-ECB","rc2-ecb",NID_rc2_ecb,0,NULL,0},
{"RC2-CFB","rc2-cfb",NID_rc2_cfb64,0,NULL,0},
{"RC2-OFB","rc2-ofb",NID_rc2_ofb64,0,NULL,0},
{"SHA","sha",NID_sha,5,&(lvalues[225]),0},
{"RSA-SHA","shaWithRSAEncryption",NID_shaWithRSAEncryption,5,
        &(lvalues[230]),0},
{"DES-EDE-CBC","des-ede-cbc",NID_des_ede_cbc,0,NULL,0},
{"DES-EDE3-CBC","des-ede3-cbc",NID_des_ede3_cbc,8,&(lvalues[235]),0},
{"DES-OFB","des-ofb",NID_des_ofb64,5,&(lvalues[243]),0},
{"IDEA-OFB","idea-ofb",NID_idea_ofb64,0,NULL,0},
{"pkcs9","pkcs9",NID_pkcs9,8,&(lvalues[248]),0},
{"emailAddress","emailAddress",NID_pkcs9_emailAddress,9,
```



```
        &(lvalues[256]),0},
{"unstructuredName","unstructuredName",NID_pkcs9_unstructuredName,9,
        &(lvalues[265]),0},
{"contentType","contentType",NID_pkcs9_contentType,9,&(lvalues[274]),0},
{"messageDigest","messageDigest",NID_pkcs9_messageDigest,9,
        &(lvalues[283]),0},
{"signingTime","signingTime",NID_pkcs9_signingTime,9,&(lvalues[292]),0},
{"countersignature","countersignature",NID_pkcs9_countersignature,9,
        &(lvalues[301]),0},
{"challengePassword","challengePassword",NID_pkcs9_challengePassword,
        9,&(lvalues[310]),0},
{"unstructuredAddress","unstructuredAddress",
        NID_pkcs9_unstructuredAddress,9,&(lvalues[319]),0},
{"extendedCertificateAttributes","extendedCertificateAttributes",
        NID_pkcs9_extCertAttributes,9,&(lvalues[328]),0},
{"Netscape","Netscape Communications Corp.",NID_netscape,7,
        &(lvalues[337]),0},
{"nsCertExt","Netscape Certificate Extension",
        NID_netscape_cert_extension,8,&(lvalues[344]),0},
{"nsDataType","Netscape Data Type",NID_netscape_data_type,8,
        &(lvalues[352]),0},
{"DES-EDE-CFB","des-ede-cfb",NID_des_ede_cfb64,0,NULL,0},
{"DES-EDE3-CFB","des-ede3-cfb",NID_des_ede3_cfb64,0,NULL,0},
{"DES-EDE-OFB","des-ede-ofb",NID_des_ede_ofb64,0,NULL,0},
{"DES-EDE3-OFB","des-ede3-ofb",NID_des_ede3_ofb64,0,NULL,0},
{"SHA1","sha1",NID_sha1,5,&(lvalues[360]),0},
{"RSA-SHA1","sha1WithRSAEncryption",NID_sha1WithRSAEncryption,9,
        &(lvalues[365]),0},
{"DSA-SHA","dsaWithSHA",NID_dsaWithSHA,5,&(lvalues[374]),0},
{"DSA-old","dsaEncryption-old",NID_dsa_2,5,&(lvalues[379]),0},
{"PBE-SHA1-RC2-64","pbeWithSHA1AndRC2-CBC",NID_pbeWithSHA1AndRC2_CBC,
        9,&(lvalues[384]),0},
{"PBKDF2","PBKDF2",NID_id_pbkdf2,9,&(lvalues[393]),0},
{"DSA-SHA1-old","dsaWithSHA1-old",NID_dsaWithSHA1_2,5,&(lvalues[402]),0},
{"nsCertType","Netscape Cert Type",NID_netscape_cert_type,9,
        &(lvalues[407]),0},
{"nsBaseUrl","Netscape Base Url",NID_netscape_base_url,9,
        &(lvalues[416]),0},
{"nsRevocationUrl","Netscape Revocation Url",
        NID_netscape_revocation_url,9,&(lvalues[425]),0},
{"nsCaRevocationUrl","Netscape CA Revocation Url",
        NID_netscape_ca_revocation_url,9,&(lvalues[434]),0},
{"nsRenewalUrl","Netscape Renewal Url",NID_netscape_renewal_url,9,
        &(lvalues[443]),0},
{"nsCaPolicyUrl","Netscape CA Policy Url",NID_netscape_ca_policy_url,
        9,&(lvalues[452]),0},
{"nsSslServerName","Netscape SSL Server Name",
        NID_netscape_ssl_server_name,9,&(lvalues[461]),0},
{"nsComment","Netscape Comment",NID_netscape_comment,9,&(lvalues[470]),0},
{"nsCertSequence","Netscape Certificate Sequence",
        NID_netscape_cert_sequence,9,&(lvalues[479]),0},
{"DESX-CBC","desx-cbc",NID_desx_cbc,0,NULL,0},
{"id-ce","id-ce",NID_id_ce,2,&(lvalues[488]),0},
{"subjectKeyIdentifier","X509v3 Subject Key Identifier",
        NID_subject_key_identifier,3,&(lvalues[490]),0},
{"keyUsage","X509v3 Key Usage",NID_key_usage,3,&(lvalues[493]),0},
{"privateKeyUsagePeriod","X509v3 Private Key Usage Period",
        NID_private_key_usage_period,3,&(lvalues[496]),0},
{"subjectAltName","X509v3 Subject Alternative Name",
        NID_subject_alt_name,3,&(lvalues[499]),0},
{"issuerAltName","X509v3 Issuer Alternative Name",NID_issuer_alt_name,
        3,&(lvalues[502]),0},
{"basicConstraints","X509v3 Basic Constraints",NID_basic_constraints,
        3,&(lvalues[505]),0},
{"crlNumber","X509v3 CRL Number",NID_crl_number,3,&(lvalues[508]),0},
{"certificatePolicies","X509v3 Certificate Policies",
```

```
1052         NID_certificate_policies,3,&(lvalues[511]),0},
1053 {"authorityKeyIdentifier","X509v3 Authority Key Identifier",
1054         NID_authority_key_identifier,3,&(lvalues[514]),0},
1055 {"BF-CBC","bf-cbc",NID_bf_cbc,9,&(lvalues[517]),0},
1056 {"BF-ECB","bf-ecb",NID_bf_ecb,0,NULL,0},
1057 {"BF-CFB","bf-cfb",NID_bf_cfb64,0,NULL,0},
1058 {"BF-OFB","bf-ofb",NID_bf_ofb64,0,NULL,0},
1059 {"MDC2","mdc2",NID_mdc2,4,&(lvalues[526]),0},
1060 {"RSA-MDC2","mdc2WithRSA",NID_mdc2WithRSA,4,&(lvalues[530]),0},
1061 {"RC4-40","rc4-40",NID_rc4_40,0,NULL,0},
1062 {"RC2-40-CBC","rc2-40-cbc",NID_rc2_40_cbc,0,NULL,0},
1063 {"GN","givenName",NID_givenName,3,&(lvalues[534]),0},
1064 {"SN","surname",NID_surname,3,&(lvalues[537]),0},
1065 {"initials","initials",NID_initials,3,&(lvalues[540]),0},
1066 {NULL,NULL,NID_undef,0,NULL,0},
1067 {"crlDistributionPoints","X509v3 CRL Distribution Points",
1068         NID_crl_distribution_points,3,&(lvalues[543]),0},
1069 {"RSA-NP-MD5","md5WithRSA",NID_md5WithRSA,5,&(lvalues[546]),0},
1070 {"serialNumber","serialNumber",NID_serialNumber,3,&(lvalues[551]),0},
1071 {"title","title",NID_title,3,&(lvalues[554]),0},
1072 {"description","description",NID_description,3,&(lvalues[557]),0},
1073 {"CAST5-CBC","cast5-cbc",NID_cast5_cbc,9,&(lvalues[560]),0},
1074 {"CAST5-ECB","cast5-ecb",NID_cast5_ecb,0,NULL,0},
1075 {"CAST5-CFB","cast5-cfb",NID_cast5_cfb64,0,NULL,0},
1076 {"CAST5-OFB","cast5-ofb",NID_cast5_ofb64,0,NULL,0},
1077 {"pbeWithMD5AndCast5CBC","pbeWithMD5AndCast5CBC",
1078         NID_pbeWithMD5AndCast5_CBC,9,&(lvalues[569]),0},
1079 {"DSA-SHA1","dsaWithSHA1",NID_dsaWithSHA1,7,&(lvalues[578]),0},
1080 {"MD5-SHA1","md5-sha1",NID_md5_sha1,0,NULL,0},
1081 {"RSA-SHA1-2","sha1WithRSA",NID_sha1WithRSA,5,&(lvalues[585]),0},
1082 {"DSA","dsaEncryption",NID_dsa,7,&(lvalues[590]),0},
1083 {"RIPEMD160","ripemd160",NID_ripemd160,5,&(lvalues[597]),0},
1084 {NULL,NULL,NID_undef,0,NULL,0},
1085 {"RSA-RIPEMD160","ripemd160WithRSA",NID_ripemd160WithRSA,6,
1086         &(lvalues[602]),0},
1087 {"RC5-CBC","rc5-cbc",NID_rc5_cbc,8,&(lvalues[608]),0},
1088 {"RC5-ECB","rc5-ecb",NID_rc5_ecb,0,NULL,0},
1089 {"RC5-CFB","rc5-cfb",NID_rc5_cfb64,0,NULL,0},
1090 {"RC5-OFB","rc5-ofb",NID_rc5_ofb64,0,NULL,0},
1091 {"RLE","run length compression",NID_rle_compression,6,&(lvalues[616]),0},
1092 {"ZLIB","zlib compression",NID_zlib_compression,11,&(lvalues[622]),0},
1093 {"extendedKeyUsage","X509v3 Extended Key Usage",NID_ext_key_usage,3,
1094         &(lvalues[633]),0},
1095 {"PKIX","PKIX",NID_id_pkix,6,&(lvalues[636]),0},
1096 {"id-kp","id-kp",NID_id_kp,7,&(lvalues[642]),0},
1097 {"serverAuth","TLS Web Server Authentication",NID_server_auth,8,
1098         &(lvalues[649]),0},
1099 {"clientAuth","TLS Web Client Authentication",NID_client_auth,8,
1100         &(lvalues[657]),0},
1101 {"codeSigning","Code Signing",NID_code_sign,8,&(lvalues[665]),0},
1102 {"emailProtection","E-mail Protection",NID_email_protect,8,
1103         &(lvalues[673]),0},
1104 {"timeStamping","Time Stamping",NID_time_stamp,8,&(lvalues[681]),0},
1105 {"msCodeInd","Microsoft Individual Code Signing",NID_ms_code_ind,10,
1106         &(lvalues[689]),0},
1107 {"msCodeCom","Microsoft Commercial Code Signing",NID_ms_code_com,10,
1108         &(lvalues[699]),0},
1109 {"msCTLSign","Microsoft Trust List Signing",NID_ms_ctl_sign,10,
1110         &(lvalues[709]),0},
1111 {"msSGC","Microsoft Server Gated Crypto",NID_ms_sgc,10,&(lvalues[719]),0},
1112 {"msEFS","Microsoft Encrypted File System",NID_ms_efs,10,
1113         &(lvalues[729]),0},
1114 {"nsSGC","Netscape Server Gated Crypto",NID_ns_sgc,9,&(lvalues[739]),0},
1115 {"deltaCRL","X509v3 Delta CRL Indicator",NID_delta_crl,3,
1116         &(lvalues[748]),0},
1117 {"CRLReason","X509v3 CRL Reason Code",NID_crl_reason,3,&(lvalues[751]),0},
```



```
1118 {"invalidityDate","Invalidity Date",NID_invalidity_date,3,
1119         &(lvalues[754]),0},
1120 {"SXNetID","Strong Extranet ID",NID_sxnet,5,&(lvalues[757]),0},
1121 {"PBE-SHA1-RC4-128","pbeWithSHA1And128BitRC4",
1122         NID_pbe_WithSHA1And128BitRC4,10,&(lvalues[762]),0},
1123 {"PBE-SHA1-RC4-40","pbeWithSHA1And40BitRC4",
1124         NID_pbe_WithSHA1And40BitRC4,10,&(lvalues[772]),0},
1125 {"PBE-SHA1-3DES","pbeWithSHA1And3-KeyTripleDES-CBC",
1126         NID_pbe_WithSHA1And3_Key_TripleDES_CBC,10,&(lvalues[782]),0},
1127 {"PBE-SHA1-2DES","pbeWithSHA1And2-KeyTripleDES-CBC",
1128         NID_pbe_WithSHA1And2_Key_TripleDES_CBC,10,&(lvalues[792]),0},
1129 {"PBE-SHA1-RC2-128","pbeWithSHA1And128BitRC2-CBC",
1130         NID_pbe_WithSHA1And128BitRC2_CBC,10,&(lvalues[802]),0},
1131 {"PBE-SHA1-RC2-40","pbeWithSHA1And40BitRC2-CBC",
1132         NID_pbe_WithSHA1And40BitRC2_CBC,10,&(lvalues[812]),0},
1133 {"keyBag","keyBag",NID_keyBag,11,&(lvalues[822]),0},
1134 {"pkcs8ShroudedKeyBag","pkcs8ShroudedKeyBag",NID_pkcs8ShroudedKeyBag,
1135         11,&(lvalues[833]),0},
1136 {"certBag","certBag",NID_certBag,11,&(lvalues[844]),0},
1137 {"crlBag","crlBag",NID_crlBag,11,&(lvalues[855]),0},
1138 {"secretBag","secretBag",NID_secretBag,11,&(lvalues[866]),0},
1139 {"safeContentsBag","safeContentsBag",NID_safeContentsBag,11,
1140         &(lvalues[877]),0},
1141 {"friendlyName","friendlyName",NID_friendlyName,9,&(lvalues[888]),0},
1142 {"localKeyID","localKeyID",NID_localKeyID,9,&(lvalues[897]),0},
1143 {"x509Certificate","x509Certificate",NID_x509Certificate,10,
1144         &(lvalues[906]),0},
1145 {"sdsiCertificate","sdsiCertificate",NID_sdsiCertificate,10,
1146         &(lvalues[916]),0},
1147 {"x509Crl","x509Crl",NID_x509Crl,10,&(lvalues[926]),0},
1148 {"PBES2","PBES2",NID_pbes2,9,&(lvalues[936]),0},
1149 {"PBMAC1","PBMAC1",NID_pbmac1,9,&(lvalues[945]),0},
1150 {"hmacWithSHA1","hmacWithSHA1",NID_hmacWithSHA1,8,&(lvalues[954]),0},
1151 {"id-qt-cps","Policy Qualifier CPS",NID_id_qt_cps,8,&(lvalues[962]),0},
1152 {"id-qt-unotice","Policy Qualifier User Notice",NID_id_qt_unotice,8,
1153         &(lvalues[970]),0},
1154 {"RC2-64-CBC","rc2-64-cbc",NID_rc2_64_cbc,0,NULL,0},
1155 {"SMIME-CAPS","S/MIME Capabilities",NID_SMIMECapabilities,9,
1156         &(lvalues[978]),0},
1157 {"PBE-MD2-RC2-64","pbeWithMD2AndRC2-CBC",NID_pbeWithMD2AndRC2_CBC,9,
1158         &(lvalues[987]),0},
1159 {"PBE-MD5-RC2-64","pbeWithMD5AndRC2-CBC",NID_pbeWithMD5AndRC2_CBC,9,
1160         &(lvalues[996]),0},
1161 {"PBE-SHA1-DES","pbeWithSHA1AndDES-CBC",NID_pbeWithSHA1AndDES_CBC,9,
1162         &(lvalues[1005]),0},
1163 {"msExtReq","Microsoft Extension Request",NID_ms_ext_req,10,
1164         &(lvalues[1014]),0},
1165 {"extReq","Extension Request",NID_ext_req,9,&(lvalues[1024]),0},
1166 {"name","name",NID_name,3,&(lvalues[1033]),0},
1167 {"dnQualifier","dnQualifier",NID_dnQualifier,3,&(lvalues[1036]),0},
1168 {"id-pe","id-pe",NID_id_pe,7,&(lvalues[1039]),0},
1169 {"id-ad","id-ad",NID_id_ad,7,&(lvalues[1046]),0},
1170 {"authorityInfoAccess","Authority Information Access",NID_info_access,
1171         8,&(lvalues[1053]),0},
1172 {"OCSP","OCSP",NID_ad_OCSP,8,&(lvalues[1061]),0},
1173 {"caIssuers","CA Issuers",NID_ad_ca_issuers,8,&(lvalues[1069]),0},
1174 {"OCSPSigning","OCSP Signing",NID_OCSP_sign,8,&(lvalues[1077]),0},
1175 {"ISO","iso",NID_iso,0,NULL,0},
1176 {"member-body","ISO Member Body",NID_member_body,1,&(lvalues[1085]),0},
1177 {"ISO-US","ISO US Member Body",NID_ISO_US,3,&(lvalues[1086]),0},
1178 {"X9-57","X9.57",NID_X9_57,5,&(lvalues[1089]),0},
1179 {"X9cm","X9.57 CM ?",NID_X9cm,6,&(lvalues[1094]),0},
1180 {"pkcs1","pkcs1",NID_pkcs1,8,&(lvalues[1100]),0},
1181 {"pkcs5","pkcs5",NID_pkcs5,8,&(lvalues[1108]),0},
1182 {"SMIME","S/MIME",NID_SMIME,9,&(lvalues[1116]),0},
1183 {"id-smime-mod","id-smime-mod",NID_id_smime_mod,10,&(lvalues[1125]),0},
```

```
{"id-smime-ct","id-smime-ct",NID_id_smime_ct,10,&(lvalues[1135]),0},
{"id-smime-aa","id-smime-aa",NID_id_smime_aa,10,&(lvalues[1145]),0},
{"id-smime-alg","id-smime-alg",NID_id_smime_alg,10,&(lvalues[1155]),0},
{"id-smime-cd","id-smime-cd",NID_id_smime_cd,10,&(lvalues[1165]),0},
{"id-smime-spq","id-smime-spq",NID_id_smime_spq,10,&(lvalues[1175]),0},
{"id-smime-cti","id-smime-cti",NID_id_smime_cti,10,&(lvalues[1185]),0},
{"id-smime-mod-cms","id-smime-mod-cms",NID_id_smime_mod_cms,11,
        &(lvalues[1195]),0},
{"id-smime-mod-ess","id-smime-mod-ess",NID_id_smime_mod_ess,11,
        &(lvalues[1206]),0},
{"id-smime-mod-oid","id-smime-mod-oid",NID_id_smime_mod_oid,11,
        &(lvalues[1217]),0},
{"id-smime-mod-msg-v3","id-smime-mod-msg-v3",NID_id_smime_mod_msg_v3,
        11,&(lvalues[1228]),0},
{"id-smime-mod-ets-eSignature-88","id-smime-mod-ets-eSignature-88",
        NID_id_smime_mod_ets_eSignature_88,11,&(lvalues[1239]),0},
{"id-smime-mod-ets-eSignature-97","id-smime-mod-ets-eSignature-97",
        NID_id_smime_mod_ets_eSignature_97,11,&(lvalues[1250]),0},
{"id-smime-mod-ets-eSigPolicy-88","id-smime-mod-ets-eSigPolicy-88",
        NID_id_smime_mod_ets_eSigPolicy_88,11,&(lvalues[1261]),0},
{"id-smime-mod-ets-eSigPolicy-97","id-smime-mod-ets-eSigPolicy-97",
        NID_id_smime_mod_ets_eSigPolicy_97,11,&(lvalues[1272]),0},
{"id-smime-ct-receipt","id-smime-ct-receipt",NID_id_smime_ct_receipt,
        11,&(lvalues[1283]),0},
{"id-smime-ct-authData","id-smime-ct-authData",
        NID_id_smime_ct_authData,11,&(lvalues[1294]),0},
{"id-smime-ct-publishCert","id-smime-ct-publishCert",
        NID_id_smime_ct_publishCert,11,&(lvalues[1305]),0},
{"id-smime-ct-TSTInfo","id-smime-ct-TSTInfo",NID_id_smime_ct_TSTInfo,
        11,&(lvalues[1316]),0},
{"id-smime-ct-TDTInfo","id-smime-ct-TDTInfo",NID_id_smime_ct_TDTInfo,
        11,&(lvalues[1327]),0},
{"id-smime-ct-contentInfo","id-smime-ct-contentInfo",
        NID_id_smime_ct_contentInfo,11,&(lvalues[1338]),0},
{"id-smime-ct-DVCSRequestData","id-smime-ct-DVCSRequestData",
        NID_id_smime_ct_DVCSRequestData,11,&(lvalues[1349]),0},
{"id-smime-ct-DVCSResponseData","id-smime-ct-DVCSResponseData",
        NID_id_smime_ct_DVCSResponseData,11,&(lvalues[1360]),0},
{"id-smime-aa-receiptRequest","id-smime-aa-receiptRequest",
        NID_id_smime_aa_receiptRequest,11,&(lvalues[1371]),0},
{"id-smime-aa-securityLabel","id-smime-aa-securityLabel",
        NID_id_smime_aa_securityLabel,11,&(lvalues[1382]),0},
{"id-smime-aa-mlExpandHistory","id-smime-aa-mlExpandHistory",
        NID_id_smime_aa_mlExpandHistory,11,&(lvalues[1393]),0},
{"id-smime-aa-contentHint","id-smime-aa-contentHint",
        NID_id_smime_aa_contentHint,11,&(lvalues[1404]),0},
{"id-smime-aa-msgSigDigest","id-smime-aa-msgSigDigest",
        NID_id_smime_aa_msgSigDigest,11,&(lvalues[1415]),0},
{"id-smime-aa-encapContentType","id-smime-aa-encapContentType",
        NID_id_smime_aa_encapContentType,11,&(lvalues[1426]),0},
{"id-smime-aa-contentIdentifier","id-smime-aa-contentIdentifier",
        NID_id_smime_aa_contentIdentifier,11,&(lvalues[1437]),0},
{"id-smime-aa-macValue","id-smime-aa-macValue",
        NID_id_smime_aa_macValue,11,&(lvalues[1448]),0},
{"id-smime-aa-equivalentLabels","id-smime-aa-equivalentLabels",
        NID_id_smime_aa_equivalentLabels,11,&(lvalues[1459]),0},
{"id-smime-aa-contentReference","id-smime-aa-contentReference",
        NID_id_smime_aa_contentReference,11,&(lvalues[1470]),0},
{"id-smime-aa-encrypKeyPref","id-smime-aa-encrypKeyPref",
        NID_id_smime_aa_encrypKeyPref,11,&(lvalues[1481]),0},
{"id-smime-aa-signingCertificate","id-smime-aa-signingCertificate",
        NID_id_smime_aa_signingCertificate,11,&(lvalues[1492]),0},
{"id-smime-aa-smimeEncryptCerts","id-smime-aa-smimeEncryptCerts",
        NID_id_smime_aa_smimeEncryptCerts,11,&(lvalues[1503]),0},
{"id-smime-aa-timeStampToken","id-smime-aa-timeStampToken",
        NID_id_smime_aa_timeStampToken,11,&(lvalues[1514]),0},
```



```
{"id-smime-aa-ets-sigPolicyId","id-smime-aa-ets-sigPolicyId",
        NID_id_smime_aa_ets_sigPolicyId,11,&(lvalues[1525]),0},
{"id-smime-aa-ets-commitmentType","id-smime-aa-ets-commitmentType",
        NID_id_smime_aa_ets_commitmentType,11,&(lvalues[1536]),0},
{"id-smime-aa-ets-signerLocation","id-smime-aa-ets-signerLocation",
        NID_id_smime_aa_ets_signerLocation,11,&(lvalues[1547]),0},
{"id-smime-aa-ets-signerAttr","id-smime-aa-ets-signerAttr",
        NID_id_smime_aa_ets_signerAttr,11,&(lvalues[1558]),0},
{"id-smime-aa-ets-otherSigCert","id-smime-aa-ets-otherSigCert",
        NID_id_smime_aa_ets_otherSigCert,11,&(lvalues[1569]),0},
{"id-smime-aa-ets-contentTimestamp",
        "id-smime-aa-ets-contentTimestamp",
        NID_id_smime_aa_ets_contentTimestamp,11,&(lvalues[1580]),0},
{"id-smime-aa-ets-CertificateRefs","id-smime-aa-ets-CertificateRefs",
        NID_id_smime_aa_ets_CertificateRefs,11,&(lvalues[1591]),0},
{"id-smime-aa-ets-RevocationRefs","id-smime-aa-ets-RevocationRefs",
        NID_id_smime_aa_ets_RevocationRefs,11,&(lvalues[1602]),0},
{"id-smime-aa-ets-certValues","id-smime-aa-ets-certValues",
        NID_id_smime_aa_ets_certValues,11,&(lvalues[1613]),0},
{"id-smime-aa-ets-revocationValues",
        "id-smime-aa-ets-revocationValues",
        NID_id_smime_aa_ets_revocationValues,11,&(lvalues[1624]),0},
{"id-smime-aa-ets-escTimeStamp","id-smime-aa-ets-escTimeStamp",
        NID_id_smime_aa_ets_escTimeStamp,11,&(lvalues[1635]),0},
{"id-smime-aa-ets-certCRLTimestamp",
        "id-smime-aa-ets-certCRLTimestamp",
        NID_id_smime_aa_ets_certCRLTimestamp,11,&(lvalues[1646]),0},
{"id-smime-aa-ets-archiveTimeStamp",
        "id-smime-aa-ets-archiveTimeStamp",
        NID_id_smime_aa_ets_archiveTimeStamp,11,&(lvalues[1657]),0},
{"id-smime-aa-signatureType","id-smime-aa-signatureType",
        NID_id_smime_aa_signatureType,11,&(lvalues[1668]),0},
{"id-smime-aa-dvcs-dvc","id-smime-aa-dvcs-dvc",
        NID_id_smime_aa_dvcs_dvc,11,&(lvalues[1679]),0},
{"id-smime-alg-ESDHwith3DES","id-smime-alg-ESDHwith3DES",
        NID_id_smime_alg_ESDHwith3DES,11,&(lvalues[1690]),0},
{"id-smime-alg-ESDHwithRC2","id-smime-alg-ESDHwithRC2",
        NID_id_smime_alg_ESDHwithRC2,11,&(lvalues[1701]),0},
{"id-smime-alg-3DESwrap","id-smime-alg-3DESwrap",
        NID_id_smime_alg_3DESwrap,11,&(lvalues[1712]),0},
{"id-smime-alg-RC2wrap","id-smime-alg-RC2wrap",
        NID_id_smime_alg_RC2wrap,11,&(lvalues[1723]),0},
{"id-smime-alg-ESDH","id-smime-alg-ESDH",NID_id_smime_alg_ESDH,11,
        &(lvalues[1734]),0},
{"id-smime-alg-CMS3DESwrap","id-smime-alg-CMS3DESwrap",
        NID_id_smime_alg_CMS3DESwrap,11,&(lvalues[1745]),0},
{"id-smime-alg-CMSRC2wrap","id-smime-alg-CMSRC2wrap",
        NID_id_smime_alg_CMSRC2wrap,11,&(lvalues[1756]),0},
{"id-smime-cd-ldap","id-smime-cd-ldap",NID_id_smime_cd_ldap,11,
        &(lvalues[1767]),0},
{"id-smime-spq-ets-sqt-uri","id-smime-spq-ets-sqt-uri",
        NID_id_smime_spq_ets_sqt_uri,11,&(lvalues[1778]),0},
{"id-smime-spq-ets-sqt-unotice","id-smime-spq-ets-sqt-unotice",
        NID_id_smime_spq_ets_sqt_unotice,11,&(lvalues[1789]),0},
{"id-smime-cti-ets-proofOfOrigin","id-smime-cti-ets-proofOfOrigin",
        NID_id_smime_cti_ets_proofOfOrigin,11,&(lvalues[1800]),0},
{"id-smime-cti-ets-proofOfReceipt","id-smime-cti-ets-proofOfReceipt",
        NID_id_smime_cti_ets_proofOfReceipt,11,&(lvalues[1811]),0},
{"id-smime-cti-ets-proofOfDelivery",
        "id-smime-cti-ets-proofOfDelivery",
        NID_id_smime_cti_ets_proofOfDelivery,11,&(lvalues[1822]),0},
{"id-smime-cti-ets-proofOfSender","id-smime-cti-ets-proofOfSender",
        NID_id_smime_cti_ets_proofOfSender,11,&(lvalues[1833]),0},
{"id-smime-cti-ets-proofOfApproval",
        "id-smime-cti-ets-proofOfApproval",
        NID_id_smime_cti_ets_proofOfApproval,11,&(lvalues[1844]),0},
```

```
{"id-smime-cti-ets-proofOfCreation",
        "id-smime-cti-ets-proofOfCreation",
        NID_id_smime_cti_ets_proofOfCreation,11,&(lvalues[1855]),0},
{"MD4","md4",NID_md4,8,&(lvalues[1866]),0},
{"id-pkix-mod","id-pkix-mod",NID_id_pkix_mod,7,&(lvalues[1874]),0},
{"id-qt","id-qt",NID_id_qt,7,&(lvalues[1881]),0},
{"id-it","id-it",NID_id_it,7,&(lvalues[1888]),0},
{"id-pkip","id-pkip",NID_id_pkip,7,&(lvalues[1895]),0},
{"id-alg","id-alg",NID_id_alg,7,&(lvalues[1902]),0},
{"id-cmc","id-cmc",NID_id_cmc,7,&(lvalues[1909]),0},
{"id-on","id-on",NID_id_on,7,&(lvalues[1916]),0},
{"id-pda","id-pda",NID_id_pda,7,&(lvalues[1923]),0},
{"id-aca","id-aca",NID_id_aca,7,&(lvalues[1930]),0},
{"id-qcs","id-qcs",NID_id_qcs,7,&(lvalues[1937]),0},
{"id-cct","id-cct",NID_id_cct,7,&(lvalues[1944]),0},
{"id-pkix1-explicit-88","id-pkix1-explicit-88",
        NID_id_pkix1_explicit_88,8,&(lvalues[1951]),0},
{"id-pkix1-implicit-88","id-pkix1-implicit-88",
        NID_id_pkix1_implicit_88,8,&(lvalues[1959]),0},
{"id-pkix1-explicit-93","id-pkix1-explicit-93",
        NID_id_pkix1_explicit_93,8,&(lvalues[1967]),0},
{"id-pkix1-implicit-93","id-pkix1-implicit-93",
        NID_id_pkix1_implicit_93,8,&(lvalues[1975]),0},
{"id-mod-crmf","id-mod-crmf",NID_id_mod_crmf,8,&(lvalues[1983]),0},
{"id-mod-cmc","id-mod-cmc",NID_id_mod_cmc,8,&(lvalues[1991]),0},
{"id-mod-kea-profile-88","id-mod-kea-profile-88",
        NID_id_mod_kea_profile_88,8,&(lvalues[1999]),0},
{"id-mod-kea-profile-93","id-mod-kea-profile-93",
        NID_id_mod_kea_profile_93,8,&(lvalues[2007]),0},
{"id-mod-cmp","id-mod-cmp",NID_id_mod_cmp,8,&(lvalues[2015]),0},
{"id-mod-qualified-cert-88","id-mod-qualified-cert-88",
        NID_id_mod_qualified_cert_88,8,&(lvalues[2023]),0},
{"id-mod-qualified-cert-93","id-mod-qualified-cert-93",
        NID_id_mod_qualified_cert_93,8,&(lvalues[2031]),0},
{"id-mod-attribute-cert","id-mod-attribute-cert",
        NID_id_mod_attribute_cert,8,&(lvalues[2039]),0},
{"id-mod-timestamp-protocol","id-mod-timestamp-protocol",
        NID_id_mod_timestamp_protocol,8,&(lvalues[2047]),0},
{"id-mod-ocsp","id-mod-ocsp",NID_id_mod_ocsp,8,&(lvalues[2055]),0},
{"id-mod-dvcs","id-mod-dvcs",NID_id_mod_dvcs,8,&(lvalues[2063]),0},
{"id-mod-cmp2000","id-mod-cmp2000",NID_id_mod_cmp2000,8,
        &(lvalues[2071]),0},
{"biometricInfo","Biometric Info",NID_biometricInfo,8,&(lvalues[2079]),0},
{"qcStatements","qcStatements",NID_qcStatements,8,&(lvalues[2087]),0},
{"ac-auditEntity","ac-auditEntity",NID_ac_auditEntity,8,
        &(lvalues[2095]),0},
{"ac-targeting","ac-targeting",NID_ac_targeting,8,&(lvalues[2103]),0},
{"aaControls","aaControls",NID_aaControls,8,&(lvalues[2111]),0},
{"sbgp-ipAddrBlock","sbgp-ipAddrBlock",NID_sbgp_ipAddrBlock,8,
        &(lvalues[2119]),0},
{"sbgp-autonomousSysNum","sbgp-autonomousSysNum",
        NID_sbgp_autonomousSysNum,8,&(lvalues[2127]),0},
{"sbgp-routerIdentifier","sbgp-routerIdentifier",
        NID_sbgp_routerIdentifier,8,&(lvalues[2135]),0},
{"textNotice","textNotice",NID_textNotice,8,&(lvalues[2143]),0},
{"ipsecEndSystem","IPSec End System",NID_ipsecEndSystem,8,
        &(lvalues[2151]),0},
{"ipsecTunnel","IPSec Tunnel",NID_ipsecTunnel,8,&(lvalues[2159]),0},
{"ipsecUser","IPSec User",NID_ipsecUser,8,&(lvalues[2167]),0},
{"DVCS","dvcs",NID_dvcs,8,&(lvalues[2175]),0},
{"id-it-caProtEncCert","id-it-caProtEncCert",NID_id_it_caProtEncCert,
        8,&(lvalues[2183]),0},
{"id-it-signKeyPairTypes","id-it-signKeyPairTypes",
        NID_id_it_signKeyPairTypes,8,&(lvalues[2191]),0},
{"id-it-encKeyPairTypes","id-it-encKeyPairTypes",
        NID_id_it_encKeyPairTypes,8,&(lvalues[2199]),0},
```



```
{"id-it-preferredSymmAlg","id-it-preferredSymmAlg",
        NID_id_it_preferredSymmAlg,8,&(lvalues[2207]),0},
{"id-it-caKeyUpdateInfo","id-it-caKeyUpdateInfo",
        NID_id_it_caKeyUpdateInfo,8,&(lvalues[2215]),0},
{"id-it-currentCRL","id-it-currentCRL",NID_id_it_currentCRL,8,
        &(lvalues[2223]),0},
{"id-it-unsupportedOIDs","id-it-unsupportedOIDs",
        NID_id_it_unsupportedOIDs,8,&(lvalues[2231]),0},
{"id-it-subscriptionRequest","id-it-subscriptionRequest",
        NID_id_it_subscriptionRequest,8,&(lvalues[2239]),0},
{"id-it-subscriptionResponse","id-it-subscriptionResponse",
        NID_id_it_subscriptionResponse,8,&(lvalues[2247]),0},
{"id-it-keyPairParamReq","id-it-keyPairParamReq",
        NID_id_it_keyPairParamReq,8,&(lvalues[2255]),0},
{"id-it-keyPairParamRep","id-it-keyPairParamRep",
        NID_id_it_keyPairParamRep,8,&(lvalues[2263]),0},
{"id-it-revPassphrase","id-it-revPassphrase",NID_id_it_revPassphrase,
        8,&(lvalues[2271]),0},
{"id-it-implicitConfirm","id-it-implicitConfirm",
        NID_id_it_implicitConfirm,8,&(lvalues[2279]),0},
{"id-it-confirmWaitTime","id-it-confirmWaitTime",
        NID_id_it_confirmWaitTime,8,&(lvalues[2287]),0},
{"id-it-origPKIMessage","id-it-origPKIMessage",
        NID_id_it_origPKIMessage,8,&(lvalues[2295]),0},
{"id-regCtrl","id-regCtrl",NID_id_regCtrl,8,&(lvalues[2303]),0},
{"id-regInfo","id-regInfo",NID_id_regInfo,8,&(lvalues[2311]),0},
{"id-regCtrl-regToken","id-regCtrl-regToken",NID_id_regCtrl_regToken,
        9,&(lvalues[2319]),0},
{"id-regCtrl-authenticator","id-regCtrl-authenticator",
        NID_id_regCtrl_authenticator,9,&(lvalues[2328]),0},
{"id-regCtrl-pkiPublicationInfo","id-regCtrl-pkiPublicationInfo",
        NID_id_regCtrl_pkiPublicationInfo,9,&(lvalues[2337]),0},
{"id-regCtrl-pkiArchiveOptions","id-regCtrl-pkiArchiveOptions",
        NID_id_regCtrl_pkiArchiveOptions,9,&(lvalues[2346]),0},
{"id-regCtrl-oldCertID","id-regCtrl-oldCertID",
        NID_id_regCtrl_oldCertID,9,&(lvalues[2355]),0},
{"id-regCtrl-protocolEncrKey","id-regCtrl-protocolEncrKey",
        NID_id_regCtrl_protocolEncrKey,9,&(lvalues[2364]),0},
{"id-regInfo-utf8Pairs","id-regInfo-utf8Pairs",
        NID_id_regInfo_utf8Pairs,9,&(lvalues[2373]),0},
{"id-regInfo-certReq","id-regInfo-certReq",NID_id_regInfo_certReq,9,
        &(lvalues[2382]),0},
{"id-alg-des40","id-alg-des40",NID_id_alg_des40,8,&(lvalues[2391]),0},
{"id-alg-noSignature","id-alg-noSignature",NID_id_alg_noSignature,8,
        &(lvalues[2399]),0},
{"id-alg-dh-sig-hmac-sha1","id-alg-dh-sig-hmac-sha1",
        NID_id_alg_dh_sig_hmac_sha1,8,&(lvalues[2407]),0},
{"id-alg-dh-pop","id-alg-dh-pop",NID_id_alg_dh_pop,8,&(lvalues[2415]),0},
{"id-cmc-statusInfo","id-cmc-statusInfo",NID_id_cmc_statusInfo,8,
        &(lvalues[2423]),0},
{"id-cmc-identification","id-cmc-identification",
        NID_id_cmc_identification,8,&(lvalues[2431]),0},
{"id-cmc-identityProof","id-cmc-identityProof",
        NID_id_cmc_identityProof,8,&(lvalues[2439]),0},
{"id-cmc-dataReturn","id-cmc-dataReturn",NID_id_cmc_dataReturn,8,
        &(lvalues[2447]),0},
{"id-cmc-transactionId","id-cmc-transactionId",
        NID_id_cmc_transactionId,8,&(lvalues[2455]),0},
{"id-cmc-senderNonce","id-cmc-senderNonce",NID_id_cmc_senderNonce,8,
        &(lvalues[2463]),0},
{"id-cmc-recipientNonce","id-cmc-recipientNonce",
        NID_id_cmc_recipientNonce,8,&(lvalues[2471]),0},
{"id-cmc-addExtensions","id-cmc-addExtensions",
        NID_id_cmc_addExtensions,8,&(lvalues[2479]),0},
{"id-cmc-encryptedPOP","id-cmc-encryptedPOP",NID_id_cmc_encryptedPOP,
        8,&(lvalues[2487]),0},
```

```
{"id-cmc-decryptedPOP","id-cmc-decryptedPOP",NID_id_cmc_decryptedPOP,
        8,&(lvalues[2495]),0},
{"id-cmc-lraPOPWitness","id-cmc-lraPOPWitness",
        NID_id_cmc_lraPOPWitness,8,&(lvalues[2503]),0},
{"id-cmc-getCert","id-cmc-getCert",NID_id_cmc_getCert,8,
        &(lvalues[2511]),0},
{"id-cmc-getCRL","id-cmc-getCRL",NID_id_cmc_getCRL,8,&(lvalues[2519]),0},
{"id-cmc-revokeRequest","id-cmc-revokeRequest",
        NID_id_cmc_revokeRequest,8,&(lvalues[2527]),0},
{"id-cmc-regInfo","id-cmc-regInfo",NID_id_cmc_regInfo,8,
        &(lvalues[2535]),0},
{"id-cmc-responseInfo","id-cmc-responseInfo",NID_id_cmc_responseInfo,
        8,&(lvalues[2543]),0},
{"id-cmc-queryPending","id-cmc-queryPending",NID_id_cmc_queryPending,
        8,&(lvalues[2551]),0},
{"id-cmc-popLinkRandom","id-cmc-popLinkRandom",
        NID_id_cmc_popLinkRandom,8,&(lvalues[2559]),0},
{"id-cmc-popLinkWitness","id-cmc-popLinkWitness",
        NID_id_cmc_popLinkWitness,8,&(lvalues[2567]),0},
{"id-cmc-confirmCertAcceptance","id-cmc-confirmCertAcceptance",
        NID_id_cmc_confirmCertAcceptance,8,&(lvalues[2575]),0},
{"id-on-personalData","id-on-personalData",NID_id_on_personalData,8,
        &(lvalues[2583]),0},
{"id-pda-dateOfBirth","id-pda-dateOfBirth",NID_id_pda_dateOfBirth,8,
        &(lvalues[2591]),0},
{"id-pda-placeOfBirth","id-pda-placeOfBirth",NID_id_pda_placeOfBirth,
        8,&(lvalues[2599]),0},
{NULL,NULL,NID_undef,0,NULL,0},
{"id-pda-gender","id-pda-gender",NID_id_pda_gender,8,&(lvalues[2607]),0},
{"id-pda-countryOfCitizenship","id-pda-countryOfCitizenship",
        NID_id_pda_countryOfCitizenship,8,&(lvalues[2615]),0},
{"id-pda-countryOfResidence","id-pda-countryOfResidence",
        NID_id_pda_countryOfResidence,8,&(lvalues[2623]),0},
{"id-aca-authenticationInfo","id-aca-authenticationInfo",
        NID_id_aca_authenticationInfo,8,&(lvalues[2631]),0},
{"id-aca-accessIdentity","id-aca-accessIdentity",
        NID_id_aca_accessIdentity,8,&(lvalues[2639]),0},
{"id-aca-chargingIdentity","id-aca-chargingIdentity",
        NID_id_aca_chargingIdentity,8,&(lvalues[2647]),0},
{"id-aca-group","id-aca-group",NID_id_aca_group,8,&(lvalues[2655]),0},
{"id-aca-role","id-aca-role",NID_id_aca_role,8,&(lvalues[2663]),0},
{"id-qcs-pkixQCSyntax-v1","id-qcs-pkixQCSyntax-v1",
        NID_id_qcs_pkixQCSyntax_v1,8,&(lvalues[2671]),0},
{"id-cct-crs","id-cct-crs",NID_id_cct_crs,8,&(lvalues[2679]),0},
{"id-cct-PKIData","id-cct-PKIData",NID_id_cct_PKIData,8,
        &(lvalues[2687]),0},
{"id-cct-PKIResponse","id-cct-PKIResponse",NID_id_cct_PKIResponse,8,
        &(lvalues[2695]),0},
{"ad_timestamping","AD Time Stamping",NID_ad_timeStamping,8,
        &(lvalues[2703]),0},
{"AD_DVCS","ad dvcs",NID_ad_dvcs,8,&(lvalues[2711]),0},
{"basicOCSPResponse","Basic OCSP Response",NID_id_pkix_OCSP_basic,9,
        &(lvalues[2719]),0},
{"Nonce","OCSP Nonce",NID_id_pkix_OCSP_Nonce,9,&(lvalues[2728]),0},
{"CrlID","OCSP CRL ID",NID_id_pkix_OCSP_CrlID,9,&(lvalues[2737]),0},
{"acceptableResponses","Acceptable OCSP Responses",
        NID_id_pkix_OCSP_acceptableResponses,9,&(lvalues[2746]),0},
{"noCheck","OCSP No Check",NID_id_pkix_OCSP_noCheck,9,&(lvalues[2755]),0},
{"archiveCutoff","OCSP Archive Cutoff",NID_id_pkix_OCSP_archiveCutoff,
        9,&(lvalues[2764]),0},
{"serviceLocator","OCSP Service Locator",
        NID_id_pkix_OCSP_serviceLocator,9,&(lvalues[2773]),0},
{"extendedStatus","Extended OCSP Status",
        NID_id_pkix_OCSP_extendedStatus,9,&(lvalues[2782]),0},
{"valid","valid",NID_id_pkix_OCSP_valid,9,&(lvalues[2791]),0},
{"path","path",NID_id_pkix_OCSP_path,9,&(lvalues[2800]),0},
```



```
{"trustRoot","Trust Root",NID_id_pkix_OCSP_trustRoot,9,
        &(lvalues[2809]),0},
{"algorithm","algorithm",NID_algorithm,4,&(lvalues[2818]),0},
{"rsaSignature","rsaSignature",NID_rsaSignature,5,&(lvalues[2822]),0},
{"X500algorithms","directory services - algorithms",
        NID_X500algorithms,2,&(lvalues[2827]),0},
{"ORG","org",NID_org,1,&(lvalues[2829]),0},
{"DOD","dod",NID_dod,2,&(lvalues[2830]),0},
{"IANA","iana",NID_iana,3,&(lvalues[2832]),0},
{"directory","Directory",NID_Directory,4,&(lvalues[2835]),0},
{"mgmt","Management",NID_Management,4,&(lvalues[2839]),0},
{"experimental","Experimental",NID_Experimental,4,&(lvalues[2843]),0},
{"private","Private",NID_Private,4,&(lvalues[2847]),0},
{"security","Security",NID_Security,4,&(lvalues[2851]),0},
{"snmpv2","SNMPv2",NID_SNMPv2,4,&(lvalues[2855]),0},
{"Mail","Mail",NID_Mail,4,&(lvalues[2859]),0},
{"enterprises","Enterprises",NID_Enterprises,5,&(lvalues[2863]),0},
{"dcobject","dcObject",NID_dcObject,9,&(lvalues[2868]),0},
{"DC","domainComponent",NID_domainComponent,10,&(lvalues[2877]),0},
{"domain","Domain",NID_Domain,10,&(lvalues[2887]),0},
{"NULL","NULL",NID_joint_iso_ccitt,0,NULL,0},
{"selected-attribute-types","Selected Attribute Types",
        NID_selected_attribute_types,3,&(lvalues[2897]),0},
{"clearance","clearance",NID_clearance,4,&(lvalues[2900]),0},
{"RSA-MD4","md4WithRSAEncryption",NID_md4WithRSAEncryption,9,
        &(lvalues[2904]),0},
{"ac-proxying","ac-proxying",NID_ac_proxying,8,&(lvalues[2913]),0},
{"subjectInfoAccess","Subject Information Access",NID_sinfo_access,8,
        &(lvalues[2921]),0},
{"id-aca-encAttrs","id-aca-encAttrs",NID_id_aca_encAttrs,8,
        &(lvalues[2929]),0},
{"role","role",NID_role,3,&(lvalues[2937]),0},
{"policyConstraints","X509v3 Policy Constraints",
        NID_policy_constraints,3,&(lvalues[2940]),0},
{"targetInformation","X509v3 AC Targeting",NID_target_information,3,
        &(lvalues[2943]),0},
{"noRevAvail","X509v3 No Revocation Available",NID_no_rev_avail,3,
        &(lvalues[2946]),0},
{"NULL","NULL",NID_ccitt,0,NULL,0},
{"ansi-X9-62","ANSI X9.62",NID_ansi_X9_62,5,&(lvalues[2949]),0},
{"prime-field","prime-field",NID_X9_62_prime_field,7,&(lvalues[2954]),0},
{"characteristic-two-field","characteristic-two-field",
        NID_X9_62_characteristic_two_field,7,&(lvalues[2961]),0},
{"id-ecPublicKey","id-ecPublicKey",NID_X9_62_id_ecPublicKey,7,
        &(lvalues[2968]),0},
{"prime192v1","prime192v1",NID_X9_62_prime192v1,8,&(lvalues[2975]),0},
{"prime192v2","prime192v2",NID_X9_62_prime192v2,8,&(lvalues[2983]),0},
{"prime192v3","prime192v3",NID_X9_62_prime192v3,8,&(lvalues[2991]),0},
{"prime239v1","prime239v1",NID_X9_62_prime239v1,8,&(lvalues[2999]),0},
{"prime239v2","prime239v2",NID_X9_62_prime239v2,8,&(lvalues[3007]),0},
{"prime239v3","prime239v3",NID_X9_62_prime239v3,8,&(lvalues[3015]),0},
{"prime256v1","prime256v1",NID_X9_62_prime256v1,8,&(lvalues[3023]),0},
{"ecdsa-with-SHA1","ecdsa-with-SHA1",NID_ecdsa_with_SHA1,7,
        &(lvalues[3031]),0},
{"CSPName","Microsoft CSP Name",NID_ms_csp_name,9,&(lvalues[3038]),0},
{"AES-128-ECB","aes-128-ecb",NID_aes_128_ecb,9,&(lvalues[3047]),0},
{"AES-128-CBC","aes-128-cbc",NID_aes_128_cbc,9,&(lvalues[3056]),0},
{"AES-128-OFB","aes-128-ofb",NID_aes_128_ofb128,9,&(lvalues[3065]),0},
{"AES-128-CFB","aes-128-cfb",NID_aes_128_cfb128,9,&(lvalues[3074]),0},
{"AES-192-ECB","aes-192-ecb",NID_aes_192_ecb,9,&(lvalues[3083]),0},
{"AES-192-CBC","aes-192-cbc",NID_aes_192_cbc,9,&(lvalues[3092]),0},
{"AES-192-OFB","aes-192-ofb",NID_aes_192_ofb128,9,&(lvalues[3101]),0},
{"AES-192-CFB","aes-192-cfb",NID_aes_192_cfb128,9,&(lvalues[3110]),0},
{"AES-256-ECB","aes-256-ecb",NID_aes_256_ecb,9,&(lvalues[3119]),0},
{"AES-256-CBC","aes-256-cbc",NID_aes_256_cbc,9,&(lvalues[3128]),0},
{"AES-256-OFB","aes-256-ofb",NID_aes_256_ofb128,9,&(lvalues[3137]),0},
```

```
{"AES-256-CFB","aes-256-cfb",NID_aes_256_cfb128,9,&(lvalues[3146]),0},
{"holdInstructionCode","Hold Instruction Code",
        NID_hold_instruction_code,3,&(lvalues[3155]),0},
{"holdInstructionNone","Hold Instruction None",
        NID_hold_instruction_none,7,&(lvalues[3158]),0},
{"holdInstructionCallIssuer","Hold Instruction Call Issuer",
        NID_hold_instruction_call_issuer,7,&(lvalues[3165]),0},
{"holdInstructionReject","Hold Instruction Reject",
        NID_hold_instruction_reject,7,&(lvalues[3172]),0},
{"data","data",NID_data,1,&(lvalues[3179]),0},
{"pss","pss",NID_pss,3,&(lvalues[3180]),0},
{"ucl","ucl",NID_ucl,7,&(lvalues[3183]),0},
{"pilot","pilot",NID_pilot,8,&(lvalues[3190]),0},
{"pilotAttributeType","pilotAttributeType",NID_pilotAttributeType,9,
        &(lvalues[3198]),0},
{"pilotAttributeSyntax","pilotAttributeSyntax",
        NID_pilotAttributeSyntax,9,&(lvalues[3207]),0},
{"pilotObjectClass","pilotObjectClass",NID_pilotObjectClass,9,
        &(lvalues[3216]),0},
{"pilotGroups","pilotGroups",NID_pilotGroups,9,&(lvalues[3225]),0},
{"iA5StringSyntax","iA5StringSyntax",NID_iA5StringSyntax,10,
        &(lvalues[3234]),0},
{"caseIgnoreIA5StringSyntax","caseIgnoreIA5StringSyntax",
        NID_caseIgnoreIA5StringSyntax,10,&(lvalues[3244]),0},
{"pilotObject","pilotObject",NID_pilotObject,10,&(lvalues[3254]),0},
{"pilotPerson","pilotPerson",NID_pilotPerson,10,&(lvalues[3264]),0},
{"account","account",NID_account,10,&(lvalues[3274]),0},
{"document","document",NID_document,10,&(lvalues[3284]),0},
{"room","room",NID_room,10,&(lvalues[3294]),0},
{"documentSeries","documentSeries",NID_documentSeries,10,
        &(lvalues[3304]),0},
{"rFC822localPart","rFC822localPart",NID_rFC822localPart,10,
        &(lvalues[3314]),0},
{"dNSDomain","dNSDomain",NID_dNSDomain,10,&(lvalues[3324]),0},
{"domainRelatedObject","domainRelatedObject",NID_domainRelatedObject,
        10,&(lvalues[3334]),0},
{"friendlyCountry","friendlyCountry",NID_friendlyCountry,10,
        &(lvalues[3344]),0},
{"simpleSecurityObject","simpleSecurityObject",
        NID_simpleSecurityObject,10,&(lvalues[3354]),0},
{"pilotOrganization","pilotOrganization",NID_pilotOrganization,10,
        &(lvalues[3364]),0},
{"pilotDSA","pilotDSA",NID_pilotDSA,10,&(lvalues[3374]),0},
{"qualityLabelledData","qualityLabelledData",NID_qualityLabelledData,
        10,&(lvalues[3384]),0},
{"UID","userId",NID_userId,10,&(lvalues[3394]),0},
{"textEncodedORAddress","textEncodedORAddress",
        NID_textEncodedORAddress,10,&(lvalues[3404]),0},
{"mail","rfc822Mailbox",NID_rfc822Mailbox,10,&(lvalues[3414]),0},
{"info","info",NID_info,10,&(lvalues[3424]),0},
{"favouriteDrink","favouriteDrink",NID_favouriteDrink,10,
        &(lvalues[3434]),0},
{"roomNumber","roomNumber",NID_roomNumber,10,&(lvalues[3444]),0},
{"photo","photo",NID_photo,10,&(lvalues[3454]),0},
{"userClass","userClass",NID_userClass,10,&(lvalues[3464]),0},
{"host","host",NID_host,10,&(lvalues[3474]),0},
{"manager","manager",NID_manager,10,&(lvalues[3484]),0},
{"documentIdentifier","documentIdentifier",NID_documentIdentifier,10,
        &(lvalues[3494]),0},
{"documentTitle","documentTitle",NID_documentTitle,10,&(lvalues[3504]),0},
{"documentVersion","documentVersion",NID_documentVersion,10,
        &(lvalues[3514]),0},
{"documentAuthor","documentAuthor",NID_documentAuthor,10,
        &(lvalues[3524]),0},
{"documentLocation","documentLocation",NID_documentLocation,10,
        &(lvalues[3534]),0},
```



```
{"homeTelephoneNumber","homeTelephoneNumber",NID_homeTelephoneNumber,
        10,&(lvalues[3544]),0},
{"secretary","secretary",NID_secretary,10,&(lvalues[3554]),0},
{"otherMailbox","otherMailbox",NID_otherMailbox,10,&(lvalues[3564]),0},
{"lastModifiedTime","lastModifiedTime",NID_lastModifiedTime,10,
        &(lvalues[3574]),0},
{"lastModifiedBy","lastModifiedBy",NID_lastModifiedBy,10,
        &(lvalues[3584]),0},
{"aRecord","aRecord",NID_aRecord,10,&(lvalues[3594]),0},
{"pilotAttributeType27","pilotAttributeType27",
        NID_pilotAttributeType27,10,&(lvalues[3604]),0},
{"mXRecord","mXRecord",NID_mXRecord,10,&(lvalues[3614]),0},
{"nSRecord","nSRecord",NID_nSRecord,10,&(lvalues[3624]),0},
{"sOARecord","sOARecord",NID_sOARecord,10,&(lvalues[3634]),0},
{"cNAMERecord","cNAMERecord",NID_cNAMERecord,10,&(lvalues[3644]),0},
{"associatedDomain","associatedDomain",NID_associatedDomain,10,
        &(lvalues[3654]),0},
{"associatedName","associatedName",NID_associatedName,10,
        &(lvalues[3664]),0},
{"homePostalAddress","homePostalAddress",NID_homePostalAddress,10,
        &(lvalues[3674]),0},
{"personalTitle","personalTitle",NID_personalTitle,10,&(lvalues[3684]),0},
{"mobileTelephoneNumber","mobileTelephoneNumber",
        NID_mobileTelephoneNumber,10,&(lvalues[3694]),0},
{"pagerTelephoneNumber","pagerTelephoneNumber",
        NID_pagerTelephoneNumber,10,&(lvalues[3704]),0},
{"friendlyCountryName","friendlyCountryName",NID_friendlyCountryName,
        10,&(lvalues[3714]),0},
{"organizationalStatus","organizationalStatus",
        NID_organizationalStatus,10,&(lvalues[3724]),0},
{"janetMailbox","janetMailbox",NID_janetMailbox,10,&(lvalues[3734]),0},
{"mailPreferenceOption","mailPreferenceOption",
        NID_mailPreferenceOption,10,&(lvalues[3744]),0},
{"buildingName","buildingName",NID_buildingName,10,&(lvalues[3754]),0},
{"dSAQuality","dSAQuality",NID_dSAQuality,10,&(lvalues[3764]),0},
{"singleLevelQuality","singleLevelQuality",NID_singleLevelQuality,10,
        &(lvalues[3774]),0},
{"subtreeMinimumQuality","subtreeMinimumQuality",
        NID_subtreeMinimumQuality,10,&(lvalues[3784]),0},
{"subtreeMaximumQuality","subtreeMaximumQuality",
        NID_subtreeMaximumQuality,10,&(lvalues[3794]),0},
{"personalSignature","personalSignature",NID_personalSignature,10,
        &(lvalues[3804]),0},
{"dITRedirect","dITRedirect",NID_dITRedirect,10,&(lvalues[3814]),0},
{"audio","audio",NID_audio,10,&(lvalues[3824]),0},
{"documentPublisher","documentPublisher",NID_documentPublisher,10,
        &(lvalues[3834]),0},
{"x500UniqueIdentifier","x500UniqueIdentifier",
        NID_x500UniqueIdentifier,3,&(lvalues[3844]),0},
{"mime-mhs","MIME MHS",NID_mime_mhs,5,&(lvalues[3847]),0},
{"mime-mhs-headings","mime-mhs-headings",NID_mime_mhs_headings,6,
        &(lvalues[3852]),0},
{"mime-mhs-bodies","mime-mhs-bodies",NID_mime_mhs_bodies,6,
        &(lvalues[3858]),0},
{"id-hex-partial-message","id-hex-partial-message",
        NID_id_hex_partial_message,7,&(lvalues[3864]),0},
{"id-hex-multipart-message","id-hex-multipart-message",
        NID_id_hex_multipart_message,7,&(lvalues[3871]),0},
{"generationQualifier","generationQualifier",NID_generationQualifier,
        3,&(lvalues[3878]),0},
{"pseudonym","pseudonym",NID_pseudonym,3,&(lvalues[3881]),0},
{NULL,NULL,NID_undef,0,NULL,0},
{"id-set","Secure Electronic Transactions",NID_id_set,2,
        &(lvalues[3884]),0},
{"set-ctype","content types",NID_set_ctype,3,&(lvalues[3886]),0},
{"set-msgExt","message extensions",NID_set_msgExt,3,&(lvalues[3889]),0},
```

```
{"set-attr","set-attr",NID_set_attr,3,&(lvalues[3892]),0},
{"set-policy","set-policy",NID_set_policy,3,&(lvalues[3895]),0},
{"set-certExt","certificate extensions",NID_set_certExt,3,
        &(lvalues[3898]),0},
{"set-brand","set-brand",NID_set_brand,3,&(lvalues[3901]),0},
{"setct-PANData","setct-PANData",NID_setct_PANData,4,&(lvalues[3904]),0},
{"setct-PANToken","setct-PANToken",NID_setct_PANToken,4,
        &(lvalues[3908]),0},
{"setct-PANOnly","setct-PANOnly",NID_setct_PANOnly,4,&(lvalues[3912]),0},
{"setct-OIData","setct-OIData",NID_setct_OIData,4,&(lvalues[3916]),0},
{"setct-PI","setct-PI",NID_setct_PI,4,&(lvalues[3920]),0},
{"setct-PIData","setct-PIData",NID_setct_PIData,4,&(lvalues[3924]),0},
{"setct-PIDataUnsigned","setct-PIDataUnsigned",
        NID_setct_PIDataUnsigned,4,&(lvalues[3928]),0},
{"setct-HODInput","setct-HODInput",NID_setct_HODInput,4,
        &(lvalues[3932]),0},
{"setct-AuthResBaggage","setct-AuthResBaggage",
        NID_setct_AuthResBaggage,4,&(lvalues[3936]),0},
{"setct-AuthRevReqBaggage","setct-AuthRevReqBaggage",
        NID_setct_AuthRevReqBaggage,4,&(lvalues[3940]),0},
{"setct-AuthRevResBaggage","setct-AuthRevResBaggage",
        NID_setct_AuthRevResBaggage,4,&(lvalues[3944]),0},
{"setct-CapTokenSeq","setct-CapTokenSeq",NID_setct_CapTokenSeq,4,
        &(lvalues[3948]),0},
{"setct-PInitResData","setct-PInitResData",NID_setct_PInitResData,4,
        &(lvalues[3952]),0},
{"setct-PI-TBS","setct-PI-TBS",NID_setct_PI_TBS,4,&(lvalues[3956]),0},
{"setct-PResData","setct-PResData",NID_setct_PResData,4,
        &(lvalues[3960]),0},
{"setct-AuthReqTBS","setct-AuthReqTBS",NID_setct_AuthReqTBS,4,
        &(lvalues[3964]),0},
{"setct-AuthResTBS","setct-AuthResTBS",NID_setct_AuthResTBS,4,
        &(lvalues[3968]),0},
{"setct-AuthResTBSX","setct-AuthResTBSX",NID_setct_AuthResTBSX,4,
        &(lvalues[3972]),0},
{"setct-AuthTokenTBS","setct-AuthTokenTBS",NID_setct_AuthTokenTBS,4,
        &(lvalues[3976]),0},
{"setct-CapTokenData","setct-CapTokenData",NID_setct_CapTokenData,4,
        &(lvalues[3980]),0},
{"setct-CapTokenTBS","setct-CapTokenTBS",NID_setct_CapTokenTBS,4,
        &(lvalues[3984]),0},
{"setct-AcqCardCodeMsg","setct-AcqCardCodeMsg",
        NID_setct_AcqCardCodeMsg,4,&(lvalues[3988]),0},
{"setct-AuthRevReqTBS","setct-AuthRevReqTBS",NID_setct_AuthRevReqTBS,
        4,&(lvalues[3992]),0},
{"setct-AuthRevResData","setct-AuthRevResData",
        NID_setct_AuthRevResData,4,&(lvalues[3996]),0},
{"setct-AuthRevResTBS","setct-AuthRevResTBS",NID_setct_AuthRevResTBS,
        4,&(lvalues[4000]),0},
{"setct-CapReqTBS","setct-CapReqTBS",NID_setct_CapReqTBS,4,
        &(lvalues[4004]),0},
{"setct-CapReqTBSX","setct-CapReqTBSX",NID_setct_CapReqTBSX,4,
        &(lvalues[4008]),0},
{"setct-CapResData","setct-CapResData",NID_setct_CapResData,4,
        &(lvalues[4012]),0},
{"setct-CapRevReqTBS","setct-CapRevReqTBS",NID_setct_CapRevReqTBS,4,
        &(lvalues[4016]),0},
{"setct-CapRevReqTBSX","setct-CapRevReqTBSX",NID_setct_CapRevReqTBSX,
        4,&(lvalues[4020]),0},
{"setct-CapRevResData","setct-CapRevResData",NID_setct_CapRevResData,
        4,&(lvalues[4024]),0},
{"setct-CredReqTBS","setct-CredReqTBS",NID_setct_CredReqTBS,4,
        &(lvalues[4028]),0},
{"setct-CredReqTBSX","setct-CredReqTBSX",NID_setct_CredReqTBSX,4,
        &(lvalues[4032]),0},
{"setct-CredResData","setct-CredResData",NID_setct_CredResData,4,
```



```
        &(lvalues[4036]),0},
{"setct-CredRevReqTBS","setct-CredRevReqTBS",NID_setct_CredRevReqTBS,
        4,&(lvalues[4040]),0},
{"setct-CredRevReqTBSX","setct-CredRevReqTBSX",
        NID_setct_CredRevReqTBSX,4,&(lvalues[4044]),0},
{"setct-CredRevResData","setct-CredRevResData",
        NID_setct_CredRevResData,4,&(lvalues[4048]),0},
{"setct-PCertReqData","setct-PCertReqData",NID_setct_PCertReqData,4,
        &(lvalues[4052]),0},
{"setct-PCertResTBS","setct-PCertResTBS",NID_setct_PCertResTBS,4,
        &(lvalues[4056]),0},
{"setct-BatchAdminReqData","setct-BatchAdminReqData",
        NID_setct_BatchAdminReqData,4,&(lvalues[4060]),0},
{"setct-BatchAdminResData","setct-BatchAdminResData",
        NID_setct_BatchAdminResData,4,&(lvalues[4064]),0},
{"setct-CardCInitResTBS","setct-CardCInitResTBS",
        NID_setct_CardCInitResTBS,4,&(lvalues[4068]),0},
{"setct-MeAqCInitResTBS","setct-MeAqCInitResTBS",
        NID_setct_MeAqCInitResTBS,4,&(lvalues[4072]),0},
{"setct-RegFormResTBS","setct-RegFormResTBS",NID_setct_RegFormResTBS,
        4,&(lvalues[4076]),0},
{"setct-CertReqData","setct-CertReqData",NID_setct_CertReqData,4,
        &(lvalues[4080]),0},
{"setct-CertReqTBS","setct-CertReqTBS",NID_setct_CertReqTBS,4,
        &(lvalues[4084]),0},
{"setct-CertResData","setct-CertResData",NID_setct_CertResData,4,
        &(lvalues[4088]),0},
{"setct-CertInqReqTBS","setct-CertInqReqTBS",NID_setct_CertInqReqTBS,
        4,&(lvalues[4092]),0},
{"setct-ErrorTBS","setct-ErrorTBS",NID_setct_ErrorTBS,4,
        &(lvalues[4096]),0},
{"setct-PIDualSignedTBE","setct-PIDualSignedTBE",
        NID_setct_PIDualSignedTBE,4,&(lvalues[4100]),0},
{"setct-PIUnsignedTBE","setct-PIUnsignedTBE",NID_setct_PIUnsignedTBE,
        4,&(lvalues[4104]),0},
{"setct-AuthReqTBE","setct-AuthReqTBE",NID_setct_AuthReqTBE,4,
        &(lvalues[4108]),0},
{"setct-AuthResTBE","setct-AuthResTBE",NID_setct_AuthResTBE,4,
        &(lvalues[4112]),0},
{"setct-AuthResTBEX","setct-AuthResTBEX",NID_setct_AuthResTBEX,4,
        &(lvalues[4116]),0},
{"setct-AuthTokenTBE","setct-AuthTokenTBE",NID_setct_AuthTokenTBE,4,
        &(lvalues[4120]),0},
{"setct-CapTokenTBE","setct-CapTokenTBE",NID_setct_CapTokenTBE,4,
        &(lvalues[4124]),0},
{"setct-CapTokenTBEX","setct-CapTokenTBEX",NID_setct_CapTokenTBEX,4,
        &(lvalues[4128]),0},
{"setct-AcqCardCodeMsgTBE","setct-AcqCardCodeMsgTBE",
        NID_setct_AcqCardCodeMsgTBE,4,&(lvalues[4132]),0},
{"setct-AuthRevReqTBE","setct-AuthRevReqTBE",NID_setct_AuthRevReqTBE,
        4,&(lvalues[4136]),0},
{"setct-AuthRevResTBE","setct-AuthRevResTBE",NID_setct_AuthRevResTBE,
        4,&(lvalues[4140]),0},
{"setct-AuthRevResTBEB","setct-AuthRevResTBEB",
        NID_setct_AuthRevResTBEB,4,&(lvalues[4144]),0},
{"setct-CapReqTBE","setct-CapReqTBE",NID_setct_CapReqTBE,4,
        &(lvalues[4148]),0},
{"setct-CapReqTBEX","setct-CapReqTBEX",NID_setct_CapReqTBEX,4,
        &(lvalues[4152]),0},
{"setct-CapResTBE","setct-CapResTBE",NID_setct_CapResTBE,4,
        &(lvalues[4156]),0},
{"setct-CapRevReqTBE","setct-CapRevReqTBE",NID_setct_CapRevReqTBE,4,
        &(lvalues[4160]),0},
{"setct-CapRevReqTBEX","setct-CapRevReqTBEX",NID_setct_CapRevReqTBEX,
        4,&(lvalues[4164]),0},
{"setct-CapRevResTBE","setct-CapRevResTBE",NID_setct_CapRevResTBE,4,
```

```
1844         &(lvalues[4168]),0},
1845 {"setct-CredReqTBE","setct-CredReqTBE",NID_setct_CredReqTBE,4,
1846         &(lvalues[4172]),0},
1847 {"setct-CredReqTBEX","setct-CredReqTBEX",NID_setct_CredReqTBEX,4,
1848         &(lvalues[4176]),0},
1849 {"setct-CredResTBE","setct-CredResTBE",NID_setct_CredResTBE,4,
1850         &(lvalues[4180]),0},
1851 {"setct-CredRevReqTBE","setct-CredRevReqTBE",NID_setct_CredRevReqTBE,
1852         4,&(lvalues[4184]),0},
1853 {"setct-CredRevReqTBEX","setct-CredRevReqTBEX",
1854         NID_setct_CredRevReqTBEX,4,&(lvalues[4188]),0},
1855 {"setct-CredRevResTBE","setct-CredRevResTBE",NID_setct_CredRevResTBE,
1856         4,&(lvalues[4192]),0},
1857 {"setct-BatchAdminReqTBE","setct-BatchAdminReqTBE",
1858         NID_setct_BatchAdminReqTBE,4,&(lvalues[4196]),0},
1859 {"setct-BatchAdminResTBE","setct-BatchAdminResTBE",
1860         NID_setct_BatchAdminResTBE,4,&(lvalues[4200]),0},
1861 {"setct-RegFormReqTBE","setct-RegFormReqTBE",NID_setct_RegFormReqTBE,
1862         4,&(lvalues[4204]),0},
1863 {"setct-CertReqTBE","setct-CertReqTBE",NID_setct_CertReqTBE,4,
1864         &(lvalues[4208]),0},
1865 {"setct-CertReqTBEX","setct-CertReqTBEX",NID_setct_CertReqTBEX,4,
1866         &(lvalues[4212]),0},
1867 {"setct-CertResTBE","setct-CertResTBE",NID_setct_CertResTBE,4,
1868         &(lvalues[4216]),0},
1869 {"setct-CRLNotificationTBS","setct-CRLNotificationTBS",
1870         NID_setct_CRLNotificationTBS,4,&(lvalues[4220]),0},
1871 {"setct-CRLNotificationResTBS","setct-CRLNotificationResTBS",
1872         NID_setct_CRLNotificationResTBS,4,&(lvalues[4224]),0},
1873 {"setct-BCIDistributionTBS","setct-BCIDistributionTBS",
1874         NID_setct_BCIDistributionTBS,4,&(lvalues[4228]),0},
1875 {"setext-genCrypt","generic cryptogram",NID_setext_genCrypt,4,
1876         &(lvalues[4232]),0},
1877 {"setext-miAuth","merchant initiated auth",NID_setext_miAuth,4,
1878         &(lvalues[4236]),0},
1879 {"setext-pinSecure","setext-pinSecure",NID_setext_pinSecure,4,
1880         &(lvalues[4240]),0},
1881 {"setext-pinAny","setext-pinAny",NID_setext_pinAny,4,&(lvalues[4244]),0},
1882 {"setext-track2","setext-track2",NID_setext_track2,4,&(lvalues[4248]),0},
1883 {"setext-cv","additional verification",NID_setext_cv,4,
1884         &(lvalues[4252]),0},
1885 {"set-policy-root","set-policy-root",NID_set_policy_root,4,
1886         &(lvalues[4256]),0},
1887 {"setCext-hashedRoot","setCext-hashedRoot",NID_setCext_hashedRoot,4,
1888         &(lvalues[4260]),0},
1889 {"setCext-certType","setCext-certType",NID_setCext_certType,4,
1890         &(lvalues[4264]),0},
1891 {"setCext-merchData","setCext-merchData",NID_setCext_merchData,4,
1892         &(lvalues[4268]),0},
1893 {"setCext-cCertRequired","setCext-cCertRequired",
1894         NID_setCext_cCertRequired,4,&(lvalues[4272]),0},
1895 {"setCext-tunneling","setCext-tunneling",NID_setCext_tunneling,4,
1896         &(lvalues[4276]),0},
1897 {"setCext-setExt","setCext-setExt",NID_setCext_setExt,4,
1898         &(lvalues[4280]),0},
1899 {"setCext-setQualf","setCext-setQualf",NID_setCext_setQualf,4,
1900         &(lvalues[4284]),0},
1901 {"setCext-PGWYcapabilities","setCext-PGWYcapabilities",
1902         NID_setCext_PGWYcapabilities,4,&(lvalues[4288]),0},
1903 {"setCext-TokenIdentifier","setCext-TokenIdentifier",
1904         NID_setCext_TokenIdentifier,4,&(lvalues[4292]),0},
1905 {"setCext-Track2Data","setCext-Track2Data",NID_setCext_Track2Data,4,
1906         &(lvalues[4296]),0},
1907 {"setCext-TokenType","setCext-TokenType",NID_setCext_TokenType,4,
1908         &(lvalues[4300]),0},
1909 {"setCext-IssuerCapabilities","setCext-IssuerCapabilities",
```



```
1910         NID_setCext_IssuerCapabilities,4,&(lvalues[4304]),0},
1911 {"setAttr-Cert","setAttr-Cert",NID_setAttr_Cert,4,&(lvalues[4308]),0},
1912 {"setAttr-PGWYcap","payment gateway capabilities",NID_setAttr_PGWYcap,
1913         4,&(lvalues[4312]),0},
1914 {"setAttr-TokenType","setAttr-TokenType",NID_setAttr_TokenType,4,
1915         &(lvalues[4316]),0},
1916 {"setAttr-IssCap","issuer capabilities",NID_setAttr_IssCap,4,
1917         &(lvalues[4320]),0},
1918 {"set-rootKeyThumb","set-rootKeyThumb",NID_set_rootKeyThumb,5,
1919         &(lvalues[4324]),0},
1920 {"set-addPolicy","set-addPolicy",NID_set_addPolicy,5,&(lvalues[4329]),0},
1921 {"setAttr-Token-EMV","setAttr-Token-EMV",NID_setAttr_Token_EMV,5,
1922         &(lvalues[4334]),0},
1923 {"setAttr-Token-B0Prime","setAttr-Token-B0Prime",
1924         NID_setAttr_Token_B0Prime,5,&(lvalues[4339]),0},
1925 {"setAttr-IssCap-CVM","setAttr-IssCap-CVM",NID_setAttr_IssCap_CVM,5,
1926         &(lvalues[4344]),0},
1927 {"setAttr-IssCap-T2","setAttr-IssCap-T2",NID_setAttr_IssCap_T2,5,
1928         &(lvalues[4349]),0},
1929 {"setAttr-IssCap-Sig","setAttr-IssCap-Sig",NID_setAttr_IssCap_Sig,5,
1930         &(lvalues[4354]),0},
1931 {"setAttr-GenCryptgrm","generate cryptogram",NID_setAttr_GenCryptgrm,
1932         6,&(lvalues[4359]),0},
1933 {"setAttr-T2Enc","encrypted track 2",NID_setAttr_T2Enc,6,
1934         &(lvalues[4365]),0},
1935 {"setAttr-T2cleartxt","cleartext track 2",NID_setAttr_T2cleartxt,6,
1936         &(lvalues[4371]),0},
1937 {"setAttr-TokICCsig","ICC or token signature",NID_setAttr_TokICCsig,6,
1938         &(lvalues[4377]),0},
1939 {"setAttr-SecDevSig","secure device signature",NID_setAttr_SecDevSig,
1940         6,&(lvalues[4383]),0},
1941 {"set-brand-IATA-ATA","set-brand-IATA-ATA",NID_set_brand_IATA_ATA,4,
1942         &(lvalues[4389]),0},
1943 {"set-brand-Diners","set-brand-Diners",NID_set_brand_Diners,4,
1944         &(lvalues[4393]),0},
1945 {"set-brand-AmericanExpress","set-brand-AmericanExpress",
1946         NID_set_brand_AmericanExpress,4,&(lvalues[4397]),0},
1947 {"set-brand-JCB","set-brand-JCB",NID_set_brand_JCB,4,&(lvalues[4401]),0},
1948 {"set-brand-Visa","set-brand-Visa",NID_set_brand_Visa,4,
1949         &(lvalues[4405]),0},
1950 {"set-brand-MasterCard","set-brand-MasterCard",
1951         NID_set_brand_MasterCard,4,&(lvalues[4409]),0},
1952 {"set-brand-Novus","set-brand-Novus",NID_set_brand_Novus,5,
1953         &(lvalues[4413]),0},
1954 {"DES-CDMF","des-cdmf",NID_des_cdmf,8,&(lvalues[4418]),0},
1955 {"rsaOAEPEncryptionSET","rsaOAEPEncryptionSET",
1956         NID_rsaOAEPEncryptionSET,9,&(lvalues[4426]),0},
1957 {"ITU-T","itu-t",NID_itu_t,0,NULL,0},
1958 {"JOINT-ISO-ITU-T","joint-iso-itu-t",NID_joint_iso_itu_t,0,NULL,0},
1959 {"international-organizations","International Organizations",
1960         NID_international_organizations,1,&(lvalues[4435]),0},
1961 {"msSmartcardLogin","Microsoft Smartcardlogin",NID_ms_smartcard_login,
1962         10,&(lvalues[4436]),0},
1963 {"msUPN","Microsoft Universal Principal Name",NID_ms_upn,10,
1964         &(lvalues[4446]),0},
1965 {"AES-128-CFB1","aes-128-cfb1",NID_aes_128_cfb1,0,NULL,0},
1966 {"AES-192-CFB1","aes-192-cfb1",NID_aes_192_cfb1,0,NULL,0},
1967 {"AES-256-CFB1","aes-256-cfb1",NID_aes_256_cfb1,0,NULL,0},
1968 {"AES-128-CFB8","aes-128-cfb8",NID_aes_128_cfb8,0,NULL,0},
1969 {"AES-192-CFB8","aes-192-cfb8",NID_aes_192_cfb8,0,NULL,0},
1970 {"AES-256-CFB8","aes-256-cfb8",NID_aes_256_cfb8,0,NULL,0},
1971 {"DES-CFB1","des-cfb1",NID_des_cfb1,0,NULL,0},
1972 {"DES-CFB8","des-cfb8",NID_des_cfb8,0,NULL,0},
1973 {"DES-EDE3-CFB1","des-ede3-cfb1",NID_des_ede3_cfb1,0,NULL,0},
1974 {"DES-EDE3-CFB8","des-ede3-cfb8",NID_des_ede3_cfb8,0,NULL,0},
1975 {"street","streetAddress",NID_streetAddress,3,&(lvalues[4456]),0},
```

```
{"postalCode","postalCode",NID_postalCode,3,&(lvalues[4459]),0},
{"id-ppl","id-ppl",NID_id_ppl,7,&(lvalues[4462]),0},
{"proxyCertInfo","Proxy Certificate Information",NID_proxyCertInfo,8,
        &(lvalues[4469]),0},
{"id-ppl-anyLanguage","Any language",NID_id_ppl_anyLanguage,8,
        &(lvalues[4477]),0},
{"id-ppl-inheritAll","Inherit all",NID_id_ppl_inheritAll,8,
        &(lvalues[4485]),0},
{"nameConstraints","X509v3 Name Constraints",NID_name_constraints,3,
        &(lvalues[4493]),0},
{"id-ppl-independent","Independent",NID_Independent,8,&(lvalues[4496]),0},
{"RSA-SHA256","sha256WithRSAEncryption",NID_sha256WithRSAEncryption,9,
        &(lvalues[4504]),0},
{"RSA-SHA384","sha384WithRSAEncryption",NID_sha384WithRSAEncryption,9,
        &(lvalues[4513]),0},
{"RSA-SHA512","sha512WithRSAEncryption",NID_sha512WithRSAEncryption,9,
        &(lvalues[4522]),0},
{"RSA-SHA224","sha224WithRSAEncryption",NID_sha224WithRSAEncryption,9,
        &(lvalues[4531]),0},
{"SHA256","sha256",NID_sha256,9,&(lvalues[4540]),0},
{"SHA384","sha384",NID_sha384,9,&(lvalues[4549]),0},
{"SHA512","sha512",NID_sha512,9,&(lvalues[4558]),0},
{"SHA224","sha224",NID_sha224,9,&(lvalues[4567]),0},
{"identified-organization","identified-organization",
        NID_identified_organization,1,&(lvalues[4576]),0},
{"certicom-arc","certicom-arc",NID_certicom_arc,3,&(lvalues[4577]),0},
{"wap","wap",NID_wap,2,&(lvalues[4580]),0},
{"wap-wsg","wap-wsg",NID_wap_wsg,3,&(lvalues[4582]),0},
{"id-characteristic-two-basis","id-characteristic-two-basis",
        NID_X9_62_id_characteristic_two_basis,8,&(lvalues[4585]),0},
{"onBasis","onBasis",NID_X9_62_onBasis,9,&(lvalues[4593]),0},
{"tpBasis","tpBasis",NID_X9_62_tpBasis,9,&(lvalues[4602]),0},
{"ppBasis","ppBasis",NID_X9_62_ppBasis,9,&(lvalues[4611]),0},
{"c2pnb163v1","c2pnb163v1",NID_X9_62_c2pnb163v1,8,&(lvalues[4620]),0},
{"c2pnb163v2","c2pnb163v2",NID_X9_62_c2pnb163v2,8,&(lvalues[4628]),0},
{"c2pnb163v3","c2pnb163v3",NID_X9_62_c2pnb163v3,8,&(lvalues[4636]),0},
{"c2pnb176v1","c2pnb176v1",NID_X9_62_c2pnb176v1,8,&(lvalues[4644]),0},
{"c2tnb191v1","c2tnb191v1",NID_X9_62_c2tnb191v1,8,&(lvalues[4652]),0},
{"c2tnb191v2","c2tnb191v2",NID_X9_62_c2tnb191v2,8,&(lvalues[4660]),0},
{"c2tnb191v3","c2tnb191v3",NID_X9_62_c2tnb191v3,8,&(lvalues[4668]),0},
{"c2onb191v4","c2onb191v4",NID_X9_62_c2onb191v4,8,&(lvalues[4676]),0},
{"c2onb191v5","c2onb191v5",NID_X9_62_c2onb191v5,8,&(lvalues[4684]),0},
{"c2pnb208w1","c2pnb208w1",NID_X9_62_c2pnb208w1,8,&(lvalues[4692]),0},
{"c2tnb239v1","c2tnb239v1",NID_X9_62_c2tnb239v1,8,&(lvalues[4700]),0},
{"c2tnb239v2","c2tnb239v2",NID_X9_62_c2tnb239v2,8,&(lvalues[4708]),0},
{"c2tnb239v3","c2tnb239v3",NID_X9_62_c2tnb239v3,8,&(lvalues[4716]),0},
{"c2onb239v4","c2onb239v4",NID_X9_62_c2onb239v4,8,&(lvalues[4724]),0},
{"c2onb239v5","c2onb239v5",NID_X9_62_c2onb239v5,8,&(lvalues[4732]),0},
{"c2pnb272w1","c2pnb272w1",NID_X9_62_c2pnb272w1,8,&(lvalues[4740]),0},
{"c2pnb304w1","c2pnb304w1",NID_X9_62_c2pnb304w1,8,&(lvalues[4748]),0},
{"c2tnb359v1","c2tnb359v1",NID_X9_62_c2tnb359v1,8,&(lvalues[4756]),0},
{"c2pnb368w1","c2pnb368w1",NID_X9_62_c2pnb368w1,8,&(lvalues[4764]),0},
{"c2tnb431r1","c2tnb431r1",NID_X9_62_c2tnb431r1,8,&(lvalues[4772]),0},
{"secp112r1","secp112r1",NID_secp112r1,5,&(lvalues[4780]),0},
{"secp112r2","secp112r2",NID_secp112r2,5,&(lvalues[4785]),0},
{"secp128r1","secp128r1",NID_secp128r1,5,&(lvalues[4790]),0},
{"secp128r2","secp128r2",NID_secp128r2,5,&(lvalues[4795]),0},
{"secp160k1","secp160k1",NID_secp160k1,5,&(lvalues[4800]),0},
{"secp160r1","secp160r1",NID_secp160r1,5,&(lvalues[4805]),0},
{"secp160r2","secp160r2",NID_secp160r2,5,&(lvalues[4810]),0},
{"secp192k1","secp192k1",NID_secp192k1,5,&(lvalues[4815]),0},
{"secp224k1","secp224k1",NID_secp224k1,5,&(lvalues[4820]),0},
{"secp224r1","secp224r1",NID_secp224r1,5,&(lvalues[4825]),0},
{"secp256k1","secp256k1",NID_secp256k1,5,&(lvalues[4830]),0},
{"secp384r1","secp384r1",NID_secp384r1,5,&(lvalues[4835]),0},
{"secp521r1","secp521r1",NID_secp521r1,5,&(lvalues[4840]),0},
```



```
{"sect113r1","sect113r1",NID_sect113r1,5,&(lvalues[4845]),0},
{"sect113r2","sect113r2",NID_sect113r2,5,&(lvalues[4850]),0},
{"sect131r1","sect131r1",NID_sect131r1,5,&(lvalues[4855]),0},
{"sect131r2","sect131r2",NID_sect131r2,5,&(lvalues[4860]),0},
{"sect163k1","sect163k1",NID_sect163k1,5,&(lvalues[4865]),0},
{"sect163r1","sect163r1",NID_sect163r1,5,&(lvalues[4870]),0},
{"sect163r2","sect163r2",NID_sect163r2,5,&(lvalues[4875]),0},
{"sect193r1","sect193r1",NID_sect193r1,5,&(lvalues[4880]),0},
{"sect193r2","sect193r2",NID_sect193r2,5,&(lvalues[4885]),0},
{"sect233k1","sect233k1",NID_sect233k1,5,&(lvalues[4890]),0},
{"sect233r1","sect233r1",NID_sect233r1,5,&(lvalues[4895]),0},
{"sect239k1","sect239k1",NID_sect239k1,5,&(lvalues[4900]),0},
{"sect283k1","sect283k1",NID_sect283k1,5,&(lvalues[4905]),0},
{"sect283r1","sect283r1",NID_sect283r1,5,&(lvalues[4910]),0},
{"sect409k1","sect409k1",NID_sect409k1,5,&(lvalues[4915]),0},
{"sect409r1","sect409r1",NID_sect409r1,5,&(lvalues[4920]),0},
{"sect571k1","sect571k1",NID_sect571k1,5,&(lvalues[4925]),0},
{"sect571r1","sect571r1",NID_sect571r1,5,&(lvalues[4930]),0},
{"wap-wsg-idm-ecid-wtls1","wap-wsg-idm-ecid-wtls1",
        NID_wap_wsg_idm_ecid_wtls1,5,&(lvalues[4935]),0},
{"wap-wsg-idm-ecid-wtls3","wap-wsg-idm-ecid-wtls3",
        NID_wap_wsg_idm_ecid_wtls3,5,&(lvalues[4940]),0},
{"wap-wsg-idm-ecid-wtls4","wap-wsg-idm-ecid-wtls4",
        NID_wap_wsg_idm_ecid_wtls4,5,&(lvalues[4945]),0},
{"wap-wsg-idm-ecid-wtls5","wap-wsg-idm-ecid-wtls5",
        NID_wap_wsg_idm_ecid_wtls5,5,&(lvalues[4950]),0},
{"wap-wsg-idm-ecid-wtls6","wap-wsg-idm-ecid-wtls6",
        NID_wap_wsg_idm_ecid_wtls6,5,&(lvalues[4955]),0},
{"wap-wsg-idm-ecid-wtls7","wap-wsg-idm-ecid-wtls7",
        NID_wap_wsg_idm_ecid_wtls7,5,&(lvalues[4960]),0},
{"wap-wsg-idm-ecid-wtls8","wap-wsg-idm-ecid-wtls8",
        NID_wap_wsg_idm_ecid_wtls8,5,&(lvalues[4965]),0},
{"wap-wsg-idm-ecid-wtls9","wap-wsg-idm-ecid-wtls9",
        NID_wap_wsg_idm_ecid_wtls9,5,&(lvalues[4970]),0},
{"wap-wsg-idm-ecid-wtls10","wap-wsg-idm-ecid-wtls10",
        NID_wap_wsg_idm_ecid_wtls10,5,&(lvalues[4975]),0},
{"wap-wsg-idm-ecid-wtls11","wap-wsg-idm-ecid-wtls11",
        NID_wap_wsg_idm_ecid_wtls11,5,&(lvalues[4980]),0},
{"wap-wsg-idm-ecid-wtls12","wap-wsg-idm-ecid-wtls12",
        NID_wap_wsg_idm_ecid_wtls12,5,&(lvalues[4985]),0},
{"anyPolicy","X509v3 Any Policy",NID_any_policy,4,&(lvalues[4990]),0},
{"policyMappings","X509v3 Policy Mappings",NID_policy_mappings,3,
        &(lvalues[4994]),0},
{"inhibitAnyPolicy","X509v3 Inhibit Any Policy",
        NID_inhibit_any_policy,3,&(lvalues[4997]),0},
{"Oakley-EC2N-3","ipsec3",NID_ipsec3,0,NULL,0},
{"Oakley-EC2N-4","ipsec4",NID_ipsec4,0,NULL,0},
{"CAMELLIA-128-CBC","camellia-128-cbc",NID_camellia_128_cbc,11,
        &(lvalues[5000]),0},
{"CAMELLIA-192-CBC","camellia-192-cbc",NID_camellia_192_cbc,11,
        &(lvalues[5011]),0},
{"CAMELLIA-256-CBC","camellia-256-cbc",NID_camellia_256_cbc,11,
        &(lvalues[5022]),0},
{"CAMELLIA-128-ECB","camellia-128-ecb",NID_camellia_128_ecb,8,
        &(lvalues[5033]),0},
{"CAMELLIA-192-ECB","camellia-192-ecb",NID_camellia_192_ecb,8,
        &(lvalues[5041]),0},
{"CAMELLIA-256-ECB","camellia-256-ecb",NID_camellia_256_ecb,8,
        &(lvalues[5049]),0},
{"CAMELLIA-128-CFB","camellia-128-cfb",NID_camellia_128_cfb128,8,
        &(lvalues[5057]),0},
{"CAMELLIA-192-CFB","camellia-192-cfb",NID_camellia_192_cfb128,8,
        &(lvalues[5065]),0},
{"CAMELLIA-256-CFB","camellia-256-cfb",NID_camellia_256_cfb128,8,
        &(lvalues[5073]),0},
{"CAMELLIA-128-CFB1","camellia-128-cfb1",NID_camellia_128_cfb1,0,NULL,0},
```

```
{"CAMELLIA-192-CFB1","camellia-192-cfb1",NID_camellia_192_cfb1,0,NULL,0},
{"CAMELLIA-256-CFB1","camellia-256-cfb1",NID_camellia_256_cfb1,0,NULL,0},
{"CAMELLIA-128-CFB8","camellia-128-cfb8",NID_camellia_128_cfb8,0,NULL,0},
{"CAMELLIA-192-CFB8","camellia-192-cfb8",NID_camellia_192_cfb8,0,NULL,0},
{"CAMELLIA-256-CFB8","camellia-256-cfb8",NID_camellia_256_cfb8,0,NULL,0},
{"CAMELLIA-128-OFB","camellia-128-ofb",NID_camellia_128_ofb128,8,
        &(lvalues[5081]),0},
{"CAMELLIA-192-OFB","camellia-192-ofb",NID_camellia_192_ofb128,8,
        &(lvalues[5089]),0},
{"CAMELLIA-256-OFB","camellia-256-ofb",NID_camellia_256_ofb128,8,
        &(lvalues[5097]),0},
{"subjectDirectoryAttributes","X509v3 Subject Directory Attributes",
        NID_subject_directory_attributes,3,&(lvalues[5105]),0},
{"issuingDistributionPoint","X509v3 Issuing Distrubution Point",
        NID_issuing_distribution_point,3,&(lvalues[5108]),0},
{"certificateIssuer","X509v3 Certificate Issuer",
        NID_certificate_issuer,3,&(lvalues[5111]),0},
{NULL,NULL,NID_undef,0,NULL,0},
{"KISA","kisa",NID_kisa,6,&(lvalues[5114]),0},
{NULL,NULL,NID_undef,0,NULL,0},
{NULL,NULL,NID_undef,0,NULL,0},
{"SEED-ECB","seed-ecb",NID_seed_ecb,8,&(lvalues[5120]),0},
{"SEED-CBC","seed-cbc",NID_seed_cbc,8,&(lvalues[5128]),0},
{"SEED-OFB","seed-ofb",NID_seed_ofb128,8,&(lvalues[5136]),0},
{"SEED-CFB","seed-cfb",NID_seed_cfb128,8,&(lvalues[5144]),0},
{"HMAC-MD5","hmac-md5",NID_hmac_md5,8,&(lvalues[5152]),0},
{"HMAC-SHA1","hmac-sha1",NID_hmac_sha1,8,&(lvalues[5160]),0},
{"id-PasswordBasedMAC","password based MAC",NID_id_PasswordBasedMAC,9,
        &(lvalues[5168]),0},
{"id-DHBasedMac","Diffie-Hellman based MAC",NID_id_DHBasedMac,9,
        &(lvalues[5177]),0},
{"id-it-suppLangTags","id-it-suppLangTags",NID_id_it_suppLangTags,8,
        &(lvalues[5186]),0},
{"caRepository","CA Repository",NID_caRepository,8,&(lvalues[5194]),0},
{"id-smime-ct-compressedData","id-smime-ct-compressedData",
        NID_id_smime_ct_compressedData,11,&(lvalues[5202]),0},
{"id-ct-asciiTextWithCRLF","id-ct-asciiTextWithCRLF",
        NID_id_ct_asciiTextWithCRLF,11,&(lvalues[5213]),0},
{"id-aes128-wrap","id-aes128-wrap",NID_id_aes128_wrap,9,
        &(lvalues[5224]),0},
{"id-aes192-wrap","id-aes192-wrap",NID_id_aes192_wrap,9,
        &(lvalues[5233]),0},
{"id-aes256-wrap","id-aes256-wrap",NID_id_aes256_wrap,9,
        &(lvalues[5242]),0},
{"ecdsa-with-Recommended","ecdsa-with-Recommended",
        NID_ecdsa_with_Recommended,7,&(lvalues[5251]),0},
{"ecdsa-with-Specified","ecdsa-with-Specified",
        NID_ecdsa_with_Specified,7,&(lvalues[5258]),0},
{"ecdsa-with-SHA224","ecdsa-with-SHA224",NID_ecdsa_with_SHA224,8,
        &(lvalues[5265]),0},
{"ecdsa-with-SHA256","ecdsa-with-SHA256",NID_ecdsa_with_SHA256,8,
        &(lvalues[5273]),0},
{"ecdsa-with-SHA384","ecdsa-with-SHA384",NID_ecdsa_with_SHA384,8,
        &(lvalues[5281]),0},
{"ecdsa-with-SHA512","ecdsa-with-SHA512",NID_ecdsa_with_SHA512,8,
        &(lvalues[5289]),0},
{"hmacWithMD5","hmacWithMD5",NID_hmacWithMD5,8,&(lvalues[5297]),0},
{"hmacWithSHA224","hmacWithSHA224",NID_hmacWithSHA224,8,
        &(lvalues[5305]),0},
{"hmacWithSHA256","hmacWithSHA256",NID_hmacWithSHA256,8,
        &(lvalues[5313]),0},
{"hmacWithSHA384","hmacWithSHA384",NID_hmacWithSHA384,8,
        &(lvalues[5321]),0},
{"hmacWithSHA512","hmacWithSHA512",NID_hmacWithSHA512,8,
        &(lvalues[5329]),0},
{"dsa_with_SHA224","dsa_with_SHA224",NID_dsa_with_SHA224,9,
```



```
        &(lvalues[5337]),0},
{"dsa_with_SHA256","dsa_with_SHA256",NID_dsa_with_SHA256,9,
        &(lvalues[5346]),0},
{"whirlpool","whirlpool",NID_whirlpool,6,&(lvalues[5355]),0},
{"cryptopro","cryptopro",NID_cryptopro,5,&(lvalues[5361]),0},
{"cryptocom","cryptocom",NID_cryptocom,5,&(lvalues[5366]),0},
{"id-GostR3411-94-with-GostR3410-2001",
        "GOST R 34.11-94 with GOST R 34.10-2001",
        NID_id_GostR3411_94_with_GostR3410_2001,6,&(lvalues[5371]),0},
{"id-GostR3411-94-with-GostR3410-94",
        "GOST R 34.11-94 with GOST R 34.10-94",
        NID_id_GostR3411_94_with_GostR3410_94,6,&(lvalues[5377]),0},
{"md_gost94","GOST R 34.11-94",NID_id_GostR3411_94,6,&(lvalues[5383]),0},
{"id-HMACGostR3411-94","HMAC GOST 34.11-94",NID_id_HMACGostR3411_94,6,
        &(lvalues[5389]),0},
{"gost2001","GOST R 34.10-2001",NID_id_GostR3410_2001,6,
        &(lvalues[5395]),0},
{"gost94","GOST R 34.10-94",NID_id_GostR3410_94,6,&(lvalues[5401]),0},
{"gost89","GOST 28147-89",NID_id_Gost28147_89,6,&(lvalues[5407]),0},
{"gost89-cnt","gost89-cnt",NID_gost89_cnt,0,NULL,0},
{"gost-mac","GOST 28147-89 MAC",NID_id_Gost28147_89_MAC,6,
        &(lvalues[5413]),0},
{"prf-gostr3411-94","GOST R 34.11-94 PRF",NID_id_GostR3411_94_prf,6,
        &(lvalues[5419]),0},
{"id-GostR3410-2001DH","GOST R 34.10-2001 DH",NID_id_GostR3410_2001DH,
        6,&(lvalues[5425]),0},
{"id-GostR3410-94DH","GOST R 34.10-94 DH",NID_id_GostR3410_94DH,6,
        &(lvalues[5431]),0},
{"id-Gost28147-89-CryptoPro-KeyMeshing",
        "id-Gost28147-89-CryptoPro-KeyMeshing",
        NID_id_Gost28147_89_CryptoPro_KeyMeshing,7,&(lvalues[5437]),0},
{"id-Gost28147-89-None-KeyMeshing","id-Gost28147-89-None-KeyMeshing",
        NID_id_Gost28147_89_None_KeyMeshing,7,&(lvalues[5444]),0},
{"id-GostR3411-94-TestParamSet","id-GostR3411-94-TestParamSet",
        NID_id_GostR3411_94_TestParamSet,7,&(lvalues[5451]),0},
{"id-GostR3411-94-CryptoProParamSet",
        "id-GostR3411-94-CryptoProParamSet",
        NID_id_GostR3411_94_CryptoProParamSet,7,&(lvalues[5458]),0},
{"id-Gost28147-89-TestParamSet","id-Gost28147-89-TestParamSet",
        NID_id_Gost28147_89_TestParamSet,7,&(lvalues[5465]),0},
{"id-Gost28147-89-CryptoPro-A-ParamSet",
        "id-Gost28147-89-CryptoPro-A-ParamSet",
        NID_id_Gost28147_89_CryptoPro_A_ParamSet,7,&(lvalues[5472]),0},
{"id-Gost28147-89-CryptoPro-B-ParamSet",
        "id-Gost28147-89-CryptoPro-B-ParamSet",
        NID_id_Gost28147_89_CryptoPro_B_ParamSet,7,&(lvalues[5479]),0},
{"id-Gost28147-89-CryptoPro-C-ParamSet",
        "id-Gost28147-89-CryptoPro-C-ParamSet",
        NID_id_Gost28147_89_CryptoPro_C_ParamSet,7,&(lvalues[5486]),0},
{"id-Gost28147-89-CryptoPro-D-ParamSet",
        "id-Gost28147-89-CryptoPro-D-ParamSet",
        NID_id_Gost28147_89_CryptoPro_D_ParamSet,7,&(lvalues[5493]),0},
{"id-Gost28147-89-CryptoPro-Oscar-1-1-ParamSet",
        "id-Gost28147-89-CryptoPro-Oscar-1-1-ParamSet",
        NID_id_Gost28147_89_CryptoPro_Oscar_1_1_ParamSet,7,&(lvalues[5500]),
        0},
{"id-Gost28147-89-CryptoPro-Oscar-1-0-ParamSet",
        "id-Gost28147-89-CryptoPro-Oscar-1-0-ParamSet",
        NID_id_Gost28147_89_CryptoPro_Oscar_1_0_ParamSet,7,&(lvalues[5507]),
        0},
{"id-Gost28147-89-CryptoPro-RIC-1-ParamSet",
        "id-Gost28147-89-CryptoPro-RIC-1-ParamSet",
        NID_id_Gost28147_89_CryptoPro_RIC_1_ParamSet,7,&(lvalues[5514]),0},
{"id-GostR3410-94-TestParamSet","id-GostR3410-94-TestParamSet",
        NID_id_GostR3410_94_TestParamSet,7,&(lvalues[5521]),0},
{"id-GostR3410-94-CryptoPro-A-ParamSet",
```

```
2240         "id-GostR3410-94-CryptoPro-A-ParamSet",
2241         NID_id_GostR3410_94_CryptoPro_A_ParamSet,7,&(lvalues[5528]),0},
2242 {"id-GostR3410-94-CryptoPro-B-ParamSet",
2243         "id-GostR3410-94-CryptoPro-B-ParamSet",
2244         NID_id_GostR3410_94_CryptoPro_B_ParamSet,7,&(lvalues[5535]),0},
2245 {"id-GostR3410-94-CryptoPro-C-ParamSet",
2246         "id-GostR3410-94-CryptoPro-C-ParamSet",
2247         NID_id_GostR3410_94_CryptoPro_C_ParamSet,7,&(lvalues[5542]),0},
2248 {"id-GostR3410-94-CryptoPro-D-ParamSet",
2249         "id-GostR3410-94-CryptoPro-D-ParamSet",
2250         NID_id_GostR3410_94_CryptoPro_D_ParamSet,7,&(lvalues[5549]),0},
2251 {"id-GostR3410-94-CryptoPro-XchA-ParamSet",
2252         "id-GostR3410-94-CryptoPro-XchA-ParamSet",
2253         NID_id_GostR3410_94_CryptoPro_XchA_ParamSet,7,&(lvalues[5556]),0},
2254 {"id-GostR3410-94-CryptoPro-XchB-ParamSet",
2255         "id-GostR3410-94-CryptoPro-XchB-ParamSet",
2256         NID_id_GostR3410_94_CryptoPro_XchB_ParamSet,7,&(lvalues[5563]),0},
2257 {"id-GostR3410-94-CryptoPro-XchC-ParamSet",
2258         "id-GostR3410-94-CryptoPro-XchC-ParamSet",
2259         NID_id_GostR3410_94_CryptoPro_XchC_ParamSet,7,&(lvalues[5570]),0},
2260 {"id-GostR3410-2001-TestParamSet","id-GostR3410-2001-TestParamSet",
2261         NID_id_GostR3410_2001_TestParamSet,7,&(lvalues[5577]),0},
2262 {"id-GostR3410-2001-CryptoPro-A-ParamSet",
2263         "id-GostR3410-2001-CryptoPro-A-ParamSet",
2264         NID_id_GostR3410_2001_CryptoPro_A_ParamSet,7,&(lvalues[5584]),0},
2265 {"id-GostR3410-2001-CryptoPro-B-ParamSet",
2266         "id-GostR3410-2001-CryptoPro-B-ParamSet",
2267         NID_id_GostR3410_2001_CryptoPro_B_ParamSet,7,&(lvalues[5591]),0},
2268 {"id-GostR3410-2001-CryptoPro-C-ParamSet",
2269         "id-GostR3410-2001-CryptoPro-C-ParamSet",
2270         NID_id_GostR3410_2001_CryptoPro_C_ParamSet,7,&(lvalues[5598]),0},
2271 {"id-GostR3410-2001-CryptoPro-XchA-ParamSet",
2272         "id-GostR3410-2001-CryptoPro-XchA-ParamSet",
2273         NID_id_GostR3410_2001_CryptoPro_XchA_ParamSet,7,&(lvalues[5605]),0},

2275 {"id-GostR3410-2001-CryptoPro-XchB-ParamSet",
2276         "id-GostR3410-2001-CryptoPro-XchB-ParamSet",
2277         NID_id_GostR3410_2001_CryptoPro_XchB_ParamSet,7,&(lvalues[5612]),0},

2279 {"id-GostR3410-94-a","id-GostR3410-94-a",NID_id_GostR3410_94_a,7,
2280         &(lvalues[5619]),0},
2281 {"id-GostR3410-94-aBis","id-GostR3410-94-aBis",
2282         NID_id_GostR3410_94_aBis,7,&(lvalues[5626]),0},
2283 {"id-GostR3410-94-b","id-GostR3410-94-b",NID_id_GostR3410_94_b,7,
2284         &(lvalues[5633]),0},
2285 {"id-GostR3410-94-bBis","id-GostR3410-94-bBis",
2286         NID_id_GostR3410_94_bBis,7,&(lvalues[5640]),0},
2287 {"id-Gost28147-89-cc","GOST 28147-89 Cryptocom ParamSet",
2288         NID_id_Gost28147_89_cc,8,&(lvalues[5647]),0},
2289 {"gost94cc","GOST 34.10-94 Cryptocom",NID_id_GostR3410_94_cc,8,
2290         &(lvalues[5655]),0},
2291 {"gost2001cc","GOST 34.10-2001 Cryptocom",NID_id_GostR3410_2001_cc,8,
2292         &(lvalues[5663]),0},
2293 {"id-GostR3411-94-with-GostR3410-94-cc",
2294         "GOST R 34.11-94 with GOST R 34.10-94 Cryptocom",
2295         NID_id_GostR3411_94_with_GostR3410_94_cc,8,&(lvalues[5671]),0},
2296 {"id-GostR3411-94-with-GostR3410-2001-cc",
2297         "GOST R 34.11-94 with GOST R 34.10-2001 Cryptocom",
2298         NID_id_GostR3411_94_with_GostR3410_2001_cc,8,&(lvalues[5679]),0},
2299 {"id-GostR3410-2001-ParamSet-cc",
2300         "GOST R 3410-2001 Parameter Set Cryptocom",
2301         NID_id_GostR3410_2001_ParamSet_cc,8,&(lvalues[5687]),0},
2302 {"HMAC","hmac",NID_hmac,0,NULL,0},
2303 {"LocalKeySet","Microsoft Local Key set",NID_LocalKeySet,9,
2304         &(lvalues[5695]),0},
2305 {"freshestCRL","X509v3 Freshest CRL",NID_freshest_crl,3,
```



```
2306         &(lvalues[5704]),0},
2307 {"id-on-permanentIdentifier","Permanent Identifier",
2308         NID_id_on_permanentIdentifier,8,&(lvalues[5707]),0},
2309 {"searchGuide","searchGuide",NID_searchGuide,3,&(lvalues[5715]),0},
2310 {"businessCategory","businessCategory",NID_businessCategory,3,
2311         &(lvalues[5718]),0},
2312 {"postalAddress","postalAddress",NID_postalAddress,3,&(lvalues[5721]),0},
2313 {"postOfficeBox","postOfficeBox",NID_postOfficeBox,3,&(lvalues[5724]),0},
2314 {"physicalDeliveryOfficeName","physicalDeliveryOfficeName",
2315         NID_physicalDeliveryOfficeName,3,&(lvalues[5727]),0},
2316 {"telephoneNumber","telephoneNumber",NID_telephoneNumber,3,
2317         &(lvalues[5730]),0},
2318 {"telexNumber","telexNumber",NID_telexNumber,3,&(lvalues[5733]),0},
2319 {"teletexTerminalIdentifier","teletexTerminalIdentifier",
2320         NID_teletexTerminalIdentifier,3,&(lvalues[5736]),0},
2321 {"facsimileTelephoneNumber","facsimileTelephoneNumber",
2322         NID_facsimileTelephoneNumber,3,&(lvalues[5739]),0},
2323 {"x121Address","x121Address",NID_x121Address,3,&(lvalues[5742]),0},
2324 {"internationaliSDNNumber","internationaliSDNNumber",
2325         NID_internationaliSDNNumber,3,&(lvalues[5745]),0},
2326 {"registeredAddress","registeredAddress",NID_registeredAddress,3,
2327         &(lvalues[5748]),0},
2328 {"destinationIndicator","destinationIndicator",
2329         NID_destinationIndicator,3,&(lvalues[5751]),0},
2330 {"preferredDeliveryMethod","preferredDeliveryMethod",
2331         NID_preferredDeliveryMethod,3,&(lvalues[5754]),0},
2332 {"presentationAddress","presentationAddress",NID_presentationAddress,
2333         3,&(lvalues[5757]),0},
2334 {"supportedApplicationContext","supportedApplicationContext",
2335         NID_supportedApplicationContext,3,&(lvalues[5760]),0},
2336 {"member","member",NID_member,3,&(lvalues[5763]),0},
2337 {"owner","owner",NID_owner,3,&(lvalues[5766]),0},
2338 {"roleOccupant","roleOccupant",NID_roleOccupant,3,&(lvalues[5769]),0},
2339 {"seeAlso","seeAlso",NID_seeAlso,3,&(lvalues[5772]),0},
2340 {"userPassword","userPassword",NID_userPassword,3,&(lvalues[5775]),0},
2341 {"userCertificate","userCertificate",NID_userCertificate,3,
2342         &(lvalues[5778]),0},
2343 {"cACertificate","cACertificate",NID_cACertificate,3,&(lvalues[5781]),0},
2344 {"authorityRevocationList","authorityRevocationList",
2345         NID_authorityRevocationList,3,&(lvalues[5784]),0},
2346 {"certificateRevocationList","certificateRevocationList",
2347         NID_certificateRevocationList,3,&(lvalues[5787]),0},
2348 {"crossCertificatePair","crossCertificatePair",
2349         NID_crossCertificatePair,3,&(lvalues[5790]),0},
2350 {"enhancedSearchGuide","enhancedSearchGuide",NID_enhancedSearchGuide,
2351         3,&(lvalues[5793]),0},
2352 {"protocolInformation","protocolInformation",NID_protocolInformation,
2353         3,&(lvalues[5796]),0},
2354 {"distinguishedName","distinguishedName",NID_distinguishedName,3,
2355         &(lvalues[5799]),0},
2356 {"uniqueMember","uniqueMember",NID_uniqueMember,3,&(lvalues[5802]),0},
2357 {"houseIdentifier","houseIdentifier",NID_houseIdentifier,3,
2358         &(lvalues[5805]),0},
2359 {"supportedAlgorithms","supportedAlgorithms",NID_supportedAlgorithms,
2360         3,&(lvalues[5808]),0},
2361 {"deltaRevocationList","deltaRevocationList",NID_deltaRevocationList,
2362         3,&(lvalues[5811]),0},
2363 {"dmdName","dmdName",NID_dmdName,3,&(lvalues[5814]),0},
2364 {"id-alg-PWRI-KEK","id-alg-PWRI-KEK",NID_id_alg_PWRI_KEK,11,
2365         &(lvalues[5817]),0},
2366 {"CMAC","cmac",NID_cmac,0,NULL,0},
2367 {"id-aes128-GCM","aes-128-gcm",NID_aes_128_gcm,9,&(lvalues[5828]),0},
2368 {"id-aes128-CCM","aes-128-ccm",NID_aes_128_ccm,9,&(lvalues[5837]),0},
2369 {"id-aes128-wrap-pad","id-aes128-wrap-pad",NID_id_aes128_wrap_pad,9,
2370         &(lvalues[5846]),0},
2371 {"id-aes192-GCM","aes-192-gcm",NID_aes_192_gcm,9,&(lvalues[5855]),0},
```

```
{"id-aes192-CCM","aes-192-ccm",NID_aes_192_ccm,9,&(lvalues[5864]),0},
{"id-aes192-wrap-pad","id-aes192-wrap-pad",NID_id_aes192_wrap_pad,9,
        &(lvalues[5873]),0},
{"id-aes256-GCM","aes-256-gcm",NID_aes_256_gcm,9,&(lvalues[5882]),0},
{"id-aes256-CCM","aes-256-ccm",NID_aes_256_ccm,9,&(lvalues[5891]),0},
{"id-aes256-wrap-pad","id-aes256-wrap-pad",NID_id_aes256_wrap_pad,9,
        &(lvalues[5900]),0},
{"AES-128-CTR","aes-128-ctr",NID_aes_128_ctr,0,NULL,0},
{"AES-192-CTR","aes-192-ctr",NID_aes_192_ctr,0,NULL,0},
{"AES-256-CTR","aes-256-ctr",NID_aes_256_ctr,0,NULL,0},
{"id-camellia128-wrap","id-camellia128-wrap",NID_id_camellia128_wrap,
        11,&(lvalues[5909]),0},
{"id-camellia192-wrap","id-camellia192-wrap",NID_id_camellia192_wrap,
        11,&(lvalues[5920]),0},
{"id-camellia256-wrap","id-camellia256-wrap",NID_id_camellia256_wrap,
        11,&(lvalues[5931]),0},
{"anyExtendedKeyUsage","Any Extended Key Usage",
        NID_anyExtendedKeyUsage,4,&(lvalues[5942]),0},
{"MGF1","mgf1",NID_mgf1,9,&(lvalues[5946]),0},
{"RSASSA-PSS","rsassaPss",NID_rsassaPss,9,&(lvalues[5955]),0},
{"AES-128-XTS","aes-128-xts",NID_aes_128_xts,0,NULL,0},
{"AES-256-XTS","aes-256-xts",NID_aes_256_xts,0,NULL,0},
{"RC4-HMAC-MD5","rc4-hmac-md5",NID_rc4_hmac_md5,0,NULL,0},
{"AES-128-CBC-HMAC-SHA1","aes-128-cbc-hmac-sha1",
        NID_aes_128_cbc_hmac_sha1,0,NULL,0},
{"AES-192-CBC-HMAC-SHA1","aes-192-cbc-hmac-sha1",
        NID_aes_192_cbc_hmac_sha1,0,NULL,0},
{"AES-256-CBC-HMAC-SHA1","aes-256-cbc-hmac-sha1",
        NID_aes_256_cbc_hmac_sha1,0,NULL,0},
{"RSAES-OAEP","rsaesOaep",NID_rsaesOaep,9,&(lvalues[5964]),0},
};

static const unsigned int sn_objs[NUM_SN]={
364,    /* "AD_DVCS" */
419,    /* "AES-128-CBC" */
916,    /* "AES-128-CBC-HMAC-SHA1" */
421,    /* "AES-128-CFB" */
650,    /* "AES-128-CFB1" */
653,    /* "AES-128-CFB8" */
904,    /* "AES-128-CTR" */
418,    /* "AES-128-ECB" */
420,    /* "AES-128-OFB" */
913,    /* "AES-128-XTS" */
423,    /* "AES-192-CBC" */
917,    /* "AES-192-CBC-HMAC-SHA1" */
425,    /* "AES-192-CFB" */
651,    /* "AES-192-CFB1" */
654,    /* "AES-192-CFB8" */
905,    /* "AES-192-CTR" */
422,    /* "AES-192-ECB" */
424,    /* "AES-192-OFB" */
427,    /* "AES-256-CBC" */
918,    /* "AES-256-CBC-HMAC-SHA1" */
429,    /* "AES-256-CFB" */
652,    /* "AES-256-CFB1" */
655,    /* "AES-256-CFB8" */
906,    /* "AES-256-CTR" */
426,    /* "AES-256-ECB" */
428,    /* "AES-256-OFB" */
914,    /* "AES-256-XTS" */
91,     /* "BF-CBC" */
93,     /* "BF-CFB" */
92,     /* "BF-ECB" */
94,     /* "BF-OFB" */
14,     /* "C" */
751,    /* "CAMELLIA-128-CBC" */
```



```
757,    /* "CAMELLIA-128-CFB" */
760,    /* "CAMELLIA-128-CFB1" */
763,    /* "CAMELLIA-128-CFB8" */
754,    /* "CAMELLIA-128-ECB" */
766,    /* "CAMELLIA-128-OFB" */
752,    /* "CAMELLIA-192-CBC" */
758,    /* "CAMELLIA-192-CFB" */
761,    /* "CAMELLIA-192-CFB1" */
764,    /* "CAMELLIA-192-CFB8" */
755,    /* "CAMELLIA-192-ECB" */
767,    /* "CAMELLIA-192-OFB" */
753,    /* "CAMELLIA-256-CBC" */
759,    /* "CAMELLIA-256-CFB" */
762,    /* "CAMELLIA-256-CFB1" */
765,    /* "CAMELLIA-256-CFB8" */
756,    /* "CAMELLIA-256-ECB" */
768,    /* "CAMELLIA-256-OFB" */
108,    /* "CAST5-CBC" */
110,    /* "CAST5-CFB" */
109,    /* "CAST5-ECB" */
111,    /* "CAST5-OFB" */
894,    /* "CMAC" */
13,     /* "CN" */
141,    /* "CRLReason" */
417,    /* "CSPName" */
367,    /* "CrlID" */
391,    /* "DC" */
31,     /* "DES-CBC" */
643,    /* "DES-CDMF" */
30,     /* "DES-CFB" */
656,    /* "DES-CFB1" */
657,    /* "DES-CFB8" */
29,     /* "DES-ECB" */
32,     /* "DES-EDE" */
43,     /* "DES-EDE-CBC" */
60,     /* "DES-EDE-CFB" */
62,     /* "DES-EDE-OFB" */
33,     /* "DES-EDE3" */
44,     /* "DES-EDE3-CBC" */
61,     /* "DES-EDE3-CFB" */
658,    /* "DES-EDE3-CFB1" */
659,    /* "DES-EDE3-CFB8" */
63,     /* "DES-EDE3-OFB" */
45,     /* "DES-OFB" */
80,     /* "DESX-CBC" */
380,    /* "DOD" */
116,    /* "DSA" */
66,     /* "DSA-SHA" */
113,    /* "DSA-SHA1" */
70,     /* "DSA-SHA1-old" */
67,     /* "DSA-old" */
297,    /* "DVCS" */
99,     /* "GN" */
855,    /* "HMAC" */
780,    /* "HMAC-MD5" */
781,    /* "HMAC-SHA1" */
381,    /* "IANA" */
34,     /* "IDEA-CBC" */
35,     /* "IDEA-CFB" */
36,     /* "IDEA-ECB" */
46,     /* "IDEA-OFB" */
181,    /* "ISO" */
183,    /* "ISO-US" */
645,    /* "ITU-T" */
646,    /* "JOINT-ISO-ITU-T" */
773,    /* "KISA" */
```

```
2504 15,     /* "L" */
2505 856,    /* "LocalKeySet" */
2506  3,     /* "MD2" */
2507 257,    /* "MD4" */
2508  4,     /* "MD5" */
2509 114,    /* "MD5-SHA1" */
2510 95,     /* "MDC2" */
2511 911,    /* "MGF1" */
2512 388,    /* "Mail" */
2513 393,    /* "NULL" */
2514 404,    /* "NULL" */
2515 57,     /* "Netscape" */
2516 366,    /* "Nonce" */
2517 17,     /* "O" */
2518 178,    /* "OCSP" */
2519 180,    /* "OCSPSigning" */
2520 379,    /* "ORG" */
2521 18,     /* "OU" */
2522 749,    /* "Oakley-EC2N-3" */
2523 750,    /* "Oakley-EC2N-4" */
2524  9,     /* "PBE-MD2-DES" */
2525 168,    /* "PBE-MD2-RC2-64" */
2526 10,     /* "PBE-MD5-DES" */
2527 169,    /* "PBE-MD5-RC2-64" */
2528 147,    /* "PBE-SHA1-2DES" */
2529 146,    /* "PBE-SHA1-3DES" */
2530 170,    /* "PBE-SHA1-DES" */
2531 148,    /* "PBE-SHA1-RC2-128" */
2532 149,    /* "PBE-SHA1-RC2-40" */
2533 68,     /* "PBE-SHA1-RC2-64" */
2534 144,    /* "PBE-SHA1-RC4-128" */
2535 145,    /* "PBE-SHA1-RC4-40" */
2536 161,    /* "PBES2" */
2537 69,     /* "PBKDF2" */
2538 162,    /* "PBMAC1" */
2539 127,    /* "PKIX" */
2540 98,     /* "RC2-40-CBC" */
2541 166,    /* "RC2-64-CBC" */
2542 37,     /* "RC2-CBC" */
2543 39,     /* "RC2-CFB" */
2544 38,     /* "RC2-ECB" */
2545 40,     /* "RC2-OFB" */
2546  5,     /* "RC4" */
2547 97,     /* "RC4-40" */
2548 915,    /* "RC4-HMAC-MD5" */
2549 120,    /* "RC5-CBC" */
2550 122,    /* "RC5-CFB" */
2551 121,    /* "RC5-ECB" */
2552 123,    /* "RC5-OFB" */
2553 117,    /* "RIPEMD160" */
2554 124,    /* "RLE" */
2555 19,     /* "RSA" */
2556  7,     /* "RSA-MD2" */
2557 396,    /* "RSA-MD4" */
2558  8,     /* "RSA-MD5" */
2559 96,     /* "RSA-MDC2" */
2560 104,    /* "RSA-NP-MD5" */
2561 119,    /* "RSA-RIPEMD160" */
2562 42,     /* "RSA-SHA" */
2563 65,     /* "RSA-SHA1" */
2564 115,    /* "RSA-SHA1-2" */
2565 671,    /* "RSA-SHA224" */
2566 668,    /* "RSA-SHA256" */
2567 669,    /* "RSA-SHA384" */
2568 670,    /* "RSA-SHA512" */
2569 919,    /* "RSAES-OAEP" */
```



```
2570 912,    /* "RSASSA-PSS" */
2571 777,    /* "SEED-CBC" */
2572 779,    /* "SEED-CFB" */
2573 776,    /* "SEED-ECB" */
2574 778,    /* "SEED-OFB" */
2575 41,     /* "SHA" */
2576 64,     /* "SHA1" */
2577 675,    /* "SHA224" */
2578 672,    /* "SHA256" */
2579 673,    /* "SHA384" */
2580 674,    /* "SHA512" */
2581 188,    /* "SMIME" */
2582 167,    /* "SMIME-CAPS" */
2583 100,    /* "SN" */
2584 16,     /* "ST" */
2585 143,    /* "SXNetID" */
2586 458,    /* "UID" */
2587  0,     /* "UNDEF" */
2588 11,     /* "X500" */
2589 378,    /* "X500algorithms" */
2590 12,     /* "X509" */
2591 184,    /* "X9-57" */
2592 185,    /* "X9cm" */
2593 125,    /* "ZLIB" */
2594 478,    /* "aRecord" */
2595 289,    /* "aaControls" */
2596 287,    /* "ac-auditEntity" */
2597 397,    /* "ac-proxying" */
2598 288,    /* "ac-targeting" */
2599 368,    /* "acceptableResponses" */
2600 446,    /* "account" */
2601 363,    /* "ad_timestamping" */
2602 376,    /* "algorithm" */
2603 405,    /* "ansi-X9-62" */
2604 910,    /* "anyExtendedKeyUsage" */
2605 746,    /* "anyPolicy" */
2606 370,    /* "archiveCutoff" */
2607 484,    /* "associatedDomain" */
2608 485,    /* "associatedName" */
2609 501,    /* "audio" */
2610 177,    /* "authorityInfoAccess" */
2611 90,     /* "authorityKeyIdentifier" */
2612 882,    /* "authorityRevocationList" */
2613 87,     /* "basicConstraints" */
2614 365,    /* "basicOCSPResponse" */
2615 285,    /* "biometricInfo" */
2616 494,    /* "buildingName" */
2617 860,    /* "businessCategory" */
2618 691,    /* "c2onb191v4" */
2619 692,    /* "c2onb191v5" */
2620 697,    /* "c2onb239v4" */
2621 698,    /* "c2onb239v5" */
2622 684,    /* "c2pnb163v1" */
2623 685,    /* "c2pnb163v2" */
2624 686,    /* "c2pnb163v3" */
2625 687,    /* "c2pnb176v1" */
2626 693,    /* "c2pnb208w1" */
2627 699,    /* "c2pnb272w1" */
2628 700,    /* "c2pnb304w1" */
2629 702,    /* "c2pnb368w1" */
2630 688,    /* "c2tnb191v1" */
2631 689,    /* "c2tnb191v2" */
2632 690,    /* "c2tnb191v3" */
2633 694,    /* "c2tnb239v1" */
2634 695,    /* "c2tnb239v2" */
2635 696,    /* "c2tnb239v3" */
```

```
2636 701,    /* "c2tnb359v1" */
2637 703,    /* "c2tnb431r1" */
2638 881,    /* "cACertificate" */
2639 483,    /* "cNAMERecord" */
2640 179,    /* "caIssuers" */
2641 785,    /* "caRepository" */
2642 443,    /* "caseIgnoreIA5StringSyntax" */
2643 152,    /* "certBag" */
2644 677,    /* "certicom-arc" */
2645 771,    /* "certificateIssuer" */
2646 89,     /* "certificatePolicies" */
2647 883,    /* "certificateRevocationList" */
2648 54,     /* "challengePassword" */
2649 407,    /* "characteristic-two-field" */
2650 395,    /* "clearance" */
2651 130,    /* "clientAuth" */
2652 131,    /* "codeSigning" */
2653 50,     /* "contentType" */
2654 53,     /* "countersignature" */
2655 153,    /* "crlBag" */
2656 103,    /* "crlDistributionPoints" */
2657 88,     /* "crlNumber" */
2658 884,    /* "crossCertificatePair" */
2659 806,    /* "cryptocom" */
2660 805,    /* "cryptopro" */
2661 500,    /* "dITRedirect" */
2662 451,    /* "dNSDomain" */
2663 495,    /* "dSAQuality" */
2664 434,    /* "data" */
2665 390,    /* "dcobject" */
2666 140,    /* "deltaCRL" */
2667 891,    /* "deltaRevocationList" */
2668 107,    /* "description" */
2669 871,    /* "destinationIndicator" */
2670 28,     /* "dhKeyAgreement" */
2671 382,    /* "directory" */
2672 887,    /* "distinguishedName" */
2673 892,    /* "dmdName" */
2674 174,    /* "dnQualifier" */
2675 447,    /* "document" */
2676 471,    /* "documentAuthor" */
2677 468,    /* "documentIdentifier" */
2678 472,    /* "documentLocation" */
2679 502,    /* "documentPublisher" */
2680 449,    /* "documentSeries" */
2681 469,    /* "documentTitle" */
2682 470,    /* "documentVersion" */
2683 392,    /* "domain" */
2684 452,    /* "domainRelatedObject" */
2685 802,    /* "dsa_with_SHA224" */
2686 803,    /* "dsa_with_SHA256" */
2687 791,    /* "ecdsa-with-Recommended" */
2688 416,    /* "ecdsa-with-SHA1" */
2689 793,    /* "ecdsa-with-SHA224" */
2690 794,    /* "ecdsa-with-SHA256" */
2691 795,    /* "ecdsa-with-SHA384" */
2692 796,    /* "ecdsa-with-SHA512" */
2693 792,    /* "ecdsa-with-Specified" */
2694 48,     /* "emailAddress" */
2695 132,    /* "emailProtection" */
2696 885,    /* "enhancedSearchGuide" */
2697 389,    /* "enterprises" */
2698 384,    /* "experimental" */
2699 172,    /* "extReq" */
2700 56,     /* "extendedCertificateAttributes" */
2701 126,    /* "extendedKeyUsage" */
```



```
2702 372,    /* "extendedStatus" */
2703 867,    /* "facsimileTelephoneNumber" */
2704 462,    /* "favouriteDrink" */
2705 857,    /* "freshestCRL" */
2706 453,    /* "friendlyCountry" */
2707 490,    /* "friendlyCountryName" */
2708 156,    /* "friendlyName" */
2709 509,    /* "generationQualifier" */
2710 815,    /* "gost-mac" */
2711 811,    /* "gost2001" */
2712 851,    /* "gost2001cc" */
2713 813,    /* "gost89" */
2714 814,    /* "gost89-cnt" */
2715 812,    /* "gost94" */
2716 850,    /* "gost94cc" */
2717 797,    /* "hmacWithMD5" */
2718 163,    /* "hmacWithSHA1" */
2719 798,    /* "hmacWithSHA224" */
2720 799,    /* "hmacWithSHA256" */
2721 800,    /* "hmacWithSHA384" */
2722 801,    /* "hmacWithSHA512" */
2723 432,    /* "holdInstructionCallIssuer" */
2724 430,    /* "holdInstructionCode" */
2725 431,    /* "holdInstructionNone" */
2726 433,    /* "holdInstructionReject" */
2727 486,    /* "homePostalAddress" */
2728 473,    /* "homeTelephoneNumber" */
2729 466,    /* "host" */
2730 889,    /* "houseIdentifier" */
2731 442,    /* "iA5StringSyntax" */
2732 783,    /* "id-DHBasedMac" */
2733 824,    /* "id-Gost28147-89-CryptoPro-A-ParamSet" */
2734 825,    /* "id-Gost28147-89-CryptoPro-B-ParamSet" */
2735 826,    /* "id-Gost28147-89-CryptoPro-C-ParamSet" */
2736 827,    /* "id-Gost28147-89-CryptoPro-D-ParamSet" */
2737 819,    /* "id-Gost28147-89-CryptoPro-KeyMeshing" */
2738 829,    /* "id-Gost28147-89-CryptoPro-Oscar-1-0-ParamSet" */
2739 828,    /* "id-Gost28147-89-CryptoPro-Oscar-1-1-ParamSet" */
2740 830,    /* "id-Gost28147-89-CryptoPro-RIC-1-ParamSet" */
2741 820,    /* "id-Gost28147-89-None-KeyMeshing" */
2742 823,    /* "id-Gost28147-89-TestParamSet" */
2743 849,    /* "id-Gost28147-89-cc" */
2744 840,    /* "id-GostR3410-2001-CryptoPro-A-ParamSet" */
2745 841,    /* "id-GostR3410-2001-CryptoPro-B-ParamSet" */
2746 842,    /* "id-GostR3410-2001-CryptoPro-C-ParamSet" */
2747 843,    /* "id-GostR3410-2001-CryptoPro-XchA-ParamSet" */
2748 844,    /* "id-GostR3410-2001-CryptoPro-XchB-ParamSet" */
2749 854,    /* "id-GostR3410-2001-ParamSet-cc" */
2750 839,    /* "id-GostR3410-2001-TestParamSet" */
2751 817,    /* "id-GostR3410-2001DH" */
2752 832,    /* "id-GostR3410-94-CryptoPro-A-ParamSet" */
2753 833,    /* "id-GostR3410-94-CryptoPro-B-ParamSet" */
2754 834,    /* "id-GostR3410-94-CryptoPro-C-ParamSet" */
2755 835,    /* "id-GostR3410-94-CryptoPro-D-ParamSet" */
2756 836,    /* "id-GostR3410-94-CryptoPro-XchA-ParamSet" */
2757 837,    /* "id-GostR3410-94-CryptoPro-XchB-ParamSet" */
2758 838,    /* "id-GostR3410-94-CryptoPro-XchC-ParamSet" */
2759 831,    /* "id-GostR3410-94-TestParamSet" */
2760 845,    /* "id-GostR3410-94-a" */
2761 846,    /* "id-GostR3410-94-aBis" */
2762 847,    /* "id-GostR3410-94-b" */
2763 848,    /* "id-GostR3410-94-bBis" */
2764 818,    /* "id-GostR3410-94DH" */
2765 822,    /* "id-GostR3411-94-CryptoProParamSet" */
2766 821,    /* "id-GostR3411-94-TestParamSet" */
2767 807,    /* "id-GostR3411-94-with-GostR3410-2001" */
```

```
2768 853,    /* "id-GostR3411-94-with-GostR3410-2001-cc" */
2769 808,    /* "id-GostR3411-94-with-GostR3410-94" */
2770 852,    /* "id-GostR3411-94-with-GostR3410-94-cc" */
2771 810,    /* "id-HMACGostR3411-94" */
2772 782,    /* "id-PasswordBasedMAC" */
2773 266,    /* "id-aca" */
2774 355,    /* "id-aca-accessIdentity" */
2775 354,    /* "id-aca-authenticationInfo" */
2776 356,    /* "id-aca-chargingIdentity" */
2777 399,    /* "id-aca-encAttrs" */
2778 357,    /* "id-aca-group" */
2779 358,    /* "id-aca-role" */
2780 176,    /* "id-ad" */
2781 896,    /* "id-aes128-CCM" */
2782 895,    /* "id-aes128-GCM" */
2783 788,    /* "id-aes128-wrap" */
2784 897,    /* "id-aes128-wrap-pad" */
2785 899,    /* "id-aes192-CCM" */
2786 898,    /* "id-aes192-GCM" */
2787 789,    /* "id-aes192-wrap" */
2788 900,    /* "id-aes192-wrap-pad" */
2789 902,    /* "id-aes256-CCM" */
2790 901,    /* "id-aes256-GCM" */
2791 790,    /* "id-aes256-wrap" */
2792 903,    /* "id-aes256-wrap-pad" */
2793 262,    /* "id-alg" */
2794 893,    /* "id-alg-PWRI-KEK" */
2795 323,    /* "id-alg-des40" */
2796 326,    /* "id-alg-dh-pop" */
2797 325,    /* "id-alg-dh-sig-hmac-sha1" */
2798 324,    /* "id-alg-noSignature" */
2799 907,    /* "id-camellia128-wrap" */
2800 908,    /* "id-camellia192-wrap" */
2801 909,    /* "id-camellia256-wrap" */
2802 268,    /* "id-cct" */
2803 361,    /* "id-cct-PKIData" */
2804 362,    /* "id-cct-PKIResponse" */
2805 360,    /* "id-cct-crs" */
2806 81,     /* "id-ce" */
2807 680,    /* "id-characteristic-two-basis" */
2808 263,    /* "id-cmc" */
2809 334,    /* "id-cmc-addExtensions" */
2810 346,    /* "id-cmc-confirmCertAcceptance" */
2811 330,    /* "id-cmc-dataReturn" */
2812 336,    /* "id-cmc-decryptedPOP" */
2813 335,    /* "id-cmc-encryptedPOP" */
2814 339,    /* "id-cmc-getCRL" */
2815 338,    /* "id-cmc-getCert" */
2816 328,    /* "id-cmc-identification" */
2817 329,    /* "id-cmc-identityProof" */
2818 337,    /* "id-cmc-lraPOPWitness" */
2819 344,    /* "id-cmc-popLinkRandom" */
2820 345,    /* "id-cmc-popLinkWitness" */
2821 343,    /* "id-cmc-queryPending" */
2822 333,    /* "id-cmc-recipientNonce" */
2823 341,    /* "id-cmc-regInfo" */
2824 342,    /* "id-cmc-responseInfo" */
2825 340,    /* "id-cmc-revokeRequest" */
2826 332,    /* "id-cmc-senderNonce" */
2827 327,    /* "id-cmc-statusInfo" */
2828 331,    /* "id-cmc-transactionId" */
2829 787,    /* "id-ct-asciiTextWithCRLF" */
2830 408,    /* "id-ecPublicKey" */
2831 508,    /* "id-hex-multipart-message" */
2832 507,    /* "id-hex-partial-message" */
2833 260,    /* "id-it" */
```



```
2834 302,    /* "id-it-caKeyUpdateInfo" */
2835 298,    /* "id-it-caProtEncCert" */
2836 311,    /* "id-it-confirmWaitTime" */
2837 303,    /* "id-it-currentCRL" */
2838 300,    /* "id-it-encKeyPairTypes" */
2839 310,    /* "id-it-implicitConfirm" */
2840 308,    /* "id-it-keyPairParamRep" */
2841 307,    /* "id-it-keyPairParamReq" */
2842 312,    /* "id-it-origPKIMessage" */
2843 301,    /* "id-it-preferredSymmAlg" */
2844 309,    /* "id-it-revPassphrase" */
2845 299,    /* "id-it-signKeyPairTypes" */
2846 305,    /* "id-it-subscriptionRequest" */
2847 306,    /* "id-it-subscriptionResponse" */
2848 784,    /* "id-it-suppLangTags" */
2849 304,    /* "id-it-unsupportedOIDs" */
2850 128,    /* "id-kp" */
2851 280,    /* "id-mod-attribute-cert" */
2852 274,    /* "id-mod-cmc" */
2853 277,    /* "id-mod-cmp" */
2854 284,    /* "id-mod-cmp2000" */
2855 273,    /* "id-mod-crmf" */
2856 283,    /* "id-mod-dvcs" */
2857 275,    /* "id-mod-kea-profile-88" */
2858 276,    /* "id-mod-kea-profile-93" */
2859 282,    /* "id-mod-ocsp" */
2860 278,    /* "id-mod-qualified-cert-88" */
2861 279,    /* "id-mod-qualified-cert-93" */
2862 281,    /* "id-mod-timestamp-protocol" */
2863 264,    /* "id-on" */
2864 858,    /* "id-on-permanentIdentifier" */
2865 347,    /* "id-on-personalData" */
2866 265,    /* "id-pda" */
2867 352,    /* "id-pda-countryOfCitizenship" */
2868 353,    /* "id-pda-countryOfResidence" */
2869 348,    /* "id-pda-dateOfBirth" */
2870 351,    /* "id-pda-gender" */
2871 349,    /* "id-pda-placeOfBirth" */
2872 175,    /* "id-pe" */
2873 261,    /* "id-pkip" */
2874 258,    /* "id-pkix-mod" */
2875 269,    /* "id-pkix1-explicit-88" */
2876 271,    /* "id-pkix1-explicit-93" */
2877 270,    /* "id-pkix1-implicit-88" */
2878 272,    /* "id-pkix1-implicit-93" */
2879 662,    /* "id-ppl" */
2880 664,    /* "id-ppl-anyLanguage" */
2881 667,    /* "id-ppl-independent" */
2882 665,    /* "id-ppl-inheritAll" */
2883 267,    /* "id-qcs" */
2884 359,    /* "id-qcs-pkixQCSyntax-v1" */
2885 259,    /* "id-qt" */
2886 164,    /* "id-qt-cps" */
2887 165,    /* "id-qt-unotice" */
2888 313,    /* "id-regCtrl" */
2889 316,    /* "id-regCtrl-authenticator" */
2890 319,    /* "id-regCtrl-oldCertID" */
2891 318,    /* "id-regCtrl-pkiArchiveOptions" */
2892 317,    /* "id-regCtrl-pkiPublicationInfo" */
2893 320,    /* "id-regCtrl-protocolEncrKey" */
2894 315,    /* "id-regCtrl-regToken" */
2895 314,    /* "id-regInfo" */
2896 322,    /* "id-regInfo-certReq" */
2897 321,    /* "id-regInfo-utf8Pairs" */
2898 512,    /* "id-set" */
2899 191,    /* "id-smime-aa" */
```

```
2900 215,    /* "id-smime-aa-contentHint" */
2901 218,    /* "id-smime-aa-contentIdentifier" */
2902 221,    /* "id-smime-aa-contentReference" */
2903 240,    /* "id-smime-aa-dvcs-dvc" */
2904 217,    /* "id-smime-aa-encapContentType" */
2905 222,    /* "id-smime-aa-encrypKeyPref" */
2906 220,    /* "id-smime-aa-equivalentLabels" */
2907 232,    /* "id-smime-aa-ets-CertificateRefs" */
2908 233,    /* "id-smime-aa-ets-RevocationRefs" */
2909 238,    /* "id-smime-aa-ets-archiveTimeStamp" */
2910 237,    /* "id-smime-aa-ets-certCRLTimestamp" */
2911 234,    /* "id-smime-aa-ets-certValues" */
2912 227,    /* "id-smime-aa-ets-commitmentType" */
2913 231,    /* "id-smime-aa-ets-contentTimestamp" */
2914 236,    /* "id-smime-aa-ets-escTimeStamp" */
2915 230,    /* "id-smime-aa-ets-otherSigCert" */
2916 235,    /* "id-smime-aa-ets-revocationValues" */
2917 226,    /* "id-smime-aa-ets-sigPolicyId" */
2918 229,    /* "id-smime-aa-ets-signerAttr" */
2919 228,    /* "id-smime-aa-ets-signerLocation" */
2920 219,    /* "id-smime-aa-macValue" */
2921 214,    /* "id-smime-aa-mlExpandHistory" */
2922 216,    /* "id-smime-aa-msgSigDigest" */
2923 212,    /* "id-smime-aa-receiptRequest" */
2924 213,    /* "id-smime-aa-securityLabel" */
2925 239,    /* "id-smime-aa-signatureType" */
2926 223,    /* "id-smime-aa-signingCertificate" */
2927 224,    /* "id-smime-aa-smimeEncryptCerts" */
2928 225,    /* "id-smime-aa-timeStampToken" */
2929 192,    /* "id-smime-alg" */
2930 243,    /* "id-smime-alg-3DESwrap" */
2931 246,    /* "id-smime-alg-CMS3DESwrap" */
2932 247,    /* "id-smime-alg-CMSRC2wrap" */
2933 245,    /* "id-smime-alg-ESDH" */
2934 241,    /* "id-smime-alg-ESDHwith3DES" */
2935 242,    /* "id-smime-alg-ESDHwithRC2" */
2936 244,    /* "id-smime-alg-RC2wrap" */
2937 193,    /* "id-smime-cd" */
2938 248,    /* "id-smime-cd-ldap" */
2939 190,    /* "id-smime-ct" */
2940 210,    /* "id-smime-ct-DVCSRequestData" */
2941 211,    /* "id-smime-ct-DVCSResponseData" */
2942 208,    /* "id-smime-ct-TDTInfo" */
2943 207,    /* "id-smime-ct-TSTInfo" */
2944 205,    /* "id-smime-ct-authData" */
2945 786,    /* "id-smime-ct-compressedData" */
2946 209,    /* "id-smime-ct-contentInfo" */
2947 206,    /* "id-smime-ct-publishCert" */
2948 204,    /* "id-smime-ct-receipt" */
2949 195,    /* "id-smime-cti" */
2950 255,    /* "id-smime-cti-ets-proofOfApproval" */
2951 256,    /* "id-smime-cti-ets-proofOfCreation" */
2952 253,    /* "id-smime-cti-ets-proofOfDelivery" */
2953 251,    /* "id-smime-cti-ets-proofOfOrigin" */
2954 252,    /* "id-smime-cti-ets-proofOfReceipt" */
2955 254,    /* "id-smime-cti-ets-proofOfSender" */
2956 189,    /* "id-smime-mod" */
2957 196,    /* "id-smime-mod-cms" */
2958 197,    /* "id-smime-mod-ess" */
2959 202,    /* "id-smime-mod-ets-eSigPolicy-88" */
2960 203,    /* "id-smime-mod-ets-eSigPolicy-97" */
2961 200,    /* "id-smime-mod-ets-eSignature-88" */
2962 201,    /* "id-smime-mod-ets-eSignature-97" */
2963 199,    /* "id-smime-mod-msg-v3" */
2964 198,    /* "id-smime-mod-oid" */
2965 194,    /* "id-smime-spq" */
```



```
2966 250,    /* "id-smime-spq-ets-sqt-unotice" */
2967 249,    /* "id-smime-spq-ets-sqt-uri" */
2968 676,    /* "identified-organization" */
2969 461,    /* "info" */
2970 748,    /* "inhibitAnyPolicy" */
2971 101,    /* "initials" */
2972 647,    /* "international-organizations" */
2973 869,    /* "internationaliSDNNumber" */
2974 142,    /* "invalidityDate" */
2975 294,    /* "ipsecEndSystem" */
2976 295,    /* "ipsecTunnel" */
2977 296,    /* "ipsecUser" */
2978 86,     /* "issuerAltName" */
2979 770,    /* "issuingDistributionPoint" */
2980 492,    /* "janetMailbox" */
2981 150,    /* "keyBag" */
2982 83,     /* "keyUsage" */
2983 477,    /* "lastModifiedBy" */
2984 476,    /* "lastModifiedTime" */
2985 157,    /* "localKeyID" */
2986 480,    /* "mXRecord" */
2987 460,    /* "mail" */
2988 493,    /* "mailPreferenceOption" */
2989 467,    /* "manager" */
2990 809,    /* "md_gost94" */
2991 875,    /* "member" */
2992 182,    /* "member-body" */
2993 51,     /* "messageDigest" */
2994 383,    /* "mgmt" */
2995 504,    /* "mime-mhs" */
2996 506,    /* "mime-mhs-bodies" */
2997 505,    /* "mime-mhs-headings" */
2998 488,    /* "mobileTelephoneNumber" */
2999 136,    /* "msCTLSign" */
3000 135,    /* "msCodeCom" */
3001 134,    /* "msCodeInd" */
3002 138,    /* "msEFS" */
3003 171,    /* "msExtReq" */
3004 137,    /* "msSGC" */
3005 648,    /* "msSmartcardLogin" */
3006 649,    /* "msUPN" */
3007 481,    /* "nSRecord" */
3008 173,    /* "name" */
3009 666,    /* "nameConstraints" */
3010 369,    /* "noCheck" */
3011 403,    /* "noRevAvail" */
3012 72,     /* "nsBaseUrl" */
3013 76,     /* "nsCaPolicyUrl" */
3014 74,     /* "nsCaRevocationUrl" */
3015 58,     /* "nsCertExt" */
3016 79,     /* "nsCertSequence" */
3017 71,     /* "nsCertType" */
3018 78,     /* "nsComment" */
3019 59,     /* "nsDataType" */
3020 75,     /* "nsRenewalUrl" */
3021 73,     /* "nsRevocationUrl" */
3022 139,    /* "nsSGC" */
3023 77,     /* "nsSslServerName" */
3024 681,    /* "onBasis" */
3025 491,    /* "organizationalStatus" */
3026 475,    /* "otherMailbox" */
3027 876,    /* "owner" */
3028 489,    /* "pagerTelephoneNumber" */
3029 374,    /* "path" */
3030 112,    /* "pbeWithMD5AndCast5CBC" */
3031 499,    /* "personalSignature" */
```

```
3032 487,    /* "personalTitle" */
3033 464,    /* "photo" */
3034 863,    /* "physicalDeliveryOfficeName" */
3035 437,    /* "pilot" */
3036 439,    /* "pilotAttributeSyntax" */
3037 438,    /* "pilotAttributeType" */
3038 479,    /* "pilotAttributeType27" */
3039 456,    /* "pilotDSA" */
3040 441,    /* "pilotGroups" */
3041 444,    /* "pilotObject" */
3042 440,    /* "pilotObjectClass" */
3043 455,    /* "pilotOrganization" */
3044 445,    /* "pilotPerson" */
3045  2,    /* "pkcs" */
3046 186,    /* "pkcs1" */
3047 27,    /* "pkcs3" */
3048 187,    /* "pkcs5" */
3049 20,    /* "pkcs7" */
3050 21,    /* "pkcs7-data" */
3051 25,    /* "pkcs7-digestData" */
3052 26,    /* "pkcs7-encryptedData" */
3053 23,    /* "pkcs7-envelopedData" */
3054 24,    /* "pkcs7-signedAndEnvelopedData" */
3055 22,    /* "pkcs7-signedData" */
3056 151,    /* "pkcs8ShroudedKeyBag" */
3057 47,    /* "pkcs9" */
3058 401,    /* "policyConstraints" */
3059 747,    /* "policyMappings" */
3060 862,    /* "postOfficeBox" */
3061 861,    /* "postalAddress" */
3062 661,    /* "postalCode" */
3063 683,    /* "ppBasis" */
3064 872,    /* "preferredDeliveryMethod" */
3065 873,    /* "presentationAddress" */
3066 816,    /* "prf-gostr3411-94" */
3067 406,    /* "prime-field" */
3068 409,    /* "prime192v1" */
3069 410,    /* "prime192v2" */
3070 411,    /* "prime192v3" */
3071 412,    /* "prime239v1" */
3072 413,    /* "prime239v2" */
3073 414,    /* "prime239v3" */
3074 415,    /* "prime256v1" */
3075 385,    /* "private" */
3076 84,    /* "privateKeyUsagePeriod" */
3077 886,    /* "protocolInformation" */
3078 663,    /* "proxyCertInfo" */
3079 510,    /* "pseudonym" */
3080 435,    /* "pss" */
3081 286,    /* "qcStatements" */
3082 457,    /* "qualityLabelledData" */
3083 450,    /* "rFC822localPart" */
3084 870,    /* "registeredAddress" */
3085 400,    /* "role" */
3086 877,    /* "roleOccupant" */
3087 448,    /* "room" */
3088 463,    /* "roomNumber" */
3089  6,    /* "rsaEncryption" */
3090 644,    /* "rsaOAEPEncryptionSET" */
3091 377,    /* "rsaSignature" */
3092  1,    /* "rsadsi" */
3093 482,    /* "sOARecord" */
3094 155,    /* "safeContentsBag" */
3095 291,    /* "sbgp-autonomousSysNum" */
3096 290,    /* "sbgp-ipAddrBlock" */
3097 292,    /* "sbgp-routerIdentifier" */
```



```
3098 159,    /* "sdsiCertificate" */
3099 859,    /* "searchGuide" */
3100 704,    /* "secp112r1" */
3101 705,    /* "secp112r2" */
3102 706,    /* "secp128r1" */
3103 707,    /* "secp128r2" */
3104 708,    /* "secp160k1" */
3105 709,    /* "secp160r1" */
3106 710,    /* "secp160r2" */
3107 711,    /* "secp192k1" */
3108 712,    /* "secp224k1" */
3109 713,    /* "secp224r1" */
3110 714,    /* "secp256k1" */
3111 715,    /* "secp384r1" */
3112 716,    /* "secp521r1" */
3113 154,    /* "secretBag" */
3114 474,    /* "secretary" */
3115 717,    /* "sect113r1" */
3116 718,    /* "sect113r2" */
3117 719,    /* "sect131r1" */
3118 720,    /* "sect131r2" */
3119 721,    /* "sect163k1" */
3120 722,    /* "sect163r1" */
3121 723,    /* "sect163r2" */
3122 724,    /* "sect193r1" */
3123 725,    /* "sect193r2" */
3124 726,    /* "sect233k1" */
3125 727,    /* "sect233r1" */
3126 728,    /* "sect239k1" */
3127 729,    /* "sect283k1" */
3128 730,    /* "sect283r1" */
3129 731,    /* "sect409k1" */
3130 732,    /* "sect409r1" */
3131 733,    /* "sect571k1" */
3132 734,    /* "sect571r1" */
3133 386,    /* "security" */
3134 878,    /* "seeAlso" */
3135 394,    /* "selected-attribute-types" */
3136 105,    /* "serialNumber" */
3137 129,    /* "serverAuth" */
3138 371,    /* "serviceLocator" */
3139 625,    /* "set-addPolicy" */
3140 515,    /* "set-attr" */
3141 518,    /* "set-brand" */
3142 638,    /* "set-brand-AmericanExpress" */
3143 637,    /* "set-brand-Diners" */
3144 636,    /* "set-brand-IATA-ATA" */
3145 639,    /* "set-brand-JCB" */
3146 641,    /* "set-brand-MasterCard" */
3147 642,    /* "set-brand-Novus" */
3148 640,    /* "set-brand-Visa" */
3149 517,    /* "set-certExt" */
3150 513,    /* "set-ctype" */
3151 514,    /* "set-msgExt" */
3152 516,    /* "set-policy" */
3153 607,    /* "set-policy-root" */
3154 624,    /* "set-rootKeyThumb" */
3155 620,    /* "setAttr-Cert" */
3156 631,    /* "setAttr-GenCryptgrm" */
3157 623,    /* "setAttr-IssCap" */
3158 628,    /* "setAttr-IssCap-CVM" */
3159 630,    /* "setAttr-IssCap-Sig" */
3160 629,    /* "setAttr-IssCap-T2" */
3161 621,    /* "setAttr-PGWYcap" */
3162 635,    /* "setAttr-SecDevSig" */
3163 632,    /* "setAttr-T2Enc" */
```

```
3164 633,    /* "setAttr-T2cleartxt" */
3165 634,    /* "setAttr-TokICCsig" */
3166 627,    /* "setAttr-Token-B0Prime" */
3167 626,    /* "setAttr-Token-EMV" */
3168 622,    /* "setAttr-TokenType" */
3169 619,    /* "setCext-IssuerCapabilities" */
3170 615,    /* "setCext-PGWYcapabilities" */
3171 616,    /* "setCext-TokenIdentifier" */
3172 618,    /* "setCext-TokenType" */
3173 617,    /* "setCext-Track2Data" */
3174 611,    /* "setCext-cCertRequired" */
3175 609,    /* "setCext-certType" */
3176 608,    /* "setCext-hashedRoot" */
3177 610,    /* "setCext-merchData" */
3178 613,    /* "setCext-setExt" */
3179 614,    /* "setCext-setQualf" */
3180 612,    /* "setCext-tunneling" */
3181 540,    /* "setct-AcqCardCodeMsg" */
3182 576,    /* "setct-AcqCardCodeMsgTBE" */
3183 570,    /* "setct-AuthReqTBE" */
3184 534,    /* "setct-AuthReqTBS" */
3185 527,    /* "setct-AuthResBaggage" */
3186 571,    /* "setct-AuthResTBE" */
3187 572,    /* "setct-AuthResTBEX" */
3188 535,    /* "setct-AuthResTBS" */
3189 536,    /* "setct-AuthResTBSX" */
3190 528,    /* "setct-AuthRevReqBaggage" */
3191 577,    /* "setct-AuthRevReqTBE" */
3192 541,    /* "setct-AuthRevReqTBS" */
3193 529,    /* "setct-AuthRevResBaggage" */
3194 542,    /* "setct-AuthRevResData" */
3195 578,    /* "setct-AuthRevResTBE" */
3196 579,    /* "setct-AuthRevResTBEB" */
3197 543,    /* "setct-AuthRevResTBS" */
3198 573,    /* "setct-AuthTokenTBE" */
3199 537,    /* "setct-AuthTokenTBS" */
3200 600,    /* "setct-BCIDistributionTBS" */
3201 558,    /* "setct-BatchAdminReqData" */
3202 592,    /* "setct-BatchAdminReqTBE" */
3203 559,    /* "setct-BatchAdminResData" */
3204 593,    /* "setct-BatchAdminResTBE" */
3205 599,    /* "setct-CRLNotificationResTBS" */
3206 598,    /* "setct-CRLNotificationTBS" */
3207 580,    /* "setct-CapReqTBE" */
3208 581,    /* "setct-CapReqTBEX" */
3209 544,    /* "setct-CapReqTBS" */
3210 545,    /* "setct-CapReqTBSX" */
3211 546,    /* "setct-CapResData" */
3212 582,    /* "setct-CapResTBE" */
3213 583,    /* "setct-CapRevReqTBE" */
3214 584,    /* "setct-CapRevReqTBEX" */
3215 547,    /* "setct-CapRevReqTBS" */
3216 548,    /* "setct-CapRevReqTBSX" */
3217 549,    /* "setct-CapRevResData" */
3218 585,    /* "setct-CapRevResTBE" */
3219 538,    /* "setct-CapTokenData" */
3220 530,    /* "setct-CapTokenSeq" */
3221 574,    /* "setct-CapTokenTBE" */
3222 575,    /* "setct-CapTokenTBEX" */
3223 539,    /* "setct-CapTokenTBS" */
3224 560,    /* "setct-CardCInitResTBS" */
3225 566,    /* "setct-CertInqReqTBS" */
3226 563,    /* "setct-CertReqData" */
3227 595,    /* "setct-CertReqTBE" */
3228 596,    /* "setct-CertReqTBEX" */
3229 564,    /* "setct-CertReqTBS" */
```



```
3230 565,    /* "setct-CertResData" */
3231 597,    /* "setct-CertResTBE" */
3232 586,    /* "setct-CredReqTBE" */
3233 587,    /* "setct-CredReqTBEX" */
3234 550,    /* "setct-CredReqTBS" */
3235 551,    /* "setct-CredReqTBSX" */
3236 552,    /* "setct-CredResData" */
3237 588,    /* "setct-CredResTBE" */
3238 589,    /* "setct-CredRevReqTBE" */
3239 590,    /* "setct-CredRevReqTBEX" */
3240 553,    /* "setct-CredRevReqTBS" */
3241 554,    /* "setct-CredRevReqTBSX" */
3242 555,    /* "setct-CredRevResData" */
3243 591,    /* "setct-CredRevResTBE" */
3244 567,    /* "setct-ErrorTBS" */
3245 526,    /* "setct-HODInput" */
3246 561,    /* "setct-MeAqCInitResTBS" */
3247 522,    /* "setct-OIData" */
3248 519,    /* "setct-PANData" */
3249 521,    /* "setct-PANOnly" */
3250 520,    /* "setct-PANToken" */
3251 556,    /* "setct-PCertReqData" */
3252 557,    /* "setct-PCertResTBS" */
3253 523,    /* "setct-PI" */
3254 532,    /* "setct-PI-TBS" */
3255 524,    /* "setct-PIData" */
3256 525,    /* "setct-PIDataUnsigned" */
3257 568,    /* "setct-PIDualSignedTBE" */
3258 569,    /* "setct-PIUnsignedTBE" */
3259 531,    /* "setct-PInitResData" */
3260 533,    /* "setct-PResData" */
3261 594,    /* "setct-RegFormReqTBE" */
3262 562,    /* "setct-RegFormResTBS" */
3263 606,    /* "setext-cv" */
3264 601,    /* "setext-genCrypt" */
3265 602,    /* "setext-miAuth" */
3266 604,    /* "setext-pinAny" */
3267 603,    /* "setext-pinSecure" */
3268 605,    /* "setext-track2" */
3269 52,     /* "signingTime" */
3270 454,    /* "simpleSecurityObject" */
3271 496,    /* "singleLevelQuality" */
3272 387,    /* "snmpv2" */
3273 660,    /* "street" */
3274 85,     /* "subjectAltName" */
3275 769,    /* "subjectDirectoryAttributes" */
3276 398,    /* "subjectInfoAccess" */
3277 82,     /* "subjectKeyIdentifier" */
3278 498,    /* "subtreeMaximumQuality" */
3279 497,    /* "subtreeMinimumQuality" */
3280 890,    /* "supportedAlgorithms" */
3281 874,    /* "supportedApplicationContext" */
3282 402,    /* "targetInformation" */
3283 864,    /* "telephoneNumber" */
3284 866,    /* "teletexTerminalIdentifier" */
3285 865,    /* "telexNumber" */
3286 459,    /* "textEncodedORAddress" */
3287 293,    /* "textNotice" */
3288 133,    /* "timeStamping" */
3289 106,    /* "title" */
3290 682,    /* "tpBasis" */
3291 375,    /* "trustRoot" */
3292 436,    /* "ucl" */
3293 888,    /* "uniqueMember" */
3294 55,     /* "unstructuredAddress" */
3295 49,     /* "unstructuredName" */
```

```
3296 880,    /* "userCertificate" */
3297 465,    /* "userClass" */
3298 879,    /* "userPassword" */
3299 373,    /* "valid" */
3300 678,    /* "wap" */
3301 679,    /* "wap-wsg" */
3302 735,    /* "wap-wsg-idm-ecid-wtls1" */
3303 743,    /* "wap-wsg-idm-ecid-wtls10" */
3304 744,    /* "wap-wsg-idm-ecid-wtls11" */
3305 745,    /* "wap-wsg-idm-ecid-wtls12" */
3306 736,    /* "wap-wsg-idm-ecid-wtls3" */
3307 737,    /* "wap-wsg-idm-ecid-wtls4" */
3308 738,    /* "wap-wsg-idm-ecid-wtls5" */
3309 739,    /* "wap-wsg-idm-ecid-wtls6" */
3310 740,    /* "wap-wsg-idm-ecid-wtls7" */
3311 741,    /* "wap-wsg-idm-ecid-wtls8" */
3312 742,    /* "wap-wsg-idm-ecid-wtls9" */
3313 804,    /* "whirlpool" */
3314 868,    /* "x121Address" */
3315 503,    /* "x500UniqueIdentifier" */
3316 158,    /* "x509Certificate" */
3317 160,    /* "x509Crl" */
3318 };

3320 static const unsigned int ln_objs[NUM_LN]={
3321 363,    /* "AD Time Stamping" */
3322 405,    /* "ANSI X9.62" */
3323 368,    /* "Acceptable OCSP Responses" */
3324 910,    /* "Any Extended Key Usage" */
3325 664,    /* "Any language" */
3326 177,    /* "Authority Information Access" */
3327 365,    /* "Basic OCSP Response" */
3328 285,    /* "Biometric Info" */
3329 179,    /* "CA Issuers" */
3330 785,    /* "CA Repository" */
3331 131,    /* "Code Signing" */
3332 783,    /* "Diffie-Hellman based MAC" */
3333 382,    /* "Directory" */
3334 392,    /* "Domain" */
3335 132,    /* "E-mail Protection" */
3336 389,    /* "Enterprises" */
3337 384,    /* "Experimental" */
3338 372,    /* "Extended OCSP Status" */
3339 172,    /* "Extension Request" */
3340 813,    /* "GOST 28147-89" */
3341 849,    /* "GOST 28147-89 Cryptocom ParamSet" */
3342 815,    /* "GOST 28147-89 MAC" */
3343 851,    /* "GOST 34.10-2001 Cryptocom" */
3344 850,    /* "GOST 34.10-94 Cryptocom" */
3345 811,    /* "GOST R 34.10-2001" */
3346 817,    /* "GOST R 34.10-2001 DH" */
3347 812,    /* "GOST R 34.10-94" */
3348 818,    /* "GOST R 34.10-94 DH" */
3349 809,    /* "GOST R 34.11-94" */
3350 816,    /* "GOST R 34.11-94 PRF" */
3351 807,    /* "GOST R 34.11-94 with GOST R 34.10-2001" */
3352 853,    /* "GOST R 34.11-94 with GOST R 34.10-2001 Cryptocom" */
3353 808,    /* "GOST R 34.11-94 with GOST R 34.10-94" */
3354 852,    /* "GOST R 34.11-94 with GOST R 34.10-94 Cryptocom" */
3355 854,    /* "GOST R 3410-2001 Parameter Set Cryptocom" */
3356 810,    /* "HMAC GOST 34.11-94" */
3357 432,    /* "Hold Instruction Call Issuer" */
3358 430,    /* "Hold Instruction Code" */
3359 431,    /* "Hold Instruction None" */
3360 433,    /* "Hold Instruction Reject" */
3361 634,    /* "ICC or token signature" */
```



```
3362 294,    /* "IPSec End System" */
3363 295,    /* "IPSec Tunnel" */
3364 296,    /* "IPSec User" */
3365 182,    /* "ISO Member Body" */
3366 183,    /* "ISO US Member Body" */
3367 667,    /* "Independent" */
3368 665,    /* "Inherit all" */
3369 647,    /* "International Organizations" */
3370 142,    /* "Invalidity Date" */
3371 504,    /* "MIME MHS" */
3372 388,    /* "Mail" */
3373 383,    /* "Management" */
3374 417,    /* "Microsoft CSP Name" */
3375 135,    /* "Microsoft Commercial Code Signing" */
3376 138,    /* "Microsoft Encrypted File System" */
3377 171,    /* "Microsoft Extension Request" */
3378 134,    /* "Microsoft Individual Code Signing" */
3379 856,    /* "Microsoft Local Key set" */
3380 137,    /* "Microsoft Server Gated Crypto" */
3381 648,    /* "Microsoft Smartcardlogin" */
3382 136,    /* "Microsoft Trust List Signing" */
3383 649,    /* "Microsoft Universal Principal Name" */
3384 393,    /* "NULL" */
3385 404,    /* "NULL" */
3386 72,     /* "Netscape Base Url" */
3387 76,     /* "Netscape CA Policy Url" */
3388 74,     /* "Netscape CA Revocation Url" */
3389 71,     /* "Netscape Cert Type" */
3390 58,     /* "Netscape Certificate Extension" */
3391 79,     /* "Netscape Certificate Sequence" */
3392 78,     /* "Netscape Comment" */
3393 57,     /* "Netscape Communications Corp." */
3394 59,     /* "Netscape Data Type" */
3395 75,     /* "Netscape Renewal Url" */
3396 73,     /* "Netscape Revocation Url" */
3397 77,     /* "Netscape SSL Server Name" */
3398 139,    /* "Netscape Server Gated Crypto" */
3399 178,    /* "OCSP" */
3400 370,    /* "OCSP Archive Cutoff" */
3401 367,    /* "OCSP CRL ID" */
3402 369,    /* "OCSP No Check" */
3403 366,    /* "OCSP Nonce" */
3404 371,    /* "OCSP Service Locator" */
3405 180,    /* "OCSP Signing" */
3406 161,    /* "PBES2" */
3407 69,     /* "PBKDF2" */
3408 162,    /* "PBMAC1" */
3409 127,    /* "PKIX" */
3410 858,    /* "Permanent Identifier" */
3411 164,    /* "Policy Qualifier CPS" */
3412 165,    /* "Policy Qualifier User Notice" */
3413 385,    /* "Private" */
3414 663,    /* "Proxy Certificate Information" */
3415  1,     /* "RSA Data Security, Inc." */
3416  2,     /* "RSA Data Security, Inc. PKCS" */
3417 188,    /* "S/MIME" */
3418 167,    /* "S/MIME Capabilities" */
3419 387,    /* "SNMPv2" */
3420 512,    /* "Secure Electronic Transactions" */
3421 386,    /* "Security" */
3422 394,    /* "Selected Attribute Types" */
3423 143,    /* "Strong Extranet ID" */
3424 398,    /* "Subject Information Access" */
3425 130,    /* "TLS Web Client Authentication" */
3426 129,    /* "TLS Web Server Authentication" */
3427 133,    /* "Time Stamping" */
```

```
3428 375,    /* "Trust Root" */
3429 12,     /* "X509" */
3430 402,    /* "X509v3 AC Targeting" */
3431 746,    /* "X509v3 Any Policy" */
3432 90,     /* "X509v3 Authority Key Identifier" */
3433 87,     /* "X509v3 Basic Constraints" */
3434 103,    /* "X509v3 CRL Distribution Points" */
3435 88,     /* "X509v3 CRL Number" */
3436 141,    /* "X509v3 CRL Reason Code" */
3437 771,    /* "X509v3 Certificate Issuer" */
3438 89,     /* "X509v3 Certificate Policies" */
3439 140,    /* "X509v3 Delta CRL Indicator" */
3440 126,    /* "X509v3 Extended Key Usage" */
3441 857,    /* "X509v3 Freshest CRL" */
3442 748,    /* "X509v3 Inhibit Any Policy" */
3443 86,     /* "X509v3 Issuer Alternative Name" */
3444 770,    /* "X509v3 Issuing Distrubution Point" */
3445 83,     /* "X509v3 Key Usage" */
3446 666,    /* "X509v3 Name Constraints" */
3447 403,    /* "X509v3 No Revocation Available" */
3448 401,    /* "X509v3 Policy Constraints" */
3449 747,    /* "X509v3 Policy Mappings" */
3450 84,     /* "X509v3 Private Key Usage Period" */
3451 85,     /* "X509v3 Subject Alternative Name" */
3452 769,    /* "X509v3 Subject Directory Attributes" */
3453 82,     /* "X509v3 Subject Key Identifier" */
3454 184,    /* "X9.57" */
3455 185,    /* "X9.57 CM ?" */
3456 478,    /* "aRecord" */
3457 289,    /* "aaControls" */
3458 287,    /* "ac-auditEntity" */
3459 397,    /* "ac-proxying" */
3460 288,    /* "ac-targeting" */
3461 446,    /* "account" */
3462 364,    /* "ad dvcs" */
3463 606,    /* "additional verification" */
3464 419,    /* "aes-128-cbc" */
3465 916,    /* "aes-128-cbc-hmac-sha1" */
3466 896,    /* "aes-128-ccm" */
3467 421,    /* "aes-128-cfb" */
3468 650,    /* "aes-128-cfb1" */
3469 653,    /* "aes-128-cfb8" */
3470 904,    /* "aes-128-ctr" */
3471 418,    /* "aes-128-ecb" */
3472 895,    /* "aes-128-gcm" */
3473 420,    /* "aes-128-ofb" */
3474 913,    /* "aes-128-xts" */
3475 423,    /* "aes-192-cbc" */
3476 917,    /* "aes-192-cbc-hmac-sha1" */
3477 899,    /* "aes-192-ccm" */
3478 425,    /* "aes-192-cfb" */
3479 651,    /* "aes-192-cfb1" */
3480 654,    /* "aes-192-cfb8" */
3481 905,    /* "aes-192-ctr" */
3482 422,    /* "aes-192-ecb" */
3483 898,    /* "aes-192-gcm" */
3484 424,    /* "aes-192-ofb" */
3485 427,    /* "aes-256-cbc" */
3486 918,    /* "aes-256-cbc-hmac-sha1" */
3487 902,    /* "aes-256-ccm" */
3488 429,    /* "aes-256-cfb" */
3489 652,    /* "aes-256-cfb1" */
3490 655,    /* "aes-256-cfb8" */
3491 906,    /* "aes-256-ctr" */
3492 426,    /* "aes-256-ecb" */
3493 901,    /* "aes-256-gcm" */
```



```
3494 428,    /* "aes-256-ofb" */
3495 914,    /* "aes-256-xts" */
3496 376,    /* "algorithm" */
3497 484,    /* "associatedDomain" */
3498 485,    /* "associatedName" */
3499 501,    /* "audio" */
3500 882,    /* "authorityRevocationList" */
3501 91,     /* "bf-cbc" */
3502 93,     /* "bf-cfb" */
3503 92,     /* "bf-ecb" */
3504 94,     /* "bf-ofb" */
3505 494,    /* "buildingName" */
3506 860,    /* "businessCategory" */
3507 691,    /* "c2onb191v4" */
3508 692,    /* "c2onb191v5" */
3509 697,    /* "c2onb239v4" */
3510 698,    /* "c2onb239v5" */
3511 684,    /* "c2pnb163v1" */
3512 685,    /* "c2pnb163v2" */
3513 686,    /* "c2pnb163v3" */
3514 687,    /* "c2pnb176v1" */
3515 693,    /* "c2pnb208w1" */
3516 699,    /* "c2pnb272w1" */
3517 700,    /* "c2pnb304w1" */
3518 702,    /* "c2pnb368w1" */
3519 688,    /* "c2tnb191v1" */
3520 689,    /* "c2tnb191v2" */
3521 690,    /* "c2tnb191v3" */
3522 694,    /* "c2tnb239v1" */
3523 695,    /* "c2tnb239v2" */
3524 696,    /* "c2tnb239v3" */
3525 701,    /* "c2tnb359v1" */
3526 703,    /* "c2tnb431r1" */
3527 881,    /* "cACertificate" */
3528 483,    /* "cNAMERecord" */
3529 751,    /* "camellia-128-cbc" */
3530 757,    /* "camellia-128-cfb" */
3531 760,    /* "camellia-128-cfb1" */
3532 763,    /* "camellia-128-cfb8" */
3533 754,    /* "camellia-128-ecb" */
3534 766,    /* "camellia-128-ofb" */
3535 752,    /* "camellia-192-cbc" */
3536 758,    /* "camellia-192-cfb" */
3537 761,    /* "camellia-192-cfb1" */
3538 764,    /* "camellia-192-cfb8" */
3539 755,    /* "camellia-192-ecb" */
3540 767,    /* "camellia-192-ofb" */
3541 753,    /* "camellia-256-cbc" */
3542 759,    /* "camellia-256-cfb" */
3543 762,    /* "camellia-256-cfb1" */
3544 765,    /* "camellia-256-cfb8" */
3545 756,    /* "camellia-256-ecb" */
3546 768,    /* "camellia-256-ofb" */
3547 443,    /* "caseIgnoreIA5StringSyntax" */
3548 108,    /* "cast5-cbc" */
3549 110,    /* "cast5-cfb" */
3550 109,    /* "cast5-ecb" */
3551 111,    /* "cast5-ofb" */
3552 152,    /* "certBag" */
3553 677,    /* "certicom-arc" */
3554 517,    /* "certificate extensions" */
3555 883,    /* "certificateRevocationList" */
3556 54,     /* "challengePassword" */
3557 407,    /* "characteristic-two-field" */
3558 395,    /* "clearance" */
3559 633,    /* "cleartext track 2" */
```

```
3560 894,    /* "cmac" */
3561 13,     /* "commonName" */
3562 513,    /* "content types" */
3563 50,     /* "contentType" */
3564 53,     /* "countersignature" */
3565 14,     /* "countryName" */
3566 153,    /* "crlBag" */
3567 884,    /* "crossCertificatePair" */
3568 806,    /* "cryptocom" */
3569 805,    /* "cryptopro" */
3570 500,    /* "dITRedirect" */
3571 451,    /* "dNSDomain" */
3572 495,    /* "dSAQuality" */
3573 434,    /* "data" */
3574 390,    /* "dcObject" */
3575 891,    /* "deltaRevocationList" */
3576 31,     /* "des-cbc" */
3577 643,    /* "des-cdmf" */
3578 30,     /* "des-cfb" */
3579 656,    /* "des-cfb1" */
3580 657,    /* "des-cfb8" */
3581 29,     /* "des-ecb" */
3582 32,     /* "des-ede" */
3583 43,     /* "des-ede-cbc" */
3584 60,     /* "des-ede-cfb" */
3585 62,     /* "des-ede-ofb" */
3586 33,     /* "des-ede3" */
3587 44,     /* "des-ede3-cbc" */
3588 61,     /* "des-ede3-cfb" */
3589 658,    /* "des-ede3-cfb1" */
3590 659,    /* "des-ede3-cfb8" */
3591 63,     /* "des-ede3-ofb" */
3592 45,     /* "des-ofb" */
3593 107,    /* "description" */
3594 871,    /* "destinationIndicator" */
3595 80,     /* "desx-cbc" */
3596 28,     /* "dhKeyAgreement" */
3597 11,     /* "directory services (X.500)" */
3598 378,    /* "directory services - algorithms" */
3599 887,    /* "distinguishedName" */
3600 892,    /* "dmdName" */
3601 174,    /* "dnQualifier" */
3602 447,    /* "document" */
3603 471,    /* "documentAuthor" */
3604 468,    /* "documentIdentifier" */
3605 472,    /* "documentLocation" */
3606 502,    /* "documentPublisher" */
3607 449,    /* "documentSeries" */
3608 469,    /* "documentTitle" */
3609 470,    /* "documentVersion" */
3610 380,    /* "dod" */
3611 391,    /* "domainComponent" */
3612 452,    /* "domainRelatedObject" */
3613 116,    /* "dsaEncryption" */
3614 67,     /* "dsaEncryption-old" */
3615 66,     /* "dsaWithSHA" */
3616 113,    /* "dsaWithSHA1" */
3617 70,     /* "dsaWithSHA1-old" */
3618 802,    /* "dsa_with_SHA224" */
3619 803,    /* "dsa_with_SHA256" */
3620 297,    /* "dvcs" */
3621 791,    /* "ecdsa-with-Recommended" */
3622 416,    /* "ecdsa-with-SHA1" */
3623 793,    /* "ecdsa-with-SHA224" */
3624 794,    /* "ecdsa-with-SHA256" */
3625 795,    /* "ecdsa-with-SHA384" */
```



```
3626 796,    /* "ecdsa-with-SHA512" */
3627 792,    /* "ecdsa-with-Specified" */
3628 48,     /* "emailAddress" */
3629 632,    /* "encrypted track 2" */
3630 885,    /* "enhancedSearchGuide" */
3631 56,     /* "extendedCertificateAttributes" */
3632 867,    /* "facsimileTelephoneNumber" */
3633 462,    /* "favouriteDrink" */
3634 453,    /* "friendlyCountry" */
3635 490,    /* "friendlyCountryName" */
3636 156,    /* "friendlyName" */
3637 631,    /* "generate cryptogram" */
3638 509,    /* "generationQualifier" */
3639 601,    /* "generic cryptogram" */
3640 99,     /* "givenName" */
3641 814,    /* "gost89-cnt" */
3642 855,    /* "hmac" */
3643 780,    /* "hmac-md5" */
3644 781,    /* "hmac-sha1" */
3645 797,    /* "hmacWithMD5" */
3646 163,    /* "hmacWithSHA1" */
3647 798,    /* "hmacWithSHA224" */
3648 799,    /* "hmacWithSHA256" */
3649 800,    /* "hmacWithSHA384" */
3650 801,    /* "hmacWithSHA512" */
3651 486,    /* "homePostalAddress" */
3652 473,    /* "homeTelephoneNumber" */
3653 466,    /* "host" */
3654 889,    /* "houseIdentifier" */
3655 442,    /* "iA5StringSyntax" */
3656 381,    /* "iana" */
3657 824,    /* "id-Gost28147-89-CryptoPro-A-ParamSet" */
3658 825,    /* "id-Gost28147-89-CryptoPro-B-ParamSet" */
3659 826,    /* "id-Gost28147-89-CryptoPro-C-ParamSet" */
3660 827,    /* "id-Gost28147-89-CryptoPro-D-ParamSet" */
3661 819,    /* "id-Gost28147-89-CryptoPro-KeyMeshing" */
3662 829,    /* "id-Gost28147-89-CryptoPro-Oscar-1-0-ParamSet" */
3663 828,    /* "id-Gost28147-89-CryptoPro-Oscar-1-1-ParamSet" */
3664 830,    /* "id-Gost28147-89-CryptoPro-RIC-1-ParamSet" */
3665 820,    /* "id-Gost28147-89-None-KeyMeshing" */
3666 823,    /* "id-Gost28147-89-TestParamSet" */
3667 840,    /* "id-GostR3410-2001-CryptoPro-A-ParamSet" */
3668 841,    /* "id-GostR3410-2001-CryptoPro-B-ParamSet" */
3669 842,    /* "id-GostR3410-2001-CryptoPro-C-ParamSet" */
3670 843,    /* "id-GostR3410-2001-CryptoPro-XchA-ParamSet" */
3671 844,    /* "id-GostR3410-2001-CryptoPro-XchB-ParamSet" */
3672 839,    /* "id-GostR3410-2001-TestParamSet" */
3673 832,    /* "id-GostR3410-94-CryptoPro-A-ParamSet" */
3674 833,    /* "id-GostR3410-94-CryptoPro-B-ParamSet" */
3675 834,    /* "id-GostR3410-94-CryptoPro-C-ParamSet" */
3676 835,    /* "id-GostR3410-94-CryptoPro-D-ParamSet" */
3677 836,    /* "id-GostR3410-94-CryptoPro-XchA-ParamSet" */
3678 837,    /* "id-GostR3410-94-CryptoPro-XchB-ParamSet" */
3679 838,    /* "id-GostR3410-94-CryptoPro-XchC-ParamSet" */
3680 831,    /* "id-GostR3410-94-TestParamSet" */
3681 845,    /* "id-GostR3410-94-a" */
3682 846,    /* "id-GostR3410-94-aBis" */
3683 847,    /* "id-GostR3410-94-b" */
3684 848,    /* "id-GostR3410-94-bBis" */
3685 822,    /* "id-GostR3411-94-CryptoProParamSet" */
3686 821,    /* "id-GostR3411-94-TestParamSet" */
3687 266,    /* "id-aca" */
3688 355,    /* "id-aca-accessIdentity" */
3689 354,    /* "id-aca-authenticationInfo" */
3690 356,    /* "id-aca-chargingIdentity" */
3691 399,    /* "id-aca-encAttrs" */
```

```
3692 357,    /* "id-aca-group" */
3693 358,    /* "id-aca-role" */
3694 176,    /* "id-ad" */
3695 788,    /* "id-aes128-wrap" */
3696 897,    /* "id-aes128-wrap-pad" */
3697 789,    /* "id-aes192-wrap" */
3698 900,    /* "id-aes192-wrap-pad" */
3699 790,    /* "id-aes256-wrap" */
3700 903,    /* "id-aes256-wrap-pad" */
3701 262,    /* "id-alg" */
3702 893,    /* "id-alg-PWRI-KEK" */
3703 323,    /* "id-alg-des40" */
3704 326,    /* "id-alg-dh-pop" */
3705 325,    /* "id-alg-dh-sig-hmac-sha1" */
3706 324,    /* "id-alg-noSignature" */
3707 907,    /* "id-camellia128-wrap" */
3708 908,    /* "id-camellia192-wrap" */
3709 909,    /* "id-camellia256-wrap" */
3710 268,    /* "id-cct" */
3711 361,    /* "id-cct-PKIData" */
3712 362,    /* "id-cct-PKIResponse" */
3713 360,    /* "id-cct-crs" */
3714 81,     /* "id-ce" */
3715 680,    /* "id-characteristic-two-basis" */
3716 263,    /* "id-cmc" */
3717 334,    /* "id-cmc-addExtensions" */
3718 346,    /* "id-cmc-confirmCertAcceptance" */
3719 330,    /* "id-cmc-dataReturn" */
3720 336,    /* "id-cmc-decryptedPOP" */
3721 335,    /* "id-cmc-encryptedPOP" */
3722 339,    /* "id-cmc-getCRL" */
3723 338,    /* "id-cmc-getCert" */
3724 328,    /* "id-cmc-identification" */
3725 329,    /* "id-cmc-identityProof" */
3726 337,    /* "id-cmc-lraPOPWitness" */
3727 344,    /* "id-cmc-popLinkRandom" */
3728 345,    /* "id-cmc-popLinkWitness" */
3729 343,    /* "id-cmc-queryPending" */
3730 333,    /* "id-cmc-recipientNonce" */
3731 341,    /* "id-cmc-regInfo" */
3732 342,    /* "id-cmc-responseInfo" */
3733 340,    /* "id-cmc-revokeRequest" */
3734 332,    /* "id-cmc-senderNonce" */
3735 327,    /* "id-cmc-statusInfo" */
3736 331,    /* "id-cmc-transactionId" */
3737 787,    /* "id-ct-asciiTextWithCRLF" */
3738 408,    /* "id-ecPublicKey" */
3739 508,    /* "id-hex-multipart-message" */
3740 507,    /* "id-hex-partial-message" */
3741 260,    /* "id-it" */
3742 302,    /* "id-it-caKeyUpdateInfo" */
3743 298,    /* "id-it-caProtEncCert" */
3744 311,    /* "id-it-confirmWaitTime" */
3745 303,    /* "id-it-currentCRL" */
3746 300,    /* "id-it-encKeyPairTypes" */
3747 310,    /* "id-it-implicitConfirm" */
3748 308,    /* "id-it-keyPairParamRep" */
3749 307,    /* "id-it-keyPairParamReq" */
3750 312,    /* "id-it-origPKIMessage" */
3751 301,    /* "id-it-preferredSymmAlg" */
3752 309,    /* "id-it-revPassphrase" */
3753 299,    /* "id-it-signKeyPairTypes" */
3754 305,    /* "id-it-subscriptionRequest" */
3755 306,    /* "id-it-subscriptionResponse" */
3756 784,    /* "id-it-suppLangTags" */
3757 304,    /* "id-it-unsupportedOIDs" */
```



```
3758 128,    /* "id-kp" */
3759 280,    /* "id-mod-attribute-cert" */
3760 274,    /* "id-mod-cmc" */
3761 277,    /* "id-mod-cmp" */
3762 284,    /* "id-mod-cmp2000" */
3763 273,    /* "id-mod-crmf" */
3764 283,    /* "id-mod-dvcs" */
3765 275,    /* "id-mod-kea-profile-88" */
3766 276,    /* "id-mod-kea-profile-93" */
3767 282,    /* "id-mod-ocsp" */
3768 278,    /* "id-mod-qualified-cert-88" */
3769 279,    /* "id-mod-qualified-cert-93" */
3770 281,    /* "id-mod-timestamp-protocol" */
3771 264,    /* "id-on" */
3772 347,    /* "id-on-personalData" */
3773 265,    /* "id-pda" */
3774 352,    /* "id-pda-countryOfCitizenship" */
3775 353,    /* "id-pda-countryOfResidence" */
3776 348,    /* "id-pda-dateOfBirth" */
3777 351,    /* "id-pda-gender" */
3778 349,    /* "id-pda-placeOfBirth" */
3779 175,    /* "id-pe" */
3780 261,    /* "id-pkip" */
3781 258,    /* "id-pkix-mod" */
3782 269,    /* "id-pkix1-explicit-88" */
3783 271,    /* "id-pkix1-explicit-93" */
3784 270,    /* "id-pkix1-implicit-88" */
3785 272,    /* "id-pkix1-implicit-93" */
3786 662,    /* "id-ppl" */
3787 267,    /* "id-qcs" */
3788 359,    /* "id-qcs-pkixQCSyntax-v1" */
3789 259,    /* "id-qt" */
3790 313,    /* "id-regCtrl" */
3791 316,    /* "id-regCtrl-authenticator" */
3792 319,    /* "id-regCtrl-oldCertID" */
3793 318,    /* "id-regCtrl-pkiArchiveOptions" */
3794 317,    /* "id-regCtrl-pkiPublicationInfo" */
3795 320,    /* "id-regCtrl-protocolEncrKey" */
3796 315,    /* "id-regCtrl-regToken" */
3797 314,    /* "id-regInfo" */
3798 322,    /* "id-regInfo-certReq" */
3799 321,    /* "id-regInfo-utf8Pairs" */
3800 191,    /* "id-smime-aa" */
3801 215,    /* "id-smime-aa-contentHint" */
3802 218,    /* "id-smime-aa-contentIdentifier" */
3803 221,    /* "id-smime-aa-contentReference" */
3804 240,    /* "id-smime-aa-dvcs-dvc" */
3805 217,    /* "id-smime-aa-encapContentType" */
3806 222,    /* "id-smime-aa-encrypKeyPref" */
3807 220,    /* "id-smime-aa-equivalentLabels" */
3808 232,    /* "id-smime-aa-ets-CertificateRefs" */
3809 233,    /* "id-smime-aa-ets-RevocationRefs" */
3810 238,    /* "id-smime-aa-ets-archiveTimeStamp" */
3811 237,    /* "id-smime-aa-ets-certCRLTimestamp" */
3812 234,    /* "id-smime-aa-ets-certValues" */
3813 227,    /* "id-smime-aa-ets-commitmentType" */
3814 231,    /* "id-smime-aa-ets-contentTimestamp" */
3815 236,    /* "id-smime-aa-ets-escTimeStamp" */
3816 230,    /* "id-smime-aa-ets-otherSigCert" */
3817 235,    /* "id-smime-aa-ets-revocationValues" */
3818 226,    /* "id-smime-aa-ets-sigPolicyId" */
3819 229,    /* "id-smime-aa-ets-signerAttr" */
3820 228,    /* "id-smime-aa-ets-signerLocation" */
3821 219,    /* "id-smime-aa-macValue" */
3822 214,    /* "id-smime-aa-mlExpandHistory" */
3823 216,    /* "id-smime-aa-msgSigDigest" */
```

```
3824 212,    /* "id-smime-aa-receiptRequest" */
3825 213,    /* "id-smime-aa-securityLabel" */
3826 239,    /* "id-smime-aa-signatureType" */
3827 223,    /* "id-smime-aa-signingCertificate" */
3828 224,    /* "id-smime-aa-smimeEncryptCerts" */
3829 225,    /* "id-smime-aa-timeStampToken" */
3830 192,    /* "id-smime-alg" */
3831 243,    /* "id-smime-alg-3DESwrap" */
3832 246,    /* "id-smime-alg-CMS3DESwrap" */
3833 247,    /* "id-smime-alg-CMSRC2wrap" */
3834 245,    /* "id-smime-alg-ESDH" */
3835 241,    /* "id-smime-alg-ESDHwith3DES" */
3836 242,    /* "id-smime-alg-ESDHwithRC2" */
3837 244,    /* "id-smime-alg-RC2wrap" */
3838 193,    /* "id-smime-cd" */
3839 248,    /* "id-smime-cd-ldap" */
3840 190,    /* "id-smime-ct" */
3841 210,    /* "id-smime-ct-DVCSRequestData" */
3842 211,    /* "id-smime-ct-DVCSResponseData" */
3843 208,    /* "id-smime-ct-TDTInfo" */
3844 207,    /* "id-smime-ct-TSTInfo" */
3845 205,    /* "id-smime-ct-authData" */
3846 786,    /* "id-smime-ct-compressedData" */
3847 209,    /* "id-smime-ct-contentInfo" */
3848 206,    /* "id-smime-ct-publishCert" */
3849 204,    /* "id-smime-ct-receipt" */
3850 195,    /* "id-smime-cti" */
3851 255,    /* "id-smime-cti-ets-proofOfApproval" */
3852 256,    /* "id-smime-cti-ets-proofOfCreation" */
3853 253,    /* "id-smime-cti-ets-proofOfDelivery" */
3854 251,    /* "id-smime-cti-ets-proofOfOrigin" */
3855 252,    /* "id-smime-cti-ets-proofOfReceipt" */
3856 254,    /* "id-smime-cti-ets-proofOfSender" */
3857 189,    /* "id-smime-mod" */
3858 196,    /* "id-smime-mod-cms" */
3859 197,    /* "id-smime-mod-ess" */
3860 202,    /* "id-smime-mod-ets-eSigPolicy-88" */
3861 203,    /* "id-smime-mod-ets-eSigPolicy-97" */
3862 200,    /* "id-smime-mod-ets-eSignature-88" */
3863 201,    /* "id-smime-mod-ets-eSignature-97" */
3864 199,    /* "id-smime-mod-msg-v3" */
3865 198,    /* "id-smime-mod-oid" */
3866 194,    /* "id-smime-spq" */
3867 250,    /* "id-smime-spq-ets-sqt-unotice" */
3868 249,    /* "id-smime-spq-ets-sqt-uri" */
3869 34,     /* "idea-cbc" */
3870 35,     /* "idea-cfb" */
3871 36,     /* "idea-ecb" */
3872 46,     /* "idea-ofb" */
3873 676,    /* "identified-organization" */
3874 461,    /* "info" */
3875 101,    /* "initials" */
3876 869,    /* "internationaliSDNNumber" */
3877 749,    /* "ipsec3" */
3878 750,    /* "ipsec4" */
3879 181,    /* "iso" */
3880 623,    /* "issuer capabilities" */
3881 645,    /* "itu-t" */
3882 492,    /* "janetMailbox" */
3883 646,    /* "joint-iso-itu-t" */
3884 150,    /* "keyBag" */
3885 773,    /* "kisa" */
3886 477,    /* "lastModifiedBy" */
3887 476,    /* "lastModifiedTime" */
3888 157,    /* "localKeyID" */
3889 15,     /* "localityName" */
```



```
3890 480,    /* "mXRecord" */
3891 493,    /* "mailPreferenceOption" */
3892 467,    /* "manager" */
3893  3,     /* "md2" */
3894  7,     /* "md2WithRSAEncryption" */
3895 257,    /* "md4" */
3896 396,    /* "md4WithRSAEncryption" */
3897  4,     /* "md5" */
3898 114,    /* "md5-sha1" */
3899 104,    /* "md5WithRSA" */
3900  8,     /* "md5WithRSAEncryption" */
3901 95,     /* "mdc2" */
3902 96,     /* "mdc2WithRSA" */
3903 875,    /* "member" */
3904 602,    /* "merchant initiated auth" */
3905 514,    /* "message extensions" */
3906 51,     /* "messageDigest" */
3907 911,    /* "mgf1" */
3908 506,    /* "mime-mhs-bodies" */
3909 505,    /* "mime-mhs-headings" */
3910 488,    /* "mobileTelephoneNumber" */
3911 481,    /* "nSRecord" */
3912 173,    /* "name" */
3913 681,    /* "onBasis" */
3914 379,    /* "org" */
3915 17,     /* "organizationName" */
3916 491,    /* "organizationalStatus" */
3917 18,     /* "organizationalUnitName" */
3918 475,    /* "otherMailbox" */
3919 876,    /* "owner" */
3920 489,    /* "pagerTelephoneNumber" */
3921 782,    /* "password based MAC" */
3922 374,    /* "path" */
3923 621,    /* "payment gateway capabilities" */
3924  9,     /* "pbeWithMD2AndDES-CBC" */
3925 168,    /* "pbeWithMD2AndRC2-CBC" */
3926 112,    /* "pbeWithMD5AndCast5CBC" */
3927 10,     /* "pbeWithMD5AndDES-CBC" */
3928 169,    /* "pbeWithMD5AndRC2-CBC" */
3929 148,    /* "pbeWithSHA1And128BitRC2-CBC" */
3930 144,    /* "pbeWithSHA1And128BitRC4" */
3931 147,    /* "pbeWithSHA1And2-KeyTripleDES-CBC" */
3932 146,    /* "pbeWithSHA1And3-KeyTripleDES-CBC" */
3933 149,    /* "pbeWithSHA1And40BitRC2-CBC" */
3934 145,    /* "pbeWithSHA1And40BitRC4" */
3935 170,    /* "pbeWithSHA1AndDES-CBC" */
3936 68,     /* "pbeWithSHA1AndRC2-CBC" */
3937 499,    /* "personalSignature" */
3938 487,    /* "personalTitle" */
3939 464,    /* "photo" */
3940 863,    /* "physicalDeliveryOfficeName" */
3941 437,    /* "pilot" */
3942 439,    /* "pilotAttributeSyntax" */
3943 438,    /* "pilotAttributeType" */
3944 479,    /* "pilotAttributeType27" */
3945 456,    /* "pilotDSA" */
3946 441,    /* "pilotGroups" */
3947 444,    /* "pilotObject" */
3948 440,    /* "pilotObjectClass" */
3949 455,    /* "pilotOrganization" */
3950 445,    /* "pilotPerson" */
3951 186,    /* "pkcs1" */
3952 27,     /* "pkcs3" */
3953 187,    /* "pkcs5" */
3954 20,     /* "pkcs7" */
3955 21,     /* "pkcs7-data" */
```

```
3956 25,     /* "pkcs7-digestData" */
3957 26,     /* "pkcs7-encryptedData" */
3958 23,     /* "pkcs7-envelopedData" */
3959 24,     /* "pkcs7-signedAndEnvelopedData" */
3960 22,     /* "pkcs7-signedData" */
3961 151,    /* "pkcs8ShroudedKeyBag" */
3962 47,     /* "pkcs9" */
3963 862,    /* "postOfficeBox" */
3964 861,    /* "postalAddress" */
3965 661,    /* "postalCode" */
3966 683,    /* "ppBasis" */
3967 872,    /* "preferredDeliveryMethod" */
3968 873,    /* "presentationAddress" */
3969 406,    /* "prime-field" */
3970 409,    /* "prime192v1" */
3971 410,    /* "prime192v2" */
3972 411,    /* "prime192v3" */
3973 412,    /* "prime239v1" */
3974 413,    /* "prime239v2" */
3975 414,    /* "prime239v3" */
3976 415,    /* "prime256v1" */
3977 886,    /* "protocolInformation" */
3978 510,    /* "pseudonym" */
3979 435,    /* "pss" */
3980 286,    /* "qcStatements" */
3981 457,    /* "qualityLabelledData" */
3982 450,    /* "rFC822localPart" */
3983 98,     /* "rc2-40-cbc" */
3984 166,    /* "rc2-64-cbc" */
3985 37,     /* "rc2-cbc" */
3986 39,     /* "rc2-cfb" */
3987 38,     /* "rc2-ecb" */
3988 40,     /* "rc2-ofb" */
3989  5,     /* "rc4" */
3990 97,     /* "rc4-40" */
3991 915,    /* "rc4-hmac-md5" */
3992 120,    /* "rc5-cbc" */
3993 122,    /* "rc5-cfb" */
3994 121,    /* "rc5-ecb" */
3995 123,    /* "rc5-ofb" */
3996 870,    /* "registeredAddress" */
3997 460,    /* "rfc822Mailbox" */
3998 117,    /* "ripemd160" */
3999 119,    /* "ripemd160WithRSA" */
4000 400,    /* "role" */
4001 877,    /* "roleOccupant" */
4002 448,    /* "room" */
4003 463,    /* "roomNumber" */
4004 19,     /* "rsa" */
4005  6,     /* "rsaEncryption" */
4006 644,    /* "rsaOAEPEncryptionSET" */
4007 377,    /* "rsaSignature" */
4008 919,    /* "rsaesOaep" */
4009 912,    /* "rsassaPss" */
4010 124,    /* "run length compression" */
4011 482,    /* "sOARecord" */
4012 155,    /* "safeContentsBag" */
4013 291,    /* "sbgp-autonomousSysNum" */
4014 290,    /* "sbgp-ipAddrBlock" */
4015 292,    /* "sbgp-routerIdentifier" */
4016 159,    /* "sdsiCertificate" */
4017 859,    /* "searchGuide" */
4018 704,    /* "secp112r1" */
4019 705,    /* "secp112r2" */
4020 706,    /* "secp128r1" */
4021 707,    /* "secp128r2" */
```



```
4022 708,    /* "secp160k1" */
4023 709,    /* "secp160r1" */
4024 710,    /* "secp160r2" */
4025 711,    /* "secp192k1" */
4026 712,    /* "secp224k1" */
4027 713,    /* "secp224r1" */
4028 714,    /* "secp256k1" */
4029 715,    /* "secp384r1" */
4030 716,    /* "secp521r1" */
4031 154,    /* "secretBag" */
4032 474,    /* "secretary" */
4033 717,    /* "sect113r1" */
4034 718,    /* "sect113r2" */
4035 719,    /* "sect131r1" */
4036 720,    /* "sect131r2" */
4037 721,    /* "sect163k1" */
4038 722,    /* "sect163r1" */
4039 723,    /* "sect163r2" */
4040 724,    /* "sect193r1" */
4041 725,    /* "sect193r2" */
4042 726,    /* "sect233k1" */
4043 727,    /* "sect233r1" */
4044 728,    /* "sect239k1" */
4045 729,    /* "sect283k1" */
4046 730,    /* "sect283r1" */
4047 731,    /* "sect409k1" */
4048 732,    /* "sect409r1" */
4049 733,    /* "sect571k1" */
4050 734,    /* "sect571r1" */
4051 635,    /* "secure device signature" */
4052 878,    /* "seeAlso" */
4053 777,    /* "seed-cbc" */
4054 779,    /* "seed-cfb" */
4055 776,    /* "seed-ecb" */
4056 778,    /* "seed-ofb" */
4057 105,    /* "serialNumber" */
4058 625,    /* "set-addPolicy" */
4059 515,    /* "set-attr" */
4060 518,    /* "set-brand" */
4061 638,    /* "set-brand-AmericanExpress" */
4062 637,    /* "set-brand-Diners" */
4063 636,    /* "set-brand-IATA-ATA" */
4064 639,    /* "set-brand-JCB" */
4065 641,    /* "set-brand-MasterCard" */
4066 642,    /* "set-brand-Novus" */
4067 640,    /* "set-brand-Visa" */
4068 516,    /* "set-policy" */
4069 607,    /* "set-policy-root" */
4070 624,    /* "set-rootKeyThumb" */
4071 620,    /* "setAttr-Cert" */
4072 628,    /* "setAttr-IssCap-CVM" */
4073 630,    /* "setAttr-IssCap-Sig" */
4074 629,    /* "setAttr-IssCap-T2" */
4075 627,    /* "setAttr-Token-B0Prime" */
4076 626,    /* "setAttr-Token-EMV" */
4077 622,    /* "setAttr-TokenType" */
4078 619,    /* "setCext-IssuerCapabilities" */
4079 615,    /* "setCext-PGWYcapabilities" */
4080 616,    /* "setCext-TokenIdentifier" */
4081 618,    /* "setCext-TokenType" */
4082 617,    /* "setCext-Track2Data" */
4083 611,    /* "setCext-cCertRequired" */
4084 609,    /* "setCext-certType" */
4085 608,    /* "setCext-hashedRoot" */
4086 610,    /* "setCext-merchData" */
4087 613,    /* "setCext-setExt" */
```

```
4088 614,    /* "setCext-setQualf" */
4089 612,    /* "setCext-tunneling" */
4090 540,    /* "setct-AcqCardCodeMsg" */
4091 576,    /* "setct-AcqCardCodeMsgTBE" */
4092 570,    /* "setct-AuthReqTBE" */
4093 534,    /* "setct-AuthReqTBS" */
4094 527,    /* "setct-AuthResBaggage" */
4095 571,    /* "setct-AuthResTBE" */
4096 572,    /* "setct-AuthResTBEX" */
4097 535,    /* "setct-AuthResTBS" */
4098 536,    /* "setct-AuthResTBSX" */
4099 528,    /* "setct-AuthRevReqBaggage" */
4100 577,    /* "setct-AuthRevReqTBE" */
4101 541,    /* "setct-AuthRevReqTBS" */
4102 529,    /* "setct-AuthRevResBaggage" */
4103 542,    /* "setct-AuthRevResData" */
4104 578,    /* "setct-AuthRevResTBE" */
4105 579,    /* "setct-AuthRevResTBEB" */
4106 543,    /* "setct-AuthRevResTBS" */
4107 573,    /* "setct-AuthTokenTBE" */
4108 537,    /* "setct-AuthTokenTBS" */
4109 600,    /* "setct-BCIDistributionTBS" */
4110 558,    /* "setct-BatchAdminReqData" */
4111 592,    /* "setct-BatchAdminReqTBE" */
4112 559,    /* "setct-BatchAdminResData" */
4113 593,    /* "setct-BatchAdminResTBE" */
4114 599,    /* "setct-CRLNotificationResTBS" */
4115 598,    /* "setct-CRLNotificationTBS" */
4116 580,    /* "setct-CapReqTBE" */
4117 581,    /* "setct-CapReqTBEX" */
4118 544,    /* "setct-CapReqTBS" */
4119 545,    /* "setct-CapReqTBSX" */
4120 546,    /* "setct-CapResData" */
4121 582,    /* "setct-CapResTBE" */
4122 583,    /* "setct-CapRevReqTBE" */
4123 584,    /* "setct-CapRevReqTBEX" */
4124 547,    /* "setct-CapRevReqTBS" */
4125 548,    /* "setct-CapRevReqTBSX" */
4126 549,    /* "setct-CapRevResData" */
4127 585,    /* "setct-CapRevResTBE" */
4128 538,    /* "setct-CapTokenData" */
4129 530,    /* "setct-CapTokenSeq" */
4130 574,    /* "setct-CapTokenTBE" */
4131 575,    /* "setct-CapTokenTBEX" */
4132 539,    /* "setct-CapTokenTBS" */
4133 560,    /* "setct-CardCInitResTBS" */
4134 566,    /* "setct-CertInqReqTBS" */
4135 563,    /* "setct-CertReqData" */
4136 595,    /* "setct-CertReqTBE" */
4137 596,    /* "setct-CertReqTBEX" */
4138 564,    /* "setct-CertReqTBS" */
4139 565,    /* "setct-CertResData" */
4140 597,    /* "setct-CertResTBE" */
4141 586,    /* "setct-CredReqTBE" */
4142 587,    /* "setct-CredReqTBEX" */
4143 550,    /* "setct-CredReqTBS" */
4144 551,    /* "setct-CredReqTBSX" */
4145 552,    /* "setct-CredResData" */
4146 588,    /* "setct-CredResTBE" */
4147 589,    /* "setct-CredRevReqTBE" */
4148 590,    /* "setct-CredRevReqTBEX" */
4149 553,    /* "setct-CredRevReqTBS" */
4150 554,    /* "setct-CredRevReqTBSX" */
4151 555,    /* "setct-CredRevResData" */
4152 591,    /* "setct-CredRevResTBE" */
4153 567,    /* "setct-ErrorTBS" */
```



```
4154 526,    /* "setct-HODInput" */
4155 561,    /* "setct-MeAqCInitResTBS" */
4156 522,    /* "setct-OIData" */
4157 519,    /* "setct-PANData" */
4158 521,    /* "setct-PANOnly" */
4159 520,    /* "setct-PANToken" */
4160 556,    /* "setct-PCertReqData" */
4161 557,    /* "setct-PCertResTBS" */
4162 523,    /* "setct-PI" */
4163 532,    /* "setct-PI-TBS" */
4164 524,    /* "setct-PIData" */
4165 525,    /* "setct-PIDataUnsigned" */
4166 568,    /* "setct-PIDualSignedTBE" */
4167 569,    /* "setct-PIUnsignedTBE" */
4168 531,    /* "setct-PInitResData" */
4169 533,    /* "setct-PResData" */
4170 594,    /* "setct-RegFormReqTBE" */
4171 562,    /* "setct-RegFormResTBS" */
4172 604,    /* "setext-pinAny" */
4173 603,    /* "setext-pinSecure" */
4174 605,    /* "setext-track2" */
4175 41,     /* "sha" */
4176 64,     /* "sha1" */
4177 115,    /* "sha1WithRSA" */
4178 65,     /* "sha1WithRSAEncryption" */
4179 675,    /* "sha224" */
4180 671,    /* "sha224WithRSAEncryption" */
4181 672,    /* "sha256" */
4182 668,    /* "sha256WithRSAEncryption" */
4183 673,    /* "sha384" */
4184 669,    /* "sha384WithRSAEncryption" */
4185 674,    /* "sha512" */
4186 670,    /* "sha512WithRSAEncryption" */
4187 42,     /* "shaWithRSAEncryption" */
4188 52,     /* "signingTime" */
4189 454,    /* "simpleSecurityObject" */
4190 496,    /* "singleLevelQuality" */
4191 16,     /* "stateOrProvinceName" */
4192 660,    /* "streetAddress" */
4193 498,    /* "subtreeMaximumQuality" */
4194 497,    /* "subtreeMinimumQuality" */
4195 890,    /* "supportedAlgorithms" */
4196 874,    /* "supportedApplicationContext" */
4197 100,    /* "surname" */
4198 864,    /* "telephoneNumber" */
4199 866,    /* "teletexTerminalIdentifier" */
4200 865,    /* "telexNumber" */
4201 459,    /* "textEncodedORAddress" */
4202 293,    /* "textNotice" */
4203 106,    /* "title" */
4204 682,    /* "tpBasis" */
4205 436,    /* "ucl" */
4206  0,     /* "undefined" */
4207 888,    /* "uniqueMember" */
4208 55,     /* "unstructuredAddress" */
4209 49,     /* "unstructuredName" */
4210 880,    /* "userCertificate" */
4211 465,    /* "userClass" */
4212 458,    /* "userId" */
4213 879,    /* "userPassword" */
4214 373,    /* "valid" */
4215 678,    /* "wap" */
4216 679,    /* "wap-wsg" */
4217 735,    /* "wap-wsg-idm-ecid-wtls1" */
4218 743,    /* "wap-wsg-idm-ecid-wtls10" */
4219 744,    /* "wap-wsg-idm-ecid-wtls11" */
```

```
4220 745,    /* "wap-wsg-idm-ecid-wtls12" */
4221 736,    /* "wap-wsg-idm-ecid-wtls3" */
4222 737,    /* "wap-wsg-idm-ecid-wtls4" */
4223 738,    /* "wap-wsg-idm-ecid-wtls5" */
4224 739,    /* "wap-wsg-idm-ecid-wtls6" */
4225 740,    /* "wap-wsg-idm-ecid-wtls7" */
4226 741,    /* "wap-wsg-idm-ecid-wtls8" */
4227 742,    /* "wap-wsg-idm-ecid-wtls9" */
4228 804,    /* "whirlpool" */
4229 868,    /* "x121Address" */
4230 503,    /* "x500UniqueIdentifier" */
4231 158,    /* "x509Certificate" */
4232 160,    /* "x509Crl" */
4233 125,    /* "zlib compression" */
4234 };

4236 static const unsigned int obj_objs[NUM_OBJ]={
4237  0,     /* OBJ_undef                          0 */
4238 181,    /* OBJ_iso                            1 */
4239 393,    /* OBJ_joint_iso_ccitt                OBJ_joint_iso_itu_t */
4240 404,    /* OBJ_ccitt                          OBJ_itu_t */
4241 645,    /* OBJ_itu_t                          0 */
4242 646,    /* OBJ_joint_iso_itu_t                2 */
4243 434,    /* OBJ_data                           0 9 */
4244 182,    /* OBJ_member_body                    1 2 */
4245 379,    /* OBJ_org                            1 3 */
4246 676,    /* OBJ_identified_organization        1 3 */
4247 11,     /* OBJ_X500                           2 5 */
4248 647,    /* OBJ_international_organizations    2 23 */
4249 380,    /* OBJ_dod                            1 3 6 */
4250 12,     /* OBJ_X509                           2 5 4 */
4251 378,    /* OBJ_X500algorithms                 2 5 8 */
4252 81,     /* OBJ_id_ce                          2 5 29 */
4253 512,    /* OBJ_id_set                         2 23 42 */
4254 678,    /* OBJ_wap                            2 23 43 */
4255 435,    /* OBJ_pss                            0 9 2342 */
4256 183,    /* OBJ_ISO_US                         1 2 840 */
4257 381,    /* OBJ_iana                           1 3 6 1 */
4258 677,    /* OBJ_certicom_arc                   1 3 132 */
4259 394,    /* OBJ_selected_attribute_types       2 5 1 5 */
4260 13,     /* OBJ_commonName                     2 5 4 3 */
4261 100,    /* OBJ_surname                        2 5 4 4 */
4262 105,    /* OBJ_serialNumber                   2 5 4 5 */
4263 14,     /* OBJ_countryName                    2 5 4 6 */
4264 15,     /* OBJ_localityName                   2 5 4 7 */
4265 16,     /* OBJ_stateOrProvinceName            2 5 4 8 */
4266 660,    /* OBJ_streetAddress                  2 5 4 9 */
4267 17,     /* OBJ_organizationName               2 5 4 10 */
4268 18,     /* OBJ_organizationalUnitName         2 5 4 11 */
4269 106,    /* OBJ_title                          2 5 4 12 */
4270 107,    /* OBJ_description                    2 5 4 13 */
4271 859,    /* OBJ_searchGuide                    2 5 4 14 */
4272 860,    /* OBJ_businessCategory               2 5 4 15 */
4273 861,    /* OBJ_postalAddress                  2 5 4 16 */
4274 661,    /* OBJ_postalCode                     2 5 4 17 */
4275 862,    /* OBJ_postOfficeBox                  2 5 4 18 */
4276 863,    /* OBJ_physicalDeliveryOfficeName     2 5 4 19 */
4277 864,    /* OBJ_telephoneNumber                2 5 4 20 */
4278 865,    /* OBJ_telexNumber                    2 5 4 21 */
4279 866,    /* OBJ_teletexTerminalIdentifier      2 5 4 22 */
4280 867,    /* OBJ_facsimileTelephoneNumber       2 5 4 23 */
4281 868,    /* OBJ_x121Address                    2 5 4 24 */
4282 869,    /* OBJ_internationaliSDNNumber        2 5 4 25 */
4283 870,    /* OBJ_registeredAddress              2 5 4 26 */
4284 871,    /* OBJ_destinationIndicator           2 5 4 27 */
4285 872,    /* OBJ_preferredDeliveryMethod        2 5 4 28 */
```



```
4286 873,    /* OBJ_presentationAddress            2 5 4 29 */
4287 874,    /* OBJ_supportedApplicationContext    2 5 4 30 */
4288 875,    /* OBJ_member                         2 5 4 31 */
4289 876,    /* OBJ_owner                          2 5 4 32 */
4290 877,    /* OBJ_roleOccupant                   2 5 4 33 */
4291 878,    /* OBJ_seeAlso                        2 5 4 34 */
4292 879,    /* OBJ_userPassword                   2 5 4 35 */
4293 880,    /* OBJ_userCertificate                2 5 4 36 */
4294 881,    /* OBJ_cACertificate                  2 5 4 37 */
4295 882,    /* OBJ_authorityRevocationList        2 5 4 38 */
4296 883,    /* OBJ_certificateRevocationList      2 5 4 39 */
4297 884,    /* OBJ_crossCertificatePair           2 5 4 40 */
4298 173,    /* OBJ_name                           2 5 4 41 */
4299 99,     /* OBJ_givenName                      2 5 4 42 */
4300 101,    /* OBJ_initials                       2 5 4 43 */
4301 509,    /* OBJ_generationQualifier            2 5 4 44 */
4302 503,    /* OBJ_x500UniqueIdentifier           2 5 4 45 */
4303 174,    /* OBJ_dnQualifier                    2 5 4 46 */
4304 885,    /* OBJ_enhancedSearchGuide            2 5 4 47 */
4305 886,    /* OBJ_protocolInformation            2 5 4 48 */
4306 887,    /* OBJ_distinguishedName              2 5 4 49 */
4307 888,    /* OBJ_uniqueMember                   2 5 4 50 */
4308 889,    /* OBJ_houseIdentifier                2 5 4 51 */
4309 890,    /* OBJ_supportedAlgorithms            2 5 4 52 */
4310 891,    /* OBJ_deltaRevocationList            2 5 4 53 */
4311 892,    /* OBJ_dmdName                        2 5 4 54 */
4312 510,    /* OBJ_pseudonym                      2 5 4 65 */
4313 400,    /* OBJ_role                           2 5 4 72 */
4314 769,    /* OBJ_subject_directory_attributes   2 5 29 9 */
4315 82,     /* OBJ_subject_key_identifier         2 5 29 14 */
4316 83,     /* OBJ_key_usage                      2 5 29 15 */
4317 84,     /* OBJ_private_key_usage_period       2 5 29 16 */
4318 85,     /* OBJ_subject_alt_name               2 5 29 17 */
4319 86,     /* OBJ_issuer_alt_name                2 5 29 18 */
4320 87,     /* OBJ_basic_constraints              2 5 29 19 */
4321 88,     /* OBJ_crl_number                     2 5 29 20 */
4322 141,    /* OBJ_crl_reason                     2 5 29 21 */
4323 430,    /* OBJ_hold_instruction_code          2 5 29 23 */
4324 142,    /* OBJ_invalidity_date                2 5 29 24 */
4325 140,    /* OBJ_delta_crl                      2 5 29 27 */
4326 770,    /* OBJ_issuing_distribution_point     2 5 29 28 */
4327 771,    /* OBJ_certificate_issuer             2 5 29 29 */
4328 666,    /* OBJ_name_constraints               2 5 29 30 */
4329 103,    /* OBJ_crl_distribution_points        2 5 29 31 */
4330 89,     /* OBJ_certificate_policies           2 5 29 32 */
4331 747,    /* OBJ_policy_mappings                2 5 29 33 */
4332 90,     /* OBJ_authority_key_identifier       2 5 29 35 */
4333 401,    /* OBJ_policy_constraints             2 5 29 36 */
4334 126,    /* OBJ_ext_key_usage                  2 5 29 37 */
4335 857,    /* OBJ_freshest_crl                   2 5 29 46 */
4336 748,    /* OBJ_inhibit_any_policy             2 5 29 54 */
4337 402,    /* OBJ_target_information             2 5 29 55 */
4338 403,    /* OBJ_no_rev_avail                   2 5 29 56 */
4339 513,    /* OBJ_set_ctype                      2 23 42 0 */
4340 514,    /* OBJ_set_msgExt                     2 23 42 1 */
4341 515,    /* OBJ_set_attr                       2 23 42 3 */
4342 516,    /* OBJ_set_policy                     2 23 42 5 */
4343 517,    /* OBJ_set_certExt                    2 23 42 7 */
4344 518,    /* OBJ_set_brand                      2 23 42 8 */
4345 679,    /* OBJ_wap_wsg                        2 23 43 1 */
4346 382,    /* OBJ_Directory                      1 3 6 1 1 */
4347 383,    /* OBJ_Management                     1 3 6 1 2 */
4348 384,    /* OBJ_Experimental                   1 3 6 1 3 */
4349 385,    /* OBJ_Private                        1 3 6 1 4 */
4350 386,    /* OBJ_Security                       1 3 6 1 5 */
4351 387,    /* OBJ_SNMPv2                         1 3 6 1 6 */
```

```
4352 388,     /* OBJ_Mail                          1 3 6 1 7 */
4353 376,     /* OBJ_algorithm                     1 3 14 3 2 */
4354 395,     /* OBJ_clearance                     2 5 1 5 55 */
4355 19,      /* OBJ_rsa                           2 5 8 1 1 */
4356 96,      /* OBJ_mdc2WithRSA                   2 5 8 3 100 */
4357 95,      /* OBJ_mdc2                          2 5 8 3 101 */
4358 746,     /* OBJ_any_policy                    2 5 29 32 0 */
4359 910,     /* OBJ_anyExtendedKeyUsage           2 5 29 37 0 */
4360 519,     /* OBJ_setct_PANData                 2 23 42 0 0 */
4361 520,     /* OBJ_setct_PANToken                2 23 42 0 1 */
4362 521,     /* OBJ_setct_PANOnly                 2 23 42 0 2 */
4363 522,     /* OBJ_setct_OIData                  2 23 42 0 3 */
4364 523,     /* OBJ_setct_PI                      2 23 42 0 4 */
4365 524,     /* OBJ_setct_PIData                  2 23 42 0 5 */
4366 525,     /* OBJ_setct_PIDataUnsigned          2 23 42 0 6 */
4367 526,     /* OBJ_setct_HODInput                2 23 42 0 7 */
4368 527,     /* OBJ_setct_AuthResBaggage          2 23 42 0 8 */
4369 528,     /* OBJ_setct_AuthRevReqBaggage       2 23 42 0 9 */
4370 529,     /* OBJ_setct_AuthRevResBaggage       2 23 42 0 10 */
4371 530,     /* OBJ_setct_CapTokenSeq             2 23 42 0 11 */
4372 531,     /* OBJ_setct_PInitResData            2 23 42 0 12 */
4373 532,     /* OBJ_setct_PI_TBS                  2 23 42 0 13 */
4374 533,     /* OBJ_setct_PResData                2 23 42 0 14 */
4375 534,     /* OBJ_setct_AuthReqTBS              2 23 42 0 16 */
4376 535,     /* OBJ_setct_AuthResTBS              2 23 42 0 17 */
4377 536,     /* OBJ_setct_AuthResTBSX             2 23 42 0 18 */
4378 537,     /* OBJ_setct_AuthTokenTBS            2 23 42 0 19 */
4379 538,     /* OBJ_setct_CapTokenData            2 23 42 0 20 */
4380 539,     /* OBJ_setct_CapTokenTBS             2 23 42 0 21 */
4381 540,     /* OBJ_setct_AcqCardCodeMsg          2 23 42 0 22 */
4382 541,     /* OBJ_setct_AuthRevReqTBS           2 23 42 0 23 */
4383 542,     /* OBJ_setct_AuthRevResData          2 23 42 0 24 */
4384 543,     /* OBJ_setct_AuthRevResTBS           2 23 42 0 25 */
4385 544,     /* OBJ_setct_CapReqTBS               2 23 42 0 26 */
4386 545,     /* OBJ_setct_CapReqTBSX              2 23 42 0 27 */
4387 546,     /* OBJ_setct_CapResData              2 23 42 0 28 */
4388 547,     /* OBJ_setct_CapRevReqTBS            2 23 42 0 29 */
4389 548,     /* OBJ_setct_CapRevReqTBSX           2 23 42 0 30 */
4390 549,     /* OBJ_setct_CapRevResData           2 23 42 0 31 */
4391 550,     /* OBJ_setct_CredReqTBS              2 23 42 0 32 */
4392 551,     /* OBJ_setct_CredReqTBSX             2 23 42 0 33 */
4393 552,     /* OBJ_setct_CredResData             2 23 42 0 34 */
4394 553,     /* OBJ_setct_CredRevReqTBS           2 23 42 0 35 */
4395 554,     /* OBJ_setct_CredRevReqTBSX          2 23 42 0 36 */
4396 555,     /* OBJ_setct_CredRevResData          2 23 42 0 37 */
4397 556,     /* OBJ_setct_PCertReqData            2 23 42 0 38 */
4398 557,     /* OBJ_setct_PCertResTBS             2 23 42 0 39 */
4399 558,     /* OBJ_setct_BatchAdminReqData       2 23 42 0 40 */
4400 559,     /* OBJ_setct_BatchAdminResData       2 23 42 0 41 */
4401 560,     /* OBJ_setct_CardCInitResTBS         2 23 42 0 42 */
4402 561,     /* OBJ_setct_MeAqCInitResTBS         2 23 42 0 43 */
4403 562,     /* OBJ_setct_RegFormResTBS           2 23 42 0 44 */
4404 563,     /* OBJ_setct_CertReqData             2 23 42 0 45 */
4405 564,     /* OBJ_setct_CertReqTBS              2 23 42 0 46 */
4406 565,     /* OBJ_setct_CertResData             2 23 42 0 47 */
4407 566,     /* OBJ_setct_CertInqReqTBS           2 23 42 0 48 */
4408 567,     /* OBJ_setct_ErrorTBS                2 23 42 0 49 */
4409 568,     /* OBJ_setct_PIDualSignedTBE         2 23 42 0 50 */
4410 569,     /* OBJ_setct_PIUnsignedTBE           2 23 42 0 51 */
4411 570,     /* OBJ_setct_AuthReqTBE              2 23 42 0 52 */
4412 571,     /* OBJ_setct_AuthResTBE              2 23 42 0 53 */
4413 572,     /* OBJ_setct_AuthResTBEX             2 23 42 0 54 */
4414 573,     /* OBJ_setct_AuthTokenTBE            2 23 42 0 55 */
4415 574,     /* OBJ_setct_CapTokenTBE             2 23 42 0 56 */
4416 575,     /* OBJ_setct_CapTokenTBEX            2 23 42 0 57 */
4417 576,     /* OBJ_setct_AcqCardCodeMsgTBE       2 23 42 0 58 */
```



```
4418 577,     /* OBJ_setct_AuthRevReqTBE           2 23 42 0 59 */
4419 578,     /* OBJ_setct_AuthRevResTBE           2 23 42 0 60 */
4420 579,     /* OBJ_setct_AuthRevResTBEB          2 23 42 0 61 */
4421 580,     /* OBJ_setct_CapReqTBE               2 23 42 0 62 */
4422 581,     /* OBJ_setct_CapReqTBEX              2 23 42 0 63 */
4423 582,     /* OBJ_setct_CapResTBE               2 23 42 0 64 */
4424 583,     /* OBJ_setct_CapRevReqTBE            2 23 42 0 65 */
4425 584,     /* OBJ_setct_CapRevReqTBEX           2 23 42 0 66 */
4426 585,     /* OBJ_setct_CapRevResTBE            2 23 42 0 67 */
4427 586,     /* OBJ_setct_CredReqTBE              2 23 42 0 68 */
4428 587,     /* OBJ_setct_CredReqTBEX             2 23 42 0 69 */
4429 588,     /* OBJ_setct_CredResTBE              2 23 42 0 70 */
4430 589,     /* OBJ_setct_CredRevReqTBE           2 23 42 0 71 */
4431 590,     /* OBJ_setct_CredRevReqTBEX          2 23 42 0 72 */
4432 591,     /* OBJ_setct_CredRevResTBE           2 23 42 0 73 */
4433 592,     /* OBJ_setct_BatchAdminReqTBE        2 23 42 0 74 */
4434 593,     /* OBJ_setct_BatchAdminResTBE        2 23 42 0 75 */
4435 594,     /* OBJ_setct_RegFormReqTBE           2 23 42 0 76 */
4436 595,     /* OBJ_setct_CertReqTBE              2 23 42 0 77 */
4437 596,     /* OBJ_setct_CertReqTBEX             2 23 42 0 78 */
4438 597,     /* OBJ_setct_CertResTBE              2 23 42 0 79 */
4439 598,     /* OBJ_setct_CRLNotificationTBS      2 23 42 0 80 */
4440 599,     /* OBJ_setct_CRLNotificationResTBS   2 23 42 0 81 */
4441 600,     /* OBJ_setct_BCIDistributionTBS      2 23 42 0 82 */
4442 601,     /* OBJ_setext_genCrypt               2 23 42 1 1 */
4443 602,     /* OBJ_setext_miAuth                 2 23 42 1 3 */
4444 603,     /* OBJ_setext_pinSecure              2 23 42 1 4 */
4445 604,     /* OBJ_setext_pinAny                 2 23 42 1 5 */
4446 605,     /* OBJ_setext_track2                 2 23 42 1 7 */
4447 606,     /* OBJ_setext_cv                     2 23 42 1 8 */
4448 620,     /* OBJ_setAttr_Cert                  2 23 42 3 0 */
4449 621,     /* OBJ_setAttr_PGWYcap               2 23 42 3 1 */
4450 622,     /* OBJ_setAttr_TokenType             2 23 42 3 2 */
4451 623,     /* OBJ_setAttr_IssCap                2 23 42 3 3 */
4452 607,     /* OBJ_set_policy_root               2 23 42 5 0 */
4453 608,     /* OBJ_setCext_hashedRoot            2 23 42 7 0 */
4454 609,     /* OBJ_setCext_certType              2 23 42 7 1 */
4455 610,     /* OBJ_setCext_merchData             2 23 42 7 2 */
4456 611,     /* OBJ_setCext_cCertRequired         2 23 42 7 3 */
4457 612,     /* OBJ_setCext_tunneling             2 23 42 7 4 */
4458 613,     /* OBJ_setCext_setExt                2 23 42 7 5 */
4459 614,     /* OBJ_setCext_setQualf              2 23 42 7 6 */
4460 615,     /* OBJ_setCext_PGWYcapabilities      2 23 42 7 7 */
4461 616,     /* OBJ_setCext_TokenIdentifier       2 23 42 7 8 */
4462 617,     /* OBJ_setCext_Track2Data            2 23 42 7 9 */
4463 618,     /* OBJ_setCext_TokenType             2 23 42 7 10 */
4464 619,     /* OBJ_setCext_IssuerCapabilities    2 23 42 7 11 */
4465 636,     /* OBJ_set_brand_IATA_ATA            2 23 42 8 1 */
4466 640,     /* OBJ_set_brand_Visa                2 23 42 8 4 */
4467 641,     /* OBJ_set_brand_MasterCard          2 23 42 8 5 */
4468 637,     /* OBJ_set_brand_Diners              2 23 42 8 30 */
4469 638,     /* OBJ_set_brand_AmericanExpress     2 23 42 8 34 */
4470 639,     /* OBJ_set_brand_JCB                 2 23 42 8 35 */
4471 805,     /* OBJ_cryptopro                     1 2 643 2 2 */
4472 806,     /* OBJ_cryptocom                     1 2 643 2 9 */
4473 184,     /* OBJ_X9_57                         1 2 840 10040 */
4474 405,     /* OBJ_ansi_X9_62                    1 2 840 10045 */
4475 389,     /* OBJ_Enterprises                   1 3 6 1 4 1 */
4476 504,     /* OBJ_mime_mhs                      1 3 6 1 7 1 */
4477 104,     /* OBJ_md5WithRSA                    1 3 14 3 2 3 */
4478 29,      /* OBJ_des_ecb                       1 3 14 3 2 6 */
4479 31,      /* OBJ_des_cbc                       1 3 14 3 2 7 */
4480 45,      /* OBJ_des_ofb64                     1 3 14 3 2 8 */
4481 30,      /* OBJ_des_cfb64                     1 3 14 3 2 9 */
4482 377,     /* OBJ_rsaSignature                  1 3 14 3 2 11 */
4483 67,      /* OBJ_dsa_2                         1 3 14 3 2 12 */
```

```
4484 66,     /* OBJ_dsaWithSHA                  1 3 14 3 2 13 */
4485 42,     /* OBJ_shaWithRSAEncryption        1 3 14 3 2 15 */
4486 32,     /* OBJ_des_ede_ecb                 1 3 14 3 2 17 */
4487 41,     /* OBJ_sha                         1 3 14 3 2 18 */
4488 64,     /* OBJ_sha1                        1 3 14 3 2 26 */
4489 70,     /* OBJ_dsaWithSHA1_2               1 3 14 3 2 27 */
4490 115,    /* OBJ_sha1WithRSA                 1 3 14 3 2 29 */
4491 117,    /* OBJ_ripemd160                   1 3 36 3 2 1 */
4492 143,    /* OBJ_sxnet                       1 3 101 1 4 1 */
4493 721,    /* OBJ_sect163k1                   1 3 132 0 1 */
4494 722,    /* OBJ_sect163r1                   1 3 132 0 2 */
4495 728,    /* OBJ_sect239k1                   1 3 132 0 3 */
4496 717,    /* OBJ_sect113r1                   1 3 132 0 4 */
4497 718,    /* OBJ_sect113r2                   1 3 132 0 5 */
4498 704,    /* OBJ_secp112r1                   1 3 132 0 6 */
4499 705,    /* OBJ_secp112r2                   1 3 132 0 7 */
4500 709,    /* OBJ_secp160r1                   1 3 132 0 8 */
4501 708,    /* OBJ_secp160k1                   1 3 132 0 9 */
4502 714,    /* OBJ_secp256k1                   1 3 132 0 10 */
4503 723,    /* OBJ_sect163r2                   1 3 132 0 15 */
4504 729,    /* OBJ_sect283k1                   1 3 132 0 16 */
4505 730,    /* OBJ_sect283r1                   1 3 132 0 17 */
4506 719,    /* OBJ_sect131r1                   1 3 132 0 22 */
4507 720,    /* OBJ_sect131r2                   1 3 132 0 23 */
4508 724,    /* OBJ_sect193r1                   1 3 132 0 24 */
4509 725,    /* OBJ_sect193r2                   1 3 132 0 25 */
4510 726,    /* OBJ_sect233k1                   1 3 132 0 26 */
4511 727,    /* OBJ_sect233r1                   1 3 132 0 27 */
4512 706,    /* OBJ_secp128r1                   1 3 132 0 28 */
4513 707,    /* OBJ_secp128r2                   1 3 132 0 29 */
4514 710,    /* OBJ_secp160r2                   1 3 132 0 30 */
4515 711,    /* OBJ_secp192k1                   1 3 132 0 31 */
4516 712,    /* OBJ_secp224k1                   1 3 132 0 32 */
4517 713,    /* OBJ_secp224r1                   1 3 132 0 33 */
4518 715,    /* OBJ_secp384r1                   1 3 132 0 34 */
4519 716,    /* OBJ_secp521r1                   1 3 132 0 35 */
4520 731,    /* OBJ_sect409k1                   1 3 132 0 36 */
4521 732,    /* OBJ_sect409r1                   1 3 132 0 37 */
4522 733,    /* OBJ_sect571k1                   1 3 132 0 38 */
4523 734,    /* OBJ_sect571r1                   1 3 132 0 39 */
4524 624,    /* OBJ_set_rootKeyThumb            2 23 42 3 0 0 */
4525 625,    /* OBJ_set_addPolicy               2 23 42 3 0 1 */
4526 626,    /* OBJ_setAttr_Token_EMV           2 23 42 3 2 1 */
4527 627,    /* OBJ_setAttr_Token_B0Prime       2 23 42 3 2 2 */
4528 628,    /* OBJ_setAttr_IssCap_CVM          2 23 42 3 3 3 */
4529 629,    /* OBJ_setAttr_IssCap_T2           2 23 42 3 3 4 */
4530 630,    /* OBJ_setAttr_IssCap_Sig          2 23 42 3 3 5 */
4531 642,    /* OBJ_set_brand_Novus             2 23 42 8 6011 */
4532 735,    /* OBJ_wap_wsg_idm_ecid_wtls1      2 23 43 1 4 1 */
4533 736,    /* OBJ_wap_wsg_idm_ecid_wtls3      2 23 43 1 4 3 */
4534 737,    /* OBJ_wap_wsg_idm_ecid_wtls4      2 23 43 1 4 4 */
4535 738,    /* OBJ_wap_wsg_idm_ecid_wtls5      2 23 43 1 4 5 */
4536 739,    /* OBJ_wap_wsg_idm_ecid_wtls6      2 23 43 1 4 6 */
4537 740,    /* OBJ_wap_wsg_idm_ecid_wtls7      2 23 43 1 4 7 */
4538 741,    /* OBJ_wap_wsg_idm_ecid_wtls8      2 23 43 1 4 8 */
4539 742,    /* OBJ_wap_wsg_idm_ecid_wtls9      2 23 43 1 4 9 */
4540 743,    /* OBJ_wap_wsg_idm_ecid_wtls10     2 23 43 1 4 10 */
4541 744,    /* OBJ_wap_wsg_idm_ecid_wtls11     2 23 43 1 4 11 */
4542 745,    /* OBJ_wap_wsg_idm_ecid_wtls12     2 23 43 1 4 12 */
4543 804,    /* OBJ_whirlpool                   1 0 10118 3 0 55 */
4544 124,    /* OBJ_rle_compression             1 1 1 1 666 1 */
4545 773,    /* OBJ_kisa                        1 2 410 200004 */
4546 807,    /* OBJ_id_GostR3411_94_with_GostR3410_2001 1 2 643 2 2 3 */
4547 808,    /* OBJ_id_GostR3411_94_with_GostR3410_94 1 2 643 2 2 4 */
4548 809,    /* OBJ_id_GostR3411_94             1 2 643 2 2 9 */
4549 810,    /* OBJ_id_HMACGostR3411_94         1 2 643 2 2 10 */
```



```
4550 811,    /* OBJ_id_GostR3410_2001           1 2 643 2 2 19 */
4551 812,    /* OBJ_id_GostR3410_94             1 2 643 2 2 20 */
4552 813,    /* OBJ_id_Gost28147_89             1 2 643 2 2 21 */
4553 815,    /* OBJ_id_Gost28147_89_MAC         1 2 643 2 2 22 */
4554 816,    /* OBJ_id_GostR3411_94_prf         1 2 643 2 2 23 */
4555 817,    /* OBJ_id_GostR3410_2001DH         1 2 643 2 2 98 */
4556 818,    /* OBJ_id_GostR3410_94DH           1 2 643 2 2 99 */
4557  1,     /* OBJ_rsadsi                      1 2 840 113549 */
4558 185,    /* OBJ_X9cm                        1 2 840 10040 4 */
4559 127,    /* OBJ_id_pkix                     1 3 6 1 5 5 7 */
4560 505,    /* OBJ_mime_mhs_headings           1 3 6 1 7 1 1 */
4561 506,    /* OBJ_mime_mhs_bodies             1 3 6 1 7 1 2 */
4562 119,    /* OBJ_ripemd160WithRSA            1 3 36 3 3 1 2 */
4563 631,    /* OBJ_setAttr_GenCryptgrm         2 23 42 3 3 3 1 */
4564 632,    /* OBJ_setAttr_T2Enc               2 23 42 3 3 4 1 */
4565 633,    /* OBJ_setAttr_T2cleartxt          2 23 42 3 3 4 2 */
4566 634,    /* OBJ_setAttr_TokICCsig           2 23 42 3 3 5 1 */
4567 635,    /* OBJ_setAttr_SecDevSig           2 23 42 3 3 5 2 */
4568 436,    /* OBJ_ucl                         0 9 2342 19200300 */
4569 820,    /* OBJ_id_Gost28147_89_None_KeyMeshing 1 2 643 2 2 14 0 */
4570 819,    /* OBJ_id_Gost28147_89_CryptoPro_KeyMeshing 1 2 643 2 2 14 1 */
4571 845,    /* OBJ_id_GostR3410_94_a           1 2 643 2 2 20 1 */
4572 846,    /* OBJ_id_GostR3410_94_aBis        1 2 643 2 2 20 2 */
4573 847,    /* OBJ_id_GostR3410_94_b           1 2 643 2 2 20 3 */
4574 848,    /* OBJ_id_GostR3410_94_bBis        1 2 643 2 2 20 4 */
4575 821,    /* OBJ_id_GostR3411_94_TestParamSet 1 2 643 2 2 30 0 */
4576 822,    /* OBJ_id_GostR3411_94_CryptoProParamSet 1 2 643 2 2 30 1 */
4577 823,    /* OBJ_id_Gost28147_89_TestParamSet 1 2 643 2 2 31 0 */
4578 824,    /* OBJ_id_Gost28147_89_CryptoPro_A_ParamSet 1 2 643 2 2 31 1 */
4579 825,    /* OBJ_id_Gost28147_89_CryptoPro_B_ParamSet 1 2 643 2 2 31 2 */
4580 826,    /* OBJ_id_Gost28147_89_CryptoPro_C_ParamSet 1 2 643 2 2 31 3 */
4581 827,    /* OBJ_id_Gost28147_89_CryptoPro_D_ParamSet 1 2 643 2 2 31 4 */
4582 828,    /* OBJ_id_Gost28147_89_CryptoPro_Oscar_1_1_ParamSet 1 2 643 2 2 31 5 */
4583 829,    /* OBJ_id_Gost28147_89_CryptoPro_Oscar_1_0_ParamSet 1 2 643 2 2 31 6 */
4584 830,    /* OBJ_id_Gost28147_89_CryptoPro_RIC_1_ParamSet 1 2 643 2 2 31 7 */
4585 831,    /* OBJ_id_GostR3410_94_TestParamSet 1 2 643 2 2 32 0 */
4586 832,    /* OBJ_id_GostR3410_94_CryptoPro_A_ParamSet 1 2 643 2 2 32 2 */
4587 833,    /* OBJ_id_GostR3410_94_CryptoPro_B_ParamSet 1 2 643 2 2 32 3 */
4588 834,    /* OBJ_id_GostR3410_94_CryptoPro_C_ParamSet 1 2 643 2 2 32 4 */
4589 835,    /* OBJ_id_GostR3410_94_CryptoPro_D_ParamSet 1 2 643 2 2 32 5 */
4590 836,    /* OBJ_id_GostR3410_94_CryptoPro_XchA_ParamSet 1 2 643 2 2 33 1 */
4591 837,    /* OBJ_id_GostR3410_94_CryptoPro_XchB_ParamSet 1 2 643 2 2 33 2 */
4592 838,    /* OBJ_id_GostR3410_94_CryptoPro_XchC_ParamSet 1 2 643 2 2 33 3 */
4593 839,    /* OBJ_id_GostR3410_2001_TestParamSet 1 2 643 2 2 35 0 */
4594 840,    /* OBJ_id_GostR3410_2001_CryptoPro_A_ParamSet 1 2 643 2 2 35 1 */
4595 841,    /* OBJ_id_GostR3410_2001_CryptoPro_B_ParamSet 1 2 643 2 2 35 2 */
4596 842,    /* OBJ_id_GostR3410_2001_CryptoPro_C_ParamSet 1 2 643 2 2 35 3 */
4597 843,    /* OBJ_id_GostR3410_2001_CryptoPro_XchA_ParamSet 1 2 643 2 2 36 0 */
4598 844,    /* OBJ_id_GostR3410_2001_CryptoPro_XchB_ParamSet 1 2 643 2 2 36 1 */
4599  2,     /* OBJ_pkcs                        1 2 840 113549 1 */
4600 431,    /* OBJ_hold_instruction_none       1 2 840 10040 2 1 */
4601 432,    /* OBJ_hold_instruction_call_issuer 1 2 840 10040 2 2 */
4602 433,    /* OBJ_hold_instruction_reject     1 2 840 10040 2 3 */
4603 116,    /* OBJ_dsa                         1 2 840 10040 4 1 */
4604 113,    /* OBJ_dsaWithSHA1                 1 2 840 10040 4 3 */
4605 406,    /* OBJ_X9_62_prime_field           1 2 840 10045 1 1 */
4606 407,    /* OBJ_X9_62_characteristic_two_field 1 2 840 10045 1 2 */
4607 408,    /* OBJ_X9_62_id_ecPublicKey        1 2 840 10045 2 1 */
4608 416,    /* OBJ_ecdsa_with_SHA1             1 2 840 10045 4 1 */
4609 791,    /* OBJ_ecdsa_with_Recommended      1 2 840 10045 4 2 */
4610 792,    /* OBJ_ecdsa_with_Specified        1 2 840 10045 4 3 */
4611 258,    /* OBJ_id_pkix_mod                 1 3 6 1 5 5 7 0 */
4612 175,    /* OBJ_id_pe                       1 3 6 1 5 5 7 1 */
4613 259,    /* OBJ_id_qt                       1 3 6 1 5 5 7 2 */
4614 128,    /* OBJ_id_kp                       1 3 6 1 5 5 7 3 */
4615 260,    /* OBJ_id_it                       1 3 6 1 5 5 7 4 */
```

```
4616 261,    /* OBJ_id_pkip                        1 3 6 1 5 5 7 5 */
4617 262,    /* OBJ_id_alg                         1 3 6 1 5 5 7 6 */
4618 263,    /* OBJ_id_cmc                         1 3 6 1 5 5 7 7 */
4619 264,    /* OBJ_id_on                          1 3 6 1 5 5 7 8 */
4620 265,    /* OBJ_id_pda                         1 3 6 1 5 5 7 9 */
4621 266,    /* OBJ_id_aca                         1 3 6 1 5 5 7 10 */
4622 267,    /* OBJ_id_qcs                         1 3 6 1 5 5 7 11 */
4623 268,    /* OBJ_id_cct                         1 3 6 1 5 5 7 12 */
4624 662,    /* OBJ_id_ppl                         1 3 6 1 5 5 7 21 */
4625 176,    /* OBJ_id_ad                          1 3 6 1 5 5 7 48 */
4626 507,    /* OBJ_id_hex_partial_message         1 3 6 1 7 1 1 1 */
4627 508,    /* OBJ_id_hex_multipart_message       1 3 6 1 7 1 1 2 */
4628 57,     /* OBJ_netscape                       2 16 840 1 113730 */
4629 754,    /* OBJ_camellia_128_ecb               0 3 4401 5 3 1 9 1 */
4630 766,    /* OBJ_camellia_128_ofb128            0 3 4401 5 3 1 9 3 */
4631 757,    /* OBJ_camellia_128_cfb128            0 3 4401 5 3 1 9 4 */
4632 755,    /* OBJ_camellia_192_ecb               0 3 4401 5 3 1 9 21 */
4633 767,    /* OBJ_camellia_192_ofb128            0 3 4401 5 3 1 9 23 */
4634 758,    /* OBJ_camellia_192_cfb128            0 3 4401 5 3 1 9 24 */
4635 756,    /* OBJ_camellia_256_ecb               0 3 4401 5 3 1 9 41 */
4636 768,    /* OBJ_camellia_256_ofb128            0 3 4401 5 3 1 9 43 */
4637 759,    /* OBJ_camellia_256_cfb128            0 3 4401 5 3 1 9 44 */
4638 437,    /* OBJ_pilot                          0 9 2342 19200300 100 */
4639 776,    /* OBJ_seed_ecb                       1 2 410 200004 1 3 */
4640 777,    /* OBJ_seed_cbc                       1 2 410 200004 1 4 */
4641 779,    /* OBJ_seed_cfb128                    1 2 410 200004 1 5 */
4642 778,    /* OBJ_seed_ofb128                    1 2 410 200004 1 6 */
4643 852,    /* OBJ_id_GostR3411_94_with_GostR3410_94_cc 1 2 643 2 9 1 3 3 */
4644 853,    /* OBJ_id_GostR3411_94_with_GostR3410_2001_cc 1 2 643 2 9 1 3 4 */
4645 850,    /* OBJ_id_GostR3410_94_cc             1 2 643 2 9 1 5 3 */
4646 851,    /* OBJ_id_GostR3410_2001_cc           1 2 643 2 9 1 5 4 */
4647 849,    /* OBJ_id_Gost28147_89_cc             1 2 643 2 9 1 6 1 */
4648 854,    /* OBJ_id_GostR3410_2001_ParamSet_cc 1 2 643 2 9 1 8 1 */
4649 186,    /* OBJ_pkcs1                          1 2 840 113549 1 1 */
4650 27,     /* OBJ_pkcs3                          1 2 840 113549 1 3 */
4651 187,    /* OBJ_pkcs5                          1 2 840 113549 1 5 */
4652 20,     /* OBJ_pkcs7                          1 2 840 113549 1 7 */
4653 47,     /* OBJ_pkcs9                          1 2 840 113549 1 9 */
4654  3,     /* OBJ_md2                            1 2 840 113549 2 2 */
4655 257,    /* OBJ_md4                            1 2 840 113549 2 4 */
4656  4,     /* OBJ_md5                            1 2 840 113549 2 5 */
4657 797,    /* OBJ_hmacWithMD5                    1 2 840 113549 2 6 */
4658 163,    /* OBJ_hmacWithSHA1                   1 2 840 113549 2 7 */
4659 798,    /* OBJ_hmacWithSHA224                 1 2 840 113549 2 8 */
4660 799,    /* OBJ_hmacWithSHA256                 1 2 840 113549 2 9 */
4661 800,    /* OBJ_hmacWithSHA384                 1 2 840 113549 2 10 */
4662 801,    /* OBJ_hmacWithSHA512                 1 2 840 113549 2 11 */
4663 37,     /* OBJ_rc2_cbc                        1 2 840 113549 3 2 */
4664  5,     /* OBJ_rc4                            1 2 840 113549 3 4 */
4665 44,     /* OBJ_des_ede3_cbc                   1 2 840 113549 3 7 */
4666 120,    /* OBJ_rc5_cbc                        1 2 840 113549 3 8 */
4667 643,    /* OBJ_des_cdmf                       1 2 840 113549 3 10 */
4668 680,    /* OBJ_X9_62_id_characteristic_two_basis 1 2 840 10045 1 2 3 */
4669 684,    /* OBJ_X9_62_c2pnb163v1               1 2 840 10045 3 0 1 */
4670 685,    /* OBJ_X9_62_c2pnb163v2               1 2 840 10045 3 0 2 */
4671 686,    /* OBJ_X9_62_c2pnb163v3               1 2 840 10045 3 0 3 */
4672 687,    /* OBJ_X9_62_c2pnb176v1               1 2 840 10045 3 0 4 */
4673 688,    /* OBJ_X9_62_c2tnb191v1               1 2 840 10045 3 0 5 */
4674 689,    /* OBJ_X9_62_c2tnb191v2               1 2 840 10045 3 0 6 */
4675 690,    /* OBJ_X9_62_c2tnb191v3               1 2 840 10045 3 0 7 */
4676 691,    /* OBJ_X9_62_c2onb191v4               1 2 840 10045 3 0 8 */
4677 692,    /* OBJ_X9_62_c2onb191v5               1 2 840 10045 3 0 9 */
4678 693,    /* OBJ_X9_62_c2pnb208w1               1 2 840 10045 3 0 10 */
4679 694,    /* OBJ_X9_62_c2tnb239v1               1 2 840 10045 3 0 11 */
4680 695,    /* OBJ_X9_62_c2tnb239v2               1 2 840 10045 3 0 12 */
4681 696,    /* OBJ_X9_62_c2tnb239v3               1 2 840 10045 3 0 13 */
```



```
4682 697,    /* OBJ_X9_62_c2onb239v4               1 2 840 10045 3 0 14 */
4683 698,    /* OBJ_X9_62_c2onb239v5               1 2 840 10045 3 0 15 */
4684 699,    /* OBJ_X9_62_c2pnb272w1               1 2 840 10045 3 0 16 */
4685 700,    /* OBJ_X9_62_c2pnb304w1               1 2 840 10045 3 0 17 */
4686 701,    /* OBJ_X9_62_c2tnb359v1               1 2 840 10045 3 0 18 */
4687 702,    /* OBJ_X9_62_c2pnb368w1               1 2 840 10045 3 0 19 */
4688 703,    /* OBJ_X9_62_c2tnb431r1               1 2 840 10045 3 0 20 */
4689 409,    /* OBJ_X9_62_prime192v1               1 2 840 10045 3 1 1 */
4690 410,    /* OBJ_X9_62_prime192v2               1 2 840 10045 3 1 2 */
4691 411,    /* OBJ_X9_62_prime192v3               1 2 840 10045 3 1 3 */
4692 412,    /* OBJ_X9_62_prime239v1               1 2 840 10045 3 1 4 */
4693 413,    /* OBJ_X9_62_prime239v2               1 2 840 10045 3 1 5 */
4694 414,    /* OBJ_X9_62_prime239v3               1 2 840 10045 3 1 6 */
4695 415,    /* OBJ_X9_62_prime256v1               1 2 840 10045 3 1 7 */
4696 793,    /* OBJ_ecdsa_with_SHA224              1 2 840 10045 4 3 1 */
4697 794,    /* OBJ_ecdsa_with_SHA256              1 2 840 10045 4 3 2 */
4698 795,    /* OBJ_ecdsa_with_SHA384              1 2 840 10045 4 3 3 */
4699 796,    /* OBJ_ecdsa_with_SHA512              1 2 840 10045 4 3 4 */
4700 269,    /* OBJ_id_pkix1_explicit_88           1 3 6 1 5 5 7 0 1 */
4701 270,    /* OBJ_id_pkix1_implicit_88           1 3 6 1 5 5 7 0 2 */
4702 271,    /* OBJ_id_pkix1_explicit_93           1 3 6 1 5 5 7 0 3 */
4703 272,    /* OBJ_id_pkix1_implicit_93           1 3 6 1 5 5 7 0 4 */
4704 273,    /* OBJ_id_mod_crmf                    1 3 6 1 5 5 7 0 5 */
4705 274,    /* OBJ_id_mod_cmc                     1 3 6 1 5 5 7 0 6 */
4706 275,    /* OBJ_id_mod_kea_profile_88          1 3 6 1 5 5 7 0 7 */
4707 276,    /* OBJ_id_mod_kea_profile_93          1 3 6 1 5 5 7 0 8 */
4708 277,    /* OBJ_id_mod_cmp                     1 3 6 1 5 5 7 0 9 */
4709 278,    /* OBJ_id_mod_qualified_cert_88       1 3 6 1 5 5 7 0 10 */
4710 279,    /* OBJ_id_mod_qualified_cert_93       1 3 6 1 5 5 7 0 11 */
4711 280,    /* OBJ_id_mod_attribute_cert          1 3 6 1 5 5 7 0 12 */
4712 281,    /* OBJ_id_mod_timestamp_protocol      1 3 6 1 5 5 7 0 13 */
4713 282,    /* OBJ_id_mod_ocsp                    1 3 6 1 5 5 7 0 14 */
4714 283,    /* OBJ_id_mod_dvcs                    1 3 6 1 5 5 7 0 15 */
4715 284,    /* OBJ_id_mod_cmp2000                 1 3 6 1 5 5 7 0 16 */
4716 177,    /* OBJ_info_access                    1 3 6 1 5 5 7 1 1 */
4717 285,    /* OBJ_biometricInfo                  1 3 6 1 5 5 7 1 2 */
4718 286,    /* OBJ_qcStatements                   1 3 6 1 5 5 7 1 3 */
4719 287,    /* OBJ_ac_auditEntity                 1 3 6 1 5 5 7 1 4 */
4720 288,    /* OBJ_ac_targeting                   1 3 6 1 5 5 7 1 5 */
4721 289,    /* OBJ_aaControls                     1 3 6 1 5 5 7 1 6 */
4722 290,    /* OBJ_sbgp_ipAddrBlock               1 3 6 1 5 5 7 1 7 */
4723 291,    /* OBJ_sbgp_autonomousSysNum          1 3 6 1 5 5 7 1 8 */
4724 292,    /* OBJ_sbgp_routerIdentifier          1 3 6 1 5 5 7 1 9 */
4725 397,    /* OBJ_ac_proxying                    1 3 6 1 5 5 7 1 10 */
4726 398,    /* OBJ_sinfo_access                   1 3 6 1 5 5 7 1 11 */
4727 663,    /* OBJ_proxyCertInfo                  1 3 6 1 5 5 7 1 14 */
4728 164,    /* OBJ_id_qt_cps                      1 3 6 1 5 5 7 2 1 */
4729 165,    /* OBJ_id_qt_unotice                  1 3 6 1 5 5 7 2 2 */
4730 293,    /* OBJ_textNotice                     1 3 6 1 5 5 7 2 3 */
4731 129,    /* OBJ_server_auth                    1 3 6 1 5 5 7 3 1 */
4732 130,    /* OBJ_client_auth                    1 3 6 1 5 5 7 3 2 */
4733 131,    /* OBJ_code_sign                      1 3 6 1 5 5 7 3 3 */
4734 132,    /* OBJ_email_protect                  1 3 6 1 5 5 7 3 4 */
4735 294,    /* OBJ_ipsecEndSystem                 1 3 6 1 5 5 7 3 5 */
4736 295,    /* OBJ_ipsecTunnel                    1 3 6 1 5 5 7 3 6 */
4737 296,    /* OBJ_ipsecUser                      1 3 6 1 5 5 7 3 7 */
4738 133,    /* OBJ_time_stamp                     1 3 6 1 5 5 7 3 8 */
4739 180,    /* OBJ_OCSP_sign                      1 3 6 1 5 5 7 3 9 */
4740 297,    /* OBJ_dvcs                           1 3 6 1 5 5 7 3 10 */
4741 298,    /* OBJ_id_it_caProtEncCert            1 3 6 1 5 5 7 4 1 */
4742 299,    /* OBJ_id_it_signKeyPairTypes         1 3 6 1 5 5 7 4 2 */
4743 300,    /* OBJ_id_it_encKeyPairTypes          1 3 6 1 5 5 7 4 3 */
4744 301,    /* OBJ_id_it_preferredSymmAlg         1 3 6 1 5 5 7 4 4 */
4745 302,    /* OBJ_id_it_caKeyUpdateInfo          1 3 6 1 5 5 7 4 5 */
4746 303,    /* OBJ_id_it_currentCRL               1 3 6 1 5 5 7 4 6 */
4747 304,    /* OBJ_id_it_unsupportedOIDs          1 3 6 1 5 5 7 4 7 */
```

```
4748 305,    /* OBJ_id_it_subscriptionRequest    1 3 6 1 5 5 7 4 8 */
4749 306,    /* OBJ_id_it_subscriptionResponse   1 3 6 1 5 5 7 4 9 */
4750 307,    /* OBJ_id_it_keyPairParamReq        1 3 6 1 5 5 7 4 10 */
4751 308,    /* OBJ_id_it_keyPairParamRep        1 3 6 1 5 5 7 4 11 */
4752 309,    /* OBJ_id_it_revPassphrase          1 3 6 1 5 5 7 4 12 */
4753 310,    /* OBJ_id_it_implicitConfirm        1 3 6 1 5 5 7 4 13 */
4754 311,    /* OBJ_id_it_confirmWaitTime        1 3 6 1 5 5 7 4 14 */
4755 312,    /* OBJ_id_it_origPKIMessage         1 3 6 1 5 5 7 4 15 */
4756 784,    /* OBJ_id_it_suppLangTags           1 3 6 1 5 5 7 4 16 */
4757 313,    /* OBJ_id_regCtrl                   1 3 6 1 5 5 7 5 1 */
4758 314,    /* OBJ_id_regInfo                   1 3 6 1 5 5 7 5 2 */
4759 323,    /* OBJ_id_alg_des40                 1 3 6 1 5 5 7 6 1 */
4760 324,    /* OBJ_id_alg_noSignature           1 3 6 1 5 5 7 6 2 */
4761 325,    /* OBJ_id_alg_dh_sig_hmac_sha1      1 3 6 1 5 5 7 6 3 */
4762 326,    /* OBJ_id_alg_dh_pop                1 3 6 1 5 5 7 6 4 */
4763 327,    /* OBJ_id_cmc_statusInfo            1 3 6 1 5 5 7 7 1 */
4764 328,    /* OBJ_id_cmc_identification        1 3 6 1 5 5 7 7 2 */
4765 329,    /* OBJ_id_cmc_identityProof         1 3 6 1 5 5 7 7 3 */
4766 330,    /* OBJ_id_cmc_dataReturn            1 3 6 1 5 5 7 7 4 */
4767 331,    /* OBJ_id_cmc_transactionId         1 3 6 1 5 5 7 7 5 */
4768 332,    /* OBJ_id_cmc_senderNonce           1 3 6 1 5 5 7 7 6 */
4769 333,    /* OBJ_id_cmc_recipientNonce        1 3 6 1 5 5 7 7 7 */
4770 334,    /* OBJ_id_cmc_addExtensions         1 3 6 1 5 5 7 7 8 */
4771 335,    /* OBJ_id_cmc_encryptedPOP          1 3 6 1 5 5 7 7 9 */
4772 336,    /* OBJ_id_cmc_decryptedPOP          1 3 6 1 5 5 7 7 10 */
4773 337,    /* OBJ_id_cmc_lraPOPWitness         1 3 6 1 5 5 7 7 11 */
4774 338,    /* OBJ_id_cmc_getCert               1 3 6 1 5 5 7 7 15 */
4775 339,    /* OBJ_id_cmc_getCRL                1 3 6 1 5 5 7 7 16 */
4776 340,    /* OBJ_id_cmc_revokeRequest         1 3 6 1 5 5 7 7 17 */
4777 341,    /* OBJ_id_cmc_regInfo               1 3 6 1 5 5 7 7 18 */
4778 342,    /* OBJ_id_cmc_responseInfo          1 3 6 1 5 5 7 7 19 */
4779 343,    /* OBJ_id_cmc_queryPending          1 3 6 1 5 5 7 7 21 */
4780 344,    /* OBJ_id_cmc_popLinkRandom         1 3 6 1 5 5 7 7 22 */
4781 345,    /* OBJ_id_cmc_popLinkWitness        1 3 6 1 5 5 7 7 23 */
4782 346,    /* OBJ_id_cmc_confirmCertAcceptance 1 3 6 1 5 5 7 7 24 */
4783 347,    /* OBJ_id_on_personalData           1 3 6 1 5 5 7 8 1 */
4784 858,    /* OBJ_id_on_permanentIdentifier    1 3 6 1 5 5 7 8 3 */
4785 348,    /* OBJ_id_pda_dateOfBirth           1 3 6 1 5 5 7 9 1 */
4786 349,    /* OBJ_id_pda_placeOfBirth          1 3 6 1 5 5 7 9 2 */
4787 351,    /* OBJ_id_pda_gender                1 3 6 1 5 5 7 9 3 */
4788 352,    /* OBJ_id_pda_countryOfCitizenship  1 3 6 1 5 5 7 9 4 */
4789 353,    /* OBJ_id_pda_countryOfResidence    1 3 6 1 5 5 7 9 5 */
4790 354,    /* OBJ_id_aca_authenticationInfo    1 3 6 1 5 5 7 10 1 */
4791 355,    /* OBJ_id_aca_accessIdentity        1 3 6 1 5 5 7 10 2 */
4792 356,    /* OBJ_id_aca_chargingIdentity      1 3 6 1 5 5 7 10 3 */
4793 357,    /* OBJ_id_aca_group                 1 3 6 1 5 5 7 10 4 */
4794 358,    /* OBJ_id_aca_role                  1 3 6 1 5 5 7 10 5 */
4795 399,    /* OBJ_id_aca_encAttrs              1 3 6 1 5 5 7 10 6 */
4796 359,    /* OBJ_id_qcs_pkixQCSyntax_v1       1 3 6 1 5 5 7 11 1 */
4797 360,    /* OBJ_id_cct_crs                   1 3 6 1 5 5 7 12 1 */
4798 361,    /* OBJ_id_cct_PKIData               1 3 6 1 5 5 7 12 2 */
4799 362,    /* OBJ_id_cct_PKIResponse           1 3 6 1 5 5 7 12 3 */
4800 664,    /* OBJ_id_ppl_anyLanguage           1 3 6 1 5 5 7 21 0 */
4801 665,    /* OBJ_id_ppl_inheritAll            1 3 6 1 5 5 7 21 1 */
4802 667,    /* OBJ_Independent                  1 3 6 1 5 5 7 21 2 */
4803 178,    /* OBJ_ad_OCSP                      1 3 6 1 5 5 7 48 1 */
4804 179,    /* OBJ_ad_ca_issuers                1 3 6 1 5 5 7 48 2 */
4805 363,    /* OBJ_ad_timeStamping              1 3 6 1 5 5 7 48 3 */
4806 364,    /* OBJ_ad_dvcs                      1 3 6 1 5 5 7 48 4 */
4807 785,    /* OBJ_caRepository                 1 3 6 1 5 5 7 48 5 */
4808 780,    /* OBJ_hmac_md5                     1 3 6 1 5 5 8 1 1 */
4809 781,    /* OBJ_hmac_sha1                    1 3 6 1 5 5 8 1 2 */
4810 58,     /* OBJ_netscape_cert_extension      2 16 840 1 113730 1 */
4811 59,     /* OBJ_netscape_data_type           2 16 840 1 113730 2 */
4812 438,    /* OBJ_pilotAttributeType           0 9 2342 19200300 100 1 */
4813 439,    /* OBJ_pilotAttributeSyntax         0 9 2342 19200300 100 3 */
```



```
4814 440,    /* OBJ_pilotObjectClass             0 9 2342 19200300 100 4 */
4815 441,    /* OBJ_pilotGroups                  0 9 2342 19200300 100 10 */
4816 108,    /* OBJ_cast5_cbc                    1 2 840 113533 7 66 10 */
4817 112,    /* OBJ_pbeWithMD5AndCast5_CBC       1 2 840 113533 7 66 12 */
4818 782,    /* OBJ_id_PasswordBasedMAC          1 2 840 113533 7 66 13 */
4819 783,    /* OBJ_id_DHBasedMac                1 2 840 113533 7 66 30 */
4820  6,     /* OBJ_rsaEncryption                1 2 840 113549 1 1 1 */
4821  7,     /* OBJ_md2WithRSAEncryption         1 2 840 113549 1 1 2 */
4822 396,    /* OBJ_md4WithRSAEncryption         1 2 840 113549 1 1 3 */
4823  8,     /* OBJ_md5WithRSAEncryption         1 2 840 113549 1 1 4 */
4824 65,     /* OBJ_sha1WithRSAEncryption        1 2 840 113549 1 1 5 */
4825 644,    /* OBJ_rsaOAEPEncryptionSET         1 2 840 113549 1 1 6 */
4826 919,    /* OBJ_rsaesOaep                    1 2 840 113549 1 1 7 */
4827 911,    /* OBJ_mgf1                         1 2 840 113549 1 1 8 */
4828 912,    /* OBJ_rsassaPss                    1 2 840 113549 1 1 10 */
4829 668,    /* OBJ_sha256WithRSAEncryption      1 2 840 113549 1 1 11 */
4830 669,    /* OBJ_sha384WithRSAEncryption      1 2 840 113549 1 1 12 */
4831 670,    /* OBJ_sha512WithRSAEncryption      1 2 840 113549 1 1 13 */
4832 671,    /* OBJ_sha224WithRSAEncryption      1 2 840 113549 1 1 14 */
4833 28,     /* OBJ_dhKeyAgreement               1 2 840 113549 1 3 1 */
4834  9,     /* OBJ_pbeWithMD2AndDES_CBC         1 2 840 113549 1 5 1 */
4835 10,     /* OBJ_pbeWithMD5AndDES_CBC         1 2 840 113549 1 5 3 */
4836 168,    /* OBJ_pbeWithMD2AndRC2_CBC         1 2 840 113549 1 5 4 */
4837 169,    /* OBJ_pbeWithMD5AndRC2_CBC         1 2 840 113549 1 5 6 */
4838 170,    /* OBJ_pbeWithSHA1AndDES_CBC        1 2 840 113549 1 5 10 */
4839 68,     /* OBJ_pbeWithSHA1AndRC2_CBC        1 2 840 113549 1 5 11 */
4840 69,     /* OBJ_id_pbkdf2                    1 2 840 113549 1 5 12 */
4841 161,    /* OBJ_pbes2                        1 2 840 113549 1 5 13 */
4842 162,    /* OBJ_pbmac1                       1 2 840 113549 1 5 14 */
4843 21,     /* OBJ_pkcs7_data                   1 2 840 113549 1 7 1 */
4844 22,     /* OBJ_pkcs7_signed                 1 2 840 113549 1 7 2 */
4845 23,     /* OBJ_pkcs7_enveloped              1 2 840 113549 1 7 3 */
4846 24,     /* OBJ_pkcs7_signedAndEnveloped     1 2 840 113549 1 7 4 */
4847 25,     /* OBJ_pkcs7_digest                 1 2 840 113549 1 7 5 */
4848 26,     /* OBJ_pkcs7_encrypted              1 2 840 113549 1 7 6 */
4849 48,     /* OBJ_pkcs9_emailAddress           1 2 840 113549 1 9 1 */
4850 49,     /* OBJ_pkcs9_unstructuredName       1 2 840 113549 1 9 2 */
4851 50,     /* OBJ_pkcs9_contentType            1 2 840 113549 1 9 3 */
4852 51,     /* OBJ_pkcs9_messageDigest          1 2 840 113549 1 9 4 */
4853 52,     /* OBJ_pkcs9_signingTime            1 2 840 113549 1 9 5 */
4854 53,     /* OBJ_pkcs9_countersignature       1 2 840 113549 1 9 6 */
4855 54,     /* OBJ_pkcs9_challengePassword      1 2 840 113549 1 9 7 */
4856 55,     /* OBJ_pkcs9_unstructuredAddress    1 2 840 113549 1 9 8 */
4857 56,     /* OBJ_pkcs9_extCertAttributes      1 2 840 113549 1 9 9 */
4858 172,    /* OBJ_ext_req                      1 2 840 113549 1 9 14 */
4859 167,    /* OBJ_SMIMECapabilities            1 2 840 113549 1 9 15 */
4860 188,    /* OBJ_SMIME                        1 2 840 113549 1 9 16 */
4861 156,    /* OBJ_friendlyName                 1 2 840 113549 1 9 20 */
4862 157,    /* OBJ_localKeyID                   1 2 840 113549 1 9 21 */
4863 681,    /* OBJ_X9_62_onBasis                1 2 840 10045 1 2 3 1 */
4864 682,    /* OBJ_X9_62_tpBasis                1 2 840 10045 1 2 3 2 */
4865 683,    /* OBJ_X9_62_ppBasis                1 2 840 10045 1 2 3 3 */
4866 417,    /* OBJ_ms_csp_name                  1 3 6 1 4 1 311 17 1 */
4867 856,    /* OBJ_LocalKeySet                  1 3 6 1 4 1 311 17 2 */
4868 390,    /* OBJ_dcObject                     1 3 6 1 4 1 1466 344 */
4869 91,     /* OBJ_bf_cbc                       1 3 6 1 4 1 3029 1 2 */
4870 315,    /* OBJ_id_regCtrl_regToken          1 3 6 1 5 5 7 5 1 1 */
4871 316,    /* OBJ_id_regCtrl_authenticator     1 3 6 1 5 5 7 5 1 2 */
4872 317,    /* OBJ_id_regCtrl_pkiPublicationInfo 1 3 6 1 5 5 7 5 1 3 */
4873 318,    /* OBJ_id_regCtrl_pkiArchiveOptions 1 3 6 1 5 5 7 5 1 4 */
4874 319,    /* OBJ_id_regCtrl_oldCertID         1 3 6 1 5 5 7 5 1 5 */
4875 320,    /* OBJ_id_regCtrl_protocolEncrKey   1 3 6 1 5 5 7 5 1 6 */
4876 321,    /* OBJ_id_regInfo_utf8Pairs         1 3 6 1 5 5 7 5 2 1 */
4877 322,    /* OBJ_id_regInfo_certReq           1 3 6 1 5 5 7 5 2 2 */
4878 365,    /* OBJ_id_pkix_OCSP_basic           1 3 6 1 5 5 7 48 1 1 */
4879 366,    /* OBJ_id_pkix_OCSP_Nonce           1 3 6 1 5 5 7 48 1 2 */
```

```
4880 367,    /* OBJ_id_pkix_OCSP_CrlID                1 3 6 1 5 5 7 48 1 3 */
4881 368,    /* OBJ_id_pkix_OCSP_acceptableResponses 1 3 6 1 5 5 7 48 1 4 */
4882 369,    /* OBJ_id_pkix_OCSP_noCheck              1 3 6 1 5 5 7 48 1 5 */
4883 370,    /* OBJ_id_pkix_OCSP_archiveCutoff        1 3 6 1 5 5 7 48 1 6 */
4884 371,    /* OBJ_id_pkix_OCSP_serviceLocator       1 3 6 1 5 5 7 48 1 7 */
4885 372,    /* OBJ_id_pkix_OCSP_extendedStatus       1 3 6 1 5 5 7 48 1 8 */
4886 373,    /* OBJ_id_pkix_OCSP_valid                1 3 6 1 5 5 7 48 1 9 */
4887 374,    /* OBJ_id_pkix_OCSP_path                 1 3 6 1 5 5 7 48 1 10 */
4888 375,    /* OBJ_id_pkix_OCSP_trustRoot            1 3 6 1 5 5 7 48 1 11 */
4889 418,    /* OBJ_aes_128_ecb                       2 16 840 1 101 3 4 1 1 */
4890 419,    /* OBJ_aes_128_cbc                       2 16 840 1 101 3 4 1 2 */
4891 420,    /* OBJ_aes_128_ofb128                    2 16 840 1 101 3 4 1 3 */
4892 421,    /* OBJ_aes_128_cfb128                    2 16 840 1 101 3 4 1 4 */
4893 788,    /* OBJ_id_aes128_wrap                    2 16 840 1 101 3 4 1 5 */
4894 895,    /* OBJ_aes_128_gcm                       2 16 840 1 101 3 4 1 6 */
4895 896,    /* OBJ_aes_128_ccm                       2 16 840 1 101 3 4 1 7 */
4896 897,    /* OBJ_id_aes128_wrap_pad                2 16 840 1 101 3 4 1 8 */
4897 422,    /* OBJ_aes_192_ecb                       2 16 840 1 101 3 4 1 21 */
4898 423,    /* OBJ_aes_192_cbc                       2 16 840 1 101 3 4 1 22 */
4899 424,    /* OBJ_aes_192_ofb128                    2 16 840 1 101 3 4 1 23 */
4900 425,    /* OBJ_aes_192_cfb128                    2 16 840 1 101 3 4 1 24 */
4901 789,    /* OBJ_id_aes192_wrap                    2 16 840 1 101 3 4 1 25 */
4902 898,    /* OBJ_aes_192_gcm                       2 16 840 1 101 3 4 1 26 */
4903 899,    /* OBJ_aes_192_ccm                       2 16 840 1 101 3 4 1 27 */
4904 900,    /* OBJ_id_aes192_wrap_pad                2 16 840 1 101 3 4 1 28 */
4905 426,    /* OBJ_aes_256_ecb                       2 16 840 1 101 3 4 1 41 */
4906 427,    /* OBJ_aes_256_cbc                       2 16 840 1 101 3 4 1 42 */
4907 428,    /* OBJ_aes_256_ofb128                    2 16 840 1 101 3 4 1 43 */
4908 429,    /* OBJ_aes_256_cfb128                    2 16 840 1 101 3 4 1 44 */
4909 790,    /* OBJ_id_aes256_wrap                    2 16 840 1 101 3 4 1 45 */
4910 901,    /* OBJ_aes_256_gcm                       2 16 840 1 101 3 4 1 46 */
4911 902,    /* OBJ_aes_256_ccm                       2 16 840 1 101 3 4 1 47 */
4912 903,    /* OBJ_id_aes256_wrap_pad                2 16 840 1 101 3 4 1 48 */
4913 672,    /* OBJ_sha256                            2 16 840 1 101 3 4 2 1 */
4914 673,    /* OBJ_sha384                            2 16 840 1 101 3 4 2 2 */
4915 674,    /* OBJ_sha512                            2 16 840 1 101 3 4 2 3 */
4916 675,    /* OBJ_sha224                            2 16 840 1 101 3 4 2 4 */
4917 802,    /* OBJ_dsa_with_SHA224                   2 16 840 1 101 3 4 3 1 */
4918 803,    /* OBJ_dsa_with_SHA256                   2 16 840 1 101 3 4 3 2 */
4919 71,     /* OBJ_netscape_cert_type                2 16 840 1 113730 1 1 */
4920 72,     /* OBJ_netscape_base_url                 2 16 840 1 113730 1 2 */
4921 73,     /* OBJ_netscape_revocation_url           2 16 840 1 113730 1 3 */
4922 74,     /* OBJ_netscape_ca_revocation_url        2 16 840 1 113730 1 4 */
4923 75,     /* OBJ_netscape_renewal_url              2 16 840 1 113730 1 7 */
4924 76,     /* OBJ_netscape_ca_policy_url            2 16 840 1 113730 1 8 */
4925 77,     /* OBJ_netscape_ssl_server_name          2 16 840 1 113730 1 12 */
4926 78,     /* OBJ_netscape_comment                  2 16 840 1 113730 1 13 */
4927 79,     /* OBJ_netscape_cert_sequence            2 16 840 1 113730 2 5 */
4928 139,    /* OBJ_ns_sgc                            2 16 840 1 113730 4 1 */
4929 458,    /* OBJ_userId                            0 9 2342 19200300 100 1 1 */
4930 459,    /* OBJ_textEncodedORAddress              0 9 2342 19200300 100 1 2 */
4931 460,    /* OBJ_rfc822Mailbox                     0 9 2342 19200300 100 1 3 */
4932 461,    /* OBJ_info                              0 9 2342 19200300 100 1 4 */
4933 462,    /* OBJ_favouriteDrink                    0 9 2342 19200300 100 1 5 */
4934 463,    /* OBJ_roomNumber                        0 9 2342 19200300 100 1 6 */
4935 464,    /* OBJ_photo                             0 9 2342 19200300 100 1 7 */
4936 465,    /* OBJ_userClass                         0 9 2342 19200300 100 1 8 */
4937 466,    /* OBJ_host                              0 9 2342 19200300 100 1 9 */
4938 467,    /* OBJ_manager                           0 9 2342 19200300 100 1 10 */
4939 468,    /* OBJ_documentIdentifier                0 9 2342 19200300 100 1 11 */
4940 469,    /* OBJ_documentTitle                     0 9 2342 19200300 100 1 12 */
4941 470,    /* OBJ_documentVersion                   0 9 2342 19200300 100 1 13 */
4942 471,    /* OBJ_documentAuthor                    0 9 2342 19200300 100 1 14 */
4943 472,    /* OBJ_documentLocation                  0 9 2342 19200300 100 1 15 */
4944 473,    /* OBJ_homeTelephoneNumber               0 9 2342 19200300 100 1 20 */
4945 474,    /* OBJ_secretary                         0 9 2342 19200300 100 1 21 */
```



```
4946 475,    /* OBJ_otherMailbox                      0 9 2342 19200300 100 1 22 */
4947 476,    /* OBJ_lastModifiedTime                  0 9 2342 19200300 100 1 23 */
4948 477,    /* OBJ_lastModifiedBy                    0 9 2342 19200300 100 1 24 */
4949 391,    /* OBJ_domainComponent                   0 9 2342 19200300 100 1 25 */
4950 478,    /* OBJ_aRecord                           0 9 2342 19200300 100 1 26 */
4951 479,    /* OBJ_pilotAttributeType27              0 9 2342 19200300 100 1 27 */
4952 480,    /* OBJ_mXRecord                          0 9 2342 19200300 100 1 28 */
4953 481,    /* OBJ_nSRecord                          0 9 2342 19200300 100 1 29 */
4954 482,    /* OBJ_sOARecord                         0 9 2342 19200300 100 1 30 */
4955 483,    /* OBJ_cNAMERecord                       0 9 2342 19200300 100 1 31 */
4956 484,    /* OBJ_associatedDomain                  0 9 2342 19200300 100 1 37 */
4957 485,    /* OBJ_associatedName                    0 9 2342 19200300 100 1 38 */
4958 486,    /* OBJ_homePostalAddress                 0 9 2342 19200300 100 1 39 */
4959 487,    /* OBJ_personalTitle                     0 9 2342 19200300 100 1 40 */
4960 488,    /* OBJ_mobileTelephoneNumber             0 9 2342 19200300 100 1 41 */
4961 489,    /* OBJ_pagerTelephoneNumber              0 9 2342 19200300 100 1 42 */
4962 490,    /* OBJ_friendlyCountryName               0 9 2342 19200300 100 1 43 */
4963 491,    /* OBJ_organizationalStatus              0 9 2342 19200300 100 1 45 */
4964 492,    /* OBJ_janetMailbox                      0 9 2342 19200300 100 1 46 */
4965 493,    /* OBJ_mailPreferenceOption              0 9 2342 19200300 100 1 47 */
4966 494,    /* OBJ_buildingName                      0 9 2342 19200300 100 1 48 */
4967 495,    /* OBJ_dSAQuality                        0 9 2342 19200300 100 1 49 */
4968 496,    /* OBJ_singleLevelQuality                0 9 2342 19200300 100 1 50 */
4969 497,    /* OBJ_subtreeMinimumQuality             0 9 2342 19200300 100 1 51 */
4970 498,    /* OBJ_subtreeMaximumQuality             0 9 2342 19200300 100 1 52 */
4971 499,    /* OBJ_personalSignature                 0 9 2342 19200300 100 1 53 */
4972 500,    /* OBJ_dITRedirect                       0 9 2342 19200300 100 1 54 */
4973 501,    /* OBJ_audio                             0 9 2342 19200300 100 1 55 */
4974 502,    /* OBJ_documentPublisher                 0 9 2342 19200300 100 1 56 */
4975 442,    /* OBJ_iA5StringSyntax                   0 9 2342 19200300 100 3 4 */
4976 443,    /* OBJ_caseIgnoreIA5StringSyntax         0 9 2342 19200300 100 3 5 */
4977 444,    /* OBJ_pilotObject                       0 9 2342 19200300 100 4 3 */
4978 445,    /* OBJ_pilotPerson                       0 9 2342 19200300 100 4 4 */
4979 446,    /* OBJ_account                           0 9 2342 19200300 100 4 5 */
4980 447,    /* OBJ_document                          0 9 2342 19200300 100 4 6 */
4981 448,    /* OBJ_room                              0 9 2342 19200300 100 4 7 */
4982 449,    /* OBJ_documentSeries                    0 9 2342 19200300 100 4 9 */
4983 392,    /* OBJ_Domain                            0 9 2342 19200300 100 4 13 */
4984 450,    /* OBJ_rFC822localPart                   0 9 2342 19200300 100 4 14 */
4985 451,    /* OBJ_dNSDomain                         0 9 2342 19200300 100 4 15 */
4986 452,    /* OBJ_domainRelatedObject               0 9 2342 19200300 100 4 17 */
4987 453,    /* OBJ_friendlyCountry                   0 9 2342 19200300 100 4 18 */
4988 454,    /* OBJ_simpleSecurityObject              0 9 2342 19200300 100 4 19 */
4989 455,    /* OBJ_pilotOrganization                 0 9 2342 19200300 100 4 20 */
4990 456,    /* OBJ_pilotDSA                          0 9 2342 19200300 100 4 21 */
4991 457,    /* OBJ_qualityLabelledData               0 9 2342 19200300 100 4 22 */
4992 189,    /* OBJ_id_smime_mod                      1 2 840 113549 1 9 16 0 */
4993 190,    /* OBJ_id_smime_ct                       1 2 840 113549 1 9 16 1 */
4994 191,    /* OBJ_id_smime_aa                       1 2 840 113549 1 9 16 2 */
4995 192,    /* OBJ_id_smime_alg                      1 2 840 113549 1 9 16 3 */
4996 193,    /* OBJ_id_smime_cd                       1 2 840 113549 1 9 16 4 */
4997 194,    /* OBJ_id_smime_spq                      1 2 840 113549 1 9 16 5 */
4998 195,    /* OBJ_id_smime_cti                      1 2 840 113549 1 9 16 6 */
4999 158,    /* OBJ_x509Certificate                   1 2 840 113549 1 9 22 1 */
5000 159,    /* OBJ_sdsiCertificate                   1 2 840 113549 1 9 22 2 */
5001 160,    /* OBJ_x509Crl                           1 2 840 113549 1 9 23 1 */
5002 144,    /* OBJ_pbe_WithSHA1And128BitRC4          1 2 840 113549 1 12 1 1 */
5003 145,    /* OBJ_pbe_WithSHA1And40BitRC4           1 2 840 113549 1 12 1 2 */
5004 146,    /* OBJ_pbe_WithSHA1And3_Key_TripleDES_CBC 1 2 840 113549 1 12 1 3 */
5005 147,    /* OBJ_pbe_WithSHA1And2_Key_TripleDES_CBC 1 2 840 113549 1 12 1 4 */
5006 148,    /* OBJ_pbe_WithSHA1And128BitRC2_CBC 1 2 840 113549 1 12 1 5 */
5007 149,    /* OBJ_pbe_WithSHA1And40BitRC2_CBC  1 2 840 113549 1 12 1 6 */
5008 171,    /* OBJ_ms_ext_req                        1 3 6 1 4 1 311 2 1 14 */
5009 134,    /* OBJ_ms_code_ind                       1 3 6 1 4 1 311 2 1 21 */
5010 135,    /* OBJ_ms_code_com                       1 3 6 1 4 1 311 2 1 22 */
5011 136,    /* OBJ_ms_ctl_sign                       1 3 6 1 4 1 311 10 3 1 */
```

```
5012 137,    /* OBJ_ms_sgc                      1 3 6 1 4 1 311 10 3 3 */
5013 138,    /* OBJ_ms_efs                      1 3 6 1 4 1 311 10 3 4 */
5014 648,    /* OBJ_ms_smartcard_login          1 3 6 1 4 1 311 20 2 2 */
5015 649,    /* OBJ_ms_upn                      1 3 6 1 4 1 311 20 2 3 */
5016 751,    /* OBJ_camellia_128_cbc            1 2 392 200011 61 1 1 1 2 */
5017 752,    /* OBJ_camellia_192_cbc            1 2 392 200011 61 1 1 1 3 */
5018 753,    /* OBJ_camellia_256_cbc            1 2 392 200011 61 1 1 1 4 */
5019 907,    /* OBJ_id_camellia128_wrap         1 2 392 200011 61 1 1 3 2 */
5020 908,    /* OBJ_id_camellia192_wrap         1 2 392 200011 61 1 1 3 3 */
5021 909,    /* OBJ_id_camellia256_wrap         1 2 392 200011 61 1 1 3 4 */
5022 196,    /* OBJ_id_smime_mod_cms            1 2 840 113549 1 9 16 0 1 */
5023 197,    /* OBJ_id_smime_mod_ess            1 2 840 113549 1 9 16 0 2 */
5024 198,    /* OBJ_id_smime_mod_oid            1 2 840 113549 1 9 16 0 3 */
5025 199,    /* OBJ_id_smime_mod_msg_v3         1 2 840 113549 1 9 16 0 4 */
5026 200,    /* OBJ_id_smime_mod_ets_eSignature_88 1 2 840 113549 1 9 16 0 5 */
5027 201,    /* OBJ_id_smime_mod_ets_eSignature_97 1 2 840 113549 1 9 16 0 6 */
5028 202,    /* OBJ_id_smime_mod_ets_eSigPolicy_88 1 2 840 113549 1 9 16 0 7 */
5029 203,    /* OBJ_id_smime_mod_ets_eSigPolicy_97 1 2 840 113549 1 9 16 0 8 */
5030 204,    /* OBJ_id_smime_ct_receipt         1 2 840 113549 1 9 16 1 1 */
5031 205,    /* OBJ_id_smime_ct_authData        1 2 840 113549 1 9 16 1 2 */
5032 206,    /* OBJ_id_smime_ct_publishCert     1 2 840 113549 1 9 16 1 3 */
5033 207,    /* OBJ_id_smime_ct_TSTInfo         1 2 840 113549 1 9 16 1 4 */
5034 208,    /* OBJ_id_smime_ct_TDTInfo         1 2 840 113549 1 9 16 1 5 */
5035 209,    /* OBJ_id_smime_ct_contentInfo     1 2 840 113549 1 9 16 1 6 */
5036 210,    /* OBJ_id_smime_ct_DVCSRequestData 1 2 840 113549 1 9 16 1 7 */
5037 211,    /* OBJ_id_smime_ct_DVCSResponseData 1 2 840 113549 1 9 16 1 8 */
5038 786,    /* OBJ_id_smime_ct_compressedData  1 2 840 113549 1 9 16 1 9 */
5039 787,    /* OBJ_id_ct_asciiTextWithCRLF     1 2 840 113549 1 9 16 1 27 */
5040 212,    /* OBJ_id_smime_aa_receiptRequest  1 2 840 113549 1 9 16 2 1 */
5041 213,    /* OBJ_id_smime_aa_securityLabel   1 2 840 113549 1 9 16 2 2 */
5042 214,    /* OBJ_id_smime_aa_mlExpandHistory 1 2 840 113549 1 9 16 2 3 */
5043 215,    /* OBJ_id_smime_aa_contentHint     1 2 840 113549 1 9 16 2 4 */
5044 216,    /* OBJ_id_smime_aa_msgSigDigest    1 2 840 113549 1 9 16 2 5 */
5045 217,    /* OBJ_id_smime_aa_encapContentType 1 2 840 113549 1 9 16 2 6 */
5046 218,    /* OBJ_id_smime_aa_contentIdentifier 1 2 840 113549 1 9 16 2 7 */
5047 219,    /* OBJ_id_smime_aa_macValue        1 2 840 113549 1 9 16 2 8 */
5048 220,    /* OBJ_id_smime_aa_equivalentLabels 1 2 840 113549 1 9 16 2 9 */
5049 221,    /* OBJ_id_smime_aa_contentReference 1 2 840 113549 1 9 16 2 10 */
5050 222,    /* OBJ_id_smime_aa_encrypKeyPref   1 2 840 113549 1 9 16 2 11 */
5051 223,    /* OBJ_id_smime_aa_signingCertificate 1 2 840 113549 1 9 16 2 12 */
5052 224,    /* OBJ_id_smime_aa_smimeEncryptCerts 1 2 840 113549 1 9 16 2 13 */
5053 225,    /* OBJ_id_smime_aa_timeStampToken  1 2 840 113549 1 9 16 2 14 */
5054 226,    /* OBJ_id_smime_aa_ets_sigPolicyId 1 2 840 113549 1 9 16 2 15 */
5055 227,    /* OBJ_id_smime_aa_ets_commitmentType 1 2 840 113549 1 9 16 2 16 */
5056 228,    /* OBJ_id_smime_aa_ets_signerLocation 1 2 840 113549 1 9 16 2 17 */
5057 229,    /* OBJ_id_smime_aa_ets_signerAttr  1 2 840 113549 1 9 16 2 18 */
5058 230,    /* OBJ_id_smime_aa_ets_otherSigCert 1 2 840 113549 1 9 16 2 19 */
5059 231,    /* OBJ_id_smime_aa_ets_contentTimestamp 1 2 840 113549 1 9 16 2 20 */
5060 232,    /* OBJ_id_smime_aa_ets_CertificateRefs 1 2 840 113549 1 9 16 2 21 */
5061 233,    /* OBJ_id_smime_aa_ets_RevocationRefs 1 2 840 113549 1 9 16 2 22 */
5062 234,    /* OBJ_id_smime_aa_ets_certValues  1 2 840 113549 1 9 16 2 23 */
5063 235,    /* OBJ_id_smime_aa_ets_revocationValues 1 2 840 113549 1 9 16 2 24 */
5064 236,    /* OBJ_id_smime_aa_ets_escTimeStamp 1 2 840 113549 1 9 16 2 25 */
5065 237,    /* OBJ_id_smime_aa_ets_certCRLTimestamp 1 2 840 113549 1 9 16 2 26 */
5066 238,    /* OBJ_id_smime_aa_ets_archiveTimeStamp 1 2 840 113549 1 9 16 2 27 */
5067 239,    /* OBJ_id_smime_aa_signatureType   1 2 840 113549 1 9 16 2 28 */
5068 240,    /* OBJ_id_smime_aa_dvcs_dvc        1 2 840 113549 1 9 16 2 29 */
5069 241,    /* OBJ_id_smime_alg_ESDHwith3DES   1 2 840 113549 1 9 16 3 1 */
5070 242,    /* OBJ_id_smime_alg_ESDHwithRC2    1 2 840 113549 1 9 16 3 2 */
5071 243,    /* OBJ_id_smime_alg_3DESwrap       1 2 840 113549 1 9 16 3 3 */
5072 244,    /* OBJ_id_smime_alg_RC2wrap        1 2 840 113549 1 9 16 3 4 */
5073 245,    /* OBJ_id_smime_alg_ESDH           1 2 840 113549 1 9 16 3 5 */
5074 246,    /* OBJ_id_smime_alg_CMS3DESwrap    1 2 840 113549 1 9 16 3 6 */
5075 247,    /* OBJ_id_smime_alg_CMSRC2wrap     1 2 840 113549 1 9 16 3 7 */
5076 125,    /* OBJ_zlib_compression            1 2 840 113549 1 9 16 3 8 */
5077 893,    /* OBJ_id_alg_PWRI_KEK             1 2 840 113549 1 9 16 3 9 */
```



```
5078 248,    /* OBJ_id_smime_cd_ldap            1 2 840 113549 1 9 16 4 1 */
5079 249,    /* OBJ_id_smime_spq_ets_sqt_uri    1 2 840 113549 1 9 16 5 1 */
5080 250,    /* OBJ_id_smime_spq_ets_sqt_unotice 1 2 840 113549 1 9 16 5 2 */
5081 251,    /* OBJ_id_smime_cti_ets_proofOfOrigin 1 2 840 113549 1 9 16 6 1 */
5082 252,    /* OBJ_id_smime_cti_ets_proofOfReceipt 1 2 840 113549 1 9 16 6 2 */
5083 253,    /* OBJ_id_smime_cti_ets_proofOfDelivery 1 2 840 113549 1 9 16 6 3 */
5084 254,    /* OBJ_id_smime_cti_ets_proofOfSender 1 2 840 113549 1 9 16 6 4 */
5085 255,    /* OBJ_id_smime_cti_ets_proofOfApproval 1 2 840 113549 1 9 16 6 5 */
5086 256,    /* OBJ_id_smime_cti_ets_proofOfCreation 1 2 840 113549 1 9 16 6 6 */
5087 150,    /* OBJ_keyBag                      1 2 840 113549 1 12 10 1 1 */
5088 151,    /* OBJ_pkcs8ShroudedKeyBag         1 2 840 113549 1 12 10 1 2 */
5089 152,    /* OBJ_certBag                     1 2 840 113549 1 12 10 1 3 */
5090 153,    /* OBJ_crlBag                      1 2 840 113549 1 12 10 1 4 */
5091 154,    /* OBJ_secretBag                   1 2 840 113549 1 12 10 1 5 */
5092 155,    /* OBJ_safeContentsBag             1 2 840 113549 1 12 10 1 6 */
5093 34,     /* OBJ_idea_cbc                    1 3 6 1 4 1 188 7 1 1 2 */
5094 };
5095 #endif /* ! codereview */
```

```
**********************************************************
    2546 Wed Aug 13 19:51:39 2014
new/usr/src/lib/openssl/include/obj_xref.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* AUTOGENERATED BY objxref.pl, DO NOT EDIT */

typedef struct
        {
        int sign_id;
        int hash_id;
        int pkey_id;
        } nid_triple;

static const nid_triple sigoid_srt[] =
        {
        {NID_md2WithRSAEncryption, NID_md2, NID_rsaEncryption},
        {NID_md5WithRSAEncryption, NID_md5, NID_rsaEncryption},
        {NID_shaWithRSAEncryption, NID_sha, NID_rsaEncryption},
        {NID_sha1WithRSAEncryption, NID_sha1, NID_rsaEncryption},
        {NID_dsaWithSHA, NID_sha, NID_dsa},
        {NID_dsaWithSHA1_2, NID_sha1, NID_dsa_2},
        {NID_mdc2WithRSA, NID_mdc2, NID_rsaEncryption},
        {NID_md5WithRSA, NID_md5, NID_rsa},
        {NID_dsaWithSHA1, NID_sha1, NID_dsa},
        {NID_sha1WithRSA, NID_sha1, NID_rsa},
        {NID_ripemd160WithRSA, NID_ripemd160, NID_rsaEncryption},
        {NID_md4WithRSAEncryption, NID_md4, NID_rsaEncryption},
        {NID_ecdsa_with_SHA1, NID_sha1, NID_X9_62_id_ecPublicKey},
        {NID_sha256WithRSAEncryption, NID_sha256, NID_rsaEncryption},
        {NID_sha384WithRSAEncryption, NID_sha384, NID_rsaEncryption},
        {NID_sha512WithRSAEncryption, NID_sha512, NID_rsaEncryption},
        {NID_sha224WithRSAEncryption, NID_sha224, NID_rsaEncryption},
        {NID_ecdsa_with_Recommended, NID_undef, NID_X9_62_id_ecPublicKey},
        {NID_ecdsa_with_Specified, NID_undef, NID_X9_62_id_ecPublicKey},
        {NID_ecdsa_with_SHA224, NID_sha224, NID_X9_62_id_ecPublicKey},
        {NID_ecdsa_with_SHA256, NID_sha256, NID_X9_62_id_ecPublicKey},
        {NID_ecdsa_with_SHA384, NID_sha384, NID_X9_62_id_ecPublicKey},
        {NID_ecdsa_with_SHA512, NID_sha512, NID_X9_62_id_ecPublicKey},
        {NID_dsa_with_SHA224, NID_sha224, NID_dsa},
        {NID_dsa_with_SHA256, NID_sha256, NID_dsa},
        {NID_id_GostR3411_94_with_GostR3410_2001, NID_id_GostR3411_94, NID_id_Go
        {NID_id_GostR3411_94_with_GostR3410_94, NID_id_GostR3411_94, NID_id_Gost
        {NID_id_GostR3411_94_with_GostR3410_94_cc, NID_id_GostR3411_94, NID_id_G
        {NID_id_GostR3411_94_with_GostR3410_2001_cc, NID_id_GostR3411_94, NID_id
        {NID_rsassaPss, NID_undef, NID_rsaEncryption},
        };

static const nid_triple * const sigoid_srt_xref[] =
        {
        &sigoid_srt[29],
        &sigoid_srt[17],
        &sigoid_srt[18],
        &sigoid_srt[0],
        &sigoid_srt[1],
        &sigoid_srt[7],
        &sigoid_srt[2],
        &sigoid_srt[4],
        &sigoid_srt[3],
        &sigoid_srt[9],
        &sigoid_srt[5],
        &sigoid_srt[8],
        &sigoid_srt[12],
        &sigoid_srt[6],
        &sigoid_srt[10],
        &sigoid_srt[11],
```



```
        &sigoid_srt[13],
        &sigoid_srt[24],
        &sigoid_srt[20],
        &sigoid_srt[14],
        &sigoid_srt[21],
        &sigoid_srt[15],
        &sigoid_srt[22],
        &sigoid_srt[16],
        &sigoid_srt[23],
        &sigoid_srt[19],
        &sigoid_srt[25],
        &sigoid_srt[26],
        &sigoid_srt[27],
        &sigoid_srt[28],
        };
#endif /* ! codereview */
```

```
**********************************************************
    5506 Wed Aug 13 19:51:39 2014
new/usr/src/lib/openssl/include/openssl/aes.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes.h -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#ifndef HEADER_AES_H
#define HEADER_AES_H

#include <openssl/opensslconf.h>

#ifdef OPENSSL_NO_AES
#error AES is disabled.
#endif

#include <stddef.h>
```



```
#define AES_ENCRYPT     1
#define AES_DECRYPT     0

/* Because array size can't be a const in C, the following two are macros.
   Both sizes are in bytes. */
#define AES_MAXNR 14
#define AES_BLOCK_SIZE 16

#ifdef  __cplusplus
extern "C" {
#endif

/* This should be a hidden type, but EVP requires that the size be known */
struct aes_key_st {
#ifdef AES_LONG
    unsigned long rd_key[4 *(AES_MAXNR + 1)];
#else
    unsigned int rd_key[4 *(AES_MAXNR + 1)];
#endif
    int rounds;
};
typedef struct aes_key_st AES_KEY;

const char *AES_options(void);

int AES_set_encrypt_key(const unsigned char *userKey, const int bits,
        AES_KEY *key);
int AES_set_decrypt_key(const unsigned char *userKey, const int bits,
        AES_KEY *key);

int private_AES_set_encrypt_key(const unsigned char *userKey, const int bits,
        AES_KEY *key);
int private_AES_set_decrypt_key(const unsigned char *userKey, const int bits,
        AES_KEY *key);

void AES_encrypt(const unsigned char *in, unsigned char *out,
        const AES_KEY *key);
void AES_decrypt(const unsigned char *in, unsigned char *out,
        const AES_KEY *key);

void AES_ecb_encrypt(const unsigned char *in, unsigned char *out,
        const AES_KEY *key, const int enc);
void AES_cbc_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
        unsigned char *ivec, const int enc);
void AES_cfb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
        unsigned char *ivec, int *num, const int enc);
void AES_cfb1_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
        unsigned char *ivec, int *num, const int enc);
void AES_cfb8_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
        unsigned char *ivec, int *num, const int enc);
void AES_ofb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
        unsigned char *ivec, int *num);
void AES_ctr128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
        unsigned char ivec[AES_BLOCK_SIZE],
        unsigned char ecount_buf[AES_BLOCK_SIZE],
        unsigned int *num);
/* NB: the IV is _two_ blocks long */
void AES_ige_encrypt(const unsigned char *in, unsigned char *out,
                     size_t length, const AES_KEY *key,
```

```
128                     unsigned char *ivec, const int enc);
129 /* NB: the IV is _four_ blocks long */
130 void AES_bi_ige_encrypt(const unsigned char *in, unsigned char *out,
131                         size_t length, const AES_KEY *key,
132                         const AES_KEY *key2, const unsigned char *ivec,
133                         const int enc);

135 int AES_wrap_key(AES_KEY *key, const unsigned char *iv,
136                 unsigned char *out,
137                 const unsigned char *in, unsigned int inlen);
138 int AES_unwrap_key(AES_KEY *key, const unsigned char *iv,
139                 unsigned char *out,
140                 const unsigned char *in, unsigned int inlen);


143 #ifdef  __cplusplus
144 }
145 #endif

147 #endif /* !HEADER_AES_H */
148 #endif /* ! codereview */
```

```
**********************************************************
   52174 Wed Aug 13 19:51:40 2014
new/usr/src/lib/openssl/include/openssl/asn1.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/asn1.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_ASN1_H
#define HEADER_ASN1_H
```



```
#include <time.h>
#include <openssl/e_os2.h>
#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/stack.h>
#include <openssl/safestack.h>

#include <openssl/symhacks.h>

#include <openssl/ossl_typ.h>
#ifndef OPENSSL_NO_DEPRECATED
#include <openssl/bn.h>
#endif

#ifdef OPENSSL_BUILD_SHLIBCRYPTO
# undef OPENSSL_EXTERN
# define OPENSSL_EXTERN OPENSSL_EXPORT
#endif

#ifdef  __cplusplus
extern "C" {
#endif

#define V_ASN1_UNIVERSAL                0x00
#define V_ASN1_APPLICATION              0x40
#define V_ASN1_CONTEXT_SPECIFIC         0x80
#define V_ASN1_PRIVATE                  0xc0

#define V_ASN1_CONSTRUCTED              0x20
#define V_ASN1_PRIMITIVE_TAG            0x1f
#define V_ASN1_PRIMATIVE_TAG            0x1f

#define V_ASN1_APP_CHOOSE               -2      /* let the recipient choose */
#define V_ASN1_OTHER                    -3      /* used in ASN1_TYPE */
#define V_ASN1_ANY                      -4      /* used in ASN1 template code */

#define V_ASN1_NEG                      0x100   /* negative flag */

#define V_ASN1_UNDEF                    -1
#define V_ASN1_EOC                      0
#define V_ASN1_BOOLEAN                  1       /**/
#define V_ASN1_INTEGER                  2
#define V_ASN1_NEG_INTEGER              (2 | V_ASN1_NEG)
#define V_ASN1_BIT_STRING               3
#define V_ASN1_OCTET_STRING             4
#define V_ASN1_NULL                     5
#define V_ASN1_OBJECT                   6
#define V_ASN1_OBJECT_DESCRIPTOR        7
#define V_ASN1_EXTERNAL                 8
#define V_ASN1_REAL                     9
#define V_ASN1_ENUMERATED               10
#define V_ASN1_NEG_ENUMERATED           (10 | V_ASN1_NEG)
#define V_ASN1_UTF8STRING               12
#define V_ASN1_SEQUENCE                 16
#define V_ASN1_SET                      17
#define V_ASN1_NUMERICSTRING            18      /**/
#define V_ASN1_PRINTABLESTRING          19
#define V_ASN1_T61STRING                20
#define V_ASN1_TELETEXSTRING            20      /* alias */
#define V_ASN1_VIDEOTEXSTRING           21      /**/
#define V_ASN1_IA5STRING                22
#define V_ASN1_UTCTIME                  23
#define V_ASN1_GENERALIZEDTIME          24      /**/
#define V_ASN1_GRAPHICSTRING            25      /**/
#define V_ASN1_ISO64STRING              26      /**/
```

```
#define V_ASN1_VISIBLESTRING            26      /* alias */
#define V_ASN1_GENERALSTRING            27      /**/
#define V_ASN1_UNIVERSALSTRING          28      /**/
#define V_ASN1_BMPSTRING                30

/* For use with d2i_ASN1_type_bytes() */
#define B_ASN1_NUMERICSTRING    0x0001
#define B_ASN1_PRINTABLESTRING  0x0002
#define B_ASN1_T61STRING        0x0004
#define B_ASN1_TELETEXSTRING    0x0004
#define B_ASN1_VIDEOTEXSTRING   0x0008
#define B_ASN1_IA5STRING        0x0010
#define B_ASN1_GRAPHICSTRING    0x0020
#define B_ASN1_ISO64STRING      0x0040
#define B_ASN1_VISIBLESTRING    0x0040
#define B_ASN1_GENERALSTRING    0x0080
#define B_ASN1_UNIVERSALSTRING  0x0100
#define B_ASN1_OCTET_STRING     0x0200
#define B_ASN1_BIT_STRING       0x0400
#define B_ASN1_BMPSTRING        0x0800
#define B_ASN1_UNKNOWN          0x1000
#define B_ASN1_UTF8STRING       0x2000
#define B_ASN1_UTCTIME          0x4000
#define B_ASN1_GENERALIZEDTIME  0x8000
#define B_ASN1_SEQUENCE         0x10000

/* For use with ASN1_mbstring_copy() */
#define MBSTRING_FLAG           0x1000
#define MBSTRING_UTF8           (MBSTRING_FLAG)
#define MBSTRING_ASC            (MBSTRING_FLAG|1)
#define MBSTRING_BMP            (MBSTRING_FLAG|2)
#define MBSTRING_UNIV           (MBSTRING_FLAG|4)

#define SMIME_OLDMIME           0x400
#define SMIME_CRLFEOL           0x800
#define SMIME_STREAM            0x1000

struct X509_algor_st;
DECLARE_STACK_OF(X509_ALGOR)

#define DECLARE_ASN1_SET_OF(type) /* filled in by mkstack.pl */
#define IMPLEMENT_ASN1_SET_OF(type) /* nothing, no longer needed */

/* We MUST make sure that, except for constness, asn1_ctx_st and
   asn1_const_ctx are exactly the same.  Fortunately, as soon as
   the old ASN1 parsing macros are gone, we can throw this away
   as well... */
typedef struct asn1_ctx_st
        {
        unsigned char *p;/* work char pointer */
        int eos;        /* end of sequence read for indefinite encoding */
        int error;      /* error code to use when returning an error */
        int inf;        /* constructed if 0x20, indefinite is 0x21 */
        int tag;        /* tag from last 'get object' */
        int xclass;     /* class from last 'get object' */
        long slen;      /* length of last 'get object' */
        unsigned char *max; /* largest value of p allowed */
        unsigned char *q;/* temporary variable */
        unsigned char **pp;/* variable */
        int line;       /* used in error processing */
        } ASN1_CTX;

typedef struct asn1_const_ctx_st
        {
        const unsigned char *p;/* work char pointer */
        int eos;        /* end of sequence read for indefinite encoding */
```



```
        int error;      /* error code to use when returning an error */
        int inf;        /* constructed if 0x20, indefinite is 0x21 */
        int tag;        /* tag from last 'get object' */
        int xclass;     /* class from last 'get object' */
        long slen;      /* length of last 'get object' */
        const unsigned char *max; /* largest value of p allowed */
        const unsigned char *q;/* temporary variable */
        const unsigned char **pp;/* variable */
        int line;       /* used in error processing */
        } ASN1_const_CTX;

/* These are used internally in the ASN1_OBJECT to keep track of
 * whether the names and data need to be free()ed */
#define ASN1_OBJECT_FLAG_DYNAMIC         0x01   /* internal use */
#define ASN1_OBJECT_FLAG_CRITICAL        0x02   /* critical x509v3 object id */
#define ASN1_OBJECT_FLAG_DYNAMIC_STRINGS 0x04   /* internal use */
#define ASN1_OBJECT_FLAG_DYNAMIC_DATA    0x08   /* internal use */
typedef struct asn1_object_st
        {
        const char *sn,*ln;
        int nid;
        int length;
        const unsigned char *data;      /* data remains const after init */
        int flags;      /* Should we free this one */
        } ASN1_OBJECT;

#define ASN1_STRING_FLAG_BITS_LEFT 0x08 /* Set if 0x07 has bits left value */
/* This indicates that the ASN1_STRING is not a real value but just a place
 * holder for the location where indefinite length constructed data should
 * be inserted in the memory buffer
 */
#define ASN1_STRING_FLAG_NDEF 0x010

/* This flag is used by the CMS code to indicate that a string is not
 * complete and is a place holder for content when it had all been
 * accessed. The flag will be reset when content has been written to it.
 */

#define ASN1_STRING_FLAG_CONT 0x020
/* This flag is used by ASN1 code to indicate an ASN1_STRING is an MSTRING
 * type.
 */
#define ASN1_STRING_FLAG_MSTRING 0x040
/* This is the base type that holds just about everything :-) */
struct asn1_string_st
        {
        int length;
        int type;
        unsigned char *data;
        /* The value of the following field depends on the type being
         * held.  It is mostly being used for BIT_STRING so if the
         * input data has a non-zero 'unused bits' value, it will be
         * handled correctly */
        long flags;
        };

/* ASN1_ENCODING structure: this is used to save the received
 * encoding of an ASN1 type. This is useful to get round
 * problems with invalid encodings which can break signatures.
 */

typedef struct ASN1_ENCODING_st
        {
        unsigned char *enc;     /* DER encoding */
        long len;               /* Length of encoding */
        int modified;           /* set to 1 if 'enc' is invalid */
```

```
260         } ASN1_ENCODING;

262 /* Used with ASN1 LONG type: if a long is set to this it is omitted */
263 #define ASN1_LONG_UNDEF 0x7fffffffL

265 #define STABLE_FLAGS_MALLOC     0x01
266 #define STABLE_NO_MASK          0x02
267 #define DIRSTRING_TYPE  \
268  (B_ASN1_PRINTABLESTRING|B_ASN1_T61STRING|B_ASN1_BMPSTRING|B_ASN1_UTF8STRING)
269 #define PKCS9STRING_TYPE (DIRSTRING_TYPE|B_ASN1_IA5STRING)

271 typedef struct asn1_string_table_st {
272         int nid;
273         long minsize;
274         long maxsize;
275         unsigned long mask;
276         unsigned long flags;
277 } ASN1_STRING_TABLE;

279 DECLARE_STACK_OF(ASN1_STRING_TABLE)

281 /* size limits: this stuff is taken straight from RFC2459 */

283 #define ub_name                         32768
284 #define ub_common_name                  64
285 #define ub_locality_name                128
286 #define ub_state_name                   128
287 #define ub_organization_name            64
288 #define ub_organization_unit_name       64
289 #define ub_title                        64
290 #define ub_email_address                128

292 /* Declarations for template structures: for full definitions
293  * see asn1t.h
294  */
295 typedef struct ASN1_TEMPLATE_st ASN1_TEMPLATE;
296 typedef struct ASN1_TLC_st ASN1_TLC;
297 /* This is just an opaque pointer */
298 typedef struct ASN1_VALUE_st ASN1_VALUE;

300 /* Declare ASN1 functions: the implement macro in in asn1t.h */

302 #define DECLARE_ASN1_FUNCTIONS(type) DECLARE_ASN1_FUNCTIONS_name(type, type)

304 #define DECLARE_ASN1_ALLOC_FUNCTIONS(type) \
305         DECLARE_ASN1_ALLOC_FUNCTIONS_name(type, type)

307 #define DECLARE_ASN1_FUNCTIONS_name(type, name) \
308         DECLARE_ASN1_ALLOC_FUNCTIONS_name(type, name) \
309         DECLARE_ASN1_ENCODE_FUNCTIONS(type, name, name)

311 #define DECLARE_ASN1_FUNCTIONS_fname(type, itname, name) \
312         DECLARE_ASN1_ALLOC_FUNCTIONS_name(type, name) \
313         DECLARE_ASN1_ENCODE_FUNCTIONS(type, itname, name)

315 #define DECLARE_ASN1_ENCODE_FUNCTIONS(type, itname, name) \
316         type *d2i_##name(type **a, const unsigned char **in, long len); \
317         int i2d_##name(type *a, unsigned char **out); \
318         DECLARE_ASN1_ITEM(itname)

320 #define DECLARE_ASN1_ENCODE_FUNCTIONS_const(type, name) \
321         type *d2i_##name(type **a, const unsigned char **in, long len); \
322         int i2d_##name(const type *a, unsigned char **out); \
323         DECLARE_ASN1_ITEM(name)

325 #define DECLARE_ASN1_NDEF_FUNCTION(name) \
```



```
326         int i2d_##name##_NDEF(name *a, unsigned char **out);

328 #define DECLARE_ASN1_FUNCTIONS_const(name) \
329         DECLARE_ASN1_ALLOC_FUNCTIONS(name) \
330         DECLARE_ASN1_ENCODE_FUNCTIONS_const(name, name)

332 #define DECLARE_ASN1_ALLOC_FUNCTIONS_name(type, name) \
333         type *name##_new(void); \
334         void name##_free(type *a);

336 #define DECLARE_ASN1_PRINT_FUNCTION(stname) \
337         DECLARE_ASN1_PRINT_FUNCTION_fname(stname, stname)

339 #define DECLARE_ASN1_PRINT_FUNCTION_fname(stname, fname) \
340         int fname##_print_ctx(BIO *out, stname *x, int indent, \
341                                         const ASN1_PCTX *pctx);

343 #define D2I_OF(type) type *(*)(type **,const unsigned char **,long)
344 #define I2D_OF(type) int (*)(type *,unsigned char **)
345 #define I2D_OF_const(type) int (*)(const type *,unsigned char **)

347 #define CHECKED_D2I_OF(type, d2i) \
348     ((d2i_of_void*) (1 ? d2i : ((D2I_OF(type))0)))
349 #define CHECKED_I2D_OF(type, i2d) \
350     ((i2d_of_void*) (1 ? i2d : ((I2D_OF(type))0)))
351 #define CHECKED_NEW_OF(type, xnew) \
352     ((void *(*)(void)) (1 ? xnew : ((type *(*)(void))0)))
353 #define CHECKED_PTR_OF(type, p) \
354     ((void*) (1 ? p : (type*)0))
355 #define CHECKED_PPTR_OF(type, p) \
356     ((void**) (1 ? p : (type**)0))

358 #define TYPEDEF_D2I_OF(type) typedef type *d2i_of_##type(type **,const unsigned
359 #define TYPEDEF_I2D_OF(type) typedef int i2d_of_##type(type *,unsigned char **)
360 #define TYPEDEF_D2I2D_OF(type) TYPEDEF_D2I_OF(type); TYPEDEF_I2D_OF(type)

362 TYPEDEF_D2I2D_OF(void);

364 /* The following macros and typedefs allow an ASN1_ITEM
365  * to be embedded in a structure and referenced. Since
366  * the ASN1_ITEM pointers need to be globally accessible
367  * (possibly from shared libraries) they may exist in
368  * different forms. On platforms that support it the
369  * ASN1_ITEM structure itself will be globally exported.
370  * Other platforms will export a function that returns
371  * an ASN1_ITEM pointer.
372  *
373  * To handle both cases transparently the macros below
374  * should be used instead of hard coding an ASN1_ITEM
375  * pointer in a structure.
376  *
377  * The structure will look like this:
378  *
379  * typedef struct SOMETHING_st {
380  *      ...
381  *      ASN1_ITEM_EXP *iptr;
382  *      ...
383  * } SOMETHING;
384  *
385  * It would be initialised as e.g.:
386  *
387  * SOMETHING somevar = {...,ASN1_ITEM_ref(X509),...};
388  *
389  * and the actual pointer extracted with:
390  *
391  * const ASN1_ITEM *it = ASN1_ITEM_ptr(somevar.iptr);
```

```
392  *
393  * Finally an ASN1_ITEM pointer can be extracted from an
394  * appropriate reference with: ASN1_ITEM_rptr(X509). This
395  * would be used when a function takes an ASN1_ITEM * argument.
396  *
397  */

399 #ifndef OPENSSL_EXPORT_VAR_AS_FUNCTION

401 /* ASN1_ITEM pointer exported type */
402 typedef const ASN1_ITEM ASN1_ITEM_EXP;

404 /* Macro to obtain ASN1_ITEM pointer from exported type */
405 #define ASN1_ITEM_ptr(iptr) (iptr)

407 /* Macro to include ASN1_ITEM pointer from base type */
408 #define ASN1_ITEM_ref(iptr) (&(iptr##_it))

410 #define ASN1_ITEM_rptr(ref) (&(ref##_it))

412 #define DECLARE_ASN1_ITEM(name) \
413         OPENSSL_EXTERN const ASN1_ITEM name##_it;

415 #else

417 /* Platforms that can't easily handle shared global variables are declared
418  * as functions returning ASN1_ITEM pointers.
419  */

421 /* ASN1_ITEM pointer exported type */
422 typedef const ASN1_ITEM * ASN1_ITEM_EXP(void);

424 /* Macro to obtain ASN1_ITEM pointer from exported type */
425 #define ASN1_ITEM_ptr(iptr) (iptr())

427 /* Macro to include ASN1_ITEM pointer from base type */
428 #define ASN1_ITEM_ref(iptr) (iptr##_it)

430 #define ASN1_ITEM_rptr(ref) (ref##_it())

432 #define DECLARE_ASN1_ITEM(name) \
433         const ASN1_ITEM * name##_it(void);

435 #endif

437 /* Parameters used by ASN1_STRING_print_ex() */

439 /* These determine which characters to escape:
440  * RFC2253 special characters, control characters and
441  * MSB set characters
442  */

444 #define ASN1_STRFLGS_ESC_2253           1
445 #define ASN1_STRFLGS_ESC_CTRL           2
446 #define ASN1_STRFLGS_ESC_MSB            4


449 /* This flag determines how we do escaping: normally
450  * RC2253 backslash only, set this to use backslash and
451  * quote.
452  */

454 #define ASN1_STRFLGS_ESC_QUOTE          8


457 /* These three flags are internal use only. */
```



```
459 /* Character is a valid PrintableString character */
460 #define CHARTYPE_PRINTABLESTRING        0x10
461 /* Character needs escaping if it is the first character */
462 #define CHARTYPE_FIRST_ESC_2253         0x20
463 /* Character needs escaping if it is the last character */
464 #define CHARTYPE_LAST_ESC_2253          0x40

466 /* NB the internal flags are safely reused below by flags
467  * handled at the top level.
468  */

470 /* If this is set we convert all character strings
471  * to UTF8 first
472  */

474 #define ASN1_STRFLGS_UTF8_CONVERT       0x10

476 /* If this is set we don't attempt to interpret content:
477  * just assume all strings are 1 byte per character. This
478  * will produce some pretty odd looking output!
479  */

481 #define ASN1_STRFLGS_IGNORE_TYPE        0x20

483 /* If this is set we include the string type in the output */
484 #define ASN1_STRFLGS_SHOW_TYPE          0x40

486 /* This determines which strings to display and which to
487  * 'dump' (hex dump of content octets or DER encoding). We can
488  * only dump non character strings or everything. If we
489  * don't dump 'unknown' they are interpreted as character
490  * strings with 1 octet per character and are subject to
491  * the usual escaping options.
492  */

494 #define ASN1_STRFLGS_DUMP_ALL           0x80
495 #define ASN1_STRFLGS_DUMP_UNKNOWN       0x100

497 /* These determine what 'dumping' does, we can dump the
498  * content octets or the DER encoding: both use the
499  * RFC2253 #XXXXX notation.
500  */

502 #define ASN1_STRFLGS_DUMP_DER           0x200

504 /* All the string flags consistent with RFC2253,
505  * escaping control characters isn't essential in
506  * RFC2253 but it is advisable anyway.
507  */

509 #define ASN1_STRFLGS_RFC2253    (ASN1_STRFLGS_ESC_2253 | \
510                                 ASN1_STRFLGS_ESC_CTRL | \
511                                 ASN1_STRFLGS_ESC_MSB | \
512                                 ASN1_STRFLGS_UTF8_CONVERT | \
513                                 ASN1_STRFLGS_DUMP_UNKNOWN | \
514                                 ASN1_STRFLGS_DUMP_DER)

516 DECLARE_STACK_OF(ASN1_INTEGER)
517 DECLARE_ASN1_SET_OF(ASN1_INTEGER)

519 DECLARE_STACK_OF(ASN1_GENERALSTRING)

521 typedef struct asn1_type_st
522         {
523         int type;
```

```
524         union   {
525                 char *ptr;
526                 ASN1_BOOLEAN            boolean;
527                 ASN1_STRING *           asn1_string;
528                 ASN1_OBJECT *           object;
529                 ASN1_INTEGER *          integer;
530                 ASN1_ENUMERATED *       enumerated;
531                 ASN1_BIT_STRING *       bit_string;
532                 ASN1_OCTET_STRING *     octet_string;
533                 ASN1_PRINTABLESTRING *  printablestring;
534                 ASN1_T61STRING *        t61string;
535                 ASN1_IA5STRING *        ia5string;
536                 ASN1_GENERALSTRING *    generalstring;
537                 ASN1_BMPSTRING *        bmpstring;
538                 ASN1_UNIVERSALSTRING *  universalstring;
539                 ASN1_UTCTIME *          utctime;
540                 ASN1_GENERALIZEDTIME *  generalizedtime;
541                 ASN1_VISIBLESTRING *    visiblestring;
542                 ASN1_UTF8STRING *       utf8string;
543                 /* set and sequence are left complete and still
544                  * contain the set or sequence bytes */
545                 ASN1_STRING *           set;
546                 ASN1_STRING *           sequence;
547                 ASN1_VALUE *            asn1_value;
548                 } value;
549         } ASN1_TYPE;

551 DECLARE_STACK_OF(ASN1_TYPE)
552 DECLARE_ASN1_SET_OF(ASN1_TYPE)

554 typedef STACK_OF(ASN1_TYPE) ASN1_SEQUENCE_ANY;

556 DECLARE_ASN1_ENCODE_FUNCTIONS_const(ASN1_SEQUENCE_ANY, ASN1_SEQUENCE_ANY)
557 DECLARE_ASN1_ENCODE_FUNCTIONS_const(ASN1_SEQUENCE_ANY, ASN1_SET_ANY)

559 typedef struct NETSCAPE_X509_st
560         {
561         ASN1_OCTET_STRING *header;
562         X509 *cert;
563         } NETSCAPE_X509;

565 /* This is used to contain a list of bit names */
566 typedef struct BIT_STRING_BITNAME_st {
567         int bitnum;
568         const char *lname;
569         const char *sname;
570 } BIT_STRING_BITNAME;


573 #define M_ASN1_STRING_length(x) ((x)->length)
574 #define M_ASN1_STRING_length_set(x, n)  ((x)->length = (n))
575 #define M_ASN1_STRING_type(x)   ((x)->type)
576 #define M_ASN1_STRING_data(x)   ((x)->data)

578 /* Macros for string operations */
579 #define M_ASN1_BIT_STRING_new() (ASN1_BIT_STRING *)\
580                 ASN1_STRING_type_new(V_ASN1_BIT_STRING)
581 #define M_ASN1_BIT_STRING_free(a)       ASN1_STRING_free((ASN1_STRING *)a)
582 #define M_ASN1_BIT_STRING_dup(a) (ASN1_BIT_STRING *)\
583                 ASN1_STRING_dup((const ASN1_STRING *)a)
584 #define M_ASN1_BIT_STRING_cmp(a,b) ASN1_STRING_cmp(\
585                 (const ASN1_STRING *)a,(const ASN1_STRING *)b)
586 #define M_ASN1_BIT_STRING_set(a,b,c) ASN1_STRING_set((ASN1_STRING *)a,b,c)

588 #define M_ASN1_INTEGER_new()    (ASN1_INTEGER *)\
589                 ASN1_STRING_type_new(V_ASN1_INTEGER)
```



```
590 #define M_ASN1_INTEGER_free(a)          ASN1_STRING_free((ASN1_STRING *)a)
591 #define M_ASN1_INTEGER_dup(a) (ASN1_INTEGER *)\
592                 ASN1_STRING_dup((const ASN1_STRING *)a)
593 #define M_ASN1_INTEGER_cmp(a,b) ASN1_STRING_cmp(\
594                 (const ASN1_STRING *)a,(const ASN1_STRING *)b)

596 #define M_ASN1_ENUMERATED_new() (ASN1_ENUMERATED *)\
597                 ASN1_STRING_type_new(V_ASN1_ENUMERATED)
598 #define M_ASN1_ENUMERATED_free(a)       ASN1_STRING_free((ASN1_STRING *)a)
599 #define M_ASN1_ENUMERATED_dup(a) (ASN1_ENUMERATED *)\
600                 ASN1_STRING_dup((const ASN1_STRING *)a)
601 #define M_ASN1_ENUMERATED_cmp(a,b)      ASN1_STRING_cmp(\
602                 (const ASN1_STRING *)a,(const ASN1_STRING *)b)

604 #define M_ASN1_OCTET_STRING_new()       (ASN1_OCTET_STRING *)\
605                 ASN1_STRING_type_new(V_ASN1_OCTET_STRING)
606 #define M_ASN1_OCTET_STRING_free(a)     ASN1_STRING_free((ASN1_STRING *)a)
607 #define M_ASN1_OCTET_STRING_dup(a) (ASN1_OCTET_STRING *)\
608                 ASN1_STRING_dup((const ASN1_STRING *)a)
609 #define M_ASN1_OCTET_STRING_cmp(a,b) ASN1_STRING_cmp(\
610                 (const ASN1_STRING *)a,(const ASN1_STRING *)b)
611 #define M_ASN1_OCTET_STRING_set(a,b,c)  ASN1_STRING_set((ASN1_STRING *)a,b,c)
612 #define M_ASN1_OCTET_STRING_print(a,b)  ASN1_STRING_print(a,(ASN1_STRING *)b)
613 #define M_i2d_ASN1_OCTET_STRING(a,pp) \
614                 i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_OCTET_STRING,\
615                 V_ASN1_UNIVERSAL)

617 #define B_ASN1_TIME \
618                         B_ASN1_UTCTIME | \
619                         B_ASN1_GENERALIZEDTIME

621 #define B_ASN1_PRINTABLE \
622                         B_ASN1_NUMERICSTRING| \
623                         B_ASN1_PRINTABLESTRING| \
624                         B_ASN1_T61STRING| \
625                         B_ASN1_IA5STRING| \
626                         B_ASN1_BIT_STRING| \
627                         B_ASN1_UNIVERSALSTRING|\
628                         B_ASN1_BMPSTRING|\
629                         B_ASN1_UTF8STRING|\
630                         B_ASN1_SEQUENCE|\
631                         B_ASN1_UNKNOWN

633 #define B_ASN1_DIRECTORYSTRING \
634                         B_ASN1_PRINTABLESTRING| \
635                         B_ASN1_TELETEXSTRING|\
636                         B_ASN1_BMPSTRING|\
637                         B_ASN1_UNIVERSALSTRING|\
638                         B_ASN1_UTF8STRING

640 #define B_ASN1_DISPLAYTEXT \
641                         B_ASN1_IA5STRING| \
642                         B_ASN1_VISIBLESTRING| \
643                         B_ASN1_BMPSTRING|\
644                         B_ASN1_UTF8STRING

646 #define M_ASN1_PRINTABLE_new()  ASN1_STRING_type_new(V_ASN1_T61STRING)
647 #define M_ASN1_PRINTABLE_free(a)        ASN1_STRING_free((ASN1_STRING *)a)
648 #define M_i2d_ASN1_PRINTABLE(a,pp) i2d_ASN1_bytes((ASN1_STRING *)a,\
649                 pp,a->type,V_ASN1_UNIVERSAL)
650 #define M_d2i_ASN1_PRINTABLE(a,pp,l) \
651                 d2i_ASN1_type_bytes((ASN1_STRING **)a,pp,l, \
652                         B_ASN1_PRINTABLE)

654 #define M_DIRECTORYSTRING_new() ASN1_STRING_type_new(V_ASN1_PRINTABLESTRING)
655 #define M_DIRECTORYSTRING_free(a)       ASN1_STRING_free((ASN1_STRING *)a)
```

```
#define M_i2d_DIRECTORYSTRING(a,pp) i2d_ASN1_bytes((ASN1_STRING *)a,\
                                                pp,a->type,V_ASN1_UNIVERSAL)
#define M_d2i_DIRECTORYSTRING(a,pp,l) \
                d2i_ASN1_type_bytes((ASN1_STRING **)a,pp,l, \
                        B_ASN1_DIRECTORYSTRING)

#define M_DISPLAYTEXT_new() ASN1_STRING_type_new(V_ASN1_VISIBLESTRING)
#define M_DISPLAYTEXT_free(a) ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_DISPLAYTEXT(a,pp) i2d_ASN1_bytes((ASN1_STRING *)a,\
                                                pp,a->type,V_ASN1_UNIVERSAL)
#define M_d2i_DISPLAYTEXT(a,pp,l) \
                d2i_ASN1_type_bytes((ASN1_STRING **)a,pp,l, \
                        B_ASN1_DISPLAYTEXT)

#define M_ASN1_PRINTABLESTRING_new() (ASN1_PRINTABLESTRING *)\
                ASN1_STRING_type_new(V_ASN1_PRINTABLESTRING)
#define M_ASN1_PRINTABLESTRING_free(a)  ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_ASN1_PRINTABLESTRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_PRINTABLESTRING,\
                V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_PRINTABLESTRING(a,pp,l) \
                (ASN1_PRINTABLESTRING *)d2i_ASN1_type_bytes\
                ((ASN1_STRING **)a,pp,l,B_ASN1_PRINTABLESTRING)

#define M_ASN1_T61STRING_new()  (ASN1_T61STRING *)\
                ASN1_STRING_type_new(V_ASN1_T61STRING)
#define M_ASN1_T61STRING_free(a)        ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_ASN1_T61STRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_T61STRING,\
                V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_T61STRING(a,pp,l) \
                (ASN1_T61STRING *)d2i_ASN1_type_bytes\
                ((ASN1_STRING **)a,pp,l,B_ASN1_T61STRING)

#define M_ASN1_IA5STRING_new()  (ASN1_IA5STRING *)\
                ASN1_STRING_type_new(V_ASN1_IA5STRING)
#define M_ASN1_IA5STRING_free(a)        ASN1_STRING_free((ASN1_STRING *)a)
#define M_ASN1_IA5STRING_dup(a) \
                (ASN1_IA5STRING *)ASN1_STRING_dup((const ASN1_STRING *)a)
#define M_i2d_ASN1_IA5STRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_IA5STRING,\
                        V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_IA5STRING(a,pp,l) \
                (ASN1_IA5STRING *)d2i_ASN1_type_bytes((ASN1_STRING **)a,pp,l,\
                        B_ASN1_IA5STRING)

#define M_ASN1_UTCTIME_new()    (ASN1_UTCTIME *)\
                ASN1_STRING_type_new(V_ASN1_UTCTIME)
#define M_ASN1_UTCTIME_free(a)  ASN1_STRING_free((ASN1_STRING *)a)
#define M_ASN1_UTCTIME_dup(a) (ASN1_UTCTIME *)\
                ASN1_STRING_dup((const ASN1_STRING *)a)

#define M_ASN1_GENERALIZEDTIME_new()    (ASN1_GENERALIZEDTIME *)\
                ASN1_STRING_type_new(V_ASN1_GENERALIZEDTIME)
#define M_ASN1_GENERALIZEDTIME_free(a)  ASN1_STRING_free((ASN1_STRING *)a)
#define M_ASN1_GENERALIZEDTIME_dup(a) (ASN1_GENERALIZEDTIME *)ASN1_STRING_dup(\
        (const ASN1_STRING *)a)

#define M_ASN1_TIME_new()       (ASN1_TIME *)\
                ASN1_STRING_type_new(V_ASN1_UTCTIME)
#define M_ASN1_TIME_free(a)     ASN1_STRING_free((ASN1_STRING *)a)
#define M_ASN1_TIME_dup(a) (ASN1_TIME *)\
        ASN1_STRING_dup((const ASN1_STRING *)a)

#define M_ASN1_GENERALSTRING_new()      (ASN1_GENERALSTRING *)\
                ASN1_STRING_type_new(V_ASN1_GENERALSTRING)
```



```
#define M_ASN1_GENERALSTRING_free(a)    ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_ASN1_GENERALSTRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_GENERALSTRING,\
                        V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_GENERALSTRING(a,pp,l) \
                (ASN1_GENERALSTRING *)d2i_ASN1_type_bytes\
                ((ASN1_STRING **)a,pp,l,B_ASN1_GENERALSTRING)

#define M_ASN1_UNIVERSALSTRING_new()    (ASN1_UNIVERSALSTRING *)\
                ASN1_STRING_type_new(V_ASN1_UNIVERSALSTRING)
#define M_ASN1_UNIVERSALSTRING_free(a)  ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_ASN1_UNIVERSALSTRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_UNIVERSALSTRING,\
                        V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_UNIVERSALSTRING(a,pp,l) \
                (ASN1_UNIVERSALSTRING *)d2i_ASN1_type_bytes\
                ((ASN1_STRING **)a,pp,l,B_ASN1_UNIVERSALSTRING)

#define M_ASN1_BMPSTRING_new()  (ASN1_BMPSTRING *)\
                ASN1_STRING_type_new(V_ASN1_BMPSTRING)
#define M_ASN1_BMPSTRING_free(a)        ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_ASN1_BMPSTRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_BMPSTRING,\
                        V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_BMPSTRING(a,pp,l) \
                (ASN1_BMPSTRING *)d2i_ASN1_type_bytes\
                ((ASN1_STRING **)a,pp,l,B_ASN1_BMPSTRING)

#define M_ASN1_VISIBLESTRING_new()      (ASN1_VISIBLESTRING *)\
                ASN1_STRING_type_new(V_ASN1_VISIBLESTRING)
#define M_ASN1_VISIBLESTRING_free(a)    ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_ASN1_VISIBLESTRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_VISIBLESTRING,\
                        V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_VISIBLESTRING(a,pp,l) \
                (ASN1_VISIBLESTRING *)d2i_ASN1_type_bytes\
                ((ASN1_STRING **)a,pp,l,B_ASN1_VISIBLESTRING)

#define M_ASN1_UTF8STRING_new() (ASN1_UTF8STRING *)\
                ASN1_STRING_type_new(V_ASN1_UTF8STRING)
#define M_ASN1_UTF8STRING_free(a)       ASN1_STRING_free((ASN1_STRING *)a)
#define M_i2d_ASN1_UTF8STRING(a,pp) \
                i2d_ASN1_bytes((ASN1_STRING *)a,pp,V_ASN1_UTF8STRING,\
                        V_ASN1_UNIVERSAL)
#define M_d2i_ASN1_UTF8STRING(a,pp,l) \
                (ASN1_UTF8STRING *)d2i_ASN1_type_bytes\
                ((ASN1_STRING **)a,pp,l,B_ASN1_UTF8STRING)

  /* for the is_set parameter to i2d_ASN1_SET */
#define IS_SEQUENCE     0
#define IS_SET          1

DECLARE_ASN1_FUNCTIONS_fname(ASN1_TYPE, ASN1_ANY, ASN1_TYPE)

int ASN1_TYPE_get(ASN1_TYPE *a);
void ASN1_TYPE_set(ASN1_TYPE *a, int type, void *value);
int ASN1_TYPE_set1(ASN1_TYPE *a, int type, const void *value);
int             ASN1_TYPE_cmp(ASN1_TYPE *a, ASN1_TYPE *b);

ASN1_OBJECT *   ASN1_OBJECT_new(void );
void            ASN1_OBJECT_free(ASN1_OBJECT *a);
int             i2d_ASN1_OBJECT(ASN1_OBJECT *a,unsigned char **pp);
ASN1_OBJECT *   c2i_ASN1_OBJECT(ASN1_OBJECT **a,const unsigned char **pp,
                        long length);
ASN1_OBJECT *   d2i_ASN1_OBJECT(ASN1_OBJECT **a,const unsigned char **pp,
                        long length);
```

```
789 DECLARE_ASN1_ITEM(ASN1_OBJECT)

791 DECLARE_STACK_OF(ASN1_OBJECT)
792 DECLARE_ASN1_SET_OF(ASN1_OBJECT)

794 ASN1_STRING *   ASN1_STRING_new(void);
795 void            ASN1_STRING_free(ASN1_STRING *a);
796 int             ASN1_STRING_copy(ASN1_STRING *dst, const ASN1_STRING *str);
797 ASN1_STRING *   ASN1_STRING_dup(const ASN1_STRING *a);
798 ASN1_STRING *   ASN1_STRING_type_new(int type );
799 int             ASN1_STRING_cmp(const ASN1_STRING *a, const ASN1_STRING *b);
800   /* Since this is used to store all sorts of things, via macros, for now, make
801      its data void * */
802 int             ASN1_STRING_set(ASN1_STRING *str, const void *data, int len);
803 void            ASN1_STRING_set0(ASN1_STRING *str, void *data, int len);
804 int ASN1_STRING_length(const ASN1_STRING *x);
805 void ASN1_STRING_length_set(ASN1_STRING *x, int n);
806 int ASN1_STRING_type(ASN1_STRING *x);
807 unsigned char * ASN1_STRING_data(ASN1_STRING *x);

809 DECLARE_ASN1_FUNCTIONS(ASN1_BIT_STRING)
810 int             i2c_ASN1_BIT_STRING(ASN1_BIT_STRING *a,unsigned char **pp);
811 ASN1_BIT_STRING *c2i_ASN1_BIT_STRING(ASN1_BIT_STRING **a,const unsigned char **p
812                         long length);
813 int             ASN1_BIT_STRING_set(ASN1_BIT_STRING *a, unsigned char *d,
814                         int length );
815 int             ASN1_BIT_STRING_set_bit(ASN1_BIT_STRING *a, int n, int value);
816 int             ASN1_BIT_STRING_get_bit(ASN1_BIT_STRING *a, int n);
817 int            ASN1_BIT_STRING_check(ASN1_BIT_STRING *a,
818                                      unsigned char *flags, int flags_len);

820 #ifndef OPENSSL_NO_BIO
821 int ASN1_BIT_STRING_name_print(BIO *out, ASN1_BIT_STRING *bs,
822                                BIT_STRING_BITNAME *tbl, int indent);
823 #endif
824 int ASN1_BIT_STRING_num_asc(char *name, BIT_STRING_BITNAME *tbl);
825 int ASN1_BIT_STRING_set_asc(ASN1_BIT_STRING *bs, char *name, int value,
826                                BIT_STRING_BITNAME *tbl);

828 int             i2d_ASN1_BOOLEAN(int a,unsigned char **pp);
829 int             d2i_ASN1_BOOLEAN(int *a,const unsigned char **pp,long length);

831 DECLARE_ASN1_FUNCTIONS(ASN1_INTEGER)
832 int             i2c_ASN1_INTEGER(ASN1_INTEGER *a,unsigned char **pp);
833 ASN1_INTEGER *c2i_ASN1_INTEGER(ASN1_INTEGER **a,const unsigned char **pp,
834                         long length);
835 ASN1_INTEGER *d2i_ASN1_UINTEGER(ASN1_INTEGER **a,const unsigned char **pp,
836                         long length);
837 ASN1_INTEGER *  ASN1_INTEGER_dup(const ASN1_INTEGER *x);
838 int ASN1_INTEGER_cmp(const ASN1_INTEGER *x, const ASN1_INTEGER *y);

840 DECLARE_ASN1_FUNCTIONS(ASN1_ENUMERATED)

842 int ASN1_UTCTIME_check(ASN1_UTCTIME *a);
843 ASN1_UTCTIME *ASN1_UTCTIME_set(ASN1_UTCTIME *s,time_t t);
844 ASN1_UTCTIME *ASN1_UTCTIME_adj(ASN1_UTCTIME *s, time_t t,
845                                int offset_day, long offset_sec);
846 int ASN1_UTCTIME_set_string(ASN1_UTCTIME *s, const char *str);
847 int ASN1_UTCTIME_cmp_time_t(const ASN1_UTCTIME *s, time_t t);
848 #if 0
849 time_t ASN1_UTCTIME_get(const ASN1_UTCTIME *s);
850 #endif

852 int ASN1_GENERALIZEDTIME_check(ASN1_GENERALIZEDTIME *a);
853 ASN1_GENERALIZEDTIME *ASN1_GENERALIZEDTIME_set(ASN1_GENERALIZEDTIME *s,time_t t)
```



```
854 ASN1_GENERALIZEDTIME *ASN1_GENERALIZEDTIME_adj(ASN1_GENERALIZEDTIME *s,
855             time_t t, int offset_day, long offset_sec);
856 int ASN1_GENERALIZEDTIME_set_string(ASN1_GENERALIZEDTIME *s, const char *str);

858 DECLARE_ASN1_FUNCTIONS(ASN1_OCTET_STRING)
859 ASN1_OCTET_STRING *     ASN1_OCTET_STRING_dup(const ASN1_OCTET_STRING *a);
860 int     ASN1_OCTET_STRING_cmp(const ASN1_OCTET_STRING *a, const ASN1_OCTET_STRIN
861 int     ASN1_OCTET_STRING_set(ASN1_OCTET_STRING *str, const unsigned char *data,

863 DECLARE_ASN1_FUNCTIONS(ASN1_VISIBLESTRING)
864 DECLARE_ASN1_FUNCTIONS(ASN1_UNIVERSALSTRING)
865 DECLARE_ASN1_FUNCTIONS(ASN1_UTF8STRING)
866 DECLARE_ASN1_FUNCTIONS(ASN1_NULL)
867 DECLARE_ASN1_FUNCTIONS(ASN1_BMPSTRING)

869 int UTF8_getc(const unsigned char *str, int len, unsigned long *val);
870 int UTF8_putc(unsigned char *str, int len, unsigned long value);

872 DECLARE_ASN1_FUNCTIONS_name(ASN1_STRING, ASN1_PRINTABLE)

874 DECLARE_ASN1_FUNCTIONS_name(ASN1_STRING, DIRECTORYSTRING)
875 DECLARE_ASN1_FUNCTIONS_name(ASN1_STRING, DISPLAYTEXT)
876 DECLARE_ASN1_FUNCTIONS(ASN1_PRINTABLESTRING)
877 DECLARE_ASN1_FUNCTIONS(ASN1_T61STRING)
878 DECLARE_ASN1_FUNCTIONS(ASN1_IA5STRING)
879 DECLARE_ASN1_FUNCTIONS(ASN1_GENERALSTRING)
880 DECLARE_ASN1_FUNCTIONS(ASN1_UTCTIME)
881 DECLARE_ASN1_FUNCTIONS(ASN1_GENERALIZEDTIME)
882 DECLARE_ASN1_FUNCTIONS(ASN1_TIME)

884 DECLARE_ASN1_ITEM(ASN1_OCTET_STRING_NDEF)

886 ASN1_TIME *ASN1_TIME_set(ASN1_TIME *s,time_t t);
887 ASN1_TIME *ASN1_TIME_adj(ASN1_TIME *s,time_t t,
888                                  int offset_day, long offset_sec);
889 int ASN1_TIME_check(ASN1_TIME *t);
890 ASN1_GENERALIZEDTIME *ASN1_TIME_to_generalizedtime(ASN1_TIME *t, ASN1_GENERALIZE
891 int ASN1_TIME_set_string(ASN1_TIME *s, const char *str);

893 int i2d_ASN1_SET(STACK_OF(OPENSSL_BLOCK) *a, unsigned char **pp,
894                  i2d_of_void *i2d, int ex_tag, int ex_class,
895                  int is_set);
896 STACK_OF(OPENSSL_BLOCK) *d2i_ASN1_SET(STACK_OF(OPENSSL_BLOCK) **a,
897                                 const unsigned char **pp,
898                                 long length, d2i_of_void *d2i,
899                                 void (*free_func)(OPENSSL_BLOCK), int ex_tag,
900                                 int ex_class);

902 #ifndef OPENSSL_NO_BIO
903 int i2a_ASN1_INTEGER(BIO *bp, ASN1_INTEGER *a);
904 int a2i_ASN1_INTEGER(BIO *bp,ASN1_INTEGER *bs,char *buf,int size);
905 int i2a_ASN1_ENUMERATED(BIO *bp, ASN1_ENUMERATED *a);
906 int a2i_ASN1_ENUMERATED(BIO *bp,ASN1_ENUMERATED *bs,char *buf,int size);
907 int i2a_ASN1_OBJECT(BIO *bp,ASN1_OBJECT *a);
908 int a2i_ASN1_STRING(BIO *bp,ASN1_STRING *bs,char *buf,int size);
909 int i2a_ASN1_STRING(BIO *bp, ASN1_STRING *a, int type);
910 #endif
911 int i2t_ASN1_OBJECT(char *buf,int buf_len,ASN1_OBJECT *a);

913 int a2d_ASN1_OBJECT(unsigned char *out,int olen, const char *buf, int num);
914 ASN1_OBJECT *ASN1_OBJECT_create(int nid, unsigned char *data,int len,
915         const char *sn, const char *ln);

917 int ASN1_INTEGER_set(ASN1_INTEGER *a, long v);
918 long ASN1_INTEGER_get(const ASN1_INTEGER *a);
919 ASN1_INTEGER *BN_to_ASN1_INTEGER(const BIGNUM *bn, ASN1_INTEGER *ai);
```

```
BIGNUM *ASN1_INTEGER_to_BN(const ASN1_INTEGER *ai,BIGNUM *bn);

int ASN1_ENUMERATED_set(ASN1_ENUMERATED *a, long v);
long ASN1_ENUMERATED_get(ASN1_ENUMERATED *a);
ASN1_ENUMERATED *BN_to_ASN1_ENUMERATED(BIGNUM *bn, ASN1_ENUMERATED *ai);
BIGNUM *ASN1_ENUMERATED_to_BN(ASN1_ENUMERATED *ai,BIGNUM *bn);

/* General */
/* given a string, return the correct type, max is the maximum length */
int ASN1_PRINTABLE_type(const unsigned char *s, int max);

int i2d_ASN1_bytes(ASN1_STRING *a, unsigned char **pp, int tag, int xclass);
ASN1_STRING *d2i_ASN1_bytes(ASN1_STRING **a, const unsigned char **pp,
        long length, int Ptag, int Pclass);
unsigned long ASN1_tag2bit(int tag);
/* type is one or more of the B_ASN1_ values. */
ASN1_STRING *d2i_ASN1_type_bytes(ASN1_STRING **a,const unsigned char **pp,
                long length,int type);

/* PARSING */
int asn1_Finish(ASN1_CTX *c);
int asn1_const_Finish(ASN1_const_CTX *c);

/* SPECIALS */
int ASN1_get_object(const unsigned char **pp, long *plength, int *ptag,
        int *pclass, long omax);
int ASN1_check_infinite_end(unsigned char **p,long len);
int ASN1_const_check_infinite_end(const unsigned char **p,long len);
void ASN1_put_object(unsigned char **pp, int constructed, int length,
        int tag, int xclass);
int ASN1_put_eoc(unsigned char **pp);
int ASN1_object_size(int constructed, int length, int tag);

/* Used to implement other functions */
void *ASN1_dup(i2d_of_void *i2d, d2i_of_void *d2i, void *x);

#define ASN1_dup_of(type,i2d,d2i,x) \
    ((type*)ASN1_dup(CHECKED_I2D_OF(type, i2d), \
                     CHECKED_D2I_OF(type, d2i), \
                     CHECKED_PTR_OF(type, x)))

#define ASN1_dup_of_const(type,i2d,d2i,x) \
    ((type*)ASN1_dup(CHECKED_I2D_OF(const type, i2d), \
                     CHECKED_D2I_OF(type, d2i), \
                     CHECKED_PTR_OF(const type, x)))

void *ASN1_item_dup(const ASN1_ITEM *it, void *x);

/* ASN1 alloc/free macros for when a type is only used internally */

#define M_ASN1_new_of(type) (type *)ASN1_item_new(ASN1_ITEM_rptr(type))
#define M_ASN1_free_of(x, type) \
                ASN1_item_free(CHECKED_PTR_OF(type, x), ASN1_ITEM_rptr(type))

#ifndef OPENSSL_NO_FP_API
void *ASN1_d2i_fp(void *(*xnew)(void), d2i_of_void *d2i, FILE *in, void **x);

#define ASN1_d2i_fp_of(type,xnew,d2i,in,x) \
    ((type*)ASN1_d2i_fp(CHECKED_NEW_OF(type, xnew), \
                        CHECKED_D2I_OF(type, d2i), \
                        in, \
                        CHECKED_PPTR_OF(type, x)))

void *ASN1_item_d2i_fp(const ASN1_ITEM *it, FILE *in, void *x);
int ASN1_i2d_fp(i2d_of_void *i2d,FILE *out,void *x);
```



```
#define ASN1_i2d_fp_of(type,i2d,out,x) \
    (ASN1_i2d_fp(CHECKED_I2D_OF(type, i2d), \
                 out, \
                 CHECKED_PTR_OF(type, x)))

#define ASN1_i2d_fp_of_const(type,i2d,out,x) \
    (ASN1_i2d_fp(CHECKED_I2D_OF(const type, i2d), \
                 out, \
                 CHECKED_PTR_OF(const type, x)))

int ASN1_item_i2d_fp(const ASN1_ITEM *it, FILE *out, void *x);
int ASN1_STRING_print_ex_fp(FILE *fp, ASN1_STRING *str, unsigned long flags);
#endif

int ASN1_STRING_to_UTF8(unsigned char **out, ASN1_STRING *in);

#ifndef OPENSSL_NO_BIO
void *ASN1_d2i_bio(void *(*xnew)(void), d2i_of_void *d2i, BIO *in, void **x);

#define ASN1_d2i_bio_of(type,xnew,d2i,in,x) \
    ((type*)ASN1_d2i_bio( CHECKED_NEW_OF(type, xnew), \
                          CHECKED_D2I_OF(type, d2i), \
                          in, \
                          CHECKED_PPTR_OF(type, x)))

void *ASN1_item_d2i_bio(const ASN1_ITEM *it, BIO *in, void *x);
int ASN1_i2d_bio(i2d_of_void *i2d,BIO *out, unsigned char *x);

#define ASN1_i2d_bio_of(type,i2d,out,x) \
    (ASN1_i2d_bio(CHECKED_I2D_OF(type, i2d), \
                  out, \
                  CHECKED_PTR_OF(type, x)))

#define ASN1_i2d_bio_of_const(type,i2d,out,x) \
    (ASN1_i2d_bio(CHECKED_I2D_OF(const type, i2d), \
                  out, \
                  CHECKED_PTR_OF(const type, x)))

int ASN1_item_i2d_bio(const ASN1_ITEM *it, BIO *out, void *x);
int ASN1_UTCTIME_print(BIO *fp, const ASN1_UTCTIME *a);
int ASN1_GENERALIZEDTIME_print(BIO *fp, const ASN1_GENERALIZEDTIME *a);
int ASN1_TIME_print(BIO *fp, const ASN1_TIME *a);
int ASN1_STRING_print(BIO *bp, const ASN1_STRING *v);
int ASN1_STRING_print_ex(BIO *out, ASN1_STRING *str, unsigned long flags);
int ASN1_bn_print(BIO *bp, const char *number, const BIGNUM *num,
                                  unsigned char *buf, int off);
int ASN1_parse(BIO *bp,const unsigned char *pp,long len,int indent);
int ASN1_parse_dump(BIO *bp,const unsigned char *pp,long len,int indent,int dump
#endif
const char *ASN1_tag2str(int tag);

/* Used to load and write netscape format cert */

DECLARE_ASN1_FUNCTIONS(NETSCAPE_X509)

int ASN1_UNIVERSALSTRING_to_string(ASN1_UNIVERSALSTRING *s);

int ASN1_TYPE_set_octetstring(ASN1_TYPE *a,
        unsigned char *data, int len);
int ASN1_TYPE_get_octetstring(ASN1_TYPE *a,
        unsigned char *data, int max_len);
int ASN1_TYPE_set_int_octetstring(ASN1_TYPE *a, long num,
        unsigned char *data, int len);
int ASN1_TYPE_get_int_octetstring(ASN1_TYPE *a,long *num,
        unsigned char *data, int max_len);
```

```
STACK_OF(OPENSSL_BLOCK) *ASN1_seq_unpack(const unsigned char *buf, int len,
                                 d2i_of_void *d2i, void (*free_func)(OPENSSL_BLO
unsigned char *ASN1_seq_pack(STACK_OF(OPENSSL_BLOCK) *safes, i2d_of_void *i2d,
                             unsigned char **buf, int *len );
void *ASN1_unpack_string(ASN1_STRING *oct, d2i_of_void *d2i);
void *ASN1_item_unpack(ASN1_STRING *oct, const ASN1_ITEM *it);
ASN1_STRING *ASN1_pack_string(void *obj, i2d_of_void *i2d,
                              ASN1_OCTET_STRING **oct);

#define ASN1_pack_string_of(type,obj,i2d,oct) \
    (ASN1_pack_string(CHECKED_PTR_OF(type, obj), \
                      CHECKED_I2D_OF(type, i2d), \
                      oct))

ASN1_STRING *ASN1_item_pack(void *obj, const ASN1_ITEM *it, ASN1_OCTET_STRING **

void ASN1_STRING_set_default_mask(unsigned long mask);
int ASN1_STRING_set_default_mask_asc(const char *p);
unsigned long ASN1_STRING_get_default_mask(void);
int ASN1_mbstring_copy(ASN1_STRING **out, const unsigned char *in, int len,
                                          int inform, unsigned long mask);
int ASN1_mbstring_ncopy(ASN1_STRING **out, const unsigned char *in, int len,
                                           int inform, unsigned long mask,
                                           long minsize, long maxsize);

ASN1_STRING *ASN1_STRING_set_by_NID(ASN1_STRING **out,
                const unsigned char *in, int inlen, int inform, int nid);
ASN1_STRING_TABLE *ASN1_STRING_TABLE_get(int nid);
int ASN1_STRING_TABLE_add(int, long, long, unsigned long, unsigned long);
void ASN1_STRING_TABLE_cleanup(void);

/* ASN1 template functions */

/* Old API compatible functions */
ASN1_VALUE *ASN1_item_new(const ASN1_ITEM *it);
void ASN1_item_free(ASN1_VALUE *val, const ASN1_ITEM *it);
ASN1_VALUE * ASN1_item_d2i(ASN1_VALUE **val, const unsigned char **in, long len,
int ASN1_item_i2d(ASN1_VALUE *val, unsigned char **out, const ASN1_ITEM *it);
int ASN1_item_ndef_i2d(ASN1_VALUE *val, unsigned char **out, const ASN1_ITEM *it

void ASN1_add_oid_module(void);

ASN1_TYPE *ASN1_generate_nconf(char *str, CONF *nconf);
ASN1_TYPE *ASN1_generate_v3(char *str, X509V3_CTX *cnf);

/* ASN1 Print flags */

/* Indicate missing OPTIONAL fields */
#define ASN1_PCTX_FLAGS_SHOW_ABSENT             0x001
/* Mark start and end of SEQUENCE */
#define ASN1_PCTX_FLAGS_SHOW_SEQUENCE           0x002
/* Mark start and end of SEQUENCE/SET OF */
#define ASN1_PCTX_FLAGS_SHOW_SSOF               0x004
/* Show the ASN1 type of primitives */
#define ASN1_PCTX_FLAGS_SHOW_TYPE               0x008
/* Don't show ASN1 type of ANY */
#define ASN1_PCTX_FLAGS_NO_ANY_TYPE             0x010
/* Don't show ASN1 type of MSTRINGs */
#define ASN1_PCTX_FLAGS_NO_MSTRING_TYPE         0x020
/* Don't show field names in SEQUENCE */
#define ASN1_PCTX_FLAGS_NO_FIELD_NAME           0x040
/* Show structure names of each SEQUENCE field */
#define ASN1_PCTX_FLAGS_SHOW_FIELD_STRUCT_NAME  0x080
/* Don't show structure name even at top level */
#define ASN1_PCTX_FLAGS_NO_STRUCT_NAME          0x100
```



```
int ASN1_item_print(BIO *out, ASN1_VALUE *ifld, int indent,
                                  const ASN1_ITEM *it, const ASN1_PCTX *pctx);
ASN1_PCTX *ASN1_PCTX_new(void);
void ASN1_PCTX_free(ASN1_PCTX *p);
unsigned long ASN1_PCTX_get_flags(ASN1_PCTX *p);
void ASN1_PCTX_set_flags(ASN1_PCTX *p, unsigned long flags);
unsigned long ASN1_PCTX_get_nm_flags(ASN1_PCTX *p);
void ASN1_PCTX_set_nm_flags(ASN1_PCTX *p, unsigned long flags);
unsigned long ASN1_PCTX_get_cert_flags(ASN1_PCTX *p);
void ASN1_PCTX_set_cert_flags(ASN1_PCTX *p, unsigned long flags);
unsigned long ASN1_PCTX_get_oid_flags(ASN1_PCTX *p);
void ASN1_PCTX_set_oid_flags(ASN1_PCTX *p, unsigned long flags);
unsigned long ASN1_PCTX_get_str_flags(ASN1_PCTX *p);
void ASN1_PCTX_set_str_flags(ASN1_PCTX *p, unsigned long flags);

BIO_METHOD *BIO_f_asn1(void);

BIO *BIO_new_NDEF(BIO *out, ASN1_VALUE *val, const ASN1_ITEM *it);

int i2d_ASN1_bio_stream(BIO *out, ASN1_VALUE *val, BIO *in, int flags,
                                  const ASN1_ITEM *it);
int PEM_write_bio_ASN1_stream(BIO *out, ASN1_VALUE *val, BIO *in, int flags,
                                  const char *hdr,
                                  const ASN1_ITEM *it);
int SMIME_write_ASN1(BIO *bio, ASN1_VALUE *val, BIO *data, int flags,
                                  int ctype_nid, int econt_nid,
                                  STACK_OF(X509_ALGOR) *mdalgs,
                                  const ASN1_ITEM *it);
ASN1_VALUE *SMIME_read_ASN1(BIO *bio, BIO **bcont, const ASN1_ITEM *it);
int SMIME_crlf_copy(BIO *in, BIO *out, int flags);
int SMIME_text(BIO *in, BIO *out);

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_ASN1_strings(void);

/* Error codes for the ASN1 functions. */

/* Function codes. */
#define ASN1_F_A2D_ASN1_OBJECT                          100
#define ASN1_F_A2I_ASN1_ENUMERATED                      101
#define ASN1_F_A2I_ASN1_INTEGER                         102
#define ASN1_F_A2I_ASN1_STRING                          103
#define ASN1_F_APPEND_EXP                               176
#define ASN1_F_ASN1_BIT_STRING_SET_BIT                  183
#define ASN1_F_ASN1_CB                                  177
#define ASN1_F_ASN1_CHECK_TLEN                          104
#define ASN1_F_ASN1_COLLATE_PRIMITIVE                   105
#define ASN1_F_ASN1_COLLECT                             106
#define ASN1_F_ASN1_D2I_EX_PRIMITIVE                    108
#define ASN1_F_ASN1_D2I_FP                              109
#define ASN1_F_ASN1_D2I_READ_BIO                        107
#define ASN1_F_ASN1_DIGEST                              184
#define ASN1_F_ASN1_DO_ADB                              110
#define ASN1_F_ASN1_DUP                                 111
#define ASN1_F_ASN1_ENUMERATED_SET                      112
#define ASN1_F_ASN1_ENUMERATED_TO_BN                    113
#define ASN1_F_ASN1_EX_C2I                              204
#define ASN1_F_ASN1_FIND_END                            190
#define ASN1_F_ASN1_GENERALIZEDTIME_ADJ                 216
#define ASN1_F_ASN1_GENERALIZEDTIME_SET                 185
#define ASN1_F_ASN1_GENERATE_V3                         178
#define ASN1_F_ASN1_GET_OBJECT                          114
#define ASN1_F_ASN1_HEADER_NEW                          115
```

```
#define ASN1_F_ASN1_I2D_BIO                             116
#define ASN1_F_ASN1_I2D_FP                              117
#define ASN1_F_ASN1_INTEGER_SET                         118
#define ASN1_F_ASN1_INTEGER_TO_BN                       119
#define ASN1_F_ASN1_ITEM_D2I_FP                         206
#define ASN1_F_ASN1_ITEM_DUP                            191
#define ASN1_F_ASN1_ITEM_EX_COMBINE_NEW                 121
#define ASN1_F_ASN1_ITEM_EX_D2I                         120
#define ASN1_F_ASN1_ITEM_I2D_BIO                        192
#define ASN1_F_ASN1_ITEM_I2D_FP                         193
#define ASN1_F_ASN1_ITEM_PACK                           198
#define ASN1_F_ASN1_ITEM_SIGN                           195
#define ASN1_F_ASN1_ITEM_SIGN_CTX                       220
#define ASN1_F_ASN1_ITEM_UNPACK                         199
#define ASN1_F_ASN1_ITEM_VERIFY                         197
#define ASN1_F_ASN1_MBSTRING_NCOPY                      122
#define ASN1_F_ASN1_OBJECT_NEW                          123
#define ASN1_F_ASN1_OUTPUT_DATA                         214
#define ASN1_F_ASN1_PACK_STRING                         124
#define ASN1_F_ASN1_PCTX_NEW                            205
#define ASN1_F_ASN1_PKCS5_PBE_SET                       125
#define ASN1_F_ASN1_SEQ_PACK                            126
#define ASN1_F_ASN1_SEQ_UNPACK                          127
#define ASN1_F_ASN1_SIGN                                128
#define ASN1_F_ASN1_STR2TYPE                            179
#define ASN1_F_ASN1_STRING_SET                          186
#define ASN1_F_ASN1_STRING_TABLE_ADD                    129
#define ASN1_F_ASN1_STRING_TYPE_NEW                     130
#define ASN1_F_ASN1_TEMPLATE_EX_D2I                     132
#define ASN1_F_ASN1_TEMPLATE_NEW                        133
#define ASN1_F_ASN1_TEMPLATE_NOEXP_D2I                  131
#define ASN1_F_ASN1_TIME_ADJ                            217
#define ASN1_F_ASN1_TIME_SET                            175
#define ASN1_F_ASN1_TYPE_GET_INT_OCTETSTRING            134
#define ASN1_F_ASN1_TYPE_GET_OCTETSTRING                135
#define ASN1_F_ASN1_UNPACK_STRING                       136
#define ASN1_F_ASN1_UTCTIME_ADJ                         218
#define ASN1_F_ASN1_UTCTIME_SET                         187
#define ASN1_F_ASN1_VERIFY                              137
#define ASN1_F_B64_READ_ASN1                            209
#define ASN1_F_B64_WRITE_ASN1                           210
#define ASN1_F_BIO_NEW_NDEF                             208
#define ASN1_F_BITSTR_CB                                180
#define ASN1_F_BN_TO_ASN1_ENUMERATED                    138
#define ASN1_F_BN_TO_ASN1_INTEGER                       139
#define ASN1_F_C2I_ASN1_BIT_STRING                      189
#define ASN1_F_C2I_ASN1_INTEGER                         194
#define ASN1_F_C2I_ASN1_OBJECT                          196
#define ASN1_F_COLLECT_DATA                             140
#define ASN1_F_D2I_ASN1_BIT_STRING                      141
#define ASN1_F_D2I_ASN1_BOOLEAN                         142
#define ASN1_F_D2I_ASN1_BYTES                           143
#define ASN1_F_D2I_ASN1_GENERALIZEDTIME                 144
#define ASN1_F_D2I_ASN1_HEADER                          145
#define ASN1_F_D2I_ASN1_INTEGER                         146
#define ASN1_F_D2I_ASN1_OBJECT                          147
#define ASN1_F_D2I_ASN1_SET                             148
#define ASN1_F_D2I_ASN1_TYPE_BYTES                      149
#define ASN1_F_D2I_ASN1_UINTEGER                        150
#define ASN1_F_D2I_ASN1_UTCTIME                         151
#define ASN1_F_D2I_AUTOPRIVATEKEY                       207
#define ASN1_F_D2I_NETSCAPE_RSA                         152
#define ASN1_F_D2I_NETSCAPE_RSA_2                       153
#define ASN1_F_D2I_PRIVATEKEY                           154
#define ASN1_F_D2I_PUBLICKEY                            155
#define ASN1_F_D2I_RSA_NET                              200
```



```
#define ASN1_F_D2I_RSA_NET_2                            201
#define ASN1_F_D2I_X509                                 156
#define ASN1_F_D2I_X509_CINF                            157
#define ASN1_F_D2I_X509_PKEY                            159
#define ASN1_F_I2D_ASN1_BIO_STREAM                      211
#define ASN1_F_I2D_ASN1_SET                             188
#define ASN1_F_I2D_ASN1_TIME                            160
#define ASN1_F_I2D_DSA_PUBKEY                           161
#define ASN1_F_I2D_EC_PUBKEY                            181
#define ASN1_F_I2D_PRIVATEKEY                           163
#define ASN1_F_I2D_PUBLICKEY                            164
#define ASN1_F_I2D_RSA_NET                              162
#define ASN1_F_I2D_RSA_PUBKEY                           165
#define ASN1_F_LONG_C2I                                 166
#define ASN1_F_OID_MODULE_INIT                          174
#define ASN1_F_PARSE_TAGGING                            182
#define ASN1_F_PKCS5_PBE2_SET_IV                        167
#define ASN1_F_PKCS5_PBE_SET                            202
#define ASN1_F_PKCS5_PBE_SET0_ALGOR                     215
#define ASN1_F_PKCS5_PBKDF2_SET                         219
#define ASN1_F_SMIME_READ_ASN1                          212
#define ASN1_F_SMIME_TEXT                               213
#define ASN1_F_X509_CINF_NEW                            168
#define ASN1_F_X509_CRL_ADD0_REVOKED                    169
#define ASN1_F_X509_INFO_NEW                            170
#define ASN1_F_X509_NAME_ENCODE                         203
#define ASN1_F_X509_NAME_EX_D2I                         158
#define ASN1_F_X509_NAME_EX_NEW                         171
#define ASN1_F_X509_NEW                                 172
#define ASN1_F_X509_PKEY_NEW                            173

/* Reason codes. */
#define ASN1_R_ADDING_OBJECT                            171
#define ASN1_R_ASN1_PARSE_ERROR                         203
#define ASN1_R_ASN1_SIG_PARSE_ERROR                     204
#define ASN1_R_AUX_ERROR                                100
#define ASN1_R_BAD_CLASS                                101
#define ASN1_R_BAD_OBJECT_HEADER                        102
#define ASN1_R_BAD_PASSWORD_READ                        103
#define ASN1_R_BAD_TAG                                  104
#define ASN1_R_BMPSTRING_IS_WRONG_LENGTH                214
#define ASN1_R_BN_LIB                                   105
#define ASN1_R_BOOLEAN_IS_WRONG_LENGTH                  106
#define ASN1_R_BUFFER_TOO_SMALL                         107
#define ASN1_R_CIPHER_HAS_NO_OBJECT_IDENTIFIER          108
#define ASN1_R_CONTEXT_NOT_INITIALISED                  217
#define ASN1_R_DATA_IS_WRONG                            109
#define ASN1_R_DECODE_ERROR                             110
#define ASN1_R_DECODING_ERROR                           111
#define ASN1_R_DEPTH_EXCEEDED                           174
#define ASN1_R_DIGEST_AND_KEY_TYPE_NOT_SUPPORTED        198
#define ASN1_R_ENCODE_ERROR                             112
#define ASN1_R_ERROR_GETTING_TIME                       173
#define ASN1_R_ERROR_LOADING_SECTION                    172
#define ASN1_R_ERROR_PARSING_SET_ELEMENT                113
#define ASN1_R_ERROR_SETTING_CIPHER_PARAMS              114
#define ASN1_R_EXPECTING_AN_INTEGER                     115
#define ASN1_R_EXPECTING_AN_OBJECT                      116
#define ASN1_R_EXPECTING_A_BOOLEAN                      117
#define ASN1_R_EXPECTING_A_TIME                         118
#define ASN1_R_EXPLICIT_LENGTH_MISMATCH                 119
#define ASN1_R_EXPLICIT_TAG_NOT_CONSTRUCTED             120
#define ASN1_R_FIELD_MISSING                            121
#define ASN1_R_FIRST_NUM_TOO_LARGE                      122
#define ASN1_R_HEADER_TOO_LONG                          123
#define ASN1_R_ILLEGAL_BITSTRING_FORMAT                 175
```

```
#define ASN1_R_ILLEGAL_BOOLEAN                           176
#define ASN1_R_ILLEGAL_CHARACTERS                        124
#define ASN1_R_ILLEGAL_FORMAT                            177
#define ASN1_R_ILLEGAL_HEX                               178
#define ASN1_R_ILLEGAL_IMPLICIT_TAG                      179
#define ASN1_R_ILLEGAL_INTEGER                           180
#define ASN1_R_ILLEGAL_NESTED_TAGGING                    181
#define ASN1_R_ILLEGAL_NULL                              125
#define ASN1_R_ILLEGAL_NULL_VALUE                        182
#define ASN1_R_ILLEGAL_OBJECT                            183
#define ASN1_R_ILLEGAL_OPTIONAL_ANY                      126
#define ASN1_R_ILLEGAL_OPTIONS_ON_ITEM_TEMPLATE          170
#define ASN1_R_ILLEGAL_TAGGED_ANY                        127
#define ASN1_R_ILLEGAL_TIME_VALUE                        184
#define ASN1_R_INTEGER_NOT_ASCII_FORMAT                  185
#define ASN1_R_INTEGER_TOO_LARGE_FOR_LONG                128
#define ASN1_R_INVALID_BMPSTRING_LENGTH                  129
#define ASN1_R_INVALID_DIGIT                             130
#define ASN1_R_INVALID_MIME_TYPE                         205
#define ASN1_R_INVALID_MODIFIER                          186
#define ASN1_R_INVALID_NUMBER                            187
#define ASN1_R_INVALID_OBJECT_ENCODING                   216
#define ASN1_R_INVALID_SEPARATOR                         131
#define ASN1_R_INVALID_TIME_FORMAT                       132
#define ASN1_R_INVALID_UNIVERSALSTRING_LENGTH            133
#define ASN1_R_INVALID_UTF8STRING                        134
#define ASN1_R_IV_TOO_LARGE                              135
#define ASN1_R_LENGTH_ERROR                              136
#define ASN1_R_LIST_ERROR                                188
#define ASN1_R_MIME_NO_CONTENT_TYPE                      206
#define ASN1_R_MIME_PARSE_ERROR                          207
#define ASN1_R_MIME_SIG_PARSE_ERROR                      208
#define ASN1_R_MISSING_EOC                               137
#define ASN1_R_MISSING_SECOND_NUMBER                     138
#define ASN1_R_MISSING_VALUE                             189
#define ASN1_R_MSTRING_NOT_UNIVERSAL                     139
#define ASN1_R_MSTRING_WRONG_TAG                         140
#define ASN1_R_NESTED_ASN1_STRING                        197
#define ASN1_R_NON_HEX_CHARACTERS                        141
#define ASN1_R_NOT_ASCII_FORMAT                          190
#define ASN1_R_NOT_ENOUGH_DATA                           142
#define ASN1_R_NO_CONTENT_TYPE                           209
#define ASN1_R_NO_DEFAULT_DIGEST                         201
#define ASN1_R_NO_MATCHING_CHOICE_TYPE                   143
#define ASN1_R_NO_MULTIPART_BODY_FAILURE                 210
#define ASN1_R_NO_MULTIPART_BOUNDARY                     211
#define ASN1_R_NO_SIG_CONTENT_TYPE                       212
#define ASN1_R_NULL_IS_WRONG_LENGTH                      144
#define ASN1_R_OBJECT_NOT_ASCII_FORMAT                   191
#define ASN1_R_ODD_NUMBER_OF_CHARS                       145
#define ASN1_R_PRIVATE_KEY_HEADER_MISSING                146
#define ASN1_R_SECOND_NUMBER_TOO_LARGE                   147
#define ASN1_R_SEQUENCE_LENGTH_MISMATCH                  148
#define ASN1_R_SEQUENCE_NOT_CONSTRUCTED                  149
#define ASN1_R_SEQUENCE_OR_SET_NEEDS_CONFIG              192
#define ASN1_R_SHORT_LINE                                150
#define ASN1_R_SIG_INVALID_MIME_TYPE                     213
#define ASN1_R_STREAMING_NOT_SUPPORTED                   202
#define ASN1_R_STRING_TOO_LONG                           151
#define ASN1_R_STRING_TOO_SHORT                          152
#define ASN1_R_TAG_VALUE_TOO_HIGH                        153
#define ASN1_R_THE_ASN1_OBJECT_IDENTIFIER_IS_NOT_KNOWN_FOR_THIS_MD 154
#define ASN1_R_TIME_NOT_ASCII_FORMAT                     193
#define ASN1_R_TOO_LONG                                  155
#define ASN1_R_TYPE_NOT_CONSTRUCTED                      156
#define ASN1_R_UNABLE_TO_DECODE_RSA_KEY                  157
```



```
#define ASN1_R_UNABLE_TO_DECODE_RSA_PRIVATE_KEY          158
#define ASN1_R_UNEXPECTED_EOC                            159
#define ASN1_R_UNIVERSALSTRING_IS_WRONG_LENGTH           215
#define ASN1_R_UNKNOWN_FORMAT                            160
#define ASN1_R_UNKNOWN_MESSAGE_DIGEST_ALGORITHM          161
#define ASN1_R_UNKNOWN_OBJECT_TYPE                       162
#define ASN1_R_UNKNOWN_PUBLIC_KEY_TYPE                   163
#define ASN1_R_UNKNOWN_SIGNATURE_ALGORITHM               199
#define ASN1_R_UNKNOWN_TAG                               194
#define ASN1_R_UNKOWN_FORMAT                             195
#define ASN1_R_UNSUPPORTED_ANY_DEFINED_BY_TYPE           164
#define ASN1_R_UNSUPPORTED_CIPHER                        165
#define ASN1_R_UNSUPPORTED_ENCRYPTION_ALGORITHM          166
#define ASN1_R_UNSUPPORTED_PUBLIC_KEY_TYPE               167
#define ASN1_R_UNSUPPORTED_TYPE                          196
#define ASN1_R_WRONG_PUBLIC_KEY_TYPE                     200
#define ASN1_R_WRONG_TAG                                 168
#define ASN1_R_WRONG_TYPE                                169

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   19136 Wed Aug 13 19:51:40 2014
new/usr/src/lib/openssl/include/openssl/asn1_mac.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/asn1_mac.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_ASN1_MAC_H
#define HEADER_ASN1_MAC_H
```



```
#include <openssl/asn1.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifndef ASN1_MAC_ERR_LIB
#define ASN1_MAC_ERR_LIB        ERR_LIB_ASN1
#endif

#define ASN1_MAC_H_err(f,r,line) \
        ERR_PUT_error(ASN1_MAC_ERR_LIB,(f),(r),__FILE__,(line))

#define M_ASN1_D2I_vars(a,type,func) \
        ASN1_const_CTX c; \
        type ret=NULL; \
        \
        c.pp=(const unsigned char **)pp; \
        c.q= *(const unsigned char **)pp; \
        c.error=ERR_R_NESTED_ASN1_ERROR; \
        if ((a == NULL) || ((*a) == NULL)) \
                { if ((ret=(type)func()) == NULL) \
                        { c.line=__LINE__; goto err; } } \
        else    ret=(*a);

#define M_ASN1_D2I_Init() \
        c.p= *(const unsigned char **)pp; \
        c.max=(length == 0)?0:(c.p+length);

#define M_ASN1_D2I_Finish_2(a) \
        if (!asn1_const_Finish(&c)) \
                { c.line=__LINE__; goto err; } \
        *(const unsigned char **)pp=c.p; \
        if (a != NULL) (*a)=ret; \
        return(ret);

#define M_ASN1_D2I_Finish(a,func,e) \
        M_ASN1_D2I_Finish_2(a); \
err:\
        ASN1_MAC_H_err((e),c.error,c.line); \
        asn1_add_error(*(const unsigned char **)pp,(int)(c.q- *pp)); \
        if ((ret != NULL) && ((a == NULL) || (*a != ret))) func(ret); \
        return(NULL)

#define M_ASN1_D2I_start_sequence() \
        if (!asn1_GetSequence(&c,&length)) \
                { c.line=__LINE__; goto err; }
/* Begin reading ASN1 without a surrounding sequence */
#define M_ASN1_D2I_begin() \
        c.slen = length;

/* End reading ASN1 with no check on length */
#define M_ASN1_D2I_Finish_nolen(a, func, e) \
        *pp=c.p; \
        if (a != NULL) (*a)=ret; \
        return(ret); \
err:\
        ASN1_MAC_H_err((e),c.error,c.line); \
        asn1_add_error(*pp,(int)(c.q- *pp)); \
        if ((ret != NULL) && ((a == NULL) || (*a != ret))) func(ret); \
        return(NULL)

#define M_ASN1_D2I_end_sequence() \
        (((c.inf&1) == 0)?(c.slen <= 0): \
                (c.eos=ASN1_const_check_infinite_end(&c.p,c.slen)))
```

```
/* Don't use this with d2i_ASN1_BOOLEAN() */
#define M_ASN1_D2I_get(b, func) \
        c.q=c.p; \
        if (func(&(b),&c.p,c.slen) == NULL) \
                {c.line=__LINE__; goto err; } \
        c.slen-=(c.p-c.q);

/* Don't use this with d2i_ASN1_BOOLEAN() */
#define M_ASN1_D2I_get_x(type,b,func) \
        c.q=c.p; \
        if (((D2I_OF(type))func)(&(b),&c.p,c.slen) == NULL) \
                {c.line=__LINE__; goto err; } \
        c.slen-=(c.p-c.q);

/* use this instead () */
#define M_ASN1_D2I_get_int(b,func) \
        c.q=c.p; \
        if (func(&(b),&c.p,c.slen) < 0) \
                {c.line=__LINE__; goto err; } \
        c.slen-=(c.p-c.q);

#define M_ASN1_D2I_get_opt(b,func,type) \
        if ((c.slen != 0) && ((M_ASN1_next & (~V_ASN1_CONSTRUCTED)) \
                == (V_ASN1_UNIVERSAL|(type)))) \
                { \
                M_ASN1_D2I_get(b,func); \
                }

#define M_ASN1_D2I_get_int_opt(b,func,type) \
        if ((c.slen != 0) && ((M_ASN1_next & (~V_ASN1_CONSTRUCTED)) \
                == (V_ASN1_UNIVERSAL|(type)))) \
                { \
                M_ASN1_D2I_get_int(b,func); \
                }

#define M_ASN1_D2I_get_imp(b,func, type) \
        M_ASN1_next=(_tmp& V_ASN1_CONSTRUCTED)|type; \
        c.q=c.p; \
        if (func(&(b),&c.p,c.slen) == NULL) \
                {c.line=__LINE__; M_ASN1_next_prev = _tmp; goto err; } \
        c.slen-=(c.p-c.q);\
        M_ASN1_next_prev=_tmp;

#define M_ASN1_D2I_get_IMP_opt(b,func,tag,type) \
        if ((c.slen != 0) && ((M_ASN1_next & (~V_ASN1_CONSTRUCTED)) == \
                (V_ASN1_CONTEXT_SPECIFIC|(tag)))) \
                { \
                unsigned char _tmp = M_ASN1_next; \
                M_ASN1_D2I_get_imp(b,func, type);\
                }

#define M_ASN1_D2I_get_set(r,func,free_func) \
                M_ASN1_D2I_get_imp_set(r,func,free_func, \
                        V_ASN1_SET,V_ASN1_UNIVERSAL);

#define M_ASN1_D2I_get_set_type(type,r,func,free_func) \
                M_ASN1_D2I_get_imp_set_type(type,r,func,free_func, \
                        V_ASN1_SET,V_ASN1_UNIVERSAL);

#define M_ASN1_D2I_get_set_opt(r,func,free_func) \
        if ((c.slen != 0) && (M_ASN1_next == (V_ASN1_UNIVERSAL| \
                V_ASN1_CONSTRUCTED|V_ASN1_SET)))\
                { M_ASN1_D2I_get_set(r,func,free_func); }

#define M_ASN1_D2I_get_set_opt_type(type,r,func,free_func) \
        if ((c.slen != 0) && (M_ASN1_next == (V_ASN1_UNIVERSAL| \
```



```
                V_ASN1_CONSTRUCTED|V_ASN1_SET)))\
                { M_ASN1_D2I_get_set_type(type,r,func,free_func); }

#define M_ASN1_I2D_len_SET_opt(a,f) \
        if ((a != NULL) && (sk_num(a) != 0)) \
                M_ASN1_I2D_len_SET(a,f);

#define M_ASN1_I2D_put_SET_opt(a,f) \
        if ((a != NULL) && (sk_num(a) != 0)) \
                M_ASN1_I2D_put_SET(a,f);

#define M_ASN1_I2D_put_SEQUENCE_opt(a,f) \
        if ((a != NULL) && (sk_num(a) != 0)) \
                M_ASN1_I2D_put_SEQUENCE(a,f);

#define M_ASN1_I2D_put_SEQUENCE_opt_type(type,a,f) \
        if ((a != NULL) && (sk_##type##_num(a) != 0)) \
                M_ASN1_I2D_put_SEQUENCE_type(type,a,f);

#define M_ASN1_D2I_get_IMP_set_opt(b,func,free_func,tag) \
        if ((c.slen != 0) && \
                (M_ASN1_next == \
                (V_ASN1_CONTEXT_SPECIFIC|V_ASN1_CONSTRUCTED|(tag))))\
                { \
                M_ASN1_D2I_get_imp_set(b,func,free_func,\
                        tag,V_ASN1_CONTEXT_SPECIFIC); \
                }

#define M_ASN1_D2I_get_IMP_set_opt_type(type,b,func,free_func,tag) \
        if ((c.slen != 0) && \
                (M_ASN1_next == \
                (V_ASN1_CONTEXT_SPECIFIC|V_ASN1_CONSTRUCTED|(tag))))\
                { \
                M_ASN1_D2I_get_imp_set_type(type,b,func,free_func,\
                        tag,V_ASN1_CONTEXT_SPECIFIC); \
                }

#define M_ASN1_D2I_get_seq(r,func,free_func) \
                M_ASN1_D2I_get_imp_set(r,func,free_func,\
                        V_ASN1_SEQUENCE,V_ASN1_UNIVERSAL);

#define M_ASN1_D2I_get_seq_type(type,r,func,free_func) \
                M_ASN1_D2I_get_imp_set_type(type,r,func,free_func,\
                                        V_ASN1_SEQUENCE,V_ASN1_UNIVERSAL)

#define M_ASN1_D2I_get_seq_opt(r,func,free_func) \
        if ((c.slen != 0) && (M_ASN1_next == (V_ASN1_UNIVERSAL| \
                V_ASN1_CONSTRUCTED|V_ASN1_SEQUENCE)))\
                { M_ASN1_D2I_get_seq(r,func,free_func); }

#define M_ASN1_D2I_get_seq_opt_type(type,r,func,free_func) \
        if ((c.slen != 0) && (M_ASN1_next == (V_ASN1_UNIVERSAL| \
                V_ASN1_CONSTRUCTED|V_ASN1_SEQUENCE)))\
                { M_ASN1_D2I_get_seq_type(type,r,func,free_func); }

#define M_ASN1_D2I_get_IMP_set(r,func,free_func,x) \
                M_ASN1_D2I_get_imp_set(r,func,free_func,\
                        x,V_ASN1_CONTEXT_SPECIFIC);

#define M_ASN1_D2I_get_IMP_set_type(type,r,func,free_func,x) \
                M_ASN1_D2I_get_imp_set_type(type,r,func,free_func,\
                        x,V_ASN1_CONTEXT_SPECIFIC);

#define M_ASN1_D2I_get_imp_set(r,func,free_func,a,b) \
        c.q=c.p; \
        if (d2i_ASN1_SET(&(r),&c.p,c.slen,(char *(*)())func,\
```

```
260                 (void (*)())free_func,a,b) == NULL) \
261                 { c.line=__LINE__; goto err; } \
262         c.slen-=(c.p-c.q);

264 #define M_ASN1_D2I_get_imp_set_type(type,r,func,free_func,a,b) \
265         c.q=c.p; \
266         if (d2i_ASN1_SET_OF_##type(&(r),&c.p,c.slen,func,\
267                                    free_func,a,b) == NULL) \
268                 { c.line=__LINE__; goto err; } \
269         c.slen-=(c.p-c.q);

271 #define M_ASN1_D2I_get_set_strings(r,func,a,b) \
272         c.q=c.p; \
273         if (d2i_ASN1_STRING_SET(&(r),&c.p,c.slen,a,b) == NULL) \
274                 { c.line=__LINE__; goto err; } \
275         c.slen-=(c.p-c.q);

277 #define M_ASN1_D2I_get_EXP_opt(r,func,tag) \
278         if ((c.slen != 0L) && (M_ASN1_next == \
279                 (V_ASN1_CONSTRUCTED|V_ASN1_CONTEXT_SPECIFIC|tag))) \
280                 { \
281                 int Tinf,Ttag,Tclass; \
282                 long Tlen; \
283                 \
284                 c.q=c.p; \
285                 Tinf=ASN1_get_object(&c.p,&Tlen,&Ttag,&Tclass,c.slen); \
286                 if (Tinf & 0x80) \
287                         { c.error=ERR_R_BAD_ASN1_OBJECT_HEADER; \
288                         c.line=__LINE__; goto err; } \
289                 if (Tinf == (V_ASN1_CONSTRUCTED+1)) \
290                                         Tlen = c.slen - (c.p - c.q) - 2; \
291                 if (func(&(r),&c.p,Tlen) == NULL) \
292                         { c.line=__LINE__; goto err; } \
293                 if (Tinf == (V_ASN1_CONSTRUCTED+1)) { \
294                         Tlen = c.slen - (c.p - c.q); \
295                         if(!ASN1_const_check_infinite_end(&c.p, Tlen)) \
296                                 { c.error=ERR_R_MISSING_ASN1_EOS; \
297                                 c.line=__LINE__; goto err; } \
298                 }\
299                 c.slen-=(c.p-c.q); \
300                 }

302 #define M_ASN1_D2I_get_EXP_set_opt(r,func,free_func,tag,b) \
303         if ((c.slen != 0) && (M_ASN1_next == \
304                 (V_ASN1_CONSTRUCTED|V_ASN1_CONTEXT_SPECIFIC|tag))) \
305                 { \
306                 int Tinf,Ttag,Tclass; \
307                 long Tlen; \
308                 \
309                 c.q=c.p; \
310                 Tinf=ASN1_get_object(&c.p,&Tlen,&Ttag,&Tclass,c.slen); \
311                 if (Tinf & 0x80) \
312                         { c.error=ERR_R_BAD_ASN1_OBJECT_HEADER; \
313                         c.line=__LINE__; goto err; } \
314                 if (Tinf == (V_ASN1_CONSTRUCTED+1)) \
315                                         Tlen = c.slen - (c.p - c.q) - 2; \
316                 if (d2i_ASN1_SET(&(r),&c.p,Tlen,(char *(*)())func, \
317                         (void (*)())free_func, \
318                         b,V_ASN1_UNIVERSAL) == NULL) \
319                         { c.line=__LINE__; goto err; } \
320                 if (Tinf == (V_ASN1_CONSTRUCTED+1)) { \
321                         Tlen = c.slen - (c.p - c.q); \
322                         if(!ASN1_check_infinite_end(&c.p, Tlen)) \
323                                 { c.error=ERR_R_MISSING_ASN1_EOS; \
324                                 c.line=__LINE__; goto err; } \
325                 }\
```



```
326                 c.slen-=(c.p-c.q); \
327                 }

329 #define M_ASN1_D2I_get_EXP_set_opt_type(type,r,func,free_func,tag,b) \
330         if ((c.slen != 0) && (M_ASN1_next == \
331                 (V_ASN1_CONSTRUCTED|V_ASN1_CONTEXT_SPECIFIC|tag))) \
332                 { \
333                 int Tinf,Ttag,Tclass; \
334                 long Tlen; \
335                 \
336                 c.q=c.p; \
337                 Tinf=ASN1_get_object(&c.p,&Tlen,&Ttag,&Tclass,c.slen); \
338                 if (Tinf & 0x80) \
339                         { c.error=ERR_R_BAD_ASN1_OBJECT_HEADER; \
340                         c.line=__LINE__; goto err; } \
341                 if (Tinf == (V_ASN1_CONSTRUCTED+1)) \
342                                         Tlen = c.slen - (c.p - c.q) - 2; \
343                 if (d2i_ASN1_SET_OF_##type(&(r),&c.p,Tlen,func, \
344                         free_func,b,V_ASN1_UNIVERSAL) == NULL) \
345                         { c.line=__LINE__; goto err; } \
346                 if (Tinf == (V_ASN1_CONSTRUCTED+1)) { \
347                         Tlen = c.slen - (c.p - c.q); \
348                         if(!ASN1_check_infinite_end(&c.p, Tlen)) \
349                                 { c.error=ERR_R_MISSING_ASN1_EOS; \
350                                 c.line=__LINE__; goto err; } \
351                 }\
352                 c.slen-=(c.p-c.q); \
353                 }

355 /* New macros */
356 #define M_ASN1_New_Malloc(ret,type) \
357         if ((ret=(type *)OPENSSL_malloc(sizeof(type))) == NULL) \
358                 { c.line=__LINE__; goto err2; }

360 #define M_ASN1_New(arg,func) \
361         if (((arg)=func()) == NULL) return(NULL)

363 #define M_ASN1_New_Error(a) \
364 /*      err:    ASN1_MAC_H_err((a),ERR_R_NESTED_ASN1_ERROR,c.line); \
365                 return(NULL);*/ \
366         err2:   ASN1_MAC_H_err((a),ERR_R_MALLOC_FAILURE,c.line); \
367                 return(NULL)


370 /* BIG UGLY WARNING!  This is so damn ugly I wanna puke.  Unfortunately,
371    some macros that use ASN1_const_CTX still insist on writing in the input
372    stream.  ARGH!  ARGH!  ARGH!  Let's get rid of this macro package.
373    Please?                                          -- Richard Levitte */
374 #define M_ASN1_next             (*((unsigned char *)(c.p)))
375 #define M_ASN1_next_prev        (*((unsigned char *)(c.q)))

377 /*************************************************/

379 #define M_ASN1_I2D_vars(a)      int r=0,ret=0; \
380                                 unsigned char *p; \
381                                 if (a == NULL) return(0)

383 /* Length Macros */
384 #define M_ASN1_I2D_len(a,f)     ret+=f(a,NULL)
385 #define M_ASN1_I2D_len_IMP_opt(a,f)     if (a != NULL) M_ASN1_I2D_len(a,f)

387 #define M_ASN1_I2D_len_SET(a,f) \
388                 ret+=i2d_ASN1_SET(a,NULL,f,V_ASN1_SET,V_ASN1_UNIVERSAL,IS_SET);

390 #define M_ASN1_I2D_len_SET_type(type,a,f) \
391                 ret+=i2d_ASN1_SET_OF_##type(a,NULL,f,V_ASN1_SET, \
```

```
392                                         V_ASN1_UNIVERSAL,IS_SET);

394 #define M_ASN1_I2D_len_SEQUENCE(a,f) \
395                 ret+=i2d_ASN1_SET(a,NULL,f,V_ASN1_SEQUENCE,V_ASN1_UNIVERSAL, \
396                                   IS_SEQUENCE);

398 #define M_ASN1_I2D_len_SEQUENCE_type(type,a,f) \
399                 ret+=i2d_ASN1_SET_OF_##type(a,NULL,f,V_ASN1_SEQUENCE, \
400                                             V_ASN1_UNIVERSAL,IS_SEQUENCE)

402 #define M_ASN1_I2D_len_SEQUENCE_opt(a,f) \
403                 if ((a != NULL) && (sk_num(a) != 0)) \
404                         M_ASN1_I2D_len_SEQUENCE(a,f);

406 #define M_ASN1_I2D_len_SEQUENCE_opt_type(type,a,f) \
407                 if ((a != NULL) && (sk_##type##_num(a) != 0)) \
408                         M_ASN1_I2D_len_SEQUENCE_type(type,a,f);

410 #define M_ASN1_I2D_len_IMP_SET(a,f,x) \
411                 ret+=i2d_ASN1_SET(a,NULL,f,x,V_ASN1_CONTEXT_SPECIFIC,IS_SET);

413 #define M_ASN1_I2D_len_IMP_SET_type(type,a,f,x) \
414                 ret+=i2d_ASN1_SET_OF_##type(a,NULL,f,x, \
415                                             V_ASN1_CONTEXT_SPECIFIC,IS_SET);

417 #define M_ASN1_I2D_len_IMP_SET_opt(a,f,x) \
418                 if ((a != NULL) && (sk_num(a) != 0)) \
419                         ret+=i2d_ASN1_SET(a,NULL,f,x,V_ASN1_CONTEXT_SPECIFIC, \
420                                           IS_SET);

422 #define M_ASN1_I2D_len_IMP_SET_opt_type(type,a,f,x) \
423                 if ((a != NULL) && (sk_##type##_num(a) != 0)) \
424                         ret+=i2d_ASN1_SET_OF_##type(a,NULL,f,x, \
425                                                     V_ASN1_CONTEXT_SPECIFIC,IS_SET);

427 #define M_ASN1_I2D_len_IMP_SEQUENCE(a,f,x) \
428                 ret+=i2d_ASN1_SET(a,NULL,f,x,V_ASN1_CONTEXT_SPECIFIC, \
429                                   IS_SEQUENCE);

431 #define M_ASN1_I2D_len_IMP_SEQUENCE_opt(a,f,x) \
432                 if ((a != NULL) && (sk_num(a) != 0)) \
433                         ret+=i2d_ASN1_SET(a,NULL,f,x,V_ASN1_CONTEXT_SPECIFIC, \
434                                           IS_SEQUENCE);

436 #define M_ASN1_I2D_len_IMP_SEQUENCE_opt_type(type,a,f,x) \
437                 if ((a != NULL) && (sk_##type##_num(a) != 0)) \
438                         ret+=i2d_ASN1_SET_OF_##type(a,NULL,f,x, \
439                                                     V_ASN1_CONTEXT_SPECIFIC, \
440                                                     IS_SEQUENCE);

442 #define M_ASN1_I2D_len_EXP_opt(a,f,mtag,v) \
443                 if (a != NULL)\
444                         { \
445                         v=f(a,NULL); \
446                         ret+=ASN1_object_size(1,v,mtag); \
447                         }

449 #define M_ASN1_I2D_len_EXP_SET_opt(a,f,mtag,tag,v) \
450                 if ((a != NULL) && (sk_num(a) != 0))\
451                         { \
452                         v=i2d_ASN1_SET(a,NULL,f,tag,V_ASN1_UNIVERSAL,IS_SET); \
453                         ret+=ASN1_object_size(1,v,mtag); \
454                         }

456 #define M_ASN1_I2D_len_EXP_SEQUENCE_opt(a,f,mtag,tag,v) \
457                 if ((a != NULL) && (sk_num(a) != 0))\
```



```
458                         { \
459                         v=i2d_ASN1_SET(a,NULL,f,tag,V_ASN1_UNIVERSAL, \
460                                        IS_SEQUENCE); \
461                         ret+=ASN1_object_size(1,v,mtag); \
462                         }

464 #define M_ASN1_I2D_len_EXP_SEQUENCE_opt_type(type,a,f,mtag,tag,v) \
465                 if ((a != NULL) && (sk_##type##_num(a) != 0))\
466                         { \
467                         v=i2d_ASN1_SET_OF_##type(a,NULL,f,tag, \
468                                                  V_ASN1_UNIVERSAL, \
469                                                  IS_SEQUENCE); \
470                         ret+=ASN1_object_size(1,v,mtag); \
471                         }

473 /* Put Macros */
474 #define M_ASN1_I2D_put(a,f)     f(a,&p)

476 #define M_ASN1_I2D_put_IMP_opt(a,f,t)   \
477                 if (a != NULL) \
478                         { \
479                         unsigned char *q=p; \
480                         f(a,&p); \
481                         *q=(V_ASN1_CONTEXT_SPECIFIC|t|(*q&V_ASN1_CONSTRUCTED));\
482                         }

484 #define M_ASN1_I2D_put_SET(a,f) i2d_ASN1_SET(a,&p,f,V_ASN1_SET,\
485                         V_ASN1_UNIVERSAL,IS_SET)
486 #define M_ASN1_I2D_put_SET_type(type,a,f) \
487      i2d_ASN1_SET_OF_##type(a,&p,f,V_ASN1_SET,V_ASN1_UNIVERSAL,IS_SET)
488 #define M_ASN1_I2D_put_IMP_SET(a,f,x) i2d_ASN1_SET(a,&p,f,x,\
489                         V_ASN1_CONTEXT_SPECIFIC,IS_SET)
490 #define M_ASN1_I2D_put_IMP_SET_type(type,a,f,x) \
491      i2d_ASN1_SET_OF_##type(a,&p,f,x,V_ASN1_CONTEXT_SPECIFIC,IS_SET)
492 #define M_ASN1_I2D_put_IMP_SEQUENCE(a,f,x) i2d_ASN1_SET(a,&p,f,x,\
493                         V_ASN1_CONTEXT_SPECIFIC,IS_SEQUENCE)

495 #define M_ASN1_I2D_put_SEQUENCE(a,f) i2d_ASN1_SET(a,&p,f,V_ASN1_SEQUENCE,\
496                                         V_ASN1_UNIVERSAL,IS_SEQUENCE)

498 #define M_ASN1_I2D_put_SEQUENCE_type(type,a,f) \
499      i2d_ASN1_SET_OF_##type(a,&p,f,V_ASN1_SEQUENCE,V_ASN1_UNIVERSAL, \
500                             IS_SEQUENCE)

502 #define M_ASN1_I2D_put_SEQUENCE_opt(a,f) \
503                 if ((a != NULL) && (sk_num(a) != 0)) \
504                         M_ASN1_I2D_put_SEQUENCE(a,f);

506 #define M_ASN1_I2D_put_IMP_SET_opt(a,f,x) \
507                 if ((a != NULL) && (sk_num(a) != 0)) \
508                         { i2d_ASN1_SET(a,&p,f,x,V_ASN1_CONTEXT_SPECIFIC, \
509                                        IS_SET); }

511 #define M_ASN1_I2D_put_IMP_SET_opt_type(type,a,f,x) \
512                 if ((a != NULL) && (sk_##type##_num(a) != 0)) \
513                         { i2d_ASN1_SET_OF_##type(a,&p,f,x, \
514                                                  V_ASN1_CONTEXT_SPECIFIC, \
515                                                  IS_SET); }

517 #define M_ASN1_I2D_put_IMP_SEQUENCE_opt(a,f,x) \
518                 if ((a != NULL) && (sk_num(a) != 0)) \
519                         { i2d_ASN1_SET(a,&p,f,x,V_ASN1_CONTEXT_SPECIFIC, \
520                                        IS_SEQUENCE); }

522 #define M_ASN1_I2D_put_IMP_SEQUENCE_opt_type(type,a,f,x) \
523                 if ((a != NULL) && (sk_##type##_num(a) != 0)) \
```

```
524                         { i2d_ASN1_SET_OF_##type(a,&p,f,x, \
525                                                  V_ASN1_CONTEXT_SPECIFIC, \
526                                                  IS_SEQUENCE); }

528 #define M_ASN1_I2D_put_EXP_opt(a,f,tag,v) \
529                 if (a != NULL) \
530                         { \
531                         ASN1_put_object(&p,1,v,tag,V_ASN1_CONTEXT_SPECIFIC); \
532                         f(a,&p); \
533                         }

535 #define M_ASN1_I2D_put_EXP_SET_opt(a,f,mtag,tag,v) \
536                 if ((a != NULL) && (sk_num(a) != 0)) \
537                         { \
538                         ASN1_put_object(&p,1,v,mtag,V_ASN1_CONTEXT_SPECIFIC); \
539                         i2d_ASN1_SET(a,&p,f,tag,V_ASN1_UNIVERSAL,IS_SET); \
540                         }

542 #define M_ASN1_I2D_put_EXP_SEQUENCE_opt(a,f,mtag,tag,v) \
543                 if ((a != NULL) && (sk_num(a) != 0)) \
544                         { \
545                         ASN1_put_object(&p,1,v,mtag,V_ASN1_CONTEXT_SPECIFIC); \
546                         i2d_ASN1_SET(a,&p,f,tag,V_ASN1_UNIVERSAL,IS_SEQUENCE); \
547                         }

549 #define M_ASN1_I2D_put_EXP_SEQUENCE_opt_type(type,a,f,mtag,tag,v) \
550                 if ((a != NULL) && (sk_##type##_num(a) != 0)) \
551                         { \
552                         ASN1_put_object(&p,1,v,mtag,V_ASN1_CONTEXT_SPECIFIC); \
553                         i2d_ASN1_SET_OF_##type(a,&p,f,tag,V_ASN1_UNIVERSAL, \
554                                                IS_SEQUENCE); \
555                         }

557 #define M_ASN1_I2D_seq_total() \
558                 r=ASN1_object_size(1,ret,V_ASN1_SEQUENCE); \
559                 if (pp == NULL) return(r); \
560                 p= *pp; \
561                 ASN1_put_object(&p,1,ret,V_ASN1_SEQUENCE,V_ASN1_UNIVERSAL)

563 #define M_ASN1_I2D_INF_seq_start(tag,ctx) \
564                 *(p++)=(V_ASN1_CONSTRUCTED|(tag)|(ctx)); \
565                 *(p++)=0x80

567 #define M_ASN1_I2D_INF_seq_end() *(p++)=0x00; *(p++)=0x00

569 #define M_ASN1_I2D_finish()     *pp=p; \
570                                 return(r);

572 int asn1_GetSequence(ASN1_const_CTX *c, long *length);
573 void asn1_add_error(const unsigned char *address,int offset);
574 #ifdef  __cplusplus
575 }
576 #endif

578 #endif
579 #endif /* ! codereview */
```

```
**********************************************************
   30069 Wed Aug 13 19:51:40 2014
new/usr/src/lib/openssl/include/openssl/asn1t.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* asn1t.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#ifndef HEADER_ASN1T_H
#define HEADER_ASN1T_H

#include <stddef.h>
```



```
#include <openssl/e_os2.h>
#include <openssl/asn1.h>

#ifdef OPENSSL_BUILD_SHLIBCRYPTO
# undef OPENSSL_EXTERN
# define OPENSSL_EXTERN OPENSSL_EXPORT
#endif

/* ASN1 template defines, structures and functions */

#ifdef  __cplusplus
extern "C" {
#endif


#ifndef OPENSSL_EXPORT_VAR_AS_FUNCTION

/* Macro to obtain ASN1_ADB pointer from a type (only used internally) */
#define ASN1_ADB_ptr(iptr) ((const ASN1_ADB *)(iptr))


/* Macros for start and end of ASN1_ITEM definition */

#define ASN1_ITEM_start(itname) \
        OPENSSL_GLOBAL const ASN1_ITEM itname##_it = {

#define ASN1_ITEM_end(itname) \
                };

#else

/* Macro to obtain ASN1_ADB pointer from a type (only used internally) */
#define ASN1_ADB_ptr(iptr) ((const ASN1_ADB *)(iptr()))


/* Macros for start and end of ASN1_ITEM definition */

#define ASN1_ITEM_start(itname) \
        const ASN1_ITEM * itname##_it(void) \
        { \
                static const ASN1_ITEM local_it = {

#define ASN1_ITEM_end(itname) \
                }; \
        return &local_it; \
        }

#endif


/* Macros to aid ASN1 template writing */

#define ASN1_ITEM_TEMPLATE(tname) \
        static const ASN1_TEMPLATE tname##_item_tt

#define ASN1_ITEM_TEMPLATE_END(tname) \
        ;\
        ASN1_ITEM_start(tname) \
                ASN1_ITYPE_PRIMITIVE,\
                -1,\
                &tname##_item_tt,\
                0,\
                NULL,\
                0,\
                #tname \
        ASN1_ITEM_end(tname)
```

```
130 /* This is a ASN1 type which just embeds a template */

132 /* This pair helps declare a SEQUENCE. We can do:
133  *
134  *      ASN1_SEQUENCE(stname) = {
135  *              ... SEQUENCE components ...
136  *      } ASN1_SEQUENCE_END(stname)
137  *
138  *      This will produce an ASN1_ITEM called stname_it
139  *      for a structure called stname.
140  *
141  *      If you want the same structure but a different
142  *      name then use:
143  *
144  *      ASN1_SEQUENCE(itname) = {
145  *              ... SEQUENCE components ...
146  *      } ASN1_SEQUENCE_END_name(stname, itname)
147  *
148  *      This will create an item called itname_it using
149  *      a structure called stname.
150  */

152 #define ASN1_SEQUENCE(tname) \
153         static const ASN1_TEMPLATE tname##_seq_tt[]

155 #define ASN1_SEQUENCE_END(stname) ASN1_SEQUENCE_END_name(stname, stname)

157 #define ASN1_SEQUENCE_END_name(stname, tname) \
158         ;\
159         ASN1_ITEM_start(tname) \
160                 ASN1_ITYPE_SEQUENCE,\
161                 V_ASN1_SEQUENCE,\
162                 tname##_seq_tt,\
163                 sizeof(tname##_seq_tt) / sizeof(ASN1_TEMPLATE),\
164                 NULL,\
165                 sizeof(stname),\
166                 #stname \
167         ASN1_ITEM_end(tname)

169 #define ASN1_NDEF_SEQUENCE(tname) \
170         ASN1_SEQUENCE(tname)

172 #define ASN1_NDEF_SEQUENCE_cb(tname, cb) \
173         ASN1_SEQUENCE_cb(tname, cb)

175 #define ASN1_SEQUENCE_cb(tname, cb) \
176         static const ASN1_AUX tname##_aux = {NULL, 0, 0, 0, cb, 0}; \
177         ASN1_SEQUENCE(tname)

179 #define ASN1_BROKEN_SEQUENCE(tname) \
180         static const ASN1_AUX tname##_aux = {NULL, ASN1_AFLG_BROKEN, 0, 0, 0, 0}
181         ASN1_SEQUENCE(tname)

183 #define ASN1_SEQUENCE_ref(tname, cb, lck) \
184         static const ASN1_AUX tname##_aux = {NULL, ASN1_AFLG_REFCOUNT, offsetof(
185         ASN1_SEQUENCE(tname)

187 #define ASN1_SEQUENCE_enc(tname, enc, cb) \
188         static const ASN1_AUX tname##_aux = {NULL, ASN1_AFLG_ENCODING, 0, 0, cb,
189         ASN1_SEQUENCE(tname)

191 #define ASN1_NDEF_SEQUENCE_END(tname) \
192         ;\
193         ASN1_ITEM_start(tname) \
```



```
194                 ASN1_ITYPE_NDEF_SEQUENCE,\
195                 V_ASN1_SEQUENCE,\
196                 tname##_seq_tt,\
197                 sizeof(tname##_seq_tt) / sizeof(ASN1_TEMPLATE),\
198                 NULL,\
199                 sizeof(tname),\
200                 #tname \
201         ASN1_ITEM_end(tname)

203 #define ASN1_BROKEN_SEQUENCE_END(stname) ASN1_SEQUENCE_END_ref(stname, stname)

205 #define ASN1_SEQUENCE_END_enc(stname, tname) ASN1_SEQUENCE_END_ref(stname, tname

207 #define ASN1_SEQUENCE_END_cb(stname, tname) ASN1_SEQUENCE_END_ref(stname, tname)

209 #define ASN1_SEQUENCE_END_ref(stname, tname) \
210         ;\
211         ASN1_ITEM_start(tname) \
212                 ASN1_ITYPE_SEQUENCE,\
213                 V_ASN1_SEQUENCE,\
214                 tname##_seq_tt,\
215                 sizeof(tname##_seq_tt) / sizeof(ASN1_TEMPLATE),\
216                 &tname##_aux,\
217                 sizeof(stname),\
218                 #stname \
219         ASN1_ITEM_end(tname)

221 #define ASN1_NDEF_SEQUENCE_END_cb(stname, tname) \
222         ;\
223         ASN1_ITEM_start(tname) \
224                 ASN1_ITYPE_NDEF_SEQUENCE,\
225                 V_ASN1_SEQUENCE,\
226                 tname##_seq_tt,\
227                 sizeof(tname##_seq_tt) / sizeof(ASN1_TEMPLATE),\
228                 &tname##_aux,\
229                 sizeof(stname),\
230                 #stname \
231         ASN1_ITEM_end(tname)


234 /* This pair helps declare a CHOICE type. We can do:
235  *
236  *      ASN1_CHOICE(chname) = {
237  *              ... CHOICE options ...
238  *      ASN1_CHOICE_END(chname)
239  *
240  *      This will produce an ASN1_ITEM called chname_it
241  *      for a structure called chname. The structure
242  *      definition must look like this:
243  *      typedef struct {
244  *              int type;
245  *              union {
246  *                      ASN1_SOMETHING *opt1;
247  *                      ASN1_SOMEOTHER *opt2;
248  *              } value;
249  *      } chname;
250  *
251  *      the name of the selector must be 'type'.
252  *      to use an alternative selector name use the
253  *      ASN1_CHOICE_END_selector() version.
254  */

256 #define ASN1_CHOICE(tname) \
257         static const ASN1_TEMPLATE tname##_ch_tt[]

259 #define ASN1_CHOICE_cb(tname, cb) \
```

```
260         static const ASN1_AUX tname##_aux = {NULL, 0, 0, 0, cb, 0}; \
261         ASN1_CHOICE(tname)

263 #define ASN1_CHOICE_END(stname) ASN1_CHOICE_END_name(stname, stname)

265 #define ASN1_CHOICE_END_name(stname, tname) ASN1_CHOICE_END_selector(stname, tna

267 #define ASN1_CHOICE_END_selector(stname, tname, selname) \
268         ;\
269         ASN1_ITEM_start(tname) \
270                 ASN1_ITYPE_CHOICE,\
271                 offsetof(stname,selname) ,\
272                 tname##_ch_tt,\
273                 sizeof(tname##_ch_tt) / sizeof(ASN1_TEMPLATE),\
274                 NULL,\
275                 sizeof(stname),\
276                 #stname \
277         ASN1_ITEM_end(tname)

279 #define ASN1_CHOICE_END_cb(stname, tname, selname) \
280         ;\
281         ASN1_ITEM_start(tname) \
282                 ASN1_ITYPE_CHOICE,\
283                 offsetof(stname,selname) ,\
284                 tname##_ch_tt,\
285                 sizeof(tname##_ch_tt) / sizeof(ASN1_TEMPLATE),\
286                 &tname##_aux,\
287                 sizeof(stname),\
288                 #stname \
289         ASN1_ITEM_end(tname)

291 /* This helps with the template wrapper form of ASN1_ITEM */

293 #define ASN1_EX_TEMPLATE_TYPE(flags, tag, name, type) { \
294         (flags), (tag), 0,\
295         #name, ASN1_ITEM_ref(type) }

297 /* These help with SEQUENCE or CHOICE components */

299 /* used to declare other types */

301 #define ASN1_EX_TYPE(flags, tag, stname, field, type) { \
302         (flags), (tag), offsetof(stname, field),\
303         #field, ASN1_ITEM_ref(type) }

305 /* used when the structure is combined with the parent */

307 #define ASN1_EX_COMBINE(flags, tag, type) { \
308         (flags)|ASN1_TFLG_COMBINE, (tag), 0, NULL, ASN1_ITEM_ref(type) }

310 /* implicit and explicit helper macros */

312 #define ASN1_IMP_EX(stname, field, type, tag, ex) \
313                 ASN1_EX_TYPE(ASN1_TFLG_IMPLICIT | ex, tag, stname, field, type)

315 #define ASN1_EXP_EX(stname, field, type, tag, ex) \
316                 ASN1_EX_TYPE(ASN1_TFLG_EXPLICIT | ex, tag, stname, field, type)

318 /* Any defined by macros: the field used is in the table itself */

320 #ifndef OPENSSL_EXPORT_VAR_AS_FUNCTION
321 #define ASN1_ADB_OBJECT(tblname) { ASN1_TFLG_ADB_OID, -1, 0, #tblname, (const AS
322 #define ASN1_ADB_INTEGER(tblname) { ASN1_TFLG_ADB_INT, -1, 0, #tblname, (const A
323 #else
324 #define ASN1_ADB_OBJECT(tblname) { ASN1_TFLG_ADB_OID, -1, 0, #tblname, tblname##
325 #define ASN1_ADB_INTEGER(tblname) { ASN1_TFLG_ADB_INT, -1, 0, #tblname, tblname#
```



```
326 #endif
327 /* Plain simple type */
328 #define ASN1_SIMPLE(stname, field, type) ASN1_EX_TYPE(0,0, stname, field, type)

330 /* OPTIONAL simple type */
331 #define ASN1_OPT(stname, field, type) ASN1_EX_TYPE(ASN1_TFLG_OPTIONAL, 0, stname

333 /* IMPLICIT tagged simple type */
334 #define ASN1_IMP(stname, field, type, tag) ASN1_IMP_EX(stname, field, type, tag,

336 /* IMPLICIT tagged OPTIONAL simple type */
337 #define ASN1_IMP_OPT(stname, field, type, tag) ASN1_IMP_EX(stname, field, type,

339 /* Same as above but EXPLICIT */

341 #define ASN1_EXP(stname, field, type, tag) ASN1_EXP_EX(stname, field, type, tag,
342 #define ASN1_EXP_OPT(stname, field, type, tag) ASN1_EXP_EX(stname, field, type,

344 /* SEQUENCE OF type */
345 #define ASN1_SEQUENCE_OF(stname, field, type) \
346                 ASN1_EX_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, stname, field, type)

348 /* OPTIONAL SEQUENCE OF */
349 #define ASN1_SEQUENCE_OF_OPT(stname, field, type) \
350                 ASN1_EX_TYPE(ASN1_TFLG_SEQUENCE_OF|ASN1_TFLG_OPTIONAL, 0, stname

352 /* Same as above but for SET OF */

354 #define ASN1_SET_OF(stname, field, type) \
355                 ASN1_EX_TYPE(ASN1_TFLG_SET_OF, 0, stname, field, type)

357 #define ASN1_SET_OF_OPT(stname, field, type) \
358                 ASN1_EX_TYPE(ASN1_TFLG_SET_OF|ASN1_TFLG_OPTIONAL, 0, stname, fie

360 /* Finally compound types of SEQUENCE, SET, IMPLICIT, EXPLICIT and OPTIONAL */

362 #define ASN1_IMP_SET_OF(stname, field, type, tag) \
363                         ASN1_IMP_EX(stname, field, type, tag, ASN1_TFLG_SET_OF)

365 #define ASN1_EXP_SET_OF(stname, field, type, tag) \
366                         ASN1_EXP_EX(stname, field, type, tag, ASN1_TFLG_SET_OF)

368 #define ASN1_IMP_SET_OF_OPT(stname, field, type, tag) \
369                         ASN1_IMP_EX(stname, field, type, tag, ASN1_TFLG_SET_OF|A

371 #define ASN1_EXP_SET_OF_OPT(stname, field, type, tag) \
372                         ASN1_EXP_EX(stname, field, type, tag, ASN1_TFLG_SET_OF|A

374 #define ASN1_IMP_SEQUENCE_OF(stname, field, type, tag) \
375                         ASN1_IMP_EX(stname, field, type, tag, ASN1_TFLG_SEQUENCE

377 #define ASN1_IMP_SEQUENCE_OF_OPT(stname, field, type, tag) \
378                         ASN1_IMP_EX(stname, field, type, tag, ASN1_TFLG_SEQUENCE

380 #define ASN1_EXP_SEQUENCE_OF(stname, field, type, tag) \
381                         ASN1_EXP_EX(stname, field, type, tag, ASN1_TFLG_SEQUENCE

383 #define ASN1_EXP_SEQUENCE_OF_OPT(stname, field, type, tag) \
384                         ASN1_EXP_EX(stname, field, type, tag, ASN1_TFLG_SEQUENCE

386 /* EXPLICIT using indefinite length constructed form */
387 #define ASN1_NDEF_EXP(stname, field, type, tag) \
388                         ASN1_EXP_EX(stname, field, type, tag, ASN1_TFLG_NDEF)

390 /* EXPLICIT OPTIONAL using indefinite length constructed form */
391 #define ASN1_NDEF_EXP_OPT(stname, field, type, tag) \
```

```
392                         ASN1_EXP_EX(stname, field, type, tag, ASN1_TFLG_OPTIONAL

394 /* Macros for the ASN1_ADB structure */

396 #define ASN1_ADB(name) \
397         static const ASN1_ADB_TABLE name##_adbtbl[]

399 #ifndef OPENSSL_EXPORT_VAR_AS_FUNCTION

401 #define ASN1_ADB_END(name, flags, field, app_table, def, none) \
402         ;\
403         static const ASN1_ADB name##_adb = {\
404                 flags,\
405                 offsetof(name, field),\
406                 app_table,\
407                 name##_adbtbl,\
408                 sizeof(name##_adbtbl) / sizeof(ASN1_ADB_TABLE),\
409                 def,\
410                 none\
411         }

413 #else

415 #define ASN1_ADB_END(name, flags, field, app_table, def, none) \
416         ;\
417         static const ASN1_ITEM *name##_adb(void) \
418         { \
419         static const ASN1_ADB internal_adb = \
420                 {\
421                 flags,\
422                 offsetof(name, field),\
423                 app_table,\
424                 name##_adbtbl,\
425                 sizeof(name##_adbtbl) / sizeof(ASN1_ADB_TABLE),\
426                 def,\
427                 none\
428                 }; \
429                 return (const ASN1_ITEM *) &internal_adb; \
430         } \
431         void dummy_function(void)

433 #endif

435 #define ADB_ENTRY(val, template) {val, template}

437 #define ASN1_ADB_TEMPLATE(name) \
438         static const ASN1_TEMPLATE name##_tt

440 /* This is the ASN1 template structure that defines
441  * a wrapper round the actual type. It determines the
442  * actual position of the field in the value structure,
443  * various flags such as OPTIONAL and the field name.
444  */

446 struct ASN1_TEMPLATE_st {
447 unsigned long flags;            /* Various flags */
448 long tag;                       /* tag, not used if no tagging */
449 unsigned long offset;           /* Offset of this field in structure */
450 #ifndef NO_ASN1_FIELD_NAMES
451 const char *field_name;         /* Field name */
452 #endif
453 ASN1_ITEM_EXP *item;            /* Relevant ASN1_ITEM or ASN1_ADB */
454 };

456 /* Macro to extract ASN1_ITEM and ASN1_ADB pointer from ASN1_TEMPLATE */
```



```
458 #define ASN1_TEMPLATE_item(t) (t->item_ptr)
459 #define ASN1_TEMPLATE_adb(t) (t->item_ptr)

461 typedef struct ASN1_ADB_TABLE_st ASN1_ADB_TABLE;
462 typedef struct ASN1_ADB_st ASN1_ADB;

464 struct ASN1_ADB_st {
465         unsigned long flags;    /* Various flags */
466         unsigned long offset;   /* Offset of selector field */
467         STACK_OF(ASN1_ADB_TABLE) **app_items; /* Application defined items */
468         const ASN1_ADB_TABLE *tbl;      /* Table of possible types */
469         long tblcount;          /* Number of entries in tbl */
470         const ASN1_TEMPLATE *default_tt;  /* Type to use if no match */
471         const ASN1_TEMPLATE *null_tt;  /* Type to use if selector is NULL */
472 };

474 struct ASN1_ADB_TABLE_st {
475         long value;             /* NID for an object or value for an int */
476         const ASN1_TEMPLATE tt;         /* item for this value */
477 };

479 /* template flags */

481 /* Field is optional */
482 #define ASN1_TFLG_OPTIONAL      (0x1)

484 /* Field is a SET OF */
485 #define ASN1_TFLG_SET_OF        (0x1 << 1)

487 /* Field is a SEQUENCE OF */
488 #define ASN1_TFLG_SEQUENCE_OF   (0x2 << 1)

490 /* Special case: this refers to a SET OF that
491  * will be sorted into DER order when encoded *and*
492  * the corresponding STACK will be modified to match
493  * the new order.
494  */
495 #define ASN1_TFLG_SET_ORDER     (0x3 << 1)

497 /* Mask for SET OF or SEQUENCE OF */
498 #define ASN1_TFLG_SK_MASK       (0x3 << 1)

500 /* These flags mean the tag should be taken from the
501  * tag field. If EXPLICIT then the underlying type
502  * is used for the inner tag.
503  */

505 /* IMPLICIT tagging */
506 #define ASN1_TFLG_IMPTAG        (0x1 << 3)


509 /* EXPLICIT tagging, inner tag from underlying type */
510 #define ASN1_TFLG_EXPTAG        (0x2 << 3)

512 #define ASN1_TFLG_TAG_MASK      (0x3 << 3)

514 /* context specific IMPLICIT */
515 #define ASN1_TFLG_IMPLICIT      ASN1_TFLG_IMPTAG|ASN1_TFLG_CONTEXT

517 /* context specific EXPLICIT */
518 #define ASN1_TFLG_EXPLICIT      ASN1_TFLG_EXPTAG|ASN1_TFLG_CONTEXT

520 /* If tagging is in force these determine the
521  * type of tag to use. Otherwise the tag is
522  * determined by the underlying type. These
523  * values reflect the actual octet format.
```

```
 */

/* Universal tag */
#define ASN1_TFLG_UNIVERSAL     (0x0<<6)
/* Application tag */
#define ASN1_TFLG_APPLICATION   (0x1<<6)
/* Context specific tag */
#define ASN1_TFLG_CONTEXT       (0x2<<6)
/* Private tag */
#define ASN1_TFLG_PRIVATE       (0x3<<6)

#define ASN1_TFLG_TAG_CLASS     (0x3<<6)

/* These are for ANY DEFINED BY type. In this case
 * the 'item' field points to an ASN1_ADB structure
 * which contains a table of values to decode the
 * relevant type
 */

#define ASN1_TFLG_ADB_MASK      (0x3<<8)

#define ASN1_TFLG_ADB_OID       (0x1<<8)

#define ASN1_TFLG_ADB_INT       (0x1<<9)

/* This flag means a parent structure is passed
 * instead of the field: this is useful is a
 * SEQUENCE is being combined with a CHOICE for
 * example. Since this means the structure and
 * item name will differ we need to use the
 * ASN1_CHOICE_END_name() macro for example.
 */

#define ASN1_TFLG_COMBINE       (0x1<<10)

/* This flag when present in a SEQUENCE OF, SET OF
 * or EXPLICIT causes indefinite length constructed
 * encoding to be used if required.
 */

#define ASN1_TFLG_NDEF          (0x1<<11)

/* This is the actual ASN1 item itself */

struct ASN1_ITEM_st {
char itype;                     /* The item type, primitive, SEQUENCE, CHOICE or
long utype;                     /* underlying type */
const ASN1_TEMPLATE *templates; /* If SEQUENCE or CHOICE this contains the conte
long tcount;                    /* Number of templates if SEQUENCE or CHOICE */
const void *funcs;              /* functions that handle this type */
long size;                      /* Structure size (usually)*/
#ifndef NO_ASN1_FIELD_NAMES
const char *sname;              /* Structure name */
#endif
};

/* These are values for the itype field and
 * determine how the type is interpreted.
 *
 * For PRIMITIVE types the underlying type
 * determines the behaviour if items is NULL.
 *
 * Otherwise templates must contain a single
 * template and the type is treated in the
 * same way as the type specified in the template.
 *
```



```
 * For SEQUENCE types the templates field points
 * to the members, the size field is the
 * structure size.
 *
 * For CHOICE types the templates field points
 * to each possible member (typically a union)
 * and the 'size' field is the offset of the
 * selector.
 *
 * The 'funcs' field is used for application
 * specific functions.
 *
 * For COMPAT types the funcs field gives a
 * set of functions that handle this type, this
 * supports the old d2i, i2d convention.
 *
 * The EXTERN type uses a new style d2i/i2d.
 * The new style should be used where possible
 * because it avoids things like the d2i IMPLICIT
 * hack.
 *
 * MSTRING is a multiple string type, it is used
 * for a CHOICE of character strings where the
 * actual strings all occupy an ASN1_STRING
 * structure. In this case the 'utype' field
 * has a special meaning, it is used as a mask
 * of acceptable types using the B_ASN1 constants.
 *
 * NDEF_SEQUENCE is the same as SEQUENCE except
 * that it will use indefinite length constructed
 * encoding if requested.
 *
 */

#define ASN1_ITYPE_PRIMITIVE            0x0

#define ASN1_ITYPE_SEQUENCE             0x1

#define ASN1_ITYPE_CHOICE               0x2

#define ASN1_ITYPE_COMPAT               0x3

#define ASN1_ITYPE_EXTERN               0x4

#define ASN1_ITYPE_MSTRING              0x5

#define ASN1_ITYPE_NDEF_SEQUENCE        0x6

/* Cache for ASN1 tag and length, so we
 * don't keep re-reading it for things
 * like CHOICE
 */

struct ASN1_TLC_st{
        char valid;     /* Values below are valid */
        int ret;        /* return value */
        long plen;      /* length */
        int ptag;       /* class value */
        int pclass;     /* class value */
        int hdrlen;     /* header length */
};

/* Typedefs for ASN1 function pointers */

typedef ASN1_VALUE * ASN1_new_func(void);
typedef void ASN1_free_func(ASN1_VALUE *a);
```

```
656 typedef ASN1_VALUE * ASN1_d2i_func(ASN1_VALUE **a, const unsigned char ** in, lo
657 typedef int ASN1_i2d_func(ASN1_VALUE * a, unsigned char **in);

659 typedef int ASN1_ex_d2i(ASN1_VALUE **pval, const unsigned char **in, long len, c
660                                     int tag, int aclass, char opt, ASN1_TLC

662 typedef int ASN1_ex_i2d(ASN1_VALUE **pval, unsigned char **out, const ASN1_ITEM
663 typedef int ASN1_ex_new_func(ASN1_VALUE **pval, const ASN1_ITEM *it);
664 typedef void ASN1_ex_free_func(ASN1_VALUE **pval, const ASN1_ITEM *it);

666 typedef int ASN1_ex_print_func(BIO *out, ASN1_VALUE **pval,
667                                                 int indent, const char *fname,
668                                                 const ASN1_PCTX *pctx);

670 typedef int ASN1_primitive_i2c(ASN1_VALUE **pval, unsigned char *cont, int *puty
671 typedef int ASN1_primitive_c2i(ASN1_VALUE **pval, const unsigned char *cont, int
672 typedef int ASN1_primitive_print(BIO *out, ASN1_VALUE **pval, const ASN1_ITEM *i

674 typedef struct ASN1_COMPAT_FUNCS_st {
675         ASN1_new_func *asn1_new;
676         ASN1_free_func *asn1_free;
677         ASN1_d2i_func *asn1_d2i;
678         ASN1_i2d_func *asn1_i2d;
679 } ASN1_COMPAT_FUNCS;

681 typedef struct ASN1_EXTERN_FUNCS_st {
682         void *app_data;
683         ASN1_ex_new_func *asn1_ex_new;
684         ASN1_ex_free_func *asn1_ex_free;
685         ASN1_ex_free_func *asn1_ex_clear;
686         ASN1_ex_d2i *asn1_ex_d2i;
687         ASN1_ex_i2d *asn1_ex_i2d;
688         ASN1_ex_print_func *asn1_ex_print;
689 } ASN1_EXTERN_FUNCS;

691 typedef struct ASN1_PRIMITIVE_FUNCS_st {
692         void *app_data;
693         unsigned long flags;
694         ASN1_ex_new_func *prim_new;
695         ASN1_ex_free_func *prim_free;
696         ASN1_ex_free_func *prim_clear;
697         ASN1_primitive_c2i *prim_c2i;
698         ASN1_primitive_i2c *prim_i2c;
699         ASN1_primitive_print *prim_print;
700 } ASN1_PRIMITIVE_FUNCS;

702 /* This is the ASN1_AUX structure: it handles various
703  * miscellaneous requirements. For example the use of
704  * reference counts and an informational callback.
705  *
706  * The "informational callback" is called at various
707  * points during the ASN1 encoding and decoding. It can
708  * be used to provide minor customisation of the structures
709  * used. This is most useful where the supplied routines
710  * *almost* do the right thing but need some extra help
711  * at a few points. If the callback returns zero then
712  * it is assumed a fatal error has occurred and the
713  * main operation should be abandoned.
714  *
715  * If major changes in the default behaviour are required
716  * then an external type is more appropriate.
717  */

719 typedef int ASN1_aux_cb(int operation, ASN1_VALUE **in, const ASN1_ITEM *it,
720                                 void *exarg);
```



```
722 typedef struct ASN1_AUX_st {
723         void *app_data;
724         int flags;
725         int ref_offset;         /* Offset of reference value */
726         int ref_lock;           /* Lock type to use */
727         ASN1_aux_cb *asn1_cb;
728         int enc_offset;         /* Offset of ASN1_ENCODING structure */
729 } ASN1_AUX;

731 /* For print related callbacks exarg points to this structure */
732 typedef struct ASN1_PRINT_ARG_st {
733         BIO *out;
734         int indent;
735         const ASN1_PCTX *pctx;
736 } ASN1_PRINT_ARG;

738 /* For streaming related callbacks exarg points to this structure */
739 typedef struct ASN1_STREAM_ARG_st {
740         /* BIO to stream through */
741         BIO *out;
742         /* BIO with filters appended */
743         BIO *ndef_bio;
744         /* Streaming I/O boundary */
745         unsigned char **boundary;
746 } ASN1_STREAM_ARG;

748 /* Flags in ASN1_AUX */

750 /* Use a reference count */
751 #define ASN1_AFLG_REFCOUNT      1
752 /* Save the encoding of structure (useful for signatures) */
753 #define ASN1_AFLG_ENCODING      2
754 /* The Sequence length is invalid */
755 #define ASN1_AFLG_BROKEN        4

757 /* operation values for asn1_cb */

759 #define ASN1_OP_NEW_PRE         0
760 #define ASN1_OP_NEW_POST        1
761 #define ASN1_OP_FREE_PRE        2
762 #define ASN1_OP_FREE_POST       3
763 #define ASN1_OP_D2I_PRE         4
764 #define ASN1_OP_D2I_POST        5
765 #define ASN1_OP_I2D_PRE         6
766 #define ASN1_OP_I2D_POST        7
767 #define ASN1_OP_PRINT_PRE       8
768 #define ASN1_OP_PRINT_POST      9
769 #define ASN1_OP_STREAM_PRE      10
770 #define ASN1_OP_STREAM_POST     11
771 #define ASN1_OP_DETACHED_PRE    12
772 #define ASN1_OP_DETACHED_POST   13

774 /* Macro to implement a primitive type */
775 #define IMPLEMENT_ASN1_TYPE(stname) IMPLEMENT_ASN1_TYPE_ex(stname, stname, 0)
776 #define IMPLEMENT_ASN1_TYPE_ex(itname, vname, ex) \
777                                 ASN1_ITEM_start(itname) \
778                                         ASN1_ITYPE_PRIMITIVE, V_##vname, NULL, 0
779                                 ASN1_ITEM_end(itname)

781 /* Macro to implement a multi string type */
782 #define IMPLEMENT_ASN1_MSTRING(itname, mask) \
783                                 ASN1_ITEM_start(itname) \
784                                         ASN1_ITYPE_MSTRING, mask, NULL, 0, NULL,
785                                 ASN1_ITEM_end(itname)

787 /* Macro to implement an ASN1_ITEM in terms of old style funcs */
```

```
789 #define IMPLEMENT_COMPAT_ASN1(sname) IMPLEMENT_COMPAT_ASN1_type(sname, V_ASN1_SE

791 #define IMPLEMENT_COMPAT_ASN1_type(sname, tag) \
792         static const ASN1_COMPAT_FUNCS sname##_ff = { \
793                 (ASN1_new_func *)sname##_new, \
794                 (ASN1_free_func *)sname##_free, \
795                 (ASN1_d2i_func *)d2i_##sname, \
796                 (ASN1_i2d_func *)i2d_##sname, \
797         }; \
798         ASN1_ITEM_start(sname) \
799                 ASN1_ITYPE_COMPAT, \
800                 tag, \
801                 NULL, \
802                 0, \
803                 &sname##_ff, \
804                 0, \
805                 #sname \
806         ASN1_ITEM_end(sname)

808 #define IMPLEMENT_EXTERN_ASN1(sname, tag, fptrs) \
809         ASN1_ITEM_start(sname) \
810                 ASN1_ITYPE_EXTERN, \
811                 tag, \
812                 NULL, \
813                 0, \
814                 &fptrs, \
815                 0, \
816                 #sname \
817         ASN1_ITEM_end(sname)

819 /* Macro to implement standard functions in terms of ASN1_ITEM structures */

821 #define IMPLEMENT_ASN1_FUNCTIONS(stname) IMPLEMENT_ASN1_FUNCTIONS_fname(stname,

823 #define IMPLEMENT_ASN1_FUNCTIONS_name(stname, itname) IMPLEMENT_ASN1_FUNCTIONS_f

825 #define IMPLEMENT_ASN1_FUNCTIONS_ENCODE_name(stname, itname) \
826                         IMPLEMENT_ASN1_FUNCTIONS_ENCODE_fname(stname, itname, it

828 #define IMPLEMENT_STATIC_ASN1_ALLOC_FUNCTIONS(stname) \
829                 IMPLEMENT_ASN1_ALLOC_FUNCTIONS_pfname(static, stname, stname, st

831 #define IMPLEMENT_ASN1_ALLOC_FUNCTIONS(stname) \
832                 IMPLEMENT_ASN1_ALLOC_FUNCTIONS_fname(stname, stname, stname)

834 #define IMPLEMENT_ASN1_ALLOC_FUNCTIONS_pfname(pre, stname, itname, fname) \
835         pre stname *fname##_new(void) \
836         { \
837                 return (stname *)ASN1_item_new(ASN1_ITEM_rptr(itname)); \
838         } \
839         pre void fname##_free(stname *a) \
840         { \
841                 ASN1_item_free((ASN1_VALUE *)a, ASN1_ITEM_rptr(itname)); \
842         }

844 #define IMPLEMENT_ASN1_ALLOC_FUNCTIONS_fname(stname, itname, fname) \
845         stname *fname##_new(void) \
846         { \
847                 return (stname *)ASN1_item_new(ASN1_ITEM_rptr(itname)); \
848         } \
849         void fname##_free(stname *a) \
850         { \
851                 ASN1_item_free((ASN1_VALUE *)a, ASN1_ITEM_rptr(itname)); \
852         }
```



```
854 #define IMPLEMENT_ASN1_FUNCTIONS_fname(stname, itname, fname) \
855         IMPLEMENT_ASN1_ENCODE_FUNCTIONS_fname(stname, itname, fname) \
856         IMPLEMENT_ASN1_ALLOC_FUNCTIONS_fname(stname, itname, fname)

858 #define IMPLEMENT_ASN1_ENCODE_FUNCTIONS_fname(stname, itname, fname) \
859         stname *d2i_##fname(stname **a, const unsigned char **in, long len) \
860         { \
861                 return (stname *)ASN1_item_d2i((ASN1_VALUE **)a, in, len, ASN1_I
862         } \
863         int i2d_##fname(stname *a, unsigned char **out) \
864         { \
865                 return ASN1_item_i2d((ASN1_VALUE *)a, out, ASN1_ITEM_rptr(itname
866         }

868 #define IMPLEMENT_ASN1_NDEF_FUNCTION(stname) \
869         int i2d_##stname##_NDEF(stname *a, unsigned char **out) \
870         { \
871                 return ASN1_item_ndef_i2d((ASN1_VALUE *)a, out, ASN1_ITEM_rptr(s
872         }

874 /* This includes evil casts to remove const: they will go away when full
875  * ASN1 constification is done.
876  */
877 #define IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(stname, itname, fname) \
878         stname *d2i_##fname(stname **a, const unsigned char **in, long len) \
879         { \
880                 return (stname *)ASN1_item_d2i((ASN1_VALUE **)a, in, len, ASN1_I
881         } \
882         int i2d_##fname(const stname *a, unsigned char **out) \
883         { \
884                 return ASN1_item_i2d((ASN1_VALUE *)a, out, ASN1_ITEM_rptr(itname
885         }

887 #define IMPLEMENT_ASN1_DUP_FUNCTION(stname) \
888         stname * stname##_dup(stname *x) \
889         { \
890         return ASN1_item_dup(ASN1_ITEM_rptr(stname), x); \
891         }

893 #define IMPLEMENT_ASN1_PRINT_FUNCTION(stname) \
894         IMPLEMENT_ASN1_PRINT_FUNCTION_fname(stname, stname, stname)

896 #define IMPLEMENT_ASN1_PRINT_FUNCTION_fname(stname, itname, fname) \
897         int fname##_print_ctx(BIO *out, stname *x, int indent, \
898                                                 const ASN1_PCTX *pctx) \
899         { \
900                 return ASN1_item_print(out, (ASN1_VALUE *)x, indent, \
901                         ASN1_ITEM_rptr(itname), pctx); \
902         }

904 #define IMPLEMENT_ASN1_FUNCTIONS_const(name) \
905                 IMPLEMENT_ASN1_FUNCTIONS_const_fname(name, name, name)

907 #define IMPLEMENT_ASN1_FUNCTIONS_const_fname(stname, itname, fname) \
908         IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(stname, itname, fname) \
909         IMPLEMENT_ASN1_ALLOC_FUNCTIONS_fname(stname, itname, fname)

911 /* external definitions for primitive types */

913 DECLARE_ASN1_ITEM(ASN1_BOOLEAN)
914 DECLARE_ASN1_ITEM(ASN1_TBOOLEAN)
915 DECLARE_ASN1_ITEM(ASN1_FBOOLEAN)
916 DECLARE_ASN1_ITEM(ASN1_SEQUENCE)
917 DECLARE_ASN1_ITEM(CBIGNUM)
918 DECLARE_ASN1_ITEM(BIGNUM)
919 DECLARE_ASN1_ITEM(LONG)
```

```
DECLARE_ASN1_ITEM(ZLONG)

DECLARE_STACK_OF(ASN1_VALUE)

/* Functions used internally by the ASN1 code */

int ASN1_item_ex_new(ASN1_VALUE **pval, const ASN1_ITEM *it);
void ASN1_item_ex_free(ASN1_VALUE **pval, const ASN1_ITEM *it);
int ASN1_template_new(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt);
int ASN1_primitive_new(ASN1_VALUE **pval, const ASN1_ITEM *it);

void ASN1_template_free(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt);
int ASN1_template_d2i(ASN1_VALUE **pval, const unsigned char **in, long len, con
int ASN1_item_ex_d2i(ASN1_VALUE **pval, const unsigned char **in, long len, cons
                               int tag, int aclass, char opt, ASN1_TLC *ctx);

int ASN1_item_ex_i2d(ASN1_VALUE **pval, unsigned char **out, const ASN1_ITEM *it
int ASN1_template_i2d(ASN1_VALUE **pval, unsigned char **out, const ASN1_TEMPLAT
void ASN1_primitive_free(ASN1_VALUE **pval, const ASN1_ITEM *it);

int asn1_ex_i2c(ASN1_VALUE **pval, unsigned char *cont, int *putype, const ASN1_
int asn1_ex_c2i(ASN1_VALUE **pval, const unsigned char *cont, int len, int utype

int asn1_get_choice_selector(ASN1_VALUE **pval, const ASN1_ITEM *it);
int asn1_set_choice_selector(ASN1_VALUE **pval, int value, const ASN1_ITEM *it);

ASN1_VALUE ** asn1_get_field_ptr(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt);

const ASN1_TEMPLATE *asn1_do_adb(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt, int

int asn1_do_lock(ASN1_VALUE **pval, int op, const ASN1_ITEM *it);

void asn1_enc_init(ASN1_VALUE **pval, const ASN1_ITEM *it);
void asn1_enc_free(ASN1_VALUE **pval, const ASN1_ITEM *it);
int asn1_enc_restore(int *len, unsigned char **out, ASN1_VALUE **pval, const ASN
int asn1_enc_save(ASN1_VALUE **pval, const unsigned char *in, int inlen, const A

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   32973 Wed Aug 13 19:51:40 2014
new/usr/src/lib/openssl/include/openssl/bio.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bio.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_BIO_H
#define HEADER_BIO_H
```



```
#include <openssl/e_os2.h>

#ifndef OPENSSL_NO_FP_API
# include <stdio.h>
#endif
#include <stdarg.h>

#include <openssl/crypto.h>

#ifndef OPENSSL_NO_SCTP
# ifndef OPENSSL_SYS_VMS
# include <stdint.h>
# else
# include <inttypes.h>
# endif
#endif

#ifdef  __cplusplus
extern "C" {
#endif

/* These are the 'types' of BIOs */
#define BIO_TYPE_NONE           0
#define BIO_TYPE_MEM            (1|0x0400)
#define BIO_TYPE_FILE           (2|0x0400)

#define BIO_TYPE_FD             (4|0x0400|0x0100)
#define BIO_TYPE_SOCKET         (5|0x0400|0x0100)
#define BIO_TYPE_NULL           (6|0x0400)
#define BIO_TYPE_SSL            (7|0x0200)
#define BIO_TYPE_MD             (8|0x0200)              /* passive filter */
#define BIO_TYPE_BUFFER         (9|0x0200)              /* filter */
#define BIO_TYPE_CIPHER         (10|0x0200)             /* filter */
#define BIO_TYPE_BASE64         (11|0x0200)             /* filter */
#define BIO_TYPE_CONNECT        (12|0x0400|0x0100)      /* socket - connect */
#define BIO_TYPE_ACCEPT         (13|0x0400|0x0100)      /* socket for accept */
#define BIO_TYPE_PROXY_CLIENT   (14|0x0200)             /* client proxy BIO */
#define BIO_TYPE_PROXY_SERVER   (15|0x0200)             /* server proxy BIO */
#define BIO_TYPE_NBIO_TEST      (16|0x0200)             /* server proxy BIO */
#define BIO_TYPE_NULL_FILTER    (17|0x0200)
#define BIO_TYPE_BER            (18|0x0200)             /* BER -> bin filter */
#define BIO_TYPE_BIO            (19|0x0400)             /* (half a) BIO pair */
#define BIO_TYPE_LINEBUFFER     (20|0x0200)             /* filter */
#define BIO_TYPE_DGRAM          (21|0x0400|0x0100)
#ifndef OPENSSL_NO_SCTP
#define BIO_TYPE_DGRAM_SCTP     (24|0x0400|0x0100)
#endif
#define BIO_TYPE_ASN1           (22|0x0200)             /* filter */
#define BIO_TYPE_COMP           (23|0x0200)             /* filter */

#define BIO_TYPE_DESCRIPTOR     0x0100  /* socket, fd, connect or accept */
#define BIO_TYPE_FILTER         0x0200
#define BIO_TYPE_SOURCE_SINK    0x0400

/* BIO_FILENAME_READ|BIO_CLOSE to open or close on free.
 * BIO_set_fp(in,stdin,BIO_NOCLOSE); */
#define BIO_NOCLOSE             0x00
#define BIO_CLOSE               0x01

/* These are used in the following macros and are passed to
 * BIO_ctrl() */
#define BIO_CTRL_RESET          1  /* opt - rewind/zero etc */
#define BIO_CTRL_EOF            2  /* opt - are we at the eof */
#define BIO_CTRL_INFO           3  /* opt - extra tit-bits */
#define BIO_CTRL_SET            4  /* man - set the 'IO' type */
#define BIO_CTRL_GET            5  /* man - get the 'IO' type */
```

```
#define BIO_CTRL_PUSH           6  /* opt - internal, used to signify change */
#define BIO_CTRL_POP            7  /* opt - internal, used to signify change */
#define BIO_CTRL_GET_CLOSE      8  /* man - set the 'close' on free */
#define BIO_CTRL_SET_CLOSE      9  /* man - set the 'close' on free */
#define BIO_CTRL_PENDING        10  /* opt - is their more data buffered */
#define BIO_CTRL_FLUSH          11  /* opt - 'flush' buffered output */
#define BIO_CTRL_DUP            12  /* man - extra stuff for 'duped' BIO */
#define BIO_CTRL_WPENDING       13  /* opt - number of bytes still to write */
/* callback is int cb(BIO *bio,state,ret); */
#define BIO_CTRL_SET_CALLBACK   14  /* opt - set callback function */
#define BIO_CTRL_GET_CALLBACK   15  /* opt - set callback function */

#define BIO_CTRL_SET_FILENAME   30      /* BIO_s_file special */

/* dgram BIO stuff */
#define BIO_CTRL_DGRAM_CONNECT       31  /* BIO dgram special */
#define BIO_CTRL_DGRAM_SET_CONNECTED 32  /* allow for an externally
                                          * connected socket to be
                                          * passed in */
#define BIO_CTRL_DGRAM_SET_RECV_TIMEOUT 33 /* setsockopt, essentially */
#define BIO_CTRL_DGRAM_GET_RECV_TIMEOUT 34 /* getsockopt, essentially */
#define BIO_CTRL_DGRAM_SET_SEND_TIMEOUT 35 /* setsockopt, essentially */
#define BIO_CTRL_DGRAM_GET_SEND_TIMEOUT 36 /* getsockopt, essentially */

#define BIO_CTRL_DGRAM_GET_RECV_TIMER_EXP 37 /* flag whether the last */
#define BIO_CTRL_DGRAM_GET_SEND_TIMER_EXP 38 /* I/O operation tiemd out */

/* #ifdef IP_MTU_DISCOVER */
#define BIO_CTRL_DGRAM_MTU_DISCOVER       39 /* set DF bit on egress packets */
/* #endif */

#define BIO_CTRL_DGRAM_QUERY_MTU          40 /* as kernel for current MTU */
#define BIO_CTRL_DGRAM_GET_FALLBACK_MTU   47
#define BIO_CTRL_DGRAM_GET_MTU            41 /* get cached value for MTU */
#define BIO_CTRL_DGRAM_SET_MTU            42 /* set cached value for
                                              * MTU. want to use this
                                              * if asking the kernel
                                              * fails */

#define BIO_CTRL_DGRAM_MTU_EXCEEDED       43 /* check whether the MTU
                                              * was exceed in the
                                              * previous write
                                              * operation */

#define BIO_CTRL_DGRAM_GET_PEER           46
#define BIO_CTRL_DGRAM_SET_PEER           44 /* Destination for the data */

#define BIO_CTRL_DGRAM_SET_NEXT_TIMEOUT   45 /* Next DTLS handshake timeout to
                                              * adjust socket timeouts */

#ifndef OPENSSL_NO_SCTP
/* SCTP stuff */
#define BIO_CTRL_DGRAM_SCTP_SET_IN_HANDSHAKE    50
#define BIO_CTRL_DGRAM_SCTP_ADD_AUTH_KEY                51
#define BIO_CTRL_DGRAM_SCTP_NEXT_AUTH_KEY               52
#define BIO_CTRL_DGRAM_SCTP_AUTH_CCS_RCVD               53
#define BIO_CTRL_DGRAM_SCTP_GET_SNDINFO         60
#define BIO_CTRL_DGRAM_SCTP_SET_SNDINFO         61
#define BIO_CTRL_DGRAM_SCTP_GET_RCVINFO         62
#define BIO_CTRL_DGRAM_SCTP_SET_RCVINFO         63
#define BIO_CTRL_DGRAM_SCTP_GET_PRINFO                  64
#define BIO_CTRL_DGRAM_SCTP_SET_PRINFO                  65
#define BIO_CTRL_DGRAM_SCTP_SAVE_SHUTDOWN               70
#endif

/* modifiers */
```



```
#define BIO_FP_READ             0x02
#define BIO_FP_WRITE            0x04
#define BIO_FP_APPEND           0x08
#define BIO_FP_TEXT             0x10

#define BIO_FLAGS_READ          0x01
#define BIO_FLAGS_WRITE         0x02
#define BIO_FLAGS_IO_SPECIAL    0x04
#define BIO_FLAGS_RWS (BIO_FLAGS_READ|BIO_FLAGS_WRITE|BIO_FLAGS_IO_SPECIAL)
#define BIO_FLAGS_SHOULD_RETRY  0x08
#ifndef BIO_FLAGS_UPLINK
/* "UPLINK" flag denotes file descriptors provided by application.
   It defaults to 0, as most platforms don't require UPLINK interface. */
#define BIO_FLAGS_UPLINK        0
#endif

/* Used in BIO_gethostbyname() */
#define BIO_GHBN_CTRL_HITS              1
#define BIO_GHBN_CTRL_MISSES            2
#define BIO_GHBN_CTRL_CACHE_SIZE        3
#define BIO_GHBN_CTRL_GET_ENTRY         4
#define BIO_GHBN_CTRL_FLUSH             5

/* Mostly used in the SSL BIO */
/* Not used anymore
 * #define BIO_FLAGS_PROTOCOL_DELAYED_READ 0x10
 * #define BIO_FLAGS_PROTOCOL_DELAYED_WRITE 0x20
 * #define BIO_FLAGS_PROTOCOL_STARTUP   0x40
 */

#define BIO_FLAGS_BASE64_NO_NL  0x100

/* This is used with memory BIOs: it means we shouldn't free up or change the
 * data in any way.
 */
#define BIO_FLAGS_MEM_RDONLY    0x200

typedef struct bio_st BIO;

void BIO_set_flags(BIO *b, int flags);
int  BIO_test_flags(const BIO *b, int flags);
void BIO_clear_flags(BIO *b, int flags);

#define BIO_get_flags(b) BIO_test_flags(b, ~(0x0))
#define BIO_set_retry_special(b) \
                BIO_set_flags(b, (BIO_FLAGS_IO_SPECIAL|BIO_FLAGS_SHOULD_RETRY))
#define BIO_set_retry_read(b) \
                BIO_set_flags(b, (BIO_FLAGS_READ|BIO_FLAGS_SHOULD_RETRY))
#define BIO_set_retry_write(b) \
                BIO_set_flags(b, (BIO_FLAGS_WRITE|BIO_FLAGS_SHOULD_RETRY))

/* These are normally used internally in BIOs */
#define BIO_clear_retry_flags(b) \
                BIO_clear_flags(b, (BIO_FLAGS_RWS|BIO_FLAGS_SHOULD_RETRY))
#define BIO_get_retry_flags(b) \
                BIO_test_flags(b, (BIO_FLAGS_RWS|BIO_FLAGS_SHOULD_RETRY))

/* These should be used by the application to tell why we should retry */
#define BIO_should_read(a)              BIO_test_flags(a, BIO_FLAGS_READ)
#define BIO_should_write(a)             BIO_test_flags(a, BIO_FLAGS_WRITE)
#define BIO_should_io_special(a)        BIO_test_flags(a, BIO_FLAGS_IO_SPECIAL)
#define BIO_retry_type(a)               BIO_test_flags(a, BIO_FLAGS_RWS)
#define BIO_should_retry(a)             BIO_test_flags(a, BIO_FLAGS_SHOULD_RETRY

/* The next three are used in conjunction with the
 * BIO_should_io_special() condition.  After this returns true,
```

```
 * BIO *BIO_get_retry_BIO(BIO *bio, int *reason); will walk the BIO
 * stack and return the 'reason' for the special and the offending BIO.
 * Given a BIO, BIO_get_retry_reason(bio) will return the code. */
/* Returned from the SSL bio when the certificate retrieval code had an error */
#define BIO_RR_SSL_X509_LOOKUP          0x01
/* Returned from the connect BIO when a connect would have blocked */
#define BIO_RR_CONNECT                  0x02
/* Returned from the accept BIO when an accept would have blocked */
#define BIO_RR_ACCEPT                   0x03

/* These are passed by the BIO callback */
#define BIO_CB_FREE     0x01
#define BIO_CB_READ     0x02
#define BIO_CB_WRITE    0x03
#define BIO_CB_PUTS     0x04
#define BIO_CB_GETS     0x05
#define BIO_CB_CTRL     0x06

/* The callback is called before and after the underling operation,
 * The BIO_CB_RETURN flag indicates if it is after the call */
#define BIO_CB_RETURN   0x80
#define BIO_CB_return(a) ((a)|BIO_CB_RETURN))
#define BIO_cb_pre(a)   (!((a)&BIO_CB_RETURN))
#define BIO_cb_post(a)  ((a)&BIO_CB_RETURN)

long (*BIO_get_callback(const BIO *b)) (struct bio_st *,int,const char *,int, lo
void BIO_set_callback(BIO *b,
        long (*callback)(struct bio_st *,int,const char *,int, long,long));
char *BIO_get_callback_arg(const BIO *b);
void BIO_set_callback_arg(BIO *b, char *arg);

const char * BIO_method_name(const BIO *b);
int BIO_method_type(const BIO *b);

typedef void bio_info_cb(struct bio_st *, int, const char *, int, long, long);

typedef struct bio_method_st
        {
        int type;
        const char *name;
        int (*bwrite)(BIO *, const char *, int);
        int (*bread)(BIO *, char *, int);
        int (*bputs)(BIO *, const char *);
        int (*bgets)(BIO *, char *, int);
        long (*ctrl)(BIO *, int, long, void *);
        int (*create)(BIO *);
        int (*destroy)(BIO *);
        long (*callback_ctrl)(BIO *, int, bio_info_cb *);
        } BIO_METHOD;

struct bio_st
        {
        BIO_METHOD *method;
        /* bio, mode, argp, argi, argl, ret */
        long (*callback)(struct bio_st *,int,const char *,int, long,long);
        char *cb_arg; /* first argument for the callback */

        int init;
        int shutdown;
        int flags;      /* extra storage */
        int retry_reason;
        int num;
        void *ptr;
        struct bio_st *next_bio;        /* used by filter BIOs */
        struct bio_st *prev_bio;        /* used by filter BIOs */
        int references;
```



```
        unsigned long num_read;
        unsigned long num_write;

        CRYPTO_EX_DATA ex_data;
        };

DECLARE_STACK_OF(BIO)

typedef struct bio_f_buffer_ctx_struct
        {
        /* Buffers are setup like this:
         *
         * <---------------------- size ----------------------->
         * +---------------------------------------------------+
         * | consumed | remaining          | free space        |
         * +---------------------------------------------------+
         * <-- off --><------- len ------->
         */

        /* BIO *bio; */ /* this is now in the BIO struct */
        int ibuf_size;  /* how big is the input buffer */
        int obuf_size;  /* how big is the output buffer */

        char *ibuf;             /* the char array */
        int ibuf_len;           /* how many bytes are in it */
        int ibuf_off;           /* write/read offset */

        char *obuf;             /* the char array */
        int obuf_len;           /* how many bytes are in it */
        int obuf_off;           /* write/read offset */
        } BIO_F_BUFFER_CTX;

/* Prefix and suffix callback in ASN1 BIO */
typedef int asn1_ps_func(BIO *b, unsigned char **pbuf, int *plen, void *parg);

#ifndef OPENSSL_NO_SCTP
/* SCTP parameter structs */
struct bio_dgram_sctp_sndinfo
        {
        uint16_t snd_sid;
        uint16_t snd_flags;
        uint32_t snd_ppid;
        uint32_t snd_context;
        };

struct bio_dgram_sctp_rcvinfo
        {
        uint16_t rcv_sid;
        uint16_t rcv_ssn;
        uint16_t rcv_flags;
        uint32_t rcv_ppid;
        uint32_t rcv_tsn;
        uint32_t rcv_cumtsn;
        uint32_t rcv_context;
        };

struct bio_dgram_sctp_prinfo
        {
        uint16_t pr_policy;
        uint32_t pr_value;
        };
#endif

/* connect BIO stuff */
#define BIO_CONN_S_BEFORE               1
#define BIO_CONN_S_GET_IP               2
```

```
#define BIO_CONN_S_GET_PORT             3
#define BIO_CONN_S_CREATE_SOCKET        4
#define BIO_CONN_S_CONNECT              5
#define BIO_CONN_S_OK                   6
#define BIO_CONN_S_BLOCKED_CONNECT      7
#define BIO_CONN_S_NBIO                 8
/*#define BIO_CONN_get_param_hostname   BIO_ctrl */

#define BIO_C_SET_CONNECT                       100
#define BIO_C_DO_STATE_MACHINE                  101
#define BIO_C_SET_NBIO                          102
#define BIO_C_SET_PROXY_PARAM                   103
#define BIO_C_SET_FD                            104
#define BIO_C_GET_FD                            105
#define BIO_C_SET_FILE_PTR                      106
#define BIO_C_GET_FILE_PTR                      107
#define BIO_C_SET_FILENAME                      108
#define BIO_C_SET_SSL                           109
#define BIO_C_GET_SSL                           110
#define BIO_C_SET_MD                            111
#define BIO_C_GET_MD                            112
#define BIO_C_GET_CIPHER_STATUS                 113
#define BIO_C_SET_BUF_MEM                       114
#define BIO_C_GET_BUF_MEM_PTR                   115
#define BIO_C_GET_BUFF_NUM_LINES                116
#define BIO_C_SET_BUFF_SIZE                     117
#define BIO_C_SET_ACCEPT                        118
#define BIO_C_SSL_MODE                          119
#define BIO_C_GET_MD_CTX                        120
#define BIO_C_GET_PROXY_PARAM                   121
#define BIO_C_SET_BUFF_READ_DATA                122 /* data to read first */
#define BIO_C_GET_CONNECT                       123
#define BIO_C_GET_ACCEPT                        124
#define BIO_C_SET_SSL_RENEGOTIATE_BYTES         125
#define BIO_C_GET_SSL_NUM_RENEGOTIATES          126
#define BIO_C_SET_SSL_RENEGOTIATE_TIMEOUT       127
#define BIO_C_FILE_SEEK                         128
#define BIO_C_GET_CIPHER_CTX                    129
#define BIO_C_SET_BUF_MEM_EOF_RETURN            130/*return end of input value*/
#define BIO_C_SET_BIND_MODE                     131
#define BIO_C_GET_BIND_MODE                     132
#define BIO_C_FILE_TELL                         133
#define BIO_C_GET_SOCKS                         134
#define BIO_C_SET_SOCKS                         135

#define BIO_C_SET_WRITE_BUF_SIZE                136/* for BIO_s_bio */
#define BIO_C_GET_WRITE_BUF_SIZE                137
#define BIO_C_MAKE_BIO_PAIR                     138
#define BIO_C_DESTROY_BIO_PAIR                  139
#define BIO_C_GET_WRITE_GUARANTEE               140
#define BIO_C_GET_READ_REQUEST                  141
#define BIO_C_SHUTDOWN_WR                       142
#define BIO_C_NREAD0                            143
#define BIO_C_NREAD                             144
#define BIO_C_NWRITE0                           145
#define BIO_C_NWRITE                            146
#define BIO_C_RESET_READ_REQUEST                147
#define BIO_C_SET_MD_CTX                        148

#define BIO_C_SET_PREFIX                        149
#define BIO_C_GET_PREFIX                        150
#define BIO_C_SET_SUFFIX                        151
#define BIO_C_GET_SUFFIX                        152

#define BIO_C_SET_EX_ARG                        153
#define BIO_C_GET_EX_ARG                        154
```



```
#define BIO_set_app_data(s,arg)         BIO_set_ex_data(s,0,arg)
#define BIO_get_app_data(s)             BIO_get_ex_data(s,0)

/* BIO_s_connect() and BIO_s_socks4a_connect() */
#define BIO_set_conn_hostname(b,name) BIO_ctrl(b,BIO_C_SET_CONNECT,0,(char *)nam
#define BIO_set_conn_port(b,port) BIO_ctrl(b,BIO_C_SET_CONNECT,1,(char *)port)
#define BIO_set_conn_ip(b,ip)     BIO_ctrl(b,BIO_C_SET_CONNECT,2,(char *)ip)
#define BIO_set_conn_int_port(b,port) BIO_ctrl(b,BIO_C_SET_CONNECT,3,(char *)por
#define BIO_get_conn_hostname(b)  BIO_ptr_ctrl(b,BIO_C_GET_CONNECT,0)
#define BIO_get_conn_port(b)      BIO_ptr_ctrl(b,BIO_C_GET_CONNECT,1)
#define BIO_get_conn_ip(b)              BIO_ptr_ctrl(b,BIO_C_GET_CONNECT,2)
#define BIO_get_conn_int_port(b) BIO_int_ctrl(b,BIO_C_GET_CONNECT,3,0)


#define BIO_set_nbio(b,n)       BIO_ctrl(b,BIO_C_SET_NBIO,(n),NULL)

/* BIO_s_accept_socket() */
#define BIO_set_accept_port(b,name) BIO_ctrl(b,BIO_C_SET_ACCEPT,0,(char *)name)
#define BIO_get_accept_port(b)  BIO_ptr_ctrl(b,BIO_C_GET_ACCEPT,0)
/* #define BIO_set_nbio(b,n)    BIO_ctrl(b,BIO_C_SET_NBIO,(n),NULL) */
#define BIO_set_nbio_accept(b,n) BIO_ctrl(b,BIO_C_SET_ACCEPT,1,(n)?(void *)"a":N
#define BIO_set_accept_bios(b,bio) BIO_ctrl(b,BIO_C_SET_ACCEPT,2,(char *)bio)

#define BIO_BIND_NORMAL                 0
#define BIO_BIND_REUSEADDR_IF_UNUSED    1
#define BIO_BIND_REUSEADDR              2
#define BIO_set_bind_mode(b,mode) BIO_ctrl(b,BIO_C_SET_BIND_MODE,mode,NULL)
#define BIO_get_bind_mode(b,mode) BIO_ctrl(b,BIO_C_GET_BIND_MODE,0,NULL)

#define BIO_do_connect(b)       BIO_do_handshake(b)
#define BIO_do_accept(b)        BIO_do_handshake(b)
#define BIO_do_handshake(b)     BIO_ctrl(b,BIO_C_DO_STATE_MACHINE,0,NULL)

/* BIO_s_proxy_client() */
#define BIO_set_url(b,url)      BIO_ctrl(b,BIO_C_SET_PROXY_PARAM,0,(char *)(url)
#define BIO_set_proxies(b,p)    BIO_ctrl(b,BIO_C_SET_PROXY_PARAM,1,(char *)(p))
/* BIO_set_nbio(b,n) */
#define BIO_set_filter_bio(b,s) BIO_ctrl(b,BIO_C_SET_PROXY_PARAM,2,(char *)(s))
/* BIO *BIO_get_filter_bio(BIO *bio); */
#define BIO_set_proxy_cb(b,cb) BIO_callback_ctrl(b,BIO_C_SET_PROXY_PARAM,3,(void
#define BIO_set_proxy_header(b,sk) BIO_ctrl(b,BIO_C_SET_PROXY_PARAM,4,(char *)sk
#define BIO_set_no_connect_return(b,bool) BIO_int_ctrl(b,BIO_C_SET_PROXY_PARAM,5

#define BIO_get_proxy_header(b,skp) BIO_ctrl(b,BIO_C_GET_PROXY_PARAM,0,(char *)s
#define BIO_get_proxies(b,pxy_p) BIO_ctrl(b,BIO_C_GET_PROXY_PARAM,1,(char *)(pxy
#define BIO_get_url(b,url)      BIO_ctrl(b,BIO_C_GET_PROXY_PARAM,2,(char *)(url)
#define BIO_get_no_connect_return(b)    BIO_ctrl(b,BIO_C_GET_PROXY_PARAM,5,NULL)

#define BIO_set_fd(b,fd,c)      BIO_int_ctrl(b,BIO_C_SET_FD,c,fd)
#define BIO_get_fd(b,c)         BIO_ctrl(b,BIO_C_GET_FD,0,(char *)c)

#define BIO_set_fp(b,fp,c)      BIO_ctrl(b,BIO_C_SET_FILE_PTR,c,(char *)fp)
#define BIO_get_fp(b,fpp)       BIO_ctrl(b,BIO_C_GET_FILE_PTR,0,(char *)fpp)

#define BIO_seek(b,ofs) (int)BIO_ctrl(b,BIO_C_FILE_SEEK,ofs,NULL)
#define BIO_tell(b)     (int)BIO_ctrl(b,BIO_C_FILE_TELL,0,NULL)

/* name is cast to lose const, but might be better to route through a function
   so we can do it safely */
#ifdef CONST_STRICT
/* If you are wondering why this isn't defined, its because CONST_STRICT is
 * purely a compile-time kludge to allow const to be checked.
 */
int BIO_read_filename(BIO *b,const char *name);
#else
```

```
#define BIO_read_filename(b,name) BIO_ctrl(b,BIO_C_SET_FILENAME, \
                BIO_CLOSE|BIO_FP_READ,(char *)name)
#endif
#define BIO_write_filename(b,name) BIO_ctrl(b,BIO_C_SET_FILENAME, \
                BIO_CLOSE|BIO_FP_WRITE,name)
#define BIO_append_filename(b,name) BIO_ctrl(b,BIO_C_SET_FILENAME, \
                BIO_CLOSE|BIO_FP_APPEND,name)
#define BIO_rw_filename(b,name) BIO_ctrl(b,BIO_C_SET_FILENAME, \
                BIO_CLOSE|BIO_FP_READ|BIO_FP_WRITE,name)

/* WARNING WARNING, this ups the reference count on the read bio of the
 * SSL structure.  This is because the ssl read BIO is now pointed to by
 * the next_bio field in the bio.  So when you free the BIO, make sure
 * you are doing a BIO_free_all() to catch the underlying BIO. */
#define BIO_set_ssl(b,ssl,c)    BIO_ctrl(b,BIO_C_SET_SSL,c,(char *)ssl)
#define BIO_get_ssl(b,sslp)     BIO_ctrl(b,BIO_C_GET_SSL,0,(char *)sslp)
#define BIO_set_ssl_mode(b,client)      BIO_ctrl(b,BIO_C_SSL_MODE,client,NULL)
#define BIO_set_ssl_renegotiate_bytes(b,num) \
        BIO_ctrl(b,BIO_C_SET_SSL_RENEGOTIATE_BYTES,num,NULL);
#define BIO_get_num_renegotiates(b) \
        BIO_ctrl(b,BIO_C_GET_SSL_NUM_RENEGOTIATES,0,NULL);
#define BIO_set_ssl_renegotiate_timeout(b,seconds) \
        BIO_ctrl(b,BIO_C_SET_SSL_RENEGOTIATE_TIMEOUT,seconds,NULL);

/* defined in evp.h */
/* #define BIO_set_md(b,md)     BIO_ctrl(b,BIO_C_SET_MD,1,(char *)md) */

#define BIO_get_mem_data(b,pp)  BIO_ctrl(b,BIO_CTRL_INFO,0,(char *)pp)
#define BIO_set_mem_buf(b,bm,c) BIO_ctrl(b,BIO_C_SET_BUF_MEM,c,(char *)bm)
#define BIO_get_mem_ptr(b,pp)   BIO_ctrl(b,BIO_C_GET_BUF_MEM_PTR,0,(char *)pp)
#define BIO_set_mem_eof_return(b,v) \
                                BIO_ctrl(b,BIO_C_SET_BUF_MEM_EOF_RETURN,v,NULL)

/* For the BIO_f_buffer() type */
#define BIO_get_buffer_num_lines(b)     BIO_ctrl(b,BIO_C_GET_BUFF_NUM_LINES,0,NU
#define BIO_set_buffer_size(b,size)     BIO_ctrl(b,BIO_C_SET_BUFF_SIZE,size,NULL
#define BIO_set_read_buffer_size(b,size) BIO_int_ctrl(b,BIO_C_SET_BUFF_SIZE,size
#define BIO_set_write_buffer_size(b,size) BIO_int_ctrl(b,BIO_C_SET_BUFF_SIZE,siz
#define BIO_set_buffer_read_data(b,buf,num) BIO_ctrl(b,BIO_C_SET_BUFF_READ_DATA,

/* Don't use the next one unless you know what you are doing :-) */
#define BIO_dup_state(b,ret)    BIO_ctrl(b,BIO_CTRL_DUP,0,(char *)(ret))

#define BIO_reset(b)            (int)BIO_ctrl(b,BIO_CTRL_RESET,0,NULL)
#define BIO_eof(b)              (int)BIO_ctrl(b,BIO_CTRL_EOF,0,NULL)
#define BIO_set_close(b,c)      (int)BIO_ctrl(b,BIO_CTRL_SET_CLOSE,(c),NULL)
#define BIO_get_close(b)        (int)BIO_ctrl(b,BIO_CTRL_GET_CLOSE,0,NULL)
#define BIO_pending(b)          (int)BIO_ctrl(b,BIO_CTRL_PENDING,0,NULL)
#define BIO_wpending(b)         (int)BIO_ctrl(b,BIO_CTRL_WPENDING,0,NULL)
/* ...pending macros have inappropriate return type */
size_t BIO_ctrl_pending(BIO *b);
size_t BIO_ctrl_wpending(BIO *b);
#define BIO_flush(b)            (int)BIO_ctrl(b,BIO_CTRL_FLUSH,0,NULL)
#define BIO_get_info_callback(b,cbp) (int)BIO_ctrl(b,BIO_CTRL_GET_CALLBACK,0, \
                                                   cbp)
#define BIO_set_info_callback(b,cb) (int)BIO_callback_ctrl(b,BIO_CTRL_SET_CALLBA

/* For the BIO_f_buffer() type */
#define BIO_buffer_get_num_lines(b) BIO_ctrl(b,BIO_CTRL_GET,0,NULL)

/* For BIO_s_bio() */
#define BIO_set_write_buf_size(b,size) (int)BIO_ctrl(b,BIO_C_SET_WRITE_BUF_SIZE,
#define BIO_get_write_buf_size(b,size) (size_t)BIO_ctrl(b,BIO_C_GET_WRITE_BUF_SI
#define BIO_make_bio_pair(b1,b2)   (int)BIO_ctrl(b1,BIO_C_MAKE_BIO_PAIR,0,b2)
#define BIO_destroy_bio_pair(b)    (int)BIO_ctrl(b,BIO_C_DESTROY_BIO_PAIR,0,NULL
#define BIO_shutdown_wr(b) (int)BIO_ctrl(b, BIO_C_SHUTDOWN_WR, 0, NULL)
```



```
/* macros with inappropriate type -- but ...pending macros use int too: */
#define BIO_get_write_guarantee(b) (int)BIO_ctrl(b,BIO_C_GET_WRITE_GUARANTEE,0,N
#define BIO_get_read_request(b)    (int)BIO_ctrl(b,BIO_C_GET_READ_REQUEST,0,NULL
size_t BIO_ctrl_get_write_guarantee(BIO *b);
size_t BIO_ctrl_get_read_request(BIO *b);
int BIO_ctrl_reset_read_request(BIO *b);

/* ctrl macros for dgram */
#define BIO_ctrl_dgram_connect(b,peer)  \
                     (int)BIO_ctrl(b,BIO_CTRL_DGRAM_CONNECT,0, (char *)peer)
#define BIO_ctrl_set_connected(b, state, peer) \
         (int)BIO_ctrl(b, BIO_CTRL_DGRAM_SET_CONNECTED, state, (char *)peer)
#define BIO_dgram_recv_timedout(b) \
         (int)BIO_ctrl(b, BIO_CTRL_DGRAM_GET_RECV_TIMER_EXP, 0, NULL)
#define BIO_dgram_send_timedout(b) \
         (int)BIO_ctrl(b, BIO_CTRL_DGRAM_GET_SEND_TIMER_EXP, 0, NULL)
#define BIO_dgram_get_peer(b,peer) \
         (int)BIO_ctrl(b, BIO_CTRL_DGRAM_GET_PEER, 0, (char *)peer)
#define BIO_dgram_set_peer(b,peer) \
         (int)BIO_ctrl(b, BIO_CTRL_DGRAM_SET_PEER, 0, (char *)peer)

/* These two aren't currently implemented */
/* int BIO_get_ex_num(BIO *bio); */
/* void BIO_set_ex_free_func(BIO *bio,int idx,void (*cb)()); */
int BIO_set_ex_data(BIO *bio,int idx,void *data);
void *BIO_get_ex_data(BIO *bio,int idx);
int BIO_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
        CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
unsigned long BIO_number_read(BIO *bio);
unsigned long BIO_number_written(BIO *bio);

/* For BIO_f_asn1() */
int BIO_asn1_set_prefix(BIO *b, asn1_ps_func *prefix,
                                        asn1_ps_func *prefix_free);
int BIO_asn1_get_prefix(BIO *b, asn1_ps_func **pprefix,
                                        asn1_ps_func **pprefix_free);
int BIO_asn1_set_suffix(BIO *b, asn1_ps_func *suffix,
                                        asn1_ps_func *suffix_free);
int BIO_asn1_get_suffix(BIO *b, asn1_ps_func **psuffix,
                                        asn1_ps_func **psuffix_free);

# ifndef OPENSSL_NO_FP_API
BIO_METHOD *BIO_s_file(void );
BIO *BIO_new_file(const char *filename, const char *mode);
BIO *BIO_new_fp(FILE *stream, int close_flag);
# define BIO_s_file_internal    BIO_s_file
# endif
BIO *   BIO_new(BIO_METHOD *type);
int     BIO_set(BIO *a,BIO_METHOD *type);
int     BIO_free(BIO *a);
void    BIO_vfree(BIO *a);
int     BIO_read(BIO *b, void *data, int len);
int     BIO_gets(BIO *bp,char *buf, int size);
int     BIO_write(BIO *b, const void *data, int len);
int     BIO_puts(BIO *bp,const char *buf);
int     BIO_indent(BIO *b,int indent,int max);
long    BIO_ctrl(BIO *bp,int cmd,long larg,void *parg);
long BIO_callback_ctrl(BIO *b, int cmd, void (*fp)(struct bio_st *, int, const c
char *  BIO_ptr_ctrl(BIO *bp,int cmd,long larg);
long    BIO_int_ctrl(BIO *bp,int cmd,long larg,int iarg);
BIO *   BIO_push(BIO *b,BIO *append);
BIO *   BIO_pop(BIO *b);
void    BIO_free_all(BIO *a);
BIO *   BIO_find_type(BIO *b,int bio_type);
BIO *   BIO_next(BIO *b);
BIO *   BIO_get_retry_BIO(BIO *bio, int *reason);
```

```
656 int     BIO_get_retry_reason(BIO *bio);
657 BIO *   BIO_dup_chain(BIO *in);

659 int BIO_nread0(BIO *bio, char **buf);
660 int BIO_nread(BIO *bio, char **buf, int num);
661 int BIO_nwrite0(BIO *bio, char **buf);
662 int BIO_nwrite(BIO *bio, char **buf, int num);

664 long BIO_debug_callback(BIO *bio,int cmd,const char *argp,int argi,
665         long argl,long ret);

667 BIO_METHOD *BIO_s_mem(void);
668 BIO *BIO_new_mem_buf(void *buf, int len);
669 BIO_METHOD *BIO_s_socket(void);
670 BIO_METHOD *BIO_s_connect(void);
671 BIO_METHOD *BIO_s_accept(void);
672 BIO_METHOD *BIO_s_fd(void);
673 #ifndef OPENSSL_SYS_OS2
674 BIO_METHOD *BIO_s_log(void);
675 #endif
676 BIO_METHOD *BIO_s_bio(void);
677 BIO_METHOD *BIO_s_null(void);
678 BIO_METHOD *BIO_f_null(void);
679 BIO_METHOD *BIO_f_buffer(void);
680 #ifdef OPENSSL_SYS_VMS
681 BIO_METHOD *BIO_f_linebuffer(void);
682 #endif
683 BIO_METHOD *BIO_f_nbio_test(void);
684 #ifndef OPENSSL_NO_DGRAM
685 BIO_METHOD *BIO_s_datagram(void);
686 #ifndef OPENSSL_NO_SCTP
687 BIO_METHOD *BIO_s_datagram_sctp(void);
688 #endif
689 #endif

691 /* BIO_METHOD *BIO_f_ber(void); */

693 int BIO_sock_should_retry(int i);
694 int BIO_sock_non_fatal_error(int error);
695 int BIO_dgram_non_fatal_error(int error);

697 int BIO_fd_should_retry(int i);
698 int BIO_fd_non_fatal_error(int error);
699 int BIO_dump_cb(int (*cb)(const void *data, size_t len, void *u),
700                 void *u, const char *s, int len);
701 int BIO_dump_indent_cb(int (*cb)(const void *data, size_t len, void *u),
702                        void *u, const char *s, int len, int indent);
703 int BIO_dump(BIO *b,const char *bytes,int len);
704 int BIO_dump_indent(BIO *b,const char *bytes,int len,int indent);
705 #ifndef OPENSSL_NO_FP_API
706 int BIO_dump_fp(FILE *fp, const char *s, int len);
707 int BIO_dump_indent_fp(FILE *fp, const char *s, int len, int indent);
708 #endif
709 struct hostent *BIO_gethostbyname(const char *name);
710 /* We might want a thread-safe interface too:
711  * struct hostent *BIO_gethostbyname_r(const char *name,
712  *     struct hostent *result, void *buffer, size_t buflen);
713  * or something similar (caller allocates a struct hostent,
714  * pointed to by "result", and additional buffer space for the various
715  * substructures; if the buffer does not suffice, NULL is returned
716  * and an appropriate error code is set).
717  */
718 int BIO_sock_error(int sock);
719 int BIO_socket_ioctl(int fd, long type, void *arg);
720 int BIO_socket_nbio(int fd,int mode);
721 int BIO_get_port(const char *str, unsigned short *port_ptr);
```



```
722 int BIO_get_host_ip(const char *str, unsigned char *ip);
723 int BIO_get_accept_socket(char *host_port,int mode);
724 int BIO_accept(int sock,char **ip_port);
725 int BIO_sock_init(void );
726 void BIO_sock_cleanup(void);
727 int BIO_set_tcp_ndelay(int sock,int turn_on);

729 BIO *BIO_new_socket(int sock, int close_flag);
730 BIO *BIO_new_dgram(int fd, int close_flag);
731 #ifndef OPENSSL_NO_SCTP
732 BIO *BIO_new_dgram_sctp(int fd, int close_flag);
733 int BIO_dgram_is_sctp(BIO *bio);
734 int BIO_dgram_sctp_notification_cb(BIO *b,
735                                    void (*handle_notifications)(BIO *bio, void *
736                                    void *context);
737 int BIO_dgram_sctp_wait_for_dry(BIO *b);
738 int BIO_dgram_sctp_msg_waiting(BIO *b);
739 #endif
740 BIO *BIO_new_fd(int fd, int close_flag);
741 BIO *BIO_new_connect(char *host_port);
742 BIO *BIO_new_accept(char *host_port);

744 int BIO_new_bio_pair(BIO **bio1, size_t writebuf1,
745         BIO **bio2, size_t writebuf2);
746 /* If successful, returns 1 and in *bio1, *bio2 two BIO pair endpoints.
747  * Otherwise returns 0 and sets *bio1 and *bio2 to NULL.
748  * Size 0 uses default value.
749  */

751 void BIO_copy_next_retry(BIO *b);

753 /*long BIO_ghbn_ctrl(int cmd,int iarg,char *parg);*/

755 #ifdef __GNUC__
756 #  define __bio_h__attr__ __attribute__
757 #else
758 #  define __bio_h__attr__(x)
759 #endif
760 int BIO_printf(BIO *bio, const char *format, ...)
761         __bio_h__attr__((__format__(__printf__,2,3)));
762 int BIO_vprintf(BIO *bio, const char *format, va_list args)
763         __bio_h__attr__((__format__(__printf__,2,0)));
764 int BIO_snprintf(char *buf, size_t n, const char *format, ...)
765         __bio_h__attr__((__format__(__printf__,3,4)));
766 int BIO_vsnprintf(char *buf, size_t n, const char *format, va_list args)
767         __bio_h__attr__((__format__(__printf__,3,0)));
768 #undef __bio_h__attr__

770 /* BEGIN ERROR CODES */
771 /* The following lines are auto generated by the script mkerr.pl. Any changes
772  * made after this point may be overwritten when the script is next run.
773  */
774 void ERR_load_BIO_strings(void);

776 /* Error codes for the BIO functions. */

778 /* Function codes. */
779 #define BIO_F_ACPT_STATE                                100
780 #define BIO_F_BIO_ACCEPT                                101
781 #define BIO_F_BIO_BER_GET_HEADER                        102
782 #define BIO_F_BIO_CALLBACK_CTRL                         131
783 #define BIO_F_BIO_CTRL                                  103
784 #define BIO_F_BIO_GETHOSTBYNAME                         120
785 #define BIO_F_BIO_GETS                                  104
786 #define BIO_F_BIO_GET_ACCEPT_SOCKET                     105
787 #define BIO_F_BIO_GET_HOST_IP                           106
```

```
#define BIO_F_BIO_GET_PORT                              107
#define BIO_F_BIO_MAKE_PAIR                             121
#define BIO_F_BIO_NEW                                   108
#define BIO_F_BIO_NEW_FILE                              109
#define BIO_F_BIO_NEW_MEM_BUF                           126
#define BIO_F_BIO_NREAD                                 123
#define BIO_F_BIO_NREAD0                                124
#define BIO_F_BIO_NWRITE                                125
#define BIO_F_BIO_NWRITE0                               122
#define BIO_F_BIO_PUTS                                  110
#define BIO_F_BIO_READ                                  111
#define BIO_F_BIO_SOCK_INIT                             112
#define BIO_F_BIO_WRITE                                 113
#define BIO_F_BUFFER_CTRL                               114
#define BIO_F_CONN_CTRL                                 127
#define BIO_F_CONN_STATE                                115
#define BIO_F_DGRAM_SCTP_READ                           132
#define BIO_F_FILE_CTRL                                 116
#define BIO_F_FILE_READ                                 130
#define BIO_F_LINEBUFFER_CTRL                           129
#define BIO_F_MEM_READ                                  128
#define BIO_F_MEM_WRITE                                 117
#define BIO_F_SSL_NEW                                   118
#define BIO_F_WSASTARTUP                                119

/* Reason codes. */
#define BIO_R_ACCEPT_ERROR                              100
#define BIO_R_BAD_FOPEN_MODE                            101
#define BIO_R_BAD_HOSTNAME_LOOKUP                       102
#define BIO_R_BROKEN_PIPE                               124
#define BIO_R_CONNECT_ERROR                             103
#define BIO_R_EOF_ON_MEMORY_BIO                         127
#define BIO_R_ERROR_SETTING_NBIO                        104
#define BIO_R_ERROR_SETTING_NBIO_ON_ACCEPTED_SOCKET     105
#define BIO_R_ERROR_SETTING_NBIO_ON_ACCEPT_SOCKET       106
#define BIO_R_GETHOSTBYNAME_ADDR_IS_NOT_AF_INET         107
#define BIO_R_INVALID_ARGUMENT                          125
#define BIO_R_INVALID_IP_ADDRESS                        108
#define BIO_R_IN_USE                                    123
#define BIO_R_KEEPALIVE                                 109
#define BIO_R_NBIO_CONNECT_ERROR                        110
#define BIO_R_NO_ACCEPT_PORT_SPECIFIED                  111
#define BIO_R_NO_HOSTNAME_SPECIFIED                     112
#define BIO_R_NO_PORT_DEFINED                           113
#define BIO_R_NO_PORT_SPECIFIED                         114
#define BIO_R_NO_SUCH_FILE                              128
#define BIO_R_NULL_PARAMETER                            115
#define BIO_R_TAG_MISMATCH                              116
#define BIO_R_UNABLE_TO_BIND_SOCKET                     117
#define BIO_R_UNABLE_TO_CREATE_SOCKET                   118
#define BIO_R_UNABLE_TO_LISTEN_SOCKET                   119
#define BIO_R_UNINITIALIZED                             120
#define BIO_R_UNSUPPORTED_METHOD                        121
#define BIO_R_WRITE_TO_READ_ONLY_BIO                    126
#define BIO_R_WSASTARTUP                                122

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    5136 Wed Aug 13 19:51:40 2014
new/usr/src/lib/openssl/include/openssl/blowfish.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/blowfish.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_BLOWFISH_H
#define HEADER_BLOWFISH_H
```



```
#include <openssl/e_os2.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef OPENSSL_NO_BF
#error BF is disabled.
#endif

#define BF_ENCRYPT      1
#define BF_DECRYPT      0

/*
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 * ! BF_LONG has to be at least 32 bits wide. If it's wider, then !
 * ! BF_LONG_LOG2 has to be defined along.                        !
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 */

#if defined(__LP32__)
#define BF_LONG unsigned long
#elif defined(OPENSSL_SYS_CRAY) || defined(__ILP64__)
#define BF_LONG unsigned long
#define BF_LONG_LOG2 3
/*
 * _CRAY note. I could declare short, but I have no idea what impact
 * does it have on performance on none-T3E machines. I could declare
 * int, but at least on C90 sizeof(int) can be chosen at compile time.
 * So I've chosen long...
 *                                      <appro@fy.chalmers.se>
 */
#else
#define BF_LONG unsigned int
#endif

#define BF_ROUNDS       16
#define BF_BLOCK        8

typedef struct bf_key_st
        {
        BF_LONG P[BF_ROUNDS+2];
        BF_LONG S[4*256];
        } BF_KEY;

#ifdef OPENSSL_FIPS
void private_BF_set_key(BF_KEY *key, int len, const unsigned char *data);
#endif
void BF_set_key(BF_KEY *key, int len, const unsigned char *data);

void BF_encrypt(BF_LONG *data,const BF_KEY *key);
void BF_decrypt(BF_LONG *data,const BF_KEY *key);

void BF_ecb_encrypt(const unsigned char *in, unsigned char *out,
        const BF_KEY *key, int enc);
void BF_cbc_encrypt(const unsigned char *in, unsigned char *out, long length,
        const BF_KEY *schedule, unsigned char *ivec, int enc);
void BF_cfb64_encrypt(const unsigned char *in, unsigned char *out, long length,
        const BF_KEY *schedule, unsigned char *ivec, int *num, int enc);
void BF_ofb64_encrypt(const unsigned char *in, unsigned char *out, long length,
        const BF_KEY *schedule, unsigned char *ivec, int *num);
const char *BF_options(void);

#ifdef  __cplusplus
}
#endif
```

```
129 #endif
130 #endif /* ! codereview */
```

```
**********************************************************
   36546 Wed Aug 13 19:51:40 2014
new/usr/src/lib/openssl/include/openssl/bn.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * Portions of the attached software ("Contribution") are developed by
 * SUN MICROSYSTEMS, INC., and are contributed to the OpenSSL project.
 *
 * The Contribution is licensed pursuant to the Eric Young open source
 * license provided above.
 *
 * The binary polynomial arithmetic software is originally written by
 * Sheueling Chang Shantz and Douglas Stebila of Sun Microsystems Laboratories.
 *
 */

#ifndef HEADER_BN_H
#define HEADER_BN_H
```

```
#include <openssl/e_os2.h>
#ifndef OPENSSL_NO_FP_API
#include <stdio.h> /* FILE */
#endif
#include <openssl/ossl_typ.h>
#include <openssl/crypto.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* These preprocessor symbols control various aspects of the bignum headers and
 * library code. They're not defined by any "normal" configuration, as they are
 * intended for development and testing purposes. NB: defining all three can be
 * useful for debugging application code as well as openssl itself.
 *
 * BN_DEBUG - turn on various debugging alterations to the bignum code
 * BN_DEBUG_RAND - uses random poisoning of unused words to trip up
 * mismanagement of bignum internals. You must also define BN_DEBUG.
 */
/* #define BN_DEBUG */
/* #define BN_DEBUG_RAND */

#ifndef OPENSSL_SMALL_FOOTPRINT
#define BN_MUL_COMBA
#define BN_SQR_COMBA
#define BN_RECURSION
#endif

/* This next option uses the C libraries (2 word)/(1 word) function.
 * If it is not defined, I use my C version (which is slower).
 * The reason for this flag is that when the particular C compiler
 * library routine is used, and the library is linked with a different
 * compiler, the library is missing.  This mostly happens when the
 * library is built with gcc and then linked using normal cc.  This would
 * be a common occurrence because gcc normally produces code that is
 * 2 times faster than system compilers for the big number stuff.
 * For machines with only one compiler (or shared libraries), this should
 * be on.  Again this in only really a problem on machines
 * using "long long's", are 32bit, and are not using my assembler code. */
#if defined(OPENSSL_SYS_MSDOS) || defined(OPENSSL_SYS_WINDOWS) || \
    defined(OPENSSL_SYS_WIN32) || defined(linux)
# ifndef BN_DIV2W
#  define BN_DIV2W
# endif
#endif

/* assuming long is 64bit - this is the DEC Alpha
 * unsigned long long is only 64 bits :-(, don't define
 * BN_LLONG for the DEC Alpha */
#ifdef SIXTY_FOUR_BIT_LONG
#define BN_ULLONG       unsigned long long
#define BN_ULONG        unsigned long
#define BN_LONG         long
#define BN_BITS         128
#define BN_BYTES        8
#define BN_BITS2        64
#define BN_BITS4        32
#define BN_MASK         (0xffffffffffffffffffffffffffffffffLL)
#define BN_MASK2        (0xffffffffffffffffL)
#define BN_MASK2l       (0xffffffffL)
#define BN_MASK2h       (0xffffffff00000000L)
#define BN_MASK2h1      (0xffffffff80000000L)
#define BN_TBIT         (0x8000000000000000L)
#define BN_DEC_CONV     (10000000000000000000UL)
#define BN_DEC_FMT1     "%lu"
```



```
#define BN_DEC_FMT2     "%019lu"
#define BN_DEC_NUM      19
#define BN_HEX_FMT1     "%lX"
#define BN_HEX_FMT2     "%016lX"
#endif

/* This is where the long long data type is 64 bits, but long is 32.
 * For machines where there are 64bit registers, this is the mode to use.
 * IRIX, on R4000 and above should use this mode, along with the relevant
 * assembler code :-).  Do NOT define BN_LLONG.
 */
#ifdef SIXTY_FOUR_BIT
#undef BN_LLONG
#undef BN_ULLONG
#define BN_ULONG        unsigned long long
#define BN_LONG         long long
#define BN_BITS         128
#define BN_BYTES        8
#define BN_BITS2        64
#define BN_BITS4        32
#define BN_MASK2        (0xffffffffffffffffLL)
#define BN_MASK2l       (0xffffffffL)
#define BN_MASK2h       (0xffffffff00000000LL)
#define BN_MASK2h1      (0xffffffff80000000LL)
#define BN_TBIT         (0x8000000000000000LL)
#define BN_DEC_CONV     (10000000000000000000ULL)
#define BN_DEC_FMT1     "%llu"
#define BN_DEC_FMT2     "%019llu"
#define BN_DEC_NUM      19
#define BN_HEX_FMT1     "%llX"
#define BN_HEX_FMT2     "%016llX"
#endif

#ifdef THIRTY_TWO_BIT
#ifdef BN_LLONG
# if defined(_WIN32) && !defined(__GNUC__)
#  define BN_ULLONG     unsigned __int64
#  define BN_MASK       (0xffffffffffffffffI64)
# else
#  define BN_ULLONG     unsigned long long
#  define BN_MASK       (0xffffffffffffffffLL)
# endif
#endif
#define BN_ULONG        unsigned int
#define BN_LONG         int
#define BN_BITS         64
#define BN_BYTES        4
#define BN_BITS2        32
#define BN_BITS4        16
#define BN_MASK2        (0xffffffffL)
#define BN_MASK2l       (0xffff)
#define BN_MASK2h1      (0xffff8000L)
#define BN_MASK2h       (0xffff0000L)
#define BN_TBIT         (0x80000000L)
#define BN_DEC_CONV     (1000000000L)
#define BN_DEC_FMT1     "%u"
#define BN_DEC_FMT2     "%09u"
#define BN_DEC_NUM      9
#define BN_HEX_FMT1     "%X"
#define BN_HEX_FMT2     "%08X"
#endif

/* 2011-02-22 SMS.
 * In various places, a size_t variable or a type cast to size_t was
 * used to perform integer-only operations on pointers.  This failed on
 * VMS with 64-bit pointers (CC /POINTER_SIZE = 64) because size_t is
```

```
 * still only 32 bits.  What's needed in these cases is an integer type
 * with the same size as a pointer, which size_t is not certain to be.
 * The only fix here is VMS-specific.
 */
#if defined(OPENSSL_SYS_VMS)
# if __INITIAL_POINTER_SIZE == 64
#  define PTR_SIZE_INT long long
# else /* __INITIAL_POINTER_SIZE == 64 */
#  define PTR_SIZE_INT int
# endif /* __INITIAL_POINTER_SIZE == 64 [else] */
#else /* defined(OPENSSL_SYS_VMS) */
# define PTR_SIZE_INT size_t
#endif /* defined(OPENSSL_SYS_VMS) [else] */

#define BN_DEFAULT_BITS 1280

#define BN_FLG_MALLOCED         0x01
#define BN_FLG_STATIC_DATA      0x02
#define BN_FLG_CONSTTIME        0x04 /* avoid leaking exponent information throu
                                      * BN_mod_exp_mont() will call BN_mod_exp_m
                                      * BN_div() will call BN_div_no_branch,
                                      * BN_mod_inverse() will call BN_mod_invers
                                      */

#ifndef OPENSSL_NO_DEPRECATED
#define BN_FLG_EXP_CONSTTIME BN_FLG_CONSTTIME /* deprecated name for the flag */
                                     /* avoid leaking exponent information thro
                                     * (BN_mod_exp_mont() will call BN_mod_exp_
#endif

#ifndef OPENSSL_NO_DEPRECATED
#define BN_FLG_FREE             0x8000  /* used for debuging */
#endif
#define BN_set_flags(b,n)       ((b)->flags|=(n))
#define BN_get_flags(b,n)       ((b)->flags&(n))

/* get a clone of a BIGNUM with changed flags, for *temporary* use only
 * (the two BIGNUMs cannot not be used in parallel!) */
#define BN_with_flags(dest,b,n)  ((dest)->d=(b)->d, \
                                  (dest)->top=(b)->top, \
                                  (dest)->dmax=(b)->dmax, \
                                  (dest)->neg=(b)->neg, \
                                  (dest)->flags=(((dest)->flags & BN_FLG_MALLOCE
                                                 |  ((b)->flags & ~BN_FLG_MALLOC
                                                 |  BN_FLG_STATIC_DATA \
                                                 |  (n)))

/* Already declared in ossl_typ.h */
#if 0
typedef struct bignum_st BIGNUM;
/* Used for temp variables (declaration hidden in bn_lcl.h) */
typedef struct bignum_ctx BN_CTX;
typedef struct bn_blinding_st BN_BLINDING;
typedef struct bn_mont_ctx_st BN_MONT_CTX;
typedef struct bn_recp_ctx_st BN_RECP_CTX;
typedef struct bn_gencb_st BN_GENCB;
#endif

struct bignum_st
        {
        BN_ULONG *d;    /* Pointer to an array of 'BN_BITS2' bit chunks. */
        int top;        /* Index of last used d +1. */
        /* The next are internal book keeping for bn_expand. */
        int dmax;       /* Size of the d array. */
        int neg;        /* one if the number is negative */
        int flags;
```



```
        };

/* Used for montgomery multiplication */
struct bn_mont_ctx_st
        {
        int ri;         /* number of bits in R */
        BIGNUM RR;      /* used to convert to montgomery form */
        BIGNUM N;       /* The modulus */
        BIGNUM Ni;      /* R*(1/R mod N) - N*Ni = 1
                         * (Ni is only stored for bignum algorithm) */
        BN_ULONG n0[2];/* least significant word(s) of Ni;
                          (type changed with 0.9.9, was "BN_ULONG n0;" before) *
        int flags;
        };

/* Used for reciprocal division/mod functions
 * It cannot be shared between threads
 */
struct bn_recp_ctx_st
        {
        BIGNUM N;       /* the divisor */
        BIGNUM Nr;      /* the reciprocal */
        int num_bits;
        int shift;
        int flags;
        };

/* Used for slow "generation" functions. */
struct bn_gencb_st
        {
        unsigned int ver;       /* To handle binary (in)compatibility */
        void *arg;              /* callback-specific data */
        union
                {
                /* if(ver==1) - handles old style callbacks */
                void (*cb_1)(int, int, void *);
                /* if(ver==2) - new callback style */
                int (*cb_2)(int, int, BN_GENCB *);
                } cb;
        };
/* Wrapper function to make using BN_GENCB easier,  */
int BN_GENCB_call(BN_GENCB *cb, int a, int b);
/* Macro to populate a BN_GENCB structure with an "old"-style callback */
#define BN_GENCB_set_old(gencb, callback, cb_arg) { \
                BN_GENCB *tmp_gencb = (gencb); \
                tmp_gencb->ver = 1; \
                tmp_gencb->arg = (cb_arg); \
                tmp_gencb->cb.cb_1 = (callback); }
/* Macro to populate a BN_GENCB structure with a "new"-style callback */
#define BN_GENCB_set(gencb, callback, cb_arg) { \
                BN_GENCB *tmp_gencb = (gencb); \
                tmp_gencb->ver = 2; \
                tmp_gencb->arg = (cb_arg); \
                tmp_gencb->cb.cb_2 = (callback); }

#define BN_prime_checks 0 /* default: select number of iterations
                             based on the size of the number */

/* number of Miller-Rabin iterations for an error rate  of less than 2^-80
 * for random 'b'-bit input, b >= 100 (taken from table 4.4 in the Handbook
 * of Applied Cryptography [Menezes, van Oorschot, Vanstone; CRC Press 1996];
 * original paper: Damgaard, Landrock, Pomerance: Average case error estimates
 * for the strong probable prime test. -- Math. Comp. 61 (1993) 177-194) */
#define BN_prime_checks_for_size(b) ((b) >= 1300 ?  2 : \
                                (b) >=  850 ?  3 : \
                                (b) >=  650 ?  4 : \
```

```
392                                (b) >=  550 ?  5 : \
393                                (b) >=  450 ?  6 : \
394                                (b) >=  400 ?  7 : \
395                                (b) >=  350 ?  8 : \
396                                (b) >=  300 ?  9 : \
397                                (b) >=  250 ? 12 : \
398                                (b) >=  200 ? 15 : \
399                                (b) >=  150 ? 18 : \
400                                /* b >= 100 */ 27)

402 #define BN_num_bytes(a) ((BN_num_bits(a)+7)/8)

404 /* Note that BN_abs_is_word didn't work reliably for w == 0 until 0.9.8 */
405 #define BN_abs_is_word(a,w) ((((a)->top == 1) && ((a)->d[0] == (BN_ULONG)(w))) |
406                                 (((w) == 0) && ((a)->top == 0)))
407 #define BN_is_zero(a)       ((a)->top == 0)
408 #define BN_is_one(a)        (BN_abs_is_word((a),1) && !(a)->neg)
409 #define BN_is_word(a,w)     (BN_abs_is_word((a),(w)) && (!(w) || !(a)->neg))
410 #define BN_is_odd(a)        (((a)->top > 0) && ((a)->d[0] & 1))

412 #define BN_one(a)       (BN_set_word((a),1))
413 #define BN_zero_ex(a) \
414         do { \
415                 BIGNUM *_tmp_bn = (a); \
416                 _tmp_bn->top = 0; \
417                 _tmp_bn->neg = 0; \
418         } while(0)
419 #ifdef OPENSSL_NO_DEPRECATED
420 #define BN_zero(a)      BN_zero_ex(a)
421 #else
422 #define BN_zero(a)      (BN_set_word((a),0))
423 #endif

425 const BIGNUM *BN_value_one(void);
426 char *  BN_options(void);
427 BN_CTX *BN_CTX_new(void);
428 #ifndef OPENSSL_NO_DEPRECATED
429 void    BN_CTX_init(BN_CTX *c);
430 #endif
431 void    BN_CTX_free(BN_CTX *c);
432 void    BN_CTX_start(BN_CTX *ctx);
433 BIGNUM *BN_CTX_get(BN_CTX *ctx);
434 void    BN_CTX_end(BN_CTX *ctx);
435 int     BN_rand(BIGNUM *rnd, int bits, int top,int bottom);
436 int     BN_pseudo_rand(BIGNUM *rnd, int bits, int top,int bottom);
437 int     BN_rand_range(BIGNUM *rnd, const BIGNUM *range);
438 int     BN_pseudo_rand_range(BIGNUM *rnd, const BIGNUM *range);
439 int     BN_num_bits(const BIGNUM *a);
440 int     BN_num_bits_word(BN_ULONG);
441 BIGNUM *BN_new(void);
442 void    BN_init(BIGNUM *);
443 void    BN_clear_free(BIGNUM *a);
444 BIGNUM *BN_copy(BIGNUM *a, const BIGNUM *b);
445 void    BN_swap(BIGNUM *a, BIGNUM *b);
446 BIGNUM *BN_bin2bn(const unsigned char *s,int len,BIGNUM *ret);
447 int     BN_bn2bin(const BIGNUM *a, unsigned char *to);
448 BIGNUM *BN_mpi2bn(const unsigned char *s,int len,BIGNUM *ret);
449 int     BN_bn2mpi(const BIGNUM *a, unsigned char *to);
450 int     BN_sub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);
451 int     BN_usub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);
452 int     BN_uadd(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);
453 int     BN_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b);
454 int     BN_mul(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, BN_CTX *ctx);
455 int     BN_sqr(BIGNUM *r, const BIGNUM *a,BN_CTX *ctx);
456 /** BN_set_negative sets sign of a BIGNUM
457  * \param  b  pointer to the BIGNUM object
```



```
458  * \param  n  0 if the BIGNUM b should be positive and a value != 0 otherwise
459  */
460 void    BN_set_negative(BIGNUM *b, int n);
461 /** BN_is_negative returns 1 if the BIGNUM is negative
462  * \param  a  pointer to the BIGNUM object
463  * \return 1 if a < 0 and 0 otherwise
464  */
465 #define BN_is_negative(a) ((a)->neg != 0)

467 int     BN_div(BIGNUM *dv, BIGNUM *rem, const BIGNUM *m, const BIGNUM *d,
468         BN_CTX *ctx);
469 #define BN_mod(rem,m,d,ctx) BN_div(NULL,(rem),(m),(d),(ctx))
470 int     BN_nnmod(BIGNUM *r, const BIGNUM *m, const BIGNUM *d, BN_CTX *ctx);
471 int     BN_mod_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m,
472 int     BN_mod_add_quick(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGN
473 int     BN_mod_sub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m,
474 int     BN_mod_sub_quick(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGN
475 int     BN_mod_mul(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
476         const BIGNUM *m, BN_CTX *ctx);
477 int     BN_mod_sqr(BIGNUM *r, const BIGNUM *a, const BIGNUM *m, BN_CTX *ctx);
478 int     BN_mod_lshift1(BIGNUM *r, const BIGNUM *a, const BIGNUM *m, BN_CTX *ctx)
479 int     BN_mod_lshift1_quick(BIGNUM *r, const BIGNUM *a, const BIGNUM *m);
480 int     BN_mod_lshift(BIGNUM *r, const BIGNUM *a, int n, const BIGNUM *m, BN_CTX
481 int     BN_mod_lshift_quick(BIGNUM *r, const BIGNUM *a, int n, const BIGNUM *m);

483 BN_ULONG BN_mod_word(const BIGNUM *a, BN_ULONG w);
484 BN_ULONG BN_div_word(BIGNUM *a, BN_ULONG w);
485 int     BN_mul_word(BIGNUM *a, BN_ULONG w);
486 int     BN_add_word(BIGNUM *a, BN_ULONG w);
487 int     BN_sub_word(BIGNUM *a, BN_ULONG w);
488 int     BN_set_word(BIGNUM *a, BN_ULONG w);
489 BN_ULONG BN_get_word(const BIGNUM *a);

491 int     BN_cmp(const BIGNUM *a, const BIGNUM *b);
492 void    BN_free(BIGNUM *a);
493 int     BN_is_bit_set(const BIGNUM *a, int n);
494 int     BN_lshift(BIGNUM *r, const BIGNUM *a, int n);
495 int     BN_lshift1(BIGNUM *r, const BIGNUM *a);
496 int     BN_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,BN_CTX *ctx);

498 int     BN_mod_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
499         const BIGNUM *m,BN_CTX *ctx);
500 int     BN_mod_exp_mont(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
501         const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx);
502 int BN_mod_exp_mont_consttime(BIGNUM *rr, const BIGNUM *a, const BIGNUM *p,
503         const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *in_mont);
504 int     BN_mod_exp_mont_word(BIGNUM *r, BN_ULONG a, const BIGNUM *p,
505         const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx);
506 int     BN_mod_exp2_mont(BIGNUM *r, const BIGNUM *a1, const BIGNUM *p1,
507         const BIGNUM *a2, const BIGNUM *p2,const BIGNUM *m,
508         BN_CTX *ctx,BN_MONT_CTX *m_ctx);
509 int     BN_mod_exp_simple(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
510         const BIGNUM *m,BN_CTX *ctx);

512 int     BN_mask_bits(BIGNUM *a,int n);
513 #ifndef OPENSSL_NO_FP_API
514 int     BN_print_fp(FILE *fp, const BIGNUM *a);
515 #endif
516 #ifdef HEADER_BIO_H
517 int     BN_print(BIO *fp, const BIGNUM *a);
518 #else
519 int     BN_print(void *fp, const BIGNUM *a);
520 #endif
521 int     BN_reciprocal(BIGNUM *r, const BIGNUM *m, int len, BN_CTX *ctx);
522 int     BN_rshift(BIGNUM *r, const BIGNUM *a, int n);
523 int     BN_rshift1(BIGNUM *r, const BIGNUM *a);
```

```
524 void    BN_clear(BIGNUM *a);
525 BIGNUM *BN_dup(const BIGNUM *a);
526 int     BN_ucmp(const BIGNUM *a, const BIGNUM *b);
527 int     BN_set_bit(BIGNUM *a, int n);
528 int     BN_clear_bit(BIGNUM *a, int n);
529 char *  BN_bn2hex(const BIGNUM *a);
530 char *  BN_bn2dec(const BIGNUM *a);
531 int     BN_hex2bn(BIGNUM **a, const char *str);
532 int     BN_dec2bn(BIGNUM **a, const char *str);
533 int     BN_asc2bn(BIGNUM **a, const char *str);
534 int     BN_gcd(BIGNUM *r,const BIGNUM *a,const BIGNUM *b,BN_CTX *ctx);
535 int     BN_kronecker(const BIGNUM *a,const BIGNUM *b,BN_CTX *ctx); /* returns -2
536 BIGNUM *BN_mod_inverse(BIGNUM *ret,
537         const BIGNUM *a, const BIGNUM *n,BN_CTX *ctx);
538 BIGNUM *BN_mod_sqrt(BIGNUM *ret,
539         const BIGNUM *a, const BIGNUM *n,BN_CTX *ctx);

541 void    BN_consttime_swap(BN_ULONG swap, BIGNUM *a, BIGNUM *b, int nwords);

543 /* Deprecated versions */
544 #ifndef OPENSSL_NO_DEPRECATED
545 BIGNUM *BN_generate_prime(BIGNUM *ret,int bits,int safe,
546         const BIGNUM *add, const BIGNUM *rem,
547         void (*callback)(int,int,void *),void *cb_arg);
548 int     BN_is_prime(const BIGNUM *p,int nchecks,
549         void (*callback)(int,int,void *),
550         BN_CTX *ctx,void *cb_arg);
551 int     BN_is_prime_fasttest(const BIGNUM *p,int nchecks,
552         void (*callback)(int,int,void *),BN_CTX *ctx,void *cb_arg,
553         int do_trial_division);
554 #endif /* !defined(OPENSSL_NO_DEPRECATED) */

556 /* Newer versions */
557 int     BN_generate_prime_ex(BIGNUM *ret,int bits,int safe, const BIGNUM *add,
558                 const BIGNUM *rem, BN_GENCB *cb);
559 int     BN_is_prime_ex(const BIGNUM *p,int nchecks, BN_CTX *ctx, BN_GENCB *cb);
560 int     BN_is_prime_fasttest_ex(const BIGNUM *p,int nchecks, BN_CTX *ctx,
561                 int do_trial_division, BN_GENCB *cb);

563 int BN_X931_generate_Xpq(BIGNUM *Xp, BIGNUM *Xq, int nbits, BN_CTX *ctx);

565 int BN_X931_derive_prime_ex(BIGNUM *p, BIGNUM *p1, BIGNUM *p2,
566                         const BIGNUM *Xp, const BIGNUM *Xp1, const BIGNUM *Xp2,
567                         const BIGNUM *e, BN_CTX *ctx, BN_GENCB *cb);
568 int BN_X931_generate_prime_ex(BIGNUM *p, BIGNUM *p1, BIGNUM *p2,
569                         BIGNUM *Xp1, BIGNUM *Xp2,
570                         const BIGNUM *Xp,
571                         const BIGNUM *e, BN_CTX *ctx,
572                         BN_GENCB *cb);

574 BN_MONT_CTX *BN_MONT_CTX_new(void );
575 void BN_MONT_CTX_init(BN_MONT_CTX *ctx);
576 int BN_mod_mul_montgomery(BIGNUM *r,const BIGNUM *a,const BIGNUM *b,
577         BN_MONT_CTX *mont, BN_CTX *ctx);
578 #define BN_to_montgomery(r,a,mont,ctx)  BN_mod_mul_montgomery(\
579         (r),(a),&((mont)->RR),(mont),(ctx))
580 int BN_from_montgomery(BIGNUM *r,const BIGNUM *a,
581         BN_MONT_CTX *mont, BN_CTX *ctx);
582 void BN_MONT_CTX_free(BN_MONT_CTX *mont);
583 int BN_MONT_CTX_set(BN_MONT_CTX *mont,const BIGNUM *mod,BN_CTX *ctx);
584 BN_MONT_CTX *BN_MONT_CTX_copy(BN_MONT_CTX *to,BN_MONT_CTX *from);
585 BN_MONT_CTX *BN_MONT_CTX_set_locked(BN_MONT_CTX **pmont, int lock,
586                                     const BIGNUM *mod, BN_CTX *ctx);

588 /* BN_BLINDING flags */
589 #define BN_BLINDING_NO_UPDATE   0x00000001
```



```
590 #define BN_BLINDING_NO_RECREATE 0x00000002

592 BN_BLINDING *BN_BLINDING_new(const BIGNUM *A, const BIGNUM *Ai, BIGNUM *mod);
593 void BN_BLINDING_free(BN_BLINDING *b);
594 int BN_BLINDING_update(BN_BLINDING *b,BN_CTX *ctx);
595 int BN_BLINDING_convert(BIGNUM *n, BN_BLINDING *b, BN_CTX *ctx);
596 int BN_BLINDING_invert(BIGNUM *n, BN_BLINDING *b, BN_CTX *ctx);
597 int BN_BLINDING_convert_ex(BIGNUM *n, BIGNUM *r, BN_BLINDING *b, BN_CTX *);
598 int BN_BLINDING_invert_ex(BIGNUM *n, const BIGNUM *r, BN_BLINDING *b, BN_CTX *);
599 #ifndef OPENSSL_NO_DEPRECATED
600 unsigned long BN_BLINDING_get_thread_id(const BN_BLINDING *);
601 void BN_BLINDING_set_thread_id(BN_BLINDING *, unsigned long);
602 #endif
603 CRYPTO_THREADID *BN_BLINDING_thread_id(BN_BLINDING *);
604 unsigned long BN_BLINDING_get_flags(const BN_BLINDING *);
605 void BN_BLINDING_set_flags(BN_BLINDING *, unsigned long);
606 BN_BLINDING *BN_BLINDING_create_param(BN_BLINDING *b,
607         const BIGNUM *e, BIGNUM *m, BN_CTX *ctx,
608         int (*bn_mod_exp)(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
609                           const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx),
610         BN_MONT_CTX *m_ctx);

612 #ifndef OPENSSL_NO_DEPRECATED
613 void BN_set_params(int mul,int high,int low,int mont);
614 int BN_get_params(int which); /* 0, mul, 1 high, 2 low, 3 mont */
615 #endif

617 void    BN_RECP_CTX_init(BN_RECP_CTX *recp);
618 BN_RECP_CTX *BN_RECP_CTX_new(void);
619 void    BN_RECP_CTX_free(BN_RECP_CTX *recp);
620 int     BN_RECP_CTX_set(BN_RECP_CTX *recp,const BIGNUM *rdiv,BN_CTX *ctx);
621 int     BN_mod_mul_reciprocal(BIGNUM *r, const BIGNUM *x, const BIGNUM *y,
622         BN_RECP_CTX *recp,BN_CTX *ctx);
623 int     BN_mod_exp_recp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
624         const BIGNUM *m, BN_CTX *ctx);
625 int     BN_div_recp(BIGNUM *dv, BIGNUM *rem, const BIGNUM *m,
626         BN_RECP_CTX *recp, BN_CTX *ctx);

628 #ifndef OPENSSL_NO_EC2M

630 /* Functions for arithmetic over binary polynomials represented by BIGNUMs.
631  *
632  * The BIGNUM::neg property of BIGNUMs representing binary polynomials is
633  * ignored.
634  *
635  * Note that input arguments are not const so that their bit arrays can
636  * be expanded to the appropriate size if needed.
637  */

639 int     BN_GF2m_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b); /*r = a + b*/
640 #define BN_GF2m_sub(r, a, b) BN_GF2m_add(r, a, b)
641 int     BN_GF2m_mod(BIGNUM *r, const BIGNUM *a, const BIGNUM *p); /*r=a mod p*/
642 int     BN_GF2m_mod_mul(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
643         const BIGNUM *p, BN_CTX *ctx); /* r = (a * b) mod p */
644 int     BN_GF2m_mod_sqr(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
645         BN_CTX *ctx); /* r = (a * a) mod p */
646 int     BN_GF2m_mod_inv(BIGNUM *r, const BIGNUM *b, const BIGNUM *p,
647         BN_CTX *ctx); /* r = (1 / b) mod p */
648 int     BN_GF2m_mod_div(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
649         const BIGNUM *p, BN_CTX *ctx); /* r = (a / b) mod p */
650 int     BN_GF2m_mod_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
651         const BIGNUM *p, BN_CTX *ctx); /* r = (a ^ b) mod p */
652 int     BN_GF2m_mod_sqrt(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
653         BN_CTX *ctx); /* r = sqrt(a) mod p */
654 int     BN_GF2m_mod_solve_quad(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
655         BN_CTX *ctx); /* r^2 + r = a mod p */
```

```
656 #define BN_GF2m_cmp(a, b) BN_ucmp((a), (b))
657 /* Some functions allow for representation of the irreducible polynomials
658  * as an unsigned int[], say p.  The irreducible f(t) is then of the form:
659  *     t^p[0] + t^p[1] + ... + t^p[k]
660  * where m = p[0] > p[1] > ... > p[k] = 0.
661  */
662 int     BN_GF2m_mod_arr(BIGNUM *r, const BIGNUM *a, const int p[]);
663         /* r = a mod p */
664 int     BN_GF2m_mod_mul_arr(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
665         const int p[], BN_CTX *ctx); /* r = (a * b) mod p */
666 int     BN_GF2m_mod_sqr_arr(BIGNUM *r, const BIGNUM *a, const int p[],
667         BN_CTX *ctx); /* r = (a * a) mod p */
668 int     BN_GF2m_mod_inv_arr(BIGNUM *r, const BIGNUM *b, const int p[],
669         BN_CTX *ctx); /* r = (1 / b) mod p */
670 int     BN_GF2m_mod_div_arr(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
671         const int p[], BN_CTX *ctx); /* r = (a / b) mod p */
672 int     BN_GF2m_mod_exp_arr(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
673         const int p[], BN_CTX *ctx); /* r = (a ^ b) mod p */
674 int     BN_GF2m_mod_sqrt_arr(BIGNUM *r, const BIGNUM *a,
675         const int p[], BN_CTX *ctx); /* r = sqrt(a) mod p */
676 int     BN_GF2m_mod_solve_quad_arr(BIGNUM *r, const BIGNUM *a,
677         const int p[], BN_CTX *ctx); /* r^2 + r = a mod p */
678 int     BN_GF2m_poly2arr(const BIGNUM *a, int p[], int max);
679 int     BN_GF2m_arr2poly(const int p[], BIGNUM *a);

681 #endif

683 /* faster mod functions for the 'NIST primes'
684  * 0 <= a < p^2 */
685 int BN_nist_mod_192(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx);
686 int BN_nist_mod_224(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx);
687 int BN_nist_mod_256(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx);
688 int BN_nist_mod_384(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx);
689 int BN_nist_mod_521(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx);

691 const BIGNUM *BN_get0_nist_prime_192(void);
692 const BIGNUM *BN_get0_nist_prime_224(void);
693 const BIGNUM *BN_get0_nist_prime_256(void);
694 const BIGNUM *BN_get0_nist_prime_384(void);
695 const BIGNUM *BN_get0_nist_prime_521(void);

697 /* library internal functions */

699 #define bn_expand(a,bits) ((((((bits+BN_BITS2-1))/BN_BITS2)) <= (a)->dmax)?\
700         (a):bn_expand2((a),(bits+BN_BITS2-1)/BN_BITS2))
701 #define bn_wexpand(a,words) (((words) <= (a)->dmax)?(a):bn_expand2((a),(words)))
702 BIGNUM *bn_expand2(BIGNUM *a, int words);
703 #ifndef OPENSSL_NO_DEPRECATED
704 BIGNUM *bn_dup_expand(const BIGNUM *a, int words); /* unused */
705 #endif

707 /* Bignum consistency macros
708  * There is one "API" macro, bn_fix_top(), for stripping leading zeroes from
709  * bignum data after direct manipulations on the data. There is also an
710  * "internal" macro, bn_check_top(), for verifying that there are no leading
711  * zeroes. Unfortunately, some auditing is required due to the fact that
712  * bn_fix_top() has become an overabused duct-tape because bignum data is
713  * occasionally passed around in an inconsistent state. So the following
714  * changes have been made to sort this out;
715  * - bn_fix_top()s implementation has been moved to bn_correct_top()
716  * - if BN_DEBUG isn't defined, bn_fix_top() maps to bn_correct_top(), and
717  *   bn_check_top() is as before.
718  * - if BN_DEBUG *is* defined;
719  *   - bn_check_top() tries to pollute unused words even if the bignum 'top' is
720  *     consistent. (ed: only if BN_DEBUG_RAND is defined)
721  *   - bn_fix_top() maps to bn_check_top() rather than "fixing" anything.
```



```
722  * The idea is to have debug builds flag up inconsistent bignums when they
723  * occur. If that occurs in a bn_fix_top(), we examine the code in question; if
724  * the use of bn_fix_top() was appropriate (ie. it follows directly after code
725  * that manipulates the bignum) it is converted to bn_correct_top(), and if it
726  * was not appropriate, we convert it permanently to bn_check_top() and track
727  * down the cause of the bug. Eventually, no internal code should be using the
728  * bn_fix_top() macro. External applications and libraries should try this with
729  * their own code too, both in terms of building against the openssl headers
730  * with BN_DEBUG defined *and* linking with a version of OpenSSL built with it
731  * defined. This not only improves external code, it provides more test
732  * coverage for openssl's own code.
733  */

735 #ifdef BN_DEBUG

737 /* We only need assert() when debugging */
738 #include <assert.h>

740 #ifdef BN_DEBUG_RAND
741 /* To avoid "make update" cvs wars due to BN_DEBUG, use some tricks */
742 #ifndef RAND_pseudo_bytes
743 int RAND_pseudo_bytes(unsigned char *buf,int num);
744 #define BN_DEBUG_TRIX
745 #endif
746 #define bn_pollute(a) \
747         do { \
748                 const BIGNUM *_bnum1 = (a); \
749                 if(_bnum1->top < _bnum1->dmax) { \
750                         unsigned char _tmp_char; \
751                         /* We cast away const without the compiler knowing, any
752                          * *genuinely* constant variables that aren't mutable \
753                          * wouldn't be constructed with top!=dmax. */ \
754                         BN_ULONG *_not_const; \
755                         memcpy(&_not_const, &_bnum1->d, sizeof(BN_ULONG*)); \
756                         RAND_pseudo_bytes(&_tmp_char, 1); \
757                         memset((unsigned char *)(_not_const + _bnum1->top), _tmp
758                                 (_bnum1->dmax - _bnum1->top) * sizeof(BN_ULONG))
759                 } \
760         } while(0)
761 #ifdef BN_DEBUG_TRIX
762 #undef RAND_pseudo_bytes
763 #endif
764 #else
765 #define bn_pollute(a)
766 #endif
767 #define bn_check_top(a) \
768         do { \
769                 const BIGNUM *_bnum2 = (a); \
770                 if (_bnum2 != NULL) { \
771                         assert((_bnum2->top == 0) || \
772                                 (_bnum2->d[_bnum2->top - 1] != 0)); \
773                         bn_pollute(_bnum2); \
774                 } \
775         } while(0)

777 #define bn_fix_top(a)           bn_check_top(a)

779 #define bn_check_size(bn, bits) bn_wcheck_size(bn, ((bits+BN_BITS2-1))/BN_BITS2)
780 #define bn_wcheck_size(bn, words) \
781         do { \
782                 const BIGNUM *_bnum2 = (bn); \
783                 assert(words <= (_bnum2)->dmax && words >= (_bnum2)->top); \
784         } while(0)

786 #else /* !BN_DEBUG */
```

```
#define bn_pollute(a)
#define bn_check_top(a)
#define bn_fix_top(a)          bn_correct_top(a)
#define bn_check_size(bn, bits)
#define bn_wcheck_size(bn, words)

#endif

#define bn_correct_top(a) \
        { \
        BN_ULONG *ftl; \
        int tmp_top = (a)->top; \
        if (tmp_top > 0) \
                { \
                for (ftl= &((a)->d[tmp_top-1]); tmp_top > 0; tmp_top--) \
                        if (*(ftl--)) break; \
                (a)->top = tmp_top; \
                } \
        bn_pollute(a); \
        }

BN_ULONG bn_mul_add_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w)
BN_ULONG bn_mul_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w);
void     bn_sqr_words(BN_ULONG *rp, const BN_ULONG *ap, int num);
BN_ULONG bn_div_words(BN_ULONG h, BN_ULONG l, BN_ULONG d);
BN_ULONG bn_add_words(BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp,int n
BN_ULONG bn_sub_words(BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp,int n

/* Primes from RFC 2409 */
BIGNUM *get_rfc2409_prime_768(BIGNUM *bn);
BIGNUM *get_rfc2409_prime_1024(BIGNUM *bn);

/* Primes from RFC 3526 */
BIGNUM *get_rfc3526_prime_1536(BIGNUM *bn);
BIGNUM *get_rfc3526_prime_2048(BIGNUM *bn);
BIGNUM *get_rfc3526_prime_3072(BIGNUM *bn);
BIGNUM *get_rfc3526_prime_4096(BIGNUM *bn);
BIGNUM *get_rfc3526_prime_6144(BIGNUM *bn);
BIGNUM *get_rfc3526_prime_8192(BIGNUM *bn);

int BN_bntest_rand(BIGNUM *rnd, int bits, int top,int bottom);

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_BN_strings(void);

/* Error codes for the BN functions. */

/* Function codes. */
#define BN_F_BNRAND                                     127
#define BN_F_BN_BLINDING_CONVERT_EX                     100
#define BN_F_BN_BLINDING_CREATE_PARAM                   128
#define BN_F_BN_BLINDING_INVERT_EX                      101
#define BN_F_BN_BLINDING_NEW                            102
#define BN_F_BN_BLINDING_UPDATE                         103
#define BN_F_BN_BN2DEC                                  104
#define BN_F_BN_BN2HEX                                  105
#define BN_F_BN_CTX_GET                                 116
#define BN_F_BN_CTX_NEW                                 106
#define BN_F_BN_CTX_START                               129
#define BN_F_BN_DIV                                     107
#define BN_F_BN_DIV_NO_BRANCH                           138
#define BN_F_BN_DIV_RECP                                130
#define BN_F_BN_EXP                                     123
```



```
#define BN_F_BN_EXPAND2                                 108
#define BN_F_BN_EXPAND_INTERNAL                         120
#define BN_F_BN_GF2M_MOD                                131
#define BN_F_BN_GF2M_MOD_EXP                            132
#define BN_F_BN_GF2M_MOD_MUL                            133
#define BN_F_BN_GF2M_MOD_SOLVE_QUAD                     134
#define BN_F_BN_GF2M_MOD_SOLVE_QUAD_ARR                 135
#define BN_F_BN_GF2M_MOD_SQR                            136
#define BN_F_BN_GF2M_MOD_SQRT                           137
#define BN_F_BN_MOD_EXP2_MONT                           118
#define BN_F_BN_MOD_EXP_MONT                            109
#define BN_F_BN_MOD_EXP_MONT_CONSTTIME                  124
#define BN_F_BN_MOD_EXP_MONT_WORD                       117
#define BN_F_BN_MOD_EXP_RECP                            125
#define BN_F_BN_MOD_EXP_SIMPLE                          126
#define BN_F_BN_MOD_INVERSE                             110
#define BN_F_BN_MOD_INVERSE_NO_BRANCH                   139
#define BN_F_BN_MOD_LSHIFT_QUICK                        119
#define BN_F_BN_MOD_MUL_RECIPROCAL                      111
#define BN_F_BN_MOD_SQRT                                121
#define BN_F_BN_MPI2BN                                  112
#define BN_F_BN_NEW                                     113
#define BN_F_BN_RAND                                    114
#define BN_F_BN_RAND_RANGE                              122
#define BN_F_BN_USUB                                    115

/* Reason codes. */
#define BN_R_ARG2_LT_ARG3                               100
#define BN_R_BAD_RECIPROCAL                             101
#define BN_R_BIGNUM_TOO_LONG                            114
#define BN_R_CALLED_WITH_EVEN_MODULUS                   102
#define BN_R_DIV_BY_ZERO                                103
#define BN_R_ENCODING_ERROR                             104
#define BN_R_EXPAND_ON_STATIC_BIGNUM_DATA               105
#define BN_R_INPUT_NOT_REDUCED                          110
#define BN_R_INVALID_LENGTH                             106
#define BN_R_INVALID_RANGE                              115
#define BN_R_NOT_A_SQUARE                               111
#define BN_R_NOT_INITIALIZED                            107
#define BN_R_NO_INVERSE                                 108
#define BN_R_NO_SOLUTION                                116
#define BN_R_P_IS_NOT_PRIME                             112
#define BN_R_TOO_MANY_ITERATIONS                        113
#define BN_R_TOO_MANY_TEMPORARY_VARIABLES               109

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4646 Wed Aug 13 19:51:41 2014
new/usr/src/lib/openssl/include/openssl/buffer.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/buffer/buffer.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_BUFFER_H
#define HEADER_BUFFER_H
```



```
#include <openssl/ossl_typ.h>

#ifdef  __cplusplus
extern "C" {
#endif

#include <stddef.h>

#if !defined(NO_SYS_TYPES_H)
#include <sys/types.h>
#endif

/* Already declared in ossl_typ.h */
/* typedef struct buf_mem_st BUF_MEM; */

struct buf_mem_st
        {
        size_t length;  /* current number of bytes */
        char *data;
        size_t max;     /* size of buffer */
        };

BUF_MEM *BUF_MEM_new(void);
void    BUF_MEM_free(BUF_MEM *a);
int     BUF_MEM_grow(BUF_MEM *str, size_t len);
int     BUF_MEM_grow_clean(BUF_MEM *str, size_t len);
char *  BUF_strdup(const char *str);
char *  BUF_strndup(const char *str, size_t siz);
void *  BUF_memdup(const void *data, size_t siz);
void    BUF_reverse(unsigned char *out, const unsigned char *in, size_t siz);

/* safe string functions */
size_t BUF_strlcpy(char *dst,const char *src,size_t siz);
size_t BUF_strlcat(char *dst,const char *src,size_t siz);


/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_BUF_strings(void);

/* Error codes for the BUF functions. */

/* Function codes. */
#define BUF_F_BUF_MEMDUP                                103
#define BUF_F_BUF_MEM_GROW                              100
#define BUF_F_BUF_MEM_GROW_CLEAN                        105
#define BUF_F_BUF_MEM_NEW                               101
#define BUF_F_BUF_STRDUP                                102
#define BUF_F_BUF_STRNDUP                               104

/* Reason codes. */

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4954 Wed Aug 13 19:51:41 2014
new/usr/src/lib/openssl/include/openssl/camellia.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/camellia/camellia.h -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#ifndef HEADER_CAMELLIA_H
#define HEADER_CAMELLIA_H

#include <openssl/opensslconf.h>

#ifdef OPENSSL_NO_CAMELLIA
#error CAMELLIA is disabled.
#endif

#include <stddef.h>
```



```
#define CAMELLIA_ENCRYPT        1
#define CAMELLIA_DECRYPT        0

/* Because array size can't be a const in C, the following two are macros.
   Both sizes are in bytes. */

#ifdef  __cplusplus
extern "C" {
#endif

/* This should be a hidden type, but EVP requires that the size be known */

#define CAMELLIA_BLOCK_SIZE 16
#define CAMELLIA_TABLE_BYTE_LEN 272
#define CAMELLIA_TABLE_WORD_LEN (CAMELLIA_TABLE_BYTE_LEN / 4)

typedef unsigned int KEY_TABLE_TYPE[CAMELLIA_TABLE_WORD_LEN]; /* to match with W

struct camellia_key_st
        {
        union   {
                double d;       /* ensures 64-bit align */
                KEY_TABLE_TYPE rd_key;
                } u;
        int grand_rounds;
        };
typedef struct camellia_key_st CAMELLIA_KEY;

#ifdef OPENSSL_FIPS
int private_Camellia_set_key(const unsigned char *userKey, const int bits,
        CAMELLIA_KEY *key);
#endif
int Camellia_set_key(const unsigned char *userKey, const int bits,
        CAMELLIA_KEY *key);

void Camellia_encrypt(const unsigned char *in, unsigned char *out,
        const CAMELLIA_KEY *key);
void Camellia_decrypt(const unsigned char *in, unsigned char *out,
        const CAMELLIA_KEY *key);

void Camellia_ecb_encrypt(const unsigned char *in, unsigned char *out,
        const CAMELLIA_KEY *key, const int enc);
void Camellia_cbc_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char *ivec, const int enc);
void Camellia_cfb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char *ivec, int *num, const int enc);
void Camellia_cfb1_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char *ivec, int *num, const int enc);
void Camellia_cfb8_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char *ivec, int *num, const int enc);
void Camellia_ofb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char *ivec, int *num);
void Camellia_ctr128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char ivec[CAMELLIA_BLOCK_SIZE],
        unsigned char ecount_buf[CAMELLIA_BLOCK_SIZE],
        unsigned int *num);

#ifdef  __cplusplus
}
```

```
128 #endif

130 #endif /* !HEADER_Camellia_H */
131 #endif /* ! codereview */
```

```
**********************************************************
    4484 Wed Aug 13 19:51:41 2014
new/usr/src/lib/openssl/include/openssl/cast.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/cast.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_CAST_H
#define HEADER_CAST_H
```



```
#ifdef  __cplusplus
extern "C" {
#endif

#include <openssl/opensslconf.h>

#ifdef OPENSSL_NO_CAST
#error CAST is disabled.
#endif

#define CAST_ENCRYPT    1
#define CAST_DECRYPT    0

#define CAST_LONG unsigned int

#define CAST_BLOCK      8
#define CAST_KEY_LENGTH 16

typedef struct cast_key_st
        {
        CAST_LONG data[32];
        int short_key;  /* Use reduced rounds for short key */
        } CAST_KEY;

#ifdef OPENSSL_FIPS
void private_CAST_set_key(CAST_KEY *key, int len, const unsigned char *data);
#endif
void CAST_set_key(CAST_KEY *key, int len, const unsigned char *data);
void CAST_ecb_encrypt(const unsigned char *in, unsigned char *out, const CAST_KE
                      int enc);
void CAST_encrypt(CAST_LONG *data, const CAST_KEY *key);
void CAST_decrypt(CAST_LONG *data, const CAST_KEY *key);
void CAST_cbc_encrypt(const unsigned char *in, unsigned char *out, long length,
                      const CAST_KEY *ks, unsigned char *iv, int enc);
void CAST_cfb64_encrypt(const unsigned char *in, unsigned char *out,
                        long length, const CAST_KEY *schedule, unsigned char *iv
                        int *num, int enc);
void CAST_ofb64_encrypt(const unsigned char *in, unsigned char *out,
                        long length, const CAST_KEY *schedule, unsigned char *iv
                        int *num);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    3242 Wed Aug 13 19:51:41 2014
new/usr/src/lib/openssl/include/openssl/cmac.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cmac/cmac.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */


#ifndef HEADER_CMAC_H
#define HEADER_CMAC_H

#ifdef __cplusplus
extern "C" {
#endif
```



```
#include <openssl/evp.h>

/* Opaque */
typedef struct CMAC_CTX_st CMAC_CTX;

CMAC_CTX *CMAC_CTX_new(void);
void CMAC_CTX_cleanup(CMAC_CTX *ctx);
void CMAC_CTX_free(CMAC_CTX *ctx);
EVP_CIPHER_CTX *CMAC_CTX_get0_cipher_ctx(CMAC_CTX *ctx);
int CMAC_CTX_copy(CMAC_CTX *out, const CMAC_CTX *in);

int CMAC_Init(CMAC_CTX *ctx, const void *key, size_t keylen,
                          const EVP_CIPHER *cipher, ENGINE *impl);
int CMAC_Update(CMAC_CTX *ctx, const void *data, size_t dlen);
int CMAC_Final(CMAC_CTX *ctx, unsigned char *out, size_t *poutlen);
int CMAC_resume(CMAC_CTX *ctx);

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   19919 Wed Aug 13 19:51:41 2014
new/usr/src/lib/openssl/include/openssl/cms.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */


#ifndef HEADER_CMS_H
#define HEADER_CMS_H

#include <openssl/x509.h>

#ifdef OPENSSL_NO_CMS
#error CMS is disabled.
```



```
#endif

#ifdef __cplusplus
extern "C" {
#endif


typedef struct CMS_ContentInfo_st CMS_ContentInfo;
typedef struct CMS_SignerInfo_st CMS_SignerInfo;
typedef struct CMS_CertificateChoices CMS_CertificateChoices;
typedef struct CMS_RevocationInfoChoice_st CMS_RevocationInfoChoice;
typedef struct CMS_RecipientInfo_st CMS_RecipientInfo;
typedef struct CMS_ReceiptRequest_st CMS_ReceiptRequest;
typedef struct CMS_Receipt_st CMS_Receipt;

DECLARE_STACK_OF(CMS_SignerInfo)
DECLARE_STACK_OF(GENERAL_NAMES)
DECLARE_ASN1_FUNCTIONS(CMS_ContentInfo)
DECLARE_ASN1_FUNCTIONS(CMS_ReceiptRequest)
DECLARE_ASN1_PRINT_FUNCTION(CMS_ContentInfo)

#define CMS_SIGNERINFO_ISSUER_SERIAL    0
#define CMS_SIGNERINFO_KEYIDENTIFIER    1

#define CMS_RECIPINFO_TRANS             0
#define CMS_RECIPINFO_AGREE             1
#define CMS_RECIPINFO_KEK               2
#define CMS_RECIPINFO_PASS              3
#define CMS_RECIPINFO_OTHER             4

/* S/MIME related flags */

#define CMS_TEXT                        0x1
#define CMS_NOCERTS                     0x2
#define CMS_NO_CONTENT_VERIFY           0x4
#define CMS_NO_ATTR_VERIFY              0x8
#define CMS_NOSIGS                      \
                        (CMS_NO_CONTENT_VERIFY|CMS_NO_ATTR_VERIFY)
#define CMS_NOINTERN                    0x10
#define CMS_NO_SIGNER_CERT_VERIFY       0x20
#define CMS_NOVERIFY                    0x20
#define CMS_DETACHED                    0x40
#define CMS_BINARY                      0x80
#define CMS_NOATTR                      0x100
#define CMS_NOSMIMECAP                  0x200
#define CMS_NOOLDMIMETYPE               0x400
#define CMS_CRLFEOL                     0x800
#define CMS_STREAM                      0x1000
#define CMS_NOCRL                       0x2000
#define CMS_PARTIAL                     0x4000
#define CMS_REUSE_DIGEST                0x8000
#define CMS_USE_KEYID                   0x10000
#define CMS_DEBUG_DECRYPT               0x20000

const ASN1_OBJECT *CMS_get0_type(CMS_ContentInfo *cms);

BIO *CMS_dataInit(CMS_ContentInfo *cms, BIO *icont);
int CMS_dataFinal(CMS_ContentInfo *cms, BIO *bio);

ASN1_OCTET_STRING **CMS_get0_content(CMS_ContentInfo *cms);
int CMS_is_detached(CMS_ContentInfo *cms);
int CMS_set_detached(CMS_ContentInfo *cms, int detached);

#ifdef HEADER_PEM_H
DECLARE_PEM_rw_const(CMS, CMS_ContentInfo)
#endif
```

```
int CMS_stream(unsigned char ***boundary, CMS_ContentInfo *cms);
CMS_ContentInfo *d2i_CMS_bio(BIO *bp, CMS_ContentInfo **cms);
int i2d_CMS_bio(BIO *bp, CMS_ContentInfo *cms);

BIO *BIO_new_CMS(BIO *out, CMS_ContentInfo *cms);
int i2d_CMS_bio_stream(BIO *out, CMS_ContentInfo *cms, BIO *in, int flags);
int PEM_write_bio_CMS_stream(BIO *out, CMS_ContentInfo *cms, BIO *in, int flags)
CMS_ContentInfo *SMIME_read_CMS(BIO *bio, BIO **bcont);
int SMIME_write_CMS(BIO *bio, CMS_ContentInfo *cms, BIO *data, int flags);

int CMS_final(CMS_ContentInfo *cms, BIO *data, BIO *dcont, unsigned int flags);

CMS_ContentInfo *CMS_sign(X509 *signcert, EVP_PKEY *pkey, STACK_OF(X509) *certs,
                                                BIO *data, unsigned int flags);

CMS_ContentInfo *CMS_sign_receipt(CMS_SignerInfo *si,
                                        X509 *signcert, EVP_PKEY *pkey,
                                        STACK_OF(X509) *certs,
                                        unsigned int flags);

int CMS_data(CMS_ContentInfo *cms, BIO *out, unsigned int flags);
CMS_ContentInfo *CMS_data_create(BIO *in, unsigned int flags);

int CMS_digest_verify(CMS_ContentInfo *cms, BIO *dcont, BIO *out,
                                                        unsigned int flags);
CMS_ContentInfo *CMS_digest_create(BIO *in, const EVP_MD *md,
                                                        unsigned int flags);

int CMS_EncryptedData_decrypt(CMS_ContentInfo *cms,
                                const unsigned char *key, size_t keylen,
                                BIO *dcont, BIO *out, unsigned int flags);

CMS_ContentInfo *CMS_EncryptedData_encrypt(BIO *in, const EVP_CIPHER *cipher,
                                        const unsigned char *key, size_t keylen,
                                        unsigned int flags);

int CMS_EncryptedData_set1_key(CMS_ContentInfo *cms, const EVP_CIPHER *ciph,
                                const unsigned char *key, size_t keylen);

int CMS_verify(CMS_ContentInfo *cms, STACK_OF(X509) *certs,
                  X509_STORE *store, BIO *dcont, BIO *out, unsigned int flags);

int CMS_verify_receipt(CMS_ContentInfo *rcms, CMS_ContentInfo *ocms,
                        STACK_OF(X509) *certs,
                        X509_STORE *store, unsigned int flags);

STACK_OF(X509) *CMS_get0_signers(CMS_ContentInfo *cms);

CMS_ContentInfo *CMS_encrypt(STACK_OF(X509) *certs, BIO *in,
                                const EVP_CIPHER *cipher, unsigned int flags);

int CMS_decrypt(CMS_ContentInfo *cms, EVP_PKEY *pkey, X509 *cert,
                                BIO *dcont, BIO *out,
                                unsigned int flags);

int CMS_decrypt_set1_pkey(CMS_ContentInfo *cms, EVP_PKEY *pk, X509 *cert);
int CMS_decrypt_set1_key(CMS_ContentInfo *cms,
                                unsigned char *key, size_t keylen,
                                unsigned char *id, size_t idlen);
int CMS_decrypt_set1_password(CMS_ContentInfo *cms,
                                unsigned char *pass, ossl_ssize_t passlen);

STACK_OF(CMS_RecipientInfo) *CMS_get0_RecipientInfos(CMS_ContentInfo *cms);
int CMS_RecipientInfo_type(CMS_RecipientInfo *ri);
CMS_ContentInfo *CMS_EnvelopedData_create(const EVP_CIPHER *cipher);
```



```
CMS_RecipientInfo *CMS_add1_recipient_cert(CMS_ContentInfo *cms,
                                                X509 *recip, unsigned int flags);
int CMS_RecipientInfo_set0_pkey(CMS_RecipientInfo *ri, EVP_PKEY *pkey);
int CMS_RecipientInfo_ktri_cert_cmp(CMS_RecipientInfo *ri, X509 *cert);
int CMS_RecipientInfo_ktri_get0_algs(CMS_RecipientInfo *ri,
                                        EVP_PKEY **pk, X509 **recip,
                                        X509_ALGOR **palg);
int CMS_RecipientInfo_ktri_get0_signer_id(CMS_RecipientInfo *ri,
                                        ASN1_OCTET_STRING **keyid,
                                        X509_NAME **issuer, ASN1_INTEGER **sno);

CMS_RecipientInfo *CMS_add0_recipient_key(CMS_ContentInfo *cms, int nid,
                                        unsigned char *key, size_t keylen,
                                        unsigned char *id, size_t idlen,
                                        ASN1_GENERALIZEDTIME *date,
                                        ASN1_OBJECT *otherTypeId,
                                        ASN1_TYPE *otherType);

int CMS_RecipientInfo_kekri_get0_id(CMS_RecipientInfo *ri,
                                        X509_ALGOR **palg,
                                        ASN1_OCTET_STRING **pid,
                                        ASN1_GENERALIZEDTIME **pdate,
                                        ASN1_OBJECT **potherid,
                                        ASN1_TYPE **pothertype);

int CMS_RecipientInfo_set0_key(CMS_RecipientInfo *ri,
                                unsigned char *key, size_t keylen);

int CMS_RecipientInfo_kekri_id_cmp(CMS_RecipientInfo *ri,
                                        const unsigned char *id, size_t idlen);

int CMS_RecipientInfo_set0_password(CMS_RecipientInfo *ri,
                                        unsigned char *pass,
                                        ossl_ssize_t passlen);

CMS_RecipientInfo *CMS_add0_recipient_password(CMS_ContentInfo *cms,
                                        int iter, int wrap_nid, int pbe_nid,
                                        unsigned char *pass,
                                        ossl_ssize_t passlen,
                                        const EVP_CIPHER *kekciph);

int CMS_RecipientInfo_decrypt(CMS_ContentInfo *cms, CMS_RecipientInfo *ri);

int CMS_uncompress(CMS_ContentInfo *cms, BIO *dcont, BIO *out,
                                                        unsigned int flags);
CMS_ContentInfo *CMS_compress(BIO *in, int comp_nid, unsigned int flags);

int CMS_set1_eContentType(CMS_ContentInfo *cms, const ASN1_OBJECT *oid);
const ASN1_OBJECT *CMS_get0_eContentType(CMS_ContentInfo *cms);

CMS_CertificateChoices *CMS_add0_CertificateChoices(CMS_ContentInfo *cms);
int CMS_add0_cert(CMS_ContentInfo *cms, X509 *cert);
int CMS_add1_cert(CMS_ContentInfo *cms, X509 *cert);
STACK_OF(X509) *CMS_get1_certs(CMS_ContentInfo *cms);

CMS_RevocationInfoChoice *CMS_add0_RevocationInfoChoice(CMS_ContentInfo *cms);
int CMS_add0_crl(CMS_ContentInfo *cms, X509_CRL *crl);
int CMS_add1_crl(CMS_ContentInfo *cms, X509_CRL *crl);
STACK_OF(X509_CRL) *CMS_get1_crls(CMS_ContentInfo *cms);

int CMS_SignedData_init(CMS_ContentInfo *cms);
CMS_SignerInfo *CMS_add1_signer(CMS_ContentInfo *cms,
                        X509 *signer, EVP_PKEY *pk, const EVP_MD *md,
                        unsigned int flags);
STACK_OF(CMS_SignerInfo) *CMS_get0_SignerInfos(CMS_ContentInfo *cms);
```

```
260 void CMS_SignerInfo_set1_signer_cert(CMS_SignerInfo *si, X509 *signer);
261 int CMS_SignerInfo_get0_signer_id(CMS_SignerInfo *si,
262                                   ASN1_OCTET_STRING **keyid,
263                                   X509_NAME **issuer, ASN1_INTEGER **sno);
264 int CMS_SignerInfo_cert_cmp(CMS_SignerInfo *si, X509 *cert);
265 int CMS_set1_signers_certs(CMS_ContentInfo *cms, STACK_OF(X509) *certs,
266                            unsigned int flags);
267 void CMS_SignerInfo_get0_algs(CMS_SignerInfo *si, EVP_PKEY **pk, X509 **signer,
268                               X509_ALGOR **pdig, X509_ALGOR **psig);
269 int CMS_SignerInfo_sign(CMS_SignerInfo *si);
270 int CMS_SignerInfo_verify(CMS_SignerInfo *si);
271 int CMS_SignerInfo_verify_content(CMS_SignerInfo *si, BIO *chain);

273 int CMS_add_smimecap(CMS_SignerInfo *si, STACK_OF(X509_ALGOR) *algs);
274 int CMS_add_simple_smimecap(STACK_OF(X509_ALGOR) **algs,
275                             int algnid, int keysize);
276 int CMS_add_standard_smimecap(STACK_OF(X509_ALGOR) **smcap);

278 int CMS_signed_get_attr_count(const CMS_SignerInfo *si);
279 int CMS_signed_get_attr_by_NID(const CMS_SignerInfo *si, int nid,
280                                int lastpos);
281 int CMS_signed_get_attr_by_OBJ(const CMS_SignerInfo *si, ASN1_OBJECT *obj,
282                                int lastpos);
283 X509_ATTRIBUTE *CMS_signed_get_attr(const CMS_SignerInfo *si, int loc);
284 X509_ATTRIBUTE *CMS_signed_delete_attr(CMS_SignerInfo *si, int loc);
285 int CMS_signed_add1_attr(CMS_SignerInfo *si, X509_ATTRIBUTE *attr);
286 int CMS_signed_add1_attr_by_OBJ(CMS_SignerInfo *si,
287                                 const ASN1_OBJECT *obj, int type,
288                                 const void *bytes, int len);
289 int CMS_signed_add1_attr_by_NID(CMS_SignerInfo *si,
290                                 int nid, int type,
291                                 const void *bytes, int len);
292 int CMS_signed_add1_attr_by_txt(CMS_SignerInfo *si,
293                                 const char *attrname, int type,
294                                 const void *bytes, int len);
295 void *CMS_signed_get0_data_by_OBJ(CMS_SignerInfo *si, ASN1_OBJECT *oid,
296                                   int lastpos, int type);

298 int CMS_unsigned_get_attr_count(const CMS_SignerInfo *si);
299 int CMS_unsigned_get_attr_by_NID(const CMS_SignerInfo *si, int nid,
300                                  int lastpos);
301 int CMS_unsigned_get_attr_by_OBJ(const CMS_SignerInfo *si, ASN1_OBJECT *obj,
302                                  int lastpos);
303 X509_ATTRIBUTE *CMS_unsigned_get_attr(const CMS_SignerInfo *si, int loc);
304 X509_ATTRIBUTE *CMS_unsigned_delete_attr(CMS_SignerInfo *si, int loc);
305 int CMS_unsigned_add1_attr(CMS_SignerInfo *si, X509_ATTRIBUTE *attr);
306 int CMS_unsigned_add1_attr_by_OBJ(CMS_SignerInfo *si,
307                                   const ASN1_OBJECT *obj, int type,
308                                   const void *bytes, int len);
309 int CMS_unsigned_add1_attr_by_NID(CMS_SignerInfo *si,
310                                   int nid, int type,
311                                   const void *bytes, int len);
312 int CMS_unsigned_add1_attr_by_txt(CMS_SignerInfo *si,
313                                   const char *attrname, int type,
314                                   const void *bytes, int len);
315 void *CMS_unsigned_get0_data_by_OBJ(CMS_SignerInfo *si, ASN1_OBJECT *oid,
316                                     int lastpos, int type);

318 #ifdef HEADER_X509V3_H

320 int CMS_get1_ReceiptRequest(CMS_SignerInfo *si, CMS_ReceiptRequest **prr);
321 CMS_ReceiptRequest *CMS_ReceiptRequest_create0(unsigned char *id, int idlen,
322                                int allorfirst,
323                                STACK_OF(GENERAL_NAMES) *receiptList,
324                                STACK_OF(GENERAL_NAMES) *receiptsTo);
325 int CMS_add1_ReceiptRequest(CMS_SignerInfo *si, CMS_ReceiptRequest *rr);
```



```
326 void CMS_ReceiptRequest_get0_values(CMS_ReceiptRequest *rr,
327                                     ASN1_STRING **pcid,
328                                     int *pallorfirst,
329                                     STACK_OF(GENERAL_NAMES) **plist,
330                                     STACK_OF(GENERAL_NAMES) **prto);

332 #endif

334 /* BEGIN ERROR CODES */
335 /* The following lines are auto generated by the script mkerr.pl. Any changes
336  * made after this point may be overwritten when the script is next run.
337  */
338 void ERR_load_CMS_strings(void);

340 /* Error codes for the CMS functions. */

342 /* Function codes. */
343 #define CMS_F_CHECK_CONTENT                              99
344 #define CMS_F_CMS_ADD0_CERT                              164
345 #define CMS_F_CMS_ADD0_RECIPIENT_KEY                     100
346 #define CMS_F_CMS_ADD0_RECIPIENT_PASSWORD                165
347 #define CMS_F_CMS_ADD1_RECEIPTREQUEST                    158
348 #define CMS_F_CMS_ADD1_RECIPIENT_CERT                    101
349 #define CMS_F_CMS_ADD1_SIGNER                            102
350 #define CMS_F_CMS_ADD1_SIGNINGTIME                       103
351 #define CMS_F_CMS_COMPRESS                               104
352 #define CMS_F_CMS_COMPRESSEDDATA_CREATE                  105
353 #define CMS_F_CMS_COMPRESSEDDATA_INIT_BIO                106
354 #define CMS_F_CMS_COPY_CONTENT                           107
355 #define CMS_F_CMS_COPY_MESSAGEDIGEST                     108
356 #define CMS_F_CMS_DATA                                   109
357 #define CMS_F_CMS_DATAFINAL                              110
358 #define CMS_F_CMS_DATAINIT                               111
359 #define CMS_F_CMS_DECRYPT                                112
360 #define CMS_F_CMS_DECRYPT_SET1_KEY                       113
361 #define CMS_F_CMS_DECRYPT_SET1_PASSWORD                  166
362 #define CMS_F_CMS_DECRYPT_SET1_PKEY                      114
363 #define CMS_F_CMS_DIGESTALGORITHM_FIND_CTX               115
364 #define CMS_F_CMS_DIGESTALGORITHM_INIT_BIO               116
365 #define CMS_F_CMS_DIGESTEDDATA_DO_FINAL                  117
366 #define CMS_F_CMS_DIGEST_VERIFY                          118
367 #define CMS_F_CMS_ENCODE_RECEIPT                         161
368 #define CMS_F_CMS_ENCRYPT                                119
369 #define CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO              120
370 #define CMS_F_CMS_ENCRYPTEDDATA_DECRYPT                  121
371 #define CMS_F_CMS_ENCRYPTEDDATA_ENCRYPT                  122
372 #define CMS_F_CMS_ENCRYPTEDDATA_SET1_KEY                 123
373 #define CMS_F_CMS_ENVELOPEDDATA_CREATE                   124
374 #define CMS_F_CMS_ENVELOPEDDATA_INIT_BIO                 125
375 #define CMS_F_CMS_ENVELOPED_DATA_INIT                    126
376 #define CMS_F_CMS_FINAL                                  127
377 #define CMS_F_CMS_GET0_CERTIFICATE_CHOICES               128
378 #define CMS_F_CMS_GET0_CONTENT                           129
379 #define CMS_F_CMS_GET0_ECONTENT_TYPE                     130
380 #define CMS_F_CMS_GET0_ENVELOPED                         131
381 #define CMS_F_CMS_GET0_REVOCATION_CHOICES                132
382 #define CMS_F_CMS_GET0_SIGNED                            133
383 #define CMS_F_CMS_MSGSIGDIGEST_ADD1                      162
384 #define CMS_F_CMS_RECEIPTREQUEST_CREATE0                 159
385 #define CMS_F_CMS_RECEIPT_VERIFY                         160
386 #define CMS_F_CMS_RECIPIENTINFO_DECRYPT                  134
387 #define CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT            135
388 #define CMS_F_CMS_RECIPIENTINFO_KEKRI_ENCRYPT            136
389 #define CMS_F_CMS_RECIPIENTINFO_KEKRI_GET0_ID            137
390 #define CMS_F_CMS_RECIPIENTINFO_KEKRI_ID_CMP             138
391 #define CMS_F_CMS_RECIPIENTINFO_KTRI_CERT_CMP            139
```

```
392 #define CMS_F_CMS_RECIPIENTINFO_KTRI_DECRYPT             140
393 #define CMS_F_CMS_RECIPIENTINFO_KTRI_ENCRYPT             141
394 #define CMS_F_CMS_RECIPIENTINFO_KTRI_GET0_ALGS           142
395 #define CMS_F_CMS_RECIPIENTINFO_KTRI_GET0_SIGNER_ID      143
396 #define CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT               167
397 #define CMS_F_CMS_RECIPIENTINFO_SET0_KEY                 144
398 #define CMS_F_CMS_RECIPIENTINFO_SET0_PASSWORD            168
399 #define CMS_F_CMS_RECIPIENTINFO_SET0_PKEY                145
400 #define CMS_F_CMS_SET1_SIGNERIDENTIFIER                  146
401 #define CMS_F_CMS_SET_DETACHED                           147
402 #define CMS_F_CMS_SIGN                                   148
403 #define CMS_F_CMS_SIGNED_DATA_INIT                       149
404 #define CMS_F_CMS_SIGNERINFO_CONTENT_SIGN                150
405 #define CMS_F_CMS_SIGNERINFO_SIGN                        151
406 #define CMS_F_CMS_SIGNERINFO_VERIFY                      152
407 #define CMS_F_CMS_SIGNERINFO_VERIFY_CERT                 153
408 #define CMS_F_CMS_SIGNERINFO_VERIFY_CONTENT              154
409 #define CMS_F_CMS_SIGN_RECEIPT                           163
410 #define CMS_F_CMS_STREAM                                 155
411 #define CMS_F_CMS_UNCOMPRESS                             156
412 #define CMS_F_CMS_VERIFY                                 157

414 /* Reason codes. */
415 #define CMS_R_ADD_SIGNER_ERROR                           99
416 #define CMS_R_CERTIFICATE_ALREADY_PRESENT                175
417 #define CMS_R_CERTIFICATE_HAS_NO_KEYID                   160
418 #define CMS_R_CERTIFICATE_VERIFY_ERROR                   100
419 #define CMS_R_CIPHER_INITIALISATION_ERROR                101
420 #define CMS_R_CIPHER_PARAMETER_INITIALISATION_ERROR      102
421 #define CMS_R_CMS_DATAFINAL_ERROR                        103
422 #define CMS_R_CMS_LIB                                    104
423 #define CMS_R_CONTENTIDENTIFIER_MISMATCH                 170
424 #define CMS_R_CONTENT_NOT_FOUND                          105
425 #define CMS_R_CONTENT_TYPE_MISMATCH                      171
426 #define CMS_R_CONTENT_TYPE_NOT_COMPRESSED_DATA           106
427 #define CMS_R_CONTENT_TYPE_NOT_ENVELOPED_DATA            107
428 #define CMS_R_CONTENT_TYPE_NOT_SIGNED_DATA               108
429 #define CMS_R_CONTENT_VERIFY_ERROR                       109
430 #define CMS_R_CTRL_ERROR                                 110
431 #define CMS_R_CTRL_FAILURE                               111
432 #define CMS_R_DECRYPT_ERROR                              112
433 #define CMS_R_DIGEST_ERROR                               161
434 #define CMS_R_ERROR_GETTING_PUBLIC_KEY                   113
435 #define CMS_R_ERROR_READING_MESSAGEDIGEST_ATTRIBUTE      114
436 #define CMS_R_ERROR_SETTING_KEY                          115
437 #define CMS_R_ERROR_SETTING_RECIPIENTINFO                116
438 #define CMS_R_INVALID_ENCRYPTED_KEY_LENGTH               117
439 #define CMS_R_INVALID_KEY_ENCRYPTION_PARAMETER           176
440 #define CMS_R_INVALID_KEY_LENGTH                         118
441 #define CMS_R_MD_BIO_INIT_ERROR                          119
442 #define CMS_R_MESSAGEDIGEST_ATTRIBUTE_WRONG_LENGTH       120
443 #define CMS_R_MESSAGEDIGEST_WRONG_LENGTH                 121
444 #define CMS_R_MSGSIGDIGEST_ERROR                         172
445 #define CMS_R_MSGSIGDIGEST_VERIFICATION_FAILURE          162
446 #define CMS_R_MSGSIGDIGEST_WRONG_LENGTH                  163
447 #define CMS_R_NEED_ONE_SIGNER                            164
448 #define CMS_R_NOT_A_SIGNED_RECEIPT                       165
449 #define CMS_R_NOT_ENCRYPTED_DATA                         122
450 #define CMS_R_NOT_KEK                                    123
451 #define CMS_R_NOT_KEY_TRANSPORT                          124
452 #define CMS_R_NOT_PWRI                                   177
453 #define CMS_R_NOT_SUPPORTED_FOR_THIS_KEY_TYPE            125
454 #define CMS_R_NO_CIPHER                                  126
455 #define CMS_R_NO_CONTENT                                 127
456 #define CMS_R_NO_CONTENT_TYPE                            173
457 #define CMS_R_NO_DEFAULT_DIGEST                          128
```



```
458 #define CMS_R_NO_DIGEST_SET                              129
459 #define CMS_R_NO_KEY                                     130
460 #define CMS_R_NO_KEY_OR_CERT                             174
461 #define CMS_R_NO_MATCHING_DIGEST                         131
462 #define CMS_R_NO_MATCHING_RECIPIENT                      132
463 #define CMS_R_NO_MATCHING_SIGNATURE                      166
464 #define CMS_R_NO_MSGSIGDIGEST                            167
465 #define CMS_R_NO_PASSWORD                                178
466 #define CMS_R_NO_PRIVATE_KEY                             133
467 #define CMS_R_NO_PUBLIC_KEY                              134
468 #define CMS_R_NO_RECEIPT_REQUEST                         168
469 #define CMS_R_NO_SIGNERS                                 135
470 #define CMS_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE     136
471 #define CMS_R_RECEIPT_DECODE_ERROR                       169
472 #define CMS_R_RECIPIENT_ERROR                            137
473 #define CMS_R_SIGNER_CERTIFICATE_NOT_FOUND               138
474 #define CMS_R_SIGNFINAL_ERROR                            139
475 #define CMS_R_SMIME_TEXT_ERROR                           140
476 #define CMS_R_STORE_INIT_ERROR                           141
477 #define CMS_R_TYPE_NOT_COMPRESSED_DATA                   142
478 #define CMS_R_TYPE_NOT_DATA                              143
479 #define CMS_R_TYPE_NOT_DIGESTED_DATA                     144
480 #define CMS_R_TYPE_NOT_ENCRYPTED_DATA                    145
481 #define CMS_R_TYPE_NOT_ENVELOPED_DATA                    146
482 #define CMS_R_UNABLE_TO_FINALIZE_CONTEXT                 147
483 #define CMS_R_UNKNOWN_CIPHER                             148
484 #define CMS_R_UNKNOWN_DIGEST_ALGORIHM                    149
485 #define CMS_R_UNKNOWN_ID                                 150
486 #define CMS_R_UNSUPPORTED_COMPRESSION_ALGORITHM          151
487 #define CMS_R_UNSUPPORTED_CONTENT_TYPE                   152
488 #define CMS_R_UNSUPPORTED_KEK_ALGORITHM                  153
489 #define CMS_R_UNSUPPORTED_KEY_ENCRYPTION_ALGORITHM       179
490 #define CMS_R_UNSUPPORTED_RECIPIENT_TYPE                 154
491 #define CMS_R_UNSUPPORTED_RECPIENTINFO_TYPE              155
492 #define CMS_R_UNSUPPORTED_TYPE                           156
493 #define CMS_R_UNWRAP_ERROR                               157
494 #define CMS_R_UNWRAP_FAILURE                             180
495 #define CMS_R_VERIFICATION_FAILURE                       158
496 #define CMS_R_WRAP_ERROR                                 159

498 #ifdef  __cplusplus
499 }
500 #endif
501 #endif
502 #endif /* ! codereview */
```

```
**********************************************************
    1978 Wed Aug 13 19:51:41 2014
new/usr/src/lib/openssl/include/openssl/comp.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#ifndef HEADER_COMP_H
#define HEADER_COMP_H

#include <openssl/crypto.h>

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct comp_ctx_st COMP_CTX;

typedef struct comp_method_st
        {
        int type;               /* NID for compression library */
        const char *name;       /* A text string to identify the library */
        int (*init)(COMP_CTX *ctx);
        void (*finish)(COMP_CTX *ctx);
        int (*compress)(COMP_CTX *ctx,
                        unsigned char *out, unsigned int olen,
                        unsigned char *in, unsigned int ilen);
        int (*expand)(COMP_CTX *ctx,
                      unsigned char *out, unsigned int olen,
                      unsigned char *in, unsigned int ilen);
        /* The following two do NOTHING, but are kept for backward compatibility
        long (*ctrl)(void);
        long (*callback_ctrl)(void);
        } COMP_METHOD;

struct comp_ctx_st
        {
        COMP_METHOD *meth;
        unsigned long compress_in;
        unsigned long compress_out;
        unsigned long expand_in;
        unsigned long expand_out;

        CRYPTO_EX_DATA  ex_data;
        };


COMP_CTX *COMP_CTX_new(COMP_METHOD *meth);
void COMP_CTX_free(COMP_CTX *ctx);
int COMP_compress_block(COMP_CTX *ctx, unsigned char *out, int olen,
        unsigned char *in, int ilen);
int COMP_expand_block(COMP_CTX *ctx, unsigned char *out, int olen,
        unsigned char *in, int ilen);
COMP_METHOD *COMP_rle(void );
COMP_METHOD *COMP_zlib(void );
void COMP_zlib_cleanup(void);

#ifdef HEADER_BIO_H
#ifdef ZLIB
BIO_METHOD *BIO_f_zlib(void);
#endif
#endif

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
```



```
void ERR_load_COMP_strings(void);

/* Error codes for the COMP functions. */

/* Function codes. */
#define COMP_F_BIO_ZLIB_FLUSH                           99
#define COMP_F_BIO_ZLIB_NEW                             100
#define COMP_F_BIO_ZLIB_READ                            101
#define COMP_F_BIO_ZLIB_WRITE                           102

/* Reason codes. */
#define COMP_R_ZLIB_DEFLATE_ERROR                       99
#define COMP_R_ZLIB_INFLATE_ERROR                       100
#define COMP_R_ZLIB_NOT_SUPPORTED                       101

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    9840 Wed Aug 13 19:51:42 2014
new/usr/src/lib/openssl/include/openssl/conf.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/conf/conf.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef  HEADER_CONF_H
#define HEADER_CONF_H
```



```
#include <openssl/bio.h>
#include <openssl/lhash.h>
#include <openssl/stack.h>
#include <openssl/safestack.h>
#include <openssl/e_os2.h>

#include <openssl/ossl_typ.h>

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct
        {
        char *section;
        char *name;
        char *value;
        } CONF_VALUE;

DECLARE_STACK_OF(CONF_VALUE)
DECLARE_LHASH_OF(CONF_VALUE);

struct conf_st;
struct conf_method_st;
typedef struct conf_method_st CONF_METHOD;

struct conf_method_st
        {
        const char *name;
        CONF *(*create)(CONF_METHOD *meth);
        int (*init)(CONF *conf);
        int (*destroy)(CONF *conf);
        int (*destroy_data)(CONF *conf);
        int (*load_bio)(CONF *conf, BIO *bp, long *eline);
        int (*dump)(const CONF *conf, BIO *bp);
        int (*is_number)(const CONF *conf, char c);
        int (*to_int)(const CONF *conf, char c);
        int (*load)(CONF *conf, const char *name, long *eline);
        };

/* Module definitions */

typedef struct conf_imodule_st CONF_IMODULE;
typedef struct conf_module_st CONF_MODULE;

DECLARE_STACK_OF(CONF_MODULE)
DECLARE_STACK_OF(CONF_IMODULE)

/* DSO module function typedefs */
typedef int conf_init_func(CONF_IMODULE *md, const CONF *cnf);
typedef void conf_finish_func(CONF_IMODULE *md);

#define CONF_MFLAGS_IGNORE_ERRORS       0x1
#define CONF_MFLAGS_IGNORE_RETURN_CODES 0x2
#define CONF_MFLAGS_SILENT              0x4
#define CONF_MFLAGS_NO_DSO              0x8
#define CONF_MFLAGS_IGNORE_MISSING_FILE 0x10
#define CONF_MFLAGS_DEFAULT_SECTION     0x20

int CONF_set_default_method(CONF_METHOD *meth);
void CONF_set_nconf(CONF *conf,LHASH_OF(CONF_VALUE) *hash);
LHASH_OF(CONF_VALUE) *CONF_load(LHASH_OF(CONF_VALUE) *conf,const char *file,
                                long *eline);
#ifndef OPENSSL_NO_FP_API
LHASH_OF(CONF_VALUE) *CONF_load_fp(LHASH_OF(CONF_VALUE) *conf, FILE *fp,
                                   long *eline);
```

```
#endif
LHASH_OF(CONF_VALUE) *CONF_load_bio(LHASH_OF(CONF_VALUE) *conf, BIO *bp,long *el
STACK_OF(CONF_VALUE) *CONF_get_section(LHASH_OF(CONF_VALUE) *conf,
                                       const char *section);
char *CONF_get_string(LHASH_OF(CONF_VALUE) *conf,const char *group,
                      const char *name);
long CONF_get_number(LHASH_OF(CONF_VALUE) *conf,const char *group,
                     const char *name);
void CONF_free(LHASH_OF(CONF_VALUE) *conf);
int CONF_dump_fp(LHASH_OF(CONF_VALUE) *conf, FILE *out);
int CONF_dump_bio(LHASH_OF(CONF_VALUE) *conf, BIO *out);

void OPENSSL_config(const char *config_name);
void OPENSSL_no_config(void);

/* New conf code.  The semantics are different from the functions above.
   If that wasn't the case, the above functions would have been replaced */

struct conf_st
        {
        CONF_METHOD *meth;
        void *meth_data;
        LHASH_OF(CONF_VALUE) *data;
        };

CONF *NCONF_new(CONF_METHOD *meth);
CONF_METHOD *NCONF_default(void);
CONF_METHOD *NCONF_WIN32(void);
#if 0 /* Just to give you an idea of what I have in mind */
CONF_METHOD *NCONF_XML(void);
#endif
void NCONF_free(CONF *conf);
void NCONF_free_data(CONF *conf);

int NCONF_load(CONF *conf,const char *file,long *eline);
#ifndef OPENSSL_NO_FP_API
int NCONF_load_fp(CONF *conf, FILE *fp,long *eline);
#endif
int NCONF_load_bio(CONF *conf, BIO *bp,long *eline);
STACK_OF(CONF_VALUE) *NCONF_get_section(const CONF *conf,const char *section);
char *NCONF_get_string(const CONF *conf,const char *group,const char *name);
int NCONF_get_number_e(const CONF *conf,const char *group,const char *name,
                       long *result);
int NCONF_dump_fp(const CONF *conf, FILE *out);
int NCONF_dump_bio(const CONF *conf, BIO *out);

#if 0 /* The following function has no error checking,
         and should therefore be avoided */
long NCONF_get_number(CONF *conf,char *group,char *name);
#else
#define NCONF_get_number(c,g,n,r) NCONF_get_number_e(c,g,n,r)
#endif

/* Module functions */

int CONF_modules_load(const CONF *cnf, const char *appname,
                      unsigned long flags);
int CONF_modules_load_file(const char *filename, const char *appname,
                           unsigned long flags);
void CONF_modules_unload(int all);
void CONF_modules_finish(void);
void CONF_modules_free(void);
int CONF_module_add(const char *name, conf_init_func *ifunc,
                    conf_finish_func *ffunc);

const char *CONF_imodule_get_name(const CONF_IMODULE *md);
```



```
const char *CONF_imodule_get_value(const CONF_IMODULE *md);
void *CONF_imodule_get_usr_data(const CONF_IMODULE *md);
void CONF_imodule_set_usr_data(CONF_IMODULE *md, void *usr_data);
CONF_MODULE *CONF_imodule_get_module(const CONF_IMODULE *md);
unsigned long CONF_imodule_get_flags(const CONF_IMODULE *md);
void CONF_imodule_set_flags(CONF_IMODULE *md, unsigned long flags);
void *CONF_module_get_usr_data(CONF_MODULE *pmod);
void CONF_module_set_usr_data(CONF_MODULE *pmod, void *usr_data);

char *CONF_get1_default_config_file(void);

int CONF_parse_list(const char *list, int sep, int nospc,
        int (*list_cb)(const char *elem, int len, void *usr), void *arg);

void OPENSSL_load_builtin_modules(void);

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_CONF_strings(void);

/* Error codes for the CONF functions. */

/* Function codes. */
#define CONF_F_CONF_DUMP_FP                             104
#define CONF_F_CONF_LOAD                                100
#define CONF_F_CONF_LOAD_BIO                            102
#define CONF_F_CONF_LOAD_FP                             103
#define CONF_F_CONF_MODULES_LOAD                        116
#define CONF_F_CONF_PARSE_LIST                          119
#define CONF_F_DEF_LOAD                                 120
#define CONF_F_DEF_LOAD_BIO                             121
#define CONF_F_MODULE_INIT                              115
#define CONF_F_MODULE_LOAD_DSO                          117
#define CONF_F_MODULE_RUN                               118
#define CONF_F_NCONF_DUMP_BIO                           105
#define CONF_F_NCONF_DUMP_FP                            106
#define CONF_F_NCONF_GET_NUMBER                         107
#define CONF_F_NCONF_GET_NUMBER_E                       112
#define CONF_F_NCONF_GET_SECTION                        108
#define CONF_F_NCONF_GET_STRING                         109
#define CONF_F_NCONF_LOAD                               113
#define CONF_F_NCONF_LOAD_BIO                           110
#define CONF_F_NCONF_LOAD_FP                            114
#define CONF_F_NCONF_NEW                                111
#define CONF_F_STR_COPY                                 101

/* Reason codes. */
#define CONF_R_ERROR_LOADING_DSO                        110
#define CONF_R_LIST_CANNOT_BE_NULL                      115
#define CONF_R_MISSING_CLOSE_SQUARE_BRACKET             100
#define CONF_R_MISSING_EQUAL_SIGN                       101
#define CONF_R_MISSING_FINISH_FUNCTION                  111
#define CONF_R_MISSING_INIT_FUNCTION                    112
#define CONF_R_MODULE_INITIALIZATION_ERROR              109
#define CONF_R_NO_CLOSE_BRACE                           102
#define CONF_R_NO_CONF                                  105
#define CONF_R_NO_CONF_OR_ENVIRONMENT_VARIABLE          106
#define CONF_R_NO_SECTION                               107
#define CONF_R_NO_SUCH_FILE                             114
#define CONF_R_NO_VALUE                                 108
#define CONF_R_UNABLE_TO_CREATE_NEW_SECTION             103
#define CONF_R_UNKNOWN_MODULE_NAME                      113
#define CONF_R_VARIABLE_HAS_NO_VALUE                    104
```

```
260 #ifdef  __cplusplus
261 }
262 #endif
263 #endif
264 #endif /* ! codereview */
```

```
**********************************************************
    4073 Wed Aug 13 19:51:42 2014
new/usr/src/lib/openssl/include/openssl/conf_api.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* conf_api.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef  HEADER_CONF_API_H
#define HEADER_CONF_API_H
```



```
#include <openssl/lhash.h>
#include <openssl/conf.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* Up until OpenSSL 0.9.5a, this was new_section */
CONF_VALUE *_CONF_new_section(CONF *conf, const char *section);
/* Up until OpenSSL 0.9.5a, this was get_section */
CONF_VALUE *_CONF_get_section(const CONF *conf, const char *section);
/* Up until OpenSSL 0.9.5a, this was CONF_get_section */
STACK_OF(CONF_VALUE) *_CONF_get_section_values(const CONF *conf,
                                               const char *section);

int _CONF_add_string(CONF *conf, CONF_VALUE *section, CONF_VALUE *value);
char *_CONF_get_string(const CONF *conf, const char *section,
                       const char *name);
long _CONF_get_number(const CONF *conf, const char *section, const char *name);

int _CONF_new_data(CONF *conf);
void _CONF_free_data(CONF *conf);

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   24328 Wed Aug 13 19:51:42 2014
new/usr/src/lib/openssl/include/openssl/crypto.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/crypto.h */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
```



```
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */

#ifndef HEADER_CRYPTO_H
#define HEADER_CRYPTO_H

#include <stdlib.h>

#include <openssl/e_os2.h>

#ifndef OPENSSL_NO_FP_API
#include <stdio.h>
#endif
```

```
#include <openssl/stack.h>
#include <openssl/safestack.h>
#include <openssl/opensslv.h>
#include <openssl/ossl_typ.h>

#ifdef CHARSET_EBCDIC
#include <openssl/ebcdic.h>
#endif

/* Resolve problems on some operating systems with symbol names that clash
   one way or another */
#include <openssl/symhacks.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* Backward compatibility to SSLeay */
/* This is more to be used to check the correct DLL is being used
 * in the MS world. */
#define SSLEAY_VERSION_NUMBER   OPENSSL_VERSION_NUMBER
#define SSLEAY_VERSION          0
/* #define SSLEAY_OPTIONS       1 no longer supported */
#define SSLEAY_CFLAGS           2
#define SSLEAY_BUILT_ON         3
#define SSLEAY_PLATFORM         4
#define SSLEAY_DIR              5

/* Already declared in ossl_typ.h */
#if 0
typedef struct crypto_ex_data_st CRYPTO_EX_DATA;
/* Called when a new object is created */
typedef int CRYPTO_EX_new(void *parent, void *ptr, CRYPTO_EX_DATA *ad,
                          int idx, long argl, void *argp);
/* Called when an object is free()ed */
typedef void CRYPTO_EX_free(void *parent, void *ptr, CRYPTO_EX_DATA *ad,
                            int idx, long argl, void *argp);
/* Called when we need to dup an object */
typedef int CRYPTO_EX_dup(CRYPTO_EX_DATA *to, CRYPTO_EX_DATA *from, void *from_d
                          int idx, long argl, void *argp);
#endif

/* A generic structure to pass assorted data in a expandable way */
typedef struct openssl_item_st
        {
        int code;
        void *value;            /* Not used for flag attributes */
        size_t value_size;      /* Max size of value for output, length for inpu
        size_t *value_length;   /* Returned length of value for output */
        } OPENSSL_ITEM;


/* When changing the CRYPTO_LOCK_* list, be sure to maintin the text lock
 * names in cryptlib.c
 */

#define CRYPTO_LOCK_ERR                 1
#define CRYPTO_LOCK_EX_DATA             2
#define CRYPTO_LOCK_X509                3
#define CRYPTO_LOCK_X509_INFO           4
#define CRYPTO_LOCK_X509_PKEY           5
#define CRYPTO_LOCK_X509_CRL            6
#define CRYPTO_LOCK_X509_REQ            7
#define CRYPTO_LOCK_DSA                 8
#define CRYPTO_LOCK_RSA                 9
#define CRYPTO_LOCK_EVP_PKEY            10
```



```
#define CRYPTO_LOCK_X509_STORE          11
#define CRYPTO_LOCK_SSL_CTX             12
#define CRYPTO_LOCK_SSL_CERT            13
#define CRYPTO_LOCK_SSL_SESSION         14
#define CRYPTO_LOCK_SSL_SESS_CERT       15
#define CRYPTO_LOCK_SSL                 16
#define CRYPTO_LOCK_SSL_METHOD          17
#define CRYPTO_LOCK_RAND                18
#define CRYPTO_LOCK_RAND2               19
#define CRYPTO_LOCK_MALLOC              20
#define CRYPTO_LOCK_BIO                 21
#define CRYPTO_LOCK_GETHOSTBYNAME       22
#define CRYPTO_LOCK_GETSERVBYNAME       23
#define CRYPTO_LOCK_READDIR             24
#define CRYPTO_LOCK_RSA_BLINDING        25
#define CRYPTO_LOCK_DH                  26
#define CRYPTO_LOCK_MALLOC2             27
#define CRYPTO_LOCK_DSO                 28
#define CRYPTO_LOCK_DYNLOCK             29
#define CRYPTO_LOCK_ENGINE              30
#define CRYPTO_LOCK_UI                  31
#define CRYPTO_LOCK_ECDSA               32
#define CRYPTO_LOCK_EC                  33
#define CRYPTO_LOCK_ECDH                34
#define CRYPTO_LOCK_BN                  35
#define CRYPTO_LOCK_EC_PRE_COMP         36
#define CRYPTO_LOCK_STORE               37
#define CRYPTO_LOCK_COMP                38
#define CRYPTO_LOCK_FIPS                39
#define CRYPTO_LOCK_FIPS2               40
#define CRYPTO_NUM_LOCKS                41

#define CRYPTO_LOCK             1
#define CRYPTO_UNLOCK           2
#define CRYPTO_READ             4
#define CRYPTO_WRITE            8

#ifndef OPENSSL_NO_LOCKING
#ifndef CRYPTO_w_lock
#define CRYPTO_w_lock(type)     \
        CRYPTO_lock(CRYPTO_LOCK|CRYPTO_WRITE,type,__FILE__,__LINE__)
#define CRYPTO_w_unlock(type)   \
        CRYPTO_lock(CRYPTO_UNLOCK|CRYPTO_WRITE,type,__FILE__,__LINE__)
#define CRYPTO_r_lock(type)     \
        CRYPTO_lock(CRYPTO_LOCK|CRYPTO_READ,type,__FILE__,__LINE__)
#define CRYPTO_r_unlock(type)   \
        CRYPTO_lock(CRYPTO_UNLOCK|CRYPTO_READ,type,__FILE__,__LINE__)
#define CRYPTO_add(addr,amount,type)    \
        CRYPTO_add_lock(addr,amount,type,__FILE__,__LINE__)
#endif
#else
#define CRYPTO_w_lock(a)
#define CRYPTO_w_unlock(a)
#define CRYPTO_r_lock(a)
#define CRYPTO_r_unlock(a)
#define CRYPTO_add(a,b,c)       ((*(a))+=(b))
#endif

/* Some applications as well as some parts of OpenSSL need to allocate
   and deallocate locks in a dynamic fashion.  The following typedef
   makes this possible in a type-safe manner.  */
/* struct CRYPTO_dynlock_value has to be defined by the application. */
typedef struct
        {
        int references;
        struct CRYPTO_dynlock_value *data;
```

```
260         } CRYPTO_dynlock;


263 /* The following can be used to detect memory leaks in the SSLeay library.
264  * It used, it turns on malloc checking */

266 #define CRYPTO_MEM_CHECK_OFF    0x0     /* an enume */
267 #define CRYPTO_MEM_CHECK_ON     0x1     /* a bit */
268 #define CRYPTO_MEM_CHECK_ENABLE 0x2     /* a bit */
269 #define CRYPTO_MEM_CHECK_DISABLE 0x3    /* an enume */

271 /* The following are bit values to turn on or off options connected to the
272  * malloc checking functionality */

274 /* Adds time to the memory checking information */
275 #define V_CRYPTO_MDEBUG_TIME    0x1 /* a bit */
276 /* Adds thread number to the memory checking information */
277 #define V_CRYPTO_MDEBUG_THREAD  0x2 /* a bit */

279 #define V_CRYPTO_MDEBUG_ALL (V_CRYPTO_MDEBUG_TIME | V_CRYPTO_MDEBUG_THREAD)


282 /* predec of the BIO type */
283 typedef struct bio_st BIO_dummy;

285 struct crypto_ex_data_st
286         {
287         STACK_OF(void) *sk;
288         int dummy; /* gcc is screwing up this data structure :-( */
289         };
290 DECLARE_STACK_OF(void)

292 /* This stuff is basically class callback functions
293  * The current classes are SSL_CTX, SSL, SSL_SESSION, and a few more */

295 typedef struct crypto_ex_data_func_st
296         {
297         long argl;      /* Arbitary long */
298         void *argp;     /* Arbitary void * */
299         CRYPTO_EX_new *new_func;
300         CRYPTO_EX_free *free_func;
301         CRYPTO_EX_dup *dup_func;
302         } CRYPTO_EX_DATA_FUNCS;

304 DECLARE_STACK_OF(CRYPTO_EX_DATA_FUNCS)

306 /* Per class, we have a STACK of CRYPTO_EX_DATA_FUNCS for each CRYPTO_EX_DATA
307  * entry.
308  */

310 #define CRYPTO_EX_INDEX_BIO             0
311 #define CRYPTO_EX_INDEX_SSL             1
312 #define CRYPTO_EX_INDEX_SSL_CTX         2
313 #define CRYPTO_EX_INDEX_SSL_SESSION     3
314 #define CRYPTO_EX_INDEX_X509_STORE      4
315 #define CRYPTO_EX_INDEX_X509_STORE_CTX  5
316 #define CRYPTO_EX_INDEX_RSA             6
317 #define CRYPTO_EX_INDEX_DSA             7
318 #define CRYPTO_EX_INDEX_DH              8
319 #define CRYPTO_EX_INDEX_ENGINE          9
320 #define CRYPTO_EX_INDEX_X509            10
321 #define CRYPTO_EX_INDEX_UI              11
322 #define CRYPTO_EX_INDEX_ECDSA           12
323 #define CRYPTO_EX_INDEX_ECDH            13
324 #define CRYPTO_EX_INDEX_COMP            14
325 #define CRYPTO_EX_INDEX_STORE           15
```



```
327 /* Dynamically assigned indexes start from this value (don't use directly, use
328  * via CRYPTO_ex_data_new_class). */
329 #define CRYPTO_EX_INDEX_USER            100


332 /* This is the default callbacks, but we can have others as well:
333  * this is needed in Win32 where the application malloc and the
334  * library malloc may not be the same.
335  */
336 #define CRYPTO_malloc_init()    CRYPTO_set_mem_functions(\
337         malloc, realloc, free)

339 #if defined CRYPTO_MDEBUG_ALL || defined CRYPTO_MDEBUG_TIME || defined CRYPTO_MD
340 # ifndef CRYPTO_MDEBUG /* avoid duplicate #define */
341 #  define CRYPTO_MDEBUG
342 # endif
343 #endif

345 /* Set standard debugging functions (not done by default
346  * unless CRYPTO_MDEBUG is defined) */
347 #define CRYPTO_malloc_debug_init()      do {\
348         CRYPTO_set_mem_debug_functions(\
349                 CRYPTO_dbg_malloc,\
350                 CRYPTO_dbg_realloc,\
351                 CRYPTO_dbg_free,\
352                 CRYPTO_dbg_set_options,\
353                 CRYPTO_dbg_get_options);\
354         } while(0)

356 int CRYPTO_mem_ctrl(int mode);
357 int CRYPTO_is_mem_check_on(void);

359 /* for applications */
360 #define MemCheck_start() CRYPTO_mem_ctrl(CRYPTO_MEM_CHECK_ON)
361 #define MemCheck_stop() CRYPTO_mem_ctrl(CRYPTO_MEM_CHECK_OFF)

363 /* for library-internal use */
364 #define MemCheck_on()   CRYPTO_mem_ctrl(CRYPTO_MEM_CHECK_ENABLE)
365 #define MemCheck_off()  CRYPTO_mem_ctrl(CRYPTO_MEM_CHECK_DISABLE)
366 #define is_MemCheck_on() CRYPTO_is_mem_check_on()

368 #define OPENSSL_malloc(num)     CRYPTO_malloc((int)num,__FILE__,__LINE__)
369 #define OPENSSL_strdup(str)     CRYPTO_strdup((str),__FILE__,__LINE__)
370 #define OPENSSL_realloc(addr,num) \
371         CRYPTO_realloc((char *)addr,(int)num,__FILE__,__LINE__)
372 #define OPENSSL_realloc_clean(addr,old_num,num) \
373         CRYPTO_realloc_clean(addr,old_num,num,__FILE__,__LINE__)
374 #define OPENSSL_remalloc(addr,num) \
375         CRYPTO_remalloc((char **)addr,(int)num,__FILE__,__LINE__)
376 #define OPENSSL_freeFunc        CRYPTO_free
377 #define OPENSSL_free(addr)      CRYPTO_free(addr)

379 #define OPENSSL_malloc_locked(num) \
380         CRYPTO_malloc_locked((int)num,__FILE__,__LINE__)
381 #define OPENSSL_free_locked(addr) CRYPTO_free_locked(addr)


384 const char *SSLeay_version(int type);
385 unsigned long SSLeay(void);

387 int OPENSSL_issetugid(void);

389 /* An opaque type representing an implementation of "ex_data" support */
390 typedef struct st_CRYPTO_EX_DATA_IMPL   CRYPTO_EX_DATA_IMPL;
391 /* Return an opaque pointer to the current "ex_data" implementation */
```

```
392 const CRYPTO_EX_DATA_IMPL *CRYPTO_get_ex_data_implementation(void);
393 /* Sets the "ex_data" implementation to be used (if it's not too late) */
394 int CRYPTO_set_ex_data_implementation(const CRYPTO_EX_DATA_IMPL *i);
395 /* Get a new "ex_data" class, and return the corresponding "class_index" */
396 int CRYPTO_ex_data_new_class(void);
397 /* Within a given class, get/register a new index */
398 int CRYPTO_get_ex_new_index(int class_index, long argl, void *argp,
399                 CRYPTO_EX_new *new_func, CRYPTO_EX_dup *dup_func,
400                 CRYPTO_EX_free *free_func);
401 /* Initialise/duplicate/free CRYPTO_EX_DATA variables corresponding to a given
402  * class (invokes whatever per-class callbacks are applicable) */
403 int CRYPTO_new_ex_data(int class_index, void *obj, CRYPTO_EX_DATA *ad);
404 int CRYPTO_dup_ex_data(int class_index, CRYPTO_EX_DATA *to,
405                 CRYPTO_EX_DATA *from);
406 void CRYPTO_free_ex_data(int class_index, void *obj, CRYPTO_EX_DATA *ad);
407 /* Get/set data in a CRYPTO_EX_DATA variable corresponding to a particular index
408  * (relative to the class type involved) */
409 int CRYPTO_set_ex_data(CRYPTO_EX_DATA *ad, int idx, void *val);
410 void *CRYPTO_get_ex_data(const CRYPTO_EX_DATA *ad,int idx);
411 /* This function cleans up all "ex_data" state. It mustn't be called under
412  * potential race-conditions. */
413 void CRYPTO_cleanup_all_ex_data(void);

415 int CRYPTO_get_new_lockid(char *name);

417 int CRYPTO_num_locks(void); /* return CRYPTO_NUM_LOCKS (shared libs!) */
418 void CRYPTO_lock(int mode, int type,const char *file,int line);
419 void CRYPTO_set_locking_callback(void (*func)(int mode,int type,
420                                               const char *file,int line));
421 void (*CRYPTO_get_locking_callback(void))(int mode,int type,const char *file,
422                 int line);
423 void CRYPTO_set_add_lock_callback(int (*func)(int *num,int mount,int type,
424                                               const char *file, int line));
425 int (*CRYPTO_get_add_lock_callback(void))(int *num,int mount,int type,
426                                           const char *file,int line);

428 /* Don't use this structure directly. */
429 typedef struct crypto_threadid_st
430         {
431         void *ptr;
432         unsigned long val;
433         } CRYPTO_THREADID;
434 /* Only use CRYPTO_THREADID_set_[numeric|pointer]() within callbacks */
435 void CRYPTO_THREADID_set_numeric(CRYPTO_THREADID *id, unsigned long val);
436 void CRYPTO_THREADID_set_pointer(CRYPTO_THREADID *id, void *ptr);
437 int CRYPTO_THREADID_set_callback(void (*threadid_func)(CRYPTO_THREADID *));
438 void (*CRYPTO_THREADID_get_callback(void))(CRYPTO_THREADID *);
439 void CRYPTO_THREADID_current(CRYPTO_THREADID *id);
440 int CRYPTO_THREADID_cmp(const CRYPTO_THREADID *a, const CRYPTO_THREADID *b);
441 void CRYPTO_THREADID_cpy(CRYPTO_THREADID *dest, const CRYPTO_THREADID *src);
442 unsigned long CRYPTO_THREADID_hash(const CRYPTO_THREADID *id);
443 #ifndef OPENSSL_NO_DEPRECATED
444 void CRYPTO_set_id_callback(unsigned long (*func)(void));
445 unsigned long (*CRYPTO_get_id_callback(void))(void);
446 unsigned long CRYPTO_thread_id(void);
447 #endif

449 const char *CRYPTO_get_lock_name(int type);
450 int CRYPTO_add_lock(int *pointer,int amount,int type, const char *file,
451                     int line);

453 int CRYPTO_get_new_dynlockid(void);
454 void CRYPTO_destroy_dynlockid(int i);
455 struct CRYPTO_dynlock_value *CRYPTO_get_dynlock_value(int i);
456 void CRYPTO_set_dynlock_create_callback(struct CRYPTO_dynlock_value *(*dyn_creat
457 void CRYPTO_set_dynlock_lock_callback(void (*dyn_lock_function)(int mode, struct
```



```
458 void CRYPTO_set_dynlock_destroy_callback(void (*dyn_destroy_function)(struct CRY
459 struct CRYPTO_dynlock_value *(*CRYPTO_get_dynlock_create_callback(void))(const c
460 void (*CRYPTO_get_dynlock_lock_callback(void))(int mode, struct CRYPTO_dynlock_v
461 void (*CRYPTO_get_dynlock_destroy_callback(void))(struct CRYPTO_dynlock_value *l

463 /* CRYPTO_set_mem_functions includes CRYPTO_set_locked_mem_functions --
464  * call the latter last if you need different functions */
465 int CRYPTO_set_mem_functions(void *(*m)(size_t),void *(*r)(void *,size_t), void
466 int CRYPTO_set_locked_mem_functions(void *(*m)(size_t), void (*free_func)(void *
467 int CRYPTO_set_mem_ex_functions(void *(*m)(size_t,const char *,int),
468                                 void *(*r)(void *,size_t,const char *,int),
469                                 void (*f)(void *));
470 int CRYPTO_set_locked_mem_ex_functions(void *(*m)(size_t,const char *,int),
471                                        void (*free_func)(void *));
472 int CRYPTO_set_mem_debug_functions(void (*m)(void *,int,const char *,int,int),
473                                    void (*r)(void *,void *,int,const char *,int,
474                                    void (*f)(void *,int),
475                                    void (*so)(long),
476                                    long (*go)(void));
477 void CRYPTO_get_mem_functions(void *(**m)(size_t),void *(**r)(void *, size_t), v
478 void CRYPTO_get_locked_mem_functions(void *(**m)(size_t), void (**f)(void *));
479 void CRYPTO_get_mem_ex_functions(void *(**m)(size_t,const char *,int),
480                                  void *(**r)(void *, size_t,const char *,int),
481                                  void (**f)(void *));
482 void CRYPTO_get_locked_mem_ex_functions(void *(**m)(size_t,const char *,int),
483                                         void (**f)(void *));
484 void CRYPTO_get_mem_debug_functions(void (**m)(void *,int,const char *,int,int),
485                                     void (**r)(void *,void *,int,const char *,in
486                                     void (**f)(void *,int),
487                                     void (**so)(long),
488                                     long (**go)(void));

490 void *CRYPTO_malloc_locked(int num, const char *file, int line);
491 void CRYPTO_free_locked(void *ptr);
492 void *CRYPTO_malloc(int num, const char *file, int line);
493 char *CRYPTO_strdup(const char *str, const char *file, int line);
494 void CRYPTO_free(void *ptr);
495 void *CRYPTO_realloc(void *addr,int num, const char *file, int line);
496 void *CRYPTO_realloc_clean(void *addr,int old_num,int num,const char *file,
497                            int line);
498 void *CRYPTO_remalloc(void *addr,int num, const char *file, int line);

500 void OPENSSL_cleanse(void *ptr, size_t len);

502 void CRYPTO_set_mem_debug_options(long bits);
503 long CRYPTO_get_mem_debug_options(void);

505 #define CRYPTO_push_info(info) \
506         CRYPTO_push_info_(info, __FILE__, __LINE__);
507 int CRYPTO_push_info_(const char *info, const char *file, int line);
508 int CRYPTO_pop_info(void);
509 int CRYPTO_remove_all_info(void);


512 /* Default debugging functions (enabled by CRYPTO_malloc_debug_init() macro;
513  * used as default in CRYPTO_MDEBUG compilations): */
514 /* The last argument has the following significance:
515  *
516  * 0:   called before the actual memory allocation has taken place
517  * 1:   called after the actual memory allocation has taken place
518  */
519 void CRYPTO_dbg_malloc(void *addr,int num,const char *file,int line,int before_p
520 void CRYPTO_dbg_realloc(void *addr1,void *addr2,int num,const char *file,int lin
521 void CRYPTO_dbg_free(void *addr,int before_p);
522 /* Tell the debugging code about options.  By default, the following values
523  * apply:
```

```
 *
 * 0:                           Clear all options.
 * V_CRYPTO_MDEBUG_TIME (1):    Set the "Show Time" option.
 * V_CRYPTO_MDEBUG_THREAD (2):  Set the "Show Thread Number" option.
 * V_CRYPTO_MDEBUG_ALL (3):     1 + 2
 */
void CRYPTO_dbg_set_options(long bits);
long CRYPTO_dbg_get_options(void);


#ifndef OPENSSL_NO_FP_API
void CRYPTO_mem_leaks_fp(FILE *);
#endif
void CRYPTO_mem_leaks(struct bio_st *bio);
/* unsigned long order, char *file, int line, int num_bytes, char *addr */
typedef void *CRYPTO_MEM_LEAK_CB(unsigned long, const char *, int, int, void *);
void CRYPTO_mem_leaks_cb(CRYPTO_MEM_LEAK_CB *cb);

/* die if we have to */
void OpenSSLDie(const char *file,int line,const char *assertion);
#define OPENSSL_assert(e)       (void)((e) ? 0 : (OpenSSLDie(__FILE__, __LINE__,

unsigned long *OPENSSL_ia32cap_loc(void);
#define OPENSSL_ia32cap (*(OPENSSL_ia32cap_loc()))
int OPENSSL_isservice(void);

int FIPS_mode(void);
int FIPS_mode_set(int r);

void OPENSSL_init(void);

#define fips_md_init(alg) fips_md_init_ctx(alg, alg)

#ifdef OPENSSL_FIPS
#define fips_md_init_ctx(alg, cx) \
        int alg##_Init(cx##_CTX *c) \
        { \
        if (FIPS_mode()) OpenSSLDie(__FILE__, __LINE__, \
                "Low level API call to digest " #alg " forbidden in FIPS mode!")
        return private_##alg##_Init(c); \
        } \
        int private_##alg##_Init(cx##_CTX *c)

#define fips_cipher_abort(alg) \
        if (FIPS_mode()) OpenSSLDie(__FILE__, __LINE__, \
                "Low level API call to cipher " #alg " forbidden in FIPS mode!")

#else
#define fips_md_init_ctx(alg, cx) \
        int alg##_Init(cx##_CTX *c)
#define fips_cipher_abort(alg) while(0)
#endif

/* CRYPTO_memcmp returns zero iff the |len| bytes at |a| and |b| are equal. It
 * takes an amount of time dependent on |len|, but independent of the contents
 * of |a| and |b|. Unlike memcmp, it cannot be used to put elements into a
 * defined order as the return value when a != b is undefined, other than to be
 * non-zero. */
int CRYPTO_memcmp(const void *a, const void *b, size_t len);

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_CRYPTO_strings(void);
```



```
/* Error codes for the CRYPTO functions. */

/* Function codes. */
#define CRYPTO_F_CRYPTO_GET_EX_NEW_INDEX                100
#define CRYPTO_F_CRYPTO_GET_NEW_DYNLOCKID               103
#define CRYPTO_F_CRYPTO_GET_NEW_LOCKID                  101
#define CRYPTO_F_CRYPTO_SET_EX_DATA                     102
#define CRYPTO_F_DEF_ADD_INDEX                          104
#define CRYPTO_F_DEF_GET_CLASS                          105
#define CRYPTO_F_FIPS_MODE_SET                          109
#define CRYPTO_F_INT_DUP_EX_DATA                        106
#define CRYPTO_F_INT_FREE_EX_DATA                       107
#define CRYPTO_F_INT_NEW_EX_DATA                        108

/* Reason codes. */
#define CRYPTO_R_FIPS_MODE_NOT_SUPPORTED                101
#define CRYPTO_R_NO_DYNLOCK_CREATE_CALLBACK             100

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   10842 Wed Aug 13 19:51:42 2014
new/usr/src/lib/openssl/include/openssl/des.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/des.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_NEW_DES_H
#define HEADER_NEW_DES_H
```



```
#include <openssl/e_os2.>      /* OPENSSL_EXTERN, OPENSSL_NO_DES,
                                  DES_LONG (via openssl/opensslconf.h */

#ifdef OPENSSL_NO_DES
#error DES is disabled.
#endif

#ifdef OPENSSL_BUILD_SHLIBCRYPTO
# undef OPENSSL_EXTERN
# define OPENSSL_EXTERN OPENSSL_EXPORT
#endif

#ifdef __cplusplus
extern "C" {
#endif

typedef unsigned char DES_cblock[8];
typedef /* const */ unsigned char const_DES_cblock[8];
/* With "const", gcc 2.8.1 on Solaris thinks that DES_cblock *
 * and const_DES_cblock * are incompatible pointer types. */

typedef struct DES_ks
    {
    union
        {
        DES_cblock cblock;
        /* make sure things are correct size on machines with
         * 8 byte longs */
        DES_LONG deslong[2];
        } ks[16];
    } DES_key_schedule;

#ifndef OPENSSL_DISABLE_OLD_DES_SUPPORT
# ifndef OPENSSL_ENABLE_OLD_DES_SUPPORT
#  define OPENSSL_ENABLE_OLD_DES_SUPPORT
# endif
#endif

#ifdef OPENSSL_ENABLE_OLD_DES_SUPPORT
# include <openssl/des_old.h>
#endif

#define DES_KEY_SZ      (sizeof(DES_cblock))
#define DES_SCHEDULE_SZ (sizeof(DES_key_schedule))

#define DES_ENCRYPT     1
#define DES_DECRYPT     0

#define DES_CBC_MODE    0
#define DES_PCBC_MODE   1

#define DES_ecb2_encrypt(i,o,k1,k2,e) \
        DES_ecb3_encrypt((i),(o),(k1),(k2),(k1),(e))

#define DES_ede2_cbc_encrypt(i,o,l,k1,k2,iv,e) \
        DES_ede3_cbc_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(e))

#define DES_ede2_cfb64_encrypt(i,o,l,k1,k2,iv,n,e) \
        DES_ede3_cfb64_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(n),(e))

#define DES_ede2_ofb64_encrypt(i,o,l,k1,k2,iv,n) \
        DES_ede3_ofb64_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(n))

OPENSSL_DECLARE_GLOBAL(int,DES_check_key);      /* defaults to false */
#define DES_check_key OPENSSL_GLOBAL_REF(DES_check_key)
OPENSSL_DECLARE_GLOBAL(int,DES_rw_mode);        /* defaults to DES_PCBC_MODE */
```

```
#define DES_rw_mode OPENSSL_GLOBAL_REF(DES_rw_mode)

const char *DES_options(void);
void DES_ecb3_encrypt(const_DES_cblock *input, DES_cblock *output,
                      DES_key_schedule *ks1,DES_key_schedule *ks2,
                      DES_key_schedule *ks3, int enc);
DES_LONG DES_cbc_cksum(const unsigned char *input,DES_cblock *output,
                       long length,DES_key_schedule *schedule,
                       const_DES_cblock *ivec);
/* DES_cbc_encrypt does not update the IV!  Use DES_ncbc_encrypt instead. */
void DES_cbc_encrypt(const unsigned char *input,unsigned char *output,
                     long length,DES_key_schedule *schedule,DES_cblock *ivec,
                     int enc);
void DES_ncbc_encrypt(const unsigned char *input,unsigned char *output,
                      long length,DES_key_schedule *schedule,DES_cblock *ivec,
                      int enc);
void DES_xcbc_encrypt(const unsigned char *input,unsigned char *output,
                      long length,DES_key_schedule *schedule,DES_cblock *ivec,
                      const_DES_cblock *inw,const_DES_cblock *outw,int enc);
void DES_cfb_encrypt(const unsigned char *in,unsigned char *out,int numbits,
                     long length,DES_key_schedule *schedule,DES_cblock *ivec,
                     int enc);
void DES_ecb_encrypt(const_DES_cblock *input,DES_cblock *output,
                     DES_key_schedule *ks,int enc);

/*      This is the DES encryption function that gets called by just about
        every other DES routine in the library.  You should not use this
        function except to implement 'modes' of DES.  I say this because the
        functions that call this routine do the conversion from 'char *' to
        long, and this needs to be done to make sure 'non-aligned' memory
        access do not occur.  The characters are loaded 'little endian'.
        Data is a pointer to 2 unsigned long's and ks is the
        DES_key_schedule to use.  enc, is non zero specifies encryption,
        zero if decryption. */
void DES_encrypt1(DES_LONG *data,DES_key_schedule *ks, int enc);

/*      This functions is the same as DES_encrypt1() except that the DES
        initial permutation (IP) and final permutation (FP) have been left
        out.  As for DES_encrypt1(), you should not use this function.
        It is used by the routines in the library that implement triple DES.
        IP() DES_encrypt2() DES_encrypt2() DES_encrypt2() FP() is the same
        as DES_encrypt1() DES_encrypt1() DES_encrypt1() except faster :-). */
void DES_encrypt2(DES_LONG *data,DES_key_schedule *ks, int enc);

void DES_encrypt3(DES_LONG *data, DES_key_schedule *ks1,
                  DES_key_schedule *ks2, DES_key_schedule *ks3);
void DES_decrypt3(DES_LONG *data, DES_key_schedule *ks1,
                  DES_key_schedule *ks2, DES_key_schedule *ks3);
void DES_ede3_cbc_encrypt(const unsigned char *input,unsigned char *output,
                          long length,
                          DES_key_schedule *ks1,DES_key_schedule *ks2,
                          DES_key_schedule *ks3,DES_cblock *ivec,int enc);
void DES_ede3_cbcm_encrypt(const unsigned char *in,unsigned char *out,
                           long length,
                           DES_key_schedule *ks1,DES_key_schedule *ks2,
                           DES_key_schedule *ks3,
                           DES_cblock *ivec1,DES_cblock *ivec2,
                           int enc);
void DES_ede3_cfb64_encrypt(const unsigned char *in,unsigned char *out,
                            long length,DES_key_schedule *ks1,
                            DES_key_schedule *ks2,DES_key_schedule *ks3,
                            DES_cblock *ivec,int *num,int enc);
void DES_ede3_cfb_encrypt(const unsigned char *in,unsigned char *out,
                          int numbits,long length,DES_key_schedule *ks1,
                          DES_key_schedule *ks2,DES_key_schedule *ks3,
                          DES_cblock *ivec,int enc);
```



```
void DES_ede3_ofb64_encrypt(const unsigned char *in,unsigned char *out,
                            long length,DES_key_schedule *ks1,
                            DES_key_schedule *ks2,DES_key_schedule *ks3,
                            DES_cblock *ivec,int *num);
#if 0
void DES_xwhite_in2out(const_DES_cblock *DES_key,const_DES_cblock *in_white,
                       DES_cblock *out_white);
#endif

int DES_enc_read(int fd,void *buf,int len,DES_key_schedule *sched,
                 DES_cblock *iv);
int DES_enc_write(int fd,const void *buf,int len,DES_key_schedule *sched,
                  DES_cblock *iv);
char *DES_fcrypt(const char *buf,const char *salt, char *ret);
char *DES_crypt(const char *buf,const char *salt);
void DES_ofb_encrypt(const unsigned char *in,unsigned char *out,int numbits,
                     long length,DES_key_schedule *schedule,DES_cblock *ivec);
void DES_pcbc_encrypt(const unsigned char *input,unsigned char *output,
                      long length,DES_key_schedule *schedule,DES_cblock *ivec,
                      int enc);
DES_LONG DES_quad_cksum(const unsigned char *input,DES_cblock output[],
                        long length,int out_count,DES_cblock *seed);
int DES_random_key(DES_cblock *ret);
void DES_set_odd_parity(DES_cblock *key);
int DES_check_key_parity(const_DES_cblock *key);
int DES_is_weak_key(const_DES_cblock *key);
/* DES_set_key (= set_key = DES_key_sched = key_sched) calls
 * DES_set_key_checked if global variable DES_check_key is set,
 * DES_set_key_unchecked otherwise. */
int DES_set_key(const_DES_cblock *key,DES_key_schedule *schedule);
int DES_key_sched(const_DES_cblock *key,DES_key_schedule *schedule);
int DES_set_key_checked(const_DES_cblock *key,DES_key_schedule *schedule);
void DES_set_key_unchecked(const_DES_cblock *key,DES_key_schedule *schedule);
#ifdef OPENSSL_FIPS
void private_DES_set_key_unchecked(const_DES_cblock *key,DES_key_schedule *sched
#endif
void DES_string_to_key(const char *str,DES_cblock *key);
void DES_string_to_2keys(const char *str,DES_cblock *key1,DES_cblock *key2);
void DES_cfb64_encrypt(const unsigned char *in,unsigned char *out,long length,
                       DES_key_schedule *schedule,DES_cblock *ivec,int *num,
                       int enc);
void DES_ofb64_encrypt(const unsigned char *in,unsigned char *out,long length,
                       DES_key_schedule *schedule,DES_cblock *ivec,int *num);

int DES_read_password(DES_cblock *key, const char *prompt, int verify);
int DES_read_2passwords(DES_cblock *key1, DES_cblock *key2, const char *prompt,
        int verify);

#define DES_fixup_key_parity DES_set_odd_parity

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
   18231 Wed Aug 13 19:51:42 2014
new/usr/src/lib/openssl/include/openssl/des_old.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/des_old.h -*- mode:C; c-file-style: "eay" -*- */

/* WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING
 *
 * The function names in here are deprecated and are only present to
 * provide an interface compatible with openssl 0.9.6 and older as
 * well as libdes.  OpenSSL now provides functions where "des_" has
 * been replaced with "DES_" in the names, to make it possible to
 * make incompatible changes that are needed for C type security and
 * other stuff.
 *
 * This include files has two compatibility modes:
 *
 *   - If OPENSSL_DES_LIBDES_COMPATIBILITY is defined, you get an API
 *     that is compatible with libdes and SSLeay.
 *   - If OPENSSL_DES_LIBDES_COMPATIBILITY isn't defined, you get an
 *     API that is compatible with OpenSSL 0.9.5x to 0.9.6x.
 *
 * Note that these modes break earlier snapshots of OpenSSL, where
 * libdes compatibility was the only available mode or (later on) the
 * prefered compatibility mode.  However, after much consideration
 * (and more or less violent discussions with external parties), it
 * was concluded that OpenSSL should be compatible with earlier versions
 * of itself before anything else.  Also, in all honesty, libdes is
 * an old beast that shouldn't really be used any more.
 *
 * Please consider starting to use the DES_ functions rather than the
 * des_ ones.  The des_ functions will disappear completely before
 * OpenSSL 1.0!
 *
 * WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING
 */

/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
```



```
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_DES_H
#define HEADER_DES_H

#include <openssl/e_os2.h>      /* OPENSSL_EXTERN, OPENSSL_NO_DES, DES_LONG */

#ifdef OPENSSL_NO_DES
#error DES is disabled.
#endif

#ifndef HEADER_NEW_DES_H
#error You must include des.h, not des_old.h directly.
#endif

#ifdef _KERBEROS_DES_H
#error <openssl/des_old.h> replaces <kerberos/des.h>.
#endif

#include <openssl/symhacks.h>

#ifdef OPENSSL_BUILD_SHLIBCRYPTO
# undef OPENSSL_EXTERN
# define OPENSSL_EXTERN OPENSSL_EXPORT
#endif

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef _
#undef _
#endif

typedef unsigned char _ossl_old_des_cblock[8];
typedef struct _ossl_old_des_ks_struct
        {
        union   {
                _ossl_old_des_cblock _;
```

```
128                 /* make sure things are correct size on machines with
129                  * 8 byte longs */
130                 DES_LONG pad[2];
131                 } ks;
132         } _ossl_old_des_key_schedule[16];

133 #ifndef OPENSSL_DES_LIBDES_COMPATIBILITY
134 #define des_cblock DES_cblock
135 #define const_des_cblock const_DES_cblock
136 #define des_key_schedule DES_key_schedule
137 #define des_ecb3_encrypt(i,o,k1,k2,k3,e)\
138         DES_ecb3_encrypt((i),(o),&(k1),&(k2),&(k3),(e))
139 #define des_ede3_cbc_encrypt(i,o,l,k1,k2,k3,iv,e)\
140         DES_ede3_cbc_encrypt((i),(o),(l),&(k1),&(k2),&(k3),(iv),(e))
141 #define des_ede3_cbcm_encrypt(i,o,l,k1,k2,k3,iv1,iv2,e)\
142         DES_ede3_cbcm_encrypt((i),(o),(l),&(k1),&(k2),&(k3),(iv1),(iv2),(e))
143 #define des_ede3_cfb64_encrypt(i,o,l,k1,k2,k3,iv,n,e)\
144         DES_ede3_cfb64_encrypt((i),(o),(l),&(k1),&(k2),&(k3),(iv),(n),(e))
145 #define des_ede3_ofb64_encrypt(i,o,l,k1,k2,k3,iv,n)\
146         DES_ede3_ofb64_encrypt((i),(o),(l),&(k1),&(k2),&(k3),(iv),(n))
147 #define des_options()\
148         DES_options()
149 #define des_cbc_cksum(i,o,l,k,iv)\
150         DES_cbc_cksum((i),(o),(l),&(k),(iv))
151 #define des_cbc_encrypt(i,o,l,k,iv,e)\
152         DES_cbc_encrypt((i),(o),(l),&(k),(iv),(e))
153 #define des_ncbc_encrypt(i,o,l,k,iv,e)\
154         DES_ncbc_encrypt((i),(o),(l),&(k),(iv),(e))
155 #define des_xcbc_encrypt(i,o,l,k,iv,inw,outw,e)\
156         DES_xcbc_encrypt((i),(o),(l),&(k),(iv),(inw),(outw),(e))
157 #define des_cfb_encrypt(i,o,n,l,k,iv,e)\
158         DES_cfb_encrypt((i),(o),(n),(l),&(k),(iv),(e))
159 #define des_ecb_encrypt(i,o,k,e)\
160         DES_ecb_encrypt((i),(o),&(k),(e))
161 #define des_encrypt1(d,k,e)\
162         DES_encrypt1((d),&(k),(e))
163 #define des_encrypt2(d,k,e)\
164         DES_encrypt2((d),&(k),(e))
165 #define des_encrypt3(d,k1,k2,k3)\
166         DES_encrypt3((d),&(k1),&(k2),&(k3))
167 #define des_decrypt3(d,k1,k2,k3)\
168         DES_decrypt3((d),&(k1),&(k2),&(k3))
169 #define des_xwhite_in2out(k,i,o)\
170         DES_xwhite_in2out((k),(i),(o))
171 #define des_enc_read(f,b,l,k,iv)\
172         DES_enc_read((f),(b),(l),&(k),(iv))
173 #define des_enc_write(f,b,l,k,iv)\
174         DES_enc_write((f),(b),(l),&(k),(iv))
175 #define des_fcrypt(b,s,r)\
176         DES_fcrypt((b),(s),(r))
177 #if 0
178 #define des_crypt(b,s)\
179         DES_crypt((b),(s))
180 #if !defined(PERL5) && !defined(__FreeBSD__) && !defined(NeXT) && !defined(__Ope
181 #define crypt(b,s)\
182         DES_crypt((b),(s))
183 #endif
184 #endif
185 #define des_ofb_encrypt(i,o,n,l,k,iv)\
186         DES_ofb_encrypt((i),(o),(n),(l),&(k),(iv))
187 #define des_pcbc_encrypt(i,o,l,k,iv,e)\
188         DES_pcbc_encrypt((i),(o),(l),&(k),(iv),(e))
189 #define des_quad_cksum(i,o,l,c,s)\
190         DES_quad_cksum((i),(o),(l),(c),(s))
191 #define des_random_seed(k)\
192         _ossl_096_des_random_seed((k))
193
```



```
194 #define des_random_key(r)\
195         DES_random_key((r))
196 #define des_read_password(k,p,v) \
197         DES_read_password((k),(p),(v))
198 #define des_read_2passwords(k1,k2,p,v) \
199         DES_read_2passwords((k1),(k2),(p),(v))
200 #define des_set_odd_parity(k)\
201         DES_set_odd_parity((k))
202 #define des_check_key_parity(k)\
203         DES_check_key_parity((k))
204 #define des_is_weak_key(k)\
205         DES_is_weak_key((k))
206 #define des_set_key(k,ks)\
207         DES_set_key((k),&(ks))
208 #define des_key_sched(k,ks)\
209         DES_key_sched((k),&(ks))
210 #define des_set_key_checked(k,ks)\
211         DES_set_key_checked((k),&(ks))
212 #define des_set_key_unchecked(k,ks)\
213         DES_set_key_unchecked((k),&(ks))
214 #define des_string_to_key(s,k)\
215         DES_string_to_key((s),(k))
216 #define des_string_to_2keys(s,k1,k2)\
217         DES_string_to_2keys((s),(k1),(k2))
218 #define des_cfb64_encrypt(i,o,l,ks,iv,n,e)\
219         DES_cfb64_encrypt((i),(o),(l),&(ks),(iv),(n),(e))
220 #define des_ofb64_encrypt(i,o,l,ks,iv,n)\
221         DES_ofb64_encrypt((i),(o),(l),&(ks),(iv),(n))


224 #define des_ecb2_encrypt(i,o,k1,k2,e) \
225         des_ecb3_encrypt((i),(o),(k1),(k2),(k1),(e))

227 #define des_ede2_cbc_encrypt(i,o,l,k1,k2,iv,e) \
228         des_ede3_cbc_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(e))

230 #define des_ede2_cfb64_encrypt(i,o,l,k1,k2,iv,n,e) \
231         des_ede3_cfb64_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(n),(e))

233 #define des_ede2_ofb64_encrypt(i,o,l,k1,k2,iv,n) \
234         des_ede3_ofb64_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(n))

236 #define des_check_key DES_check_key
237 #define des_rw_mode DES_rw_mode
238 #else /* libdes compatibility */
239 /* Map all symbol names to _ossl_old_des_* form, so we avoid all
240    clashes with libdes */
241 #define des_cblock _ossl_old_des_cblock
242 #define des_key_schedule _ossl_old_des_key_schedule
243 #define des_ecb3_encrypt(i,o,k1,k2,k3,e)\
244         _ossl_old_des_ecb3_encrypt((i),(o),(k1),(k2),(k3),(e))
245 #define des_ede3_cbc_encrypt(i,o,l,k1,k2,k3,iv,e)\
246         _ossl_old_des_ede3_cbc_encrypt((i),(o),(l),(k1),(k2),(k3),(iv),(e))
247 #define des_ede3_cfb64_encrypt(i,o,l,k1,k2,k3,iv,n,e)\
248         _ossl_old_des_ede3_cfb64_encrypt((i),(o),(l),(k1),(k2),(k3),(iv),(n),(e)
249 #define des_ede3_ofb64_encrypt(i,o,l,k1,k2,k3,iv,n)\
250         _ossl_old_des_ede3_ofb64_encrypt((i),(o),(l),(k1),(k2),(k3),(iv),(n))
251 #define des_options()\
252         _ossl_old_des_options()
253 #define des_cbc_cksum(i,o,l,k,iv)\
254         _ossl_old_des_cbc_cksum((i),(o),(l),(k),(iv))
255 #define des_cbc_encrypt(i,o,l,k,iv,e)\
256         _ossl_old_des_cbc_encrypt((i),(o),(l),(k),(iv),(e))
257 #define des_ncbc_encrypt(i,o,l,k,iv,e)\
258         _ossl_old_des_ncbc_encrypt((i),(o),(l),(k),(iv),(e))
259 #define des_xcbc_encrypt(i,o,l,k,iv,inw,outw,e)\
```

```
260         _ossl_old_des_xcbc_encrypt((i),(o),(l),(k),(iv),(inw),(outw),(e))
261 #define des_cfb_encrypt(i,o,n,l,k,iv,e)\
262         _ossl_old_des_cfb_encrypt((i),(o),(n),(l),(k),(iv),(e))
263 #define des_ecb_encrypt(i,o,k,e)\
264         _ossl_old_des_ecb_encrypt((i),(o),(k),(e))
265 #define des_encrypt(d,k,e)\
266         _ossl_old_des_encrypt((d),(k),(e))
267 #define des_encrypt2(d,k,e)\
268         _ossl_old_des_encrypt2((d),(k),(e))
269 #define des_encrypt3(d,k1,k2,k3)\
270         _ossl_old_des_encrypt3((d),(k1),(k2),(k3))
271 #define des_decrypt3(d,k1,k2,k3)\
272         _ossl_old_des_decrypt3((d),(k1),(k2),(k3))
273 #define des_xwhite_in2out(k,i,o)\
274         _ossl_old_des_xwhite_in2out((k),(i),(o))
275 #define des_enc_read(f,b,l,k,iv)\
276         _ossl_old_des_enc_read((f),(b),(l),(k),(iv))
277 #define des_enc_write(f,b,l,k,iv)\
278         _ossl_old_des_enc_write((f),(b),(l),(k),(iv))
279 #define des_fcrypt(b,s,r)\
280         _ossl_old_des_fcrypt((b),(s),(r))
281 #define des_crypt(b,s)\
282         _ossl_old_des_crypt((b),(s))
283 #if 0
284 #define crypt(b,s)\
285         _ossl_old_crypt((b),(s))
286 #endif
287 #define des_ofb_encrypt(i,o,n,l,k,iv)\
288         _ossl_old_des_ofb_encrypt((i),(o),(n),(l),(k),(iv))
289 #define des_pcbc_encrypt(i,o,l,k,iv,e)\
290         _ossl_old_des_pcbc_encrypt((i),(o),(l),(k),(iv),(e))
291 #define des_quad_cksum(i,o,l,c,s)\
292         _ossl_old_des_quad_cksum((i),(o),(l),(c),(s))
293 #define des_random_seed(k)\
294         _ossl_old_des_random_seed((k))
295 #define des_random_key(r)\
296         _ossl_old_des_random_key((r))
297 #define des_read_password(k,p,v) \
298         _ossl_old_des_read_password((k),(p),(v))
299 #define des_read_2passwords(k1,k2,p,v) \
300         _ossl_old_des_read_2passwords((k1),(k2),(p),(v))
301 #define des_set_odd_parity(k)\
302         _ossl_old_des_set_odd_parity((k))
303 #define des_is_weak_key(k)\
304         _ossl_old_des_is_weak_key((k))
305 #define des_set_key(k,ks)\
306         _ossl_old_des_set_key((k),(ks))
307 #define des_key_sched(k,ks)\
308         _ossl_old_des_key_sched((k),(ks))
309 #define des_string_to_key(s,k)\
310         _ossl_old_des_string_to_key((s),(k))
311 #define des_string_to_2keys(s,k1,k2)\
312         _ossl_old_des_string_to_2keys((s),(k1),(k2))
313 #define des_cfb64_encrypt(i,o,l,ks,iv,n,e)\
314         _ossl_old_des_cfb64_encrypt((i),(o),(l),(ks),(iv),(n),(e))
315 #define des_ofb64_encrypt(i,o,l,ks,iv,n)\
316         _ossl_old_des_ofb64_encrypt((i),(o),(l),(ks),(iv),(n))


319 #define des_ecb2_encrypt(i,o,k1,k2,e) \
320         des_ecb3_encrypt((i),(o),(k1),(k2),(k1),(e))

322 #define des_ede2_cbc_encrypt(i,o,l,k1,k2,iv,e) \
323         des_ede3_cbc_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(e))

325 #define des_ede2_cfb64_encrypt(i,o,l,k1,k2,iv,n,e) \
```



```
326         des_ede3_cfb64_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(n),(e))

328 #define des_ede2_ofb64_encrypt(i,o,l,k1,k2,iv,n) \
329         des_ede3_ofb64_encrypt((i),(o),(l),(k1),(k2),(k1),(iv),(n))

331 #define des_check_key DES_check_key
332 #define des_rw_mode DES_rw_mode
333 #endif

335 const char *_ossl_old_des_options(void);
336 void _ossl_old_des_ecb3_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
337         _ossl_old_des_key_schedule ks1,_ossl_old_des_key_schedule ks2,
338         _ossl_old_des_key_schedule ks3, int enc);
339 DES_LONG _ossl_old_des_cbc_cksum(_ossl_old_des_cblock *input,_ossl_old_des_cbloc
340         long length,_ossl_old_des_key_schedule schedule,_ossl_old_des_cblock *iv
341 void _ossl_old_des_cbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
342         _ossl_old_des_key_schedule schedule,_ossl_old_des_cblock *ivec,int enc);
343 void _ossl_old_des_ncbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
344         _ossl_old_des_key_schedule schedule,_ossl_old_des_cblock *ivec,int enc);
345 void _ossl_old_des_xcbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
346         _ossl_old_des_key_schedule schedule,_ossl_old_des_cblock *ivec,
347         _ossl_old_des_cblock *inw,_ossl_old_des_cblock *outw,int enc);
348 void _ossl_old_des_cfb_encrypt(unsigned char *in,unsigned char *out,int numbits,
349         long length,_ossl_old_des_key_schedule schedule,_ossl_old_des_cblock *iv
350 void _ossl_old_des_ecb_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
351         _ossl_old_des_key_schedule ks,int enc);
352 void _ossl_old_des_encrypt(DES_LONG *data,_ossl_old_des_key_schedule ks, int enc
353 void _ossl_old_des_encrypt2(DES_LONG *data,_ossl_old_des_key_schedule ks, int en
354 void _ossl_old_des_encrypt3(DES_LONG *data, _ossl_old_des_key_schedule ks1,
355         _ossl_old_des_key_schedule ks2, _ossl_old_des_key_schedule ks3);
356 void _ossl_old_des_decrypt3(DES_LONG *data, _ossl_old_des_key_schedule ks1,
357         _ossl_old_des_key_schedule ks2, _ossl_old_des_key_schedule ks3);
358 void _ossl_old_des_ede3_cbc_encrypt(_ossl_old_des_cblock *input, _ossl_old_des_c
359         long length, _ossl_old_des_key_schedule ks1, _ossl_old_des_key_schedule
360         _ossl_old_des_key_schedule ks3, _ossl_old_des_cblock *ivec, int enc);
361 void _ossl_old_des_ede3_cfb64_encrypt(unsigned char *in, unsigned char *out,
362         long length, _ossl_old_des_key_schedule ks1, _ossl_old_des_key_schedule
363         _ossl_old_des_key_schedule ks3, _ossl_old_des_cblock *ivec, int *num, in
364 void _ossl_old_des_ede3_ofb64_encrypt(unsigned char *in, unsigned char *out,
365         long length, _ossl_old_des_key_schedule ks1, _ossl_old_des_key_schedule
366         _ossl_old_des_key_schedule ks3, _ossl_old_des_cblock *ivec, int *num);
367 #if 0
368 void _ossl_old_des_xwhite_in2out(_ossl_old_des_cblock (*des_key), _ossl_old_des_
369         _ossl_old_des_cblock (*out_white));
370 #endif

372 int _ossl_old_des_enc_read(int fd,char *buf,int len,_ossl_old_des_key_schedule s
373         _ossl_old_des_cblock *iv);
374 int _ossl_old_des_enc_write(int fd,char *buf,int len,_ossl_old_des_key_schedule
375         _ossl_old_des_cblock *iv);
376 char *_ossl_old_des_fcrypt(const char *buf,const char *salt, char *ret);
377 char *_ossl_old_des_crypt(const char *buf,const char *salt);
378 #if !defined(PERL5) && !defined(NeXT)
379 char *_ossl_old_crypt(const char *buf,const char *salt);
380 #endif
381 void _ossl_old_des_ofb_encrypt(unsigned char *in,unsigned char *out,
382         int numbits,long length,_ossl_old_des_key_schedule schedule,_ossl_old_de
383 void _ossl_old_des_pcbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
384         _ossl_old_des_key_schedule schedule,_ossl_old_des_cblock *ivec,int enc);
385 DES_LONG _ossl_old_des_quad_cksum(_ossl_old_des_cblock *input,_ossl_old_des_cblo
386         long length,int out_count,_ossl_old_des_cblock *seed);
387 void _ossl_old_des_random_seed(_ossl_old_des_cblock key);
388 void _ossl_old_des_random_key(_ossl_old_des_cblock ret);
389 int _ossl_old_des_read_password(_ossl_old_des_cblock *key,const char *prompt,int
390 int _ossl_old_des_read_2passwords(_ossl_old_des_cblock *key1,_ossl_old_des_cbloc
391         const char *prompt,int verify);
```

```
392 void _ossl_old_des_set_odd_parity(_ossl_old_des_cblock *key);
393 int _ossl_old_des_is_weak_key(_ossl_old_des_cblock *key);
394 int _ossl_old_des_set_key(_ossl_old_des_cblock *key,_ossl_old_des_key_schedule s
395 int _ossl_old_des_key_sched(_ossl_old_des_cblock *key,_ossl_old_des_key_schedule
396 void _ossl_old_des_string_to_key(char *str,_ossl_old_des_cblock *key);
397 void _ossl_old_des_string_to_2keys(char *str,_ossl_old_des_cblock *key1,_ossl_ol
398 void _ossl_old_des_cfb64_encrypt(unsigned char *in, unsigned char *out, long len
399         _ossl_old_des_key_schedule schedule, _ossl_old_des_cblock *ivec, int *nu
400 void _ossl_old_des_ofb64_encrypt(unsigned char *in, unsigned char *out, long len
401         _ossl_old_des_key_schedule schedule, _ossl_old_des_cblock *ivec, int *nu

403 void _ossl_096_des_random_seed(des_cblock *key);

405 /* The following definitions provide compatibility with the MIT Kerberos
406  * library. The _ossl_old_des_key_schedule structure is not binary compatible. *

408 #define _KERBEROS_DES_H

410 #define KRBDES_ENCRYPT DES_ENCRYPT
411 #define KRBDES_DECRYPT DES_DECRYPT

413 #ifdef KERBEROS
414 #  define ENCRYPT DES_ENCRYPT
415 #  define DECRYPT DES_DECRYPT
416 #endif

418 #ifndef NCOMPAT
419 #  define C_Block des_cblock
420 #  define Key_schedule des_key_schedule
421 #  define KEY_SZ DES_KEY_SZ
422 #  define string_to_key des_string_to_key
423 #  define read_pw_string des_read_pw_string
424 #  define random_key des_random_key
425 #  define pcbc_encrypt des_pcbc_encrypt
426 #  define set_key des_set_key
427 #  define key_sched des_key_sched
428 #  define ecb_encrypt des_ecb_encrypt
429 #  define cbc_encrypt des_cbc_encrypt
430 #  define ncbc_encrypt des_ncbc_encrypt
431 #  define xcbc_encrypt des_xcbc_encrypt
432 #  define cbc_cksum des_cbc_cksum
433 #  define quad_cksum des_quad_cksum
434 #  define check_parity des_check_key_parity
435 #endif

437 #define des_fixup_key_parity DES_fixup_key_parity

439 #ifdef  __cplusplus
440 }
441 #endif

443 /* for DES_read_pw_string et al */
444 #include <openssl/ui_compat.h>

446 #endif
447 #endif /* ! codereview */
```

```
**********************************************************
    9970 Wed Aug 13 19:51:42 2014
new/usr/src/lib/openssl/include/openssl/dh.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dh/dh.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_DH_H
#define HEADER_DH_H
```



```
#include <openssl/e_os2.h>

#ifdef OPENSSL_NO_DH
#error DH is disabled.
#endif

#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/ossl_typ.h>
#ifndef OPENSSL_NO_DEPRECATED
#include <openssl/bn.h>
#endif

#ifndef OPENSSL_DH_MAX_MODULUS_BITS
# define OPENSSL_DH_MAX_MODULUS_BITS    10000
#endif

#define DH_FLAG_CACHE_MONT_P     0x01
#define DH_FLAG_NO_EXP_CONSTTIME 0x02 /* new with 0.9.7h; the built-in DH
                                       * implementation now uses constant time
                                       * modular exponentiation for secret expon
                                       * by default. This flag causes the
                                       * faster variable sliding window method t
                                       * be used for all exponents.
                                       */

/* If this flag is set the DH method is FIPS compliant and can be used
 * in FIPS mode. This is set in the validated module method. If an
 * application sets this flag in its own methods it is its reposibility
 * to ensure the result is compliant.
 */

#define DH_FLAG_FIPS_METHOD                     0x0400

/* If this flag is set the operations normally disabled in FIPS mode are
 * permitted it is then the applications responsibility to ensure that the
 * usage is compliant.
 */

#define DH_FLAG_NON_FIPS_ALLOW                  0x0400

#ifdef __cplusplus
extern "C" {
#endif

/* Already defined in ossl_typ.h */
/* typedef struct dh_st DH; */
/* typedef struct dh_method DH_METHOD; */

struct dh_method
        {
        const char *name;
        /* Methods here */
        int (*generate_key)(DH *dh);
        int (*compute_key)(unsigned char *key,const BIGNUM *pub_key,DH *dh);
        int (*bn_mod_exp)(const DH *dh, BIGNUM *r, const BIGNUM *a,
                                const BIGNUM *p, const BIGNUM *m, BN_CTX *ctx,
                                BN_MONT_CTX *m_ctx); /* Can be null */

        int (*init)(DH *dh);
        int (*finish)(DH *dh);
        int flags;
        char *app_data;
        /* If this is non-NULL, it will be used to generate parameters */
        int (*generate_params)(DH *dh, int prime_len, int generator, BN_GENCB *c
```

```
128         };

130 struct dh_st
131         {
132         /* This first argument is used to pick up errors when
133          * a DH is passed instead of a EVP_PKEY */
134         int pad;
135         int version;
136         BIGNUM *p;
137         BIGNUM *g;
138         long length; /* optional */
139         BIGNUM *pub_key;        /* g^x */
140         BIGNUM *priv_key;       /* x */

142         int flags;
143         BN_MONT_CTX *method_mont_p;
144         /* Place holders if we want to do X9.42 DH */
145         BIGNUM *q;
146         BIGNUM *j;
147         unsigned char *seed;
148         int seedlen;
149         BIGNUM *counter;

151         int references;
152         CRYPTO_EX_DATA ex_data;
153         const DH_METHOD *meth;
154         ENGINE *engine;
155         };

157 #define DH_GENERATOR_2          2
158 /* #define DH_GENERATOR_3       3 */
159 #define DH_GENERATOR_5          5

161 /* DH_check error codes */
162 #define DH_CHECK_P_NOT_PRIME            0x01
163 #define DH_CHECK_P_NOT_SAFE_PRIME       0x02
164 #define DH_UNABLE_TO_CHECK_GENERATOR    0x04
165 #define DH_NOT_SUITABLE_GENERATOR       0x08

167 /* DH_check_pub_key error codes */
168 #define DH_CHECK_PUBKEY_TOO_SMALL       0x01
169 #define DH_CHECK_PUBKEY_TOO_LARGE       0x02

171 /* primes p where (p-1)/2 is prime too are called "safe"; we define
172    this for backward compatibility: */
173 #define DH_CHECK_P_NOT_STRONG_PRIME     DH_CHECK_P_NOT_SAFE_PRIME

175 #define d2i_DHparams_fp(fp,x) (DH *)ASN1_d2i_fp((char *(*)())DH_new, \
176                 (char *(*)())d2i_DHparams,(fp),(unsigned char **)(x))
177 #define i2d_DHparams_fp(fp,x) ASN1_i2d_fp(i2d_DHparams,(fp), \
178                 (unsigned char *)(x))
179 #define d2i_DHparams_bio(bp,x) ASN1_d2i_bio_of(DH,DH_new,d2i_DHparams,bp,x)
180 #define i2d_DHparams_bio(bp,x) ASN1_i2d_bio_of_const(DH,i2d_DHparams,bp,x)

182 DH *DHparams_dup(DH *);

184 const DH_METHOD *DH_OpenSSL(void);

186 void DH_set_default_method(const DH_METHOD *meth);
187 const DH_METHOD *DH_get_default_method(void);
188 int DH_set_method(DH *dh, const DH_METHOD *meth);
189 DH *DH_new_method(ENGINE *engine);

191 DH *    DH_new(void);
192 void    DH_free(DH *dh);
193 int     DH_up_ref(DH *dh);
```



```
194 int     DH_size(const DH *dh);
195 int DH_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
196             CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
197 int DH_set_ex_data(DH *d, int idx, void *arg);
198 void *DH_get_ex_data(DH *d, int idx);

200 /* Deprecated version */
201 #ifndef OPENSSL_NO_DEPRECATED
202 DH *    DH_generate_parameters(int prime_len,int generator,
203                 void (*callback)(int,int,void *),void *cb_arg);
204 #endif /* !defined(OPENSSL_NO_DEPRECATED) */

206 /* New version */
207 int     DH_generate_parameters_ex(DH *dh, int prime_len,int generator, BN_GENCB

209 int     DH_check(const DH *dh,int *codes);
210 int     DH_check_pub_key(const DH *dh,const BIGNUM *pub_key, int *codes);
211 int     DH_generate_key(DH *dh);
212 int     DH_compute_key(unsigned char *key,const BIGNUM *pub_key,DH *dh);
213 DH *    d2i_DHparams(DH **a,const unsigned char **pp, long length);
214 int     i2d_DHparams(const DH *a,unsigned char **pp);
215 #ifndef OPENSSL_NO_FP_API
216 int     DHparams_print_fp(FILE *fp, const DH *x);
217 #endif
218 #ifndef OPENSSL_NO_BIO
219 int     DHparams_print(BIO *bp, const DH *x);
220 #else
221 int     DHparams_print(char *bp, const DH *x);
222 #endif

224 #define EVP_PKEY_CTX_set_dh_paramgen_prime_len(ctx, len) \
225         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_DH, EVP_PKEY_OP_PARAMGEN, \
226                           EVP_PKEY_CTRL_DH_PARAMGEN_PRIME_LEN, len, NULL)

228 #define EVP_PKEY_CTX_set_dh_paramgen_generator(ctx, gen) \
229         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_DH, EVP_PKEY_OP_PARAMGEN, \
230                           EVP_PKEY_CTRL_DH_PARAMGEN_GENERATOR, gen, NULL)

232 #define EVP_PKEY_CTRL_DH_PARAMGEN_PRIME_LEN     (EVP_PKEY_ALG_CTRL + 1)
233 #define EVP_PKEY_CTRL_DH_PARAMGEN_GENERATOR     (EVP_PKEY_ALG_CTRL + 2)


236 /* BEGIN ERROR CODES */
237 /* The following lines are auto generated by the script mkerr.pl. Any changes
238  * made after this point may be overwritten when the script is next run.
239  */
240 void ERR_load_DH_strings(void);

242 /* Error codes for the DH functions. */

244 /* Function codes. */
245 #define DH_F_COMPUTE_KEY                                102
246 #define DH_F_DHPARAMS_PRINT_FP                          101
247 #define DH_F_DH_BUILTIN_GENPARAMS                       106
248 #define DH_F_DH_COMPUTE_KEY                             114
249 #define DH_F_DH_GENERATE_KEY                            115
250 #define DH_F_DH_GENERATE_PARAMETERS_EX                  116
251 #define DH_F_DH_NEW_METHOD                              105
252 #define DH_F_DH_PARAM_DECODE                            107
253 #define DH_F_DH_PRIV_DECODE                             110
254 #define DH_F_DH_PRIV_ENCODE                             111
255 #define DH_F_DH_PUB_DECODE                              108
256 #define DH_F_DH_PUB_ENCODE                              109
257 #define DH_F_DO_DH_PRINT                                100
258 #define DH_F_GENERATE_KEY                               103
259 #define DH_F_GENERATE_PARAMETERS                        104
```

```
260 #define DH_F_PKEY_DH_DERIVE                             112
261 #define DH_F_PKEY_DH_KEYGEN                             113

263 /* Reason codes. */
264 #define DH_R_BAD_GENERATOR                              101
265 #define DH_R_BN_DECODE_ERROR                            109
266 #define DH_R_BN_ERROR                                   106
267 #define DH_R_DECODE_ERROR                               104
268 #define DH_R_INVALID_PUBKEY                             102
269 #define DH_R_KEYS_NOT_SET                               108
270 #define DH_R_KEY_SIZE_TOO_SMALL                         110
271 #define DH_R_MODULUS_TOO_LARGE                          103
272 #define DH_R_NON_FIPS_METHOD                            111
273 #define DH_R_NO_PARAMETERS_SET                          107
274 #define DH_R_NO_PRIVATE_VALUE                           100
275 #define DH_R_PARAMETER_ENCODING_ERROR                   105

277 #ifdef  __cplusplus
278 }
279 #endif
280 #endif
281 #endif /* ! codereview */
```

```
**********************************************************
   12051 Wed Aug 13 19:51:43 2014
new/usr/src/lib/openssl/include/openssl/dsa.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/*
 * The DSS routines are based on patches supplied by
 * Steven Schoch <schoch@sheba.arc.nasa.gov>.  He basically did the
```



```
 * work and I have just tweaked them a little to fit into my
 * stylistic vision for SSLeay :-) */

#ifndef HEADER_DSA_H
#define HEADER_DSA_H

#include <openssl/e_os2.h>

#ifdef OPENSSL_NO_DSA
#error DSA is disabled.
#endif

#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/crypto.h>
#include <openssl/ossl_typ.h>

#ifndef OPENSSL_NO_DEPRECATED
#include <openssl/bn.h>
#ifndef OPENSSL_NO_DH
# include <openssl/dh.h>
#endif
#endif

#ifndef OPENSSL_DSA_MAX_MODULUS_BITS
# define OPENSSL_DSA_MAX_MODULUS_BITS   10000
#endif

#define DSA_FLAG_CACHE_MONT_P   0x01
#define DSA_FLAG_NO_EXP_CONSTTIME       0x02 /* new with 0.9.7h; the built-in DS
                                              * implementation now uses constant
                                              * modular exponentiation for secre
                                              * by default. This flag causes the
                                              * faster variable sliding window m
                                              * be used for all exponents.
                                              */

/* If this flag is set the DSA method is FIPS compliant and can be used
 * in FIPS mode. This is set in the validated module method. If an
 * application sets this flag in its own methods it is its reposibility
 * to ensure the result is compliant.
 */

#define DSA_FLAG_FIPS_METHOD                    0x0400

/* If this flag is set the operations normally disabled in FIPS mode are
 * permitted it is then the applications responsibility to ensure that the
 * usage is compliant.
 */

#define DSA_FLAG_NON_FIPS_ALLOW                 0x0400

#ifdef  __cplusplus
extern "C" {
#endif

/* Already defined in ossl_typ.h */
/* typedef struct dsa_st DSA; */
/* typedef struct dsa_method DSA_METHOD; */

typedef struct DSA_SIG_st
        {
        BIGNUM *r;
        BIGNUM *s;
        } DSA_SIG;
```

```
129 struct dsa_method
130         {
131         const char *name;
132         DSA_SIG * (*dsa_do_sign)(const unsigned char *dgst, int dlen, DSA *dsa);
133         int (*dsa_sign_setup)(DSA *dsa, BN_CTX *ctx_in, BIGNUM **kinvp,
134                                                         BIGNUM **rp);
135         int (*dsa_do_verify)(const unsigned char *dgst, int dgst_len,
136                              DSA_SIG *sig, DSA *dsa);
137         int (*dsa_mod_exp)(DSA *dsa, BIGNUM *rr, BIGNUM *a1, BIGNUM *p1,
138                         BIGNUM *a2, BIGNUM *p2, BIGNUM *m, BN_CTX *ctx,
139                         BN_MONT_CTX *in_mont);
140         int (*bn_mod_exp)(DSA *dsa, BIGNUM *r, BIGNUM *a, const BIGNUM *p,
141                           const BIGNUM *m, BN_CTX *ctx,
142                           BN_MONT_CTX *m_ctx); /* Can be null */
143         int (*init)(DSA *dsa);
144         int (*finish)(DSA *dsa);
145         int flags;
146         char *app_data;
147         /* If this is non-NULL, it is used to generate DSA parameters */
148         int (*dsa_paramgen)(DSA *dsa, int bits,
149                         const unsigned char *seed, int seed_len,
150                         int *counter_ret, unsigned long *h_ret,
151                         BN_GENCB *cb);
152         /* If this is non-NULL, it is used to generate DSA keys */
153         int (*dsa_keygen)(DSA *dsa);
154         };

156 struct dsa_st
157         {
158         /* This first variable is used to pick up errors where
159          * a DSA is passed instead of of a EVP_PKEY */
160         int pad;
161         long version;
162         int write_params;
163         BIGNUM *p;
164         BIGNUM *q;      /* == 20 */
165         BIGNUM *g;

167         BIGNUM *pub_key;  /* y public key */
168         BIGNUM *priv_key; /* x private key */

170         BIGNUM *kinv;   /* Signing pre-calc */
171         BIGNUM *r;      /* Signing pre-calc */

173         int flags;
174         /* Normally used to cache montgomery values */
175         BN_MONT_CTX *method_mont_p;
176         int references;
177         CRYPTO_EX_DATA ex_data;
178         const DSA_METHOD *meth;
179         /* functional reference if 'meth' is ENGINE-provided */
180         ENGINE *engine;
181         };

183 #define d2i_DSAparams_fp(fp,x) (DSA *)ASN1_d2i_fp((char *(*)())DSA_new, \
184                 (char *(*)())d2i_DSAparams,(fp),(unsigned char **)(x))
185 #define i2d_DSAparams_fp(fp,x) ASN1_i2d_fp(i2d_DSAparams,(fp), \
186                 (unsigned char *)(x))
187 #define d2i_DSAparams_bio(bp,x) ASN1_d2i_bio_of(DSA,DSA_new,d2i_DSAparams,bp,x)
188 #define i2d_DSAparams_bio(bp,x) ASN1_i2d_bio_of_const(DSA,i2d_DSAparams,bp,x)


191 DSA *DSAparams_dup(DSA *x);
192 DSA_SIG * DSA_SIG_new(void);
193 void    DSA_SIG_free(DSA_SIG *a);
```



```
194 int     i2d_DSA_SIG(const DSA_SIG *a, unsigned char **pp);
195 DSA_SIG * d2i_DSA_SIG(DSA_SIG **v, const unsigned char **pp, long length);

197 DSA_SIG * DSA_do_sign(const unsigned char *dgst,int dlen,DSA *dsa);
198 int     DSA_do_verify(const unsigned char *dgst,int dgst_len,
199                       DSA_SIG *sig,DSA *dsa);

201 const DSA_METHOD *DSA_OpenSSL(void);

203 void    DSA_set_default_method(const DSA_METHOD *);
204 const DSA_METHOD *DSA_get_default_method(void);
205 int     DSA_set_method(DSA *dsa, const DSA_METHOD *);

207 DSA *   DSA_new(void);
208 DSA *   DSA_new_method(ENGINE *engine);
209 void    DSA_free (DSA *r);
210 /* "up" the DSA object's reference count */
211 int     DSA_up_ref(DSA *r);
212 int     DSA_size(const DSA *);
213         /* next 4 return -1 on error */
214 int     DSA_sign_setup( DSA *dsa,BN_CTX *ctx_in,BIGNUM **kinvp,BIGNUM **rp);
215 int     DSA_sign(int type,const unsigned char *dgst,int dlen,
216                 unsigned char *sig, unsigned int *siglen, DSA *dsa);
217 int     DSA_verify(int type,const unsigned char *dgst,int dgst_len,
218                 const unsigned char *sigbuf, int siglen, DSA *dsa);
219 int DSA_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
220                          CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
221 int DSA_set_ex_data(DSA *d, int idx, void *arg);
222 void *DSA_get_ex_data(DSA *d, int idx);

224 DSA *   d2i_DSAPublicKey(DSA **a, const unsigned char **pp, long length);
225 DSA *   d2i_DSAPrivateKey(DSA **a, const unsigned char **pp, long length);
226 DSA *   d2i_DSAparams(DSA **a, const unsigned char **pp, long length);

228 /* Deprecated version */
229 #ifndef OPENSSL_NO_DEPRECATED
230 DSA *   DSA_generate_parameters(int bits,
231                 unsigned char *seed,int seed_len,
232                 int *counter_ret, unsigned long *h_ret,void
233                 (*callback)(int, int, void *),void *cb_arg);
234 #endif /* !defined(OPENSSL_NO_DEPRECATED) */

236 /* New version */
237 int     DSA_generate_parameters_ex(DSA *dsa, int bits,
238                 const unsigned char *seed,int seed_len,
239                 int *counter_ret, unsigned long *h_ret, BN_GENCB *cb);

241 int     DSA_generate_key(DSA *a);
242 int     i2d_DSAPublicKey(const DSA *a, unsigned char **pp);
243 int     i2d_DSAPrivateKey(const DSA *a, unsigned char **pp);
244 int     i2d_DSAparams(const DSA *a,unsigned char **pp);

246 #ifndef OPENSSL_NO_BIO
247 int     DSAparams_print(BIO *bp, const DSA *x);
248 int     DSA_print(BIO *bp, const DSA *x, int off);
249 #endif
250 #ifndef OPENSSL_NO_FP_API
251 int     DSAparams_print_fp(FILE *fp, const DSA *x);
252 int     DSA_print_fp(FILE *bp, const DSA *x, int off);
253 #endif

255 #define DSS_prime_checks 50
256 /* Primality test according to FIPS PUB 186[-1], Appendix 2.1:
257  * 50 rounds of Rabin-Miller */
258 #define DSA_is_prime(n, callback, cb_arg) \
259         BN_is_prime(n, DSS_prime_checks, callback, NULL, cb_arg)
```

```
#ifndef OPENSSL_NO_DH
/* Convert DSA structure (key or just parameters) into DH structure
 * (be careful to avoid small subgroup attacks when using this!) */
DH *DSA_dup_DH(const DSA *r);
#endif

#define EVP_PKEY_CTX_set_dsa_paramgen_bits(ctx, nbits) \
        EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_DSA, EVP_PKEY_OP_PARAMGEN, \
                          EVP_PKEY_CTRL_DSA_PARAMGEN_BITS, nbits, NULL)

#define EVP_PKEY_CTRL_DSA_PARAMGEN_BITS         (EVP_PKEY_ALG_CTRL + 1)
#define EVP_PKEY_CTRL_DSA_PARAMGEN_Q_BITS       (EVP_PKEY_ALG_CTRL + 2)
#define EVP_PKEY_CTRL_DSA_PARAMGEN_MD           (EVP_PKEY_ALG_CTRL + 3)

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_DSA_strings(void);

/* Error codes for the DSA functions. */

/* Function codes. */
#define DSA_F_D2I_DSA_SIG                               110
#define DSA_F_DO_DSA_PRINT                              104
#define DSA_F_DSAPARAMS_PRINT                           100
#define DSA_F_DSAPARAMS_PRINT_FP                        101
#define DSA_F_DSA_DO_SIGN                               112
#define DSA_F_DSA_DO_VERIFY                             113
#define DSA_F_DSA_GENERATE_KEY                          124
#define DSA_F_DSA_GENERATE_PARAMETERS_EX                123
#define DSA_F_DSA_NEW_METHOD                            103
#define DSA_F_DSA_PARAM_DECODE                          119
#define DSA_F_DSA_PRINT_FP                              105
#define DSA_F_DSA_PRIV_DECODE                           115
#define DSA_F_DSA_PRIV_ENCODE                           116
#define DSA_F_DSA_PUB_DECODE                            117
#define DSA_F_DSA_PUB_ENCODE                            118
#define DSA_F_DSA_SIGN                                  106
#define DSA_F_DSA_SIGN_SETUP                            107
#define DSA_F_DSA_SIG_NEW                               109
#define DSA_F_DSA_SIG_PRINT                             125
#define DSA_F_DSA_VERIFY                                108
#define DSA_F_I2D_DSA_SIG                               111
#define DSA_F_OLD_DSA_PRIV_DECODE                       122
#define DSA_F_PKEY_DSA_CTRL                             120
#define DSA_F_PKEY_DSA_KEYGEN                           121
#define DSA_F_SIG_CB                                    114

/* Reason codes. */
#define DSA_R_BAD_Q_VALUE                               102
#define DSA_R_BN_DECODE_ERROR                           108
#define DSA_R_BN_ERROR                                  109
#define DSA_R_DATA_TOO_LARGE_FOR_KEY_SIZE               100
#define DSA_R_DECODE_ERROR                              104
#define DSA_R_INVALID_DIGEST_TYPE                       106
#define DSA_R_MISSING_PARAMETERS                        101
#define DSA_R_MODULUS_TOO_LARGE                         103
#define DSA_R_NEED_NEW_SETUP_VALUES                     110
#define DSA_R_NON_FIPS_DSA_METHOD                       111
#define DSA_R_NO_PARAMETERS_SET                         107
#define DSA_R_PARAMETER_ENCODING_ERROR                  105

#ifdef  __cplusplus
}
```



```
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   18093 Wed Aug 13 19:51:43 2014
new/usr/src/lib/openssl/include/openssl/dso.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso.h -*- mode:C; c-file-style: "eay" -*- */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_DSO_H
#define HEADER_DSO_H
```



```
#include <openssl/crypto.h>

#ifdef __cplusplus
extern "C" {
#endif

/* These values are used as commands to DSO_ctrl() */
#define DSO_CTRL_GET_FLAGS      1
#define DSO_CTRL_SET_FLAGS      2
#define DSO_CTRL_OR_FLAGS       3

/* By default, DSO_load() will translate the provided filename into a form
 * typical for the platform (more specifically the DSO_METHOD) using the
 * dso_name_converter function of the method. Eg. win32 will transform "blah"
 * into "blah.dll", and dlfcn will transform it into "libblah.so". The
 * behaviour can be overriden by setting the name_converter callback in the DSO
 * object (using DSO_set_name_converter()). This callback could even utilise
 * the DSO_METHOD's converter too if it only wants to override behaviour for
 * one or two possible DSO methods. However, the following flag can be set in a
 * DSO to prevent *any* native name-translation at all - eg. if the caller has
 * prompted the user for a path to a driver library so the filename should be
 * interpreted as-is. */
#define DSO_FLAG_NO_NAME_TRANSLATION            0x01
/* An extra flag to give if only the extension should be added as
 * translation.  This is obviously only of importance on Unix and
 * other operating systems where the translation also may prefix
 * the name with something, like 'lib', and ignored everywhere else.
 * This flag is also ignored if DSO_FLAG_NO_NAME_TRANSLATION is used
 * at the same time. */
#define DSO_FLAG_NAME_TRANSLATION_EXT_ONLY      0x02

/* The following flag controls the translation of symbol names to upper
 * case.  This is currently only being implemented for OpenVMS.
 */
#define DSO_FLAG_UPCASE_SYMBOL                  0x10

/* This flag loads the library with public symbols.
 * Meaning: The exported symbols of this library are public
 * to all libraries loaded after this library.
 * At the moment only implemented in unix.
 */
#define DSO_FLAG_GLOBAL_SYMBOLS                 0x20


typedef void (*DSO_FUNC_TYPE)(void);

typedef struct dso_st DSO;

/* The function prototype used for method functions (or caller-provided
 * callbacks) that transform filenames. They are passed a DSO structure pointer
 * (or NULL if they are to be used independantly of a DSO object) and a
 * filename to transform. They should either return NULL (if there is an error
 * condition) or a newly allocated string containing the transformed form that
 * the caller will need to free with OPENSSL_free() when done. */
typedef char* (*DSO_NAME_CONVERTER_FUNC)(DSO *, const char *);
/* The function prototype used for method functions (or caller-provided
 * callbacks) that merge two file specifications. They are passed a
 * DSO structure pointer (or NULL if they are to be used independantly of
 * a DSO object) and two file specifications to merge. They should
 * either return NULL (if there is an error condition) or a newly allocated
 * string containing the result of merging that the caller will need
 * to free with OPENSSL_free() when done.
 * Here, merging means that bits and pieces are taken from each of the
 * file specifications and added together in whatever fashion that is
 * sensible for the DSO method in question.  The only rule that really
 * applies is that if the two specification contain pieces of the same
```

```
 * type, the copy from the first string takes priority.  One could see
 * it as the first specification is the one given by the user and the
 * second being a bunch of defaults to add on if they're missing in the
 * first. */
typedef char* (*DSO_MERGER_FUNC)(DSO *, const char *, const char *);

typedef struct dso_meth_st
        {
        const char *name;
        /* Loads a shared library, NB: new DSO_METHODs must ensure that a
         * successful load populates the loaded_filename field, and likewise a
         * successful unload OPENSSL_frees and NULLs it out. */
        int (*dso_load)(DSO *dso);
        /* Unloads a shared library */
        int (*dso_unload)(DSO *dso);
        /* Binds a variable */
        void *(*dso_bind_var)(DSO *dso, const char *symname);
        /* Binds a function - assumes a return type of DSO_FUNC_TYPE.
         * This should be cast to the real function prototype by the
         * caller. Platforms that don't have compatible representations
         * for different prototypes (this is possible within ANSI C)
         * are highly unlikely to have shared libraries at all, let
         * alone a DSO_METHOD implemented for them. */
        DSO_FUNC_TYPE (*dso_bind_func)(DSO *dso, const char *symname);

/* I don't think this would actually be used in any circumstances. */
#if 0
        /* Unbinds a variable */
        int (*dso_unbind_var)(DSO *dso, char *symname, void *symptr);
        /* Unbinds a function */
        int (*dso_unbind_func)(DSO *dso, char *symname, DSO_FUNC_TYPE symptr);
#endif
        /* The generic (yuck) "ctrl()" function. NB: Negative return
         * values (rather than zero) indicate errors. */
        long (*dso_ctrl)(DSO *dso, int cmd, long larg, void *parg);
        /* The default DSO_METHOD-specific function for converting filenames to
         * a canonical native form. */
        DSO_NAME_CONVERTER_FUNC dso_name_converter;
        /* The default DSO_METHOD-specific function for converting filenames to
         * a canonical native form. */
        DSO_MERGER_FUNC dso_merger;

        /* [De]Initialisation handlers. */
        int (*init)(DSO *dso);
        int (*finish)(DSO *dso);

        /* Return pathname of the module containing location */
        int (*pathbyaddr)(void *addr,char *path,int sz);
        /* Perform global symbol lookup, i.e. among *all* modules */
        void *(*globallookup)(const char *symname);
        } DSO_METHOD;

/**********************************************************************/
/* The low-level handle type used to refer to a loaded shared library */

struct dso_st
        {
        DSO_METHOD *meth;
        /* Standard dlopen uses a (void *). Win32 uses a HANDLE. VMS
         * doesn't use anything but will need to cache the filename
         * for use in the dso_bind handler. All in all, let each
         * method control its own destiny. "Handles" and such go in
         * a STACK. */
        STACK_OF(void) *meth_data;
        int references;
        int flags;
```



```
        /* For use by applications etc ... use this for your bits'n'pieces,
         * don't touch meth_data! */
        CRYPTO_EX_DATA ex_data;
        /* If this callback function pointer is set to non-NULL, then it will
         * be used in DSO_load() in place of meth->dso_name_converter. NB: This
         * should normally set using DSO_set_name_converter(). */
        DSO_NAME_CONVERTER_FUNC name_converter;
        /* If this callback function pointer is set to non-NULL, then it will
         * be used in DSO_load() in place of meth->dso_merger. NB: This
         * should normally set using DSO_set_merger(). */
        DSO_MERGER_FUNC merger;
        /* This is populated with (a copy of) the platform-independant
         * filename used for this DSO. */
        char *filename;
        /* This is populated with (a copy of) the translated filename by which
         * the DSO was actually loaded. It is NULL iff the DSO is not currently
         * loaded. NB: This is here because the filename translation process
         * may involve a callback being invoked more than once not only to
         * convert to a platform-specific form, but also to try different
         * filenames in the process of trying to perform a load. As such, this
         * variable can be used to indicate (a) whether this DSO structure
         * corresponds to a loaded library or not, and (b) the filename with
         * which it was actually loaded. */
        char *loaded_filename;
        };


DSO *   DSO_new(void);
DSO *   DSO_new_method(DSO_METHOD *method);
int     DSO_free(DSO *dso);
int     DSO_flags(DSO *dso);
int     DSO_up_ref(DSO *dso);
long    DSO_ctrl(DSO *dso, int cmd, long larg, void *parg);

/* This function sets the DSO's name_converter callback. If it is non-NULL,
 * then it will be used instead of the associated DSO_METHOD's function. If
 * oldcb is non-NULL then it is set to the function pointer value being
 * replaced. Return value is non-zero for success. */
int     DSO_set_name_converter(DSO *dso, DSO_NAME_CONVERTER_FUNC cb,
                               DSO_NAME_CONVERTER_FUNC *oldcb);
/* These functions can be used to get/set the platform-independant filename
 * used for a DSO. NB: set will fail if the DSO is already loaded. */
const char *DSO_get_filename(DSO *dso);
int     DSO_set_filename(DSO *dso, const char *filename);
/* This function will invoke the DSO's name_converter callback to translate a
 * filename, or if the callback isn't set it will instead use the DSO_METHOD's
 * converter. If "filename" is NULL, the "filename" in the DSO itself will be
 * used. If the DSO_FLAG_NO_NAME_TRANSLATION flag is set, then the filename is
 * simply duplicated. NB: This function is usually called from within a
 * DSO_METHOD during the processing of a DSO_load() call, and is exposed so that
 * caller-created DSO_METHODs can do the same thing. A non-NULL return value
 * will need to be OPENSSL_free()'d. */
char    *DSO_convert_filename(DSO *dso, const char *filename);
/* This function will invoke the DSO's merger callback to merge two file
 * specifications, or if the callback isn't set it will instead use the
 * DSO_METHOD's merger.  A non-NULL return value will need to be
 * OPENSSL_free()'d. */
char    *DSO_merge(DSO *dso, const char *filespec1, const char *filespec2);
/* If the DSO is currently loaded, this returns the filename that it was loaded
 * under, otherwise it returns NULL. So it is also useful as a test as to
 * whether the DSO is currently loaded. NB: This will not necessarily return
 * the same value as DSO_convert_filename(dso, dso->filename), because the
 * DSO_METHOD's load function may have tried a variety of filenames (with
 * and/or without the aid of the converters) before settling on the one it
 * actually loaded. */
const char *DSO_get_loaded_filename(DSO *dso);
```

```
261 void    DSO_set_default_method(DSO_METHOD *meth);
262 DSO_METHOD *DSO_get_default_method(void);
263 DSO_METHOD *DSO_get_method(DSO *dso);
264 DSO_METHOD *DSO_set_method(DSO *dso, DSO_METHOD *meth);

266 /* The all-singing all-dancing load function, you normally pass NULL
267  * for the first and third parameters. Use DSO_up and DSO_free for
268  * subsequent reference count handling. Any flags passed in will be set
269  * in the constructed DSO after its init() function but before the
270  * load operation. If 'dso' is non-NULL, 'flags' is ignored. */
271 DSO *DSO_load(DSO *dso, const char *filename, DSO_METHOD *meth, int flags);

273 /* This function binds to a variable inside a shared library. */
274 void *DSO_bind_var(DSO *dso, const char *symname);

276 /* This function binds to a function inside a shared library. */
277 DSO_FUNC_TYPE DSO_bind_func(DSO *dso, const char *symname);

279 /* This method is the default, but will beg, borrow, or steal whatever
280  * method should be the default on any particular platform (including
281  * DSO_METH_null() if necessary). */
282 DSO_METHOD *DSO_METHOD_openssl(void);

284 /* This method is defined for all platforms - if a platform has no
285  * DSO support then this will be the only method! */
286 DSO_METHOD *DSO_METHOD_null(void);

288 /* If DSO_DLFCN is defined, the standard dlfcn.h-style functions
289  * (dlopen, dlclose, dlsym, etc) will be used and incorporated into
290  * this method. If not, this method will return NULL. */
291 DSO_METHOD *DSO_METHOD_dlfcn(void);

293 /* If DSO_DL is defined, the standard dl.h-style functions (shl_load,
294  * shl_unload, shl_findsym, etc) will be used and incorporated into
295  * this method. If not, this method will return NULL. */
296 DSO_METHOD *DSO_METHOD_dl(void);

298 /* If WIN32 is defined, use DLLs. If not, return NULL. */
299 DSO_METHOD *DSO_METHOD_win32(void);

301 /* If VMS is defined, use shared images. If not, return NULL. */
302 DSO_METHOD *DSO_METHOD_vms(void);

304 /* This function writes null-terminated pathname of DSO module
305  * containing 'addr' into 'sz' large caller-provided 'path' and
306  * returns the number of characters [including trailing zero]
307  * written to it. If 'sz' is 0 or negative, 'path' is ignored and
308  * required amount of charachers [including trailing zero] to
309  * accomodate pathname is returned. If 'addr' is NULL, then
310  * pathname of cryptolib itself is returned. Negative or zero
311  * return value denotes error.
312  */
313 int DSO_pathbyaddr(void *addr,char *path,int sz);

315 /* This function should be used with caution! It looks up symbols in
316  * *all* loaded modules and if module gets unloaded by somebody else
317  * attempt to dereference the pointer is doomed to have fatal
318  * consequences. Primary usage for this function is to probe *core*
319  * system functionality, e.g. check if getnameinfo(3) is available
320  * at run-time without bothering about OS-specific details such as
321  * libc.so.versioning or where does it actually reside: in libc
322  * itself or libsocket. */
323 void *DSO_global_lookup(const char *name);

325 /* If BeOS is defined, use shared images. If not, return NULL. */
```



```
326 DSO_METHOD *DSO_METHOD_beos(void);

328 /* BEGIN ERROR CODES */
329 /* The following lines are auto generated by the script mkerr.pl. Any changes
330  * made after this point may be overwritten when the script is next run.
331  */
332 void ERR_load_DSO_strings(void);

334 /* Error codes for the DSO functions. */

336 /* Function codes. */
337 #define DSO_F_BEOS_BIND_FUNC                                144
338 #define DSO_F_BEOS_BIND_VAR                                 145
339 #define DSO_F_BEOS_LOAD                                     146
340 #define DSO_F_BEOS_NAME_CONVERTER                           147
341 #define DSO_F_BEOS_UNLOAD                                   148
342 #define DSO_F_DLFCN_BIND_FUNC                               100
343 #define DSO_F_DLFCN_BIND_VAR                                101
344 #define DSO_F_DLFCN_LOAD                                    102
345 #define DSO_F_DLFCN_MERGER                                  130
346 #define DSO_F_DLFCN_NAME_CONVERTER                          123
347 #define DSO_F_DLFCN_UNLOAD                                  103
348 #define DSO_F_DL_BIND_FUNC                                  104
349 #define DSO_F_DL_BIND_VAR                                   105
350 #define DSO_F_DL_LOAD                                       106
351 #define DSO_F_DL_MERGER                                     131
352 #define DSO_F_DL_NAME_CONVERTER                             124
353 #define DSO_F_DL_UNLOAD                                     107
354 #define DSO_F_DSO_BIND_FUNC                                 108
355 #define DSO_F_DSO_BIND_VAR                                  109
356 #define DSO_F_DSO_CONVERT_FILENAME                          126
357 #define DSO_F_DSO_CTRL                                      110
358 #define DSO_F_DSO_FREE                                      111
359 #define DSO_F_DSO_GET_FILENAME                              127
360 #define DSO_F_DSO_GET_LOADED_FILENAME                       128
361 #define DSO_F_DSO_GLOBAL_LOOKUP                             139
362 #define DSO_F_DSO_LOAD                                      112
363 #define DSO_F_DSO_MERGE                                     132
364 #define DSO_F_DSO_NEW_METHOD                                113
365 #define DSO_F_DSO_PATHBYADDR                                140
366 #define DSO_F_DSO_SET_FILENAME                              129
367 #define DSO_F_DSO_SET_NAME_CONVERTER                        122
368 #define DSO_F_DSO_UP_REF                                    114
369 #define DSO_F_GLOBAL_LOOKUP_FUNC                            138
370 #define DSO_F_PATHBYADDR                                    137
371 #define DSO_F_VMS_BIND_SYM                                  115
372 #define DSO_F_VMS_LOAD                                      116
373 #define DSO_F_VMS_MERGER                                    133
374 #define DSO_F_VMS_UNLOAD                                    117
375 #define DSO_F_WIN32_BIND_FUNC                               118
376 #define DSO_F_WIN32_BIND_VAR                                119
377 #define DSO_F_WIN32_GLOBALLOOKUP                            142
378 #define DSO_F_WIN32_GLOBALLOOKUP_FUNC                       143
379 #define DSO_F_WIN32_JOINER                                  135
380 #define DSO_F_WIN32_LOAD                                    120
381 #define DSO_F_WIN32_MERGER                                  134
382 #define DSO_F_WIN32_NAME_CONVERTER                          125
383 #define DSO_F_WIN32_PATHBYADDR                              141
384 #define DSO_F_WIN32_SPLITTER                                136
385 #define DSO_F_WIN32_UNLOAD                                  121

387 /* Reason codes. */
388 #define DSO_R_CTRL_FAILED                                   100
389 #define DSO_R_DSO_ALREADY_LOADED                            110
390 #define DSO_R_EMPTY_FILE_STRUCTURE                          113
391 #define DSO_R_FAILURE                                       114
```

```
392 #define DSO_R_FILENAME_TOO_BIG                          101
393 #define DSO_R_FINISH_FAILED                             102
394 #define DSO_R_INCORRECT_FILE_SYNTAX                     115
395 #define DSO_R_LOAD_FAILED                               103
396 #define DSO_R_NAME_TRANSLATION_FAILED                   109
397 #define DSO_R_NO_FILENAME                               111
398 #define DSO_R_NO_FILE_SPECIFICATION                     116
399 #define DSO_R_NULL_HANDLE                               104
400 #define DSO_R_SET_FILENAME_FAILED                       112
401 #define DSO_R_STACK_ERROR                               105
402 #define DSO_R_SYM_FAILURE                               106
403 #define DSO_R_UNLOAD_FAILED                             107
404 #define DSO_R_UNSUPPORTED                               108

406 #ifdef  __cplusplus
407 }
408 #endif
409 #endif
410 #endif /* ! codereview */
```

```
**********************************************************
    8016 Wed Aug 13 19:51:43 2014
new/usr/src/lib/openssl/include/openssl/dtls1.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/dtls1.h */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_DTLS1_H
#define HEADER_DTLS1_H
```



```
#include <openssl/buffer.h>
#include <openssl/pqueue.h>
#ifdef OPENSSL_SYS_VMS
#include <resource.h>
#include <sys/timeb.h>
#endif
#ifdef OPENSSL_SYS_WIN32
/* Needed for struct timeval */
#include <winsock.h>
#elif defined(OPENSSL_SYS_NETWARE) && !defined(_WINSOCK2API_)
#include <sys/timeval.h>
#else
#if defined(OPENSSL_SYS_VXWORKS)
#include <sys/times.h>
#else
#include <sys/time.h>
#endif
#endif

#ifdef  __cplusplus
extern "C" {
#endif

#define DTLS1_VERSION                   0xFEFF
#define DTLS1_BAD_VER                   0x0100

#if 0
/* this alert description is not specified anywhere... */
#define DTLS1_AD_MISSING_HANDSHAKE_MESSAGE    110
#endif

/* lengths of messages */
#define DTLS1_COOKIE_LENGTH                     256

#define DTLS1_RT_HEADER_LENGTH                  13

#define DTLS1_HM_HEADER_LENGTH                  12

#define DTLS1_HM_BAD_FRAGMENT                   -2
#define DTLS1_HM_FRAGMENT_RETRY                 -3

#define DTLS1_CCS_HEADER_LENGTH                  1

#ifdef DTLS1_AD_MISSING_HANDSHAKE_MESSAGE
#define DTLS1_AL_HEADER_LENGTH                   7
#else
#define DTLS1_AL_HEADER_LENGTH                   2
#endif

#ifndef OPENSSL_NO_SSL_INTERN

#ifndef OPENSSL_NO_SCTP
#define DTLS1_SCTP_AUTH_LABEL   "EXPORTER_DTLS_OVER_SCTP"
#endif

typedef struct dtls1_bitmap_st
        {
        unsigned long map;              /* track 32 packets on 32-bit systems
                                           and 64 - on 64-bit systems */
        unsigned char max_seq_num[8];   /* max record number seen so far,
                                           64-bit value in big-endian
                                           encoding */
        } DTLS1_BITMAP;

struct dtls1_retransmit_state
```

```
128         {
129         EVP_CIPHER_CTX *enc_write_ctx;  /* cryptographic state */
130         EVP_MD_CTX *write_hash;                 /* used for mac generation */
131 #ifndef OPENSSL_NO_COMP
132         COMP_CTX *compress;                             /* compression */
133 #else
134         char *compress;
135 #endif
136         SSL_SESSION *session;
137         unsigned short epoch;
138         };

140 struct hm_header_st
141         {
142         unsigned char type;
143         unsigned long msg_len;
144         unsigned short seq;
145         unsigned long frag_off;
146         unsigned long frag_len;
147         unsigned int is_ccs;
148         struct dtls1_retransmit_state saved_retransmit_state;
149         };

151 struct ccs_header_st
152         {
153         unsigned char type;
154         unsigned short seq;
155         };

157 struct dtls1_timeout_st
158         {
159         /* Number of read timeouts so far */
160         unsigned int read_timeouts;

162         /* Number of write timeouts so far */
163         unsigned int write_timeouts;

165         /* Number of alerts received so far */
166         unsigned int num_alerts;
167         };

169 typedef struct record_pqueue_st
170         {
171         unsigned short epoch;
172         pqueue q;
173         } record_pqueue;

175 typedef struct hm_fragment_st
176         {
177         struct hm_header_st msg_header;
178         unsigned char *fragment;
179         unsigned char *reassembly;
180         } hm_fragment;

182 typedef struct dtls1_state_st
183         {
184         unsigned int send_cookie;
185         unsigned char cookie[DTLS1_COOKIE_LENGTH];
186         unsigned char rcvd_cookie[DTLS1_COOKIE_LENGTH];
187         unsigned int cookie_len;

189         /*
190          * The current data and handshake epoch.  This is initially
191          * undefined, and starts at zero once the initial handshake is
192          * completed
193          */
```



```
194         unsigned short r_epoch;
195         unsigned short w_epoch;

197         /* records being received in the current epoch */
198         DTLS1_BITMAP bitmap;

200         /* renegotiation starts a new set of sequence numbers */
201         DTLS1_BITMAP next_bitmap;

203         /* handshake message numbers */
204         unsigned short handshake_write_seq;
205         unsigned short next_handshake_write_seq;

207         unsigned short handshake_read_seq;

209         /* save last sequence number for retransmissions */
210         unsigned char last_write_sequence[8];

212         /* Received handshake records (processed and unprocessed) */
213         record_pqueue unprocessed_rcds;
214         record_pqueue processed_rcds;

216         /* Buffered handshake messages */
217         pqueue buffered_messages;

219         /* Buffered (sent) handshake records */
220         pqueue sent_messages;

222         /* Buffered application records.
223          * Only for records between CCS and Finished
224          * to prevent either protocol violation or
225          * unnecessary message loss.
226          */
227         record_pqueue buffered_app_data;

229         /* Is set when listening for new connections with dtls1_listen() */
230         unsigned int listen;

232         unsigned int mtu; /* max DTLS packet size */

234         struct hm_header_st w_msg_hdr;
235         struct hm_header_st r_msg_hdr;

237         struct dtls1_timeout_st timeout;

239         /* Indicates when the last handshake msg or heartbeat sent will timeout
240         struct timeval next_timeout;

242         /* Timeout duration */
243         unsigned short timeout_duration;

245         /* storage for Alert/Handshake protocol data received but not
246          * yet processed by ssl3_read_bytes: */
247         unsigned char alert_fragment[DTLS1_AL_HEADER_LENGTH];
248         unsigned int alert_fragment_len;
249         unsigned char handshake_fragment[DTLS1_HM_HEADER_LENGTH];
250         unsigned int handshake_fragment_len;

252         unsigned int retransmitting;
253         unsigned int change_cipher_spec_ok;

255 #ifndef OPENSSL_NO_SCTP
256         /* used when SSL_ST_XX_FLUSH is entered */
257         int next_state;

259         int shutdown_received;
```

```
260 #endif

262         } DTLS1_STATE;

264 typedef struct dtls1_record_data_st
265         {
266         unsigned char *packet;
267         unsigned int   packet_length;
268         SSL3_BUFFER    rbuf;
269         SSL3_RECORD    rrec;
270 #ifndef OPENSSL_NO_SCTP
271         struct bio_dgram_sctp_rcvinfo recordinfo;
272 #endif
273         } DTLS1_RECORD_DATA;

275 #endif

277 /* Timeout multipliers (timeout slice is defined in apps/timeouts.h */
278 #define DTLS1_TMO_READ_COUNT                      2
279 #define DTLS1_TMO_WRITE_COUNT                     2

281 #define DTLS1_TMO_ALERT_COUNT                     12

283 #ifdef  __cplusplus
284 }
285 #endif
286 #endif
287 #endif /* ! codereview */
```

```
**********************************************************
   10442 Wed Aug 13 19:51:43 2014
new/usr/src/lib/openssl/include/openssl/e_os2.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* e_os2.h */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/opensslconf.h>

#ifndef HEADER_E_OS2_H
#define HEADER_E_OS2_H

#ifdef  __cplusplus
```



```
extern "C" {
#endif

/*****************************************************************************
 * Detect operating systems.  This probably needs completing.
 * The result is that at least one OPENSSL_SYS_os macro should be defined.
 * However, if none is defined, Unix is assumed.
 **/

#define OPENSSL_SYS_UNIX

/* ---------------------- Macintosh, before MacOS X ---------------------- */
#if defined(__MWERKS__) && defined(macintosh) || defined(OPENSSL_SYSNAME_MAC)
# undef OPENSSL_SYS_UNIX
# define OPENSSL_SYS_MACINTOSH_CLASSIC
#endif

/* ---------------------- NetWare ---------------------------------------- */
#if defined(NETWARE) || defined(OPENSSL_SYSNAME_NETWARE)
# undef OPENSSL_SYS_UNIX
# define OPENSSL_SYS_NETWARE
#endif

/* ---------------------- Microsoft operating systems ---------------------- */

/* Note that MSDOS actually denotes 32-bit environments running on top of
   MS-DOS, such as DJGPP one. */
#if defined(OPENSSL_SYSNAME_MSDOS)
# undef OPENSSL_SYS_UNIX
# define OPENSSL_SYS_MSDOS
#endif

/* For 32 bit environment, there seems to be the CygWin environment and then
   all the others that try to do the same thing Microsoft does... */
#if defined(OPENSSL_SYSNAME_UWIN)
# undef OPENSSL_SYS_UNIX
# define OPENSSL_SYS_WIN32_UWIN
#else
# if defined(__CYGWIN32__) || defined(OPENSSL_SYSNAME_CYGWIN32)
#  undef OPENSSL_SYS_UNIX
#  define OPENSSL_SYS_WIN32_CYGWIN
# else
#  if defined(_WIN32) || defined(OPENSSL_SYSNAME_WIN32)
#   undef OPENSSL_SYS_UNIX
#   define OPENSSL_SYS_WIN32
#  endif
#  if defined(OPENSSL_SYSNAME_WINNT)
#   undef OPENSSL_SYS_UNIX
#   define OPENSSL_SYS_WINNT
#  endif
#  if defined(OPENSSL_SYSNAME_WINCE)
#   undef OPENSSL_SYS_UNIX
#   define OPENSSL_SYS_WINCE
#  endif
# endif
#endif

/* Anything that tries to look like Microsoft is "Windows" */
#if defined(OPENSSL_SYS_WIN32) || defined(OPENSSL_SYS_WINNT) || defined(OPENSSL_
# undef OPENSSL_SYS_UNIX
# define OPENSSL_SYS_WINDOWS
# ifndef OPENSSL_SYS_MSDOS
#  define OPENSSL_SYS_MSDOS
# endif
#endif
```

```
/* DLL settings.  This part is a bit tough, because it's up to the application
   implementor how he or she will link the application, so it requires some
   macro to be used. */
#ifdef OPENSSL_SYS_WINDOWS
# ifndef OPENSSL_OPT_WINDLL
#  if defined(_WINDLL) /* This is used when building OpenSSL to indicate that
                          DLL linkage should be used */
#   define OPENSSL_OPT_WINDLL
#  endif
# endif
#endif

/* ------------------------------- OpenVMS ------------------------------- */
#if defined(__VMS) || defined(VMS) || defined(OPENSSL_SYSNAME_VMS)
# undef OPENSSL_SYS_UNIX
# define OPENSSL_SYS_VMS
# if defined(__DECC)
#  define OPENSSL_SYS_VMS_DECC
# elif defined(__DECCXX)
#  define OPENSSL_SYS_VMS_DECC
#  define OPENSSL_SYS_VMS_DECCXX
# else
#  define OPENSSL_SYS_VMS_NODECC
# endif
#endif

/* -------------------------------- OS/2 -------------------------------- */
#if defined(__EMX__) || defined(__OS2__)
# undef OPENSSL_SYS_UNIX
# define OPENSSL_SYS_OS2
#endif

/* -------------------------------- Unix -------------------------------- */
#ifdef OPENSSL_SYS_UNIX
# if defined(linux) || defined(__linux__) || defined(OPENSSL_SYSNAME_LINUX)
#  define OPENSSL_SYS_LINUX
# endif
# ifdef OPENSSL_SYSNAME_MPE
#  define OPENSSL_SYS_MPE
# endif
# ifdef OPENSSL_SYSNAME_SNI
#  define OPENSSL_SYS_SNI
# endif
# ifdef OPENSSL_SYSNAME_ULTRASPARC
#  define OPENSSL_SYS_ULTRASPARC
# endif
# ifdef OPENSSL_SYSNAME_NEWS4
#  define OPENSSL_SYS_NEWS4
# endif
# ifdef OPENSSL_SYSNAME_MACOSX
#  define OPENSSL_SYS_MACOSX
# endif
# ifdef OPENSSL_SYSNAME_MACOSX_RHAPSODY
#  define OPENSSL_SYS_MACOSX_RHAPSODY
#  define OPENSSL_SYS_MACOSX
# endif
# ifdef OPENSSL_SYSNAME_SUNOS
#  define OPENSSL_SYS_SUNOS
#endif
# if defined(_CRAY) || defined(OPENSSL_SYSNAME_CRAY)
#  define OPENSSL_SYS_CRAY
# endif
# if defined(_AIX) || defined(OPENSSL_SYSNAME_AIX)
#  define OPENSSL_SYS_AIX
# endif
#endif
```



```
/* -------------------------------- VOS -------------------------------- */
#if defined(__VOS__) || defined(OPENSSL_SYSNAME_VOS)
# define OPENSSL_SYS_VOS
#ifdef __HPPA__
# define OPENSSL_SYS_VOS_HPPA
#endif
#ifdef __IA32__
# define OPENSSL_SYS_VOS_IA32
#endif
#endif

/* ------------------------------ VxWorks ------------------------------ */
#ifdef OPENSSL_SYSNAME_VXWORKS
# define OPENSSL_SYS_VXWORKS
#endif

/* -------------------------------- BeOS -------------------------------- */
#if defined(__BEOS__)
# define OPENSSL_SYS_BEOS
# include <sys/socket.h>
# if defined(BONE_VERSION)
#  define OPENSSL_SYS_BEOS_BONE
# else
#  define OPENSSL_SYS_BEOS_R5
# endif
#endif

/**
 * That's it for OS-specific stuff
 *****************************************************************************/


/* Specials for I/O an exit */
#ifdef OPENSSL_SYS_MSDOS
# define OPENSSL_UNISTD_IO <io.h>
# define OPENSSL_DECLARE_EXIT extern void exit(int);
#else
# define OPENSSL_UNISTD_IO OPENSSL_UNISTD
# define OPENSSL_DECLARE_EXIT /* declared in unistd.h */
#endif

/* Definitions of OPENSSL_GLOBAL and OPENSSL_EXTERN, to define and declare
   certain global symbols that, with some compilers under VMS, have to be
   defined and declared explicitely with globaldef and globalref.
   Definitions of OPENSSL_EXPORT and OPENSSL_IMPORT, to define and declare
   DLL exports and imports for compilers under Win32.  These are a little
   more complicated to use.  Basically, for any library that exports some
   global variables, the following code must be present in the header file
   that declares them, before OPENSSL_EXTERN is used:

   #ifdef SOME_BUILD_FLAG_MACRO
   # undef OPENSSL_EXTERN
   # define OPENSSL_EXTERN OPENSSL_EXPORT
   #endif

   The default is to have OPENSSL_EXPORT, OPENSSL_IMPORT and OPENSSL_GLOBAL
   have some generally sensible values, and for OPENSSL_EXTERN to have the
   value OPENSSL_IMPORT.
*/

#if defined(OPENSSL_SYS_VMS_NODECC)
# define OPENSSL_EXPORT globalref
# define OPENSSL_IMPORT globalref
# define OPENSSL_GLOBAL globaldef
#elif defined(OPENSSL_SYS_WINDOWS) && defined(OPENSSL_OPT_WINDLL)
```

```
# define OPENSSL_EXPORT extern __declspec(dllexport)
# define OPENSSL_IMPORT extern __declspec(dllimport)
# define OPENSSL_GLOBAL
#else
# define OPENSSL_EXPORT extern
# define OPENSSL_IMPORT extern
# define OPENSSL_GLOBAL
#endif
#define OPENSSL_EXTERN OPENSSL_IMPORT

/* Macros to allow global variables to be reached through function calls when
   required (if a shared library version requires it, for example.
   The way it's done allows definitions like this:

        // in foobar.c
        OPENSSL_IMPLEMENT_GLOBAL(int,foobar,0)
        // in foobar.h
        OPENSSL_DECLARE_GLOBAL(int,foobar);
        #define foobar OPENSSL_GLOBAL_REF(foobar)
*/
#ifdef OPENSSL_EXPORT_VAR_AS_FUNCTION
# define OPENSSL_IMPLEMENT_GLOBAL(type,name,value)                      \
        type *_shadow_##name(void)                                      \
        { static type _hide_##name=value; return &_hide_##name; }
# define OPENSSL_DECLARE_GLOBAL(type,name) type *_shadow_##name(void)
# define OPENSSL_GLOBAL_REF(name) (*(_shadow_##name()))
#else
# define OPENSSL_IMPLEMENT_GLOBAL(type,name,value) OPENSSL_GLOBAL type _shadow_#
# define OPENSSL_DECLARE_GLOBAL(type,name) OPENSSL_EXPORT type _shadow_##name
# define OPENSSL_GLOBAL_REF(name) _shadow_##name
#endif

#if defined(OPENSSL_SYS_MACINTOSH_CLASSIC) && macintosh==1 && !defined(MAC_OS_GU
#  define ossl_ssize_t long
#endif

#ifdef OPENSSL_SYS_MSDOS
#  define ossl_ssize_t long
#endif

#if defined(NeXT) || defined(OPENSSL_SYS_NEWS4) || defined(OPENSSL_SYS_SUNOS)
#  define ssize_t int
#endif

#if defined(__ultrix) && !defined(ssize_t)
#  define ossl_ssize_t int
#endif

#ifndef ossl_ssize_t
#  define ossl_ssize_t ssize_t
#endif

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
     540 Wed Aug 13 19:51:43 2014
new/usr/src/lib/openssl/include/openssl/ebcdic.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ebcdic.h */

#ifndef HEADER_EBCDIC_H
#define HEADER_EBCDIC_H

#include <sys/types.h>

/* Avoid name clashes with other applications */
#define os_toascii   _openssl_os_toascii
#define os_toebcdic  _openssl_os_toebcdic
#define ebcdic2ascii _openssl_ebcdic2ascii
#define ascii2ebcdic _openssl_ascii2ebcdic

extern const unsigned char os_toascii[256];
extern const unsigned char os_toebcdic[256];
void *ebcdic2ascii(void *dest, const void *srce, size_t count);
void *ascii2ebcdic(void *dest, const void *srce, size_t count);

#endif
#endif /* ! codereview */
```

```
**********************************************************
   40611 Wed Aug 13 19:51:43 2014
new/usr/src/lib/openssl/include/openssl/engine.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* openssl/engine.h */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 1999-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
```



```
 */

#ifndef HEADER_ENGINE_H
#define HEADER_ENGINE_H

#include <openssl/opensslconf.h>

#ifdef OPENSSL_NO_ENGINE
#error ENGINE is disabled.
#endif

#ifndef OPENSSL_NO_DEPRECATED
#include <openssl/bn.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#ifndef OPENSSL_NO_ECDH
#include <openssl/ecdh.h>
#endif
#ifndef OPENSSL_NO_ECDSA
#include <openssl/ecdsa.h>
#endif
#include <openssl/rand.h>
#include <openssl/ui.h>
#include <openssl/err.h>
#endif

#include <openssl/ossl_typ.h>
#include <openssl/symhacks.h>

#include <openssl/x509.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* These flags are used to control combinations of algorithm (methods)
 * by bitwise "OR"ing. */
#define ENGINE_METHOD_RSA               (unsigned int)0x0001
#define ENGINE_METHOD_DSA               (unsigned int)0x0002
#define ENGINE_METHOD_DH                (unsigned int)0x0004
#define ENGINE_METHOD_RAND              (unsigned int)0x0008
#define ENGINE_METHOD_ECDH              (unsigned int)0x0010
#define ENGINE_METHOD_ECDSA             (unsigned int)0x0020
#define ENGINE_METHOD_CIPHERS           (unsigned int)0x0040
#define ENGINE_METHOD_DIGESTS           (unsigned int)0x0080
#define ENGINE_METHOD_STORE             (unsigned int)0x0100
#define ENGINE_METHOD_PKEY_METHS        (unsigned int)0x0200
#define ENGINE_METHOD_PKEY_ASN1_METHS   (unsigned int)0x0400
/* Obvious all-or-nothing cases. */
#define ENGINE_METHOD_ALL               (unsigned int)0xFFFF
#define ENGINE_METHOD_NONE              (unsigned int)0x0000

/* This(ese) flag(s) controls behaviour of the ENGINE_TABLE mechanism used
 * internally to control registration of ENGINE implementations, and can be set
 * by ENGINE_set_table_flags(). The "NOINIT" flag prevents attempts to
 * initialise registered ENGINEs if they are not already initialised. */
#define ENGINE_TABLE_FLAG_NOINIT        (unsigned int)0x0001

/* ENGINE flags that can be set by ENGINE_set_flags(). */
```

```
/* #define ENGINE_FLAGS_MALLOCED        0x0001 */ /* Not used */

/* This flag is for ENGINEs that wish to handle the various 'CMD'-related
 * control commands on their own. Without this flag, ENGINE_ctrl() handles these
 * control commands on behalf of the ENGINE using their "cmd_defns" data. */
#define ENGINE_FLAGS_MANUAL_CMD_CTRL    (int)0x0002

/* This flag is for ENGINEs who return new duplicate structures when found via
 * "ENGINE_by_id()". When an ENGINE must store state (eg. if ENGINE_ctrl()
 * commands are called in sequence as part of some stateful process like
 * key-generation setup and execution), it can set this flag - then each attempt
 * to obtain the ENGINE will result in it being copied into a new structure.
 * Normally, ENGINEs don't declare this flag so ENGINE_by_id() just increments
 * the existing ENGINE's structural reference count. */
#define ENGINE_FLAGS_BY_ID_COPY         (int)0x0004

/* This flag if for an ENGINE that does not want its methods registered as
 * part of ENGINE_register_all_complete() for example if the methods are
 * not usable as default methods.
 */

#define ENGINE_FLAGS_NO_REGISTER_ALL    (int)0x0008

/* ENGINEs can support their own command types, and these flags are used in
 * ENGINE_CTRL_GET_CMD_FLAGS to indicate to the caller what kind of input each
 * command expects. Currently only numeric and string input is supported. If a
 * control command supports none of the _NUMERIC, _STRING, or _NO_INPUT options,
 * then it is regarded as an "internal" control command - and not for use in
 * config setting situations. As such, they're not available to the
 * ENGINE_ctrl_cmd_string() function, only raw ENGINE_ctrl() access. Changes to
 * this list of 'command types' should be reflected carefully in
 * ENGINE_cmd_is_executable() and ENGINE_ctrl_cmd_string(). */

/* accepts a 'long' input value (3rd parameter to ENGINE_ctrl) */
#define ENGINE_CMD_FLAG_NUMERIC         (unsigned int)0x0001
/* accepts string input (cast from 'void*' to 'const char *', 4th parameter to
 * ENGINE_ctrl) */
#define ENGINE_CMD_FLAG_STRING          (unsigned int)0x0002
/* Indicates that the control command takes *no* input. Ie. the control command
 * is unparameterised. */
#define ENGINE_CMD_FLAG_NO_INPUT        (unsigned int)0x0004
/* Indicates that the control command is internal. This control command won't
 * be shown in any output, and is only usable through the ENGINE_ctrl_cmd()
 * function. */
#define ENGINE_CMD_FLAG_INTERNAL        (unsigned int)0x0008

/* NB: These 3 control commands are deprecated and should not be used. ENGINEs
 * relying on these commands should compile conditional support for
 * compatibility (eg. if these symbols are defined) but should also migrate the
 * same functionality to their own ENGINE-specific control functions that can be
 * "discovered" by calling applications. The fact these control commands
 * wouldn't be "executable" (ie. usable by text-based config) doesn't change the
 * fact that application code can find and use them without requiring per-ENGINE
 * hacking. */

/* These flags are used to tell the ctrl function what should be done.
 * All command numbers are shared between all engines, even if some don't
 * make sense to some engines.  In such a case, they do nothing but return
 * the error ENGINE_R_CTRL_COMMAND_NOT_IMPLEMENTED. */
#define ENGINE_CTRL_SET_LOGSTREAM               1
#define ENGINE_CTRL_SET_PASSWORD_CALLBACK       2
#define ENGINE_CTRL_HUP                         3 /* Close and reinitialise any
                                                     handles/connections etc. */
#define ENGINE_CTRL_SET_USER_INTERFACE          4 /* Alternative to callback */
#define ENGINE_CTRL_SET_CALLBACK_DATA           5 /* User-specific data, used
                                                     when calling the password
```



```
                                                     callback and the user
                                                     interface */
#define ENGINE_CTRL_LOAD_CONFIGURATION          6 /* Load a configuration, given
                                                     a string that represents a
                                                     file name or so */
#define ENGINE_CTRL_LOAD_SECTION                7 /* Load data from a given
                                                     section in the already load
                                                     configuration */

/* These control commands allow an application to deal with an arbitrary engine
 * in a dynamic way. Warn: Negative return values indicate errors FOR THESE
 * COMMANDS because zero is used to indicate 'end-of-list'. Other commands,
 * including ENGINE-specific command types, return zero for an error.
 *
 * An ENGINE can choose to implement these ctrl functions, and can internally
 * manage things however it chooses - it does so by setting the
 * ENGINE_FLAGS_MANUAL_CMD_CTRL flag (using ENGINE_set_flags()). Otherwise the
 * ENGINE_ctrl() code handles this on the ENGINE's behalf using the cmd_defns
 * data (set using ENGINE_set_cmd_defns()). This means an ENGINE's ctrl()
 * handler need only implement its own commands - the above "meta" commands will
 * be taken care of. */

/* Returns non-zero if the supplied ENGINE has a ctrl() handler. If "not", then
 * all the remaining control commands will return failure, so it is worth
 * checking this first if the caller is trying to "discover" the engine's
 * capabilities and doesn't want errors generated unnecessarily. */
#define ENGINE_CTRL_HAS_CTRL_FUNCTION           10
/* Returns a positive command number for the first command supported by the
 * engine. Returns zero if no ctrl commands are supported. */
#define ENGINE_CTRL_GET_FIRST_CMD_TYPE          11
/* The 'long' argument specifies a command implemented by the engine, and the
 * return value is the next command supported, or zero if there are no more. */
#define ENGINE_CTRL_GET_NEXT_CMD_TYPE           12
/* The 'void*' argument is a command name (cast from 'const char *'), and the
 * return value is the command that corresponds to it. */
#define ENGINE_CTRL_GET_CMD_FROM_NAME           13
/* The next two allow a command to be converted into its corresponding string
 * form. In each case, the 'long' argument supplies the command. In the NAME_LEN
 * case, the return value is the length of the command name (not counting a
 * trailing EOL). In the NAME case, the 'void*' argument must be a string buffer
 * large enough, and it will be populated with the name of the command (WITH a
 * trailing EOL). */
#define ENGINE_CTRL_GET_NAME_LEN_FROM_CMD       14
#define ENGINE_CTRL_GET_NAME_FROM_CMD           15
/* The next two are similar but give a "short description" of a command. */
#define ENGINE_CTRL_GET_DESC_LEN_FROM_CMD       16
#define ENGINE_CTRL_GET_DESC_FROM_CMD           17
/* With this command, the return value is the OR'd combination of
 * ENGINE_CMD_FLAG_*** values that indicate what kind of input a given
 * engine-specific ctrl command expects. */
#define ENGINE_CTRL_GET_CMD_FLAGS               18

/* ENGINE implementations should start the numbering of their own control
 * commands from this value. (ie. ENGINE_CMD_BASE, ENGINE_CMD_BASE + 1, etc). */
#define ENGINE_CMD_BASE                         200

/* NB: These 2 nCipher "chil" control commands are deprecated, and their
 * functionality is now available through ENGINE-specific control commands
 * (exposed through the above-mentioned 'CMD'-handling). Code using these 2
 * commands should be migrated to the more general command handling before these
 * are removed. */

/* Flags specific to the nCipher "chil" engine */
#define ENGINE_CTRL_CHIL_SET_FORKCHECK          100
        /* Depending on the value of the (long)i argument, this sets or
         * unsets the SimpleForkCheck flag in the CHIL API to enable or
```

```
260          * disable checking and workarounds for applications that fork().
261          */
262 #define ENGINE_CTRL_CHIL_NO_LOCKING            101
263         /* This prevents the initialisation function from providing mutex
264          * callbacks to the nCipher library. */

266 /* If an ENGINE supports its own specific control commands and wishes the
267  * framework to handle the above 'ENGINE_CMD_***'-manipulation commands on its
268  * behalf, it should supply a null-terminated array of ENGINE_CMD_DEFN entries
269  * to ENGINE_set_cmd_defns(). It should also implement a ctrl() handler that
270  * supports the stated commands (ie. the "cmd_num" entries as described by the
271  * array). NB: The array must be ordered in increasing order of cmd_num.
272  * "null-terminated" means that the last ENGINE_CMD_DEFN element has cmd_num set
273  * to zero and/or cmd_name set to NULL. */
274 typedef struct ENGINE_CMD_DEFN_st
275         {
276         unsigned int cmd_num; /* The command number */
277         const char *cmd_name; /* The command name itself */
278         const char *cmd_desc; /* A short description of the command */
279         unsigned int cmd_flags; /* The input the command expects */
280         } ENGINE_CMD_DEFN;

282 /* Generic function pointer */
283 typedef int (*ENGINE_GEN_FUNC_PTR)(void);
284 /* Generic function pointer taking no arguments */
285 typedef int (*ENGINE_GEN_INT_FUNC_PTR)(ENGINE *);
286 /* Specific control function pointer */
287 typedef int (*ENGINE_CTRL_FUNC_PTR)(ENGINE *, int, long, void *, void (*f)(void)
288 /* Generic load_key function pointer */
289 typedef EVP_PKEY * (*ENGINE_LOAD_KEY_PTR)(ENGINE *, const char *,
290         UI_METHOD *ui_method, void *callback_data);
291 typedef int (*ENGINE_SSL_CLIENT_CERT_PTR)(ENGINE *, SSL *ssl,
292         STACK_OF(X509_NAME) *ca_dn, X509 **pcert, EVP_PKEY **pkey,
293         STACK_OF(X509) **pother, UI_METHOD *ui_method, void *callback_data);
294 /* These callback types are for an ENGINE's handler for cipher and digest logic.
295  * These handlers have these prototypes;
296  *   int foo(ENGINE *e, const EVP_CIPHER **cipher, const int **nids, int nid);
297  *   int foo(ENGINE *e, const EVP_MD **digest, const int **nids, int nid);
298  * Looking at how to implement these handlers in the case of cipher support, if
299  * the framework wants the EVP_CIPHER for 'nid', it will call;
300  *   foo(e, &p_evp_cipher, NULL, nid);    (return zero for failure)
301  * If the framework wants a list of supported 'nid's, it will call;
302  *   foo(e, NULL, &p_nids, 0); (returns number of 'nids' or -1 for error)
303  */
304 /* Returns to a pointer to the array of supported cipher 'nid's. If the second
305  * parameter is non-NULL it is set to the size of the returned array. */
306 typedef int (*ENGINE_CIPHERS_PTR)(ENGINE *, const EVP_CIPHER **, const int **, i
307 typedef int (*ENGINE_DIGESTS_PTR)(ENGINE *, const EVP_MD **, const int **, int);
308 typedef int (*ENGINE_PKEY_METHS_PTR)(ENGINE *, EVP_PKEY_METHOD **, const int **,
309 typedef int (*ENGINE_PKEY_ASN1_METHS_PTR)(ENGINE *, EVP_PKEY_ASN1_METHOD **, con
310 /* STRUCTURE functions ... all of these functions deal with pointers to ENGINE
311  * structures where the pointers have a "structural reference". This means that
312  * their reference is to allowed access to the structure but it does not imply
313  * that the structure is functional. To simply increment or decrement the
314  * structural reference count, use ENGINE_by_id and ENGINE_free. NB: This is not
315  * required when iterating using ENGINE_get_next as it will automatically
316  * decrement the structural reference count of the "current" ENGINE and
317  * increment the structural reference count of the ENGINE it returns (unless it
318  * is NULL). */

320 /* Get the first/last "ENGINE" type available. */
321 ENGINE *ENGINE_get_first(void);
322 ENGINE *ENGINE_get_last(void);
323 /* Iterate to the next/previous "ENGINE" type (NULL = end of the list). */
324 ENGINE *ENGINE_get_next(ENGINE *e);
325 ENGINE *ENGINE_get_prev(ENGINE *e);
```



```
326 /* Add another "ENGINE" type into the array. */
327 int ENGINE_add(ENGINE *e);
328 /* Remove an existing "ENGINE" type from the array. */
329 int ENGINE_remove(ENGINE *e);
330 /* Retrieve an engine from the list by its unique "id" value. */
331 ENGINE *ENGINE_by_id(const char *id);
332 /* Add all the built-in engines. */
333 void ENGINE_load_openssl(void);
334 void ENGINE_load_dynamic(void);
335 #ifndef OPENSSL_NO_STATIC_ENGINE
336 void ENGINE_load_4758cca(void);
337 void ENGINE_load_aep(void);
338 void ENGINE_load_atalla(void);
339 void ENGINE_load_chil(void);
340 void ENGINE_load_cswift(void);
341 void ENGINE_load_nuron(void);
342 void ENGINE_load_sureware(void);
343 void ENGINE_load_ubsec(void);
344 void ENGINE_load_padlock(void);
345 void ENGINE_load_capi(void);
346 #ifndef OPENSSL_NO_GMP
347 void ENGINE_load_gmp(void);
348 #endif
349 #ifndef OPENSSL_NO_GOST
350 void ENGINE_load_gost(void);
351 #endif
352 #endif
353 void ENGINE_load_cryptodev(void);
354 void ENGINE_load_pk11(void);
355 void ENGINE_load_rsax(void);
356 void ENGINE_load_rdrand(void);
357 void ENGINE_load_builtin_engines(void);

359 /* Get and set global flags (ENGINE_TABLE_FLAG_***) for the implementation
360  * "registry" handling. */
361 unsigned int ENGINE_get_table_flags(void);
362 void ENGINE_set_table_flags(unsigned int flags);

364 /* Manage registration of ENGINEs per "table". For each type, there are 3
365  * functions;
366  *   ENGINE_register_***(e) - registers the implementation from 'e' (if it has o
367  *   ENGINE_unregister_***(e) - unregister the implementation from 'e'
368  *   ENGINE_register_all_***() - call ENGINE_register_***() for each 'e' in the
369  * Cleanup is automatically registered from each table when required, so
370  * ENGINE_cleanup() will reverse any "register" operations. */

372 int ENGINE_register_RSA(ENGINE *e);
373 void ENGINE_unregister_RSA(ENGINE *e);
374 void ENGINE_register_all_RSA(void);

376 int ENGINE_register_DSA(ENGINE *e);
377 void ENGINE_unregister_DSA(ENGINE *e);
378 void ENGINE_register_all_DSA(void);

380 int ENGINE_register_ECDH(ENGINE *e);
381 void ENGINE_unregister_ECDH(ENGINE *e);
382 void ENGINE_register_all_ECDH(void);

384 int ENGINE_register_ECDSA(ENGINE *e);
385 void ENGINE_unregister_ECDSA(ENGINE *e);
386 void ENGINE_register_all_ECDSA(void);

388 int ENGINE_register_DH(ENGINE *e);
389 void ENGINE_unregister_DH(ENGINE *e);
390 void ENGINE_register_all_DH(void);
```

```
int ENGINE_register_RAND(ENGINE *e);
void ENGINE_unregister_RAND(ENGINE *e);
void ENGINE_register_all_RAND(void);

int ENGINE_register_STORE(ENGINE *e);
void ENGINE_unregister_STORE(ENGINE *e);
void ENGINE_register_all_STORE(void);

int ENGINE_register_ciphers(ENGINE *e);
void ENGINE_unregister_ciphers(ENGINE *e);
void ENGINE_register_all_ciphers(void);

int ENGINE_register_digests(ENGINE *e);
void ENGINE_unregister_digests(ENGINE *e);
void ENGINE_register_all_digests(void);

int ENGINE_register_pkey_meths(ENGINE *e);
void ENGINE_unregister_pkey_meths(ENGINE *e);
void ENGINE_register_all_pkey_meths(void);

int ENGINE_register_pkey_asn1_meths(ENGINE *e);
void ENGINE_unregister_pkey_asn1_meths(ENGINE *e);
void ENGINE_register_all_pkey_asn1_meths(void);

/* These functions register all support from the above categories. Note, use of
 * these functions can result in static linkage of code your application may not
 * need. If you only need a subset of functionality, consider using more
 * selective initialisation. */
int ENGINE_register_complete(ENGINE *e);
int ENGINE_register_all_complete(void);

/* Send parametrised control commands to the engine. The possibilities to send
 * down an integer, a pointer to data or a function pointer are provided. Any of
 * the parameters may or may not be NULL, depending on the command number. In
 * actuality, this function only requires a structural (rather than functional)
 * reference to an engine, but many control commands may require the engine be
 * functional. The caller should be aware of trying commands that require an
 * operational ENGINE, and only use functional references in such situations. */
int ENGINE_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)(void));

/* This function tests if an ENGINE-specific command is usable as a "setting".
 * Eg. in an application's config file that gets processed through
 * ENGINE_ctrl_cmd_string(). If this returns zero, it is not available to
 * ENGINE_ctrl_cmd_string(), only ENGINE_ctrl(). */
int ENGINE_cmd_is_executable(ENGINE *e, int cmd);

/* This function works like ENGINE_ctrl() with the exception of taking a
 * command name instead of a command number, and can handle optional commands.
 * See the comment on ENGINE_ctrl_cmd_string() for an explanation on how to
 * use the cmd_name and cmd_optional. */
int ENGINE_ctrl_cmd(ENGINE *e, const char *cmd_name,
        long i, void *p, void (*f)(void), int cmd_optional);

/* This function passes a command-name and argument to an ENGINE. The cmd_name
 * is converted to a command number and the control command is called using
 * 'arg' as an argument (unless the ENGINE doesn't support such a command, in
 * which case no control command is called). The command is checked for input
 * flags, and if necessary the argument will be converted to a numeric value. If
 * cmd_optional is non-zero, then if the ENGINE doesn't support the given
 * cmd_name the return value will be success anyway. This function is intended
 * for applications to use so that users (or config files) can supply
 * engine-specific config data to the ENGINE at run-time to control behaviour of
 * specific engines. As such, it shouldn't be used for calling ENGINE_ctrl()
 * functions that return data, deal with binary data, or that are otherwise
 * supposed to be used directly through ENGINE_ctrl() in application code. Any
 * "return" data from an ENGINE_ctrl() operation in this function will be lost -
```



```
 * the return value is interpreted as failure if the return value is zero,
 * success otherwise, and this function returns a boolean value as a result. In
 * other words, vendors of 'ENGINE'-enabled devices should write ENGINE
 * implementations with parameterisations that work in this scheme, so that
 * compliant ENGINE-based applications can work consistently with the same
 * configuration for the same ENGINE-enabled devices, across applications. */
int ENGINE_ctrl_cmd_string(ENGINE *e, const char *cmd_name, const char *arg,
                           int cmd_optional);

/* These functions are useful for manufacturing new ENGINE structures. They
 * don't address reference counting at all - one uses them to populate an ENGINE
 * structure with personalised implementations of things prior to using it
 * directly or adding it to the builtin ENGINE list in OpenSSL. These are also
 * here so that the ENGINE structure doesn't have to be exposed and break binary
 * compatibility! */
ENGINE *ENGINE_new(void);
int ENGINE_free(ENGINE *e);
int ENGINE_up_ref(ENGINE *e);
int ENGINE_set_id(ENGINE *e, const char *id);
int ENGINE_set_name(ENGINE *e, const char *name);
int ENGINE_set_RSA(ENGINE *e, const RSA_METHOD *rsa_meth);
int ENGINE_set_DSA(ENGINE *e, const DSA_METHOD *dsa_meth);
int ENGINE_set_ECDH(ENGINE *e, const ECDH_METHOD *ecdh_meth);
int ENGINE_set_ECDSA(ENGINE *e, const ECDSA_METHOD *ecdsa_meth);
int ENGINE_set_DH(ENGINE *e, const DH_METHOD *dh_meth);
int ENGINE_set_RAND(ENGINE *e, const RAND_METHOD *rand_meth);
int ENGINE_set_STORE(ENGINE *e, const STORE_METHOD *store_meth);
int ENGINE_set_destroy_function(ENGINE *e, ENGINE_GEN_INT_FUNC_PTR destroy_f);
int ENGINE_set_init_function(ENGINE *e, ENGINE_GEN_INT_FUNC_PTR init_f);
int ENGINE_set_finish_function(ENGINE *e, ENGINE_GEN_INT_FUNC_PTR finish_f);
int ENGINE_set_ctrl_function(ENGINE *e, ENGINE_CTRL_FUNC_PTR ctrl_f);
int ENGINE_set_load_privkey_function(ENGINE *e, ENGINE_LOAD_KEY_PTR loadpriv_f);
int ENGINE_set_load_pubkey_function(ENGINE *e, ENGINE_LOAD_KEY_PTR loadpub_f);
int ENGINE_set_load_ssl_client_cert_function(ENGINE *e,
                                 ENGINE_SSL_CLIENT_CERT_PTR loadssl_f);
int ENGINE_set_ciphers(ENGINE *e, ENGINE_CIPHERS_PTR f);
int ENGINE_set_digests(ENGINE *e, ENGINE_DIGESTS_PTR f);
int ENGINE_set_pkey_meths(ENGINE *e, ENGINE_PKEY_METHS_PTR f);
int ENGINE_set_pkey_asn1_meths(ENGINE *e, ENGINE_PKEY_ASN1_METHS_PTR f);
int ENGINE_set_flags(ENGINE *e, int flags);
int ENGINE_set_cmd_defns(ENGINE *e, const ENGINE_CMD_DEFN *defns);
/* These functions allow control over any per-structure ENGINE data. */
int ENGINE_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
                CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
int ENGINE_set_ex_data(ENGINE *e, int idx, void *arg);
void *ENGINE_get_ex_data(const ENGINE *e, int idx);

/* This function cleans up anything that needs it. Eg. the ENGINE_add() function
 * automatically ensures the list cleanup function is registered to be called
 * from ENGINE_cleanup(). Similarly, all ENGINE_register_*** functions ensure
 * ENGINE_cleanup() will clean up after them. */
void ENGINE_cleanup(void);

/* These return values from within the ENGINE structure. These can be useful
 * with functional references as well as structural references - it depends
 * which you obtained. Using the result for functional purposes if you only
 * obtained a structural reference may be problematic! */
const char *ENGINE_get_id(const ENGINE *e);
const char *ENGINE_get_name(const ENGINE *e);
const RSA_METHOD *ENGINE_get_RSA(const ENGINE *e);
const DSA_METHOD *ENGINE_get_DSA(const ENGINE *e);
const ECDH_METHOD *ENGINE_get_ECDH(const ENGINE *e);
const ECDSA_METHOD *ENGINE_get_ECDSA(const ENGINE *e);
const DH_METHOD *ENGINE_get_DH(const ENGINE *e);
const RAND_METHOD *ENGINE_get_RAND(const ENGINE *e);
const STORE_METHOD *ENGINE_get_STORE(const ENGINE *e);
```

```
524 ENGINE_GEN_INT_FUNC_PTR ENGINE_get_destroy_function(const ENGINE *e);
525 ENGINE_GEN_INT_FUNC_PTR ENGINE_get_init_function(const ENGINE *e);
526 ENGINE_GEN_INT_FUNC_PTR ENGINE_get_finish_function(const ENGINE *e);
527 ENGINE_CTRL_FUNC_PTR ENGINE_get_ctrl_function(const ENGINE *e);
528 ENGINE_LOAD_KEY_PTR ENGINE_get_load_privkey_function(const ENGINE *e);
529 ENGINE_LOAD_KEY_PTR ENGINE_get_load_pubkey_function(const ENGINE *e);
530 ENGINE_SSL_CLIENT_CERT_PTR ENGINE_get_ssl_client_cert_function(const ENGINE *e);
531 ENGINE_CIPHERS_PTR ENGINE_get_ciphers(const ENGINE *e);
532 ENGINE_DIGESTS_PTR ENGINE_get_digests(const ENGINE *e);
533 ENGINE_PKEY_METHS_PTR ENGINE_get_pkey_meths(const ENGINE *e);
534 ENGINE_PKEY_ASN1_METHS_PTR ENGINE_get_pkey_asn1_meths(const ENGINE *e);
535 const EVP_CIPHER *ENGINE_get_cipher(ENGINE *e, int nid);
536 const EVP_MD *ENGINE_get_digest(ENGINE *e, int nid);
537 const EVP_PKEY_METHOD *ENGINE_get_pkey_meth(ENGINE *e, int nid);
538 const EVP_PKEY_ASN1_METHOD *ENGINE_get_pkey_asn1_meth(ENGINE *e, int nid);
539 const EVP_PKEY_ASN1_METHOD *ENGINE_get_pkey_asn1_meth_str(ENGINE *e,
540                                         const char *str, int len);
541 const EVP_PKEY_ASN1_METHOD *ENGINE_pkey_asn1_find_str(ENGINE **pe,
542                                         const char *str, int len);
543 const ENGINE_CMD_DEFN *ENGINE_get_cmd_defns(const ENGINE *e);
544 int ENGINE_get_flags(const ENGINE *e);

546 /* FUNCTIONAL functions. These functions deal with ENGINE structures
547  * that have (or will) be initialised for use. Broadly speaking, the
548  * structural functions are useful for iterating the list of available
549  * engine types, creating new engine types, and other "list" operations.
550  * These functions actually deal with ENGINEs that are to be used. As
551  * such these functions can fail (if applicable) when particular
552  * engines are unavailable - eg. if a hardware accelerator is not
553  * attached or not functioning correctly. Each ENGINE has 2 reference
554  * counts; structural and functional. Every time a functional reference
555  * is obtained or released, a corresponding structural reference is
556  * automatically obtained or released too. */

558 /* Initialise a engine type for use (or up its reference count if it's
559  * already in use). This will fail if the engine is not currently
560  * operational and cannot initialise. */
561 int ENGINE_init(ENGINE *e);
562 /* Free a functional reference to a engine type. This does not require
563  * a corresponding call to ENGINE_free as it also releases a structural
564  * reference. */
565 int ENGINE_finish(ENGINE *e);

567 /* The following functions handle keys that are stored in some secondary
568  * location, handled by the engine.  The storage may be on a card or
569  * whatever. */
570 EVP_PKEY *ENGINE_load_private_key(ENGINE *e, const char *key_id,
571         UI_METHOD *ui_method, void *callback_data);
572 EVP_PKEY *ENGINE_load_public_key(ENGINE *e, const char *key_id,
573         UI_METHOD *ui_method, void *callback_data);
574 int ENGINE_load_ssl_client_cert(ENGINE *e, SSL *s,
575         STACK_OF(X509_NAME) *ca_dn, X509 **pcert, EVP_PKEY **ppkey,
576         STACK_OF(X509) **pother,
577         UI_METHOD *ui_method, void *callback_data);

579 /* This returns a pointer for the current ENGINE structure that
580  * is (by default) performing any RSA operations. The value returned
581  * is an incremented reference, so it should be free'd (ENGINE_finish)
582  * before it is discarded. */
583 ENGINE *ENGINE_get_default_RSA(void);
584 /* Same for the other "methods" */
585 ENGINE *ENGINE_get_default_DSA(void);
586 ENGINE *ENGINE_get_default_ECDH(void);
587 ENGINE *ENGINE_get_default_ECDSA(void);
588 ENGINE *ENGINE_get_default_DH(void);
589 ENGINE *ENGINE_get_default_RAND(void);
```



```
590 /* These functions can be used to get a functional reference to perform
591  * ciphering or digesting corresponding to "nid". */
592 ENGINE *ENGINE_get_cipher_engine(int nid);
593 ENGINE *ENGINE_get_digest_engine(int nid);
594 ENGINE *ENGINE_get_pkey_meth_engine(int nid);
595 ENGINE *ENGINE_get_pkey_asn1_meth_engine(int nid);

597 /* This sets a new default ENGINE structure for performing RSA
598  * operations. If the result is non-zero (success) then the ENGINE
599  * structure will have had its reference count up'd so the caller
600  * should still free their own reference 'e'. */
601 int ENGINE_set_default_RSA(ENGINE *e);
602 int ENGINE_set_default_string(ENGINE *e, const char *def_list);
603 /* Same for the other "methods" */
604 int ENGINE_set_default_DSA(ENGINE *e);
605 int ENGINE_set_default_ECDH(ENGINE *e);
606 int ENGINE_set_default_ECDSA(ENGINE *e);
607 int ENGINE_set_default_DH(ENGINE *e);
608 int ENGINE_set_default_RAND(ENGINE *e);
609 int ENGINE_set_default_ciphers(ENGINE *e);
610 int ENGINE_set_default_digests(ENGINE *e);
611 int ENGINE_set_default_pkey_meths(ENGINE *e);
612 int ENGINE_set_default_pkey_asn1_meths(ENGINE *e);

614 /* The combination "set" - the flags are bitwise "OR"d from the
615  * ENGINE_METHOD_*** defines above. As with the "ENGINE_register_complete()"
616  * function, this function can result in unnecessary static linkage. If your
617  * application requires only specific functionality, consider using more
618  * selective functions. */
619 int ENGINE_set_default(ENGINE *e, unsigned int flags);

621 void ENGINE_add_conf_module(void);

623 /* Deprecated functions ... */
624 /* int ENGINE_clear_defaults(void); */

626 /**************************/
627 /* DYNAMIC ENGINE SUPPORT */
628 /**************************/

630 /* Binary/behaviour compatibility levels */
631 #define OSSL_DYNAMIC_VERSION            (unsigned long)0x00020000
632 /* Binary versions older than this are too old for us (whether we're a loader or
633  * a loadee) */
634 #define OSSL_DYNAMIC_OLDEST             (unsigned long)0x00020000

636 /* When compiling an ENGINE entirely as an external shared library, loadable by
637  * the "dynamic" ENGINE, these types are needed. The 'dynamic_fns' structure
638  * type provides the calling application's (or library's) error functionality
639  * and memory management function pointers to the loaded library. These should
640  * be used/set in the loaded library code so that the loading application's
641  * 'state' will be used/changed in all operations. The 'static_state' pointer
642  * allows the loaded library to know if it shares the same static data as the
643  * calling application (or library), and thus whether these callbacks need to be
644  * set or not. */
645 typedef void *(*dyn_MEM_malloc_cb)(size_t);
646 typedef void *(*dyn_MEM_realloc_cb)(void *, size_t);
647 typedef void (*dyn_MEM_free_cb)(void *);
648 typedef struct st_dynamic_MEM_fns {
649         dyn_MEM_malloc_cb                       malloc_cb;
650         dyn_MEM_realloc_cb                      realloc_cb;
651         dyn_MEM_free_cb                         free_cb;
652         } dynamic_MEM_fns;
653 /* FIXME: Perhaps the memory and locking code (crypto.h) should declare and use
654  * these types so we (and any other dependant code) can simplify a bit?? */
655 typedef void (*dyn_lock_locking_cb)(int,int,const char *,int);
```

```
656 typedef int (*dyn_lock_add_lock_cb)(int*,int,int,const char *,int);
657 typedef struct CRYPTO_dynlock_value *(*dyn_dynlock_create_cb)(
658                                         const char *,int);
659 typedef void (*dyn_dynlock_lock_cb)(int,struct CRYPTO_dynlock_value *,
660                                         const char *,int);
661 typedef void (*dyn_dynlock_destroy_cb)(struct CRYPTO_dynlock_value *,
662                                         const char *,int);
663 typedef struct st_dynamic_LOCK_fns {
664         dyn_lock_locking_cb                     lock_locking_cb;
665         dyn_lock_add_lock_cb                    lock_add_lock_cb;
666         dyn_dynlock_create_cb                   dynlock_create_cb;
667         dyn_dynlock_lock_cb                     dynlock_lock_cb;
668         dyn_dynlock_destroy_cb                  dynlock_destroy_cb;
669         } dynamic_LOCK_fns;
670 /* The top-level structure */
671 typedef struct st_dynamic_fns {
672         void                                    *static_state;
673         const ERR_FNS                           *err_fns;
674         const CRYPTO_EX_DATA_IMPL               *ex_data_fns;
675         dynamic_MEM_fns                         mem_fns;
676         dynamic_LOCK_fns                        lock_fns;
677         } dynamic_fns;

679 /* The version checking function should be of this prototype. NB: The
680  * ossl_version value passed in is the OSSL_DYNAMIC_VERSION of the loading code.
681  * If this function returns zero, it indicates a (potential) version
682  * incompatibility and the loaded library doesn't believe it can proceed.
683  * Otherwise, the returned value is the (latest) version supported by the
684  * loading library. The loader may still decide that the loaded code's version
685  * is unsatisfactory and could veto the load. The function is expected to
686  * be implemented with the symbol name "v_check", and a default implementation
687  * can be fully instantiated with IMPLEMENT_DYNAMIC_CHECK_FN(). */
688 typedef unsigned long (*dynamic_v_check_fn)(unsigned long ossl_version);
689 #define IMPLEMENT_DYNAMIC_CHECK_FN() \
690         OPENSSL_EXPORT unsigned long v_check(unsigned long v); \
691         OPENSSL_EXPORT unsigned long v_check(unsigned long v) { \
692                 if(v >= OSSL_DYNAMIC_OLDEST) return OSSL_DYNAMIC_VERSION; \
693                 return 0; }

695 /* This function is passed the ENGINE structure to initialise with its own
696  * function and command settings. It should not adjust the structural or
697  * functional reference counts. If this function returns zero, (a) the load will
698  * be aborted, (b) the previous ENGINE state will be memcpy'd back onto the
699  * structure, and (c) the shared library will be unloaded. So implementations
700  * should do their own internal cleanup in failure circumstances otherwise they
701  * could leak. The 'id' parameter, if non-NULL, represents the ENGINE id that
702  * the loader is looking for. If this is NULL, the shared library can choose to
703  * return failure or to initialise a 'default' ENGINE. If non-NULL, the shared
704  * library must initialise only an ENGINE matching the passed 'id'. The function
705  * is expected to be implemented with the symbol name "bind_engine". A standard
706  * implementation can be instantiated with IMPLEMENT_DYNAMIC_BIND_FN(fn) where
707  * the parameter 'fn' is a callback function that populates the ENGINE structure
708  * and returns an int value (zero for failure). 'fn' should have prototype;
709  *    [static] int fn(ENGINE *e, const char *id); */
710 typedef int (*dynamic_bind_engine)(ENGINE *e, const char *id,
711                                    const dynamic_fns *fns);
712 #define IMPLEMENT_DYNAMIC_BIND_FN(fn) \
713         OPENSSL_EXPORT \
714         int bind_engine(ENGINE *e, const char *id, const dynamic_fns *fns); \
715         OPENSSL_EXPORT \
716         int bind_engine(ENGINE *e, const char *id, const dynamic_fns *fns) { \
717                 if(ENGINE_get_static_state() == fns->static_state) goto skip_cbs; \
718                 if(!CRYPTO_set_mem_functions(fns->mem_fns.malloc_cb, \
719                         fns->mem_fns.realloc_cb, fns->mem_fns.free_cb)) \
720                         return 0; \
721                 CRYPTO_set_locking_callback(fns->lock_fns.lock_locking_cb); \
```



```
722                 CRYPTO_set_add_lock_callback(fns->lock_fns.lock_add_lock_cb); \
723                 CRYPTO_set_dynlock_create_callback(fns->lock_fns.dynlock_create_
724                 CRYPTO_set_dynlock_lock_callback(fns->lock_fns.dynlock_lock_cb); \
725                 CRYPTO_set_dynlock_destroy_callback(fns->lock_fns.dynlock_destro
726                 if(!CRYPTO_set_ex_data_implementation(fns->ex_data_fns)) \
727                         return 0; \
728                 if(!ERR_set_implementation(fns->err_fns)) return 0; \
729         skip_cbs: \
730                 if(!fn(e,id)) return 0; \
731                 return 1; }

733 /* If the loading application (or library) and the loaded ENGINE library share
734  * the same static data (eg. they're both dynamically linked to the same
735  * libcrypto.so) we need a way to avoid trying to set system callbacks - this
736  * would fail, and for the same reason that it's unnecessary to try. If the
737  * loaded ENGINE has (or gets from through the loader) its own copy of the
738  * libcrypto static data, we will need to set the callbacks. The easiest way to
739  * detect this is to have a function that returns a pointer to some static data
740  * and let the loading application and loaded ENGINE compare their respective
741  * values. */
742 void *ENGINE_get_static_state(void);

744 #if defined(__OpenBSD__) || defined(__FreeBSD__) || defined(HAVE_CRYPTODEV)
745 void ENGINE_setup_bsd_cryptodev(void);
746 #endif

748 /* BEGIN ERROR CODES */
749 /* The following lines are auto generated by the script mkerr.pl. Any changes
750  * made after this point may be overwritten when the script is next run.
751  */
752 void ERR_load_ENGINE_strings(void);

754 /* Error codes for the ENGINE functions. */

756 /* Function codes. */
757 #define ENGINE_F_DYNAMIC_CTRL                           180
758 #define ENGINE_F_DYNAMIC_GET_DATA_CTX                   181
759 #define ENGINE_F_DYNAMIC_LOAD                           182
760 #define ENGINE_F_DYNAMIC_SET_DATA_CTX                   183
761 #define ENGINE_F_ENGINE_ADD                             105
762 #define ENGINE_F_ENGINE_BY_ID                           106
763 #define ENGINE_F_ENGINE_CMD_IS_EXECUTABLE               170
764 #define ENGINE_F_ENGINE_CTRL                            142
765 #define ENGINE_F_ENGINE_CTRL_CMD                        178
766 #define ENGINE_F_ENGINE_CTRL_CMD_STRING                 171
767 #define ENGINE_F_ENGINE_FINISH                          107
768 #define ENGINE_F_ENGINE_FREE_UTIL                       108
769 #define ENGINE_F_ENGINE_GET_CIPHER                      185
770 #define ENGINE_F_ENGINE_GET_DEFAULT_TYPE                177
771 #define ENGINE_F_ENGINE_GET_DIGEST                      186
772 #define ENGINE_F_ENGINE_GET_NEXT                        115
773 #define ENGINE_F_ENGINE_GET_PKEY_ASN1_METH              193
774 #define ENGINE_F_ENGINE_GET_PKEY_METH                   192
775 #define ENGINE_F_ENGINE_GET_PREV                        116
776 #define ENGINE_F_ENGINE_INIT                            119
777 #define ENGINE_F_ENGINE_LIST_ADD                        120
778 #define ENGINE_F_ENGINE_LIST_REMOVE                     121
779 #define ENGINE_F_ENGINE_LOAD_PRIVATE_KEY                150
780 #define ENGINE_F_ENGINE_LOAD_PUBLIC_KEY                 151
781 #define ENGINE_F_ENGINE_LOAD_SSL_CLIENT_CERT            194
782 #define ENGINE_F_ENGINE_NEW                             122
783 #define ENGINE_F_ENGINE_REMOVE                          123
784 #define ENGINE_F_ENGINE_SET_DEFAULT_STRING              189
785 #define ENGINE_F_ENGINE_SET_DEFAULT_TYPE                126
786 #define ENGINE_F_ENGINE_SET_ID                          129
787 #define ENGINE_F_ENGINE_SET_NAME                        130
```

```
788 #define ENGINE_F_ENGINE_TABLE_REGISTER                  184
789 #define ENGINE_F_ENGINE_UNLOAD_KEY                      152
790 #define ENGINE_F_ENGINE_UNLOCKED_FINISH                 191
791 #define ENGINE_F_ENGINE_UP_REF                          190
792 #define ENGINE_F_INT_CTRL_HELPER                        172
793 #define ENGINE_F_INT_ENGINE_CONFIGURE                   188
794 #define ENGINE_F_INT_ENGINE_MODULE_INIT                 187
795 #define ENGINE_F_LOG_MESSAGE                            141

797 /* Reason codes. */
798 #define ENGINE_R_ALREADY_LOADED                         100
799 #define ENGINE_R_ARGUMENT_IS_NOT_A_NUMBER               133
800 #define ENGINE_R_CMD_NOT_EXECUTABLE                     134
801 #define ENGINE_R_COMMAND_TAKES_INPUT                    135
802 #define ENGINE_R_COMMAND_TAKES_NO_INPUT                 136
803 #define ENGINE_R_CONFLICTING_ENGINE_ID                  103
804 #define ENGINE_R_CTRL_COMMAND_NOT_IMPLEMENTED           119
805 #define ENGINE_R_DH_NOT_IMPLEMENTED                     139
806 #define ENGINE_R_DSA_NOT_IMPLEMENTED                    140
807 #define ENGINE_R_DSO_FAILURE                            104
808 #define ENGINE_R_DSO_NOT_FOUND                          132
809 #define ENGINE_R_ENGINES_SECTION_ERROR                  148
810 #define ENGINE_R_ENGINE_CONFIGURATION_ERROR             102
811 #define ENGINE_R_ENGINE_IS_NOT_IN_LIST                  105
812 #define ENGINE_R_ENGINE_SECTION_ERROR                   149
813 #define ENGINE_R_FAILED_LOADING_PRIVATE_KEY             128
814 #define ENGINE_R_FAILED_LOADING_PUBLIC_KEY              129
815 #define ENGINE_R_FINISH_FAILED                          106
816 #define ENGINE_R_GET_HANDLE_FAILED                      107
817 #define ENGINE_R_ID_OR_NAME_MISSING                     108
818 #define ENGINE_R_INIT_FAILED                            109
819 #define ENGINE_R_INTERNAL_LIST_ERROR                    110
820 #define ENGINE_R_INVALID_ARGUMENT                       143
821 #define ENGINE_R_INVALID_CMD_NAME                       137
822 #define ENGINE_R_INVALID_CMD_NUMBER                     138
823 #define ENGINE_R_INVALID_INIT_VALUE                     151
824 #define ENGINE_R_INVALID_STRING                         150
825 #define ENGINE_R_NOT_INITIALISED                        117
826 #define ENGINE_R_NOT_LOADED                             112
827 #define ENGINE_R_NO_CONTROL_FUNCTION                    120
828 #define ENGINE_R_NO_INDEX                               144
829 #define ENGINE_R_NO_LOAD_FUNCTION                       125
830 #define ENGINE_R_NO_REFERENCE                           130
831 #define ENGINE_R_NO_SUCH_ENGINE                         116
832 #define ENGINE_R_NO_UNLOAD_FUNCTION                     126
833 #define ENGINE_R_PROVIDE_PARAMETERS                     113
834 #define ENGINE_R_RSA_NOT_IMPLEMENTED                    141
835 #define ENGINE_R_UNIMPLEMENTED_CIPHER                   146
836 #define ENGINE_R_UNIMPLEMENTED_DIGEST                   147
837 #define ENGINE_R_UNIMPLEMENTED_PUBLIC_KEY_METHOD        101
838 #define ENGINE_R_VERSION_INCOMPATIBILITY                145

840 #ifdef  __cplusplus
841 }
842 #endif
843 #endif
844 #endif /* ! codereview */
```

```
**********************************************************
   15855 Wed Aug 13 19:51:44 2014
new/usr/src/lib/openssl/include/openssl/err.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/err/err.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_ERR_H
#define HEADER_ERR_H

#include <openssl/e_os2.h>

#ifndef OPENSSL_NO_FP_API
#include <stdio.h>
#include <stdlib.h>
#endif

#include <openssl/ossl_typ.h>
#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#ifndef OPENSSL_NO_LHASH
#include <openssl/lhash.h>
```

```
#endif

#ifdef  __cplusplus
extern "C" {
#endif

#ifndef OPENSSL_NO_ERR
#define ERR_PUT_error(a,b,c,d,e)        ERR_put_error(a,b,c,d,e)
#else
#define ERR_PUT_error(a,b,c,d,e)        ERR_put_error(a,b,c,NULL,0)
#endif

#include <errno.h>

#define ERR_TXT_MALLOCED        0x01
#define ERR_TXT_STRING          0x02

#define ERR_FLAG_MARK           0x01

#define ERR_NUM_ERRORS  16
typedef struct err_state_st
        {
        CRYPTO_THREADID tid;
        int err_flags[ERR_NUM_ERRORS];
        unsigned long err_buffer[ERR_NUM_ERRORS];
        char *err_data[ERR_NUM_ERRORS];
        int err_data_flags[ERR_NUM_ERRORS];
        const char *err_file[ERR_NUM_ERRORS];
        int err_line[ERR_NUM_ERRORS];
        int top,bottom;
        } ERR_STATE;

/* library */
#define ERR_LIB_NONE            1
#define ERR_LIB_SYS             2
#define ERR_LIB_BN              3
#define ERR_LIB_RSA             4
#define ERR_LIB_DH              5
#define ERR_LIB_EVP             6
#define ERR_LIB_BUF             7
#define ERR_LIB_OBJ             8
#define ERR_LIB_PEM             9
#define ERR_LIB_DSA             10
#define ERR_LIB_X509            11
/* #define ERR_LIB_METH         12 */
#define ERR_LIB_ASN1            13
#define ERR_LIB_CONF            14
#define ERR_LIB_CRYPTO          15
#define ERR_LIB_EC              16
#define ERR_LIB_SSL             20
/* #define ERR_LIB_SSL23        21 */
/* #define ERR_LIB_SSL2         22 */
/* #define ERR_LIB_SSL3         23 */
/* #define ERR_LIB_RSAREF       30 */
/* #define ERR_LIB_PROXY        31 */
#define ERR_LIB_BIO             32
#define ERR_LIB_PKCS7           33
#define ERR_LIB_X509V3          34
#define ERR_LIB_PKCS12          35
#define ERR_LIB_RAND            36
#define ERR_LIB_DSO             37
#define ERR_LIB_ENGINE          38
#define ERR_LIB_OCSP            39
#define ERR_LIB_UI              40
#define ERR_LIB_COMP            41
#define ERR_LIB_ECDSA           42
```



```
#define ERR_LIB_ECDH            43
#define ERR_LIB_STORE           44
#define ERR_LIB_FIPS            45
#define ERR_LIB_CMS             46
#define ERR_LIB_TS              47
#define ERR_LIB_HMAC            48
#define ERR_LIB_JPAKE           49

#define ERR_LIB_USER            128

#define SYSerr(f,r)  ERR_PUT_error(ERR_LIB_SYS,(f),(r),__FILE__,__LINE__)
#define BNerr(f,r)   ERR_PUT_error(ERR_LIB_BN,(f),(r),__FILE__,__LINE__)
#define RSAerr(f,r)  ERR_PUT_error(ERR_LIB_RSA,(f),(r),__FILE__,__LINE__)
#define DHerr(f,r)   ERR_PUT_error(ERR_LIB_DH,(f),(r),__FILE__,__LINE__)
#define EVPerr(f,r)  ERR_PUT_error(ERR_LIB_EVP,(f),(r),__FILE__,__LINE__)
#define BUFerr(f,r)  ERR_PUT_error(ERR_LIB_BUF,(f),(r),__FILE__,__LINE__)
#define OBJerr(f,r)  ERR_PUT_error(ERR_LIB_OBJ,(f),(r),__FILE__,__LINE__)
#define PEMerr(f,r)  ERR_PUT_error(ERR_LIB_PEM,(f),(r),__FILE__,__LINE__)
#define DSAerr(f,r)  ERR_PUT_error(ERR_LIB_DSA,(f),(r),__FILE__,__LINE__)
#define X509err(f,r) ERR_PUT_error(ERR_LIB_X509,(f),(r),__FILE__,__LINE__)
#define ASN1err(f,r) ERR_PUT_error(ERR_LIB_ASN1,(f),(r),__FILE__,__LINE__)
#define CONFerr(f,r) ERR_PUT_error(ERR_LIB_CONF,(f),(r),__FILE__,__LINE__)
#define CRYPTOerr(f,r) ERR_PUT_error(ERR_LIB_CRYPTO,(f),(r),__FILE__,__LINE__)
#define ECerr(f,r)   ERR_PUT_error(ERR_LIB_EC,(f),(r),__FILE__,__LINE__)
#define SSLerr(f,r)  ERR_PUT_error(ERR_LIB_SSL,(f),(r),__FILE__,__LINE__)
#define BIOerr(f,r)  ERR_PUT_error(ERR_LIB_BIO,(f),(r),__FILE__,__LINE__)
#define PKCS7err(f,r) ERR_PUT_error(ERR_LIB_PKCS7,(f),(r),__FILE__,__LINE__)
#define X509V3err(f,r) ERR_PUT_error(ERR_LIB_X509V3,(f),(r),__FILE__,__LINE__)
#define PKCS12err(f,r) ERR_PUT_error(ERR_LIB_PKCS12,(f),(r),__FILE__,__LINE__)
#define RANDerr(f,r) ERR_PUT_error(ERR_LIB_RAND,(f),(r),__FILE__,__LINE__)
#define DSOerr(f,r) ERR_PUT_error(ERR_LIB_DSO,(f),(r),__FILE__,__LINE__)
#define ENGINEerr(f,r) ERR_PUT_error(ERR_LIB_ENGINE,(f),(r),__FILE__,__LINE__)
#define OCSPerr(f,r) ERR_PUT_error(ERR_LIB_OCSP,(f),(r),__FILE__,__LINE__)
#define UIerr(f,r) ERR_PUT_error(ERR_LIB_UI,(f),(r),__FILE__,__LINE__)
#define COMPerr(f,r) ERR_PUT_error(ERR_LIB_COMP,(f),(r),__FILE__,__LINE__)
#define ECDSAerr(f,r)  ERR_PUT_error(ERR_LIB_ECDSA,(f),(r),__FILE__,__LINE__)
#define ECDHerr(f,r)  ERR_PUT_error(ERR_LIB_ECDH,(f),(r),__FILE__,__LINE__)
#define STOREerr(f,r) ERR_PUT_error(ERR_LIB_STORE,(f),(r),__FILE__,__LINE__)
#define FIPSerr(f,r) ERR_PUT_error(ERR_LIB_FIPS,(f),(r),__FILE__,__LINE__)
#define CMSerr(f,r) ERR_PUT_error(ERR_LIB_CMS,(f),(r),__FILE__,__LINE__)
#define TSerr(f,r) ERR_PUT_error(ERR_LIB_TS,(f),(r),__FILE__,__LINE__)
#define HMACerr(f,r) ERR_PUT_error(ERR_LIB_HMAC,(f),(r),__FILE__,__LINE__)
#define JPAKEerr(f,r) ERR_PUT_error(ERR_LIB_JPAKE,(f),(r),__FILE__,__LINE__)

/* Borland C seems too stupid to be able to shift and do longs in
 * the pre-processor :-( */
#define ERR_PACK(l,f,r)                 (((((unsigned long)l)&0xffL)*0x1000000)| \
                                        ((((unsigned long)f)&0xfffL)*0x1000)| \
                                        ((((unsigned long)r)&0xfffL)))
#define ERR_GET_LIB(l)                  (int)((((unsigned long)l)>>24L)&0xffL)
#define ERR_GET_FUNC(l)                 (int)((((unsigned long)l)>>12L)&0xfffL)
#define ERR_GET_REASON(l)               (int)(((l)&0xfffL)
#define ERR_FATAL_ERROR(l)              (int)((l)&ERR_R_FATAL)


/* OS functions */
#define SYS_F_FOPEN             1
#define SYS_F_CONNECT           2
#define SYS_F_GETSERVBYNAME     3
#define SYS_F_SOCKET            4
#define SYS_F_IOCTLSOCKET       5
#define SYS_F_BIND              6
#define SYS_F_LISTEN            7
#define SYS_F_ACCEPT            8
#define SYS_F_WSASTARTUP        9 /* Winsock stuff */
#define SYS_F_OPENDIR           10
```

```
260 #define SYS_F_FREAD             11


263 /* reasons */
264 #define ERR_R_SYS_LIB   ERR_LIB_SYS       /* 2 */
265 #define ERR_R_BN_LIB    ERR_LIB_BN        /* 3 */
266 #define ERR_R_RSA_LIB   ERR_LIB_RSA       /* 4 */
267 #define ERR_R_DH_LIB    ERR_LIB_DH        /* 5 */
268 #define ERR_R_EVP_LIB   ERR_LIB_EVP       /* 6 */
269 #define ERR_R_BUF_LIB   ERR_LIB_BUF       /* 7 */
270 #define ERR_R_OBJ_LIB   ERR_LIB_OBJ       /* 8 */
271 #define ERR_R_PEM_LIB   ERR_LIB_PEM       /* 9 */
272 #define ERR_R_DSA_LIB   ERR_LIB_DSA      /* 10 */
273 #define ERR_R_X509_LIB  ERR_LIB_X509     /* 11 */
274 #define ERR_R_ASN1_LIB  ERR_LIB_ASN1     /* 13 */
275 #define ERR_R_CONF_LIB  ERR_LIB_CONF     /* 14 */
276 #define ERR_R_CRYPTO_LIB ERR_LIB_CRYPTO  /* 15 */
277 #define ERR_R_EC_LIB    ERR_LIB_EC       /* 16 */
278 #define ERR_R_SSL_LIB   ERR_LIB_SSL      /* 20 */
279 #define ERR_R_BIO_LIB   ERR_LIB_BIO      /* 32 */
280 #define ERR_R_PKCS7_LIB ERR_LIB_PKCS7    /* 33 */
281 #define ERR_R_X509V3_LIB ERR_LIB_X509V3  /* 34 */
282 #define ERR_R_PKCS12_LIB ERR_LIB_PKCS12  /* 35 */
283 #define ERR_R_RAND_LIB  ERR_LIB_RAND     /* 36 */
284 #define ERR_R_DSO_LIB   ERR_LIB_DSO      /* 37 */
285 #define ERR_R_ENGINE_LIB ERR_LIB_ENGINE  /* 38 */
286 #define ERR_R_OCSP_LIB  ERR_LIB_OCSP     /* 39 */
287 #define ERR_R_UI_LIB    ERR_LIB_UI       /* 40 */
288 #define ERR_R_COMP_LIB  ERR_LIB_COMP     /* 41 */
289 #define ERR_R_ECDSA_LIB ERR_LIB_ECDSA    /* 42 */
290 #define ERR_R_ECDH_LIB  ERR_LIB_ECDH     /* 43 */
291 #define ERR_R_STORE_LIB ERR_LIB_STORE    /* 44 */
292 #define ERR_R_TS_LIB    ERR_LIB_TS       /* 45 */

294 #define ERR_R_NESTED_ASN1_ERROR                 58
295 #define ERR_R_BAD_ASN1_OBJECT_HEADER            59
296 #define ERR_R_BAD_GET_ASN1_OBJECT_CALL          60
297 #define ERR_R_EXPECTING_AN_ASN1_SEQUENCE        61
298 #define ERR_R_ASN1_LENGTH_MISMATCH              62
299 #define ERR_R_MISSING_ASN1_EOS                  63

301 /* fatal error */
302 #define ERR_R_FATAL                             64
303 #define ERR_R_MALLOC_FAILURE                    (1|ERR_R_FATAL)
304 #define ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED       (2|ERR_R_FATAL)
305 #define ERR_R_PASSED_NULL_PARAMETER             (3|ERR_R_FATAL)
306 #define ERR_R_INTERNAL_ERROR                    (4|ERR_R_FATAL)
307 #define ERR_R_DISABLED                          (5|ERR_R_FATAL)

309 /* 99 is the maximum possible ERR_R_... code, higher values
310  * are reserved for the individual libraries */


313 typedef struct ERR_string_data_st
314         {
315         unsigned long error;
316         const char *string;
317         } ERR_STRING_DATA;

319 void ERR_put_error(int lib, int func,int reason,const char *file,int line);
320 void ERR_set_error_data(char *data,int flags);

322 unsigned long ERR_get_error(void);
323 unsigned long ERR_get_error_line(const char **file,int *line);
324 unsigned long ERR_get_error_line_data(const char **file,int *line,
325                                       const char **data, int *flags);
```



```
326 unsigned long ERR_peek_error(void);
327 unsigned long ERR_peek_error_line(const char **file,int *line);
328 unsigned long ERR_peek_error_line_data(const char **file,int *line,
329                                        const char **data,int *flags);
330 unsigned long ERR_peek_last_error(void);
331 unsigned long ERR_peek_last_error_line(const char **file,int *line);
332 unsigned long ERR_peek_last_error_line_data(const char **file,int *line,
333                                        const char **data,int *flags);
334 void ERR_clear_error(void );
335 char *ERR_error_string(unsigned long e,char *buf);
336 void ERR_error_string_n(unsigned long e, char *buf, size_t len);
337 const char *ERR_lib_error_string(unsigned long e);
338 const char *ERR_func_error_string(unsigned long e);
339 const char *ERR_reason_error_string(unsigned long e);
340 void ERR_print_errors_cb(int (*cb)(const char *str, size_t len, void *u),
341                          void *u);
342 #ifndef OPENSSL_NO_FP_API
343 void ERR_print_errors_fp(FILE *fp);
344 #endif
345 #ifndef OPENSSL_NO_BIO
346 void ERR_print_errors(BIO *bp);
347 #endif
348 void ERR_add_error_data(int num, ...);
349 void ERR_add_error_vdata(int num, va_list args);
350 void ERR_load_strings(int lib,ERR_STRING_DATA str[]);
351 void ERR_unload_strings(int lib,ERR_STRING_DATA str[]);
352 void ERR_load_ERR_strings(void);
353 void ERR_load_crypto_strings(void);
354 void ERR_free_strings(void);

356 void ERR_remove_thread_state(const CRYPTO_THREADID *tid);
357 #ifndef OPENSSL_NO_DEPRECATED
358 void ERR_remove_state(unsigned long pid); /* if zero we look it up */
359 #endif
360 ERR_STATE *ERR_get_state(void);

362 #ifndef OPENSSL_NO_LHASH
363 LHASH_OF(ERR_STRING_DATA) *ERR_get_string_table(void);
364 LHASH_OF(ERR_STATE) *ERR_get_err_state_table(void);
365 void ERR_release_err_state_table(LHASH_OF(ERR_STATE) **hash);
366 #endif

368 int ERR_get_next_error_library(void);

370 int ERR_set_mark(void);
371 int ERR_pop_to_mark(void);

373 /* Already defined in ossl_typ.h */
374 /* typedef struct st_ERR_FNS ERR_FNS; */
375 /* An application can use this function and provide the return value to loaded
376  * modules that should use the application's ERR state/functionality */
377 const ERR_FNS *ERR_get_implementation(void);
378 /* A loaded module should call this function prior to any ERR operations using
379  * the application's "ERR_FNS". */
380 int ERR_set_implementation(const ERR_FNS *fns);

382 #ifdef  __cplusplus
383 }
384 #endif

386 #endif
387 #endif /* ! codereview */
```

```
**********************************************************
   52775 Wed Aug 13 19:51:44 2014
new/usr/src/lib/openssl/include/openssl/evp.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/evp.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_ENVELOPE_H
#define HEADER_ENVELOPE_H
```



```
#ifdef OPENSSL_ALGORITHM_DEFINES
# include <openssl/opensslconf.h>
#else
# define OPENSSL_ALGORITHM_DEFINES
# include <openssl/opensslconf.h>
# undef OPENSSL_ALGORITHM_DEFINES
#endif

#include <openssl/ossl_typ.h>

#include <openssl/symhacks.h>

#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif

/*
#define EVP_RC2_KEY_SIZE                16
#define EVP_RC4_KEY_SIZE                16
#define EVP_BLOWFISH_KEY_SIZE           16
#define EVP_CAST5_KEY_SIZE              16
#define EVP_RC5_32_12_16_KEY_SIZE       16
*/
#define EVP_MAX_MD_SIZE                 64      /* longest known is SHA512 */
#define EVP_MAX_KEY_LENGTH              64
#define EVP_MAX_IV_LENGTH               16
#define EVP_MAX_BLOCK_LENGTH            32

#define PKCS5_SALT_LEN                  8
/* Default PKCS#5 iteration count */
#define PKCS5_DEFAULT_ITER              2048

#include <openssl/objects.h>

#define EVP_PK_RSA      0x0001
#define EVP_PK_DSA      0x0002
#define EVP_PK_DH       0x0004
#define EVP_PK_EC       0x0008
#define EVP_PKT_SIGN    0x0010
#define EVP_PKT_ENC     0x0020
#define EVP_PKT_EXCH    0x0040
#define EVP_PKS_RSA     0x0100
#define EVP_PKS_DSA     0x0200
#define EVP_PKS_EC      0x0400
#define EVP_PKT_EXP     0x1000 /* <= 512 bit key */

#define EVP_PKEY_NONE   NID_undef
#define EVP_PKEY_RSA    NID_rsaEncryption
#define EVP_PKEY_RSA2   NID_rsa
#define EVP_PKEY_DSA    NID_dsa
#define EVP_PKEY_DSA1   NID_dsa_2
#define EVP_PKEY_DSA2   NID_dsaWithSHA
#define EVP_PKEY_DSA3   NID_dsaWithSHA1
#define EVP_PKEY_DSA4   NID_dsaWithSHA1_2
#define EVP_PKEY_DH     NID_dhKeyAgreement
#define EVP_PKEY_EC     NID_X9_62_id_ecPublicKey
#define EVP_PKEY_HMAC   NID_hmac
#define EVP_PKEY_CMAC   NID_cmac

#ifdef __cplusplus
extern "C" {
#endif

/* Type needs to be a bit field
 * Sub-type needs to be for variations on the method, as in, can it do
 * arbitrary encryption.... */
```

```
128 struct evp_pkey_st
129         {
130         int type;
131         int save_type;
132         int references;
133         const EVP_PKEY_ASN1_METHOD *ameth;
134         ENGINE *engine;
135         union   {
136                 char *ptr;
137 #ifndef OPENSSL_NO_RSA
138                 struct rsa_st *rsa;     /* RSA */
139 #endif
140 #ifndef OPENSSL_NO_DSA
141                 struct dsa_st *dsa;     /* DSA */
142 #endif
143 #ifndef OPENSSL_NO_DH
144                 struct dh_st *dh;       /* DH */
145 #endif
146 #ifndef OPENSSL_NO_EC
147                 struct ec_key_st *ec;   /* ECC */
148 #endif
149                 } pkey;
150         int save_parameters;
151         STACK_OF(X509_ATTRIBUTE) *attributes; /* [ 0 ] */
152         } /* EVP_PKEY */;

154 #define EVP_PKEY_MO_SIGN        0x0001
155 #define EVP_PKEY_MO_VERIFY      0x0002
156 #define EVP_PKEY_MO_ENCRYPT     0x0004
157 #define EVP_PKEY_MO_DECRYPT     0x0008

159 #ifndef EVP_MD
160 struct env_md_st
161         {
162         int type;
163         int pkey_type;
164         int md_size;
165         unsigned long flags;
166         int (*init)(EVP_MD_CTX *ctx);
167         int (*update)(EVP_MD_CTX *ctx,const void *data,size_t count);
168         int (*final)(EVP_MD_CTX *ctx,unsigned char *md);
169         int (*copy)(EVP_MD_CTX *to,const EVP_MD_CTX *from);
170         int (*cleanup)(EVP_MD_CTX *ctx);

172         /* FIXME: prototype these some day */
173         int (*sign)(int type, const unsigned char *m, unsigned int m_length,
174                     unsigned char *sigret, unsigned int *siglen, void *key);
175         int (*verify)(int type, const unsigned char *m, unsigned int m_length,
176                       const unsigned char *sigbuf, unsigned int siglen,
177                       void *key);
178         int required_pkey_type[5]; /*EVP_PKEY_xxx */
179         int block_size;
180         int ctx_size; /* how big does the ctx->md_data need to be */
181         /* control function */
182         int (*md_ctrl)(EVP_MD_CTX *ctx, int cmd, int p1, void *p2);
183         } /* EVP_MD */;

185 typedef int evp_sign_method(int type,const unsigned char *m,
186                             unsigned int m_length,unsigned char *sigret,
187                             unsigned int *siglen, void *key);
188 typedef int evp_verify_method(int type,const unsigned char *m,
189                             unsigned int m_length,const unsigned char *sigbuf,
190                             unsigned int siglen, void *key);

192 #define EVP_MD_FLAG_ONESHOT     0x0001 /* digest can only handle a single
193                                         * block */
```



```
195 #define EVP_MD_FLAG_PKEY_DIGEST 0x0002 /* digest is a "clone" digest used
196                                         * which is a copy of an existing
197                                         * one for a specific public key type.
198                                         * EVP_dss1() etc */

200 /* Digest uses EVP_PKEY_METHOD for signing instead of MD specific signing */

202 #define EVP_MD_FLAG_PKEY_METHOD_SIGNATURE       0x0004

204 /* DigestAlgorithmIdentifier flags... */

206 #define EVP_MD_FLAG_DIGALGID_MASK               0x0018

208 /* NULL or absent parameter accepted. Use NULL */

210 #define EVP_MD_FLAG_DIGALGID_NULL               0x0000

212 /* NULL or absent parameter accepted. Use NULL for PKCS#1 otherwise absent */

214 #define EVP_MD_FLAG_DIGALGID_ABSENT             0x0008

216 /* Custom handling via ctrl */

218 #define EVP_MD_FLAG_DIGALGID_CUSTOM             0x0018

220 #define EVP_MD_FLAG_FIPS        0x0400 /* Note if suitable for use in FIPS mode

222 /* Digest ctrls */

224 #define EVP_MD_CTRL_DIGALGID                    0x1
225 #define EVP_MD_CTRL_MICALG                      0x2

227 /* Minimum Algorithm specific ctrl value */

229 #define EVP_MD_CTRL_ALG_CTRL                    0x1000

231 #define EVP_PKEY_NULL_method    NULL,NULL,{0,0,0,0}

233 #ifndef OPENSSL_NO_DSA
234 #define EVP_PKEY_DSA_method     (evp_sign_method *)DSA_sign, \
235                                 (evp_verify_method *)DSA_verify, \
236                                 {EVP_PKEY_DSA,EVP_PKEY_DSA2,EVP_PKEY_DSA3, \
237                                         EVP_PKEY_DSA4,0}
238 #else
239 #define EVP_PKEY_DSA_method     EVP_PKEY_NULL_method
240 #endif

242 #ifndef OPENSSL_NO_ECDSA
243 #define EVP_PKEY_ECDSA_method   (evp_sign_method *)ECDSA_sign, \
244                                 (evp_verify_method *)ECDSA_verify, \
245                                  {EVP_PKEY_EC,0,0,0}
246 #else
247 #define EVP_PKEY_ECDSA_method   EVP_PKEY_NULL_method
248 #endif

250 #ifndef OPENSSL_NO_RSA
251 #define EVP_PKEY_RSA_method     (evp_sign_method *)RSA_sign, \
252                                 (evp_verify_method *)RSA_verify, \
253                                 {EVP_PKEY_RSA,EVP_PKEY_RSA2,0,0}
254 #define EVP_PKEY_RSA_ASN1_OCTET_STRING_method \
255                                 (evp_sign_method *)RSA_sign_ASN1_OCTET_STRING, \
256                                 (evp_verify_method *)RSA_verify_ASN1_OCTET_STRIN
257                                 {EVP_PKEY_RSA,EVP_PKEY_RSA2,0,0}
258 #else
259 #define EVP_PKEY_RSA_method     EVP_PKEY_NULL_method
```

```
#define EVP_PKEY_RSA_ASN1_OCTET_STRING_method EVP_PKEY_NULL_method
#endif

#endif /* !EVP_MD */

struct env_md_ctx_st
        {
        const EVP_MD *digest;
        ENGINE *engine; /* functional reference if 'digest' is ENGINE-provided *
        unsigned long flags;
        void *md_data;
        /* Public key context for sign/verify */
        EVP_PKEY_CTX *pctx;
        /* Update function: usually copied from EVP_MD */
        int (*update)(EVP_MD_CTX *ctx,const void *data,size_t count);
        } /* EVP_MD_CTX */;

/* values for EVP_MD_CTX flags */

#define EVP_MD_CTX_FLAG_ONESHOT         0x0001 /* digest update will be called
                                                * once only */
#define EVP_MD_CTX_FLAG_CLEANED         0x0002 /* context has already been
                                                * cleaned */
#define EVP_MD_CTX_FLAG_REUSE           0x0004 /* Don't free up ctx->md_data
                                                * in EVP_MD_CTX_cleanup */
/* FIPS and pad options are ignored in 1.0.0, definitions are here
 * so we don't accidentally reuse the values for other purposes.
 */

#define EVP_MD_CTX_FLAG_NON_FIPS_ALLOW  0x0008  /* Allow use of non FIPS digest
                                                 * in FIPS mode */

/* The following PAD options are also currently ignored in 1.0.0, digest
 * parameters are handled through EVP_DigestSign*() and EVP_DigestVerify*()
 * instead.
 */
#define EVP_MD_CTX_FLAG_PAD_MASK        0xF0    /* RSA mode to use */
#define EVP_MD_CTX_FLAG_PAD_PKCS1       0x00    /* PKCS#1 v1.5 mode */
#define EVP_MD_CTX_FLAG_PAD_X931        0x10    /* X9.31 mode */
#define EVP_MD_CTX_FLAG_PAD_PSS         0x20    /* PSS mode */

#define EVP_MD_CTX_FLAG_NO_INIT         0x0100 /* Don't initialize md_data */

struct evp_cipher_st
        {
        int nid;
        int block_size;
        int key_len;            /* Default value for variable length ciphers */
        int iv_len;
        unsigned long flags;    /* Various flags */
        int (*init)(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                    const unsigned char *iv, int enc);  /* init key */
        int (*do_cipher)(EVP_CIPHER_CTX *ctx, unsigned char *out,
                         const unsigned char *in, size_t inl);/* encrypt/decrypt
        int (*cleanup)(EVP_CIPHER_CTX *); /* cleanup ctx */
        int ctx_size;           /* how big ctx->cipher_data needs to be */
        int (*set_asn1_parameters)(EVP_CIPHER_CTX *, ASN1_TYPE *); /* Populate a
        int (*get_asn1_parameters)(EVP_CIPHER_CTX *, ASN1_TYPE *); /* Get parame
        int (*ctrl)(EVP_CIPHER_CTX *, int type, int arg, void *ptr); /* Miscella
        void *app_data;         /* Application data */
        } /* EVP_CIPHER */;

/* Values for cipher flags */

/* Modes for ciphers */
```



```
#define         EVP_CIPH_STREAM_CIPHER          0x0
#define         EVP_CIPH_ECB_MODE               0x1
#define         EVP_CIPH_CBC_MODE               0x2
#define         EVP_CIPH_CFB_MODE               0x3
#define         EVP_CIPH_OFB_MODE               0x4
#define         EVP_CIPH_CTR_MODE               0x5
#define         EVP_CIPH_GCM_MODE               0x6
#define         EVP_CIPH_CCM_MODE               0x7
#define         EVP_CIPH_XTS_MODE               0x10001
#define         EVP_CIPH_MODE                   0xF0007
/* Set if variable length cipher */
#define         EVP_CIPH_VARIABLE_LENGTH        0x8
/* Set if the iv handling should be done by the cipher itself */
#define         EVP_CIPH_CUSTOM_IV              0x10
/* Set if the cipher's init() function should be called if key is NULL */
#define         EVP_CIPH_ALWAYS_CALL_INIT       0x20
/* Call ctrl() to init cipher parameters */
#define         EVP_CIPH_CTRL_INIT              0x40
/* Don't use standard key length function */
#define         EVP_CIPH_CUSTOM_KEY_LENGTH      0x80
/* Don't use standard block padding */
#define         EVP_CIPH_NO_PADDING             0x100
/* cipher handles random key generation */
#define         EVP_CIPH_RAND_KEY               0x200
/* cipher has its own additional copying logic */
#define         EVP_CIPH_CUSTOM_COPY            0x400
/* Allow use default ASN1 get/set iv */
#define         EVP_CIPH_FLAG_DEFAULT_ASN1      0x1000
/* Buffer length in bits not bytes: CFB1 mode only */
#define         EVP_CIPH_FLAG_LENGTH_BITS       0x2000
/* Note if suitable for use in FIPS mode */
#define         EVP_CIPH_FLAG_FIPS              0x4000
/* Allow non FIPS cipher in FIPS mode */
#define         EVP_CIPH_FLAG_NON_FIPS_ALLOW    0x8000
/* Cipher handles any and all padding logic as well
 * as finalisation.
 */
#define         EVP_CIPH_FLAG_CUSTOM_CIPHER     0x100000
#define         EVP_CIPH_FLAG_AEAD_CIPHER       0x200000

/* ctrl() values */

#define         EVP_CTRL_INIT                   0x0
#define         EVP_CTRL_SET_KEY_LENGTH         0x1
#define         EVP_CTRL_GET_RC2_KEY_BITS       0x2
#define         EVP_CTRL_SET_RC2_KEY_BITS       0x3
#define         EVP_CTRL_GET_RC5_ROUNDS         0x4
#define         EVP_CTRL_SET_RC5_ROUNDS         0x5
#define         EVP_CTRL_RAND_KEY               0x6
#define         EVP_CTRL_PBE_PRF_NID            0x7
#define         EVP_CTRL_COPY                   0x8
#define         EVP_CTRL_GCM_SET_IVLEN          0x9
#define         EVP_CTRL_GCM_GET_TAG            0x10
#define         EVP_CTRL_GCM_SET_TAG            0x11
#define         EVP_CTRL_GCM_SET_IV_FIXED       0x12
#define         EVP_CTRL_GCM_IV_GEN             0x13
#define         EVP_CTRL_CCM_SET_IVLEN          EVP_CTRL_GCM_SET_IVLEN
#define         EVP_CTRL_CCM_GET_TAG            EVP_CTRL_GCM_GET_TAG
#define         EVP_CTRL_CCM_SET_TAG            EVP_CTRL_GCM_SET_TAG
#define         EVP_CTRL_CCM_SET_L              0x14
#define         EVP_CTRL_CCM_SET_MSGLEN         0x15
/* AEAD cipher deduces payload length and returns number of bytes
 * required to store MAC and eventual padding. Subsequent call to
 * EVP_Cipher even appends/verifies MAC.
 */
#define         EVP_CTRL_AEAD_TLS1_AAD          0x16
```

```
392 /* Used by composite AEAD ciphers, no-op in GCM, CCM... */
393 #define         EVP_CTRL_AEAD_SET_MAC_KEY       0x17
394 /* Set the GCM invocation field, decrypt only */
395 #define         EVP_CTRL_GCM_SET_IV_INV         0x18

397 /* GCM TLS constants */
398 /* Length of fixed part of IV derived from PRF */
399 #define EVP_GCM_TLS_FIXED_IV_LEN                        4
400 /* Length of explicit part of IV part of TLS records */
401 #define EVP_GCM_TLS_EXPLICIT_IV_LEN                     8
402 /* Length of tag for TLS */
403 #define EVP_GCM_TLS_TAG_LEN                             16

405 typedef struct evp_cipher_info_st
406         {
407         const EVP_CIPHER *cipher;
408         unsigned char iv[EVP_MAX_IV_LENGTH];
409         } EVP_CIPHER_INFO;

411 struct evp_cipher_ctx_st
412         {
413         const EVP_CIPHER *cipher;
414         ENGINE *engine; /* functional reference if 'cipher' is ENGINE-provided *
415         int encrypt;            /* encrypt or decrypt */
416         int buf_len;            /* number we have left */

418         unsigned char  oiv[EVP_MAX_IV_LENGTH];  /* original iv */
419         unsigned char  iv[EVP_MAX_IV_LENGTH];   /* working iv */
420         unsigned char buf[EVP_MAX_BLOCK_LENGTH];/* saved partial block */
421         int num;                                /* used by cfb/ofb/ctr mode */

423         void *app_data;         /* application stuff */
424         int key_len;            /* May change for variable length cipher */
425         unsigned long flags;    /* Various flags */
426         void *cipher_data; /* per EVP data */
427         int final_used;
428         int block_mask;
429         unsigned char final[EVP_MAX_BLOCK_LENGTH];/* possible final block */
430         } /* EVP_CIPHER_CTX */;

432 typedef struct evp_Encode_Ctx_st
433         {
434         int num;        /* number saved in a partial encode/decode */
435         int length;     /* The length is either the output line length
436                          * (in input bytes) or the shortest input line
437                          * length that is ok.  Once decoding begins,
438                          * the length is adjusted up each time a longer
439                          * line is decoded */
440         unsigned char enc_data[80];     /* data to encode */
441         int line_num;   /* number read on current line */
442         int expect_nl;
443         } EVP_ENCODE_CTX;

445 /* Password based encryption function */
446 typedef int (EVP_PBE_KEYGEN)(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
447                 ASN1_TYPE *param, const EVP_CIPHER *cipher,
448                 const EVP_MD *md, int en_de);

450 #ifndef OPENSSL_NO_RSA
451 #define EVP_PKEY_assign_RSA(pkey,rsa) EVP_PKEY_assign((pkey),EVP_PKEY_RSA,\
452                                         (char *)(rsa))
453 #endif

455 #ifndef OPENSSL_NO_DSA
456 #define EVP_PKEY_assign_DSA(pkey,dsa) EVP_PKEY_assign((pkey),EVP_PKEY_DSA,\
457                                         (char *)(dsa))
```



```
458 #endif

460 #ifndef OPENSSL_NO_DH
461 #define EVP_PKEY_assign_DH(pkey,dh) EVP_PKEY_assign((pkey),EVP_PKEY_DH,\
462                                         (char *)(dh))
463 #endif

465 #ifndef OPENSSL_NO_EC
466 #define EVP_PKEY_assign_EC_KEY(pkey,eckey) EVP_PKEY_assign((pkey),EVP_PKEY_EC,\
467                                         (char *)(eckey))
468 #endif

470 /* Add some extra combinations */
471 #define EVP_get_digestbynid(a) EVP_get_digestbyname(OBJ_nid2sn(a))
472 #define EVP_get_digestbyobj(a) EVP_get_digestbynid(OBJ_obj2nid(a))
473 #define EVP_get_cipherbynid(a) EVP_get_cipherbyname(OBJ_nid2sn(a))
474 #define EVP_get_cipherbyobj(a) EVP_get_cipherbynid(OBJ_obj2nid(a))

476 int EVP_MD_type(const EVP_MD *md);
477 #define EVP_MD_nid(e)                   EVP_MD_type(e)
478 #define EVP_MD_name(e)                  OBJ_nid2sn(EVP_MD_nid(e))
479 int EVP_MD_pkey_type(const EVP_MD *md);
480 int EVP_MD_size(const EVP_MD *md);
481 int EVP_MD_block_size(const EVP_MD *md);
482 unsigned long EVP_MD_flags(const EVP_MD *md);

484 const EVP_MD *EVP_MD_CTX_md(const EVP_MD_CTX *ctx);
485 #define EVP_MD_CTX_size(e)              EVP_MD_size(EVP_MD_CTX_md(e))
486 #define EVP_MD_CTX_block_size(e)        EVP_MD_block_size(EVP_MD_CTX_md(e))
487 #define EVP_MD_CTX_type(e)              EVP_MD_type(EVP_MD_CTX_md(e))

489 int EVP_CIPHER_nid(const EVP_CIPHER *cipher);
490 #define EVP_CIPHER_name(e)              OBJ_nid2sn(EVP_CIPHER_nid(e))
491 int EVP_CIPHER_block_size(const EVP_CIPHER *cipher);
492 int EVP_CIPHER_key_length(const EVP_CIPHER *cipher);
493 int EVP_CIPHER_iv_length(const EVP_CIPHER *cipher);
494 unsigned long EVP_CIPHER_flags(const EVP_CIPHER *cipher);
495 #define EVP_CIPHER_mode(e)              (EVP_CIPHER_flags(e) & EVP_CIPH_MODE)

497 const EVP_CIPHER * EVP_CIPHER_CTX_cipher(const EVP_CIPHER_CTX *ctx);
498 int EVP_CIPHER_CTX_nid(const EVP_CIPHER_CTX *ctx);
499 int EVP_CIPHER_CTX_block_size(const EVP_CIPHER_CTX *ctx);
500 int EVP_CIPHER_CTX_key_length(const EVP_CIPHER_CTX *ctx);
501 int EVP_CIPHER_CTX_iv_length(const EVP_CIPHER_CTX *ctx);
502 int EVP_CIPHER_CTX_copy(EVP_CIPHER_CTX *out, const EVP_CIPHER_CTX *in);
503 void * EVP_CIPHER_CTX_get_app_data(const EVP_CIPHER_CTX *ctx);
504 void EVP_CIPHER_CTX_set_app_data(EVP_CIPHER_CTX *ctx, void *data);
505 #define EVP_CIPHER_CTX_type(c)          EVP_CIPHER_type(EVP_CIPHER_CTX_cipher(c))
506 unsigned long EVP_CIPHER_CTX_flags(const EVP_CIPHER_CTX *ctx);
507 #define EVP_CIPHER_CTX_mode(e)          (EVP_CIPHER_CTX_flags(e) & EVP_CIPH_MODE

509 #define EVP_ENCODE_LENGTH(l)    (((l+2)/3*4)+(l/48+1)*2+80)
510 #define EVP_DECODE_LENGTH(l)    ((l+3)/4*3+80)

512 #define EVP_SignInit_ex(a,b,c)          EVP_DigestInit_ex(a,b,c)
513 #define EVP_SignInit(a,b)               EVP_DigestInit(a,b)
514 #define EVP_SignUpdate(a,b,c)           EVP_DigestUpdate(a,b,c)
515 #define EVP_VerifyInit_ex(a,b,c)        EVP_DigestInit_ex(a,b,c)
516 #define EVP_VerifyInit(a,b)             EVP_DigestInit(a,b)
517 #define EVP_VerifyUpdate(a,b,c)         EVP_DigestUpdate(a,b,c)
518 #define EVP_OpenUpdate(a,b,c,d,e)       EVP_DecryptUpdate(a,b,c,d,e)
519 #define EVP_SealUpdate(a,b,c,d,e)       EVP_EncryptUpdate(a,b,c,d,e)
520 #define EVP_DigestSignUpdate(a,b,c)     EVP_DigestUpdate(a,b,c)
521 #define EVP_DigestVerifyUpdate(a,b,c)   EVP_DigestUpdate(a,b,c)

523 #ifdef CONST_STRICT
```

```
524 void BIO_set_md(BIO *,const EVP_MD *md);
525 #else
526 # define BIO_set_md(b,md)               BIO_ctrl(b,BIO_C_SET_MD,0,(char *)md)
527 #endif
528 #define BIO_get_md(b,mdp)               BIO_ctrl(b,BIO_C_GET_MD,0,(char *)mdp)
529 #define BIO_get_md_ctx(b,mdcp)     BIO_ctrl(b,BIO_C_GET_MD_CTX,0,(char *)mdcp)
530 #define BIO_set_md_ctx(b,mdcp)     BIO_ctrl(b,BIO_C_SET_MD_CTX,0,(char *)mdcp)
531 #define BIO_get_cipher_status(b)        BIO_ctrl(b,BIO_C_GET_CIPHER_STATUS,0,NUL
532 #define BIO_get_cipher_ctx(b,c_pp)      BIO_ctrl(b,BIO_C_GET_CIPHER_CTX,0,(char

534 int EVP_Cipher(EVP_CIPHER_CTX *c,
535                unsigned char *out,
536                const unsigned char *in,
537                unsigned int inl);

539 #define EVP_add_cipher_alias(n,alias) \
540         OBJ_NAME_add((alias),OBJ_NAME_TYPE_CIPHER_METH|OBJ_NAME_ALIAS,(n))
541 #define EVP_add_digest_alias(n,alias) \
542         OBJ_NAME_add((alias),OBJ_NAME_TYPE_MD_METH|OBJ_NAME_ALIAS,(n))
543 #define EVP_delete_cipher_alias(alias) \
544         OBJ_NAME_remove(alias,OBJ_NAME_TYPE_CIPHER_METH|OBJ_NAME_ALIAS);
545 #define EVP_delete_digest_alias(alias) \
546         OBJ_NAME_remove(alias,OBJ_NAME_TYPE_MD_METH|OBJ_NAME_ALIAS);

548 void    EVP_MD_CTX_init(EVP_MD_CTX *ctx);
549 int     EVP_MD_CTX_cleanup(EVP_MD_CTX *ctx);
550 EVP_MD_CTX *EVP_MD_CTX_create(void);
551 void    EVP_MD_CTX_destroy(EVP_MD_CTX *ctx);
552 int     EVP_MD_CTX_copy_ex(EVP_MD_CTX *out,const EVP_MD_CTX *in);
553 void    EVP_MD_CTX_set_flags(EVP_MD_CTX *ctx, int flags);
554 void    EVP_MD_CTX_clear_flags(EVP_MD_CTX *ctx, int flags);
555 int     EVP_MD_CTX_test_flags(const EVP_MD_CTX *ctx,int flags);
556 int     EVP_DigestInit_ex(EVP_MD_CTX *ctx, const EVP_MD *type, ENGINE *impl);
557 int     EVP_DigestUpdate(EVP_MD_CTX *ctx,const void *d,
558                          size_t cnt);
559 int     EVP_DigestFinal_ex(EVP_MD_CTX *ctx,unsigned char *md,unsigned int *s);
560 int     EVP_Digest(const void *data, size_t count,
561                unsigned char *md, unsigned int *size, const EVP_MD *type, ENGIN

563 int     EVP_MD_CTX_copy(EVP_MD_CTX *out,const EVP_MD_CTX *in);
564 int     EVP_DigestInit(EVP_MD_CTX *ctx, const EVP_MD *type);
565 int     EVP_DigestFinal(EVP_MD_CTX *ctx,unsigned char *md,unsigned int *s);

567 int     EVP_read_pw_string(char *buf,int length,const char *prompt,int verify);
568 int     EVP_read_pw_string_min(char *buf,int minlen,int maxlen,const char *promp
569 void    EVP_set_pw_prompt(const char *prompt);
570 char *  EVP_get_pw_prompt(void);

572 int     EVP_BytesToKey(const EVP_CIPHER *type,const EVP_MD *md,
573                const unsigned char *salt, const unsigned char *data,
574                int datal, int count, unsigned char *key,unsigned char *iv);

576 void    EVP_CIPHER_CTX_set_flags(EVP_CIPHER_CTX *ctx, int flags);
577 void    EVP_CIPHER_CTX_clear_flags(EVP_CIPHER_CTX *ctx, int flags);
578 int     EVP_CIPHER_CTX_test_flags(const EVP_CIPHER_CTX *ctx,int flags);

580 int     EVP_EncryptInit(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *cipher,
581                const unsigned char *key, const unsigned char *iv);
582 int     EVP_EncryptInit_ex(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *cipher, ENGINE
583                const unsigned char *key, const unsigned char *iv);
584 int     EVP_EncryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out,
585                int *outl, const unsigned char *in, int inl);
586 int     EVP_EncryptFinal_ex(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl);
587 int     EVP_EncryptFinal(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl);

589 int     EVP_DecryptInit(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *cipher,
```



```
590                const unsigned char *key, const unsigned char *iv);
591 int     EVP_DecryptInit_ex(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *cipher, ENGINE
592                const unsigned char *key, const unsigned char *iv);
593 int     EVP_DecryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out,
594                int *outl, const unsigned char *in, int inl);
595 int     EVP_DecryptFinal(EVP_CIPHER_CTX *ctx, unsigned char *outm, int *outl);
596 int     EVP_DecryptFinal_ex(EVP_CIPHER_CTX *ctx, unsigned char *outm, int *outl)

598 int     EVP_CipherInit(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *cipher,
599                        const unsigned char *key,const unsigned char *iv,
600                        int enc);
601 int     EVP_CipherInit_ex(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *cipher, ENGINE *
602                        const unsigned char *key,const unsigned char *iv,
603                        int enc);
604 int     EVP_CipherUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out,
605                int *outl, const unsigned char *in, int inl);
606 int     EVP_CipherFinal(EVP_CIPHER_CTX *ctx, unsigned char *outm, int *outl);
607 int     EVP_CipherFinal_ex(EVP_CIPHER_CTX *ctx, unsigned char *outm, int *outl);

609 int     EVP_SignFinal(EVP_MD_CTX *ctx,unsigned char *md,unsigned int *s,
610                EVP_PKEY *pkey);

612 int     EVP_VerifyFinal(EVP_MD_CTX *ctx,const unsigned char *sigbuf,
613                unsigned int siglen,EVP_PKEY *pkey);

615 int     EVP_DigestSignInit(EVP_MD_CTX *ctx, EVP_PKEY_CTX **pctx,
616                        const EVP_MD *type, ENGINE *e, EVP_PKEY *pkey);
617 int     EVP_DigestSignFinal(EVP_MD_CTX *ctx,
618                        unsigned char *sigret, size_t *siglen);

620 int     EVP_DigestVerifyInit(EVP_MD_CTX *ctx, EVP_PKEY_CTX **pctx,
621                        const EVP_MD *type, ENGINE *e, EVP_PKEY *pkey);
622 int     EVP_DigestVerifyFinal(EVP_MD_CTX *ctx,
623                        unsigned char *sig, size_t siglen);

625 int     EVP_OpenInit(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *type,
626                const unsigned char *ek, int ekl, const unsigned char *iv,
627                EVP_PKEY *priv);
628 int     EVP_OpenFinal(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl);

630 int     EVP_SealInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type,
631                 unsigned char **ek, int *ekl, unsigned char *iv,
632                EVP_PKEY **pubk, int npubk);
633 int     EVP_SealFinal(EVP_CIPHER_CTX *ctx,unsigned char *out,int *outl);

635 void    EVP_EncodeInit(EVP_ENCODE_CTX *ctx);
636 void    EVP_EncodeUpdate(EVP_ENCODE_CTX *ctx,unsigned char *out,int *outl,
637                const unsigned char *in,int inl);
638 void    EVP_EncodeFinal(EVP_ENCODE_CTX *ctx,unsigned char *out,int *outl);
639 int     EVP_EncodeBlock(unsigned char *t, const unsigned char *f, int n);

641 void    EVP_DecodeInit(EVP_ENCODE_CTX *ctx);
642 int     EVP_DecodeUpdate(EVP_ENCODE_CTX *ctx,unsigned char *out,int *outl,
643                const unsigned char *in, int inl);
644 int     EVP_DecodeFinal(EVP_ENCODE_CTX *ctx, unsigned
645                char *out, int *outl);
646 int     EVP_DecodeBlock(unsigned char *t, const unsigned char *f, int n);

648 void EVP_CIPHER_CTX_init(EVP_CIPHER_CTX *a);
649 int EVP_CIPHER_CTX_cleanup(EVP_CIPHER_CTX *a);
650 EVP_CIPHER_CTX *EVP_CIPHER_CTX_new(void);
651 void EVP_CIPHER_CTX_free(EVP_CIPHER_CTX *a);
652 int EVP_CIPHER_CTX_set_key_length(EVP_CIPHER_CTX *x, int keylen);
653 int EVP_CIPHER_CTX_set_padding(EVP_CIPHER_CTX *c, int pad);
654 int EVP_CIPHER_CTX_ctrl(EVP_CIPHER_CTX *ctx, int type, int arg, void *ptr);
655 int EVP_CIPHER_CTX_rand_key(EVP_CIPHER_CTX *ctx, unsigned char *key);
```

```
#ifndef OPENSSL_NO_BIO
BIO_METHOD *BIO_f_md(void);
BIO_METHOD *BIO_f_base64(void);
BIO_METHOD *BIO_f_cipher(void);
BIO_METHOD *BIO_f_reliable(void);
void BIO_set_cipher(BIO *b,const EVP_CIPHER *c,const unsigned char *k,
                    const unsigned char *i, int enc);
#endif

const EVP_MD *EVP_md_null(void);
#ifndef OPENSSL_NO_MD2
const EVP_MD *EVP_md2(void);
#endif
#ifndef OPENSSL_NO_MD4
const EVP_MD *EVP_md4(void);
#endif
#ifndef OPENSSL_NO_MD5
const EVP_MD *EVP_md5(void);
#endif
#ifndef OPENSSL_NO_SHA
const EVP_MD *EVP_sha(void);
const EVP_MD *EVP_sha1(void);
const EVP_MD *EVP_dss(void);
const EVP_MD *EVP_dss1(void);
const EVP_MD *EVP_ecdsa(void);
#endif
#ifndef OPENSSL_NO_SHA256
const EVP_MD *EVP_sha224(void);
const EVP_MD *EVP_sha256(void);
#endif
#ifndef OPENSSL_NO_SHA512
const EVP_MD *EVP_sha384(void);
const EVP_MD *EVP_sha512(void);
#endif
#ifndef OPENSSL_NO_MDC2
const EVP_MD *EVP_mdc2(void);
#endif
#ifndef OPENSSL_NO_RIPEMD
const EVP_MD *EVP_ripemd160(void);
#endif
#ifndef OPENSSL_NO_WHIRLPOOL
const EVP_MD *EVP_whirlpool(void);
#endif
const EVP_CIPHER *EVP_enc_null(void);           /* does nothing :-) */
#ifndef OPENSSL_NO_DES
const EVP_CIPHER *EVP_des_ecb(void);
const EVP_CIPHER *EVP_des_ede(void);
const EVP_CIPHER *EVP_des_ede3(void);
const EVP_CIPHER *EVP_des_ede_ecb(void);
const EVP_CIPHER *EVP_des_ede3_ecb(void);
const EVP_CIPHER *EVP_des_cfb64(void);
# define EVP_des_cfb EVP_des_cfb64
const EVP_CIPHER *EVP_des_cfb1(void);
const EVP_CIPHER *EVP_des_cfb8(void);
const EVP_CIPHER *EVP_des_ede_cfb64(void);
# define EVP_des_ede_cfb EVP_des_ede_cfb64
#if 0
const EVP_CIPHER *EVP_des_ede_cfb1(void);
const EVP_CIPHER *EVP_des_ede_cfb8(void);
#endif
const EVP_CIPHER *EVP_des_ede3_cfb64(void);
# define EVP_des_ede3_cfb EVP_des_ede3_cfb64
const EVP_CIPHER *EVP_des_ede3_cfb1(void);
const EVP_CIPHER *EVP_des_ede3_cfb8(void);
const EVP_CIPHER *EVP_des_ofb(void);
```



```
const EVP_CIPHER *EVP_des_ede_ofb(void);
const EVP_CIPHER *EVP_des_ede3_ofb(void);
const EVP_CIPHER *EVP_des_cbc(void);
const EVP_CIPHER *EVP_des_ede_cbc(void);
const EVP_CIPHER *EVP_des_ede3_cbc(void);
const EVP_CIPHER *EVP_desx_cbc(void);
/* This should now be supported through the dev_crypto ENGINE. But also, why are
 * rc4 and md5 declarations made here inside a "NO_DES" precompiler branch? */
#if 0
# ifdef OPENSSL_OPENBSD_DEV_CRYPTO
const EVP_CIPHER *EVP_dev_crypto_des_ede3_cbc(void);
const EVP_CIPHER *EVP_dev_crypto_rc4(void);
const EVP_MD *EVP_dev_crypto_md5(void);
# endif
#endif
#endif
#ifndef OPENSSL_NO_RC4
const EVP_CIPHER *EVP_rc4(void);
const EVP_CIPHER *EVP_rc4_40(void);
#ifndef OPENSSL_NO_MD5
const EVP_CIPHER *EVP_rc4_hmac_md5(void);
#endif
#endif
#ifndef OPENSSL_NO_IDEA
const EVP_CIPHER *EVP_idea_ecb(void);
const EVP_CIPHER *EVP_idea_cfb64(void);
# define EVP_idea_cfb EVP_idea_cfb64
const EVP_CIPHER *EVP_idea_ofb(void);
const EVP_CIPHER *EVP_idea_cbc(void);
#endif
#ifndef OPENSSL_NO_RC2
const EVP_CIPHER *EVP_rc2_ecb(void);
const EVP_CIPHER *EVP_rc2_cbc(void);
const EVP_CIPHER *EVP_rc2_40_cbc(void);
const EVP_CIPHER *EVP_rc2_64_cbc(void);
const EVP_CIPHER *EVP_rc2_cfb64(void);
# define EVP_rc2_cfb EVP_rc2_cfb64
const EVP_CIPHER *EVP_rc2_ofb(void);
#endif
#ifndef OPENSSL_NO_BF
const EVP_CIPHER *EVP_bf_ecb(void);
const EVP_CIPHER *EVP_bf_cbc(void);
const EVP_CIPHER *EVP_bf_cfb64(void);
# define EVP_bf_cfb EVP_bf_cfb64
const EVP_CIPHER *EVP_bf_ofb(void);
#endif
#ifndef OPENSSL_NO_CAST
const EVP_CIPHER *EVP_cast5_ecb(void);
const EVP_CIPHER *EVP_cast5_cbc(void);
const EVP_CIPHER *EVP_cast5_cfb64(void);
# define EVP_cast5_cfb EVP_cast5_cfb64
const EVP_CIPHER *EVP_cast5_ofb(void);
#endif
#ifndef OPENSSL_NO_RC5
const EVP_CIPHER *EVP_rc5_32_12_16_cbc(void);
const EVP_CIPHER *EVP_rc5_32_12_16_ecb(void);
const EVP_CIPHER *EVP_rc5_32_12_16_cfb64(void);
# define EVP_rc5_32_12_16_cfb EVP_rc5_32_12_16_cfb64
const EVP_CIPHER *EVP_rc5_32_12_16_ofb(void);
#endif
#ifndef OPENSSL_NO_AES
const EVP_CIPHER *EVP_aes_128_ecb(void);
const EVP_CIPHER *EVP_aes_128_cbc(void);
const EVP_CIPHER *EVP_aes_128_cfb1(void);
const EVP_CIPHER *EVP_aes_128_cfb8(void);
const EVP_CIPHER *EVP_aes_128_cfb128(void);
```

```
788 # define EVP_aes_128_cfb EVP_aes_128_cfb128
789 const EVP_CIPHER *EVP_aes_128_ofb(void);
790 const EVP_CIPHER *EVP_aes_128_ctr(void);
791 const EVP_CIPHER *EVP_aes_128_ccm(void);
792 const EVP_CIPHER *EVP_aes_128_gcm(void);
793 const EVP_CIPHER *EVP_aes_128_xts(void);
794 const EVP_CIPHER *EVP_aes_192_ecb(void);
795 const EVP_CIPHER *EVP_aes_192_cbc(void);
796 const EVP_CIPHER *EVP_aes_192_cfb1(void);
797 const EVP_CIPHER *EVP_aes_192_cfb8(void);
798 const EVP_CIPHER *EVP_aes_192_cfb128(void);
799 # define EVP_aes_192_cfb EVP_aes_192_cfb128
800 const EVP_CIPHER *EVP_aes_192_ofb(void);
801 const EVP_CIPHER *EVP_aes_192_ctr(void);
802 const EVP_CIPHER *EVP_aes_192_ccm(void);
803 const EVP_CIPHER *EVP_aes_192_gcm(void);
804 const EVP_CIPHER *EVP_aes_256_ecb(void);
805 const EVP_CIPHER *EVP_aes_256_cbc(void);
806 const EVP_CIPHER *EVP_aes_256_cfb1(void);
807 const EVP_CIPHER *EVP_aes_256_cfb8(void);
808 const EVP_CIPHER *EVP_aes_256_cfb128(void);
809 # define EVP_aes_256_cfb EVP_aes_256_cfb128
810 const EVP_CIPHER *EVP_aes_256_ofb(void);
811 const EVP_CIPHER *EVP_aes_256_ctr(void);
812 const EVP_CIPHER *EVP_aes_256_ccm(void);
813 const EVP_CIPHER *EVP_aes_256_gcm(void);
814 const EVP_CIPHER *EVP_aes_256_xts(void);
815 #if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA1)
816 const EVP_CIPHER *EVP_aes_128_cbc_hmac_sha1(void);
817 const EVP_CIPHER *EVP_aes_256_cbc_hmac_sha1(void);
818 #endif
819 #endif
820 #ifndef OPENSSL_NO_CAMELLIA
821 const EVP_CIPHER *EVP_camellia_128_ecb(void);
822 const EVP_CIPHER *EVP_camellia_128_cbc(void);
823 const EVP_CIPHER *EVP_camellia_128_cfb1(void);
824 const EVP_CIPHER *EVP_camellia_128_cfb8(void);
825 const EVP_CIPHER *EVP_camellia_128_cfb128(void);
826 # define EVP_camellia_128_cfb EVP_camellia_128_cfb128
827 const EVP_CIPHER *EVP_camellia_128_ofb(void);
828 const EVP_CIPHER *EVP_camellia_192_ecb(void);
829 const EVP_CIPHER *EVP_camellia_192_cbc(void);
830 const EVP_CIPHER *EVP_camellia_192_cfb1(void);
831 const EVP_CIPHER *EVP_camellia_192_cfb8(void);
832 const EVP_CIPHER *EVP_camellia_192_cfb128(void);
833 # define EVP_camellia_192_cfb EVP_camellia_192_cfb128
834 const EVP_CIPHER *EVP_camellia_192_ofb(void);
835 const EVP_CIPHER *EVP_camellia_256_ecb(void);
836 const EVP_CIPHER *EVP_camellia_256_cbc(void);
837 const EVP_CIPHER *EVP_camellia_256_cfb1(void);
838 const EVP_CIPHER *EVP_camellia_256_cfb8(void);
839 const EVP_CIPHER *EVP_camellia_256_cfb128(void);
840 # define EVP_camellia_256_cfb EVP_camellia_256_cfb128
841 const EVP_CIPHER *EVP_camellia_256_ofb(void);
842 #endif

844 #ifndef OPENSSL_NO_SEED
845 const EVP_CIPHER *EVP_seed_ecb(void);
846 const EVP_CIPHER *EVP_seed_cbc(void);
847 const EVP_CIPHER *EVP_seed_cfb128(void);
848 # define EVP_seed_cfb EVP_seed_cfb128
849 const EVP_CIPHER *EVP_seed_ofb(void);
850 #endif

852 void OPENSSL_add_all_algorithms_noconf(void);
853 void OPENSSL_add_all_algorithms_conf(void);
```



```
855 #ifdef OPENSSL_LOAD_CONF
856 #define OpenSSL_add_all_algorithms() \
857                 OPENSSL_add_all_algorithms_conf()
858 #else
859 #define OpenSSL_add_all_algorithms() \
860                 OPENSSL_add_all_algorithms_noconf()
861 #endif

863 void OpenSSL_add_all_ciphers(void);
864 void OpenSSL_add_all_digests(void);
865 #define SSLeay_add_all_algorithms() OpenSSL_add_all_algorithms()
866 #define SSLeay_add_all_ciphers() OpenSSL_add_all_ciphers()
867 #define SSLeay_add_all_digests() OpenSSL_add_all_digests()

869 int EVP_add_cipher(const EVP_CIPHER *cipher);
870 int EVP_add_digest(const EVP_MD *digest);

872 const EVP_CIPHER *EVP_get_cipherbyname(const char *name);
873 const EVP_MD *EVP_get_digestbyname(const char *name);
874 void EVP_cleanup(void);

876 void EVP_CIPHER_do_all(void (*fn)(const EVP_CIPHER *ciph,
877                 const char *from, const char *to, void *x), void *arg);
878 void EVP_CIPHER_do_all_sorted(void (*fn)(const EVP_CIPHER *ciph,
879                 const char *from, const char *to, void *x), void *arg);

881 void EVP_MD_do_all(void (*fn)(const EVP_MD *ciph,
882                 const char *from, const char *to, void *x), void *arg);
883 void EVP_MD_do_all_sorted(void (*fn)(const EVP_MD *ciph,
884                 const char *from, const char *to, void *x), void *arg);

886 int             EVP_PKEY_decrypt_old(unsigned char *dec_key,
887                         const unsigned char *enc_key,int enc_key_len,
888                         EVP_PKEY *private_key);
889 int             EVP_PKEY_encrypt_old(unsigned char *enc_key,
890                         const unsigned char *key,int key_len,
891                         EVP_PKEY *pub_key);
892 int             EVP_PKEY_type(int type);
893 int             EVP_PKEY_id(const EVP_PKEY *pkey);
894 int             EVP_PKEY_base_id(const EVP_PKEY *pkey);
895 int             EVP_PKEY_bits(EVP_PKEY *pkey);
896 int             EVP_PKEY_size(EVP_PKEY *pkey);
897 int             EVP_PKEY_set_type(EVP_PKEY *pkey,int type);
898 int             EVP_PKEY_set_type_str(EVP_PKEY *pkey, const char *str, int len);
899 int             EVP_PKEY_assign(EVP_PKEY *pkey,int type,void *key);
900 void *          EVP_PKEY_get0(EVP_PKEY *pkey);

902 #ifndef OPENSSL_NO_RSA
903 struct rsa_st;
904 int EVP_PKEY_set1_RSA(EVP_PKEY *pkey,struct rsa_st *key);
905 struct rsa_st *EVP_PKEY_get1_RSA(EVP_PKEY *pkey);
906 #endif
907 #ifndef OPENSSL_NO_DSA
908 struct dsa_st;
909 int EVP_PKEY_set1_DSA(EVP_PKEY *pkey,struct dsa_st *key);
910 struct dsa_st *EVP_PKEY_get1_DSA(EVP_PKEY *pkey);
911 #endif
912 #ifndef OPENSSL_NO_DH
913 struct dh_st;
914 int EVP_PKEY_set1_DH(EVP_PKEY *pkey,struct dh_st *key);
915 struct dh_st *EVP_PKEY_get1_DH(EVP_PKEY *pkey);
916 #endif
917 #ifndef OPENSSL_NO_EC
918 struct ec_key_st;
919 int EVP_PKEY_set1_EC_KEY(EVP_PKEY *pkey,struct ec_key_st *key);
```

```
struct ec_key_st *EVP_PKEY_get1_EC_KEY(EVP_PKEY *pkey);
#endif

EVP_PKEY *      EVP_PKEY_new(void);
void            EVP_PKEY_free(EVP_PKEY *pkey);

EVP_PKEY *      d2i_PublicKey(int type,EVP_PKEY **a, const unsigned char **pp,
                        long length);
int             i2d_PublicKey(EVP_PKEY *a, unsigned char **pp);

EVP_PKEY *      d2i_PrivateKey(int type,EVP_PKEY **a, const unsigned char **pp,
                        long length);
EVP_PKEY *      d2i_AutoPrivateKey(EVP_PKEY **a, const unsigned char **pp,
                        long length);
int             i2d_PrivateKey(EVP_PKEY *a, unsigned char **pp);

int EVP_PKEY_copy_parameters(EVP_PKEY *to, const EVP_PKEY *from);
int EVP_PKEY_missing_parameters(const EVP_PKEY *pkey);
int EVP_PKEY_save_parameters(EVP_PKEY *pkey,int mode);
int EVP_PKEY_cmp_parameters(const EVP_PKEY *a, const EVP_PKEY *b);

int EVP_PKEY_cmp(const EVP_PKEY *a, const EVP_PKEY *b);

int EVP_PKEY_print_public(BIO *out, const EVP_PKEY *pkey,
                                int indent, ASN1_PCTX *pctx);
int EVP_PKEY_print_private(BIO *out, const EVP_PKEY *pkey,
                                int indent, ASN1_PCTX *pctx);
int EVP_PKEY_print_params(BIO *out, const EVP_PKEY *pkey,
                                int indent, ASN1_PCTX *pctx);

int EVP_PKEY_get_default_digest_nid(EVP_PKEY *pkey, int *pnid);

int EVP_CIPHER_type(const EVP_CIPHER *ctx);

/* calls methods */
int EVP_CIPHER_param_to_asn1(EVP_CIPHER_CTX *c, ASN1_TYPE *type);
int EVP_CIPHER_asn1_to_param(EVP_CIPHER_CTX *c, ASN1_TYPE *type);

/* These are used by EVP_CIPHER methods */
int EVP_CIPHER_set_asn1_iv(EVP_CIPHER_CTX *c,ASN1_TYPE *type);
int EVP_CIPHER_get_asn1_iv(EVP_CIPHER_CTX *c,ASN1_TYPE *type);

/* PKCS5 password based encryption */
int PKCS5_PBE_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
                        ASN1_TYPE *param, const EVP_CIPHER *cipher, const EVP_M
                        int en_de);
int PKCS5_PBKDF2_HMAC_SHA1(const char *pass, int passlen,
                          const unsigned char *salt, int saltlen, int iter,
                          int keylen, unsigned char *out);
int PKCS5_PBKDF2_HMAC(const char *pass, int passlen,
                          const unsigned char *salt, int saltlen, int iter,
                          const EVP_MD *digest,
                      int keylen, unsigned char *out);
int PKCS5_v2_PBE_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
                        ASN1_TYPE *param, const EVP_CIPHER *cipher, const EVP_M
                        int en_de);

void PKCS5_PBE_add(void);

int EVP_PBE_CipherInit (ASN1_OBJECT *pbe_obj, const char *pass, int passlen,
             ASN1_TYPE *param, EVP_CIPHER_CTX *ctx, int en_de);

/* PBE type */

/* Can appear as the outermost AlgorithmIdentifier */
#define EVP_PBE_TYPE_OUTER      0x0
```



```
/* Is an PRF type OID */
#define EVP_PBE_TYPE_PRF        0x1

int EVP_PBE_alg_add_type(int pbe_type, int pbe_nid, int cipher_nid, int md_nid,
             EVP_PBE_KEYGEN *keygen);
int EVP_PBE_alg_add(int nid, const EVP_CIPHER *cipher, const EVP_MD *md,
                    EVP_PBE_KEYGEN *keygen);
int EVP_PBE_find(int type, int pbe_nid,
                        int *pcnid, int *pmnid, EVP_PBE_KEYGEN **pkeygen);
void EVP_PBE_cleanup(void);

#define ASN1_PKEY_ALIAS         0x1
#define ASN1_PKEY_DYNAMIC       0x2
#define ASN1_PKEY_SIGPARAM_NULL 0x4

#define ASN1_PKEY_CTRL_PKCS7_SIGN       0x1
#define ASN1_PKEY_CTRL_PKCS7_ENCRYPT    0x2
#define ASN1_PKEY_CTRL_DEFAULT_MD_NID   0x3
#define ASN1_PKEY_CTRL_CMS_SIGN         0x5
#define ASN1_PKEY_CTRL_CMS_ENVELOPE     0x7

int EVP_PKEY_asn1_get_count(void);
const EVP_PKEY_ASN1_METHOD *EVP_PKEY_asn1_get0(int idx);
const EVP_PKEY_ASN1_METHOD *EVP_PKEY_asn1_find(ENGINE **pe, int type);
const EVP_PKEY_ASN1_METHOD *EVP_PKEY_asn1_find_str(ENGINE **pe,
                                        const char *str, int len);
int EVP_PKEY_asn1_add0(const EVP_PKEY_ASN1_METHOD *ameth);
int EVP_PKEY_asn1_add_alias(int to, int from);
int EVP_PKEY_asn1_get0_info(int *ppkey_id, int *pkey_base_id, int *ppkey_flags,
                                const char **pinfo, const char **ppem_str,
                                        const EVP_PKEY_ASN1_METHOD *ameth);

const EVP_PKEY_ASN1_METHOD* EVP_PKEY_get0_asn1(EVP_PKEY *pkey);
EVP_PKEY_ASN1_METHOD* EVP_PKEY_asn1_new(int id, int flags,
                                        const char *pem_str, const char *info);
void EVP_PKEY_asn1_copy(EVP_PKEY_ASN1_METHOD *dst,
                        const EVP_PKEY_ASN1_METHOD *src);
void EVP_PKEY_asn1_free(EVP_PKEY_ASN1_METHOD *ameth);
void EVP_PKEY_asn1_set_public(EVP_PKEY_ASN1_METHOD *ameth,
                int (*pub_decode)(EVP_PKEY *pk, X509_PUBKEY *pub),
                int (*pub_encode)(X509_PUBKEY *pub, const EVP_PKEY *pk),
                int (*pub_cmp)(const EVP_PKEY *a, const EVP_PKEY *b),
                int (*pub_print)(BIO *out, const EVP_PKEY *pkey, int indent,
                                                        ASN1_PCTX *pctx),
                int (*pkey_size)(const EVP_PKEY *pk),
                int (*pkey_bits)(const EVP_PKEY *pk));
void EVP_PKEY_asn1_set_private(EVP_PKEY_ASN1_METHOD *ameth,
                int (*priv_decode)(EVP_PKEY *pk, PKCS8_PRIV_KEY_INFO *p8inf),
                int (*priv_encode)(PKCS8_PRIV_KEY_INFO *p8, const EVP_PKEY *pk),
                int (*priv_print)(BIO *out, const EVP_PKEY *pkey, int indent,
                                                        ASN1_PCTX *pctx));
void EVP_PKEY_asn1_set_param(EVP_PKEY_ASN1_METHOD *ameth,
                int (*param_decode)(EVP_PKEY *pkey,
                                const unsigned char **pder, int derlen),
                int (*param_encode)(const EVP_PKEY *pkey, unsigned char **pder),
                int (*param_missing)(const EVP_PKEY *pk),
                int (*param_copy)(EVP_PKEY *to, const EVP_PKEY *from),
                int (*param_cmp)(const EVP_PKEY *a, const EVP_PKEY *b),
                int (*param_print)(BIO *out, const EVP_PKEY *pkey, int indent,
                                                        ASN1_PCTX *pctx));

void EVP_PKEY_asn1_set_free(EVP_PKEY_ASN1_METHOD *ameth,
                void (*pkey_free)(EVP_PKEY *pkey));
void EVP_PKEY_asn1_set_ctrl(EVP_PKEY_ASN1_METHOD *ameth,
                int (*pkey_ctrl)(EVP_PKEY *pkey, int op,
                                                        long arg1, void *arg2));
```

```
#define EVP_PKEY_OP_UNDEFINED           0
#define EVP_PKEY_OP_PARAMGEN            (1<<1)
#define EVP_PKEY_OP_KEYGEN              (1<<2)
#define EVP_PKEY_OP_SIGN                (1<<3)
#define EVP_PKEY_OP_VERIFY              (1<<4)
#define EVP_PKEY_OP_VERIFYRECOVER       (1<<5)
#define EVP_PKEY_OP_SIGNCTX             (1<<6)
#define EVP_PKEY_OP_VERIFYCTX           (1<<7)
#define EVP_PKEY_OP_ENCRYPT             (1<<8)
#define EVP_PKEY_OP_DECRYPT             (1<<9)
#define EVP_PKEY_OP_DERIVE              (1<<10)

#define EVP_PKEY_OP_TYPE_SIG    \
        (EVP_PKEY_OP_SIGN | EVP_PKEY_OP_VERIFY | EVP_PKEY_OP_VERIFYRECOVER \
                | EVP_PKEY_OP_SIGNCTX | EVP_PKEY_OP_VERIFYCTX)

#define EVP_PKEY_OP_TYPE_CRYPT \
        (EVP_PKEY_OP_ENCRYPT | EVP_PKEY_OP_DECRYPT)

#define EVP_PKEY_OP_TYPE_NOGEN \
        (EVP_PKEY_OP_SIG | EVP_PKEY_OP_CRYPT | EVP_PKEY_OP_DERIVE)

#define EVP_PKEY_OP_TYPE_GEN \
                (EVP_PKEY_OP_PARAMGEN | EVP_PKEY_OP_KEYGEN)

#define  EVP_PKEY_CTX_set_signature_md(ctx, md) \
                EVP_PKEY_CTX_ctrl(ctx, -1, EVP_PKEY_OP_TYPE_SIG,  \
                                  EVP_PKEY_CTRL_MD, 0, (void *)md)

#define EVP_PKEY_CTRL_MD                1
#define EVP_PKEY_CTRL_PEER_KEY          2

#define EVP_PKEY_CTRL_PKCS7_ENCRYPT     3
#define EVP_PKEY_CTRL_PKCS7_DECRYPT     4

#define EVP_PKEY_CTRL_PKCS7_SIGN        5

#define EVP_PKEY_CTRL_SET_MAC_KEY       6

#define EVP_PKEY_CTRL_DIGESTINIT        7

/* Used by GOST key encryption in TLS */
#define EVP_PKEY_CTRL_SET_IV            8

#define EVP_PKEY_CTRL_CMS_ENCRYPT       9
#define EVP_PKEY_CTRL_CMS_DECRYPT       10
#define EVP_PKEY_CTRL_CMS_SIGN          11

#define EVP_PKEY_CTRL_CIPHER            12

#define EVP_PKEY_ALG_CTRL               0x1000


#define EVP_PKEY_FLAG_AUTOARGLEN        2
/* Method handles all operations: don't assume any digest related
 * defaults.
 */
#define EVP_PKEY_FLAG_SIGCTX_CUSTOM     4

const EVP_PKEY_METHOD *EVP_PKEY_meth_find(int type);
EVP_PKEY_METHOD* EVP_PKEY_meth_new(int id, int flags);
void EVP_PKEY_meth_get0_info(int *ppkey_id, int *pflags,
                             const EVP_PKEY_METHOD *meth);
void EVP_PKEY_meth_copy(EVP_PKEY_METHOD *dst, const EVP_PKEY_METHOD *src);
```



```
void EVP_PKEY_meth_free(EVP_PKEY_METHOD *pmeth);
int EVP_PKEY_meth_add0(const EVP_PKEY_METHOD *pmeth);

EVP_PKEY_CTX *EVP_PKEY_CTX_new(EVP_PKEY *pkey, ENGINE *e);
EVP_PKEY_CTX *EVP_PKEY_CTX_new_id(int id, ENGINE *e);
EVP_PKEY_CTX *EVP_PKEY_CTX_dup(EVP_PKEY_CTX *ctx);
void EVP_PKEY_CTX_free(EVP_PKEY_CTX *ctx);

int EVP_PKEY_CTX_ctrl(EVP_PKEY_CTX *ctx, int keytype, int optype,
                                  int cmd, int p1, void *p2);
int EVP_PKEY_CTX_ctrl_str(EVP_PKEY_CTX *ctx, const char *type,
                                                const char *value);

int EVP_PKEY_CTX_get_operation(EVP_PKEY_CTX *ctx);
void EVP_PKEY_CTX_set0_keygen_info(EVP_PKEY_CTX *ctx, int *dat, int datlen);

EVP_PKEY *EVP_PKEY_new_mac_key(int type, ENGINE *e,
                                  const unsigned char *key, int keylen);

void EVP_PKEY_CTX_set_data(EVP_PKEY_CTX *ctx, void *data);
void *EVP_PKEY_CTX_get_data(EVP_PKEY_CTX *ctx);
EVP_PKEY *EVP_PKEY_CTX_get0_pkey(EVP_PKEY_CTX *ctx);

EVP_PKEY *EVP_PKEY_CTX_get0_peerkey(EVP_PKEY_CTX *ctx);

void EVP_PKEY_CTX_set_app_data(EVP_PKEY_CTX *ctx, void *data);
void *EVP_PKEY_CTX_get_app_data(EVP_PKEY_CTX *ctx);

int EVP_PKEY_sign_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_sign(EVP_PKEY_CTX *ctx,
                        unsigned char *sig, size_t *siglen,
                        const unsigned char *tbs, size_t tbslen);
int EVP_PKEY_verify_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_verify(EVP_PKEY_CTX *ctx,
                        const unsigned char *sig, size_t siglen,
                        const unsigned char *tbs, size_t tbslen);
int EVP_PKEY_verify_recover_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_verify_recover(EVP_PKEY_CTX *ctx,
                        unsigned char *rout, size_t *routlen,
                        const unsigned char *sig, size_t siglen);
int EVP_PKEY_encrypt_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_encrypt(EVP_PKEY_CTX *ctx,
                        unsigned char *out, size_t *outlen,
                        const unsigned char *in, size_t inlen);
int EVP_PKEY_decrypt_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_decrypt(EVP_PKEY_CTX *ctx,
                        unsigned char *out, size_t *outlen,
                        const unsigned char *in, size_t inlen);

int EVP_PKEY_derive_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_derive_set_peer(EVP_PKEY_CTX *ctx, EVP_PKEY *peer);
int EVP_PKEY_derive(EVP_PKEY_CTX *ctx, unsigned char *key, size_t *keylen);

typedef int EVP_PKEY_gen_cb(EVP_PKEY_CTX *ctx);

int EVP_PKEY_paramgen_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_paramgen(EVP_PKEY_CTX *ctx, EVP_PKEY **ppkey);
int EVP_PKEY_keygen_init(EVP_PKEY_CTX *ctx);
int EVP_PKEY_keygen(EVP_PKEY_CTX *ctx, EVP_PKEY **ppkey);

void EVP_PKEY_CTX_set_cb(EVP_PKEY_CTX *ctx, EVP_PKEY_gen_cb *cb);
EVP_PKEY_gen_cb *EVP_PKEY_CTX_get_cb(EVP_PKEY_CTX *ctx);

int EVP_PKEY_CTX_get_keygen_info(EVP_PKEY_CTX *ctx, int idx);

void EVP_PKEY_meth_set_init(EVP_PKEY_METHOD *pmeth,
```

```
1184         int (*init)(EVP_PKEY_CTX *ctx));

1186 void EVP_PKEY_meth_set_copy(EVP_PKEY_METHOD *pmeth,
1187         int (*copy)(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src));

1189 void EVP_PKEY_meth_set_cleanup(EVP_PKEY_METHOD *pmeth,
1190         void (*cleanup)(EVP_PKEY_CTX *ctx));

1192 void EVP_PKEY_meth_set_paramgen(EVP_PKEY_METHOD *pmeth,
1193         int (*paramgen_init)(EVP_PKEY_CTX *ctx),
1194         int (*paramgen)(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey));

1196 void EVP_PKEY_meth_set_keygen(EVP_PKEY_METHOD *pmeth,
1197         int (*keygen_init)(EVP_PKEY_CTX *ctx),
1198         int (*keygen)(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey));

1200 void EVP_PKEY_meth_set_sign(EVP_PKEY_METHOD *pmeth,
1201         int (*sign_init)(EVP_PKEY_CTX *ctx),
1202         int (*sign)(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
1203                                         const unsigned char *tbs, size_t tbslen)

1205 void EVP_PKEY_meth_set_verify(EVP_PKEY_METHOD *pmeth,
1206         int (*verify_init)(EVP_PKEY_CTX *ctx),
1207         int (*verify)(EVP_PKEY_CTX *ctx, const unsigned char *sig, size_t siglen
1208                                         const unsigned char *tbs, size_t tbslen)

1210 void EVP_PKEY_meth_set_verify_recover(EVP_PKEY_METHOD *pmeth,
1211         int (*verify_recover_init)(EVP_PKEY_CTX *ctx),
1212         int (*verify_recover)(EVP_PKEY_CTX *ctx,
1213                                         unsigned char *sig, size_t *siglen,
1214                                         const unsigned char *tbs, size_t tbslen)

1216 void EVP_PKEY_meth_set_signctx(EVP_PKEY_METHOD *pmeth,
1217         int (*signctx_init)(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx),
1218         int (*signctx)(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
1219                                         EVP_MD_CTX *mctx));

1221 void EVP_PKEY_meth_set_verifyctx(EVP_PKEY_METHOD *pmeth,
1222         int (*verifyctx_init)(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx),
1223         int (*verifyctx)(EVP_PKEY_CTX *ctx, const unsigned char *sig,int siglen,
1224                                         EVP_MD_CTX *mctx));

1226 void EVP_PKEY_meth_set_encrypt(EVP_PKEY_METHOD *pmeth,
1227         int (*encrypt_init)(EVP_PKEY_CTX *ctx),
1228         int (*encryptfn)(EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,
1229                                         const unsigned char *in, size_t inlen));

1231 void EVP_PKEY_meth_set_decrypt(EVP_PKEY_METHOD *pmeth,
1232         int (*decrypt_init)(EVP_PKEY_CTX *ctx),
1233         int (*decrypt)(EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,
1234                                         const unsigned char *in, size_t inlen));

1236 void EVP_PKEY_meth_set_derive(EVP_PKEY_METHOD *pmeth,
1237         int (*derive_init)(EVP_PKEY_CTX *ctx),
1238         int (*derive)(EVP_PKEY_CTX *ctx, unsigned char *key, size_t *keylen));

1240 void EVP_PKEY_meth_set_ctrl(EVP_PKEY_METHOD *pmeth,
1241         int (*ctrl)(EVP_PKEY_CTX *ctx, int type, int p1, void *p2),
1242         int (*ctrl_str)(EVP_PKEY_CTX *ctx,
1243                                         const char *type, const char *value));

1245 void EVP_add_alg_module(void);

1247 /* BEGIN ERROR CODES */
1248 /* The following lines are auto generated by the script mkerr.pl. Any changes
1249  * made after this point may be overwritten when the script is next run.
```



```
1250  */
1251 void ERR_load_EVP_strings(void);

1253 /* Error codes for the EVP functions. */

1255 /* Function codes. */
1256 #define EVP_F_AESNI_INIT_KEY                            165
1257 #define EVP_F_AESNI_XTS_CIPHER                          176
1258 #define EVP_F_AES_INIT_KEY                              133
1259 #define EVP_F_AES_XTS                                   172
1260 #define EVP_F_AES_XTS_CIPHER                            175
1261 #define EVP_F_ALG_MODULE_INIT                           177
1262 #define EVP_F_CAMELLIA_INIT_KEY                         159
1263 #define EVP_F_CMAC_INIT                                 173
1264 #define EVP_F_D2I_PKEY                                  100
1265 #define EVP_F_DO_SIGVER_INIT                            161
1266 #define EVP_F_DSAPKEY2PKCS8                             134
1267 #define EVP_F_DSA_PKEY2PKCS8                            135
1268 #define EVP_F_ECDSA_PKEY2PKCS8                          129
1269 #define EVP_F_ECKEY_PKEY2PKCS8                          132
1270 #define EVP_F_EVP_CIPHERINIT_EX                         123
1271 #define EVP_F_EVP_CIPHER_CTX_COPY                       163
1272 #define EVP_F_EVP_CIPHER_CTX_CTRL                       124
1273 #define EVP_F_EVP_CIPHER_CTX_SET_KEY_LENGTH             122
1274 #define EVP_F_EVP_DECRYPTFINAL_EX                       101
1275 #define EVP_F_EVP_DIGESTINIT_EX                         128
1276 #define EVP_F_EVP_ENCRYPTFINAL_EX                       127
1277 #define EVP_F_EVP_MD_CTX_COPY_EX                        110
1278 #define EVP_F_EVP_MD_SIZE                               162
1279 #define EVP_F_EVP_OPENINIT                              102
1280 #define EVP_F_EVP_PBE_ALG_ADD                           115
1281 #define EVP_F_EVP_PBE_ALG_ADD_TYPE                      160
1282 #define EVP_F_EVP_PBE_CIPHERINIT                        116
1283 #define EVP_F_EVP_PKCS82PKEY                            111
1284 #define EVP_F_EVP_PKCS82PKEY_BROKEN                     136
1285 #define EVP_F_EVP_PKEY2PKCS8_BROKEN                     113
1286 #define EVP_F_EVP_PKEY_COPY_PARAMETERS                  103
1287 #define EVP_F_EVP_PKEY_CTX_CTRL                         137
1288 #define EVP_F_EVP_PKEY_CTX_CTRL_STR                     150
1289 #define EVP_F_EVP_PKEY_CTX_DUP                          156
1290 #define EVP_F_EVP_PKEY_DECRYPT                          104
1291 #define EVP_F_EVP_PKEY_DECRYPT_INIT                     138
1292 #define EVP_F_EVP_PKEY_DECRYPT_OLD                      151
1293 #define EVP_F_EVP_PKEY_DERIVE                           153
1294 #define EVP_F_EVP_PKEY_DERIVE_INIT                      154
1295 #define EVP_F_EVP_PKEY_DERIVE_SET_PEER                  155
1296 #define EVP_F_EVP_PKEY_ENCRYPT                          105
1297 #define EVP_F_EVP_PKEY_ENCRYPT_INIT                     139
1298 #define EVP_F_EVP_PKEY_ENCRYPT_OLD                      152
1299 #define EVP_F_EVP_PKEY_GET1_DH                          119
1300 #define EVP_F_EVP_PKEY_GET1_DSA                         120
1301 #define EVP_F_EVP_PKEY_GET1_ECDSA                       130
1302 #define EVP_F_EVP_PKEY_GET1_EC_KEY                      131
1303 #define EVP_F_EVP_PKEY_GET1_RSA                         121
1304 #define EVP_F_EVP_PKEY_KEYGEN                           146
1305 #define EVP_F_EVP_PKEY_KEYGEN_INIT                      147
1306 #define EVP_F_EVP_PKEY_NEW                              106
1307 #define EVP_F_EVP_PKEY_PARAMGEN                         148
1308 #define EVP_F_EVP_PKEY_PARAMGEN_INIT                    149
1309 #define EVP_F_EVP_PKEY_SIGN                             140
1310 #define EVP_F_EVP_PKEY_SIGN_INIT                        141
1311 #define EVP_F_EVP_PKEY_VERIFY                           142
1312 #define EVP_F_EVP_PKEY_VERIFY_INIT                      143
1313 #define EVP_F_EVP_PKEY_VERIFY_RECOVER                   144
1314 #define EVP_F_EVP_PKEY_VERIFY_RECOVER_INIT              145
1315 #define EVP_F_EVP_RIJNDAEL                              126
```

```
#define EVP_F_EVP_SIGNFINAL                             107
#define EVP_F_EVP_VERIFYFINAL                           108
#define EVP_F_FIPS_CIPHERINIT                           166
#define EVP_F_FIPS_CIPHER_CTX_COPY                      170
#define EVP_F_FIPS_CIPHER_CTX_CTRL                      167
#define EVP_F_FIPS_CIPHER_CTX_SET_KEY_LENGTH            171
#define EVP_F_FIPS_DIGESTINIT                           168
#define EVP_F_FIPS_MD_CTX_COPY                          169
#define EVP_F_HMAC_INIT_EX                              174
#define EVP_F_INT_CTX_NEW                               157
#define EVP_F_PKCS5_PBE_KEYIVGEN                        117
#define EVP_F_PKCS5_V2_PBE_KEYIVGEN                     118
#define EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN                  164
#define EVP_F_PKCS8_SET_BROKEN                          112
#define EVP_F_PKEY_SET_TYPE                             158
#define EVP_F_RC2_MAGIC_TO_METH                         109
#define EVP_F_RC5_CTRL                                  125

/* Reason codes. */
#define EVP_R_AES_IV_SETUP_FAILED                       162
#define EVP_R_AES_KEY_SETUP_FAILED                      143
#define EVP_R_ASN1_LIB                                  140
#define EVP_R_BAD_BLOCK_LENGTH                          136
#define EVP_R_BAD_DECRYPT                               100
#define EVP_R_BAD_KEY_LENGTH                            137
#define EVP_R_BN_DECODE_ERROR                           112
#define EVP_R_BN_PUBKEY_ERROR                           113
#define EVP_R_BUFFER_TOO_SMALL                          155
#define EVP_R_CAMELLIA_KEY_SETUP_FAILED                 157
#define EVP_R_CIPHER_PARAMETER_ERROR                    122
#define EVP_R_COMMAND_NOT_SUPPORTED                     147
#define EVP_R_CTRL_NOT_IMPLEMENTED                      132
#define EVP_R_CTRL_OPERATION_NOT_IMPLEMENTED            133
#define EVP_R_DATA_NOT_MULTIPLE_OF_BLOCK_LENGTH         138
#define EVP_R_DECODE_ERROR                              114
#define EVP_R_DIFFERENT_KEY_TYPES                       101
#define EVP_R_DIFFERENT_PARAMETERS                      153
#define EVP_R_DISABLED_FOR_FIPS                         163
#define EVP_R_ENCODE_ERROR                              115
#define EVP_R_ERROR_LOADING_SECTION                     165
#define EVP_R_ERROR_SETTING_FIPS_MODE                   166
#define EVP_R_EVP_PBE_CIPHERINIT_ERROR                  119
#define EVP_R_EXPECTING_AN_RSA_KEY                      127
#define EVP_R_EXPECTING_A_DH_KEY                        128
#define EVP_R_EXPECTING_A_DSA_KEY                       129
#define EVP_R_EXPECTING_A_ECDSA_KEY                     141
#define EVP_R_EXPECTING_A_EC_KEY                        142
#define EVP_R_FIPS_MODE_NOT_SUPPORTED                   167
#define EVP_R_INITIALIZATION_ERROR                      134
#define EVP_R_INPUT_NOT_INITIALIZED                     111
#define EVP_R_INVALID_DIGEST                            152
#define EVP_R_INVALID_FIPS_MODE                         168
#define EVP_R_INVALID_KEY_LENGTH                        130
#define EVP_R_INVALID_OPERATION                         148
#define EVP_R_IV_TOO_LARGE                              102
#define EVP_R_KEYGEN_FAILURE                            120
#define EVP_R_MESSAGE_DIGEST_IS_NULL                    159
#define EVP_R_METHOD_NOT_SUPPORTED                      144
#define EVP_R_MISSING_PARAMETERS                        103
#define EVP_R_NO_CIPHER_SET                             131
#define EVP_R_NO_DEFAULT_DIGEST                         158
#define EVP_R_NO_DIGEST_SET                             139
#define EVP_R_NO_DSA_PARAMETERS                         116
#define EVP_R_NO_KEY_SET                                154
#define EVP_R_NO_OPERATION_SET                          149
#define EVP_R_NO_SIGN_FUNCTION_CONFIGURED               104
```



```
#define EVP_R_NO_VERIFY_FUNCTION_CONFIGURED             105
#define EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE  150
#define EVP_R_OPERATON_NOT_INITIALIZED                  151
#define EVP_R_PKCS8_UNKNOWN_BROKEN_TYPE                 117
#define EVP_R_PRIVATE_KEY_DECODE_ERROR                  145
#define EVP_R_PRIVATE_KEY_ENCODE_ERROR                  146
#define EVP_R_PUBLIC_KEY_NOT_RSA                        106
#define EVP_R_TOO_LARGE                                 164
#define EVP_R_UNKNOWN_CIPHER                            160
#define EVP_R_UNKNOWN_DIGEST                            161
#define EVP_R_UNKNOWN_OPTION                            169
#define EVP_R_UNKNOWN_PBE_ALGORITHM                     121
#define EVP_R_UNSUPORTED_NUMBER_OF_ROUNDS               135
#define EVP_R_UNSUPPORTED_ALGORITHM                     156
#define EVP_R_UNSUPPORTED_CIPHER                        107
#define EVP_R_UNSUPPORTED_KEYLENGTH                     123
#define EVP_R_UNSUPPORTED_KEY_DERIVATION_FUNCTION       124
#define EVP_R_UNSUPPORTED_KEY_SIZE                      108
#define EVP_R_UNSUPPORTED_PRF                           125
#define EVP_R_UNSUPPORTED_PRIVATE_KEY_ALGORITHM         118
#define EVP_R_UNSUPPORTED_SALT_TYPE                     126
#define EVP_R_WRONG_FINAL_BLOCK_LENGTH                  109
#define EVP_R_WRONG_PUBLIC_KEY_TYPE                     110

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4470 Wed Aug 13 19:51:44 2014
new/usr/src/lib/openssl/include/openssl/hmac.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/hmac/hmac.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
#ifndef HEADER_HMAC_H
#define HEADER_HMAC_H

#include <openssl/opensslconf.h>
```



```
#ifdef OPENSSL_NO_HMAC
#error HMAC is disabled.
#endif

#include <openssl/evp.h>

#define HMAC_MAX_MD_CBLOCK      128     /* largest known is SHA512 */

#ifdef __cplusplus
extern "C" {
#endif

typedef struct hmac_ctx_st
        {
        const EVP_MD *md;
        EVP_MD_CTX md_ctx;
        EVP_MD_CTX i_ctx;
        EVP_MD_CTX o_ctx;
        unsigned int key_length;
        unsigned char key[HMAC_MAX_MD_CBLOCK];
        } HMAC_CTX;

#define HMAC_size(e)    (EVP_MD_size((e)->md))


void HMAC_CTX_init(HMAC_CTX *ctx);
void HMAC_CTX_cleanup(HMAC_CTX *ctx);

#define HMAC_cleanup(ctx) HMAC_CTX_cleanup(ctx) /* deprecated */

int HMAC_Init(HMAC_CTX *ctx, const void *key, int len,
              const EVP_MD *md); /* deprecated */
int HMAC_Init_ex(HMAC_CTX *ctx, const void *key, int len,
                 const EVP_MD *md, ENGINE *impl);
int HMAC_Update(HMAC_CTX *ctx, const unsigned char *data, size_t len);
int HMAC_Final(HMAC_CTX *ctx, unsigned char *md, unsigned int *len);
unsigned char *HMAC(const EVP_MD *evp_md, const void *key, int key_len,
                    const unsigned char *d, size_t n, unsigned char *md,
                    unsigned int *md_len);
int HMAC_CTX_copy(HMAC_CTX *dctx, HMAC_CTX *sctx);

void HMAC_CTX_set_flags(HMAC_CTX *ctx, unsigned long flags);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    7637 Wed Aug 13 19:51:44 2014
new/usr/src/lib/openssl/include/openssl/krb5_asn.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* krb5_asn.h */
/* Written by Vern Staats <staatsvr@asc.hpc.mil> for the OpenSSL project,
** using ocsp/{*.h,*asn*.c} as a starting point
*/

/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_KRB5_ASN_H
#define HEADER_KRB5_ASN_H
```



```
/*
#include <krb5.h>
*/
#include <openssl/safestack.h>

#ifdef  __cplusplus
extern "C" {
#endif


/*      ASN.1 from Kerberos RFC 1510
*/

/*      EncryptedData ::=   SEQUENCE {
**              etype[0]                        INTEGER, -- EncryptionType
**              kvno[1]                         INTEGER OPTIONAL,
**              cipher[2]                       OCTET STRING -- ciphertext
**      }
*/
typedef struct  krb5_encdata_st
        {
        ASN1_INTEGER                    *etype;
        ASN1_INTEGER                    *kvno;
        ASN1_OCTET_STRING               *cipher;
        }       KRB5_ENCDATA;

DECLARE_STACK_OF(KRB5_ENCDATA)

/*      PrincipalName ::=   SEQUENCE {
**              name-type[0]                    INTEGER,
**              name-string[1]                  SEQUENCE OF GeneralString
**      }
*/
typedef struct  krb5_princname_st
        {
        ASN1_INTEGER                    *nametype;
        STACK_OF(ASN1_GENERALSTRING)    *namestring;
        }       KRB5_PRINCNAME;

DECLARE_STACK_OF(KRB5_PRINCNAME)


/*      Ticket ::=      [APPLICATION 1] SEQUENCE {
**              tkt-vno[0]                      INTEGER,
**              realm[1]                        Realm,
**              sname[2]                        PrincipalName,
**              enc-part[3]                     EncryptedData
**      }
*/
typedef struct  krb5_tktbody_st
        {
        ASN1_INTEGER                    *tktvno;
        ASN1_GENERALSTRING              *realm;
        KRB5_PRINCNAME                  *sname;
        KRB5_ENCDATA                    *encdata;
        }       KRB5_TKTBODY;

typedef STACK_OF(KRB5_TKTBODY) KRB5_TICKET;
DECLARE_STACK_OF(KRB5_TKTBODY)


/*      AP-REQ ::=      [APPLICATION 14] SEQUENCE {
**              pvno[0]                         INTEGER,
**              msg-type[1]                     INTEGER,
**              ap-options[2]                   APOptions,
```

```
**              ticket[3]                       Ticket,
**              authenticator[4]                EncryptedData
**      }
**
**      APOptions ::=   BIT STRING {
**              reserved(0), use-session-key(1), mutual-required(2) }
*/
typedef struct  krb5_ap_req_st
        {
        ASN1_INTEGER                    *pvno;
        ASN1_INTEGER                    *msgtype;
        ASN1_BIT_STRING                 *apoptions;
        KRB5_TICKET                     *ticket;
        KRB5_ENCDATA                    *authenticator;
        }       KRB5_APREQBODY;

typedef STACK_OF(KRB5_APREQBODY) KRB5_APREQ;
DECLARE_STACK_OF(KRB5_APREQBODY)


/*      Authenticator Stuff     */


/*      Checksum ::=   SEQUENCE {
**              cksumtype[0]                    INTEGER,
**              checksum[1]                     OCTET STRING
**      }
*/
typedef struct  krb5_checksum_st
        {
        ASN1_INTEGER                    *ctype;
        ASN1_OCTET_STRING               *checksum;
        }       KRB5_CHECKSUM;

DECLARE_STACK_OF(KRB5_CHECKSUM)


/*      EncryptionKey ::=   SEQUENCE {
**              keytype[0]                      INTEGER,
**              keyvalue[1]                     OCTET STRING
**      }
*/
typedef struct  krb5_encryptionkey_st
        {
        ASN1_INTEGER                    *ktype;
        ASN1_OCTET_STRING               *keyvalue;
        }       KRB5_ENCKEY;

DECLARE_STACK_OF(KRB5_ENCKEY)


/*      AuthorizationData ::=   SEQUENCE OF SEQUENCE {
**              ad-type[0]                      INTEGER,
**              ad-data[1]                      OCTET STRING
**      }
*/
typedef struct  krb5_authorization_st
        {
        ASN1_INTEGER                    *adtype;
        ASN1_OCTET_STRING               *addata;
        }       KRB5_AUTHDATA;

DECLARE_STACK_OF(KRB5_AUTHDATA)


/*      -- Unencrypted authenticator
```



```
**      Authenticator ::=    [APPLICATION 2] SEQUENCE    {
**              authenticator-vno[0]            INTEGER,
**              crealm[1]                       Realm,
**              cname[2]                        PrincipalName,
**              cksum[3]                        Checksum OPTIONAL,
**              cusec[4]                        INTEGER,
**              ctime[5]                        KerberosTime,
**              subkey[6]                       EncryptionKey OPTIONAL,
**              seq-number[7]                   INTEGER OPTIONAL,
**              authorization-data[8]           AuthorizationData OPTIONAL
**      }
*/
typedef struct  krb5_authenticator_st
        {
        ASN1_INTEGER                    *avno;
        ASN1_GENERALSTRING              *crealm;
        KRB5_PRINCNAME                  *cname;
        KRB5_CHECKSUM                   *cksum;
        ASN1_INTEGER                    *cusec;
        ASN1_GENERALIZEDTIME            *ctime;
        KRB5_ENCKEY                     *subkey;
        ASN1_INTEGER                    *seqnum;
        KRB5_AUTHDATA                   *authorization;
        }       KRB5_AUTHENTBODY;

typedef STACK_OF(KRB5_AUTHENTBODY) KRB5_AUTHENT;
DECLARE_STACK_OF(KRB5_AUTHENTBODY)


/*  DECLARE_ASN1_FUNCTIONS(type) = DECLARE_ASN1_FUNCTIONS_name(type, type) =
**      type *name##_new(void);
**      void name##_free(type *a);
**      DECLARE_ASN1_ENCODE_FUNCTIONS(type, name, name) =
**       DECLARE_ASN1_ENCODE_FUNCTIONS(type, itname, name) =
**        type *d2i_##name(type **a, const unsigned char **in, long len);
**        int i2d_##name(type *a, unsigned char **out);
**        DECLARE_ASN1_ITEM(itname) = OPENSSL_EXTERN const ASN1_ITEM itname##_it
*/

DECLARE_ASN1_FUNCTIONS(KRB5_ENCDATA)
DECLARE_ASN1_FUNCTIONS(KRB5_PRINCNAME)
DECLARE_ASN1_FUNCTIONS(KRB5_TKTBODY)
DECLARE_ASN1_FUNCTIONS(KRB5_APREQBODY)
DECLARE_ASN1_FUNCTIONS(KRB5_TICKET)
DECLARE_ASN1_FUNCTIONS(KRB5_APREQ)

DECLARE_ASN1_FUNCTIONS(KRB5_CHECKSUM)
DECLARE_ASN1_FUNCTIONS(KRB5_ENCKEY)
DECLARE_ASN1_FUNCTIONS(KRB5_AUTHDATA)
DECLARE_ASN1_FUNCTIONS(KRB5_AUTHENTBODY)
DECLARE_ASN1_FUNCTIONS(KRB5_AUTHENT)


/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    6289 Wed Aug 13 19:51:44 2014
new/usr/src/lib/openssl/include/openssl/kssl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/kssl.h -*- mode: C; c-file-style: "eay" -*- */
/* Written by Vern Staats <staatsvr@asc.hpc.mil> for the OpenSSL project 2000.
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/*
**      19990701        VRS     Started.
*/
```



```
#ifndef KSSL_H
#define KSSL_H

#include <openssl/opensslconf.h>

#ifndef OPENSSL_NO_KRB5

#include <stdio.h>
#include <ctype.h>
#include <krb5.h>
#ifdef OPENSSL_SYS_WIN32
/* These can sometimes get redefined indirectly by krb5 header files
 * after they get undefed in ossl_typ.h
 */
#undef X509_NAME
#undef X509_EXTENSIONS
#undef OCSP_REQUEST
#undef OCSP_RESPONSE
#endif

#ifdef  __cplusplus
extern "C" {
#endif

/*
**      Depending on which KRB5 implementation used, some types from
**      the other may be missing.  Resolve that here and now
*/
#ifdef KRB5_HEIMDAL
typedef unsigned char krb5_octet;
#define FAR
#else

#ifndef FAR
#define FAR
#endif

#endif

/*      Uncomment this to debug kssl problems or
**      to trace usage of the Kerberos session key
**
**      #define         KSSL_DEBUG
*/

#ifndef KRB5SVC
#define KRB5SVC "host"
#endif

#ifndef KRB5KEYTAB
#define KRB5KEYTAB      "/etc/krb5.keytab"
#endif

#ifndef KRB5SENDAUTH
#define KRB5SENDAUTH    1
#endif

#ifndef KRB5CHECKAUTH
#define KRB5CHECKAUTH   1
#endif

#ifndef KSSL_CLOCKSKEW
#define KSSL_CLOCKSKEW  300;
#endif
```

```
#define KSSL_ERR_MAX    255
typedef struct kssl_err_st  {
        int  reason;
        char text[KSSL_ERR_MAX+1];
        } KSSL_ERR;


/*      Context for passing
**              (1) Kerberos session key to SSL, and
**              (2)     Config data between application and SSL lib
*/
typedef struct kssl_ctx_st
        {
                                /*      used by:    disposition:            */
        char *service_name;     /*      C,S         default ok (kssl)       */
        char *service_host;     /*      C           input, REQUIRED         */
        char *client_princ;     /*      S           output from krb5 ticket */
        char *keytab_file;      /*      S           NULL (/etc/krb5.keytab) */
        char *cred_cache;       /*      C           NULL (default)          */
        krb5_enctype enctype;
        int length;
        krb5_octet FAR *key;
        } KSSL_CTX;

#define KSSL_CLIENT     1
#define KSSL_SERVER     2
#define KSSL_SERVICE    3
#define KSSL_KEYTAB     4

#define KSSL_CTX_OK     0
#define KSSL_CTX_ERR    1
#define KSSL_NOMEM      2

/* Public (for use by applications that use OpenSSL with Kerberos 5 support */
krb5_error_code kssl_ctx_setstring(KSSL_CTX *kssl_ctx, int which, char *text);
KSSL_CTX *kssl_ctx_new(void);
KSSL_CTX *kssl_ctx_free(KSSL_CTX *kssl_ctx);
void kssl_ctx_show(KSSL_CTX *kssl_ctx);
krb5_error_code kssl_ctx_setprinc(KSSL_CTX *kssl_ctx, int which,
        krb5_data *realm, krb5_data *entity, int nentities);
krb5_error_code kssl_cget_tkt(KSSL_CTX *kssl_ctx,  krb5_data **enc_tktp,
        krb5_data *authenp, KSSL_ERR *kssl_err);
krb5_error_code kssl_sget_tkt(KSSL_CTX *kssl_ctx,  krb5_data *indata,
        krb5_ticket_times *ttimes, KSSL_ERR *kssl_err);
krb5_error_code kssl_ctx_setkey(KSSL_CTX *kssl_ctx, krb5_keyblock *session);
void    kssl_err_set(KSSL_ERR *kssl_err, int reason, char *text);
void kssl_krb5_free_data_contents(krb5_context context, krb5_data *data);
krb5_error_code  kssl_build_principal_2(krb5_context context,
                        krb5_principal *princ, int rlen, const char *realm,
                        int slen, const char *svc, int hlen, const char *host);
krb5_error_code  kssl_validate_times(krb5_timestamp atime,
                                     krb5_ticket_times *ttimes);
krb5_error_code  kssl_check_authent(KSSL_CTX *kssl_ctx, krb5_data *authentp,
                                    krb5_timestamp *atimep, KSSL_ERR *kssl_err);
unsigned char   *kssl_skip_confound(krb5_enctype enctype, unsigned char *authn);

void SSL_set0_kssl_ctx(SSL *s, KSSL_CTX *kctx);
KSSL_CTX * SSL_get0_kssl_ctx(SSL *s);
char *kssl_ctx_get0_client_princ(KSSL_CTX *kctx);

#ifdef  __cplusplus
}
#endif
#endif  /* OPENSSL_NO_KRB5      */
#endif  /* KSSL_H       */
#endif /* ! codereview */
```

```
**********************************************************
    9085 Wed Aug 13 19:51:44 2014
new/usr/src/lib/openssl/include/openssl/lhash.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/lhash/lhash.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Header for dynamic hash table routines
 * Author - Eric Young
 */
```



```
#ifndef HEADER_LHASH_H
#define HEADER_LHASH_H

#include <openssl/e_os2.h>
#ifndef OPENSSL_NO_FP_API
#include <stdio.h>
#endif

#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct lhash_node_st
        {
        void *data;
        struct lhash_node_st *next;
#ifndef OPENSSL_NO_HASH_COMP
        unsigned long hash;
#endif
        } LHASH_NODE;

typedef int (*LHASH_COMP_FN_TYPE)(const void *, const void *);
typedef unsigned long (*LHASH_HASH_FN_TYPE)(const void *);
typedef void (*LHASH_DOALL_FN_TYPE)(void *);
typedef void (*LHASH_DOALL_ARG_FN_TYPE)(void *, void *);

/* Macros for declaring and implementing type-safe wrappers for LHASH callbacks.
 * This way, callbacks can be provided to LHASH structures without function
 * pointer casting and the macro-defined callbacks provide per-variable casting
 * before deferring to the underlying type-specific callbacks. NB: It is
 * possible to place a "static" in front of both the DECLARE and IMPLEMENT
 * macros if the functions are strictly internal. */

/* First: "hash" functions */
#define DECLARE_LHASH_HASH_FN(name, o_type) \
        unsigned long name##_LHASH_HASH(const void *);
#define IMPLEMENT_LHASH_HASH_FN(name, o_type) \
        unsigned long name##_LHASH_HASH(const void *arg) { \
                const o_type *a = arg; \
                return name##_hash(a); }
#define LHASH_HASH_FN(name) name##_LHASH_HASH

/* Second: "compare" functions */
#define DECLARE_LHASH_COMP_FN(name, o_type) \
        int name##_LHASH_COMP(const void *, const void *);
#define IMPLEMENT_LHASH_COMP_FN(name, o_type) \
        int name##_LHASH_COMP(const void *arg1, const void *arg2) { \
                const o_type *a = arg1;                 \
                const o_type *b = arg2; \
                return name##_cmp(a,b); }
#define LHASH_COMP_FN(name) name##_LHASH_COMP

/* Third: "doall" functions */
#define DECLARE_LHASH_DOALL_FN(name, o_type) \
        void name##_LHASH_DOALL(void *);
#define IMPLEMENT_LHASH_DOALL_FN(name, o_type) \
        void name##_LHASH_DOALL(void *arg) { \
                o_type *a = arg; \
                name##_doall(a); }
#define LHASH_DOALL_FN(name) name##_LHASH_DOALL
```

```
/* Fourth: "doall_arg" functions */
#define DECLARE_LHASH_DOALL_ARG_FN(name, o_type, a_type) \
        void name##_LHASH_DOALL_ARG(void *, void *);
#define IMPLEMENT_LHASH_DOALL_ARG_FN(name, o_type, a_type) \
        void name##_LHASH_DOALL_ARG(void *arg1, void *arg2) { \
                o_type *a = arg1; \
                a_type *b = arg2; \
                name##_doall_arg(a, b); }
#define LHASH_DOALL_ARG_FN(name) name##_LHASH_DOALL_ARG

typedef struct lhash_st
        {
        LHASH_NODE **b;
        LHASH_COMP_FN_TYPE comp;
        LHASH_HASH_FN_TYPE hash;
        unsigned int num_nodes;
        unsigned int num_alloc_nodes;
        unsigned int p;
        unsigned int pmax;
        unsigned long up_load; /* load times 256 */
        unsigned long down_load; /* load times 256 */
        unsigned long num_items;

        unsigned long num_expands;
        unsigned long num_expand_reallocs;
        unsigned long num_contracts;
        unsigned long num_contract_reallocs;
        unsigned long num_hash_calls;
        unsigned long num_comp_calls;
        unsigned long num_insert;
        unsigned long num_replace;
        unsigned long num_delete;
        unsigned long num_no_delete;
        unsigned long num_retrieve;
        unsigned long num_retrieve_miss;
        unsigned long num_hash_comps;

        int error;
        } _LHASH;       /* Do not use _LHASH directly, use LHASH_OF
                         * and friends */

#define LH_LOAD_MULT    256

/* Indicates a malloc() error in the last call, this is only bad
 * in lh_insert(). */
#define lh_error(lh)    ((lh)->error)

_LHASH *lh_new(LHASH_HASH_FN_TYPE h, LHASH_COMP_FN_TYPE c);
void lh_free(_LHASH *lh);
void *lh_insert(_LHASH *lh, void *data);
void *lh_delete(_LHASH *lh, const void *data);
void *lh_retrieve(_LHASH *lh, const void *data);
void lh_doall(_LHASH *lh, LHASH_DOALL_FN_TYPE func);
void lh_doall_arg(_LHASH *lh, LHASH_DOALL_ARG_FN_TYPE func, void *arg);
unsigned long lh_strhash(const char *c);
unsigned long lh_num_items(const _LHASH *lh);

#ifndef OPENSSL_NO_FP_API
void lh_stats(const _LHASH *lh, FILE *out);
void lh_node_stats(const _LHASH *lh, FILE *out);
void lh_node_usage_stats(const _LHASH *lh, FILE *out);
#endif

#ifndef OPENSSL_NO_BIO
void lh_stats_bio(const _LHASH *lh, BIO *out);
void lh_node_stats_bio(const _LHASH *lh, BIO *out);
```



```
void lh_node_usage_stats_bio(const _LHASH *lh, BIO *out);
#endif

/* Type checking... */

#define LHASH_OF(type) struct lhash_st_##type

#define DECLARE_LHASH_OF(type) LHASH_OF(type) { int dummy; }

#define CHECKED_LHASH_OF(type,lh) \
  ((_LHASH *)CHECKED_PTR_OF(LHASH_OF(type),lh))

/* Define wrapper functions. */
#define LHM_lh_new(type, name) \
  ((LHASH_OF(type) *)lh_new(LHASH_HASH_FN(name), LHASH_COMP_FN(name)))
#define LHM_lh_error(type, lh) \
  lh_error(CHECKED_LHASH_OF(type,lh))
#define LHM_lh_insert(type, lh, inst) \
  ((type *)lh_insert(CHECKED_LHASH_OF(type, lh), \
                     CHECKED_PTR_OF(type, inst)))
#define LHM_lh_retrieve(type, lh, inst) \
  ((type *)lh_retrieve(CHECKED_LHASH_OF(type, lh), \
                       CHECKED_PTR_OF(type, inst)))
#define LHM_lh_delete(type, lh, inst) \
  ((type *)lh_delete(CHECKED_LHASH_OF(type, lh),                        \
                     CHECKED_PTR_OF(type, inst)))
#define LHM_lh_doall(type, lh,fn) lh_doall(CHECKED_LHASH_OF(type, lh), fn)
#define LHM_lh_doall_arg(type, lh, fn, arg_type, arg) \
  lh_doall_arg(CHECKED_LHASH_OF(type, lh), fn, CHECKED_PTR_OF(arg_type, arg))
#define LHM_lh_num_items(type, lh) lh_num_items(CHECKED_LHASH_OF(type, lh))
#define LHM_lh_down_load(type, lh) (CHECKED_LHASH_OF(type, lh)->down_load)
#define LHM_lh_node_stats_bio(type, lh, out) \
  lh_node_stats_bio(CHECKED_LHASH_OF(type, lh), out)
#define LHM_lh_node_usage_stats_bio(type, lh, out) \
  lh_node_usage_stats_bio(CHECKED_LHASH_OF(type, lh), out)
#define LHM_lh_stats_bio(type, lh, out) \
  lh_stats_bio(CHECKED_LHASH_OF(type, lh), out)
#define LHM_lh_free(type, lh) lh_free(CHECKED_LHASH_OF(type, lh))

DECLARE_LHASH_OF(OPENSSL_STRING);
DECLARE_LHASH_OF(OPENSSL_CSTRING);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    3952 Wed Aug 13 19:51:45 2014
new/usr/src/lib/openssl/include/openssl/md2.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md/md2.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_MD2_H
#define HEADER_MD2_H
```



```
#include <openssl/opensslconf.h> /* OPENSSL_NO_MD2, MD2_INT */
#ifdef OPENSSL_NO_MD2
#error MD2 is disabled.
#endif
#include <stddef.h>

#define MD2_DIGEST_LENGTH       16
#define MD2_BLOCK               16

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct MD2state_st
        {
        unsigned int num;
        unsigned char data[MD2_BLOCK];
        MD2_INT cksm[MD2_BLOCK];
        MD2_INT state[MD2_BLOCK];
        } MD2_CTX;

const char *MD2_options(void);
#ifdef OPENSSL_FIPS
int private_MD2_Init(MD2_CTX *c);
#endif
int MD2_Init(MD2_CTX *c);
int MD2_Update(MD2_CTX *c, const unsigned char *data, size_t len);
int MD2_Final(unsigned char *md, MD2_CTX *c);
unsigned char *MD2(const unsigned char *d, size_t n,unsigned char *md);
#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    4686 Wed Aug 13 19:51:45 2014
new/usr/src/lib/openssl/include/openssl/md4.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md4/md4.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_MD4_H
#define HEADER_MD4_H
```



```
#include <openssl/e_os2.h>
#include <stddef.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef OPENSSL_NO_MD4
#error MD4 is disabled.
#endif

/*
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 * ! MD4_LONG has to be at least 32 bits wide. If it's wider, then !
 * ! MD4_LONG_LOG2 has to be defined along.                        !
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 */

#if defined(__LP32__)
#define MD4_LONG unsigned long
#elif defined(OPENSSL_SYS_CRAY) || defined(__ILP64__)
#define MD4_LONG unsigned long
#define MD4_LONG_LOG2 3
/*
 * _CRAY note. I could declare short, but I have no idea what impact
 * does it have on performance on none-T3E machines. I could declare
 * int, but at least on C90 sizeof(int) can be chosen at compile time.
 * So I've chosen long...
 *                                      <appro@fy.chalmers.se>
 */
#else
#define MD4_LONG unsigned int
#endif

#define MD4_CBLOCK      64
#define MD4_LBLOCK      (MD4_CBLOCK/4)
#define MD4_DIGEST_LENGTH 16

typedef struct MD4state_st
        {
        MD4_LONG A,B,C,D;
        MD4_LONG Nl,Nh;
        MD4_LONG data[MD4_LBLOCK];
        unsigned int num;
        } MD4_CTX;

#ifdef OPENSSL_FIPS
int private_MD4_Init(MD4_CTX *c);
#endif
int MD4_Init(MD4_CTX *c);
int MD4_Update(MD4_CTX *c, const void *data, size_t len);
int MD4_Final(unsigned char *md, MD4_CTX *c);
unsigned char *MD4(const unsigned char *d, size_t n, unsigned char *md);
void MD4_Transform(MD4_CTX *c, const unsigned char *b);
#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    4686 Wed Aug 13 19:51:45 2014
new/usr/src/lib/openssl/include/openssl/md5.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md5/md5.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_MD5_H
#define HEADER_MD5_H
```



```
#include <openssl/e_os2.h>
#include <stddef.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef OPENSSL_NO_MD5
#error MD5 is disabled.
#endif

/*
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 * ! MD5_LONG has to be at least 32 bits wide. If it's wider, then !
 * ! MD5_LONG_LOG2 has to be defined along.                        !
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 */

#if defined(__LP32__)
#define MD5_LONG unsigned long
#elif defined(OPENSSL_SYS_CRAY) || defined(__ILP64__)
#define MD5_LONG unsigned long
#define MD5_LONG_LOG2 3
/*
 * _CRAY note. I could declare short, but I have no idea what impact
 * does it have on performance on none-T3E machines. I could declare
 * int, but at least on C90 sizeof(int) can be chosen at compile time.
 * So I've chosen long...
 *                                      <appro@fy.chalmers.se>
 */
#else
#define MD5_LONG unsigned int
#endif

#define MD5_CBLOCK      64
#define MD5_LBLOCK      (MD5_CBLOCK/4)
#define MD5_DIGEST_LENGTH 16

typedef struct MD5state_st
        {
        MD5_LONG A,B,C,D;
        MD5_LONG Nl,Nh;
        MD5_LONG data[MD5_LBLOCK];
        unsigned int num;
        } MD5_CTX;

#ifdef OPENSSL_FIPS
int private_MD5_Init(MD5_CTX *c);
#endif
int MD5_Init(MD5_CTX *c);
int MD5_Update(MD5_CTX *c, const void *data, size_t len);
int MD5_Final(unsigned char *md, MD5_CTX *c);
unsigned char *MD5(const unsigned char *d, size_t n, unsigned char *md);
void MD5_Transform(MD5_CTX *c, const unsigned char *b);
#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    5619 Wed Aug 13 19:51:45 2014
new/usr/src/lib/openssl/include/openssl/modes.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project. All rights reserved.
 *
 * Rights for redistribution and usage in source and binary
 * forms are granted according to the OpenSSL license.
 */

#include <stddef.h>

typedef void (*block128_f)(const unsigned char in[16],
                        unsigned char out[16],
                        const void *key);

typedef void (*cbc128_f)(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], int enc);

typedef void (*ctr128_f)(const unsigned char *in, unsigned char *out,
                        size_t blocks, const void *key,
                        const unsigned char ivec[16]);

typedef void (*ccm128_f)(const unsigned char *in, unsigned char *out,
                        size_t blocks, const void *key,
                        const unsigned char ivec[16],unsigned char cmac[16]);

void CRYPTO_cbc128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block);
void CRYPTO_cbc128_decrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block);

void CRYPTO_ctr128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], unsigned char ecount_buf[16],
                        unsigned int *num, block128_f block);

void CRYPTO_ctr128_encrypt_ctr32(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], unsigned char ecount_buf[16],
                        unsigned int *num, ctr128_f ctr);

void CRYPTO_ofb128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], int *num,
                        block128_f block);

void CRYPTO_cfb128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], int *num,
                        int enc, block128_f block);
void CRYPTO_cfb128_8_encrypt(const unsigned char *in, unsigned char *out,
                        size_t length, const void *key,
                        unsigned char ivec[16], int *num,
                        int enc, block128_f block);
void CRYPTO_cfb128_1_encrypt(const unsigned char *in, unsigned char *out,
                        size_t bits, const void *key,
                        unsigned char ivec[16], int *num,
                        int enc, block128_f block);

size_t CRYPTO_cts128_encrypt_block(const unsigned char *in, unsigned char *out,
```



```
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block);
size_t CRYPTO_cts128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], cbc128_f cbc);
size_t CRYPTO_cts128_decrypt_block(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block);
size_t CRYPTO_cts128_decrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], cbc128_f cbc);

size_t CRYPTO_nistcts128_encrypt_block(const unsigned char *in, unsigned char *o
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block);
size_t CRYPTO_nistcts128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], cbc128_f cbc);
size_t CRYPTO_nistcts128_decrypt_block(const unsigned char *in, unsigned char *o
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block);
size_t CRYPTO_nistcts128_decrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], cbc128_f cbc);

typedef struct gcm128_context GCM128_CONTEXT;

GCM128_CONTEXT *CRYPTO_gcm128_new(void *key, block128_f block);
void CRYPTO_gcm128_init(GCM128_CONTEXT *ctx,void *key,block128_f block);
void CRYPTO_gcm128_setiv(GCM128_CONTEXT *ctx, const unsigned char *iv,
                        size_t len);
int CRYPTO_gcm128_aad(GCM128_CONTEXT *ctx, const unsigned char *aad,
                        size_t len);
int CRYPTO_gcm128_encrypt(GCM128_CONTEXT *ctx,
                        const unsigned char *in, unsigned char *out,
                        size_t len);
int CRYPTO_gcm128_decrypt(GCM128_CONTEXT *ctx,
                        const unsigned char *in, unsigned char *out,
                        size_t len);
int CRYPTO_gcm128_encrypt_ctr32(GCM128_CONTEXT *ctx,
                        const unsigned char *in, unsigned char *out,
                        size_t len, ctr128_f stream);
int CRYPTO_gcm128_decrypt_ctr32(GCM128_CONTEXT *ctx,
                        const unsigned char *in, unsigned char *out,
                        size_t len, ctr128_f stream);
int CRYPTO_gcm128_finish(GCM128_CONTEXT *ctx,const unsigned char *tag,
                        size_t len);
void CRYPTO_gcm128_tag(GCM128_CONTEXT *ctx, unsigned char *tag, size_t len);
void CRYPTO_gcm128_release(GCM128_CONTEXT *ctx);

typedef struct ccm128_context CCM128_CONTEXT;

void CRYPTO_ccm128_init(CCM128_CONTEXT *ctx,
        unsigned int M, unsigned int L, void *key,block128_f block);
int CRYPTO_ccm128_setiv(CCM128_CONTEXT *ctx,
        const unsigned char *nonce, size_t nlen, size_t mlen);
void CRYPTO_ccm128_aad(CCM128_CONTEXT *ctx,
        const unsigned char *aad, size_t alen);
int CRYPTO_ccm128_encrypt(CCM128_CONTEXT *ctx,
        const unsigned char *inp, unsigned char *out, size_t len);
int CRYPTO_ccm128_decrypt(CCM128_CONTEXT *ctx,
        const unsigned char *inp, unsigned char *out, size_t len);
int CRYPTO_ccm128_encrypt_ccm64(CCM128_CONTEXT *ctx,
        const unsigned char *inp, unsigned char *out, size_t len,
        ccm128_f stream);
int CRYPTO_ccm128_decrypt_ccm64(CCM128_CONTEXT *ctx,
```

```
128         const unsigned char *inp, unsigned char *out, size_t len,
129         ccm128_f stream);
130 size_t CRYPTO_ccm128_tag(CCM128_CONTEXT *ctx, unsigned char *tag, size_t len);

132 typedef struct xts128_context XTS128_CONTEXT;

134 int CRYPTO_xts128_encrypt(const XTS128_CONTEXT *ctx, const unsigned char iv[16],
135         const unsigned char *inp, unsigned char *out, size_t len, int enc);
136 #endif /* ! codereview */
```

```
**********************************************************
  136207 Wed Aug 13 19:51:45 2014
new/usr/src/lib/openssl/include/openssl/obj_mac.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/objects/obj_mac.h */

/* THIS FILE IS GENERATED FROM objects.txt by objects.pl via the
 * following command:
 * perl objects.pl objects.txt obj_mac.num obj_mac.h
 */

/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
```



```
 * [including the GNU Public Licence.]
 */

#define SN_undef                        "UNDEF"
#define LN_undef                        "undefined"
#define NID_undef                       0
#define OBJ_undef                       0L

#define SN_itu_t                "ITU-T"
#define LN_itu_t                "itu-t"
#define NID_itu_t               645
#define OBJ_itu_t               0L

#define NID_ccitt               404
#define OBJ_ccitt               OBJ_itu_t

#define SN_iso          "ISO"
#define LN_iso          "iso"
#define NID_iso         181
#define OBJ_iso         1L

#define SN_joint_iso_itu_t              "JOINT-ISO-ITU-T"
#define LN_joint_iso_itu_t              "joint-iso-itu-t"
#define NID_joint_iso_itu_t             646
#define OBJ_joint_iso_itu_t             2L

#define NID_joint_iso_ccitt             393
#define OBJ_joint_iso_ccitt             OBJ_joint_iso_itu_t

#define SN_member_body          "member-body"
#define LN_member_body          "ISO Member Body"
#define NID_member_body         182
#define OBJ_member_body         OBJ_iso,2L

#define SN_identified_organization              "identified-organization"
#define NID_identified_organization             676
#define OBJ_identified_organization             OBJ_iso,3L

#define SN_hmac_md5             "HMAC-MD5"
#define LN_hmac_md5             "hmac-md5"
#define NID_hmac_md5            780
#define OBJ_hmac_md5            OBJ_identified_organization,6L,1L,5L,5L,8L,1L,1L

#define SN_hmac_sha1            "HMAC-SHA1"
#define LN_hmac_sha1            "hmac-sha1"
#define NID_hmac_sha1           781
#define OBJ_hmac_sha1           OBJ_identified_organization,6L,1L,5L,5L,8L,1L,2L

#define SN_certicom_arc         "certicom-arc"
#define NID_certicom_arc                677
#define OBJ_certicom_arc                OBJ_identified_organization,132L

#define SN_international_organizations          "international-organizations"
#define LN_international_organizations          "International Organizations"
#define NID_international_organizations         647
#define OBJ_international_organizations         OBJ_joint_iso_itu_t,23L

#define SN_wap          "wap"
#define NID_wap         678
#define OBJ_wap         OBJ_international_organizations,43L

#define SN_wap_wsg              "wap-wsg"
#define NID_wap_wsg             679
#define OBJ_wap_wsg             OBJ_wap,1L

#define SN_selected_attribute_types             "selected-attribute-types"
```

```
#define LN_selected_attribute_types             "Selected Attribute Types"
#define NID_selected_attribute_types            394
#define OBJ_selected_attribute_types            OBJ_joint_iso_itu_t,5L,1L,5L

#define SN_clearance            "clearance"
#define NID_clearance           395
#define OBJ_clearance           OBJ_selected_attribute_types,55L

#define SN_ISO_US               "ISO-US"
#define LN_ISO_US               "ISO US Member Body"
#define NID_ISO_US              183
#define OBJ_ISO_US              OBJ_member_body,840L

#define SN_X9_57                "X9-57"
#define LN_X9_57                "X9.57"
#define NID_X9_57               184
#define OBJ_X9_57               OBJ_ISO_US,10040L

#define SN_X9cm         "X9cm"
#define LN_X9cm         "X9.57 CM ?"
#define NID_X9cm                185
#define OBJ_X9cm                OBJ_X9_57,4L

#define SN_dsa          "DSA"
#define LN_dsa          "dsaEncryption"
#define NID_dsa         116
#define OBJ_dsa         OBJ_X9cm,1L

#define SN_dsaWithSHA1          "DSA-SHA1"
#define LN_dsaWithSHA1          "dsaWithSHA1"
#define NID_dsaWithSHA1         113
#define OBJ_dsaWithSHA1         OBJ_X9cm,3L

#define SN_ansi_X9_62           "ansi-X9-62"
#define LN_ansi_X9_62           "ANSI X9.62"
#define NID_ansi_X9_62          405
#define OBJ_ansi_X9_62          OBJ_ISO_US,10045L

#define OBJ_X9_62_id_fieldType          OBJ_ansi_X9_62,1L

#define SN_X9_62_prime_field            "prime-field"
#define NID_X9_62_prime_field           406
#define OBJ_X9_62_prime_field           OBJ_X9_62_id_fieldType,1L

#define SN_X9_62_characteristic_two_field               "characteristic-two-fiel
#define NID_X9_62_characteristic_two_field              407
#define OBJ_X9_62_characteristic_two_field              OBJ_X9_62_id_fieldType,2

#define SN_X9_62_id_characteristic_two_basis            "id-characteristic-two-b
#define NID_X9_62_id_characteristic_two_basis           680
#define OBJ_X9_62_id_characteristic_two_basis           OBJ_X9_62_characteristic

#define SN_X9_62_onBasis                "onBasis"
#define NID_X9_62_onBasis               681
#define OBJ_X9_62_onBasis               OBJ_X9_62_id_characteristic_two_basis,1L

#define SN_X9_62_tpBasis                "tpBasis"
#define NID_X9_62_tpBasis               682
#define OBJ_X9_62_tpBasis               OBJ_X9_62_id_characteristic_two_basis,2L

#define SN_X9_62_ppBasis                "ppBasis"
#define NID_X9_62_ppBasis               683
#define OBJ_X9_62_ppBasis               OBJ_X9_62_id_characteristic_two_basis,3L

#define OBJ_X9_62_id_publicKeyType              OBJ_ansi_X9_62,2L
```



```
#define SN_X9_62_id_ecPublicKey         "id-ecPublicKey"
#define NID_X9_62_id_ecPublicKey                408
#define OBJ_X9_62_id_ecPublicKey                OBJ_X9_62_id_publicKeyType,1L

#define OBJ_X9_62_ellipticCurve         OBJ_ansi_X9_62,3L

#define OBJ_X9_62_c_TwoCurve            OBJ_X9_62_ellipticCurve,0L

#define SN_X9_62_c2pnb163v1             "c2pnb163v1"
#define NID_X9_62_c2pnb163v1            684
#define OBJ_X9_62_c2pnb163v1            OBJ_X9_62_c_TwoCurve,1L

#define SN_X9_62_c2pnb163v2             "c2pnb163v2"
#define NID_X9_62_c2pnb163v2            685
#define OBJ_X9_62_c2pnb163v2            OBJ_X9_62_c_TwoCurve,2L

#define SN_X9_62_c2pnb163v3             "c2pnb163v3"
#define NID_X9_62_c2pnb163v3            686
#define OBJ_X9_62_c2pnb163v3            OBJ_X9_62_c_TwoCurve,3L

#define SN_X9_62_c2pnb176v1             "c2pnb176v1"
#define NID_X9_62_c2pnb176v1            687
#define OBJ_X9_62_c2pnb176v1            OBJ_X9_62_c_TwoCurve,4L

#define SN_X9_62_c2tnb191v1             "c2tnb191v1"
#define NID_X9_62_c2tnb191v1            688
#define OBJ_X9_62_c2tnb191v1            OBJ_X9_62_c_TwoCurve,5L

#define SN_X9_62_c2tnb191v2             "c2tnb191v2"
#define NID_X9_62_c2tnb191v2            689
#define OBJ_X9_62_c2tnb191v2            OBJ_X9_62_c_TwoCurve,6L

#define SN_X9_62_c2tnb191v3             "c2tnb191v3"
#define NID_X9_62_c2tnb191v3            690
#define OBJ_X9_62_c2tnb191v3            OBJ_X9_62_c_TwoCurve,7L

#define SN_X9_62_c2onb191v4             "c2onb191v4"
#define NID_X9_62_c2onb191v4            691
#define OBJ_X9_62_c2onb191v4            OBJ_X9_62_c_TwoCurve,8L

#define SN_X9_62_c2onb191v5             "c2onb191v5"
#define NID_X9_62_c2onb191v5            692
#define OBJ_X9_62_c2onb191v5            OBJ_X9_62_c_TwoCurve,9L

#define SN_X9_62_c2pnb208w1             "c2pnb208w1"
#define NID_X9_62_c2pnb208w1            693
#define OBJ_X9_62_c2pnb208w1            OBJ_X9_62_c_TwoCurve,10L

#define SN_X9_62_c2tnb239v1             "c2tnb239v1"
#define NID_X9_62_c2tnb239v1            694
#define OBJ_X9_62_c2tnb239v1            OBJ_X9_62_c_TwoCurve,11L

#define SN_X9_62_c2tnb239v2             "c2tnb239v2"
#define NID_X9_62_c2tnb239v2            695
#define OBJ_X9_62_c2tnb239v2            OBJ_X9_62_c_TwoCurve,12L

#define SN_X9_62_c2tnb239v3             "c2tnb239v3"
#define NID_X9_62_c2tnb239v3            696
#define OBJ_X9_62_c2tnb239v3            OBJ_X9_62_c_TwoCurve,13L

#define SN_X9_62_c2onb239v4             "c2onb239v4"
#define NID_X9_62_c2onb239v4            697
#define OBJ_X9_62_c2onb239v4            OBJ_X9_62_c_TwoCurve,14L

#define SN_X9_62_c2onb239v5             "c2onb239v5"
#define NID_X9_62_c2onb239v5            698
```

```
260 #define OBJ_X9_62_c2onb239v5             OBJ_X9_62_c_TwoCurve,15L

262 #define SN_X9_62_c2pnb272w1              "c2pnb272w1"
263 #define NID_X9_62_c2pnb272w1             699
264 #define OBJ_X9_62_c2pnb272w1             OBJ_X9_62_c_TwoCurve,16L

266 #define SN_X9_62_c2pnb304w1              "c2pnb304w1"
267 #define NID_X9_62_c2pnb304w1             700
268 #define OBJ_X9_62_c2pnb304w1             OBJ_X9_62_c_TwoCurve,17L

270 #define SN_X9_62_c2tnb359v1              "c2tnb359v1"
271 #define NID_X9_62_c2tnb359v1             701
272 #define OBJ_X9_62_c2tnb359v1             OBJ_X9_62_c_TwoCurve,18L

274 #define SN_X9_62_c2pnb368w1              "c2pnb368w1"
275 #define NID_X9_62_c2pnb368w1             702
276 #define OBJ_X9_62_c2pnb368w1             OBJ_X9_62_c_TwoCurve,19L

278 #define SN_X9_62_c2tnb431r1              "c2tnb431r1"
279 #define NID_X9_62_c2tnb431r1             703
280 #define OBJ_X9_62_c2tnb431r1             OBJ_X9_62_c_TwoCurve,20L

282 #define OBJ_X9_62_primeCurve             OBJ_X9_62_ellipticCurve,1L

284 #define SN_X9_62_prime192v1              "prime192v1"
285 #define NID_X9_62_prime192v1             409
286 #define OBJ_X9_62_prime192v1             OBJ_X9_62_primeCurve,1L

288 #define SN_X9_62_prime192v2              "prime192v2"
289 #define NID_X9_62_prime192v2             410
290 #define OBJ_X9_62_prime192v2             OBJ_X9_62_primeCurve,2L

292 #define SN_X9_62_prime192v3              "prime192v3"
293 #define NID_X9_62_prime192v3             411
294 #define OBJ_X9_62_prime192v3             OBJ_X9_62_primeCurve,3L

296 #define SN_X9_62_prime239v1              "prime239v1"
297 #define NID_X9_62_prime239v1             412
298 #define OBJ_X9_62_prime239v1             OBJ_X9_62_primeCurve,4L

300 #define SN_X9_62_prime239v2              "prime239v2"
301 #define NID_X9_62_prime239v2             413
302 #define OBJ_X9_62_prime239v2             OBJ_X9_62_primeCurve,5L

304 #define SN_X9_62_prime239v3              "prime239v3"
305 #define NID_X9_62_prime239v3             414
306 #define OBJ_X9_62_prime239v3             OBJ_X9_62_primeCurve,6L

308 #define SN_X9_62_prime256v1              "prime256v1"
309 #define NID_X9_62_prime256v1             415
310 #define OBJ_X9_62_prime256v1             OBJ_X9_62_primeCurve,7L

312 #define OBJ_X9_62_id_ecSigType           OBJ_ansi_X9_62,4L

314 #define SN_ecdsa_with_SHA1               "ecdsa-with-SHA1"
315 #define NID_ecdsa_with_SHA1              416
316 #define OBJ_ecdsa_with_SHA1              OBJ_X9_62_id_ecSigType,1L

318 #define SN_ecdsa_with_Recommended                "ecdsa-with-Recommended"
319 #define NID_ecdsa_with_Recommended               791
320 #define OBJ_ecdsa_with_Recommended               OBJ_X9_62_id_ecSigType,2L

322 #define SN_ecdsa_with_Specified          "ecdsa-with-Specified"
323 #define NID_ecdsa_with_Specified                 792
324 #define OBJ_ecdsa_with_Specified                 OBJ_X9_62_id_ecSigType,3L
```



```
326 #define SN_ecdsa_with_SHA224             "ecdsa-with-SHA224"
327 #define NID_ecdsa_with_SHA224            793
328 #define OBJ_ecdsa_with_SHA224            OBJ_ecdsa_with_Specified,1L

330 #define SN_ecdsa_with_SHA256             "ecdsa-with-SHA256"
331 #define NID_ecdsa_with_SHA256            794
332 #define OBJ_ecdsa_with_SHA256            OBJ_ecdsa_with_Specified,2L

334 #define SN_ecdsa_with_SHA384             "ecdsa-with-SHA384"
335 #define NID_ecdsa_with_SHA384            795
336 #define OBJ_ecdsa_with_SHA384            OBJ_ecdsa_with_Specified,3L

338 #define SN_ecdsa_with_SHA512             "ecdsa-with-SHA512"
339 #define NID_ecdsa_with_SHA512            796
340 #define OBJ_ecdsa_with_SHA512            OBJ_ecdsa_with_Specified,4L

342 #define OBJ_secg_ellipticCurve           OBJ_certicom_arc,0L

344 #define SN_secp112r1             "secp112r1"
345 #define NID_secp112r1            704
346 #define OBJ_secp112r1            OBJ_secg_ellipticCurve,6L

348 #define SN_secp112r2             "secp112r2"
349 #define NID_secp112r2            705
350 #define OBJ_secp112r2            OBJ_secg_ellipticCurve,7L

352 #define SN_secp128r1             "secp128r1"
353 #define NID_secp128r1            706
354 #define OBJ_secp128r1            OBJ_secg_ellipticCurve,28L

356 #define SN_secp128r2             "secp128r2"
357 #define NID_secp128r2            707
358 #define OBJ_secp128r2            OBJ_secg_ellipticCurve,29L

360 #define SN_secp160k1             "secp160k1"
361 #define NID_secp160k1            708
362 #define OBJ_secp160k1            OBJ_secg_ellipticCurve,9L

364 #define SN_secp160r1             "secp160r1"
365 #define NID_secp160r1            709
366 #define OBJ_secp160r1            OBJ_secg_ellipticCurve,8L

368 #define SN_secp160r2             "secp160r2"
369 #define NID_secp160r2            710
370 #define OBJ_secp160r2            OBJ_secg_ellipticCurve,30L

372 #define SN_secp192k1             "secp192k1"
373 #define NID_secp192k1            711
374 #define OBJ_secp192k1            OBJ_secg_ellipticCurve,31L

376 #define SN_secp224k1             "secp224k1"
377 #define NID_secp224k1            712
378 #define OBJ_secp224k1            OBJ_secg_ellipticCurve,32L

380 #define SN_secp224r1             "secp224r1"
381 #define NID_secp224r1            713
382 #define OBJ_secp224r1            OBJ_secg_ellipticCurve,33L

384 #define SN_secp256k1             "secp256k1"
385 #define NID_secp256k1            714
386 #define OBJ_secp256k1            OBJ_secg_ellipticCurve,10L

388 #define SN_secp384r1             "secp384r1"
389 #define NID_secp384r1            715
390 #define OBJ_secp384r1            OBJ_secg_ellipticCurve,34L
```

```
392 #define SN_secp521r1            "secp521r1"
393 #define NID_secp521r1           716
394 #define OBJ_secp521r1           OBJ_secg_ellipticCurve,35L

396 #define SN_sect113r1            "sect113r1"
397 #define NID_sect113r1           717
398 #define OBJ_sect113r1           OBJ_secg_ellipticCurve,4L

400 #define SN_sect113r2            "sect113r2"
401 #define NID_sect113r2           718
402 #define OBJ_sect113r2           OBJ_secg_ellipticCurve,5L

404 #define SN_sect131r1            "sect131r1"
405 #define NID_sect131r1           719
406 #define OBJ_sect131r1           OBJ_secg_ellipticCurve,22L

408 #define SN_sect131r2            "sect131r2"
409 #define NID_sect131r2           720
410 #define OBJ_sect131r2           OBJ_secg_ellipticCurve,23L

412 #define SN_sect163k1            "sect163k1"
413 #define NID_sect163k1           721
414 #define OBJ_sect163k1           OBJ_secg_ellipticCurve,1L

416 #define SN_sect163r1            "sect163r1"
417 #define NID_sect163r1           722
418 #define OBJ_sect163r1           OBJ_secg_ellipticCurve,2L

420 #define SN_sect163r2            "sect163r2"
421 #define NID_sect163r2           723
422 #define OBJ_sect163r2           OBJ_secg_ellipticCurve,15L

424 #define SN_sect193r1            "sect193r1"
425 #define NID_sect193r1           724
426 #define OBJ_sect193r1           OBJ_secg_ellipticCurve,24L

428 #define SN_sect193r2            "sect193r2"
429 #define NID_sect193r2           725
430 #define OBJ_sect193r2           OBJ_secg_ellipticCurve,25L

432 #define SN_sect233k1            "sect233k1"
433 #define NID_sect233k1           726
434 #define OBJ_sect233k1           OBJ_secg_ellipticCurve,26L

436 #define SN_sect233r1            "sect233r1"
437 #define NID_sect233r1           727
438 #define OBJ_sect233r1           OBJ_secg_ellipticCurve,27L

440 #define SN_sect239k1            "sect239k1"
441 #define NID_sect239k1           728
442 #define OBJ_sect239k1           OBJ_secg_ellipticCurve,3L

444 #define SN_sect283k1            "sect283k1"
445 #define NID_sect283k1           729
446 #define OBJ_sect283k1           OBJ_secg_ellipticCurve,16L

448 #define SN_sect283r1            "sect283r1"
449 #define NID_sect283r1           730
450 #define OBJ_sect283r1           OBJ_secg_ellipticCurve,17L

452 #define SN_sect409k1            "sect409k1"
453 #define NID_sect409k1           731
454 #define OBJ_sect409k1           OBJ_secg_ellipticCurve,36L

456 #define SN_sect409r1            "sect409r1"
457 #define NID_sect409r1           732
```



```
458 #define OBJ_sect409r1           OBJ_secg_ellipticCurve,37L

460 #define SN_sect571k1            "sect571k1"
461 #define NID_sect571k1           733
462 #define OBJ_sect571k1           OBJ_secg_ellipticCurve,38L

464 #define SN_sect571r1            "sect571r1"
465 #define NID_sect571r1           734
466 #define OBJ_sect571r1           OBJ_secg_ellipticCurve,39L

468 #define OBJ_wap_wsg_idm_ecid            OBJ_wap_wsg,4L

470 #define SN_wap_wsg_idm_ecid_wtls1               "wap-wsg-idm-ecid-wtls1"
471 #define NID_wap_wsg_idm_ecid_wtls1              735
472 #define OBJ_wap_wsg_idm_ecid_wtls1              OBJ_wap_wsg_idm_ecid,1L

474 #define SN_wap_wsg_idm_ecid_wtls3               "wap-wsg-idm-ecid-wtls3"
475 #define NID_wap_wsg_idm_ecid_wtls3              736
476 #define OBJ_wap_wsg_idm_ecid_wtls3              OBJ_wap_wsg_idm_ecid,3L

478 #define SN_wap_wsg_idm_ecid_wtls4               "wap-wsg-idm-ecid-wtls4"
479 #define NID_wap_wsg_idm_ecid_wtls4              737
480 #define OBJ_wap_wsg_idm_ecid_wtls4              OBJ_wap_wsg_idm_ecid,4L

482 #define SN_wap_wsg_idm_ecid_wtls5               "wap-wsg-idm-ecid-wtls5"
483 #define NID_wap_wsg_idm_ecid_wtls5              738
484 #define OBJ_wap_wsg_idm_ecid_wtls5              OBJ_wap_wsg_idm_ecid,5L

486 #define SN_wap_wsg_idm_ecid_wtls6               "wap-wsg-idm-ecid-wtls6"
487 #define NID_wap_wsg_idm_ecid_wtls6              739
488 #define OBJ_wap_wsg_idm_ecid_wtls6              OBJ_wap_wsg_idm_ecid,6L

490 #define SN_wap_wsg_idm_ecid_wtls7               "wap-wsg-idm-ecid-wtls7"
491 #define NID_wap_wsg_idm_ecid_wtls7              740
492 #define OBJ_wap_wsg_idm_ecid_wtls7              OBJ_wap_wsg_idm_ecid,7L

494 #define SN_wap_wsg_idm_ecid_wtls8               "wap-wsg-idm-ecid-wtls8"
495 #define NID_wap_wsg_idm_ecid_wtls8              741
496 #define OBJ_wap_wsg_idm_ecid_wtls8              OBJ_wap_wsg_idm_ecid,8L

498 #define SN_wap_wsg_idm_ecid_wtls9               "wap-wsg-idm-ecid-wtls9"
499 #define NID_wap_wsg_idm_ecid_wtls9              742
500 #define OBJ_wap_wsg_idm_ecid_wtls9              OBJ_wap_wsg_idm_ecid,9L

502 #define SN_wap_wsg_idm_ecid_wtls10              "wap-wsg-idm-ecid-wtls10"
503 #define NID_wap_wsg_idm_ecid_wtls10             743
504 #define OBJ_wap_wsg_idm_ecid_wtls10             OBJ_wap_wsg_idm_ecid,10L

506 #define SN_wap_wsg_idm_ecid_wtls11              "wap-wsg-idm-ecid-wtls11"
507 #define NID_wap_wsg_idm_ecid_wtls11             744
508 #define OBJ_wap_wsg_idm_ecid_wtls11             OBJ_wap_wsg_idm_ecid,11L

510 #define SN_wap_wsg_idm_ecid_wtls12              "wap-wsg-idm-ecid-wtls12"
511 #define NID_wap_wsg_idm_ecid_wtls12             745
512 #define OBJ_wap_wsg_idm_ecid_wtls12             OBJ_wap_wsg_idm_ecid,12L

514 #define SN_cast5_cbc            "CAST5-CBC"
515 #define LN_cast5_cbc            "cast5-cbc"
516 #define NID_cast5_cbc           108
517 #define OBJ_cast5_cbc           OBJ_ISO_US,113533L,7L,66L,10L

519 #define SN_cast5_ecb            "CAST5-ECB"
520 #define LN_cast5_ecb            "cast5-ecb"
521 #define NID_cast5_ecb           109

523 #define SN_cast5_cfb64          "CAST5-CFB"
```

```
#define LN_cast5_cfb64          "cast5-cfb"
#define NID_cast5_cfb64         110

#define SN_cast5_ofb64          "CAST5-OFB"
#define LN_cast5_ofb64          "cast5-ofb"
#define NID_cast5_ofb64         111

#define LN_pbeWithMD5AndCast5_CBC               "pbeWithMD5AndCast5CBC"
#define NID_pbeWithMD5AndCast5_CBC              112
#define OBJ_pbeWithMD5AndCast5_CBC              OBJ_ISO_US,113533L,7L,66L,12L

#define SN_id_PasswordBasedMAC          "id-PasswordBasedMAC"
#define LN_id_PasswordBasedMAC          "password based MAC"
#define NID_id_PasswordBasedMAC         782
#define OBJ_id_PasswordBasedMAC         OBJ_ISO_US,113533L,7L,66L,13L

#define SN_id_DHBasedMac                "id-DHBasedMac"
#define LN_id_DHBasedMac                "Diffie-Hellman based MAC"
#define NID_id_DHBasedMac               783
#define OBJ_id_DHBasedMac               OBJ_ISO_US,113533L,7L,66L,30L

#define SN_rsadsi               "rsadsi"
#define LN_rsadsi               "RSA Data Security, Inc."
#define NID_rsadsi              1
#define OBJ_rsadsi              OBJ_ISO_US,113549L

#define SN_pkcs         "pkcs"
#define LN_pkcs         "RSA Data Security, Inc. PKCS"
#define NID_pkcs                2
#define OBJ_pkcs                OBJ_rsadsi,1L

#define SN_pkcs1                "pkcs1"
#define NID_pkcs1               186
#define OBJ_pkcs1               OBJ_pkcs,1L

#define LN_rsaEncryption                "rsaEncryption"
#define NID_rsaEncryption               6
#define OBJ_rsaEncryption               OBJ_pkcs1,1L

#define SN_md2WithRSAEncryption         "RSA-MD2"
#define LN_md2WithRSAEncryption         "md2WithRSAEncryption"
#define NID_md2WithRSAEncryption                7
#define OBJ_md2WithRSAEncryption                OBJ_pkcs1,2L

#define SN_md4WithRSAEncryption         "RSA-MD4"
#define LN_md4WithRSAEncryption         "md4WithRSAEncryption"
#define NID_md4WithRSAEncryption                396
#define OBJ_md4WithRSAEncryption                OBJ_pkcs1,3L

#define SN_md5WithRSAEncryption         "RSA-MD5"
#define LN_md5WithRSAEncryption         "md5WithRSAEncryption"
#define NID_md5WithRSAEncryption                8
#define OBJ_md5WithRSAEncryption                OBJ_pkcs1,4L

#define SN_sha1WithRSAEncryption                "RSA-SHA1"
#define LN_sha1WithRSAEncryption                "sha1WithRSAEncryption"
#define NID_sha1WithRSAEncryption               65
#define OBJ_sha1WithRSAEncryption               OBJ_pkcs1,5L

#define SN_rsaesOaep            "RSAES-OAEP"
#define LN_rsaesOaep            "rsaesOaep"
#define NID_rsaesOaep           919
#define OBJ_rsaesOaep           OBJ_pkcs1,7L

#define SN_mgf1         "MGF1"
#define LN_mgf1         "mgf1"
```



```
#define NID_mgf1                911
#define OBJ_mgf1                OBJ_pkcs1,8L

#define SN_rsassaPss            "RSASSA-PSS"
#define LN_rsassaPss            "rsassaPss"
#define NID_rsassaPss           912
#define OBJ_rsassaPss           OBJ_pkcs1,10L

#define SN_sha256WithRSAEncryption              "RSA-SHA256"
#define LN_sha256WithRSAEncryption              "sha256WithRSAEncryption"
#define NID_sha256WithRSAEncryption             668
#define OBJ_sha256WithRSAEncryption             OBJ_pkcs1,11L

#define SN_sha384WithRSAEncryption              "RSA-SHA384"
#define LN_sha384WithRSAEncryption              "sha384WithRSAEncryption"
#define NID_sha384WithRSAEncryption             669
#define OBJ_sha384WithRSAEncryption             OBJ_pkcs1,12L

#define SN_sha512WithRSAEncryption              "RSA-SHA512"
#define LN_sha512WithRSAEncryption              "sha512WithRSAEncryption"
#define NID_sha512WithRSAEncryption             670
#define OBJ_sha512WithRSAEncryption             OBJ_pkcs1,13L

#define SN_sha224WithRSAEncryption              "RSA-SHA224"
#define LN_sha224WithRSAEncryption              "sha224WithRSAEncryption"
#define NID_sha224WithRSAEncryption             671
#define OBJ_sha224WithRSAEncryption             OBJ_pkcs1,14L

#define SN_pkcs3                "pkcs3"
#define NID_pkcs3               27
#define OBJ_pkcs3               OBJ_pkcs,3L

#define LN_dhKeyAgreement               "dhKeyAgreement"
#define NID_dhKeyAgreement              28
#define OBJ_dhKeyAgreement              OBJ_pkcs3,1L

#define SN_pkcs5                "pkcs5"
#define NID_pkcs5               187
#define OBJ_pkcs5               OBJ_pkcs,5L

#define SN_pbeWithMD2AndDES_CBC         "PBE-MD2-DES"
#define LN_pbeWithMD2AndDES_CBC         "pbeWithMD2AndDES-CBC"
#define NID_pbeWithMD2AndDES_CBC                9
#define OBJ_pbeWithMD2AndDES_CBC                OBJ_pkcs5,1L

#define SN_pbeWithMD5AndDES_CBC         "PBE-MD5-DES"
#define LN_pbeWithMD5AndDES_CBC         "pbeWithMD5AndDES-CBC"
#define NID_pbeWithMD5AndDES_CBC                10
#define OBJ_pbeWithMD5AndDES_CBC                OBJ_pkcs5,3L

#define SN_pbeWithMD2AndRC2_CBC         "PBE-MD2-RC2-64"
#define LN_pbeWithMD2AndRC2_CBC         "pbeWithMD2AndRC2-CBC"
#define NID_pbeWithMD2AndRC2_CBC                168
#define OBJ_pbeWithMD2AndRC2_CBC                OBJ_pkcs5,4L

#define SN_pbeWithMD5AndRC2_CBC         "PBE-MD5-RC2-64"
#define LN_pbeWithMD5AndRC2_CBC         "pbeWithMD5AndRC2-CBC"
#define NID_pbeWithMD5AndRC2_CBC                169
#define OBJ_pbeWithMD5AndRC2_CBC                OBJ_pkcs5,6L

#define SN_pbeWithSHA1AndDES_CBC                "PBE-SHA1-DES"
#define LN_pbeWithSHA1AndDES_CBC                "pbeWithSHA1AndDES-CBC"
#define NID_pbeWithSHA1AndDES_CBC               170
#define OBJ_pbeWithSHA1AndDES_CBC               OBJ_pkcs5,10L

#define SN_pbeWithSHA1AndRC2_CBC                "PBE-SHA1-RC2-64"
```

```
656 #define LN_pbeWithSHA1AndRC2_CBC                "pbeWithSHA1AndRC2-CBC"
657 #define NID_pbeWithSHA1AndRC2_CBC               68
658 #define OBJ_pbeWithSHA1AndRC2_CBC               OBJ_pkcs5,11L

660 #define LN_id_pbkdf2            "PBKDF2"
661 #define NID_id_pbkdf2           69
662 #define OBJ_id_pbkdf2           OBJ_pkcs5,12L

664 #define LN_pbes2                "PBES2"
665 #define NID_pbes2               161
666 #define OBJ_pbes2               OBJ_pkcs5,13L

668 #define LN_pbmac1               "PBMAC1"
669 #define NID_pbmac1              162
670 #define OBJ_pbmac1              OBJ_pkcs5,14L

672 #define SN_pkcs7                "pkcs7"
673 #define NID_pkcs7               20
674 #define OBJ_pkcs7               OBJ_pkcs,7L

676 #define LN_pkcs7_data           "pkcs7-data"
677 #define NID_pkcs7_data          21
678 #define OBJ_pkcs7_data          OBJ_pkcs7,1L

680 #define LN_pkcs7_signed         "pkcs7-signedData"
681 #define NID_pkcs7_signed                22
682 #define OBJ_pkcs7_signed                OBJ_pkcs7,2L

684 #define LN_pkcs7_enveloped              "pkcs7-envelopedData"
685 #define NID_pkcs7_enveloped             23
686 #define OBJ_pkcs7_enveloped             OBJ_pkcs7,3L

688 #define LN_pkcs7_signedAndEnveloped             "pkcs7-signedAndEnvelopedData"
689 #define NID_pkcs7_signedAndEnveloped            24
690 #define OBJ_pkcs7_signedAndEnveloped            OBJ_pkcs7,4L

692 #define LN_pkcs7_digest         "pkcs7-digestData"
693 #define NID_pkcs7_digest                25
694 #define OBJ_pkcs7_digest                OBJ_pkcs7,5L

696 #define LN_pkcs7_encrypted              "pkcs7-encryptedData"
697 #define NID_pkcs7_encrypted             26
698 #define OBJ_pkcs7_encrypted             OBJ_pkcs7,6L

700 #define SN_pkcs9                "pkcs9"
701 #define NID_pkcs9               47
702 #define OBJ_pkcs9               OBJ_pkcs,9L

704 #define LN_pkcs9_emailAddress           "emailAddress"
705 #define NID_pkcs9_emailAddress          48
706 #define OBJ_pkcs9_emailAddress          OBJ_pkcs9,1L

708 #define LN_pkcs9_unstructuredName               "unstructuredName"
709 #define NID_pkcs9_unstructuredName              49
710 #define OBJ_pkcs9_unstructuredName              OBJ_pkcs9,2L

712 #define LN_pkcs9_contentType            "contentType"
713 #define NID_pkcs9_contentType           50
714 #define OBJ_pkcs9_contentType           OBJ_pkcs9,3L

716 #define LN_pkcs9_messageDigest          "messageDigest"
717 #define NID_pkcs9_messageDigest         51
718 #define OBJ_pkcs9_messageDigest         OBJ_pkcs9,4L

720 #define LN_pkcs9_signingTime            "signingTime"
721 #define NID_pkcs9_signingTime           52
```



```
722 #define OBJ_pkcs9_signingTime           OBJ_pkcs9,5L

724 #define LN_pkcs9_countersignature               "countersignature"
725 #define NID_pkcs9_countersignature              53
726 #define OBJ_pkcs9_countersignature              OBJ_pkcs9,6L

728 #define LN_pkcs9_challengePassword              "challengePassword"
729 #define NID_pkcs9_challengePassword             54
730 #define OBJ_pkcs9_challengePassword             OBJ_pkcs9,7L

732 #define LN_pkcs9_unstructuredAddress            "unstructuredAddress"
733 #define NID_pkcs9_unstructuredAddress           55
734 #define OBJ_pkcs9_unstructuredAddress           OBJ_pkcs9,8L

736 #define LN_pkcs9_extCertAttributes              "extendedCertificateAttributes"
737 #define NID_pkcs9_extCertAttributes             56
738 #define OBJ_pkcs9_extCertAttributes             OBJ_pkcs9,9L

740 #define SN_ext_req              "extReq"
741 #define LN_ext_req              "Extension Request"
742 #define NID_ext_req             172
743 #define OBJ_ext_req             OBJ_pkcs9,14L

745 #define SN_SMIMECapabilities            "SMIME-CAPS"
746 #define LN_SMIMECapabilities            "S/MIME Capabilities"
747 #define NID_SMIMECapabilities           167
748 #define OBJ_SMIMECapabilities           OBJ_pkcs9,15L

750 #define SN_SMIME                "SMIME"
751 #define LN_SMIME                "S/MIME"
752 #define NID_SMIME               188
753 #define OBJ_SMIME               OBJ_pkcs9,16L

755 #define SN_id_smime_mod         "id-smime-mod"
756 #define NID_id_smime_mod                189
757 #define OBJ_id_smime_mod                OBJ_SMIME,0L

759 #define SN_id_smime_ct          "id-smime-ct"
760 #define NID_id_smime_ct         190
761 #define OBJ_id_smime_ct         OBJ_SMIME,1L

763 #define SN_id_smime_aa          "id-smime-aa"
764 #define NID_id_smime_aa         191
765 #define OBJ_id_smime_aa         OBJ_SMIME,2L

767 #define SN_id_smime_alg         "id-smime-alg"
768 #define NID_id_smime_alg                192
769 #define OBJ_id_smime_alg                OBJ_SMIME,3L

771 #define SN_id_smime_cd          "id-smime-cd"
772 #define NID_id_smime_cd         193
773 #define OBJ_id_smime_cd         OBJ_SMIME,4L

775 #define SN_id_smime_spq         "id-smime-spq"
776 #define NID_id_smime_spq                194
777 #define OBJ_id_smime_spq                OBJ_SMIME,5L

779 #define SN_id_smime_cti         "id-smime-cti"
780 #define NID_id_smime_cti                195
781 #define OBJ_id_smime_cti                OBJ_SMIME,6L

783 #define SN_id_smime_mod_cms             "id-smime-mod-cms"
784 #define NID_id_smime_mod_cms            196
785 #define OBJ_id_smime_mod_cms            OBJ_id_smime_mod,1L

787 #define SN_id_smime_mod_ess             "id-smime-mod-ess"
```

```
#define NID_id_smime_mod_ess            197
#define OBJ_id_smime_mod_ess            OBJ_id_smime_mod,2L

#define SN_id_smime_mod_oid             "id-smime-mod-oid"
#define NID_id_smime_mod_oid            198
#define OBJ_id_smime_mod_oid            OBJ_id_smime_mod,3L

#define SN_id_smime_mod_msg_v3          "id-smime-mod-msg-v3"
#define NID_id_smime_mod_msg_v3         199
#define OBJ_id_smime_mod_msg_v3         OBJ_id_smime_mod,4L

#define SN_id_smime_mod_ets_eSignature_88               "id-smime-mod-ets-eSigna
#define NID_id_smime_mod_ets_eSignature_88              200
#define OBJ_id_smime_mod_ets_eSignature_88              OBJ_id_smime_mod,5L

#define SN_id_smime_mod_ets_eSignature_97               "id-smime-mod-ets-eSigna
#define NID_id_smime_mod_ets_eSignature_97              201
#define OBJ_id_smime_mod_ets_eSignature_97              OBJ_id_smime_mod,6L

#define SN_id_smime_mod_ets_eSigPolicy_88               "id-smime-mod-ets-eSigPo
#define NID_id_smime_mod_ets_eSigPolicy_88              202
#define OBJ_id_smime_mod_ets_eSigPolicy_88              OBJ_id_smime_mod,7L

#define SN_id_smime_mod_ets_eSigPolicy_97               "id-smime-mod-ets-eSigPo
#define NID_id_smime_mod_ets_eSigPolicy_97              203
#define OBJ_id_smime_mod_ets_eSigPolicy_97              OBJ_id_smime_mod,8L

#define SN_id_smime_ct_receipt          "id-smime-ct-receipt"
#define NID_id_smime_ct_receipt         204
#define OBJ_id_smime_ct_receipt         OBJ_id_smime_ct,1L

#define SN_id_smime_ct_authData         "id-smime-ct-authData"
#define NID_id_smime_ct_authData                205
#define OBJ_id_smime_ct_authData                OBJ_id_smime_ct,2L

#define SN_id_smime_ct_publishCert              "id-smime-ct-publishCert"
#define NID_id_smime_ct_publishCert             206
#define OBJ_id_smime_ct_publishCert             OBJ_id_smime_ct,3L

#define SN_id_smime_ct_TSTInfo          "id-smime-ct-TSTInfo"
#define NID_id_smime_ct_TSTInfo         207
#define OBJ_id_smime_ct_TSTInfo         OBJ_id_smime_ct,4L

#define SN_id_smime_ct_TDTInfo          "id-smime-ct-TDTInfo"
#define NID_id_smime_ct_TDTInfo         208
#define OBJ_id_smime_ct_TDTInfo         OBJ_id_smime_ct,5L

#define SN_id_smime_ct_contentInfo              "id-smime-ct-contentInfo"
#define NID_id_smime_ct_contentInfo             209
#define OBJ_id_smime_ct_contentInfo             OBJ_id_smime_ct,6L

#define SN_id_smime_ct_DVCSRequestData          "id-smime-ct-DVCSRequestData"
#define NID_id_smime_ct_DVCSRequestData         210
#define OBJ_id_smime_ct_DVCSRequestData         OBJ_id_smime_ct,7L

#define SN_id_smime_ct_DVCSResponseData         "id-smime-ct-DVCSResponseData"
#define NID_id_smime_ct_DVCSResponseData                211
#define OBJ_id_smime_ct_DVCSResponseData                OBJ_id_smime_ct,8L

#define SN_id_smime_ct_compressedData           "id-smime-ct-compressedData"
#define NID_id_smime_ct_compressedData          786
#define OBJ_id_smime_ct_compressedData          OBJ_id_smime_ct,9L

#define SN_id_ct_asciiTextWithCRLF              "id-ct-asciiTextWithCRLF"
#define NID_id_ct_asciiTextWithCRLF             787
#define OBJ_id_ct_asciiTextWithCRLF             OBJ_id_smime_ct,27L
```



```
#define SN_id_smime_aa_receiptRequest           "id-smime-aa-receiptRequest"
#define NID_id_smime_aa_receiptRequest          212
#define OBJ_id_smime_aa_receiptRequest          OBJ_id_smime_aa,1L

#define SN_id_smime_aa_securityLabel            "id-smime-aa-securityLabel"
#define NID_id_smime_aa_securityLabel           213
#define OBJ_id_smime_aa_securityLabel           OBJ_id_smime_aa,2L

#define SN_id_smime_aa_mlExpandHistory          "id-smime-aa-mlExpandHistory"
#define NID_id_smime_aa_mlExpandHistory         214
#define OBJ_id_smime_aa_mlExpandHistory         OBJ_id_smime_aa,3L

#define SN_id_smime_aa_contentHint              "id-smime-aa-contentHint"
#define NID_id_smime_aa_contentHint             215
#define OBJ_id_smime_aa_contentHint             OBJ_id_smime_aa,4L

#define SN_id_smime_aa_msgSigDigest             "id-smime-aa-msgSigDigest"
#define NID_id_smime_aa_msgSigDigest            216
#define OBJ_id_smime_aa_msgSigDigest            OBJ_id_smime_aa,5L

#define SN_id_smime_aa_encapContentType         "id-smime-aa-encapContentType"
#define NID_id_smime_aa_encapContentType                217
#define OBJ_id_smime_aa_encapContentType                OBJ_id_smime_aa,6L

#define SN_id_smime_aa_contentIdentifier                "id-smime-aa-contentIden
#define NID_id_smime_aa_contentIdentifier               218
#define OBJ_id_smime_aa_contentIdentifier               OBJ_id_smime_aa,7L

#define SN_id_smime_aa_macValue         "id-smime-aa-macValue"
#define NID_id_smime_aa_macValue                219
#define OBJ_id_smime_aa_macValue                OBJ_id_smime_aa,8L

#define SN_id_smime_aa_equivalentLabels         "id-smime-aa-equivalentLabels"
#define NID_id_smime_aa_equivalentLabels                220
#define OBJ_id_smime_aa_equivalentLabels                OBJ_id_smime_aa,9L

#define SN_id_smime_aa_contentReference         "id-smime-aa-contentReference"
#define NID_id_smime_aa_contentReference                221
#define OBJ_id_smime_aa_contentReference                OBJ_id_smime_aa,10L

#define SN_id_smime_aa_encrypKeyPref            "id-smime-aa-encrypKeyPref"
#define NID_id_smime_aa_encrypKeyPref           222
#define OBJ_id_smime_aa_encrypKeyPref           OBJ_id_smime_aa,11L

#define SN_id_smime_aa_signingCertificate               "id-smime-aa-signingCert
#define NID_id_smime_aa_signingCertificate              223
#define OBJ_id_smime_aa_signingCertificate              OBJ_id_smime_aa,12L

#define SN_id_smime_aa_smimeEncryptCerts                "id-smime-aa-smimeEncryp
#define NID_id_smime_aa_smimeEncryptCerts               224
#define OBJ_id_smime_aa_smimeEncryptCerts               OBJ_id_smime_aa,13L

#define SN_id_smime_aa_timeStampToken           "id-smime-aa-timeStampToken"
#define NID_id_smime_aa_timeStampToken          225
#define OBJ_id_smime_aa_timeStampToken          OBJ_id_smime_aa,14L

#define SN_id_smime_aa_ets_sigPolicyId          "id-smime-aa-ets-sigPolicyId"
#define NID_id_smime_aa_ets_sigPolicyId         226
#define OBJ_id_smime_aa_ets_sigPolicyId         OBJ_id_smime_aa,15L

#define SN_id_smime_aa_ets_commitmentType               "id-smime-aa-ets-commitm
#define NID_id_smime_aa_ets_commitmentType              227
#define OBJ_id_smime_aa_ets_commitmentType              OBJ_id_smime_aa,16L

#define SN_id_smime_aa_ets_signerLocation               "id-smime-aa-ets-signerL
```

```
#define NID_id_smime_aa_ets_signerLocation              228
#define OBJ_id_smime_aa_ets_signerLocation              OBJ_id_smime_aa,17L

#define SN_id_smime_aa_ets_signerAttr           "id-smime-aa-ets-signerAttr"
#define NID_id_smime_aa_ets_signerAttr          229
#define OBJ_id_smime_aa_ets_signerAttr          OBJ_id_smime_aa,18L

#define SN_id_smime_aa_ets_otherSigCert         "id-smime-aa-ets-otherSigCert"
#define NID_id_smime_aa_ets_otherSigCert                230
#define OBJ_id_smime_aa_ets_otherSigCert                OBJ_id_smime_aa,19L

#define SN_id_smime_aa_ets_contentTimestamp             "id-smime-aa-ets-content
#define NID_id_smime_aa_ets_contentTimestamp            231
#define OBJ_id_smime_aa_ets_contentTimestamp            OBJ_id_smime_aa,20L

#define SN_id_smime_aa_ets_CertificateRefs              "id-smime-aa-ets-Certifi
#define NID_id_smime_aa_ets_CertificateRefs             232
#define OBJ_id_smime_aa_ets_CertificateRefs             OBJ_id_smime_aa,21L

#define SN_id_smime_aa_ets_RevocationRefs               "id-smime-aa-ets-Revocat
#define NID_id_smime_aa_ets_RevocationRefs              233
#define OBJ_id_smime_aa_ets_RevocationRefs              OBJ_id_smime_aa,22L

#define SN_id_smime_aa_ets_certValues           "id-smime-aa-ets-certValues"
#define NID_id_smime_aa_ets_certValues          234
#define OBJ_id_smime_aa_ets_certValues          OBJ_id_smime_aa,23L

#define SN_id_smime_aa_ets_revocationValues             "id-smime-aa-ets-revocat
#define NID_id_smime_aa_ets_revocationValues            235
#define OBJ_id_smime_aa_ets_revocationValues            OBJ_id_smime_aa,24L

#define SN_id_smime_aa_ets_escTimeStamp         "id-smime-aa-ets-escTimeStamp"
#define NID_id_smime_aa_ets_escTimeStamp                236
#define OBJ_id_smime_aa_ets_escTimeStamp                OBJ_id_smime_aa,25L

#define SN_id_smime_aa_ets_certCRLTimestamp             "id-smime-aa-ets-certCRL
#define NID_id_smime_aa_ets_certCRLTimestamp            237
#define OBJ_id_smime_aa_ets_certCRLTimestamp            OBJ_id_smime_aa,26L

#define SN_id_smime_aa_ets_archiveTimeStamp             "id-smime-aa-ets-archive
#define NID_id_smime_aa_ets_archiveTimeStamp            238
#define OBJ_id_smime_aa_ets_archiveTimeStamp            OBJ_id_smime_aa,27L

#define SN_id_smime_aa_signatureType            "id-smime-aa-signatureType"
#define NID_id_smime_aa_signatureType           239
#define OBJ_id_smime_aa_signatureType           OBJ_id_smime_aa,28L

#define SN_id_smime_aa_dvcs_dvc         "id-smime-aa-dvcs-dvc"
#define NID_id_smime_aa_dvcs_dvc                240
#define OBJ_id_smime_aa_dvcs_dvc                OBJ_id_smime_aa,29L

#define SN_id_smime_alg_ESDHwith3DES            "id-smime-alg-ESDHwith3DES"
#define NID_id_smime_alg_ESDHwith3DES           241
#define OBJ_id_smime_alg_ESDHwith3DES           OBJ_id_smime_alg,1L

#define SN_id_smime_alg_ESDHwithRC2             "id-smime-alg-ESDHwithRC2"
#define NID_id_smime_alg_ESDHwithRC2            242
#define OBJ_id_smime_alg_ESDHwithRC2            OBJ_id_smime_alg,2L

#define SN_id_smime_alg_3DESwrap                "id-smime-alg-3DESwrap"
#define NID_id_smime_alg_3DESwrap               243
#define OBJ_id_smime_alg_3DESwrap               OBJ_id_smime_alg,3L

#define SN_id_smime_alg_RC2wrap         "id-smime-alg-RC2wrap"
#define NID_id_smime_alg_RC2wrap                244
#define OBJ_id_smime_alg_RC2wrap                OBJ_id_smime_alg,4L
```



```
#define SN_id_smime_alg_ESDH            "id-smime-alg-ESDH"
#define NID_id_smime_alg_ESDH           245
#define OBJ_id_smime_alg_ESDH           OBJ_id_smime_alg,5L

#define SN_id_smime_alg_CMS3DESwrap             "id-smime-alg-CMS3DESwrap"
#define NID_id_smime_alg_CMS3DESwrap            246
#define OBJ_id_smime_alg_CMS3DESwrap            OBJ_id_smime_alg,6L

#define SN_id_smime_alg_CMSRC2wrap              "id-smime-alg-CMSRC2wrap"
#define NID_id_smime_alg_CMSRC2wrap             247
#define OBJ_id_smime_alg_CMSRC2wrap             OBJ_id_smime_alg,7L

#define SN_id_alg_PWRI_KEK              "id-alg-PWRI-KEK"
#define NID_id_alg_PWRI_KEK             893
#define OBJ_id_alg_PWRI_KEK             OBJ_id_smime_alg,9L

#define SN_id_smime_cd_ldap             "id-smime-cd-ldap"
#define NID_id_smime_cd_ldap            248
#define OBJ_id_smime_cd_ldap            OBJ_id_smime_cd,1L

#define SN_id_smime_spq_ets_sqt_uri             "id-smime-spq-ets-sqt-uri"
#define NID_id_smime_spq_ets_sqt_uri            249
#define OBJ_id_smime_spq_ets_sqt_uri            OBJ_id_smime_spq,1L

#define SN_id_smime_spq_ets_sqt_unotice         "id-smime-spq-ets-sqt-unotice"
#define NID_id_smime_spq_ets_sqt_unotice                250
#define OBJ_id_smime_spq_ets_sqt_unotice                OBJ_id_smime_spq,2L

#define SN_id_smime_cti_ets_proofOfOrigin               "id-smime-cti-ets-proofO
#define NID_id_smime_cti_ets_proofOfOrigin              251
#define OBJ_id_smime_cti_ets_proofOfOrigin              OBJ_id_smime_cti,1L

#define SN_id_smime_cti_ets_proofOfReceipt              "id-smime-cti-ets-proofO
#define NID_id_smime_cti_ets_proofOfReceipt             252
#define OBJ_id_smime_cti_ets_proofOfReceipt             OBJ_id_smime_cti,2L

#define SN_id_smime_cti_ets_proofOfDelivery             "id-smime-cti-ets-proofO
#define NID_id_smime_cti_ets_proofOfDelivery            253
#define OBJ_id_smime_cti_ets_proofOfDelivery            OBJ_id_smime_cti,3L

#define SN_id_smime_cti_ets_proofOfSender               "id-smime-cti-ets-proofO
#define NID_id_smime_cti_ets_proofOfSender              254
#define OBJ_id_smime_cti_ets_proofOfSender              OBJ_id_smime_cti,4L

#define SN_id_smime_cti_ets_proofOfApproval             "id-smime-cti-ets-proofO
#define NID_id_smime_cti_ets_proofOfApproval            255
#define OBJ_id_smime_cti_ets_proofOfApproval            OBJ_id_smime_cti,5L

#define SN_id_smime_cti_ets_proofOfCreation             "id-smime-cti-ets-proofO
#define NID_id_smime_cti_ets_proofOfCreation            256
#define OBJ_id_smime_cti_ets_proofOfCreation            OBJ_id_smime_cti,6L

#define LN_friendlyName         "friendlyName"
#define NID_friendlyName                156
#define OBJ_friendlyName                OBJ_pkcs9,20L

#define LN_localKeyID           "localKeyID"
#define NID_localKeyID          157
#define OBJ_localKeyID          OBJ_pkcs9,21L

#define SN_ms_csp_name          "CSPName"
#define LN_ms_csp_name          "Microsoft CSP Name"
#define NID_ms_csp_name         417
#define OBJ_ms_csp_name         1L,3L,6L,1L,4L,1L,311L,17L,1L
```

```
#define SN_LocalKeySet          "LocalKeySet"
#define LN_LocalKeySet          "Microsoft Local Key set"
#define NID_LocalKeySet         856
#define OBJ_LocalKeySet         1L,3L,6L,1L,4L,1L,311L,17L,2L

#define OBJ_certTypes           OBJ_pkcs9,22L

#define LN_x509Certificate              "x509Certificate"
#define NID_x509Certificate             158
#define OBJ_x509Certificate             OBJ_certTypes,1L

#define LN_sdsiCertificate              "sdsiCertificate"
#define NID_sdsiCertificate             159
#define OBJ_sdsiCertificate             OBJ_certTypes,2L

#define OBJ_crlTypes            OBJ_pkcs9,23L

#define LN_x509Crl              "x509Crl"
#define NID_x509Crl             160
#define OBJ_x509Crl             OBJ_crlTypes,1L

#define OBJ_pkcs12              OBJ_pkcs,12L

#define OBJ_pkcs12_pbeids               OBJ_pkcs12,1L

#define SN_pbe_WithSHA1And128BitRC4             "PBE-SHA1-RC4-128"
#define LN_pbe_WithSHA1And128BitRC4             "pbeWithSHA1And128BitRC4"
#define NID_pbe_WithSHA1And128BitRC4            144
#define OBJ_pbe_WithSHA1And128BitRC4            OBJ_pkcs12_pbeids,1L

#define SN_pbe_WithSHA1And40BitRC4              "PBE-SHA1-RC4-40"
#define LN_pbe_WithSHA1And40BitRC4              "pbeWithSHA1And40BitRC4"
#define NID_pbe_WithSHA1And40BitRC4             145
#define OBJ_pbe_WithSHA1And40BitRC4             OBJ_pkcs12_pbeids,2L

#define SN_pbe_WithSHA1And3_Key_TripleDES_CBC           "PBE-SHA1-3DES"
#define LN_pbe_WithSHA1And3_Key_TripleDES_CBC           "pbeWithSHA1And3-KeyTrip
#define NID_pbe_WithSHA1And3_Key_TripleDES_CBC          146
#define OBJ_pbe_WithSHA1And3_Key_TripleDES_CBC          OBJ_pkcs12_pbeids,3L

#define SN_pbe_WithSHA1And2_Key_TripleDES_CBC           "PBE-SHA1-2DES"
#define LN_pbe_WithSHA1And2_Key_TripleDES_CBC           "pbeWithSHA1And2-KeyTrip
#define NID_pbe_WithSHA1And2_Key_TripleDES_CBC          147
#define OBJ_pbe_WithSHA1And2_Key_TripleDES_CBC          OBJ_pkcs12_pbeids,4L

#define SN_pbe_WithSHA1And128BitRC2_CBC         "PBE-SHA1-RC2-128"
#define LN_pbe_WithSHA1And128BitRC2_CBC         "pbeWithSHA1And128BitRC2-CBC"
#define NID_pbe_WithSHA1And128BitRC2_CBC                148
#define OBJ_pbe_WithSHA1And128BitRC2_CBC                OBJ_pkcs12_pbeids,5L

#define SN_pbe_WithSHA1And40BitRC2_CBC          "PBE-SHA1-RC2-40"
#define LN_pbe_WithSHA1And40BitRC2_CBC          "pbeWithSHA1And40BitRC2-CBC"
#define NID_pbe_WithSHA1And40BitRC2_CBC         149
#define OBJ_pbe_WithSHA1And40BitRC2_CBC         OBJ_pkcs12_pbeids,6L

#define OBJ_pkcs12_Version1             OBJ_pkcs12,10L

#define OBJ_pkcs12_BagIds               OBJ_pkcs12_Version1,1L

#define LN_keyBag               "keyBag"
#define NID_keyBag              150
#define OBJ_keyBag              OBJ_pkcs12_BagIds,1L

#define LN_pkcs8ShroudedKeyBag          "pkcs8ShroudedKeyBag"
#define NID_pkcs8ShroudedKeyBag         151
#define OBJ_pkcs8ShroudedKeyBag         OBJ_pkcs12_BagIds,2L
```



```
#define LN_certBag              "certBag"
#define NID_certBag             152
#define OBJ_certBag             OBJ_pkcs12_BagIds,3L

#define LN_crlBag               "crlBag"
#define NID_crlBag              153
#define OBJ_crlBag              OBJ_pkcs12_BagIds,4L

#define LN_secretBag            "secretBag"
#define NID_secretBag           154
#define OBJ_secretBag           OBJ_pkcs12_BagIds,5L

#define LN_safeContentsBag              "safeContentsBag"
#define NID_safeContentsBag             155
#define OBJ_safeContentsBag             OBJ_pkcs12_BagIds,6L

#define SN_md2          "MD2"
#define LN_md2          "md2"
#define NID_md2         3
#define OBJ_md2         OBJ_rsadsi,2L,2L

#define SN_md4          "MD4"
#define LN_md4          "md4"
#define NID_md4         257
#define OBJ_md4         OBJ_rsadsi,2L,4L

#define SN_md5          "MD5"
#define LN_md5          "md5"
#define NID_md5         4
#define OBJ_md5         OBJ_rsadsi,2L,5L

#define SN_md5_sha1             "MD5-SHA1"
#define LN_md5_sha1             "md5-sha1"
#define NID_md5_sha1            114

#define LN_hmacWithMD5          "hmacWithMD5"
#define NID_hmacWithMD5         797
#define OBJ_hmacWithMD5         OBJ_rsadsi,2L,6L

#define LN_hmacWithSHA1         "hmacWithSHA1"
#define NID_hmacWithSHA1                163
#define OBJ_hmacWithSHA1                OBJ_rsadsi,2L,7L

#define LN_hmacWithSHA224               "hmacWithSHA224"
#define NID_hmacWithSHA224              798
#define OBJ_hmacWithSHA224              OBJ_rsadsi,2L,8L

#define LN_hmacWithSHA256               "hmacWithSHA256"
#define NID_hmacWithSHA256              799
#define OBJ_hmacWithSHA256              OBJ_rsadsi,2L,9L

#define LN_hmacWithSHA384               "hmacWithSHA384"
#define NID_hmacWithSHA384              800
#define OBJ_hmacWithSHA384              OBJ_rsadsi,2L,10L

#define LN_hmacWithSHA512               "hmacWithSHA512"
#define NID_hmacWithSHA512              801
#define OBJ_hmacWithSHA512              OBJ_rsadsi,2L,11L

#define SN_rc2_cbc              "RC2-CBC"
#define LN_rc2_cbc              "rc2-cbc"
#define NID_rc2_cbc             37
#define OBJ_rc2_cbc             OBJ_rsadsi,3L,2L

#define SN_rc2_ecb              "RC2-ECB"
```

```
1184 #define LN_rc2_ecb              "rc2-ecb"
1185 #define NID_rc2_ecb             38

1187 #define SN_rc2_cfb64            "RC2-CFB"
1188 #define LN_rc2_cfb64            "rc2-cfb"
1189 #define NID_rc2_cfb64           39

1191 #define SN_rc2_ofb64            "RC2-OFB"
1192 #define LN_rc2_ofb64            "rc2-ofb"
1193 #define NID_rc2_ofb64           40

1195 #define SN_rc2_40_cbc           "RC2-40-CBC"
1196 #define LN_rc2_40_cbc           "rc2-40-cbc"
1197 #define NID_rc2_40_cbc          98

1199 #define SN_rc2_64_cbc           "RC2-64-CBC"
1200 #define LN_rc2_64_cbc           "rc2-64-cbc"
1201 #define NID_rc2_64_cbc          166

1203 #define SN_rc4          "RC4"
1204 #define LN_rc4          "rc4"
1205 #define NID_rc4         5
1206 #define OBJ_rc4         OBJ_rsadsi,3L,4L

1208 #define SN_rc4_40               "RC4-40"
1209 #define LN_rc4_40               "rc4-40"
1210 #define NID_rc4_40              97

1212 #define SN_des_ede3_cbc         "DES-EDE3-CBC"
1213 #define LN_des_ede3_cbc         "des-ede3-cbc"
1214 #define NID_des_ede3_cbc                44
1215 #define OBJ_des_ede3_cbc                OBJ_rsadsi,3L,7L

1217 #define SN_rc5_cbc              "RC5-CBC"
1218 #define LN_rc5_cbc              "rc5-cbc"
1219 #define NID_rc5_cbc             120
1220 #define OBJ_rc5_cbc             OBJ_rsadsi,3L,8L

1222 #define SN_rc5_ecb              "RC5-ECB"
1223 #define LN_rc5_ecb              "rc5-ecb"
1224 #define NID_rc5_ecb             121

1226 #define SN_rc5_cfb64            "RC5-CFB"
1227 #define LN_rc5_cfb64            "rc5-cfb"
1228 #define NID_rc5_cfb64           122

1230 #define SN_rc5_ofb64            "RC5-OFB"
1231 #define LN_rc5_ofb64            "rc5-ofb"
1232 #define NID_rc5_ofb64           123

1234 #define SN_ms_ext_req           "msExtReq"
1235 #define LN_ms_ext_req           "Microsoft Extension Request"
1236 #define NID_ms_ext_req          171
1237 #define OBJ_ms_ext_req          1L,3L,6L,1L,4L,1L,311L,2L,1L,14L

1239 #define SN_ms_code_ind          "msCodeInd"
1240 #define LN_ms_code_ind          "Microsoft Individual Code Signing"
1241 #define NID_ms_code_ind         134
1242 #define OBJ_ms_code_ind         1L,3L,6L,1L,4L,1L,311L,2L,1L,21L

1244 #define SN_ms_code_com          "msCodeCom"
1245 #define LN_ms_code_com          "Microsoft Commercial Code Signing"
1246 #define NID_ms_code_com         135
1247 #define OBJ_ms_code_com         1L,3L,6L,1L,4L,1L,311L,2L,1L,22L

1249 #define SN_ms_ctl_sign          "msCTLSign"
```



```
1250 #define LN_ms_ctl_sign          "Microsoft Trust List Signing"
1251 #define NID_ms_ctl_sign         136
1252 #define OBJ_ms_ctl_sign         1L,3L,6L,1L,4L,1L,311L,10L,3L,1L

1254 #define SN_ms_sgc               "msSGC"
1255 #define LN_ms_sgc               "Microsoft Server Gated Crypto"
1256 #define NID_ms_sgc              137
1257 #define OBJ_ms_sgc              1L,3L,6L,1L,4L,1L,311L,10L,3L,3L

1259 #define SN_ms_efs               "msEFS"
1260 #define LN_ms_efs               "Microsoft Encrypted File System"
1261 #define NID_ms_efs              138
1262 #define OBJ_ms_efs              1L,3L,6L,1L,4L,1L,311L,10L,3L,4L

1264 #define SN_ms_smartcard_login           "msSmartcardLogin"
1265 #define LN_ms_smartcard_login           "Microsoft Smartcardlogin"
1266 #define NID_ms_smartcard_login          648
1267 #define OBJ_ms_smartcard_login          1L,3L,6L,1L,4L,1L,311L,20L,2L,2L

1269 #define SN_ms_upn               "msUPN"
1270 #define LN_ms_upn               "Microsoft Universal Principal Name"
1271 #define NID_ms_upn              649
1272 #define OBJ_ms_upn              1L,3L,6L,1L,4L,1L,311L,20L,2L,3L

1274 #define SN_idea_cbc             "IDEA-CBC"
1275 #define LN_idea_cbc             "idea-cbc"
1276 #define NID_idea_cbc            34
1277 #define OBJ_idea_cbc            1L,3L,6L,1L,4L,1L,188L,7L,1L,1L,2L

1279 #define SN_idea_ecb             "IDEA-ECB"
1280 #define LN_idea_ecb             "idea-ecb"
1281 #define NID_idea_ecb            36

1283 #define SN_idea_cfb64           "IDEA-CFB"
1284 #define LN_idea_cfb64           "idea-cfb"
1285 #define NID_idea_cfb64          35

1287 #define SN_idea_ofb64           "IDEA-OFB"
1288 #define LN_idea_ofb64           "idea-ofb"
1289 #define NID_idea_ofb64          46

1291 #define SN_bf_cbc               "BF-CBC"
1292 #define LN_bf_cbc               "bf-cbc"
1293 #define NID_bf_cbc              91
1294 #define OBJ_bf_cbc              1L,3L,6L,1L,4L,1L,3029L,1L,2L

1296 #define SN_bf_ecb               "BF-ECB"
1297 #define LN_bf_ecb               "bf-ecb"
1298 #define NID_bf_ecb              92

1300 #define SN_bf_cfb64             "BF-CFB"
1301 #define LN_bf_cfb64             "bf-cfb"
1302 #define NID_bf_cfb64            93

1304 #define SN_bf_ofb64             "BF-OFB"
1305 #define LN_bf_ofb64             "bf-ofb"
1306 #define NID_bf_ofb64            94

1308 #define SN_id_pkix              "PKIX"
1309 #define NID_id_pkix             127
1310 #define OBJ_id_pkix             1L,3L,6L,1L,5L,5L,7L

1312 #define SN_id_pkix_mod          "id-pkix-mod"
1313 #define NID_id_pkix_mod         258
1314 #define OBJ_id_pkix_mod         OBJ_id_pkix,0L
```

```
#define SN_id_pe                "id-pe"
#define NID_id_pe               175
#define OBJ_id_pe               OBJ_id_pkix,1L

#define SN_id_qt                "id-qt"
#define NID_id_qt               259
#define OBJ_id_qt               OBJ_id_pkix,2L

#define SN_id_kp                "id-kp"
#define NID_id_kp               128
#define OBJ_id_kp               OBJ_id_pkix,3L

#define SN_id_it                "id-it"
#define NID_id_it               260
#define OBJ_id_it               OBJ_id_pkix,4L

#define SN_id_pkip              "id-pkip"
#define NID_id_pkip             261
#define OBJ_id_pkip             OBJ_id_pkix,5L

#define SN_id_alg               "id-alg"
#define NID_id_alg              262
#define OBJ_id_alg              OBJ_id_pkix,6L

#define SN_id_cmc               "id-cmc"
#define NID_id_cmc              263
#define OBJ_id_cmc              OBJ_id_pkix,7L

#define SN_id_on                "id-on"
#define NID_id_on               264
#define OBJ_id_on               OBJ_id_pkix,8L

#define SN_id_pda               "id-pda"
#define NID_id_pda              265
#define OBJ_id_pda              OBJ_id_pkix,9L

#define SN_id_aca               "id-aca"
#define NID_id_aca              266
#define OBJ_id_aca              OBJ_id_pkix,10L

#define SN_id_qcs               "id-qcs"
#define NID_id_qcs              267
#define OBJ_id_qcs              OBJ_id_pkix,11L

#define SN_id_cct               "id-cct"
#define NID_id_cct              268
#define OBJ_id_cct              OBJ_id_pkix,12L

#define SN_id_ppl               "id-ppl"
#define NID_id_ppl              662
#define OBJ_id_ppl              OBJ_id_pkix,21L

#define SN_id_ad                "id-ad"
#define NID_id_ad               176
#define OBJ_id_ad               OBJ_id_pkix,48L

#define SN_id_pkix1_explicit_88         "id-pkix1-explicit-88"
#define NID_id_pkix1_explicit_88                269
#define OBJ_id_pkix1_explicit_88                OBJ_id_pkix_mod,1L

#define SN_id_pkix1_implicit_88         "id-pkix1-implicit-88"
#define NID_id_pkix1_implicit_88                270
#define OBJ_id_pkix1_implicit_88                OBJ_id_pkix_mod,2L

#define SN_id_pkix1_explicit_93         "id-pkix1-explicit-93"
#define NID_id_pkix1_explicit_93                271
```



```
#define OBJ_id_pkix1_explicit_93                OBJ_id_pkix_mod,3L

#define SN_id_pkix1_implicit_93         "id-pkix1-implicit-93"
#define NID_id_pkix1_implicit_93                272
#define OBJ_id_pkix1_implicit_93                OBJ_id_pkix_mod,4L

#define SN_id_mod_crmf          "id-mod-crmf"
#define NID_id_mod_crmf         273
#define OBJ_id_mod_crmf         OBJ_id_pkix_mod,5L

#define SN_id_mod_cmc           "id-mod-cmc"
#define NID_id_mod_cmc          274
#define OBJ_id_mod_cmc          OBJ_id_pkix_mod,6L

#define SN_id_mod_kea_profile_88                "id-mod-kea-profile-88"
#define NID_id_mod_kea_profile_88               275
#define OBJ_id_mod_kea_profile_88               OBJ_id_pkix_mod,7L

#define SN_id_mod_kea_profile_93                "id-mod-kea-profile-93"
#define NID_id_mod_kea_profile_93               276
#define OBJ_id_mod_kea_profile_93               OBJ_id_pkix_mod,8L

#define SN_id_mod_cmp           "id-mod-cmp"
#define NID_id_mod_cmp          277
#define OBJ_id_mod_cmp          OBJ_id_pkix_mod,9L

#define SN_id_mod_qualified_cert_88             "id-mod-qualified-cert-88"
#define NID_id_mod_qualified_cert_88            278
#define OBJ_id_mod_qualified_cert_88            OBJ_id_pkix_mod,10L

#define SN_id_mod_qualified_cert_93             "id-mod-qualified-cert-93"
#define NID_id_mod_qualified_cert_93            279
#define OBJ_id_mod_qualified_cert_93            OBJ_id_pkix_mod,11L

#define SN_id_mod_attribute_cert                "id-mod-attribute-cert"
#define NID_id_mod_attribute_cert               280
#define OBJ_id_mod_attribute_cert               OBJ_id_pkix_mod,12L

#define SN_id_mod_timestamp_protocol            "id-mod-timestamp-protocol"
#define NID_id_mod_timestamp_protocol           281
#define OBJ_id_mod_timestamp_protocol           OBJ_id_pkix_mod,13L

#define SN_id_mod_ocsp          "id-mod-ocsp"
#define NID_id_mod_ocsp         282
#define OBJ_id_mod_ocsp         OBJ_id_pkix_mod,14L

#define SN_id_mod_dvcs          "id-mod-dvcs"
#define NID_id_mod_dvcs         283
#define OBJ_id_mod_dvcs         OBJ_id_pkix_mod,15L

#define SN_id_mod_cmp2000               "id-mod-cmp2000"
#define NID_id_mod_cmp2000              284
#define OBJ_id_mod_cmp2000              OBJ_id_pkix_mod,16L

#define SN_info_access          "authorityInfoAccess"
#define LN_info_access          "Authority Information Access"
#define NID_info_access         177
#define OBJ_info_access         OBJ_id_pe,1L

#define SN_biometricInfo                "biometricInfo"
#define LN_biometricInfo                "Biometric Info"
#define NID_biometricInfo               285
#define OBJ_biometricInfo               OBJ_id_pe,2L

#define SN_qcStatements         "qcStatements"
#define NID_qcStatements                286
```

```
1448 #define OBJ_qcStatements                OBJ_id_pe,3L

1450 #define SN_ac_auditEntity               "ac-auditEntity"
1451 #define NID_ac_auditEntity              287
1452 #define OBJ_ac_auditEntity              OBJ_id_pe,4L

1454 #define SN_ac_targeting         "ac-targeting"
1455 #define NID_ac_targeting                288
1456 #define OBJ_ac_targeting                OBJ_id_pe,5L

1458 #define SN_aaControls           "aaControls"
1459 #define NID_aaControls          289
1460 #define OBJ_aaControls          OBJ_id_pe,6L

1462 #define SN_sbgp_ipAddrBlock             "sbgp-ipAddrBlock"
1463 #define NID_sbgp_ipAddrBlock            290
1464 #define OBJ_sbgp_ipAddrBlock            OBJ_id_pe,7L

1466 #define SN_sbgp_autonomousSysNum                "sbgp-autonomousSysNum"
1467 #define NID_sbgp_autonomousSysNum               291
1468 #define OBJ_sbgp_autonomousSysNum               OBJ_id_pe,8L

1470 #define SN_sbgp_routerIdentifier                "sbgp-routerIdentifier"
1471 #define NID_sbgp_routerIdentifier               292
1472 #define OBJ_sbgp_routerIdentifier               OBJ_id_pe,9L

1474 #define SN_ac_proxying          "ac-proxying"
1475 #define NID_ac_proxying         397
1476 #define OBJ_ac_proxying         OBJ_id_pe,10L

1478 #define SN_sinfo_access         "subjectInfoAccess"
1479 #define LN_sinfo_access         "Subject Information Access"
1480 #define NID_sinfo_access                398
1481 #define OBJ_sinfo_access                OBJ_id_pe,11L

1483 #define SN_proxyCertInfo                "proxyCertInfo"
1484 #define LN_proxyCertInfo                "Proxy Certificate Information"
1485 #define NID_proxyCertInfo               663
1486 #define OBJ_proxyCertInfo               OBJ_id_pe,14L

1488 #define SN_id_qt_cps            "id-qt-cps"
1489 #define LN_id_qt_cps            "Policy Qualifier CPS"
1490 #define NID_id_qt_cps           164
1491 #define OBJ_id_qt_cps           OBJ_id_qt,1L

1493 #define SN_id_qt_unotice                "id-qt-unotice"
1494 #define LN_id_qt_unotice                "Policy Qualifier User Notice"
1495 #define NID_id_qt_unotice               165
1496 #define OBJ_id_qt_unotice               OBJ_id_qt,2L

1498 #define SN_textNotice           "textNotice"
1499 #define NID_textNotice          293
1500 #define OBJ_textNotice          OBJ_id_qt,3L

1502 #define SN_server_auth          "serverAuth"
1503 #define LN_server_auth          "TLS Web Server Authentication"
1504 #define NID_server_auth         129
1505 #define OBJ_server_auth         OBJ_id_kp,1L

1507 #define SN_client_auth          "clientAuth"
1508 #define LN_client_auth          "TLS Web Client Authentication"
1509 #define NID_client_auth         130
1510 #define OBJ_client_auth         OBJ_id_kp,2L

1512 #define SN_code_sign            "codeSigning"
1513 #define LN_code_sign            "Code Signing"
```



```
1514 #define NID_code_sign           131
1515 #define OBJ_code_sign           OBJ_id_kp,3L

1517 #define SN_email_protect                "emailProtection"
1518 #define LN_email_protect                "E-mail Protection"
1519 #define NID_email_protect               132
1520 #define OBJ_email_protect               OBJ_id_kp,4L

1522 #define SN_ipsecEndSystem               "ipsecEndSystem"
1523 #define LN_ipsecEndSystem               "IPSec End System"
1524 #define NID_ipsecEndSystem              294
1525 #define OBJ_ipsecEndSystem              OBJ_id_kp,5L

1527 #define SN_ipsecTunnel          "ipsecTunnel"
1528 #define LN_ipsecTunnel          "IPSec Tunnel"
1529 #define NID_ipsecTunnel         295
1530 #define OBJ_ipsecTunnel         OBJ_id_kp,6L

1532 #define SN_ipsecUser            "ipsecUser"
1533 #define LN_ipsecUser            "IPSec User"
1534 #define NID_ipsecUser           296
1535 #define OBJ_ipsecUser           OBJ_id_kp,7L

1537 #define SN_time_stamp           "timeStamping"
1538 #define LN_time_stamp           "Time Stamping"
1539 #define NID_time_stamp          133
1540 #define OBJ_time_stamp          OBJ_id_kp,8L

1542 #define SN_OCSP_sign            "OCSPSigning"
1543 #define LN_OCSP_sign            "OCSP Signing"
1544 #define NID_OCSP_sign           180
1545 #define OBJ_OCSP_sign           OBJ_id_kp,9L

1547 #define SN_dvcs         "DVCS"
1548 #define LN_dvcs         "dvcs"
1549 #define NID_dvcs                297
1550 #define OBJ_dvcs                OBJ_id_kp,10L

1552 #define SN_id_it_caProtEncCert          "id-it-caProtEncCert"
1553 #define NID_id_it_caProtEncCert         298
1554 #define OBJ_id_it_caProtEncCert         OBJ_id_it,1L

1556 #define SN_id_it_signKeyPairTypes               "id-it-signKeyPairTypes"
1557 #define NID_id_it_signKeyPairTypes              299
1558 #define OBJ_id_it_signKeyPairTypes              OBJ_id_it,2L

1560 #define SN_id_it_encKeyPairTypes                "id-it-encKeyPairTypes"
1561 #define NID_id_it_encKeyPairTypes               300
1562 #define OBJ_id_it_encKeyPairTypes               OBJ_id_it,3L

1564 #define SN_id_it_preferredSymmAlg               "id-it-preferredSymmAlg"
1565 #define NID_id_it_preferredSymmAlg              301
1566 #define OBJ_id_it_preferredSymmAlg              OBJ_id_it,4L

1568 #define SN_id_it_caKeyUpdateInfo                "id-it-caKeyUpdateInfo"
1569 #define NID_id_it_caKeyUpdateInfo               302
1570 #define OBJ_id_it_caKeyUpdateInfo               OBJ_id_it,5L

1572 #define SN_id_it_currentCRL             "id-it-currentCRL"
1573 #define NID_id_it_currentCRL            303
1574 #define OBJ_id_it_currentCRL            OBJ_id_it,6L

1576 #define SN_id_it_unsupportedOIDs                "id-it-unsupportedOIDs"
1577 #define NID_id_it_unsupportedOIDs               304
1578 #define OBJ_id_it_unsupportedOIDs               OBJ_id_it,7L
```

```
1580 #define SN_id_it_subscriptionRequest            "id-it-subscriptionRequest"
1581 #define NID_id_it_subscriptionRequest           305
1582 #define OBJ_id_it_subscriptionRequest           OBJ_id_it,8L

1584 #define SN_id_it_subscriptionResponse           "id-it-subscriptionResponse"
1585 #define NID_id_it_subscriptionResponse          306
1586 #define OBJ_id_it_subscriptionResponse          OBJ_id_it,9L

1588 #define SN_id_it_keyPairParamReq                "id-it-keyPairParamReq"
1589 #define NID_id_it_keyPairParamReq               307
1590 #define OBJ_id_it_keyPairParamReq               OBJ_id_it,10L

1592 #define SN_id_it_keyPairParamRep                "id-it-keyPairParamRep"
1593 #define NID_id_it_keyPairParamRep               308
1594 #define OBJ_id_it_keyPairParamRep               OBJ_id_it,11L

1596 #define SN_id_it_revPassphrase          "id-it-revPassphrase"
1597 #define NID_id_it_revPassphrase         309
1598 #define OBJ_id_it_revPassphrase         OBJ_id_it,12L

1600 #define SN_id_it_implicitConfirm                "id-it-implicitConfirm"
1601 #define NID_id_it_implicitConfirm               310
1602 #define OBJ_id_it_implicitConfirm               OBJ_id_it,13L

1604 #define SN_id_it_confirmWaitTime                "id-it-confirmWaitTime"
1605 #define NID_id_it_confirmWaitTime               311
1606 #define OBJ_id_it_confirmWaitTime               OBJ_id_it,14L

1608 #define SN_id_it_origPKIMessage         "id-it-origPKIMessage"
1609 #define NID_id_it_origPKIMessage                312
1610 #define OBJ_id_it_origPKIMessage                OBJ_id_it,15L

1612 #define SN_id_it_suppLangTags           "id-it-suppLangTags"
1613 #define NID_id_it_suppLangTags          784
1614 #define OBJ_id_it_suppLangTags          OBJ_id_it,16L

1616 #define SN_id_regCtrl           "id-regCtrl"
1617 #define NID_id_regCtrl          313
1618 #define OBJ_id_regCtrl          OBJ_id_pkip,1L

1620 #define SN_id_regInfo           "id-regInfo"
1621 #define NID_id_regInfo          314
1622 #define OBJ_id_regInfo          OBJ_id_pkip,2L

1624 #define SN_id_regCtrl_regToken          "id-regCtrl-regToken"
1625 #define NID_id_regCtrl_regToken         315
1626 #define OBJ_id_regCtrl_regToken         OBJ_id_regCtrl,1L

1628 #define SN_id_regCtrl_authenticator             "id-regCtrl-authenticator"
1629 #define NID_id_regCtrl_authenticator            316
1630 #define OBJ_id_regCtrl_authenticator            OBJ_id_regCtrl,2L

1632 #define SN_id_regCtrl_pkiPublicationInfo                "id-regCtrl-pkiPublicati
1633 #define NID_id_regCtrl_pkiPublicationInfo               317
1634 #define OBJ_id_regCtrl_pkiPublicationInfo               OBJ_id_regCtrl,3L

1636 #define SN_id_regCtrl_pkiArchiveOptions         "id-regCtrl-pkiArchiveOptions"
1637 #define NID_id_regCtrl_pkiArchiveOptions                318
1638 #define OBJ_id_regCtrl_pkiArchiveOptions                OBJ_id_regCtrl,4L

1640 #define SN_id_regCtrl_oldCertID         "id-regCtrl-oldCertID"
1641 #define NID_id_regCtrl_oldCertID                319
1642 #define OBJ_id_regCtrl_oldCertID                OBJ_id_regCtrl,5L

1644 #define SN_id_regCtrl_protocolEncrKey           "id-regCtrl-protocolEncrKey"
1645 #define NID_id_regCtrl_protocolEncrKey          320
```



```
1646 #define OBJ_id_regCtrl_protocolEncrKey          OBJ_id_regCtrl,6L

1648 #define SN_id_regInfo_utf8Pairs         "id-regInfo-utf8Pairs"
1649 #define NID_id_regInfo_utf8Pairs                321
1650 #define OBJ_id_regInfo_utf8Pairs                OBJ_id_regInfo,1L

1652 #define SN_id_regInfo_certReq           "id-regInfo-certReq"
1653 #define NID_id_regInfo_certReq          322
1654 #define OBJ_id_regInfo_certReq          OBJ_id_regInfo,2L

1656 #define SN_id_alg_des40         "id-alg-des40"
1657 #define NID_id_alg_des40                323
1658 #define OBJ_id_alg_des40                OBJ_id_alg,1L

1660 #define SN_id_alg_noSignature           "id-alg-noSignature"
1661 #define NID_id_alg_noSignature          324
1662 #define OBJ_id_alg_noSignature          OBJ_id_alg,2L

1664 #define SN_id_alg_dh_sig_hmac_sha1              "id-alg-dh-sig-hmac-sha1"
1665 #define NID_id_alg_dh_sig_hmac_sha1             325
1666 #define OBJ_id_alg_dh_sig_hmac_sha1             OBJ_id_alg,3L

1668 #define SN_id_alg_dh_pop                "id-alg-dh-pop"
1669 #define NID_id_alg_dh_pop               326
1670 #define OBJ_id_alg_dh_pop               OBJ_id_alg,4L

1672 #define SN_id_cmc_statusInfo            "id-cmc-statusInfo"
1673 #define NID_id_cmc_statusInfo           327
1674 #define OBJ_id_cmc_statusInfo           OBJ_id_cmc,1L

1676 #define SN_id_cmc_identification                "id-cmc-identification"
1677 #define NID_id_cmc_identification               328
1678 #define OBJ_id_cmc_identification               OBJ_id_cmc,2L

1680 #define SN_id_cmc_identityProof         "id-cmc-identityProof"
1681 #define NID_id_cmc_identityProof                329
1682 #define OBJ_id_cmc_identityProof                OBJ_id_cmc,3L

1684 #define SN_id_cmc_dataReturn            "id-cmc-dataReturn"
1685 #define NID_id_cmc_dataReturn           330
1686 #define OBJ_id_cmc_dataReturn           OBJ_id_cmc,4L

1688 #define SN_id_cmc_transactionId         "id-cmc-transactionId"
1689 #define NID_id_cmc_transactionId                331
1690 #define OBJ_id_cmc_transactionId                OBJ_id_cmc,5L

1692 #define SN_id_cmc_senderNonce           "id-cmc-senderNonce"
1693 #define NID_id_cmc_senderNonce          332
1694 #define OBJ_id_cmc_senderNonce          OBJ_id_cmc,6L

1696 #define SN_id_cmc_recipientNonce                "id-cmc-recipientNonce"
1697 #define NID_id_cmc_recipientNonce               333
1698 #define OBJ_id_cmc_recipientNonce               OBJ_id_cmc,7L

1700 #define SN_id_cmc_addExtensions         "id-cmc-addExtensions"
1701 #define NID_id_cmc_addExtensions                334
1702 #define OBJ_id_cmc_addExtensions                OBJ_id_cmc,8L

1704 #define SN_id_cmc_encryptedPOP          "id-cmc-encryptedPOP"
1705 #define NID_id_cmc_encryptedPOP         335
1706 #define OBJ_id_cmc_encryptedPOP         OBJ_id_cmc,9L

1708 #define SN_id_cmc_decryptedPOP          "id-cmc-decryptedPOP"
1709 #define NID_id_cmc_decryptedPOP         336
1710 #define OBJ_id_cmc_decryptedPOP         OBJ_id_cmc,10L
```

```
#define SN_id_cmc_lraPOPWitness         "id-cmc-lraPOPWitness"
#define NID_id_cmc_lraPOPWitness                337
#define OBJ_id_cmc_lraPOPWitness                OBJ_id_cmc,11L

#define SN_id_cmc_getCert               "id-cmc-getCert"
#define NID_id_cmc_getCert              338
#define OBJ_id_cmc_getCert              OBJ_id_cmc,15L

#define SN_id_cmc_getCRL                "id-cmc-getCRL"
#define NID_id_cmc_getCRL               339
#define OBJ_id_cmc_getCRL               OBJ_id_cmc,16L

#define SN_id_cmc_revokeRequest         "id-cmc-revokeRequest"
#define NID_id_cmc_revokeRequest                340
#define OBJ_id_cmc_revokeRequest                OBJ_id_cmc,17L

#define SN_id_cmc_regInfo               "id-cmc-regInfo"
#define NID_id_cmc_regInfo              341
#define OBJ_id_cmc_regInfo              OBJ_id_cmc,18L

#define SN_id_cmc_responseInfo          "id-cmc-responseInfo"
#define NID_id_cmc_responseInfo         342
#define OBJ_id_cmc_responseInfo         OBJ_id_cmc,19L

#define SN_id_cmc_queryPending          "id-cmc-queryPending"
#define NID_id_cmc_queryPending         343
#define OBJ_id_cmc_queryPending         OBJ_id_cmc,21L

#define SN_id_cmc_popLinkRandom         "id-cmc-popLinkRandom"
#define NID_id_cmc_popLinkRandom                344
#define OBJ_id_cmc_popLinkRandom                OBJ_id_cmc,22L

#define SN_id_cmc_popLinkWitness                "id-cmc-popLinkWitness"
#define NID_id_cmc_popLinkWitness               345
#define OBJ_id_cmc_popLinkWitness               OBJ_id_cmc,23L

#define SN_id_cmc_confirmCertAcceptance         "id-cmc-confirmCertAcceptance"
#define NID_id_cmc_confirmCertAcceptance                346
#define OBJ_id_cmc_confirmCertAcceptance                OBJ_id_cmc,24L

#define SN_id_on_personalData           "id-on-personalData"
#define NID_id_on_personalData          347
#define OBJ_id_on_personalData          OBJ_id_on,1L

#define SN_id_on_permanentIdentifier            "id-on-permanentIdentifier"
#define LN_id_on_permanentIdentifier            "Permanent Identifier"
#define NID_id_on_permanentIdentifier           858
#define OBJ_id_on_permanentIdentifier           OBJ_id_on,3L

#define SN_id_pda_dateOfBirth           "id-pda-dateOfBirth"
#define NID_id_pda_dateOfBirth          348
#define OBJ_id_pda_dateOfBirth          OBJ_id_pda,1L

#define SN_id_pda_placeOfBirth          "id-pda-placeOfBirth"
#define NID_id_pda_placeOfBirth         349
#define OBJ_id_pda_placeOfBirth         OBJ_id_pda,2L

#define SN_id_pda_gender                "id-pda-gender"
#define NID_id_pda_gender               351
#define OBJ_id_pda_gender               OBJ_id_pda,3L

#define SN_id_pda_countryOfCitizenship          "id-pda-countryOfCitizenship"
#define NID_id_pda_countryOfCitizenship         352
#define OBJ_id_pda_countryOfCitizenship         OBJ_id_pda,4L

#define SN_id_pda_countryOfResidence            "id-pda-countryOfResidence"
```



```
#define NID_id_pda_countryOfResidence           353
#define OBJ_id_pda_countryOfResidence           OBJ_id_pda,5L

#define SN_id_aca_authenticationInfo            "id-aca-authenticationInfo"
#define NID_id_aca_authenticationInfo           354
#define OBJ_id_aca_authenticationInfo           OBJ_id_aca,1L

#define SN_id_aca_accessIdentity                "id-aca-accessIdentity"
#define NID_id_aca_accessIdentity               355
#define OBJ_id_aca_accessIdentity               OBJ_id_aca,2L

#define SN_id_aca_chargingIdentity              "id-aca-chargingIdentity"
#define NID_id_aca_chargingIdentity             356
#define OBJ_id_aca_chargingIdentity             OBJ_id_aca,3L

#define SN_id_aca_group         "id-aca-group"
#define NID_id_aca_group                357
#define OBJ_id_aca_group                OBJ_id_aca,4L

#define SN_id_aca_role          "id-aca-role"
#define NID_id_aca_role         358
#define OBJ_id_aca_role         OBJ_id_aca,5L

#define SN_id_aca_encAttrs              "id-aca-encAttrs"
#define NID_id_aca_encAttrs             399
#define OBJ_id_aca_encAttrs             OBJ_id_aca,6L

#define SN_id_qcs_pkixQCSyntax_v1               "id-qcs-pkixQCSyntax-v1"
#define NID_id_qcs_pkixQCSyntax_v1              359
#define OBJ_id_qcs_pkixQCSyntax_v1              OBJ_id_qcs,1L

#define SN_id_cct_crs           "id-cct-crs"
#define NID_id_cct_crs          360
#define OBJ_id_cct_crs          OBJ_id_cct,1L

#define SN_id_cct_PKIData               "id-cct-PKIData"
#define NID_id_cct_PKIData              361
#define OBJ_id_cct_PKIData              OBJ_id_cct,2L

#define SN_id_cct_PKIResponse           "id-cct-PKIResponse"
#define NID_id_cct_PKIResponse          362
#define OBJ_id_cct_PKIResponse          OBJ_id_cct,3L

#define SN_id_ppl_anyLanguage           "id-ppl-anyLanguage"
#define LN_id_ppl_anyLanguage           "Any language"
#define NID_id_ppl_anyLanguage          664
#define OBJ_id_ppl_anyLanguage          OBJ_id_ppl,0L

#define SN_id_ppl_inheritAll            "id-ppl-inheritAll"
#define LN_id_ppl_inheritAll            "Inherit all"
#define NID_id_ppl_inheritAll           665
#define OBJ_id_ppl_inheritAll           OBJ_id_ppl,1L

#define SN_Independent          "id-ppl-independent"
#define LN_Independent          "Independent"
#define NID_Independent         667
#define OBJ_Independent         OBJ_id_ppl,2L

#define SN_ad_OCSP              "OCSP"
#define LN_ad_OCSP              "OCSP"
#define NID_ad_OCSP             178
#define OBJ_ad_OCSP             OBJ_id_ad,1L

#define SN_ad_ca_issuers                "caIssuers"
#define LN_ad_ca_issuers                "CA Issuers"
#define NID_ad_ca_issuers               179
```

```
#define OBJ_ad_ca_issuers               OBJ_id_ad,2L

#define SN_ad_timeStamping              "ad_timestamping"
#define LN_ad_timeStamping              "AD Time Stamping"
#define NID_ad_timeStamping             363
#define OBJ_ad_timeStamping             OBJ_id_ad,3L

#define SN_ad_dvcs              "AD_DVCS"
#define LN_ad_dvcs              "ad dvcs"
#define NID_ad_dvcs             364
#define OBJ_ad_dvcs             OBJ_id_ad,4L

#define SN_caRepository         "caRepository"
#define LN_caRepository         "CA Repository"
#define NID_caRepository                785
#define OBJ_caRepository                OBJ_id_ad,5L

#define OBJ_id_pkix_OCSP                OBJ_ad_OCSP

#define SN_id_pkix_OCSP_basic           "basicOCSPResponse"
#define LN_id_pkix_OCSP_basic           "Basic OCSP Response"
#define NID_id_pkix_OCSP_basic          365
#define OBJ_id_pkix_OCSP_basic          OBJ_id_pkix_OCSP,1L

#define SN_id_pkix_OCSP_Nonce           "Nonce"
#define LN_id_pkix_OCSP_Nonce           "OCSP Nonce"
#define NID_id_pkix_OCSP_Nonce          366
#define OBJ_id_pkix_OCSP_Nonce          OBJ_id_pkix_OCSP,2L

#define SN_id_pkix_OCSP_CrlID           "CrlID"
#define LN_id_pkix_OCSP_CrlID           "OCSP CRL ID"
#define NID_id_pkix_OCSP_CrlID          367
#define OBJ_id_pkix_OCSP_CrlID          OBJ_id_pkix_OCSP,3L

#define SN_id_pkix_OCSP_acceptableResponses             "acceptableResponses"
#define LN_id_pkix_OCSP_acceptableResponses             "Acceptable OCSP Respons
#define NID_id_pkix_OCSP_acceptableResponses            368
#define OBJ_id_pkix_OCSP_acceptableResponses            OBJ_id_pkix_OCSP,4L

#define SN_id_pkix_OCSP_noCheck         "noCheck"
#define LN_id_pkix_OCSP_noCheck         "OCSP No Check"
#define NID_id_pkix_OCSP_noCheck                369
#define OBJ_id_pkix_OCSP_noCheck                OBJ_id_pkix_OCSP,5L

#define SN_id_pkix_OCSP_archiveCutoff           "archiveCutoff"
#define LN_id_pkix_OCSP_archiveCutoff           "OCSP Archive Cutoff"
#define NID_id_pkix_OCSP_archiveCutoff          370
#define OBJ_id_pkix_OCSP_archiveCutoff          OBJ_id_pkix_OCSP,6L

#define SN_id_pkix_OCSP_serviceLocator          "serviceLocator"
#define LN_id_pkix_OCSP_serviceLocator          "OCSP Service Locator"
#define NID_id_pkix_OCSP_serviceLocator         371
#define OBJ_id_pkix_OCSP_serviceLocator         OBJ_id_pkix_OCSP,7L

#define SN_id_pkix_OCSP_extendedStatus          "extendedStatus"
#define LN_id_pkix_OCSP_extendedStatus          "Extended OCSP Status"
#define NID_id_pkix_OCSP_extendedStatus         372
#define OBJ_id_pkix_OCSP_extendedStatus         OBJ_id_pkix_OCSP,8L

#define SN_id_pkix_OCSP_valid           "valid"
#define NID_id_pkix_OCSP_valid          373
#define OBJ_id_pkix_OCSP_valid          OBJ_id_pkix_OCSP,9L

#define SN_id_pkix_OCSP_path            "path"
#define NID_id_pkix_OCSP_path           374
#define OBJ_id_pkix_OCSP_path           OBJ_id_pkix_OCSP,10L
```



```
#define SN_id_pkix_OCSP_trustRoot               "trustRoot"
#define LN_id_pkix_OCSP_trustRoot               "Trust Root"
#define NID_id_pkix_OCSP_trustRoot              375
#define OBJ_id_pkix_OCSP_trustRoot              OBJ_id_pkix_OCSP,11L

#define SN_algorithm            "algorithm"
#define LN_algorithm            "algorithm"
#define NID_algorithm           376
#define OBJ_algorithm           1L,3L,14L,3L,2L

#define SN_md5WithRSA           "RSA-NP-MD5"
#define LN_md5WithRSA           "md5WithRSA"
#define NID_md5WithRSA          104
#define OBJ_md5WithRSA          OBJ_algorithm,3L

#define SN_des_ecb              "DES-ECB"
#define LN_des_ecb              "des-ecb"
#define NID_des_ecb             29
#define OBJ_des_ecb             OBJ_algorithm,6L

#define SN_des_cbc              "DES-CBC"
#define LN_des_cbc              "des-cbc"
#define NID_des_cbc             31
#define OBJ_des_cbc             OBJ_algorithm,7L

#define SN_des_ofb64            "DES-OFB"
#define LN_des_ofb64            "des-ofb"
#define NID_des_ofb64           45
#define OBJ_des_ofb64           OBJ_algorithm,8L

#define SN_des_cfb64            "DES-CFB"
#define LN_des_cfb64            "des-cfb"
#define NID_des_cfb64           30
#define OBJ_des_cfb64           OBJ_algorithm,9L

#define SN_rsaSignature         "rsaSignature"
#define NID_rsaSignature                377
#define OBJ_rsaSignature                OBJ_algorithm,11L

#define SN_dsa_2                "DSA-old"
#define LN_dsa_2                "dsaEncryption-old"
#define NID_dsa_2               67
#define OBJ_dsa_2               OBJ_algorithm,12L

#define SN_dsaWithSHA           "DSA-SHA"
#define LN_dsaWithSHA           "dsaWithSHA"
#define NID_dsaWithSHA          66
#define OBJ_dsaWithSHA          OBJ_algorithm,13L

#define SN_shaWithRSAEncryption         "RSA-SHA"
#define LN_shaWithRSAEncryption         "shaWithRSAEncryption"
#define NID_shaWithRSAEncryption                42
#define OBJ_shaWithRSAEncryption                OBJ_algorithm,15L

#define SN_des_ede_ecb          "DES-EDE"
#define LN_des_ede_ecb          "des-ede"
#define NID_des_ede_ecb         32
#define OBJ_des_ede_ecb         OBJ_algorithm,17L

#define SN_des_ede3_ecb         "DES-EDE3"
#define LN_des_ede3_ecb         "des-ede3"
#define NID_des_ede3_ecb                33

#define SN_des_ede_cbc          "DES-EDE-CBC"
#define LN_des_ede_cbc          "des-ede-cbc"
```

```
#define NID_des_ede_cbc         43

#define SN_des_ede_cfb64                "DES-EDE-CFB"
#define LN_des_ede_cfb64                "des-ede-cfb"
#define NID_des_ede_cfb64               60

#define SN_des_ede3_cfb64               "DES-EDE3-CFB"
#define LN_des_ede3_cfb64               "des-ede3-cfb"
#define NID_des_ede3_cfb64              61

#define SN_des_ede_ofb64                "DES-EDE-OFB"
#define LN_des_ede_ofb64                "des-ede-ofb"
#define NID_des_ede_ofb64               62

#define SN_des_ede3_ofb64               "DES-EDE3-OFB"
#define LN_des_ede3_ofb64               "des-ede3-ofb"
#define NID_des_ede3_ofb64              63

#define SN_desx_cbc             "DESX-CBC"
#define LN_desx_cbc             "desx-cbc"
#define NID_desx_cbc            80

#define SN_sha          "SHA"
#define LN_sha          "sha"
#define NID_sha         41
#define OBJ_sha         OBJ_algorithm,18L

#define SN_sha1         "SHA1"
#define LN_sha1         "sha1"
#define NID_sha1                64
#define OBJ_sha1                OBJ_algorithm,26L

#define SN_dsaWithSHA1_2                "DSA-SHA1-old"
#define LN_dsaWithSHA1_2                "dsaWithSHA1-old"
#define NID_dsaWithSHA1_2               70
#define OBJ_dsaWithSHA1_2               OBJ_algorithm,27L

#define SN_sha1WithRSA          "RSA-SHA1-2"
#define LN_sha1WithRSA          "sha1WithRSA"
#define NID_sha1WithRSA         115
#define OBJ_sha1WithRSA         OBJ_algorithm,29L

#define SN_ripemd160            "RIPEMD160"
#define LN_ripemd160            "ripemd160"
#define NID_ripemd160           117
#define OBJ_ripemd160           1L,3L,36L,3L,2L,1L

#define SN_ripemd160WithRSA             "RSA-RIPEMD160"
#define LN_ripemd160WithRSA             "ripemd160WithRSA"
#define NID_ripemd160WithRSA            119
#define OBJ_ripemd160WithRSA            1L,3L,36L,3L,3L,1L,2L

#define SN_sxnet                "SXNetID"
#define LN_sxnet                "Strong Extranet ID"
#define NID_sxnet               143
#define OBJ_sxnet               1L,3L,101L,1L,4L,1L

#define SN_X500         "X500"
#define LN_X500         "directory services (X.500)"
#define NID_X500                11
#define OBJ_X500                2L,5L

#define SN_X509         "X509"
#define NID_X509                12
#define OBJ_X509                OBJ_X500,4L
```



```
#define SN_commonName           "CN"
#define LN_commonName           "commonName"
#define NID_commonName          13
#define OBJ_commonName          OBJ_X509,3L

#define SN_surname              "SN"
#define LN_surname              "surname"
#define NID_surname             100
#define OBJ_surname             OBJ_X509,4L

#define LN_serialNumber         "serialNumber"
#define NID_serialNumber                105
#define OBJ_serialNumber                OBJ_X509,5L

#define SN_countryName          "C"
#define LN_countryName          "countryName"
#define NID_countryName         14
#define OBJ_countryName         OBJ_X509,6L

#define SN_localityName         "L"
#define LN_localityName         "localityName"
#define NID_localityName                15
#define OBJ_localityName                OBJ_X509,7L

#define SN_stateOrProvinceName          "ST"
#define LN_stateOrProvinceName          "stateOrProvinceName"
#define NID_stateOrProvinceName         16
#define OBJ_stateOrProvinceName         OBJ_X509,8L

#define SN_streetAddress                "street"
#define LN_streetAddress                "streetAddress"
#define NID_streetAddress               660
#define OBJ_streetAddress               OBJ_X509,9L

#define SN_organizationName             "O"
#define LN_organizationName             "organizationName"
#define NID_organizationName            17
#define OBJ_organizationName            OBJ_X509,10L

#define SN_organizationalUnitName               "OU"
#define LN_organizationalUnitName               "organizationalUnitName"
#define NID_organizationalUnitName              18
#define OBJ_organizationalUnitName              OBJ_X509,11L

#define SN_title                "title"
#define LN_title                "title"
#define NID_title               106
#define OBJ_title               OBJ_X509,12L

#define LN_description          "description"
#define NID_description         107
#define OBJ_description         OBJ_X509,13L

#define LN_searchGuide          "searchGuide"
#define NID_searchGuide         859
#define OBJ_searchGuide         OBJ_X509,14L

#define LN_businessCategory             "businessCategory"
#define NID_businessCategory            860
#define OBJ_businessCategory            OBJ_X509,15L

#define LN_postalAddress                "postalAddress"
#define NID_postalAddress               861
#define OBJ_postalAddress               OBJ_X509,16L

#define LN_postalCode           "postalCode"
```

```
2108 #define NID_postalCode          661
2109 #define OBJ_postalCode          OBJ_X509,17L

2111 #define LN_postOfficeBox                "postOfficeBox"
2112 #define NID_postOfficeBox               862
2113 #define OBJ_postOfficeBox               OBJ_X509,18L

2115 #define LN_physicalDeliveryOfficeName          "physicalDeliveryOfficeName"
2116 #define NID_physicalDeliveryOfficeName         863
2117 #define OBJ_physicalDeliveryOfficeName         OBJ_X509,19L

2119 #define LN_telephoneNumber              "telephoneNumber"
2120 #define NID_telephoneNumber             864
2121 #define OBJ_telephoneNumber             OBJ_X509,20L

2123 #define LN_telexNumber          "telexNumber"
2124 #define NID_telexNumber         865
2125 #define OBJ_telexNumber         OBJ_X509,21L

2127 #define LN_teletexTerminalIdentifier           "teletexTerminalIdentifier"
2128 #define NID_teletexTerminalIdentifier          866
2129 #define OBJ_teletexTerminalIdentifier          OBJ_X509,22L

2131 #define LN_facsimileTelephoneNumber            "facsimileTelephoneNumber"
2132 #define NID_facsimileTelephoneNumber           867
2133 #define OBJ_facsimileTelephoneNumber           OBJ_X509,23L

2135 #define LN_x121Address          "x121Address"
2136 #define NID_x121Address         868
2137 #define OBJ_x121Address         OBJ_X509,24L

2139 #define LN_internationaliSDNNumber             "internationaliSDNNumber"
2140 #define NID_internationaliSDNNumber            869
2141 #define OBJ_internationaliSDNNumber            OBJ_X509,25L

2143 #define LN_registeredAddress            "registeredAddress"
2144 #define NID_registeredAddress           870
2145 #define OBJ_registeredAddress           OBJ_X509,26L

2147 #define LN_destinationIndicator         "destinationIndicator"
2148 #define NID_destinationIndicator                871
2149 #define OBJ_destinationIndicator                OBJ_X509,27L

2151 #define LN_preferredDeliveryMethod              "preferredDeliveryMethod"
2152 #define NID_preferredDeliveryMethod             872
2153 #define OBJ_preferredDeliveryMethod             OBJ_X509,28L

2155 #define LN_presentationAddress          "presentationAddress"
2156 #define NID_presentationAddress         873
2157 #define OBJ_presentationAddress         OBJ_X509,29L

2159 #define LN_supportedApplicationContext          "supportedApplicationContext"
2160 #define NID_supportedApplicationContext         874
2161 #define OBJ_supportedApplicationContext         OBJ_X509,30L

2163 #define SN_member               "member"
2164 #define NID_member              875
2165 #define OBJ_member              OBJ_X509,31L

2167 #define SN_owner                "owner"
2168 #define NID_owner               876
2169 #define OBJ_owner               OBJ_X509,32L

2171 #define LN_roleOccupant         "roleOccupant"
2172 #define NID_roleOccupant                877
2173 #define OBJ_roleOccupant                OBJ_X509,33L
```



```
2175 #define SN_seeAlso              "seeAlso"
2176 #define NID_seeAlso             878
2177 #define OBJ_seeAlso             OBJ_X509,34L

2179 #define LN_userPassword         "userPassword"
2180 #define NID_userPassword                879
2181 #define OBJ_userPassword                OBJ_X509,35L

2183 #define LN_userCertificate              "userCertificate"
2184 #define NID_userCertificate             880
2185 #define OBJ_userCertificate             OBJ_X509,36L

2187 #define LN_cACertificate                "cACertificate"
2188 #define NID_cACertificate               881
2189 #define OBJ_cACertificate               OBJ_X509,37L

2191 #define LN_authorityRevocationList              "authorityRevocationList"
2192 #define NID_authorityRevocationList             882
2193 #define OBJ_authorityRevocationList             OBJ_X509,38L

2195 #define LN_certificateRevocationList            "certificateRevocationList"
2196 #define NID_certificateRevocationList           883
2197 #define OBJ_certificateRevocationList           OBJ_X509,39L

2199 #define LN_crossCertificatePair         "crossCertificatePair"
2200 #define NID_crossCertificatePair                884
2201 #define OBJ_crossCertificatePair                OBJ_X509,40L

2203 #define SN_name         "name"
2204 #define LN_name         "name"
2205 #define NID_name                173
2206 #define OBJ_name                OBJ_X509,41L

2208 #define SN_givenName            "GN"
2209 #define LN_givenName            "givenName"
2210 #define NID_givenName           99
2211 #define OBJ_givenName           OBJ_X509,42L

2213 #define SN_initials             "initials"
2214 #define LN_initials             "initials"
2215 #define NID_initials            101
2216 #define OBJ_initials            OBJ_X509,43L

2218 #define LN_generationQualifier          "generationQualifier"
2219 #define NID_generationQualifier         509
2220 #define OBJ_generationQualifier         OBJ_X509,44L

2222 #define LN_x500UniqueIdentifier         "x500UniqueIdentifier"
2223 #define NID_x500UniqueIdentifier                503
2224 #define OBJ_x500UniqueIdentifier                OBJ_X509,45L

2226 #define SN_dnQualifier          "dnQualifier"
2227 #define LN_dnQualifier          "dnQualifier"
2228 #define NID_dnQualifier         174
2229 #define OBJ_dnQualifier         OBJ_X509,46L

2231 #define LN_enhancedSearchGuide          "enhancedSearchGuide"
2232 #define NID_enhancedSearchGuide         885
2233 #define OBJ_enhancedSearchGuide         OBJ_X509,47L

2235 #define LN_protocolInformation          "protocolInformation"
2236 #define NID_protocolInformation         886
2237 #define OBJ_protocolInformation         OBJ_X509,48L

2239 #define LN_distinguishedName            "distinguishedName"
```

```
#define NID_distinguishedName           887
#define OBJ_distinguishedName           OBJ_X509,49L

#define LN_uniqueMember         "uniqueMember"
#define NID_uniqueMember                888
#define OBJ_uniqueMember                OBJ_X509,50L

#define LN_houseIdentifier              "houseIdentifier"
#define NID_houseIdentifier             889
#define OBJ_houseIdentifier             OBJ_X509,51L

#define LN_supportedAlgorithms          "supportedAlgorithms"
#define NID_supportedAlgorithms         890
#define OBJ_supportedAlgorithms         OBJ_X509,52L

#define LN_deltaRevocationList          "deltaRevocationList"
#define NID_deltaRevocationList         891
#define OBJ_deltaRevocationList         OBJ_X509,53L

#define SN_dmdName              "dmdName"
#define NID_dmdName             892
#define OBJ_dmdName             OBJ_X509,54L

#define LN_pseudonym            "pseudonym"
#define NID_pseudonym           510
#define OBJ_pseudonym           OBJ_X509,65L

#define SN_role         "role"
#define LN_role         "role"
#define NID_role                400
#define OBJ_role                OBJ_X509,72L

#define SN_X500algorithms               "X500algorithms"
#define LN_X500algorithms               "directory services - algorithms"
#define NID_X500algorithms              378
#define OBJ_X500algorithms              OBJ_X500,8L

#define SN_rsa          "RSA"
#define LN_rsa          "rsa"
#define NID_rsa         19
#define OBJ_rsa         OBJ_X500algorithms,1L,1L

#define SN_mdc2WithRSA          "RSA-MDC2"
#define LN_mdc2WithRSA          "mdc2WithRSA"
#define NID_mdc2WithRSA         96
#define OBJ_mdc2WithRSA         OBJ_X500algorithms,3L,100L

#define SN_mdc2         "MDC2"
#define LN_mdc2         "mdc2"
#define NID_mdc2                95
#define OBJ_mdc2                OBJ_X500algorithms,3L,101L

#define SN_id_ce                "id-ce"
#define NID_id_ce               81
#define OBJ_id_ce               OBJ_X500,29L

#define SN_subject_directory_attributes         "subjectDirectoryAttributes"
#define LN_subject_directory_attributes         "X509v3 Subject Directory Attrib
#define NID_subject_directory_attributes                769
#define OBJ_subject_directory_attributes                OBJ_id_ce,9L

#define SN_subject_key_identifier               "subjectKeyIdentifier"
#define LN_subject_key_identifier               "X509v3 Subject Key Identifier"
#define NID_subject_key_identifier              82
#define OBJ_subject_key_identifier              OBJ_id_ce,14L
```



```
#define SN_key_usage            "keyUsage"
#define LN_key_usage            "X509v3 Key Usage"
#define NID_key_usage           83
#define OBJ_key_usage           OBJ_id_ce,15L

#define SN_private_key_usage_period             "privateKeyUsagePeriod"
#define LN_private_key_usage_period             "X509v3 Private Key Usage Period
#define NID_private_key_usage_period            84
#define OBJ_private_key_usage_period            OBJ_id_ce,16L

#define SN_subject_alt_name             "subjectAltName"
#define LN_subject_alt_name             "X509v3 Subject Alternative Name"
#define NID_subject_alt_name            85
#define OBJ_subject_alt_name            OBJ_id_ce,17L

#define SN_issuer_alt_name              "issuerAltName"
#define LN_issuer_alt_name              "X509v3 Issuer Alternative Name"
#define NID_issuer_alt_name             86
#define OBJ_issuer_alt_name             OBJ_id_ce,18L

#define SN_basic_constraints            "basicConstraints"
#define LN_basic_constraints            "X509v3 Basic Constraints"
#define NID_basic_constraints           87
#define OBJ_basic_constraints           OBJ_id_ce,19L

#define SN_crl_number           "crlNumber"
#define LN_crl_number           "X509v3 CRL Number"
#define NID_crl_number          88
#define OBJ_crl_number          OBJ_id_ce,20L

#define SN_crl_reason           "CRLReason"
#define LN_crl_reason           "X509v3 CRL Reason Code"
#define NID_crl_reason          141
#define OBJ_crl_reason          OBJ_id_ce,21L

#define SN_invalidity_date              "invalidityDate"
#define LN_invalidity_date              "Invalidity Date"
#define NID_invalidity_date             142
#define OBJ_invalidity_date             OBJ_id_ce,24L

#define SN_delta_crl            "deltaCRL"
#define LN_delta_crl            "X509v3 Delta CRL Indicator"
#define NID_delta_crl           140
#define OBJ_delta_crl           OBJ_id_ce,27L

#define SN_issuing_distribution_point           "issuingDistributionPoint"
#define LN_issuing_distribution_point           "X509v3 Issuing Distrubution Poi
#define NID_issuing_distribution_point          770
#define OBJ_issuing_distribution_point          OBJ_id_ce,28L

#define SN_certificate_issuer           "certificateIssuer"
#define LN_certificate_issuer           "X509v3 Certificate Issuer"
#define NID_certificate_issuer          771
#define OBJ_certificate_issuer          OBJ_id_ce,29L

#define SN_name_constraints             "nameConstraints"
#define LN_name_constraints             "X509v3 Name Constraints"
#define NID_name_constraints            666
#define OBJ_name_constraints            OBJ_id_ce,30L

#define SN_crl_distribution_points              "crlDistributionPoints"
#define LN_crl_distribution_points              "X509v3 CRL Distribution Points"
#define NID_crl_distribution_points             103
#define OBJ_crl_distribution_points             OBJ_id_ce,31L

#define SN_certificate_policies         "certificatePolicies"
```

```
2372 #define LN_certificate_policies         "X509v3 Certificate Policies"
2373 #define NID_certificate_policies                89
2374 #define OBJ_certificate_policies                OBJ_id_ce,32L

2376 #define SN_any_policy           "anyPolicy"
2377 #define LN_any_policy           "X509v3 Any Policy"
2378 #define NID_any_policy          746
2379 #define OBJ_any_policy          OBJ_certificate_policies,0L

2381 #define SN_policy_mappings              "policyMappings"
2382 #define LN_policy_mappings              "X509v3 Policy Mappings"
2383 #define NID_policy_mappings             747
2384 #define OBJ_policy_mappings             OBJ_id_ce,33L

2386 #define SN_authority_key_identifier             "authorityKeyIdentifier"
2387 #define LN_authority_key_identifier             "X509v3 Authority Key Identifier
2388 #define NID_authority_key_identifier            90
2389 #define OBJ_authority_key_identifier            OBJ_id_ce,35L

2391 #define SN_policy_constraints           "policyConstraints"
2392 #define LN_policy_constraints           "X509v3 Policy Constraints"
2393 #define NID_policy_constraints          401
2394 #define OBJ_policy_constraints          OBJ_id_ce,36L

2396 #define SN_ext_key_usage                "extendedKeyUsage"
2397 #define LN_ext_key_usage                "X509v3 Extended Key Usage"
2398 #define NID_ext_key_usage               126
2399 #define OBJ_ext_key_usage               OBJ_id_ce,37L

2401 #define SN_freshest_crl         "freshestCRL"
2402 #define LN_freshest_crl         "X509v3 Freshest CRL"
2403 #define NID_freshest_crl                857
2404 #define OBJ_freshest_crl                OBJ_id_ce,46L

2406 #define SN_inhibit_any_policy           "inhibitAnyPolicy"
2407 #define LN_inhibit_any_policy           "X509v3 Inhibit Any Policy"
2408 #define NID_inhibit_any_policy          748
2409 #define OBJ_inhibit_any_policy          OBJ_id_ce,54L

2411 #define SN_target_information           "targetInformation"
2412 #define LN_target_information           "X509v3 AC Targeting"
2413 #define NID_target_information          402
2414 #define OBJ_target_information          OBJ_id_ce,55L

2416 #define SN_no_rev_avail         "noRevAvail"
2417 #define LN_no_rev_avail         "X509v3 No Revocation Available"
2418 #define NID_no_rev_avail                403
2419 #define OBJ_no_rev_avail                OBJ_id_ce,56L

2421 #define SN_anyExtendedKeyUsage          "anyExtendedKeyUsage"
2422 #define LN_anyExtendedKeyUsage          "Any Extended Key Usage"
2423 #define NID_anyExtendedKeyUsage         910
2424 #define OBJ_anyExtendedKeyUsage         OBJ_ext_key_usage,0L

2426 #define SN_netscape             "Netscape"
2427 #define LN_netscape             "Netscape Communications Corp."
2428 #define NID_netscape            57
2429 #define OBJ_netscape            2L,16L,840L,1L,113730L

2431 #define SN_netscape_cert_extension              "nsCertExt"
2432 #define LN_netscape_cert_extension              "Netscape Certificate Extension"
2433 #define NID_netscape_cert_extension             58
2434 #define OBJ_netscape_cert_extension             OBJ_netscape,1L

2436 #define SN_netscape_data_type           "nsDataType"
2437 #define LN_netscape_data_type           "Netscape Data Type"
```



```
2438 #define NID_netscape_data_type          59
2439 #define OBJ_netscape_data_type          OBJ_netscape,2L

2441 #define SN_netscape_cert_type           "nsCertType"
2442 #define LN_netscape_cert_type           "Netscape Cert Type"
2443 #define NID_netscape_cert_type          71
2444 #define OBJ_netscape_cert_type          OBJ_netscape_cert_extension,1L

2446 #define SN_netscape_base_url            "nsBaseUrl"
2447 #define LN_netscape_base_url            "Netscape Base Url"
2448 #define NID_netscape_base_url           72
2449 #define OBJ_netscape_base_url           OBJ_netscape_cert_extension,2L

2451 #define SN_netscape_revocation_url              "nsRevocationUrl"
2452 #define LN_netscape_revocation_url              "Netscape Revocation Url"
2453 #define NID_netscape_revocation_url             73
2454 #define OBJ_netscape_revocation_url             OBJ_netscape_cert_extension,3L

2456 #define SN_netscape_ca_revocation_url           "nsCaRevocationUrl"
2457 #define LN_netscape_ca_revocation_url           "Netscape CA Revocation Url"
2458 #define NID_netscape_ca_revocation_url          74
2459 #define OBJ_netscape_ca_revocation_url          OBJ_netscape_cert_extension,4L

2461 #define SN_netscape_renewal_url         "nsRenewalUrl"
2462 #define LN_netscape_renewal_url         "Netscape Renewal Url"
2463 #define NID_netscape_renewal_url                75
2464 #define OBJ_netscape_renewal_url                OBJ_netscape_cert_extension,7L

2466 #define SN_netscape_ca_policy_url               "nsCaPolicyUrl"
2467 #define LN_netscape_ca_policy_url               "Netscape CA Policy Url"
2468 #define NID_netscape_ca_policy_url              76
2469 #define OBJ_netscape_ca_policy_url              OBJ_netscape_cert_extension,8L

2471 #define SN_netscape_ssl_server_name             "nsSslServerName"
2472 #define LN_netscape_ssl_server_name             "Netscape SSL Server Name"
2473 #define NID_netscape_ssl_server_name            77
2474 #define OBJ_netscape_ssl_server_name            OBJ_netscape_cert_extension,12L

2476 #define SN_netscape_comment             "nsComment"
2477 #define LN_netscape_comment             "Netscape Comment"
2478 #define NID_netscape_comment            78
2479 #define OBJ_netscape_comment            OBJ_netscape_cert_extension,13L

2481 #define SN_netscape_cert_sequence               "nsCertSequence"
2482 #define LN_netscape_cert_sequence               "Netscape Certificate Sequence"
2483 #define NID_netscape_cert_sequence              79
2484 #define OBJ_netscape_cert_sequence              OBJ_netscape_data_type,5L

2486 #define SN_ns_sgc               "nsSGC"
2487 #define LN_ns_sgc               "Netscape Server Gated Crypto"
2488 #define NID_ns_sgc              139
2489 #define OBJ_ns_sgc              OBJ_netscape,4L,1L

2491 #define SN_org          "ORG"
2492 #define LN_org          "org"
2493 #define NID_org         379
2494 #define OBJ_org         OBJ_iso,3L

2496 #define SN_dod          "DOD"
2497 #define LN_dod          "dod"
2498 #define NID_dod         380
2499 #define OBJ_dod         OBJ_org,6L

2501 #define SN_iana         "IANA"
2502 #define LN_iana         "iana"
2503 #define NID_iana                381
```

```
#define OBJ_iana                OBJ_dod,1L

#define OBJ_internet            OBJ_iana

#define SN_Directory            "directory"
#define LN_Directory            "Directory"
#define NID_Directory           382
#define OBJ_Directory           OBJ_internet,1L

#define SN_Management           "mgmt"
#define LN_Management           "Management"
#define NID_Management          383
#define OBJ_Management          OBJ_internet,2L

#define SN_Experimental         "experimental"
#define LN_Experimental         "Experimental"
#define NID_Experimental                384
#define OBJ_Experimental                OBJ_internet,3L

#define SN_Private              "private"
#define LN_Private              "Private"
#define NID_Private             385
#define OBJ_Private             OBJ_internet,4L

#define SN_Security             "security"
#define LN_Security             "Security"
#define NID_Security            386
#define OBJ_Security            OBJ_internet,5L

#define SN_SNMPv2               "snmpv2"
#define LN_SNMPv2               "SNMPv2"
#define NID_SNMPv2              387
#define OBJ_SNMPv2              OBJ_internet,6L

#define LN_Mail         "Mail"
#define NID_Mail                388
#define OBJ_Mail                OBJ_internet,7L

#define SN_Enterprises          "enterprises"
#define LN_Enterprises          "Enterprises"
#define NID_Enterprises         389
#define OBJ_Enterprises         OBJ_Private,1L

#define SN_dcObject             "dcobject"
#define LN_dcObject             "dcObject"
#define NID_dcObject            390
#define OBJ_dcObject            OBJ_Enterprises,1466L,344L

#define SN_mime_mhs             "mime-mhs"
#define LN_mime_mhs             "MIME MHS"
#define NID_mime_mhs            504
#define OBJ_mime_mhs            OBJ_Mail,1L

#define SN_mime_mhs_headings            "mime-mhs-headings"
#define LN_mime_mhs_headings            "mime-mhs-headings"
#define NID_mime_mhs_headings           505
#define OBJ_mime_mhs_headings           OBJ_mime_mhs,1L

#define SN_mime_mhs_bodies              "mime-mhs-bodies"
#define LN_mime_mhs_bodies              "mime-mhs-bodies"
#define NID_mime_mhs_bodies             506
#define OBJ_mime_mhs_bodies             OBJ_mime_mhs,2L

#define SN_id_hex_partial_message               "id-hex-partial-message"
#define LN_id_hex_partial_message               "id-hex-partial-message"
#define NID_id_hex_partial_message              507
```



```
#define OBJ_id_hex_partial_message              OBJ_mime_mhs_headings,1L

#define SN_id_hex_multipart_message             "id-hex-multipart-message"
#define LN_id_hex_multipart_message             "id-hex-multipart-message"
#define NID_id_hex_multipart_message            508
#define OBJ_id_hex_multipart_message            OBJ_mime_mhs_headings,2L

#define SN_rle_compression              "RLE"
#define LN_rle_compression              "run length compression"
#define NID_rle_compression             124
#define OBJ_rle_compression             1L,1L,1L,1L,666L,1L

#define SN_zlib_compression             "ZLIB"
#define LN_zlib_compression             "zlib compression"
#define NID_zlib_compression            125
#define OBJ_zlib_compression            OBJ_id_smime_alg,8L

#define OBJ_csor                2L,16L,840L,1L,101L,3L

#define OBJ_nistAlgorithms              OBJ_csor,4L

#define OBJ_aes         OBJ_nistAlgorithms,1L

#define SN_aes_128_ecb          "AES-128-ECB"
#define LN_aes_128_ecb          "aes-128-ecb"
#define NID_aes_128_ecb         418
#define OBJ_aes_128_ecb         OBJ_aes,1L

#define SN_aes_128_cbc          "AES-128-CBC"
#define LN_aes_128_cbc          "aes-128-cbc"
#define NID_aes_128_cbc         419
#define OBJ_aes_128_cbc         OBJ_aes,2L

#define SN_aes_128_ofb128               "AES-128-OFB"
#define LN_aes_128_ofb128               "aes-128-ofb"
#define NID_aes_128_ofb128              420
#define OBJ_aes_128_ofb128              OBJ_aes,3L

#define SN_aes_128_cfb128               "AES-128-CFB"
#define LN_aes_128_cfb128               "aes-128-cfb"
#define NID_aes_128_cfb128              421
#define OBJ_aes_128_cfb128              OBJ_aes,4L

#define SN_id_aes128_wrap               "id-aes128-wrap"
#define NID_id_aes128_wrap              788
#define OBJ_id_aes128_wrap              OBJ_aes,5L

#define SN_aes_128_gcm          "id-aes128-GCM"
#define LN_aes_128_gcm          "aes-128-gcm"
#define NID_aes_128_gcm         895
#define OBJ_aes_128_gcm         OBJ_aes,6L

#define SN_aes_128_ccm          "id-aes128-CCM"
#define LN_aes_128_ccm          "aes-128-ccm"
#define NID_aes_128_ccm         896
#define OBJ_aes_128_ccm         OBJ_aes,7L

#define SN_id_aes128_wrap_pad           "id-aes128-wrap-pad"
#define NID_id_aes128_wrap_pad          897
#define OBJ_id_aes128_wrap_pad          OBJ_aes,8L

#define SN_aes_192_ecb          "AES-192-ECB"
#define LN_aes_192_ecb          "aes-192-ecb"
#define NID_aes_192_ecb         422
#define OBJ_aes_192_ecb         OBJ_aes,21L
```

```
2636 #define SN_aes_192_cbc          "AES-192-CBC"
2637 #define LN_aes_192_cbc          "aes-192-cbc"
2638 #define NID_aes_192_cbc         423
2639 #define OBJ_aes_192_cbc         OBJ_aes,22L

2641 #define SN_aes_192_ofb128               "AES-192-OFB"
2642 #define LN_aes_192_ofb128               "aes-192-ofb"
2643 #define NID_aes_192_ofb128              424
2644 #define OBJ_aes_192_ofb128              OBJ_aes,23L

2646 #define SN_aes_192_cfb128               "AES-192-CFB"
2647 #define LN_aes_192_cfb128               "aes-192-cfb"
2648 #define NID_aes_192_cfb128              425
2649 #define OBJ_aes_192_cfb128              OBJ_aes,24L

2651 #define SN_id_aes192_wrap               "id-aes192-wrap"
2652 #define NID_id_aes192_wrap              789
2653 #define OBJ_id_aes192_wrap              OBJ_aes,25L

2655 #define SN_aes_192_gcm          "id-aes192-GCM"
2656 #define LN_aes_192_gcm          "aes-192-gcm"
2657 #define NID_aes_192_gcm         898
2658 #define OBJ_aes_192_gcm         OBJ_aes,26L

2660 #define SN_aes_192_ccm          "id-aes192-CCM"
2661 #define LN_aes_192_ccm          "aes-192-ccm"
2662 #define NID_aes_192_ccm         899
2663 #define OBJ_aes_192_ccm         OBJ_aes,27L

2665 #define SN_id_aes192_wrap_pad           "id-aes192-wrap-pad"
2666 #define NID_id_aes192_wrap_pad          900
2667 #define OBJ_id_aes192_wrap_pad          OBJ_aes,28L

2669 #define SN_aes_256_ecb          "AES-256-ECB"
2670 #define LN_aes_256_ecb          "aes-256-ecb"
2671 #define NID_aes_256_ecb         426
2672 #define OBJ_aes_256_ecb         OBJ_aes,41L

2674 #define SN_aes_256_cbc          "AES-256-CBC"
2675 #define LN_aes_256_cbc          "aes-256-cbc"
2676 #define NID_aes_256_cbc         427
2677 #define OBJ_aes_256_cbc         OBJ_aes,42L

2679 #define SN_aes_256_ofb128               "AES-256-OFB"
2680 #define LN_aes_256_ofb128               "aes-256-ofb"
2681 #define NID_aes_256_ofb128              428
2682 #define OBJ_aes_256_ofb128              OBJ_aes,43L

2684 #define SN_aes_256_cfb128               "AES-256-CFB"
2685 #define LN_aes_256_cfb128               "aes-256-cfb"
2686 #define NID_aes_256_cfb128              429
2687 #define OBJ_aes_256_cfb128              OBJ_aes,44L

2689 #define SN_id_aes256_wrap               "id-aes256-wrap"
2690 #define NID_id_aes256_wrap              790
2691 #define OBJ_id_aes256_wrap              OBJ_aes,45L

2693 #define SN_aes_256_gcm          "id-aes256-GCM"
2694 #define LN_aes_256_gcm          "aes-256-gcm"
2695 #define NID_aes_256_gcm         901
2696 #define OBJ_aes_256_gcm         OBJ_aes,46L

2698 #define SN_aes_256_ccm          "id-aes256-CCM"
2699 #define LN_aes_256_ccm          "aes-256-ccm"
2700 #define NID_aes_256_ccm         902
2701 #define OBJ_aes_256_ccm         OBJ_aes,47L
```



```
2703 #define SN_id_aes256_wrap_pad           "id-aes256-wrap-pad"
2704 #define NID_id_aes256_wrap_pad          903
2705 #define OBJ_id_aes256_wrap_pad          OBJ_aes,48L

2707 #define SN_aes_128_cfb1         "AES-128-CFB1"
2708 #define LN_aes_128_cfb1         "aes-128-cfb1"
2709 #define NID_aes_128_cfb1                650

2711 #define SN_aes_192_cfb1         "AES-192-CFB1"
2712 #define LN_aes_192_cfb1         "aes-192-cfb1"
2713 #define NID_aes_192_cfb1                651

2715 #define SN_aes_256_cfb1         "AES-256-CFB1"
2716 #define LN_aes_256_cfb1         "aes-256-cfb1"
2717 #define NID_aes_256_cfb1                652

2719 #define SN_aes_128_cfb8         "AES-128-CFB8"
2720 #define LN_aes_128_cfb8         "aes-128-cfb8"
2721 #define NID_aes_128_cfb8                653

2723 #define SN_aes_192_cfb8         "AES-192-CFB8"
2724 #define LN_aes_192_cfb8         "aes-192-cfb8"
2725 #define NID_aes_192_cfb8                654

2727 #define SN_aes_256_cfb8         "AES-256-CFB8"
2728 #define LN_aes_256_cfb8         "aes-256-cfb8"
2729 #define NID_aes_256_cfb8                655

2731 #define SN_aes_128_ctr          "AES-128-CTR"
2732 #define LN_aes_128_ctr          "aes-128-ctr"
2733 #define NID_aes_128_ctr         904

2735 #define SN_aes_192_ctr          "AES-192-CTR"
2736 #define LN_aes_192_ctr          "aes-192-ctr"
2737 #define NID_aes_192_ctr         905

2739 #define SN_aes_256_ctr          "AES-256-CTR"
2740 #define LN_aes_256_ctr          "aes-256-ctr"
2741 #define NID_aes_256_ctr         906

2743 #define SN_aes_128_xts          "AES-128-XTS"
2744 #define LN_aes_128_xts          "aes-128-xts"
2745 #define NID_aes_128_xts         913

2747 #define SN_aes_256_xts          "AES-256-XTS"
2748 #define LN_aes_256_xts          "aes-256-xts"
2749 #define NID_aes_256_xts         914

2751 #define SN_des_cfb1             "DES-CFB1"
2752 #define LN_des_cfb1             "des-cfb1"
2753 #define NID_des_cfb1            656

2755 #define SN_des_cfb8             "DES-CFB8"
2756 #define LN_des_cfb8             "des-cfb8"
2757 #define NID_des_cfb8            657

2759 #define SN_des_ede3_cfb1                "DES-EDE3-CFB1"
2760 #define LN_des_ede3_cfb1                "des-ede3-cfb1"
2761 #define NID_des_ede3_cfb1               658

2763 #define SN_des_ede3_cfb8                "DES-EDE3-CFB8"
2764 #define LN_des_ede3_cfb8                "des-ede3-cfb8"
2765 #define NID_des_ede3_cfb8               659

2767 #define OBJ_nist_hashalgs               OBJ_nistAlgorithms,2L
```

```
#define SN_sha256               "SHA256"
#define LN_sha256               "sha256"
#define NID_sha256              672
#define OBJ_sha256              OBJ_nist_hashalgs,1L

#define SN_sha384               "SHA384"
#define LN_sha384               "sha384"
#define NID_sha384              673
#define OBJ_sha384              OBJ_nist_hashalgs,2L

#define SN_sha512               "SHA512"
#define LN_sha512               "sha512"
#define NID_sha512              674
#define OBJ_sha512              OBJ_nist_hashalgs,3L

#define SN_sha224               "SHA224"
#define LN_sha224               "sha224"
#define NID_sha224              675
#define OBJ_sha224              OBJ_nist_hashalgs,4L

#define OBJ_dsa_with_sha2               OBJ_nistAlgorithms,3L

#define SN_dsa_with_SHA224              "dsa_with_SHA224"
#define NID_dsa_with_SHA224             802
#define OBJ_dsa_with_SHA224             OBJ_dsa_with_sha2,1L

#define SN_dsa_with_SHA256              "dsa_with_SHA256"
#define NID_dsa_with_SHA256             803
#define OBJ_dsa_with_SHA256             OBJ_dsa_with_sha2,2L

#define SN_hold_instruction_code                "holdInstructionCode"
#define LN_hold_instruction_code                "Hold Instruction Code"
#define NID_hold_instruction_code               430
#define OBJ_hold_instruction_code               OBJ_id_ce,23L

#define OBJ_holdInstruction             OBJ_X9_57,2L

#define SN_hold_instruction_none                "holdInstructionNone"
#define LN_hold_instruction_none                "Hold Instruction None"
#define NID_hold_instruction_none               431
#define OBJ_hold_instruction_none               OBJ_holdInstruction,1L

#define SN_hold_instruction_call_issuer         "holdInstructionCallIssuer"
#define LN_hold_instruction_call_issuer         "Hold Instruction Call Issuer"
#define NID_hold_instruction_call_issuer                432
#define OBJ_hold_instruction_call_issuer                OBJ_holdInstruction,2L

#define SN_hold_instruction_reject              "holdInstructionReject"
#define LN_hold_instruction_reject              "Hold Instruction Reject"
#define NID_hold_instruction_reject             433
#define OBJ_hold_instruction_reject             OBJ_holdInstruction,3L

#define SN_data         "data"
#define NID_data                434
#define OBJ_data                OBJ_itu_t,9L

#define SN_pss          "pss"
#define NID_pss         435
#define OBJ_pss         OBJ_data,2342L

#define SN_ucl          "ucl"
#define NID_ucl         436
#define OBJ_ucl         OBJ_pss,19200300L

#define SN_pilot                "pilot"
```



```
#define NID_pilot               437
#define OBJ_pilot               OBJ_ucl,100L

#define LN_pilotAttributeType           "pilotAttributeType"
#define NID_pilotAttributeType          438
#define OBJ_pilotAttributeType          OBJ_pilot,1L

#define LN_pilotAttributeSyntax         "pilotAttributeSyntax"
#define NID_pilotAttributeSyntax                439
#define OBJ_pilotAttributeSyntax                OBJ_pilot,3L

#define LN_pilotObjectClass             "pilotObjectClass"
#define NID_pilotObjectClass            440
#define OBJ_pilotObjectClass            OBJ_pilot,4L

#define LN_pilotGroups          "pilotGroups"
#define NID_pilotGroups         441
#define OBJ_pilotGroups         OBJ_pilot,10L

#define LN_iA5StringSyntax              "iA5StringSyntax"
#define NID_iA5StringSyntax             442
#define OBJ_iA5StringSyntax             OBJ_pilotAttributeSyntax,4L

#define LN_caseIgnoreIA5StringSyntax            "caseIgnoreIA5StringSyntax"
#define NID_caseIgnoreIA5StringSyntax           443
#define OBJ_caseIgnoreIA5StringSyntax           OBJ_pilotAttributeSyntax,5L

#define LN_pilotObject          "pilotObject"
#define NID_pilotObject         444
#define OBJ_pilotObject         OBJ_pilotObjectClass,3L

#define LN_pilotPerson          "pilotPerson"
#define NID_pilotPerson         445
#define OBJ_pilotPerson         OBJ_pilotObjectClass,4L

#define SN_account              "account"
#define NID_account             446
#define OBJ_account             OBJ_pilotObjectClass,5L

#define SN_document             "document"
#define NID_document            447
#define OBJ_document            OBJ_pilotObjectClass,6L

#define SN_room         "room"
#define NID_room                448
#define OBJ_room                OBJ_pilotObjectClass,7L

#define LN_documentSeries               "documentSeries"
#define NID_documentSeries              449
#define OBJ_documentSeries              OBJ_pilotObjectClass,9L

#define SN_Domain               "domain"
#define LN_Domain               "Domain"
#define NID_Domain              392
#define OBJ_Domain              OBJ_pilotObjectClass,13L

#define LN_rFC822localPart              "rFC822localPart"
#define NID_rFC822localPart             450
#define OBJ_rFC822localPart             OBJ_pilotObjectClass,14L

#define LN_dNSDomain            "dNSDomain"
#define NID_dNSDomain           451
#define OBJ_dNSDomain           OBJ_pilotObjectClass,15L

#define LN_domainRelatedObject          "domainRelatedObject"
#define NID_domainRelatedObject         452
```

```
2900 #define OBJ_domainRelatedObject         OBJ_pilotObjectClass,17L

2902 #define LN_friendlyCountry              "friendlyCountry"
2903 #define NID_friendlyCountry              453
2904 #define OBJ_friendlyCountry              OBJ_pilotObjectClass,18L

2906 #define LN_simpleSecurityObject         "simpleSecurityObject"
2907 #define NID_simpleSecurityObject                454
2908 #define OBJ_simpleSecurityObject                OBJ_pilotObjectClass,19L

2910 #define LN_pilotOrganization            "pilotOrganization"
2911 #define NID_pilotOrganization            455
2912 #define OBJ_pilotOrganization            OBJ_pilotObjectClass,20L

2914 #define LN_pilotDSA             "pilotDSA"
2915 #define NID_pilotDSA             456
2916 #define OBJ_pilotDSA             OBJ_pilotObjectClass,21L

2918 #define LN_qualityLabelledData          "qualityLabelledData"
2919 #define NID_qualityLabelledData          457
2920 #define OBJ_qualityLabelledData          OBJ_pilotObjectClass,22L

2922 #define SN_userId               "UID"
2923 #define LN_userId               "userId"
2924 #define NID_userId               458
2925 #define OBJ_userId               OBJ_pilotAttributeType,1L

2927 #define LN_textEncodedORAddress         "textEncodedORAddress"
2928 #define NID_textEncodedORAddress                459
2929 #define OBJ_textEncodedORAddress                OBJ_pilotAttributeType,2L

2931 #define SN_rfc822Mailbox                "mail"
2932 #define LN_rfc822Mailbox                "rfc822Mailbox"
2933 #define NID_rfc822Mailbox                460
2934 #define OBJ_rfc822Mailbox                OBJ_pilotAttributeType,3L

2936 #define SN_info         "info"
2937 #define NID_info                 461
2938 #define OBJ_info                 OBJ_pilotAttributeType,4L

2940 #define LN_favouriteDrink               "favouriteDrink"
2941 #define NID_favouriteDrink               462
2942 #define OBJ_favouriteDrink               OBJ_pilotAttributeType,5L

2944 #define LN_roomNumber           "roomNumber"
2945 #define NID_roomNumber           463
2946 #define OBJ_roomNumber           OBJ_pilotAttributeType,6L

2948 #define SN_photo                "photo"
2949 #define NID_photo                464
2950 #define OBJ_photo                OBJ_pilotAttributeType,7L

2952 #define LN_userClass            "userClass"
2953 #define NID_userClass            465
2954 #define OBJ_userClass            OBJ_pilotAttributeType,8L

2956 #define SN_host         "host"
2957 #define NID_host                 466
2958 #define OBJ_host                 OBJ_pilotAttributeType,9L

2960 #define SN_manager              "manager"
2961 #define NID_manager              467
2962 #define OBJ_manager              OBJ_pilotAttributeType,10L

2964 #define LN_documentIdentifier           "documentIdentifier"
2965 #define NID_documentIdentifier           468
```



```
2966 #define OBJ_documentIdentifier          OBJ_pilotAttributeType,11L

2968 #define LN_documentTitle                "documentTitle"
2969 #define NID_documentTitle                469
2970 #define OBJ_documentTitle                OBJ_pilotAttributeType,12L

2972 #define LN_documentVersion              "documentVersion"
2973 #define NID_documentVersion              470
2974 #define OBJ_documentVersion              OBJ_pilotAttributeType,13L

2976 #define LN_documentAuthor               "documentAuthor"
2977 #define NID_documentAuthor               471
2978 #define OBJ_documentAuthor               OBJ_pilotAttributeType,14L

2980 #define LN_documentLocation             "documentLocation"
2981 #define NID_documentLocation             472
2982 #define OBJ_documentLocation             OBJ_pilotAttributeType,15L

2984 #define LN_homeTelephoneNumber          "homeTelephoneNumber"
2985 #define NID_homeTelephoneNumber          473
2986 #define OBJ_homeTelephoneNumber          OBJ_pilotAttributeType,20L

2988 #define SN_secretary            "secretary"
2989 #define NID_secretary            474
2990 #define OBJ_secretary            OBJ_pilotAttributeType,21L

2992 #define LN_otherMailbox         "otherMailbox"
2993 #define NID_otherMailbox                475
2994 #define OBJ_otherMailbox                OBJ_pilotAttributeType,22L

2996 #define LN_lastModifiedTime             "lastModifiedTime"
2997 #define NID_lastModifiedTime             476
2998 #define OBJ_lastModifiedTime             OBJ_pilotAttributeType,23L

3000 #define LN_lastModifiedBy               "lastModifiedBy"
3001 #define NID_lastModifiedBy               477
3002 #define OBJ_lastModifiedBy               OBJ_pilotAttributeType,24L

3004 #define SN_domainComponent              "DC"
3005 #define LN_domainComponent              "domainComponent"
3006 #define NID_domainComponent              391
3007 #define OBJ_domainComponent              OBJ_pilotAttributeType,25L

3009 #define LN_aRecord              "aRecord"
3010 #define NID_aRecord              478
3011 #define OBJ_aRecord              OBJ_pilotAttributeType,26L

3013 #define LN_pilotAttributeType27         "pilotAttributeType27"
3014 #define NID_pilotAttributeType27                479
3015 #define OBJ_pilotAttributeType27                OBJ_pilotAttributeType,27L

3017 #define LN_mXRecord             "mXRecord"
3018 #define NID_mXRecord             480
3019 #define OBJ_mXRecord             OBJ_pilotAttributeType,28L

3021 #define LN_nSRecord             "nSRecord"
3022 #define NID_nSRecord             481
3023 #define OBJ_nSRecord             OBJ_pilotAttributeType,29L

3025 #define LN_sOARecord            "sOARecord"
3026 #define NID_sOARecord            482
3027 #define OBJ_sOARecord            OBJ_pilotAttributeType,30L

3029 #define LN_cNAMERecord          "cNAMERecord"
3030 #define NID_cNAMERecord          483
3031 #define OBJ_cNAMERecord          OBJ_pilotAttributeType,31L
```

```
#define LN_associatedDomain             "associatedDomain"
#define NID_associatedDomain            484
#define OBJ_associatedDomain            OBJ_pilotAttributeType,37L

#define LN_associatedName               "associatedName"
#define NID_associatedName              485
#define OBJ_associatedName              OBJ_pilotAttributeType,38L

#define LN_homePostalAddress            "homePostalAddress"
#define NID_homePostalAddress           486
#define OBJ_homePostalAddress           OBJ_pilotAttributeType,39L

#define LN_personalTitle                "personalTitle"
#define NID_personalTitle               487
#define OBJ_personalTitle               OBJ_pilotAttributeType,40L

#define LN_mobileTelephoneNumber                "mobileTelephoneNumber"
#define NID_mobileTelephoneNumber               488
#define OBJ_mobileTelephoneNumber               OBJ_pilotAttributeType,41L

#define LN_pagerTelephoneNumber         "pagerTelephoneNumber"
#define NID_pagerTelephoneNumber                489
#define OBJ_pagerTelephoneNumber                OBJ_pilotAttributeType,42L

#define LN_friendlyCountryName          "friendlyCountryName"
#define NID_friendlyCountryName         490
#define OBJ_friendlyCountryName         OBJ_pilotAttributeType,43L

#define LN_organizationalStatus         "organizationalStatus"
#define NID_organizationalStatus                491
#define OBJ_organizationalStatus                OBJ_pilotAttributeType,45L

#define LN_janetMailbox         "janetMailbox"
#define NID_janetMailbox                492
#define OBJ_janetMailbox                OBJ_pilotAttributeType,46L

#define LN_mailPreferenceOption         "mailPreferenceOption"
#define NID_mailPreferenceOption                493
#define OBJ_mailPreferenceOption                OBJ_pilotAttributeType,47L

#define LN_buildingName         "buildingName"
#define NID_buildingName                494
#define OBJ_buildingName                OBJ_pilotAttributeType,48L

#define LN_dSAQuality           "dSAQuality"
#define NID_dSAQuality          495
#define OBJ_dSAQuality          OBJ_pilotAttributeType,49L

#define LN_singleLevelQuality           "singleLevelQuality"
#define NID_singleLevelQuality          496
#define OBJ_singleLevelQuality          OBJ_pilotAttributeType,50L

#define LN_subtreeMinimumQuality                "subtreeMinimumQuality"
#define NID_subtreeMinimumQuality               497
#define OBJ_subtreeMinimumQuality               OBJ_pilotAttributeType,51L

#define LN_subtreeMaximumQuality                "subtreeMaximumQuality"
#define NID_subtreeMaximumQuality               498
#define OBJ_subtreeMaximumQuality               OBJ_pilotAttributeType,52L

#define LN_personalSignature            "personalSignature"
#define NID_personalSignature           499
#define OBJ_personalSignature           OBJ_pilotAttributeType,53L

#define LN_dITRedirect          "dITRedirect"
```



```
#define NID_dITRedirect         500
#define OBJ_dITRedirect         OBJ_pilotAttributeType,54L

#define SN_audio                "audio"
#define NID_audio               501
#define OBJ_audio               OBJ_pilotAttributeType,55L

#define LN_documentPublisher            "documentPublisher"
#define NID_documentPublisher           502
#define OBJ_documentPublisher           OBJ_pilotAttributeType,56L

#define SN_id_set               "id-set"
#define LN_id_set               "Secure Electronic Transactions"
#define NID_id_set              512
#define OBJ_id_set              OBJ_international_organizations,42L

#define SN_set_ctype            "set-ctype"
#define LN_set_ctype            "content types"
#define NID_set_ctype           513
#define OBJ_set_ctype           OBJ_id_set,0L

#define SN_set_msgExt           "set-msgExt"
#define LN_set_msgExt           "message extensions"
#define NID_set_msgExt          514
#define OBJ_set_msgExt          OBJ_id_set,1L

#define SN_set_attr             "set-attr"
#define NID_set_attr            515
#define OBJ_set_attr            OBJ_id_set,3L

#define SN_set_policy           "set-policy"
#define NID_set_policy          516
#define OBJ_set_policy          OBJ_id_set,5L

#define SN_set_certExt          "set-certExt"
#define LN_set_certExt          "certificate extensions"
#define NID_set_certExt         517
#define OBJ_set_certExt         OBJ_id_set,7L

#define SN_set_brand            "set-brand"
#define NID_set_brand           518
#define OBJ_set_brand           OBJ_id_set,8L

#define SN_setct_PANData                "setct-PANData"
#define NID_setct_PANData               519
#define OBJ_setct_PANData               OBJ_set_ctype,0L

#define SN_setct_PANToken               "setct-PANToken"
#define NID_setct_PANToken              520
#define OBJ_setct_PANToken              OBJ_set_ctype,1L

#define SN_setct_PANOnly                "setct-PANOnly"
#define NID_setct_PANOnly               521
#define OBJ_setct_PANOnly               OBJ_set_ctype,2L

#define SN_setct_OIData         "setct-OIData"
#define NID_setct_OIData                522
#define OBJ_setct_OIData                OBJ_set_ctype,3L

#define SN_setct_PI             "setct-PI"
#define NID_setct_PI            523
#define OBJ_setct_PI            OBJ_set_ctype,4L

#define SN_setct_PIData         "setct-PIData"
#define NID_setct_PIData                524
#define OBJ_setct_PIData                OBJ_set_ctype,5L
```

```
#define SN_setct_PIDataUnsigned         "setct-PIDataUnsigned"
#define NID_setct_PIDataUnsigned                525
#define OBJ_setct_PIDataUnsigned                OBJ_set_ctype,6L

#define SN_setct_HODInput               "setct-HODInput"
#define NID_setct_HODInput              526
#define OBJ_setct_HODInput              OBJ_set_ctype,7L

#define SN_setct_AuthResBaggage         "setct-AuthResBaggage"
#define NID_setct_AuthResBaggage                527
#define OBJ_setct_AuthResBaggage                OBJ_set_ctype,8L

#define SN_setct_AuthRevReqBaggage              "setct-AuthRevReqBaggage"
#define NID_setct_AuthRevReqBaggage             528
#define OBJ_setct_AuthRevReqBaggage             OBJ_set_ctype,9L

#define SN_setct_AuthRevResBaggage              "setct-AuthRevResBaggage"
#define NID_setct_AuthRevResBaggage             529
#define OBJ_setct_AuthRevResBaggage             OBJ_set_ctype,10L

#define SN_setct_CapTokenSeq            "setct-CapTokenSeq"
#define NID_setct_CapTokenSeq           530
#define OBJ_setct_CapTokenSeq           OBJ_set_ctype,11L

#define SN_setct_PInitResData           "setct-PInitResData"
#define NID_setct_PInitResData          531
#define OBJ_setct_PInitResData          OBJ_set_ctype,12L

#define SN_setct_PI_TBS         "setct-PI-TBS"
#define NID_setct_PI_TBS                532
#define OBJ_setct_PI_TBS                OBJ_set_ctype,13L

#define SN_setct_PResData               "setct-PResData"
#define NID_setct_PResData              533
#define OBJ_setct_PResData              OBJ_set_ctype,14L

#define SN_setct_AuthReqTBS             "setct-AuthReqTBS"
#define NID_setct_AuthReqTBS            534
#define OBJ_setct_AuthReqTBS            OBJ_set_ctype,16L

#define SN_setct_AuthResTBS             "setct-AuthResTBS"
#define NID_setct_AuthResTBS            535
#define OBJ_setct_AuthResTBS            OBJ_set_ctype,17L

#define SN_setct_AuthResTBSX            "setct-AuthResTBSX"
#define NID_setct_AuthResTBSX           536
#define OBJ_setct_AuthResTBSX           OBJ_set_ctype,18L

#define SN_setct_AuthTokenTBS           "setct-AuthTokenTBS"
#define NID_setct_AuthTokenTBS          537
#define OBJ_setct_AuthTokenTBS          OBJ_set_ctype,19L

#define SN_setct_CapTokenData           "setct-CapTokenData"
#define NID_setct_CapTokenData          538
#define OBJ_setct_CapTokenData          OBJ_set_ctype,20L

#define SN_setct_CapTokenTBS            "setct-CapTokenTBS"
#define NID_setct_CapTokenTBS           539
#define OBJ_setct_CapTokenTBS           OBJ_set_ctype,21L

#define SN_setct_AcqCardCodeMsg         "setct-AcqCardCodeMsg"
#define NID_setct_AcqCardCodeMsg                540
#define OBJ_setct_AcqCardCodeMsg                OBJ_set_ctype,22L

#define SN_setct_AuthRevReqTBS          "setct-AuthRevReqTBS"
```



```
#define NID_setct_AuthRevReqTBS         541
#define OBJ_setct_AuthRevReqTBS         OBJ_set_ctype,23L

#define SN_setct_AuthRevResData         "setct-AuthRevResData"
#define NID_setct_AuthRevResData                542
#define OBJ_setct_AuthRevResData                OBJ_set_ctype,24L

#define SN_setct_AuthRevResTBS          "setct-AuthRevResTBS"
#define NID_setct_AuthRevResTBS         543
#define OBJ_setct_AuthRevResTBS         OBJ_set_ctype,25L

#define SN_setct_CapReqTBS              "setct-CapReqTBS"
#define NID_setct_CapReqTBS             544
#define OBJ_setct_CapReqTBS             OBJ_set_ctype,26L

#define SN_setct_CapReqTBSX             "setct-CapReqTBSX"
#define NID_setct_CapReqTBSX            545
#define OBJ_setct_CapReqTBSX            OBJ_set_ctype,27L

#define SN_setct_CapResData             "setct-CapResData"
#define NID_setct_CapResData            546
#define OBJ_setct_CapResData            OBJ_set_ctype,28L

#define SN_setct_CapRevReqTBS           "setct-CapRevReqTBS"
#define NID_setct_CapRevReqTBS          547
#define OBJ_setct_CapRevReqTBS          OBJ_set_ctype,29L

#define SN_setct_CapRevReqTBSX          "setct-CapRevReqTBSX"
#define NID_setct_CapRevReqTBSX         548
#define OBJ_setct_CapRevReqTBSX         OBJ_set_ctype,30L

#define SN_setct_CapRevResData          "setct-CapRevResData"
#define NID_setct_CapRevResData         549
#define OBJ_setct_CapRevResData         OBJ_set_ctype,31L

#define SN_setct_CredReqTBS             "setct-CredReqTBS"
#define NID_setct_CredReqTBS            550
#define OBJ_setct_CredReqTBS            OBJ_set_ctype,32L

#define SN_setct_CredReqTBSX            "setct-CredReqTBSX"
#define NID_setct_CredReqTBSX           551
#define OBJ_setct_CredReqTBSX           OBJ_set_ctype,33L

#define SN_setct_CredResData            "setct-CredResData"
#define NID_setct_CredResData           552
#define OBJ_setct_CredResData           OBJ_set_ctype,34L

#define SN_setct_CredRevReqTBS          "setct-CredRevReqTBS"
#define NID_setct_CredRevReqTBS         553
#define OBJ_setct_CredRevReqTBS         OBJ_set_ctype,35L

#define SN_setct_CredRevReqTBSX         "setct-CredRevReqTBSX"
#define NID_setct_CredRevReqTBSX                554
#define OBJ_setct_CredRevReqTBSX                OBJ_set_ctype,36L

#define SN_setct_CredRevResData         "setct-CredRevResData"
#define NID_setct_CredRevResData                555
#define OBJ_setct_CredRevResData                OBJ_set_ctype,37L

#define SN_setct_PCertReqData           "setct-PCertReqData"
#define NID_setct_PCertReqData          556
#define OBJ_setct_PCertReqData          OBJ_set_ctype,38L

#define SN_setct_PCertResTBS            "setct-PCertResTBS"
#define NID_setct_PCertResTBS           557
#define OBJ_setct_PCertResTBS           OBJ_set_ctype,39L
```

```
3297 #define SN_setct_BatchAdminReqData              "setct-BatchAdminReqData"
3298 #define NID_setct_BatchAdminReqData             558
3299 #define OBJ_setct_BatchAdminReqData             OBJ_set_ctype,40L

3301 #define SN_setct_BatchAdminResData              "setct-BatchAdminResData"
3302 #define NID_setct_BatchAdminResData             559
3303 #define OBJ_setct_BatchAdminResData             OBJ_set_ctype,41L

3305 #define SN_setct_CardCInitResTBS                "setct-CardCInitResTBS"
3306 #define NID_setct_CardCInitResTBS               560
3307 #define OBJ_setct_CardCInitResTBS               OBJ_set_ctype,42L

3309 #define SN_setct_MeAqCInitResTBS                "setct-MeAqCInitResTBS"
3310 #define NID_setct_MeAqCInitResTBS               561
3311 #define OBJ_setct_MeAqCInitResTBS               OBJ_set_ctype,43L

3313 #define SN_setct_RegFormResTBS          "setct-RegFormResTBS"
3314 #define NID_setct_RegFormResTBS         562
3315 #define OBJ_setct_RegFormResTBS         OBJ_set_ctype,44L

3317 #define SN_setct_CertReqData            "setct-CertReqData"
3318 #define NID_setct_CertReqData           563
3319 #define OBJ_setct_CertReqData           OBJ_set_ctype,45L

3321 #define SN_setct_CertReqTBS             "setct-CertReqTBS"
3322 #define NID_setct_CertReqTBS            564
3323 #define OBJ_setct_CertReqTBS            OBJ_set_ctype,46L

3325 #define SN_setct_CertResData            "setct-CertResData"
3326 #define NID_setct_CertResData           565
3327 #define OBJ_setct_CertResData           OBJ_set_ctype,47L

3329 #define SN_setct_CertInqReqTBS          "setct-CertInqReqTBS"
3330 #define NID_setct_CertInqReqTBS         566
3331 #define OBJ_setct_CertInqReqTBS         OBJ_set_ctype,48L

3333 #define SN_setct_ErrorTBS               "setct-ErrorTBS"
3334 #define NID_setct_ErrorTBS              567
3335 #define OBJ_setct_ErrorTBS              OBJ_set_ctype,49L

3337 #define SN_setct_PIDualSignedTBE                "setct-PIDualSignedTBE"
3338 #define NID_setct_PIDualSignedTBE               568
3339 #define OBJ_setct_PIDualSignedTBE               OBJ_set_ctype,50L

3341 #define SN_setct_PIUnsignedTBE          "setct-PIUnsignedTBE"
3342 #define NID_setct_PIUnsignedTBE         569
3343 #define OBJ_setct_PIUnsignedTBE         OBJ_set_ctype,51L

3345 #define SN_setct_AuthReqTBE             "setct-AuthReqTBE"
3346 #define NID_setct_AuthReqTBE            570
3347 #define OBJ_setct_AuthReqTBE            OBJ_set_ctype,52L

3349 #define SN_setct_AuthResTBE             "setct-AuthResTBE"
3350 #define NID_setct_AuthResTBE            571
3351 #define OBJ_setct_AuthResTBE            OBJ_set_ctype,53L

3353 #define SN_setct_AuthResTBEX            "setct-AuthResTBEX"
3354 #define NID_setct_AuthResTBEX           572
3355 #define OBJ_setct_AuthResTBEX           OBJ_set_ctype,54L

3357 #define SN_setct_AuthTokenTBE           "setct-AuthTokenTBE"
3358 #define NID_setct_AuthTokenTBE          573
3359 #define OBJ_setct_AuthTokenTBE          OBJ_set_ctype,55L

3361 #define SN_setct_CapTokenTBE            "setct-CapTokenTBE"
```



```
3362 #define NID_setct_CapTokenTBE           574
3363 #define OBJ_setct_CapTokenTBE           OBJ_set_ctype,56L

3365 #define SN_setct_CapTokenTBEX           "setct-CapTokenTBEX"
3366 #define NID_setct_CapTokenTBEX          575
3367 #define OBJ_setct_CapTokenTBEX          OBJ_set_ctype,57L

3369 #define SN_setct_AcqCardCodeMsgTBE              "setct-AcqCardCodeMsgTBE"
3370 #define NID_setct_AcqCardCodeMsgTBE             576
3371 #define OBJ_setct_AcqCardCodeMsgTBE             OBJ_set_ctype,58L

3373 #define SN_setct_AuthRevReqTBE          "setct-AuthRevReqTBE"
3374 #define NID_setct_AuthRevReqTBE         577
3375 #define OBJ_setct_AuthRevReqTBE         OBJ_set_ctype,59L

3377 #define SN_setct_AuthRevResTBE          "setct-AuthRevResTBE"
3378 #define NID_setct_AuthRevResTBE         578
3379 #define OBJ_setct_AuthRevResTBE         OBJ_set_ctype,60L

3381 #define SN_setct_AuthRevResTBEB         "setct-AuthRevResTBEB"
3382 #define NID_setct_AuthRevResTBEB                579
3383 #define OBJ_setct_AuthRevResTBEB                OBJ_set_ctype,61L

3385 #define SN_setct_CapReqTBE              "setct-CapReqTBE"
3386 #define NID_setct_CapReqTBE             580
3387 #define OBJ_setct_CapReqTBE             OBJ_set_ctype,62L

3389 #define SN_setct_CapReqTBEX             "setct-CapReqTBEX"
3390 #define NID_setct_CapReqTBEX            581
3391 #define OBJ_setct_CapReqTBEX            OBJ_set_ctype,63L

3393 #define SN_setct_CapResTBE              "setct-CapResTBE"
3394 #define NID_setct_CapResTBE             582
3395 #define OBJ_setct_CapResTBE             OBJ_set_ctype,64L

3397 #define SN_setct_CapRevReqTBE           "setct-CapRevReqTBE"
3398 #define NID_setct_CapRevReqTBE          583
3399 #define OBJ_setct_CapRevReqTBE          OBJ_set_ctype,65L

3401 #define SN_setct_CapRevReqTBEX          "setct-CapRevReqTBEX"
3402 #define NID_setct_CapRevReqTBEX         584
3403 #define OBJ_setct_CapRevReqTBEX         OBJ_set_ctype,66L

3405 #define SN_setct_CapRevResTBE           "setct-CapRevResTBE"
3406 #define NID_setct_CapRevResTBE          585
3407 #define OBJ_setct_CapRevResTBE          OBJ_set_ctype,67L

3409 #define SN_setct_CredReqTBE             "setct-CredReqTBE"
3410 #define NID_setct_CredReqTBE            586
3411 #define OBJ_setct_CredReqTBE            OBJ_set_ctype,68L

3413 #define SN_setct_CredReqTBEX            "setct-CredReqTBEX"
3414 #define NID_setct_CredReqTBEX           587
3415 #define OBJ_setct_CredReqTBEX           OBJ_set_ctype,69L

3417 #define SN_setct_CredResTBE             "setct-CredResTBE"
3418 #define NID_setct_CredResTBE            588
3419 #define OBJ_setct_CredResTBE            OBJ_set_ctype,70L

3421 #define SN_setct_CredRevReqTBE          "setct-CredRevReqTBE"
3422 #define NID_setct_CredRevReqTBE         589
3423 #define OBJ_setct_CredRevReqTBE         OBJ_set_ctype,71L

3425 #define SN_setct_CredRevReqTBEX         "setct-CredRevReqTBEX"
3426 #define NID_setct_CredRevReqTBEX                590
3427 #define OBJ_setct_CredRevReqTBEX                OBJ_set_ctype,72L
```

```
3429 #define SN_setct_CredRevResTBE          "setct-CredRevResTBE"
3430 #define NID_setct_CredRevResTBE         591
3431 #define OBJ_setct_CredRevResTBE         OBJ_set_ctype,73L

3433 #define SN_setct_BatchAdminReqTBE               "setct-BatchAdminReqTBE"
3434 #define NID_setct_BatchAdminReqTBE              592
3435 #define OBJ_setct_BatchAdminReqTBE              OBJ_set_ctype,74L

3437 #define SN_setct_BatchAdminResTBE               "setct-BatchAdminResTBE"
3438 #define NID_setct_BatchAdminResTBE              593
3439 #define OBJ_setct_BatchAdminResTBE              OBJ_set_ctype,75L

3441 #define SN_setct_RegFormReqTBE          "setct-RegFormReqTBE"
3442 #define NID_setct_RegFormReqTBE         594
3443 #define OBJ_setct_RegFormReqTBE         OBJ_set_ctype,76L

3445 #define SN_setct_CertReqTBE             "setct-CertReqTBE"
3446 #define NID_setct_CertReqTBE            595
3447 #define OBJ_setct_CertReqTBE            OBJ_set_ctype,77L

3449 #define SN_setct_CertReqTBEX            "setct-CertReqTBEX"
3450 #define NID_setct_CertReqTBEX           596
3451 #define OBJ_setct_CertReqTBEX           OBJ_set_ctype,78L

3453 #define SN_setct_CertResTBE             "setct-CertResTBE"
3454 #define NID_setct_CertResTBE            597
3455 #define OBJ_setct_CertResTBE            OBJ_set_ctype,79L

3457 #define SN_setct_CRLNotificationTBS             "setct-CRLNotificationTBS"
3458 #define NID_setct_CRLNotificationTBS            598
3459 #define OBJ_setct_CRLNotificationTBS            OBJ_set_ctype,80L

3461 #define SN_setct_CRLNotificationResTBS          "setct-CRLNotificationResTBS"
3462 #define NID_setct_CRLNotificationResTBS         599
3463 #define OBJ_setct_CRLNotificationResTBS         OBJ_set_ctype,81L

3465 #define SN_setct_BCIDistributionTBS             "setct-BCIDistributionTBS"
3466 #define NID_setct_BCIDistributionTBS            600
3467 #define OBJ_setct_BCIDistributionTBS            OBJ_set_ctype,82L

3469 #define SN_setext_genCrypt              "setext-genCrypt"
3470 #define LN_setext_genCrypt              "generic cryptogram"
3471 #define NID_setext_genCrypt             601
3472 #define OBJ_setext_genCrypt             OBJ_set_msgExt,1L

3474 #define SN_setext_miAuth                "setext-miAuth"
3475 #define LN_setext_miAuth                "merchant initiated auth"
3476 #define NID_setext_miAuth               602
3477 #define OBJ_setext_miAuth               OBJ_set_msgExt,3L

3479 #define SN_setext_pinSecure             "setext-pinSecure"
3480 #define NID_setext_pinSecure            603
3481 #define OBJ_setext_pinSecure            OBJ_set_msgExt,4L

3483 #define SN_setext_pinAny                "setext-pinAny"
3484 #define NID_setext_pinAny               604
3485 #define OBJ_setext_pinAny               OBJ_set_msgExt,5L

3487 #define SN_setext_track2                "setext-track2"
3488 #define NID_setext_track2               605
3489 #define OBJ_setext_track2               OBJ_set_msgExt,7L

3491 #define SN_setext_cv            "setext-cv"
3492 #define LN_setext_cv            "additional verification"
3493 #define NID_setext_cv           606
```



```
3494 #define OBJ_setext_cv           OBJ_set_msgExt,8L

3496 #define SN_set_policy_root              "set-policy-root"
3497 #define NID_set_policy_root             607
3498 #define OBJ_set_policy_root             OBJ_set_policy,0L

3500 #define SN_setCext_hashedRoot           "setCext-hashedRoot"
3501 #define NID_setCext_hashedRoot          608
3502 #define OBJ_setCext_hashedRoot          OBJ_set_certExt,0L

3504 #define SN_setCext_certType             "setCext-certType"
3505 #define NID_setCext_certType            609
3506 #define OBJ_setCext_certType            OBJ_set_certExt,1L

3508 #define SN_setCext_merchData            "setCext-merchData"
3509 #define NID_setCext_merchData           610
3510 #define OBJ_setCext_merchData           OBJ_set_certExt,2L

3512 #define SN_setCext_cCertRequired                "setCext-cCertRequired"
3513 #define NID_setCext_cCertRequired               611
3514 #define OBJ_setCext_cCertRequired               OBJ_set_certExt,3L

3516 #define SN_setCext_tunneling            "setCext-tunneling"
3517 #define NID_setCext_tunneling           612
3518 #define OBJ_setCext_tunneling           OBJ_set_certExt,4L

3520 #define SN_setCext_setExt               "setCext-setExt"
3521 #define NID_setCext_setExt              613
3522 #define OBJ_setCext_setExt              OBJ_set_certExt,5L

3524 #define SN_setCext_setQualf             "setCext-setQualf"
3525 #define NID_setCext_setQualf            614
3526 #define OBJ_setCext_setQualf            OBJ_set_certExt,6L

3528 #define SN_setCext_PGWYcapabilities             "setCext-PGWYcapabilities"
3529 #define NID_setCext_PGWYcapabilities            615
3530 #define OBJ_setCext_PGWYcapabilities            OBJ_set_certExt,7L

3532 #define SN_setCext_TokenIdentifier              "setCext-TokenIdentifier"
3533 #define NID_setCext_TokenIdentifier             616
3534 #define OBJ_setCext_TokenIdentifier             OBJ_set_certExt,8L

3536 #define SN_setCext_Track2Data           "setCext-Track2Data"
3537 #define NID_setCext_Track2Data          617
3538 #define OBJ_setCext_Track2Data          OBJ_set_certExt,9L

3540 #define SN_setCext_TokenType            "setCext-TokenType"
3541 #define NID_setCext_TokenType           618
3542 #define OBJ_setCext_TokenType           OBJ_set_certExt,10L

3544 #define SN_setCext_IssuerCapabilities           "setCext-IssuerCapabilities"
3545 #define NID_setCext_IssuerCapabilities          619
3546 #define OBJ_setCext_IssuerCapabilities          OBJ_set_certExt,11L

3548 #define SN_setAttr_Cert         "setAttr-Cert"
3549 #define NID_setAttr_Cert                620
3550 #define OBJ_setAttr_Cert                OBJ_set_attr,0L

3552 #define SN_setAttr_PGWYcap              "setAttr-PGWYcap"
3553 #define LN_setAttr_PGWYcap              "payment gateway capabilities"
3554 #define NID_setAttr_PGWYcap             621
3555 #define OBJ_setAttr_PGWYcap             OBJ_set_attr,1L

3557 #define SN_setAttr_TokenType            "setAttr-TokenType"
3558 #define NID_setAttr_TokenType           622
3559 #define OBJ_setAttr_TokenType           OBJ_set_attr,2L
```

```
3561 #define SN_setAttr_IssCap               "setAttr-IssCap"
3562 #define LN_setAttr_IssCap               "issuer capabilities"
3563 #define NID_setAttr_IssCap              623
3564 #define OBJ_setAttr_IssCap              OBJ_set_attr,3L

3566 #define SN_set_rootKeyThumb             "set-rootKeyThumb"
3567 #define NID_set_rootKeyThumb            624
3568 #define OBJ_set_rootKeyThumb            OBJ_setAttr_Cert,0L

3570 #define SN_set_addPolicy                "set-addPolicy"
3571 #define NID_set_addPolicy               625
3572 #define OBJ_set_addPolicy               OBJ_setAttr_Cert,1L

3574 #define SN_setAttr_Token_EMV            "setAttr-Token-EMV"
3575 #define NID_setAttr_Token_EMV           626
3576 #define OBJ_setAttr_Token_EMV           OBJ_setAttr_TokenType,1L

3578 #define SN_setAttr_Token_B0Prime                "setAttr-Token-B0Prime"
3579 #define NID_setAttr_Token_B0Prime               627
3580 #define OBJ_setAttr_Token_B0Prime               OBJ_setAttr_TokenType,2L

3582 #define SN_setAttr_IssCap_CVM           "setAttr-IssCap-CVM"
3583 #define NID_setAttr_IssCap_CVM          628
3584 #define OBJ_setAttr_IssCap_CVM          OBJ_setAttr_IssCap,3L

3586 #define SN_setAttr_IssCap_T2            "setAttr-IssCap-T2"
3587 #define NID_setAttr_IssCap_T2           629
3588 #define OBJ_setAttr_IssCap_T2           OBJ_setAttr_IssCap,4L

3590 #define SN_setAttr_IssCap_Sig           "setAttr-IssCap-Sig"
3591 #define NID_setAttr_IssCap_Sig          630
3592 #define OBJ_setAttr_IssCap_Sig          OBJ_setAttr_IssCap,5L

3594 #define SN_setAttr_GenCryptgrm          "setAttr-GenCryptgrm"
3595 #define LN_setAttr_GenCryptgrm          "generate cryptogram"
3596 #define NID_setAttr_GenCryptgrm         631
3597 #define OBJ_setAttr_GenCryptgrm         OBJ_setAttr_IssCap_CVM,1L

3599 #define SN_setAttr_T2Enc                "setAttr-T2Enc"
3600 #define LN_setAttr_T2Enc                "encrypted track 2"
3601 #define NID_setAttr_T2Enc               632
3602 #define OBJ_setAttr_T2Enc               OBJ_setAttr_IssCap_T2,1L

3604 #define SN_setAttr_T2cleartxt           "setAttr-T2cleartxt"
3605 #define LN_setAttr_T2cleartxt           "cleartext track 2"
3606 #define NID_setAttr_T2cleartxt          633
3607 #define OBJ_setAttr_T2cleartxt          OBJ_setAttr_IssCap_T2,2L

3609 #define SN_setAttr_TokICCsig            "setAttr-TokICCsig"
3610 #define LN_setAttr_TokICCsig            "ICC or token signature"
3611 #define NID_setAttr_TokICCsig           634
3612 #define OBJ_setAttr_TokICCsig           OBJ_setAttr_IssCap_Sig,1L

3614 #define SN_setAttr_SecDevSig            "setAttr-SecDevSig"
3615 #define LN_setAttr_SecDevSig            "secure device signature"
3616 #define NID_setAttr_SecDevSig           635
3617 #define OBJ_setAttr_SecDevSig           OBJ_setAttr_IssCap_Sig,2L

3619 #define SN_set_brand_IATA_ATA           "set-brand-IATA-ATA"
3620 #define NID_set_brand_IATA_ATA          636
3621 #define OBJ_set_brand_IATA_ATA          OBJ_set_brand,1L

3623 #define SN_set_brand_Diners             "set-brand-Diners"
3624 #define NID_set_brand_Diners            637
3625 #define OBJ_set_brand_Diners            OBJ_set_brand,30L
```



```
3627 #define SN_set_brand_AmericanExpress            "set-brand-AmericanExpress"
3628 #define NID_set_brand_AmericanExpress           638
3629 #define OBJ_set_brand_AmericanExpress           OBJ_set_brand,34L

3631 #define SN_set_brand_JCB                "set-brand-JCB"
3632 #define NID_set_brand_JCB               639
3633 #define OBJ_set_brand_JCB               OBJ_set_brand,35L

3635 #define SN_set_brand_Visa               "set-brand-Visa"
3636 #define NID_set_brand_Visa              640
3637 #define OBJ_set_brand_Visa              OBJ_set_brand,4L

3639 #define SN_set_brand_MasterCard         "set-brand-MasterCard"
3640 #define NID_set_brand_MasterCard                641
3641 #define OBJ_set_brand_MasterCard                OBJ_set_brand,5L

3643 #define SN_set_brand_Novus              "set-brand-Novus"
3644 #define NID_set_brand_Novus             642
3645 #define OBJ_set_brand_Novus             OBJ_set_brand,6011L

3647 #define SN_des_cdmf             "DES-CDMF"
3648 #define LN_des_cdmf             "des-cdmf"
3649 #define NID_des_cdmf            643
3650 #define OBJ_des_cdmf            OBJ_rsadsi,3L,10L

3652 #define SN_rsaOAEPEncryptionSET         "rsaOAEPEncryptionSET"
3653 #define NID_rsaOAEPEncryptionSET                644
3654 #define OBJ_rsaOAEPEncryptionSET                OBJ_rsadsi,1L,1L,6L

3656 #define SN_ipsec3               "Oakley-EC2N-3"
3657 #define LN_ipsec3               "ipsec3"
3658 #define NID_ipsec3              749

3660 #define SN_ipsec4               "Oakley-EC2N-4"
3661 #define LN_ipsec4               "ipsec4"
3662 #define NID_ipsec4              750

3664 #define SN_whirlpool            "whirlpool"
3665 #define NID_whirlpool           804
3666 #define OBJ_whirlpool           OBJ_iso,0L,10118L,3L,0L,55L

3668 #define SN_cryptopro            "cryptopro"
3669 #define NID_cryptopro           805
3670 #define OBJ_cryptopro           OBJ_member_body,643L,2L,2L

3672 #define SN_cryptocom            "cryptocom"
3673 #define NID_cryptocom           806
3674 #define OBJ_cryptocom           OBJ_member_body,643L,2L,9L

3676 #define SN_id_GostR3411_94_with_GostR3410_2001          "id-GostR3411-94-with-Go
3677 #define LN_id_GostR3411_94_with_GostR3410_2001          "GOST R 34.11-94 with GO
3678 #define NID_id_GostR3411_94_with_GostR3410_2001         807
3679 #define OBJ_id_GostR3411_94_with_GostR3410_2001         OBJ_cryptopro,3L

3681 #define SN_id_GostR3411_94_with_GostR3410_94            "id-GostR3411-94-with-Go
3682 #define LN_id_GostR3411_94_with_GostR3410_94            "GOST R 34.11-94 with GO
3683 #define NID_id_GostR3411_94_with_GostR3410_94           808
3684 #define OBJ_id_GostR3411_94_with_GostR3410_94           OBJ_cryptopro,4L

3686 #define SN_id_GostR3411_94              "md_gost94"
3687 #define LN_id_GostR3411_94              "GOST R 34.11-94"
3688 #define NID_id_GostR3411_94             809
3689 #define OBJ_id_GostR3411_94             OBJ_cryptopro,9L

3691 #define SN_id_HMACGostR3411_94          "id-HMACGostR3411-94"
```

```
3692 #define LN_id_HMACGostR3411_94          "HMAC GOST 34.11-94"
3693 #define NID_id_HMACGostR3411_94         810
3694 #define OBJ_id_HMACGostR3411_94         OBJ_cryptopro,10L

3696 #define SN_id_GostR3410_2001            "gost2001"
3697 #define LN_id_GostR3410_2001            "GOST R 34.10-2001"
3698 #define NID_id_GostR3410_2001           811
3699 #define OBJ_id_GostR3410_2001           OBJ_cryptopro,19L

3701 #define SN_id_GostR3410_94              "gost94"
3702 #define LN_id_GostR3410_94              "GOST R 34.10-94"
3703 #define NID_id_GostR3410_94             812
3704 #define OBJ_id_GostR3410_94             OBJ_cryptopro,20L

3706 #define SN_id_Gost28147_89              "gost89"
3707 #define LN_id_Gost28147_89              "GOST 28147-89"
3708 #define NID_id_Gost28147_89             813
3709 #define OBJ_id_Gost28147_89             OBJ_cryptopro,21L

3711 #define SN_gost89_cnt           "gost89-cnt"
3712 #define NID_gost89_cnt          814

3714 #define SN_id_Gost28147_89_MAC          "gost-mac"
3715 #define LN_id_Gost28147_89_MAC          "GOST 28147-89 MAC"
3716 #define NID_id_Gost28147_89_MAC         815
3717 #define OBJ_id_Gost28147_89_MAC         OBJ_cryptopro,22L

3719 #define SN_id_GostR3411_94_prf          "prf-gostr3411-94"
3720 #define LN_id_GostR3411_94_prf          "GOST R 34.11-94 PRF"
3721 #define NID_id_GostR3411_94_prf         816
3722 #define OBJ_id_GostR3411_94_prf         OBJ_cryptopro,23L

3724 #define SN_id_GostR3410_2001DH          "id-GostR3410-2001DH"
3725 #define LN_id_GostR3410_2001DH          "GOST R 34.10-2001 DH"
3726 #define NID_id_GostR3410_2001DH         817
3727 #define OBJ_id_GostR3410_2001DH         OBJ_cryptopro,98L

3729 #define SN_id_GostR3410_94DH            "id-GostR3410-94DH"
3730 #define LN_id_GostR3410_94DH            "GOST R 34.10-94 DH"
3731 #define NID_id_GostR3410_94DH           818
3732 #define OBJ_id_GostR3410_94DH           OBJ_cryptopro,99L

3734 #define SN_id_Gost28147_89_CryptoPro_KeyMeshing         "id-Gost28147-89-CryptoP
3735 #define NID_id_Gost28147_89_CryptoPro_KeyMeshing                819
3736 #define OBJ_id_Gost28147_89_CryptoPro_KeyMeshing                OBJ_cryptopro,14

3738 #define SN_id_Gost28147_89_None_KeyMeshing              "id-Gost28147-89-None-Ke
3739 #define NID_id_Gost28147_89_None_KeyMeshing             820
3740 #define OBJ_id_Gost28147_89_None_KeyMeshing             OBJ_cryptopro,14L,0L

3742 #define SN_id_GostR3411_94_TestParamSet         "id-GostR3411-94-TestParamSet"
3743 #define NID_id_GostR3411_94_TestParamSet                821
3744 #define OBJ_id_GostR3411_94_TestParamSet                OBJ_cryptopro,30L,0L

3746 #define SN_id_GostR3411_94_CryptoProParamSet            "id-GostR3411-94-CryptoP
3747 #define NID_id_GostR3411_94_CryptoProParamSet           822
3748 #define OBJ_id_GostR3411_94_CryptoProParamSet           OBJ_cryptopro,30L,1L

3750 #define SN_id_Gost28147_89_TestParamSet         "id-Gost28147-89-TestParamSet"
3751 #define NID_id_Gost28147_89_TestParamSet                823
3752 #define OBJ_id_Gost28147_89_TestParamSet                OBJ_cryptopro,31L,0L

3754 #define SN_id_Gost28147_89_CryptoPro_A_ParamSet         "id-Gost28147-89-CryptoP
3755 #define NID_id_Gost28147_89_CryptoPro_A_ParamSet                824
3756 #define OBJ_id_Gost28147_89_CryptoPro_A_ParamSet                OBJ_cryptopro,31
```



```
3758 #define SN_id_Gost28147_89_CryptoPro_B_ParamSet         "id-Gost28147-89-CryptoP
3759 #define NID_id_Gost28147_89_CryptoPro_B_ParamSet                825
3760 #define OBJ_id_Gost28147_89_CryptoPro_B_ParamSet                OBJ_cryptopro,31

3762 #define SN_id_Gost28147_89_CryptoPro_C_ParamSet         "id-Gost28147-89-CryptoP
3763 #define NID_id_Gost28147_89_CryptoPro_C_ParamSet                826
3764 #define OBJ_id_Gost28147_89_CryptoPro_C_ParamSet                OBJ_cryptopro,31

3766 #define SN_id_Gost28147_89_CryptoPro_D_ParamSet         "id-Gost28147-89-CryptoP
3767 #define NID_id_Gost28147_89_CryptoPro_D_ParamSet                827
3768 #define OBJ_id_Gost28147_89_CryptoPro_D_ParamSet                OBJ_cryptopro,31

3770 #define SN_id_Gost28147_89_CryptoPro_Oscar_1_1_ParamSet         "id-Gost28147-89
3771 #define NID_id_Gost28147_89_CryptoPro_Oscar_1_1_ParamSet                828
3772 #define OBJ_id_Gost28147_89_CryptoPro_Oscar_1_1_ParamSet                OBJ_cryp

3774 #define SN_id_Gost28147_89_CryptoPro_Oscar_1_0_ParamSet         "id-Gost28147-89
3775 #define NID_id_Gost28147_89_CryptoPro_Oscar_1_0_ParamSet                829
3776 #define OBJ_id_Gost28147_89_CryptoPro_Oscar_1_0_ParamSet                OBJ_cryp

3778 #define SN_id_Gost28147_89_CryptoPro_RIC_1_ParamSet             "id-Gost28147-89
3779 #define NID_id_Gost28147_89_CryptoPro_RIC_1_ParamSet            830
3780 #define OBJ_id_Gost28147_89_CryptoPro_RIC_1_ParamSet            OBJ_cryptopro,31

3782 #define SN_id_GostR3410_94_TestParamSet         "id-GostR3410-94-TestParamSet"
3783 #define NID_id_GostR3410_94_TestParamSet                831
3784 #define OBJ_id_GostR3410_94_TestParamSet                OBJ_cryptopro,32L,0L

3786 #define SN_id_GostR3410_94_CryptoPro_A_ParamSet         "id-GostR3410-94-CryptoP
3787 #define NID_id_GostR3410_94_CryptoPro_A_ParamSet                832
3788 #define OBJ_id_GostR3410_94_CryptoPro_A_ParamSet                OBJ_cryptopro,32

3790 #define SN_id_GostR3410_94_CryptoPro_B_ParamSet         "id-GostR3410-94-CryptoP
3791 #define NID_id_GostR3410_94_CryptoPro_B_ParamSet                833
3792 #define OBJ_id_GostR3410_94_CryptoPro_B_ParamSet                OBJ_cryptopro,32

3794 #define SN_id_GostR3410_94_CryptoPro_C_ParamSet         "id-GostR3410-94-CryptoP
3795 #define NID_id_GostR3410_94_CryptoPro_C_ParamSet                834
3796 #define OBJ_id_GostR3410_94_CryptoPro_C_ParamSet                OBJ_cryptopro,32

3798 #define SN_id_GostR3410_94_CryptoPro_D_ParamSet         "id-GostR3410-94-CryptoP
3799 #define NID_id_GostR3410_94_CryptoPro_D_ParamSet                835
3800 #define OBJ_id_GostR3410_94_CryptoPro_D_ParamSet                OBJ_cryptopro,32

3802 #define SN_id_GostR3410_94_CryptoPro_XchA_ParamSet              "id-GostR3410-94
3803 #define NID_id_GostR3410_94_CryptoPro_XchA_ParamSet             836
3804 #define OBJ_id_GostR3410_94_CryptoPro_XchA_ParamSet             OBJ_cryptopro,33

3806 #define SN_id_GostR3410_94_CryptoPro_XchB_ParamSet              "id-GostR3410-94
3807 #define NID_id_GostR3410_94_CryptoPro_XchB_ParamSet             837
3808 #define OBJ_id_GostR3410_94_CryptoPro_XchB_ParamSet             OBJ_cryptopro,33

3810 #define SN_id_GostR3410_94_CryptoPro_XchC_ParamSet              "id-GostR3410-94
3811 #define NID_id_GostR3410_94_CryptoPro_XchC_ParamSet             838
3812 #define OBJ_id_GostR3410_94_CryptoPro_XchC_ParamSet             OBJ_cryptopro,33

3814 #define SN_id_GostR3410_2001_TestParamSet               "id-GostR3410-2001-TestP
3815 #define NID_id_GostR3410_2001_TestParamSet              839
3816 #define OBJ_id_GostR3410_2001_TestParamSet              OBJ_cryptopro,35L,0L

3818 #define SN_id_GostR3410_2001_CryptoPro_A_ParamSet               "id-GostR3410-20
3819 #define NID_id_GostR3410_2001_CryptoPro_A_ParamSet              840
3820 #define OBJ_id_GostR3410_2001_CryptoPro_A_ParamSet              OBJ_cryptopro,35

3822 #define SN_id_GostR3410_2001_CryptoPro_B_ParamSet               "id-GostR3410-20
3823 #define NID_id_GostR3410_2001_CryptoPro_B_ParamSet              841
```

```
#define OBJ_id_GostR3410_2001_CryptoPro_B_ParamSet              OBJ_cryptopro,35

#define SN_id_GostR3410_2001_CryptoPro_C_ParamSet               "id-GostR3410-20
#define NID_id_GostR3410_2001_CryptoPro_C_ParamSet              842
#define OBJ_id_GostR3410_2001_CryptoPro_C_ParamSet              OBJ_cryptopro,35

#define SN_id_GostR3410_2001_CryptoPro_XchA_ParamSet            "id-GostR3410-20
#define NID_id_GostR3410_2001_CryptoPro_XchA_ParamSet           843
#define OBJ_id_GostR3410_2001_CryptoPro_XchA_ParamSet           OBJ_cryptopro,36

#define SN_id_GostR3410_2001_CryptoPro_XchB_ParamSet            "id-GostR3410-20
#define NID_id_GostR3410_2001_CryptoPro_XchB_ParamSet           844
#define OBJ_id_GostR3410_2001_CryptoPro_XchB_ParamSet           OBJ_cryptopro,36

#define SN_id_GostR3410_94_a            "id-GostR3410-94-a"
#define NID_id_GostR3410_94_a           845
#define OBJ_id_GostR3410_94_a           OBJ_id_GostR3410_94,1L

#define SN_id_GostR3410_94_aBis         "id-GostR3410-94-aBis"
#define NID_id_GostR3410_94_aBis                846
#define OBJ_id_GostR3410_94_aBis                OBJ_id_GostR3410_94,2L

#define SN_id_GostR3410_94_b            "id-GostR3410-94-b"
#define NID_id_GostR3410_94_b           847
#define OBJ_id_GostR3410_94_b           OBJ_id_GostR3410_94,3L

#define SN_id_GostR3410_94_bBis         "id-GostR3410-94-bBis"
#define NID_id_GostR3410_94_bBis                848
#define OBJ_id_GostR3410_94_bBis                OBJ_id_GostR3410_94,4L

#define SN_id_Gost28147_89_cc           "id-Gost28147-89-cc"
#define LN_id_Gost28147_89_cc           "GOST 28147-89 Cryptocom ParamSet"
#define NID_id_Gost28147_89_cc          849
#define OBJ_id_Gost28147_89_cc          OBJ_cryptocom,1L,6L,1L

#define SN_id_GostR3410_94_cc           "gost94cc"
#define LN_id_GostR3410_94_cc           "GOST 34.10-94 Cryptocom"
#define NID_id_GostR3410_94_cc          850
#define OBJ_id_GostR3410_94_cc          OBJ_cryptocom,1L,5L,3L

#define SN_id_GostR3410_2001_cc         "gost2001cc"
#define LN_id_GostR3410_2001_cc         "GOST 34.10-2001 Cryptocom"
#define NID_id_GostR3410_2001_cc                851
#define OBJ_id_GostR3410_2001_cc                OBJ_cryptocom,1L,5L,4L

#define SN_id_GostR3411_94_with_GostR3410_94_cc         "id-GostR3411-94-with-Go
#define LN_id_GostR3411_94_with_GostR3410_94_cc         "GOST R 34.11-94 with GO
#define NID_id_GostR3411_94_with_GostR3410_94_cc                852
#define OBJ_id_GostR3411_94_with_GostR3410_94_cc                OBJ_cryptocom,1L

#define SN_id_GostR3411_94_with_GostR3410_2001_cc               "id-GostR3411-94
#define LN_id_GostR3411_94_with_GostR3410_2001_cc               "GOST R 34.11-94
#define NID_id_GostR3411_94_with_GostR3410_2001_cc              853
#define OBJ_id_GostR3411_94_with_GostR3410_2001_cc              OBJ_cryptocom,1L

#define SN_id_GostR3410_2001_ParamSet_cc                "id-GostR3410-2001-Param
#define LN_id_GostR3410_2001_ParamSet_cc                "GOST R 3410-2001 Parame
#define NID_id_GostR3410_2001_ParamSet_cc               854
#define OBJ_id_GostR3410_2001_ParamSet_cc               OBJ_cryptocom,1L,8L,1L

#define SN_camellia_128_cbc             "CAMELLIA-128-CBC"
#define LN_camellia_128_cbc             "camellia-128-cbc"
#define NID_camellia_128_cbc            751
#define OBJ_camellia_128_cbc            1L,2L,392L,200011L,61L,1L,1L,1L,2L

#define SN_camellia_192_cbc             "CAMELLIA-192-CBC"
```



```
#define LN_camellia_192_cbc             "camellia-192-cbc"
#define NID_camellia_192_cbc            752
#define OBJ_camellia_192_cbc            1L,2L,392L,200011L,61L,1L,1L,1L,3L

#define SN_camellia_256_cbc             "CAMELLIA-256-CBC"
#define LN_camellia_256_cbc             "camellia-256-cbc"
#define NID_camellia_256_cbc            753
#define OBJ_camellia_256_cbc            1L,2L,392L,200011L,61L,1L,1L,1L,4L

#define SN_id_camellia128_wrap          "id-camellia128-wrap"
#define NID_id_camellia128_wrap         907
#define OBJ_id_camellia128_wrap         1L,2L,392L,200011L,61L,1L,1L,3L,2L

#define SN_id_camellia192_wrap          "id-camellia192-wrap"
#define NID_id_camellia192_wrap         908
#define OBJ_id_camellia192_wrap         1L,2L,392L,200011L,61L,1L,1L,3L,3L

#define SN_id_camellia256_wrap          "id-camellia256-wrap"
#define NID_id_camellia256_wrap         909
#define OBJ_id_camellia256_wrap         1L,2L,392L,200011L,61L,1L,1L,3L,4L

#define OBJ_ntt_ds              0L,3L,4401L,5L

#define OBJ_camellia            OBJ_ntt_ds,3L,1L,9L

#define SN_camellia_128_ecb             "CAMELLIA-128-ECB"
#define LN_camellia_128_ecb             "camellia-128-ecb"
#define NID_camellia_128_ecb            754
#define OBJ_camellia_128_ecb            OBJ_camellia,1L

#define SN_camellia_128_ofb128          "CAMELLIA-128-OFB"
#define LN_camellia_128_ofb128          "camellia-128-ofb"
#define NID_camellia_128_ofb128         766
#define OBJ_camellia_128_ofb128         OBJ_camellia,3L

#define SN_camellia_128_cfb128          "CAMELLIA-128-CFB"
#define LN_camellia_128_cfb128          "camellia-128-cfb"
#define NID_camellia_128_cfb128         757
#define OBJ_camellia_128_cfb128         OBJ_camellia,4L

#define SN_camellia_192_ecb             "CAMELLIA-192-ECB"
#define LN_camellia_192_ecb             "camellia-192-ecb"
#define NID_camellia_192_ecb            755
#define OBJ_camellia_192_ecb            OBJ_camellia,21L

#define SN_camellia_192_ofb128          "CAMELLIA-192-OFB"
#define LN_camellia_192_ofb128          "camellia-192-ofb"
#define NID_camellia_192_ofb128         767
#define OBJ_camellia_192_ofb128         OBJ_camellia,23L

#define SN_camellia_192_cfb128          "CAMELLIA-192-CFB"
#define LN_camellia_192_cfb128          "camellia-192-cfb"
#define NID_camellia_192_cfb128         758
#define OBJ_camellia_192_cfb128         OBJ_camellia,24L

#define SN_camellia_256_ecb             "CAMELLIA-256-ECB"
#define LN_camellia_256_ecb             "camellia-256-ecb"
#define NID_camellia_256_ecb            756
#define OBJ_camellia_256_ecb            OBJ_camellia,41L

#define SN_camellia_256_ofb128          "CAMELLIA-256-OFB"
#define LN_camellia_256_ofb128          "camellia-256-ofb"
#define NID_camellia_256_ofb128         768
#define OBJ_camellia_256_ofb128         OBJ_camellia,43L

#define SN_camellia_256_cfb128          "CAMELLIA-256-CFB"
```

```
#define LN_camellia_256_cfb128          "camellia-256-cfb"
#define NID_camellia_256_cfb128         759
#define OBJ_camellia_256_cfb128         OBJ_camellia,44L

#define SN_camellia_128_cfb1            "CAMELLIA-128-CFB1"
#define LN_camellia_128_cfb1            "camellia-128-cfb1"
#define NID_camellia_128_cfb1           760

#define SN_camellia_192_cfb1            "CAMELLIA-192-CFB1"
#define LN_camellia_192_cfb1            "camellia-192-cfb1"
#define NID_camellia_192_cfb1           761

#define SN_camellia_256_cfb1            "CAMELLIA-256-CFB1"
#define LN_camellia_256_cfb1            "camellia-256-cfb1"
#define NID_camellia_256_cfb1           762

#define SN_camellia_128_cfb8            "CAMELLIA-128-CFB8"
#define LN_camellia_128_cfb8            "camellia-128-cfb8"
#define NID_camellia_128_cfb8           763

#define SN_camellia_192_cfb8            "CAMELLIA-192-CFB8"
#define LN_camellia_192_cfb8            "camellia-192-cfb8"
#define NID_camellia_192_cfb8           764

#define SN_camellia_256_cfb8            "CAMELLIA-256-CFB8"
#define LN_camellia_256_cfb8            "camellia-256-cfb8"
#define NID_camellia_256_cfb8           765

#define SN_kisa         "KISA"
#define LN_kisa         "kisa"
#define NID_kisa                773
#define OBJ_kisa                OBJ_member_body,410L,200004L

#define SN_seed_ecb             "SEED-ECB"
#define LN_seed_ecb             "seed-ecb"
#define NID_seed_ecb            776
#define OBJ_seed_ecb            OBJ_kisa,1L,3L

#define SN_seed_cbc             "SEED-CBC"
#define LN_seed_cbc             "seed-cbc"
#define NID_seed_cbc            777
#define OBJ_seed_cbc            OBJ_kisa,1L,4L

#define SN_seed_cfb128          "SEED-CFB"
#define LN_seed_cfb128          "seed-cfb"
#define NID_seed_cfb128         779
#define OBJ_seed_cfb128         OBJ_kisa,1L,5L

#define SN_seed_ofb128          "SEED-OFB"
#define LN_seed_ofb128          "seed-ofb"
#define NID_seed_ofb128         778
#define OBJ_seed_ofb128         OBJ_kisa,1L,6L

#define SN_hmac         "HMAC"
#define LN_hmac         "hmac"
#define NID_hmac                855

#define SN_cmac         "CMAC"
#define LN_cmac         "cmac"
#define NID_cmac                894

#define SN_rc4_hmac_md5         "RC4-HMAC-MD5"
#define LN_rc4_hmac_md5         "rc4-hmac-md5"
#define NID_rc4_hmac_md5                915

#define SN_aes_128_cbc_hmac_sha1                "AES-128-CBC-HMAC-SHA1"
```



```
#define LN_aes_128_cbc_hmac_sha1                "aes-128-cbc-hmac-sha1"
#define NID_aes_128_cbc_hmac_sha1               916

#define SN_aes_192_cbc_hmac_sha1                "AES-192-CBC-HMAC-SHA1"
#define LN_aes_192_cbc_hmac_sha1                "aes-192-cbc-hmac-sha1"
#define NID_aes_192_cbc_hmac_sha1               917

#define SN_aes_256_cbc_hmac_sha1                "AES-256-CBC-HMAC-SHA1"
#define LN_aes_256_cbc_hmac_sha1                "aes-256-cbc-hmac-sha1"
#define NID_aes_256_cbc_hmac_sha1               918
#endif /* ! codereview */
```

```
**********************************************************
   36633 Wed Aug 13 19:51:45 2014
new/usr/src/lib/openssl/include/openssl/objects.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/objects/objects.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_OBJECTS_H
#define HEADER_OBJECTS_H
```



```
#define USE_OBJ_MAC

#ifdef USE_OBJ_MAC
#include <openssl/obj_mac.h>
#else
#define SN_undef                        "UNDEF"
#define LN_undef                        "undefined"
#define NID_undef                       0
#define OBJ_undef                       0L

#define SN_Algorithm                    "Algorithm"
#define LN_algorithm                    "algorithm"
#define NID_algorithm                   38
#define OBJ_algorithm                   1L,3L,14L,3L,2L

#define LN_rsadsi                       "rsadsi"
#define NID_rsadsi                      1
#define OBJ_rsadsi                      1L,2L,840L,113549L

#define LN_pkcs                         "pkcs"
#define NID_pkcs                        2
#define OBJ_pkcs                        OBJ_rsadsi,1L

#define SN_md2                          "MD2"
#define LN_md2                          "md2"
#define NID_md2                         3
#define OBJ_md2                         OBJ_rsadsi,2L,2L

#define SN_md5                          "MD5"
#define LN_md5                          "md5"
#define NID_md5                         4
#define OBJ_md5                         OBJ_rsadsi,2L,5L

#define SN_rc4                          "RC4"
#define LN_rc4                          "rc4"
#define NID_rc4                         5
#define OBJ_rc4                         OBJ_rsadsi,3L,4L

#define LN_rsaEncryption                "rsaEncryption"
#define NID_rsaEncryption               6
#define OBJ_rsaEncryption               OBJ_pkcs,1L,1L

#define SN_md2WithRSAEncryption         "RSA-MD2"
#define LN_md2WithRSAEncryption         "md2WithRSAEncryption"
#define NID_md2WithRSAEncryption        7
#define OBJ_md2WithRSAEncryption        OBJ_pkcs,1L,2L

#define SN_md5WithRSAEncryption         "RSA-MD5"
#define LN_md5WithRSAEncryption         "md5WithRSAEncryption"
#define NID_md5WithRSAEncryption        8
#define OBJ_md5WithRSAEncryption        OBJ_pkcs,1L,4L

#define SN_pbeWithMD2AndDES_CBC         "PBE-MD2-DES"
#define LN_pbeWithMD2AndDES_CBC         "pbeWithMD2AndDES-CBC"
#define NID_pbeWithMD2AndDES_CBC        9
#define OBJ_pbeWithMD2AndDES_CBC        OBJ_pkcs,5L,1L

#define SN_pbeWithMD5AndDES_CBC         "PBE-MD5-DES"
#define LN_pbeWithMD5AndDES_CBC         "pbeWithMD5AndDES-CBC"
#define NID_pbeWithMD5AndDES_CBC        10
#define OBJ_pbeWithMD5AndDES_CBC        OBJ_pkcs,5L,3L

#define LN_X500                         "X500"
#define NID_X500                        11
#define OBJ_X500                        2L,5L
```

```
128 #define LN_X509                         "X509"
129 #define NID_X509                        12
130 #define OBJ_X509                        OBJ_X500,4L

132 #define SN_commonName                   "CN"
133 #define LN_commonName                   "commonName"
134 #define NID_commonName                  13
135 #define OBJ_commonName                  OBJ_X509,3L

137 #define SN_countryName                  "C"
138 #define LN_countryName                  "countryName"
139 #define NID_countryName                 14
140 #define OBJ_countryName                 OBJ_X509,6L

142 #define SN_localityName                 "L"
143 #define LN_localityName                 "localityName"
144 #define NID_localityName                15
145 #define OBJ_localityName                OBJ_X509,7L

147 /* Postal Address? PA */

149 /* should be "ST" (rfc1327) but MS uses 'S' */
150 #define SN_stateOrProvinceName          "ST"
151 #define LN_stateOrProvinceName          "stateOrProvinceName"
152 #define NID_stateOrProvinceName         16
153 #define OBJ_stateOrProvinceName         OBJ_X509,8L

155 #define SN_organizationName             "O"
156 #define LN_organizationName             "organizationName"
157 #define NID_organizationName            17
158 #define OBJ_organizationName            OBJ_X509,10L

160 #define SN_organizationalUnitName       "OU"
161 #define LN_organizationalUnitName       "organizationalUnitName"
162 #define NID_organizationalUnitName      18
163 #define OBJ_organizationalUnitName      OBJ_X509,11L

165 #define SN_rsa                          "RSA"
166 #define LN_rsa                          "rsa"
167 #define NID_rsa                         19
168 #define OBJ_rsa                         OBJ_X500,8L,1L,1L

170 #define LN_pkcs7                        "pkcs7"
171 #define NID_pkcs7                       20
172 #define OBJ_pkcs7                       OBJ_pkcs,7L

174 #define LN_pkcs7_data                   "pkcs7-data"
175 #define NID_pkcs7_data                  21
176 #define OBJ_pkcs7_data                  OBJ_pkcs7,1L

178 #define LN_pkcs7_signed                 "pkcs7-signedData"
179 #define NID_pkcs7_signed                22
180 #define OBJ_pkcs7_signed                OBJ_pkcs7,2L

182 #define LN_pkcs7_enveloped              "pkcs7-envelopedData"
183 #define NID_pkcs7_enveloped             23
184 #define OBJ_pkcs7_enveloped             OBJ_pkcs7,3L

186 #define LN_pkcs7_signedAndEnveloped     "pkcs7-signedAndEnvelopedData"
187 #define NID_pkcs7_signedAndEnveloped    24
188 #define OBJ_pkcs7_signedAndEnveloped    OBJ_pkcs7,4L

190 #define LN_pkcs7_digest                 "pkcs7-digestData"
191 #define NID_pkcs7_digest                25
192 #define OBJ_pkcs7_digest                OBJ_pkcs7,5L
```



```
194 #define LN_pkcs7_encrypted              "pkcs7-encryptedData"
195 #define NID_pkcs7_encrypted             26
196 #define OBJ_pkcs7_encrypted             OBJ_pkcs7,6L

198 #define LN_pkcs3                        "pkcs3"
199 #define NID_pkcs3                       27
200 #define OBJ_pkcs3                       OBJ_pkcs,3L

202 #define LN_dhKeyAgreement               "dhKeyAgreement"
203 #define NID_dhKeyAgreement              28
204 #define OBJ_dhKeyAgreement              OBJ_pkcs3,1L

206 #define SN_des_ecb                      "DES-ECB"
207 #define LN_des_ecb                      "des-ecb"
208 #define NID_des_ecb                     29
209 #define OBJ_des_ecb                     OBJ_algorithm,6L

211 #define SN_des_cfb64                    "DES-CFB"
212 #define LN_des_cfb64                    "des-cfb"
213 #define NID_des_cfb64                   30
214 /* IV + num */
215 #define OBJ_des_cfb64                   OBJ_algorithm,9L

217 #define SN_des_cbc                      "DES-CBC"
218 #define LN_des_cbc                      "des-cbc"
219 #define NID_des_cbc                     31
220 /* IV */
221 #define OBJ_des_cbc                     OBJ_algorithm,7L

223 #define SN_des_ede                      "DES-EDE"
224 #define LN_des_ede                      "des-ede"
225 #define NID_des_ede                     32
226 /* ?? */
227 #define OBJ_des_ede                     OBJ_algorithm,17L

229 #define SN_des_ede3                     "DES-EDE3"
230 #define LN_des_ede3                     "des-ede3"
231 #define NID_des_ede3                    33

233 #define SN_idea_cbc                     "IDEA-CBC"
234 #define LN_idea_cbc                     "idea-cbc"
235 #define NID_idea_cbc                    34
236 #define OBJ_idea_cbc                    1L,3L,6L,1L,4L,1L,188L,7L,1L,1L,2L

238 #define SN_idea_cfb64                   "IDEA-CFB"
239 #define LN_idea_cfb64                   "idea-cfb"
240 #define NID_idea_cfb64                  35

242 #define SN_idea_ecb                     "IDEA-ECB"
243 #define LN_idea_ecb                     "idea-ecb"
244 #define NID_idea_ecb                    36

246 #define SN_rc2_cbc                      "RC2-CBC"
247 #define LN_rc2_cbc                      "rc2-cbc"
248 #define NID_rc2_cbc                     37
249 #define OBJ_rc2_cbc                     OBJ_rsadsi,3L,2L

251 #define SN_rc2_ecb                      "RC2-ECB"
252 #define LN_rc2_ecb                      "rc2-ecb"
253 #define NID_rc2_ecb                     38

255 #define SN_rc2_cfb64                    "RC2-CFB"
256 #define LN_rc2_cfb64                    "rc2-cfb"
257 #define NID_rc2_cfb64                   39

259 #define SN_rc2_ofb64                    "RC2-OFB"
```

```
#define LN_rc2_ofb64                    "rc2-ofb"
#define NID_rc2_ofb64                   40

#define SN_sha                          "SHA"
#define LN_sha                          "sha"
#define NID_sha                         41
#define OBJ_sha                         OBJ_algorithm,18L

#define SN_shaWithRSAEncryption         "RSA-SHA"
#define LN_shaWithRSAEncryption         "shaWithRSAEncryption"
#define NID_shaWithRSAEncryption        42
#define OBJ_shaWithRSAEncryption        OBJ_algorithm,15L

#define SN_des_ede_cbc                  "DES-EDE-CBC"
#define LN_des_ede_cbc                  "des-ede-cbc"
#define NID_des_ede_cbc                 43

#define SN_des_ede3_cbc                 "DES-EDE3-CBC"
#define LN_des_ede3_cbc                 "des-ede3-cbc"
#define NID_des_ede3_cbc                44
#define OBJ_des_ede3_cbc                OBJ_rsadsi,3L,7L

#define SN_des_ofb64                    "DES-OFB"
#define LN_des_ofb64                    "des-ofb"
#define NID_des_ofb64                   45
#define OBJ_des_ofb64                   OBJ_algorithm,8L

#define SN_idea_ofb64                   "IDEA-OFB"
#define LN_idea_ofb64                   "idea-ofb"
#define NID_idea_ofb64                  46

#define LN_pkcs9                        "pkcs9"
#define NID_pkcs9                       47
#define OBJ_pkcs9                       OBJ_pkcs,9L

#define SN_pkcs9_emailAddress           "Email"
#define LN_pkcs9_emailAddress           "emailAddress"
#define NID_pkcs9_emailAddress          48
#define OBJ_pkcs9_emailAddress          OBJ_pkcs9,1L

#define LN_pkcs9_unstructuredName       "unstructuredName"
#define NID_pkcs9_unstructuredName      49
#define OBJ_pkcs9_unstructuredName      OBJ_pkcs9,2L

#define LN_pkcs9_contentType            "contentType"
#define NID_pkcs9_contentType           50
#define OBJ_pkcs9_contentType           OBJ_pkcs9,3L

#define LN_pkcs9_messageDigest          "messageDigest"
#define NID_pkcs9_messageDigest         51
#define OBJ_pkcs9_messageDigest         OBJ_pkcs9,4L

#define LN_pkcs9_signingTime            "signingTime"
#define NID_pkcs9_signingTime           52
#define OBJ_pkcs9_signingTime           OBJ_pkcs9,5L

#define LN_pkcs9_countersignature       "countersignature"
#define NID_pkcs9_countersignature      53
#define OBJ_pkcs9_countersignature      OBJ_pkcs9,6L

#define LN_pkcs9_challengePassword      "challengePassword"
#define NID_pkcs9_challengePassword     54
#define OBJ_pkcs9_challengePassword     OBJ_pkcs9,7L

#define LN_pkcs9_unstructuredAddress    "unstructuredAddress"
#define NID_pkcs9_unstructuredAddress   55
```



```
#define OBJ_pkcs9_unstructuredAddress   OBJ_pkcs9,8L

#define LN_pkcs9_extCertAttributes      "extendedCertificateAttributes"
#define NID_pkcs9_extCertAttributes     56
#define OBJ_pkcs9_extCertAttributes     OBJ_pkcs9,9L

#define SN_netscape                     "Netscape"
#define LN_netscape                     "Netscape Communications Corp."
#define NID_netscape                    57
#define OBJ_netscape                    2L,16L,840L,1L,113730L

#define SN_netscape_cert_extension      "nsCertExt"
#define LN_netscape_cert_extension      "Netscape Certificate Extension"
#define NID_netscape_cert_extension     58
#define OBJ_netscape_cert_extension     OBJ_netscape,1L

#define SN_netscape_data_type           "nsDataType"
#define LN_netscape_data_type           "Netscape Data Type"
#define NID_netscape_data_type          59
#define OBJ_netscape_data_type          OBJ_netscape,2L

#define SN_des_ede_cfb64                "DES-EDE-CFB"
#define LN_des_ede_cfb64                "des-ede-cfb"
#define NID_des_ede_cfb64               60

#define SN_des_ede3_cfb64               "DES-EDE3-CFB"
#define LN_des_ede3_cfb64               "des-ede3-cfb"
#define NID_des_ede3_cfb64              61

#define SN_des_ede_ofb64                "DES-EDE-OFB"
#define LN_des_ede_ofb64                "des-ede-ofb"
#define NID_des_ede_ofb64               62

#define SN_des_ede3_ofb64               "DES-EDE3-OFB"
#define LN_des_ede3_ofb64               "des-ede3-ofb"
#define NID_des_ede3_ofb64              63

/* I'm not sure about the object ID */
#define SN_sha1                         "SHA1"
#define LN_sha1                         "sha1"
#define NID_sha1                        64
#define OBJ_sha1                        OBJ_algorithm,26L
/* 28 Jun 1996 - eay */
/* #define OBJ_sha1                     1L,3L,14L,2L,26L,05L <- wrong */

#define SN_sha1WithRSAEncryption        "RSA-SHA1"
#define LN_sha1WithRSAEncryption        "sha1WithRSAEncryption"
#define NID_sha1WithRSAEncryption       65
#define OBJ_sha1WithRSAEncryption       OBJ_pkcs,1L,5L

#define SN_dsaWithSHA                   "DSA-SHA"
#define LN_dsaWithSHA                   "dsaWithSHA"
#define NID_dsaWithSHA                  66
#define OBJ_dsaWithSHA                  OBJ_algorithm,13L

#define SN_dsa_2                        "DSA-old"
#define LN_dsa_2                        "dsaEncryption-old"
#define NID_dsa_2                       67
#define OBJ_dsa_2                       OBJ_algorithm,12L

/* proposed by microsoft to RSA */
#define SN_pbeWithSHA1AndRC2_CBC        "PBE-SHA1-RC2-64"
#define LN_pbeWithSHA1AndRC2_CBC        "pbeWithSHA1AndRC2-CBC"
#define NID_pbeWithSHA1AndRC2_CBC       68
#define OBJ_pbeWithSHA1AndRC2_CBC       OBJ_pkcs,5L,11L
```

```
392 /* proposed by microsoft to RSA as pbeWithSHA1AndRC4: it is now
393  * defined explicitly in PKCS#5 v2.0 as id-PBKDF2 which is something
394  * completely different.
395  */
396 #define LN_id_pbkdf2                    "PBKDF2"
397 #define NID_id_pbkdf2                   69
398 #define OBJ_id_pbkdf2                   OBJ_pkcs,5L,12L

400 #define SN_dsaWithSHA1_2                "DSA-SHA1-old"
401 #define LN_dsaWithSHA1_2                "dsaWithSHA1-old"
402 #define NID_dsaWithSHA1_2               70
403 /* Got this one from 'sdn706r20.pdf' which is actually an NSA document :-) */
404 #define OBJ_dsaWithSHA1_2               OBJ_algorithm,27L

406 #define SN_netscape_cert_type           "nsCertType"
407 #define LN_netscape_cert_type           "Netscape Cert Type"
408 #define NID_netscape_cert_type          71
409 #define OBJ_netscape_cert_type          OBJ_netscape_cert_extension,1L

411 #define SN_netscape_base_url            "nsBaseUrl"
412 #define LN_netscape_base_url            "Netscape Base Url"
413 #define NID_netscape_base_url           72
414 #define OBJ_netscape_base_url           OBJ_netscape_cert_extension,2L

416 #define SN_netscape_revocation_url      "nsRevocationUrl"
417 #define LN_netscape_revocation_url      "Netscape Revocation Url"
418 #define NID_netscape_revocation_url     73
419 #define OBJ_netscape_revocation_url     OBJ_netscape_cert_extension,3L

421 #define SN_netscape_ca_revocation_url   "nsCaRevocationUrl"
422 #define LN_netscape_ca_revocation_url   "Netscape CA Revocation Url"
423 #define NID_netscape_ca_revocation_url  74
424 #define OBJ_netscape_ca_revocation_url  OBJ_netscape_cert_extension,4L

426 #define SN_netscape_renewal_url         "nsRenewalUrl"
427 #define LN_netscape_renewal_url         "Netscape Renewal Url"
428 #define NID_netscape_renewal_url        75
429 #define OBJ_netscape_renewal_url        OBJ_netscape_cert_extension,7L

431 #define SN_netscape_ca_policy_url       "nsCaPolicyUrl"
432 #define LN_netscape_ca_policy_url       "Netscape CA Policy Url"
433 #define NID_netscape_ca_policy_url      76
434 #define OBJ_netscape_ca_policy_url      OBJ_netscape_cert_extension,8L

436 #define SN_netscape_ssl_server_name     "nsSslServerName"
437 #define LN_netscape_ssl_server_name     "Netscape SSL Server Name"
438 #define NID_netscape_ssl_server_name    77
439 #define OBJ_netscape_ssl_server_name    OBJ_netscape_cert_extension,12L

441 #define SN_netscape_comment             "nsComment"
442 #define LN_netscape_comment             "Netscape Comment"
443 #define NID_netscape_comment            78
444 #define OBJ_netscape_comment            OBJ_netscape_cert_extension,13L

446 #define SN_netscape_cert_sequence       "nsCertSequence"
447 #define LN_netscape_cert_sequence       "Netscape Certificate Sequence"
448 #define NID_netscape_cert_sequence      79
449 #define OBJ_netscape_cert_sequence      OBJ_netscape_data_type,5L

451 #define SN_desx_cbc                     "DESX-CBC"
452 #define LN_desx_cbc                     "desx-cbc"
453 #define NID_desx_cbc                    80

455 #define SN_id_ce                        "id-ce"
456 #define NID_id_ce                       81
457 #define OBJ_id_ce                       2L,5L,29L
```



```
459 #define SN_subject_key_identifier       "subjectKeyIdentifier"
460 #define LN_subject_key_identifier       "X509v3 Subject Key Identifier"
461 #define NID_subject_key_identifier      82
462 #define OBJ_subject_key_identifier      OBJ_id_ce,14L

464 #define SN_key_usage                    "keyUsage"
465 #define LN_key_usage                    "X509v3 Key Usage"
466 #define NID_key_usage                   83
467 #define OBJ_key_usage                   OBJ_id_ce,15L

469 #define SN_private_key_usage_period     "privateKeyUsagePeriod"
470 #define LN_private_key_usage_period     "X509v3 Private Key Usage Period"
471 #define NID_private_key_usage_period    84
472 #define OBJ_private_key_usage_period    OBJ_id_ce,16L

474 #define SN_subject_alt_name             "subjectAltName"
475 #define LN_subject_alt_name             "X509v3 Subject Alternative Name"
476 #define NID_subject_alt_name            85
477 #define OBJ_subject_alt_name            OBJ_id_ce,17L

479 #define SN_issuer_alt_name              "issuerAltName"
480 #define LN_issuer_alt_name              "X509v3 Issuer Alternative Name"
481 #define NID_issuer_alt_name             86
482 #define OBJ_issuer_alt_name             OBJ_id_ce,18L

484 #define SN_basic_constraints            "basicConstraints"
485 #define LN_basic_constraints            "X509v3 Basic Constraints"
486 #define NID_basic_constraints           87
487 #define OBJ_basic_constraints           OBJ_id_ce,19L

489 #define SN_crl_number                   "crlNumber"
490 #define LN_crl_number                   "X509v3 CRL Number"
491 #define NID_crl_number                  88
492 #define OBJ_crl_number                  OBJ_id_ce,20L

494 #define SN_certificate_policies         "certificatePolicies"
495 #define LN_certificate_policies         "X509v3 Certificate Policies"
496 #define NID_certificate_policies        89
497 #define OBJ_certificate_policies        OBJ_id_ce,32L

499 #define SN_authority_key_identifier     "authorityKeyIdentifier"
500 #define LN_authority_key_identifier     "X509v3 Authority Key Identifier"
501 #define NID_authority_key_identifier    90
502 #define OBJ_authority_key_identifier    OBJ_id_ce,35L

504 #define SN_bf_cbc                       "BF-CBC"
505 #define LN_bf_cbc                       "bf-cbc"
506 #define NID_bf_cbc                      91
507 #define OBJ_bf_cbc                      1L,3L,6L,1L,4L,1L,3029L,1L,2L

509 #define SN_bf_ecb                       "BF-ECB"
510 #define LN_bf_ecb                       "bf-ecb"
511 #define NID_bf_ecb                      92

513 #define SN_bf_cfb64                     "BF-CFB"
514 #define LN_bf_cfb64                     "bf-cfb"
515 #define NID_bf_cfb64                    93

517 #define SN_bf_ofb64                     "BF-OFB"
518 #define LN_bf_ofb64                     "bf-ofb"
519 #define NID_bf_ofb64                    94

521 #define SN_mdc2                         "MDC2"
522 #define LN_mdc2                         "mdc2"
523 #define NID_mdc2                        95
```

```
524 #define OBJ_mdc2                        2L,5L,8L,3L,101L
525 /* An alternative?                      1L,3L,14L,3L,2L,19L */

527 #define SN_mdc2WithRSA                  "RSA-MDC2"
528 #define LN_mdc2WithRSA                  "mdc2withRSA"
529 #define NID_mdc2WithRSA                 96
530 #define OBJ_mdc2WithRSA                 2L,5L,8L,3L,100L

532 #define SN_rc4_40                       "RC4-40"
533 #define LN_rc4_40                       "rc4-40"
534 #define NID_rc4_40                      97

536 #define SN_rc2_40_cbc                   "RC2-40-CBC"
537 #define LN_rc2_40_cbc                   "rc2-40-cbc"
538 #define NID_rc2_40_cbc                  98

540 #define SN_givenName                    "G"
541 #define LN_givenName                    "givenName"
542 #define NID_givenName                   99
543 #define OBJ_givenName                   OBJ_X509,42L

545 #define SN_surname                      "S"
546 #define LN_surname                      "surname"
547 #define NID_surname                     100
548 #define OBJ_surname                     OBJ_X509,4L

550 #define SN_initials                     "I"
551 #define LN_initials                     "initials"
552 #define NID_initials                    101
553 #define OBJ_initials                    OBJ_X509,43L

555 #define SN_uniqueIdentifier             "UID"
556 #define LN_uniqueIdentifier             "uniqueIdentifier"
557 #define NID_uniqueIdentifier            102
558 #define OBJ_uniqueIdentifier            OBJ_X509,45L

560 #define SN_crl_distribution_points      "crlDistributionPoints"
561 #define LN_crl_distribution_points      "X509v3 CRL Distribution Points"
562 #define NID_crl_distribution_points     103
563 #define OBJ_crl_distribution_points     OBJ_id_ce,31L

565 #define SN_md5WithRSA                   "RSA-NP-MD5"
566 #define LN_md5WithRSA                   "md5WithRSA"
567 #define NID_md5WithRSA                  104
568 #define OBJ_md5WithRSA                  OBJ_algorithm,3L

570 #define SN_serialNumber                 "SN"
571 #define LN_serialNumber                 "serialNumber"
572 #define NID_serialNumber                105
573 #define OBJ_serialNumber                OBJ_X509,5L

575 #define SN_title                        "T"
576 #define LN_title                        "title"
577 #define NID_title                       106
578 #define OBJ_title                       OBJ_X509,12L

580 #define SN_description                  "D"
581 #define LN_description                  "description"
582 #define NID_description                 107
583 #define OBJ_description                 OBJ_X509,13L

585 /* CAST5 is CAST-128, I'm just sticking with the documentation */
586 #define SN_cast5_cbc                    "CAST5-CBC"
587 #define LN_cast5_cbc                    "cast5-cbc"
588 #define NID_cast5_cbc                   108
589 #define OBJ_cast5_cbc                   1L,2L,840L,113533L,7L,66L,10L
```



```
591 #define SN_cast5_ecb                    "CAST5-ECB"
592 #define LN_cast5_ecb                    "cast5-ecb"
593 #define NID_cast5_ecb                   109

595 #define SN_cast5_cfb64                  "CAST5-CFB"
596 #define LN_cast5_cfb64                  "cast5-cfb"
597 #define NID_cast5_cfb64                 110

599 #define SN_cast5_ofb64                  "CAST5-OFB"
600 #define LN_cast5_ofb64                  "cast5-ofb"
601 #define NID_cast5_ofb64                 111

603 #define LN_pbeWithMD5AndCast5_CBC       "pbeWithMD5AndCast5CBC"
604 #define NID_pbeWithMD5AndCast5_CBC      112
605 #define OBJ_pbeWithMD5AndCast5_CBC      1L,2L,840L,113533L,7L,66L,12L

607 /* This is one sun will soon be using :-(
608  * id-dsa-with-sha1 ID  ::= {
609  *   iso(1) member-body(2) us(840) x9-57 (10040) x9cm(4) 3 }
610  */
611 #define SN_dsaWithSHA1                  "DSA-SHA1"
612 #define LN_dsaWithSHA1                  "dsaWithSHA1"
613 #define NID_dsaWithSHA1                 113
614 #define OBJ_dsaWithSHA1                 1L,2L,840L,10040L,4L,3L

616 #define NID_md5_sha1                    114
617 #define SN_md5_sha1                     "MD5-SHA1"
618 #define LN_md5_sha1                     "md5-sha1"

620 #define SN_sha1WithRSA                  "RSA-SHA1-2"
621 #define LN_sha1WithRSA                  "sha1WithRSA"
622 #define NID_sha1WithRSA                 115
623 #define OBJ_sha1WithRSA                 OBJ_algorithm,29L

625 #define SN_dsa                          "DSA"
626 #define LN_dsa                          "dsaEncryption"
627 #define NID_dsa                         116
628 #define OBJ_dsa                         1L,2L,840L,10040L,4L,1L

630 #define SN_ripemd160                    "RIPEMD160"
631 #define LN_ripemd160                    "ripemd160"
632 #define NID_ripemd160                   117
633 #define OBJ_ripemd160                   1L,3L,36L,3L,2L,1L

635 /* The name should actually be rsaSignatureWithripemd160, but I'm going
636  * to continue using the convention I'm using with the other ciphers */
637 #define SN_ripemd160WithRSA             "RSA-RIPEMD160"
638 #define LN_ripemd160WithRSA             "ripemd160WithRSA"
639 #define NID_ripemd160WithRSA            119
640 #define OBJ_ripemd160WithRSA            1L,3L,36L,3L,3L,1L,2L

642 /* Taken from rfc2040
643  *  RC5_CBC_Parameters ::= SEQUENCE {
644  *      version           INTEGER (v1_0(16)),
645  *      rounds            INTEGER (8..127),
646  *      blockSizeInBits   INTEGER (64, 128),
647  *      iv                OCTET STRING OPTIONAL
648  *      }
649  */
650 #define SN_rc5_cbc                      "RC5-CBC"
651 #define LN_rc5_cbc                      "rc5-cbc"
652 #define NID_rc5_cbc                     120
653 #define OBJ_rc5_cbc                     OBJ_rsadsi,3L,8L

655 #define SN_rc5_ecb                      "RC5-ECB"
```

```
#define LN_rc5_ecb                      "rc5-ecb"
#define NID_rc5_ecb                     121

#define SN_rc5_cfb64                    "RC5-CFB"
#define LN_rc5_cfb64                    "rc5-cfb"
#define NID_rc5_cfb64                   122

#define SN_rc5_ofb64                    "RC5-OFB"
#define LN_rc5_ofb64                    "rc5-ofb"
#define NID_rc5_ofb64                   123

#define SN_rle_compression              "RLE"
#define LN_rle_compression              "run length compression"
#define NID_rle_compression             124
#define OBJ_rle_compression             1L,1L,1L,1L,666L,1L

#define SN_zlib_compression             "ZLIB"
#define LN_zlib_compression             "zlib compression"
#define NID_zlib_compression            125
#define OBJ_zlib_compression            1L,1L,1L,1L,666L,2L

#define SN_ext_key_usage                "extendedKeyUsage"
#define LN_ext_key_usage                "X509v3 Extended Key Usage"
#define NID_ext_key_usage               126
#define OBJ_ext_key_usage               OBJ_id_ce,37

#define SN_id_pkix                      "PKIX"
#define NID_id_pkix                     127
#define OBJ_id_pkix                     1L,3L,6L,1L,5L,5L,7L

#define SN_id_kp                        "id-kp"
#define NID_id_kp                       128
#define OBJ_id_kp                       OBJ_id_pkix,3L

/* PKIX extended key usage OIDs */

#define SN_server_auth                  "serverAuth"
#define LN_server_auth                  "TLS Web Server Authentication"
#define NID_server_auth                 129
#define OBJ_server_auth                 OBJ_id_kp,1L

#define SN_client_auth                  "clientAuth"
#define LN_client_auth                  "TLS Web Client Authentication"
#define NID_client_auth                 130
#define OBJ_client_auth                 OBJ_id_kp,2L

#define SN_code_sign                    "codeSigning"
#define LN_code_sign                    "Code Signing"
#define NID_code_sign                   131
#define OBJ_code_sign                   OBJ_id_kp,3L

#define SN_email_protect                "emailProtection"
#define LN_email_protect                "E-mail Protection"
#define NID_email_protect               132
#define OBJ_email_protect               OBJ_id_kp,4L

#define SN_time_stamp                   "timeStamping"
#define LN_time_stamp                   "Time Stamping"
#define NID_time_stamp                  133
#define OBJ_time_stamp                  OBJ_id_kp,8L

/* Additional extended key usage OIDs: Microsoft */

#define SN_ms_code_ind                  "msCodeInd"
#define LN_ms_code_ind                  "Microsoft Individual Code Signing"
#define NID_ms_code_ind                 134
```



```
#define OBJ_ms_code_ind                 1L,3L,6L,1L,4L,1L,311L,2L,1L,21L

#define SN_ms_code_com                  "msCodeCom"
#define LN_ms_code_com                  "Microsoft Commercial Code Signing"
#define NID_ms_code_com                 135
#define OBJ_ms_code_com                 1L,3L,6L,1L,4L,1L,311L,2L,1L,22L

#define SN_ms_ctl_sign                  "msCTLSign"
#define LN_ms_ctl_sign                  "Microsoft Trust List Signing"
#define NID_ms_ctl_sign                 136
#define OBJ_ms_ctl_sign                 1L,3L,6L,1L,4L,1L,311L,10L,3L,1L

#define SN_ms_sgc                       "msSGC"
#define LN_ms_sgc                       "Microsoft Server Gated Crypto"
#define NID_ms_sgc                      137
#define OBJ_ms_sgc                      1L,3L,6L,1L,4L,1L,311L,10L,3L,3L

#define SN_ms_efs                       "msEFS"
#define LN_ms_efs                       "Microsoft Encrypted File System"
#define NID_ms_efs                      138
#define OBJ_ms_efs                      1L,3L,6L,1L,4L,1L,311L,10L,3L,4L

/* Additional usage: Netscape */

#define SN_ns_sgc                       "nsSGC"
#define LN_ns_sgc                       "Netscape Server Gated Crypto"
#define NID_ns_sgc                      139
#define OBJ_ns_sgc                      OBJ_netscape,4L,1L

#define SN_delta_crl                    "deltaCRL"
#define LN_delta_crl                    "X509v3 Delta CRL Indicator"
#define NID_delta_crl                   140
#define OBJ_delta_crl                   OBJ_id_ce,27L

#define SN_crl_reason                   "CRLReason"
#define LN_crl_reason                   "CRL Reason Code"
#define NID_crl_reason                  141
#define OBJ_crl_reason                  OBJ_id_ce,21L

#define SN_invalidity_date              "invalidityDate"
#define LN_invalidity_date              "Invalidity Date"
#define NID_invalidity_date             142
#define OBJ_invalidity_date             OBJ_id_ce,24L

#define SN_sxnet                        "SXNetID"
#define LN_sxnet                        "Strong Extranet ID"
#define NID_sxnet                       143
#define OBJ_sxnet                       1L,3L,101L,1L,4L,1L

/* PKCS12 and related OBJECT IDENTIFIERS */

#define OBJ_pkcs12                      OBJ_pkcs,12L
#define OBJ_pkcs12_pbeids               OBJ_pkcs12, 1

#define SN_pbe_WithSHA1And128BitRC4     "PBE-SHA1-RC4-128"
#define LN_pbe_WithSHA1And128BitRC4     "pbeWithSHA1And128BitRC4"
#define NID_pbe_WithSHA1And128BitRC4    144
#define OBJ_pbe_WithSHA1And128BitRC4    OBJ_pkcs12_pbeids, 1L

#define SN_pbe_WithSHA1And40BitRC4      "PBE-SHA1-RC4-40"
#define LN_pbe_WithSHA1And40BitRC4      "pbeWithSHA1And40BitRC4"
#define NID_pbe_WithSHA1And40BitRC4     145
#define OBJ_pbe_WithSHA1And40BitRC4     OBJ_pkcs12_pbeids, 2L

#define SN_pbe_WithSHA1And3_Key_TripleDES_CBC   "PBE-SHA1-3DES"
#define LN_pbe_WithSHA1And3_Key_TripleDES_CBC   "pbeWithSHA1And3-KeyTripleDES-CB
```

```
#define NID_pbe_WithSHA1And3_Key_TripleDES_CBC  146
#define OBJ_pbe_WithSHA1And3_Key_TripleDES_CBC  OBJ_pkcs12_pbeids, 3L

#define SN_pbe_WithSHA1And2_Key_TripleDES_CBC   "PBE-SHA1-2DES"
#define LN_pbe_WithSHA1And2_Key_TripleDES_CBC   "pbeWithSHA1And2-KeyTripleDES-CB
#define NID_pbe_WithSHA1And2_Key_TripleDES_CBC  147
#define OBJ_pbe_WithSHA1And2_Key_TripleDES_CBC  OBJ_pkcs12_pbeids, 4L

#define SN_pbe_WithSHA1And128BitRC2_CBC         "PBE-SHA1-RC2-128"
#define LN_pbe_WithSHA1And128BitRC2_CBC         "pbeWithSHA1And128BitRC2-CBC"
#define NID_pbe_WithSHA1And128BitRC2_CBC        148
#define OBJ_pbe_WithSHA1And128BitRC2_CBC        OBJ_pkcs12_pbeids, 5L

#define SN_pbe_WithSHA1And40BitRC2_CBC  "PBE-SHA1-RC2-40"
#define LN_pbe_WithSHA1And40BitRC2_CBC  "pbeWithSHA1And40BitRC2-CBC"
#define NID_pbe_WithSHA1And40BitRC2_CBC 149
#define OBJ_pbe_WithSHA1And40BitRC2_CBC OBJ_pkcs12_pbeids, 6L

#define OBJ_pkcs12_Version1     OBJ_pkcs12, 10L

#define OBJ_pkcs12_BagIds       OBJ_pkcs12_Version1, 1L

#define LN_keyBag               "keyBag"
#define NID_keyBag              150
#define OBJ_keyBag              OBJ_pkcs12_BagIds, 1L

#define LN_pkcs8ShroudedKeyBag  "pkcs8ShroudedKeyBag"
#define NID_pkcs8ShroudedKeyBag 151
#define OBJ_pkcs8ShroudedKeyBag OBJ_pkcs12_BagIds, 2L

#define LN_certBag              "certBag"
#define NID_certBag             152
#define OBJ_certBag             OBJ_pkcs12_BagIds, 3L

#define LN_crlBag               "crlBag"
#define NID_crlBag              153
#define OBJ_crlBag              OBJ_pkcs12_BagIds, 4L

#define LN_secretBag            "secretBag"
#define NID_secretBag           154
#define OBJ_secretBag           OBJ_pkcs12_BagIds, 5L

#define LN_safeContentsBag      "safeContentsBag"
#define NID_safeContentsBag     155
#define OBJ_safeContentsBag     OBJ_pkcs12_BagIds, 6L

#define LN_friendlyName         "friendlyName"
#define NID_friendlyName        156
#define OBJ_friendlyName        OBJ_pkcs9, 20L

#define LN_localKeyID           "localKeyID"
#define NID_localKeyID          157
#define OBJ_localKeyID          OBJ_pkcs9, 21L

#define OBJ_certTypes           OBJ_pkcs9, 22L

#define LN_x509Certificate      "x509Certificate"
#define NID_x509Certificate     158
#define OBJ_x509Certificate     OBJ_certTypes, 1L

#define LN_sdsiCertificate      "sdsiCertificate"
#define NID_sdsiCertificate     159
#define OBJ_sdsiCertificate     OBJ_certTypes, 2L

#define OBJ_crlTypes            OBJ_pkcs9, 23L
```



```
#define LN_x509Crl              "x509Crl"
#define NID_x509Crl             160
#define OBJ_x509Crl             OBJ_crlTypes, 1L

/* PKCS#5 v2 OIDs */

#define LN_pbes2                "PBES2"
#define NID_pbes2               161
#define OBJ_pbes2               OBJ_pkcs,5L,13L

#define LN_pbmac1               "PBMAC1"
#define NID_pbmac1              162
#define OBJ_pbmac1              OBJ_pkcs,5L,14L

#define LN_hmacWithSHA1         "hmacWithSHA1"
#define NID_hmacWithSHA1        163
#define OBJ_hmacWithSHA1        OBJ_rsadsi,2L,7L

/* Policy Qualifier Ids */

#define LN_id_qt_cps            "Policy Qualifier CPS"
#define SN_id_qt_cps            "id-qt-cps"
#define NID_id_qt_cps           164
#define OBJ_id_qt_cps           OBJ_id_pkix,2L,1L

#define LN_id_qt_unotice        "Policy Qualifier User Notice"
#define SN_id_qt_unotice        "id-qt-unotice"
#define NID_id_qt_unotice       165
#define OBJ_id_qt_unotice       OBJ_id_pkix,2L,2L

#define SN_rc2_64_cbc                   "RC2-64-CBC"
#define LN_rc2_64_cbc                   "rc2-64-cbc"
#define NID_rc2_64_cbc                  166

#define SN_SMIMECapabilities            "SMIME-CAPS"
#define LN_SMIMECapabilities            "S/MIME Capabilities"
#define NID_SMIMECapabilities           167
#define OBJ_SMIMECapabilities           OBJ_pkcs9,15L

#define SN_pbeWithMD2AndRC2_CBC         "PBE-MD2-RC2-64"
#define LN_pbeWithMD2AndRC2_CBC         "pbeWithMD2AndRC2-CBC"
#define NID_pbeWithMD2AndRC2_CBC        168
#define OBJ_pbeWithMD2AndRC2_CBC        OBJ_pkcs,5L,4L

#define SN_pbeWithMD5AndRC2_CBC         "PBE-MD5-RC2-64"
#define LN_pbeWithMD5AndRC2_CBC         "pbeWithMD5AndRC2-CBC"
#define NID_pbeWithMD5AndRC2_CBC        169
#define OBJ_pbeWithMD5AndRC2_CBC        OBJ_pkcs,5L,6L

#define SN_pbeWithSHA1AndDES_CBC        "PBE-SHA1-DES"
#define LN_pbeWithSHA1AndDES_CBC        "pbeWithSHA1AndDES-CBC"
#define NID_pbeWithSHA1AndDES_CBC       170
#define OBJ_pbeWithSHA1AndDES_CBC       OBJ_pkcs,5L,10L

/* Extension request OIDs */

#define LN_ms_ext_req                   "Microsoft Extension Request"
#define SN_ms_ext_req                   "msExtReq"
#define NID_ms_ext_req                  171
#define OBJ_ms_ext_req                  1L,3L,6L,1L,4L,1L,311L,2L,1L,14L

#define LN_ext_req                      "Extension Request"
#define SN_ext_req                      "extReq"
#define NID_ext_req                     172
#define OBJ_ext_req                     OBJ_pkcs9,14L
```

```
#define SN_name                         "name"
#define LN_name                         "name"
#define NID_name                        173
#define OBJ_name                        OBJ_X509,41L

#define SN_dnQualifier                  "dnQualifier"
#define LN_dnQualifier                  "dnQualifier"
#define NID_dnQualifier                 174
#define OBJ_dnQualifier                 OBJ_X509,46L

#define SN_id_pe                        "id-pe"
#define NID_id_pe                       175
#define OBJ_id_pe                       OBJ_id_pkix,1L

#define SN_id_ad                        "id-ad"
#define NID_id_ad                       176
#define OBJ_id_ad                       OBJ_id_pkix,48L

#define SN_info_access                  "authorityInfoAccess"
#define LN_info_access                  "Authority Information Access"
#define NID_info_access                 177
#define OBJ_info_access                 OBJ_id_pe,1L

#define SN_ad_OCSP                      "OCSP"
#define LN_ad_OCSP                      "OCSP"
#define NID_ad_OCSP                     178
#define OBJ_ad_OCSP                     OBJ_id_ad,1L

#define SN_ad_ca_issuers                "caIssuers"
#define LN_ad_ca_issuers                "CA Issuers"
#define NID_ad_ca_issuers               179
#define OBJ_ad_ca_issuers               OBJ_id_ad,2L

#define SN_OCSP_sign                    "OCSPSigning"
#define LN_OCSP_sign                    "OCSP Signing"
#define NID_OCSP_sign                   180
#define OBJ_OCSP_sign                   OBJ_id_kp,9L
#endif /* USE_OBJ_MAC */

#include <openssl/bio.h>
#include <openssl/asn1.h>

#define OBJ_NAME_TYPE_UNDEF             0x00
#define OBJ_NAME_TYPE_MD_METH           0x01
#define OBJ_NAME_TYPE_CIPHER_METH       0x02
#define OBJ_NAME_TYPE_PKEY_METH         0x03
#define OBJ_NAME_TYPE_COMP_METH         0x04
#define OBJ_NAME_TYPE_NUM               0x05

#define OBJ_NAME_ALIAS                  0x8000

#define OBJ_BSEARCH_VALUE_ON_NOMATCH            0x01
#define OBJ_BSEARCH_FIRST_VALUE_ON_MATCH        0x02


#ifdef  __cplusplus
extern "C" {
#endif

typedef struct obj_name_st
        {
        int type;
        int alias;
        const char *name;
        const char *data;
        } OBJ_NAME;
```



```
#define         OBJ_create_and_add_object(a,b,c) OBJ_create(a,b,c)


int OBJ_NAME_init(void);
int OBJ_NAME_new_index(unsigned long (*hash_func)(const char *),
                       int (*cmp_func)(const char *, const char *),
                       void (*free_func)(const char *, int, const char *));
const char *OBJ_NAME_get(const char *name,int type);
int OBJ_NAME_add(const char *name,int type,const char *data);
int OBJ_NAME_remove(const char *name,int type);
void OBJ_NAME_cleanup(int type); /* -1 for everything */
void OBJ_NAME_do_all(int type,void (*fn)(const OBJ_NAME *,void *arg),
                     void *arg);
void OBJ_NAME_do_all_sorted(int type,void (*fn)(const OBJ_NAME *,void *arg),
                            void *arg);

ASN1_OBJECT *   OBJ_dup(const ASN1_OBJECT *o);
ASN1_OBJECT *   OBJ_nid2obj(int n);
const char *    OBJ_nid2ln(int n);
const char *    OBJ_nid2sn(int n);
int             OBJ_obj2nid(const ASN1_OBJECT *o);
ASN1_OBJECT *   OBJ_txt2obj(const char *s, int no_name);
int     OBJ_obj2txt(char *buf, int buf_len, const ASN1_OBJECT *a, int no_name);
int             OBJ_txt2nid(const char *s);
int             OBJ_ln2nid(const char *s);
int             OBJ_sn2nid(const char *s);
int             OBJ_cmp(const ASN1_OBJECT *a,const ASN1_OBJECT *b);
const void *    OBJ_bsearch_(const void *key,const void *base,int num,int size,
                             int (*cmp)(const void *, const void *));
const void *    OBJ_bsearch_ex_(const void *key,const void *base,int num,
                                int size,
                                int (*cmp)(const void *, const void *),
                                int flags);

#define _DECLARE_OBJ_BSEARCH_CMP_FN(scope, type1, type2, nm)    \
  static int nm##_cmp_BSEARCH_CMP_FN(const void *, const void *); \
  static int nm##_cmp(type1 const *, type2 const *); \
  scope type2 * OBJ_bsearch_##nm(type1 *key, type2 const *base, int num)

#define DECLARE_OBJ_BSEARCH_CMP_FN(type1, type2, cmp)   \
  _DECLARE_OBJ_BSEARCH_CMP_FN(static, type1, type2, cmp)
#define DECLARE_OBJ_BSEARCH_GLOBAL_CMP_FN(type1, type2, nm)     \
  type2 * OBJ_bsearch_##nm(type1 *key, type2 const *base, int num)

/*
 * Unsolved problem: if a type is actually a pointer type, like
 * nid_triple is, then its impossible to get a const where you need
 * it. Consider:
 *
 * typedef int nid_triple[3];
 * const void *a_;
 * const nid_triple const *a = a_;
 *
 * The assignement discards a const because what you really want is:
 *
 * const int const * const *a = a_;
 *
 * But if you do that, you lose the fact that a is an array of 3 ints,
 * which breaks comparison functions.
 *
 * Thus we end up having to cast, sadly, or unpack the
 * declarations. Or, as I finally did in this case, delcare nid_triple
 * to be a struct, which it should have been in the first place.
 *
 * Ben, August 2008.
```

```
 *
 * Also, strictly speaking not all types need be const, but handling
 * the non-constness means a lot of complication, and in practice
 * comparison routines do always not touch their arguments.
 */

#define IMPLEMENT_OBJ_BSEARCH_CMP_FN(type1, type2, nm)  \
  static int nm##_cmp_BSEARCH_CMP_FN(const void *a_, const void *b_)    \
      { \
      type1 const *a = a_; \
      type2 const *b = b_; \
      return nm##_cmp(a,b); \
      } \
  static type2 *OBJ_bsearch_##nm(type1 *key, type2 const *base, int num) \
      { \
      return (type2 *)OBJ_bsearch_(key, base, num, sizeof(type2), \
                                   nm##_cmp_BSEARCH_CMP_FN); \
      } \
      extern void dummy_prototype(void)

#define IMPLEMENT_OBJ_BSEARCH_GLOBAL_CMP_FN(type1, type2, nm)   \
  static int nm##_cmp_BSEARCH_CMP_FN(const void *a_, const void *b_)    \
      { \
      type1 const *a = a_; \
      type2 const *b = b_; \
      return nm##_cmp(a,b); \
      } \
  type2 *OBJ_bsearch_##nm(type1 *key, type2 const *base, int num) \
      { \
      return (type2 *)OBJ_bsearch_(key, base, num, sizeof(type2), \
                                   nm##_cmp_BSEARCH_CMP_FN); \
      } \
      extern void dummy_prototype(void)

#define OBJ_bsearch(type1,key,type2,base,num,cmp)                       \
  ((type2 *)OBJ_bsearch_(CHECKED_PTR_OF(type1,key),CHECKED_PTR_OF(type2,base), \
                         num,sizeof(type2),                             \
                         ((void)CHECKED_PTR_OF(type1,cmp##_type_1),     \
                          (void)CHECKED_PTR_OF(type2,cmp##_type_2),     \
                          cmp##_BSEARCH_CMP_FN)))

#define OBJ_bsearch_ex(type1,key,type2,base,num,cmp,flags)
  ((type2 *)OBJ_bsearch_ex_(CHECKED_PTR_OF(type1,key),CHECKED_PTR_OF(type2,base)
                         num,sizeof(type2),                             \
                         ((void)CHECKED_PTR_OF(type1,cmp##_type_1),     \
                          (void)type_2=CHECKED_PTR_OF(type2,cmp##_type_2), \
                          cmp##_BSEARCH_CMP_FN)),flags)

int             OBJ_new_nid(int num);
int             OBJ_add_object(const ASN1_OBJECT *obj);
int             OBJ_create(const char *oid,const char *sn,const char *ln);
void            OBJ_cleanup(void );
int             OBJ_create_objects(BIO *in);

int OBJ_find_sigid_algs(int signid, int *pdig_nid, int *ppkey_nid);
int OBJ_find_sigid_by_algs(int *psignid, int dig_nid, int pkey_nid);
int OBJ_add_sigid(int signid, int dig_id, int pkey_id);
void OBJ_sigid_free(void);

extern int obj_cleanup_defer;
void check_defer(int nid);

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
```



```
void ERR_load_OBJ_strings(void);

/* Error codes for the OBJ functions. */

/* Function codes. */
#define OBJ_F_OBJ_ADD_OBJECT                    105
#define OBJ_F_OBJ_CREATE                        100
#define OBJ_F_OBJ_DUP                           101
#define OBJ_F_OBJ_NAME_NEW_INDEX                106
#define OBJ_F_OBJ_NID2LN                        102
#define OBJ_F_OBJ_NID2OBJ                       103
#define OBJ_F_OBJ_NID2SN                        104

/* Reason codes. */
#define OBJ_R_MALLOC_FAILURE                    100
#define OBJ_R_UNKNOWN_NID                       101

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   24028 Wed Aug 13 19:51:46 2014
new/usr/src/lib/openssl/include/openssl/ocsp.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp.h */
/* Written by Tom Titchener <Tom_Titchener@groove.net> for the OpenSSL
 * project. */

/* History:
   This file was transfered to Richard Levitte from CertCo by Kathy
   Weinhold in mid-spring 2000 to be included in OpenSSL or released
   as a patch kit. */

/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
```



```
 */

#ifndef HEADER_OCSP_H
#define HEADER_OCSP_H

#include <openssl/ossl_typ.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/safestack.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* Various flags and values */

#define OCSP_DEFAULT_NONCE_LENGTH       16

#define OCSP_NOCERTS                    0x1
#define OCSP_NOINTERN                   0x2
#define OCSP_NOSIGS                     0x4
#define OCSP_NOCHAIN                    0x8
#define OCSP_NOVERIFY                   0x10
#define OCSP_NOEXPLICIT                 0x20
#define OCSP_NOCASIGN                   0x40
#define OCSP_NODELEGATED                0x80
#define OCSP_NOCHECKS                   0x100
#define OCSP_TRUSTOTHER                 0x200
#define OCSP_RESPID_KEY                 0x400
#define OCSP_NOTIME                     0x800

/*   CertID ::= SEQUENCE {
 *       hashAlgorithm           AlgorithmIdentifier,
 *       issuerNameHash     OCTET STRING, -- Hash of Issuer's DN
 *       issuerKeyHash      OCTET STRING, -- Hash of Issuers public key (excludi
 *       serialNumber       CertificateSerialNumber }
 */
typedef struct ocsp_cert_id_st
        {
        X509_ALGOR *hashAlgorithm;
        ASN1_OCTET_STRING *issuerNameHash;
        ASN1_OCTET_STRING *issuerKeyHash;
        ASN1_INTEGER *serialNumber;
        } OCSP_CERTID;

DECLARE_STACK_OF(OCSP_CERTID)

/*   Request ::=     SEQUENCE {
 *       reqCert                    CertID,
 *       singleRequestExtensions    [0] EXPLICIT Extensions OPTIONAL }
 */
typedef struct ocsp_one_request_st
        {
        OCSP_CERTID *reqCert;
        STACK_OF(X509_EXTENSION) *singleRequestExtensions;
        } OCSP_ONEREQ;

DECLARE_STACK_OF(OCSP_ONEREQ)
DECLARE_ASN1_SET_OF(OCSP_ONEREQ)


/*   TBSRequest      ::=     SEQUENCE {
 *       version             [0] EXPLICIT Version DEFAULT v1,
 *       requestorName       [1] EXPLICIT GeneralName OPTIONAL,
 *       requestList             SEQUENCE OF Request,
 *       requestExtensions   [2] EXPLICIT Extensions OPTIONAL }
```

```
 */
typedef struct ocsp_req_info_st
        {
        ASN1_INTEGER *version;
        GENERAL_NAME *requestorName;
        STACK_OF(OCSP_ONEREQ) *requestList;
        STACK_OF(X509_EXTENSION) *requestExtensions;
        } OCSP_REQINFO;

/*   Signature       ::=     SEQUENCE {
 *       signatureAlgorithm   AlgorithmIdentifier,
 *       signature            BIT STRING,
 *       certs                [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
 */
typedef struct ocsp_signature_st
        {
        X509_ALGOR *signatureAlgorithm;
        ASN1_BIT_STRING *signature;
        STACK_OF(X509) *certs;
        } OCSP_SIGNATURE;

/*   OCSPRequest     ::=     SEQUENCE {
 *       tbsRequest                  TBSRequest,
 *       optionalSignature   [0]     EXPLICIT Signature OPTIONAL }
 */
typedef struct ocsp_request_st
        {
        OCSP_REQINFO *tbsRequest;
        OCSP_SIGNATURE *optionalSignature; /* OPTIONAL */
        } OCSP_REQUEST;

/*   OCSPResponseStatus ::= ENUMERATED {
 *       successful            (0),      --Response has valid confirmations
 *       malformedRequest      (1),      --Illegal confirmation request
 *       internalError         (2),      --Internal error in issuer
 *       tryLater              (3),      --Try again later
 *                                       --(4) is not used
 *       sigRequired           (5),      --Must sign the request
 *       unauthorized          (6)       --Request unauthorized
 *   }
 */
#define OCSP_RESPONSE_STATUS_SUCCESSFUL         0
#define OCSP_RESPONSE_STATUS_MALFORMEDREQUEST     1
#define OCSP_RESPONSE_STATUS_INTERNALERROR        2
#define OCSP_RESPONSE_STATUS_TRYLATER             3
#define OCSP_RESPONSE_STATUS_SIGREQUIRED          5
#define OCSP_RESPONSE_STATUS_UNAUTHORIZED         6

/*   ResponseBytes ::=       SEQUENCE {
 *       responseType   OBJECT IDENTIFIER,
 *       response       OCTET STRING }
 */
typedef struct ocsp_resp_bytes_st
        {
        ASN1_OBJECT *responseType;
        ASN1_OCTET_STRING *response;
        } OCSP_RESPBYTES;

/*   OCSPResponse ::= SEQUENCE {
 *      responseStatus         OCSPResponseStatus,
 *      responseBytes          [0] EXPLICIT ResponseBytes OPTIONAL }
 */
struct ocsp_response_st
        {
        ASN1_ENUMERATED *responseStatus;
        OCSP_RESPBYTES  *responseBytes;
```



```
        };

/*   ResponderID ::= CHOICE {
 *      byName   [1] Name,
 *      byKey    [2] KeyHash }
 */
#define V_OCSP_RESPID_NAME 0
#define V_OCSP_RESPID_KEY  1
struct ocsp_responder_id_st
        {
        int type;
        union   {
                X509_NAME* byName;
                ASN1_OCTET_STRING *byKey;
                } value;
        };

DECLARE_STACK_OF(OCSP_RESPID)
DECLARE_ASN1_FUNCTIONS(OCSP_RESPID)

/*   KeyHash ::= OCTET STRING --SHA-1 hash of responder's public key
 *                            --(excluding the tag and length fields)
 */

/*   RevokedInfo ::= SEQUENCE {
 *       revocationTime              GeneralizedTime,
 *       revocationReason    [0]     EXPLICIT CRLReason OPTIONAL }
 */
typedef struct ocsp_revoked_info_st
        {
        ASN1_GENERALIZEDTIME *revocationTime;
        ASN1_ENUMERATED *revocationReason;
        } OCSP_REVOKEDINFO;

/*   CertStatus ::= CHOICE {
 *       good                [0]     IMPLICIT NULL,
 *       revoked             [1]     IMPLICIT RevokedInfo,
 *       unknown             [2]     IMPLICIT UnknownInfo }
 */
#define V_OCSP_CERTSTATUS_GOOD    0
#define V_OCSP_CERTSTATUS_REVOKED 1
#define V_OCSP_CERTSTATUS_UNKNOWN 2
typedef struct ocsp_cert_status_st
        {
        int type;
        union   {
                ASN1_NULL *good;
                OCSP_REVOKEDINFO *revoked;
                ASN1_NULL *unknown;
                } value;
        } OCSP_CERTSTATUS;

/*   SingleResponse ::= SEQUENCE {
 *      certID                       CertID,
 *      certStatus                   CertStatus,
 *      thisUpdate                   GeneralizedTime,
 *      nextUpdate           [0]     EXPLICIT GeneralizedTime OPTIONAL,
 *      singleExtensions     [1]     EXPLICIT Extensions OPTIONAL }
 */
typedef struct ocsp_single_response_st
        {
        OCSP_CERTID *certId;
        OCSP_CERTSTATUS *certStatus;
        ASN1_GENERALIZEDTIME *thisUpdate;
        ASN1_GENERALIZEDTIME *nextUpdate;
        STACK_OF(X509_EXTENSION) *singleExtensions;
```

```
260         } OCSP_SINGLERESP;

262 DECLARE_STACK_OF(OCSP_SINGLERESP)
263 DECLARE_ASN1_SET_OF(OCSP_SINGLERESP)

265 /*   ResponseData ::= SEQUENCE {
266  *      version             [0] EXPLICIT Version DEFAULT v1,
267  *      responderID             ResponderID,
268  *      producedAt              GeneralizedTime,
269  *      responses               SEQUENCE OF SingleResponse,
270  *      responseExtensions  [1] EXPLICIT Extensions OPTIONAL }
271  */
272 typedef struct ocsp_response_data_st
273         {
274         ASN1_INTEGER *version;
275         OCSP_RESPID  *responderId;
276         ASN1_GENERALIZEDTIME *producedAt;
277         STACK_OF(OCSP_SINGLERESP) *responses;
278         STACK_OF(X509_EXTENSION) *responseExtensions;
279         } OCSP_RESPDATA;

281 /*   BasicOCSPResponse       ::= SEQUENCE {
282  *      tbsResponseData      ResponseData,
283  *      signatureAlgorithm   AlgorithmIdentifier,
284  *      signature            BIT STRING,
285  *      certs                [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }
286  */
287   /* Note 1:
288      The value for "signature" is specified in the OCSP rfc2560 as follows:
289      "The value for the signature SHALL be computed on the hash of the DER
290      encoding ResponseData."  This means that you must hash the DER-encoded
291      tbsResponseData, and then run it through a crypto-signing function, which
292      will (at least w/RSA) do a hash-'n'-private-encrypt operation.  This seems
293      a bit odd, but that's the spec.  Also note that the data structures do not
294      leave anywhere to independently specify the algorithm used for the initial
295      hash. So, we look at the signature-specification algorithm, and try to do
296      something intelligent.     -- Kathy Weinhold, CertCo */
297   /* Note 2:
298      It seems that the mentioned passage from RFC 2560 (section 4.2.1) is open
299      for interpretation.  I've done tests against another responder, and found
300      that it doesn't do the double hashing that the RFC seems to say one
301      should.  Therefore, all relevant functions take a flag saying which
302      variant should be used.    -- Richard Levitte, OpenSSL team and CeloCom */
303 typedef struct ocsp_basic_response_st
304         {
305         OCSP_RESPDATA *tbsResponseData;
306         X509_ALGOR *signatureAlgorithm;
307         ASN1_BIT_STRING *signature;
308         STACK_OF(X509) *certs;
309         } OCSP_BASICRESP;

311 /*
312  *   CRLReason ::= ENUMERATED {
313  *        unspecified             (0),
314  *        keyCompromise           (1),
315  *        cACompromise            (2),
316  *        affiliationChanged      (3),
317  *        superseded              (4),
318  *        cessationOfOperation    (5),
319  *        certificateHold         (6),
320  *        removeFromCRL           (8) }
321  */
322 #define OCSP_REVOKED_STATUS_NOSTATUS               -1
323 #define OCSP_REVOKED_STATUS_UNSPECIFIED             0
324 #define OCSP_REVOKED_STATUS_KEYCOMPROMISE           1
325 #define OCSP_REVOKED_STATUS_CACOMPROMISE            2
```



```
326 #define OCSP_REVOKED_STATUS_AFFILIATIONCHANGED      3
327 #define OCSP_REVOKED_STATUS_SUPERSEDED              4
328 #define OCSP_REVOKED_STATUS_CESSATIONOFOPERATION    5
329 #define OCSP_REVOKED_STATUS_CERTIFICATEHOLD         6
330 #define OCSP_REVOKED_STATUS_REMOVEFROMCRL           8

332 /* CrlID ::= SEQUENCE {
333  *     crlUrl               [0]     EXPLICIT IA5String OPTIONAL,
334  *     crlNum               [1]     EXPLICIT INTEGER OPTIONAL,
335  *     crlTime              [2]     EXPLICIT GeneralizedTime OPTIONAL }
336  */
337 typedef struct ocsp_crl_id_st
338         {
339         ASN1_IA5STRING *crlUrl;
340         ASN1_INTEGER *crlNum;
341         ASN1_GENERALIZEDTIME *crlTime;
342         } OCSP_CRLID;

344 /* ServiceLocator ::= SEQUENCE {
345  *      issuer    Name,
346  *      locator   AuthorityInfoAccessSyntax OPTIONAL }
347  */
348 typedef struct ocsp_service_locator_st
349         {
350         X509_NAME* issuer;
351         STACK_OF(ACCESS_DESCRIPTION) *locator;
352         } OCSP_SERVICELOC;

354 #define PEM_STRING_OCSP_REQUEST "OCSP REQUEST"
355 #define PEM_STRING_OCSP_RESPONSE "OCSP RESPONSE"

357 #define d2i_OCSP_REQUEST_bio(bp,p) ASN1_d2i_bio_of(OCSP_REQUEST,OCSP_REQUEST_new

359 #define d2i_OCSP_RESPONSE_bio(bp,p) ASN1_d2i_bio_of(OCSP_RESPONSE,OCSP_RESPONSE_

361 #define PEM_read_bio_OCSP_REQUEST(bp,x,cb) (OCSP_REQUEST *)PEM_ASN1_read_bio( \
362      (char *(*)())d2i_OCSP_REQUEST,PEM_STRING_OCSP_REQUEST,bp,(char **)x,cb,NULL

364 #define PEM_read_bio_OCSP_RESPONSE(bp,x,cb)(OCSP_RESPONSE *)PEM_ASN1_read_bio(\
365      (char *(*)())d2i_OCSP_RESPONSE,PEM_STRING_OCSP_RESPONSE,bp,(char **)x,cb,NU

367 #define PEM_write_bio_OCSP_REQUEST(bp,o) \
368     PEM_ASN1_write_bio((int (*)())i2d_OCSP_REQUEST,PEM_STRING_OCSP_REQUEST,\
369                        bp,(char *)o, NULL,NULL,0,NULL,NULL)

371 #define PEM_write_bio_OCSP_RESPONSE(bp,o) \
372     PEM_ASN1_write_bio((int (*)())i2d_OCSP_RESPONSE,PEM_STRING_OCSP_RESPONSE,\
373                        bp,(char *)o, NULL,NULL,0,NULL,NULL)

375 #define i2d_OCSP_RESPONSE_bio(bp,o) ASN1_i2d_bio_of(OCSP_RESPONSE,i2d_OCSP_RESPO

377 #define i2d_OCSP_REQUEST_bio(bp,o) ASN1_i2d_bio_of(OCSP_REQUEST,i2d_OCSP_REQUEST

379 #define OCSP_REQUEST_sign(o,pkey,md) \
380         ASN1_item_sign(ASN1_ITEM_rptr(OCSP_REQINFO),\
381                 o->optionalSignature->signatureAlgorithm,NULL,\
382                 o->optionalSignature->signature,o->tbsRequest,pkey,md)

384 #define OCSP_BASICRESP_sign(o,pkey,md,d) \
385         ASN1_item_sign(ASN1_ITEM_rptr(OCSP_RESPDATA),o->signatureAlgorithm,NULL,
386                 o->signature,o->tbsResponseData,pkey,md)

388 #define OCSP_REQUEST_verify(a,r) ASN1_item_verify(ASN1_ITEM_rptr(OCSP_REQINFO),\
389         a->optionalSignature->signatureAlgorithm,\
390         a->optionalSignature->signature,a->tbsRequest,r)
```

```
#define OCSP_BASICRESP_verify(a,r,d) ASN1_item_verify(ASN1_ITEM_rptr(OCSP_RESPDA
        a->signatureAlgorithm,a->signature,a->tbsResponseData,r)

#define ASN1_BIT_STRING_digest(data,type,md,len) \
        ASN1_item_digest(ASN1_ITEM_rptr(ASN1_BIT_STRING),type,data,md,len)

#define OCSP_CERTSTATUS_dup(cs)\
                (OCSP_CERTSTATUS*)ASN1_dup((int(*)())i2d_OCSP_CERTSTATUS,\
                (char *(*)())d2i_OCSP_CERTSTATUS,(char *)(cs))

OCSP_CERTID *OCSP_CERTID_dup(OCSP_CERTID *id);

OCSP_RESPONSE *OCSP_sendreq_bio(BIO *b, char *path, OCSP_REQUEST *req);
OCSP_REQ_CTX *OCSP_sendreq_new(BIO *io, char *path, OCSP_REQUEST *req,
                                                        int maxline);
int OCSP_sendreq_nbio(OCSP_RESPONSE **presp, OCSP_REQ_CTX *rctx);
void OCSP_REQ_CTX_free(OCSP_REQ_CTX *rctx);
int OCSP_REQ_CTX_set1_req(OCSP_REQ_CTX *rctx, OCSP_REQUEST *req);
int OCSP_REQ_CTX_add1_header(OCSP_REQ_CTX *rctx,
                const char *name, const char *value);

OCSP_CERTID *OCSP_cert_to_id(const EVP_MD *dgst, X509 *subject, X509 *issuer);

OCSP_CERTID *OCSP_cert_id_new(const EVP_MD *dgst,
                              X509_NAME *issuerName,
                              ASN1_BIT_STRING* issuerKey,
                              ASN1_INTEGER *serialNumber);

OCSP_ONEREQ *OCSP_request_add0_id(OCSP_REQUEST *req, OCSP_CERTID *cid);

int OCSP_request_add1_nonce(OCSP_REQUEST *req, unsigned char *val, int len);
int OCSP_basic_add1_nonce(OCSP_BASICRESP *resp, unsigned char *val, int len);
int OCSP_check_nonce(OCSP_REQUEST *req, OCSP_BASICRESP *bs);
int OCSP_copy_nonce(OCSP_BASICRESP *resp, OCSP_REQUEST *req);

int OCSP_request_set1_name(OCSP_REQUEST *req, X509_NAME *nm);
int OCSP_request_add1_cert(OCSP_REQUEST *req, X509 *cert);

int OCSP_request_sign(OCSP_REQUEST   *req,
                      X509           *signer,
                      EVP_PKEY       *key,
                      const EVP_MD   *dgst,
                      STACK_OF(X509) *certs,
                      unsigned long flags);

int OCSP_response_status(OCSP_RESPONSE *resp);
OCSP_BASICRESP *OCSP_response_get1_basic(OCSP_RESPONSE *resp);

int OCSP_resp_count(OCSP_BASICRESP *bs);
OCSP_SINGLERESP *OCSP_resp_get0(OCSP_BASICRESP *bs, int idx);
int OCSP_resp_find(OCSP_BASICRESP *bs, OCSP_CERTID *id, int last);
int OCSP_single_get0_status(OCSP_SINGLERESP *single, int *reason,
                            ASN1_GENERALIZEDTIME **revtime,
                            ASN1_GENERALIZEDTIME **thisupd,
                            ASN1_GENERALIZEDTIME **nextupd);
int OCSP_resp_find_status(OCSP_BASICRESP *bs, OCSP_CERTID *id, int *status,
                            int *reason,
                            ASN1_GENERALIZEDTIME **revtime,
                            ASN1_GENERALIZEDTIME **thisupd,
                            ASN1_GENERALIZEDTIME **nextupd);
int OCSP_check_validity(ASN1_GENERALIZEDTIME *thisupd,
                        ASN1_GENERALIZEDTIME *nextupd,
                        long sec, long maxsec);

int OCSP_request_verify(OCSP_REQUEST *req, STACK_OF(X509) *certs, X509_STORE *st
```



```
int OCSP_parse_url(char *url, char **phost, char **pport, char **ppath, int *pss

int OCSP_id_issuer_cmp(OCSP_CERTID *a, OCSP_CERTID *b);
int OCSP_id_cmp(OCSP_CERTID *a, OCSP_CERTID *b);

int OCSP_request_onereq_count(OCSP_REQUEST *req);
OCSP_ONEREQ *OCSP_request_onereq_get0(OCSP_REQUEST *req, int i);
OCSP_CERTID *OCSP_onereq_get0_id(OCSP_ONEREQ *one);
int OCSP_id_get0_info(ASN1_OCTET_STRING **piNameHash, ASN1_OBJECT **pmd,
                      ASN1_OCTET_STRING **pikeyHash,
                      ASN1_INTEGER **pserial, OCSP_CERTID *cid);
int OCSP_request_is_signed(OCSP_REQUEST *req);
OCSP_RESPONSE *OCSP_response_create(int status, OCSP_BASICRESP *bs);
OCSP_SINGLERESP *OCSP_basic_add1_status(OCSP_BASICRESP *rsp,
                                        OCSP_CERTID *cid,
                                        int status, int reason,
                                        ASN1_TIME *revtime,
                                      ASN1_TIME *thisupd, ASN1_TIME *nextupd);
int OCSP_basic_add1_cert(OCSP_BASICRESP *resp, X509 *cert);
int OCSP_basic_sign(OCSP_BASICRESP *brsp,
                      X509 *signer, EVP_PKEY *key, const EVP_MD *dgst,
                      STACK_OF(X509) *certs, unsigned long flags);

X509_EXTENSION *OCSP_crlID_new(char *url, long *n, char *tim);

X509_EXTENSION *OCSP_accept_responses_new(char **oids);

X509_EXTENSION *OCSP_archive_cutoff_new(char* tim);

X509_EXTENSION *OCSP_url_svcloc_new(X509_NAME* issuer, char **urls);

int OCSP_REQUEST_get_ext_count(OCSP_REQUEST *x);
int OCSP_REQUEST_get_ext_by_NID(OCSP_REQUEST *x, int nid, int lastpos);
int OCSP_REQUEST_get_ext_by_OBJ(OCSP_REQUEST *x, ASN1_OBJECT *obj, int lastpos);
int OCSP_REQUEST_get_ext_by_critical(OCSP_REQUEST *x, int crit, int lastpos);
X509_EXTENSION *OCSP_REQUEST_get_ext(OCSP_REQUEST *x, int loc);
X509_EXTENSION *OCSP_REQUEST_delete_ext(OCSP_REQUEST *x, int loc);
void *OCSP_REQUEST_get1_ext_d2i(OCSP_REQUEST *x, int nid, int *crit, int *idx);
int OCSP_REQUEST_add1_ext_i2d(OCSP_REQUEST *x, int nid, void *value, int crit,
                                                        unsigned long flags);
int OCSP_REQUEST_add_ext(OCSP_REQUEST *x, X509_EXTENSION *ex, int loc);

int OCSP_ONEREQ_get_ext_count(OCSP_ONEREQ *x);
int OCSP_ONEREQ_get_ext_by_NID(OCSP_ONEREQ *x, int nid, int lastpos);
int OCSP_ONEREQ_get_ext_by_OBJ(OCSP_ONEREQ *x, ASN1_OBJECT *obj, int lastpos);
int OCSP_ONEREQ_get_ext_by_critical(OCSP_ONEREQ *x, int crit, int lastpos);
X509_EXTENSION *OCSP_ONEREQ_get_ext(OCSP_ONEREQ *x, int loc);
X509_EXTENSION *OCSP_ONEREQ_delete_ext(OCSP_ONEREQ *x, int loc);
void *OCSP_ONEREQ_get1_ext_d2i(OCSP_ONEREQ *x, int nid, int *crit, int *idx);
int OCSP_ONEREQ_add1_ext_i2d(OCSP_ONEREQ *x, int nid, void *value, int crit,
                                                        unsigned long flags);
int OCSP_ONEREQ_add_ext(OCSP_ONEREQ *x, X509_EXTENSION *ex, int loc);

int OCSP_BASICRESP_get_ext_count(OCSP_BASICRESP *x);
int OCSP_BASICRESP_get_ext_by_NID(OCSP_BASICRESP *x, int nid, int lastpos);
int OCSP_BASICRESP_get_ext_by_OBJ(OCSP_BASICRESP *x, ASN1_OBJECT *obj, int lastp
int OCSP_BASICRESP_get_ext_by_critical(OCSP_BASICRESP *x, int crit, int lastpos)
X509_EXTENSION *OCSP_BASICRESP_get_ext(OCSP_BASICRESP *x, int loc);
X509_EXTENSION *OCSP_BASICRESP_delete_ext(OCSP_BASICRESP *x, int loc);
void *OCSP_BASICRESP_get1_ext_d2i(OCSP_BASICRESP *x, int nid, int *crit, int *id
int OCSP_BASICRESP_add1_ext_i2d(OCSP_BASICRESP *x, int nid, void *value, int cri
                                                        unsigned long flags);
int OCSP_BASICRESP_add_ext(OCSP_BASICRESP *x, X509_EXTENSION *ex, int loc);

int OCSP_SINGLERESP_get_ext_count(OCSP_SINGLERESP *x);
int OCSP_SINGLERESP_get_ext_by_NID(OCSP_SINGLERESP *x, int nid, int lastpos);
```

```
524 int OCSP_SINGLERESP_get_ext_by_OBJ(OCSP_SINGLERESP *x, ASN1_OBJECT *obj, int las
525 int OCSP_SINGLERESP_get_ext_by_critical(OCSP_SINGLERESP *x, int crit, int lastpo
526 X509_EXTENSION *OCSP_SINGLERESP_get_ext(OCSP_SINGLERESP *x, int loc);
527 X509_EXTENSION *OCSP_SINGLERESP_delete_ext(OCSP_SINGLERESP *x, int loc);
528 void *OCSP_SINGLERESP_get1_ext_d2i(OCSP_SINGLERESP *x, int nid, int *crit, int *
529 int OCSP_SINGLERESP_add1_ext_i2d(OCSP_SINGLERESP *x, int nid, void *value, int c
530                                                     unsigned long flags);
531 int OCSP_SINGLERESP_add_ext(OCSP_SINGLERESP *x, X509_EXTENSION *ex, int loc);

533 DECLARE_ASN1_FUNCTIONS(OCSP_SINGLERESP)
534 DECLARE_ASN1_FUNCTIONS(OCSP_CERTSTATUS)
535 DECLARE_ASN1_FUNCTIONS(OCSP_REVOKEDINFO)
536 DECLARE_ASN1_FUNCTIONS(OCSP_BASICRESP)
537 DECLARE_ASN1_FUNCTIONS(OCSP_RESPDATA)
538 DECLARE_ASN1_FUNCTIONS(OCSP_RESPID)
539 DECLARE_ASN1_FUNCTIONS(OCSP_RESPONSE)
540 DECLARE_ASN1_FUNCTIONS(OCSP_RESPBYTES)
541 DECLARE_ASN1_FUNCTIONS(OCSP_ONEREQ)
542 DECLARE_ASN1_FUNCTIONS(OCSP_CERTID)
543 DECLARE_ASN1_FUNCTIONS(OCSP_REQUEST)
544 DECLARE_ASN1_FUNCTIONS(OCSP_SIGNATURE)
545 DECLARE_ASN1_FUNCTIONS(OCSP_REQINFO)
546 DECLARE_ASN1_FUNCTIONS(OCSP_CRLID)
547 DECLARE_ASN1_FUNCTIONS(OCSP_SERVICELOC)

549 const char *OCSP_response_status_str(long s);
550 const char *OCSP_cert_status_str(long s);
551 const char *OCSP_crl_reason_str(long s);

553 int OCSP_REQUEST_print(BIO *bp, OCSP_REQUEST* a, unsigned long flags);
554 int OCSP_RESPONSE_print(BIO *bp, OCSP_RESPONSE* o, unsigned long flags);

556 int OCSP_basic_verify(OCSP_BASICRESP *bs, STACK_OF(X509) *certs,
557                                 X509_STORE *st, unsigned long flags);

559 /* BEGIN ERROR CODES */
560 /* The following lines are auto generated by the script mkerr.pl. Any changes
561  * made after this point may be overwritten when the script is next run.
562  */
563 void ERR_load_OCSP_strings(void);

565 /* Error codes for the OCSP functions. */

567 /* Function codes. */
568 #define OCSP_F_ASN1_STRING_ENCODE                       100
569 #define OCSP_F_D2I_OCSP_NONCE                           102
570 #define OCSP_F_OCSP_BASIC_ADD1_STATUS                   103
571 #define OCSP_F_OCSP_BASIC_SIGN                          104
572 #define OCSP_F_OCSP_BASIC_VERIFY                        105
573 #define OCSP_F_OCSP_CERT_ID_NEW                         101
574 #define OCSP_F_OCSP_CHECK_DELEGATED                     106
575 #define OCSP_F_OCSP_CHECK_IDS                           107
576 #define OCSP_F_OCSP_CHECK_ISSUER                        108
577 #define OCSP_F_OCSP_CHECK_VALIDITY                      115
578 #define OCSP_F_OCSP_MATCH_ISSUERID                      109
579 #define OCSP_F_OCSP_PARSE_URL                           114
580 #define OCSP_F_OCSP_REQUEST_SIGN                        110
581 #define OCSP_F_OCSP_REQUEST_VERIFY                      116
582 #define OCSP_F_OCSP_RESPONSE_GET1_BASIC                 111
583 #define OCSP_F_OCSP_SENDREQ_BIO                         112
584 #define OCSP_F_OCSP_SENDREQ_NBIO                        117
585 #define OCSP_F_PARSE_HTTP_LINE1                         118
586 #define OCSP_F_REQUEST_VERIFY                           113

588 /* Reason codes. */
589 #define OCSP_R_BAD_DATA                                 100
```



```
590 #define OCSP_R_CERTIFICATE_VERIFY_ERROR                 101
591 #define OCSP_R_DIGEST_ERR                               102
592 #define OCSP_R_ERROR_IN_NEXTUPDATE_FIELD                122
593 #define OCSP_R_ERROR_IN_THISUPDATE_FIELD                123
594 #define OCSP_R_ERROR_PARSING_URL                        121
595 #define OCSP_R_MISSING_OCSPSIGNING_USAGE                103
596 #define OCSP_R_NEXTUPDATE_BEFORE_THISUPDATE             124
597 #define OCSP_R_NOT_BASIC_RESPONSE                       104
598 #define OCSP_R_NO_CERTIFICATES_IN_CHAIN                 105
599 #define OCSP_R_NO_CONTENT                               106
600 #define OCSP_R_NO_PUBLIC_KEY                            107
601 #define OCSP_R_NO_RESPONSE_DATA                         108
602 #define OCSP_R_NO_REVOKED_TIME                          109
603 #define OCSP_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE   110
604 #define OCSP_R_REQUEST_NOT_SIGNED                       128
605 #define OCSP_R_RESPONSE_CONTAINS_NO_REVOCATION_DATA     111
606 #define OCSP_R_ROOT_CA_NOT_TRUSTED                      112
607 #define OCSP_R_SERVER_READ_ERROR                        113
608 #define OCSP_R_SERVER_RESPONSE_ERROR                    114
609 #define OCSP_R_SERVER_RESPONSE_PARSE_ERROR              115
610 #define OCSP_R_SERVER_WRITE_ERROR                       116
611 #define OCSP_R_SIGNATURE_FAILURE                        117
612 #define OCSP_R_SIGNER_CERTIFICATE_NOT_FOUND             118
613 #define OCSP_R_STATUS_EXPIRED                           125
614 #define OCSP_R_STATUS_NOT_YET_VALID                     126
615 #define OCSP_R_STATUS_TOO_OLD                           127
616 #define OCSP_R_UNKNOWN_MESSAGE_DIGEST                   119
617 #define OCSP_R_UNKNOWN_NID                              120
618 #define OCSP_R_UNSUPPORTED_REQUESTORNAME_TYPE           129

620 #ifdef  __cplusplus
621 }
622 #endif
623 #endif
624 #endif /* ! codereview */
```

```
**********************************************************
   10126 Wed Aug 13 19:51:46 2014
new/usr/src/lib/openssl/include/openssl/opensslconf.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* opensslconf.h */
/* WARNING: Generated automatically from opensslconf.h.in by Configure. */

/* OpenSSL was configured with the following options: */
#ifndef OPENSSL_DOING_MAKEDEPEND


#ifndef OPENSSL_NO_EC
# define OPENSSL_NO_EC
#endif
#ifndef OPENSSL_NO_EC_NISTP_64_GCC_128
# define OPENSSL_NO_EC_NISTP_64_GCC_128
#endif
#ifndef OPENSSL_NO_ECDH
# define OPENSSL_NO_ECDH
#endif
#ifndef OPENSSL_NO_ECDSA
# define OPENSSL_NO_ECDSA
#endif
#ifndef OPENSSL_NO_GMP
# define OPENSSL_NO_GMP
#endif
#ifndef OPENSSL_NO_GOST
# define OPENSSL_NO_GOST
#endif
#ifndef OPENSSL_NO_HW_4758_CCA
# define OPENSSL_NO_HW_4758_CCA
#endif
#ifndef OPENSSL_NO_HW_AEP
# define OPENSSL_NO_HW_AEP
#endif
#ifndef OPENSSL_NO_HW_ATALLA
# define OPENSSL_NO_HW_ATALLA
#endif
#ifndef OPENSSL_NO_HW_CHIL
# define OPENSSL_NO_HW_CHIL
#endif
#ifndef OPENSSL_NO_HW_CSWIFT
# define OPENSSL_NO_HW_CSWIFT
#endif
#ifndef OPENSSL_NO_HW_GMP
# define OPENSSL_NO_HW_GMP
#endif
#ifndef OPENSSL_NO_HW_NCIPHER
# define OPENSSL_NO_HW_NCIPHER
#endif
#ifndef OPENSSL_NO_HW_NURON
# define OPENSSL_NO_HW_NURON
#endif
#ifndef OPENSSL_NO_HW_PADLOCK
# define OPENSSL_NO_HW_PADLOCK
#endif
#ifndef OPENSSL_NO_HW_SUREWARE
# define OPENSSL_NO_HW_SUREWARE
#endif
#ifndef OPENSSL_NO_HW_UBSEC
# define OPENSSL_NO_HW_UBSEC
#endif
#ifndef OPENSSL_NO_IDEA
# define OPENSSL_NO_IDEA
#endif
```



```
#ifndef OPENSSL_NO_JPAKE
# define OPENSSL_NO_JPAKE
#endif
#ifndef OPENSSL_NO_KRB5
# define OPENSSL_NO_KRB5
#endif
#ifndef OPENSSL_NO_MDC2
# define OPENSSL_NO_MDC2
#endif
#ifndef OPENSSL_NO_RC3
# define OPENSSL_NO_RC3
#endif
#ifndef OPENSSL_NO_RC5
# define OPENSSL_NO_RC5
#endif
#ifndef OPENSSL_NO_RFC3779
# define OPENSSL_NO_RFC3779
#endif
#ifndef OPENSSL_NO_SCTP
# define OPENSSL_NO_SCTP
#endif
#ifndef OPENSSL_NO_SEED
# define OPENSSL_NO_SEED
#endif
#ifndef OPENSSL_NO_STORE
# define OPENSSL_NO_STORE
#endif
#ifndef OPENSSL_NO_UNIT_TEST
# define OPENSSL_NO_UNIT_TEST
#endif
#ifndef OPENSSL_NO_WHIRLPOOL
# define OPENSSL_NO_WHIRLPOOL
#endif
#ifndef OPENSSL_NO_WHRLPOOL
# define OPENSSL_NO_WHRLPOOL
#endif

#endif /* OPENSSL_DOING_MAKEDEPEND */

#ifndef OPENSSL_THREADS
# define OPENSSL_THREADS
#endif
#ifndef OPENSSL_NO_STATIC_ENGINE
# define OPENSSL_NO_STATIC_ENGINE
#endif

/* The OPENSSL_NO_* macros are also defined as NO_* if the application
   asks for it.  This is a transient feature that is provided for those
   who haven't had the time to do the appropriate changes in their
   applications.  */
#ifdef OPENSSL_ALGORITHM_DEFINES
# if defined(OPENSSL_NO_EC) && !defined(NO_EC)
#  define NO_EC
# endif
# if defined(OPENSSL_NO_EC_NISTP_64_GCC_128) && !defined(NO_EC_NISTP_64_GCC_128)
#  define NO_EC_NISTP_64_GCC_128
# endif
# if defined(OPENSSL_NO_ECDH) && !defined(NO_ECDH)
#  define NO_ECDH
# endif
# if defined(OPENSSL_NO_ECDSA) && !defined(NO_ECDSA)
#  define NO_ECDSA
# endif
# if defined(OPENSSL_NO_GMP) && !defined(NO_GMP)
#  define NO_GMP
# endif
```

```
128 # if defined(OPENSSL_NO_GOST) && !defined(NO_GOST)
129 #  define NO_GOST
130 # endif
131 # if defined(OPENSSL_NO_HW_4758_CCA) && !defined(NO_HW_4758_CCA)
132 #  define NO_HW_4758_CCA
133 # endif
134 # if defined(OPENSSL_NO_HW_AEP) && !defined(NO_HW_AEP)
135 #  define NO_HW_AEP
136 # endif
137 # if defined(OPENSSL_NO_HW_ATALLA) && !defined(NO_HW_ATALLA)
138 #  define NO_HW_ATALLA
139 # endif
140 # if defined(OPENSSL_NO_HW_CHIL) && !defined(NO_HW_CHIL)
141 #  define NO_HW_CHIL
142 # endif
143 # if defined(OPENSSL_NO_HW_CSWIFT) && !defined(NO_HW_CSWIFT)
144 #  define NO_HW_CSWIFT
145 # endif
146 # if defined(OPENSSL_NO_HW_GMP) && !defined(NO_HW_GMP)
147 #  define NO_HW_GMP
148 # endif
149 # if defined(OPENSSL_NO_HW_NCIPHER) && !defined(NO_HW_NCIPHER)
150 #  define NO_HW_NCIPHER
151 # endif
152 # if defined(OPENSSL_NO_HW_NURON) && !defined(NO_HW_NURON)
153 #  define NO_HW_NURON
154 # endif
155 # if defined(OPENSSL_NO_HW_PADLOCK) && !defined(NO_HW_PADLOCK)
156 #  define NO_HW_PADLOCK
157 # endif
158 # if defined(OPENSSL_NO_HW_SUREWARE) && !defined(NO_HW_SUREWARE)
159 #  define NO_HW_SUREWARE
160 # endif
161 # if defined(OPENSSL_NO_HW_UBSEC) && !defined(NO_HW_UBSEC)
162 #  define NO_HW_UBSEC
163 # endif
164 # if defined(OPENSSL_NO_IDEA) && !defined(NO_IDEA)
165 #  define NO_IDEA
166 # endif
167 # if defined(OPENSSL_NO_JPAKE) && !defined(NO_JPAKE)
168 #  define NO_JPAKE
169 # endif
170 # if defined(OPENSSL_NO_KRB5) && !defined(NO_KRB5)
171 #  define NO_KRB5
172 # endif
173 # if defined(OPENSSL_NO_MDC2) && !defined(NO_MDC2)
174 #  define NO_MDC2
175 # endif
176 # if defined(OPENSSL_NO_RC3) && !defined(NO_RC3)
177 #  define NO_RC3
178 # endif
179 # if defined(OPENSSL_NO_RC5) && !defined(NO_RC5)
180 #  define NO_RC5
181 # endif
182 # if defined(OPENSSL_NO_RFC3779) && !defined(NO_RFC3779)
183 #  define NO_RFC3779
184 # endif
185 # if defined(OPENSSL_NO_SCTP) && !defined(NO_SCTP)
186 #  define NO_SCTP
187 # endif
188 # if defined(OPENSSL_NO_SEED) && !defined(NO_SEED)
189 #  define NO_SEED
190 # endif
191 # if defined(OPENSSL_NO_STORE) && !defined(NO_STORE)
192 #  define NO_STORE
193 # endif
```



```
194 # if defined(OPENSSL_NO_UNIT_TEST) && !defined(NO_UNIT_TEST)
195 #  define NO_UNIT_TEST
196 # endif
197 # if defined(OPENSSL_NO_WHIRLPOOL) && !defined(NO_WHIRLPOOL)
198 #  define NO_WHIRLPOOL
199 # endif
200 # if defined(OPENSSL_NO_WHRLPOOL) && !defined(NO_WHRLPOOL)
201 #  define NO_WHRLPOOL
202 # endif
203 #endif

205 #define OPENSSL_CPUID_OBJ

207 /* crypto/opensslconf.h.in */

209 #include <openssl/sunw_prefix.h>

211 /* Generate 80386 code? */
212 #undef I386_ONLY

214 #if !(defined(VMS) || defined(__VMS)) /* VMS uses logical names instead */
215 #if defined(HEADER_CRYPTLIB_H) && !defined(OPENSSLDIR)
216 #if defined(__x86_64)
217 #define ENGINESDIR "/usr/lib/openssl/engines/64"
218 #else
219 #define ENGINESDIR "/usr/lib/openssl/engines"
220 #endif
221 #define OPENSSLDIR "/etc/openssl"
222 #endif
223 #endif

225 #undef OPENSSL_UNISTD
226 #define OPENSSL_UNISTD <unistd.h>

228 #undef OPENSSL_EXPORT_VAR_AS_FUNCTION

230 #if defined(HEADER_IDEA_H) && !defined(IDEA_INT)
231 #define IDEA_INT unsigned int
232 #endif

234 #if defined(HEADER_MD2_H) && !defined(MD2_INT)
235 #define MD2_INT unsigned int
236 #endif

238 #if defined(HEADER_RC2_H) && !defined(RC2_INT)
239 /* I need to put in a mod for the alpha - eay */
240 #define RC2_INT unsigned int
241 #endif

243 #if defined(HEADER_RC4_H)
244 #if !defined(RC4_INT)
245 /* using int types make the structure larger but make the code faster
246  * on most boxes I have tested - up to %20 faster. */
247 /*
248  * I don't know what does "most" mean, but declaring "int" is a must on:
249  * - Intel P6 because partial register stalls are very expensive;
250  * - elder Alpha because it lacks byte load/store instructions;
251  */
252 #define RC4_INT unsigned int
253 #endif
254 #if !defined(RC4_CHUNK)
255 /*
256  * This enables code handling data aligned at natural CPU word
257  * boundary. See crypto/rc4/rc4_enc.c for further details.
258  */
259 #if defined(__x86_64)
```

```
#define RC4_CHUNK unsigned long
#else
#undef RC4_CHUNK
#endif  /* __x86_64 */
#endif
#endif

#if (defined(HEADER_NEW_DES_H) || defined(HEADER_DES_H)) && !defined(DES_LONG)
/* If this is set to 'unsigned int' on a DEC Alpha, this gives about a
 * %20 speed up (longs are 8 bytes, int's are 4). */
#ifndef DES_LONG
#if defined(__x86_64)
#define DES_LONG unsigned int
#else
#define DES_LONG unsigned long
#endif
#endif
#endif

#if defined(HEADER_BN_H) && !defined(CONFIG_HEADER_BN_H)
#define CONFIG_HEADER_BN_H
#if defined(__x86_64)
#undef BN_LLONG
#else
#define BN_LLONG
#endif

/* Should we define BN_DIV2W here? */

/* Only one for the following should be defined */
#if defined(__x86_64)
#define SIXTY_FOUR_BIT_LONG
#undef THIRTY_TWO_BIT
#else
#undef SIXTY_FOUR_BIT_LONG
#undef SIXTY_FOUR_BIT
#define THIRTY_TWO_BIT
#endif
#undef SIXTY_FOUR_BIT
#endif

#if defined(HEADER_RC4_LOCL_H) && !defined(CONFIG_HEADER_RC4_LOCL_H)
#define CONFIG_HEADER_RC4_LOCL_H
/* if this is defined data[i] is used instead of *data, this is a %20
 * speedup on x86 */
#if defined(__x86_64)
#undef  RC4_INDEX
#else
#define RC4_INDEX
#endif /* __x86_64 */
#endif

#if defined(HEADER_BF_LOCL_H) && !defined(CONFIG_HEADER_BF_LOCL_H)
#define CONFIG_HEADER_BF_LOCL_H
#define BF_PTR
#endif /* HEADER_BF_LOCL_H */

#if defined(HEADER_DES_LOCL_H) && !defined(CONFIG_HEADER_DES_LOCL_H)
#define CONFIG_HEADER_DES_LOCL_H
#ifndef DES_DEFAULT_OPTIONS
/* the following is tweaked from a config script, that is why it is a
 * protected undef/define */
#ifndef DES_PTR
#if defined(__x86_64)
#undef DES_PTR
#else
```



```
#define DES_PTR
#endif /* __x86_64 */
#endif

/* This helps C compiler generate the correct code for multiple functional
 * units.  It reduces register dependancies at the expense of 2 more
 * registers */
#ifndef DES_RISC1
#if defined(__x86_64)
#undef DES_RISC1
#else
#define        DES_RISC1
#endif /* __x86_64 */
#endif

#ifndef DES_RISC2
#undef DES_RISC2
#endif

#if defined(DES_RISC1) && defined(DES_RISC2)
YOU SHOULD NOT HAVE BOTH DES_RISC1 AND DES_RISC2 DEFINED!!!!!
#endif

/* Unroll the inner loop, this sometimes helps, sometimes hinders.
 * Very mucy CPU dependant */
#ifndef DES_UNROLL
#define DES_UNROLL
#endif

/* These default values were supplied by
 * Peter Gutman <pgut001@cs.auckland.ac.nz>
 * They are only used if nothing else has been defined */
#if !defined(DES_PTR) && !defined(DES_RISC1) && !defined(DES_RISC2) && !defined(
/* Special defines which change the way the code is built depending on the
   CPU and OS.  For SGI machines you can use _MIPS_SZLONG (32 or 64) to find
   even newer MIPS CPU's, but at the moment one size fits all for
   optimization options.  Older Sparc's work better with only UNROLL, but
   there's no way to tell at compile time what it is you're running on */

#if defined( sun )              /* Newer Sparc's */
#  define DES_PTR
#  define DES_RISC1
#  define DES_UNROLL
#elif defined( __ultrix )       /* Older MIPS */
#  define DES_PTR
#  define DES_RISC2
#  define DES_UNROLL
#elif defined( __osf1__ )       /* Alpha */
#  define DES_PTR
#  define DES_RISC2
#elif defined ( _AIX )          /* RS6000 */
  /* Unknown */
#elif defined( __hpux )         /* HP-PA */
  /* Unknown */
#elif defined( __aux )          /* 68K */
  /* Unknown */
#elif defined( __dgux )         /* 88K (but P6 in latest boxes) */
#  define DES_UNROLL
#elif defined( __sgi )          /* Newer MIPS */
#  define DES_PTR
#  define DES_RISC2
#  define DES_UNROLL
#elif defined(i386) || defined(__i386__)        /* x86 boxes, should be gcc */
#  define DES_PTR
#  define DES_RISC1
#  define DES_UNROLL
```

```
392 #endif /* Systems-specific speed defines */
393 #endif

395 #endif /* DES_DEFAULT_OPTIONS */
396 #endif /* HEADER_DES_LOCL_H */
397 #endif /* ! codereview */
```

```
**********************************************************
    3750 Wed Aug 13 19:51:46 2014
new/usr/src/lib/openssl/include/openssl/opensslv.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#ifndef HEADER_OPENSSLV_H
#define HEADER_OPENSSLV_H

/* Numeric release version identifier:
 * MNNFFPPS: major minor fix patch status
 * The status nibble has one of the values 0 for development, 1 to e for betas
 * 1 to 14, and f for release.  The patch level is exactly that.
 * For example:
 * 0.9.3-dev      0x00903000
 * 0.9.3-beta1    0x00903001
 * 0.9.3-beta2-dev 0x00903002
 * 0.9.3-beta2    0x00903002 (same as ...beta2-dev)
 * 0.9.3          0x0090300f
 * 0.9.3a         0x0090301f
 * 0.9.4          0x0090400f
 * 1.2.3z         0x102031af
 *
 * For continuity reasons (because 0.9.5 is already out, and is coded
 * 0x00905100), between 0.9.5 and 0.9.6 the coding of the patch level
 * part is slightly different, by setting the highest bit.  This means
 * that 0.9.5a looks like this: 0x0090581f.  At 0.9.6, we can start
 * with 0x0090600S...
 *
 * (Prior to 0.9.3-dev a different scheme was used: 0.9.2b is 0x0922.)
 * (Prior to 0.9.5a beta1, a different scheme was used: MMNNFFRBB for
 *  major minor fix final patch/beta)
 */
#define OPENSSL_VERSION_NUMBER  0x1000109fL
#ifdef OPENSSL_FIPS
#define OPENSSL_VERSION_TEXT    "OpenSSL 1.0.1i-fips 6 Aug 2014"
#else
#define OPENSSL_VERSION_TEXT    "OpenSSL 1.0.1i 6 Aug 2014"
#endif
#define OPENSSL_VERSION_PTEXT   " part of " OPENSSL_VERSION_TEXT


/* The macros below are to be used for shared library (.so, .dll, ...)
 * versioning.  That kind of versioning works a bit differently between
 * operating systems.  The most usual scheme is to set a major and a minor
 * number, and have the runtime loader check that the major number is equal
 * to what it was at application link time, while the minor number has to
 * be greater or equal to what it was at application link time.  With this
 * scheme, the version number is usually part of the file name, like this:
 *
 *      libcrypto.so.0.9
 *
 * Some unixen also make a softlink with the major verson number only:
 *
 *      libcrypto.so.0
 *
 * On Tru64 and IRIX 6.x it works a little bit differently.  There, the
 * shared library version is stored in the file, and is actually a series
 * of versions, separated by colons.  The rightmost version present in the
 * library when linking an application is stored in the application to be
 * matched at run time.  When the application is run, a check is done to
 * see if the library version stored in the application matches any of the
 * versions in the version string of the library itself.
 * This version string can be constructed in any way, depending on what
 * kind of matching is desired.  However, to implement the same scheme as
 * the one used in the other unixen, all compatible versions, from lowest
 * to highest, should be part of the string.  Consecutive builds would
```



```
 * give the following versions strings:
 *
 *      3.0
 *      3.0:3.1
 *      3.0:3.1:3.2
 *      4.0
 *      4.0:4.1
 *
 * Notice how version 4 is completely incompatible with version, and
 * therefore give the breach you can see.
 *
 * There may be other schemes as well that I haven't yet discovered.
 *
 * So, here's the way it works here: first of all, the library version
 * number doesn't need at all to match the overall OpenSSL version.
 * However, it's nice and more understandable if it actually does.
 * The current library version is stored in the macro SHLIB_VERSION_NUMBER,
 * which is just a piece of text in the format "M.m.e" (Major, minor, edit).
 * For the sake of Tru64, IRIX, and any other OS that behaves in similar ways,
 * we need to keep a history of version numbers, which is done in the
 * macro SHLIB_VERSION_HISTORY.  The numbers are separated by colons and
 * should only keep the versions that are binary compatible with the current.
 */
#define SHLIB_VERSION_HISTORY ""
#define SHLIB_VERSION_NUMBER "1.0.0"


#endif /* HEADER_OPENSSLV_H */
#endif /* ! codereview */
```

```
**********************************************************
    7395 Wed Aug 13 19:51:46 2014
new/usr/src/lib/openssl/include/openssl/ossl_typ.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_OPENSSL_TYPES_H
#define HEADER_OPENSSL_TYPES_H

#include <openssl/e_os2.h>

#ifdef NO_ASN1_TYPEDEFS
#define ASN1_INTEGER            ASN1_STRING
```



```
#define ASN1_ENUMERATED         ASN1_STRING
#define ASN1_BIT_STRING         ASN1_STRING
#define ASN1_OCTET_STRING       ASN1_STRING
#define ASN1_PRINTABLESTRING    ASN1_STRING
#define ASN1_T61STRING          ASN1_STRING
#define ASN1_IA5STRING          ASN1_STRING
#define ASN1_UTCTIME            ASN1_STRING
#define ASN1_GENERALIZEDTIME    ASN1_STRING
#define ASN1_TIME               ASN1_STRING
#define ASN1_GENERALSTRING      ASN1_STRING
#define ASN1_UNIVERSALSTRING    ASN1_STRING
#define ASN1_BMPSTRING          ASN1_STRING
#define ASN1_VISIBLESTRING      ASN1_STRING
#define ASN1_UTF8STRING         ASN1_STRING
#define ASN1_BOOLEAN            int
#define ASN1_NULL               int
#else
typedef struct asn1_string_st ASN1_INTEGER;
typedef struct asn1_string_st ASN1_ENUMERATED;
typedef struct asn1_string_st ASN1_BIT_STRING;
typedef struct asn1_string_st ASN1_OCTET_STRING;
typedef struct asn1_string_st ASN1_PRINTABLESTRING;
typedef struct asn1_string_st ASN1_T61STRING;
typedef struct asn1_string_st ASN1_IA5STRING;
typedef struct asn1_string_st ASN1_GENERALSTRING;
typedef struct asn1_string_st ASN1_UNIVERSALSTRING;
typedef struct asn1_string_st ASN1_BMPSTRING;
typedef struct asn1_string_st ASN1_UTCTIME;
typedef struct asn1_string_st ASN1_TIME;
typedef struct asn1_string_st ASN1_GENERALIZEDTIME;
typedef struct asn1_string_st ASN1_VISIBLESTRING;
typedef struct asn1_string_st ASN1_UTF8STRING;
typedef struct asn1_string_st ASN1_STRING;
typedef int ASN1_BOOLEAN;
typedef int ASN1_NULL;
#endif

typedef struct ASN1_ITEM_st ASN1_ITEM;
typedef struct asn1_pctx_st ASN1_PCTX;

#ifdef OPENSSL_SYS_WIN32
#undef X509_NAME
#undef X509_EXTENSIONS
#undef X509_CERT_PAIR
#undef PKCS7_ISSUER_AND_SERIAL
#undef OCSP_REQUEST
#undef OCSP_RESPONSE
#endif

#ifdef BIGNUM
#undef BIGNUM
#endif
typedef struct bignum_st BIGNUM;
typedef struct bignum_ctx BN_CTX;
typedef struct bn_blinding_st BN_BLINDING;
typedef struct bn_mont_ctx_st BN_MONT_CTX;
typedef struct bn_recp_ctx_st BN_RECP_CTX;
typedef struct bn_gencb_st BN_GENCB;

typedef struct buf_mem_st BUF_MEM;

typedef struct evp_cipher_st EVP_CIPHER;
typedef struct evp_cipher_ctx_st EVP_CIPHER_CTX;
typedef struct env_md_st EVP_MD;
typedef struct env_md_ctx_st EVP_MD_CTX;
typedef struct evp_pkey_st EVP_PKEY;
```

```
129 typedef struct evp_pkey_asn1_method_st EVP_PKEY_ASN1_METHOD;

131 typedef struct evp_pkey_method_st EVP_PKEY_METHOD;
132 typedef struct evp_pkey_ctx_st EVP_PKEY_CTX;

134 typedef struct dh_st DH;
135 typedef struct dh_method DH_METHOD;

137 typedef struct dsa_st DSA;
138 typedef struct dsa_method DSA_METHOD;

140 typedef struct rsa_st RSA;
141 typedef struct rsa_meth_st RSA_METHOD;

143 typedef struct rand_meth_st RAND_METHOD;

145 typedef struct ecdh_method ECDH_METHOD;
146 typedef struct ecdsa_method ECDSA_METHOD;

148 typedef struct x509_st X509;
149 typedef struct X509_algor_st X509_ALGOR;
150 typedef struct X509_crl_st X509_CRL;
151 typedef struct x509_crl_method_st X509_CRL_METHOD;
152 typedef struct x509_revoked_st X509_REVOKED;
153 typedef struct X509_name_st X509_NAME;
154 typedef struct X509_pubkey_st X509_PUBKEY;
155 typedef struct x509_store_st X509_STORE;
156 typedef struct x509_store_ctx_st X509_STORE_CTX;

158 typedef struct pkcs8_priv_key_info_st PKCS8_PRIV_KEY_INFO;

160 typedef struct v3_ext_ctx X509V3_CTX;
161 typedef struct conf_st CONF;

163 typedef struct store_st STORE;
164 typedef struct store_method_st STORE_METHOD;

166 typedef struct ui_st UI;
167 typedef struct ui_method_st UI_METHOD;

169 typedef struct st_ERR_FNS ERR_FNS;

171 typedef struct engine_st ENGINE;
172 typedef struct ssl_st SSL;
173 typedef struct ssl_ctx_st SSL_CTX;

175 typedef struct X509_POLICY_NODE_st X509_POLICY_NODE;
176 typedef struct X509_POLICY_LEVEL_st X509_POLICY_LEVEL;
177 typedef struct X509_POLICY_TREE_st X509_POLICY_TREE;
178 typedef struct X509_POLICY_CACHE_st X509_POLICY_CACHE;

180 typedef struct AUTHORITY_KEYID_st AUTHORITY_KEYID;
181 typedef struct DIST_POINT_st DIST_POINT;
182 typedef struct ISSUING_DIST_POINT_st ISSUING_DIST_POINT;
183 typedef struct NAME_CONSTRAINTS_st NAME_CONSTRAINTS;

185   /* If placed in pkcs12.h, we end up with a circular depency with pkcs7.h */
186 #define DECLARE_PKCS12_STACK_OF(type) /* Nothing */
187 #define IMPLEMENT_PKCS12_STACK_OF(type) /* Nothing */

189 typedef struct crypto_ex_data_st CRYPTO_EX_DATA;
190 /* Callback types for crypto.h */
191 typedef int CRYPTO_EX_new(void *parent, void *ptr, CRYPTO_EX_DATA *ad,
192                           int idx, long argl, void *argp);
193 typedef void CRYPTO_EX_free(void *parent, void *ptr, CRYPTO_EX_DATA *ad,
```



```
194                             int idx, long argl, void *argp);
195 typedef int CRYPTO_EX_dup(CRYPTO_EX_DATA *to, CRYPTO_EX_DATA *from, void *from_d
196                           int idx, long argl, void *argp);

198 typedef struct ocsp_req_ctx_st OCSP_REQ_CTX;
199 typedef struct ocsp_response_st OCSP_RESPONSE;
200 typedef struct ocsp_responder_id_st OCSP_RESPID;

202 #endif /* def HEADER_OPENSSL_TYPES_H */
203 #endif /* ! codereview */
```

```
**********************************************************
   21971 Wed Aug 13 19:51:46 2014
new/usr/src/lib/openssl/include/openssl/pem.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_PEM_H
#define HEADER_PEM_H
```



```
#include <openssl/e_os2.h>
#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#ifndef OPENSSL_NO_STACK
#include <openssl/stack.h>
#endif
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pem2.h>

#ifdef  __cplusplus
extern "C" {
#endif

#define PEM_BUFSIZE             1024

#define PEM_OBJ_UNDEF           0
#define PEM_OBJ_X509            1
#define PEM_OBJ_X509_REQ        2
#define PEM_OBJ_CRL             3
#define PEM_OBJ_SSL_SESSION     4
#define PEM_OBJ_PRIV_KEY        10
#define PEM_OBJ_PRIV_RSA        11
#define PEM_OBJ_PRIV_DSA        12
#define PEM_OBJ_PRIV_DH         13
#define PEM_OBJ_PUB_RSA         14
#define PEM_OBJ_PUB_DSA         15
#define PEM_OBJ_PUB_DH          16
#define PEM_OBJ_DHPARAMS        17
#define PEM_OBJ_DSAPARAMS       18
#define PEM_OBJ_PRIV_RSA_PUBLIC 19
#define PEM_OBJ_PRIV_ECDSA      20
#define PEM_OBJ_PUB_ECDSA       21
#define PEM_OBJ_ECPARAMETERS    22

#define PEM_ERROR               30
#define PEM_DEK_DES_CBC         40
#define PEM_DEK_IDEA_CBC        45
#define PEM_DEK_DES_EDE         50
#define PEM_DEK_DES_ECB         60
#define PEM_DEK_RSA             70
#define PEM_DEK_RSA_MD2         80
#define PEM_DEK_RSA_MD5         90

#define PEM_MD_MD2              NID_md2
#define PEM_MD_MD5              NID_md5
#define PEM_MD_SHA              NID_sha
#define PEM_MD_MD2_RSA          NID_md2WithRSAEncryption
#define PEM_MD_MD5_RSA          NID_md5WithRSAEncryption
#define PEM_MD_SHA_RSA          NID_sha1WithRSAEncryption

#define PEM_STRING_X509_OLD     "X509 CERTIFICATE"
#define PEM_STRING_X509         "CERTIFICATE"
#define PEM_STRING_X509_PAIR    "CERTIFICATE PAIR"
#define PEM_STRING_X509_TRUSTED "TRUSTED CERTIFICATE"
#define PEM_STRING_X509_REQ_OLD "NEW CERTIFICATE REQUEST"
#define PEM_STRING_X509_REQ     "CERTIFICATE REQUEST"
#define PEM_STRING_X509_CRL     "X509 CRL"
#define PEM_STRING_EVP_PKEY     "ANY PRIVATE KEY"
#define PEM_STRING_PUBLIC       "PUBLIC KEY"
#define PEM_STRING_RSA          "RSA PRIVATE KEY"
#define PEM_STRING_RSA_PUBLIC   "RSA PUBLIC KEY"
#define PEM_STRING_DSA          "DSA PRIVATE KEY"
#define PEM_STRING_DSA_PUBLIC   "DSA PUBLIC KEY"
#define PEM_STRING_PKCS7        "PKCS7"
```

```
#define PEM_STRING_PKCS7_SIGNED "PKCS #7 SIGNED DATA"
#define PEM_STRING_PKCS8        "ENCRYPTED PRIVATE KEY"
#define PEM_STRING_PKCS8INF     "PRIVATE KEY"
#define PEM_STRING_DHPARAMS     "DH PARAMETERS"
#define PEM_STRING_SSL_SESSION  "SSL SESSION PARAMETERS"
#define PEM_STRING_DSAPARAMS    "DSA PARAMETERS"
#define PEM_STRING_ECDSA_PUBLIC "ECDSA PUBLIC KEY"
#define PEM_STRING_ECPARAMETERS "EC PARAMETERS"
#define PEM_STRING_ECPRIVATEKEY "EC PRIVATE KEY"
#define PEM_STRING_PARAMETERS   "PARAMETERS"
#define PEM_STRING_CMS          "CMS"

  /* Note that this structure is initialised by PEM_SealInit and cleaned up
     by PEM_SealFinal (at least for now) */
typedef struct PEM_Encode_Seal_st
        {
        EVP_ENCODE_CTX encode;
        EVP_MD_CTX md;
        EVP_CIPHER_CTX cipher;
        } PEM_ENCODE_SEAL_CTX;

/* enc_type is one off */
#define PEM_TYPE_ENCRYPTED      10
#define PEM_TYPE_MIC_ONLY       20
#define PEM_TYPE_MIC_CLEAR      30
#define PEM_TYPE_CLEAR          40

typedef struct pem_recip_st
        {
        char *name;
        X509_NAME *dn;

        int cipher;
        int key_enc;
        /*      char iv[8]; unused and wrong size */
        } PEM_USER;

typedef struct pem_ctx_st
        {
        int type;               /* what type of object */

        struct  {
                int version;
                int mode;
                } proc_type;

        char *domain;

        struct  {
                int cipher;
        /* unused, and wrong size
           unsigned char iv[8]; */
                } DEK_info;

        PEM_USER *originator;

        int num_recipient;
        PEM_USER **recipient;

        /* XXX(ben): don#t think this is used!
                STACK *x509_chain;      / * certificate chain */
        EVP_MD *md;             /* signature type */

        int md_enc;             /* is the md encrypted or not? */
        int md_len;             /* length of md_data */
        char *md_data;          /* message digest, could be pkey encrypted */
```



```
        EVP_CIPHER *dec;        /* date encryption cipher */
        int key_len;            /* key length */
        unsigned char *key;     /* key */
        /* unused, and wrong size
           unsigned char iv[8]; */


        int  data_enc;          /* is the data encrypted */
        int data_len;
        unsigned char *data;
        } PEM_CTX;

/* These macros make the PEM_read/PEM_write functions easier to maintain and
 * write. Now they are all implemented with either:
 * IMPLEMENT_PEM_rw(...) or IMPLEMENT_PEM_rw_cb(...)
 */

#ifdef OPENSSL_NO_FP_API

#define IMPLEMENT_PEM_read_fp(name, type, str, asn1) /**/
#define IMPLEMENT_PEM_write_fp(name, type, str, asn1) /**/
#define IMPLEMENT_PEM_write_fp_const(name, type, str, asn1) /**/
#define IMPLEMENT_PEM_write_cb_fp(name, type, str, asn1) /**/
#define IMPLEMENT_PEM_write_cb_fp_const(name, type, str, asn1) /**/

#else

#define IMPLEMENT_PEM_read_fp(name, type, str, asn1) \
type *PEM_read_##name(FILE *fp, type **x, pem_password_cb *cb, void *u)\
{ \
return PEM_ASN1_read((d2i_of_void *)d2i_##asn1, str,fp,(void **)x,cb,u); \
}

#define IMPLEMENT_PEM_write_fp(name, type, str, asn1) \
int PEM_write_##name(FILE *fp, type *x) \
{ \
return PEM_ASN1_write((i2d_of_void *)i2d_##asn1,str,fp,x,NULL,NULL,0,NULL,NULL);
}

#define IMPLEMENT_PEM_write_fp_const(name, type, str, asn1) \
int PEM_write_##name(FILE *fp, const type *x) \
{ \
return PEM_ASN1_write((i2d_of_void *)i2d_##asn1,str,fp,(void *)x,NULL,NULL,0,NUL
}

#define IMPLEMENT_PEM_write_cb_fp(name, type, str, asn1) \
int PEM_write_##name(FILE *fp, type *x, const EVP_CIPHER *enc, \
             unsigned char *kstr, int klen, pem_password_cb *cb, \
                  void *u) \
        { \
        return PEM_ASN1_write((i2d_of_void *)i2d_##asn1,str,fp,x,enc,kstr,klen,c
        }

#define IMPLEMENT_PEM_write_cb_fp_const(name, type, str, asn1) \
int PEM_write_##name(FILE *fp, type *x, const EVP_CIPHER *enc, \
             unsigned char *kstr, int klen, pem_password_cb *cb, \
                  void *u) \
        { \
        return PEM_ASN1_write((i2d_of_void *)i2d_##asn1,str,fp,x,enc,kstr,klen,c
        }

#endif

#define IMPLEMENT_PEM_read_bio(name, type, str, asn1) \
type *PEM_read_bio_##name(BIO *bp, type **x, pem_password_cb *cb, void *u)\
```

```
260 { \
261 return PEM_ASN1_read_bio((d2i_of_void *)d2i_##asn1, str,bp,(void **)x,cb,u); \
262 }

264 #define IMPLEMENT_PEM_write_bio(name, type, str, asn1) \
265 int PEM_write_bio_##name(BIO *bp, type *x) \
266 { \
267 return PEM_ASN1_write_bio((i2d_of_void *)i2d_##asn1,str,bp,x,NULL,NULL,0,NULL,NU
268 }

270 #define IMPLEMENT_PEM_write_bio_const(name, type, str, asn1) \
271 int PEM_write_bio_##name(BIO *bp, const type *x) \
272 { \
273 return PEM_ASN1_write_bio((i2d_of_void *)i2d_##asn1,str,bp,(void *)x,NULL,NULL,0
274 }

276 #define IMPLEMENT_PEM_write_cb_bio(name, type, str, asn1) \
277 int PEM_write_bio_##name(BIO *bp, type *x, const EVP_CIPHER *enc, \
278              unsigned char *kstr, int klen, pem_password_cb *cb, void *u) \
279         { \
280         return PEM_ASN1_write_bio((i2d_of_void *)i2d_##asn1,str,bp,x,enc,kstr,kl
281         }

283 #define IMPLEMENT_PEM_write_cb_bio_const(name, type, str, asn1) \
284 int PEM_write_bio_##name(BIO *bp, type *x, const EVP_CIPHER *enc, \
285              unsigned char *kstr, int klen, pem_password_cb *cb, void *u) \
286         { \
287         return PEM_ASN1_write_bio((i2d_of_void *)i2d_##asn1,str,bp,(void *)x,enc
288         }

290 #define IMPLEMENT_PEM_write(name, type, str, asn1) \
291         IMPLEMENT_PEM_write_bio(name, type, str, asn1) \
292         IMPLEMENT_PEM_write_fp(name, type, str, asn1)

294 #define IMPLEMENT_PEM_write_const(name, type, str, asn1) \
295         IMPLEMENT_PEM_write_bio_const(name, type, str, asn1) \
296         IMPLEMENT_PEM_write_fp_const(name, type, str, asn1)

298 #define IMPLEMENT_PEM_write_cb(name, type, str, asn1) \
299         IMPLEMENT_PEM_write_cb_bio(name, type, str, asn1) \
300         IMPLEMENT_PEM_write_cb_fp(name, type, str, asn1)

302 #define IMPLEMENT_PEM_write_cb_const(name, type, str, asn1) \
303         IMPLEMENT_PEM_write_cb_bio_const(name, type, str, asn1) \
304         IMPLEMENT_PEM_write_cb_fp_const(name, type, str, asn1)

306 #define IMPLEMENT_PEM_read(name, type, str, asn1) \
307         IMPLEMENT_PEM_read_bio(name, type, str, asn1) \
308         IMPLEMENT_PEM_read_fp(name, type, str, asn1)

310 #define IMPLEMENT_PEM_rw(name, type, str, asn1) \
311         IMPLEMENT_PEM_read(name, type, str, asn1) \
312         IMPLEMENT_PEM_write(name, type, str, asn1)

314 #define IMPLEMENT_PEM_rw_const(name, type, str, asn1) \
315         IMPLEMENT_PEM_read(name, type, str, asn1) \
316         IMPLEMENT_PEM_write_const(name, type, str, asn1)

318 #define IMPLEMENT_PEM_rw_cb(name, type, str, asn1) \
319         IMPLEMENT_PEM_read(name, type, str, asn1) \
320         IMPLEMENT_PEM_write_cb(name, type, str, asn1)

322 /* These are the same except they are for the declarations */

324 #if defined(OPENSSL_NO_FP_API)
```



```
326 #define DECLARE_PEM_read_fp(name, type) /**/
327 #define DECLARE_PEM_write_fp(name, type) /**/
328 #define DECLARE_PEM_write_cb_fp(name, type) /**/

330 #else

332 #define DECLARE_PEM_read_fp(name, type) \
333         type *PEM_read_##name(FILE *fp, type **x, pem_password_cb *cb, void *u);

335 #define DECLARE_PEM_write_fp(name, type) \
336         int PEM_write_##name(FILE *fp, type *x);

338 #define DECLARE_PEM_write_fp_const(name, type) \
339         int PEM_write_##name(FILE *fp, const type *x);

341 #define DECLARE_PEM_write_cb_fp(name, type) \
342         int PEM_write_##name(FILE *fp, type *x, const EVP_CIPHER *enc, \
343              unsigned char *kstr, int klen, pem_password_cb *cb, void *u);

345 #endif

347 #ifndef OPENSSL_NO_BIO
348 #define DECLARE_PEM_read_bio(name, type) \
349         type *PEM_read_bio_##name(BIO *bp, type **x, pem_password_cb *cb, void *

351 #define DECLARE_PEM_write_bio(name, type) \
352         int PEM_write_bio_##name(BIO *bp, type *x);

354 #define DECLARE_PEM_write_bio_const(name, type) \
355         int PEM_write_bio_##name(BIO *bp, const type *x);

357 #define DECLARE_PEM_write_cb_bio(name, type) \
358         int PEM_write_bio_##name(BIO *bp, type *x, const EVP_CIPHER *enc, \
359              unsigned char *kstr, int klen, pem_password_cb *cb, void *u);

361 #else

363 #define DECLARE_PEM_read_bio(name, type) /**/
364 #define DECLARE_PEM_write_bio(name, type) /**/
365 #define DECLARE_PEM_write_bio_const(name, type) /**/
366 #define DECLARE_PEM_write_cb_bio(name, type) /**/

368 #endif

370 #define DECLARE_PEM_write(name, type) \
371         DECLARE_PEM_write_bio(name, type) \
372         DECLARE_PEM_write_fp(name, type)

374 #define DECLARE_PEM_write_const(name, type) \
375         DECLARE_PEM_write_bio_const(name, type) \
376         DECLARE_PEM_write_fp_const(name, type)

378 #define DECLARE_PEM_write_cb(name, type) \
379         DECLARE_PEM_write_cb_bio(name, type) \
380         DECLARE_PEM_write_cb_fp(name, type)

382 #define DECLARE_PEM_read(name, type) \
383         DECLARE_PEM_read_bio(name, type) \
384         DECLARE_PEM_read_fp(name, type)

386 #define DECLARE_PEM_rw(name, type) \
387         DECLARE_PEM_read(name, type) \
388         DECLARE_PEM_write(name, type)

390 #define DECLARE_PEM_rw_const(name, type) \
391         DECLARE_PEM_read(name, type) \
```

```
392         DECLARE_PEM_write_const(name, type)

394 #define DECLARE_PEM_rw_cb(name, type) \
395         DECLARE_PEM_read(name, type) \
396         DECLARE_PEM_write_cb(name, type)

398 #if 1
399 /* "userdata": new with OpenSSL 0.9.4 */
400 typedef int pem_password_cb(char *buf, int size, int rwflag, void *userdata);
401 #else
402 /* OpenSSL 0.9.3, 0.9.3a */
403 typedef int pem_password_cb(char *buf, int size, int rwflag);
404 #endif

406 int     PEM_get_EVP_CIPHER_INFO(char *header, EVP_CIPHER_INFO *cipher);
407 int     PEM_do_header (EVP_CIPHER_INFO *cipher, unsigned char *data,long *len,
408         pem_password_cb *callback,void *u);

410 #ifndef OPENSSL_NO_BIO
411 int     PEM_read_bio(BIO *bp, char **name, char **header,
412                 unsigned char **data,long *len);
413 int     PEM_write_bio(BIO *bp,const char *name,char *hdr,unsigned char *data,
414                 long len);
415 int PEM_bytes_read_bio(unsigned char **pdata, long *plen, char **pnm, const char
416             pem_password_cb *cb, void *u);
417 void *  PEM_ASN1_read_bio(d2i_of_void *d2i, const char *name, BIO *bp,
418                           void **x, pem_password_cb *cb, void *u);
419 int     PEM_ASN1_write_bio(i2d_of_void *i2d,const char *name,BIO *bp, void *x,
420                            const EVP_CIPHER *enc,unsigned char *kstr,int klen,
421                            pem_password_cb *cb, void *u);

423 STACK_OF(X509_INFO) *   PEM_X509_INFO_read_bio(BIO *bp, STACK_OF(X509_INFO) *sk,
424 int     PEM_X509_INFO_write_bio(BIO *bp,X509_INFO *xi, EVP_CIPHER *enc,
425                 unsigned char *kstr, int klen, pem_password_cb *cd, void *u);
426 #endif

428 int     PEM_read(FILE *fp, char **name, char **header,
429                 unsigned char **data,long *len);
430 int     PEM_write(FILE *fp,char *name,char *hdr,unsigned char *data,long len);
431 void *  PEM_ASN1_read(d2i_of_void *d2i, const char *name, FILE *fp, void **x,
432                       pem_password_cb *cb, void *u);
433 int     PEM_ASN1_write(i2d_of_void *i2d,const char *name,FILE *fp,
434                        void *x,const EVP_CIPHER *enc,unsigned char *kstr,
435                        int klen,pem_password_cb *callback, void *u);
436 STACK_OF(X509_INFO) *   PEM_X509_INFO_read(FILE *fp, STACK_OF(X509_INFO) *sk,
437         pem_password_cb *cb, void *u);

439 int     PEM_SealInit(PEM_ENCODE_SEAL_CTX *ctx, EVP_CIPHER *type,
440                 EVP_MD *md_type, unsigned char **ek, int *ekl,
441                 unsigned char *iv, EVP_PKEY **pubk, int npubk);
442 void    PEM_SealUpdate(PEM_ENCODE_SEAL_CTX *ctx, unsigned char *out, int *outl,
443                 unsigned char *in, int inl);
444 int     PEM_SealFinal(PEM_ENCODE_SEAL_CTX *ctx, unsigned char *sig,int *sigl,
445                 unsigned char *out, int *outl, EVP_PKEY *priv);

447 void    PEM_SignInit(EVP_MD_CTX *ctx, EVP_MD *type);
448 void    PEM_SignUpdate(EVP_MD_CTX *ctx,unsigned char *d,unsigned int cnt);
449 int     PEM_SignFinal(EVP_MD_CTX *ctx, unsigned char *sigret,
450                 unsigned int *siglen, EVP_PKEY *pkey);

452 int     PEM_def_callback(char *buf, int num, int w, void *key);
453 void    PEM_proc_type(char *buf, int type);
454 void    PEM_dek_info(char *buf, const char *type, int len, char *str);


457 #include <openssl/symhacks.h>
```



```
459 DECLARE_PEM_rw(X509, X509)

461 DECLARE_PEM_rw(X509_AUX, X509)

463 DECLARE_PEM_rw(X509_CERT_PAIR, X509_CERT_PAIR)

465 DECLARE_PEM_rw(X509_REQ, X509_REQ)
466 DECLARE_PEM_write(X509_REQ_NEW, X509_REQ)

468 DECLARE_PEM_rw(X509_CRL, X509_CRL)

470 DECLARE_PEM_rw(PKCS7, PKCS7)

472 DECLARE_PEM_rw(NETSCAPE_CERT_SEQUENCE, NETSCAPE_CERT_SEQUENCE)

474 DECLARE_PEM_rw(PKCS8, X509_SIG)

476 DECLARE_PEM_rw(PKCS8_PRIV_KEY_INFO, PKCS8_PRIV_KEY_INFO)

478 #ifndef OPENSSL_NO_RSA

480 DECLARE_PEM_rw_cb(RSAPrivateKey, RSA)

482 DECLARE_PEM_rw_const(RSAPublicKey, RSA)
483 DECLARE_PEM_rw(RSA_PUBKEY, RSA)

485 #endif

487 #ifndef OPENSSL_NO_DSA

489 DECLARE_PEM_rw_cb(DSAPrivateKey, DSA)

491 DECLARE_PEM_rw(DSA_PUBKEY, DSA)

493 DECLARE_PEM_rw_const(DSAparams, DSA)

495 #endif

497 #ifndef OPENSSL_NO_EC
498 DECLARE_PEM_rw_const(ECPKParameters, EC_GROUP)
499 DECLARE_PEM_rw_cb(ECPrivateKey, EC_KEY)
500 DECLARE_PEM_rw(EC_PUBKEY, EC_KEY)
501 #endif

503 #ifndef OPENSSL_NO_DH

505 DECLARE_PEM_rw_const(DHparams, DH)

507 #endif

509 DECLARE_PEM_rw_cb(PrivateKey, EVP_PKEY)

511 DECLARE_PEM_rw(PUBKEY, EVP_PKEY)

513 int PEM_write_bio_PKCS8PrivateKey_nid(BIO *bp, EVP_PKEY *x, int nid,
514                                       char *kstr, int klen,
515                                       pem_password_cb *cb, void *u);
516 int PEM_write_bio_PKCS8PrivateKey(BIO *, EVP_PKEY *, const EVP_CIPHER *,
517                                       char *, int, pem_password_cb *, void *);
518 int i2d_PKCS8PrivateKey_bio(BIO *bp, EVP_PKEY *x, const EVP_CIPHER *enc,
519                                       char *kstr, int klen,
520                                       pem_password_cb *cb, void *u);
521 int i2d_PKCS8PrivateKey_nid_bio(BIO *bp, EVP_PKEY *x, int nid,
522                                       char *kstr, int klen,
523                                       pem_password_cb *cb, void *u);
```

```
EVP_PKEY *d2i_PKCS8PrivateKey_bio(BIO *bp, EVP_PKEY **x, pem_password_cb *cb, vo

int i2d_PKCS8PrivateKey_fp(FILE *fp, EVP_PKEY *x, const EVP_CIPHER *enc,
                           char *kstr, int klen,
                           pem_password_cb *cb, void *u);
int i2d_PKCS8PrivateKey_nid_fp(FILE *fp, EVP_PKEY *x, int nid,
                           char *kstr, int klen,
                           pem_password_cb *cb, void *u);
int PEM_write_PKCS8PrivateKey_nid(FILE *fp, EVP_PKEY *x, int nid,
                           char *kstr, int klen,
                           pem_password_cb *cb, void *u);

EVP_PKEY *d2i_PKCS8PrivateKey_fp(FILE *fp, EVP_PKEY **x, pem_password_cb *cb, vo

int PEM_write_PKCS8PrivateKey(FILE *fp,EVP_PKEY *x,const EVP_CIPHER *enc,
                          char *kstr,int klen, pem_password_cb *cd, void *u)

EVP_PKEY *PEM_read_bio_Parameters(BIO *bp, EVP_PKEY **x);
int PEM_write_bio_Parameters(BIO *bp, EVP_PKEY *x);


EVP_PKEY *b2i_PrivateKey(const unsigned char **in, long length);
EVP_PKEY *b2i_PublicKey(const unsigned char **in, long length);
EVP_PKEY *b2i_PrivateKey_bio(BIO *in);
EVP_PKEY *b2i_PublicKey_bio(BIO *in);
int i2b_PrivateKey_bio(BIO *out, EVP_PKEY *pk);
int i2b_PublicKey_bio(BIO *out, EVP_PKEY *pk);
#ifndef OPENSSL_NO_RC4
EVP_PKEY *b2i_PVK_bio(BIO *in, pem_password_cb *cb, void *u);
int i2b_PVK_bio(BIO *out, EVP_PKEY *pk, int enclevel,
                pem_password_cb *cb, void *u);
#endif


/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_PEM_strings(void);

/* Error codes for the PEM functions. */

/* Function codes. */
#define PEM_F_B2I_DSS                                   127
#define PEM_F_B2I_PVK_BIO                               128
#define PEM_F_B2I_RSA                                   129
#define PEM_F_CHECK_BITLEN_DSA                          130
#define PEM_F_CHECK_BITLEN_RSA                          131
#define PEM_F_D2I_PKCS8PRIVATEKEY_BIO                   120
#define PEM_F_D2I_PKCS8PRIVATEKEY_FP                    121
#define PEM_F_DO_B2I                                    132
#define PEM_F_DO_B2I_BIO                                133
#define PEM_F_DO_BLOB_HEADER                            134
#define PEM_F_DO_PK8PKEY                                126
#define PEM_F_DO_PK8PKEY_FP                             125
#define PEM_F_DO_PVK_BODY                               135
#define PEM_F_DO_PVK_HEADER                             136
#define PEM_F_I2B_PVK                                   137
#define PEM_F_I2B_PVK_BIO                               138
#define PEM_F_LOAD_IV                                   101
#define PEM_F_PEM_ASN1_READ                             102
#define PEM_F_PEM_ASN1_READ_BIO                         103
#define PEM_F_PEM_ASN1_WRITE                            104
#define PEM_F_PEM_ASN1_WRITE_BIO                        105
#define PEM_F_PEM_DEF_CALLBACK                          100
#define PEM_F_PEM_DO_HEADER                             106
```



```
#define PEM_F_PEM_F_PEM_WRITE_PKCS8PRIVATEKEY           118
#define PEM_F_PEM_GET_EVP_CIPHER_INFO                   107
#define PEM_F_PEM_PK8PKEY                               119
#define PEM_F_PEM_READ                                  108
#define PEM_F_PEM_READ_BIO                              109
#define PEM_F_PEM_READ_BIO_PARAMETERS                   140
#define PEM_F_PEM_READ_BIO_PRIVATEKEY                   123
#define PEM_F_PEM_READ_PRIVATEKEY                       124
#define PEM_F_PEM_SEALFINAL                             110
#define PEM_F_PEM_SEALINIT                              111
#define PEM_F_PEM_SIGNFINAL                             112
#define PEM_F_PEM_WRITE                                 113
#define PEM_F_PEM_WRITE_BIO                             114
#define PEM_F_PEM_WRITE_PRIVATEKEY                      139
#define PEM_F_PEM_X509_INFO_READ                        115
#define PEM_F_PEM_X509_INFO_READ_BIO                    116
#define PEM_F_PEM_X509_INFO_WRITE_BIO                   117

/* Reason codes. */
#define PEM_R_BAD_BASE64_DECODE                         100
#define PEM_R_BAD_DECRYPT                               101
#define PEM_R_BAD_END_LINE                              102
#define PEM_R_BAD_IV_CHARS                              103
#define PEM_R_BAD_MAGIC_NUMBER                          116
#define PEM_R_BAD_PASSWORD_READ                         104
#define PEM_R_BAD_VERSION_NUMBER                        117
#define PEM_R_BIO_WRITE_FAILURE                         118
#define PEM_R_CIPHER_IS_NULL                            127
#define PEM_R_ERROR_CONVERTING_PRIVATE_KEY              115
#define PEM_R_EXPECTING_PRIVATE_KEY_BLOB                119
#define PEM_R_EXPECTING_PUBLIC_KEY_BLOB                 120
#define PEM_R_INCONSISTENT_HEADER                       121
#define PEM_R_KEYBLOB_HEADER_PARSE_ERROR                122
#define PEM_R_KEYBLOB_TOO_SHORT                         123
#define PEM_R_NOT_DEK_INFO                              105
#define PEM_R_NOT_ENCRYPTED                             106
#define PEM_R_NOT_PROC_TYPE                             107
#define PEM_R_NO_START_LINE                             108
#define PEM_R_PROBLEMS_GETTING_PASSWORD                 109
#define PEM_R_PUBLIC_KEY_NO_RSA                         110
#define PEM_R_PVK_DATA_TOO_SHORT                        124
#define PEM_R_PVK_TOO_SHORT                             125
#define PEM_R_READ_KEY                                  111
#define PEM_R_SHORT_HEADER                              112
#define PEM_R_UNSUPPORTED_CIPHER                        113
#define PEM_R_UNSUPPORTED_ENCRYPTION                    114
#define PEM_R_UNSUPPORTED_KEY_COMPONENTS                126

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

****
2862 Wed Aug 13 19:51:46 2014
new/usr/src/lib/openssl/include/openssl/pem2.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/*
 * This header only exists to break a circular dependency between pem and err
 * Ben 30 Jan 1999.
 */

#ifdef __cplusplus
extern "C" {
```



```
#endif

#ifndef HEADER_PEM_H
void ERR_load_PEM_strings(void);
#endif

#ifdef __cplusplus
}
#endif
#endif /* ! codereview */
```

```
**********************************************************
   12651 Wed Aug 13 19:51:47 2014
new/usr/src/lib/openssl/include/openssl/pkcs12.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pkcs12.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_PKCS12_H
#define HEADER_PKCS12_H
```



```
#include <openssl/bio.h>
#include <openssl/x509.h>

#ifdef __cplusplus
extern "C" {
#endif

#define PKCS12_KEY_ID   1
#define PKCS12_IV_ID    2
#define PKCS12_MAC_ID   3

/* Default iteration count */
#ifndef PKCS12_DEFAULT_ITER
#define PKCS12_DEFAULT_ITER     PKCS5_DEFAULT_ITER
#endif

#define PKCS12_MAC_KEY_LENGTH 20

#define PKCS12_SALT_LEN 8

/* Uncomment out next line for unicode password and names, otherwise ASCII */

/*#define PBE_UNICODE*/

#ifdef PBE_UNICODE
#define PKCS12_key_gen PKCS12_key_gen_uni
#define PKCS12_add_friendlyname PKCS12_add_friendlyname_uni
#else
#define PKCS12_key_gen PKCS12_key_gen_asc
#define PKCS12_add_friendlyname PKCS12_add_friendlyname_asc
#endif

/* MS key usage constants */

#define KEY_EX  0x10
#define KEY_SIG 0x80

typedef struct {
X509_SIG *dinfo;
ASN1_OCTET_STRING *salt;
ASN1_INTEGER *iter;     /* defaults to 1 */
} PKCS12_MAC_DATA;

typedef struct {
ASN1_INTEGER *version;
PKCS12_MAC_DATA *mac;
PKCS7 *authsafes;
} PKCS12;

typedef struct {
ASN1_OBJECT *type;
union {
        struct pkcs12_bag_st *bag; /* secret, crl and certbag */
        struct pkcs8_priv_key_info_st   *keybag; /* keybag */
        X509_SIG *shkeybag; /* shrouded key bag */
        STACK_OF(PKCS12_SAFEBAG) *safes;
        ASN1_TYPE *other;
}value;
STACK_OF(X509_ATTRIBUTE) *attrib;
} PKCS12_SAFEBAG;

DECLARE_STACK_OF(PKCS12_SAFEBAG)
DECLARE_ASN1_SET_OF(PKCS12_SAFEBAG)
DECLARE_PKCS12_STACK_OF(PKCS12_SAFEBAG)

typedef struct pkcs12_bag_st {
```

```
ASN1_OBJECT *type;
union {
        ASN1_OCTET_STRING *x509cert;
        ASN1_OCTET_STRING *x509crl;
        ASN1_OCTET_STRING *octet;
        ASN1_IA5STRING *sdsicert;
        ASN1_TYPE *other; /* Secret or other bag */
}value;
} PKCS12_BAGS;

#define PKCS12_ERROR    0
#define PKCS12_OK       1

/* Compatibility macros */

#define M_PKCS12_x5092certbag PKCS12_x5092certbag
#define M_PKCS12_x509crl2certbag PKCS12_x509crl2certbag

#define M_PKCS12_certbag2x509 PKCS12_certbag2x509
#define M_PKCS12_certbag2x509crl PKCS12_certbag2x509crl

#define M_PKCS12_unpack_p7data PKCS12_unpack_p7data
#define M_PKCS12_pack_authsafes PKCS12_pack_authsafes
#define M_PKCS12_unpack_authsafes PKCS12_unpack_authsafes
#define M_PKCS12_unpack_p7encdata PKCS12_unpack_p7encdata

#define M_PKCS12_decrypt_skey PKCS12_decrypt_skey
#define M_PKCS8_decrypt PKCS8_decrypt

#define M_PKCS12_bag_type(bg) OBJ_obj2nid((bg)->type)
#define M_PKCS12_cert_bag_type(bg) OBJ_obj2nid((bg)->value.bag->type)
#define M_PKCS12_crl_bag_type M_PKCS12_cert_bag_type

#define PKCS12_get_attr(bag, attr_nid) \
                         PKCS12_get_attr_gen(bag->attrib, attr_nid)

#define PKCS8_get_attr(p8, attr_nid) \
                 PKCS12_get_attr_gen(p8->attributes, attr_nid)

#define PKCS12_mac_present(p12) ((p12)->mac ? 1 : 0)


PKCS12_SAFEBAG *PKCS12_x5092certbag(X509 *x509);
PKCS12_SAFEBAG *PKCS12_x509crl2certbag(X509_CRL *crl);
X509 *PKCS12_certbag2x509(PKCS12_SAFEBAG *bag);
X509_CRL *PKCS12_certbag2x509crl(PKCS12_SAFEBAG *bag);

PKCS12_SAFEBAG *PKCS12_item_pack_safebag(void *obj, const ASN1_ITEM *it, int nid
             int nid2);
PKCS12_SAFEBAG *PKCS12_MAKE_KEYBAG(PKCS8_PRIV_KEY_INFO *p8);
PKCS8_PRIV_KEY_INFO *PKCS8_decrypt(X509_SIG *p8, const char *pass, int passlen);
PKCS8_PRIV_KEY_INFO *PKCS12_decrypt_skey(PKCS12_SAFEBAG *bag, const char *pass,
                                                        int passlen);
X509_SIG *PKCS8_encrypt(int pbe_nid, const EVP_CIPHER *cipher,
                        const char *pass, int passlen,
                        unsigned char *salt, int saltlen, int iter,
                        PKCS8_PRIV_KEY_INFO *p8);
PKCS12_SAFEBAG *PKCS12_MAKE_SHKEYBAG(int pbe_nid, const char *pass,
                                     int passlen, unsigned char *salt,
                                     int saltlen, int iter,
                                     PKCS8_PRIV_KEY_INFO *p8);
PKCS7 *PKCS12_pack_p7data(STACK_OF(PKCS12_SAFEBAG) *sk);
STACK_OF(PKCS12_SAFEBAG) *PKCS12_unpack_p7data(PKCS7 *p7);
PKCS7 *PKCS12_pack_p7encdata(int pbe_nid, const char *pass, int passlen,
                             unsigned char *salt, int saltlen, int iter,
                             STACK_OF(PKCS12_SAFEBAG) *bags);
```



```
STACK_OF(PKCS12_SAFEBAG) *PKCS12_unpack_p7encdata(PKCS7 *p7, const char *pass, i

int PKCS12_pack_authsafes(PKCS12 *p12, STACK_OF(PKCS7) *safes);
STACK_OF(PKCS7) *PKCS12_unpack_authsafes(PKCS12 *p12);

int PKCS12_add_localkeyid(PKCS12_SAFEBAG *bag, unsigned char *name, int namelen)
int PKCS12_add_friendlyname_asc(PKCS12_SAFEBAG *bag, const char *name,
                                int namelen);
int PKCS12_add_CSPName_asc(PKCS12_SAFEBAG *bag, const char *name,
                           int namelen);
int PKCS12_add_friendlyname_uni(PKCS12_SAFEBAG *bag, const unsigned char *name,
                                int namelen);
int PKCS8_add_keyusage(PKCS8_PRIV_KEY_INFO *p8, int usage);
ASN1_TYPE *PKCS12_get_attr_gen(STACK_OF(X509_ATTRIBUTE) *attrs, int attr_nid);
char *PKCS12_get_friendlyname(PKCS12_SAFEBAG *bag);
unsigned char *PKCS12_pbe_crypt(X509_ALGOR *algor, const char *pass,
                                int passlen, unsigned char *in, int inlen,
                                unsigned char **data, int *datalen, int en_de);
void * PKCS12_item_decrypt_d2i(X509_ALGOR *algor, const ASN1_ITEM *it,
            const char *pass, int passlen, ASN1_OCTET_STRING *oct, int zbuf);
ASN1_OCTET_STRING *PKCS12_item_i2d_encrypt(X509_ALGOR *algor, const ASN1_ITEM *i
                                           const char *pass, int passlen,
                                           void *obj, int zbuf);
PKCS12 *PKCS12_init(int mode);
int PKCS12_key_gen_asc(const char *pass, int passlen, unsigned char *salt,
                       int saltlen, int id, int iter, int n,
                       unsigned char *out, const EVP_MD *md_type);
int PKCS12_key_gen_uni(unsigned char *pass, int passlen, unsigned char *salt, in
int PKCS12_PBE_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
                        ASN1_TYPE *param, const EVP_CIPHER *cipher, const EVP_M
                        int en_de);
int PKCS12_gen_mac(PKCS12 *p12, const char *pass, int passlen,
                   unsigned char *mac, unsigned int *maclen);
int PKCS12_verify_mac(PKCS12 *p12, const char *pass, int passlen);
int PKCS12_set_mac(PKCS12 *p12, const char *pass, int passlen,
                   unsigned char *salt, int saltlen, int iter,
                   const EVP_MD *md_type);
int PKCS12_setup_mac(PKCS12 *p12, int iter, unsigned char *salt,
                                  int saltlen, const EVP_MD *md_type);
unsigned char *OPENSSL_asc2uni(const char *asc, int asclen, unsigned char **uni,
char *OPENSSL_uni2asc(unsigned char *uni, int unilen);

DECLARE_ASN1_FUNCTIONS(PKCS12)
DECLARE_ASN1_FUNCTIONS(PKCS12_MAC_DATA)
DECLARE_ASN1_FUNCTIONS(PKCS12_SAFEBAG)
DECLARE_ASN1_FUNCTIONS(PKCS12_BAGS)

DECLARE_ASN1_ITEM(PKCS12_SAFEBAGS)
DECLARE_ASN1_ITEM(PKCS12_AUTHSAFES)

void PKCS12_PBE_add(void);
int PKCS12_parse(PKCS12 *p12, const char *pass, EVP_PKEY **pkey, X509 **cert,
                 STACK_OF(X509) **ca);
PKCS12 *PKCS12_create(char *pass, char *name, EVP_PKEY *pkey, X509 *cert,
                      STACK_OF(X509) *ca, int nid_key, int nid_cert, int iter
                                          int mac_iter, int keytype);

PKCS12_SAFEBAG *PKCS12_add_cert(STACK_OF(PKCS12_SAFEBAG) **pbags, X509 *cert);
PKCS12_SAFEBAG *PKCS12_add_key(STACK_OF(PKCS12_SAFEBAG) **pbags, EVP_PKEY *key,
                                           int key_usage, int iter,
                                           int key_nid, char *pass);
int PKCS12_add_safe(STACK_OF(PKCS7) **psafes, STACK_OF(PKCS12_SAFEBAG) *bags,
                                 int safe_nid, int iter, char *pass);
PKCS12 *PKCS12_add_safes(STACK_OF(PKCS7) *safes, int p7_nid);

int i2d_PKCS12_bio(BIO *bp, PKCS12 *p12);
```

```
260 int i2d_PKCS12_fp(FILE *fp, PKCS12 *p12);
261 PKCS12 *d2i_PKCS12_bio(BIO *bp, PKCS12 **p12);
262 PKCS12 *d2i_PKCS12_fp(FILE *fp, PKCS12 **p12);
263 int PKCS12_newpass(PKCS12 *p12, char *oldpass, char *newpass);

265 /* BEGIN ERROR CODES */
266 /* The following lines are auto generated by the script mkerr.pl. Any changes
267  * made after this point may be overwritten when the script is next run.
268  */
269 void ERR_load_PKCS12_strings(void);

271 /* Error codes for the PKCS12 functions. */

273 /* Function codes. */
274 #define PKCS12_F_PARSE_BAG                              129
275 #define PKCS12_F_PARSE_BAGS                             103
276 #define PKCS12_F_PKCS12_ADD_FRIENDLYNAME                100
277 #define PKCS12_F_PKCS12_ADD_FRIENDLYNAME_ASC            127
278 #define PKCS12_F_PKCS12_ADD_FRIENDLYNAME_UNI            102
279 #define PKCS12_F_PKCS12_ADD_LOCALKEYID                  104
280 #define PKCS12_F_PKCS12_CREATE                          105
281 #define PKCS12_F_PKCS12_GEN_MAC                         107
282 #define PKCS12_F_PKCS12_INIT                            109
283 #define PKCS12_F_PKCS12_ITEM_DECRYPT_D2I                106
284 #define PKCS12_F_PKCS12_ITEM_I2D_ENCRYPT                108
285 #define PKCS12_F_PKCS12_ITEM_PACK_SAFEBAG               117
286 #define PKCS12_F_PKCS12_KEY_GEN_ASC                     110
287 #define PKCS12_F_PKCS12_KEY_GEN_UNI                     111
288 #define PKCS12_F_PKCS12_MAKE_KEYBAG                     112
289 #define PKCS12_F_PKCS12_MAKE_SHKEYBAG                   113
290 #define PKCS12_F_PKCS12_NEWPASS                         128
291 #define PKCS12_F_PKCS12_PACK_P7DATA                     114
292 #define PKCS12_F_PKCS12_PACK_P7ENCDATA                  115
293 #define PKCS12_F_PKCS12_PARSE                           118
294 #define PKCS12_F_PKCS12_PBE_CRYPT                       119
295 #define PKCS12_F_PKCS12_PBE_KEYIVGEN                    120
296 #define PKCS12_F_PKCS12_SETUP_MAC                       122
297 #define PKCS12_F_PKCS12_SET_MAC                         123
298 #define PKCS12_F_PKCS12_UNPACK_AUTHSAFES                130
299 #define PKCS12_F_PKCS12_UNPACK_P7DATA                   131
300 #define PKCS12_F_PKCS12_VERIFY_MAC                      126
301 #define PKCS12_F_PKCS8_ADD_KEYUSAGE                     124
302 #define PKCS12_F_PKCS8_ENCRYPT                          125

304 /* Reason codes. */
305 #define PKCS12_R_CANT_PACK_STRUCTURE                    100
306 #define PKCS12_R_CONTENT_TYPE_NOT_DATA                  121
307 #define PKCS12_R_DECODE_ERROR                           101
308 #define PKCS12_R_ENCODE_ERROR                           102
309 #define PKCS12_R_ENCRYPT_ERROR                          103
310 #define PKCS12_R_ERROR_SETTING_ENCRYPTED_DATA_TYPE      120
311 #define PKCS12_R_INVALID_NULL_ARGUMENT                  104
312 #define PKCS12_R_INVALID_NULL_PKCS12_POINTER            105
313 #define PKCS12_R_IV_GEN_ERROR                           106
314 #define PKCS12_R_KEY_GEN_ERROR                          107
315 #define PKCS12_R_MAC_ABSENT                             108
316 #define PKCS12_R_MAC_GENERATION_ERROR                   109
317 #define PKCS12_R_MAC_SETUP_ERROR                        110
318 #define PKCS12_R_MAC_STRING_SET_ERROR                   111
319 #define PKCS12_R_MAC_VERIFY_ERROR                       112
320 #define PKCS12_R_MAC_VERIFY_FAILURE                     113
321 #define PKCS12_R_PARSE_ERROR                            114
322 #define PKCS12_R_PKCS12_ALGOR_CIPHERINIT_ERROR          115
323 #define PKCS12_R_PKCS12_CIPHERFINAL_ERROR               116
324 #define PKCS12_R_PKCS12_PBE_CRYPT_ERROR                 117
325 #define PKCS12_R_UNKNOWN_DIGEST_ALGORITHM               118
```



```
326 #define PKCS12_R_UNSUPPORTED_PKCS12_MODE                119

328 #ifdef  __cplusplus
329 }
330 #endif
331 #endif
332 #endif /* ! codereview */
```

```
**********************************************************
   18104 Wed Aug 13 19:51:47 2014
new/usr/src/lib/openssl/include/openssl/pkcs7.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pkcs7/pkcs7.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_PKCS7_H
#define HEADER_PKCS7_H
```



```
#include <openssl/asn1.h>
#include <openssl/bio.h>
#include <openssl/e_os2.h>

#include <openssl/symhacks.h>
#include <openssl/ossl_typ.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef OPENSSL_SYS_WIN32
/* Under Win32 thes are defined in wincrypt.h */
#undef PKCS7_ISSUER_AND_SERIAL
#undef PKCS7_SIGNER_INFO
#endif

/*
Encryption_ID           DES-CBC
Digest_ID               MD5
Digest_Encryption_ID    rsaEncryption
Key_Encryption_ID       rsaEncryption
*/

typedef struct pkcs7_issuer_and_serial_st
        {
        X509_NAME *issuer;
        ASN1_INTEGER *serial;
        } PKCS7_ISSUER_AND_SERIAL;

typedef struct pkcs7_signer_info_st
        {
        ASN1_INTEGER                    *version;       /* version 1 */
        PKCS7_ISSUER_AND_SERIAL         *issuer_and_serial;
        X509_ALGOR                      *digest_alg;
        STACK_OF(X509_ATTRIBUTE)        *auth_attr;     /* [ 0 ] */
        X509_ALGOR                      *digest_enc_alg;
        ASN1_OCTET_STRING               *enc_digest;
        STACK_OF(X509_ATTRIBUTE)        *unauth_attr;   /* [ 1 ] */

        /* The private key to sign with */
        EVP_PKEY                        *pkey;
        } PKCS7_SIGNER_INFO;

DECLARE_STACK_OF(PKCS7_SIGNER_INFO)
DECLARE_ASN1_SET_OF(PKCS7_SIGNER_INFO)

typedef struct pkcs7_recip_info_st
        {
        ASN1_INTEGER                    *version;       /* version 0 */
        PKCS7_ISSUER_AND_SERIAL         *issuer_and_serial;
        X509_ALGOR                      *key_enc_algor;
        ASN1_OCTET_STRING               *enc_key;
        X509                            *cert; /* get the pub-key from this */
        } PKCS7_RECIP_INFO;

DECLARE_STACK_OF(PKCS7_RECIP_INFO)
DECLARE_ASN1_SET_OF(PKCS7_RECIP_INFO)

typedef struct pkcs7_signed_st
        {
        ASN1_INTEGER                    *version;       /* version 1 */
        STACK_OF(X509_ALGOR)            *md_algs;       /* md used */
        STACK_OF(X509)                  *cert;          /* [ 0 ] */
        STACK_OF(X509_CRL)              *crl;           /* [ 1 ] */
        STACK_OF(PKCS7_SIGNER_INFO)     *signer_info;
```

```
129         struct pkcs7_st                 *contents;
130         } PKCS7_SIGNED;
131 /* The above structure is very very similar to PKCS7_SIGN_ENVELOPE.
132  * How about merging the two */

134 typedef struct pkcs7_enc_content_st
135         {
136         ASN1_OBJECT                     *content_type;
137         X509_ALGOR                      *algorithm;
138         ASN1_OCTET_STRING               *enc_data;      /* [ 0 ] */
139         const EVP_CIPHER                *cipher;
140         } PKCS7_ENC_CONTENT;

142 typedef struct pkcs7_enveloped_st
143         {
144         ASN1_INTEGER                    *version;       /* version 0 */
145         STACK_OF(PKCS7_RECIP_INFO)      *recipientinfo;
146         PKCS7_ENC_CONTENT               *enc_data;
147         } PKCS7_ENVELOPE;

149 typedef struct pkcs7_signedandenveloped_st
150         {
151         ASN1_INTEGER                    *version;       /* version 1 */
152         STACK_OF(X509_ALGOR)            *md_algs;       /* md used */
153         STACK_OF(X509)                  *cert;          /* [ 0 ] */
154         STACK_OF(X509_CRL)              *crl;           /* [ 1 ] */
155         STACK_OF(PKCS7_SIGNER_INFO)     *signer_info;

157         PKCS7_ENC_CONTENT               *enc_data;
158         STACK_OF(PKCS7_RECIP_INFO)      *recipientinfo;
159         } PKCS7_SIGN_ENVELOPE;

161 typedef struct pkcs7_digest_st
162         {
163         ASN1_INTEGER                    *version;       /* version 0 */
164         X509_ALGOR                      *md;            /* md used */
165         struct pkcs7_st                 *contents;
166         ASN1_OCTET_STRING               *digest;
167         } PKCS7_DIGEST;

169 typedef struct pkcs7_encrypted_st
170         {
171         ASN1_INTEGER                    *version;       /* version 0 */
172         PKCS7_ENC_CONTENT               *enc_data;
173         } PKCS7_ENCRYPT;

175 typedef struct pkcs7_st
176         {
177         /* The following is non NULL if it contains ASN1 encoding of
178          * this structure */
179         unsigned char *asn1;
180         long length;

182 #define PKCS7_S_HEADER  0
183 #define PKCS7_S_BODY    1
184 #define PKCS7_S_TAIL    2
185         int state; /* used during processing */

187         int detached;

189         ASN1_OBJECT *type;
190         /* content as defined by the type */
191         /* all encryption/message digests are applied to the 'contents',
192          * leaving out the 'type' field. */
193         union   {
```



```
194                 char *ptr;

196                 /* NID_pkcs7_data */
197                 ASN1_OCTET_STRING *data;

199                 /* NID_pkcs7_signed */
200                 PKCS7_SIGNED *sign;

202                 /* NID_pkcs7_enveloped */
203                 PKCS7_ENVELOPE *enveloped;

205                 /* NID_pkcs7_signedAndEnveloped */
206                 PKCS7_SIGN_ENVELOPE *signed_and_enveloped;

208                 /* NID_pkcs7_digest */
209                 PKCS7_DIGEST *digest;

211                 /* NID_pkcs7_encrypted */
212                 PKCS7_ENCRYPT *encrypted;

214                 /* Anything else */
215                 ASN1_TYPE *other;
216                 } d;
217         } PKCS7;

219 DECLARE_STACK_OF(PKCS7)
220 DECLARE_ASN1_SET_OF(PKCS7)
221 DECLARE_PKCS12_STACK_OF(PKCS7)

223 #define PKCS7_OP_SET_DETACHED_SIGNATURE 1
224 #define PKCS7_OP_GET_DETACHED_SIGNATURE 2

226 #define PKCS7_get_signed_attributes(si) ((si)->auth_attr)
227 #define PKCS7_get_attributes(si)        ((si)->unauth_attr)

229 #define PKCS7_type_is_signed(a) (OBJ_obj2nid((a)->type) == NID_pkcs7_signed)
230 #define PKCS7_type_is_encrypted(a) (OBJ_obj2nid((a)->type) == NID_pkcs7_encrypte
231 #define PKCS7_type_is_enveloped(a) (OBJ_obj2nid((a)->type) == NID_pkcs7_envelope
232 #define PKCS7_type_is_signedAndEnveloped(a) \
233                 (OBJ_obj2nid((a)->type) == NID_pkcs7_signedAndEnveloped)
234 #define PKCS7_type_is_data(a)   (OBJ_obj2nid((a)->type) == NID_pkcs7_data)
235 #define PKCS7_type_is_digest(a)  (OBJ_obj2nid((a)->type) == NID_pkcs7_digest)
236 #define PKCS7_type_is_encrypted(a) \
237                 (OBJ_obj2nid((a)->type) == NID_pkcs7_encrypted)

239 #define PKCS7_type_is_digest(a)   (OBJ_obj2nid((a)->type) == NID_pkcs7_digest)

241 #define PKCS7_set_detached(p,v) \
242                 PKCS7_ctrl(p,PKCS7_OP_SET_DETACHED_SIGNATURE,v,NULL)
243 #define PKCS7_get_detached(p) \
244                 PKCS7_ctrl(p,PKCS7_OP_GET_DETACHED_SIGNATURE,0,NULL)

246 #define PKCS7_is_detached(p7) (PKCS7_type_is_signed(p7) && PKCS7_get_detached(p7

248 /* S/MIME related flags */

250 #define PKCS7_TEXT              0x1
251 #define PKCS7_NOCERTS           0x2
252 #define PKCS7_NOSIGS            0x4
253 #define PKCS7_NOCHAIN           0x8
254 #define PKCS7_NOINTERN          0x10
255 #define PKCS7_NOVERIFY          0x20
256 #define PKCS7_DETACHED          0x40
257 #define PKCS7_BINARY            0x80
258 #define PKCS7_NOATTR            0x100
259 #define PKCS7_NOSMIMECAP        0x200
```

```
#define PKCS7_NOOLDMIMETYPE     0x400
#define PKCS7_CRLFEOL           0x800
#define PKCS7_STREAM            0x1000
#define PKCS7_NOCRL             0x2000
#define PKCS7_PARTIAL           0x4000
#define PKCS7_REUSE_DIGEST      0x8000

/* Flags: for compatibility with older code */

#define SMIME_TEXT      PKCS7_TEXT
#define SMIME_NOCERTS   PKCS7_NOCERTS
#define SMIME_NOSIGS    PKCS7_NOSIGS
#define SMIME_NOCHAIN   PKCS7_NOCHAIN
#define SMIME_NOINTERN  PKCS7_NOINTERN
#define SMIME_NOVERIFY  PKCS7_NOVERIFY
#define SMIME_DETACHED  PKCS7_DETACHED
#define SMIME_BINARY    PKCS7_BINARY
#define SMIME_NOATTR    PKCS7_NOATTR

DECLARE_ASN1_FUNCTIONS(PKCS7_ISSUER_AND_SERIAL)

int PKCS7_ISSUER_AND_SERIAL_digest(PKCS7_ISSUER_AND_SERIAL *data,const EVP_MD *t
        unsigned char *md,unsigned int *len);
#ifndef OPENSSL_NO_FP_API
PKCS7 *d2i_PKCS7_fp(FILE *fp,PKCS7 **p7);
int i2d_PKCS7_fp(FILE *fp,PKCS7 *p7);
#endif
PKCS7 *PKCS7_dup(PKCS7 *p7);
PKCS7 *d2i_PKCS7_bio(BIO *bp,PKCS7 **p7);
int i2d_PKCS7_bio(BIO *bp,PKCS7 *p7);
int i2d_PKCS7_bio_stream(BIO *out, PKCS7 *p7, BIO *in, int flags);
int PEM_write_bio_PKCS7_stream(BIO *out, PKCS7 *p7, BIO *in, int flags);

DECLARE_ASN1_FUNCTIONS(PKCS7_SIGNER_INFO)
DECLARE_ASN1_FUNCTIONS(PKCS7_RECIP_INFO)
DECLARE_ASN1_FUNCTIONS(PKCS7_SIGNED)
DECLARE_ASN1_FUNCTIONS(PKCS7_ENC_CONTENT)
DECLARE_ASN1_FUNCTIONS(PKCS7_ENVELOPE)
DECLARE_ASN1_FUNCTIONS(PKCS7_SIGN_ENVELOPE)
DECLARE_ASN1_FUNCTIONS(PKCS7_DIGEST)
DECLARE_ASN1_FUNCTIONS(PKCS7_ENCRYPT)
DECLARE_ASN1_FUNCTIONS(PKCS7)

DECLARE_ASN1_ITEM(PKCS7_ATTR_SIGN)
DECLARE_ASN1_ITEM(PKCS7_ATTR_VERIFY)

DECLARE_ASN1_NDEF_FUNCTION(PKCS7)
DECLARE_ASN1_PRINT_FUNCTION(PKCS7)

long PKCS7_ctrl(PKCS7 *p7, int cmd, long larg, char *parg);

int PKCS7_set_type(PKCS7 *p7, int type);
int PKCS7_set0_type_other(PKCS7 *p7, int type, ASN1_TYPE *other);
int PKCS7_set_content(PKCS7 *p7, PKCS7 *p7_data);
int PKCS7_SIGNER_INFO_set(PKCS7_SIGNER_INFO *p7i, X509 *x509, EVP_PKEY *pkey,
        const EVP_MD *dgst);
int PKCS7_SIGNER_INFO_sign(PKCS7_SIGNER_INFO *si);
int PKCS7_add_signer(PKCS7 *p7, PKCS7_SIGNER_INFO *p7i);
int PKCS7_add_certificate(PKCS7 *p7, X509 *x509);
int PKCS7_add_crl(PKCS7 *p7, X509_CRL *x509);
int PKCS7_content_new(PKCS7 *p7, int nid);
int PKCS7_dataVerify(X509_STORE *cert_store, X509_STORE_CTX *ctx,
        BIO *bio, PKCS7 *p7, PKCS7_SIGNER_INFO *si);
int PKCS7_signatureVerify(BIO *bio, PKCS7 *p7, PKCS7_SIGNER_INFO *si,
                                                                X509 *x509);
```



```
BIO *PKCS7_dataInit(PKCS7 *p7, BIO *bio);
int PKCS7_dataFinal(PKCS7 *p7, BIO *bio);
BIO *PKCS7_dataDecode(PKCS7 *p7, EVP_PKEY *pkey, BIO *in_bio, X509 *pcert);


PKCS7_SIGNER_INFO *PKCS7_add_signature(PKCS7 *p7, X509 *x509,
        EVP_PKEY *pkey, const EVP_MD *dgst);
X509 *PKCS7_cert_from_signer_info(PKCS7 *p7, PKCS7_SIGNER_INFO *si);
int PKCS7_set_digest(PKCS7 *p7, const EVP_MD *md);
STACK_OF(PKCS7_SIGNER_INFO) *PKCS7_get_signer_info(PKCS7 *p7);

PKCS7_RECIP_INFO *PKCS7_add_recipient(PKCS7 *p7, X509 *x509);
void PKCS7_SIGNER_INFO_get0_algs(PKCS7_SIGNER_INFO *si, EVP_PKEY **pk,
                                 X509_ALGOR **pdig, X509_ALGOR **psig);
void PKCS7_RECIP_INFO_get0_alg(PKCS7_RECIP_INFO *ri, X509_ALGOR **penc);
int PKCS7_add_recipient_info(PKCS7 *p7, PKCS7_RECIP_INFO *ri);
int PKCS7_RECIP_INFO_set(PKCS7_RECIP_INFO *p7i, X509 *x509);
int PKCS7_set_cipher(PKCS7 *p7, const EVP_CIPHER *cipher);
int PKCS7_stream(unsigned char ***boundary, PKCS7 *p7);

PKCS7_ISSUER_AND_SERIAL *PKCS7_get_issuer_and_serial(PKCS7 *p7, int idx);
ASN1_OCTET_STRING *PKCS7_digest_from_attributes(STACK_OF(X509_ATTRIBUTE) *sk);
int PKCS7_add_signed_attribute(PKCS7_SIGNER_INFO *p7si,int nid,int type,
        void *data);
int PKCS7_add_attribute (PKCS7_SIGNER_INFO *p7si, int nid, int atrtype,
        void *value);
ASN1_TYPE *PKCS7_get_attribute(PKCS7_SIGNER_INFO *si, int nid);
ASN1_TYPE *PKCS7_get_signed_attribute(PKCS7_SIGNER_INFO *si, int nid);
int PKCS7_set_signed_attributes(PKCS7_SIGNER_INFO *p7si,
                                STACK_OF(X509_ATTRIBUTE) *sk);
int PKCS7_set_attributes(PKCS7_SIGNER_INFO *p7si,STACK_OF(X509_ATTRIBUTE) *sk);


PKCS7 *PKCS7_sign(X509 *signcert, EVP_PKEY *pkey, STACK_OF(X509) *certs,
                                                        BIO *data, int flags);

PKCS7_SIGNER_INFO *PKCS7_sign_add_signer(PKCS7 *p7,
                        X509 *signcert, EVP_PKEY *pkey, const EVP_MD *md,
                        int flags);

int PKCS7_final(PKCS7 *p7, BIO *data, int flags);
int PKCS7_verify(PKCS7 *p7, STACK_OF(X509) *certs, X509_STORE *store,
                                        BIO *indata, BIO *out, int flags);
STACK_OF(X509) *PKCS7_get0_signers(PKCS7 *p7, STACK_OF(X509) *certs, int flags);
PKCS7 *PKCS7_encrypt(STACK_OF(X509) *certs, BIO *in, const EVP_CIPHER *cipher,
                                                            int flags);
int PKCS7_decrypt(PKCS7 *p7, EVP_PKEY *pkey, X509 *cert, BIO *data, int flags);

int PKCS7_add_attrib_smimecap(PKCS7_SIGNER_INFO *si,
                              STACK_OF(X509_ALGOR) *cap);
STACK_OF(X509_ALGOR) *PKCS7_get_smimecap(PKCS7_SIGNER_INFO *si);
int PKCS7_simple_smimecap(STACK_OF(X509_ALGOR) *sk, int nid, int arg);

int PKCS7_add_attrib_content_type(PKCS7_SIGNER_INFO *si, ASN1_OBJECT *coid);
int PKCS7_add0_attrib_signing_time(PKCS7_SIGNER_INFO *si, ASN1_TIME *t);
int PKCS7_add1_attrib_digest(PKCS7_SIGNER_INFO *si,
                             const unsigned char *md, int mdlen);

int SMIME_write_PKCS7(BIO *bio, PKCS7 *p7, BIO *data, int flags);
PKCS7 *SMIME_read_PKCS7(BIO *bio, BIO **bcont);

BIO *BIO_new_PKCS7(BIO *out, PKCS7 *p7);


/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
```

```
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_PKCS7_strings(void);

/* Error codes for the PKCS7 functions. */

/* Function codes. */
#define PKCS7_F_B64_READ_PKCS7                          120
#define PKCS7_F_B64_WRITE_PKCS7                         121
#define PKCS7_F_DO_PKCS7_SIGNED_ATTRIB                  136
#define PKCS7_F_I2D_PKCS7_BIO_STREAM                    140
#define PKCS7_F_PKCS7_ADD0_ATTRIB_SIGNING_TIME          135
#define PKCS7_F_PKCS7_ADD_ATTRIB_SMIMECAP               118
#define PKCS7_F_PKCS7_ADD_CERTIFICATE                   100
#define PKCS7_F_PKCS7_ADD_CRL                           101
#define PKCS7_F_PKCS7_ADD_RECIPIENT_INFO                102
#define PKCS7_F_PKCS7_ADD_SIGNATURE                     131
#define PKCS7_F_PKCS7_ADD_SIGNER                        103
#define PKCS7_F_PKCS7_BIO_ADD_DIGEST                    125
#define PKCS7_F_PKCS7_COPY_EXISTING_DIGEST              138
#define PKCS7_F_PKCS7_CTRL                              104
#define PKCS7_F_PKCS7_DATADECODE                        112
#define PKCS7_F_PKCS7_DATAFINAL                         128
#define PKCS7_F_PKCS7_DATAINIT                          105
#define PKCS7_F_PKCS7_DATASIGN                          106
#define PKCS7_F_PKCS7_DATAVERIFY                        107
#define PKCS7_F_PKCS7_DECRYPT                           114
#define PKCS7_F_PKCS7_DECRYPT_RINFO                     133
#define PKCS7_F_PKCS7_ENCODE_RINFO                      132
#define PKCS7_F_PKCS7_ENCRYPT                           115
#define PKCS7_F_PKCS7_FINAL                             134
#define PKCS7_F_PKCS7_FIND_DIGEST                       127
#define PKCS7_F_PKCS7_GET0_SIGNERS                      124
#define PKCS7_F_PKCS7_RECIP_INFO_SET                    130
#define PKCS7_F_PKCS7_SET_CIPHER                        108
#define PKCS7_F_PKCS7_SET_CONTENT                       109
#define PKCS7_F_PKCS7_SET_DIGEST                        126
#define PKCS7_F_PKCS7_SET_TYPE                          110
#define PKCS7_F_PKCS7_SIGN                              116
#define PKCS7_F_PKCS7_SIGNATUREVERIFY                   113
#define PKCS7_F_PKCS7_SIGNER_INFO_SET                   129
#define PKCS7_F_PKCS7_SIGNER_INFO_SIGN                  139
#define PKCS7_F_PKCS7_SIGN_ADD_SIGNER                   137
#define PKCS7_F_PKCS7_SIMPLE_SMIMECAP                   119
#define PKCS7_F_PKCS7_VERIFY                            117
#define PKCS7_F_SMIME_READ_PKCS7                        122
#define PKCS7_F_SMIME_TEXT                              123

/* Reason codes. */
#define PKCS7_R_CERTIFICATE_VERIFY_ERROR                117
#define PKCS7_R_CIPHER_HAS_NO_OBJECT_IDENTIFIER         144
#define PKCS7_R_CIPHER_NOT_INITIALIZED                  116
#define PKCS7_R_CONTENT_AND_DATA_PRESENT                118
#define PKCS7_R_CTRL_ERROR                              152
#define PKCS7_R_DECODE_ERROR                            130
#define PKCS7_R_DECRYPTED_KEY_IS_WRONG_LENGTH           100
#define PKCS7_R_DECRYPT_ERROR                           119
#define PKCS7_R_DIGEST_FAILURE                          101
#define PKCS7_R_ENCRYPTION_CTRL_FAILURE                 149
#define PKCS7_R_ENCRYPTION_NOT_SUPPORTED_FOR_THIS_KEY_TYPE 150
#define PKCS7_R_ERROR_ADDING_RECIPIENT                  120
#define PKCS7_R_ERROR_SETTING_CIPHER                    121
#define PKCS7_R_INVALID_MIME_TYPE                       131
#define PKCS7_R_INVALID_NULL_POINTER                    143
#define PKCS7_R_INVALID_SIGNED_DATA_TYPE                155
#define PKCS7_R_MIME_NO_CONTENT_TYPE                    132
```



```
#define PKCS7_R_MIME_PARSE_ERROR                        133
#define PKCS7_R_MIME_SIG_PARSE_ERROR                    134
#define PKCS7_R_MISSING_CERIPEND_INFO                   103
#define PKCS7_R_NO_CONTENT                              122
#define PKCS7_R_NO_CONTENT_TYPE                         135
#define PKCS7_R_NO_DEFAULT_DIGEST                       151
#define PKCS7_R_NO_MATCHING_DIGEST_TYPE_FOUND           154
#define PKCS7_R_NO_MULTIPART_BODY_FAILURE               136
#define PKCS7_R_NO_MULTIPART_BOUNDARY                   137
#define PKCS7_R_NO_RECIPIENT_MATCHES_CERTIFICATE        115
#define PKCS7_R_NO_RECIPIENT_MATCHES_KEY                146
#define PKCS7_R_NO_SIGNATURES_ON_DATA                   123
#define PKCS7_R_NO_SIGNERS                              142
#define PKCS7_R_NO_SIG_CONTENT_TYPE                     138
#define PKCS7_R_OPERATION_NOT_SUPPORTED_ON_THIS_TYPE    104
#define PKCS7_R_PKCS7_ADD_SIGNATURE_ERROR               124
#define PKCS7_R_PKCS7_ADD_SIGNER_ERROR                  153
#define PKCS7_R_PKCS7_DATAFINAL                         126
#define PKCS7_R_PKCS7_DATAFINAL_ERROR                   125
#define PKCS7_R_PKCS7_DATASIGN                          145
#define PKCS7_R_PKCS7_PARSE_ERROR                       139
#define PKCS7_R_PKCS7_SIG_PARSE_ERROR                   140
#define PKCS7_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE  127
#define PKCS7_R_SIGNATURE_FAILURE                       105
#define PKCS7_R_SIGNER_CERTIFICATE_NOT_FOUND            128
#define PKCS7_R_SIGNING_CTRL_FAILURE                    147
#define PKCS7_R_SIGNING_NOT_SUPPORTED_FOR_THIS_KEY_TYPE 148
#define PKCS7_R_SIG_INVALID_MIME_TYPE                   141
#define PKCS7_R_SMIME_TEXT_ERROR                        129
#define PKCS7_R_UNABLE_TO_FIND_CERTIFICATE              106
#define PKCS7_R_UNABLE_TO_FIND_MEM_BIO                  107
#define PKCS7_R_UNABLE_TO_FIND_MESSAGE_DIGEST           108
#define PKCS7_R_UNKNOWN_DIGEST_TYPE                     109
#define PKCS7_R_UNKNOWN_OPERATION                       110
#define PKCS7_R_UNSUPPORTED_CIPHER_TYPE                 111
#define PKCS7_R_UNSUPPORTED_CONTENT_TYPE                112
#define PKCS7_R_WRONG_CONTENT_TYPE                      113
#define PKCS7_R_WRONG_PKCS7_TYPE                        114

#ifdef  __cplusplus
}
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3576 Wed Aug 13 19:51:47 2014
new/usr/src/lib/openssl/include/openssl/pqueue.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pqueue/pqueue.h */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_PQUEUE_H
#define HEADER_PQUEUE_H
```



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct _pqueue *pqueue;

typedef struct _pitem
        {
        unsigned char priority[8]; /* 64-bit value in big-endian encoding */
        void *data;
        struct _pitem *next;
        } pitem;

typedef struct _pitem *piterator;

pitem *pitem_new(unsigned char *prio64be, void *data);
void   pitem_free(pitem *item);

pqueue pqueue_new(void);
void   pqueue_free(pqueue pq);

pitem *pqueue_insert(pqueue pq, pitem *item);
pitem *pqueue_peek(pqueue pq);
pitem *pqueue_pop(pqueue pq);
pitem *pqueue_find(pqueue pq, unsigned char *prio64be);
pitem *pqueue_iterator(pqueue pq);
pitem *pqueue_next(piterator *iter);

void   pqueue_print(pqueue pq);
int    pqueue_size(pqueue pq);

#endif /* ! HEADER_PQUEUE_H */
#endif /* ! codereview */
```

```
**********************************************************
    5572 Wed Aug 13 19:51:47 2014
new/usr/src/lib/openssl/include/openssl/rand.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/rand.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_RAND_H
#define HEADER_RAND_H
```



```
#include <stdlib.h>
#include <openssl/ossl_typ.h>
#include <openssl/e_os2.h>

#if defined(OPENSSL_SYS_WINDOWS)
#include <windows.h>
#endif

#ifdef  __cplusplus
extern "C" {
#endif

#if defined(OPENSSL_FIPS)
#define FIPS_RAND_SIZE_T size_t
#endif

/* Already defined in ossl_typ.h */
/* typedef struct rand_meth_st RAND_METHOD; */

struct rand_meth_st
        {
        void (*seed)(const void *buf, int num);
        int (*bytes)(unsigned char *buf, int num);
        void (*cleanup)(void);
        void (*add)(const void *buf, int num, double entropy);
        int (*pseudorand)(unsigned char *buf, int num);
        int (*status)(void);
        };

#ifdef BN_DEBUG
extern int rand_predictable;
#endif

int RAND_set_rand_method(const RAND_METHOD *meth);
const RAND_METHOD *RAND_get_rand_method(void);
#ifndef OPENSSL_NO_ENGINE
int RAND_set_rand_engine(ENGINE *engine);
#endif
RAND_METHOD *RAND_SSLeay(void);
void RAND_cleanup(void );
int  RAND_bytes(unsigned char *buf,int num);
int  RAND_pseudo_bytes(unsigned char *buf,int num);
void RAND_seed(const void *buf,int num);
void RAND_add(const void *buf,int num,double entropy);
int  RAND_load_file(const char *file,long max_bytes);
int  RAND_write_file(const char *file);
const char *RAND_file_name(char *file,size_t num);
int RAND_status(void);
int RAND_query_egd_bytes(const char *path, unsigned char *buf, int bytes);
int RAND_egd(const char *path);
int RAND_egd_bytes(const char *path,int bytes);
int RAND_poll(void);

#if defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_WIN32)

void RAND_screen(void);
int RAND_event(UINT, WPARAM, LPARAM);

#endif

#ifdef OPENSSL_FIPS
void RAND_set_fips_drbg_type(int type, int flags);
int RAND_init_fips(void);
#endif

/* BEGIN ERROR CODES */
```

```
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_RAND_strings(void);

/* Error codes for the RAND functions. */

/* Function codes. */
#define RAND_F_RAND_GET_RAND_METHOD                     101
#define RAND_F_RAND_INIT_FIPS                           102
#define RAND_F_SSLEAY_RAND_BYTES                        100

/* Reason codes. */
#define RAND_R_DUAL_EC_DRBG_DISABLED                    104
#define RAND_R_ERROR_INITIALISING_DRBG                  102
#define RAND_R_ERROR_INSTANTIATING_DRBG                 103
#define RAND_R_NO_FIPS_RANDOM_METHOD_SET                101
#define RAND_R_PRNG_NOT_SEEDED                          100

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4398 Wed Aug 13 19:51:47 2014
new/usr/src/lib/openssl/include/openssl/rc2.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc2/rc2.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_RC2_H
#define HEADER_RC2_H
```



```
#include <openssl/opensslconf.h> /* OPENSSL_NO_RC2, RC2_INT */
#ifdef OPENSSL_NO_RC2
#error RC2 is disabled.
#endif

#define RC2_ENCRYPT     1
#define RC2_DECRYPT     0

#define RC2_BLOCK       8
#define RC2_KEY_LENGTH  16

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct rc2_key_st
        {
        RC2_INT data[64];
        } RC2_KEY;

#ifdef OPENSSL_FIPS
void private_RC2_set_key(RC2_KEY *key, int len, const unsigned char *data,int bi
#endif
void RC2_set_key(RC2_KEY *key, int len, const unsigned char *data,int bits);
void RC2_ecb_encrypt(const unsigned char *in,unsigned char *out,RC2_KEY *key,
                     int enc);
void RC2_encrypt(unsigned long *data,RC2_KEY *key);
void RC2_decrypt(unsigned long *data,RC2_KEY *key);
void RC2_cbc_encrypt(const unsigned char *in, unsigned char *out, long length,
        RC2_KEY *ks, unsigned char *iv, int enc);
void RC2_cfb64_encrypt(const unsigned char *in, unsigned char *out,
                       long length, RC2_KEY *schedule, unsigned char *ivec,
                       int *num, int enc);
void RC2_ofb64_encrypt(const unsigned char *in, unsigned char *out,
                       long length, RC2_KEY *schedule, unsigned char *ivec,
                       int *num);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    3790 Wed Aug 13 19:51:47 2014
new/usr/src/lib/openssl/include/openssl/rc4.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc4/rc4.h */
/* Copyright (C) 1995-1997 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_RC4_H
#define HEADER_RC4_H
```



```
#include <openssl/opensslconf.h> /* OPENSSL_NO_RC4, RC4_INT */
#ifdef OPENSSL_NO_RC4
#error RC4 is disabled.
#endif

#include <stddef.h>

#ifdef  __cplusplus
extern "C" {
#endif

typedef struct rc4_key_st
        {
        RC4_INT x,y;
        RC4_INT data[256];
        } RC4_KEY;


const char *RC4_options(void);
void RC4_set_key(RC4_KEY *key, int len, const unsigned char *data);
void private_RC4_set_key(RC4_KEY *key, int len, const unsigned char *data);
void RC4(RC4_KEY *key, size_t len, const unsigned char *indata,
                unsigned char *outdata);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    4321 Wed Aug 13 19:51:48 2014
new/usr/src/lib/openssl/include/openssl/ripemd.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ripemd/ripemd.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_RIPEMD_H
#define HEADER_RIPEMD_H
```



```
#include <openssl/e_os2.h>
#include <stddef.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef OPENSSL_NO_RIPEMD
#error RIPEMD is disabled.
#endif

#if defined(__LP32__)
#define RIPEMD160_LONG unsigned long
#elif defined(OPENSSL_SYS_CRAY) || defined(__ILP64__)
#define RIPEMD160_LONG unsigned long
#define RIPEMD160_LONG_LOG2 3
#else
#define RIPEMD160_LONG unsigned int
#endif

#define RIPEMD160_CBLOCK        64
#define RIPEMD160_LBLOCK        (RIPEMD160_CBLOCK/4)
#define RIPEMD160_DIGEST_LENGTH 20

typedef struct RIPEMD160state_st
        {
        RIPEMD160_LONG A,B,C,D,E;
        RIPEMD160_LONG Nl,Nh;
        RIPEMD160_LONG data[RIPEMD160_LBLOCK];
        unsigned int   num;
        } RIPEMD160_CTX;

#ifdef OPENSSL_FIPS
int private_RIPEMD160_Init(RIPEMD160_CTX *c);
#endif
int RIPEMD160_Init(RIPEMD160_CTX *c);
int RIPEMD160_Update(RIPEMD160_CTX *c, const void *data, size_t len);
int RIPEMD160_Final(unsigned char *md, RIPEMD160_CTX *c);
unsigned char *RIPEMD160(const unsigned char *d, size_t n,
        unsigned char *md);
void RIPEMD160_Transform(RIPEMD160_CTX *c, const unsigned char *b);
#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
   22855 Wed Aug 13 19:51:48 2014
new/usr/src/lib/openssl/include/openssl/rsa.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_RSA_H
#define HEADER_RSA_H
```



```
#include <openssl/asn1.h>

#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/crypto.h>
#include <openssl/ossl_typ.h>
#ifndef OPENSSL_NO_DEPRECATED
#include <openssl/bn.h>
#endif

#ifdef OPENSSL_NO_RSA
#error RSA is disabled.
#endif

#ifdef __cplusplus
extern "C" {
#endif

/* Declared already in ossl_typ.h */
/* typedef struct rsa_st RSA; */
/* typedef struct rsa_meth_st RSA_METHOD; */

struct rsa_meth_st
        {
        const char *name;
        int (*rsa_pub_enc)(int flen,const unsigned char *from,
                           unsigned char *to,
                           RSA *rsa,int padding);
        int (*rsa_pub_dec)(int flen,const unsigned char *from,
                           unsigned char *to,
                           RSA *rsa,int padding);
        int (*rsa_priv_enc)(int flen,const unsigned char *from,
                            unsigned char *to,
                            RSA *rsa,int padding);
        int (*rsa_priv_dec)(int flen,const unsigned char *from,
                            unsigned char *to,
                            RSA *rsa,int padding);
        int (*rsa_mod_exp)(BIGNUM *r0,const BIGNUM *I,RSA *rsa,BN_CTX *ctx); /*
        int (*bn_mod_exp)(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
                          const BIGNUM *m, BN_CTX *ctx,
                          BN_MONT_CTX *m_ctx); /* Can be null */
        int (*init)(RSA *rsa);          /* called at new */
        int (*finish)(RSA *rsa);        /* called at free */
        int flags;                      /* RSA_METHOD_FLAG_* things */
        char *app_data;                 /* may be needed! */
/* New sign and verify functions: some libraries don't allow arbitrary data
 * to be signed/verified: this allows them to be used. Note: for this to work
 * the RSA_public_decrypt() and RSA_private_encrypt() should *NOT* be used
 * RSA_sign(), RSA_verify() should be used instead. Note: for backwards
 * compatibility this functionality is only enabled if the RSA_FLAG_SIGN_VER
 * option is set in 'flags'.
 */
        int (*rsa_sign)(int type,
                const unsigned char *m, unsigned int m_length,
                unsigned char *sigret, unsigned int *siglen, const RSA *rsa);
        int (*rsa_verify)(int dtype,
                const unsigned char *m, unsigned int m_length,
                const unsigned char *sigbuf, unsigned int siglen,
                                                        const RSA *rsa);
/* If this callback is NULL, the builtin software RSA key-gen will be used. This
 * is for behavioural compatibility whilst the code gets rewired, but one day
 * it would be nice to assume there are no such things as "builtin software"
 * implementations. */
        int (*rsa_keygen)(RSA *rsa, int bits, BIGNUM *e, BN_GENCB *cb);
        };
```

```
129 struct rsa_st
130         {
131         /* The first parameter is used to pickup errors where
132          * this is passed instead of aEVP_PKEY, it is set to 0 */
133         int pad;
134         long version;
135         const RSA_METHOD *meth;
136         /* functional reference if 'meth' is ENGINE-provided */
137         ENGINE *engine;
138         BIGNUM *n;
139         BIGNUM *e;
140         BIGNUM *d;
141         BIGNUM *p;
142         BIGNUM *q;
143         BIGNUM *dmp1;
144         BIGNUM *dmq1;
145         BIGNUM *iqmp;
146         /* be careful using this if the RSA structure is shared */
147         CRYPTO_EX_DATA ex_data;
148         int references;
149         int flags;

151         /* Used to cache montgomery values */
152         BN_MONT_CTX *_method_mod_n;
153         BN_MONT_CTX *_method_mod_p;
154         BN_MONT_CTX *_method_mod_q;

156         /* all BIGNUM values are actually in the following data, if it is not
157          * NULL */
158         char *bignum_data;
159         BN_BLINDING *blinding;
160         BN_BLINDING *mt_blinding;
161         };

163 #ifndef OPENSSL_RSA_MAX_MODULUS_BITS
164 # define OPENSSL_RSA_MAX_MODULUS_BITS   16384
165 #endif

167 #ifndef OPENSSL_RSA_SMALL_MODULUS_BITS
168 # define OPENSSL_RSA_SMALL_MODULUS_BITS 3072
169 #endif
170 #ifndef OPENSSL_RSA_MAX_PUBEXP_BITS
171 # define OPENSSL_RSA_MAX_PUBEXP_BITS    64 /* exponent limit enforced for "large
172 #endif

174 #define RSA_3   0x3L
175 #define RSA_F4  0x10001L

177 #define RSA_METHOD_FLAG_NO_CHECK        0x0001 /* don't check pub/private match

179 #define RSA_FLAG_CACHE_PUBLIC           0x0002
180 #define RSA_FLAG_CACHE_PRIVATE          0x0004
181 #define RSA_FLAG_BLINDING               0x0008
182 #define RSA_FLAG_THREAD_SAFE            0x0010
183 /* This flag means the private key operations will be handled by rsa_mod_exp
184  * and that they do not depend on the private key components being present:
185  * for example a key stored in external hardware. Without this flag bn_mod_exp
186  * gets called when private key components are absent.
187  */
188 #define RSA_FLAG_EXT_PKEY               0x0020

190 /* This flag in the RSA_METHOD enables the new rsa_sign, rsa_verify functions.
191  */
192 #define RSA_FLAG_SIGN_VER               0x0040
```



```
194 #define RSA_FLAG_NO_BLINDING            0x0080 /* new with 0.9.6j and 0.9.7b; th
195                                                 * RSA implementation now uses bl
196                                                 * default (ignoring RSA_FLAG_BLI
197                                                 * but other engines might not ne
198                                                 */
199 #define RSA_FLAG_NO_CONSTTIME           0x0100 /* new with 0.9.8f; the built-in
200                                                 * implementation now uses consta
201                                                 * operations by default in priva
202                                                 * e.g., constant time modular ex
203                                                 * modular inverse without leakin
204                                                 * division without leaking branc
205                                                 * flag disables these constant t
206                                                 * operations and results in fast
207                                                 * private key operations.
208                                                 */
209 #ifndef OPENSSL_NO_DEPRECATED
210 #define RSA_FLAG_NO_EXP_CONSTTIME RSA_FLAG_NO_CONSTTIME /* deprecated name for t
211                                                 /* new with 0.9.7h; the built-in
212                                                 * implementation now uses consta
213                                                 * modular exponentiation for sec
214                                                 * by default. This flag causes t
215                                                 * faster variable sliding window
216                                                 * be used for all exponents.
217                                                 */
218 #endif


221 #define EVP_PKEY_CTX_set_rsa_padding(ctx, pad) \
222         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, -1, EVP_PKEY_CTRL_RSA_PADDING, \
223                                 pad, NULL)

225 #define EVP_PKEY_CTX_get_rsa_padding(ctx, ppad) \
226         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, -1, \
227                                 EVP_PKEY_CTRL_GET_RSA_PADDING, 0, ppad)

229 #define EVP_PKEY_CTX_set_rsa_pss_saltlen(ctx, len) \
230         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, \
231                                 (EVP_PKEY_OP_SIGN|EVP_PKEY_OP_VERIFY), \
232                                 EVP_PKEY_CTRL_RSA_PSS_SALTLEN, \
233                                 len, NULL)

235 #define EVP_PKEY_CTX_get_rsa_pss_saltlen(ctx, plen) \
236         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, \
237                                 (EVP_PKEY_OP_SIGN|EVP_PKEY_OP_VERIFY), \
238                                 EVP_PKEY_CTRL_GET_RSA_PSS_SALTLEN, \
239                                 0, plen)

241 #define EVP_PKEY_CTX_set_rsa_keygen_bits(ctx, bits) \
242         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, EVP_PKEY_OP_KEYGEN, \
243                                 EVP_PKEY_CTRL_RSA_KEYGEN_BITS, bits, NULL)

245 #define EVP_PKEY_CTX_set_rsa_keygen_pubexp(ctx, pubexp) \
246         EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, EVP_PKEY_OP_KEYGEN, \
247                                 EVP_PKEY_CTRL_RSA_KEYGEN_PUBEXP, 0, pubexp)

249 #define  EVP_PKEY_CTX_set_rsa_mgf1_md(ctx, md)  \
250                 EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, EVP_PKEY_OP_TYPE_SIG,  \
251                                 EVP_PKEY_CTRL_RSA_MGF1_MD, 0, (void *)md)

253 #define  EVP_PKEY_CTX_get_rsa_mgf1_md(ctx, pmd) \
254                 EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_RSA, EVP_PKEY_OP_TYPE_SIG,  \
255                                 EVP_PKEY_CTRL_GET_RSA_MGF1_MD, 0, (void *)pmd)

257 #define EVP_PKEY_CTRL_RSA_PADDING       (EVP_PKEY_ALG_CTRL + 1)
258 #define EVP_PKEY_CTRL_RSA_PSS_SALTLEN   (EVP_PKEY_ALG_CTRL + 2)
```

```
#define EVP_PKEY_CTRL_RSA_KEYGEN_BITS   (EVP_PKEY_ALG_CTRL + 3)
#define EVP_PKEY_CTRL_RSA_KEYGEN_PUBEXP (EVP_PKEY_ALG_CTRL + 4)
#define EVP_PKEY_CTRL_RSA_MGF1_MD       (EVP_PKEY_ALG_CTRL + 5)

#define EVP_PKEY_CTRL_GET_RSA_PADDING           (EVP_PKEY_ALG_CTRL + 6)
#define EVP_PKEY_CTRL_GET_RSA_PSS_SALTLEN       (EVP_PKEY_ALG_CTRL + 7)
#define EVP_PKEY_CTRL_GET_RSA_MGF1_MD           (EVP_PKEY_ALG_CTRL + 8)

#define RSA_PKCS1_PADDING       1
#define RSA_SSLV23_PADDING      2
#define RSA_NO_PADDING          3
#define RSA_PKCS1_OAEP_PADDING  4
#define RSA_X931_PADDING        5
/* EVP_PKEY_ only */
#define RSA_PKCS1_PSS_PADDING   6

#define RSA_PKCS1_PADDING_SIZE  11

#define RSA_set_app_data(s,arg)         RSA_set_ex_data(s,0,arg)
#define RSA_get_app_data(s)             RSA_get_ex_data(s,0)

RSA *   RSA_new(void);
RSA *   RSA_new_method(ENGINE *engine);
int     RSA_size(const RSA *rsa);

/* Deprecated version */
#ifndef OPENSSL_NO_DEPRECATED
RSA *   RSA_generate_key(int bits, unsigned long e,void
                (*callback)(int,int,void *),void *cb_arg);
#endif /* !defined(OPENSSL_NO_DEPRECATED) */

/* New version */
int     RSA_generate_key_ex(RSA *rsa, int bits, BIGNUM *e, BN_GENCB *cb);

int     RSA_check_key(const RSA *);
        /* next 4 return -1 on error */
int     RSA_public_encrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
int     RSA_private_encrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
int     RSA_public_decrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
int     RSA_private_decrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
void    RSA_free (RSA *r);
/* "up" the RSA object's reference count */
int     RSA_up_ref(RSA *r);

int     RSA_flags(const RSA *r);

void RSA_set_default_method(const RSA_METHOD *meth);
const RSA_METHOD *RSA_get_default_method(void);
const RSA_METHOD *RSA_get_method(const RSA *rsa);
int RSA_set_method(RSA *rsa, const RSA_METHOD *meth);

/* This function needs the memory locking malloc callbacks to be installed */
int RSA_memory_lock(RSA *r);

/* these are the actual SSLeay RSA functions */
const RSA_METHOD *RSA_PKCS1_SSLeay(void);

const RSA_METHOD *RSA_null_method(void);

DECLARE_ASN1_ENCODE_FUNCTIONS_const(RSA, RSAPublicKey)
DECLARE_ASN1_ENCODE_FUNCTIONS_const(RSA, RSAPrivateKey)
```



```
typedef struct rsa_pss_params_st
        {
        X509_ALGOR *hashAlgorithm;
        X509_ALGOR *maskGenAlgorithm;
        ASN1_INTEGER *saltLength;
        ASN1_INTEGER *trailerField;
        } RSA_PSS_PARAMS;

DECLARE_ASN1_FUNCTIONS(RSA_PSS_PARAMS)

#ifndef OPENSSL_NO_FP_API
int     RSA_print_fp(FILE *fp, const RSA *r,int offset);
#endif

#ifndef OPENSSL_NO_BIO
int     RSA_print(BIO *bp, const RSA *r,int offset);
#endif

#ifndef OPENSSL_NO_RC4
int i2d_RSA_NET(const RSA *a, unsigned char **pp,
                int (*cb)(char *buf, int len, const char *prompt, int verify),
                int sgckey);
RSA *d2i_RSA_NET(RSA **a, const unsigned char **pp, long length,
                 int (*cb)(char *buf, int len, const char *prompt, int verify),
                 int sgckey);

int i2d_Netscape_RSA(const RSA *a, unsigned char **pp,
                     int (*cb)(char *buf, int len, const char *prompt,
                                int verify));
RSA *d2i_Netscape_RSA(RSA **a, const unsigned char **pp, long length,
                      int (*cb)(char *buf, int len, const char *prompt,
                                int verify));
#endif

/* The following 2 functions sign and verify a X509_SIG ASN1 object
 * inside PKCS#1 padded RSA encryption */
int RSA_sign(int type, const unsigned char *m, unsigned int m_length,
        unsigned char *sigret, unsigned int *siglen, RSA *rsa);
int RSA_verify(int type, const unsigned char *m, unsigned int m_length,
        const unsigned char *sigbuf, unsigned int siglen, RSA *rsa);

/* The following 2 function sign and verify a ASN1_OCTET_STRING
 * object inside PKCS#1 padded RSA encryption */
int RSA_sign_ASN1_OCTET_STRING(int type,
        const unsigned char *m, unsigned int m_length,
        unsigned char *sigret, unsigned int *siglen, RSA *rsa);
int RSA_verify_ASN1_OCTET_STRING(int type,
        const unsigned char *m, unsigned int m_length,
        unsigned char *sigbuf, unsigned int siglen, RSA *rsa);

int RSA_blinding_on(RSA *rsa, BN_CTX *ctx);
void RSA_blinding_off(RSA *rsa);
BN_BLINDING *RSA_setup_blinding(RSA *rsa, BN_CTX *ctx);

int RSA_padding_add_PKCS1_type_1(unsigned char *to,int tlen,
        const unsigned char *f,int fl);
int RSA_padding_check_PKCS1_type_1(unsigned char *to,int tlen,
        const unsigned char *f,int fl,int rsa_len);
int RSA_padding_add_PKCS1_type_2(unsigned char *to,int tlen,
        const unsigned char *f,int fl);
int RSA_padding_check_PKCS1_type_2(unsigned char *to,int tlen,
        const unsigned char *f,int fl,int rsa_len);
int PKCS1_MGF1(unsigned char *mask, long len,
        const unsigned char *seed, long seedlen, const EVP_MD *dgst);
int RSA_padding_add_PKCS1_OAEP(unsigned char *to,int tlen,
        const unsigned char *f,int fl,
```

```
392         const unsigned char *p,int pl);
393 int RSA_padding_check_PKCS1_OAEP(unsigned char *to,int tlen,
394         const unsigned char *f,int fl,int rsa_len,
395         const unsigned char *p,int pl);
396 int RSA_padding_add_SSLv23(unsigned char *to,int tlen,
397         const unsigned char *f,int fl);
398 int RSA_padding_check_SSLv23(unsigned char *to,int tlen,
399         const unsigned char *f,int fl,int rsa_len);
400 int RSA_padding_add_none(unsigned char *to,int tlen,
401         const unsigned char *f,int fl);
402 int RSA_padding_check_none(unsigned char *to,int tlen,
403         const unsigned char *f,int fl,int rsa_len);
404 int RSA_padding_add_X931(unsigned char *to,int tlen,
405         const unsigned char *f,int fl);
406 int RSA_padding_check_X931(unsigned char *to,int tlen,
407         const unsigned char *f,int fl,int rsa_len);
408 int RSA_X931_hash_id(int nid);

410 int RSA_verify_PKCS1_PSS(RSA *rsa, const unsigned char *mHash,
411                          const EVP_MD *Hash, const unsigned char *EM, int sLen);
412 int RSA_padding_add_PKCS1_PSS(RSA *rsa, unsigned char *EM,
413                          const unsigned char *mHash,
414                          const EVP_MD *Hash, int sLen);

416 int RSA_verify_PKCS1_PSS_mgf1(RSA *rsa, const unsigned char *mHash,
417                          const EVP_MD *Hash, const EVP_MD *mgf1Hash,
418                          const unsigned char *EM, int sLen);

420 int RSA_padding_add_PKCS1_PSS_mgf1(RSA *rsa, unsigned char *EM,
421                          const unsigned char *mHash,
422                          const EVP_MD *Hash, const EVP_MD *mgf1Hash, int sLen);

424 int RSA_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
425         CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
426 int RSA_set_ex_data(RSA *r,int idx,void *arg);
427 void *RSA_get_ex_data(const RSA *r, int idx);

429 RSA *RSAPublicKey_dup(RSA *rsa);
430 RSA *RSAPrivateKey_dup(RSA *rsa);

432 /* If this flag is set the RSA method is FIPS compliant and can be used
433  * in FIPS mode. This is set in the validated module method. If an
434  * application sets this flag in its own methods it is its responsibility
435  * to ensure the result is compliant.
436  */

438 #define RSA_FLAG_FIPS_METHOD                    0x0400

440 /* If this flag is set the operations normally disabled in FIPS mode are
441  * permitted it is then the applications responsibility to ensure that the
442  * usage is compliant.
443  */

445 #define RSA_FLAG_NON_FIPS_ALLOW                 0x0400
446 /* Application has decided PRNG is good enough to generate a key: don't
447  * check.
448  */
449 #define RSA_FLAG_CHECKED                        0x0800

451 /* BEGIN ERROR CODES */
452 /* The following lines are auto generated by the script mkerr.pl. Any changes
453  * made after this point may be overwritten when the script is next run.
454  */
455 void ERR_load_RSA_strings(void);

457 /* Error codes for the RSA functions. */
```



```
459 /* Function codes. */
460 #define RSA_F_CHECK_PADDING_MD                  140
461 #define RSA_F_DO_RSA_PRINT                      146
462 #define RSA_F_INT_RSA_VERIFY                    145
463 #define RSA_F_MEMORY_LOCK                       100
464 #define RSA_F_OLD_RSA_PRIV_DECODE               147
465 #define RSA_F_PKEY_RSA_CTRL                     143
466 #define RSA_F_PKEY_RSA_CTRL_STR                 144
467 #define RSA_F_PKEY_RSA_SIGN                     142
468 #define RSA_F_PKEY_RSA_VERIFY                   154
469 #define RSA_F_PKEY_RSA_VERIFYRECOVER            141
470 #define RSA_F_RSA_BUILTIN_KEYGEN                129
471 #define RSA_F_RSA_CHECK_KEY                     123
472 #define RSA_F_RSA_EAY_PRIVATE_DECRYPT           101
473 #define RSA_F_RSA_EAY_PRIVATE_ENCRYPT           102
474 #define RSA_F_RSA_EAY_PUBLIC_DECRYPT            103
475 #define RSA_F_RSA_EAY_PUBLIC_ENCRYPT            104
476 #define RSA_F_RSA_GENERATE_KEY                  105
477 #define RSA_F_RSA_GENERATE_KEY_EX               155
478 #define RSA_F_RSA_ITEM_VERIFY                   156
479 #define RSA_F_RSA_MEMORY_LOCK                   130
480 #define RSA_F_RSA_NEW_METHOD                    106
481 #define RSA_F_RSA_NULL                          124
482 #define RSA_F_RSA_NULL_MOD_EXP                  131
483 #define RSA_F_RSA_NULL_PRIVATE_DECRYPT          132
484 #define RSA_F_RSA_NULL_PRIVATE_ENCRYPT          133
485 #define RSA_F_RSA_NULL_PUBLIC_DECRYPT           134
486 #define RSA_F_RSA_NULL_PUBLIC_ENCRYPT           135
487 #define RSA_F_RSA_PADDING_ADD_NONE              107
488 #define RSA_F_RSA_PADDING_ADD_PKCS1_OAEP        121
489 #define RSA_F_RSA_PADDING_ADD_PKCS1_PSS         125
490 #define RSA_F_RSA_PADDING_ADD_PKCS1_PSS_MGF1    148
491 #define RSA_F_RSA_PADDING_ADD_PKCS1_TYPE_1      108
492 #define RSA_F_RSA_PADDING_ADD_PKCS1_TYPE_2      109
493 #define RSA_F_RSA_PADDING_ADD_SSLV23            110
494 #define RSA_F_RSA_PADDING_ADD_X931              127
495 #define RSA_F_RSA_PADDING_CHECK_NONE            111
496 #define RSA_F_RSA_PADDING_CHECK_PKCS1_OAEP      122
497 #define RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_1    112
498 #define RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2    113
499 #define RSA_F_RSA_PADDING_CHECK_SSLV23          114
500 #define RSA_F_RSA_PADDING_CHECK_X931            128
501 #define RSA_F_RSA_PRINT                         115
502 #define RSA_F_RSA_PRINT_FP                      116
503 #define RSA_F_RSA_PRIVATE_DECRYPT               150
504 #define RSA_F_RSA_PRIVATE_ENCRYPT               151
505 #define RSA_F_RSA_PRIV_DECODE                   137
506 #define RSA_F_RSA_PRIV_ENCODE                   138
507 #define RSA_F_RSA_PUBLIC_DECRYPT                152
508 #define RSA_F_RSA_PUBLIC_ENCRYPT                153
509 #define RSA_F_RSA_PUB_DECODE                    139
510 #define RSA_F_RSA_SETUP_BLINDING                136
511 #define RSA_F_RSA_SIGN                          117
512 #define RSA_F_RSA_SIGN_ASN1_OCTET_STRING        118
513 #define RSA_F_RSA_VERIFY                        119
514 #define RSA_F_RSA_VERIFY_ASN1_OCTET_STRING      120
515 #define RSA_F_RSA_VERIFY_PKCS1_PSS              126
516 #define RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1         149

518 /* Reason codes. */
519 #define RSA_R_ALGORITHM_MISMATCH                100
520 #define RSA_R_BAD_E_VALUE                       101
521 #define RSA_R_BAD_FIXED_HEADER_DECRYPT          102
522 #define RSA_R_BAD_PAD_BYTE_COUNT                103
523 #define RSA_R_BAD_SIGNATURE                     104
```

```
524 #define RSA_R_BLOCK_TYPE_IS_NOT_01                      106
525 #define RSA_R_BLOCK_TYPE_IS_NOT_02                      107
526 #define RSA_R_DATA_GREATER_THAN_MOD_LEN                 108
527 #define RSA_R_DATA_TOO_LARGE                            109
528 #define RSA_R_DATA_TOO_LARGE_FOR_KEY_SIZE               110
529 #define RSA_R_DATA_TOO_LARGE_FOR_MODULUS                132
530 #define RSA_R_DATA_TOO_SMALL                            111
531 #define RSA_R_DATA_TOO_SMALL_FOR_KEY_SIZE               122
532 #define RSA_R_DIGEST_TOO_BIG_FOR_RSA_KEY                112
533 #define RSA_R_DMP1_NOT_CONGRUENT_TO_D                   124
534 #define RSA_R_DMQ1_NOT_CONGRUENT_TO_D                   125
535 #define RSA_R_D_E_NOT_CONGRUENT_TO_1                    123
536 #define RSA_R_FIRST_OCTET_INVALID                       133
537 #define RSA_R_ILLEGAL_OR_UNSUPPORTED_PADDING_MODE       144
538 #define RSA_R_INVALID_DIGEST_LENGTH                     143
539 #define RSA_R_INVALID_HEADER                            137
540 #define RSA_R_INVALID_KEYBITS                           145
541 #define RSA_R_INVALID_MESSAGE_LENGTH                    131
542 #define RSA_R_INVALID_MGF1_MD                           156
543 #define RSA_R_INVALID_PADDING                           138
544 #define RSA_R_INVALID_PADDING_MODE                      141
545 #define RSA_R_INVALID_PSS_PARAMETERS                    149
546 #define RSA_R_INVALID_PSS_SALTLEN                       146
547 #define RSA_R_INVALID_SALT_LENGTH                       150
548 #define RSA_R_INVALID_TRAILER                           139
549 #define RSA_R_INVALID_X931_DIGEST                       142
550 #define RSA_R_IQMP_NOT_INVERSE_OF_Q                     126
551 #define RSA_R_KEY_SIZE_TOO_SMALL                        120
552 #define RSA_R_LAST_OCTET_INVALID                        134
553 #define RSA_R_MODULUS_TOO_LARGE                         105
554 #define RSA_R_NON_FIPS_RSA_METHOD                       157
555 #define RSA_R_NO_PUBLIC_EXPONENT                        140
556 #define RSA_R_NULL_BEFORE_BLOCK_MISSING                 113
557 #define RSA_R_N_DOES_NOT_EQUAL_P_Q                      127
558 #define RSA_R_OAEP_DECODING_ERROR                       121
559 #define RSA_R_OPERATION_NOT_ALLOWED_IN_FIPS_MODE        158
560 #define RSA_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE  148
561 #define RSA_R_PADDING_CHECK_FAILED                      114
562 #define RSA_R_P_NOT_PRIME                               128
563 #define RSA_R_Q_NOT_PRIME                               129
564 #define RSA_R_RSA_OPERATIONS_NOT_SUPPORTED              130
565 #define RSA_R_SLEN_CHECK_FAILED                         136
566 #define RSA_R_SLEN_RECOVERY_FAILED                      135
567 #define RSA_R_SSLV3_ROLLBACK_ATTACK                     115
568 #define RSA_R_THE_ASN1_OBJECT_IDENTIFIER_IS_NOT_KNOWN_FOR_THIS_MD 116
569 #define RSA_R_UNKNOWN_ALGORITHM_TYPE                    117
570 #define RSA_R_UNKNOWN_MASK_DIGEST                       151
571 #define RSA_R_UNKNOWN_PADDING_TYPE                      118
572 #define RSA_R_UNKNOWN_PSS_DIGEST                        152
573 #define RSA_R_UNSUPPORTED_MASK_ALGORITHM                153
574 #define RSA_R_UNSUPPORTED_MASK_PARAMETER                154
575 #define RSA_R_UNSUPPORTED_SIGNATURE_TYPE                155
576 #define RSA_R_VALUE_MISSING                             147
577 #define RSA_R_WRONG_SIGNATURE_LENGTH                    119

579 #ifdef  __cplusplus
580 }
581 #endif
582 #endif
583 #endif /* ! codereview */
```

```
**********************************************************
  184128 Wed Aug 13 19:51:48 2014
new/usr/src/lib/openssl/include/openssl/safestack.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_SAFESTACK_H
#define HEADER_SAFESTACK_H

#include <openssl/stack.h>

#ifndef CHECKED_PTR_OF
#define CHECKED_PTR_OF(type, p) \
```



```
    ((void*) (1 ? p : (type*)0))
#endif

/* In C++ we get problems because an explicit cast is needed from (void *)
 * we use CHECKED_STACK_OF to ensure the correct type is passed in the macros
 * below.
 */

#define CHECKED_STACK_OF(type, p) \
    ((_STACK*) (1 ? p : (STACK_OF(type)*)0))

#define CHECKED_SK_FREE_FUNC(type, p) \
    ((void (*)(void *)) ((1 ? p : (void (*)(type *))0)))

#define CHECKED_SK_FREE_FUNC2(type, p) \
    ((void (*)(void *)) ((1 ? p : (void (*)(type))0)))

#define CHECKED_SK_CMP_FUNC(type, p) \
    ((int (*)(const void *, const void *)) \
        ((1 ? p : (int (*)(const type * const *, const type * const *))0)))

#define STACK_OF(type) struct stack_st_##type
#define PREDECLARE_STACK_OF(type) STACK_OF(type);

#define DECLARE_STACK_OF(type) \
STACK_OF(type) \
    { \
    _STACK stack; \
    };
#define DECLARE_SPECIAL_STACK_OF(type, type2) \
STACK_OF(type) \
    { \
    _STACK stack; \
    };

#define IMPLEMENT_STACK_OF(type) /* nada (obsolete in new safestack approach)*/


/* Strings are special: normally an lhash entry will point to a single
 * (somewhat) mutable object. In the case of strings:
 *
 * a) Instead of a single char, there is an array of chars, NUL-terminated.
 * b) The string may have be immutable.
 *
 * So, they need their own declarations. Especially important for
 * type-checking tools, such as Deputy.
 *
o * In practice, however, it appears to be hard to have a const
 * string. For now, I'm settling for dealing with the fact it is a
 * string at all.
 */
typedef char *OPENSSL_STRING;

typedef const char *OPENSSL_CSTRING;

/* Confusingly, LHASH_OF(STRING) deals with char ** throughout, but
 * STACK_OF(STRING) is really more like STACK_OF(char), only, as
 * mentioned above, instead of a single char each entry is a
 * NUL-terminated array of chars. So, we have to implement STRING
 * specially for STACK_OF. This is dealt with in the autogenerated
 * macros below.
 */

DECLARE_SPECIAL_STACK_OF(OPENSSL_STRING, char)

/* Similarly, we sometimes use a block of characters, NOT
```

```
 * nul-terminated. These should also be distinguished from "normal"
 * stacks. */

typedef void *OPENSSL_BLOCK;
DECLARE_SPECIAL_STACK_OF(OPENSSL_BLOCK, void)

/* SKM_sk_... stack macros are internal to safestack.h:
 * never use them directly, use sk_<type>_... instead */
#define SKM_sk_new(type, cmp) \
        ((STACK_OF(type) *)sk_new(CHECKED_SK_CMP_FUNC(type, cmp)))
#define SKM_sk_new_null(type) \
        ((STACK_OF(type) *)sk_new_null())
#define SKM_sk_free(type, st) \
        sk_free(CHECKED_STACK_OF(type, st))
#define SKM_sk_num(type, st) \
        sk_num(CHECKED_STACK_OF(type, st))
#define SKM_sk_value(type, st,i) \
        ((type *)sk_value(CHECKED_STACK_OF(type, st), i))
#define SKM_sk_set(type, st,i,val) \
        sk_set(CHECKED_STACK_OF(type, st), i, CHECKED_PTR_OF(type, val))
#define SKM_sk_zero(type, st) \
        sk_zero(CHECKED_STACK_OF(type, st))
#define SKM_sk_push(type, st, val) \
        sk_push(CHECKED_STACK_OF(type, st), CHECKED_PTR_OF(type, val))
#define SKM_sk_unshift(type, st, val) \
        sk_unshift(CHECKED_STACK_OF(type, st), CHECKED_PTR_OF(type, val))
#define SKM_sk_find(type, st, val) \
        sk_find(CHECKED_STACK_OF(type, st), CHECKED_PTR_OF(type, val))
#define SKM_sk_find_ex(type, st, val) \
        sk_find_ex(CHECKED_STACK_OF(type, st), \
                   CHECKED_PTR_OF(type, val))
#define SKM_sk_delete(type, st, i) \
        (type *)sk_delete(CHECKED_STACK_OF(type, st), i)
#define SKM_sk_delete_ptr(type, st, ptr) \
        (type *)sk_delete_ptr(CHECKED_STACK_OF(type, st), CHECKED_PTR_OF(type, p
#define SKM_sk_insert(type, st,val, i) \
        sk_insert(CHECKED_STACK_OF(type, st), CHECKED_PTR_OF(type, val), i)
#define SKM_sk_set_cmp_func(type, st, cmp) \
        ((int (*)(const type * const *,const type * const *)) \
        sk_set_cmp_func(CHECKED_STACK_OF(type, st), CHECKED_SK_CMP_FUNC(type, cm
#define SKM_sk_dup(type, st) \
        (STACK_OF(type) *)sk_dup(CHECKED_STACK_OF(type, st))
#define SKM_sk_pop_free(type, st, free_func) \
        sk_pop_free(CHECKED_STACK_OF(type, st), CHECKED_SK_FREE_FUNC(type, free_
#define SKM_sk_shift(type, st) \
        (type *)sk_shift(CHECKED_STACK_OF(type, st))
#define SKM_sk_pop(type, st) \
        (type *)sk_pop(CHECKED_STACK_OF(type, st))
#define SKM_sk_sort(type, st) \
        sk_sort(CHECKED_STACK_OF(type, st))
#define SKM_sk_is_sorted(type, st) \
        sk_is_sorted(CHECKED_STACK_OF(type, st))

#define SKM_ASN1_SET_OF_d2i(type, st, pp, length, d2i_func, free_func, ex_tag, e
  (STACK_OF(type) *)d2i_ASN1_SET( \
                                (STACK_OF(OPENSSL_BLOCK) **)CHECKED_PTR_OF(STACK
                                pp, length, \
                                CHECKED_D2I_OF(type, d2i_func), \
                                CHECKED_SK_FREE_FUNC(type, free_func), \
                                ex_tag, ex_class)

#define SKM_ASN1_SET_OF_i2d(type, st, pp, i2d_func, ex_tag, ex_class, is_set) \
  i2d_ASN1_SET((STACK_OF(OPENSSL_BLOCK) *)CHECKED_STACK_OF(type, st), pp, \
                                CHECKED_I2D_OF(type, i2d_func), \
                                ex_tag, ex_class, is_set)
```



```
#define SKM_ASN1_seq_pack(type, st, i2d_func, buf, len) \
        ASN1_seq_pack(CHECKED_PTR_OF(STACK_OF(type), st), \
                      CHECKED_I2D_OF(type, i2d_func), buf, len)

#define SKM_ASN1_seq_unpack(type, buf, len, d2i_func, free_func) \
        (STACK_OF(type) *)ASN1_seq_unpack(buf, len, CHECKED_D2I_OF(type, d2i_fun

#define SKM_PKCS12_decrypt_d2i(type, algor, d2i_func, free_func, pass, passlen,
        (STACK_OF(type) *)PKCS12_decrypt_d2i(algor, \
                                CHECKED_D2I_OF(type, d2i_func), \
                                CHECKED_SK_FREE_FUNC(type, free_func), \
                                pass, passlen, oct, seq)

/* This block of defines is updated by util/mkstack.pl, please do not touch! */
#define sk_ACCESS_DESCRIPTION_new(cmp) SKM_sk_new(ACCESS_DESCRIPTION, (cmp))
#define sk_ACCESS_DESCRIPTION_new_null() SKM_sk_new_null(ACCESS_DESCRIPTION)
#define sk_ACCESS_DESCRIPTION_free(st) SKM_sk_free(ACCESS_DESCRIPTION, (st))
#define sk_ACCESS_DESCRIPTION_num(st) SKM_sk_num(ACCESS_DESCRIPTION, (st))
#define sk_ACCESS_DESCRIPTION_value(st, i) SKM_sk_value(ACCESS_DESCRIPTION, (st)
#define sk_ACCESS_DESCRIPTION_set(st, i, val) SKM_sk_set(ACCESS_DESCRIPTION, (st
#define sk_ACCESS_DESCRIPTION_zero(st) SKM_sk_zero(ACCESS_DESCRIPTION, (st))
#define sk_ACCESS_DESCRIPTION_push(st, val) SKM_sk_push(ACCESS_DESCRIPTION, (st)
#define sk_ACCESS_DESCRIPTION_unshift(st, val) SKM_sk_unshift(ACCESS_DESCRIPTION
#define sk_ACCESS_DESCRIPTION_find(st, val) SKM_sk_find(ACCESS_DESCRIPTION, (st)
#define sk_ACCESS_DESCRIPTION_find_ex(st, val) SKM_sk_find_ex(ACCESS_DESCRIPTION
#define sk_ACCESS_DESCRIPTION_delete(st, i) SKM_sk_delete(ACCESS_DESCRIPTION, (s
#define sk_ACCESS_DESCRIPTION_delete_ptr(st, ptr) SKM_sk_delete_ptr(ACCESS_DESCR
#define sk_ACCESS_DESCRIPTION_insert(st, val, i) SKM_sk_insert(ACCESS_DESCRIPTIO
#define sk_ACCESS_DESCRIPTION_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ACCESS_D
#define sk_ACCESS_DESCRIPTION_dup(st) SKM_sk_dup(ACCESS_DESCRIPTION, st)
#define sk_ACCESS_DESCRIPTION_pop_free(st, free_func) SKM_sk_pop_free(ACCESS_DES
#define sk_ACCESS_DESCRIPTION_shift(st) SKM_sk_shift(ACCESS_DESCRIPTION, (st))
#define sk_ACCESS_DESCRIPTION_pop(st) SKM_sk_pop(ACCESS_DESCRIPTION, (st))
#define sk_ACCESS_DESCRIPTION_sort(st) SKM_sk_sort(ACCESS_DESCRIPTION, (st))
#define sk_ACCESS_DESCRIPTION_is_sorted(st) SKM_sk_is_sorted(ACCESS_DESCRIPTION,

#define sk_ASIdOrRange_new(cmp) SKM_sk_new(ASIdOrRange, (cmp))
#define sk_ASIdOrRange_new_null() SKM_sk_new_null(ASIdOrRange)
#define sk_ASIdOrRange_free(st) SKM_sk_free(ASIdOrRange, (st))
#define sk_ASIdOrRange_num(st) SKM_sk_num(ASIdOrRange, (st))
#define sk_ASIdOrRange_value(st, i) SKM_sk_value(ASIdOrRange, (st), (i))
#define sk_ASIdOrRange_set(st, i, val) SKM_sk_set(ASIdOrRange, (st), (i), (val))
#define sk_ASIdOrRange_zero(st) SKM_sk_zero(ASIdOrRange, (st))
#define sk_ASIdOrRange_push(st, val) SKM_sk_push(ASIdOrRange, (st), (val))
#define sk_ASIdOrRange_unshift(st, val) SKM_sk_unshift(ASIdOrRange, (st), (val))
#define sk_ASIdOrRange_find(st, val) SKM_sk_find(ASIdOrRange, (st), (val))
#define sk_ASIdOrRange_find_ex(st, val) SKM_sk_find_ex(ASIdOrRange, (st), (val))
#define sk_ASIdOrRange_delete(st, i) SKM_sk_delete(ASIdOrRange, (st), (i))
#define sk_ASIdOrRange_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASIdOrRange, (st),
#define sk_ASIdOrRange_insert(st, val, i) SKM_sk_insert(ASIdOrRange, (st), (val)
#define sk_ASIdOrRange_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASIdOrRange, (s
#define sk_ASIdOrRange_dup(st) SKM_sk_dup(ASIdOrRange, st)
#define sk_ASIdOrRange_pop_free(st, free_func) SKM_sk_pop_free(ASIdOrRange, (st)
#define sk_ASIdOrRange_shift(st) SKM_sk_shift(ASIdOrRange, (st))
#define sk_ASIdOrRange_pop(st) SKM_sk_pop(ASIdOrRange, (st))
#define sk_ASIdOrRange_sort(st) SKM_sk_sort(ASIdOrRange, (st))
#define sk_ASIdOrRange_is_sorted(st) SKM_sk_is_sorted(ASIdOrRange, (st))

#define sk_ASN1_GENERALSTRING_new(cmp) SKM_sk_new(ASN1_GENERALSTRING, (cmp))
#define sk_ASN1_GENERALSTRING_new_null() SKM_sk_new_null(ASN1_GENERALSTRING)
#define sk_ASN1_GENERALSTRING_free(st) SKM_sk_free(ASN1_GENERALSTRING, (st))
#define sk_ASN1_GENERALSTRING_num(st) SKM_sk_num(ASN1_GENERALSTRING, (st))
#define sk_ASN1_GENERALSTRING_value(st, i) SKM_sk_value(ASN1_GENERALSTRING, (st)
#define sk_ASN1_GENERALSTRING_set(st, i, val) SKM_sk_set(ASN1_GENERALSTRING, (st
#define sk_ASN1_GENERALSTRING_zero(st) SKM_sk_zero(ASN1_GENERALSTRING, (st))
#define sk_ASN1_GENERALSTRING_push(st, val) SKM_sk_push(ASN1_GENERALSTRING, (st)
```

```
#define sk_ASN1_GENERALSTRING_unshift(st, val) SKM_sk_unshift(ASN1_GENERALSTRING
#define sk_ASN1_GENERALSTRING_find(st, val) SKM_sk_find(ASN1_GENERALSTRING, (st)
#define sk_ASN1_GENERALSTRING_find_ex(st, val) SKM_sk_find_ex(ASN1_GENERALSTRING
#define sk_ASN1_GENERALSTRING_delete(st, i) SKM_sk_delete(ASN1_GENERALSTRING, (s
#define sk_ASN1_GENERALSTRING_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASN1_GENERAL
#define sk_ASN1_GENERALSTRING_insert(st, val, i) SKM_sk_insert(ASN1_GENERALSTRIN
#define sk_ASN1_GENERALSTRING_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASN1_GEN
#define sk_ASN1_GENERALSTRING_dup(st) SKM_sk_dup(ASN1_GENERALSTRING, st)
#define sk_ASN1_GENERALSTRING_pop_free(st, free_func) SKM_sk_pop_free(ASN1_GENER
#define sk_ASN1_GENERALSTRING_shift(st) SKM_sk_shift(ASN1_GENERALSTRING, (st))
#define sk_ASN1_GENERALSTRING_pop(st) SKM_sk_pop(ASN1_GENERALSTRING, (st))
#define sk_ASN1_GENERALSTRING_sort(st) SKM_sk_sort(ASN1_GENERALSTRING, (st))
#define sk_ASN1_GENERALSTRING_is_sorted(st) SKM_sk_is_sorted(ASN1_GENERALSTRING,

#define sk_ASN1_INTEGER_new(cmp) SKM_sk_new(ASN1_INTEGER, (cmp))
#define sk_ASN1_INTEGER_new_null() SKM_sk_new_null(ASN1_INTEGER)
#define sk_ASN1_INTEGER_free(st) SKM_sk_free(ASN1_INTEGER, (st))
#define sk_ASN1_INTEGER_num(st) SKM_sk_num(ASN1_INTEGER, (st))
#define sk_ASN1_INTEGER_value(st, i) SKM_sk_value(ASN1_INTEGER, (st), (i))
#define sk_ASN1_INTEGER_set(st, i, val) SKM_sk_set(ASN1_INTEGER, (st), (i), (val
#define sk_ASN1_INTEGER_zero(st) SKM_sk_zero(ASN1_INTEGER, (st))
#define sk_ASN1_INTEGER_push(st, val) SKM_sk_push(ASN1_INTEGER, (st), (val))
#define sk_ASN1_INTEGER_unshift(st, val) SKM_sk_unshift(ASN1_INTEGER, (st), (val
#define sk_ASN1_INTEGER_find(st, val) SKM_sk_find(ASN1_INTEGER, (st), (val))
#define sk_ASN1_INTEGER_find_ex(st, val) SKM_sk_find_ex(ASN1_INTEGER, (st), (val
#define sk_ASN1_INTEGER_delete(st, i) SKM_sk_delete(ASN1_INTEGER, (st), (i))
#define sk_ASN1_INTEGER_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASN1_INTEGER, (st)
#define sk_ASN1_INTEGER_insert(st, val, i) SKM_sk_insert(ASN1_INTEGER, (st), (va
#define sk_ASN1_INTEGER_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASN1_INTEGER,
#define sk_ASN1_INTEGER_dup(st) SKM_sk_dup(ASN1_INTEGER, st)
#define sk_ASN1_INTEGER_pop_free(st, free_func) SKM_sk_pop_free(ASN1_INTEGER, (s
#define sk_ASN1_INTEGER_shift(st) SKM_sk_shift(ASN1_INTEGER, (st))
#define sk_ASN1_INTEGER_pop(st) SKM_sk_pop(ASN1_INTEGER, (st))
#define sk_ASN1_INTEGER_sort(st) SKM_sk_sort(ASN1_INTEGER, (st))
#define sk_ASN1_INTEGER_is_sorted(st) SKM_sk_is_sorted(ASN1_INTEGER, (st))

#define sk_ASN1_OBJECT_new(cmp) SKM_sk_new(ASN1_OBJECT, (cmp))
#define sk_ASN1_OBJECT_new_null() SKM_sk_new_null(ASN1_OBJECT)
#define sk_ASN1_OBJECT_free(st) SKM_sk_free(ASN1_OBJECT, (st))
#define sk_ASN1_OBJECT_num(st) SKM_sk_num(ASN1_OBJECT, (st))
#define sk_ASN1_OBJECT_value(st, i) SKM_sk_value(ASN1_OBJECT, (st), (i))
#define sk_ASN1_OBJECT_set(st, i, val) SKM_sk_set(ASN1_OBJECT, (st), (i), (val))
#define sk_ASN1_OBJECT_zero(st) SKM_sk_zero(ASN1_OBJECT, (st))
#define sk_ASN1_OBJECT_push(st, val) SKM_sk_push(ASN1_OBJECT, (st), (val))
#define sk_ASN1_OBJECT_unshift(st, val) SKM_sk_unshift(ASN1_OBJECT, (st), (val))
#define sk_ASN1_OBJECT_find(st, val) SKM_sk_find(ASN1_OBJECT, (st), (val))
#define sk_ASN1_OBJECT_find_ex(st, val) SKM_sk_find_ex(ASN1_OBJECT, (st), (val))
#define sk_ASN1_OBJECT_delete(st, i) SKM_sk_delete(ASN1_OBJECT, (st), (i))
#define sk_ASN1_OBJECT_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASN1_OBJECT, (st),
#define sk_ASN1_OBJECT_insert(st, val, i) SKM_sk_insert(ASN1_OBJECT, (st), (val)
#define sk_ASN1_OBJECT_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASN1_OBJECT, (s
#define sk_ASN1_OBJECT_dup(st) SKM_sk_dup(ASN1_OBJECT, st)
#define sk_ASN1_OBJECT_pop_free(st, free_func) SKM_sk_pop_free(ASN1_OBJECT, (st)
#define sk_ASN1_OBJECT_shift(st) SKM_sk_shift(ASN1_OBJECT, (st))
#define sk_ASN1_OBJECT_pop(st) SKM_sk_pop(ASN1_OBJECT, (st))
#define sk_ASN1_OBJECT_sort(st) SKM_sk_sort(ASN1_OBJECT, (st))
#define sk_ASN1_OBJECT_is_sorted(st) SKM_sk_is_sorted(ASN1_OBJECT, (st))

#define sk_ASN1_STRING_TABLE_new(cmp) SKM_sk_new(ASN1_STRING_TABLE, (cmp))
#define sk_ASN1_STRING_TABLE_new_null() SKM_sk_new_null(ASN1_STRING_TABLE)
#define sk_ASN1_STRING_TABLE_free(st) SKM_sk_free(ASN1_STRING_TABLE, (st))
#define sk_ASN1_STRING_TABLE_num(st) SKM_sk_num(ASN1_STRING_TABLE, (st))
#define sk_ASN1_STRING_TABLE_value(st, i) SKM_sk_value(ASN1_STRING_TABLE, (st),
#define sk_ASN1_STRING_TABLE_set(st, i, val) SKM_sk_set(ASN1_STRING_TABLE, (st),
#define sk_ASN1_STRING_TABLE_zero(st) SKM_sk_zero(ASN1_STRING_TABLE, (st))
#define sk_ASN1_STRING_TABLE_push(st, val) SKM_sk_push(ASN1_STRING_TABLE, (st),
```



```
#define sk_ASN1_STRING_TABLE_unshift(st, val) SKM_sk_unshift(ASN1_STRING_TABLE,
#define sk_ASN1_STRING_TABLE_find(st, val) SKM_sk_find(ASN1_STRING_TABLE, (st),
#define sk_ASN1_STRING_TABLE_find_ex(st, val) SKM_sk_find_ex(ASN1_STRING_TABLE,
#define sk_ASN1_STRING_TABLE_delete(st, i) SKM_sk_delete(ASN1_STRING_TABLE, (st)
#define sk_ASN1_STRING_TABLE_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASN1_STRING_T
#define sk_ASN1_STRING_TABLE_insert(st, val, i) SKM_sk_insert(ASN1_STRING_TABLE,
#define sk_ASN1_STRING_TABLE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASN1_STRI
#define sk_ASN1_STRING_TABLE_dup(st) SKM_sk_dup(ASN1_STRING_TABLE, st)
#define sk_ASN1_STRING_TABLE_pop_free(st, free_func) SKM_sk_pop_free(ASN1_STRING
#define sk_ASN1_STRING_TABLE_shift(st) SKM_sk_shift(ASN1_STRING_TABLE, (st))
#define sk_ASN1_STRING_TABLE_pop(st) SKM_sk_pop(ASN1_STRING_TABLE, (st))
#define sk_ASN1_STRING_TABLE_sort(st) SKM_sk_sort(ASN1_STRING_TABLE, (st))
#define sk_ASN1_STRING_TABLE_is_sorted(st) SKM_sk_is_sorted(ASN1_STRING_TABLE, (

#define sk_ASN1_TYPE_new(cmp) SKM_sk_new(ASN1_TYPE, (cmp))
#define sk_ASN1_TYPE_new_null() SKM_sk_new_null(ASN1_TYPE)
#define sk_ASN1_TYPE_free(st) SKM_sk_free(ASN1_TYPE, (st))
#define sk_ASN1_TYPE_num(st) SKM_sk_num(ASN1_TYPE, (st))
#define sk_ASN1_TYPE_value(st, i) SKM_sk_value(ASN1_TYPE, (st), (i))
#define sk_ASN1_TYPE_set(st, i, val) SKM_sk_set(ASN1_TYPE, (st), (i), (val))
#define sk_ASN1_TYPE_zero(st) SKM_sk_zero(ASN1_TYPE, (st))
#define sk_ASN1_TYPE_push(st, val) SKM_sk_push(ASN1_TYPE, (st), (val))
#define sk_ASN1_TYPE_unshift(st, val) SKM_sk_unshift(ASN1_TYPE, (st), (val))
#define sk_ASN1_TYPE_find(st, val) SKM_sk_find(ASN1_TYPE, (st), (val))
#define sk_ASN1_TYPE_find_ex(st, val) SKM_sk_find_ex(ASN1_TYPE, (st), (val))
#define sk_ASN1_TYPE_delete(st, i) SKM_sk_delete(ASN1_TYPE, (st), (i))
#define sk_ASN1_TYPE_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASN1_TYPE, (st), (ptr
#define sk_ASN1_TYPE_insert(st, val, i) SKM_sk_insert(ASN1_TYPE, (st), (val), (i
#define sk_ASN1_TYPE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASN1_TYPE, (st),
#define sk_ASN1_TYPE_dup(st) SKM_sk_dup(ASN1_TYPE, st)
#define sk_ASN1_TYPE_pop_free(st, free_func) SKM_sk_pop_free(ASN1_TYPE, (st), (f
#define sk_ASN1_TYPE_shift(st) SKM_sk_shift(ASN1_TYPE, (st))
#define sk_ASN1_TYPE_pop(st) SKM_sk_pop(ASN1_TYPE, (st))
#define sk_ASN1_TYPE_sort(st) SKM_sk_sort(ASN1_TYPE, (st))
#define sk_ASN1_TYPE_is_sorted(st) SKM_sk_is_sorted(ASN1_TYPE, (st))

#define sk_ASN1_UTF8STRING_new(cmp) SKM_sk_new(ASN1_UTF8STRING, (cmp))
#define sk_ASN1_UTF8STRING_new_null() SKM_sk_new_null(ASN1_UTF8STRING)
#define sk_ASN1_UTF8STRING_free(st) SKM_sk_free(ASN1_UTF8STRING, (st))
#define sk_ASN1_UTF8STRING_num(st) SKM_sk_num(ASN1_UTF8STRING, (st))
#define sk_ASN1_UTF8STRING_value(st, i) SKM_sk_value(ASN1_UTF8STRING, (st), (i))
#define sk_ASN1_UTF8STRING_set(st, i, val) SKM_sk_set(ASN1_UTF8STRING, (st), (i)
#define sk_ASN1_UTF8STRING_zero(st) SKM_sk_zero(ASN1_UTF8STRING, (st))
#define sk_ASN1_UTF8STRING_push(st, val) SKM_sk_push(ASN1_UTF8STRING, (st), (val
#define sk_ASN1_UTF8STRING_unshift(st, val) SKM_sk_unshift(ASN1_UTF8STRING, (st)
#define sk_ASN1_UTF8STRING_find(st, val) SKM_sk_find(ASN1_UTF8STRING, (st), (val
#define sk_ASN1_UTF8STRING_find_ex(st, val) SKM_sk_find_ex(ASN1_UTF8STRING, (st)
#define sk_ASN1_UTF8STRING_delete(st, i) SKM_sk_delete(ASN1_UTF8STRING, (st), (i
#define sk_ASN1_UTF8STRING_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASN1_UTF8STRING
#define sk_ASN1_UTF8STRING_insert(st, val, i) SKM_sk_insert(ASN1_UTF8STRING, (st
#define sk_ASN1_UTF8STRING_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASN1_UTF8ST
#define sk_ASN1_UTF8STRING_dup(st) SKM_sk_dup(ASN1_UTF8STRING, st)
#define sk_ASN1_UTF8STRING_pop_free(st, free_func) SKM_sk_pop_free(ASN1_UTF8STRI
#define sk_ASN1_UTF8STRING_shift(st) SKM_sk_shift(ASN1_UTF8STRING, (st))
#define sk_ASN1_UTF8STRING_pop(st) SKM_sk_pop(ASN1_UTF8STRING, (st))
#define sk_ASN1_UTF8STRING_sort(st) SKM_sk_sort(ASN1_UTF8STRING, (st))
#define sk_ASN1_UTF8STRING_is_sorted(st) SKM_sk_is_sorted(ASN1_UTF8STRING, (st))

#define sk_ASN1_VALUE_new(cmp) SKM_sk_new(ASN1_VALUE, (cmp))
#define sk_ASN1_VALUE_new_null() SKM_sk_new_null(ASN1_VALUE)
#define sk_ASN1_VALUE_free(st) SKM_sk_free(ASN1_VALUE, (st))
#define sk_ASN1_VALUE_num(st) SKM_sk_num(ASN1_VALUE, (st))
#define sk_ASN1_VALUE_value(st, i) SKM_sk_value(ASN1_VALUE, (st), (i))
#define sk_ASN1_VALUE_set(st, i, val) SKM_sk_set(ASN1_VALUE, (st), (i), (val))
#define sk_ASN1_VALUE_zero(st) SKM_sk_zero(ASN1_VALUE, (st))
#define sk_ASN1_VALUE_push(st, val) SKM_sk_push(ASN1_VALUE, (st), (val))
```

```
392 #define sk_ASN1_VALUE_unshift(st, val) SKM_sk_unshift(ASN1_VALUE, (st), (val))
393 #define sk_ASN1_VALUE_find(st, val) SKM_sk_find(ASN1_VALUE, (st), (val))
394 #define sk_ASN1_VALUE_find_ex(st, val) SKM_sk_find_ex(ASN1_VALUE, (st), (val))
395 #define sk_ASN1_VALUE_delete(st, i) SKM_sk_delete(ASN1_VALUE, (st), (i))
396 #define sk_ASN1_VALUE_delete_ptr(st, ptr) SKM_sk_delete_ptr(ASN1_VALUE, (st), (p
397 #define sk_ASN1_VALUE_insert(st, val, i) SKM_sk_insert(ASN1_VALUE, (st), (val),
398 #define sk_ASN1_VALUE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ASN1_VALUE, (st)
399 #define sk_ASN1_VALUE_dup(st) SKM_sk_dup(ASN1_VALUE, st)
400 #define sk_ASN1_VALUE_pop_free(st, free_func) SKM_sk_pop_free(ASN1_VALUE, (st),
401 #define sk_ASN1_VALUE_shift(st) SKM_sk_shift(ASN1_VALUE, (st))
402 #define sk_ASN1_VALUE_pop(st) SKM_sk_pop(ASN1_VALUE, (st))
403 #define sk_ASN1_VALUE_sort(st) SKM_sk_sort(ASN1_VALUE, (st))
404 #define sk_ASN1_VALUE_is_sorted(st) SKM_sk_is_sorted(ASN1_VALUE, (st))

406 #define sk_BIO_new(cmp) SKM_sk_new(BIO, (cmp))
407 #define sk_BIO_new_null() SKM_sk_new_null(BIO)
408 #define sk_BIO_free(st) SKM_sk_free(BIO, (st))
409 #define sk_BIO_num(st) SKM_sk_num(BIO, (st))
410 #define sk_BIO_value(st, i) SKM_sk_value(BIO, (st), (i))
411 #define sk_BIO_set(st, i, val) SKM_sk_set(BIO, (st), (i), (val))
412 #define sk_BIO_zero(st) SKM_sk_zero(BIO, (st))
413 #define sk_BIO_push(st, val) SKM_sk_push(BIO, (st), (val))
414 #define sk_BIO_unshift(st, val) SKM_sk_unshift(BIO, (st), (val))
415 #define sk_BIO_find(st, val) SKM_sk_find(BIO, (st), (val))
416 #define sk_BIO_find_ex(st, val) SKM_sk_find_ex(BIO, (st), (val))
417 #define sk_BIO_delete(st, i) SKM_sk_delete(BIO, (st), (i))
418 #define sk_BIO_delete_ptr(st, ptr) SKM_sk_delete_ptr(BIO, (st), (ptr))
419 #define sk_BIO_insert(st, val, i) SKM_sk_insert(BIO, (st), (val), (i))
420 #define sk_BIO_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(BIO, (st), (cmp))
421 #define sk_BIO_dup(st) SKM_sk_dup(BIO, st)
422 #define sk_BIO_pop_free(st, free_func) SKM_sk_pop_free(BIO, (st), (free_func))
423 #define sk_BIO_shift(st) SKM_sk_shift(BIO, (st))
424 #define sk_BIO_pop(st) SKM_sk_pop(BIO, (st))
425 #define sk_BIO_sort(st) SKM_sk_sort(BIO, (st))
426 #define sk_BIO_is_sorted(st) SKM_sk_is_sorted(BIO, (st))

428 #define sk_BY_DIR_ENTRY_new(cmp) SKM_sk_new(BY_DIR_ENTRY, (cmp))
429 #define sk_BY_DIR_ENTRY_new_null() SKM_sk_new_null(BY_DIR_ENTRY)
430 #define sk_BY_DIR_ENTRY_free(st) SKM_sk_free(BY_DIR_ENTRY, (st))
431 #define sk_BY_DIR_ENTRY_num(st) SKM_sk_num(BY_DIR_ENTRY, (st))
432 #define sk_BY_DIR_ENTRY_value(st, i) SKM_sk_value(BY_DIR_ENTRY, (st), (i))
433 #define sk_BY_DIR_ENTRY_set(st, i, val) SKM_sk_set(BY_DIR_ENTRY, (st), (i), (val
434 #define sk_BY_DIR_ENTRY_zero(st) SKM_sk_zero(BY_DIR_ENTRY, (st))
435 #define sk_BY_DIR_ENTRY_push(st, val) SKM_sk_push(BY_DIR_ENTRY, (st), (val))
436 #define sk_BY_DIR_ENTRY_unshift(st, val) SKM_sk_unshift(BY_DIR_ENTRY, (st), (val
437 #define sk_BY_DIR_ENTRY_find(st, val) SKM_sk_find(BY_DIR_ENTRY, (st), (val))
438 #define sk_BY_DIR_ENTRY_find_ex(st, val) SKM_sk_find_ex(BY_DIR_ENTRY, (st), (val
439 #define sk_BY_DIR_ENTRY_delete(st, i) SKM_sk_delete(BY_DIR_ENTRY, (st), (i))
440 #define sk_BY_DIR_ENTRY_delete_ptr(st, ptr) SKM_sk_delete_ptr(BY_DIR_ENTRY, (st)
441 #define sk_BY_DIR_ENTRY_insert(st, val, i) SKM_sk_insert(BY_DIR_ENTRY, (st), (va
442 #define sk_BY_DIR_ENTRY_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(BY_DIR_ENTRY,
443 #define sk_BY_DIR_ENTRY_dup(st) SKM_sk_dup(BY_DIR_ENTRY, st)
444 #define sk_BY_DIR_ENTRY_pop_free(st, free_func) SKM_sk_pop_free(BY_DIR_ENTRY, (s
445 #define sk_BY_DIR_ENTRY_shift(st) SKM_sk_shift(BY_DIR_ENTRY, (st))
446 #define sk_BY_DIR_ENTRY_pop(st) SKM_sk_pop(BY_DIR_ENTRY, (st))
447 #define sk_BY_DIR_ENTRY_sort(st) SKM_sk_sort(BY_DIR_ENTRY, (st))
448 #define sk_BY_DIR_ENTRY_is_sorted(st) SKM_sk_is_sorted(BY_DIR_ENTRY, (st))

450 #define sk_BY_DIR_HASH_new(cmp) SKM_sk_new(BY_DIR_HASH, (cmp))
451 #define sk_BY_DIR_HASH_new_null() SKM_sk_new_null(BY_DIR_HASH)
452 #define sk_BY_DIR_HASH_free(st) SKM_sk_free(BY_DIR_HASH, (st))
453 #define sk_BY_DIR_HASH_num(st) SKM_sk_num(BY_DIR_HASH, (st))
454 #define sk_BY_DIR_HASH_value(st, i) SKM_sk_value(BY_DIR_HASH, (st), (i))
455 #define sk_BY_DIR_HASH_set(st, i, val) SKM_sk_set(BY_DIR_HASH, (st), (i), (val))
456 #define sk_BY_DIR_HASH_zero(st) SKM_sk_zero(BY_DIR_HASH, (st))
457 #define sk_BY_DIR_HASH_push(st, val) SKM_sk_push(BY_DIR_HASH, (st), (val))
```



```
458 #define sk_BY_DIR_HASH_unshift(st, val) SKM_sk_unshift(BY_DIR_HASH, (st), (val))
459 #define sk_BY_DIR_HASH_find(st, val) SKM_sk_find(BY_DIR_HASH, (st), (val))
460 #define sk_BY_DIR_HASH_find_ex(st, val) SKM_sk_find_ex(BY_DIR_HASH, (st), (val))
461 #define sk_BY_DIR_HASH_delete(st, i) SKM_sk_delete(BY_DIR_HASH, (st), (i))
462 #define sk_BY_DIR_HASH_delete_ptr(st, ptr) SKM_sk_delete_ptr(BY_DIR_HASH, (st),
463 #define sk_BY_DIR_HASH_insert(st, val, i) SKM_sk_insert(BY_DIR_HASH, (st), (val)
464 #define sk_BY_DIR_HASH_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(BY_DIR_HASH, (s
465 #define sk_BY_DIR_HASH_dup(st) SKM_sk_dup(BY_DIR_HASH, st)
466 #define sk_BY_DIR_HASH_pop_free(st, free_func) SKM_sk_pop_free(BY_DIR_HASH, (st)
467 #define sk_BY_DIR_HASH_shift(st) SKM_sk_shift(BY_DIR_HASH, (st))
468 #define sk_BY_DIR_HASH_pop(st) SKM_sk_pop(BY_DIR_HASH, (st))
469 #define sk_BY_DIR_HASH_sort(st) SKM_sk_sort(BY_DIR_HASH, (st))
470 #define sk_BY_DIR_HASH_is_sorted(st) SKM_sk_is_sorted(BY_DIR_HASH, (st))

472 #define sk_CMS_CertificateChoices_new(cmp) SKM_sk_new(CMS_CertificateChoices, (c
473 #define sk_CMS_CertificateChoices_new_null() SKM_sk_new_null(CMS_CertificateChoi
474 #define sk_CMS_CertificateChoices_free(st) SKM_sk_free(CMS_CertificateChoices, (
475 #define sk_CMS_CertificateChoices_num(st) SKM_sk_num(CMS_CertificateChoices, (st
476 #define sk_CMS_CertificateChoices_value(st, i) SKM_sk_value(CMS_CertificateChoic
477 #define sk_CMS_CertificateChoices_set(st, i, val) SKM_sk_set(CMS_CertificateChoi
478 #define sk_CMS_CertificateChoices_zero(st) SKM_sk_zero(CMS_CertificateChoices, (
479 #define sk_CMS_CertificateChoices_push(st, val) SKM_sk_push(CMS_CertificateChoic
480 #define sk_CMS_CertificateChoices_unshift(st, val) SKM_sk_unshift(CMS_Certificat
481 #define sk_CMS_CertificateChoices_find(st, val) SKM_sk_find(CMS_CertificateChoic
482 #define sk_CMS_CertificateChoices_find_ex(st, val) SKM_sk_find_ex(CMS_Certificat
483 #define sk_CMS_CertificateChoices_delete(st, i) SKM_sk_delete(CMS_CertificateCho
484 #define sk_CMS_CertificateChoices_delete_ptr(st, ptr) SKM_sk_delete_ptr(CMS_Cert
485 #define sk_CMS_CertificateChoices_insert(st, val, i) SKM_sk_insert(CMS_Certifica
486 #define sk_CMS_CertificateChoices_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CMS_
487 #define sk_CMS_CertificateChoices_dup(st) SKM_sk_dup(CMS_CertificateChoices, st)
488 #define sk_CMS_CertificateChoices_pop_free(st, free_func) SKM_sk_pop_free(CMS_Ce
489 #define sk_CMS_CertificateChoices_shift(st) SKM_sk_shift(CMS_CertificateChoices,
490 #define sk_CMS_CertificateChoices_pop(st) SKM_sk_pop(CMS_CertificateChoices, (st
491 #define sk_CMS_CertificateChoices_sort(st) SKM_sk_sort(CMS_CertificateChoices, (
492 #define sk_CMS_CertificateChoices_is_sorted(st) SKM_sk_is_sorted(CMS_Certificate

494 #define sk_CMS_RecipientInfo_new(cmp) SKM_sk_new(CMS_RecipientInfo, (cmp))
495 #define sk_CMS_RecipientInfo_new_null() SKM_sk_new_null(CMS_RecipientInfo)
496 #define sk_CMS_RecipientInfo_free(st) SKM_sk_free(CMS_RecipientInfo, (st))
497 #define sk_CMS_RecipientInfo_num(st) SKM_sk_num(CMS_RecipientInfo, (st))
498 #define sk_CMS_RecipientInfo_value(st, i) SKM_sk_value(CMS_RecipientInfo, (st),
499 #define sk_CMS_RecipientInfo_set(st, i, val) SKM_sk_set(CMS_RecipientInfo, (st),
500 #define sk_CMS_RecipientInfo_zero(st) SKM_sk_zero(CMS_RecipientInfo, (st))
501 #define sk_CMS_RecipientInfo_push(st, val) SKM_sk_push(CMS_RecipientInfo, (st),
502 #define sk_CMS_RecipientInfo_unshift(st, val) SKM_sk_unshift(CMS_RecipientInfo,
503 #define sk_CMS_RecipientInfo_find(st, val) SKM_sk_find(CMS_RecipientInfo, (st),
504 #define sk_CMS_RecipientInfo_find_ex(st, val) SKM_sk_find_ex(CMS_RecipientInfo,
505 #define sk_CMS_RecipientInfo_delete(st, i) SKM_sk_delete(CMS_RecipientInfo, (st)
506 #define sk_CMS_RecipientInfo_delete_ptr(st, ptr) SKM_sk_delete_ptr(CMS_Recipient
507 #define sk_CMS_RecipientInfo_insert(st, val, i) SKM_sk_insert(CMS_RecipientInfo,
508 #define sk_CMS_RecipientInfo_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CMS_Recip
509 #define sk_CMS_RecipientInfo_dup(st) SKM_sk_dup(CMS_RecipientInfo, st)
510 #define sk_CMS_RecipientInfo_pop_free(st, free_func) SKM_sk_pop_free(CMS_Recipie
511 #define sk_CMS_RecipientInfo_shift(st) SKM_sk_shift(CMS_RecipientInfo, (st))
512 #define sk_CMS_RecipientInfo_pop(st) SKM_sk_pop(CMS_RecipientInfo, (st))
513 #define sk_CMS_RecipientInfo_sort(st) SKM_sk_sort(CMS_RecipientInfo, (st))
514 #define sk_CMS_RecipientInfo_is_sorted(st) SKM_sk_is_sorted(CMS_RecipientInfo, (

516 #define sk_CMS_RevocationInfoChoice_new(cmp) SKM_sk_new(CMS_RevocationInfoChoice
517 #define sk_CMS_RevocationInfoChoice_new_null() SKM_sk_new_null(CMS_RevocationInf
518 #define sk_CMS_RevocationInfoChoice_free(st) SKM_sk_free(CMS_RevocationInfoChoic
519 #define sk_CMS_RevocationInfoChoice_num(st) SKM_sk_num(CMS_RevocationInfoChoice,
520 #define sk_CMS_RevocationInfoChoice_value(st, i) SKM_sk_value(CMS_RevocationInfo
521 #define sk_CMS_RevocationInfoChoice_set(st, i, val) SKM_sk_set(CMS_RevocationInf
522 #define sk_CMS_RevocationInfoChoice_zero(st) SKM_sk_zero(CMS_RevocationInfoChoic
523 #define sk_CMS_RevocationInfoChoice_push(st, val) SKM_sk_push(CMS_RevocationInfo
```

```
#define sk_CMS_RevocationInfoChoice_unshift(st, val) SKM_sk_unshift(CMS_Revocati
#define sk_CMS_RevocationInfoChoice_find(st, val) SKM_sk_find(CMS_RevocationInfo
#define sk_CMS_RevocationInfoChoice_find_ex(st, val) SKM_sk_find_ex(CMS_Revocati
#define sk_CMS_RevocationInfoChoice_delete(st, i) SKM_sk_delete(CMS_RevocationIn
#define sk_CMS_RevocationInfoChoice_delete_ptr(st, ptr) SKM_sk_delete_ptr(CMS_Re
#define sk_CMS_RevocationInfoChoice_insert(st, val, i) SKM_sk_insert(CMS_Revocat
#define sk_CMS_RevocationInfoChoice_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CM
#define sk_CMS_RevocationInfoChoice_dup(st) SKM_sk_dup(CMS_RevocationInfoChoice,
#define sk_CMS_RevocationInfoChoice_pop_free(st, free_func) SKM_sk_pop_free(CMS_
#define sk_CMS_RevocationInfoChoice_shift(st) SKM_sk_shift(CMS_RevocationInfoCho
#define sk_CMS_RevocationInfoChoice_pop(st) SKM_sk_pop(CMS_RevocationInfoChoice,
#define sk_CMS_RevocationInfoChoice_sort(st) SKM_sk_sort(CMS_RevocationInfoChoic
#define sk_CMS_RevocationInfoChoice_is_sorted(st) SKM_sk_is_sorted(CMS_Revocatio

#define sk_CMS_SignerInfo_new(cmp) SKM_sk_new(CMS_SignerInfo, (cmp))
#define sk_CMS_SignerInfo_new_null() SKM_sk_new_null(CMS_SignerInfo)
#define sk_CMS_SignerInfo_free(st) SKM_sk_free(CMS_SignerInfo, (st))
#define sk_CMS_SignerInfo_num(st) SKM_sk_num(CMS_SignerInfo, (st))
#define sk_CMS_SignerInfo_value(st, i) SKM_sk_value(CMS_SignerInfo, (st), (i))
#define sk_CMS_SignerInfo_set(st, i, val) SKM_sk_set(CMS_SignerInfo, (st), (i),
#define sk_CMS_SignerInfo_zero(st) SKM_sk_zero(CMS_SignerInfo, (st))
#define sk_CMS_SignerInfo_push(st, val) SKM_sk_push(CMS_SignerInfo, (st), (val))
#define sk_CMS_SignerInfo_unshift(st, val) SKM_sk_unshift(CMS_SignerInfo, (st),
#define sk_CMS_SignerInfo_find(st, val) SKM_sk_find(CMS_SignerInfo, (st), (val))
#define sk_CMS_SignerInfo_find_ex(st, val) SKM_sk_find_ex(CMS_SignerInfo, (st),
#define sk_CMS_SignerInfo_delete(st, i) SKM_sk_delete(CMS_SignerInfo, (st), (i))
#define sk_CMS_SignerInfo_delete_ptr(st, ptr) SKM_sk_delete_ptr(CMS_SignerInfo,
#define sk_CMS_SignerInfo_insert(st, val, i) SKM_sk_insert(CMS_SignerInfo, (st),
#define sk_CMS_SignerInfo_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CMS_SignerIn
#define sk_CMS_SignerInfo_dup(st) SKM_sk_dup(CMS_SignerInfo, st)
#define sk_CMS_SignerInfo_pop_free(st, free_func) SKM_sk_pop_free(CMS_SignerInfo
#define sk_CMS_SignerInfo_shift(st) SKM_sk_shift(CMS_SignerInfo, (st))
#define sk_CMS_SignerInfo_pop(st) SKM_sk_pop(CMS_SignerInfo, (st))
#define sk_CMS_SignerInfo_sort(st) SKM_sk_sort(CMS_SignerInfo, (st))
#define sk_CMS_SignerInfo_is_sorted(st) SKM_sk_is_sorted(CMS_SignerInfo, (st))

#define sk_CONF_IMODULE_new(cmp) SKM_sk_new(CONF_IMODULE, (cmp))
#define sk_CONF_IMODULE_new_null() SKM_sk_new_null(CONF_IMODULE)
#define sk_CONF_IMODULE_free(st) SKM_sk_free(CONF_IMODULE, (st))
#define sk_CONF_IMODULE_num(st) SKM_sk_num(CONF_IMODULE, (st))
#define sk_CONF_IMODULE_value(st, i) SKM_sk_value(CONF_IMODULE, (st), (i))
#define sk_CONF_IMODULE_set(st, i, val) SKM_sk_set(CONF_IMODULE, (st), (i), (val
#define sk_CONF_IMODULE_zero(st) SKM_sk_zero(CONF_IMODULE, (st))
#define sk_CONF_IMODULE_push(st, val) SKM_sk_push(CONF_IMODULE, (st), (val))
#define sk_CONF_IMODULE_unshift(st, val) SKM_sk_unshift(CONF_IMODULE, (st), (val
#define sk_CONF_IMODULE_find(st, val) SKM_sk_find(CONF_IMODULE, (st), (val))
#define sk_CONF_IMODULE_find_ex(st, val) SKM_sk_find_ex(CONF_IMODULE, (st), (val
#define sk_CONF_IMODULE_delete(st, i) SKM_sk_delete(CONF_IMODULE, (st), (i))
#define sk_CONF_IMODULE_delete_ptr(st, ptr) SKM_sk_delete_ptr(CONF_IMODULE, (st)
#define sk_CONF_IMODULE_insert(st, val, i) SKM_sk_insert(CONF_IMODULE, (st), (va
#define sk_CONF_IMODULE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CONF_IMODULE,
#define sk_CONF_IMODULE_dup(st) SKM_sk_dup(CONF_IMODULE, st)
#define sk_CONF_IMODULE_pop_free(st, free_func) SKM_sk_pop_free(CONF_IMODULE, (s
#define sk_CONF_IMODULE_shift(st) SKM_sk_shift(CONF_IMODULE, (st))
#define sk_CONF_IMODULE_pop(st) SKM_sk_pop(CONF_IMODULE, (st))
#define sk_CONF_IMODULE_sort(st) SKM_sk_sort(CONF_IMODULE, (st))
#define sk_CONF_IMODULE_is_sorted(st) SKM_sk_is_sorted(CONF_IMODULE, (st))

#define sk_CONF_MODULE_new(cmp) SKM_sk_new(CONF_MODULE, (cmp))
#define sk_CONF_MODULE_new_null() SKM_sk_new_null(CONF_MODULE)
#define sk_CONF_MODULE_free(st) SKM_sk_free(CONF_MODULE, (st))
#define sk_CONF_MODULE_num(st) SKM_sk_num(CONF_MODULE, (st))
#define sk_CONF_MODULE_value(st, i) SKM_sk_value(CONF_MODULE, (st), (i))
#define sk_CONF_MODULE_set(st, i, val) SKM_sk_set(CONF_MODULE, (st), (i), (val))
#define sk_CONF_MODULE_zero(st) SKM_sk_zero(CONF_MODULE, (st))
#define sk_CONF_MODULE_push(st, val) SKM_sk_push(CONF_MODULE, (st), (val))
```



```
#define sk_CONF_MODULE_unshift(st, val) SKM_sk_unshift(CONF_MODULE, (st), (val))
#define sk_CONF_MODULE_find(st, val) SKM_sk_find(CONF_MODULE, (st), (val))
#define sk_CONF_MODULE_find_ex(st, val) SKM_sk_find_ex(CONF_MODULE, (st), (val))
#define sk_CONF_MODULE_delete(st, i) SKM_sk_delete(CONF_MODULE, (st), (i))
#define sk_CONF_MODULE_delete_ptr(st, ptr) SKM_sk_delete_ptr(CONF_MODULE, (st),
#define sk_CONF_MODULE_insert(st, val, i) SKM_sk_insert(CONF_MODULE, (st), (val)
#define sk_CONF_MODULE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CONF_MODULE, (s
#define sk_CONF_MODULE_dup(st) SKM_sk_dup(CONF_MODULE, st)
#define sk_CONF_MODULE_pop_free(st, free_func) SKM_sk_pop_free(CONF_MODULE, (st)
#define sk_CONF_MODULE_shift(st) SKM_sk_shift(CONF_MODULE, (st))
#define sk_CONF_MODULE_pop(st) SKM_sk_pop(CONF_MODULE, (st))
#define sk_CONF_MODULE_sort(st) SKM_sk_sort(CONF_MODULE, (st))
#define sk_CONF_MODULE_is_sorted(st) SKM_sk_is_sorted(CONF_MODULE, (st))

#define sk_CONF_VALUE_new(cmp) SKM_sk_new(CONF_VALUE, (cmp))
#define sk_CONF_VALUE_new_null() SKM_sk_new_null(CONF_VALUE)
#define sk_CONF_VALUE_free(st) SKM_sk_free(CONF_VALUE, (st))
#define sk_CONF_VALUE_num(st) SKM_sk_num(CONF_VALUE, (st))
#define sk_CONF_VALUE_value(st, i) SKM_sk_value(CONF_VALUE, (st), (i))
#define sk_CONF_VALUE_set(st, i, val) SKM_sk_set(CONF_VALUE, (st), (i), (val))
#define sk_CONF_VALUE_zero(st) SKM_sk_zero(CONF_VALUE, (st))
#define sk_CONF_VALUE_push(st, val) SKM_sk_push(CONF_VALUE, (st), (val))
#define sk_CONF_VALUE_unshift(st, val) SKM_sk_unshift(CONF_VALUE, (st), (val))
#define sk_CONF_VALUE_find(st, val) SKM_sk_find(CONF_VALUE, (st), (val))
#define sk_CONF_VALUE_find_ex(st, val) SKM_sk_find_ex(CONF_VALUE, (st), (val))
#define sk_CONF_VALUE_delete(st, i) SKM_sk_delete(CONF_VALUE, (st), (i))
#define sk_CONF_VALUE_delete_ptr(st, ptr) SKM_sk_delete_ptr(CONF_VALUE, (st), (p
#define sk_CONF_VALUE_insert(st, val, i) SKM_sk_insert(CONF_VALUE, (st), (val),
#define sk_CONF_VALUE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CONF_VALUE, (st)
#define sk_CONF_VALUE_dup(st) SKM_sk_dup(CONF_VALUE, st)
#define sk_CONF_VALUE_pop_free(st, free_func) SKM_sk_pop_free(CONF_VALUE, (st),
#define sk_CONF_VALUE_shift(st) SKM_sk_shift(CONF_VALUE, (st))
#define sk_CONF_VALUE_pop(st) SKM_sk_pop(CONF_VALUE, (st))
#define sk_CONF_VALUE_sort(st) SKM_sk_sort(CONF_VALUE, (st))
#define sk_CONF_VALUE_is_sorted(st) SKM_sk_is_sorted(CONF_VALUE, (st))

#define sk_CRYPTO_EX_DATA_FUNCS_new(cmp) SKM_sk_new(CRYPTO_EX_DATA_FUNCS, (cmp))
#define sk_CRYPTO_EX_DATA_FUNCS_new_null() SKM_sk_new_null(CRYPTO_EX_DATA_FUNCS)
#define sk_CRYPTO_EX_DATA_FUNCS_free(st) SKM_sk_free(CRYPTO_EX_DATA_FUNCS, (st))
#define sk_CRYPTO_EX_DATA_FUNCS_num(st) SKM_sk_num(CRYPTO_EX_DATA_FUNCS, (st))
#define sk_CRYPTO_EX_DATA_FUNCS_value(st, i) SKM_sk_value(CRYPTO_EX_DATA_FUNCS,
#define sk_CRYPTO_EX_DATA_FUNCS_set(st, i, val) SKM_sk_set(CRYPTO_EX_DATA_FUNCS,
#define sk_CRYPTO_EX_DATA_FUNCS_zero(st) SKM_sk_zero(CRYPTO_EX_DATA_FUNCS, (st))
#define sk_CRYPTO_EX_DATA_FUNCS_push(st, val) SKM_sk_push(CRYPTO_EX_DATA_FUNCS,
#define sk_CRYPTO_EX_DATA_FUNCS_unshift(st, val) SKM_sk_unshift(CRYPTO_EX_DATA_F
#define sk_CRYPTO_EX_DATA_FUNCS_find(st, val) SKM_sk_find(CRYPTO_EX_DATA_FUNCS,
#define sk_CRYPTO_EX_DATA_FUNCS_find_ex(st, val) SKM_sk_find_ex(CRYPTO_EX_DATA_F
#define sk_CRYPTO_EX_DATA_FUNCS_delete(st, i) SKM_sk_delete(CRYPTO_EX_DATA_FUNCS
#define sk_CRYPTO_EX_DATA_FUNCS_delete_ptr(st, ptr) SKM_sk_delete_ptr(CRYPTO_EX_
#define sk_CRYPTO_EX_DATA_FUNCS_insert(st, val, i) SKM_sk_insert(CRYPTO_EX_DATA_
#define sk_CRYPTO_EX_DATA_FUNCS_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CRYPTO
#define sk_CRYPTO_EX_DATA_FUNCS_dup(st) SKM_sk_dup(CRYPTO_EX_DATA_FUNCS, st)
#define sk_CRYPTO_EX_DATA_FUNCS_pop_free(st, free_func) SKM_sk_pop_free(CRYPTO_E
#define sk_CRYPTO_EX_DATA_FUNCS_shift(st) SKM_sk_shift(CRYPTO_EX_DATA_FUNCS, (st
#define sk_CRYPTO_EX_DATA_FUNCS_pop(st) SKM_sk_pop(CRYPTO_EX_DATA_FUNCS, (st))
#define sk_CRYPTO_EX_DATA_FUNCS_sort(st) SKM_sk_sort(CRYPTO_EX_DATA_FUNCS, (st))
#define sk_CRYPTO_EX_DATA_FUNCS_is_sorted(st) SKM_sk_is_sorted(CRYPTO_EX_DATA_FU

#define sk_CRYPTO_dynlock_new(cmp) SKM_sk_new(CRYPTO_dynlock, (cmp))
#define sk_CRYPTO_dynlock_new_null() SKM_sk_new_null(CRYPTO_dynlock)
#define sk_CRYPTO_dynlock_free(st) SKM_sk_free(CRYPTO_dynlock, (st))
#define sk_CRYPTO_dynlock_num(st) SKM_sk_num(CRYPTO_dynlock, (st))
#define sk_CRYPTO_dynlock_value(st, i) SKM_sk_value(CRYPTO_dynlock, (st), (i))
#define sk_CRYPTO_dynlock_set(st, i, val) SKM_sk_set(CRYPTO_dynlock, (st), (i),
#define sk_CRYPTO_dynlock_zero(st) SKM_sk_zero(CRYPTO_dynlock, (st))
#define sk_CRYPTO_dynlock_push(st, val) SKM_sk_push(CRYPTO_dynlock, (st), (val))
```

```
#define sk_CRYPTO_dynlock_unshift(st, val) SKM_sk_unshift(CRYPTO_dynlock, (st),
#define sk_CRYPTO_dynlock_find(st, val) SKM_sk_find(CRYPTO_dynlock, (st), (val))
#define sk_CRYPTO_dynlock_find_ex(st, val) SKM_sk_find_ex(CRYPTO_dynlock, (st),
#define sk_CRYPTO_dynlock_delete(st, i) SKM_sk_delete(CRYPTO_dynlock, (st), (i))
#define sk_CRYPTO_dynlock_delete_ptr(st, ptr) SKM_sk_delete_ptr(CRYPTO_dynlock,
#define sk_CRYPTO_dynlock_insert(st, val, i) SKM_sk_insert(CRYPTO_dynlock, (st),
#define sk_CRYPTO_dynlock_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(CRYPTO_dynlo
#define sk_CRYPTO_dynlock_dup(st) SKM_sk_dup(CRYPTO_dynlock, st)
#define sk_CRYPTO_dynlock_pop_free(st, free_func) SKM_sk_pop_free(CRYPTO_dynlock
#define sk_CRYPTO_dynlock_shift(st) SKM_sk_shift(CRYPTO_dynlock, (st))
#define sk_CRYPTO_dynlock_pop(st) SKM_sk_pop(CRYPTO_dynlock, (st))
#define sk_CRYPTO_dynlock_sort(st) SKM_sk_sort(CRYPTO_dynlock, (st))
#define sk_CRYPTO_dynlock_is_sorted(st) SKM_sk_is_sorted(CRYPTO_dynlock, (st))

#define sk_DIST_POINT_new(cmp) SKM_sk_new(DIST_POINT, (cmp))
#define sk_DIST_POINT_new_null() SKM_sk_new_null(DIST_POINT)
#define sk_DIST_POINT_free(st) SKM_sk_free(DIST_POINT, (st))
#define sk_DIST_POINT_num(st) SKM_sk_num(DIST_POINT, (st))
#define sk_DIST_POINT_value(st, i) SKM_sk_value(DIST_POINT, (st), (i))
#define sk_DIST_POINT_set(st, i, val) SKM_sk_set(DIST_POINT, (st), (i), (val))
#define sk_DIST_POINT_zero(st) SKM_sk_zero(DIST_POINT, (st))
#define sk_DIST_POINT_push(st, val) SKM_sk_push(DIST_POINT, (st), (val))
#define sk_DIST_POINT_unshift(st, val) SKM_sk_unshift(DIST_POINT, (st), (val))
#define sk_DIST_POINT_find(st, val) SKM_sk_find(DIST_POINT, (st), (val))
#define sk_DIST_POINT_find_ex(st, val) SKM_sk_find_ex(DIST_POINT, (st), (val))
#define sk_DIST_POINT_delete(st, i) SKM_sk_delete(DIST_POINT, (st), (i))
#define sk_DIST_POINT_delete_ptr(st, ptr) SKM_sk_delete_ptr(DIST_POINT, (st), (p
#define sk_DIST_POINT_insert(st, val, i) SKM_sk_insert(DIST_POINT, (st), (val),
#define sk_DIST_POINT_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(DIST_POINT, (st)
#define sk_DIST_POINT_dup(st) SKM_sk_dup(DIST_POINT, st)
#define sk_DIST_POINT_pop_free(st, free_func) SKM_sk_pop_free(DIST_POINT, (st),
#define sk_DIST_POINT_shift(st) SKM_sk_shift(DIST_POINT, (st))
#define sk_DIST_POINT_pop(st) SKM_sk_pop(DIST_POINT, (st))
#define sk_DIST_POINT_sort(st) SKM_sk_sort(DIST_POINT, (st))
#define sk_DIST_POINT_is_sorted(st) SKM_sk_is_sorted(DIST_POINT, (st))

#define sk_ENGINE_new(cmp) SKM_sk_new(ENGINE, (cmp))
#define sk_ENGINE_new_null() SKM_sk_new_null(ENGINE)
#define sk_ENGINE_free(st) SKM_sk_free(ENGINE, (st))
#define sk_ENGINE_num(st) SKM_sk_num(ENGINE, (st))
#define sk_ENGINE_value(st, i) SKM_sk_value(ENGINE, (st), (i))
#define sk_ENGINE_set(st, i, val) SKM_sk_set(ENGINE, (st), (i), (val))
#define sk_ENGINE_zero(st) SKM_sk_zero(ENGINE, (st))
#define sk_ENGINE_push(st, val) SKM_sk_push(ENGINE, (st), (val))
#define sk_ENGINE_unshift(st, val) SKM_sk_unshift(ENGINE, (st), (val))
#define sk_ENGINE_find(st, val) SKM_sk_find(ENGINE, (st), (val))
#define sk_ENGINE_find_ex(st, val) SKM_sk_find_ex(ENGINE, (st), (val))
#define sk_ENGINE_delete(st, i) SKM_sk_delete(ENGINE, (st), (i))
#define sk_ENGINE_delete_ptr(st, ptr) SKM_sk_delete_ptr(ENGINE, (st), (ptr))
#define sk_ENGINE_insert(st, val, i) SKM_sk_insert(ENGINE, (st), (val), (i))
#define sk_ENGINE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ENGINE, (st), (cmp))
#define sk_ENGINE_dup(st) SKM_sk_dup(ENGINE, st)
#define sk_ENGINE_pop_free(st, free_func) SKM_sk_pop_free(ENGINE, (st), (free_fu
#define sk_ENGINE_shift(st) SKM_sk_shift(ENGINE, (st))
#define sk_ENGINE_pop(st) SKM_sk_pop(ENGINE, (st))
#define sk_ENGINE_sort(st) SKM_sk_sort(ENGINE, (st))
#define sk_ENGINE_is_sorted(st) SKM_sk_is_sorted(ENGINE, (st))

#define sk_ENGINE_CLEANUP_ITEM_new(cmp) SKM_sk_new(ENGINE_CLEANUP_ITEM, (cmp))
#define sk_ENGINE_CLEANUP_ITEM_new_null() SKM_sk_new_null(ENGINE_CLEANUP_ITEM)
#define sk_ENGINE_CLEANUP_ITEM_free(st) SKM_sk_free(ENGINE_CLEANUP_ITEM, (st))
#define sk_ENGINE_CLEANUP_ITEM_num(st) SKM_sk_num(ENGINE_CLEANUP_ITEM, (st))
#define sk_ENGINE_CLEANUP_ITEM_value(st, i) SKM_sk_value(ENGINE_CLEANUP_ITEM, (s
#define sk_ENGINE_CLEANUP_ITEM_set(st, i, val) SKM_sk_set(ENGINE_CLEANUP_ITEM, (
#define sk_ENGINE_CLEANUP_ITEM_zero(st) SKM_sk_zero(ENGINE_CLEANUP_ITEM, (st))
#define sk_ENGINE_CLEANUP_ITEM_push(st, val) SKM_sk_push(ENGINE_CLEANUP_ITEM, (s
```



```
#define sk_ENGINE_CLEANUP_ITEM_unshift(st, val) SKM_sk_unshift(ENGINE_CLEANUP_IT
#define sk_ENGINE_CLEANUP_ITEM_find(st, val) SKM_sk_find(ENGINE_CLEANUP_ITEM, (s
#define sk_ENGINE_CLEANUP_ITEM_find_ex(st, val) SKM_sk_find_ex(ENGINE_CLEANUP_IT
#define sk_ENGINE_CLEANUP_ITEM_delete(st, i) SKM_sk_delete(ENGINE_CLEANUP_ITEM,
#define sk_ENGINE_CLEANUP_ITEM_delete_ptr(st, ptr) SKM_sk_delete_ptr(ENGINE_CLEA
#define sk_ENGINE_CLEANUP_ITEM_insert(st, val, i) SKM_sk_insert(ENGINE_CLEANUP_I
#define sk_ENGINE_CLEANUP_ITEM_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ENGINE_
#define sk_ENGINE_CLEANUP_ITEM_dup(st) SKM_sk_dup(ENGINE_CLEANUP_ITEM, st)
#define sk_ENGINE_CLEANUP_ITEM_pop_free(st, free_func) SKM_sk_pop_free(ENGINE_CL
#define sk_ENGINE_CLEANUP_ITEM_shift(st) SKM_sk_shift(ENGINE_CLEANUP_ITEM, (st))
#define sk_ENGINE_CLEANUP_ITEM_pop(st) SKM_sk_pop(ENGINE_CLEANUP_ITEM, (st))
#define sk_ENGINE_CLEANUP_ITEM_sort(st) SKM_sk_sort(ENGINE_CLEANUP_ITEM, (st))
#define sk_ENGINE_CLEANUP_ITEM_is_sorted(st) SKM_sk_is_sorted(ENGINE_CLEANUP_ITE

#define sk_ESS_CERT_ID_new(cmp) SKM_sk_new(ESS_CERT_ID, (cmp))
#define sk_ESS_CERT_ID_new_null() SKM_sk_new_null(ESS_CERT_ID)
#define sk_ESS_CERT_ID_free(st) SKM_sk_free(ESS_CERT_ID, (st))
#define sk_ESS_CERT_ID_num(st) SKM_sk_num(ESS_CERT_ID, (st))
#define sk_ESS_CERT_ID_value(st, i) SKM_sk_value(ESS_CERT_ID, (st), (i))
#define sk_ESS_CERT_ID_set(st, i, val) SKM_sk_set(ESS_CERT_ID, (st), (i), (val))
#define sk_ESS_CERT_ID_zero(st) SKM_sk_zero(ESS_CERT_ID, (st))
#define sk_ESS_CERT_ID_push(st, val) SKM_sk_push(ESS_CERT_ID, (st), (val))
#define sk_ESS_CERT_ID_unshift(st, val) SKM_sk_unshift(ESS_CERT_ID, (st), (val))
#define sk_ESS_CERT_ID_find(st, val) SKM_sk_find(ESS_CERT_ID, (st), (val))
#define sk_ESS_CERT_ID_find_ex(st, val) SKM_sk_find_ex(ESS_CERT_ID, (st), (val))
#define sk_ESS_CERT_ID_delete(st, i) SKM_sk_delete(ESS_CERT_ID, (st), (i))
#define sk_ESS_CERT_ID_delete_ptr(st, ptr) SKM_sk_delete_ptr(ESS_CERT_ID, (st),
#define sk_ESS_CERT_ID_insert(st, val, i) SKM_sk_insert(ESS_CERT_ID, (st), (val)
#define sk_ESS_CERT_ID_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ESS_CERT_ID, (s
#define sk_ESS_CERT_ID_dup(st) SKM_sk_dup(ESS_CERT_ID, st)
#define sk_ESS_CERT_ID_pop_free(st, free_func) SKM_sk_pop_free(ESS_CERT_ID, (st)
#define sk_ESS_CERT_ID_shift(st) SKM_sk_shift(ESS_CERT_ID, (st))
#define sk_ESS_CERT_ID_pop(st) SKM_sk_pop(ESS_CERT_ID, (st))
#define sk_ESS_CERT_ID_sort(st) SKM_sk_sort(ESS_CERT_ID, (st))
#define sk_ESS_CERT_ID_is_sorted(st) SKM_sk_is_sorted(ESS_CERT_ID, (st))

#define sk_EVP_MD_new(cmp) SKM_sk_new(EVP_MD, (cmp))
#define sk_EVP_MD_new_null() SKM_sk_new_null(EVP_MD)
#define sk_EVP_MD_free(st) SKM_sk_free(EVP_MD, (st))
#define sk_EVP_MD_num(st) SKM_sk_num(EVP_MD, (st))
#define sk_EVP_MD_value(st, i) SKM_sk_value(EVP_MD, (st), (i))
#define sk_EVP_MD_set(st, i, val) SKM_sk_set(EVP_MD, (st), (i), (val))
#define sk_EVP_MD_zero(st) SKM_sk_zero(EVP_MD, (st))
#define sk_EVP_MD_push(st, val) SKM_sk_push(EVP_MD, (st), (val))
#define sk_EVP_MD_unshift(st, val) SKM_sk_unshift(EVP_MD, (st), (val))
#define sk_EVP_MD_find(st, val) SKM_sk_find(EVP_MD, (st), (val))
#define sk_EVP_MD_find_ex(st, val) SKM_sk_find_ex(EVP_MD, (st), (val))
#define sk_EVP_MD_delete(st, i) SKM_sk_delete(EVP_MD, (st), (i))
#define sk_EVP_MD_delete_ptr(st, ptr) SKM_sk_delete_ptr(EVP_MD, (st), (ptr))
#define sk_EVP_MD_insert(st, val, i) SKM_sk_insert(EVP_MD, (st), (val), (i))
#define sk_EVP_MD_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(EVP_MD, (st), (cmp))
#define sk_EVP_MD_dup(st) SKM_sk_dup(EVP_MD, st)
#define sk_EVP_MD_pop_free(st, free_func) SKM_sk_pop_free(EVP_MD, (st), (free_fu
#define sk_EVP_MD_shift(st) SKM_sk_shift(EVP_MD, (st))
#define sk_EVP_MD_pop(st) SKM_sk_pop(EVP_MD, (st))
#define sk_EVP_MD_sort(st) SKM_sk_sort(EVP_MD, (st))
#define sk_EVP_MD_is_sorted(st) SKM_sk_is_sorted(EVP_MD, (st))

#define sk_EVP_PBE_CTL_new(cmp) SKM_sk_new(EVP_PBE_CTL, (cmp))
#define sk_EVP_PBE_CTL_new_null() SKM_sk_new_null(EVP_PBE_CTL)
#define sk_EVP_PBE_CTL_free(st) SKM_sk_free(EVP_PBE_CTL, (st))
#define sk_EVP_PBE_CTL_num(st) SKM_sk_num(EVP_PBE_CTL, (st))
#define sk_EVP_PBE_CTL_value(st, i) SKM_sk_value(EVP_PBE_CTL, (st), (i))
#define sk_EVP_PBE_CTL_set(st, i, val) SKM_sk_set(EVP_PBE_CTL, (st), (i), (val))
#define sk_EVP_PBE_CTL_zero(st) SKM_sk_zero(EVP_PBE_CTL, (st))
#define sk_EVP_PBE_CTL_push(st, val) SKM_sk_push(EVP_PBE_CTL, (st), (val))
```

```
#define sk_EVP_PBE_CTL_unshift(st, val) SKM_sk_unshift(EVP_PBE_CTL, (st), (val))
#define sk_EVP_PBE_CTL_find(st, val) SKM_sk_find(EVP_PBE_CTL, (st), (val))
#define sk_EVP_PBE_CTL_find_ex(st, val) SKM_sk_find_ex(EVP_PBE_CTL, (st), (val))
#define sk_EVP_PBE_CTL_delete(st, i) SKM_sk_delete(EVP_PBE_CTL, (st), (i))
#define sk_EVP_PBE_CTL_delete_ptr(st, ptr) SKM_sk_delete_ptr(EVP_PBE_CTL, (st),
#define sk_EVP_PBE_CTL_insert(st, val, i) SKM_sk_insert(EVP_PBE_CTL, (st), (val)
#define sk_EVP_PBE_CTL_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(EVP_PBE_CTL, (s
#define sk_EVP_PBE_CTL_dup(st) SKM_sk_dup(EVP_PBE_CTL, st)
#define sk_EVP_PBE_CTL_pop_free(st, free_func) SKM_sk_pop_free(EVP_PBE_CTL, (st)
#define sk_EVP_PBE_CTL_shift(st) SKM_sk_shift(EVP_PBE_CTL, (st))
#define sk_EVP_PBE_CTL_pop(st) SKM_sk_pop(EVP_PBE_CTL, (st))
#define sk_EVP_PBE_CTL_sort(st) SKM_sk_sort(EVP_PBE_CTL, (st))
#define sk_EVP_PBE_CTL_is_sorted(st) SKM_sk_is_sorted(EVP_PBE_CTL, (st))

#define sk_EVP_PKEY_ASN1_METHOD_new(cmp) SKM_sk_new(EVP_PKEY_ASN1_METHOD, (cmp))
#define sk_EVP_PKEY_ASN1_METHOD_new_null() SKM_sk_new_null(EVP_PKEY_ASN1_METHOD)
#define sk_EVP_PKEY_ASN1_METHOD_free(st) SKM_sk_free(EVP_PKEY_ASN1_METHOD, (st))
#define sk_EVP_PKEY_ASN1_METHOD_num(st) SKM_sk_num(EVP_PKEY_ASN1_METHOD, (st))
#define sk_EVP_PKEY_ASN1_METHOD_value(st, i) SKM_sk_value(EVP_PKEY_ASN1_METHOD,
#define sk_EVP_PKEY_ASN1_METHOD_set(st, i, val) SKM_sk_set(EVP_PKEY_ASN1_METHOD,
#define sk_EVP_PKEY_ASN1_METHOD_zero(st) SKM_sk_zero(EVP_PKEY_ASN1_METHOD, (st))
#define sk_EVP_PKEY_ASN1_METHOD_push(st, val) SKM_sk_push(EVP_PKEY_ASN1_METHOD,
#define sk_EVP_PKEY_ASN1_METHOD_unshift(st, val) SKM_sk_unshift(EVP_PKEY_ASN1_ME
#define sk_EVP_PKEY_ASN1_METHOD_find(st, val) SKM_sk_find(EVP_PKEY_ASN1_METHOD,
#define sk_EVP_PKEY_ASN1_METHOD_find_ex(st, val) SKM_sk_find_ex(EVP_PKEY_ASN1_ME
#define sk_EVP_PKEY_ASN1_METHOD_delete(st, i) SKM_sk_delete(EVP_PKEY_ASN1_METHOD
#define sk_EVP_PKEY_ASN1_METHOD_delete_ptr(st, ptr) SKM_sk_delete_ptr(EVP_PKEY_A
#define sk_EVP_PKEY_ASN1_METHOD_insert(st, val, i) SKM_sk_insert(EVP_PKEY_ASN1_M
#define sk_EVP_PKEY_ASN1_METHOD_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(EVP_PK
#define sk_EVP_PKEY_ASN1_METHOD_dup(st) SKM_sk_dup(EVP_PKEY_ASN1_METHOD, st)
#define sk_EVP_PKEY_ASN1_METHOD_pop_free(st, free_func) SKM_sk_pop_free(EVP_PKEY
#define sk_EVP_PKEY_ASN1_METHOD_shift(st) SKM_sk_shift(EVP_PKEY_ASN1_METHOD, (st
#define sk_EVP_PKEY_ASN1_METHOD_pop(st) SKM_sk_pop(EVP_PKEY_ASN1_METHOD, (st))
#define sk_EVP_PKEY_ASN1_METHOD_sort(st) SKM_sk_sort(EVP_PKEY_ASN1_METHOD, (st))
#define sk_EVP_PKEY_ASN1_METHOD_is_sorted(st) SKM_sk_is_sorted(EVP_PKEY_ASN1_MET

#define sk_EVP_PKEY_METHOD_new(cmp) SKM_sk_new(EVP_PKEY_METHOD, (cmp))
#define sk_EVP_PKEY_METHOD_new_null() SKM_sk_new_null(EVP_PKEY_METHOD)
#define sk_EVP_PKEY_METHOD_free(st) SKM_sk_free(EVP_PKEY_METHOD, (st))
#define sk_EVP_PKEY_METHOD_num(st) SKM_sk_num(EVP_PKEY_METHOD, (st))
#define sk_EVP_PKEY_METHOD_value(st, i) SKM_sk_value(EVP_PKEY_METHOD, (st), (i))
#define sk_EVP_PKEY_METHOD_set(st, i, val) SKM_sk_set(EVP_PKEY_METHOD, (st), (i)
#define sk_EVP_PKEY_METHOD_zero(st) SKM_sk_zero(EVP_PKEY_METHOD, (st))
#define sk_EVP_PKEY_METHOD_push(st, val) SKM_sk_push(EVP_PKEY_METHOD, (st), (val
#define sk_EVP_PKEY_METHOD_unshift(st, val) SKM_sk_unshift(EVP_PKEY_METHOD, (st)
#define sk_EVP_PKEY_METHOD_find(st, val) SKM_sk_find(EVP_PKEY_METHOD, (st), (val
#define sk_EVP_PKEY_METHOD_find_ex(st, val) SKM_sk_find_ex(EVP_PKEY_METHOD, (st)
#define sk_EVP_PKEY_METHOD_delete(st, i) SKM_sk_delete(EVP_PKEY_METHOD, (st), (i
#define sk_EVP_PKEY_METHOD_delete_ptr(st, ptr) SKM_sk_delete_ptr(EVP_PKEY_METHOD
#define sk_EVP_PKEY_METHOD_insert(st, val, i) SKM_sk_insert(EVP_PKEY_METHOD, (st
#define sk_EVP_PKEY_METHOD_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(EVP_PKEY_ME
#define sk_EVP_PKEY_METHOD_dup(st) SKM_sk_dup(EVP_PKEY_METHOD, st)
#define sk_EVP_PKEY_METHOD_pop_free(st, free_func) SKM_sk_pop_free(EVP_PKEY_METH
#define sk_EVP_PKEY_METHOD_shift(st) SKM_sk_shift(EVP_PKEY_METHOD, (st))
#define sk_EVP_PKEY_METHOD_pop(st) SKM_sk_pop(EVP_PKEY_METHOD, (st))
#define sk_EVP_PKEY_METHOD_sort(st) SKM_sk_sort(EVP_PKEY_METHOD, (st))
#define sk_EVP_PKEY_METHOD_is_sorted(st) SKM_sk_is_sorted(EVP_PKEY_METHOD, (st))

#define sk_GENERAL_NAME_new(cmp) SKM_sk_new(GENERAL_NAME, (cmp))
#define sk_GENERAL_NAME_new_null() SKM_sk_new_null(GENERAL_NAME)
#define sk_GENERAL_NAME_free(st) SKM_sk_free(GENERAL_NAME, (st))
#define sk_GENERAL_NAME_num(st) SKM_sk_num(GENERAL_NAME, (st))
#define sk_GENERAL_NAME_value(st, i) SKM_sk_value(GENERAL_NAME, (st), (i))
#define sk_GENERAL_NAME_set(st, i, val) SKM_sk_set(GENERAL_NAME, (st), (i), (val
#define sk_GENERAL_NAME_zero(st) SKM_sk_zero(GENERAL_NAME, (st))
#define sk_GENERAL_NAME_push(st, val) SKM_sk_push(GENERAL_NAME, (st), (val))
```



```
#define sk_GENERAL_NAME_unshift(st, val) SKM_sk_unshift(GENERAL_NAME, (st), (val
#define sk_GENERAL_NAME_find(st, val) SKM_sk_find(GENERAL_NAME, (st), (val))
#define sk_GENERAL_NAME_find_ex(st, val) SKM_sk_find_ex(GENERAL_NAME, (st), (val
#define sk_GENERAL_NAME_delete(st, i) SKM_sk_delete(GENERAL_NAME, (st), (i))
#define sk_GENERAL_NAME_delete_ptr(st, ptr) SKM_sk_delete_ptr(GENERAL_NAME, (st)
#define sk_GENERAL_NAME_insert(st, val, i) SKM_sk_insert(GENERAL_NAME, (st), (va
#define sk_GENERAL_NAME_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(GENERAL_NAME,
#define sk_GENERAL_NAME_dup(st) SKM_sk_dup(GENERAL_NAME, st)
#define sk_GENERAL_NAME_pop_free(st, free_func) SKM_sk_pop_free(GENERAL_NAME, (s
#define sk_GENERAL_NAME_shift(st) SKM_sk_shift(GENERAL_NAME, (st))
#define sk_GENERAL_NAME_pop(st) SKM_sk_pop(GENERAL_NAME, (st))
#define sk_GENERAL_NAME_sort(st) SKM_sk_sort(GENERAL_NAME, (st))
#define sk_GENERAL_NAME_is_sorted(st) SKM_sk_is_sorted(GENERAL_NAME, (st))

#define sk_GENERAL_NAMES_new(cmp) SKM_sk_new(GENERAL_NAMES, (cmp))
#define sk_GENERAL_NAMES_new_null() SKM_sk_new_null(GENERAL_NAMES)
#define sk_GENERAL_NAMES_free(st) SKM_sk_free(GENERAL_NAMES, (st))
#define sk_GENERAL_NAMES_num(st) SKM_sk_num(GENERAL_NAMES, (st))
#define sk_GENERAL_NAMES_value(st, i) SKM_sk_value(GENERAL_NAMES, (st), (i))
#define sk_GENERAL_NAMES_set(st, i, val) SKM_sk_set(GENERAL_NAMES, (st), (i), (v
#define sk_GENERAL_NAMES_zero(st) SKM_sk_zero(GENERAL_NAMES, (st))
#define sk_GENERAL_NAMES_push(st, val) SKM_sk_push(GENERAL_NAMES, (st), (val))
#define sk_GENERAL_NAMES_unshift(st, val) SKM_sk_unshift(GENERAL_NAMES, (st), (v
#define sk_GENERAL_NAMES_find(st, val) SKM_sk_find(GENERAL_NAMES, (st), (val))
#define sk_GENERAL_NAMES_find_ex(st, val) SKM_sk_find_ex(GENERAL_NAMES, (st), (v
#define sk_GENERAL_NAMES_delete(st, i) SKM_sk_delete(GENERAL_NAMES, (st), (i))
#define sk_GENERAL_NAMES_delete_ptr(st, ptr) SKM_sk_delete_ptr(GENERAL_NAMES, (s
#define sk_GENERAL_NAMES_insert(st, val, i) SKM_sk_insert(GENERAL_NAMES, (st), (
#define sk_GENERAL_NAMES_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(GENERAL_NAMES
#define sk_GENERAL_NAMES_dup(st) SKM_sk_dup(GENERAL_NAMES, st)
#define sk_GENERAL_NAMES_pop_free(st, free_func) SKM_sk_pop_free(GENERAL_NAMES,
#define sk_GENERAL_NAMES_shift(st) SKM_sk_shift(GENERAL_NAMES, (st))
#define sk_GENERAL_NAMES_pop(st) SKM_sk_pop(GENERAL_NAMES, (st))
#define sk_GENERAL_NAMES_sort(st) SKM_sk_sort(GENERAL_NAMES, (st))
#define sk_GENERAL_NAMES_is_sorted(st) SKM_sk_is_sorted(GENERAL_NAMES, (st))

#define sk_GENERAL_SUBTREE_new(cmp) SKM_sk_new(GENERAL_SUBTREE, (cmp))
#define sk_GENERAL_SUBTREE_new_null() SKM_sk_new_null(GENERAL_SUBTREE)
#define sk_GENERAL_SUBTREE_free(st) SKM_sk_free(GENERAL_SUBTREE, (st))
#define sk_GENERAL_SUBTREE_num(st) SKM_sk_num(GENERAL_SUBTREE, (st))
#define sk_GENERAL_SUBTREE_value(st, i) SKM_sk_value(GENERAL_SUBTREE, (st), (i))
#define sk_GENERAL_SUBTREE_set(st, i, val) SKM_sk_set(GENERAL_SUBTREE, (st), (i)
#define sk_GENERAL_SUBTREE_zero(st) SKM_sk_zero(GENERAL_SUBTREE, (st))
#define sk_GENERAL_SUBTREE_push(st, val) SKM_sk_push(GENERAL_SUBTREE, (st), (val
#define sk_GENERAL_SUBTREE_unshift(st, val) SKM_sk_unshift(GENERAL_SUBTREE, (st)
#define sk_GENERAL_SUBTREE_find(st, val) SKM_sk_find(GENERAL_SUBTREE, (st), (val
#define sk_GENERAL_SUBTREE_find_ex(st, val) SKM_sk_find_ex(GENERAL_SUBTREE, (st)
#define sk_GENERAL_SUBTREE_delete(st, i) SKM_sk_delete(GENERAL_SUBTREE, (st), (i
#define sk_GENERAL_SUBTREE_delete_ptr(st, ptr) SKM_sk_delete_ptr(GENERAL_SUBTREE
#define sk_GENERAL_SUBTREE_insert(st, val, i) SKM_sk_insert(GENERAL_SUBTREE, (st
#define sk_GENERAL_SUBTREE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(GENERAL_SUB
#define sk_GENERAL_SUBTREE_dup(st) SKM_sk_dup(GENERAL_SUBTREE, st)
#define sk_GENERAL_SUBTREE_pop_free(st, free_func) SKM_sk_pop_free(GENERAL_SUBTR
#define sk_GENERAL_SUBTREE_shift(st) SKM_sk_shift(GENERAL_SUBTREE, (st))
#define sk_GENERAL_SUBTREE_pop(st) SKM_sk_pop(GENERAL_SUBTREE, (st))
#define sk_GENERAL_SUBTREE_sort(st) SKM_sk_sort(GENERAL_SUBTREE, (st))
#define sk_GENERAL_SUBTREE_is_sorted(st) SKM_sk_is_sorted(GENERAL_SUBTREE, (st))

#define sk_IPAddressFamily_new(cmp) SKM_sk_new(IPAddressFamily, (cmp))
#define sk_IPAddressFamily_new_null() SKM_sk_new_null(IPAddressFamily)
#define sk_IPAddressFamily_free(st) SKM_sk_free(IPAddressFamily, (st))
#define sk_IPAddressFamily_num(st) SKM_sk_num(IPAddressFamily, (st))
#define sk_IPAddressFamily_value(st, i) SKM_sk_value(IPAddressFamily, (st), (i))
#define sk_IPAddressFamily_set(st, i, val) SKM_sk_set(IPAddressFamily, (st), (i)
#define sk_IPAddressFamily_zero(st) SKM_sk_zero(IPAddressFamily, (st))
#define sk_IPAddressFamily_push(st, val) SKM_sk_push(IPAddressFamily, (st), (val
```

```
#define sk_IPAddressFamily_unshift(st, val) SKM_sk_unshift(IPAddressFamily, (st)
#define sk_IPAddressFamily_find(st, val) SKM_sk_find(IPAddressFamily, (st), (val
#define sk_IPAddressFamily_find_ex(st, val) SKM_sk_find_ex(IPAddressFamily, (st)
#define sk_IPAddressFamily_delete(st, i) SKM_sk_delete(IPAddressFamily, (st), (i
#define sk_IPAddressFamily_delete_ptr(st, ptr) SKM_sk_delete_ptr(IPAddressFamily
#define sk_IPAddressFamily_insert(st, val, i) SKM_sk_insert(IPAddressFamily, (st
#define sk_IPAddressFamily_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(IPAddressFa
#define sk_IPAddressFamily_dup(st) SKM_sk_dup(IPAddressFamily, st)
#define sk_IPAddressFamily_pop_free(st, free_func) SKM_sk_pop_free(IPAddressFami
#define sk_IPAddressFamily_shift(st) SKM_sk_shift(IPAddressFamily, (st))
#define sk_IPAddressFamily_pop(st) SKM_sk_pop(IPAddressFamily, (st))
#define sk_IPAddressFamily_sort(st) SKM_sk_sort(IPAddressFamily, (st))
#define sk_IPAddressFamily_is_sorted(st) SKM_sk_is_sorted(IPAddressFamily, (st))

#define sk_IPAddressOrRange_new(cmp) SKM_sk_new(IPAddressOrRange, (cmp))
#define sk_IPAddressOrRange_new_null() SKM_sk_new_null(IPAddressOrRange)
#define sk_IPAddressOrRange_free(st) SKM_sk_free(IPAddressOrRange, (st))
#define sk_IPAddressOrRange_num(st) SKM_sk_num(IPAddressOrRange, (st))
#define sk_IPAddressOrRange_value(st, i) SKM_sk_value(IPAddressOrRange, (st), (i
#define sk_IPAddressOrRange_set(st, i, val) SKM_sk_set(IPAddressOrRange, (st), (
#define sk_IPAddressOrRange_zero(st) SKM_sk_zero(IPAddressOrRange, (st))
#define sk_IPAddressOrRange_push(st, val) SKM_sk_push(IPAddressOrRange, (st), (v
#define sk_IPAddressOrRange_unshift(st, val) SKM_sk_unshift(IPAddressOrRange, (s
#define sk_IPAddressOrRange_find(st, val) SKM_sk_find(IPAddressOrRange, (st), (v
#define sk_IPAddressOrRange_find_ex(st, val) SKM_sk_find_ex(IPAddressOrRange, (s
#define sk_IPAddressOrRange_delete(st, i) SKM_sk_delete(IPAddressOrRange, (st),
#define sk_IPAddressOrRange_delete_ptr(st, ptr) SKM_sk_delete_ptr(IPAddressOrRan
#define sk_IPAddressOrRange_insert(st, val, i) SKM_sk_insert(IPAddressOrRange, (
#define sk_IPAddressOrRange_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(IPAddressO
#define sk_IPAddressOrRange_dup(st) SKM_sk_dup(IPAddressOrRange, st)
#define sk_IPAddressOrRange_pop_free(st, free_func) SKM_sk_pop_free(IPAddressOrR
#define sk_IPAddressOrRange_shift(st) SKM_sk_shift(IPAddressOrRange, (st))
#define sk_IPAddressOrRange_pop(st) SKM_sk_pop(IPAddressOrRange, (st))
#define sk_IPAddressOrRange_sort(st) SKM_sk_sort(IPAddressOrRange, (st))
#define sk_IPAddressOrRange_is_sorted(st) SKM_sk_is_sorted(IPAddressOrRange, (st

#define sk_KRB5_APREQBODY_new(cmp) SKM_sk_new(KRB5_APREQBODY, (cmp))
#define sk_KRB5_APREQBODY_new_null() SKM_sk_new_null(KRB5_APREQBODY)
#define sk_KRB5_APREQBODY_free(st) SKM_sk_free(KRB5_APREQBODY, (st))
#define sk_KRB5_APREQBODY_num(st) SKM_sk_num(KRB5_APREQBODY, (st))
#define sk_KRB5_APREQBODY_value(st, i) SKM_sk_value(KRB5_APREQBODY, (st), (i))
#define sk_KRB5_APREQBODY_set(st, i, val) SKM_sk_set(KRB5_APREQBODY, (st), (i),
#define sk_KRB5_APREQBODY_zero(st) SKM_sk_zero(KRB5_APREQBODY, (st))
#define sk_KRB5_APREQBODY_push(st, val) SKM_sk_push(KRB5_APREQBODY, (st), (val))
#define sk_KRB5_APREQBODY_unshift(st, val) SKM_sk_unshift(KRB5_APREQBODY, (st),
#define sk_KRB5_APREQBODY_find(st, val) SKM_sk_find(KRB5_APREQBODY, (st), (val))
#define sk_KRB5_APREQBODY_find_ex(st, val) SKM_sk_find_ex(KRB5_APREQBODY, (st),
#define sk_KRB5_APREQBODY_delete(st, i) SKM_sk_delete(KRB5_APREQBODY, (st), (i))
#define sk_KRB5_APREQBODY_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_APREQBODY,
#define sk_KRB5_APREQBODY_insert(st, val, i) SKM_sk_insert(KRB5_APREQBODY, (st),
#define sk_KRB5_APREQBODY_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_APREQBO
#define sk_KRB5_APREQBODY_dup(st) SKM_sk_dup(KRB5_APREQBODY, st)
#define sk_KRB5_APREQBODY_pop_free(st, free_func) SKM_sk_pop_free(KRB5_APREQBODY
#define sk_KRB5_APREQBODY_shift(st) SKM_sk_shift(KRB5_APREQBODY, (st))
#define sk_KRB5_APREQBODY_pop(st) SKM_sk_pop(KRB5_APREQBODY, (st))
#define sk_KRB5_APREQBODY_sort(st) SKM_sk_sort(KRB5_APREQBODY, (st))
#define sk_KRB5_APREQBODY_is_sorted(st) SKM_sk_is_sorted(KRB5_APREQBODY, (st))

#define sk_KRB5_AUTHDATA_new(cmp) SKM_sk_new(KRB5_AUTHDATA, (cmp))
#define sk_KRB5_AUTHDATA_new_null() SKM_sk_new_null(KRB5_AUTHDATA)
#define sk_KRB5_AUTHDATA_free(st) SKM_sk_free(KRB5_AUTHDATA, (st))
#define sk_KRB5_AUTHDATA_num(st) SKM_sk_num(KRB5_AUTHDATA, (st))
#define sk_KRB5_AUTHDATA_value(st, i) SKM_sk_value(KRB5_AUTHDATA, (st), (i))
#define sk_KRB5_AUTHDATA_set(st, i, val) SKM_sk_set(KRB5_AUTHDATA, (st), (i), (v
#define sk_KRB5_AUTHDATA_zero(st) SKM_sk_zero(KRB5_AUTHDATA, (st))
#define sk_KRB5_AUTHDATA_push(st, val) SKM_sk_push(KRB5_AUTHDATA, (st), (val))
```



```
#define sk_KRB5_AUTHDATA_unshift(st, val) SKM_sk_unshift(KRB5_AUTHDATA, (st), (v
#define sk_KRB5_AUTHDATA_find(st, val) SKM_sk_find(KRB5_AUTHDATA, (st), (val))
#define sk_KRB5_AUTHDATA_find_ex(st, val) SKM_sk_find_ex(KRB5_AUTHDATA, (st), (v
#define sk_KRB5_AUTHDATA_delete(st, i) SKM_sk_delete(KRB5_AUTHDATA, (st), (i))
#define sk_KRB5_AUTHDATA_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_AUTHDATA, (s
#define sk_KRB5_AUTHDATA_insert(st, val, i) SKM_sk_insert(KRB5_AUTHDATA, (st), (
#define sk_KRB5_AUTHDATA_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_AUTHDATA
#define sk_KRB5_AUTHDATA_dup(st) SKM_sk_dup(KRB5_AUTHDATA, st)
#define sk_KRB5_AUTHDATA_pop_free(st, free_func) SKM_sk_pop_free(KRB5_AUTHDATA,
#define sk_KRB5_AUTHDATA_shift(st) SKM_sk_shift(KRB5_AUTHDATA, (st))
#define sk_KRB5_AUTHDATA_pop(st) SKM_sk_pop(KRB5_AUTHDATA, (st))
#define sk_KRB5_AUTHDATA_sort(st) SKM_sk_sort(KRB5_AUTHDATA, (st))
#define sk_KRB5_AUTHDATA_is_sorted(st) SKM_sk_is_sorted(KRB5_AUTHDATA, (st))

#define sk_KRB5_AUTHENTBODY_new(cmp) SKM_sk_new(KRB5_AUTHENTBODY, (cmp))
#define sk_KRB5_AUTHENTBODY_new_null() SKM_sk_new_null(KRB5_AUTHENTBODY)
#define sk_KRB5_AUTHENTBODY_free(st) SKM_sk_free(KRB5_AUTHENTBODY, (st))
#define sk_KRB5_AUTHENTBODY_num(st) SKM_sk_num(KRB5_AUTHENTBODY, (st))
#define sk_KRB5_AUTHENTBODY_value(st, i) SKM_sk_value(KRB5_AUTHENTBODY, (st), (i
#define sk_KRB5_AUTHENTBODY_set(st, i, val) SKM_sk_set(KRB5_AUTHENTBODY, (st), (
#define sk_KRB5_AUTHENTBODY_zero(st) SKM_sk_zero(KRB5_AUTHENTBODY, (st))
#define sk_KRB5_AUTHENTBODY_push(st, val) SKM_sk_push(KRB5_AUTHENTBODY, (st), (v
#define sk_KRB5_AUTHENTBODY_unshift(st, val) SKM_sk_unshift(KRB5_AUTHENTBODY, (s
#define sk_KRB5_AUTHENTBODY_find(st, val) SKM_sk_find(KRB5_AUTHENTBODY, (st), (v
#define sk_KRB5_AUTHENTBODY_find_ex(st, val) SKM_sk_find_ex(KRB5_AUTHENTBODY, (s
#define sk_KRB5_AUTHENTBODY_delete(st, i) SKM_sk_delete(KRB5_AUTHENTBODY, (st),
#define sk_KRB5_AUTHENTBODY_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_AUTHENTBO
#define sk_KRB5_AUTHENTBODY_insert(st, val, i) SKM_sk_insert(KRB5_AUTHENTBODY, (
#define sk_KRB5_AUTHENTBODY_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_AUTHE
#define sk_KRB5_AUTHENTBODY_dup(st) SKM_sk_dup(KRB5_AUTHENTBODY, st)
#define sk_KRB5_AUTHENTBODY_pop_free(st, free_func) SKM_sk_pop_free(KRB5_AUTHENT
#define sk_KRB5_AUTHENTBODY_shift(st) SKM_sk_shift(KRB5_AUTHENTBODY, (st))
#define sk_KRB5_AUTHENTBODY_pop(st) SKM_sk_pop(KRB5_AUTHENTBODY, (st))
#define sk_KRB5_AUTHENTBODY_sort(st) SKM_sk_sort(KRB5_AUTHENTBODY, (st))
#define sk_KRB5_AUTHENTBODY_is_sorted(st) SKM_sk_is_sorted(KRB5_AUTHENTBODY, (st

#define sk_KRB5_CHECKSUM_new(cmp) SKM_sk_new(KRB5_CHECKSUM, (cmp))
#define sk_KRB5_CHECKSUM_new_null() SKM_sk_new_null(KRB5_CHECKSUM)
#define sk_KRB5_CHECKSUM_free(st) SKM_sk_free(KRB5_CHECKSUM, (st))
#define sk_KRB5_CHECKSUM_num(st) SKM_sk_num(KRB5_CHECKSUM, (st))
#define sk_KRB5_CHECKSUM_value(st, i) SKM_sk_value(KRB5_CHECKSUM, (st), (i))
#define sk_KRB5_CHECKSUM_set(st, i, val) SKM_sk_set(KRB5_CHECKSUM, (st), (i), (v
#define sk_KRB5_CHECKSUM_zero(st) SKM_sk_zero(KRB5_CHECKSUM, (st))
#define sk_KRB5_CHECKSUM_push(st, val) SKM_sk_push(KRB5_CHECKSUM, (st), (val))
#define sk_KRB5_CHECKSUM_unshift(st, val) SKM_sk_unshift(KRB5_CHECKSUM, (st), (v
#define sk_KRB5_CHECKSUM_find(st, val) SKM_sk_find(KRB5_CHECKSUM, (st), (val))
#define sk_KRB5_CHECKSUM_find_ex(st, val) SKM_sk_find_ex(KRB5_CHECKSUM, (st), (v
#define sk_KRB5_CHECKSUM_delete(st, i) SKM_sk_delete(KRB5_CHECKSUM, (st), (i))
#define sk_KRB5_CHECKSUM_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_CHECKSUM, (s
#define sk_KRB5_CHECKSUM_insert(st, val, i) SKM_sk_insert(KRB5_CHECKSUM, (st), (
#define sk_KRB5_CHECKSUM_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_CHECKSUM
#define sk_KRB5_CHECKSUM_dup(st) SKM_sk_dup(KRB5_CHECKSUM, st)
#define sk_KRB5_CHECKSUM_pop_free(st, free_func) SKM_sk_pop_free(KRB5_CHECKSUM,
#define sk_KRB5_CHECKSUM_shift(st) SKM_sk_shift(KRB5_CHECKSUM, (st))
#define sk_KRB5_CHECKSUM_pop(st) SKM_sk_pop(KRB5_CHECKSUM, (st))
#define sk_KRB5_CHECKSUM_sort(st) SKM_sk_sort(KRB5_CHECKSUM, (st))
#define sk_KRB5_CHECKSUM_is_sorted(st) SKM_sk_is_sorted(KRB5_CHECKSUM, (st))

#define sk_KRB5_ENCDATA_new(cmp) SKM_sk_new(KRB5_ENCDATA, (cmp))
#define sk_KRB5_ENCDATA_new_null() SKM_sk_new_null(KRB5_ENCDATA)
#define sk_KRB5_ENCDATA_free(st) SKM_sk_free(KRB5_ENCDATA, (st))
#define sk_KRB5_ENCDATA_num(st) SKM_sk_num(KRB5_ENCDATA, (st))
#define sk_KRB5_ENCDATA_value(st, i) SKM_sk_value(KRB5_ENCDATA, (st), (i))
#define sk_KRB5_ENCDATA_set(st, i, val) SKM_sk_set(KRB5_ENCDATA, (st), (i), (val
#define sk_KRB5_ENCDATA_zero(st) SKM_sk_zero(KRB5_ENCDATA, (st))
#define sk_KRB5_ENCDATA_push(st, val) SKM_sk_push(KRB5_ENCDATA, (st), (val))
```

```
#define sk_KRB5_ENCDATA_unshift(st, val) SKM_sk_unshift(KRB5_ENCDATA, (st), (val
#define sk_KRB5_ENCDATA_find(st, val) SKM_sk_find(KRB5_ENCDATA, (st), (val))
#define sk_KRB5_ENCDATA_find_ex(st, val) SKM_sk_find_ex(KRB5_ENCDATA, (st), (val
#define sk_KRB5_ENCDATA_delete(st, i) SKM_sk_delete(KRB5_ENCDATA, (st), (i))
#define sk_KRB5_ENCDATA_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_ENCDATA, (st)
#define sk_KRB5_ENCDATA_insert(st, val, i) SKM_sk_insert(KRB5_ENCDATA, (st), (va
#define sk_KRB5_ENCDATA_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_ENCDATA,
#define sk_KRB5_ENCDATA_dup(st) SKM_sk_dup(KRB5_ENCDATA, st)
#define sk_KRB5_ENCDATA_pop_free(st, free_func) SKM_sk_pop_free(KRB5_ENCDATA, (s
#define sk_KRB5_ENCDATA_shift(st) SKM_sk_shift(KRB5_ENCDATA, (st))
#define sk_KRB5_ENCDATA_pop(st) SKM_sk_pop(KRB5_ENCDATA, (st))
#define sk_KRB5_ENCDATA_sort(st) SKM_sk_sort(KRB5_ENCDATA, (st))
#define sk_KRB5_ENCDATA_is_sorted(st) SKM_sk_is_sorted(KRB5_ENCDATA, (st))

#define sk_KRB5_ENCKEY_new(cmp) SKM_sk_new(KRB5_ENCKEY, (cmp))
#define sk_KRB5_ENCKEY_new_null() SKM_sk_new_null(KRB5_ENCKEY)
#define sk_KRB5_ENCKEY_free(st) SKM_sk_free(KRB5_ENCKEY, (st))
#define sk_KRB5_ENCKEY_num(st) SKM_sk_num(KRB5_ENCKEY, (st))
#define sk_KRB5_ENCKEY_value(st, i) SKM_sk_value(KRB5_ENCKEY, (st), (i))
#define sk_KRB5_ENCKEY_set(st, i, val) SKM_sk_set(KRB5_ENCKEY, (st), (i), (val))
#define sk_KRB5_ENCKEY_zero(st) SKM_sk_zero(KRB5_ENCKEY, (st))
#define sk_KRB5_ENCKEY_push(st, val) SKM_sk_push(KRB5_ENCKEY, (st), (val))
#define sk_KRB5_ENCKEY_unshift(st, val) SKM_sk_unshift(KRB5_ENCKEY, (st), (val))
#define sk_KRB5_ENCKEY_find(st, val) SKM_sk_find(KRB5_ENCKEY, (st), (val))
#define sk_KRB5_ENCKEY_find_ex(st, val) SKM_sk_find_ex(KRB5_ENCKEY, (st), (val))
#define sk_KRB5_ENCKEY_delete(st, i) SKM_sk_delete(KRB5_ENCKEY, (st), (i))
#define sk_KRB5_ENCKEY_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_ENCKEY, (st),
#define sk_KRB5_ENCKEY_insert(st, val, i) SKM_sk_insert(KRB5_ENCKEY, (st), (val)
#define sk_KRB5_ENCKEY_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_ENCKEY, (s
#define sk_KRB5_ENCKEY_dup(st) SKM_sk_dup(KRB5_ENCKEY, st)
#define sk_KRB5_ENCKEY_pop_free(st, free_func) SKM_sk_pop_free(KRB5_ENCKEY, (st)
#define sk_KRB5_ENCKEY_shift(st) SKM_sk_shift(KRB5_ENCKEY, (st))
#define sk_KRB5_ENCKEY_pop(st) SKM_sk_pop(KRB5_ENCKEY, (st))
#define sk_KRB5_ENCKEY_sort(st) SKM_sk_sort(KRB5_ENCKEY, (st))
#define sk_KRB5_ENCKEY_is_sorted(st) SKM_sk_is_sorted(KRB5_ENCKEY, (st))

#define sk_KRB5_PRINCNAME_new(cmp) SKM_sk_new(KRB5_PRINCNAME, (cmp))
#define sk_KRB5_PRINCNAME_new_null() SKM_sk_new_null(KRB5_PRINCNAME)
#define sk_KRB5_PRINCNAME_free(st) SKM_sk_free(KRB5_PRINCNAME, (st))
#define sk_KRB5_PRINCNAME_num(st) SKM_sk_num(KRB5_PRINCNAME, (st))
#define sk_KRB5_PRINCNAME_value(st, i) SKM_sk_value(KRB5_PRINCNAME, (st), (i))
#define sk_KRB5_PRINCNAME_set(st, i, val) SKM_sk_set(KRB5_PRINCNAME, (st), (i),
#define sk_KRB5_PRINCNAME_zero(st) SKM_sk_zero(KRB5_PRINCNAME, (st))
#define sk_KRB5_PRINCNAME_push(st, val) SKM_sk_push(KRB5_PRINCNAME, (st), (val))
#define sk_KRB5_PRINCNAME_unshift(st, val) SKM_sk_unshift(KRB5_PRINCNAME, (st),
#define sk_KRB5_PRINCNAME_find(st, val) SKM_sk_find(KRB5_PRINCNAME, (st), (val))
#define sk_KRB5_PRINCNAME_find_ex(st, val) SKM_sk_find_ex(KRB5_PRINCNAME, (st),
#define sk_KRB5_PRINCNAME_delete(st, i) SKM_sk_delete(KRB5_PRINCNAME, (st), (i))
#define sk_KRB5_PRINCNAME_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_PRINCNAME,
#define sk_KRB5_PRINCNAME_insert(st, val, i) SKM_sk_insert(KRB5_PRINCNAME, (st),
#define sk_KRB5_PRINCNAME_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_PRINCNA
#define sk_KRB5_PRINCNAME_dup(st) SKM_sk_dup(KRB5_PRINCNAME, st)
#define sk_KRB5_PRINCNAME_pop_free(st, free_func) SKM_sk_pop_free(KRB5_PRINCNAME
#define sk_KRB5_PRINCNAME_shift(st) SKM_sk_shift(KRB5_PRINCNAME, (st))
#define sk_KRB5_PRINCNAME_pop(st) SKM_sk_pop(KRB5_PRINCNAME, (st))
#define sk_KRB5_PRINCNAME_sort(st) SKM_sk_sort(KRB5_PRINCNAME, (st))
#define sk_KRB5_PRINCNAME_is_sorted(st) SKM_sk_is_sorted(KRB5_PRINCNAME, (st))

#define sk_KRB5_TKTBODY_new(cmp) SKM_sk_new(KRB5_TKTBODY, (cmp))
#define sk_KRB5_TKTBODY_new_null() SKM_sk_new_null(KRB5_TKTBODY)
#define sk_KRB5_TKTBODY_free(st) SKM_sk_free(KRB5_TKTBODY, (st))
#define sk_KRB5_TKTBODY_num(st) SKM_sk_num(KRB5_TKTBODY, (st))
#define sk_KRB5_TKTBODY_value(st, i) SKM_sk_value(KRB5_TKTBODY, (st), (i))
#define sk_KRB5_TKTBODY_set(st, i, val) SKM_sk_set(KRB5_TKTBODY, (st), (i), (val
#define sk_KRB5_TKTBODY_zero(st) SKM_sk_zero(KRB5_TKTBODY, (st))
#define sk_KRB5_TKTBODY_push(st, val) SKM_sk_push(KRB5_TKTBODY, (st), (val))
```



```
#define sk_KRB5_TKTBODY_unshift(st, val) SKM_sk_unshift(KRB5_TKTBODY, (st), (val
#define sk_KRB5_TKTBODY_find(st, val) SKM_sk_find(KRB5_TKTBODY, (st), (val))
#define sk_KRB5_TKTBODY_find_ex(st, val) SKM_sk_find_ex(KRB5_TKTBODY, (st), (val
#define sk_KRB5_TKTBODY_delete(st, i) SKM_sk_delete(KRB5_TKTBODY, (st), (i))
#define sk_KRB5_TKTBODY_delete_ptr(st, ptr) SKM_sk_delete_ptr(KRB5_TKTBODY, (st)
#define sk_KRB5_TKTBODY_insert(st, val, i) SKM_sk_insert(KRB5_TKTBODY, (st), (va
#define sk_KRB5_TKTBODY_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(KRB5_TKTBODY,
#define sk_KRB5_TKTBODY_dup(st) SKM_sk_dup(KRB5_TKTBODY, st)
#define sk_KRB5_TKTBODY_pop_free(st, free_func) SKM_sk_pop_free(KRB5_TKTBODY, (s
#define sk_KRB5_TKTBODY_shift(st) SKM_sk_shift(KRB5_TKTBODY, (st))
#define sk_KRB5_TKTBODY_pop(st) SKM_sk_pop(KRB5_TKTBODY, (st))
#define sk_KRB5_TKTBODY_sort(st) SKM_sk_sort(KRB5_TKTBODY, (st))
#define sk_KRB5_TKTBODY_is_sorted(st) SKM_sk_is_sorted(KRB5_TKTBODY, (st))

#define sk_MEM_OBJECT_DATA_new(cmp) SKM_sk_new(MEM_OBJECT_DATA, (cmp))
#define sk_MEM_OBJECT_DATA_new_null() SKM_sk_new_null(MEM_OBJECT_DATA)
#define sk_MEM_OBJECT_DATA_free(st) SKM_sk_free(MEM_OBJECT_DATA, (st))
#define sk_MEM_OBJECT_DATA_num(st) SKM_sk_num(MEM_OBJECT_DATA, (st))
#define sk_MEM_OBJECT_DATA_value(st, i) SKM_sk_value(MEM_OBJECT_DATA, (st), (i))
#define sk_MEM_OBJECT_DATA_set(st, i, val) SKM_sk_set(MEM_OBJECT_DATA, (st), (i)
#define sk_MEM_OBJECT_DATA_zero(st) SKM_sk_zero(MEM_OBJECT_DATA, (st))
#define sk_MEM_OBJECT_DATA_push(st, val) SKM_sk_push(MEM_OBJECT_DATA, (st), (val
#define sk_MEM_OBJECT_DATA_unshift(st, val) SKM_sk_unshift(MEM_OBJECT_DATA, (st)
#define sk_MEM_OBJECT_DATA_find(st, val) SKM_sk_find(MEM_OBJECT_DATA, (st), (val
#define sk_MEM_OBJECT_DATA_find_ex(st, val) SKM_sk_find_ex(MEM_OBJECT_DATA, (st)
#define sk_MEM_OBJECT_DATA_delete(st, i) SKM_sk_delete(MEM_OBJECT_DATA, (st), (i
#define sk_MEM_OBJECT_DATA_delete_ptr(st, ptr) SKM_sk_delete_ptr(MEM_OBJECT_DATA
#define sk_MEM_OBJECT_DATA_insert(st, val, i) SKM_sk_insert(MEM_OBJECT_DATA, (st
#define sk_MEM_OBJECT_DATA_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(MEM_OBJECT_
#define sk_MEM_OBJECT_DATA_dup(st) SKM_sk_dup(MEM_OBJECT_DATA, st)
#define sk_MEM_OBJECT_DATA_pop_free(st, free_func) SKM_sk_pop_free(MEM_OBJECT_DA
#define sk_MEM_OBJECT_DATA_shift(st) SKM_sk_shift(MEM_OBJECT_DATA, (st))
#define sk_MEM_OBJECT_DATA_pop(st) SKM_sk_pop(MEM_OBJECT_DATA, (st))
#define sk_MEM_OBJECT_DATA_sort(st) SKM_sk_sort(MEM_OBJECT_DATA, (st))
#define sk_MEM_OBJECT_DATA_is_sorted(st) SKM_sk_is_sorted(MEM_OBJECT_DATA, (st))

#define sk_MIME_HEADER_new(cmp) SKM_sk_new(MIME_HEADER, (cmp))
#define sk_MIME_HEADER_new_null() SKM_sk_new_null(MIME_HEADER)
#define sk_MIME_HEADER_free(st) SKM_sk_free(MIME_HEADER, (st))
#define sk_MIME_HEADER_num(st) SKM_sk_num(MIME_HEADER, (st))
#define sk_MIME_HEADER_value(st, i) SKM_sk_value(MIME_HEADER, (st), (i))
#define sk_MIME_HEADER_set(st, i, val) SKM_sk_set(MIME_HEADER, (st), (i), (val))
#define sk_MIME_HEADER_zero(st) SKM_sk_zero(MIME_HEADER, (st))
#define sk_MIME_HEADER_push(st, val) SKM_sk_push(MIME_HEADER, (st), (val))
#define sk_MIME_HEADER_unshift(st, val) SKM_sk_unshift(MIME_HEADER, (st), (val))
#define sk_MIME_HEADER_find(st, val) SKM_sk_find(MIME_HEADER, (st), (val))
#define sk_MIME_HEADER_find_ex(st, val) SKM_sk_find_ex(MIME_HEADER, (st), (val))
#define sk_MIME_HEADER_delete(st, i) SKM_sk_delete(MIME_HEADER, (st), (i))
#define sk_MIME_HEADER_delete_ptr(st, ptr) SKM_sk_delete_ptr(MIME_HEADER, (st),
#define sk_MIME_HEADER_insert(st, val, i) SKM_sk_insert(MIME_HEADER, (st), (val)
#define sk_MIME_HEADER_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(MIME_HEADER, (s
#define sk_MIME_HEADER_dup(st) SKM_sk_dup(MIME_HEADER, st)
#define sk_MIME_HEADER_pop_free(st, free_func) SKM_sk_pop_free(MIME_HEADER, (st)
#define sk_MIME_HEADER_shift(st) SKM_sk_shift(MIME_HEADER, (st))
#define sk_MIME_HEADER_pop(st) SKM_sk_pop(MIME_HEADER, (st))
#define sk_MIME_HEADER_sort(st) SKM_sk_sort(MIME_HEADER, (st))
#define sk_MIME_HEADER_is_sorted(st) SKM_sk_is_sorted(MIME_HEADER, (st))

#define sk_MIME_PARAM_new(cmp) SKM_sk_new(MIME_PARAM, (cmp))
#define sk_MIME_PARAM_new_null() SKM_sk_new_null(MIME_PARAM)
#define sk_MIME_PARAM_free(st) SKM_sk_free(MIME_PARAM, (st))
#define sk_MIME_PARAM_num(st) SKM_sk_num(MIME_PARAM, (st))
#define sk_MIME_PARAM_value(st, i) SKM_sk_value(MIME_PARAM, (st), (i))
#define sk_MIME_PARAM_set(st, i, val) SKM_sk_set(MIME_PARAM, (st), (i), (val))
#define sk_MIME_PARAM_zero(st) SKM_sk_zero(MIME_PARAM, (st))
#define sk_MIME_PARAM_push(st, val) SKM_sk_push(MIME_PARAM, (st), (val))
```

```
#define sk_MIME_PARAM_unshift(st, val) SKM_sk_unshift(MIME_PARAM, (st), (val))
#define sk_MIME_PARAM_find(st, val) SKM_sk_find(MIME_PARAM, (st), (val))
#define sk_MIME_PARAM_find_ex(st, val) SKM_sk_find_ex(MIME_PARAM, (st), (val))
#define sk_MIME_PARAM_delete(st, i) SKM_sk_delete(MIME_PARAM, (st), (i))
#define sk_MIME_PARAM_delete_ptr(st, ptr) SKM_sk_delete_ptr(MIME_PARAM, (st), (p
#define sk_MIME_PARAM_insert(st, val, i) SKM_sk_insert(MIME_PARAM, (st), (val),
#define sk_MIME_PARAM_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(MIME_PARAM, (st)
#define sk_MIME_PARAM_dup(st) SKM_sk_dup(MIME_PARAM, st)
#define sk_MIME_PARAM_pop_free(st, free_func) SKM_sk_pop_free(MIME_PARAM, (st),
#define sk_MIME_PARAM_shift(st) SKM_sk_shift(MIME_PARAM, (st))
#define sk_MIME_PARAM_pop(st) SKM_sk_pop(MIME_PARAM, (st))
#define sk_MIME_PARAM_sort(st) SKM_sk_sort(MIME_PARAM, (st))
#define sk_MIME_PARAM_is_sorted(st) SKM_sk_is_sorted(MIME_PARAM, (st))

#define sk_NAME_FUNCS_new(cmp) SKM_sk_new(NAME_FUNCS, (cmp))
#define sk_NAME_FUNCS_new_null() SKM_sk_new_null(NAME_FUNCS)
#define sk_NAME_FUNCS_free(st) SKM_sk_free(NAME_FUNCS, (st))
#define sk_NAME_FUNCS_num(st) SKM_sk_num(NAME_FUNCS, (st))
#define sk_NAME_FUNCS_value(st, i) SKM_sk_value(NAME_FUNCS, (st), (i))
#define sk_NAME_FUNCS_set(st, i, val) SKM_sk_set(NAME_FUNCS, (st), (i), (val))
#define sk_NAME_FUNCS_zero(st) SKM_sk_zero(NAME_FUNCS, (st))
#define sk_NAME_FUNCS_push(st, val) SKM_sk_push(NAME_FUNCS, (st), (val))
#define sk_NAME_FUNCS_unshift(st, val) SKM_sk_unshift(NAME_FUNCS, (st), (val))
#define sk_NAME_FUNCS_find(st, val) SKM_sk_find(NAME_FUNCS, (st), (val))
#define sk_NAME_FUNCS_find_ex(st, val) SKM_sk_find_ex(NAME_FUNCS, (st), (val))
#define sk_NAME_FUNCS_delete(st, i) SKM_sk_delete(NAME_FUNCS, (st), (i))
#define sk_NAME_FUNCS_delete_ptr(st, ptr) SKM_sk_delete_ptr(NAME_FUNCS, (st), (p
#define sk_NAME_FUNCS_insert(st, val, i) SKM_sk_insert(NAME_FUNCS, (st), (val),
#define sk_NAME_FUNCS_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(NAME_FUNCS, (st)
#define sk_NAME_FUNCS_dup(st) SKM_sk_dup(NAME_FUNCS, st)
#define sk_NAME_FUNCS_pop_free(st, free_func) SKM_sk_pop_free(NAME_FUNCS, (st),
#define sk_NAME_FUNCS_shift(st) SKM_sk_shift(NAME_FUNCS, (st))
#define sk_NAME_FUNCS_pop(st) SKM_sk_pop(NAME_FUNCS, (st))
#define sk_NAME_FUNCS_sort(st) SKM_sk_sort(NAME_FUNCS, (st))
#define sk_NAME_FUNCS_is_sorted(st) SKM_sk_is_sorted(NAME_FUNCS, (st))

#define sk_OCSP_CERTID_new(cmp) SKM_sk_new(OCSP_CERTID, (cmp))
#define sk_OCSP_CERTID_new_null() SKM_sk_new_null(OCSP_CERTID)
#define sk_OCSP_CERTID_free(st) SKM_sk_free(OCSP_CERTID, (st))
#define sk_OCSP_CERTID_num(st) SKM_sk_num(OCSP_CERTID, (st))
#define sk_OCSP_CERTID_value(st, i) SKM_sk_value(OCSP_CERTID, (st), (i))
#define sk_OCSP_CERTID_set(st, i, val) SKM_sk_set(OCSP_CERTID, (st), (i), (val))
#define sk_OCSP_CERTID_zero(st) SKM_sk_zero(OCSP_CERTID, (st))
#define sk_OCSP_CERTID_push(st, val) SKM_sk_push(OCSP_CERTID, (st), (val))
#define sk_OCSP_CERTID_unshift(st, val) SKM_sk_unshift(OCSP_CERTID, (st), (val))
#define sk_OCSP_CERTID_find(st, val) SKM_sk_find(OCSP_CERTID, (st), (val))
#define sk_OCSP_CERTID_find_ex(st, val) SKM_sk_find_ex(OCSP_CERTID, (st), (val))
#define sk_OCSP_CERTID_delete(st, i) SKM_sk_delete(OCSP_CERTID, (st), (i))
#define sk_OCSP_CERTID_delete_ptr(st, ptr) SKM_sk_delete_ptr(OCSP_CERTID, (st),
#define sk_OCSP_CERTID_insert(st, val, i) SKM_sk_insert(OCSP_CERTID, (st), (val)
#define sk_OCSP_CERTID_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(OCSP_CERTID, (s
#define sk_OCSP_CERTID_dup(st) SKM_sk_dup(OCSP_CERTID, st)
#define sk_OCSP_CERTID_pop_free(st, free_func) SKM_sk_pop_free(OCSP_CERTID, (st)
#define sk_OCSP_CERTID_shift(st) SKM_sk_shift(OCSP_CERTID, (st))
#define sk_OCSP_CERTID_pop(st) SKM_sk_pop(OCSP_CERTID, (st))
#define sk_OCSP_CERTID_sort(st) SKM_sk_sort(OCSP_CERTID, (st))
#define sk_OCSP_CERTID_is_sorted(st) SKM_sk_is_sorted(OCSP_CERTID, (st))

#define sk_OCSP_ONEREQ_new(cmp) SKM_sk_new(OCSP_ONEREQ, (cmp))
#define sk_OCSP_ONEREQ_new_null() SKM_sk_new_null(OCSP_ONEREQ)
#define sk_OCSP_ONEREQ_free(st) SKM_sk_free(OCSP_ONEREQ, (st))
#define sk_OCSP_ONEREQ_num(st) SKM_sk_num(OCSP_ONEREQ, (st))
#define sk_OCSP_ONEREQ_value(st, i) SKM_sk_value(OCSP_ONEREQ, (st), (i))
#define sk_OCSP_ONEREQ_set(st, i, val) SKM_sk_set(OCSP_ONEREQ, (st), (i), (val))
#define sk_OCSP_ONEREQ_zero(st) SKM_sk_zero(OCSP_ONEREQ, (st))
#define sk_OCSP_ONEREQ_push(st, val) SKM_sk_push(OCSP_ONEREQ, (st), (val))
```



```
#define sk_OCSP_ONEREQ_unshift(st, val) SKM_sk_unshift(OCSP_ONEREQ, (st), (val))
#define sk_OCSP_ONEREQ_find(st, val) SKM_sk_find(OCSP_ONEREQ, (st), (val))
#define sk_OCSP_ONEREQ_find_ex(st, val) SKM_sk_find_ex(OCSP_ONEREQ, (st), (val))
#define sk_OCSP_ONEREQ_delete(st, i) SKM_sk_delete(OCSP_ONEREQ, (st), (i))
#define sk_OCSP_ONEREQ_delete_ptr(st, ptr) SKM_sk_delete_ptr(OCSP_ONEREQ, (st),
#define sk_OCSP_ONEREQ_insert(st, val, i) SKM_sk_insert(OCSP_ONEREQ, (st), (val)
#define sk_OCSP_ONEREQ_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(OCSP_ONEREQ, (s
#define sk_OCSP_ONEREQ_dup(st) SKM_sk_dup(OCSP_ONEREQ, st)
#define sk_OCSP_ONEREQ_pop_free(st, free_func) SKM_sk_pop_free(OCSP_ONEREQ, (st)
#define sk_OCSP_ONEREQ_shift(st) SKM_sk_shift(OCSP_ONEREQ, (st))
#define sk_OCSP_ONEREQ_pop(st) SKM_sk_pop(OCSP_ONEREQ, (st))
#define sk_OCSP_ONEREQ_sort(st) SKM_sk_sort(OCSP_ONEREQ, (st))
#define sk_OCSP_ONEREQ_is_sorted(st) SKM_sk_is_sorted(OCSP_ONEREQ, (st))

#define sk_OCSP_RESPID_new(cmp) SKM_sk_new(OCSP_RESPID, (cmp))
#define sk_OCSP_RESPID_new_null() SKM_sk_new_null(OCSP_RESPID)
#define sk_OCSP_RESPID_free(st) SKM_sk_free(OCSP_RESPID, (st))
#define sk_OCSP_RESPID_num(st) SKM_sk_num(OCSP_RESPID, (st))
#define sk_OCSP_RESPID_value(st, i) SKM_sk_value(OCSP_RESPID, (st), (i))
#define sk_OCSP_RESPID_set(st, i, val) SKM_sk_set(OCSP_RESPID, (st), (i), (val))
#define sk_OCSP_RESPID_zero(st) SKM_sk_zero(OCSP_RESPID, (st))
#define sk_OCSP_RESPID_push(st, val) SKM_sk_push(OCSP_RESPID, (st), (val))
#define sk_OCSP_RESPID_unshift(st, val) SKM_sk_unshift(OCSP_RESPID, (st), (val))
#define sk_OCSP_RESPID_find(st, val) SKM_sk_find(OCSP_RESPID, (st), (val))
#define sk_OCSP_RESPID_find_ex(st, val) SKM_sk_find_ex(OCSP_RESPID, (st), (val))
#define sk_OCSP_RESPID_delete(st, i) SKM_sk_delete(OCSP_RESPID, (st), (i))
#define sk_OCSP_RESPID_delete_ptr(st, ptr) SKM_sk_delete_ptr(OCSP_RESPID, (st),
#define sk_OCSP_RESPID_insert(st, val, i) SKM_sk_insert(OCSP_RESPID, (st), (val)
#define sk_OCSP_RESPID_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(OCSP_RESPID, (s
#define sk_OCSP_RESPID_dup(st) SKM_sk_dup(OCSP_RESPID, st)
#define sk_OCSP_RESPID_pop_free(st, free_func) SKM_sk_pop_free(OCSP_RESPID, (st)
#define sk_OCSP_RESPID_shift(st) SKM_sk_shift(OCSP_RESPID, (st))
#define sk_OCSP_RESPID_pop(st) SKM_sk_pop(OCSP_RESPID, (st))
#define sk_OCSP_RESPID_sort(st) SKM_sk_sort(OCSP_RESPID, (st))
#define sk_OCSP_RESPID_is_sorted(st) SKM_sk_is_sorted(OCSP_RESPID, (st))

#define sk_OCSP_SINGLERESP_new(cmp) SKM_sk_new(OCSP_SINGLERESP, (cmp))
#define sk_OCSP_SINGLERESP_new_null() SKM_sk_new_null(OCSP_SINGLERESP)
#define sk_OCSP_SINGLERESP_free(st) SKM_sk_free(OCSP_SINGLERESP, (st))
#define sk_OCSP_SINGLERESP_num(st) SKM_sk_num(OCSP_SINGLERESP, (st))
#define sk_OCSP_SINGLERESP_value(st, i) SKM_sk_value(OCSP_SINGLERESP, (st), (i))
#define sk_OCSP_SINGLERESP_set(st, i, val) SKM_sk_set(OCSP_SINGLERESP, (st), (i)
#define sk_OCSP_SINGLERESP_zero(st) SKM_sk_zero(OCSP_SINGLERESP, (st))
#define sk_OCSP_SINGLERESP_push(st, val) SKM_sk_push(OCSP_SINGLERESP, (st), (val
#define sk_OCSP_SINGLERESP_unshift(st, val) SKM_sk_unshift(OCSP_SINGLERESP, (st)
#define sk_OCSP_SINGLERESP_find(st, val) SKM_sk_find(OCSP_SINGLERESP, (st), (val
#define sk_OCSP_SINGLERESP_find_ex(st, val) SKM_sk_find_ex(OCSP_SINGLERESP, (st)
#define sk_OCSP_SINGLERESP_delete(st, i) SKM_sk_delete(OCSP_SINGLERESP, (st), (i
#define sk_OCSP_SINGLERESP_delete_ptr(st, ptr) SKM_sk_delete_ptr(OCSP_SINGLERESP
#define sk_OCSP_SINGLERESP_insert(st, val, i) SKM_sk_insert(OCSP_SINGLERESP, (st
#define sk_OCSP_SINGLERESP_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(OCSP_SINGLE
#define sk_OCSP_SINGLERESP_dup(st) SKM_sk_dup(OCSP_SINGLERESP, st)
#define sk_OCSP_SINGLERESP_pop_free(st, free_func) SKM_sk_pop_free(OCSP_SINGLERE
#define sk_OCSP_SINGLERESP_shift(st) SKM_sk_shift(OCSP_SINGLERESP, (st))
#define sk_OCSP_SINGLERESP_pop(st) SKM_sk_pop(OCSP_SINGLERESP, (st))
#define sk_OCSP_SINGLERESP_sort(st) SKM_sk_sort(OCSP_SINGLERESP, (st))
#define sk_OCSP_SINGLERESP_is_sorted(st) SKM_sk_is_sorted(OCSP_SINGLERESP, (st))

#define sk_PKCS12_SAFEBAG_new(cmp) SKM_sk_new(PKCS12_SAFEBAG, (cmp))
#define sk_PKCS12_SAFEBAG_new_null() SKM_sk_new_null(PKCS12_SAFEBAG)
#define sk_PKCS12_SAFEBAG_free(st) SKM_sk_free(PKCS12_SAFEBAG, (st))
#define sk_PKCS12_SAFEBAG_num(st) SKM_sk_num(PKCS12_SAFEBAG, (st))
#define sk_PKCS12_SAFEBAG_value(st, i) SKM_sk_value(PKCS12_SAFEBAG, (st), (i))
#define sk_PKCS12_SAFEBAG_set(st, i, val) SKM_sk_set(PKCS12_SAFEBAG, (st), (i),
#define sk_PKCS12_SAFEBAG_zero(st) SKM_sk_zero(PKCS12_SAFEBAG, (st))
#define sk_PKCS12_SAFEBAG_push(st, val) SKM_sk_push(PKCS12_SAFEBAG, (st), (val))
```

```
#define sk_PKCS12_SAFEBAG_unshift(st, val) SKM_sk_unshift(PKCS12_SAFEBAG, (st),
#define sk_PKCS12_SAFEBAG_find(st, val) SKM_sk_find(PKCS12_SAFEBAG, (st), (val))
#define sk_PKCS12_SAFEBAG_find_ex(st, val) SKM_sk_find_ex(PKCS12_SAFEBAG, (st),
#define sk_PKCS12_SAFEBAG_delete(st, i) SKM_sk_delete(PKCS12_SAFEBAG, (st), (i))
#define sk_PKCS12_SAFEBAG_delete_ptr(st, ptr) SKM_sk_delete_ptr(PKCS12_SAFEBAG,
#define sk_PKCS12_SAFEBAG_insert(st, val, i) SKM_sk_insert(PKCS12_SAFEBAG, (st),
#define sk_PKCS12_SAFEBAG_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(PKCS12_SAFEB
#define sk_PKCS12_SAFEBAG_dup(st) SKM_sk_dup(PKCS12_SAFEBAG, st)
#define sk_PKCS12_SAFEBAG_pop_free(st, free_func) SKM_sk_pop_free(PKCS12_SAFEBAG
#define sk_PKCS12_SAFEBAG_shift(st) SKM_sk_shift(PKCS12_SAFEBAG, (st))
#define sk_PKCS12_SAFEBAG_pop(st) SKM_sk_pop(PKCS12_SAFEBAG, (st))
#define sk_PKCS12_SAFEBAG_sort(st) SKM_sk_sort(PKCS12_SAFEBAG, (st))
#define sk_PKCS12_SAFEBAG_is_sorted(st) SKM_sk_is_sorted(PKCS12_SAFEBAG, (st))

#define sk_PKCS7_new(cmp) SKM_sk_new(PKCS7, (cmp))
#define sk_PKCS7_new_null() SKM_sk_new_null(PKCS7)
#define sk_PKCS7_free(st) SKM_sk_free(PKCS7, (st))
#define sk_PKCS7_num(st) SKM_sk_num(PKCS7, (st))
#define sk_PKCS7_value(st, i) SKM_sk_value(PKCS7, (st), (i))
#define sk_PKCS7_set(st, i, val) SKM_sk_set(PKCS7, (st), (i), (val))
#define sk_PKCS7_zero(st) SKM_sk_zero(PKCS7, (st))
#define sk_PKCS7_push(st, val) SKM_sk_push(PKCS7, (st), (val))
#define sk_PKCS7_unshift(st, val) SKM_sk_unshift(PKCS7, (st), (val))
#define sk_PKCS7_find(st, val) SKM_sk_find(PKCS7, (st), (val))
#define sk_PKCS7_find_ex(st, val) SKM_sk_find_ex(PKCS7, (st), (val))
#define sk_PKCS7_delete(st, i) SKM_sk_delete(PKCS7, (st), (i))
#define sk_PKCS7_delete_ptr(st, ptr) SKM_sk_delete_ptr(PKCS7, (st), (ptr))
#define sk_PKCS7_insert(st, val, i) SKM_sk_insert(PKCS7, (st), (val), (i))
#define sk_PKCS7_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(PKCS7, (st), (cmp))
#define sk_PKCS7_dup(st) SKM_sk_dup(PKCS7, st)
#define sk_PKCS7_pop_free(st, free_func) SKM_sk_pop_free(PKCS7, (st), (free_func
#define sk_PKCS7_shift(st) SKM_sk_shift(PKCS7, (st))
#define sk_PKCS7_pop(st) SKM_sk_pop(PKCS7, (st))
#define sk_PKCS7_sort(st) SKM_sk_sort(PKCS7, (st))
#define sk_PKCS7_is_sorted(st) SKM_sk_is_sorted(PKCS7, (st))

#define sk_PKCS7_RECIP_INFO_new(cmp) SKM_sk_new(PKCS7_RECIP_INFO, (cmp))
#define sk_PKCS7_RECIP_INFO_new_null() SKM_sk_new_null(PKCS7_RECIP_INFO)
#define sk_PKCS7_RECIP_INFO_free(st) SKM_sk_free(PKCS7_RECIP_INFO, (st))
#define sk_PKCS7_RECIP_INFO_num(st) SKM_sk_num(PKCS7_RECIP_INFO, (st))
#define sk_PKCS7_RECIP_INFO_value(st, i) SKM_sk_value(PKCS7_RECIP_INFO, (st), (i
#define sk_PKCS7_RECIP_INFO_set(st, i, val) SKM_sk_set(PKCS7_RECIP_INFO, (st), (
#define sk_PKCS7_RECIP_INFO_zero(st) SKM_sk_zero(PKCS7_RECIP_INFO, (st))
#define sk_PKCS7_RECIP_INFO_push(st, val) SKM_sk_push(PKCS7_RECIP_INFO, (st), (v
#define sk_PKCS7_RECIP_INFO_unshift(st, val) SKM_sk_unshift(PKCS7_RECIP_INFO, (s
#define sk_PKCS7_RECIP_INFO_find(st, val) SKM_sk_find(PKCS7_RECIP_INFO, (st), (v
#define sk_PKCS7_RECIP_INFO_find_ex(st, val) SKM_sk_find_ex(PKCS7_RECIP_INFO, (s
#define sk_PKCS7_RECIP_INFO_delete(st, i) SKM_sk_delete(PKCS7_RECIP_INFO, (st),
#define sk_PKCS7_RECIP_INFO_delete_ptr(st, ptr) SKM_sk_delete_ptr(PKCS7_RECIP_IN
#define sk_PKCS7_RECIP_INFO_insert(st, val, i) SKM_sk_insert(PKCS7_RECIP_INFO, (
#define sk_PKCS7_RECIP_INFO_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(PKCS7_RECI
#define sk_PKCS7_RECIP_INFO_dup(st) SKM_sk_dup(PKCS7_RECIP_INFO, st)
#define sk_PKCS7_RECIP_INFO_pop_free(st, free_func) SKM_sk_pop_free(PKCS7_RECIP_
#define sk_PKCS7_RECIP_INFO_shift(st) SKM_sk_shift(PKCS7_RECIP_INFO, (st))
#define sk_PKCS7_RECIP_INFO_pop(st) SKM_sk_pop(PKCS7_RECIP_INFO, (st))
#define sk_PKCS7_RECIP_INFO_sort(st) SKM_sk_sort(PKCS7_RECIP_INFO, (st))
#define sk_PKCS7_RECIP_INFO_is_sorted(st) SKM_sk_is_sorted(PKCS7_RECIP_INFO, (st

#define sk_PKCS7_SIGNER_INFO_new(cmp) SKM_sk_new(PKCS7_SIGNER_INFO, (cmp))
#define sk_PKCS7_SIGNER_INFO_new_null() SKM_sk_new_null(PKCS7_SIGNER_INFO)
#define sk_PKCS7_SIGNER_INFO_free(st) SKM_sk_free(PKCS7_SIGNER_INFO, (st))
#define sk_PKCS7_SIGNER_INFO_num(st) SKM_sk_num(PKCS7_SIGNER_INFO, (st))
#define sk_PKCS7_SIGNER_INFO_value(st, i) SKM_sk_value(PKCS7_SIGNER_INFO, (st),
#define sk_PKCS7_SIGNER_INFO_set(st, i, val) SKM_sk_set(PKCS7_SIGNER_INFO, (st),
#define sk_PKCS7_SIGNER_INFO_zero(st) SKM_sk_zero(PKCS7_SIGNER_INFO, (st))
#define sk_PKCS7_SIGNER_INFO_push(st, val) SKM_sk_push(PKCS7_SIGNER_INFO, (st),
```



```
#define sk_PKCS7_SIGNER_INFO_unshift(st, val) SKM_sk_unshift(PKCS7_SIGNER_INFO,
#define sk_PKCS7_SIGNER_INFO_find(st, val) SKM_sk_find(PKCS7_SIGNER_INFO, (st),
#define sk_PKCS7_SIGNER_INFO_find_ex(st, val) SKM_sk_find_ex(PKCS7_SIGNER_INFO,
#define sk_PKCS7_SIGNER_INFO_delete(st, i) SKM_sk_delete(PKCS7_SIGNER_INFO, (st)
#define sk_PKCS7_SIGNER_INFO_delete_ptr(st, ptr) SKM_sk_delete_ptr(PKCS7_SIGNER_
#define sk_PKCS7_SIGNER_INFO_insert(st, val, i) SKM_sk_insert(PKCS7_SIGNER_INFO,
#define sk_PKCS7_SIGNER_INFO_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(PKCS7_SIG
#define sk_PKCS7_SIGNER_INFO_dup(st) SKM_sk_dup(PKCS7_SIGNER_INFO, st)
#define sk_PKCS7_SIGNER_INFO_pop_free(st, free_func) SKM_sk_pop_free(PKCS7_SIGNE
#define sk_PKCS7_SIGNER_INFO_shift(st) SKM_sk_shift(PKCS7_SIGNER_INFO, (st))
#define sk_PKCS7_SIGNER_INFO_pop(st) SKM_sk_pop(PKCS7_SIGNER_INFO, (st))
#define sk_PKCS7_SIGNER_INFO_sort(st) SKM_sk_sort(PKCS7_SIGNER_INFO, (st))
#define sk_PKCS7_SIGNER_INFO_is_sorted(st) SKM_sk_is_sorted(PKCS7_SIGNER_INFO, (

#define sk_POLICYINFO_new(cmp) SKM_sk_new(POLICYINFO, (cmp))
#define sk_POLICYINFO_new_null() SKM_sk_new_null(POLICYINFO)
#define sk_POLICYINFO_free(st) SKM_sk_free(POLICYINFO, (st))
#define sk_POLICYINFO_num(st) SKM_sk_num(POLICYINFO, (st))
#define sk_POLICYINFO_value(st, i) SKM_sk_value(POLICYINFO, (st), (i))
#define sk_POLICYINFO_set(st, i, val) SKM_sk_set(POLICYINFO, (st), (i), (val))
#define sk_POLICYINFO_zero(st) SKM_sk_zero(POLICYINFO, (st))
#define sk_POLICYINFO_push(st, val) SKM_sk_push(POLICYINFO, (st), (val))
#define sk_POLICYINFO_unshift(st, val) SKM_sk_unshift(POLICYINFO, (st), (val))
#define sk_POLICYINFO_find(st, val) SKM_sk_find(POLICYINFO, (st), (val))
#define sk_POLICYINFO_find_ex(st, val) SKM_sk_find_ex(POLICYINFO, (st), (val))
#define sk_POLICYINFO_delete(st, i) SKM_sk_delete(POLICYINFO, (st), (i))
#define sk_POLICYINFO_delete_ptr(st, ptr) SKM_sk_delete_ptr(POLICYINFO, (st), (p
#define sk_POLICYINFO_insert(st, val, i) SKM_sk_insert(POLICYINFO, (st), (val),
#define sk_POLICYINFO_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(POLICYINFO, (st)
#define sk_POLICYINFO_dup(st) SKM_sk_dup(POLICYINFO, st)
#define sk_POLICYINFO_pop_free(st, free_func) SKM_sk_pop_free(POLICYINFO, (st),
#define sk_POLICYINFO_shift(st) SKM_sk_shift(POLICYINFO, (st))
#define sk_POLICYINFO_pop(st) SKM_sk_pop(POLICYINFO, (st))
#define sk_POLICYINFO_sort(st) SKM_sk_sort(POLICYINFO, (st))
#define sk_POLICYINFO_is_sorted(st) SKM_sk_is_sorted(POLICYINFO, (st))

#define sk_POLICYQUALINFO_new(cmp) SKM_sk_new(POLICYQUALINFO, (cmp))
#define sk_POLICYQUALINFO_new_null() SKM_sk_new_null(POLICYQUALINFO)
#define sk_POLICYQUALINFO_free(st) SKM_sk_free(POLICYQUALINFO, (st))
#define sk_POLICYQUALINFO_num(st) SKM_sk_num(POLICYQUALINFO, (st))
#define sk_POLICYQUALINFO_value(st, i) SKM_sk_value(POLICYQUALINFO, (st), (i))
#define sk_POLICYQUALINFO_set(st, i, val) SKM_sk_set(POLICYQUALINFO, (st), (i),
#define sk_POLICYQUALINFO_zero(st) SKM_sk_zero(POLICYQUALINFO, (st))
#define sk_POLICYQUALINFO_push(st, val) SKM_sk_push(POLICYQUALINFO, (st), (val))
#define sk_POLICYQUALINFO_unshift(st, val) SKM_sk_unshift(POLICYQUALINFO, (st),
#define sk_POLICYQUALINFO_find(st, val) SKM_sk_find(POLICYQUALINFO, (st), (val))
#define sk_POLICYQUALINFO_find_ex(st, val) SKM_sk_find_ex(POLICYQUALINFO, (st),
#define sk_POLICYQUALINFO_delete(st, i) SKM_sk_delete(POLICYQUALINFO, (st), (i))
#define sk_POLICYQUALINFO_delete_ptr(st, ptr) SKM_sk_delete_ptr(POLICYQUALINFO,
#define sk_POLICYQUALINFO_insert(st, val, i) SKM_sk_insert(POLICYQUALINFO, (st),
#define sk_POLICYQUALINFO_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(POLICYQUALIN
#define sk_POLICYQUALINFO_dup(st) SKM_sk_dup(POLICYQUALINFO, st)
#define sk_POLICYQUALINFO_pop_free(st, free_func) SKM_sk_pop_free(POLICYQUALINFO
#define sk_POLICYQUALINFO_shift(st) SKM_sk_shift(POLICYQUALINFO, (st))
#define sk_POLICYQUALINFO_pop(st) SKM_sk_pop(POLICYQUALINFO, (st))
#define sk_POLICYQUALINFO_sort(st) SKM_sk_sort(POLICYQUALINFO, (st))
#define sk_POLICYQUALINFO_is_sorted(st) SKM_sk_is_sorted(POLICYQUALINFO, (st))

#define sk_POLICY_MAPPING_new(cmp) SKM_sk_new(POLICY_MAPPING, (cmp))
#define sk_POLICY_MAPPING_new_null() SKM_sk_new_null(POLICY_MAPPING)
#define sk_POLICY_MAPPING_free(st) SKM_sk_free(POLICY_MAPPING, (st))
#define sk_POLICY_MAPPING_num(st) SKM_sk_num(POLICY_MAPPING, (st))
#define sk_POLICY_MAPPING_value(st, i) SKM_sk_value(POLICY_MAPPING, (st), (i))
#define sk_POLICY_MAPPING_set(st, i, val) SKM_sk_set(POLICY_MAPPING, (st), (i),
#define sk_POLICY_MAPPING_zero(st) SKM_sk_zero(POLICY_MAPPING, (st))
#define sk_POLICY_MAPPING_push(st, val) SKM_sk_push(POLICY_MAPPING, (st), (val))
```

```
#define sk_POLICY_MAPPING_unshift(st, val) SKM_sk_unshift(POLICY_MAPPING, (st),
#define sk_POLICY_MAPPING_find(st, val) SKM_sk_find(POLICY_MAPPING, (st), (val))
#define sk_POLICY_MAPPING_find_ex(st, val) SKM_sk_find_ex(POLICY_MAPPING, (st),
#define sk_POLICY_MAPPING_delete(st, i) SKM_sk_delete(POLICY_MAPPING, (st), (i))
#define sk_POLICY_MAPPING_delete_ptr(st, ptr) SKM_sk_delete_ptr(POLICY_MAPPING,
#define sk_POLICY_MAPPING_insert(st, val, i) SKM_sk_insert(POLICY_MAPPING, (st),
#define sk_POLICY_MAPPING_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(POLICY_MAPPI
#define sk_POLICY_MAPPING_dup(st) SKM_sk_dup(POLICY_MAPPING, st)
#define sk_POLICY_MAPPING_pop_free(st, free_func) SKM_sk_pop_free(POLICY_MAPPING
#define sk_POLICY_MAPPING_shift(st) SKM_sk_shift(POLICY_MAPPING, (st))
#define sk_POLICY_MAPPING_pop(st) SKM_sk_pop(POLICY_MAPPING, (st))
#define sk_POLICY_MAPPING_sort(st) SKM_sk_sort(POLICY_MAPPING, (st))
#define sk_POLICY_MAPPING_is_sorted(st) SKM_sk_is_sorted(POLICY_MAPPING, (st))

#define sk_SRP_gN_new(cmp) SKM_sk_new(SRP_gN, (cmp))
#define sk_SRP_gN_new_null() SKM_sk_new_null(SRP_gN)
#define sk_SRP_gN_free(st) SKM_sk_free(SRP_gN, (st))
#define sk_SRP_gN_num(st) SKM_sk_num(SRP_gN, (st))
#define sk_SRP_gN_value(st, i) SKM_sk_value(SRP_gN, (st), (i))
#define sk_SRP_gN_set(st, i, val) SKM_sk_set(SRP_gN, (st), (i), (val))
#define sk_SRP_gN_zero(st) SKM_sk_zero(SRP_gN, (st))
#define sk_SRP_gN_push(st, val) SKM_sk_push(SRP_gN, (st), (val))
#define sk_SRP_gN_unshift(st, val) SKM_sk_unshift(SRP_gN, (st), (val))
#define sk_SRP_gN_find(st, val) SKM_sk_find(SRP_gN, (st), (val))
#define sk_SRP_gN_find_ex(st, val) SKM_sk_find_ex(SRP_gN, (st), (val))
#define sk_SRP_gN_delete(st, i) SKM_sk_delete(SRP_gN, (st), (i))
#define sk_SRP_gN_delete_ptr(st, ptr) SKM_sk_delete_ptr(SRP_gN, (st), (ptr))
#define sk_SRP_gN_insert(st, val, i) SKM_sk_insert(SRP_gN, (st), (val), (i))
#define sk_SRP_gN_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(SRP_gN, (st), (cmp))
#define sk_SRP_gN_dup(st) SKM_sk_dup(SRP_gN, st)
#define sk_SRP_gN_pop_free(st, free_func) SKM_sk_pop_free(SRP_gN, (st), (free_fu
#define sk_SRP_gN_shift(st) SKM_sk_shift(SRP_gN, (st))
#define sk_SRP_gN_pop(st) SKM_sk_pop(SRP_gN, (st))
#define sk_SRP_gN_sort(st) SKM_sk_sort(SRP_gN, (st))
#define sk_SRP_gN_is_sorted(st) SKM_sk_is_sorted(SRP_gN, (st))

#define sk_SRP_gN_cache_new(cmp) SKM_sk_new(SRP_gN_cache, (cmp))
#define sk_SRP_gN_cache_new_null() SKM_sk_new_null(SRP_gN_cache)
#define sk_SRP_gN_cache_free(st) SKM_sk_free(SRP_gN_cache, (st))
#define sk_SRP_gN_cache_num(st) SKM_sk_num(SRP_gN_cache, (st))
#define sk_SRP_gN_cache_value(st, i) SKM_sk_value(SRP_gN_cache, (st), (i))
#define sk_SRP_gN_cache_set(st, i, val) SKM_sk_set(SRP_gN_cache, (st), (i), (val
#define sk_SRP_gN_cache_zero(st) SKM_sk_zero(SRP_gN_cache, (st))
#define sk_SRP_gN_cache_push(st, val) SKM_sk_push(SRP_gN_cache, (st), (val))
#define sk_SRP_gN_cache_unshift(st, val) SKM_sk_unshift(SRP_gN_cache, (st), (val
#define sk_SRP_gN_cache_find(st, val) SKM_sk_find(SRP_gN_cache, (st), (val))
#define sk_SRP_gN_cache_find_ex(st, val) SKM_sk_find_ex(SRP_gN_cache, (st), (val
#define sk_SRP_gN_cache_delete(st, i) SKM_sk_delete(SRP_gN_cache, (st), (i))
#define sk_SRP_gN_cache_delete_ptr(st, ptr) SKM_sk_delete_ptr(SRP_gN_cache, (st)
#define sk_SRP_gN_cache_insert(st, val, i) SKM_sk_insert(SRP_gN_cache, (st), (va
#define sk_SRP_gN_cache_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(SRP_gN_cache,
#define sk_SRP_gN_cache_dup(st) SKM_sk_dup(SRP_gN_cache, st)
#define sk_SRP_gN_cache_pop_free(st, free_func) SKM_sk_pop_free(SRP_gN_cache, (s
#define sk_SRP_gN_cache_shift(st) SKM_sk_shift(SRP_gN_cache, (st))
#define sk_SRP_gN_cache_pop(st) SKM_sk_pop(SRP_gN_cache, (st))
#define sk_SRP_gN_cache_sort(st) SKM_sk_sort(SRP_gN_cache, (st))
#define sk_SRP_gN_cache_is_sorted(st) SKM_sk_is_sorted(SRP_gN_cache, (st))

#define sk_SRP_user_pwd_new(cmp) SKM_sk_new(SRP_user_pwd, (cmp))
#define sk_SRP_user_pwd_new_null() SKM_sk_new_null(SRP_user_pwd)
#define sk_SRP_user_pwd_free(st) SKM_sk_free(SRP_user_pwd, (st))
#define sk_SRP_user_pwd_num(st) SKM_sk_num(SRP_user_pwd, (st))
#define sk_SRP_user_pwd_value(st, i) SKM_sk_value(SRP_user_pwd, (st), (i))
#define sk_SRP_user_pwd_set(st, i, val) SKM_sk_set(SRP_user_pwd, (st), (i), (val
#define sk_SRP_user_pwd_zero(st) SKM_sk_zero(SRP_user_pwd, (st))
#define sk_SRP_user_pwd_push(st, val) SKM_sk_push(SRP_user_pwd, (st), (val))
```



```
#define sk_SRP_user_pwd_unshift(st, val) SKM_sk_unshift(SRP_user_pwd, (st), (val
#define sk_SRP_user_pwd_find(st, val) SKM_sk_find(SRP_user_pwd, (st), (val))
#define sk_SRP_user_pwd_find_ex(st, val) SKM_sk_find_ex(SRP_user_pwd, (st), (val
#define sk_SRP_user_pwd_delete(st, i) SKM_sk_delete(SRP_user_pwd, (st), (i))
#define sk_SRP_user_pwd_delete_ptr(st, ptr) SKM_sk_delete_ptr(SRP_user_pwd, (st)
#define sk_SRP_user_pwd_insert(st, val, i) SKM_sk_insert(SRP_user_pwd, (st), (va
#define sk_SRP_user_pwd_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(SRP_user_pwd,
#define sk_SRP_user_pwd_dup(st) SKM_sk_dup(SRP_user_pwd, st)
#define sk_SRP_user_pwd_pop_free(st, free_func) SKM_sk_pop_free(SRP_user_pwd, (s
#define sk_SRP_user_pwd_shift(st) SKM_sk_shift(SRP_user_pwd, (st))
#define sk_SRP_user_pwd_pop(st) SKM_sk_pop(SRP_user_pwd, (st))
#define sk_SRP_user_pwd_sort(st) SKM_sk_sort(SRP_user_pwd, (st))
#define sk_SRP_user_pwd_is_sorted(st) SKM_sk_is_sorted(SRP_user_pwd, (st))

#define sk_SRTP_PROTECTION_PROFILE_new(cmp) SKM_sk_new(SRTP_PROTECTION_PROFILE,
#define sk_SRTP_PROTECTION_PROFILE_new_null() SKM_sk_new_null(SRTP_PROTECTION_PR
#define sk_SRTP_PROTECTION_PROFILE_free(st) SKM_sk_free(SRTP_PROTECTION_PROFILE,
#define sk_SRTP_PROTECTION_PROFILE_num(st) SKM_sk_num(SRTP_PROTECTION_PROFILE, (
#define sk_SRTP_PROTECTION_PROFILE_value(st, i) SKM_sk_value(SRTP_PROTECTION_PRO
#define sk_SRTP_PROTECTION_PROFILE_set(st, i, val) SKM_sk_set(SRTP_PROTECTION_PR
#define sk_SRTP_PROTECTION_PROFILE_zero(st) SKM_sk_zero(SRTP_PROTECTION_PROFILE,
#define sk_SRTP_PROTECTION_PROFILE_push(st, val) SKM_sk_push(SRTP_PROTECTION_PRO
#define sk_SRTP_PROTECTION_PROFILE_unshift(st, val) SKM_sk_unshift(SRTP_PROTECTI
#define sk_SRTP_PROTECTION_PROFILE_find(st, val) SKM_sk_find(SRTP_PROTECTION_PRO
#define sk_SRTP_PROTECTION_PROFILE_find_ex(st, val) SKM_sk_find_ex(SRTP_PROTECTI
#define sk_SRTP_PROTECTION_PROFILE_delete(st, i) SKM_sk_delete(SRTP_PROTECTION_P
#define sk_SRTP_PROTECTION_PROFILE_delete_ptr(st, ptr) SKM_sk_delete_ptr(SRTP_PR
#define sk_SRTP_PROTECTION_PROFILE_insert(st, val, i) SKM_sk_insert(SRTP_PROTECT
#define sk_SRTP_PROTECTION_PROFILE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(SRT
#define sk_SRTP_PROTECTION_PROFILE_dup(st) SKM_sk_dup(SRTP_PROTECTION_PROFILE, s
#define sk_SRTP_PROTECTION_PROFILE_pop_free(st, free_func) SKM_sk_pop_free(SRTP_
#define sk_SRTP_PROTECTION_PROFILE_shift(st) SKM_sk_shift(SRTP_PROTECTION_PROFIL
#define sk_SRTP_PROTECTION_PROFILE_pop(st) SKM_sk_pop(SRTP_PROTECTION_PROFILE, (
#define sk_SRTP_PROTECTION_PROFILE_sort(st) SKM_sk_sort(SRTP_PROTECTION_PROFILE,
#define sk_SRTP_PROTECTION_PROFILE_is_sorted(st) SKM_sk_is_sorted(SRTP_PROTECTIO

#define sk_SSL_CIPHER_new(cmp) SKM_sk_new(SSL_CIPHER, (cmp))
#define sk_SSL_CIPHER_new_null() SKM_sk_new_null(SSL_CIPHER)
#define sk_SSL_CIPHER_free(st) SKM_sk_free(SSL_CIPHER, (st))
#define sk_SSL_CIPHER_num(st) SKM_sk_num(SSL_CIPHER, (st))
#define sk_SSL_CIPHER_value(st, i) SKM_sk_value(SSL_CIPHER, (st), (i))
#define sk_SSL_CIPHER_set(st, i, val) SKM_sk_set(SSL_CIPHER, (st), (i), (val))
#define sk_SSL_CIPHER_zero(st) SKM_sk_zero(SSL_CIPHER, (st))
#define sk_SSL_CIPHER_push(st, val) SKM_sk_push(SSL_CIPHER, (st), (val))
#define sk_SSL_CIPHER_unshift(st, val) SKM_sk_unshift(SSL_CIPHER, (st), (val))
#define sk_SSL_CIPHER_find(st, val) SKM_sk_find(SSL_CIPHER, (st), (val))
#define sk_SSL_CIPHER_find_ex(st, val) SKM_sk_find_ex(SSL_CIPHER, (st), (val))
#define sk_SSL_CIPHER_delete(st, i) SKM_sk_delete(SSL_CIPHER, (st), (i))
#define sk_SSL_CIPHER_delete_ptr(st, ptr) SKM_sk_delete_ptr(SSL_CIPHER, (st), (p
#define sk_SSL_CIPHER_insert(st, val, i) SKM_sk_insert(SSL_CIPHER, (st), (val),
#define sk_SSL_CIPHER_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(SSL_CIPHER, (st)
#define sk_SSL_CIPHER_dup(st) SKM_sk_dup(SSL_CIPHER, st)
#define sk_SSL_CIPHER_pop_free(st, free_func) SKM_sk_pop_free(SSL_CIPHER, (st),
#define sk_SSL_CIPHER_shift(st) SKM_sk_shift(SSL_CIPHER, (st))
#define sk_SSL_CIPHER_pop(st) SKM_sk_pop(SSL_CIPHER, (st))
#define sk_SSL_CIPHER_sort(st) SKM_sk_sort(SSL_CIPHER, (st))
#define sk_SSL_CIPHER_is_sorted(st) SKM_sk_is_sorted(SSL_CIPHER, (st))

#define sk_SSL_COMP_new(cmp) SKM_sk_new(SSL_COMP, (cmp))
#define sk_SSL_COMP_new_null() SKM_sk_new_null(SSL_COMP)
#define sk_SSL_COMP_free(st) SKM_sk_free(SSL_COMP, (st))
#define sk_SSL_COMP_num(st) SKM_sk_num(SSL_COMP, (st))
#define sk_SSL_COMP_value(st, i) SKM_sk_value(SSL_COMP, (st), (i))
#define sk_SSL_COMP_set(st, i, val) SKM_sk_set(SSL_COMP, (st), (i), (val))
#define sk_SSL_COMP_zero(st) SKM_sk_zero(SSL_COMP, (st))
#define sk_SSL_COMP_push(st, val) SKM_sk_push(SSL_COMP, (st), (val))
```

```
#define sk_SSL_COMP_unshift(st, val) SKM_sk_unshift(SSL_COMP, (st), (val))
#define sk_SSL_COMP_find(st, val) SKM_sk_find(SSL_COMP, (st), (val))
#define sk_SSL_COMP_find_ex(st, val) SKM_sk_find_ex(SSL_COMP, (st), (val))
#define sk_SSL_COMP_delete(st, i) SKM_sk_delete(SSL_COMP, (st), (i))
#define sk_SSL_COMP_delete_ptr(st, ptr) SKM_sk_delete_ptr(SSL_COMP, (st), (ptr))
#define sk_SSL_COMP_insert(st, val, i) SKM_sk_insert(SSL_COMP, (st), (val), (i))
#define sk_SSL_COMP_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(SSL_COMP, (st), (c
#define sk_SSL_COMP_dup(st) SKM_sk_dup(SSL_COMP, st)
#define sk_SSL_COMP_pop_free(st, free_func) SKM_sk_pop_free(SSL_COMP, (st), (fre
#define sk_SSL_COMP_shift(st) SKM_sk_shift(SSL_COMP, (st))
#define sk_SSL_COMP_pop(st) SKM_sk_pop(SSL_COMP, (st))
#define sk_SSL_COMP_sort(st) SKM_sk_sort(SSL_COMP, (st))
#define sk_SSL_COMP_is_sorted(st) SKM_sk_is_sorted(SSL_COMP, (st))

#define sk_STACK_OF_X509_NAME_ENTRY_new(cmp) SKM_sk_new(STACK_OF_X509_NAME_ENTRY
#define sk_STACK_OF_X509_NAME_ENTRY_new_null() SKM_sk_new_null(STACK_OF_X509_NAM
#define sk_STACK_OF_X509_NAME_ENTRY_free(st) SKM_sk_free(STACK_OF_X509_NAME_ENTR
#define sk_STACK_OF_X509_NAME_ENTRY_num(st) SKM_sk_num(STACK_OF_X509_NAME_ENTRY,
#define sk_STACK_OF_X509_NAME_ENTRY_value(st, i) SKM_sk_value(STACK_OF_X509_NAME
#define sk_STACK_OF_X509_NAME_ENTRY_set(st, i, val) SKM_sk_set(STACK_OF_X509_NAM
#define sk_STACK_OF_X509_NAME_ENTRY_zero(st) SKM_sk_zero(STACK_OF_X509_NAME_ENTR
#define sk_STACK_OF_X509_NAME_ENTRY_push(st, val) SKM_sk_push(STACK_OF_X509_NAME
#define sk_STACK_OF_X509_NAME_ENTRY_unshift(st, val) SKM_sk_unshift(STACK_OF_X50
#define sk_STACK_OF_X509_NAME_ENTRY_find(st, val) SKM_sk_find(STACK_OF_X509_NAME
#define sk_STACK_OF_X509_NAME_ENTRY_find_ex(st, val) SKM_sk_find_ex(STACK_OF_X50
#define sk_STACK_OF_X509_NAME_ENTRY_delete(st, i) SKM_sk_delete(STACK_OF_X509_NA
#define sk_STACK_OF_X509_NAME_ENTRY_delete_ptr(st, ptr) SKM_sk_delete_ptr(STACK_
#define sk_STACK_OF_X509_NAME_ENTRY_insert(st, val, i) SKM_sk_insert(STACK_OF_X5
#define sk_STACK_OF_X509_NAME_ENTRY_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(ST
#define sk_STACK_OF_X509_NAME_ENTRY_dup(st) SKM_sk_dup(STACK_OF_X509_NAME_ENTRY,
#define sk_STACK_OF_X509_NAME_ENTRY_pop_free(st, free_func) SKM_sk_pop_free(STAC
#define sk_STACK_OF_X509_NAME_ENTRY_shift(st) SKM_sk_shift(STACK_OF_X509_NAME_EN
#define sk_STACK_OF_X509_NAME_ENTRY_pop(st) SKM_sk_pop(STACK_OF_X509_NAME_ENTRY,
#define sk_STACK_OF_X509_NAME_ENTRY_sort(st) SKM_sk_sort(STACK_OF_X509_NAME_ENTR
#define sk_STACK_OF_X509_NAME_ENTRY_is_sorted(st) SKM_sk_is_sorted(STACK_OF_X509

#define sk_STORE_ATTR_INFO_new(cmp) SKM_sk_new(STORE_ATTR_INFO, (cmp))
#define sk_STORE_ATTR_INFO_new_null() SKM_sk_new_null(STORE_ATTR_INFO)
#define sk_STORE_ATTR_INFO_free(st) SKM_sk_free(STORE_ATTR_INFO, (st))
#define sk_STORE_ATTR_INFO_num(st) SKM_sk_num(STORE_ATTR_INFO, (st))
#define sk_STORE_ATTR_INFO_value(st, i) SKM_sk_value(STORE_ATTR_INFO, (st), (i))
#define sk_STORE_ATTR_INFO_set(st, i, val) SKM_sk_set(STORE_ATTR_INFO, (st), (i)
#define sk_STORE_ATTR_INFO_zero(st) SKM_sk_zero(STORE_ATTR_INFO, (st))
#define sk_STORE_ATTR_INFO_push(st, val) SKM_sk_push(STORE_ATTR_INFO, (st), (val
#define sk_STORE_ATTR_INFO_unshift(st, val) SKM_sk_unshift(STORE_ATTR_INFO, (st)
#define sk_STORE_ATTR_INFO_find(st, val) SKM_sk_find(STORE_ATTR_INFO, (st), (val
#define sk_STORE_ATTR_INFO_find_ex(st, val) SKM_sk_find_ex(STORE_ATTR_INFO, (st)
#define sk_STORE_ATTR_INFO_delete(st, i) SKM_sk_delete(STORE_ATTR_INFO, (st), (i
#define sk_STORE_ATTR_INFO_delete_ptr(st, ptr) SKM_sk_delete_ptr(STORE_ATTR_INFO
#define sk_STORE_ATTR_INFO_insert(st, val, i) SKM_sk_insert(STORE_ATTR_INFO, (st
#define sk_STORE_ATTR_INFO_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(STORE_ATTR_
#define sk_STORE_ATTR_INFO_dup(st) SKM_sk_dup(STORE_ATTR_INFO, st)
#define sk_STORE_ATTR_INFO_pop_free(st, free_func) SKM_sk_pop_free(STORE_ATTR_IN
#define sk_STORE_ATTR_INFO_shift(st) SKM_sk_shift(STORE_ATTR_INFO, (st))
#define sk_STORE_ATTR_INFO_pop(st) SKM_sk_pop(STORE_ATTR_INFO, (st))
#define sk_STORE_ATTR_INFO_sort(st) SKM_sk_sort(STORE_ATTR_INFO, (st))
#define sk_STORE_ATTR_INFO_is_sorted(st) SKM_sk_is_sorted(STORE_ATTR_INFO, (st))

#define sk_STORE_OBJECT_new(cmp) SKM_sk_new(STORE_OBJECT, (cmp))
#define sk_STORE_OBJECT_new_null() SKM_sk_new_null(STORE_OBJECT)
#define sk_STORE_OBJECT_free(st) SKM_sk_free(STORE_OBJECT, (st))
#define sk_STORE_OBJECT_num(st) SKM_sk_num(STORE_OBJECT, (st))
#define sk_STORE_OBJECT_value(st, i) SKM_sk_value(STORE_OBJECT, (st), (i))
#define sk_STORE_OBJECT_set(st, i, val) SKM_sk_set(STORE_OBJECT, (st), (i), (val
#define sk_STORE_OBJECT_zero(st) SKM_sk_zero(STORE_OBJECT, (st))
#define sk_STORE_OBJECT_push(st, val) SKM_sk_push(STORE_OBJECT, (st), (val))
```



```
#define sk_STORE_OBJECT_unshift(st, val) SKM_sk_unshift(STORE_OBJECT, (st), (val
#define sk_STORE_OBJECT_find(st, val) SKM_sk_find(STORE_OBJECT, (st), (val))
#define sk_STORE_OBJECT_find_ex(st, val) SKM_sk_find_ex(STORE_OBJECT, (st), (val
#define sk_STORE_OBJECT_delete(st, i) SKM_sk_delete(STORE_OBJECT, (st), (i))
#define sk_STORE_OBJECT_delete_ptr(st, ptr) SKM_sk_delete_ptr(STORE_OBJECT, (st)
#define sk_STORE_OBJECT_insert(st, val, i) SKM_sk_insert(STORE_OBJECT, (st), (va
#define sk_STORE_OBJECT_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(STORE_OBJECT,
#define sk_STORE_OBJECT_dup(st) SKM_sk_dup(STORE_OBJECT, st)
#define sk_STORE_OBJECT_pop_free(st, free_func) SKM_sk_pop_free(STORE_OBJECT, (s
#define sk_STORE_OBJECT_shift(st) SKM_sk_shift(STORE_OBJECT, (st))
#define sk_STORE_OBJECT_pop(st) SKM_sk_pop(STORE_OBJECT, (st))
#define sk_STORE_OBJECT_sort(st) SKM_sk_sort(STORE_OBJECT, (st))
#define sk_STORE_OBJECT_is_sorted(st) SKM_sk_is_sorted(STORE_OBJECT, (st))

#define sk_SXNETID_new(cmp) SKM_sk_new(SXNETID, (cmp))
#define sk_SXNETID_new_null() SKM_sk_new_null(SXNETID)
#define sk_SXNETID_free(st) SKM_sk_free(SXNETID, (st))
#define sk_SXNETID_num(st) SKM_sk_num(SXNETID, (st))
#define sk_SXNETID_value(st, i) SKM_sk_value(SXNETID, (st), (i))
#define sk_SXNETID_set(st, i, val) SKM_sk_set(SXNETID, (st), (i), (val))
#define sk_SXNETID_zero(st) SKM_sk_zero(SXNETID, (st))
#define sk_SXNETID_push(st, val) SKM_sk_push(SXNETID, (st), (val))
#define sk_SXNETID_unshift(st, val) SKM_sk_unshift(SXNETID, (st), (val))
#define sk_SXNETID_find(st, val) SKM_sk_find(SXNETID, (st), (val))
#define sk_SXNETID_find_ex(st, val) SKM_sk_find_ex(SXNETID, (st), (val))
#define sk_SXNETID_delete(st, i) SKM_sk_delete(SXNETID, (st), (i))
#define sk_SXNETID_delete_ptr(st, ptr) SKM_sk_delete_ptr(SXNETID, (st), (ptr))
#define sk_SXNETID_insert(st, val, i) SKM_sk_insert(SXNETID, (st), (val), (i))
#define sk_SXNETID_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(SXNETID, (st), (cmp
#define sk_SXNETID_dup(st) SKM_sk_dup(SXNETID, st)
#define sk_SXNETID_pop_free(st, free_func) SKM_sk_pop_free(SXNETID, (st), (free_
#define sk_SXNETID_shift(st) SKM_sk_shift(SXNETID, (st))
#define sk_SXNETID_pop(st) SKM_sk_pop(SXNETID, (st))
#define sk_SXNETID_sort(st) SKM_sk_sort(SXNETID, (st))
#define sk_SXNETID_is_sorted(st) SKM_sk_is_sorted(SXNETID, (st))

#define sk_UI_STRING_new(cmp) SKM_sk_new(UI_STRING, (cmp))
#define sk_UI_STRING_new_null() SKM_sk_new_null(UI_STRING)
#define sk_UI_STRING_free(st) SKM_sk_free(UI_STRING, (st))
#define sk_UI_STRING_num(st) SKM_sk_num(UI_STRING, (st))
#define sk_UI_STRING_value(st, i) SKM_sk_value(UI_STRING, (st), (i))
#define sk_UI_STRING_set(st, i, val) SKM_sk_set(UI_STRING, (st), (i), (val))
#define sk_UI_STRING_zero(st) SKM_sk_zero(UI_STRING, (st))
#define sk_UI_STRING_push(st, val) SKM_sk_push(UI_STRING, (st), (val))
#define sk_UI_STRING_unshift(st, val) SKM_sk_unshift(UI_STRING, (st), (val))
#define sk_UI_STRING_find(st, val) SKM_sk_find(UI_STRING, (st), (val))
#define sk_UI_STRING_find_ex(st, val) SKM_sk_find_ex(UI_STRING, (st), (val))
#define sk_UI_STRING_delete(st, i) SKM_sk_delete(UI_STRING, (st), (i))
#define sk_UI_STRING_delete_ptr(st, ptr) SKM_sk_delete_ptr(UI_STRING, (st), (ptr
#define sk_UI_STRING_insert(st, val, i) SKM_sk_insert(UI_STRING, (st), (val), (i
#define sk_UI_STRING_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(UI_STRING, (st),
#define sk_UI_STRING_dup(st) SKM_sk_dup(UI_STRING, st)
#define sk_UI_STRING_pop_free(st, free_func) SKM_sk_pop_free(UI_STRING, (st), (f
#define sk_UI_STRING_shift(st) SKM_sk_shift(UI_STRING, (st))
#define sk_UI_STRING_pop(st) SKM_sk_pop(UI_STRING, (st))
#define sk_UI_STRING_sort(st) SKM_sk_sort(UI_STRING, (st))
#define sk_UI_STRING_is_sorted(st) SKM_sk_is_sorted(UI_STRING, (st))

#define sk_X509_new(cmp) SKM_sk_new(X509, (cmp))
#define sk_X509_new_null() SKM_sk_new_null(X509)
#define sk_X509_free(st) SKM_sk_free(X509, (st))
#define sk_X509_num(st) SKM_sk_num(X509, (st))
#define sk_X509_value(st, i) SKM_sk_value(X509, (st), (i))
#define sk_X509_set(st, i, val) SKM_sk_set(X509, (st), (i), (val))
#define sk_X509_zero(st) SKM_sk_zero(X509, (st))
#define sk_X509_push(st, val) SKM_sk_push(X509, (st), (val))
```

```
#define sk_X509_unshift(st, val) SKM_sk_unshift(X509, (st), (val))
#define sk_X509_find(st, val) SKM_sk_find(X509, (st), (val))
#define sk_X509_find_ex(st, val) SKM_sk_find_ex(X509, (st), (val))
#define sk_X509_delete(st, i) SKM_sk_delete(X509, (st), (i))
#define sk_X509_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509, (st), (ptr))
#define sk_X509_insert(st, val, i) SKM_sk_insert(X509, (st), (val), (i))
#define sk_X509_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509, (st), (cmp))
#define sk_X509_dup(st) SKM_sk_dup(X509, st)
#define sk_X509_pop_free(st, free_func) SKM_sk_pop_free(X509, (st), (free_func))
#define sk_X509_shift(st) SKM_sk_shift(X509, (st))
#define sk_X509_pop(st) SKM_sk_pop(X509, (st))
#define sk_X509_sort(st) SKM_sk_sort(X509, (st))
#define sk_X509_is_sorted(st) SKM_sk_is_sorted(X509, (st))

#define sk_X509V3_EXT_METHOD_new(cmp) SKM_sk_new(X509V3_EXT_METHOD, (cmp))
#define sk_X509V3_EXT_METHOD_new_null() SKM_sk_new_null(X509V3_EXT_METHOD)
#define sk_X509V3_EXT_METHOD_free(st) SKM_sk_free(X509V3_EXT_METHOD, (st))
#define sk_X509V3_EXT_METHOD_num(st) SKM_sk_num(X509V3_EXT_METHOD, (st))
#define sk_X509V3_EXT_METHOD_value(st, i) SKM_sk_value(X509V3_EXT_METHOD, (st),
#define sk_X509V3_EXT_METHOD_set(st, i, val) SKM_sk_set(X509V3_EXT_METHOD, (st),
#define sk_X509V3_EXT_METHOD_zero(st) SKM_sk_zero(X509V3_EXT_METHOD, (st))
#define sk_X509V3_EXT_METHOD_push(st, val) SKM_sk_push(X509V3_EXT_METHOD, (st),
#define sk_X509V3_EXT_METHOD_unshift(st, val) SKM_sk_unshift(X509V3_EXT_METHOD,
#define sk_X509V3_EXT_METHOD_find(st, val) SKM_sk_find(X509V3_EXT_METHOD, (st),
#define sk_X509V3_EXT_METHOD_find_ex(st, val) SKM_sk_find_ex(X509V3_EXT_METHOD,
#define sk_X509V3_EXT_METHOD_delete(st, i) SKM_sk_delete(X509V3_EXT_METHOD, (st)
#define sk_X509V3_EXT_METHOD_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509V3_EXT_ME
#define sk_X509V3_EXT_METHOD_insert(st, val, i) SKM_sk_insert(X509V3_EXT_METHOD,
#define sk_X509V3_EXT_METHOD_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509V3_EX
#define sk_X509V3_EXT_METHOD_dup(st) SKM_sk_dup(X509V3_EXT_METHOD, st)
#define sk_X509V3_EXT_METHOD_pop_free(st, free_func) SKM_sk_pop_free(X509V3_EXT_
#define sk_X509V3_EXT_METHOD_shift(st) SKM_sk_shift(X509V3_EXT_METHOD, (st))
#define sk_X509V3_EXT_METHOD_pop(st) SKM_sk_pop(X509V3_EXT_METHOD, (st))
#define sk_X509V3_EXT_METHOD_sort(st) SKM_sk_sort(X509V3_EXT_METHOD, (st))
#define sk_X509V3_EXT_METHOD_is_sorted(st) SKM_sk_is_sorted(X509V3_EXT_METHOD, (

#define sk_X509_ALGOR_new(cmp) SKM_sk_new(X509_ALGOR, (cmp))
#define sk_X509_ALGOR_new_null() SKM_sk_new_null(X509_ALGOR)
#define sk_X509_ALGOR_free(st) SKM_sk_free(X509_ALGOR, (st))
#define sk_X509_ALGOR_num(st) SKM_sk_num(X509_ALGOR, (st))
#define sk_X509_ALGOR_value(st, i) SKM_sk_value(X509_ALGOR, (st), (i))
#define sk_X509_ALGOR_set(st, i, val) SKM_sk_set(X509_ALGOR, (st), (i), (val))
#define sk_X509_ALGOR_zero(st) SKM_sk_zero(X509_ALGOR, (st))
#define sk_X509_ALGOR_push(st, val) SKM_sk_push(X509_ALGOR, (st), (val))
#define sk_X509_ALGOR_unshift(st, val) SKM_sk_unshift(X509_ALGOR, (st), (val))
#define sk_X509_ALGOR_find(st, val) SKM_sk_find(X509_ALGOR, (st), (val))
#define sk_X509_ALGOR_find_ex(st, val) SKM_sk_find_ex(X509_ALGOR, (st), (val))
#define sk_X509_ALGOR_delete(st, i) SKM_sk_delete(X509_ALGOR, (st), (i))
#define sk_X509_ALGOR_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_ALGOR, (st), (p
#define sk_X509_ALGOR_insert(st, val, i) SKM_sk_insert(X509_ALGOR, (st), (val),
#define sk_X509_ALGOR_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_ALGOR, (st)
#define sk_X509_ALGOR_dup(st) SKM_sk_dup(X509_ALGOR, st)
#define sk_X509_ALGOR_pop_free(st, free_func) SKM_sk_pop_free(X509_ALGOR, (st),
#define sk_X509_ALGOR_shift(st) SKM_sk_shift(X509_ALGOR, (st))
#define sk_X509_ALGOR_pop(st) SKM_sk_pop(X509_ALGOR, (st))
#define sk_X509_ALGOR_sort(st) SKM_sk_sort(X509_ALGOR, (st))
#define sk_X509_ALGOR_is_sorted(st) SKM_sk_is_sorted(X509_ALGOR, (st))

#define sk_X509_ATTRIBUTE_new(cmp) SKM_sk_new(X509_ATTRIBUTE, (cmp))
#define sk_X509_ATTRIBUTE_new_null() SKM_sk_new_null(X509_ATTRIBUTE)
#define sk_X509_ATTRIBUTE_free(st) SKM_sk_free(X509_ATTRIBUTE, (st))
#define sk_X509_ATTRIBUTE_num(st) SKM_sk_num(X509_ATTRIBUTE, (st))
#define sk_X509_ATTRIBUTE_value(st, i) SKM_sk_value(X509_ATTRIBUTE, (st), (i))
#define sk_X509_ATTRIBUTE_set(st, i, val) SKM_sk_set(X509_ATTRIBUTE, (st), (i),
#define sk_X509_ATTRIBUTE_zero(st) SKM_sk_zero(X509_ATTRIBUTE, (st))
#define sk_X509_ATTRIBUTE_push(st, val) SKM_sk_push(X509_ATTRIBUTE, (st), (val))
```



```
#define sk_X509_ATTRIBUTE_unshift(st, val) SKM_sk_unshift(X509_ATTRIBUTE, (st),
#define sk_X509_ATTRIBUTE_find(st, val) SKM_sk_find(X509_ATTRIBUTE, (st), (val))
#define sk_X509_ATTRIBUTE_find_ex(st, val) SKM_sk_find_ex(X509_ATTRIBUTE, (st),
#define sk_X509_ATTRIBUTE_delete(st, i) SKM_sk_delete(X509_ATTRIBUTE, (st), (i))
#define sk_X509_ATTRIBUTE_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_ATTRIBUTE,
#define sk_X509_ATTRIBUTE_insert(st, val, i) SKM_sk_insert(X509_ATTRIBUTE, (st),
#define sk_X509_ATTRIBUTE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_ATTRIBU
#define sk_X509_ATTRIBUTE_dup(st) SKM_sk_dup(X509_ATTRIBUTE, st)
#define sk_X509_ATTRIBUTE_pop_free(st, free_func) SKM_sk_pop_free(X509_ATTRIBUTE
#define sk_X509_ATTRIBUTE_shift(st) SKM_sk_shift(X509_ATTRIBUTE, (st))
#define sk_X509_ATTRIBUTE_pop(st) SKM_sk_pop(X509_ATTRIBUTE, (st))
#define sk_X509_ATTRIBUTE_sort(st) SKM_sk_sort(X509_ATTRIBUTE, (st))
#define sk_X509_ATTRIBUTE_is_sorted(st) SKM_sk_is_sorted(X509_ATTRIBUTE, (st))

#define sk_X509_CRL_new(cmp) SKM_sk_new(X509_CRL, (cmp))
#define sk_X509_CRL_new_null() SKM_sk_new_null(X509_CRL)
#define sk_X509_CRL_free(st) SKM_sk_free(X509_CRL, (st))
#define sk_X509_CRL_num(st) SKM_sk_num(X509_CRL, (st))
#define sk_X509_CRL_value(st, i) SKM_sk_value(X509_CRL, (st), (i))
#define sk_X509_CRL_set(st, i, val) SKM_sk_set(X509_CRL, (st), (i), (val))
#define sk_X509_CRL_zero(st) SKM_sk_zero(X509_CRL, (st))
#define sk_X509_CRL_push(st, val) SKM_sk_push(X509_CRL, (st), (val))
#define sk_X509_CRL_unshift(st, val) SKM_sk_unshift(X509_CRL, (st), (val))
#define sk_X509_CRL_find(st, val) SKM_sk_find(X509_CRL, (st), (val))
#define sk_X509_CRL_find_ex(st, val) SKM_sk_find_ex(X509_CRL, (st), (val))
#define sk_X509_CRL_delete(st, i) SKM_sk_delete(X509_CRL, (st), (i))
#define sk_X509_CRL_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_CRL, (st), (ptr))
#define sk_X509_CRL_insert(st, val, i) SKM_sk_insert(X509_CRL, (st), (val), (i))
#define sk_X509_CRL_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_CRL, (st), (c
#define sk_X509_CRL_dup(st) SKM_sk_dup(X509_CRL, st)
#define sk_X509_CRL_pop_free(st, free_func) SKM_sk_pop_free(X509_CRL, (st), (fre
#define sk_X509_CRL_shift(st) SKM_sk_shift(X509_CRL, (st))
#define sk_X509_CRL_pop(st) SKM_sk_pop(X509_CRL, (st))
#define sk_X509_CRL_sort(st) SKM_sk_sort(X509_CRL, (st))
#define sk_X509_CRL_is_sorted(st) SKM_sk_is_sorted(X509_CRL, (st))

#define sk_X509_EXTENSION_new(cmp) SKM_sk_new(X509_EXTENSION, (cmp))
#define sk_X509_EXTENSION_new_null() SKM_sk_new_null(X509_EXTENSION)
#define sk_X509_EXTENSION_free(st) SKM_sk_free(X509_EXTENSION, (st))
#define sk_X509_EXTENSION_num(st) SKM_sk_num(X509_EXTENSION, (st))
#define sk_X509_EXTENSION_value(st, i) SKM_sk_value(X509_EXTENSION, (st), (i))
#define sk_X509_EXTENSION_set(st, i, val) SKM_sk_set(X509_EXTENSION, (st), (i),
#define sk_X509_EXTENSION_zero(st) SKM_sk_zero(X509_EXTENSION, (st))
#define sk_X509_EXTENSION_push(st, val) SKM_sk_push(X509_EXTENSION, (st), (val))
#define sk_X509_EXTENSION_unshift(st, val) SKM_sk_unshift(X509_EXTENSION, (st),
#define sk_X509_EXTENSION_find(st, val) SKM_sk_find(X509_EXTENSION, (st), (val))
#define sk_X509_EXTENSION_find_ex(st, val) SKM_sk_find_ex(X509_EXTENSION, (st),
#define sk_X509_EXTENSION_delete(st, i) SKM_sk_delete(X509_EXTENSION, (st), (i))
#define sk_X509_EXTENSION_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_EXTENSION,
#define sk_X509_EXTENSION_insert(st, val, i) SKM_sk_insert(X509_EXTENSION, (st),
#define sk_X509_EXTENSION_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_EXTENSI
#define sk_X509_EXTENSION_dup(st) SKM_sk_dup(X509_EXTENSION, st)
#define sk_X509_EXTENSION_pop_free(st, free_func) SKM_sk_pop_free(X509_EXTENSION
#define sk_X509_EXTENSION_shift(st) SKM_sk_shift(X509_EXTENSION, (st))
#define sk_X509_EXTENSION_pop(st) SKM_sk_pop(X509_EXTENSION, (st))
#define sk_X509_EXTENSION_sort(st) SKM_sk_sort(X509_EXTENSION, (st))
#define sk_X509_EXTENSION_is_sorted(st) SKM_sk_is_sorted(X509_EXTENSION, (st))

#define sk_X509_INFO_new(cmp) SKM_sk_new(X509_INFO, (cmp))
#define sk_X509_INFO_new_null() SKM_sk_new_null(X509_INFO)
#define sk_X509_INFO_free(st) SKM_sk_free(X509_INFO, (st))
#define sk_X509_INFO_num(st) SKM_sk_num(X509_INFO, (st))
#define sk_X509_INFO_value(st, i) SKM_sk_value(X509_INFO, (st), (i))
#define sk_X509_INFO_set(st, i, val) SKM_sk_set(X509_INFO, (st), (i), (val))
#define sk_X509_INFO_zero(st) SKM_sk_zero(X509_INFO, (st))
#define sk_X509_INFO_push(st, val) SKM_sk_push(X509_INFO, (st), (val))
```

```
#define sk_X509_INFO_unshift(st, val) SKM_sk_unshift(X509_INFO, (st), (val))
#define sk_X509_INFO_find(st, val) SKM_sk_find(X509_INFO, (st), (val))
#define sk_X509_INFO_find_ex(st, val) SKM_sk_find_ex(X509_INFO, (st), (val))
#define sk_X509_INFO_delete(st, i) SKM_sk_delete(X509_INFO, (st), (i))
#define sk_X509_INFO_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_INFO, (st), (ptr
#define sk_X509_INFO_insert(st, val, i) SKM_sk_insert(X509_INFO, (st), (val), (i
#define sk_X509_INFO_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_INFO, (st),
#define sk_X509_INFO_dup(st) SKM_sk_dup(X509_INFO, st)
#define sk_X509_INFO_pop_free(st, free_func) SKM_sk_pop_free(X509_INFO, (st), (f
#define sk_X509_INFO_shift(st) SKM_sk_shift(X509_INFO, (st))
#define sk_X509_INFO_pop(st) SKM_sk_pop(X509_INFO, (st))
#define sk_X509_INFO_sort(st) SKM_sk_sort(X509_INFO, (st))
#define sk_X509_INFO_is_sorted(st) SKM_sk_is_sorted(X509_INFO, (st))

#define sk_X509_LOOKUP_new(cmp) SKM_sk_new(X509_LOOKUP, (cmp))
#define sk_X509_LOOKUP_new_null() SKM_sk_new_null(X509_LOOKUP)
#define sk_X509_LOOKUP_free(st) SKM_sk_free(X509_LOOKUP, (st))
#define sk_X509_LOOKUP_num(st) SKM_sk_num(X509_LOOKUP, (st))
#define sk_X509_LOOKUP_value(st, i) SKM_sk_value(X509_LOOKUP, (st), (i))
#define sk_X509_LOOKUP_set(st, i, val) SKM_sk_set(X509_LOOKUP, (st), (i), (val))
#define sk_X509_LOOKUP_zero(st) SKM_sk_zero(X509_LOOKUP, (st))
#define sk_X509_LOOKUP_push(st, val) SKM_sk_push(X509_LOOKUP, (st), (val))
#define sk_X509_LOOKUP_unshift(st, val) SKM_sk_unshift(X509_LOOKUP, (st), (val))
#define sk_X509_LOOKUP_find(st, val) SKM_sk_find(X509_LOOKUP, (st), (val))
#define sk_X509_LOOKUP_find_ex(st, val) SKM_sk_find_ex(X509_LOOKUP, (st), (val))
#define sk_X509_LOOKUP_delete(st, i) SKM_sk_delete(X509_LOOKUP, (st), (i))
#define sk_X509_LOOKUP_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_LOOKUP, (st),
#define sk_X509_LOOKUP_insert(st, val, i) SKM_sk_insert(X509_LOOKUP, (st), (val)
#define sk_X509_LOOKUP_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_LOOKUP, (s
#define sk_X509_LOOKUP_dup(st) SKM_sk_dup(X509_LOOKUP, st)
#define sk_X509_LOOKUP_pop_free(st, free_func) SKM_sk_pop_free(X509_LOOKUP, (st)
#define sk_X509_LOOKUP_shift(st) SKM_sk_shift(X509_LOOKUP, (st))
#define sk_X509_LOOKUP_pop(st) SKM_sk_pop(X509_LOOKUP, (st))
#define sk_X509_LOOKUP_sort(st) SKM_sk_sort(X509_LOOKUP, (st))
#define sk_X509_LOOKUP_is_sorted(st) SKM_sk_is_sorted(X509_LOOKUP, (st))

#define sk_X509_NAME_new(cmp) SKM_sk_new(X509_NAME, (cmp))
#define sk_X509_NAME_new_null() SKM_sk_new_null(X509_NAME)
#define sk_X509_NAME_free(st) SKM_sk_free(X509_NAME, (st))
#define sk_X509_NAME_num(st) SKM_sk_num(X509_NAME, (st))
#define sk_X509_NAME_value(st, i) SKM_sk_value(X509_NAME, (st), (i))
#define sk_X509_NAME_set(st, i, val) SKM_sk_set(X509_NAME, (st), (i), (val))
#define sk_X509_NAME_zero(st) SKM_sk_zero(X509_NAME, (st))
#define sk_X509_NAME_push(st, val) SKM_sk_push(X509_NAME, (st), (val))
#define sk_X509_NAME_unshift(st, val) SKM_sk_unshift(X509_NAME, (st), (val))
#define sk_X509_NAME_find(st, val) SKM_sk_find(X509_NAME, (st), (val))
#define sk_X509_NAME_find_ex(st, val) SKM_sk_find_ex(X509_NAME, (st), (val))
#define sk_X509_NAME_delete(st, i) SKM_sk_delete(X509_NAME, (st), (i))
#define sk_X509_NAME_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_NAME, (st), (ptr
#define sk_X509_NAME_insert(st, val, i) SKM_sk_insert(X509_NAME, (st), (val), (i
#define sk_X509_NAME_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_NAME, (st),
#define sk_X509_NAME_dup(st) SKM_sk_dup(X509_NAME, st)
#define sk_X509_NAME_pop_free(st, free_func) SKM_sk_pop_free(X509_NAME, (st), (f
#define sk_X509_NAME_shift(st) SKM_sk_shift(X509_NAME, (st))
#define sk_X509_NAME_pop(st) SKM_sk_pop(X509_NAME, (st))
#define sk_X509_NAME_sort(st) SKM_sk_sort(X509_NAME, (st))
#define sk_X509_NAME_is_sorted(st) SKM_sk_is_sorted(X509_NAME, (st))

#define sk_X509_NAME_ENTRY_new(cmp) SKM_sk_new(X509_NAME_ENTRY, (cmp))
#define sk_X509_NAME_ENTRY_new_null() SKM_sk_new_null(X509_NAME_ENTRY)
#define sk_X509_NAME_ENTRY_free(st) SKM_sk_free(X509_NAME_ENTRY, (st))
#define sk_X509_NAME_ENTRY_num(st) SKM_sk_num(X509_NAME_ENTRY, (st))
#define sk_X509_NAME_ENTRY_value(st, i) SKM_sk_value(X509_NAME_ENTRY, (st), (i))
#define sk_X509_NAME_ENTRY_set(st, i, val) SKM_sk_set(X509_NAME_ENTRY, (st), (i)
#define sk_X509_NAME_ENTRY_zero(st) SKM_sk_zero(X509_NAME_ENTRY, (st))
#define sk_X509_NAME_ENTRY_push(st, val) SKM_sk_push(X509_NAME_ENTRY, (st), (val
```



```
#define sk_X509_NAME_ENTRY_unshift(st, val) SKM_sk_unshift(X509_NAME_ENTRY, (st)
#define sk_X509_NAME_ENTRY_find(st, val) SKM_sk_find(X509_NAME_ENTRY, (st), (val
#define sk_X509_NAME_ENTRY_find_ex(st, val) SKM_sk_find_ex(X509_NAME_ENTRY, (st)
#define sk_X509_NAME_ENTRY_delete(st, i) SKM_sk_delete(X509_NAME_ENTRY, (st), (i
#define sk_X509_NAME_ENTRY_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_NAME_ENTRY
#define sk_X509_NAME_ENTRY_insert(st, val, i) SKM_sk_insert(X509_NAME_ENTRY, (st
#define sk_X509_NAME_ENTRY_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_NAME_E
#define sk_X509_NAME_ENTRY_dup(st) SKM_sk_dup(X509_NAME_ENTRY, st)
#define sk_X509_NAME_ENTRY_pop_free(st, free_func) SKM_sk_pop_free(X509_NAME_ENT
#define sk_X509_NAME_ENTRY_shift(st) SKM_sk_shift(X509_NAME_ENTRY, (st))
#define sk_X509_NAME_ENTRY_pop(st) SKM_sk_pop(X509_NAME_ENTRY, (st))
#define sk_X509_NAME_ENTRY_sort(st) SKM_sk_sort(X509_NAME_ENTRY, (st))
#define sk_X509_NAME_ENTRY_is_sorted(st) SKM_sk_is_sorted(X509_NAME_ENTRY, (st))

#define sk_X509_OBJECT_new(cmp) SKM_sk_new(X509_OBJECT, (cmp))
#define sk_X509_OBJECT_new_null() SKM_sk_new_null(X509_OBJECT)
#define sk_X509_OBJECT_free(st) SKM_sk_free(X509_OBJECT, (st))
#define sk_X509_OBJECT_num(st) SKM_sk_num(X509_OBJECT, (st))
#define sk_X509_OBJECT_value(st, i) SKM_sk_value(X509_OBJECT, (st), (i))
#define sk_X509_OBJECT_set(st, i, val) SKM_sk_set(X509_OBJECT, (st), (i), (val))
#define sk_X509_OBJECT_zero(st) SKM_sk_zero(X509_OBJECT, (st))
#define sk_X509_OBJECT_push(st, val) SKM_sk_push(X509_OBJECT, (st), (val))
#define sk_X509_OBJECT_unshift(st, val) SKM_sk_unshift(X509_OBJECT, (st), (val))
#define sk_X509_OBJECT_find(st, val) SKM_sk_find(X509_OBJECT, (st), (val))
#define sk_X509_OBJECT_find_ex(st, val) SKM_sk_find_ex(X509_OBJECT, (st), (val))
#define sk_X509_OBJECT_delete(st, i) SKM_sk_delete(X509_OBJECT, (st), (i))
#define sk_X509_OBJECT_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_OBJECT, (st),
#define sk_X509_OBJECT_insert(st, val, i) SKM_sk_insert(X509_OBJECT, (st), (val)
#define sk_X509_OBJECT_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_OBJECT, (s
#define sk_X509_OBJECT_dup(st) SKM_sk_dup(X509_OBJECT, st)
#define sk_X509_OBJECT_pop_free(st, free_func) SKM_sk_pop_free(X509_OBJECT, (st)
#define sk_X509_OBJECT_shift(st) SKM_sk_shift(X509_OBJECT, (st))
#define sk_X509_OBJECT_pop(st) SKM_sk_pop(X509_OBJECT, (st))
#define sk_X509_OBJECT_sort(st) SKM_sk_sort(X509_OBJECT, (st))
#define sk_X509_OBJECT_is_sorted(st) SKM_sk_is_sorted(X509_OBJECT, (st))

#define sk_X509_POLICY_DATA_new(cmp) SKM_sk_new(X509_POLICY_DATA, (cmp))
#define sk_X509_POLICY_DATA_new_null() SKM_sk_new_null(X509_POLICY_DATA)
#define sk_X509_POLICY_DATA_free(st) SKM_sk_free(X509_POLICY_DATA, (st))
#define sk_X509_POLICY_DATA_num(st) SKM_sk_num(X509_POLICY_DATA, (st))
#define sk_X509_POLICY_DATA_value(st, i) SKM_sk_value(X509_POLICY_DATA, (st), (i
#define sk_X509_POLICY_DATA_set(st, i, val) SKM_sk_set(X509_POLICY_DATA, (st), (
#define sk_X509_POLICY_DATA_zero(st) SKM_sk_zero(X509_POLICY_DATA, (st))
#define sk_X509_POLICY_DATA_push(st, val) SKM_sk_push(X509_POLICY_DATA, (st), (v
#define sk_X509_POLICY_DATA_unshift(st, val) SKM_sk_unshift(X509_POLICY_DATA, (s
#define sk_X509_POLICY_DATA_find(st, val) SKM_sk_find(X509_POLICY_DATA, (st), (v
#define sk_X509_POLICY_DATA_find_ex(st, val) SKM_sk_find_ex(X509_POLICY_DATA, (s
#define sk_X509_POLICY_DATA_delete(st, i) SKM_sk_delete(X509_POLICY_DATA, (st),
#define sk_X509_POLICY_DATA_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_POLICY_DA
#define sk_X509_POLICY_DATA_insert(st, val, i) SKM_sk_insert(X509_POLICY_DATA, (
#define sk_X509_POLICY_DATA_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_POLIC
#define sk_X509_POLICY_DATA_dup(st) SKM_sk_dup(X509_POLICY_DATA, st)
#define sk_X509_POLICY_DATA_pop_free(st, free_func) SKM_sk_pop_free(X509_POLICY_
#define sk_X509_POLICY_DATA_shift(st) SKM_sk_shift(X509_POLICY_DATA, (st))
#define sk_X509_POLICY_DATA_pop(st) SKM_sk_pop(X509_POLICY_DATA, (st))
#define sk_X509_POLICY_DATA_sort(st) SKM_sk_sort(X509_POLICY_DATA, (st))
#define sk_X509_POLICY_DATA_is_sorted(st) SKM_sk_is_sorted(X509_POLICY_DATA, (st

#define sk_X509_POLICY_NODE_new(cmp) SKM_sk_new(X509_POLICY_NODE, (cmp))
#define sk_X509_POLICY_NODE_new_null() SKM_sk_new_null(X509_POLICY_NODE)
#define sk_X509_POLICY_NODE_free(st) SKM_sk_free(X509_POLICY_NODE, (st))
#define sk_X509_POLICY_NODE_num(st) SKM_sk_num(X509_POLICY_NODE, (st))
#define sk_X509_POLICY_NODE_value(st, i) SKM_sk_value(X509_POLICY_NODE, (st), (i
#define sk_X509_POLICY_NODE_set(st, i, val) SKM_sk_set(X509_POLICY_NODE, (st), (
#define sk_X509_POLICY_NODE_zero(st) SKM_sk_zero(X509_POLICY_NODE, (st))
#define sk_X509_POLICY_NODE_push(st, val) SKM_sk_push(X509_POLICY_NODE, (st), (v
```

```
#define sk_X509_POLICY_NODE_unshift(st, val) SKM_sk_unshift(X509_POLICY_NODE, (s
#define sk_X509_POLICY_NODE_find(st, val) SKM_sk_find(X509_POLICY_NODE, (st), (v
#define sk_X509_POLICY_NODE_find_ex(st, val) SKM_sk_find_ex(X509_POLICY_NODE, (s
#define sk_X509_POLICY_NODE_delete(st, i) SKM_sk_delete(X509_POLICY_NODE, (st),
#define sk_X509_POLICY_NODE_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_POLICY_NO
#define sk_X509_POLICY_NODE_insert(st, val, i) SKM_sk_insert(X509_POLICY_NODE, (
#define sk_X509_POLICY_NODE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_POLIC
#define sk_X509_POLICY_NODE_dup(st) SKM_sk_dup(X509_POLICY_NODE, st)
#define sk_X509_POLICY_NODE_pop_free(st, free_func) SKM_sk_pop_free(X509_POLICY_
#define sk_X509_POLICY_NODE_shift(st) SKM_sk_shift(X509_POLICY_NODE, (st))
#define sk_X509_POLICY_NODE_pop(st) SKM_sk_pop(X509_POLICY_NODE, (st))
#define sk_X509_POLICY_NODE_sort(st) SKM_sk_sort(X509_POLICY_NODE, (st))
#define sk_X509_POLICY_NODE_is_sorted(st) SKM_sk_is_sorted(X509_POLICY_NODE, (st

#define sk_X509_PURPOSE_new(cmp) SKM_sk_new(X509_PURPOSE, (cmp))
#define sk_X509_PURPOSE_new_null() SKM_sk_new_null(X509_PURPOSE)
#define sk_X509_PURPOSE_free(st) SKM_sk_free(X509_PURPOSE, (st))
#define sk_X509_PURPOSE_num(st) SKM_sk_num(X509_PURPOSE, (st))
#define sk_X509_PURPOSE_value(st, i) SKM_sk_value(X509_PURPOSE, (st), (i))
#define sk_X509_PURPOSE_set(st, i, val) SKM_sk_set(X509_PURPOSE, (st), (i), (val
#define sk_X509_PURPOSE_zero(st) SKM_sk_zero(X509_PURPOSE, (st))
#define sk_X509_PURPOSE_push(st, val) SKM_sk_push(X509_PURPOSE, (st), (val))
#define sk_X509_PURPOSE_unshift(st, val) SKM_sk_unshift(X509_PURPOSE, (st), (val
#define sk_X509_PURPOSE_find(st, val) SKM_sk_find(X509_PURPOSE, (st), (val))
#define sk_X509_PURPOSE_find_ex(st, val) SKM_sk_find_ex(X509_PURPOSE, (st), (val
#define sk_X509_PURPOSE_delete(st, i) SKM_sk_delete(X509_PURPOSE, (st), (i))
#define sk_X509_PURPOSE_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_PURPOSE, (st)
#define sk_X509_PURPOSE_insert(st, val, i) SKM_sk_insert(X509_PURPOSE, (st), (va
#define sk_X509_PURPOSE_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_PURPOSE,
#define sk_X509_PURPOSE_dup(st) SKM_sk_dup(X509_PURPOSE, st)
#define sk_X509_PURPOSE_pop_free(st, free_func) SKM_sk_pop_free(X509_PURPOSE, (s
#define sk_X509_PURPOSE_shift(st) SKM_sk_shift(X509_PURPOSE, (st))
#define sk_X509_PURPOSE_pop(st) SKM_sk_pop(X509_PURPOSE, (st))
#define sk_X509_PURPOSE_sort(st) SKM_sk_sort(X509_PURPOSE, (st))
#define sk_X509_PURPOSE_is_sorted(st) SKM_sk_is_sorted(X509_PURPOSE, (st))

#define sk_X509_REVOKED_new(cmp) SKM_sk_new(X509_REVOKED, (cmp))
#define sk_X509_REVOKED_new_null() SKM_sk_new_null(X509_REVOKED)
#define sk_X509_REVOKED_free(st) SKM_sk_free(X509_REVOKED, (st))
#define sk_X509_REVOKED_num(st) SKM_sk_num(X509_REVOKED, (st))
#define sk_X509_REVOKED_value(st, i) SKM_sk_value(X509_REVOKED, (st), (i))
#define sk_X509_REVOKED_set(st, i, val) SKM_sk_set(X509_REVOKED, (st), (i), (val
#define sk_X509_REVOKED_zero(st) SKM_sk_zero(X509_REVOKED, (st))
#define sk_X509_REVOKED_push(st, val) SKM_sk_push(X509_REVOKED, (st), (val))
#define sk_X509_REVOKED_unshift(st, val) SKM_sk_unshift(X509_REVOKED, (st), (val
#define sk_X509_REVOKED_find(st, val) SKM_sk_find(X509_REVOKED, (st), (val))
#define sk_X509_REVOKED_find_ex(st, val) SKM_sk_find_ex(X509_REVOKED, (st), (val
#define sk_X509_REVOKED_delete(st, i) SKM_sk_delete(X509_REVOKED, (st), (i))
#define sk_X509_REVOKED_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_REVOKED, (st)
#define sk_X509_REVOKED_insert(st, val, i) SKM_sk_insert(X509_REVOKED, (st), (va
#define sk_X509_REVOKED_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_REVOKED,
#define sk_X509_REVOKED_dup(st) SKM_sk_dup(X509_REVOKED, st)
#define sk_X509_REVOKED_pop_free(st, free_func) SKM_sk_pop_free(X509_REVOKED, (s
#define sk_X509_REVOKED_shift(st) SKM_sk_shift(X509_REVOKED, (st))
#define sk_X509_REVOKED_pop(st) SKM_sk_pop(X509_REVOKED, (st))
#define sk_X509_REVOKED_sort(st) SKM_sk_sort(X509_REVOKED, (st))
#define sk_X509_REVOKED_is_sorted(st) SKM_sk_is_sorted(X509_REVOKED, (st))

#define sk_X509_TRUST_new(cmp) SKM_sk_new(X509_TRUST, (cmp))
#define sk_X509_TRUST_new_null() SKM_sk_new_null(X509_TRUST)
#define sk_X509_TRUST_free(st) SKM_sk_free(X509_TRUST, (st))
#define sk_X509_TRUST_num(st) SKM_sk_num(X509_TRUST, (st))
#define sk_X509_TRUST_value(st, i) SKM_sk_value(X509_TRUST, (st), (i))
#define sk_X509_TRUST_set(st, i, val) SKM_sk_set(X509_TRUST, (st), (i), (val))
#define sk_X509_TRUST_zero(st) SKM_sk_zero(X509_TRUST, (st))
#define sk_X509_TRUST_push(st, val) SKM_sk_push(X509_TRUST, (st), (val))
```



```
#define sk_X509_TRUST_unshift(st, val) SKM_sk_unshift(X509_TRUST, (st), (val))
#define sk_X509_TRUST_find(st, val) SKM_sk_find(X509_TRUST, (st), (val))
#define sk_X509_TRUST_find_ex(st, val) SKM_sk_find_ex(X509_TRUST, (st), (val))
#define sk_X509_TRUST_delete(st, i) SKM_sk_delete(X509_TRUST, (st), (i))
#define sk_X509_TRUST_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_TRUST, (st), (p
#define sk_X509_TRUST_insert(st, val, i) SKM_sk_insert(X509_TRUST, (st), (val),
#define sk_X509_TRUST_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_TRUST, (st)
#define sk_X509_TRUST_dup(st) SKM_sk_dup(X509_TRUST, st)
#define sk_X509_TRUST_pop_free(st, free_func) SKM_sk_pop_free(X509_TRUST, (st),
#define sk_X509_TRUST_shift(st) SKM_sk_shift(X509_TRUST, (st))
#define sk_X509_TRUST_pop(st) SKM_sk_pop(X509_TRUST, (st))
#define sk_X509_TRUST_sort(st) SKM_sk_sort(X509_TRUST, (st))
#define sk_X509_TRUST_is_sorted(st) SKM_sk_is_sorted(X509_TRUST, (st))

#define sk_X509_VERIFY_PARAM_new(cmp) SKM_sk_new(X509_VERIFY_PARAM, (cmp))
#define sk_X509_VERIFY_PARAM_new_null() SKM_sk_new_null(X509_VERIFY_PARAM)
#define sk_X509_VERIFY_PARAM_free(st) SKM_sk_free(X509_VERIFY_PARAM, (st))
#define sk_X509_VERIFY_PARAM_num(st) SKM_sk_num(X509_VERIFY_PARAM, (st))
#define sk_X509_VERIFY_PARAM_value(st, i) SKM_sk_value(X509_VERIFY_PARAM, (st),
#define sk_X509_VERIFY_PARAM_set(st, i, val) SKM_sk_set(X509_VERIFY_PARAM, (st),
#define sk_X509_VERIFY_PARAM_zero(st) SKM_sk_zero(X509_VERIFY_PARAM, (st))
#define sk_X509_VERIFY_PARAM_push(st, val) SKM_sk_push(X509_VERIFY_PARAM, (st),
#define sk_X509_VERIFY_PARAM_unshift(st, val) SKM_sk_unshift(X509_VERIFY_PARAM,
#define sk_X509_VERIFY_PARAM_find(st, val) SKM_sk_find(X509_VERIFY_PARAM, (st),
#define sk_X509_VERIFY_PARAM_find_ex(st, val) SKM_sk_find_ex(X509_VERIFY_PARAM,
#define sk_X509_VERIFY_PARAM_delete(st, i) SKM_sk_delete(X509_VERIFY_PARAM, (st)
#define sk_X509_VERIFY_PARAM_delete_ptr(st, ptr) SKM_sk_delete_ptr(X509_VERIFY_P
#define sk_X509_VERIFY_PARAM_insert(st, val, i) SKM_sk_insert(X509_VERIFY_PARAM,
#define sk_X509_VERIFY_PARAM_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(X509_VERI
#define sk_X509_VERIFY_PARAM_dup(st) SKM_sk_dup(X509_VERIFY_PARAM, st)
#define sk_X509_VERIFY_PARAM_pop_free(st, free_func) SKM_sk_pop_free(X509_VERIFY
#define sk_X509_VERIFY_PARAM_shift(st) SKM_sk_shift(X509_VERIFY_PARAM, (st))
#define sk_X509_VERIFY_PARAM_pop(st) SKM_sk_pop(X509_VERIFY_PARAM, (st))
#define sk_X509_VERIFY_PARAM_sort(st) SKM_sk_sort(X509_VERIFY_PARAM, (st))
#define sk_X509_VERIFY_PARAM_is_sorted(st) SKM_sk_is_sorted(X509_VERIFY_PARAM, (

#define sk_nid_triple_new(cmp) SKM_sk_new(nid_triple, (cmp))
#define sk_nid_triple_new_null() SKM_sk_new_null(nid_triple)
#define sk_nid_triple_free(st) SKM_sk_free(nid_triple, (st))
#define sk_nid_triple_num(st) SKM_sk_num(nid_triple, (st))
#define sk_nid_triple_value(st, i) SKM_sk_value(nid_triple, (st), (i))
#define sk_nid_triple_set(st, i, val) SKM_sk_set(nid_triple, (st), (i), (val))
#define sk_nid_triple_zero(st) SKM_sk_zero(nid_triple, (st))
#define sk_nid_triple_push(st, val) SKM_sk_push(nid_triple, (st), (val))
#define sk_nid_triple_unshift(st, val) SKM_sk_unshift(nid_triple, (st), (val))
#define sk_nid_triple_find(st, val) SKM_sk_find(nid_triple, (st), (val))
#define sk_nid_triple_find_ex(st, val) SKM_sk_find_ex(nid_triple, (st), (val))
#define sk_nid_triple_delete(st, i) SKM_sk_delete(nid_triple, (st), (i))
#define sk_nid_triple_delete_ptr(st, ptr) SKM_sk_delete_ptr(nid_triple, (st), (p
#define sk_nid_triple_insert(st, val, i) SKM_sk_insert(nid_triple, (st), (val),
#define sk_nid_triple_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(nid_triple, (st)
#define sk_nid_triple_dup(st) SKM_sk_dup(nid_triple, st)
#define sk_nid_triple_pop_free(st, free_func) SKM_sk_pop_free(nid_triple, (st),
#define sk_nid_triple_shift(st) SKM_sk_shift(nid_triple, (st))
#define sk_nid_triple_pop(st) SKM_sk_pop(nid_triple, (st))
#define sk_nid_triple_sort(st) SKM_sk_sort(nid_triple, (st))
#define sk_nid_triple_is_sorted(st) SKM_sk_is_sorted(nid_triple, (st))

#define sk_void_new(cmp) SKM_sk_new(void, (cmp))
#define sk_void_new_null() SKM_sk_new_null(void)
#define sk_void_free(st) SKM_sk_free(void, (st))
#define sk_void_num(st) SKM_sk_num(void, (st))
#define sk_void_value(st, i) SKM_sk_value(void, (st), (i))
#define sk_void_set(st, i, val) SKM_sk_set(void, (st), (i), (val))
#define sk_void_zero(st) SKM_sk_zero(void, (st))
#define sk_void_push(st, val) SKM_sk_push(void, (st), (val))
```

```
#define sk_void_unshift(st, val) SKM_sk_unshift(void, (st), (val))
#define sk_void_find(st, val) SKM_sk_find(void, (st), (val))
#define sk_void_find_ex(st, val) SKM_sk_find_ex(void, (st), (val))
#define sk_void_delete(st, i) SKM_sk_delete(void, (st), (i))
#define sk_void_delete_ptr(st, ptr) SKM_sk_delete_ptr(void, (st), (ptr))
#define sk_void_insert(st, val, i) SKM_sk_insert(void, (st), (val), (i))
#define sk_void_set_cmp_func(st, cmp) SKM_sk_set_cmp_func(void, (st), (cmp))
#define sk_void_dup(st) SKM_sk_dup(void, st)
#define sk_void_pop_free(st, free_func) SKM_sk_pop_free(void, (st), (free_func))
#define sk_void_shift(st) SKM_sk_shift(void, (st))
#define sk_void_pop(st) SKM_sk_pop(void, (st))
#define sk_void_sort(st) SKM_sk_sort(void, (st))
#define sk_void_is_sorted(st) SKM_sk_is_sorted(void, (st))

#define sk_OPENSSL_STRING_new(cmp) ((STACK_OF(OPENSSL_STRING) *)sk_new(CHECKED_S
#define sk_OPENSSL_STRING_new_null() ((STACK_OF(OPENSSL_STRING) *)sk_new_null())
#define sk_OPENSSL_STRING_push(st, val) sk_push(CHECKED_STACK_OF(OPENSSL_STRING,
#define sk_OPENSSL_STRING_find(st, val) sk_find(CHECKED_STACK_OF(OPENSSL_STRING,
#define sk_OPENSSL_STRING_value(st, i) ((OPENSSL_STRING)sk_value(CHECKED_STACK_O
#define sk_OPENSSL_STRING_num(st) SKM_sk_num(OPENSSL_STRING, st)
#define sk_OPENSSL_STRING_pop_free(st, free_func) sk_pop_free(CHECKED_STACK_OF(O
#define sk_OPENSSL_STRING_insert(st, val, i) sk_insert(CHECKED_STACK_OF(OPENSSL_
#define sk_OPENSSL_STRING_free(st) SKM_sk_free(OPENSSL_STRING, st)
#define sk_OPENSSL_STRING_set(st, i, val) sk_set(CHECKED_STACK_OF(OPENSSL_STRING
#define sk_OPENSSL_STRING_zero(st) SKM_sk_zero(OPENSSL_STRING, (st))
#define sk_OPENSSL_STRING_unshift(st, val) sk_unshift(CHECKED_STACK_OF(OPENSSL_S
#define sk_OPENSSL_STRING_find_ex(st, val) sk_find_ex((_STACK *)CHECKED_CONST_PT
#define sk_OPENSSL_STRING_delete(st, i) SKM_sk_delete(OPENSSL_STRING, (st), (i))
#define sk_OPENSSL_STRING_delete_ptr(st, ptr) (OPENSSL_STRING *)sk_delete_ptr(CH
#define sk_OPENSSL_STRING_set_cmp_func(st, cmp)  \
        ((int (*)(const char * const *,const char * const *)) \
        sk_set_cmp_func(CHECKED_STACK_OF(OPENSSL_STRING, st), CHECKED_SK_CMP_FUN
#define sk_OPENSSL_STRING_dup(st) SKM_sk_dup(OPENSSL_STRING, st)
#define sk_OPENSSL_STRING_shift(st) SKM_sk_shift(OPENSSL_STRING, (st))
#define sk_OPENSSL_STRING_pop(st) (char *)sk_pop(CHECKED_STACK_OF(OPENSSL_STRING
#define sk_OPENSSL_STRING_sort(st) SKM_sk_sort(OPENSSL_STRING, (st))
#define sk_OPENSSL_STRING_is_sorted(st) SKM_sk_is_sorted(OPENSSL_STRING, (st))


#define sk_OPENSSL_BLOCK_new(cmp) ((STACK_OF(OPENSSL_BLOCK) *)sk_new(CHECKED_SK_
#define sk_OPENSSL_BLOCK_new_null() ((STACK_OF(OPENSSL_BLOCK) *)sk_new_null())
#define sk_OPENSSL_BLOCK_push(st, val) sk_push(CHECKED_STACK_OF(OPENSSL_BLOCK, s
#define sk_OPENSSL_BLOCK_find(st, val) sk_find(CHECKED_STACK_OF(OPENSSL_BLOCK, s
#define sk_OPENSSL_BLOCK_value(st, i) ((OPENSSL_BLOCK)sk_value(CHECKED_STACK_OF(
#define sk_OPENSSL_BLOCK_num(st) SKM_sk_num(OPENSSL_BLOCK, st)
#define sk_OPENSSL_BLOCK_pop_free(st, free_func) sk_pop_free(CHECKED_STACK_OF(OP
#define sk_OPENSSL_BLOCK_insert(st, val, i) sk_insert(CHECKED_STACK_OF(OPENSSL_B
#define sk_OPENSSL_BLOCK_free(st) SKM_sk_free(OPENSSL_BLOCK, st)
#define sk_OPENSSL_BLOCK_set(st, i, val) sk_set(CHECKED_STACK_OF(OPENSSL_BLOCK,
#define sk_OPENSSL_BLOCK_zero(st) SKM_sk_zero(OPENSSL_BLOCK, (st))
#define sk_OPENSSL_BLOCK_unshift(st, val) sk_unshift(CHECKED_STACK_OF(OPENSSL_BL
#define sk_OPENSSL_BLOCK_find_ex(st, val) sk_find_ex((_STACK *)CHECKED_CONST_PTR
#define sk_OPENSSL_BLOCK_delete(st, i) SKM_sk_delete(OPENSSL_BLOCK, (st), (i))
#define sk_OPENSSL_BLOCK_delete_ptr(st, ptr) (OPENSSL_BLOCK *)sk_delete_ptr(CHEC
#define sk_OPENSSL_BLOCK_set_cmp_func(st, cmp)  \
        ((int (*)(const void * const *,const void * const *)) \
        sk_set_cmp_func(CHECKED_STACK_OF(OPENSSL_BLOCK, st), CHECKED_SK_CMP_FUNC
#define sk_OPENSSL_BLOCK_dup(st) SKM_sk_dup(OPENSSL_BLOCK, st)
#define sk_OPENSSL_BLOCK_shift(st) SKM_sk_shift(OPENSSL_BLOCK, (st))
#define sk_OPENSSL_BLOCK_pop(st) (void *)sk_pop(CHECKED_STACK_OF(OPENSSL_BLOCK,
#define sk_OPENSSL_BLOCK_sort(st) SKM_sk_sort(OPENSSL_BLOCK, (st))
#define sk_OPENSSL_BLOCK_is_sorted(st) SKM_sk_is_sorted(OPENSSL_BLOCK, (st))


#define sk_OPENSSL_PSTRING_new(cmp) ((STACK_OF(OPENSSL_PSTRING) *)sk_new(CHECKED
#define sk_OPENSSL_PSTRING_new_null() ((STACK_OF(OPENSSL_PSTRING) *)sk_new_null(
```



```
#define sk_OPENSSL_PSTRING_push(st, val) sk_push(CHECKED_STACK_OF(OPENSSL_PSTRIN
#define sk_OPENSSL_PSTRING_find(st, val) sk_find(CHECKED_STACK_OF(OPENSSL_PSTRIN
#define sk_OPENSSL_PSTRING_value(st, i) ((OPENSSL_PSTRING)sk_value(CHECKED_STACK
#define sk_OPENSSL_PSTRING_num(st) SKM_sk_num(OPENSSL_PSTRING, st)
#define sk_OPENSSL_PSTRING_pop_free(st, free_func) sk_pop_free(CHECKED_STACK_OF(
#define sk_OPENSSL_PSTRING_insert(st, val, i) sk_insert(CHECKED_STACK_OF(OPENSSL
#define sk_OPENSSL_PSTRING_free(st) SKM_sk_free(OPENSSL_PSTRING, st)
#define sk_OPENSSL_PSTRING_set(st, i, val) sk_set(CHECKED_STACK_OF(OPENSSL_PSTRI
#define sk_OPENSSL_PSTRING_zero(st) SKM_sk_zero(OPENSSL_PSTRING, (st))
#define sk_OPENSSL_PSTRING_unshift(st, val) sk_unshift(CHECKED_STACK_OF(OPENSSL_
#define sk_OPENSSL_PSTRING_find_ex(st, val) sk_find_ex((_STACK *)CHECKED_CONST_P
#define sk_OPENSSL_PSTRING_delete(st, i) SKM_sk_delete(OPENSSL_PSTRING, (st), (i
#define sk_OPENSSL_PSTRING_delete_ptr(st, ptr) (OPENSSL_PSTRING *)sk_delete_ptr(
#define sk_OPENSSL_PSTRING_set_cmp_func(st, cmp)  \
        ((int (*)(const OPENSSL_STRING * const *,const OPENSSL_STRING * const *)
        sk_set_cmp_func(CHECKED_STACK_OF(OPENSSL_PSTRING, st), CHECKED_SK_CMP_FU
#define sk_OPENSSL_PSTRING_dup(st) SKM_sk_dup(OPENSSL_PSTRING, st)
#define sk_OPENSSL_PSTRING_shift(st) SKM_sk_shift(OPENSSL_PSTRING, (st))
#define sk_OPENSSL_PSTRING_pop(st) (OPENSSL_STRING *)sk_pop(CHECKED_STACK_OF(OPE
#define sk_OPENSSL_PSTRING_sort(st) SKM_sk_sort(OPENSSL_PSTRING, (st))
#define sk_OPENSSL_PSTRING_is_sorted(st) SKM_sk_is_sorted(OPENSSL_PSTRING, (st))


#define d2i_ASN1_SET_OF_ACCESS_DESCRIPTION(st, pp, length, d2i_func, free_func,
        SKM_ASN1_SET_OF_d2i(ACCESS_DESCRIPTION, (st), (pp), (length), (d2i_func)
#define i2d_ASN1_SET_OF_ACCESS_DESCRIPTION(st, pp, i2d_func, ex_tag, ex_class, i
        SKM_ASN1_SET_OF_i2d(ACCESS_DESCRIPTION, (st), (pp), (i2d_func), (ex_tag)
#define ASN1_seq_pack_ACCESS_DESCRIPTION(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(ACCESS_DESCRIPTION, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_ACCESS_DESCRIPTION(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(ACCESS_DESCRIPTION, (buf), (len), (d2i_func), (free_

#define d2i_ASN1_SET_OF_ASN1_INTEGER(st, pp, length, d2i_func, free_func, ex_tag
        SKM_ASN1_SET_OF_d2i(ASN1_INTEGER, (st), (pp), (length), (d2i_func), (fre
#define i2d_ASN1_SET_OF_ASN1_INTEGER(st, pp, i2d_func, ex_tag, ex_class, is_set)
        SKM_ASN1_SET_OF_i2d(ASN1_INTEGER, (st), (pp), (i2d_func), (ex_tag), (ex_
#define ASN1_seq_pack_ASN1_INTEGER(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(ASN1_INTEGER, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_ASN1_INTEGER(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(ASN1_INTEGER, (buf), (len), (d2i_func), (free_func))

#define d2i_ASN1_SET_OF_ASN1_OBJECT(st, pp, length, d2i_func, free_func, ex_tag,
        SKM_ASN1_SET_OF_d2i(ASN1_OBJECT, (st), (pp), (length), (d2i_func), (free
#define i2d_ASN1_SET_OF_ASN1_OBJECT(st, pp, i2d_func, ex_tag, ex_class, is_set)
        SKM_ASN1_SET_OF_i2d(ASN1_OBJECT, (st), (pp), (i2d_func), (ex_tag), (ex_c
#define ASN1_seq_pack_ASN1_OBJECT(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(ASN1_OBJECT, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_ASN1_OBJECT(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(ASN1_OBJECT, (buf), (len), (d2i_func), (free_func))

#define d2i_ASN1_SET_OF_ASN1_TYPE(st, pp, length, d2i_func, free_func, ex_tag, e
        SKM_ASN1_SET_OF_d2i(ASN1_TYPE, (st), (pp), (length), (d2i_func), (free_f
#define i2d_ASN1_SET_OF_ASN1_TYPE(st, pp, i2d_func, ex_tag, ex_class, is_set) \
        SKM_ASN1_SET_OF_i2d(ASN1_TYPE, (st), (pp), (i2d_func), (ex_tag), (ex_cla
#define ASN1_seq_pack_ASN1_TYPE(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(ASN1_TYPE, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_ASN1_TYPE(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(ASN1_TYPE, (buf), (len), (d2i_func), (free_func))

#define d2i_ASN1_SET_OF_ASN1_UTF8STRING(st, pp, length, d2i_func, free_func, ex_
        SKM_ASN1_SET_OF_d2i(ASN1_UTF8STRING, (st), (pp), (length), (d2i_func), (
#define i2d_ASN1_SET_OF_ASN1_UTF8STRING(st, pp, i2d_func, ex_tag, ex_class, is_s
        SKM_ASN1_SET_OF_i2d(ASN1_UTF8STRING, (st), (pp), (i2d_func), (ex_tag), (
#define ASN1_seq_pack_ASN1_UTF8STRING(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(ASN1_UTF8STRING, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_ASN1_UTF8STRING(buf, len, d2i_func, free_func) \
```

```
2240         SKM_ASN1_seq_unpack(ASN1_UTF8STRING, (buf), (len), (d2i_func), (free_fun

2242 #define d2i_ASN1_SET_OF_DIST_POINT(st, pp, length, d2i_func, free_func, ex_tag,
2243         SKM_ASN1_SET_OF_d2i(DIST_POINT, (st), (pp), (length), (d2i_func), (free_
2244 #define i2d_ASN1_SET_OF_DIST_POINT(st, pp, i2d_func, ex_tag, ex_class, is_set) \
2245         SKM_ASN1_SET_OF_i2d(DIST_POINT, (st), (pp), (i2d_func), (ex_tag), (ex_cl
2246 #define ASN1_seq_pack_DIST_POINT(st, i2d_func, buf, len) \
2247         SKM_ASN1_seq_pack(DIST_POINT, (st), (i2d_func), (buf), (len))
2248 #define ASN1_seq_unpack_DIST_POINT(buf, len, d2i_func, free_func) \
2249         SKM_ASN1_seq_unpack(DIST_POINT, (buf), (len), (d2i_func), (free_func))

2251 #define d2i_ASN1_SET_OF_ESS_CERT_ID(st, pp, length, d2i_func, free_func, ex_tag,
2252         SKM_ASN1_SET_OF_d2i(ESS_CERT_ID, (st), (pp), (length), (d2i_func), (free
2253 #define i2d_ASN1_SET_OF_ESS_CERT_ID(st, pp, i2d_func, ex_tag, ex_class, is_set)
2254         SKM_ASN1_SET_OF_i2d(ESS_CERT_ID, (st), (pp), (i2d_func), (ex_tag), (ex_c
2255 #define ASN1_seq_pack_ESS_CERT_ID(st, i2d_func, buf, len) \
2256         SKM_ASN1_seq_pack(ESS_CERT_ID, (st), (i2d_func), (buf), (len))
2257 #define ASN1_seq_unpack_ESS_CERT_ID(buf, len, d2i_func, free_func) \
2258         SKM_ASN1_seq_unpack(ESS_CERT_ID, (buf), (len), (d2i_func), (free_func))

2260 #define d2i_ASN1_SET_OF_EVP_MD(st, pp, length, d2i_func, free_func, ex_tag, ex_c
2261         SKM_ASN1_SET_OF_d2i(EVP_MD, (st), (pp), (length), (d2i_func), (free_func
2262 #define i2d_ASN1_SET_OF_EVP_MD(st, pp, i2d_func, ex_tag, ex_class, is_set) \
2263         SKM_ASN1_SET_OF_i2d(EVP_MD, (st), (pp), (i2d_func), (ex_tag), (ex_class)
2264 #define ASN1_seq_pack_EVP_MD(st, i2d_func, buf, len) \
2265         SKM_ASN1_seq_pack(EVP_MD, (st), (i2d_func), (buf), (len))
2266 #define ASN1_seq_unpack_EVP_MD(buf, len, d2i_func, free_func) \
2267         SKM_ASN1_seq_unpack(EVP_MD, (buf), (len), (d2i_func), (free_func))

2269 #define d2i_ASN1_SET_OF_GENERAL_NAME(st, pp, length, d2i_func, free_func, ex_tag
2270         SKM_ASN1_SET_OF_d2i(GENERAL_NAME, (st), (pp), (length), (d2i_func), (fre
2271 #define i2d_ASN1_SET_OF_GENERAL_NAME(st, pp, i2d_func, ex_tag, ex_class, is_set)
2272         SKM_ASN1_SET_OF_i2d(GENERAL_NAME, (st), (pp), (i2d_func), (ex_tag), (ex_
2273 #define ASN1_seq_pack_GENERAL_NAME(st, i2d_func, buf, len) \
2274         SKM_ASN1_seq_pack(GENERAL_NAME, (st), (i2d_func), (buf), (len))
2275 #define ASN1_seq_unpack_GENERAL_NAME(buf, len, d2i_func, free_func) \
2276         SKM_ASN1_seq_unpack(GENERAL_NAME, (buf), (len), (d2i_func), (free_func))

2278 #define d2i_ASN1_SET_OF_OCSP_ONEREQ(st, pp, length, d2i_func, free_func, ex_tag,
2279         SKM_ASN1_SET_OF_d2i(OCSP_ONEREQ, (st), (pp), (length), (d2i_func), (free
2280 #define i2d_ASN1_SET_OF_OCSP_ONEREQ(st, pp, i2d_func, ex_tag, ex_class, is_set)
2281         SKM_ASN1_SET_OF_i2d(OCSP_ONEREQ, (st), (pp), (i2d_func), (ex_tag), (ex_c
2282 #define ASN1_seq_pack_OCSP_ONEREQ(st, i2d_func, buf, len) \
2283         SKM_ASN1_seq_pack(OCSP_ONEREQ, (st), (i2d_func), (buf), (len))
2284 #define ASN1_seq_unpack_OCSP_ONEREQ(buf, len, d2i_func, free_func) \
2285         SKM_ASN1_seq_unpack(OCSP_ONEREQ, (buf), (len), (d2i_func), (free_func))

2287 #define d2i_ASN1_SET_OF_OCSP_SINGLERESP(st, pp, length, d2i_func, free_func, ex_
2288         SKM_ASN1_SET_OF_d2i(OCSP_SINGLERESP, (st), (pp), (length), (d2i_func), (
2289 #define i2d_ASN1_SET_OF_OCSP_SINGLERESP(st, pp, i2d_func, ex_tag, ex_class, is_s
2290         SKM_ASN1_SET_OF_i2d(OCSP_SINGLERESP, (st), (pp), (i2d_func), (ex_tag), (
2291 #define ASN1_seq_pack_OCSP_SINGLERESP(st, i2d_func, buf, len) \
2292         SKM_ASN1_seq_pack(OCSP_SINGLERESP, (st), (i2d_func), (buf), (len))
2293 #define ASN1_seq_unpack_OCSP_SINGLERESP(buf, len, d2i_func, free_func) \
2294         SKM_ASN1_seq_unpack(OCSP_SINGLERESP, (buf), (len), (d2i_func), (free_fun

2296 #define d2i_ASN1_SET_OF_PKCS12_SAFEBAG(st, pp, length, d2i_func, free_func, ex_t
2297         SKM_ASN1_SET_OF_d2i(PKCS12_SAFEBAG, (st), (pp), (length), (d2i_func), (f
2298 #define i2d_ASN1_SET_OF_PKCS12_SAFEBAG(st, pp, i2d_func, ex_tag, ex_class, is_se
2299         SKM_ASN1_SET_OF_i2d(PKCS12_SAFEBAG, (st), (pp), (i2d_func), (ex_tag), (e
2300 #define ASN1_seq_pack_PKCS12_SAFEBAG(st, i2d_func, buf, len) \
2301         SKM_ASN1_seq_pack(PKCS12_SAFEBAG, (st), (i2d_func), (buf), (len))
2302 #define ASN1_seq_unpack_PKCS12_SAFEBAG(buf, len, d2i_func, free_func) \
2303         SKM_ASN1_seq_unpack(PKCS12_SAFEBAG, (buf), (len), (d2i_func), (free_func

2305 #define d2i_ASN1_SET_OF_PKCS7(st, pp, length, d2i_func, free_func, ex_tag, ex_cl
```



```
2306         SKM_ASN1_SET_OF_d2i(PKCS7, (st), (pp), (length), (d2i_func), (free_func)
2307 #define i2d_ASN1_SET_OF_PKCS7(st, pp, i2d_func, ex_tag, ex_class, is_set) \
2308         SKM_ASN1_SET_OF_i2d(PKCS7, (st), (pp), (i2d_func), (ex_tag), (ex_class),
2309 #define ASN1_seq_pack_PKCS7(st, i2d_func, buf, len) \
2310         SKM_ASN1_seq_pack(PKCS7, (st), (i2d_func), (buf), (len))
2311 #define ASN1_seq_unpack_PKCS7(buf, len, d2i_func, free_func) \
2312         SKM_ASN1_seq_unpack(PKCS7, (buf), (len), (d2i_func), (free_func))

2314 #define d2i_ASN1_SET_OF_PKCS7_RECIP_INFO(st, pp, length, d2i_func, free_func, ex
2315         SKM_ASN1_SET_OF_d2i(PKCS7_RECIP_INFO, (st), (pp), (length), (d2i_func),
2316 #define i2d_ASN1_SET_OF_PKCS7_RECIP_INFO(st, pp, i2d_func, ex_tag, ex_class, is_
2317         SKM_ASN1_SET_OF_i2d(PKCS7_RECIP_INFO, (st), (pp), (i2d_func), (ex_tag),
2318 #define ASN1_seq_pack_PKCS7_RECIP_INFO(st, i2d_func, buf, len) \
2319         SKM_ASN1_seq_pack(PKCS7_RECIP_INFO, (st), (i2d_func), (buf), (len))
2320 #define ASN1_seq_unpack_PKCS7_RECIP_INFO(buf, len, d2i_func, free_func) \
2321         SKM_ASN1_seq_unpack(PKCS7_RECIP_INFO, (buf), (len), (d2i_func), (free_fu

2323 #define d2i_ASN1_SET_OF_PKCS7_SIGNER_INFO(st, pp, length, d2i_func, free_func, e
2324         SKM_ASN1_SET_OF_d2i(PKCS7_SIGNER_INFO, (st), (pp), (length), (d2i_func),
2325 #define i2d_ASN1_SET_OF_PKCS7_SIGNER_INFO(st, pp, i2d_func, ex_tag, ex_class, is
2326         SKM_ASN1_SET_OF_i2d(PKCS7_SIGNER_INFO, (st), (pp), (i2d_func), (ex_tag),
2327 #define ASN1_seq_pack_PKCS7_SIGNER_INFO(st, i2d_func, buf, len) \
2328         SKM_ASN1_seq_pack(PKCS7_SIGNER_INFO, (st), (i2d_func), (buf), (len))
2329 #define ASN1_seq_unpack_PKCS7_SIGNER_INFO(buf, len, d2i_func, free_func) \
2330         SKM_ASN1_seq_unpack(PKCS7_SIGNER_INFO, (buf), (len), (d2i_func), (free_f

2332 #define d2i_ASN1_SET_OF_POLICYINFO(st, pp, length, d2i_func, free_func, ex_tag,
2333         SKM_ASN1_SET_OF_d2i(POLICYINFO, (st), (pp), (length), (d2i_func), (free_
2334 #define i2d_ASN1_SET_OF_POLICYINFO(st, pp, i2d_func, ex_tag, ex_class, is_set) \
2335         SKM_ASN1_SET_OF_i2d(POLICYINFO, (st), (pp), (i2d_func), (ex_tag), (ex_cl
2336 #define ASN1_seq_pack_POLICYINFO(st, i2d_func, buf, len) \
2337         SKM_ASN1_seq_pack(POLICYINFO, (st), (i2d_func), (buf), (len))
2338 #define ASN1_seq_unpack_POLICYINFO(buf, len, d2i_func, free_func) \
2339         SKM_ASN1_seq_unpack(POLICYINFO, (buf), (len), (d2i_func), (free_func))

2341 #define d2i_ASN1_SET_OF_POLICYQUALINFO(st, pp, length, d2i_func, free_func, ex_t
2342         SKM_ASN1_SET_OF_d2i(POLICYQUALINFO, (st), (pp), (length), (d2i_func), (f
2343 #define i2d_ASN1_SET_OF_POLICYQUALINFO(st, pp, i2d_func, ex_tag, ex_class, is_se
2344         SKM_ASN1_SET_OF_i2d(POLICYQUALINFO, (st), (pp), (i2d_func), (ex_tag), (e
2345 #define ASN1_seq_pack_POLICYQUALINFO(st, i2d_func, buf, len) \
2346         SKM_ASN1_seq_pack(POLICYQUALINFO, (st), (i2d_func), (buf), (len))
2347 #define ASN1_seq_unpack_POLICYQUALINFO(buf, len, d2i_func, free_func) \
2348         SKM_ASN1_seq_unpack(POLICYQUALINFO, (buf), (len), (d2i_func), (free_func

2350 #define d2i_ASN1_SET_OF_SXNETID(st, pp, length, d2i_func, free_func, ex_tag, ex_
2351         SKM_ASN1_SET_OF_d2i(SXNETID, (st), (pp), (length), (d2i_func), (free_fun
2352 #define i2d_ASN1_SET_OF_SXNETID(st, pp, i2d_func, ex_tag, ex_class, is_set) \
2353         SKM_ASN1_SET_OF_i2d(SXNETID, (st), (pp), (i2d_func), (ex_tag), (ex_class
2354 #define ASN1_seq_pack_SXNETID(st, i2d_func, buf, len) \
2355         SKM_ASN1_seq_pack(SXNETID, (st), (i2d_func), (buf), (len))
2356 #define ASN1_seq_unpack_SXNETID(buf, len, d2i_func, free_func) \
2357         SKM_ASN1_seq_unpack(SXNETID, (buf), (len), (d2i_func), (free_func))

2359 #define d2i_ASN1_SET_OF_X509(st, pp, length, d2i_func, free_func, ex_tag, ex_cla
2360         SKM_ASN1_SET_OF_d2i(X509, (st), (pp), (length), (d2i_func), (free_func),
2361 #define i2d_ASN1_SET_OF_X509(st, pp, i2d_func, ex_tag, ex_class, is_set) \
2362         SKM_ASN1_SET_OF_i2d(X509, (st), (pp), (i2d_func), (ex_tag), (ex_class),
2363 #define ASN1_seq_pack_X509(st, i2d_func, buf, len) \
2364         SKM_ASN1_seq_pack(X509, (st), (i2d_func), (buf), (len))
2365 #define ASN1_seq_unpack_X509(buf, len, d2i_func, free_func) \
2366         SKM_ASN1_seq_unpack(X509, (buf), (len), (d2i_func), (free_func))

2368 #define d2i_ASN1_SET_OF_X509_ALGOR(st, pp, length, d2i_func, free_func, ex_tag,
2369         SKM_ASN1_SET_OF_d2i(X509_ALGOR, (st), (pp), (length), (d2i_func), (free_
2370 #define i2d_ASN1_SET_OF_X509_ALGOR(st, pp, i2d_func, ex_tag, ex_class, is_set) \
2371         SKM_ASN1_SET_OF_i2d(X509_ALGOR, (st), (pp), (i2d_func), (ex_tag), (ex_cl
```

```
#define ASN1_seq_pack_X509_ALGOR(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(X509_ALGOR, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_X509_ALGOR(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(X509_ALGOR, (buf), (len), (d2i_func), (free_func))

#define d2i_ASN1_SET_OF_X509_ATTRIBUTE(st, pp, length, d2i_func, free_func, ex_t
        SKM_ASN1_SET_OF_d2i(X509_ATTRIBUTE, (st), (pp), (length), (d2i_func), (f
#define i2d_ASN1_SET_OF_X509_ATTRIBUTE(st, pp, i2d_func, ex_tag, ex_class, is_se
        SKM_ASN1_SET_OF_i2d(X509_ATTRIBUTE, (st), (pp), (i2d_func), (ex_tag), (e
#define ASN1_seq_pack_X509_ATTRIBUTE(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(X509_ATTRIBUTE, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_X509_ATTRIBUTE(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(X509_ATTRIBUTE, (buf), (len), (d2i_func), (free_func

#define d2i_ASN1_SET_OF_X509_CRL(st, pp, length, d2i_func, free_func, ex_tag, ex
        SKM_ASN1_SET_OF_d2i(X509_CRL, (st), (pp), (length), (d2i_func), (free_fu
#define i2d_ASN1_SET_OF_X509_CRL(st, pp, i2d_func, ex_tag, ex_class, is_set) \
        SKM_ASN1_SET_OF_i2d(X509_CRL, (st), (pp), (i2d_func), (ex_tag), (ex_clas
#define ASN1_seq_pack_X509_CRL(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(X509_CRL, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_X509_CRL(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(X509_CRL, (buf), (len), (d2i_func), (free_func))

#define d2i_ASN1_SET_OF_X509_EXTENSION(st, pp, length, d2i_func, free_func, ex_t
        SKM_ASN1_SET_OF_d2i(X509_EXTENSION, (st), (pp), (length), (d2i_func), (f
#define i2d_ASN1_SET_OF_X509_EXTENSION(st, pp, i2d_func, ex_tag, ex_class, is_se
        SKM_ASN1_SET_OF_i2d(X509_EXTENSION, (st), (pp), (i2d_func), (ex_tag), (e
#define ASN1_seq_pack_X509_EXTENSION(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(X509_EXTENSION, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_X509_EXTENSION(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(X509_EXTENSION, (buf), (len), (d2i_func), (free_func

#define d2i_ASN1_SET_OF_X509_NAME_ENTRY(st, pp, length, d2i_func, free_func, ex_
        SKM_ASN1_SET_OF_d2i(X509_NAME_ENTRY, (st), (pp), (length), (d2i_func), (
#define i2d_ASN1_SET_OF_X509_NAME_ENTRY(st, pp, i2d_func, ex_tag, ex_class, is_s
        SKM_ASN1_SET_OF_i2d(X509_NAME_ENTRY, (st), (pp), (i2d_func), (ex_tag), (
#define ASN1_seq_pack_X509_NAME_ENTRY(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(X509_NAME_ENTRY, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_X509_NAME_ENTRY(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(X509_NAME_ENTRY, (buf), (len), (d2i_func), (free_fun

#define d2i_ASN1_SET_OF_X509_REVOKED(st, pp, length, d2i_func, free_func, ex_tag
        SKM_ASN1_SET_OF_d2i(X509_REVOKED, (st), (pp), (length), (d2i_func), (fre
#define i2d_ASN1_SET_OF_X509_REVOKED(st, pp, i2d_func, ex_tag, ex_class, is_set)
        SKM_ASN1_SET_OF_i2d(X509_REVOKED, (st), (pp), (i2d_func), (ex_tag), (ex_
#define ASN1_seq_pack_X509_REVOKED(st, i2d_func, buf, len) \
        SKM_ASN1_seq_pack(X509_REVOKED, (st), (i2d_func), (buf), (len))
#define ASN1_seq_unpack_X509_REVOKED(buf, len, d2i_func, free_func) \
        SKM_ASN1_seq_unpack(X509_REVOKED, (buf), (len), (d2i_func), (free_func))

#define PKCS12_decrypt_d2i_PKCS12_SAFEBAG(algor, d2i_func, free_func, pass, pass
        SKM_PKCS12_decrypt_d2i(PKCS12_SAFEBAG, (algor), (d2i_func), (free_func),

#define PKCS12_decrypt_d2i_PKCS7(algor, d2i_func, free_func, pass, passlen, oct,
        SKM_PKCS12_decrypt_d2i(PKCS7, (algor), (d2i_func), (free_func), (pass),

#define lh_ADDED_OBJ_new() LHM_lh_new(ADDED_OBJ,added_obj)
#define lh_ADDED_OBJ_insert(lh,inst) LHM_lh_insert(ADDED_OBJ,lh,inst)
#define lh_ADDED_OBJ_retrieve(lh,inst) LHM_lh_retrieve(ADDED_OBJ,lh,inst)
#define lh_ADDED_OBJ_delete(lh,inst) LHM_lh_delete(ADDED_OBJ,lh,inst)
#define lh_ADDED_OBJ_doall(lh,fn) LHM_lh_doall(ADDED_OBJ,lh,fn)
#define lh_ADDED_OBJ_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(ADDED_OBJ,lh,fn,arg_type,arg)
#define lh_ADDED_OBJ_error(lh) LHM_lh_error(ADDED_OBJ,lh)
#define lh_ADDED_OBJ_num_items(lh) LHM_lh_num_items(ADDED_OBJ,lh)
#define lh_ADDED_OBJ_down_load(lh) LHM_lh_down_load(ADDED_OBJ,lh)
```



```
#define lh_ADDED_OBJ_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(ADDED_OBJ,lh,out)
#define lh_ADDED_OBJ_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(ADDED_OBJ,lh,out)
#define lh_ADDED_OBJ_stats_bio(lh,out) \
  LHM_lh_stats_bio(ADDED_OBJ,lh,out)
#define lh_ADDED_OBJ_free(lh) LHM_lh_free(ADDED_OBJ,lh)

#define lh_APP_INFO_new() LHM_lh_new(APP_INFO,app_info)
#define lh_APP_INFO_insert(lh,inst) LHM_lh_insert(APP_INFO,lh,inst)
#define lh_APP_INFO_retrieve(lh,inst) LHM_lh_retrieve(APP_INFO,lh,inst)
#define lh_APP_INFO_delete(lh,inst) LHM_lh_delete(APP_INFO,lh,inst)
#define lh_APP_INFO_doall(lh,fn) LHM_lh_doall(APP_INFO,lh,fn)
#define lh_APP_INFO_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(APP_INFO,lh,fn,arg_type,arg)
#define lh_APP_INFO_error(lh) LHM_lh_error(APP_INFO,lh)
#define lh_APP_INFO_num_items(lh) LHM_lh_num_items(APP_INFO,lh)
#define lh_APP_INFO_down_load(lh) LHM_lh_down_load(APP_INFO,lh)
#define lh_APP_INFO_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(APP_INFO,lh,out)
#define lh_APP_INFO_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(APP_INFO,lh,out)
#define lh_APP_INFO_stats_bio(lh,out) \
  LHM_lh_stats_bio(APP_INFO,lh,out)
#define lh_APP_INFO_free(lh) LHM_lh_free(APP_INFO,lh)

#define lh_CONF_VALUE_new() LHM_lh_new(CONF_VALUE,conf_value)
#define lh_CONF_VALUE_insert(lh,inst) LHM_lh_insert(CONF_VALUE,lh,inst)
#define lh_CONF_VALUE_retrieve(lh,inst) LHM_lh_retrieve(CONF_VALUE,lh,inst)
#define lh_CONF_VALUE_delete(lh,inst) LHM_lh_delete(CONF_VALUE,lh,inst)
#define lh_CONF_VALUE_doall(lh,fn) LHM_lh_doall(CONF_VALUE,lh,fn)
#define lh_CONF_VALUE_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(CONF_VALUE,lh,fn,arg_type,arg)
#define lh_CONF_VALUE_error(lh) LHM_lh_error(CONF_VALUE,lh)
#define lh_CONF_VALUE_num_items(lh) LHM_lh_num_items(CONF_VALUE,lh)
#define lh_CONF_VALUE_down_load(lh) LHM_lh_down_load(CONF_VALUE,lh)
#define lh_CONF_VALUE_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(CONF_VALUE,lh,out)
#define lh_CONF_VALUE_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(CONF_VALUE,lh,out)
#define lh_CONF_VALUE_stats_bio(lh,out) \
  LHM_lh_stats_bio(CONF_VALUE,lh,out)
#define lh_CONF_VALUE_free(lh) LHM_lh_free(CONF_VALUE,lh)

#define lh_ENGINE_PILE_new() LHM_lh_new(ENGINE_PILE,engine_pile)
#define lh_ENGINE_PILE_insert(lh,inst) LHM_lh_insert(ENGINE_PILE,lh,inst)
#define lh_ENGINE_PILE_retrieve(lh,inst) LHM_lh_retrieve(ENGINE_PILE,lh,inst)
#define lh_ENGINE_PILE_delete(lh,inst) LHM_lh_delete(ENGINE_PILE,lh,inst)
#define lh_ENGINE_PILE_doall(lh,fn) LHM_lh_doall(ENGINE_PILE,lh,fn)
#define lh_ENGINE_PILE_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(ENGINE_PILE,lh,fn,arg_type,arg)
#define lh_ENGINE_PILE_error(lh) LHM_lh_error(ENGINE_PILE,lh)
#define lh_ENGINE_PILE_num_items(lh) LHM_lh_num_items(ENGINE_PILE,lh)
#define lh_ENGINE_PILE_down_load(lh) LHM_lh_down_load(ENGINE_PILE,lh)
#define lh_ENGINE_PILE_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(ENGINE_PILE,lh,out)
#define lh_ENGINE_PILE_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(ENGINE_PILE,lh,out)
#define lh_ENGINE_PILE_stats_bio(lh,out) \
  LHM_lh_stats_bio(ENGINE_PILE,lh,out)
#define lh_ENGINE_PILE_free(lh) LHM_lh_free(ENGINE_PILE,lh)

#define lh_ERR_STATE_new() LHM_lh_new(ERR_STATE,err_state)
#define lh_ERR_STATE_insert(lh,inst) LHM_lh_insert(ERR_STATE,lh,inst)
#define lh_ERR_STATE_retrieve(lh,inst) LHM_lh_retrieve(ERR_STATE,lh,inst)
#define lh_ERR_STATE_delete(lh,inst) LHM_lh_delete(ERR_STATE,lh,inst)
```

```
#define lh_ERR_STATE_doall(lh,fn) LHM_lh_doall(ERR_STATE,lh,fn)
#define lh_ERR_STATE_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(ERR_STATE,lh,fn,arg_type,arg)
#define lh_ERR_STATE_error(lh) LHM_lh_error(ERR_STATE,lh)
#define lh_ERR_STATE_num_items(lh) LHM_lh_num_items(ERR_STATE,lh)
#define lh_ERR_STATE_down_load(lh) LHM_lh_down_load(ERR_STATE,lh)
#define lh_ERR_STATE_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(ERR_STATE,lh,out)
#define lh_ERR_STATE_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(ERR_STATE,lh,out)
#define lh_ERR_STATE_stats_bio(lh,out) \
  LHM_lh_stats_bio(ERR_STATE,lh,out)
#define lh_ERR_STATE_free(lh) LHM_lh_free(ERR_STATE,lh)

#define lh_ERR_STRING_DATA_new() LHM_lh_new(ERR_STRING_DATA,err_string_data)
#define lh_ERR_STRING_DATA_insert(lh,inst) LHM_lh_insert(ERR_STRING_DATA,lh,inst
#define lh_ERR_STRING_DATA_retrieve(lh,inst) LHM_lh_retrieve(ERR_STRING_DATA,lh,
#define lh_ERR_STRING_DATA_delete(lh,inst) LHM_lh_delete(ERR_STRING_DATA,lh,inst
#define lh_ERR_STRING_DATA_doall(lh,fn) LHM_lh_doall(ERR_STRING_DATA,lh,fn)
#define lh_ERR_STRING_DATA_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(ERR_STRING_DATA,lh,fn,arg_type,arg)
#define lh_ERR_STRING_DATA_error(lh) LHM_lh_error(ERR_STRING_DATA,lh)
#define lh_ERR_STRING_DATA_num_items(lh) LHM_lh_num_items(ERR_STRING_DATA,lh)
#define lh_ERR_STRING_DATA_down_load(lh) LHM_lh_down_load(ERR_STRING_DATA,lh)
#define lh_ERR_STRING_DATA_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(ERR_STRING_DATA,lh,out)
#define lh_ERR_STRING_DATA_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(ERR_STRING_DATA,lh,out)
#define lh_ERR_STRING_DATA_stats_bio(lh,out) \
  LHM_lh_stats_bio(ERR_STRING_DATA,lh,out)
#define lh_ERR_STRING_DATA_free(lh) LHM_lh_free(ERR_STRING_DATA,lh)

#define lh_EX_CLASS_ITEM_new() LHM_lh_new(EX_CLASS_ITEM,ex_class_item)
#define lh_EX_CLASS_ITEM_insert(lh,inst) LHM_lh_insert(EX_CLASS_ITEM,lh,inst)
#define lh_EX_CLASS_ITEM_retrieve(lh,inst) LHM_lh_retrieve(EX_CLASS_ITEM,lh,inst
#define lh_EX_CLASS_ITEM_delete(lh,inst) LHM_lh_delete(EX_CLASS_ITEM,lh,inst)
#define lh_EX_CLASS_ITEM_doall(lh,fn) LHM_lh_doall(EX_CLASS_ITEM,lh,fn)
#define lh_EX_CLASS_ITEM_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(EX_CLASS_ITEM,lh,fn,arg_type,arg)
#define lh_EX_CLASS_ITEM_error(lh) LHM_lh_error(EX_CLASS_ITEM,lh)
#define lh_EX_CLASS_ITEM_num_items(lh) LHM_lh_num_items(EX_CLASS_ITEM,lh)
#define lh_EX_CLASS_ITEM_down_load(lh) LHM_lh_down_load(EX_CLASS_ITEM,lh)
#define lh_EX_CLASS_ITEM_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(EX_CLASS_ITEM,lh,out)
#define lh_EX_CLASS_ITEM_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(EX_CLASS_ITEM,lh,out)
#define lh_EX_CLASS_ITEM_stats_bio(lh,out) \
  LHM_lh_stats_bio(EX_CLASS_ITEM,lh,out)
#define lh_EX_CLASS_ITEM_free(lh) LHM_lh_free(EX_CLASS_ITEM,lh)

#define lh_FUNCTION_new() LHM_lh_new(FUNCTION,function)
#define lh_FUNCTION_insert(lh,inst) LHM_lh_insert(FUNCTION,lh,inst)
#define lh_FUNCTION_retrieve(lh,inst) LHM_lh_retrieve(FUNCTION,lh,inst)
#define lh_FUNCTION_delete(lh,inst) LHM_lh_delete(FUNCTION,lh,inst)
#define lh_FUNCTION_doall(lh,fn) LHM_lh_doall(FUNCTION,lh,fn)
#define lh_FUNCTION_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(FUNCTION,lh,fn,arg_type,arg)
#define lh_FUNCTION_error(lh) LHM_lh_error(FUNCTION,lh)
#define lh_FUNCTION_num_items(lh) LHM_lh_num_items(FUNCTION,lh)
#define lh_FUNCTION_down_load(lh) LHM_lh_down_load(FUNCTION,lh)
#define lh_FUNCTION_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(FUNCTION,lh,out)
#define lh_FUNCTION_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(FUNCTION,lh,out)
#define lh_FUNCTION_stats_bio(lh,out) \
  LHM_lh_stats_bio(FUNCTION,lh,out)
```



```
#define lh_FUNCTION_free(lh) LHM_lh_free(FUNCTION,lh)

#define lh_MEM_new() LHM_lh_new(MEM,mem)
#define lh_MEM_insert(lh,inst) LHM_lh_insert(MEM,lh,inst)
#define lh_MEM_retrieve(lh,inst) LHM_lh_retrieve(MEM,lh,inst)
#define lh_MEM_delete(lh,inst) LHM_lh_delete(MEM,lh,inst)
#define lh_MEM_doall(lh,fn) LHM_lh_doall(MEM,lh,fn)
#define lh_MEM_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(MEM,lh,fn,arg_type,arg)
#define lh_MEM_error(lh) LHM_lh_error(MEM,lh)
#define lh_MEM_num_items(lh) LHM_lh_num_items(MEM,lh)
#define lh_MEM_down_load(lh) LHM_lh_down_load(MEM,lh)
#define lh_MEM_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(MEM,lh,out)
#define lh_MEM_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(MEM,lh,out)
#define lh_MEM_stats_bio(lh,out) \
  LHM_lh_stats_bio(MEM,lh,out)
#define lh_MEM_free(lh) LHM_lh_free(MEM,lh)

#define lh_OBJ_NAME_new() LHM_lh_new(OBJ_NAME,obj_name)
#define lh_OBJ_NAME_insert(lh,inst) LHM_lh_insert(OBJ_NAME,lh,inst)
#define lh_OBJ_NAME_retrieve(lh,inst) LHM_lh_retrieve(OBJ_NAME,lh,inst)
#define lh_OBJ_NAME_delete(lh,inst) LHM_lh_delete(OBJ_NAME,lh,inst)
#define lh_OBJ_NAME_doall(lh,fn) LHM_lh_doall(OBJ_NAME,lh,fn)
#define lh_OBJ_NAME_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(OBJ_NAME,lh,fn,arg_type,arg)
#define lh_OBJ_NAME_error(lh) LHM_lh_error(OBJ_NAME,lh)
#define lh_OBJ_NAME_num_items(lh) LHM_lh_num_items(OBJ_NAME,lh)
#define lh_OBJ_NAME_down_load(lh) LHM_lh_down_load(OBJ_NAME,lh)
#define lh_OBJ_NAME_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(OBJ_NAME,lh,out)
#define lh_OBJ_NAME_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(OBJ_NAME,lh,out)
#define lh_OBJ_NAME_stats_bio(lh,out) \
  LHM_lh_stats_bio(OBJ_NAME,lh,out)
#define lh_OBJ_NAME_free(lh) LHM_lh_free(OBJ_NAME,lh)

#define lh_OPENSSL_CSTRING_new() LHM_lh_new(OPENSSL_CSTRING,openssl_cstring)
#define lh_OPENSSL_CSTRING_insert(lh,inst) LHM_lh_insert(OPENSSL_CSTRING,lh,inst
#define lh_OPENSSL_CSTRING_retrieve(lh,inst) LHM_lh_retrieve(OPENSSL_CSTRING,lh,
#define lh_OPENSSL_CSTRING_delete(lh,inst) LHM_lh_delete(OPENSSL_CSTRING,lh,inst
#define lh_OPENSSL_CSTRING_doall(lh,fn) LHM_lh_doall(OPENSSL_CSTRING,lh,fn)
#define lh_OPENSSL_CSTRING_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(OPENSSL_CSTRING,lh,fn,arg_type,arg)
#define lh_OPENSSL_CSTRING_error(lh) LHM_lh_error(OPENSSL_CSTRING,lh)
#define lh_OPENSSL_CSTRING_num_items(lh) LHM_lh_num_items(OPENSSL_CSTRING,lh)
#define lh_OPENSSL_CSTRING_down_load(lh) LHM_lh_down_load(OPENSSL_CSTRING,lh)
#define lh_OPENSSL_CSTRING_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(OPENSSL_CSTRING,lh,out)
#define lh_OPENSSL_CSTRING_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(OPENSSL_CSTRING,lh,out)
#define lh_OPENSSL_CSTRING_stats_bio(lh,out) \
  LHM_lh_stats_bio(OPENSSL_CSTRING,lh,out)
#define lh_OPENSSL_CSTRING_free(lh) LHM_lh_free(OPENSSL_CSTRING,lh)

#define lh_OPENSSL_STRING_new() LHM_lh_new(OPENSSL_STRING,openssl_string)
#define lh_OPENSSL_STRING_insert(lh,inst) LHM_lh_insert(OPENSSL_STRING,lh,inst)
#define lh_OPENSSL_STRING_retrieve(lh,inst) LHM_lh_retrieve(OPENSSL_STRING,lh,in
#define lh_OPENSSL_STRING_delete(lh,inst) LHM_lh_delete(OPENSSL_STRING,lh,inst)
#define lh_OPENSSL_STRING_doall(lh,fn) LHM_lh_doall(OPENSSL_STRING,lh,fn)
#define lh_OPENSSL_STRING_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(OPENSSL_STRING,lh,fn,arg_type,arg)
#define lh_OPENSSL_STRING_error(lh) LHM_lh_error(OPENSSL_STRING,lh)
#define lh_OPENSSL_STRING_num_items(lh) LHM_lh_num_items(OPENSSL_STRING,lh)
#define lh_OPENSSL_STRING_down_load(lh) LHM_lh_down_load(OPENSSL_STRING,lh)
```

```
#define lh_OPENSSL_STRING_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(OPENSSL_STRING,lh,out)
#define lh_OPENSSL_STRING_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(OPENSSL_STRING,lh,out)
#define lh_OPENSSL_STRING_stats_bio(lh,out) \
  LHM_lh_stats_bio(OPENSSL_STRING,lh,out)
#define lh_OPENSSL_STRING_free(lh) LHM_lh_free(OPENSSL_STRING,lh)

#define lh_SSL_SESSION_new() LHM_lh_new(SSL_SESSION,ssl_session)
#define lh_SSL_SESSION_insert(lh,inst) LHM_lh_insert(SSL_SESSION,lh,inst)
#define lh_SSL_SESSION_retrieve(lh,inst) LHM_lh_retrieve(SSL_SESSION,lh,inst)
#define lh_SSL_SESSION_delete(lh,inst) LHM_lh_delete(SSL_SESSION,lh,inst)
#define lh_SSL_SESSION_doall(lh,fn) LHM_lh_doall(SSL_SESSION,lh,fn)
#define lh_SSL_SESSION_doall_arg(lh,fn,arg_type,arg) \
  LHM_lh_doall_arg(SSL_SESSION,lh,fn,arg_type,arg)
#define lh_SSL_SESSION_error(lh) LHM_lh_error(SSL_SESSION,lh)
#define lh_SSL_SESSION_num_items(lh) LHM_lh_num_items(SSL_SESSION,lh)
#define lh_SSL_SESSION_down_load(lh) LHM_lh_down_load(SSL_SESSION,lh)
#define lh_SSL_SESSION_node_stats_bio(lh,out) \
  LHM_lh_node_stats_bio(SSL_SESSION,lh,out)
#define lh_SSL_SESSION_node_usage_stats_bio(lh,out) \
  LHM_lh_node_usage_stats_bio(SSL_SESSION,lh,out)
#define lh_SSL_SESSION_stats_bio(lh,out) \
  LHM_lh_stats_bio(SSL_SESSION,lh,out)
#define lh_SSL_SESSION_free(lh) LHM_lh_free(SSL_SESSION,lh)
/* End of util/mkstack.pl block, you may now edit :-) */

#endif /* !defined HEADER_SAFESTACK_H */
#endif /* ! codereview */
```

```
**********************************************************
    7630 Wed Aug 13 19:51:48 2014
new/usr/src/lib/openssl/include/openssl/sha.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_SHA_H
#define HEADER_SHA_H
```



```
#include <openssl/e_os2.h>
#include <stddef.h>

#ifdef  __cplusplus
extern "C" {
#endif

#if defined(OPENSSL_NO_SHA) || (defined(OPENSSL_NO_SHA0) && defined(OPENSSL_NO_S
#error SHA is disabled.
#endif

#if defined(OPENSSL_FIPS)
#define FIPS_SHA_SIZE_T size_t
#endif

/*
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 * ! SHA_LONG has to be at least 32 bits wide. If it's wider, then !
 * ! SHA_LONG_LOG2 has to be defined along.                        !
 * !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 */

#if defined(__LP32__)
#define SHA_LONG unsigned long
#elif defined(OPENSSL_SYS_CRAY) || defined(__ILP64__)
#define SHA_LONG unsigned long
#define SHA_LONG_LOG2 3
#else
#define SHA_LONG unsigned int
#endif

#define SHA_LBLOCK      16
#define SHA_CBLOCK      (SHA_LBLOCK*4)  /* SHA treats input data as a
                                         * contiguous array of 32 bit
                                         * wide big-endian values. */
#define SHA_LAST_BLOCK  (SHA_CBLOCK-8)
#define SHA_DIGEST_LENGTH 20

typedef struct SHAstate_st
        {
        SHA_LONG h0,h1,h2,h3,h4;
        SHA_LONG Nl,Nh;
        SHA_LONG data[SHA_LBLOCK];
        unsigned int num;
        } SHA_CTX;

#ifndef OPENSSL_NO_SHA0
#ifdef OPENSSL_FIPS
int private_SHA_Init(SHA_CTX *c);
#endif
int SHA_Init(SHA_CTX *c);
int SHA_Update(SHA_CTX *c, const void *data, size_t len);
int SHA_Final(unsigned char *md, SHA_CTX *c);
unsigned char *SHA(const unsigned char *d, size_t n, unsigned char *md);
void SHA_Transform(SHA_CTX *c, const unsigned char *data);
#endif
#ifndef OPENSSL_NO_SHA1
#ifdef OPENSSL_FIPS
int private_SHA1_Init(SHA_CTX *c);
#endif
int SHA1_Init(SHA_CTX *c);
int SHA1_Update(SHA_CTX *c, const void *data, size_t len);
int SHA1_Final(unsigned char *md, SHA_CTX *c);
unsigned char *SHA1(const unsigned char *d, size_t n, unsigned char *md);
void SHA1_Transform(SHA_CTX *c, const unsigned char *data);
#endif
```

```
#define SHA256_CBLOCK   (SHA_LBLOCK*4)  /* SHA-256 treats input data as a
                                         * contiguous array of 32 bit
                                         * wide big-endian values. */
#define SHA224_DIGEST_LENGTH    28
#define SHA256_DIGEST_LENGTH    32

typedef struct SHA256state_st
        {
        SHA_LONG h[8];
        SHA_LONG Nl,Nh;
        SHA_LONG data[SHA_LBLOCK];
        unsigned int num,md_len;
        } SHA256_CTX;

#ifndef OPENSSL_NO_SHA256
#ifdef OPENSSL_FIPS
int private_SHA224_Init(SHA256_CTX *c);
int private_SHA256_Init(SHA256_CTX *c);
#endif
int SHA224_Init(SHA256_CTX *c);
int SHA224_Update(SHA256_CTX *c, const void *data, size_t len);
int SHA224_Final(unsigned char *md, SHA256_CTX *c);
unsigned char *SHA224(const unsigned char *d, size_t n,unsigned char *md);
int SHA256_Init(SHA256_CTX *c);
int SHA256_Update(SHA256_CTX *c, const void *data, size_t len);
int SHA256_Final(unsigned char *md, SHA256_CTX *c);
unsigned char *SHA256(const unsigned char *d, size_t n,unsigned char *md);
void SHA256_Transform(SHA256_CTX *c, const unsigned char *data);
#endif

#define SHA384_DIGEST_LENGTH    48
#define SHA512_DIGEST_LENGTH    64

#ifndef OPENSSL_NO_SHA512
/*
 * Unlike 32-bit digest algorithms, SHA-512 *relies* on SHA_LONG64
 * being exactly 64-bit wide. See Implementation Notes in sha512.c
 * for further details.
 */
#define SHA512_CBLOCK   (SHA_LBLOCK*8)  /* SHA-512 treats input data as a
                                         * contiguous array of 64 bit
                                         * wide big-endian values. */
#if (defined(_WIN32) || defined(_WIN64)) && !defined(__MINGW32__)
#define SHA_LONG64 unsigned __int64
#define U64(C)     C##UI64
#elif defined(__arch64__)
#define SHA_LONG64 unsigned long
#define U64(C)     C##UL
#else
#define SHA_LONG64 unsigned long long
#define U64(C)     C##ULL
#endif

typedef struct SHA512state_st
        {
        SHA_LONG64 h[8];
        SHA_LONG64 Nl,Nh;
        union {
                SHA_LONG64      d[SHA_LBLOCK];
                unsigned char   p[SHA512_CBLOCK];
        } u;
        unsigned int num,md_len;
        } SHA512_CTX;
#endif
```



```
#ifndef OPENSSL_NO_SHA512
#ifdef OPENSSL_FIPS
int private_SHA384_Init(SHA512_CTX *c);
int private_SHA512_Init(SHA512_CTX *c);
#endif
int SHA384_Init(SHA512_CTX *c);
int SHA384_Update(SHA512_CTX *c, const void *data, size_t len);
int SHA384_Final(unsigned char *md, SHA512_CTX *c);
unsigned char *SHA384(const unsigned char *d, size_t n,unsigned char *md);
int SHA512_Init(SHA512_CTX *c);
int SHA512_Update(SHA512_CTX *c, const void *data, size_t len);
int SHA512_Final(unsigned char *md, SHA512_CTX *c);
unsigned char *SHA512(const unsigned char *d, size_t n,unsigned char *md);
void SHA512_Transform(SHA512_CTX *c, const unsigned char *data);
#endif

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    5346 Wed Aug 13 19:51:48 2014
new/usr/src/lib/openssl/include/openssl/srp.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/srp/srp.h */
/* Written by Christophe Renou (christophe.renou@edelweb.fr) with
 * the precious help of Peter Sylvester (peter.sylvester@edelweb.fr)
 * for the EdelKey project and contributed to the OpenSSL project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#ifndef __SRP_H__
#define __SRP_H__
```



```
#ifndef OPENSSL_NO_SRP

#include <stdio.h>
#include <string.h>

#ifdef  __cplusplus
extern "C" {
#endif

#include <openssl/safestack.h>
#include <openssl/bn.h>
#include <openssl/crypto.h>

typedef struct SRP_gN_cache_st
        {
        char *b64_bn;
        BIGNUM *bn;
        } SRP_gN_cache;


DECLARE_STACK_OF(SRP_gN_cache)

typedef struct SRP_user_pwd_st
        {
        char *id;
        BIGNUM *s;
        BIGNUM *v;
        const BIGNUM *g;
        const BIGNUM *N;
        char *info;
        } SRP_user_pwd;

DECLARE_STACK_OF(SRP_user_pwd)

typedef struct SRP_VBASE_st
        {
        STACK_OF(SRP_user_pwd) *users_pwd;
        STACK_OF(SRP_gN_cache) *gN_cache;
/* to simulate a user */
        char *seed_key;
        BIGNUM *default_g;
        BIGNUM *default_N;
        } SRP_VBASE;


/*Structure interne pour retenir les couples N et g*/
typedef struct SRP_gN_st
        {
        char *id;
        BIGNUM *g;
        BIGNUM *N;
        } SRP_gN;

DECLARE_STACK_OF(SRP_gN)

SRP_VBASE *SRP_VBASE_new(char *seed_key);
int SRP_VBASE_free(SRP_VBASE *vb);
int SRP_VBASE_init(SRP_VBASE *vb, char * verifier_file);
SRP_user_pwd *SRP_VBASE_get_by_user(SRP_VBASE *vb, char *username);
char *SRP_create_verifier(const char *user, const char *pass, char **salt,
                          char **verifier, const char *N, const char *g);
int SRP_create_verifier_BN(const char *user, const char *pass, BIGNUM **salt, BI


#define SRP_NO_ERROR 0
#define SRP_ERR_VBASE_INCOMPLETE_FILE 1
```

```
#define SRP_ERR_VBASE_BN_LIB 2
#define SRP_ERR_OPEN_FILE 3
#define SRP_ERR_MEMORY 4

#define DB_srptype      0
#define DB_srpverifier  1
#define DB_srpsalt      2
#define DB_srpid        3
#define DB_srpgN        4
#define DB_srpinfo      5
#undef  DB_NUMBER
#define DB_NUMBER       6

#define DB_SRP_INDEX    'I'
#define DB_SRP_VALID    'V'
#define DB_SRP_REVOKED  'R'
#define DB_SRP_MODIF    'v'


/* see srp.c */
char * SRP_check_known_gN_param(BIGNUM* g, BIGNUM* N);
SRP_gN *SRP_get_default_gN(const char * id) ;

/* server side .... */
BIGNUM *SRP_Calc_server_key(BIGNUM *A, BIGNUM *v, BIGNUM *u, BIGNUM *b, BIGNUM *
BIGNUM *SRP_Calc_B(BIGNUM *b, BIGNUM *N, BIGNUM *g, BIGNUM *v);
int SRP_Verify_A_mod_N(BIGNUM *A, BIGNUM *N);
BIGNUM *SRP_Calc_u(BIGNUM *A, BIGNUM *B, BIGNUM *N) ;



/* client side .... */
BIGNUM *SRP_Calc_x(BIGNUM *s, const char *user, const char *pass);
BIGNUM *SRP_Calc_A(BIGNUM *a, BIGNUM *N, BIGNUM *g);
BIGNUM *SRP_Calc_client_key(BIGNUM *N, BIGNUM *B, BIGNUM *g, BIGNUM *x, BIGNUM *
int SRP_Verify_B_mod_N(BIGNUM *B, BIGNUM *N);

#define SRP_MINIMAL_N 1024

#ifdef  __cplusplus
}
#endif

#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    6631 Wed Aug 13 19:51:48 2014
new/usr/src/lib/openssl/include/openssl/srtp.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/tls1.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/*
  DTLS code by Eric Rescorla <ekr@rtfm.com>

  Copyright (C) 2006, Network Resonance, Inc.
  Copyright (C) 2011, RTFM, Inc.
*/

#ifndef HEADER_D1_SRTP_H
#define HEADER_D1_SRTP_H

#ifdef  __cplusplus
extern "C" {
#endif


#define SRTP_AES128_CM_SHA1_80 0x0001
#define SRTP_AES128_CM_SHA1_32 0x0002
```

```
#define SRTP_AES128_F8_SHA1_80 0x0003
#define SRTP_AES128_F8_SHA1_32 0x0004
#define SRTP_NULL_SHA1_80      0x0005
#define SRTP_NULL_SHA1_32      0x0006

int SSL_CTX_set_tlsext_use_srtp(SSL_CTX *ctx, const char *profiles);
int SSL_set_tlsext_use_srtp(SSL *ctx, const char *profiles);
SRTP_PROTECTION_PROFILE *SSL_get_selected_srtp_profile(SSL *s);

STACK_OF(SRTP_PROTECTION_PROFILE) *SSL_get_srtp_profiles(SSL *ssl);
SRTP_PROTECTION_PROFILE *SSL_get_selected_srtp_profile(SSL *s);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
  103976 Wed Aug 13 19:51:49 2014
new/usr/src/lib/openssl/include/openssl/ssl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/ssl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECC cipher suite support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
```

```
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#ifndef HEADER_SSL_H
#define HEADER_SSL_H

#include <openssl/e_os2.h>

#ifndef OPENSSL_NO_COMP
#include <openssl/comp.h>
#endif
#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#ifndef OPENSSL_NO_DEPRECATED
#ifndef OPENSSL_NO_X509
#include <openssl/x509.h>
#endif
#include <openssl/crypto.h>
#include <openssl/lhash.h>
#include <openssl/buffer.h>
#endif
#include <openssl/pem.h>
#include <openssl/hmac.h>

#include <openssl/kssl.h>
#include <openssl/safestack.h>
#include <openssl/symhacks.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* SSLeay version number for ASN.1 encoding of the session information */
/* Version 0 - initial version
 * Version 1 - added the optional peer certificate
 */
#define SSL_SESSION_ASN1_VERSION 0x0001

/* text strings for the ciphers */
#define SSL_TXT_NULL_WITH_MD5           SSL2_TXT_NULL_WITH_MD5
#define SSL_TXT_RC4_128_WITH_MD5        SSL2_TXT_RC4_128_WITH_MD5
#define SSL_TXT_RC4_128_EXPORT40_WITH_MD5 SSL2_TXT_RC4_128_EXPORT40_WITH_MD5
#define SSL_TXT_RC2_128_CBC_WITH_MD5    SSL2_TXT_RC2_128_CBC_WITH_MD5
#define SSL_TXT_RC2_128_CBC_EXPORT40_WITH_MD5 SSL2_TXT_RC2_128_CBC_EXPORT40_WITH
#define SSL_TXT_IDEA_128_CBC_WITH_MD5   SSL2_TXT_IDEA_128_CBC_WITH_MD5
#define SSL_TXT_DES_64_CBC_WITH_MD5     SSL2_TXT_DES_64_CBC_WITH_MD5
#define SSL_TXT_DES_64_CBC_WITH_SHA     SSL2_TXT_DES_64_CBC_WITH_SHA
#define SSL_TXT_DES_192_EDE3_CBC_WITH_MD5 SSL2_TXT_DES_192_EDE3_CBC_WITH_MD5
#define SSL_TXT_DES_192_EDE3_CBC_WITH_SHA SSL2_TXT_DES_192_EDE3_CBC_WITH_SHA

/*    VRS Additional Kerberos5 entries
 */
#define SSL_TXT_KRB5_DES_64_CBC_SHA   SSL3_TXT_KRB5_DES_64_CBC_SHA
```



```
#define SSL_TXT_KRB5_DES_192_CBC3_SHA SSL3_TXT_KRB5_DES_192_CBC3_SHA
#define SSL_TXT_KRB5_RC4_128_SHA      SSL3_TXT_KRB5_RC4_128_SHA
#define SSL_TXT_KRB5_IDEA_128_CBC_SHA SSL3_TXT_KRB5_IDEA_128_CBC_SHA
#define SSL_TXT_KRB5_DES_64_CBC_MD5   SSL3_TXT_KRB5_DES_64_CBC_MD5
#define SSL_TXT_KRB5_DES_192_CBC3_MD5 SSL3_TXT_KRB5_DES_192_CBC3_MD5
#define SSL_TXT_KRB5_RC4_128_MD5      SSL3_TXT_KRB5_RC4_128_MD5
#define SSL_TXT_KRB5_IDEA_128_CBC_MD5 SSL3_TXT_KRB5_IDEA_128_CBC_MD5

#define SSL_TXT_KRB5_DES_40_CBC_SHA   SSL3_TXT_KRB5_DES_40_CBC_SHA
#define SSL_TXT_KRB5_RC2_40_CBC_SHA   SSL3_TXT_KRB5_RC2_40_CBC_SHA
#define SSL_TXT_KRB5_RC4_40_SHA       SSL3_TXT_KRB5_RC4_40_SHA
#define SSL_TXT_KRB5_DES_40_CBC_MD5   SSL3_TXT_KRB5_DES_40_CBC_MD5
#define SSL_TXT_KRB5_RC2_40_CBC_MD5   SSL3_TXT_KRB5_RC2_40_CBC_MD5
#define SSL_TXT_KRB5_RC4_40_MD5       SSL3_TXT_KRB5_RC4_40_MD5

#define SSL_TXT_KRB5_DES_40_CBC_SHA   SSL3_TXT_KRB5_DES_40_CBC_SHA
#define SSL_TXT_KRB5_DES_40_CBC_MD5   SSL3_TXT_KRB5_DES_40_CBC_MD5
#define SSL_TXT_KRB5_DES_64_CBC_SHA   SSL3_TXT_KRB5_DES_64_CBC_SHA
#define SSL_TXT_KRB5_DES_64_CBC_MD5   SSL3_TXT_KRB5_DES_64_CBC_MD5
#define SSL_TXT_KRB5_DES_192_CBC3_SHA SSL3_TXT_KRB5_DES_192_CBC3_SHA
#define SSL_TXT_KRB5_DES_192_CBC3_MD5 SSL3_TXT_KRB5_DES_192_CBC3_MD5
#define SSL_MAX_KRB5_PRINCIPAL_LENGTH  256

#define SSL_MAX_SSL_SESSION_ID_LENGTH           32
#define SSL_MAX_SID_CTX_LENGTH                  32

#define SSL_MIN_RSA_MODULUS_LENGTH_IN_BYTES     (512/8)
#define SSL_MAX_KEY_ARG_LENGTH                  8
#define SSL_MAX_MASTER_KEY_LENGTH               48


/* These are used to specify which ciphers to use and not to use */

#define SSL_TXT_EXP40           "EXPORT40"
#define SSL_TXT_EXP56           "EXPORT56"
#define SSL_TXT_LOW             "LOW"
#define SSL_TXT_MEDIUM          "MEDIUM"
#define SSL_TXT_HIGH            "HIGH"
#define SSL_TXT_FIPS            "FIPS"

#define SSL_TXT_kFZA            "kFZA" /* unused! */
#define SSL_TXT_aFZA            "aFZA" /* unused! */
#define SSL_TXT_eFZA            "eFZA" /* unused! */
#define SSL_TXT_FZA             "FZA"  /* unused! */

#define SSL_TXT_aNULL           "aNULL"
#define SSL_TXT_eNULL           "eNULL"
#define SSL_TXT_NULL            "NULL"

#define SSL_TXT_kRSA            "kRSA"
#define SSL_TXT_kDHr            "kDHr" /* no such ciphersuites supported! */
#define SSL_TXT_kDHd            "kDHd" /* no such ciphersuites supported! */
#define SSL_TXT_kDH             "kDH"  /* no such ciphersuites supported! */
#define SSL_TXT_kEDH            "kEDH"
#define SSL_TXT_kKRB5           "kKRB5"
#define SSL_TXT_kECDHr          "kECDHr"
#define SSL_TXT_kECDHe          "kECDHe"
#define SSL_TXT_kECDH           "kECDH"
#define SSL_TXT_kEECDH          "kEECDH"
#define SSL_TXT_kPSK            "kPSK"
#define SSL_TXT_kGOST           "kGOST"
#define SSL_TXT_kSRP            "kSRP"

#define SSL_TXT_aRSA            "aRSA"
#define SSL_TXT_aDSS            "aDSS"
#define SSL_TXT_aDH             "aDH" /* no such ciphersuites supported! */
```

```
260 #define SSL_TXT_aECDH           "aECDH"
261 #define SSL_TXT_aKRB5           "aKRB5"
262 #define SSL_TXT_aECDSA          "aECDSA"
263 #define SSL_TXT_aPSK            "aPSK"
264 #define SSL_TXT_aGOST94 "aGOST94"
265 #define SSL_TXT_aGOST01 "aGOST01"
266 #define SSL_TXT_aGOST  "aGOST"
267 #define SSL_TXT_aSRP            "aSRP"

269 #define SSL_TXT_DSS             "DSS"
270 #define SSL_TXT_DH              "DH"
271 #define SSL_TXT_EDH             "EDH" /* same as "kEDH:-ADH" */
272 #define SSL_TXT_ADH             "ADH"
273 #define SSL_TXT_RSA             "RSA"
274 #define SSL_TXT_ECDH            "ECDH"
275 #define SSL_TXT_EECDH           "EECDH" /* same as "kEECDH:-AECDH" */
276 #define SSL_TXT_AECDH           "AECDH"
277 #define SSL_TXT_ECDSA           "ECDSA"
278 #define SSL_TXT_KRB5            "KRB5"
279 #define SSL_TXT_PSK             "PSK"
280 #define SSL_TXT_SRP             "SRP"

282 #define SSL_TXT_DES             "DES"
283 #define SSL_TXT_3DES            "3DES"
284 #define SSL_TXT_RC4             "RC4"
285 #define SSL_TXT_RC2             "RC2"
286 #define SSL_TXT_IDEA            "IDEA"
287 #define SSL_TXT_SEED            "SEED"
288 #define SSL_TXT_AES128          "AES128"
289 #define SSL_TXT_AES256          "AES256"
290 #define SSL_TXT_AES             "AES"
291 #define SSL_TXT_AES_GCM         "AESGCM"
292 #define SSL_TXT_CAMELLIA128     "CAMELLIA128"
293 #define SSL_TXT_CAMELLIA256     "CAMELLIA256"
294 #define SSL_TXT_CAMELLIA        "CAMELLIA"

296 #define SSL_TXT_MD5             "MD5"
297 #define SSL_TXT_SHA1            "SHA1"
298 #define SSL_TXT_SHA             "SHA" /* same as "SHA1" */
299 #define SSL_TXT_GOST94          "GOST94"
300 #define SSL_TXT_GOST89MAC               "GOST89MAC"
301 #define SSL_TXT_SHA256          "SHA256"
302 #define SSL_TXT_SHA384          "SHA384"

304 #define SSL_TXT_SSLV2           "SSLv2"
305 #define SSL_TXT_SSLV3           "SSLv3"
306 #define SSL_TXT_TLSV1           "TLSv1"
307 #define SSL_TXT_TLSV1_1         "TLSv1.1"
308 #define SSL_TXT_TLSV1_2         "TLSv1.2"

310 #define SSL_TXT_EXP             "EXP"
311 #define SSL_TXT_EXPORT          "EXPORT"

313 #define SSL_TXT_ALL             "ALL"

315 /*
316  * COMPLEMENTOF* definitions. These identifiers are used to (de-select)
317  * ciphers normally not being used.
318  * Example: "RC4" will activate all ciphers using RC4 including ciphers
319  * without authentication, which would normally disabled by DEFAULT (due
320  * the "!ADH" being part of default). Therefore "RC4:!COMPLEMENTOFDEFAULT"
321  * will make sure that it is also disabled in the specific selection.
322  * COMPLEMENTOF* identifiers are portable between version, as adjustments
323  * to the default cipher setup will also be included here.
324  *
325  * COMPLEMENTOFDEFAULT does not experience the same special treatment that
```



```
326  * DEFAULT gets, as only selection is being done and no sorting as needed
327  * for DEFAULT.
328  */
329 #define SSL_TXT_CMPALL          "COMPLEMENTOFALL"
330 #define SSL_TXT_CMPDEF          "COMPLEMENTOFDEFAULT"

332 /* The following cipher list is used by default.
333  * It also is substituted when an application-defined cipher list string
334  * starts with 'DEFAULT'. */
335 #define SSL_DEFAULT_CIPHER_LIST "ALL:!aNULL:!eNULL:!SSLv2"
336 /* As of OpenSSL 1.0.0, ssl_create_cipher_list() in ssl/ssl_ciph.c always
337  * starts with a reasonable order, and all we have to do for DEFAULT is
338  * throwing out anonymous and unencrypted ciphersuites!
339  * (The latter are not actually enabled by ALL, but "ALL:RSA" would enable
340  * some of them.)
341  */

343 /* Used in SSL_set_shutdown()/SSL_get_shutdown(); */
344 #define SSL_SENT_SHUTDOWN       1
345 #define SSL_RECEIVED_SHUTDOWN   2

347 #ifdef __cplusplus
348 }
349 #endif

351 #ifdef  __cplusplus
352 extern "C" {
353 #endif

355 #if (defined(OPENSSL_NO_RSA) || defined(OPENSSL_NO_MD5)) && !defined(OPENSSL_NO_
356 #define OPENSSL_NO_SSL2
357 #endif

359 #define SSL_FILETYPE_ASN1       X509_FILETYPE_ASN1
360 #define SSL_FILETYPE_PEM        X509_FILETYPE_PEM

362 /* This is needed to stop compilers complaining about the
363  * 'struct ssl_st *' function parameters used to prototype callbacks
364  * in SSL_CTX. */
365 typedef struct ssl_st *ssl_crock_st;
366 typedef struct tls_session_ticket_ext_st TLS_SESSION_TICKET_EXT;
367 typedef struct ssl_method_st SSL_METHOD;
368 typedef struct ssl_cipher_st SSL_CIPHER;
369 typedef struct ssl_session_st SSL_SESSION;

371 DECLARE_STACK_OF(SSL_CIPHER)

373 /* SRTP protection profiles for use with the use_srtp extension (RFC 5764)*/
374 typedef struct srtp_protection_profile_st
375         {
376         const char *name;
377         unsigned long id;
378         } SRTP_PROTECTION_PROFILE;

380 DECLARE_STACK_OF(SRTP_PROTECTION_PROFILE)

382 typedef int (*tls_session_ticket_ext_cb_fn)(SSL *s, const unsigned char *data, i
383 typedef int (*tls_session_secret_cb_fn)(SSL *s, void *secret, int *secret_len, S


386 #ifndef OPENSSL_NO_SSL_INTERN

388 /* used to hold info on the particular ciphers used */
389 struct ssl_cipher_st
390         {
391         int valid;
```

```
392         const char *name;               /* text name */
393         unsigned long id;               /* id, 4 bytes, first is version */

395         /* changed in 0.9.9: these four used to be portions of a single value 'a
396         unsigned long algorithm_mkey;   /* key exchange algorithm */
397         unsigned long algorithm_auth;   /* server authentication */
398         unsigned long algorithm_enc;    /* symmetric encryption */
399         unsigned long algorithm_mac;    /* symmetric authentication */
400         unsigned long algorithm_ssl;    /* (major) protocol version */

402         unsigned long algo_strength;    /* strength and export flags */
403         unsigned long algorithm2;       /* Extra flags */
404         int strength_bits;              /* Number of bits really used */
405         int alg_bits;                   /* Number of bits for algorithm */
406         };


409 /* Used to hold functions for SSLv2 or SSLv3/TLSv1 functions */
410 struct ssl_method_st
411         {
412         int version;
413         int (*ssl_new)(SSL *s);
414         void (*ssl_clear)(SSL *s);
415         void (*ssl_free)(SSL *s);
416         int (*ssl_accept)(SSL *s);
417         int (*ssl_connect)(SSL *s);
418         int (*ssl_read)(SSL *s,void *buf,int len);
419         int (*ssl_peek)(SSL *s,void *buf,int len);
420         int (*ssl_write)(SSL *s,const void *buf,int len);
421         int (*ssl_shutdown)(SSL *s);
422         int (*ssl_renegotiate)(SSL *s);
423         int (*ssl_renegotiate_check)(SSL *s);
424         long (*ssl_get_message)(SSL *s, int st1, int stn, int mt, long
425                 max, int *ok);
426         int (*ssl_read_bytes)(SSL *s, int type, unsigned char *buf, int len,
427                 int peek);
428         int (*ssl_write_bytes)(SSL *s, int type, const void *buf_, int len);
429         int (*ssl_dispatch_alert)(SSL *s);
430         long (*ssl_ctrl)(SSL *s,int cmd,long larg,void *parg);
431         long (*ssl_ctx_ctrl)(SSL_CTX *ctx,int cmd,long larg,void *parg);
432         const SSL_CIPHER *(*get_cipher_by_char)(const unsigned char *ptr);
433         int (*put_cipher_by_char)(const SSL_CIPHER *cipher,unsigned char *ptr);
434         int (*ssl_pending)(const SSL *s);
435         int (*num_ciphers)(void);
436         const SSL_CIPHER *(*get_cipher)(unsigned ncipher);
437         const struct ssl_method_st *(*get_ssl_method)(int version);
438         long (*get_timeout)(void);
439         struct ssl3_enc_method *ssl3_enc; /* Extra SSLv3/TLS stuff */
440         int (*ssl_version)(void);
441         long (*ssl_callback_ctrl)(SSL *s, int cb_id, void (*fp)(void));
442         long (*ssl_ctx_callback_ctrl)(SSL_CTX *s, int cb_id, void (*fp)(void));
443         };

445 /* Lets make this into an ASN.1 type structure as follows
446  * SSL_SESSION_ID ::= SEQUENCE {
447  *      version                 INTEGER,        -- structure version number
448  *      SSLversion              INTEGER,        -- SSL version number
449  *      Cipher                  OCTET STRING,   -- the 3 byte cipher ID
450  *      Session_ID              OCTET STRING,   -- the Session ID
451  *      Master_key              OCTET STRING,   -- the master key
452  *      KRB5_principal          OCTET STRING    -- optional Kerberos principal
453  *      Key_Arg [ 0 ] IMPLICIT  OCTET STRING,   -- the optional Key argument
454  *      Time [ 1 ] EXPLICIT     INTEGER,        -- optional Start Time
455  *      Timeout [ 2 ] EXPLICIT  INTEGER,        -- optional Timeout ins seconds
456  *      Peer [ 3 ] EXPLICIT     X509,           -- optional Peer Certificate
457  *      Session_ID_context [ 4 ] EXPLICIT OCTET STRING,   -- the Session ID cont
```



```
458  *      Verify_result [ 5 ] EXPLICIT INTEGER,   -- X509_V_... code for 'Peer'
459  *      HostName [ 6 ] EXPLICIT OCTET STRING,   -- optional HostName from server
460  *      PSK_identity_hint [ 7 ] EXPLICIT OCTET STRING, -- optional PSK identity
461  *      PSK_identity [ 8 ] EXPLICIT OCTET STRING,  -- optional PSK identity
462  *      Ticket_lifetime_hint [9] EXPLICIT INTEGER, -- server's lifetime hint for
463  *      Ticket [10]             EXPLICIT OCTET STRING, -- session ticket (client
464  *      Compression_meth [11]   EXPLICIT OCTET STRING, -- optional compression m
465  *      SRP_username [ 12 ] EXPLICIT OCTET STRING -- optional SRP username
466  *      }
467  * Look in ssl/ssl_asn1.c for more details
468  * I'm using EXPLICIT tags so I can read the damn things using asn1parse :-).
469  */
470 struct ssl_session_st
471         {
472         int ssl_version;        /* what ssl version session info is
473                                  * being kept in here? */

475         /* only really used in SSLv2 */
476         unsigned int key_arg_length;
477         unsigned char key_arg[SSL_MAX_KEY_ARG_LENGTH];
478         int master_key_length;
479         unsigned char master_key[SSL_MAX_MASTER_KEY_LENGTH];
480         /* session_id - valid? */
481         unsigned int session_id_length;
482         unsigned char session_id[SSL_MAX_SSL_SESSION_ID_LENGTH];
483         /* this is used to determine whether the session is being reused in
484          * the appropriate context. It is up to the application to set this,
485          * via SSL_new */
486         unsigned int sid_ctx_length;
487         unsigned char sid_ctx[SSL_MAX_SID_CTX_LENGTH];

489 #ifndef OPENSSL_NO_KRB5
490         unsigned int krb5_client_princ_len;
491         unsigned char krb5_client_princ[SSL_MAX_KRB5_PRINCIPAL_LENGTH];
492 #endif /* OPENSSL_NO_KRB5 */
493 #ifndef OPENSSL_NO_PSK
494         char *psk_identity_hint;
495         char *psk_identity;
496 #endif
497         /* Used to indicate that session resumption is not allowed.
498          * Applications can also set this bit for a new session via
499          * not_resumable_session_cb to disable session caching and tickets. */
500         int not_resumable;

502         /* The cert is the certificate used to establish this connection */
503         struct sess_cert_st /* SESS_CERT */ *sess_cert;

505         /* This is the cert for the other end.
506          * On clients, it will be the same as sess_cert->peer_key->x509
507          * (the latter is not enough as sess_cert is not retained
508          * in the external representation of sessions, see ssl_asn1.c). */
509         X509 *peer;
510         /* when app_verify_callback accepts a session where the peer's certifica
511          * is not ok, we must remember the error for session reuse: */
512         long verify_result; /* only for servers */

514         int references;
515         long timeout;
516         long time;

518         unsigned int compress_meth;     /* Need to lookup the method */

520         const SSL_CIPHER *cipher;
521         unsigned long cipher_id;        /* when ASN.1 loaded, this
522                                          * needs to be used to load
523                                          * the 'cipher' structure */
```

```
525         STACK_OF(SSL_CIPHER) *ciphers; /* shared ciphers? */

527         CRYPTO_EX_DATA ex_data; /* application specific data */

529         /* These are used to make removal of session-ids more
530          * efficient and to implement a maximum cache size. */
531         struct ssl_session_st *prev,*next;
532 #ifndef OPENSSL_NO_TLSEXT
533         char *tlsext_hostname;
534 #ifndef OPENSSL_NO_EC
535         size_t tlsext_ecpointformatlist_length;
536         unsigned char *tlsext_ecpointformatlist; /* peer's list */
537         size_t tlsext_ellipticcurvelist_length;
538         unsigned char *tlsext_ellipticcurvelist; /* peer's list */
539 #endif /* OPENSSL_NO_EC */
540         /* RFC4507 info */
541         unsigned char *tlsext_tick;     /* Session ticket */
542         size_t tlsext_ticklen;          /* Session ticket length */
543         long tlsext_tick_lifetime_hint; /* Session lifetime hint in seconds */
544 #endif
545 #ifndef OPENSSL_NO_SRP
546         char *srp_username;
547 #endif
548         };

550 #endif

552 #define SSL_OP_MICROSOFT_SESS_ID_BUG                    0x00000001L
553 #define SSL_OP_NETSCAPE_CHALLENGE_BUG                   0x00000002L
554 /* Allow initial connection to servers that don't support RI */
555 #define SSL_OP_LEGACY_SERVER_CONNECT                    0x00000004L
556 #define SSL_OP_NETSCAPE_REUSE_CIPHER_CHANGE_BUG         0x00000008L
557 #define SSL_OP_TLSEXT_PADDING                           0x00000010L
558 #define SSL_OP_MICROSOFT_BIG_SSLV3_BUFFER               0x00000020L
559 #define SSL_OP_SAFARI_ECDHE_ECDSA_BUG                   0x00000040L
560 #define SSL_OP_SSLEAY_080_CLIENT_DH_BUG                 0x00000080L
561 #define SSL_OP_TLS_D5_BUG                               0x00000100L
562 #define SSL_OP_TLS_BLOCK_PADDING_BUG                    0x00000200L

564 /* Hasn't done anything since OpenSSL 0.9.7h, retained for compatibility */
565 #define SSL_OP_MSIE_SSLV2_RSA_PADDING                   0x0
566 /* Refers to ancient SSLREF and SSLv2, retained for compatibility */
567 #define SSL_OP_SSLREF2_REUSE_CERT_TYPE_BUG              0x0

569 /* Disable SSL 3.0/TLS 1.0 CBC vulnerability workaround that was added
570  * in OpenSSL 0.9.6d.  Usually (depending on the application protocol)
571  * the workaround is not needed.  Unfortunately some broken SSL/TLS
572  * implementations cannot handle it at all, which is why we include
573  * it in SSL_OP_ALL. */
574 #define SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS              0x00000800L /* added in

576 /* SSL_OP_ALL: various bug workarounds that should be rather harmless.
577  *             This used to be 0x000FFFFFL before 0.9.7. */
578 #define SSL_OP_ALL                                      0x80000BFFL

580 /* DTLS options */
581 #define SSL_OP_NO_QUERY_MTU                0x00001000L
582 /* Turn on Cookie Exchange (on relevant for servers) */
583 #define SSL_OP_COOKIE_EXCHANGE             0x00002000L
584 /* Don't use RFC4507 ticket extension */
585 #define SSL_OP_NO_TICKET                   0x00004000L
586 /* Use Cisco's "speshul" version of DTLS_BAD_VER (as client)  */
587 #define SSL_OP_CISCO_ANYCONNECT            0x00008000L

589 /* As server, disallow session resumption on renegotiation */
```



```
590 #define SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION   0x00010000L
591 /* Don't use compression even if supported */
592 #define SSL_OP_NO_COMPRESSION                           0x00020000L
593 /* Permit unsafe legacy renegotiation */
594 #define SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION        0x00040000L
595 /* If set, always create a new key when using tmp_ecdh parameters */
596 #define SSL_OP_SINGLE_ECDH_USE                          0x00080000L
597 /* If set, always create a new key when using tmp_dh parameters */
598 #define SSL_OP_SINGLE_DH_USE                            0x00100000L
599 /* Set to always use the tmp_rsa key when doing RSA operations,
600  * even when this violates protocol specs */
601 #define SSL_OP_EPHEMERAL_RSA                            0x00200000L
602 /* Set on servers to choose the cipher according to the server's
603  * preferences */
604 #define SSL_OP_CIPHER_SERVER_PREFERENCE                 0x00400000L
605 /* If set, a server will allow a client to issue a SSLv3.0 version number
606  * as latest version supported in the premaster secret, even when TLSv1.0
607  * (version 3.1) was announced in the client hello. Normally this is
608  * forbidden to prevent version rollback attacks. */
609 #define SSL_OP_TLS_ROLLBACK_BUG                         0x00800000L

611 #define SSL_OP_NO_SSLv2                                 0x01000000L
612 #define SSL_OP_NO_SSLv3                                 0x02000000L
613 #define SSL_OP_NO_TLSv1                                 0x04000000L
614 #define SSL_OP_NO_TLSv1_2                               0x08000000L
615 #define SSL_OP_NO_TLSv1_1                               0x10000000L

617 /* These next two were never actually used for anything since SSLeay
618  * zap so we have some more flags.
619  */
620 /* The next flag deliberately changes the ciphertest, this is a check
621  * for the PKCS#1 attack */
622 #define SSL_OP_PKCS1_CHECK_1                            0x0
623 #define SSL_OP_PKCS1_CHECK_2                            0x0

625 #define SSL_OP_NETSCAPE_CA_DN_BUG                       0x20000000L
626 #define SSL_OP_NETSCAPE_DEMO_CIPHER_CHANGE_BUG          0x40000000L
627 /* Make server add server-hello extension from early version of
628  * cryptopro draft, when GOST ciphersuite is negotiated.
629  * Required for interoperability with CryptoPro CSP 3.x
630  */
631 #define SSL_OP_CRYPTOPRO_TLSEXT_BUG                     0x80000000L

633 /* Allow SSL_write(..., n) to return r with 0 < r < n (i.e. report success
634  * when just a single record has been written): */
635 #define SSL_MODE_ENABLE_PARTIAL_WRITE       0x00000001L
636 /* Make it possible to retry SSL_write() with changed buffer location
637  * (buffer contents must stay the same!); this is not the default to avoid
638  * the misconception that non-blocking SSL_write() behaves like
639  * non-blocking write(): */
640 #define SSL_MODE_ACCEPT_MOVING_WRITE_BUFFER 0x00000002L
641 /* Never bother the application with retries if the transport
642  * is blocking: */
643 #define SSL_MODE_AUTO_RETRY 0x00000004L
644 /* Don't attempt to automatically build certificate chain */
645 #define SSL_MODE_NO_AUTO_CHAIN 0x00000008L
646 /* Save RAM by releasing read and write buffers when they're empty. (SSL3 and
647  * TLS only.)  "Released" buffers are put onto a free-list in the context
648  * or just freed (depending on the context's setting for freelist_max_len). */
649 #define SSL_MODE_RELEASE_BUFFERS 0x00000010L
650 /* Send the current time in the Random fields of the ClientHello and
651  * ServerHello records for compatibility with hypothetical implementations
652  * that require it.
653  */
654 #define SSL_MODE_SEND_CLIENTHELLO_TIME 0x00000020L
655 #define SSL_MODE_SEND_SERVERHELLO_TIME 0x00000040L
```

```
/* Note: SSL[_CTX]_set_{options,mode} use |= op on the previous value,
 * they cannot be used to clear bits. */

#define SSL_CTX_set_options(ctx,op) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_OPTIONS,(op),NULL)
#define SSL_CTX_clear_options(ctx,op) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_CLEAR_OPTIONS,(op),NULL)
#define SSL_CTX_get_options(ctx) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_OPTIONS,0,NULL)
#define SSL_set_options(ssl,op) \
        SSL_ctrl((ssl),SSL_CTRL_OPTIONS,(op),NULL)
#define SSL_clear_options(ssl,op) \
        SSL_ctrl((ssl),SSL_CTRL_CLEAR_OPTIONS,(op),NULL)
#define SSL_get_options(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_OPTIONS,0,NULL)

#define SSL_CTX_set_mode(ctx,op) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_MODE,(op),NULL)
#define SSL_CTX_clear_mode(ctx,op) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_CLEAR_MODE,(op),NULL)
#define SSL_CTX_get_mode(ctx) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_MODE,0,NULL)
#define SSL_clear_mode(ssl,op) \
        SSL_ctrl((ssl),SSL_CTRL_CLEAR_MODE,(op),NULL)
#define SSL_set_mode(ssl,op) \
        SSL_ctrl((ssl),SSL_CTRL_MODE,(op),NULL)
#define SSL_get_mode(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_MODE,0,NULL)
#define SSL_set_mtu(ssl, mtu) \
        SSL_ctrl((ssl),SSL_CTRL_SET_MTU,(mtu),NULL)

#define SSL_get_secure_renegotiation_support(ssl) \
        SSL_ctrl((ssl), SSL_CTRL_GET_RI_SUPPORT, 0, NULL)

#ifndef OPENSSL_NO_HEARTBEATS
#define SSL_heartbeat(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_TLS_EXT_SEND_HEARTBEAT,0,NULL)
#endif

void SSL_CTX_set_msg_callback(SSL_CTX *ctx, void (*cb)(int write_p, int version,
void SSL_set_msg_callback(SSL *ssl, void (*cb)(int write_p, int version, int con
#define SSL_CTX_set_msg_callback_arg(ctx, arg) SSL_CTX_ctrl((ctx), SSL_CTRL_SET_
#define SSL_set_msg_callback_arg(ssl, arg) SSL_ctrl((ssl), SSL_CTRL_SET_MSG_CALL

#ifndef OPENSSL_NO_SRP

#ifndef OPENSSL_NO_SSL_INTERN

typedef struct srp_ctx_st
        {
        /* param for all the callbacks */
        void *SRP_cb_arg;
        /* set client Hello login callback */
        int (*TLS_ext_srp_username_callback)(SSL *, int *, void *);
        /* set SRP N/g param callback for verification */
        int (*SRP_verify_param_callback)(SSL *, void *);
        /* set SRP client passwd callback */
        char *(*SRP_give_srp_client_pwd_callback)(SSL *, void *);

        char *login;
        BIGNUM *N,*g,*s,*B,*A;
        BIGNUM *a,*b,*v;
        char *info;
        int strength;
```



```
        unsigned long srp_Mask;
        } SRP_CTX;

#endif

/* see tls_srp.c */
int SSL_SRP_CTX_init(SSL *s);
int SSL_CTX_SRP_CTX_init(SSL_CTX *ctx);
int SSL_SRP_CTX_free(SSL *ctx);
int SSL_CTX_SRP_CTX_free(SSL_CTX *ctx);
int SSL_srp_server_param_with_username(SSL *s, int *ad);
int SRP_generate_server_master_secret(SSL *s,unsigned char *master_key);
int SRP_Calc_A_param(SSL *s);
int SRP_generate_client_master_secret(SSL *s,unsigned char *master_key);

#endif

#if defined(OPENSSL_SYS_MSDOS) && !defined(OPENSSL_SYS_WIN32)
#define SSL_MAX_CERT_LIST_DEFAULT 1024*30 /* 30k max cert list :-) */
#else
#define SSL_MAX_CERT_LIST_DEFAULT 1024*100 /* 100k max cert list :-) */
#endif

#define SSL_SESSION_CACHE_MAX_SIZE_DEFAULT      (1024*20)

/* This callback type is used inside SSL_CTX, SSL, and in the functions that set
 * them. It is used to override the generation of SSL/TLS session IDs in a
 * server. Return value should be zero on an error, non-zero to proceed. Also,
 * callbacks should themselves check if the id they generate is unique otherwise
 * the SSL handshake will fail with an error - callbacks can do this using the
 * 'ssl' value they're passed by;
 *      SSL_has_matching_session_id(ssl, id, *id_len)
 * The length value passed in is set at the maximum size the session ID can be.
 * In SSLv2 this is 16 bytes, whereas SSLv3/TLSv1 it is 32 bytes. The callback
 * can alter this length to be less if desired, but under SSLv2 session IDs are
 * supposed to be fixed at 16 bytes so the id will be padded after the callback
 * returns in this case. It is also an error for the callback to set the size to
 * zero. */
typedef int (*GEN_SESSION_CB)(const SSL *ssl, unsigned char *id,
                              unsigned int *id_len);

typedef struct ssl_comp_st SSL_COMP;

#ifndef OPENSSL_NO_SSL_INTERN

struct ssl_comp_st
        {
        int id;
        const char *name;
#ifndef OPENSSL_NO_COMP
        COMP_METHOD *method;
#else
        char *method;
#endif
        };

DECLARE_STACK_OF(SSL_COMP)
DECLARE_LHASH_OF(SSL_SESSION);

struct ssl_ctx_st
        {
        const SSL_METHOD *method;

        STACK_OF(SSL_CIPHER) *cipher_list;
        /* same as above but sorted for lookup */
        STACK_OF(SSL_CIPHER) *cipher_list_by_id;
```

```
789         struct x509_store_st /* X509_STORE */ *cert_store;
790         LHASH_OF(SSL_SESSION) *sessions;
791         /* Most session-ids that will be cached, default is
792          * SSL_SESSION_CACHE_MAX_SIZE_DEFAULT. 0 is unlimited. */
793         unsigned long session_cache_size;
794         struct ssl_session_st *session_cache_head;
795         struct ssl_session_st *session_cache_tail;

797         /* This can have one of 2 values, ored together,
798          * SSL_SESS_CACHE_CLIENT,
799          * SSL_SESS_CACHE_SERVER,
800          * Default is SSL_SESSION_CACHE_SERVER, which means only
801          * SSL_accept which cache SSL_SESSIONS. */
802         int session_cache_mode;

804         /* If timeout is not 0, it is the default timeout value set
805          * when SSL_new() is called.  This has been put in to make
806          * life easier to set things up */
807         long session_timeout;

809         /* If this callback is not null, it will be called each
810          * time a session id is added to the cache.  If this function
811          * returns 1, it means that the callback will do a
812          * SSL_SESSION_free() when it has finished using it.  Otherwise,
813          * on 0, it means the callback has finished with it.
814          * If remove_session_cb is not null, it will be called when
815          * a session-id is removed from the cache.  After the call,
816          * OpenSSL will SSL_SESSION_free() it. */
817         int (*new_session_cb)(struct ssl_st *ssl,SSL_SESSION *sess);
818         void (*remove_session_cb)(struct ssl_ctx_st *ctx,SSL_SESSION *sess);
819         SSL_SESSION *(*get_session_cb)(struct ssl_st *ssl,
820                 unsigned char *data,int len,int *copy);

822         struct
823                 {
824                 int sess_connect;       /* SSL new conn - started */
825                 int sess_connect_renegotiate;/* SSL reneg - requested */
826                 int sess_connect_good;  /* SSL new conne/reneg - finished */
827                 int sess_accept;        /* SSL new accept - started */
828                 int sess_accept_renegotiate;/* SSL reneg - requested */
829                 int sess_accept_good;   /* SSL accept/reneg - finished */
830                 int sess_miss;          /* session lookup misses  */
831                 int sess_timeout;       /* reuse attempt on timeouted session */
832                 int sess_cache_full;    /* session removed due to full cache */
833                 int sess_hit;           /* session reuse actually done */
834                 int sess_cb_hit;        /* session-id that was not
835                                          * in the cache was
836                                          * passed back via the callback.  This
837                                          * indicates that the application is
838                                          * supplying session-id's from other
839                                          * processes - spooky :-) */
840                 } stats;

842         int references;

844         /* if defined, these override the X509_verify_cert() calls */
845         int (*app_verify_callback)(X509_STORE_CTX *, void *);
846         void *app_verify_arg;
847         /* before OpenSSL 0.9.7, 'app_verify_arg' was ignored
848          * ('app_verify_callback' was called with just one argument) */

850         /* Default password callback. */
851         pem_password_cb *default_passwd_callback;

853         /* Default password callback user data. */
```



```
854         void *default_passwd_callback_userdata;

856         /* get client cert callback */
857         int (*client_cert_cb)(SSL *ssl, X509 **x509, EVP_PKEY **pkey);

859     /* cookie generate callback */
860     int (*app_gen_cookie_cb)(SSL *ssl, unsigned char *cookie,
861         unsigned int *cookie_len);

863     /* verify cookie callback */
864     int (*app_verify_cookie_cb)(SSL *ssl, unsigned char *cookie,
865         unsigned int cookie_len);

867         CRYPTO_EX_DATA ex_data;

869         const EVP_MD *rsa_md5;/* For SSLv2 - name is 'ssl2-md5' */
870         const EVP_MD *md5;      /* For SSLv3/TLSv1 'ssl3-md5' */
871         const EVP_MD *sha1;   /* For SSLv3/TLSv1 'ssl3->sha1' */

873         STACK_OF(X509) *extra_certs;
874         STACK_OF(SSL_COMP) *comp_methods; /* stack of SSL_COMP, SSLv3/TLSv1 */


877         /* Default values used when no per-SSL value is defined follow */

879         void (*info_callback)(const SSL *ssl,int type,int val); /* used if SSL's

881         /* what we put in client cert requests */
882         STACK_OF(X509_NAME) *client_CA;


885         /* Default values to use in SSL structures follow (these are copied by S

887         unsigned long options;
888         unsigned long mode;
889         long max_cert_list;

891         struct cert_st /* CERT */ *cert;
892         int read_ahead;

894         /* callback that allows applications to peek at protocol messages */
895         void (*msg_callback)(int write_p, int version, int content_type, const v
896         void *msg_callback_arg;

898         int verify_mode;
899         unsigned int sid_ctx_length;
900         unsigned char sid_ctx[SSL_MAX_SID_CTX_LENGTH];
901         int (*default_verify_callback)(int ok,X509_STORE_CTX *ctx); /* called 'v

903         /* Default generate session ID callback. */
904         GEN_SESSION_CB generate_session_id;

906         X509_VERIFY_PARAM *param;

908 #if 0
909         int purpose;            /* Purpose setting */
910         int trust;              /* Trust setting */
911 #endif

913         int quiet_shutdown;

915         /* Maximum amount of data to send in one fragment.
916          * actual record size can be more than this due to
917          * padding and MAC overheads.
918          */
919         unsigned int max_send_fragment;
```

```
#ifndef OPENSSL_NO_ENGINE
        /* Engine to pass requests for client certs to
         */
        ENGINE *client_cert_engine;
#endif

#ifndef OPENSSL_NO_TLSEXT
        /* TLS extensions servername callback */
        int (*tlsext_servername_callback)(SSL*, int *, void *);
        void *tlsext_servername_arg;
        /* RFC 4507 session ticket keys */
        unsigned char tlsext_tick_key_name[16];
        unsigned char tlsext_tick_hmac_key[16];
        unsigned char tlsext_tick_aes_key[16];
        /* Callback to support customisation of ticket key setting */
        int (*tlsext_ticket_key_cb)(SSL *ssl,
                                    unsigned char *name, unsigned char *iv,
                                    EVP_CIPHER_CTX *ectx,
                                    HMAC_CTX *hctx, int enc);

        /* certificate status request info */
        /* Callback for status request */
        int (*tlsext_status_cb)(SSL *ssl, void *arg);
        void *tlsext_status_arg;

        /* draft-rescorla-tls-opaque-prf-input-00.txt information */
        int (*tlsext_opaque_prf_input_callback)(SSL *, void *peerinput, size_t l
        void *tlsext_opaque_prf_input_callback_arg;
#endif

#ifndef OPENSSL_NO_PSK
        char *psk_identity_hint;
        unsigned int (*psk_client_callback)(SSL *ssl, const char *hint, char *id
                unsigned int max_identity_len, unsigned char *psk,
                unsigned int max_psk_len);
        unsigned int (*psk_server_callback)(SSL *ssl, const char *identity,
                unsigned char *psk, unsigned int max_psk_len);
#endif

#ifndef OPENSSL_NO_BUF_FREELISTS
#define SSL_MAX_BUF_FREELIST_LEN_DEFAULT 32
        unsigned int freelist_max_len;
        struct ssl3_buf_freelist_st *wbuf_freelist;
        struct ssl3_buf_freelist_st *rbuf_freelist;
#endif
#ifndef OPENSSL_NO_SRP
        SRP_CTX srp_ctx; /* ctx for SRP authentication */
#endif

#ifndef OPENSSL_NO_TLSEXT

# ifndef OPENSSL_NO_NEXTPROTONEG
        /* Next protocol negotiation information */
        /* (for experimental NPN extension). */

        /* For a server, this contains a callback function by which the set of
         * advertised protocols can be provided. */
        int (*next_protos_advertised_cb)(SSL *s, const unsigned char **buf,
                                         unsigned int *len, void *arg);
        void *next_protos_advertised_cb_arg;
        /* For a client, this contains a callback function that selects the
         * next protocol from the list provided by the server. */
        int (*next_proto_select_cb)(SSL *s, unsigned char **out,
                                    unsigned char *outlen,
                                    const unsigned char *in,
```



```
                                    unsigned int inlen,
                                    void *arg);
        void *next_proto_select_cb_arg;
# endif
        /* SRTP profiles we are willing to do from RFC 5764 */
        STACK_OF(SRTP_PROTECTION_PROFILE) *srtp_profiles;
#endif
        };

#endif

#define SSL_SESS_CACHE_OFF                      0x0000
#define SSL_SESS_CACHE_CLIENT                   0x0001
#define SSL_SESS_CACHE_SERVER                   0x0002
#define SSL_SESS_CACHE_BOTH     (SSL_SESS_CACHE_CLIENT|SSL_SESS_CACHE_SERVER)
#define SSL_SESS_CACHE_NO_AUTO_CLEAR            0x0080
/* enough comments already ... see SSL_CTX_set_session_cache_mode(3) */
#define SSL_SESS_CACHE_NO_INTERNAL_LOOKUP       0x0100
#define SSL_SESS_CACHE_NO_INTERNAL_STORE        0x0200
#define SSL_SESS_CACHE_NO_INTERNAL \
        (SSL_SESS_CACHE_NO_INTERNAL_LOOKUP|SSL_SESS_CACHE_NO_INTERNAL_STORE)

LHASH_OF(SSL_SESSION) *SSL_CTX_sessions(SSL_CTX *ctx);
#define SSL_CTX_sess_number(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_NUMBER,0,NULL)
#define SSL_CTX_sess_connect(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_CONNECT,0,NULL)
#define SSL_CTX_sess_connect_good(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_CONNECT_GOOD,0,NULL)
#define SSL_CTX_sess_connect_renegotiate(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_CONNECT_RENEGOTIATE,0,NULL)
#define SSL_CTX_sess_accept(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_ACCEPT,0,NULL)
#define SSL_CTX_sess_accept_renegotiate(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_ACCEPT_RENEGOTIATE,0,NULL)
#define SSL_CTX_sess_accept_good(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_ACCEPT_GOOD,0,NULL)
#define SSL_CTX_sess_hits(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_HIT,0,NULL)
#define SSL_CTX_sess_cb_hits(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_CB_HIT,0,NULL)
#define SSL_CTX_sess_misses(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_MISSES,0,NULL)
#define SSL_CTX_sess_timeouts(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_TIMEOUTS,0,NULL)
#define SSL_CTX_sess_cache_full(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SESS_CACHE_FULL,0,NULL)

void SSL_CTX_sess_set_new_cb(SSL_CTX *ctx, int (*new_session_cb)(struct ssl_st *
int (*SSL_CTX_sess_get_new_cb(SSL_CTX *ctx))(struct ssl_st *ssl, SSL_SESSION *se
void SSL_CTX_sess_set_remove_cb(SSL_CTX *ctx, void (*remove_session_cb)(struct s
void (*SSL_CTX_sess_get_remove_cb(SSL_CTX *ctx))(struct ssl_ctx_st *ctx, SSL_SES
void SSL_CTX_sess_set_get_cb(SSL_CTX *ctx, SSL_SESSION *(*get_session_cb)(struct
SSL_SESSION *(*SSL_CTX_sess_get_get_cb(SSL_CTX *ctx))(struct ssl_st *ssl, unsign
void SSL_CTX_set_info_callback(SSL_CTX *ctx, void (*cb)(const SSL *ssl,int type,
void (*SSL_CTX_get_info_callback(SSL_CTX *ctx))(const SSL *ssl,int type,int val)
void SSL_CTX_set_client_cert_cb(SSL_CTX *ctx, int (*client_cert_cb)(SSL *ssl, X5
int (*SSL_CTX_get_client_cert_cb(SSL_CTX *ctx))(SSL *ssl, X509 **x509, EVP_PKEY
#ifndef OPENSSL_NO_ENGINE
int SSL_CTX_set_client_cert_engine(SSL_CTX *ctx, ENGINE *e);
#endif
void SSL_CTX_set_cookie_generate_cb(SSL_CTX *ctx, int (*app_gen_cookie_cb)(SSL *
void SSL_CTX_set_cookie_verify_cb(SSL_CTX *ctx, int (*app_verify_cookie_cb)(SSL
#ifndef OPENSSL_NO_NEXTPROTONEG
void SSL_CTX_set_next_protos_advertised_cb(SSL_CTX *s,
                                           int (*cb) (SSL *ssl,
```

```
                                                const unsigned char **out,
                                                unsigned int *outlen,
                                                void *arg),
                                        void *arg);
void SSL_CTX_set_next_proto_select_cb(SSL_CTX *s,
                                      int (*cb) (SSL *ssl,
                                                 unsigned char **out,
                                                 unsigned char *outlen,
                                                 const unsigned char *in,
                                                 unsigned int inlen,
                                                 void *arg),
                                      void *arg);

int SSL_select_next_proto(unsigned char **out, unsigned char *outlen,
                          const unsigned char *in, unsigned int inlen,
                          const unsigned char *client, unsigned int client_len);
void SSL_get0_next_proto_negotiated(const SSL *s,
                                    const unsigned char **data, unsigned *len);

#define OPENSSL_NPN_UNSUPPORTED 0
#define OPENSSL_NPN_NEGOTIATED  1
#define OPENSSL_NPN_NO_OVERLAP  2
#endif

#ifndef OPENSSL_NO_PSK
/* the maximum length of the buffer given to callbacks containing the
 * resulting identity/psk */
#define PSK_MAX_IDENTITY_LEN 128
#define PSK_MAX_PSK_LEN 256
void SSL_CTX_set_psk_client_callback(SSL_CTX *ctx,
        unsigned int (*psk_client_callback)(SSL *ssl, const char *hint,
                char *identity, unsigned int max_identity_len, unsigned char *ps
                unsigned int max_psk_len));
void SSL_set_psk_client_callback(SSL *ssl,
        unsigned int (*psk_client_callback)(SSL *ssl, const char *hint,
                char *identity, unsigned int max_identity_len, unsigned char *ps
                unsigned int max_psk_len));
void SSL_CTX_set_psk_server_callback(SSL_CTX *ctx,
        unsigned int (*psk_server_callback)(SSL *ssl, const char *identity,
                unsigned char *psk, unsigned int max_psk_len));
void SSL_set_psk_server_callback(SSL *ssl,
        unsigned int (*psk_server_callback)(SSL *ssl, const char *identity,
                unsigned char *psk, unsigned int max_psk_len));
int SSL_CTX_use_psk_identity_hint(SSL_CTX *ctx, const char *identity_hint);
int SSL_use_psk_identity_hint(SSL *s, const char *identity_hint);
const char *SSL_get_psk_identity_hint(const SSL *s);
const char *SSL_get_psk_identity(const SSL *s);
#endif

#define SSL_NOTHING     1
#define SSL_WRITING     2
#define SSL_READING     3
#define SSL_X509_LOOKUP 4

/* These will only be used when doing non-blocking IO */
#define SSL_want_nothing(s)     (SSL_want(s) == SSL_NOTHING)
#define SSL_want_read(s)        (SSL_want(s) == SSL_READING)
#define SSL_want_write(s)       (SSL_want(s) == SSL_WRITING)
#define SSL_want_x509_lookup(s) (SSL_want(s) == SSL_X509_LOOKUP)

#define SSL_MAC_FLAG_READ_MAC_STREAM 1
#define SSL_MAC_FLAG_WRITE_MAC_STREAM 2

#ifndef OPENSSL_NO_SSL_INTERN

struct ssl_st
```



```
        {
        /* protocol version
         * (one of SSL2_VERSION, SSL3_VERSION, TLS1_VERSION, DTLS1_VERSION)
         */
        int version;
        int type; /* SSL_ST_CONNECT or SSL_ST_ACCEPT */

        const SSL_METHOD *method; /* SSLv3 */

        /* There are 2 BIO's even though they are normally both the
         * same.  This is so data can be read and written to different
         * handlers */

#ifndef OPENSSL_NO_BIO
        BIO *rbio; /* used by SSL_read */
        BIO *wbio; /* used by SSL_write */
        BIO *bbio; /* used during session-id reuse to concatenate
                    * messages */
#else
        char *rbio; /* used by SSL_read */
        char *wbio; /* used by SSL_write */
        char *bbio;
#endif
        /* This holds a variable that indicates what we were doing
         * when a 0 or -1 is returned.  This is needed for
         * non-blocking IO so we know what request needs re-doing when
         * in SSL_accept or SSL_connect */
        int rwstate;

        /* true when we are actually in SSL_accept() or SSL_connect() */
        int in_handshake;
        int (*handshake_func)(SSL *);

        /* Imagine that here's a boolean member "init" that is
         * switched as soon as SSL_set_{accept/connect}_state
         * is called for the first time, so that "state" and
         * "handshake_func" are properly initialized.  But as
         * handshake_func is == 0 until then, we use this
         * test instead of an "init" member.
         */

        int server;     /* are we the server side? - mostly used by SSL_clear*/

        int new_session;/* Generate a new session or reuse an old one.
                         * NB: For servers, the 'new' session may actually be a
                         * cached session or even the previous session unless
                         * SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION is set
        int quiet_shutdown;/* don't send shutdown packets */
        int shutdown;   /* we have shut things down, 0x01 sent, 0x02
                         * for received */
        int state;      /* where we are */
        int rstate;     /* where we are when reading */

        BUF_MEM *init_buf;      /* buffer used during init */
        void *init_msg;         /* pointer to handshake message body, set by ssl
        int init_num;           /* amount read/written */
        int init_off;           /* amount read/written */

        /* used internally to point at a raw packet */
        unsigned char *packet;
        unsigned int packet_length;

        struct ssl2_state_st *s2; /* SSLv2 variables */
        struct ssl3_state_st *s3; /* SSLv3 variables */
        struct dtls1_state_st *d1; /* DTLSv1 variables */
```

```
1184         int read_ahead;         /* Read as many input bytes as possible
1185                                  * (for non-blocking reads) */

1187         /* callback that allows applications to peek at protocol messages */
1188         void (*msg_callback)(int write_p, int version, int content_type, const v
1189         void *msg_callback_arg;

1191         int hit;                /* reusing a previous session */

1193         X509_VERIFY_PARAM *param;

1195 #if 0
1196         int purpose;            /* Purpose setting */
1197         int trust;              /* Trust setting */
1198 #endif

1200         /* crypto */
1201         STACK_OF(SSL_CIPHER) *cipher_list;
1202         STACK_OF(SSL_CIPHER) *cipher_list_by_id;

1204         /* These are the ones being used, the ones in SSL_SESSION are
1205          * the ones to be 'copied' into these ones */
1206         int mac_flags;
1207         EVP_CIPHER_CTX *enc_read_ctx;           /* cryptographic state */
1208         EVP_MD_CTX *read_hash;          /* used for mac generation */
1209 #ifndef OPENSSL_NO_COMP
1210         COMP_CTX *expand;                       /* uncompress */
1211 #else
1212         char *expand;
1213 #endif

1215         EVP_CIPHER_CTX *enc_write_ctx;          /* cryptographic state */
1216         EVP_MD_CTX *write_hash;         /* used for mac generation */
1217 #ifndef OPENSSL_NO_COMP
1218         COMP_CTX *compress;                     /* compression */
1219 #else
1220         char *compress;
1221 #endif

1223         /* session info */

1225         /* client cert? */
1226         /* This is used to hold the server certificate used */
1227         struct cert_st /* CERT */ *cert;

1229         /* the session_id_context is used to ensure sessions are only reused
1230          * in the appropriate context */
1231         unsigned int sid_ctx_length;
1232         unsigned char sid_ctx[SSL_MAX_SID_CTX_LENGTH];

1234         /* This can also be in the session once a session is established */
1235         SSL_SESSION *session;

1237         /* Default generate session ID callback. */
1238         GEN_SESSION_CB generate_session_id;

1240         /* Used in SSL2 and SSL3 */
1241         int verify_mode;        /* 0 don't care about verify failure.
1242                                  * 1 fail if verify fails */
1243         int (*verify_callback)(int ok,X509_STORE_CTX *ctx); /* fail if callback

1245         void (*info_callback)(const SSL *ssl,int type,int val); /* optional info

1247         int error;              /* error bytes to be written */
1248         int error_code;         /* actual code */
```



```
1250 #ifndef OPENSSL_NO_KRB5
1251         KSSL_CTX *kssl_ctx;     /* Kerberos 5 context */
1252 #endif  /* OPENSSL_NO_KRB5 */

1254 #ifndef OPENSSL_NO_PSK
1255         unsigned int (*psk_client_callback)(SSL *ssl, const char *hint, char *id
1256                 unsigned int max_identity_len, unsigned char *psk,
1257                 unsigned int max_psk_len);
1258         unsigned int (*psk_server_callback)(SSL *ssl, const char *identity,
1259                 unsigned char *psk, unsigned int max_psk_len);
1260 #endif

1262         SSL_CTX *ctx;
1263         /* set this flag to 1 and a sleep(1) is put into all SSL_read()
1264          * and SSL_write() calls, good for nbio debuging :-) */
1265         int debug;

1267         /* extra application data */
1268         long verify_result;
1269         CRYPTO_EX_DATA ex_data;

1271         /* for server side, keep the list of CA_dn we can use */
1272         STACK_OF(X509_NAME) *client_CA;

1274         int references;
1275         unsigned long options; /* protocol behaviour */
1276         unsigned long mode; /* API behaviour */
1277         long max_cert_list;
1278         int first_packet;
1279         int client_version;     /* what was passed, used for
1280                                  * SSLv3/TLS rollback check */
1281         unsigned int max_send_fragment;
1282 #ifndef OPENSSL_NO_TLSEXT
1283         /* TLS extension debug callback */
1284         void (*tlsext_debug_cb)(SSL *s, int client_server, int type,
1285                                         unsigned char *data, int len,
1286                                         void *arg);
1287         void *tlsext_debug_arg;
1288         char *tlsext_hostname;
1289         int servername_done;   /* no further mod of servername
1290                                   0 : call the servername extension callback.
1291                                   1 : prepare 2, allow last ack just after in se
1292                                   2 : don't call servername callback, no ack in
1293                                */
1294         /* certificate status request info */
1295         /* Status type or -1 if no status type */
1296         int tlsext_status_type;
1297         /* Expect OCSP CertificateStatus message */
1298         int tlsext_status_expected;
1299         /* OCSP status request only */
1300         STACK_OF(OCSP_RESPID) *tlsext_ocsp_ids;
1301         X509_EXTENSIONS *tlsext_ocsp_exts;
1302         /* OCSP response received or to be sent */
1303         unsigned char *tlsext_ocsp_resp;
1304         int tlsext_ocsp_resplen;

1306         /* RFC4507 session ticket expected to be received or sent */
1307         int tlsext_ticket_expected;
1308 #ifndef OPENSSL_NO_EC
1309         size_t tlsext_ecpointformatlist_length;
1310         unsigned char *tlsext_ecpointformatlist; /* our list */
1311         size_t tlsext_ellipticcurvelist_length;
1312         unsigned char *tlsext_ellipticcurvelist; /* our list */
1313 #endif /* OPENSSL_NO_EC */

1315         /* draft-rescorla-tls-opaque-prf-input-00.txt information to be used for
```

```
1316         void *tlsext_opaque_prf_input;
1317         size_t tlsext_opaque_prf_input_len;

1319         /* TLS Session Ticket extension override */
1320         TLS_SESSION_TICKET_EXT *tlsext_session_ticket;

1322         /* TLS Session Ticket extension callback */
1323         tls_session_ticket_ext_cb_fn tls_session_ticket_ext_cb;
1324         void *tls_session_ticket_ext_cb_arg;

1326         /* TLS pre-shared secret session resumption */
1327         tls_session_secret_cb_fn tls_session_secret_cb;
1328         void *tls_session_secret_cb_arg;

1330         SSL_CTX * initial_ctx; /* initial ctx, used to store sessions */

1332 #ifndef OPENSSL_NO_NEXTPROTONEG
1333         /* Next protocol negotiation. For the client, this is the protocol that
1334          * we sent in NextProtocol and is set when handling ServerHello
1335          * extensions.
1336          *
1337          * For a server, this is the client's selected_protocol from
1338          * NextProtocol and is set when handling the NextProtocol message,
1339          * before the Finished message. */
1340         unsigned char *next_proto_negotiated;
1341         unsigned char next_proto_negotiated_len;
1342 #endif

1344 #define session_ctx initial_ctx

1346         STACK_OF(SRTP_PROTECTION_PROFILE) *srtp_profiles;  /* What we'll do */
1347         SRTP_PROTECTION_PROFILE *srtp_profile;           /* What's been chosen

1349         unsigned int tlsext_heartbeat;  /* Is use of the Heartbeat extension neg
1350                                            0: disabled
1351                                            1: enabled
1352                                            2: enabled, but not allowed to send R
1353                                         */
1354         unsigned int tlsext_hb_pending; /* Indicates if a HeartbeatRequest is in
1355         unsigned int tlsext_hb_seq;     /* HeartbeatRequest sequence number */
1356 #else
1357 #define session_ctx ctx
1358 #endif /* OPENSSL_NO_TLSEXT */

1360         int renegotiate;/* 1 if we are renegotiating.
1361                          * 2 if we are a server and are inside a handshake
1362                          * (i.e. not just sending a HelloRequest) */

1364 #ifndef OPENSSL_NO_SRP
1365         SRP_CTX srp_ctx; /* ctx for SRP authentication */
1366 #endif
1367         };

1369 #endif

1371 #ifdef __cplusplus
1372 }
1373 #endif

1375 #include <openssl/ssl2.h>
1376 #include <openssl/ssl3.h>
1377 #include <openssl/tls1.h> /* This is mostly sslv3 with a few tweaks */
1378 #include <openssl/dtls1.h> /* Datagram TLS */
1379 #include <openssl/ssl23.h>
1380 #include <openssl/srtp.h>  /* Support for the use_srtp extension */
```



```
1382 #ifdef  __cplusplus
1383 extern "C" {
1384 #endif

1386 /* compatibility */
1387 #define SSL_set_app_data(s,arg)         (SSL_set_ex_data(s,0,(char *)arg))
1388 #define SSL_get_app_data(s)             (SSL_get_ex_data(s,0))
1389 #define SSL_SESSION_set_app_data(s,a)   (SSL_SESSION_set_ex_data(s,0,(char *)a))
1390 #define SSL_SESSION_get_app_data(s)     (SSL_SESSION_get_ex_data(s,0))
1391 #define SSL_CTX_get_app_data(ctx)       (SSL_CTX_get_ex_data(ctx,0))
1392 #define SSL_CTX_set_app_data(ctx,arg)   (SSL_CTX_set_ex_data(ctx,0,(char *)arg))

1394 /* The following are the possible values for ssl->state are are
1395  * used to indicate where we are up to in the SSL connection establishment.
1396  * The macros that follow are about the only things you should need to use
1397  * and even then, only when using non-blocking IO.
1398  * It can also be useful to work out where you were when the connection
1399  * failed */

1401 #define SSL_ST_CONNECT                  0x1000
1402 #define SSL_ST_ACCEPT                   0x2000
1403 #define SSL_ST_MASK                     0x0FFF
1404 #define SSL_ST_INIT                     (SSL_ST_CONNECT|SSL_ST_ACCEPT)
1405 #define SSL_ST_BEFORE                   0x4000
1406 #define SSL_ST_OK                       0x03
1407 #define SSL_ST_RENEGOTIATE              (0x04|SSL_ST_INIT)

1409 #define SSL_CB_LOOP                     0x01
1410 #define SSL_CB_EXIT                     0x02
1411 #define SSL_CB_READ                     0x04
1412 #define SSL_CB_WRITE                    0x08
1413 #define SSL_CB_ALERT                    0x4000 /* used in callback */
1414 #define SSL_CB_READ_ALERT               (SSL_CB_ALERT|SSL_CB_READ)
1415 #define SSL_CB_WRITE_ALERT              (SSL_CB_ALERT|SSL_CB_WRITE)
1416 #define SSL_CB_ACCEPT_LOOP              (SSL_ST_ACCEPT|SSL_CB_LOOP)
1417 #define SSL_CB_ACCEPT_EXIT              (SSL_ST_ACCEPT|SSL_CB_EXIT)
1418 #define SSL_CB_CONNECT_LOOP             (SSL_ST_CONNECT|SSL_CB_LOOP)
1419 #define SSL_CB_CONNECT_EXIT             (SSL_ST_CONNECT|SSL_CB_EXIT)
1420 #define SSL_CB_HANDSHAKE_START          0x10
1421 #define SSL_CB_HANDSHAKE_DONE           0x20

1423 /* Is the SSL_connection established? */
1424 #define SSL_get_state(a)                SSL_state(a)
1425 #define SSL_is_init_finished(a)         (SSL_state(a) == SSL_ST_OK)
1426 #define SSL_in_init(a)                  (SSL_state(a)&SSL_ST_INIT)
1427 #define SSL_in_before(a)                (SSL_state(a)&SSL_ST_BEFORE)
1428 #define SSL_in_connect_init(a)          (SSL_state(a)&SSL_ST_CONNECT)
1429 #define SSL_in_accept_init(a)           (SSL_state(a)&SSL_ST_ACCEPT)

1431 /* The following 2 states are kept in ssl->rstate when reads fail,
1432  * you should not need these */
1433 #define SSL_ST_READ_HEADER                      0xF0
1434 #define SSL_ST_READ_BODY                        0xF1
1435 #define SSL_ST_READ_DONE                        0xF2

1437 /* Obtain latest Finished message
1438  *   -- that we sent (SSL_get_finished)
1439  *   -- that we expected from peer (SSL_get_peer_finished).
1440  * Returns length (0 == no Finished so far), copies up to 'count' bytes. */
1441 size_t SSL_get_finished(const SSL *s, void *buf, size_t count);
1442 size_t SSL_get_peer_finished(const SSL *s, void *buf, size_t count);

1444 /* use either SSL_VERIFY_NONE or SSL_VERIFY_PEER, the last 2 options
1445  * are 'ored' with SSL_VERIFY_PEER if they are desired */
1446 #define SSL_VERIFY_NONE                 0x00
1447 #define SSL_VERIFY_PEER                 0x01
```

```
#define SSL_VERIFY_FAIL_IF_NO_PEER_CERT 0x02
#define SSL_VERIFY_CLIENT_ONCE          0x04

#define OpenSSL_add_ssl_algorithms()    SSL_library_init()
#define SSLeay_add_ssl_algorithms()     SSL_library_init()

/* this is for backward compatibility */
#if 0 /* NEW_SSLEAY */
#define SSL_CTX_set_default_verify(a,b,c) SSL_CTX_set_verify(a,b,c)
#define SSL_set_pref_cipher(c,n)        SSL_set_cipher_list(c,n)
#define SSL_add_session(a,b)            SSL_CTX_add_session((a),(b))
#define SSL_remove_session(a,b)         SSL_CTX_remove_session((a),(b))
#define SSL_flush_sessions(a,b)         SSL_CTX_flush_sessions((a),(b))
#endif
/* More backward compatibility */
#define SSL_get_cipher(s) \
                SSL_CIPHER_get_name(SSL_get_current_cipher(s))
#define SSL_get_cipher_bits(s,np) \
                SSL_CIPHER_get_bits(SSL_get_current_cipher(s),np)
#define SSL_get_cipher_version(s) \
                SSL_CIPHER_get_version(SSL_get_current_cipher(s))
#define SSL_get_cipher_name(s) \
                SSL_CIPHER_get_name(SSL_get_current_cipher(s))
#define SSL_get_time(a)         SSL_SESSION_get_time(a)
#define SSL_set_time(a,b)       SSL_SESSION_set_time((a),(b))
#define SSL_get_timeout(a)      SSL_SESSION_get_timeout(a)
#define SSL_set_timeout(a,b)    SSL_SESSION_set_timeout((a),(b))

#define d2i_SSL_SESSION_bio(bp,s_id) ASN1_d2i_bio_of(SSL_SESSION,SSL_SESSION_new
#define i2d_SSL_SESSION_bio(bp,s_id) ASN1_i2d_bio_of(SSL_SESSION,i2d_SSL_SESSION

DECLARE_PEM_rw(SSL_SESSION, SSL_SESSION)

#define SSL_AD_REASON_OFFSET            1000 /* offset to get SSL_R_... value fr

/* These alert types are for SSLv3 and TLSv1 */
#define SSL_AD_CLOSE_NOTIFY             SSL3_AD_CLOSE_NOTIFY
#define SSL_AD_UNEXPECTED_MESSAGE       SSL3_AD_UNEXPECTED_MESSAGE /* fatal */
#define SSL_AD_BAD_RECORD_MAC           SSL3_AD_BAD_RECORD_MAC     /* fatal */
#define SSL_AD_DECRYPTION_FAILED        TLS1_AD_DECRYPTION_FAILED
#define SSL_AD_RECORD_OVERFLOW          TLS1_AD_RECORD_OVERFLOW
#define SSL_AD_DECOMPRESSION_FAILURE    SSL3_AD_DECOMPRESSION_FAILURE/* fatal */
#define SSL_AD_HANDSHAKE_FAILURE        SSL3_AD_HANDSHAKE_FAILURE/* fatal */
#define SSL_AD_NO_CERTIFICATE           SSL3_AD_NO_CERTIFICATE /* Not for TLS */
#define SSL_AD_BAD_CERTIFICATE          SSL3_AD_BAD_CERTIFICATE
#define SSL_AD_UNSUPPORTED_CERTIFICATE  SSL3_AD_UNSUPPORTED_CERTIFICATE
#define SSL_AD_CERTIFICATE_REVOKED      SSL3_AD_CERTIFICATE_REVOKED
#define SSL_AD_CERTIFICATE_EXPIRED      SSL3_AD_CERTIFICATE_EXPIRED
#define SSL_AD_CERTIFICATE_UNKNOWN      SSL3_AD_CERTIFICATE_UNKNOWN
#define SSL_AD_ILLEGAL_PARAMETER        SSL3_AD_ILLEGAL_PARAMETER   /* fatal */
#define SSL_AD_UNKNOWN_CA               TLS1_AD_UNKNOWN_CA      /* fatal */
#define SSL_AD_ACCESS_DENIED            TLS1_AD_ACCESS_DENIED   /* fatal */
#define SSL_AD_DECODE_ERROR             TLS1_AD_DECODE_ERROR    /* fatal */
#define SSL_AD_DECRYPT_ERROR            TLS1_AD_DECRYPT_ERROR
#define SSL_AD_EXPORT_RESTRICTION       TLS1_AD_EXPORT_RESTRICTION/* fatal */
#define SSL_AD_PROTOCOL_VERSION         TLS1_AD_PROTOCOL_VERSION /* fatal */
#define SSL_AD_INSUFFICIENT_SECURITY    TLS1_AD_INSUFFICIENT_SECURITY/* fatal */
#define SSL_AD_INTERNAL_ERROR           TLS1_AD_INTERNAL_ERROR  /* fatal */
#define SSL_AD_USER_CANCELLED           TLS1_AD_USER_CANCELLED
#define SSL_AD_NO_RENEGOTIATION         TLS1_AD_NO_RENEGOTIATION
#define SSL_AD_UNSUPPORTED_EXTENSION    TLS1_AD_UNSUPPORTED_EXTENSION
#define SSL_AD_CERTIFICATE_UNOBTAINABLE TLS1_AD_CERTIFICATE_UNOBTAINABLE
#define SSL_AD_UNRECOGNIZED_NAME        TLS1_AD_UNRECOGNIZED_NAME
#define SSL_AD_BAD_CERTIFICATE_STATUS_RESPONSE TLS1_AD_BAD_CERTIFICATE_STATUS_RE
#define SSL_AD_BAD_CERTIFICATE_HASH_VALUE TLS1_AD_BAD_CERTIFICATE_HASH_VALUE
#define SSL_AD_UNKNOWN_PSK_IDENTITY     TLS1_AD_UNKNOWN_PSK_IDENTITY /* fatal */
```



```
#define SSL_ERROR_NONE                  0
#define SSL_ERROR_SSL                   1
#define SSL_ERROR_WANT_READ             2
#define SSL_ERROR_WANT_WRITE            3
#define SSL_ERROR_WANT_X509_LOOKUP      4
#define SSL_ERROR_SYSCALL               5 /* look at error stack/return value/er
#define SSL_ERROR_ZERO_RETURN           6
#define SSL_ERROR_WANT_CONNECT          7
#define SSL_ERROR_WANT_ACCEPT           8

#define SSL_CTRL_NEED_TMP_RSA                   1
#define SSL_CTRL_SET_TMP_RSA                    2
#define SSL_CTRL_SET_TMP_DH                     3
#define SSL_CTRL_SET_TMP_ECDH                   4
#define SSL_CTRL_SET_TMP_RSA_CB                 5
#define SSL_CTRL_SET_TMP_DH_CB                  6
#define SSL_CTRL_SET_TMP_ECDH_CB                7

#define SSL_CTRL_GET_SESSION_REUSED             8
#define SSL_CTRL_GET_CLIENT_CERT_REQUEST        9
#define SSL_CTRL_GET_NUM_RENEGOTIATIONS         10
#define SSL_CTRL_CLEAR_NUM_RENEGOTIATIONS       11
#define SSL_CTRL_GET_TOTAL_RENEGOTIATIONS       12
#define SSL_CTRL_GET_FLAGS                      13
#define SSL_CTRL_EXTRA_CHAIN_CERT               14

#define SSL_CTRL_SET_MSG_CALLBACK               15
#define SSL_CTRL_SET_MSG_CALLBACK_ARG           16

/* only applies to datagram connections */
#define SSL_CTRL_SET_MTU                17
/* Stats */
#define SSL_CTRL_SESS_NUMBER                    20
#define SSL_CTRL_SESS_CONNECT                   21
#define SSL_CTRL_SESS_CONNECT_GOOD              22
#define SSL_CTRL_SESS_CONNECT_RENEGOTIATE       23
#define SSL_CTRL_SESS_ACCEPT                    24
#define SSL_CTRL_SESS_ACCEPT_GOOD               25
#define SSL_CTRL_SESS_ACCEPT_RENEGOTIATE        26
#define SSL_CTRL_SESS_HIT                       27
#define SSL_CTRL_SESS_CB_HIT                    28
#define SSL_CTRL_SESS_MISSES                    29
#define SSL_CTRL_SESS_TIMEOUTS                  30
#define SSL_CTRL_SESS_CACHE_FULL                31
#define SSL_CTRL_OPTIONS                        32
#define SSL_CTRL_MODE                           33

#define SSL_CTRL_GET_READ_AHEAD                 40
#define SSL_CTRL_SET_READ_AHEAD                 41
#define SSL_CTRL_SET_SESS_CACHE_SIZE            42
#define SSL_CTRL_GET_SESS_CACHE_SIZE            43
#define SSL_CTRL_SET_SESS_CACHE_MODE            44
#define SSL_CTRL_GET_SESS_CACHE_MODE            45

#define SSL_CTRL_GET_MAX_CERT_LIST              50
#define SSL_CTRL_SET_MAX_CERT_LIST              51

#define SSL_CTRL_SET_MAX_SEND_FRAGMENT          52

/* see tls1.h for macros based on these */
#ifndef OPENSSL_NO_TLSEXT
#define SSL_CTRL_SET_TLSEXT_SERVERNAME_CB       53
#define SSL_CTRL_SET_TLSEXT_SERVERNAME_ARG      54
#define SSL_CTRL_SET_TLSEXT_HOSTNAME            55
#define SSL_CTRL_SET_TLSEXT_DEBUG_CB            56
```

```
#define SSL_CTRL_SET_TLSEXT_DEBUG_ARG           57
#define SSL_CTRL_GET_TLSEXT_TICKET_KEYS         58
#define SSL_CTRL_SET_TLSEXT_TICKET_KEYS         59
#define SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT    60
#define SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT_CB 61
#define SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT_CB_ARG 62
#define SSL_CTRL_SET_TLSEXT_STATUS_REQ_CB       63
#define SSL_CTRL_SET_TLSEXT_STATUS_REQ_CB_ARG   64
#define SSL_CTRL_SET_TLSEXT_STATUS_REQ_TYPE     65
#define SSL_CTRL_GET_TLSEXT_STATUS_REQ_EXTS     66
#define SSL_CTRL_SET_TLSEXT_STATUS_REQ_EXTS     67
#define SSL_CTRL_GET_TLSEXT_STATUS_REQ_IDS      68
#define SSL_CTRL_SET_TLSEXT_STATUS_REQ_IDS      69
#define SSL_CTRL_GET_TLSEXT_STATUS_REQ_OCSP_RESP        70
#define SSL_CTRL_SET_TLSEXT_STATUS_REQ_OCSP_RESP        71

#define SSL_CTRL_SET_TLSEXT_TICKET_KEY_CB       72

#define SSL_CTRL_SET_TLS_EXT_SRP_USERNAME_CB    75
#define SSL_CTRL_SET_SRP_VERIFY_PARAM_CB                76
#define SSL_CTRL_SET_SRP_GIVE_CLIENT_PWD_CB             77

#define SSL_CTRL_SET_SRP_ARG            78
#define SSL_CTRL_SET_TLS_EXT_SRP_USERNAME               79
#define SSL_CTRL_SET_TLS_EXT_SRP_STRENGTH               80
#define SSL_CTRL_SET_TLS_EXT_SRP_PASSWORD               81
#ifndef OPENSSL_NO_HEARTBEATS
#define SSL_CTRL_TLS_EXT_SEND_HEARTBEAT                         85
#define SSL_CTRL_GET_TLS_EXT_HEARTBEAT_PENDING          86
#define SSL_CTRL_SET_TLS_EXT_HEARTBEAT_NO_REQUESTS      87
#endif
#endif

#define DTLS_CTRL_GET_TIMEOUT           73
#define DTLS_CTRL_HANDLE_TIMEOUT        74
#define DTLS_CTRL_LISTEN                        75

#define SSL_CTRL_GET_RI_SUPPORT                 76
#define SSL_CTRL_CLEAR_OPTIONS                  77
#define SSL_CTRL_CLEAR_MODE                     78

#define SSL_CTRL_GET_EXTRA_CHAIN_CERTS          82
#define SSL_CTRL_CLEAR_EXTRA_CHAIN_CERTS        83

#define DTLSv1_get_timeout(ssl, arg) \
        SSL_ctrl(ssl,DTLS_CTRL_GET_TIMEOUT,0, (void *)arg)
#define DTLSv1_handle_timeout(ssl) \
        SSL_ctrl(ssl,DTLS_CTRL_HANDLE_TIMEOUT,0, NULL)
#define DTLSv1_listen(ssl, peer) \
        SSL_ctrl(ssl,DTLS_CTRL_LISTEN,0, (void *)peer)

#define SSL_session_reused(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_GET_SESSION_REUSED,0,NULL)
#define SSL_num_renegotiations(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_GET_NUM_RENEGOTIATIONS,0,NULL)
#define SSL_clear_num_renegotiations(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_CLEAR_NUM_RENEGOTIATIONS,0,NULL)
#define SSL_total_renegotiations(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_GET_TOTAL_RENEGOTIATIONS,0,NULL)

#define SSL_CTX_need_tmp_RSA(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_NEED_TMP_RSA,0,NULL)
#define SSL_CTX_set_tmp_rsa(ctx,rsa) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_TMP_RSA,0,(char *)rsa)
#define SSL_CTX_set_tmp_dh(ctx,dh) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_TMP_DH,0,(char *)dh)
```



```
#define SSL_CTX_set_tmp_ecdh(ctx,ecdh) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_TMP_ECDH,0,(char *)ecdh)

#define SSL_need_tmp_RSA(ssl) \
        SSL_ctrl(ssl,SSL_CTRL_NEED_TMP_RSA,0,NULL)
#define SSL_set_tmp_rsa(ssl,rsa) \
        SSL_ctrl(ssl,SSL_CTRL_SET_TMP_RSA,0,(char *)rsa)
#define SSL_set_tmp_dh(ssl,dh) \
        SSL_ctrl(ssl,SSL_CTRL_SET_TMP_DH,0,(char *)dh)
#define SSL_set_tmp_ecdh(ssl,ecdh) \
        SSL_ctrl(ssl,SSL_CTRL_SET_TMP_ECDH,0,(char *)ecdh)

#define SSL_CTX_add_extra_chain_cert(ctx,x509) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_EXTRA_CHAIN_CERT,0,(char *)x509)
#define SSL_CTX_get_extra_chain_certs(ctx,px509) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_GET_EXTRA_CHAIN_CERTS,0,px509)
#define SSL_CTX_clear_extra_chain_certs(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_CLEAR_EXTRA_CHAIN_CERTS,0,NULL)

#ifndef OPENSSL_NO_BIO
BIO_METHOD *BIO_f_ssl(void);
BIO *BIO_new_ssl(SSL_CTX *ctx,int client);
BIO *BIO_new_ssl_connect(SSL_CTX *ctx);
BIO *BIO_new_buffer_ssl_connect(SSL_CTX *ctx);
int BIO_ssl_copy_session_id(BIO *to,BIO *from);
void BIO_ssl_shutdown(BIO *ssl_bio);

#endif

int     SSL_CTX_set_cipher_list(SSL_CTX *,const char *str);
SSL_CTX *SSL_CTX_new(const SSL_METHOD *meth);
void    SSL_CTX_free(SSL_CTX *);
long SSL_CTX_set_timeout(SSL_CTX *ctx,long t);
long SSL_CTX_get_timeout(const SSL_CTX *ctx);
X509_STORE *SSL_CTX_get_cert_store(const SSL_CTX *);
void SSL_CTX_set_cert_store(SSL_CTX *,X509_STORE *);
int SSL_want(const SSL *s);
int     SSL_clear(SSL *s);

void    SSL_CTX_flush_sessions(SSL_CTX *ctx,long tm);

const SSL_CIPHER *SSL_get_current_cipher(const SSL *s);
int     SSL_CIPHER_get_bits(const SSL_CIPHER *c,int *alg_bits);
char *  SSL_CIPHER_get_version(const SSL_CIPHER *c);
const char *    SSL_CIPHER_get_name(const SSL_CIPHER *c);
unsigned long   SSL_CIPHER_get_id(const SSL_CIPHER *c);

int     SSL_get_fd(const SSL *s);
int     SSL_get_rfd(const SSL *s);
int     SSL_get_wfd(const SSL *s);
const char  * SSL_get_cipher_list(const SSL *s,int n);
char *  SSL_get_shared_ciphers(const SSL *s, char *buf, int len);
int     SSL_get_read_ahead(const SSL * s);
int     SSL_pending(const SSL *s);
#ifndef OPENSSL_NO_SOCK
int     SSL_set_fd(SSL *s, int fd);
int     SSL_set_rfd(SSL *s, int fd);
int     SSL_set_wfd(SSL *s, int fd);
#endif
#ifndef OPENSSL_NO_BIO
void    SSL_set_bio(SSL *s, BIO *rbio,BIO *wbio);
BIO *   SSL_get_rbio(const SSL *s);
BIO *   SSL_get_wbio(const SSL *s);
#endif
int     SSL_set_cipher_list(SSL *s, const char *str);
void    SSL_set_read_ahead(SSL *s, int yes);
```

```
int     SSL_get_verify_mode(const SSL *s);
int     SSL_get_verify_depth(const SSL *s);
int     (*SSL_get_verify_callback(const SSL *s))(int,X509_STORE_CTX *);
void    SSL_set_verify(SSL *s, int mode,
                       int (*callback)(int ok,X509_STORE_CTX *ctx));
void    SSL_set_verify_depth(SSL *s, int depth);
#ifndef OPENSSL_NO_RSA
int     SSL_use_RSAPrivateKey(SSL *ssl, RSA *rsa);
#endif
int     SSL_use_RSAPrivateKey_ASN1(SSL *ssl, unsigned char *d, long len);
int     SSL_use_PrivateKey(SSL *ssl, EVP_PKEY *pkey);
int     SSL_use_PrivateKey_ASN1(int pk,SSL *ssl, const unsigned char *d, long le
int     SSL_use_certificate(SSL *ssl, X509 *x);
int     SSL_use_certificate_ASN1(SSL *ssl, const unsigned char *d, int len);

#ifndef OPENSSL_NO_STDIO
int     SSL_use_RSAPrivateKey_file(SSL *ssl, const char *file, int type);
int     SSL_use_PrivateKey_file(SSL *ssl, const char *file, int type);
int     SSL_use_certificate_file(SSL *ssl, const char *file, int type);
int     SSL_CTX_use_RSAPrivateKey_file(SSL_CTX *ctx, const char *file, int type)
int     SSL_CTX_use_PrivateKey_file(SSL_CTX *ctx, const char *file, int type);
int     SSL_CTX_use_certificate_file(SSL_CTX *ctx, const char *file, int type);
int     SSL_CTX_use_certificate_chain_file(SSL_CTX *ctx, const char *file); /* P
STACK_OF(X509_NAME) *SSL_load_client_CA_file(const char *file);
int     SSL_add_file_cert_subjects_to_stack(STACK_OF(X509_NAME) *stackCAs,
                                            const char *file);
#ifndef OPENSSL_SYS_VMS
#ifndef OPENSSL_SYS_MACINTOSH_CLASSIC /* XXXXX: Better scheme needed! [was: #ifn
int     SSL_add_dir_cert_subjects_to_stack(STACK_OF(X509_NAME) *stackCAs,
                                           const char *dir);
#endif
#endif

#endif

void    SSL_load_error_strings(void );
const char *SSL_state_string(const SSL *s);
const char *SSL_rstate_string(const SSL *s);
const char *SSL_state_string_long(const SSL *s);
const char *SSL_rstate_string_long(const SSL *s);
long    SSL_SESSION_get_time(const SSL_SESSION *s);
long    SSL_SESSION_set_time(SSL_SESSION *s, long t);
long    SSL_SESSION_get_timeout(const SSL_SESSION *s);
long    SSL_SESSION_set_timeout(SSL_SESSION *s, long t);
void    SSL_copy_session_id(SSL *to,const SSL *from);
X509 *SSL_SESSION_get0_peer(SSL_SESSION *s);
int SSL_SESSION_set1_id_context(SSL_SESSION *s,const unsigned char *sid_ctx,
                                unsigned int sid_ctx_len);

SSL_SESSION *SSL_SESSION_new(void);
const unsigned char *SSL_SESSION_get_id(const SSL_SESSION *s,
                                        unsigned int *len);
unsigned int SSL_SESSION_get_compress_id(const SSL_SESSION *s);
#ifndef OPENSSL_NO_FP_API
int     SSL_SESSION_print_fp(FILE *fp,const SSL_SESSION *ses);
#endif
#ifndef OPENSSL_NO_BIO
int     SSL_SESSION_print(BIO *fp,const SSL_SESSION *ses);
#endif
void    SSL_SESSION_free(SSL_SESSION *ses);
int     i2d_SSL_SESSION(SSL_SESSION *in,unsigned char **pp);
int     SSL_set_session(SSL *to, SSL_SESSION *session);
int     SSL_CTX_add_session(SSL_CTX *s, SSL_SESSION *c);
int     SSL_CTX_remove_session(SSL_CTX *,SSL_SESSION *c);
int     SSL_CTX_set_generate_session_id(SSL_CTX *, GEN_SESSION_CB);
int     SSL_set_generate_session_id(SSL *, GEN_SESSION_CB);
```



```
int     SSL_has_matching_session_id(const SSL *ssl, const unsigned char *id,
                                    unsigned int id_len);
SSL_SESSION *d2i_SSL_SESSION(SSL_SESSION **a,const unsigned char **pp,
                             long length);

#ifdef HEADER_X509_H
X509 *  SSL_get_peer_certificate(const SSL *s);
#endif

STACK_OF(X509) *SSL_get_peer_cert_chain(const SSL *s);

int SSL_CTX_get_verify_mode(const SSL_CTX *ctx);
int SSL_CTX_get_verify_depth(const SSL_CTX *ctx);
int (*SSL_CTX_get_verify_callback(const SSL_CTX *ctx))(int,X509_STORE_CTX *);
void SSL_CTX_set_verify(SSL_CTX *ctx,int mode,
                        int (*callback)(int, X509_STORE_CTX *));
void SSL_CTX_set_verify_depth(SSL_CTX *ctx,int depth);
void SSL_CTX_set_cert_verify_callback(SSL_CTX *ctx, int (*cb)(X509_STORE_CTX *,v
#ifndef OPENSSL_NO_RSA
int SSL_CTX_use_RSAPrivateKey(SSL_CTX *ctx, RSA *rsa);
#endif
int SSL_CTX_use_RSAPrivateKey_ASN1(SSL_CTX *ctx, const unsigned char *d, long le
int SSL_CTX_use_PrivateKey(SSL_CTX *ctx, EVP_PKEY *pkey);
int SSL_CTX_use_PrivateKey_ASN1(int pk,SSL_CTX *ctx,
        const unsigned char *d, long len);
int SSL_CTX_use_certificate(SSL_CTX *ctx, X509 *x);
int SSL_CTX_use_certificate_ASN1(SSL_CTX *ctx, int len, const unsigned char *d);

void SSL_CTX_set_default_passwd_cb(SSL_CTX *ctx, pem_password_cb *cb);
void SSL_CTX_set_default_passwd_cb_userdata(SSL_CTX *ctx, void *u);

int SSL_CTX_check_private_key(const SSL_CTX *ctx);
int SSL_check_private_key(const SSL *ctx);

int     SSL_CTX_set_session_id_context(SSL_CTX *ctx,const unsigned char *sid_ctx
                                       unsigned int sid_ctx_len);

SSL *   SSL_new(SSL_CTX *ctx);
int     SSL_set_session_id_context(SSL *ssl,const unsigned char *sid_ctx,
                                   unsigned int sid_ctx_len);

int SSL_CTX_set_purpose(SSL_CTX *s, int purpose);
int SSL_set_purpose(SSL *s, int purpose);
int SSL_CTX_set_trust(SSL_CTX *s, int trust);
int SSL_set_trust(SSL *s, int trust);

int SSL_CTX_set1_param(SSL_CTX *ctx, X509_VERIFY_PARAM *vpm);
int SSL_set1_param(SSL *ssl, X509_VERIFY_PARAM *vpm);

#ifndef OPENSSL_NO_SRP
int SSL_CTX_set_srp_username(SSL_CTX *ctx,char *name);
int SSL_CTX_set_srp_password(SSL_CTX *ctx,char *password);
int SSL_CTX_set_srp_strength(SSL_CTX *ctx, int strength);
int SSL_CTX_set_srp_client_pwd_callback(SSL_CTX *ctx,
                                        char *(*cb)(SSL *,void *));
int SSL_CTX_set_srp_verify_param_callback(SSL_CTX *ctx,
                                          int (*cb)(SSL *,void *));
int SSL_CTX_set_srp_username_callback(SSL_CTX *ctx,
                                      int (*cb)(SSL *,int *,void *));
int SSL_CTX_set_srp_cb_arg(SSL_CTX *ctx, void *arg);

int SSL_set_srp_server_param(SSL *s, const BIGNUM *N, const BIGNUM *g,
                             BIGNUM *sa, BIGNUM *v, char *info);
int SSL_set_srp_server_param_pw(SSL *s, const char *user, const char *pass,
                                const char *grp);
```

```
1844 BIGNUM *SSL_get_srp_g(SSL *s);
1845 BIGNUM *SSL_get_srp_N(SSL *s);

1847 char *SSL_get_srp_username(SSL *s);
1848 char *SSL_get_srp_userinfo(SSL *s);
1849 #endif

1851 void    SSL_free(SSL *ssl);
1852 int     SSL_accept(SSL *ssl);
1853 int     SSL_connect(SSL *ssl);
1854 int     SSL_read(SSL *ssl,void *buf,int num);
1855 int     SSL_peek(SSL *ssl,void *buf,int num);
1856 int     SSL_write(SSL *ssl,const void *buf,int num);
1857 long    SSL_ctrl(SSL *ssl,int cmd, long larg, void *parg);
1858 long    SSL_callback_ctrl(SSL *, int, void (*)(void));
1859 long    SSL_CTX_ctrl(SSL_CTX *ctx,int cmd, long larg, void *parg);
1860 long    SSL_CTX_callback_ctrl(SSL_CTX *, int, void (*)(void));

1862 int     SSL_get_error(const SSL *s,int ret_code);
1863 const char *SSL_get_version(const SSL *s);

1865 /* This sets the 'default' SSL version that SSL_new() will create */
1866 int SSL_CTX_set_ssl_version(SSL_CTX *ctx, const SSL_METHOD *meth);

1868 #ifndef OPENSSL_NO_SSL2
1869 const SSL_METHOD *SSLv2_method(void);          /* SSLv2 */
1870 const SSL_METHOD *SSLv2_server_method(void);   /* SSLv2 */
1871 const SSL_METHOD *SSLv2_client_method(void);   /* SSLv2 */
1872 #endif

1874 const SSL_METHOD *SSLv3_method(void);          /* SSLv3 */
1875 const SSL_METHOD *SSLv3_server_method(void);   /* SSLv3 */
1876 const SSL_METHOD *SSLv3_client_method(void);   /* SSLv3 */

1878 const SSL_METHOD *SSLv23_method(void);  /* SSLv3 but can rollback to v2 */
1879 const SSL_METHOD *SSLv23_server_method(void);   /* SSLv3 but can rollback to v2
1880 const SSL_METHOD *SSLv23_client_method(void);   /* SSLv3 but can rollback to v2

1882 const SSL_METHOD *TLSv1_method(void);          /* TLSv1.0 */
1883 const SSL_METHOD *TLSv1_server_method(void);   /* TLSv1.0 */
1884 const SSL_METHOD *TLSv1_client_method(void);   /* TLSv1.0 */

1886 const SSL_METHOD *TLSv1_1_method(void);        /* TLSv1.1 */
1887 const SSL_METHOD *TLSv1_1_server_method(void); /* TLSv1.1 */
1888 const SSL_METHOD *TLSv1_1_client_method(void); /* TLSv1.1 */

1890 const SSL_METHOD *TLSv1_2_method(void);        /* TLSv1.2 */
1891 const SSL_METHOD *TLSv1_2_server_method(void); /* TLSv1.2 */
1892 const SSL_METHOD *TLSv1_2_client_method(void); /* TLSv1.2 */


1895 const SSL_METHOD *DTLSv1_method(void);          /* DTLSv1.0 */
1896 const SSL_METHOD *DTLSv1_server_method(void);   /* DTLSv1.0 */
1897 const SSL_METHOD *DTLSv1_client_method(void);   /* DTLSv1.0 */

1899 STACK_OF(SSL_CIPHER) *SSL_get_ciphers(const SSL *s);

1901 int SSL_do_handshake(SSL *s);
1902 int SSL_renegotiate(SSL *s);
1903 int SSL_renegotiate_abbreviated(SSL *s);
1904 int SSL_renegotiate_pending(SSL *s);
1905 int SSL_shutdown(SSL *s);

1907 const SSL_METHOD *SSL_get_ssl_method(SSL *s);
1908 int SSL_set_ssl_method(SSL *s, const SSL_METHOD *method);
1909 const char *SSL_alert_type_string_long(int value);
```



```
1910 const char *SSL_alert_type_string(int value);
1911 const char *SSL_alert_desc_string_long(int value);
1912 const char *SSL_alert_desc_string(int value);

1914 void SSL_set_client_CA_list(SSL *s, STACK_OF(X509_NAME) *name_list);
1915 void SSL_CTX_set_client_CA_list(SSL_CTX *ctx, STACK_OF(X509_NAME) *name_list);
1916 STACK_OF(X509_NAME) *SSL_get_client_CA_list(const SSL *s);
1917 STACK_OF(X509_NAME) *SSL_CTX_get_client_CA_list(const SSL_CTX *s);
1918 int SSL_add_client_CA(SSL *ssl,X509 *x);
1919 int SSL_CTX_add_client_CA(SSL_CTX *ctx,X509 *x);

1921 void SSL_set_connect_state(SSL *s);
1922 void SSL_set_accept_state(SSL *s);

1924 long SSL_get_default_timeout(const SSL *s);

1926 int SSL_library_init(void );

1928 char *SSL_CIPHER_description(const SSL_CIPHER *,char *buf,int size);
1929 STACK_OF(X509_NAME) *SSL_dup_CA_list(STACK_OF(X509_NAME) *sk);

1931 SSL *SSL_dup(SSL *ssl);

1933 X509 *SSL_get_certificate(const SSL *ssl);
1934 /* EVP_PKEY */ struct evp_pkey_st *SSL_get_privatekey(SSL *ssl);

1936 void SSL_CTX_set_quiet_shutdown(SSL_CTX *ctx,int mode);
1937 int SSL_CTX_get_quiet_shutdown(const SSL_CTX *ctx);
1938 void SSL_set_quiet_shutdown(SSL *ssl,int mode);
1939 int SSL_get_quiet_shutdown(const SSL *ssl);
1940 void SSL_set_shutdown(SSL *ssl,int mode);
1941 int SSL_get_shutdown(const SSL *ssl);
1942 int SSL_version(const SSL *ssl);
1943 int SSL_CTX_set_default_verify_paths(SSL_CTX *ctx);
1944 int SSL_CTX_load_verify_locations(SSL_CTX *ctx, const char *CAfile,
1945         const char *CApath);
1946 #define SSL_get0_session SSL_get_session /* just peek at pointer */
1947 SSL_SESSION *SSL_get_session(const SSL *ssl);
1948 SSL_SESSION *SSL_get1_session(SSL *ssl); /* obtain a reference count */
1949 SSL_CTX *SSL_get_SSL_CTX(const SSL *ssl);
1950 SSL_CTX *SSL_set_SSL_CTX(SSL *ssl, SSL_CTX* ctx);
1951 void SSL_set_info_callback(SSL *ssl,
1952                            void (*cb)(const SSL *ssl,int type,int val));
1953 void (*SSL_get_info_callback(const SSL *ssl))(const SSL *ssl,int type,int val);
1954 int SSL_state(const SSL *ssl);
1955 void SSL_set_state(SSL *ssl, int state);

1957 void SSL_set_verify_result(SSL *ssl,long v);
1958 long SSL_get_verify_result(const SSL *ssl);

1960 int SSL_set_ex_data(SSL *ssl,int idx,void *data);
1961 void *SSL_get_ex_data(const SSL *ssl,int idx);
1962 int SSL_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
1963         CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);

1965 int SSL_SESSION_set_ex_data(SSL_SESSION *ss,int idx,void *data);
1966 void *SSL_SESSION_get_ex_data(const SSL_SESSION *ss,int idx);
1967 int SSL_SESSION_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
1968         CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);

1970 int SSL_CTX_set_ex_data(SSL_CTX *ssl,int idx,void *data);
1971 void *SSL_CTX_get_ex_data(const SSL_CTX *ssl,int idx);
1972 int SSL_CTX_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
1973         CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);

1975 int SSL_get_ex_data_X509_STORE_CTX_idx(void );
```

```
#define SSL_CTX_sess_set_cache_size(ctx,t) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_SESS_CACHE_SIZE,t,NULL)
#define SSL_CTX_sess_get_cache_size(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_GET_SESS_CACHE_SIZE,0,NULL)
#define SSL_CTX_set_session_cache_mode(ctx,m) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_SESS_CACHE_MODE,m,NULL)
#define SSL_CTX_get_session_cache_mode(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_GET_SESS_CACHE_MODE,0,NULL)

#define SSL_CTX_get_default_read_ahead(ctx) SSL_CTX_get_read_ahead(ctx)
#define SSL_CTX_set_default_read_ahead(ctx,m) SSL_CTX_set_read_ahead(ctx,m)
#define SSL_CTX_get_read_ahead(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_GET_READ_AHEAD,0,NULL)
#define SSL_CTX_set_read_ahead(ctx,m) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_READ_AHEAD,m,NULL)
#define SSL_CTX_get_max_cert_list(ctx) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_GET_MAX_CERT_LIST,0,NULL)
#define SSL_CTX_set_max_cert_list(ctx,m) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_MAX_CERT_LIST,m,NULL)
#define SSL_get_max_cert_list(ssl) \
        SSL_ctrl(ssl,SSL_CTRL_GET_MAX_CERT_LIST,0,NULL)
#define SSL_set_max_cert_list(ssl,m) \
        SSL_ctrl(ssl,SSL_CTRL_SET_MAX_CERT_LIST,m,NULL)

#define SSL_CTX_set_max_send_fragment(ctx,m) \
        SSL_CTX_ctrl(ctx,SSL_CTRL_SET_MAX_SEND_FRAGMENT,m,NULL)
#define SSL_set_max_send_fragment(ssl,m) \
        SSL_ctrl(ssl,SSL_CTRL_SET_MAX_SEND_FRAGMENT,m,NULL)

     /* NB: the keylength is only applicable when is_export is true */
#ifndef OPENSSL_NO_RSA
void SSL_CTX_set_tmp_rsa_callback(SSL_CTX *ctx,
                                  RSA *(*cb)(SSL *ssl,int is_export,
                                             int keylength));

void SSL_set_tmp_rsa_callback(SSL *ssl,
                                  RSA *(*cb)(SSL *ssl,int is_export,
                                             int keylength));
#endif
#ifndef OPENSSL_NO_DH
void SSL_CTX_set_tmp_dh_callback(SSL_CTX *ctx,
                                 DH *(*dh)(SSL *ssl,int is_export,
                                           int keylength));
void SSL_set_tmp_dh_callback(SSL *ssl,
                                 DH *(*dh)(SSL *ssl,int is_export,
                                           int keylength));
#endif
#ifndef OPENSSL_NO_ECDH
void SSL_CTX_set_tmp_ecdh_callback(SSL_CTX *ctx,
                                   EC_KEY *(*ecdh)(SSL *ssl,int is_export,
                                             int keylength));
void SSL_set_tmp_ecdh_callback(SSL *ssl,
                                   EC_KEY *(*ecdh)(SSL *ssl,int is_export,
                                             int keylength));
#endif

#ifndef OPENSSL_NO_COMP
const COMP_METHOD *SSL_get_current_compression(SSL *s);
const COMP_METHOD *SSL_get_current_expansion(SSL *s);
const char *SSL_COMP_get_name(const COMP_METHOD *comp);
STACK_OF(SSL_COMP) *SSL_COMP_get_compression_methods(void);
int SSL_COMP_add_compression_method(int id,COMP_METHOD *cm);
#else
const void *SSL_get_current_compression(SSL *s);
const void *SSL_get_current_expansion(SSL *s);
```



```
const char *SSL_COMP_get_name(const void *comp);
void *SSL_COMP_get_compression_methods(void);
int SSL_COMP_add_compression_method(int id,void *cm);
#endif

/* TLS extensions functions */
int SSL_set_session_ticket_ext(SSL *s, void *ext_data, int ext_len);

int SSL_set_session_ticket_ext_cb(SSL *s, tls_session_ticket_ext_cb_fn cb,
                                  void *arg);

/* Pre-shared secret session resumption functions */
int SSL_set_session_secret_cb(SSL *s, tls_session_secret_cb_fn tls_session_secre

void SSL_set_debug(SSL *s, int debug);
int SSL_cache_hit(SSL *s);

#ifndef OPENSSL_NO_UNIT_TEST
const struct openssl_ssl_test_functions *SSL_test_functions(void);
#endif

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_SSL_strings(void);

/* Error codes for the SSL functions. */

/* Function codes. */
#define SSL_F_CLIENT_CERTIFICATE                        100
#define SSL_F_CLIENT_FINISHED                           167
#define SSL_F_CLIENT_HELLO                              101
#define SSL_F_CLIENT_MASTER_KEY                         102
#define SSL_F_D2I_SSL_SESSION                           103
#define SSL_F_DO_DTLS1_WRITE                            245
#define SSL_F_DO_SSL3_WRITE                             104
#define SSL_F_DTLS1_ACCEPT                              246
#define SSL_F_DTLS1_ADD_CERT_TO_BUF                     295
#define SSL_F_DTLS1_BUFFER_RECORD                       247
#define SSL_F_DTLS1_CHECK_TIMEOUT_NUM                   316
#define SSL_F_DTLS1_CLIENT_HELLO                        248
#define SSL_F_DTLS1_CONNECT                             249
#define SSL_F_DTLS1_ENC                                 250
#define SSL_F_DTLS1_GET_HELLO_VERIFY                    251
#define SSL_F_DTLS1_GET_MESSAGE                         252
#define SSL_F_DTLS1_GET_MESSAGE_FRAGMENT                253
#define SSL_F_DTLS1_GET_RECORD                          254
#define SSL_F_DTLS1_HANDLE_TIMEOUT                      297
#define SSL_F_DTLS1_HEARTBEAT                           305
#define SSL_F_DTLS1_OUTPUT_CERT_CHAIN                   255
#define SSL_F_DTLS1_PREPROCESS_FRAGMENT                 288
#define SSL_F_DTLS1_PROCESS_OUT_OF_SEQ_MESSAGE          256
#define SSL_F_DTLS1_PROCESS_RECORD                      257
#define SSL_F_DTLS1_READ_BYTES                          258
#define SSL_F_DTLS1_READ_FAILED                         259
#define SSL_F_DTLS1_SEND_CERTIFICATE_REQUEST            260
#define SSL_F_DTLS1_SEND_CLIENT_CERTIFICATE             261
#define SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE            262
#define SSL_F_DTLS1_SEND_CLIENT_VERIFY                  263
#define SSL_F_DTLS1_SEND_HELLO_VERIFY_REQUEST           264
#define SSL_F_DTLS1_SEND_SERVER_CERTIFICATE             265
#define SSL_F_DTLS1_SEND_SERVER_HELLO                   266
#define SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE            267
#define SSL_F_DTLS1_WRITE_APP_DATA_BYTES                268
#define SSL_F_GET_CLIENT_FINISHED                       105
```

```
2108 #define SSL_F_GET_CLIENT_HELLO                          106
2109 #define SSL_F_GET_CLIENT_MASTER_KEY                     107
2110 #define SSL_F_GET_SERVER_FINISHED                       108
2111 #define SSL_F_GET_SERVER_HELLO                          109
2112 #define SSL_F_GET_SERVER_VERIFY                         110
2113 #define SSL_F_I2D_SSL_SESSION                           111
2114 #define SSL_F_READ_N                                    112
2115 #define SSL_F_REQUEST_CERTIFICATE                       113
2116 #define SSL_F_SERVER_FINISH                             239
2117 #define SSL_F_SERVER_HELLO                              114
2118 #define SSL_F_SERVER_VERIFY                             240
2119 #define SSL_F_SSL23_ACCEPT                              115
2120 #define SSL_F_SSL23_CLIENT_HELLO                        116
2121 #define SSL_F_SSL23_CONNECT                             117
2122 #define SSL_F_SSL23_GET_CLIENT_HELLO                    118
2123 #define SSL_F_SSL23_GET_SERVER_HELLO                    119
2124 #define SSL_F_SSL23_PEEK                                237
2125 #define SSL_F_SSL23_READ                                120
2126 #define SSL_F_SSL23_WRITE                               121
2127 #define SSL_F_SSL2_ACCEPT                               122
2128 #define SSL_F_SSL2_CONNECT                              123
2129 #define SSL_F_SSL2_ENC_INIT                             124
2130 #define SSL_F_SSL2_GENERATE_KEY_MATERIAL                241
2131 #define SSL_F_SSL2_PEEK                                 234
2132 #define SSL_F_SSL2_READ                                 125
2133 #define SSL_F_SSL2_READ_INTERNAL                        236
2134 #define SSL_F_SSL2_SET_CERTIFICATE                      126
2135 #define SSL_F_SSL2_WRITE                                127
2136 #define SSL_F_SSL3_ACCEPT                               128
2137 #define SSL_F_SSL3_ADD_CERT_TO_BUF                      296
2138 #define SSL_F_SSL3_CALLBACK_CTRL                        233
2139 #define SSL_F_SSL3_CHANGE_CIPHER_STATE                  129
2140 #define SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM             130
2141 #define SSL_F_SSL3_CHECK_CLIENT_HELLO                   304
2142 #define SSL_F_SSL3_CLIENT_HELLO                         131
2143 #define SSL_F_SSL3_CONNECT                              132
2144 #define SSL_F_SSL3_CTRL                                 213
2145 #define SSL_F_SSL3_CTX_CTRL                             133
2146 #define SSL_F_SSL3_DIGEST_CACHED_RECORDS                293
2147 #define SSL_F_SSL3_DO_CHANGE_CIPHER_SPEC                292
2148 #define SSL_F_SSL3_ENC                                  134
2149 #define SSL_F_SSL3_GENERATE_KEY_BLOCK                   238
2150 #define SSL_F_SSL3_GET_CERTIFICATE_REQUEST              135
2151 #define SSL_F_SSL3_GET_CERT_STATUS                      289
2152 #define SSL_F_SSL3_GET_CERT_VERIFY                      136
2153 #define SSL_F_SSL3_GET_CLIENT_CERTIFICATE               137
2154 #define SSL_F_SSL3_GET_CLIENT_HELLO                     138
2155 #define SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE              139
2156 #define SSL_F_SSL3_GET_FINISHED                         140
2157 #define SSL_F_SSL3_GET_KEY_EXCHANGE                     141
2158 #define SSL_F_SSL3_GET_MESSAGE                          142
2159 #define SSL_F_SSL3_GET_NEW_SESSION_TICKET               283
2160 #define SSL_F_SSL3_GET_NEXT_PROTO                       306
2161 #define SSL_F_SSL3_GET_RECORD                           143
2162 #define SSL_F_SSL3_GET_SERVER_CERTIFICATE               144
2163 #define SSL_F_SSL3_GET_SERVER_DONE                      145
2164 #define SSL_F_SSL3_GET_SERVER_HELLO                     146
2165 #define SSL_F_SSL3_HANDSHAKE_MAC                        285
2166 #define SSL_F_SSL3_NEW_SESSION_TICKET                   287
2167 #define SSL_F_SSL3_OUTPUT_CERT_CHAIN                    147
2168 #define SSL_F_SSL3_PEEK                                 235
2169 #define SSL_F_SSL3_READ_BYTES                           148
2170 #define SSL_F_SSL3_READ_N                               149
2171 #define SSL_F_SSL3_SEND_CERTIFICATE_REQUEST             150
2172 #define SSL_F_SSL3_SEND_CLIENT_CERTIFICATE              151
2173 #define SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE             152
```



```
2174 #define SSL_F_SSL3_SEND_CLIENT_VERIFY                   153
2175 #define SSL_F_SSL3_SEND_SERVER_CERTIFICATE              154
2176 #define SSL_F_SSL3_SEND_SERVER_HELLO                    242
2177 #define SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE             155
2178 #define SSL_F_SSL3_SETUP_KEY_BLOCK                      157
2179 #define SSL_F_SSL3_SETUP_READ_BUFFER                    156
2180 #define SSL_F_SSL3_SETUP_WRITE_BUFFER                   291
2181 #define SSL_F_SSL3_WRITE_BYTES                          158
2182 #define SSL_F_SSL3_WRITE_PENDING                        159
2183 #define SSL_F_SSL_ADD_CLIENTHELLO_RENEGOTIATE_EXT       298
2184 #define SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT                277
2185 #define SSL_F_SSL_ADD_CLIENTHELLO_USE_SRTP_EXT          307
2186 #define SSL_F_SSL_ADD_DIR_CERT_SUBJECTS_TO_STACK        215
2187 #define SSL_F_SSL_ADD_FILE_CERT_SUBJECTS_TO_STACK       216
2188 #define SSL_F_SSL_ADD_SERVERHELLO_RENEGOTIATE_EXT       299
2189 #define SSL_F_SSL_ADD_SERVERHELLO_TLSEXT                278
2190 #define SSL_F_SSL_ADD_SERVERHELLO_USE_SRTP_EXT          308
2191 #define SSL_F_SSL_BAD_METHOD                            160
2192 #define SSL_F_SSL_BYTES_TO_CIPHER_LIST                  161
2193 #define SSL_F_SSL_CERT_DUP                              221
2194 #define SSL_F_SSL_CERT_INST                             222
2195 #define SSL_F_SSL_CERT_INSTANTIATE                      214
2196 #define SSL_F_SSL_CERT_NEW                              162
2197 #define SSL_F_SSL_CHECK_PRIVATE_KEY                     163
2198 #define SSL_F_SSL_CHECK_SERVERHELLO_TLSEXT              280
2199 #define SSL_F_SSL_CHECK_SRVR_ECC_CERT_AND_ALG           279
2200 #define SSL_F_SSL_CIPHER_PROCESS_RULESTR                230
2201 #define SSL_F_SSL_CIPHER_STRENGTH_SORT                  231
2202 #define SSL_F_SSL_CLEAR                                 164
2203 #define SSL_F_SSL_COMP_ADD_COMPRESSION_METHOD           165
2204 #define SSL_F_SSL_CREATE_CIPHER_LIST                    166
2205 #define SSL_F_SSL_CTRL                                  232
2206 #define SSL_F_SSL_CTX_CHECK_PRIVATE_KEY                 168
2207 #define SSL_F_SSL_CTX_MAKE_PROFILES                     309
2208 #define SSL_F_SSL_CTX_NEW                               169
2209 #define SSL_F_SSL_CTX_SET_CIPHER_LIST                   269
2210 #define SSL_F_SSL_CTX_SET_CLIENT_CERT_ENGINE            290
2211 #define SSL_F_SSL_CTX_SET_PURPOSE                       226
2212 #define SSL_F_SSL_CTX_SET_SESSION_ID_CONTEXT            219
2213 #define SSL_F_SSL_CTX_SET_SSL_VERSION                   170
2214 #define SSL_F_SSL_CTX_SET_TRUST                         229
2215 #define SSL_F_SSL_CTX_USE_CERTIFICATE                   171
2216 #define SSL_F_SSL_CTX_USE_CERTIFICATE_ASN1              172
2217 #define SSL_F_SSL_CTX_USE_CERTIFICATE_CHAIN_FILE        220
2218 #define SSL_F_SSL_CTX_USE_CERTIFICATE_FILE              173
2219 #define SSL_F_SSL_CTX_USE_PRIVATEKEY                    174
2220 #define SSL_F_SSL_CTX_USE_PRIVATEKEY_ASN1               175
2221 #define SSL_F_SSL_CTX_USE_PRIVATEKEY_FILE               176
2222 #define SSL_F_SSL_CTX_USE_PSK_IDENTITY_HINT             272
2223 #define SSL_F_SSL_CTX_USE_RSAPRIVATEKEY                 177
2224 #define SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_ASN1            178
2225 #define SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_FILE            179
2226 #define SSL_F_SSL_DO_HANDSHAKE                          180
2227 #define SSL_F_SSL_GET_NEW_SESSION                       181
2228 #define SSL_F_SSL_GET_PREV_SESSION                      217
2229 #define SSL_F_SSL_GET_SERVER_SEND_CERT                  182
2230 #define SSL_F_SSL_GET_SERVER_SEND_PKEY                  317
2231 #define SSL_F_SSL_GET_SIGN_PKEY                         183
2232 #define SSL_F_SSL_INIT_WBIO_BUFFER                      184
2233 #define SSL_F_SSL_LOAD_CLIENT_CA_FILE                   185
2234 #define SSL_F_SSL_NEW                                   186
2235 #define SSL_F_SSL_PARSE_CLIENTHELLO_RENEGOTIATE_EXT     300
2236 #define SSL_F_SSL_PARSE_CLIENTHELLO_TLSEXT              302
2237 #define SSL_F_SSL_PARSE_CLIENTHELLO_USE_SRTP_EXT        310
2238 #define SSL_F_SSL_PARSE_SERVERHELLO_RENEGOTIATE_EXT     301
2239 #define SSL_F_SSL_PARSE_SERVERHELLO_TLSEXT              303
```

```
2240 #define SSL_F_SSL_PARSE_SERVERHELLO_USE_SRTP_EXT         311
2241 #define SSL_F_SSL_PEEK                                  270
2242 #define SSL_F_SSL_PREPARE_CLIENTHELLO_TLSEXT            281
2243 #define SSL_F_SSL_PREPARE_SERVERHELLO_TLSEXT            282
2244 #define SSL_F_SSL_READ                                  223
2245 #define SSL_F_SSL_RSA_PRIVATE_DECRYPT                   187
2246 #define SSL_F_SSL_RSA_PUBLIC_ENCRYPT                    188
2247 #define SSL_F_SSL_SESSION_NEW                           189
2248 #define SSL_F_SSL_SESSION_PRINT_FP                      190
2249 #define SSL_F_SSL_SESSION_SET1_ID_CONTEXT               312
2250 #define SSL_F_SSL_SESS_CERT_NEW                         225
2251 #define SSL_F_SSL_SET_CERT                              191
2252 #define SSL_F_SSL_SET_CIPHER_LIST                       271
2253 #define SSL_F_SSL_SET_FD                                192
2254 #define SSL_F_SSL_SET_PKEY                              193
2255 #define SSL_F_SSL_SET_PURPOSE                           227
2256 #define SSL_F_SSL_SET_RFD                               194
2257 #define SSL_F_SSL_SET_SESSION                           195
2258 #define SSL_F_SSL_SET_SESSION_ID_CONTEXT                218
2259 #define SSL_F_SSL_SET_SESSION_TICKET_EXT                294
2260 #define SSL_F_SSL_SET_TRUST                             228
2261 #define SSL_F_SSL_SET_WFD                               196
2262 #define SSL_F_SSL_SHUTDOWN                              224
2263 #define SSL_F_SSL_SRP_CTX_INIT                          313
2264 #define SSL_F_SSL_UNDEFINED_CONST_FUNCTION              243
2265 #define SSL_F_SSL_UNDEFINED_FUNCTION                    197
2266 #define SSL_F_SSL_UNDEFINED_VOID_FUNCTION               244
2267 #define SSL_F_SSL_USE_CERTIFICATE                       198
2268 #define SSL_F_SSL_USE_CERTIFICATE_ASN1                  199
2269 #define SSL_F_SSL_USE_CERTIFICATE_FILE                  200
2270 #define SSL_F_SSL_USE_PRIVATEKEY                        201
2271 #define SSL_F_SSL_USE_PRIVATEKEY_ASN1                   202
2272 #define SSL_F_SSL_USE_PRIVATEKEY_FILE                   203
2273 #define SSL_F_SSL_USE_PSK_IDENTITY_HINT                 273
2274 #define SSL_F_SSL_USE_RSAPRIVATEKEY                     204
2275 #define SSL_F_SSL_USE_RSAPRIVATEKEY_ASN1                205
2276 #define SSL_F_SSL_USE_RSAPRIVATEKEY_FILE                206
2277 #define SSL_F_SSL_VERIFY_CERT_CHAIN                     207
2278 #define SSL_F_SSL_WRITE                                 208
2279 #define SSL_F_TLS1_CERT_VERIFY_MAC                      286
2280 #define SSL_F_TLS1_CHANGE_CIPHER_STATE                  209
2281 #define SSL_F_TLS1_CHECK_SERVERHELLO_TLSEXT             274
2282 #define SSL_F_TLS1_ENC                                  210
2283 #define SSL_F_TLS1_EXPORT_KEYING_MATERIAL               314
2284 #define SSL_F_TLS1_HEARTBEAT                            315
2285 #define SSL_F_TLS1_PREPARE_CLIENTHELLO_TLSEXT           275
2286 #define SSL_F_TLS1_PREPARE_SERVERHELLO_TLSEXT           276
2287 #define SSL_F_TLS1_PRF                                  284
2288 #define SSL_F_TLS1_SETUP_KEY_BLOCK                      211
2289 #define SSL_F_WRITE_PENDING                             212

2291 /* Reason codes. */
2292 #define SSL_R_APP_DATA_IN_HANDSHAKE                     100
2293 #define SSL_R_ATTEMPT_TO_REUSE_SESSION_IN_DIFFERENT_CONTEXT 272
2294 #define SSL_R_BAD_ALERT_RECORD                          101
2295 #define SSL_R_BAD_AUTHENTICATION_TYPE                   102
2296 #define SSL_R_BAD_CHANGE_CIPHER_SPEC                    103
2297 #define SSL_R_BAD_CHECKSUM                              104
2298 #define SSL_R_BAD_DATA_RETURNED_BY_CALLBACK             106
2299 #define SSL_R_BAD_DECOMPRESSION                         107
2300 #define SSL_R_BAD_DH_G_LENGTH                           108
2301 #define SSL_R_BAD_DH_PUB_KEY_LENGTH                     109
2302 #define SSL_R_BAD_DH_P_LENGTH                           110
2303 #define SSL_R_BAD_DIGEST_LENGTH                         111
2304 #define SSL_R_BAD_DSA_SIGNATURE                         112
2305 #define SSL_R_BAD_ECC_CERT                              304
```



```
2306 #define SSL_R_BAD_ECDSA_SIGNATURE                       305
2307 #define SSL_R_BAD_ECPOINT                               306
2308 #define SSL_R_BAD_HANDSHAKE_LENGTH                      332
2309 #define SSL_R_BAD_HELLO_REQUEST                         105
2310 #define SSL_R_BAD_LENGTH                                271
2311 #define SSL_R_BAD_MAC_DECODE                            113
2312 #define SSL_R_BAD_MAC_LENGTH                            333
2313 #define SSL_R_BAD_MESSAGE_TYPE                          114
2314 #define SSL_R_BAD_PACKET_LENGTH                         115
2315 #define SSL_R_BAD_PROTOCOL_VERSION_NUMBER               116
2316 #define SSL_R_BAD_PSK_IDENTITY_HINT_LENGTH              316
2317 #define SSL_R_BAD_RESPONSE_ARGUMENT                     117
2318 #define SSL_R_BAD_RSA_DECRYPT                           118
2319 #define SSL_R_BAD_RSA_ENCRYPT                           119
2320 #define SSL_R_BAD_RSA_E_LENGTH                          120
2321 #define SSL_R_BAD_RSA_MODULUS_LENGTH                    121
2322 #define SSL_R_BAD_RSA_SIGNATURE                         122
2323 #define SSL_R_BAD_SIGNATURE                             123
2324 #define SSL_R_BAD_SRP_A_LENGTH                          347
2325 #define SSL_R_BAD_SRP_B_LENGTH                          348
2326 #define SSL_R_BAD_SRP_G_LENGTH                          349
2327 #define SSL_R_BAD_SRP_N_LENGTH                          350
2328 #define SSL_R_BAD_SRP_PARAMETERS                        371
2329 #define SSL_R_BAD_SRP_S_LENGTH                          351
2330 #define SSL_R_BAD_SRTP_MKI_VALUE                        352
2331 #define SSL_R_BAD_SRTP_PROTECTION_PROFILE_LIST          353
2332 #define SSL_R_BAD_SSL_FILETYPE                          124
2333 #define SSL_R_BAD_SSL_SESSION_ID_LENGTH                 125
2334 #define SSL_R_BAD_STATE                                 126
2335 #define SSL_R_BAD_WRITE_RETRY                           127
2336 #define SSL_R_BIO_NOT_SET                               128
2337 #define SSL_R_BLOCK_CIPHER_PAD_IS_WRONG                 129
2338 #define SSL_R_BN_LIB                                    130
2339 #define SSL_R_CA_DN_LENGTH_MISMATCH                     131
2340 #define SSL_R_CA_DN_TOO_LONG                            132
2341 #define SSL_R_CCS_RECEIVED_EARLY                        133
2342 #define SSL_R_CERTIFICATE_VERIFY_FAILED                 134
2343 #define SSL_R_CERT_LENGTH_MISMATCH                      135
2344 #define SSL_R_CHALLENGE_IS_DIFFERENT                    136
2345 #define SSL_R_CIPHER_CODE_WRONG_LENGTH                  137
2346 #define SSL_R_CIPHER_OR_HASH_UNAVAILABLE                138
2347 #define SSL_R_CIPHER_TABLE_SRC_ERROR                    139
2348 #define SSL_R_CLIENTHELLO_TLSEXT                        226
2349 #define SSL_R_COMPRESSED_LENGTH_TOO_LONG                140
2350 #define SSL_R_COMPRESSION_DISABLED                      343
2351 #define SSL_R_COMPRESSION_FAILURE                       141
2352 #define SSL_R_COMPRESSION_ID_NOT_WITHIN_PRIVATE_RANGE   307
2353 #define SSL_R_COMPRESSION_LIBRARY_ERROR                 142
2354 #define SSL_R_CONNECTION_ID_IS_DIFFERENT                143
2355 #define SSL_R_CONNECTION_TYPE_NOT_SET                   144
2356 #define SSL_R_COOKIE_MISMATCH                           308
2357 #define SSL_R_DATA_BETWEEN_CCS_AND_FINISHED             145
2358 #define SSL_R_DATA_LENGTH_TOO_LONG                      146
2359 #define SSL_R_DECRYPTION_FAILED                         147
2360 #define SSL_R_DECRYPTION_FAILED_OR_BAD_RECORD_MAC       281
2361 #define SSL_R_DH_PUBLIC_VALUE_LENGTH_IS_WRONG           148
2362 #define SSL_R_DIGEST_CHECK_FAILED                       149
2363 #define SSL_R_DTLS_MESSAGE_TOO_BIG                      334
2364 #define SSL_R_DUPLICATE_COMPRESSION_ID                  309
2365 #define SSL_R_ECC_CERT_NOT_FOR_KEY_AGREEMENT            317
2366 #define SSL_R_ECC_CERT_NOT_FOR_SIGNING                  318
2367 #define SSL_R_ECC_CERT_SHOULD_HAVE_RSA_SIGNATURE        322
2368 #define SSL_R_ECC_CERT_SHOULD_HAVE_SHA1_SIGNATURE       323
2369 #define SSL_R_ECGROUP_TOO_LARGE_FOR_CIPHER              310
2370 #define SSL_R_EMPTY_SRTP_PROTECTION_PROFILE_LIST        354
2371 #define SSL_R_ENCRYPTED_LENGTH_TOO_LONG                 150
```

```
#define SSL_R_ERROR_GENERATING_TMP_RSA_KEY 282
#define SSL_R_ERROR_IN_RECEIVED_CIPHER_LIST 151
#define SSL_R_EXCESSIVE_MESSAGE_SIZE 152
#define SSL_R_EXTRA_DATA_IN_MESSAGE 153
#define SSL_R_GOT_A_FIN_BEFORE_A_CCS 154
#define SSL_R_GOT_NEXT_PROTO_BEFORE_A_CCS 355
#define SSL_R_GOT_NEXT_PROTO_WITHOUT_EXTENSION 356
#define SSL_R_HTTPS_PROXY_REQUEST 155
#define SSL_R_HTTP_REQUEST 156
#define SSL_R_ILLEGAL_PADDING 283
#define SSL_R_INCONSISTENT_COMPRESSION 340
#define SSL_R_INVALID_CHALLENGE_LENGTH 158
#define SSL_R_INVALID_COMMAND 280
#define SSL_R_INVALID_COMPRESSION_ALGORITHM 341
#define SSL_R_INVALID_PURPOSE 278
#define SSL_R_INVALID_SRP_USERNAME 357
#define SSL_R_INVALID_STATUS_RESPONSE 328
#define SSL_R_INVALID_TICKET_KEYS_LENGTH 325
#define SSL_R_INVALID_TRUST 279
#define SSL_R_KEY_ARG_TOO_LONG 284
#define SSL_R_KRB5 285
#define SSL_R_KRB5_C_CC_PRINC 286
#define SSL_R_KRB5_C_GET_CRED 287
#define SSL_R_KRB5_C_INIT 288
#define SSL_R_KRB5_C_MK_REQ 289
#define SSL_R_KRB5_S_BAD_TICKET 290
#define SSL_R_KRB5_S_INIT 291
#define SSL_R_KRB5_S_RD_REQ 292
#define SSL_R_KRB5_S_TKT_EXPIRED 293
#define SSL_R_KRB5_S_TKT_NYV 294
#define SSL_R_KRB5_S_TKT_SKEW 295
#define SSL_R_LENGTH_MISMATCH 159
#define SSL_R_LENGTH_TOO_SHORT 160
#define SSL_R_LIBRARY_BUG 274
#define SSL_R_LIBRARY_HAS_NO_CIPHERS 161
#define SSL_R_MESSAGE_TOO_LONG 296
#define SSL_R_MISSING_DH_DSA_CERT 162
#define SSL_R_MISSING_DH_KEY 163
#define SSL_R_MISSING_DH_RSA_CERT 164
#define SSL_R_MISSING_DSA_SIGNING_CERT 165
#define SSL_R_MISSING_EXPORT_TMP_DH_KEY 166
#define SSL_R_MISSING_EXPORT_TMP_RSA_KEY 167
#define SSL_R_MISSING_RSA_CERTIFICATE 168
#define SSL_R_MISSING_RSA_ENCRYPTING_CERT 169
#define SSL_R_MISSING_RSA_SIGNING_CERT 170
#define SSL_R_MISSING_SRP_PARAM 358
#define SSL_R_MISSING_TMP_DH_KEY 171
#define SSL_R_MISSING_TMP_ECDH_KEY 311
#define SSL_R_MISSING_TMP_RSA_KEY 172
#define SSL_R_MISSING_TMP_RSA_PKEY 173
#define SSL_R_MISSING_VERIFY_MESSAGE 174
#define SSL_R_MULTIPLE_SGC_RESTARTS 346
#define SSL_R_NON_SSLV2_INITIAL_PACKET 175
#define SSL_R_NO_CERTIFICATES_RETURNED 176
#define SSL_R_NO_CERTIFICATE_ASSIGNED 177
#define SSL_R_NO_CERTIFICATE_RETURNED 178
#define SSL_R_NO_CERTIFICATE_SET 179
#define SSL_R_NO_CERTIFICATE_SPECIFIED 180
#define SSL_R_NO_CIPHERS_AVAILABLE 181
#define SSL_R_NO_CIPHERS_PASSED 182
#define SSL_R_NO_CIPHERS_SPECIFIED 183
#define SSL_R_NO_CIPHER_LIST 184
#define SSL_R_NO_CIPHER_MATCH 185
#define SSL_R_NO_CLIENT_CERT_METHOD 331
#define SSL_R_NO_CLIENT_CERT_RECEIVED 186
#define SSL_R_NO_COMPRESSION_SPECIFIED 187
```



```
#define SSL_R_NO_GOST_CERTIFICATE_SENT_BY_PEER 330
#define SSL_R_NO_METHOD_SPECIFIED 188
#define SSL_R_NO_PRIVATEKEY 189
#define SSL_R_NO_PRIVATE_KEY_ASSIGNED 190
#define SSL_R_NO_PROTOCOLS_AVAILABLE 191
#define SSL_R_NO_PUBLICKEY 192
#define SSL_R_NO_RENEGOTIATION 339
#define SSL_R_NO_REQUIRED_DIGEST 324
#define SSL_R_NO_SHARED_CIPHER 193
#define SSL_R_NO_SRTP_PROFILES 359
#define SSL_R_NO_VERIFY_CALLBACK 194
#define SSL_R_NULL_SSL_CTX 195
#define SSL_R_NULL_SSL_METHOD_PASSED 196
#define SSL_R_OLD_SESSION_CIPHER_NOT_RETURNED 197
#define SSL_R_OLD_SESSION_COMPRESSION_ALGORITHM_NOT_RETURNED 344
#define SSL_R_ONLY_TLS_ALLOWED_IN_FIPS_MODE 297
#define SSL_R_OPAQUE_PRF_INPUT_TOO_LONG 327
#define SSL_R_PACKET_LENGTH_TOO_LONG 198
#define SSL_R_PARSE_TLSEXT 227
#define SSL_R_PATH_TOO_LONG 270
#define SSL_R_PEER_DID_NOT_RETURN_A_CERTIFICATE 199
#define SSL_R_PEER_ERROR 200
#define SSL_R_PEER_ERROR_CERTIFICATE 201
#define SSL_R_PEER_ERROR_NO_CERTIFICATE 202
#define SSL_R_PEER_ERROR_NO_CIPHER 203
#define SSL_R_PEER_ERROR_UNSUPPORTED_CERTIFICATE_TYPE 204
#define SSL_R_PRE_MAC_LENGTH_TOO_LONG 205
#define SSL_R_PROBLEMS_MAPPING_CIPHER_FUNCTIONS 206
#define SSL_R_PROTOCOL_IS_SHUTDOWN 207
#define SSL_R_PSK_IDENTITY_NOT_FOUND 223
#define SSL_R_PSK_NO_CLIENT_CB 224
#define SSL_R_PSK_NO_SERVER_CB 225
#define SSL_R_PUBLIC_KEY_ENCRYPT_ERROR 208
#define SSL_R_PUBLIC_KEY_IS_NOT_RSA 209
#define SSL_R_PUBLIC_KEY_NOT_RSA 210
#define SSL_R_READ_BIO_NOT_SET 211
#define SSL_R_READ_TIMEOUT_EXPIRED 312
#define SSL_R_READ_WRONG_PACKET_TYPE 212
#define SSL_R_RECORD_LENGTH_MISMATCH 213
#define SSL_R_RECORD_TOO_LARGE 214
#define SSL_R_RECORD_TOO_SMALL 298
#define SSL_R_RENEGOTIATE_EXT_TOO_LONG 335
#define SSL_R_RENEGOTIATION_ENCODING_ERR 336
#define SSL_R_RENEGOTIATION_MISMATCH 337
#define SSL_R_REQUIRED_CIPHER_MISSING 215
#define SSL_R_REQUIRED_COMPRESSSION_ALGORITHM_MISSING 342
#define SSL_R_REUSE_CERT_LENGTH_NOT_ZERO 216
#define SSL_R_REUSE_CERT_TYPE_NOT_ZERO 217
#define SSL_R_REUSE_CIPHER_LIST_NOT_ZERO 218
#define SSL_R_SCSV_RECEIVED_WHEN_RENEGOTIATING 345
#define SSL_R_SERVERHELLO_TLSEXT 275
#define SSL_R_SESSION_ID_CONTEXT_UNINITIALIZED 277
#define SSL_R_SHORT_READ 219
#define SSL_R_SIGNATURE_ALGORITHMS_ERROR 360
#define SSL_R_SIGNATURE_FOR_NON_SIGNING_CERTIFICATE 220
#define SSL_R_SRP_A_CALC 361
#define SSL_R_SRTP_COULD_NOT_ALLOCATE_PROFILES 362
#define SSL_R_SRTP_PROTECTION_PROFILE_LIST_TOO_LONG 363
#define SSL_R_SRTP_UNKNOWN_PROTECTION_PROFILE 364
#define SSL_R_SSL23_DOING_SESSION_ID_REUSE 221
#define SSL_R_SSL2_CONNECTION_ID_TOO_LONG 299
#define SSL_R_SSL3_EXT_INVALID_ECPOINTFORMAT 321
#define SSL_R_SSL3_EXT_INVALID_SERVERNAME 319
#define SSL_R_SSL3_EXT_INVALID_SERVERNAME_TYPE 320
#define SSL_R_SSL3_SESSION_ID_TOO_LONG 300
#define SSL_R_SSL3_SESSION_ID_TOO_SHORT 222
```

```
#define SSL_R_SSLV3_ALERT_BAD_CERTIFICATE                1042
#define SSL_R_SSLV3_ALERT_BAD_RECORD_MAC                 1020
#define SSL_R_SSLV3_ALERT_CERTIFICATE_EXPIRED            1045
#define SSL_R_SSLV3_ALERT_CERTIFICATE_REVOKED            1044
#define SSL_R_SSLV3_ALERT_CERTIFICATE_UNKNOWN            1046
#define SSL_R_SSLV3_ALERT_DECOMPRESSION_FAILURE          1030
#define SSL_R_SSLV3_ALERT_HANDSHAKE_FAILURE              1040
#define SSL_R_SSLV3_ALERT_ILLEGAL_PARAMETER              1047
#define SSL_R_SSLV3_ALERT_NO_CERTIFICATE                 1041
#define SSL_R_SSLV3_ALERT_UNEXPECTED_MESSAGE             1010
#define SSL_R_SSLV3_ALERT_UNSUPPORTED_CERTIFICATE        1043
#define SSL_R_SSL_CTX_HAS_NO_DEFAULT_SSL_VERSION         228
#define SSL_R_SSL_HANDSHAKE_FAILURE                      229
#define SSL_R_SSL_LIBRARY_HAS_NO_CIPHERS                 230
#define SSL_R_SSL_SESSION_ID_CALLBACK_FAILED             301
#define SSL_R_SSL_SESSION_ID_CONFLICT                    302
#define SSL_R_SSL_SESSION_ID_CONTEXT_TOO_LONG            273
#define SSL_R_SSL_SESSION_ID_HAS_BAD_LENGTH              303
#define SSL_R_SSL_SESSION_ID_IS_DIFFERENT                231
#define SSL_R_TLSV1_ALERT_ACCESS_DENIED                  1049
#define SSL_R_TLSV1_ALERT_DECODE_ERROR                   1050
#define SSL_R_TLSV1_ALERT_DECRYPTION_FAILED              1021
#define SSL_R_TLSV1_ALERT_DECRYPT_ERROR                  1051
#define SSL_R_TLSV1_ALERT_EXPORT_RESTRICTION             1060
#define SSL_R_TLSV1_ALERT_INSUFFICIENT_SECURITY          1071
#define SSL_R_TLSV1_ALERT_INTERNAL_ERROR                 1080
#define SSL_R_TLSV1_ALERT_NO_RENEGOTIATION               1100
#define SSL_R_TLSV1_ALERT_PROTOCOL_VERSION               1070
#define SSL_R_TLSV1_ALERT_RECORD_OVERFLOW                1022
#define SSL_R_TLSV1_ALERT_UNKNOWN_CA                     1048
#define SSL_R_TLSV1_ALERT_USER_CANCELLED                 1090
#define SSL_R_TLSV1_BAD_CERTIFICATE_HASH_VALUE           1114
#define SSL_R_TLSV1_BAD_CERTIFICATE_STATUS_RESPONSE      1113
#define SSL_R_TLSV1_CERTIFICATE_UNOBTAINABLE             1111
#define SSL_R_TLSV1_UNRECOGNIZED_NAME                    1112
#define SSL_R_TLSV1_UNSUPPORTED_EXTENSION                1110
#define SSL_R_TLS_CLIENT_CERT_REQ_WITH_ANON_CIPHER       232
#define SSL_R_TLS_HEARTBEAT_PEER_DOESNT_ACCEPT           365
#define SSL_R_TLS_HEARTBEAT_PENDING                      366
#define SSL_R_TLS_ILLEGAL_EXPORTER_LABEL                 367
#define SSL_R_TLS_INVALID_ECPOINTFORMAT_LIST             157
#define SSL_R_TLS_PEER_DID_NOT_RESPOND_WITH_CERTIFICATE_LIST 233
#define SSL_R_TLS_RSA_ENCRYPTED_VALUE_LENGTH_IS_WRONG    234
#define SSL_R_TRIED_TO_USE_UNSUPPORTED_CIPHER            235
#define SSL_R_UNABLE_TO_DECODE_DH_CERTS                  236
#define SSL_R_UNABLE_TO_DECODE_ECDH_CERTS                313
#define SSL_R_UNABLE_TO_EXTRACT_PUBLIC_KEY               237
#define SSL_R_UNABLE_TO_FIND_DH_PARAMETERS               238
#define SSL_R_UNABLE_TO_FIND_ECDH_PARAMETERS             314
#define SSL_R_UNABLE_TO_FIND_PUBLIC_KEY_PARAMETERS       239
#define SSL_R_UNABLE_TO_FIND_SSL_METHOD                  240
#define SSL_R_UNABLE_TO_LOAD_SSL2_MD5_ROUTINES           241
#define SSL_R_UNABLE_TO_LOAD_SSL3_MD5_ROUTINES           242
#define SSL_R_UNABLE_TO_LOAD_SSL3_SHA1_ROUTINES          243
#define SSL_R_UNEXPECTED_MESSAGE                         244
#define SSL_R_UNEXPECTED_RECORD                          245
#define SSL_R_UNINITIALIZED                              276
#define SSL_R_UNKNOWN_ALERT_TYPE                         246
#define SSL_R_UNKNOWN_CERTIFICATE_TYPE                   247
#define SSL_R_UNKNOWN_CIPHER_RETURNED                    248
#define SSL_R_UNKNOWN_CIPHER_TYPE                        249
#define SSL_R_UNKNOWN_DIGEST                             368
#define SSL_R_UNKNOWN_KEY_EXCHANGE_TYPE                  250
#define SSL_R_UNKNOWN_PKEY_TYPE                          251
#define SSL_R_UNKNOWN_PROTOCOL                           252
#define SSL_R_UNKNOWN_REMOTE_ERROR_TYPE                  253
```



```
#define SSL_R_UNKNOWN_SSL_VERSION                        254
#define SSL_R_UNKNOWN_STATE                              255
#define SSL_R_UNSAFE_LEGACY_RENEGOTIATION_DISABLED       338
#define SSL_R_UNSUPPORTED_CIPHER                         256
#define SSL_R_UNSUPPORTED_COMPRESSION_ALGORITHM          257
#define SSL_R_UNSUPPORTED_DIGEST_TYPE                    326
#define SSL_R_UNSUPPORTED_ELLIPTIC_CURVE                 315
#define SSL_R_UNSUPPORTED_PROTOCOL                       258
#define SSL_R_UNSUPPORTED_SSL_VERSION                    259
#define SSL_R_UNSUPPORTED_STATUS_TYPE                    329
#define SSL_R_USE_SRTP_NOT_NEGOTIATED                    369
#define SSL_R_WRITE_BIO_NOT_SET                          260
#define SSL_R_WRONG_CIPHER_RETURNED                      261
#define SSL_R_WRONG_MESSAGE_TYPE                         262
#define SSL_R_WRONG_NUMBER_OF_KEY_BITS                   263
#define SSL_R_WRONG_SIGNATURE_LENGTH                     264
#define SSL_R_WRONG_SIGNATURE_SIZE                       265
#define SSL_R_WRONG_SIGNATURE_TYPE                       370
#define SSL_R_WRONG_SSL_VERSION                          266
#define SSL_R_WRONG_VERSION_NUMBER                       267
#define SSL_R_X509_LIB                                   268
#define SSL_R_X509_VERIFICATION_SETUP_PROBLEMS           269

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   10742 Wed Aug 13 19:51:49 2014
new/usr/src/lib/openssl/include/openssl/ssl2.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl2.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_SSL2_H
#define HEADER_SSL2_H
```



```
#ifdef  __cplusplus
extern "C" {
#endif

/* Protocol Version Codes */
#define SSL2_VERSION            0x0002
#define SSL2_VERSION_MAJOR      0x00
#define SSL2_VERSION_MINOR      0x02
/* #define SSL2_CLIENT_VERSION  0x0002 */
/* #define SSL2_SERVER_VERSION  0x0002 */

/* Protocol Message Codes */
#define SSL2_MT_ERROR                   0
#define SSL2_MT_CLIENT_HELLO            1
#define SSL2_MT_CLIENT_MASTER_KEY       2
#define SSL2_MT_CLIENT_FINISHED         3
#define SSL2_MT_SERVER_HELLO            4
#define SSL2_MT_SERVER_VERIFY           5
#define SSL2_MT_SERVER_FINISHED         6
#define SSL2_MT_REQUEST_CERTIFICATE     7
#define SSL2_MT_CLIENT_CERTIFICATE      8

/* Error Message Codes */
#define SSL2_PE_UNDEFINED_ERROR         0x0000
#define SSL2_PE_NO_CIPHER               0x0001
#define SSL2_PE_NO_CERTIFICATE          0x0002
#define SSL2_PE_BAD_CERTIFICATE         0x0004
#define SSL2_PE_UNSUPPORTED_CERTIFICATE_TYPE 0x0006

/* Cipher Kind Values */
#define SSL2_CK_NULL_WITH_MD5                   0x02000000 /* v3 */
#define SSL2_CK_RC4_128_WITH_MD5                0x02010080
#define SSL2_CK_RC4_128_EXPORT40_WITH_MD5       0x02020080
#define SSL2_CK_RC2_128_CBC_WITH_MD5            0x02030080
#define SSL2_CK_RC2_128_CBC_EXPORT40_WITH_MD5   0x02040080
#define SSL2_CK_IDEA_128_CBC_WITH_MD5           0x02050080
#define SSL2_CK_DES_64_CBC_WITH_MD5             0x02060040
#define SSL2_CK_DES_64_CBC_WITH_SHA             0x02060140 /* v3 */
#define SSL2_CK_DES_192_EDE3_CBC_WITH_MD5       0x020700c0
#define SSL2_CK_DES_192_EDE3_CBC_WITH_SHA       0x020701c0 /* v3 */
#define SSL2_CK_RC4_64_WITH_MD5                 0x02080080 /* MS hack */

#define SSL2_CK_DES_64_CFB64_WITH_MD5_1         0x02ff0800 /* SSLeay */
#define SSL2_CK_NULL                            0x02ff0810 /* SSLeay */

#define SSL2_TXT_DES_64_CFB64_WITH_MD5_1        "DES-CFB-M1"
#define SSL2_TXT_NULL_WITH_MD5                  "NULL-MD5"
#define SSL2_TXT_RC4_128_WITH_MD5               "RC4-MD5"
#define SSL2_TXT_RC4_128_EXPORT40_WITH_MD5      "EXP-RC4-MD5"
#define SSL2_TXT_RC2_128_CBC_WITH_MD5           "RC2-CBC-MD5"
#define SSL2_TXT_RC2_128_CBC_EXPORT40_WITH_MD5  "EXP-RC2-CBC-MD5"
#define SSL2_TXT_IDEA_128_CBC_WITH_MD5          "IDEA-CBC-MD5"
#define SSL2_TXT_DES_64_CBC_WITH_MD5            "DES-CBC-MD5"
#define SSL2_TXT_DES_64_CBC_WITH_SHA            "DES-CBC-SHA"
#define SSL2_TXT_DES_192_EDE3_CBC_WITH_MD5      "DES-CBC3-MD5"
#define SSL2_TXT_DES_192_EDE3_CBC_WITH_SHA      "DES-CBC3-SHA"
#define SSL2_TXT_RC4_64_WITH_MD5                "RC4-64-MD5"

#define SSL2_TXT_NULL                           "NULL"

/* Flags for the SSL_CIPHER.algorithm2 field */
#define SSL2_CF_5_BYTE_ENC                      0x01
#define SSL2_CF_8_BYTE_ENC                      0x02

/* Certificate Type Codes */
#define SSL2_CT_X509_CERTIFICATE                0x01
```

```
/* Authentication Type Code */
#define SSL2_AT_MD5_WITH_RSA_ENCRYPTION         0x01

#define SSL2_MAX_SSL_SESSION_ID_LENGTH          32

/* Upper/Lower Bounds */
#define SSL2_MAX_MASTER_KEY_LENGTH_IN_BITS      256
#ifdef OPENSSL_SYS_MPE
#define SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER    29998u
#else
#define SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER    32767u  /* 2^15-1 */
#endif
#define SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER    16383 /* 2^14-1 */

#define SSL2_CHALLENGE_LENGTH   16
/*#define SSL2_CHALLENGE_LENGTH 32 */
#define SSL2_MIN_CHALLENGE_LENGTH       16
#define SSL2_MAX_CHALLENGE_LENGTH       32
#define SSL2_CONNECTION_ID_LENGTH       16
#define SSL2_MAX_CONNECTION_ID_LENGTH   16
#define SSL2_SSL_SESSION_ID_LENGTH      16
#define SSL2_MAX_CERT_CHALLENGE_LENGTH  32
#define SSL2_MIN_CERT_CHALLENGE_LENGTH  16
#define SSL2_MAX_KEY_MATERIAL_LENGTH    24

#ifndef HEADER_SSL_LOCL_H
#define  CERT           char
#endif

#ifndef OPENSSL_NO_SSL_INTERN

typedef struct ssl2_state_st
        {
        int three_byte_header;
        int clear_text;         /* clear text */
        int escape;             /* not used in SSLv2 */
        int ssl2_rollback;      /* used if SSLv23 rolled back to SSLv2 */

        /* non-blocking io info, used to make sure the same
         * args were passwd */
        unsigned int wnum;      /* number of bytes sent so far */
        int wpend_tot;
        const unsigned char *wpend_buf;

        int wpend_off;  /* offset to data to write */
        int wpend_len;  /* number of bytes passwd to write */
        int wpend_ret;  /* number of bytes to return to caller */

        /* buffer raw data */
        int rbuf_left;
        int rbuf_offs;
        unsigned char *rbuf;
        unsigned char *wbuf;

        unsigned char *write_ptr;/* used to point to the start due to
                                  * 2/3 byte header. */

        unsigned int padding;
        unsigned int rlength; /* passed to ssl2_enc */
        int ract_data_length; /* Set when things are encrypted. */
        unsigned int wlength; /* passed to ssl2_enc */
        int wact_data_length; /* Set when things are decrypted. */
        unsigned char *ract_data;
        unsigned char *wact_data;
        unsigned char *mac_data;
```



```
        unsigned char *read_key;
        unsigned char *write_key;

                /* Stuff specifically to do with this SSL session */
        unsigned int challenge_length;
        unsigned char challenge[SSL2_MAX_CHALLENGE_LENGTH];
        unsigned int conn_id_length;
        unsigned char conn_id[SSL2_MAX_CONNECTION_ID_LENGTH];
        unsigned int key_material_length;
        unsigned char key_material[SSL2_MAX_KEY_MATERIAL_LENGTH*2];

        unsigned long read_sequence;
        unsigned long write_sequence;

        struct  {
                unsigned int conn_id_length;
                unsigned int cert_type;
                unsigned int cert_length;
                unsigned int csl;
                unsigned int clear;
                unsigned int enc;
                unsigned char ccl[SSL2_MAX_CERT_CHALLENGE_LENGTH];
                unsigned int cipher_spec_length;
                unsigned int session_id_length;
                unsigned int clen;
                unsigned int rlen;
                } tmp;
        } SSL2_STATE;

#endif

/* SSLv2 */
/* client */
#define SSL2_ST_SEND_CLIENT_HELLO_A             (0x10|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_HELLO_B             (0x11|SSL_ST_CONNECT)
#define SSL2_ST_GET_SERVER_HELLO_A              (0x20|SSL_ST_CONNECT)
#define SSL2_ST_GET_SERVER_HELLO_B              (0x21|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_MASTER_KEY_A        (0x30|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_MASTER_KEY_B        (0x31|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_FINISHED_A          (0x40|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_FINISHED_B          (0x41|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_CERTIFICATE_A       (0x50|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_CERTIFICATE_B       (0x51|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_CERTIFICATE_C       (0x52|SSL_ST_CONNECT)
#define SSL2_ST_SEND_CLIENT_CERTIFICATE_D       (0x53|SSL_ST_CONNECT)
#define SSL2_ST_GET_SERVER_VERIFY_A             (0x60|SSL_ST_CONNECT)
#define SSL2_ST_GET_SERVER_VERIFY_B             (0x61|SSL_ST_CONNECT)
#define SSL2_ST_GET_SERVER_FINISHED_A           (0x70|SSL_ST_CONNECT)
#define SSL2_ST_GET_SERVER_FINISHED_B           (0x71|SSL_ST_CONNECT)
#define SSL2_ST_CLIENT_START_ENCRYPTION         (0x80|SSL_ST_CONNECT)
#define SSL2_ST_X509_GET_CLIENT_CERTIFICATE     (0x90|SSL_ST_CONNECT)
/* server */
#define SSL2_ST_GET_CLIENT_HELLO_A              (0x10|SSL_ST_ACCEPT)
#define SSL2_ST_GET_CLIENT_HELLO_B              (0x11|SSL_ST_ACCEPT)
#define SSL2_ST_GET_CLIENT_HELLO_C              (0x12|SSL_ST_ACCEPT)
#define SSL2_ST_SEND_SERVER_HELLO_A             (0x20|SSL_ST_ACCEPT)
#define SSL2_ST_SEND_SERVER_HELLO_B             (0x21|SSL_ST_ACCEPT)
#define SSL2_ST_GET_CLIENT_MASTER_KEY_A         (0x30|SSL_ST_ACCEPT)
#define SSL2_ST_GET_CLIENT_MASTER_KEY_B         (0x31|SSL_ST_ACCEPT)
#define SSL2_ST_SEND_SERVER_VERIFY_A            (0x40|SSL_ST_ACCEPT)
#define SSL2_ST_SEND_SERVER_VERIFY_B            (0x41|SSL_ST_ACCEPT)
#define SSL2_ST_SEND_SERVER_VERIFY_C            (0x42|SSL_ST_ACCEPT)
#define SSL2_ST_GET_CLIENT_FINISHED_A           (0x50|SSL_ST_ACCEPT)
#define SSL2_ST_GET_CLIENT_FINISHED_B           (0x51|SSL_ST_ACCEPT)
#define SSL2_ST_SEND_SERVER_FINISHED_A          (0x60|SSL_ST_ACCEPT)
```

```
260 #define SSL2_ST_SEND_SERVER_FINISHED_B          (0x61|SSL_ST_ACCEPT)
261 #define SSL2_ST_SEND_REQUEST_CERTIFICATE_A      (0x70|SSL_ST_ACCEPT)
262 #define SSL2_ST_SEND_REQUEST_CERTIFICATE_B      (0x71|SSL_ST_ACCEPT)
263 #define SSL2_ST_SEND_REQUEST_CERTIFICATE_C      (0x72|SSL_ST_ACCEPT)
264 #define SSL2_ST_SEND_REQUEST_CERTIFICATE_D      (0x73|SSL_ST_ACCEPT)
265 #define SSL2_ST_SERVER_START_ENCRYPTION         (0x80|SSL_ST_ACCEPT)
266 #define SSL2_ST_X509_GET_SERVER_CERTIFICATE     (0x90|SSL_ST_ACCEPT)

268 #ifdef  __cplusplus
269 }
270 #endif
271 #endif
272 #endif /* ! codereview */
```

```
**********************************************************
    3735 Wed Aug 13 19:51:49 2014
new/usr/src/lib/openssl/include/openssl/ssl23.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl23.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_SSL23_H
#define HEADER_SSL23_H
```



```
#ifdef  __cplusplus
extern "C" {
#endif

/*client */
/* write to server */
#define SSL23_ST_CW_CLNT_HELLO_A        (0x210|SSL_ST_CONNECT)
#define SSL23_ST_CW_CLNT_HELLO_B        (0x211|SSL_ST_CONNECT)
/* read from server */
#define SSL23_ST_CR_SRVR_HELLO_A        (0x220|SSL_ST_CONNECT)
#define SSL23_ST_CR_SRVR_HELLO_B        (0x221|SSL_ST_CONNECT)

/* server */
/* read from client */
#define SSL23_ST_SR_CLNT_HELLO_A        (0x210|SSL_ST_ACCEPT)
#define SSL23_ST_SR_CLNT_HELLO_B        (0x211|SSL_ST_ACCEPT)

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   27096 Wed Aug 13 19:51:49 2014
new/usr/src/lib/openssl/include/openssl/ssl3.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl3.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECC cipher suite support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */

#ifndef HEADER_SSL3_H
#define HEADER_SSL3_H

#ifndef OPENSSL_NO_COMP
#include <openssl/comp.h>
#endif
#include <openssl/buffer.h>
#include <openssl/evp.h>
#include <openssl/ssl.h>

#ifdef  __cplusplus
```

```
128 extern "C" {
129 #endif

131 /* Signalling cipher suite value: from draft-ietf-tls-renegotiation-03.txt */
132 #define SSL3_CK_SCSV                            0x030000FF

134 #define SSL3_CK_RSA_NULL_MD5                    0x03000001
135 #define SSL3_CK_RSA_NULL_SHA                    0x03000002
136 #define SSL3_CK_RSA_RC4_40_MD5                  0x03000003
137 #define SSL3_CK_RSA_RC4_128_MD5                 0x03000004
138 #define SSL3_CK_RSA_RC4_128_SHA                 0x03000005
139 #define SSL3_CK_RSA_RC2_40_MD5                  0x03000006
140 #define SSL3_CK_RSA_IDEA_128_SHA                0x03000007
141 #define SSL3_CK_RSA_DES_40_CBC_SHA              0x03000008
142 #define SSL3_CK_RSA_DES_64_CBC_SHA              0x03000009
143 #define SSL3_CK_RSA_DES_192_CBC3_SHA            0x0300000A

145 #define SSL3_CK_DH_DSS_DES_40_CBC_SHA           0x0300000B
146 #define SSL3_CK_DH_DSS_DES_64_CBC_SHA           0x0300000C
147 #define SSL3_CK_DH_DSS_DES_192_CBC3_SHA         0x0300000D
148 #define SSL3_CK_DH_RSA_DES_40_CBC_SHA           0x0300000E
149 #define SSL3_CK_DH_RSA_DES_64_CBC_SHA           0x0300000F
150 #define SSL3_CK_DH_RSA_DES_192_CBC3_SHA         0x03000010

152 #define SSL3_CK_EDH_DSS_DES_40_CBC_SHA          0x03000011
153 #define SSL3_CK_EDH_DSS_DES_64_CBC_SHA          0x03000012
154 #define SSL3_CK_EDH_DSS_DES_192_CBC3_SHA        0x03000013
155 #define SSL3_CK_EDH_RSA_DES_40_CBC_SHA          0x03000014
156 #define SSL3_CK_EDH_RSA_DES_64_CBC_SHA          0x03000015
157 #define SSL3_CK_EDH_RSA_DES_192_CBC3_SHA        0x03000016

159 #define SSL3_CK_ADH_RC4_40_MD5                  0x03000017
160 #define SSL3_CK_ADH_RC4_128_MD5                 0x03000018
161 #define SSL3_CK_ADH_DES_40_CBC_SHA              0x03000019
162 #define SSL3_CK_ADH_DES_64_CBC_SHA              0x0300001A
163 #define SSL3_CK_ADH_DES_192_CBC_SHA             0x0300001B

165 #if 0
166         #define SSL3_CK_FZA_DMS_NULL_SHA                0x0300001C
167         #define SSL3_CK_FZA_DMS_FZA_SHA                 0x0300001D
168         #if 0 /* Because it clashes with KRB5, is never used any more, and is sa
169                  to remove according to David Hopwood <david.hopwood@zetnet.co.u
170                  of the ietf-tls list */
171         #define SSL3_CK_FZA_DMS_RC4_SHA                 0x0300001E
172         #endif
173 #endif

175 /*    VRS Additional Kerberos5 entries
176  */
177 #define SSL3_CK_KRB5_DES_64_CBC_SHA             0x0300001E
178 #define SSL3_CK_KRB5_DES_192_CBC3_SHA           0x0300001F
179 #define SSL3_CK_KRB5_RC4_128_SHA                0x03000020
180 #define SSL3_CK_KRB5_IDEA_128_CBC_SHA           0x03000021
181 #define SSL3_CK_KRB5_DES_64_CBC_MD5             0x03000022
182 #define SSL3_CK_KRB5_DES_192_CBC3_MD5           0x03000023
183 #define SSL3_CK_KRB5_RC4_128_MD5                0x03000024
184 #define SSL3_CK_KRB5_IDEA_128_CBC_MD5           0x03000025

186 #define SSL3_CK_KRB5_DES_40_CBC_SHA             0x03000026
187 #define SSL3_CK_KRB5_RC2_40_CBC_SHA             0x03000027
188 #define SSL3_CK_KRB5_RC4_40_SHA                 0x03000028
189 #define SSL3_CK_KRB5_DES_40_CBC_MD5             0x03000029
190 #define SSL3_CK_KRB5_RC2_40_CBC_MD5             0x0300002A
191 #define SSL3_CK_KRB5_RC4_40_MD5                 0x0300002B

193 #define SSL3_TXT_RSA_NULL_MD5                   "NULL-MD5"
```



```
194 #define SSL3_TXT_RSA_NULL_SHA                   "NULL-SHA"
195 #define SSL3_TXT_RSA_RC4_40_MD5                 "EXP-RC4-MD5"
196 #define SSL3_TXT_RSA_RC4_128_MD5                "RC4-MD5"
197 #define SSL3_TXT_RSA_RC4_128_SHA                "RC4-SHA"
198 #define SSL3_TXT_RSA_RC2_40_MD5                 "EXP-RC2-CBC-MD5"
199 #define SSL3_TXT_RSA_IDEA_128_SHA               "IDEA-CBC-SHA"
200 #define SSL3_TXT_RSA_DES_40_CBC_SHA             "EXP-DES-CBC-SHA"
201 #define SSL3_TXT_RSA_DES_64_CBC_SHA             "DES-CBC-SHA"
202 #define SSL3_TXT_RSA_DES_192_CBC3_SHA           "DES-CBC3-SHA"

204 #define SSL3_TXT_DH_DSS_DES_40_CBC_SHA          "EXP-DH-DSS-DES-CBC-SHA"
205 #define SSL3_TXT_DH_DSS_DES_64_CBC_SHA          "DH-DSS-DES-CBC-SHA"
206 #define SSL3_TXT_DH_DSS_DES_192_CBC3_SHA        "DH-DSS-DES-CBC3-SHA"
207 #define SSL3_TXT_DH_RSA_DES_40_CBC_SHA          "EXP-DH-RSA-DES-CBC-SHA"
208 #define SSL3_TXT_DH_RSA_DES_64_CBC_SHA          "DH-RSA-DES-CBC-SHA"
209 #define SSL3_TXT_DH_RSA_DES_192_CBC3_SHA        "DH-RSA-DES-CBC3-SHA"

211 #define SSL3_TXT_EDH_DSS_DES_40_CBC_SHA         "EXP-EDH-DSS-DES-CBC-SHA"
212 #define SSL3_TXT_EDH_DSS_DES_64_CBC_SHA         "EDH-DSS-DES-CBC-SHA"
213 #define SSL3_TXT_EDH_DSS_DES_192_CBC3_SHA       "EDH-DSS-DES-CBC3-SHA"
214 #define SSL3_TXT_EDH_RSA_DES_40_CBC_SHA         "EXP-EDH-RSA-DES-CBC-SHA"
215 #define SSL3_TXT_EDH_RSA_DES_64_CBC_SHA         "EDH-RSA-DES-CBC-SHA"
216 #define SSL3_TXT_EDH_RSA_DES_192_CBC3_SHA       "EDH-RSA-DES-CBC3-SHA"

218 #define SSL3_TXT_ADH_RC4_40_MD5                 "EXP-ADH-RC4-MD5"
219 #define SSL3_TXT_ADH_RC4_128_MD5                "ADH-RC4-MD5"
220 #define SSL3_TXT_ADH_DES_40_CBC_SHA             "EXP-ADH-DES-CBC-SHA"
221 #define SSL3_TXT_ADH_DES_64_CBC_SHA             "ADH-DES-CBC-SHA"
222 #define SSL3_TXT_ADH_DES_192_CBC_SHA            "ADH-DES-CBC3-SHA"

224 #if 0
225         #define SSL3_TXT_FZA_DMS_NULL_SHA               "FZA-NULL-SHA"
226         #define SSL3_TXT_FZA_DMS_FZA_SHA                "FZA-FZA-CBC-SHA"
227         #define SSL3_TXT_FZA_DMS_RC4_SHA                "FZA-RC4-SHA"
228 #endif

230 #define SSL3_TXT_KRB5_DES_64_CBC_SHA            "KRB5-DES-CBC-SHA"
231 #define SSL3_TXT_KRB5_DES_192_CBC3_SHA          "KRB5-DES-CBC3-SHA"
232 #define SSL3_TXT_KRB5_RC4_128_SHA               "KRB5-RC4-SHA"
233 #define SSL3_TXT_KRB5_IDEA_128_CBC_SHA          "KRB5-IDEA-CBC-SHA"
234 #define SSL3_TXT_KRB5_DES_64_CBC_MD5            "KRB5-DES-CBC-MD5"
235 #define SSL3_TXT_KRB5_DES_192_CBC3_MD5          "KRB5-DES-CBC3-MD5"
236 #define SSL3_TXT_KRB5_RC4_128_MD5               "KRB5-RC4-MD5"
237 #define SSL3_TXT_KRB5_IDEA_128_CBC_MD5          "KRB5-IDEA-CBC-MD5"

239 #define SSL3_TXT_KRB5_DES_40_CBC_SHA            "EXP-KRB5-DES-CBC-SHA"
240 #define SSL3_TXT_KRB5_RC2_40_CBC_SHA            "EXP-KRB5-RC2-CBC-SHA"
241 #define SSL3_TXT_KRB5_RC4_40_SHA                "EXP-KRB5-RC4-SHA"
242 #define SSL3_TXT_KRB5_DES_40_CBC_MD5            "EXP-KRB5-DES-CBC-MD5"
243 #define SSL3_TXT_KRB5_RC2_40_CBC_MD5            "EXP-KRB5-RC2-CBC-MD5"
244 #define SSL3_TXT_KRB5_RC4_40_MD5                "EXP-KRB5-RC4-MD5"

246 #define SSL3_SSL_SESSION_ID_LENGTH              32
247 #define SSL3_MAX_SSL_SESSION_ID_LENGTH          32

249 #define SSL3_MASTER_SECRET_SIZE                 48
250 #define SSL3_RANDOM_SIZE                        32
251 #define SSL3_SESSION_ID_SIZE                    32
252 #define SSL3_RT_HEADER_LENGTH                   5

254 #ifndef SSL3_ALIGN_PAYLOAD
255  /* Some will argue that this increases memory footprint, but it's
256   * not actually true. Point is that malloc has to return at least
257   * 64-bit aligned pointers, meaning that allocating 5 bytes wastes
258   * 3 bytes in either case. Suggested pre-gaping simply moves these
259   * wasted bytes from the end of allocated region to its front,
```

```
260   * but makes data payload aligned, which improves performance:-) */
261 # define SSL3_ALIGN_PAYLOAD                     8
262 #else
263 # if (SSL3_ALIGN_PAYLOAD&(SSL3_ALIGN_PAYLOAD-1))!=0
264 #  error "insane SSL3_ALIGN_PAYLOAD"
265 #  undef SSL3_ALIGN_PAYLOAD
266 # endif
267 #endif

269 /* This is the maximum MAC (digest) size used by the SSL library.
270  * Currently maximum of 20 is used by SHA1, but we reserve for
271  * future extension for 512-bit hashes.
272  */

274 #define SSL3_RT_MAX_MD_SIZE                     64

276 /* Maximum block size used in all ciphersuites. Currently 16 for AES.
277  */

279 #define SSL_RT_MAX_CIPHER_BLOCK_SIZE            16

281 #define SSL3_RT_MAX_EXTRA                       (16384)

283 /* Maximum plaintext length: defined by SSL/TLS standards */
284 #define SSL3_RT_MAX_PLAIN_LENGTH                16384
285 /* Maximum compression overhead: defined by SSL/TLS standards */
286 #define SSL3_RT_MAX_COMPRESSED_OVERHEAD         1024

288 /* The standards give a maximum encryption overhead of 1024 bytes.
289  * In practice the value is lower than this. The overhead is the maximum
290  * number of padding bytes (256) plus the mac size.
291  */
292 #define SSL3_RT_MAX_ENCRYPTED_OVERHEAD  (256 + SSL3_RT_MAX_MD_SIZE)

294 /* OpenSSL currently only uses a padding length of at most one block so
295  * the send overhead is smaller.
296  */

298 #define SSL3_RT_SEND_MAX_ENCRYPTED_OVERHEAD \
299                         (SSL_RT_MAX_CIPHER_BLOCK_SIZE + SSL3_RT_MAX_MD_SIZE)

301 /* If compression isn't used don't include the compression overhead */

303 #ifdef OPENSSL_NO_COMP
304 #define SSL3_RT_MAX_COMPRESSED_LENGTH           SSL3_RT_MAX_PLAIN_LENGTH
305 #else
306 #define SSL3_RT_MAX_COMPRESSED_LENGTH   \
307                 (SSL3_RT_MAX_PLAIN_LENGTH+SSL3_RT_MAX_COMPRESSED_OVERHEAD)
308 #endif
309 #define SSL3_RT_MAX_ENCRYPTED_LENGTH    \
310                 (SSL3_RT_MAX_ENCRYPTED_OVERHEAD+SSL3_RT_MAX_COMPRESSED_LENGTH)
311 #define SSL3_RT_MAX_PACKET_SIZE         \
312                 (SSL3_RT_MAX_ENCRYPTED_LENGTH+SSL3_RT_HEADER_LENGTH)

314 #define SSL3_MD_CLIENT_FINISHED_CONST   "\x43\x4C\x4E\x54"
315 #define SSL3_MD_SERVER_FINISHED_CONST   "\x53\x52\x56\x52"

317 #define SSL3_VERSION                    0x0300
318 #define SSL3_VERSION_MAJOR              0x03
319 #define SSL3_VERSION_MINOR              0x00

321 #define SSL3_RT_CHANGE_CIPHER_SPEC      20
322 #define SSL3_RT_ALERT                   21
323 #define SSL3_RT_HANDSHAKE               22
324 #define SSL3_RT_APPLICATION_DATA        23
325 #define TLS1_RT_HEARTBEAT               24
```



```
327 #define SSL3_AL_WARNING                 1
328 #define SSL3_AL_FATAL                   2

330 #define SSL3_AD_CLOSE_NOTIFY             0
331 #define SSL3_AD_UNEXPECTED_MESSAGE      10      /* fatal */
332 #define SSL3_AD_BAD_RECORD_MAC          20      /* fatal */
333 #define SSL3_AD_DECOMPRESSION_FAILURE   30      /* fatal */
334 #define SSL3_AD_HANDSHAKE_FAILURE       40      /* fatal */
335 #define SSL3_AD_NO_CERTIFICATE          41
336 #define SSL3_AD_BAD_CERTIFICATE         42
337 #define SSL3_AD_UNSUPPORTED_CERTIFICATE 43
338 #define SSL3_AD_CERTIFICATE_REVOKED     44
339 #define SSL3_AD_CERTIFICATE_EXPIRED     45
340 #define SSL3_AD_CERTIFICATE_UNKNOWN     46
341 #define SSL3_AD_ILLEGAL_PARAMETER       47      /* fatal */

343 #define TLS1_HB_REQUEST         1
344 #define TLS1_HB_RESPONSE        2

346 #ifndef OPENSSL_NO_SSL_INTERN

348 typedef struct ssl3_record_st
349         {
350 /*r */  int type;               /* type of record */
351 /*rw*/  unsigned int length;    /* How many bytes available */
352 /*r */  unsigned int off;       /* read/write offset into 'buf' */
353 /*rw*/  unsigned char *data;    /* pointer to the record data */
354 /*rw*/  unsigned char *input;   /* where the decode bytes are */
355 /*r */  unsigned char *comp;    /* only used with decompression - malloc()ed */
356 /*r */  unsigned long epoch;    /* epoch number, needed by DTLS1 */
357 /*r */  unsigned char seq_num[8]; /* sequence number, needed by DTLS1 */
358         } SSL3_RECORD;

360 typedef struct ssl3_buffer_st
361         {
362         unsigned char *buf;     /* at least SSL3_RT_MAX_PACKET_SIZE bytes,
363                                  * see ssl3_setup_buffers() */
364         size_t len;             /* buffer size */
365         int offset;             /* where to 'copy from' */
366         int left;               /* how many bytes left */
367         } SSL3_BUFFER;

369 #endif

371 #define SSL3_CT_RSA_SIGN                        1
372 #define SSL3_CT_DSS_SIGN                        2
373 #define SSL3_CT_RSA_FIXED_DH                    3
374 #define SSL3_CT_DSS_FIXED_DH                    4
375 #define SSL3_CT_RSA_EPHEMERAL_DH                5
376 #define SSL3_CT_DSS_EPHEMERAL_DH                6
377 #define SSL3_CT_FORTEZZA_DMS                    20
378 /* SSL3_CT_NUMBER is used to size arrays and it must be large
379  * enough to contain all of the cert types defined either for
380  * SSLv3 and TLSv1.
381  */
382 #define SSL3_CT_NUMBER                  9


385 #define SSL3_FLAGS_NO_RENEGOTIATE_CIPHERS       0x0001
386 #define SSL3_FLAGS_DELAY_CLIENT_FINISHED        0x0002
387 #define SSL3_FLAGS_POP_BUFFER                   0x0004
388 #define TLS1_FLAGS_TLS_PADDING_BUG              0x0008
389 #define TLS1_FLAGS_SKIP_CERT_VERIFY             0x0010
390 #define TLS1_FLAGS_KEEP_HANDSHAKE               0x0020
391 #define SSL3_FLAGS_CCS_OK                       0x0080
```

```
/* SSL3_FLAGS_SGC_RESTART_DONE is set when we
 * restart a handshake because of MS SGC and so prevents us
 * from restarting the handshake in a loop. It's reset on a
 * renegotiation, so effectively limits the client to one restart
 * per negotiation. This limits the possibility of a DDoS
 * attack where the client handshakes in a loop using SGC to
 * restart. Servers which permit renegotiation can still be
 * effected, but we can't prevent that.
 */
#define SSL3_FLAGS_SGC_RESTART_DONE             0x0040

#ifndef OPENSSL_NO_SSL_INTERN

typedef struct ssl3_state_st
        {
        long flags;
        int delay_buf_pop_ret;

        unsigned char read_sequence[8];
        int read_mac_secret_size;
        unsigned char read_mac_secret[EVP_MAX_MD_SIZE];
        unsigned char write_sequence[8];
        int write_mac_secret_size;
        unsigned char write_mac_secret[EVP_MAX_MD_SIZE];

        unsigned char server_random[SSL3_RANDOM_SIZE];
        unsigned char client_random[SSL3_RANDOM_SIZE];

        /* flags for countermeasure against known-IV weakness */
        int need_empty_fragments;
        int empty_fragment_done;

        /* The value of 'extra' when the buffers were initialized */
        int init_extra;

        SSL3_BUFFER rbuf;       /* read IO goes into here */
        SSL3_BUFFER wbuf;       /* write IO goes into here */

        SSL3_RECORD rrec;       /* each decoded record goes in here */
        SSL3_RECORD wrec;       /* goes out from here */

        /* storage for Alert/Handshake protocol data received but not
         * yet processed by ssl3_read_bytes: */
        unsigned char alert_fragment[2];
        unsigned int alert_fragment_len;
        unsigned char handshake_fragment[4];
        unsigned int handshake_fragment_len;

        /* partial write - check the numbers match */
        unsigned int wnum;      /* number of bytes sent so far */
        int wpend_tot;          /* number bytes written */
        int wpend_type;
        int wpend_ret;          /* number of bytes submitted */
        const unsigned char *wpend_buf;

        /* used during startup, digest all incoming/outgoing packets */
        BIO *handshake_buffer;
        /* When set of handshake digests is determined, buffer is hashed
         * and freed and MD_CTX-es for all required digests are stored in
         * this array */
        EVP_MD_CTX **handshake_dgst;
        /* this is set whenerver we see a change_cipher_spec message
         * come in when we are not looking for one */
        int change_cipher_spec;
```



```
        int warn_alert;
        int fatal_alert;
        /* we allow one fatal and one warning alert to be outstanding,
         * send close alert via the warning alert */
        int alert_dispatch;
        unsigned char send_alert[2];

        /* This flag is set when we should renegotiate ASAP, basically when
         * there is no more data in the read or write buffers */
        int renegotiate;
        int total_renegotiations;
        int num_renegotiations;

        int in_read_app_data;

        /* Opaque PRF input as used for the current handshake.
         * These fields are used only if TLSEXT_TYPE_opaque_prf_input is defined
         * (otherwise, they are merely present to improve binary compatibility) */
        void *client_opaque_prf_input;
        size_t client_opaque_prf_input_len;
        void *server_opaque_prf_input;
        size_t server_opaque_prf_input_len;

        struct  {
                /* actually only needs to be 16+20 */
                unsigned char cert_verify_md[EVP_MAX_MD_SIZE*2];

                /* actually only need to be 16+20 for SSLv3 and 12 for TLS */
                unsigned char finish_md[EVP_MAX_MD_SIZE*2];
                int finish_md_len;
                unsigned char peer_finish_md[EVP_MAX_MD_SIZE*2];
                int peer_finish_md_len;

                unsigned long message_size;
                int message_type;

                /* used to hold the new cipher we are going to use */
                const SSL_CIPHER *new_cipher;
#ifndef OPENSSL_NO_DH
                DH *dh;
#endif

#ifndef OPENSSL_NO_ECDH
                EC_KEY *ecdh; /* holds short lived ECDH key */
#endif

                /* used when SSL_ST_FLUSH_DATA is entered */
                int next_state;

                int reuse_message;

                /* used for certificate requests */
                int cert_req;
                int ctype_num;
                char ctype[SSL3_CT_NUMBER];
                STACK_OF(X509_NAME) *ca_names;

                int use_rsa_tmp;

                int key_block_length;
                unsigned char *key_block;

                const EVP_CIPHER *new_sym_enc;
                const EVP_MD *new_hash;
                int new_mac_pkey_type;
                int new_mac_secret_size;
```

```
524 #ifndef OPENSSL_NO_COMP
525                 const SSL_COMP *new_compression;
526 #else
527                 char *new_compression;
528 #endif
529                 int cert_request;
530                 } tmp;

532         /* Connection binding to prevent renegotiation attacks */
533         unsigned char previous_client_finished[EVP_MAX_MD_SIZE];
534         unsigned char previous_client_finished_len;
535         unsigned char previous_server_finished[EVP_MAX_MD_SIZE];
536         unsigned char previous_server_finished_len;
537         int send_connection_binding; /* TODOEKR */

539 #ifndef OPENSSL_NO_NEXTPROTONEG
540         /* Set if we saw the Next Protocol Negotiation extension from our peer.
541         int next_proto_neg_seen;
542 #endif

544 #ifndef OPENSSL_NO_TLSEXT
545 #ifndef OPENSSL_NO_EC
546         /* This is set to true if we believe that this is a version of Safari
547          * running on OS X 10.6 or newer. We wish to know this because Safari
548          * on 10.8 .. 10.8.3 has broken ECDHE-ECDSA support. */
549         char is_probably_safari;
550 #endif /* !OPENSSL_NO_EC */
551 #endif /* !OPENSSL_NO_TLSEXT */
552         } SSL3_STATE;

554 #endif

556 /* SSLv3 */
557 /*client */
558 /* extra state */
559 #define SSL3_ST_CW_FLUSH                (0x100|SSL_ST_CONNECT)
560 #ifndef OPENSSL_NO_SCTP
561 #define DTLS1_SCTP_ST_CW_WRITE_SOCK                     (0x310|SSL_ST_CONNECT)
562 #define DTLS1_SCTP_ST_CR_READ_SOCK                      (0x320|SSL_ST_CONNECT)
563 #endif
564 /* write to server */
565 #define SSL3_ST_CW_CLNT_HELLO_A         (0x110|SSL_ST_CONNECT)
566 #define SSL3_ST_CW_CLNT_HELLO_B         (0x111|SSL_ST_CONNECT)
567 /* read from server */
568 #define SSL3_ST_CR_SRVR_HELLO_A         (0x120|SSL_ST_CONNECT)
569 #define SSL3_ST_CR_SRVR_HELLO_B         (0x121|SSL_ST_CONNECT)
570 #define DTLS1_ST_CR_HELLO_VERIFY_REQUEST_A (0x126|SSL_ST_CONNECT)
571 #define DTLS1_ST_CR_HELLO_VERIFY_REQUEST_B (0x127|SSL_ST_CONNECT)
572 #define SSL3_ST_CR_CERT_A               (0x130|SSL_ST_CONNECT)
573 #define SSL3_ST_CR_CERT_B               (0x131|SSL_ST_CONNECT)
574 #define SSL3_ST_CR_KEY_EXCH_A           (0x140|SSL_ST_CONNECT)
575 #define SSL3_ST_CR_KEY_EXCH_B           (0x141|SSL_ST_CONNECT)
576 #define SSL3_ST_CR_CERT_REQ_A           (0x150|SSL_ST_CONNECT)
577 #define SSL3_ST_CR_CERT_REQ_B           (0x151|SSL_ST_CONNECT)
578 #define SSL3_ST_CR_SRVR_DONE_A          (0x160|SSL_ST_CONNECT)
579 #define SSL3_ST_CR_SRVR_DONE_B          (0x161|SSL_ST_CONNECT)
580 /* write to server */
581 #define SSL3_ST_CW_CERT_A               (0x170|SSL_ST_CONNECT)
582 #define SSL3_ST_CW_CERT_B               (0x171|SSL_ST_CONNECT)
583 #define SSL3_ST_CW_CERT_C               (0x172|SSL_ST_CONNECT)
584 #define SSL3_ST_CW_CERT_D               (0x173|SSL_ST_CONNECT)
585 #define SSL3_ST_CW_KEY_EXCH_A           (0x180|SSL_ST_CONNECT)
586 #define SSL3_ST_CW_KEY_EXCH_B           (0x181|SSL_ST_CONNECT)
587 #define SSL3_ST_CW_CERT_VRFY_A          (0x190|SSL_ST_CONNECT)
588 #define SSL3_ST_CW_CERT_VRFY_B          (0x191|SSL_ST_CONNECT)
589 #define SSL3_ST_CW_CHANGE_A             (0x1A0|SSL_ST_CONNECT)
```



```
590 #define SSL3_ST_CW_CHANGE_B             (0x1A1|SSL_ST_CONNECT)
591 #ifndef OPENSSL_NO_NEXTPROTONEG
592 #define SSL3_ST_CW_NEXT_PROTO_A         (0x200|SSL_ST_CONNECT)
593 #define SSL3_ST_CW_NEXT_PROTO_B         (0x201|SSL_ST_CONNECT)
594 #endif
595 #define SSL3_ST_CW_FINISHED_A           (0x1B0|SSL_ST_CONNECT)
596 #define SSL3_ST_CW_FINISHED_B           (0x1B1|SSL_ST_CONNECT)
597 /* read from server */
598 #define SSL3_ST_CR_CHANGE_A             (0x1C0|SSL_ST_CONNECT)
599 #define SSL3_ST_CR_CHANGE_B             (0x1C1|SSL_ST_CONNECT)
600 #define SSL3_ST_CR_FINISHED_A           (0x1D0|SSL_ST_CONNECT)
601 #define SSL3_ST_CR_FINISHED_B           (0x1D1|SSL_ST_CONNECT)
602 #define SSL3_ST_CR_SESSION_TICKET_A     (0x1E0|SSL_ST_CONNECT)
603 #define SSL3_ST_CR_SESSION_TICKET_B     (0x1E1|SSL_ST_CONNECT)
604 #define SSL3_ST_CR_CERT_STATUS_A        (0x1F0|SSL_ST_CONNECT)
605 #define SSL3_ST_CR_CERT_STATUS_B        (0x1F1|SSL_ST_CONNECT)

607 /* server */
608 /* extra state */
609 #define SSL3_ST_SW_FLUSH                (0x100|SSL_ST_ACCEPT)
610 #ifndef OPENSSL_NO_SCTP
611 #define DTLS1_SCTP_ST_SW_WRITE_SOCK                     (0x310|SSL_ST_ACCEPT)
612 #define DTLS1_SCTP_ST_SR_READ_SOCK                      (0x320|SSL_ST_ACCEPT)
613 #endif
614 /* read from client */
615 /* Do not change the number values, they do matter */
616 #define SSL3_ST_SR_CLNT_HELLO_A         (0x110|SSL_ST_ACCEPT)
617 #define SSL3_ST_SR_CLNT_HELLO_B         (0x111|SSL_ST_ACCEPT)
618 #define SSL3_ST_SR_CLNT_HELLO_C         (0x112|SSL_ST_ACCEPT)
619 /* write to client */
620 #define DTLS1_ST_SW_HELLO_VERIFY_REQUEST_A (0x113|SSL_ST_ACCEPT)
621 #define DTLS1_ST_SW_HELLO_VERIFY_REQUEST_B (0x114|SSL_ST_ACCEPT)
622 #define SSL3_ST_SW_HELLO_REQ_A          (0x120|SSL_ST_ACCEPT)
623 #define SSL3_ST_SW_HELLO_REQ_B          (0x121|SSL_ST_ACCEPT)
624 #define SSL3_ST_SW_HELLO_REQ_C          (0x122|SSL_ST_ACCEPT)
625 #define SSL3_ST_SW_SRVR_HELLO_A         (0x130|SSL_ST_ACCEPT)
626 #define SSL3_ST_SW_SRVR_HELLO_B         (0x131|SSL_ST_ACCEPT)
627 #define SSL3_ST_SW_CERT_A               (0x140|SSL_ST_ACCEPT)
628 #define SSL3_ST_SW_CERT_B               (0x141|SSL_ST_ACCEPT)
629 #define SSL3_ST_SW_KEY_EXCH_A           (0x150|SSL_ST_ACCEPT)
630 #define SSL3_ST_SW_KEY_EXCH_B           (0x151|SSL_ST_ACCEPT)
631 #define SSL3_ST_SW_CERT_REQ_A           (0x160|SSL_ST_ACCEPT)
632 #define SSL3_ST_SW_CERT_REQ_B           (0x161|SSL_ST_ACCEPT)
633 #define SSL3_ST_SW_SRVR_DONE_A          (0x170|SSL_ST_ACCEPT)
634 #define SSL3_ST_SW_SRVR_DONE_B          (0x171|SSL_ST_ACCEPT)
635 /* read from client */
636 #define SSL3_ST_SR_CERT_A               (0x180|SSL_ST_ACCEPT)
637 #define SSL3_ST_SR_CERT_B               (0x181|SSL_ST_ACCEPT)
638 #define SSL3_ST_SR_KEY_EXCH_A           (0x190|SSL_ST_ACCEPT)
639 #define SSL3_ST_SR_KEY_EXCH_B           (0x191|SSL_ST_ACCEPT)
640 #define SSL3_ST_SR_CERT_VRFY_A          (0x1A0|SSL_ST_ACCEPT)
641 #define SSL3_ST_SR_CERT_VRFY_B          (0x1A1|SSL_ST_ACCEPT)
642 #define SSL3_ST_SR_CHANGE_A             (0x1B0|SSL_ST_ACCEPT)
643 #define SSL3_ST_SR_CHANGE_B             (0x1B1|SSL_ST_ACCEPT)
644 #ifndef OPENSSL_NO_NEXTPROTONEG
645 #define SSL3_ST_SR_NEXT_PROTO_A         (0x210|SSL_ST_ACCEPT)
646 #define SSL3_ST_SR_NEXT_PROTO_B         (0x211|SSL_ST_ACCEPT)
647 #endif
648 #define SSL3_ST_SR_FINISHED_A           (0x1C0|SSL_ST_ACCEPT)
649 #define SSL3_ST_SR_FINISHED_B           (0x1C1|SSL_ST_ACCEPT)
650 /* write to client */
651 #define SSL3_ST_SW_CHANGE_A             (0x1D0|SSL_ST_ACCEPT)
652 #define SSL3_ST_SW_CHANGE_B             (0x1D1|SSL_ST_ACCEPT)
653 #define SSL3_ST_SW_FINISHED_A           (0x1E0|SSL_ST_ACCEPT)
654 #define SSL3_ST_SW_FINISHED_B           (0x1E1|SSL_ST_ACCEPT)
655 #define SSL3_ST_SW_SESSION_TICKET_A     (0x1F0|SSL_ST_ACCEPT)
```

```
656 #define SSL3_ST_SW_SESSION_TICKET_B     (0x1F1|SSL_ST_ACCEPT)
657 #define SSL3_ST_SW_CERT_STATUS_A        (0x200|SSL_ST_ACCEPT)
658 #define SSL3_ST_SW_CERT_STATUS_B        (0x201|SSL_ST_ACCEPT)

660 #define SSL3_MT_HELLO_REQUEST                   0
661 #define SSL3_MT_CLIENT_HELLO                    1
662 #define SSL3_MT_SERVER_HELLO                    2
663 #define SSL3_MT_NEWSESSION_TICKET               4
664 #define SSL3_MT_CERTIFICATE                     11
665 #define SSL3_MT_SERVER_KEY_EXCHANGE             12
666 #define SSL3_MT_CERTIFICATE_REQUEST             13
667 #define SSL3_MT_SERVER_DONE                     14
668 #define SSL3_MT_CERTIFICATE_VERIFY              15
669 #define SSL3_MT_CLIENT_KEY_EXCHANGE             16
670 #define SSL3_MT_FINISHED                        20
671 #define SSL3_MT_CERTIFICATE_STATUS              22
672 #ifndef OPENSSL_NO_NEXTPROTONEG
673 #define SSL3_MT_NEXT_PROTO                      67
674 #endif
675 #define DTLS1_MT_HELLO_VERIFY_REQUEST    3


678 #define SSL3_MT_CCS                             1

680 /* These are used when changing over to a new cipher */
681 #define SSL3_CC_READ            0x01
682 #define SSL3_CC_WRITE           0x02
683 #define SSL3_CC_CLIENT          0x10
684 #define SSL3_CC_SERVER          0x20
685 #define SSL3_CHANGE_CIPHER_CLIENT_WRITE (SSL3_CC_CLIENT|SSL3_CC_WRITE)
686 #define SSL3_CHANGE_CIPHER_SERVER_READ  (SSL3_CC_SERVER|SSL3_CC_READ)
687 #define SSL3_CHANGE_CIPHER_CLIENT_READ  (SSL3_CC_CLIENT|SSL3_CC_READ)
688 #define SSL3_CHANGE_CIPHER_SERVER_WRITE (SSL3_CC_SERVER|SSL3_CC_WRITE)

690 #ifdef  __cplusplus
691 }
692 #endif
693 #endif
694 #endif /* ! codereview */
```

```
**********************************************************
    4406 Wed Aug 13 19:51:49 2014
new/usr/src/lib/openssl/include/openssl/stack.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/stack/stack.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_STACK_H
#define HEADER_STACK_H
```



```
#ifdef  __cplusplus
extern "C" {
#endif

typedef struct stack_st
        {
        int num;
        char **data;
        int sorted;

        int num_alloc;
        int (*comp)(const void *, const void *);
        } _STACK;  /* Use STACK_OF(...) instead */

#define M_sk_num(sk)            ((sk) ? (sk)->num:-1)
#define M_sk_value(sk,n)        ((sk) ? (sk)->data[n] : NULL)

int sk_num(const _STACK *);
void *sk_value(const _STACK *, int);

void *sk_set(_STACK *, int, void *);

_STACK *sk_new(int (*cmp)(const void *, const void *));
_STACK *sk_new_null(void);
void sk_free(_STACK *);
void sk_pop_free(_STACK *st, void (*func)(void *));
int sk_insert(_STACK *sk, void *data, int where);
void *sk_delete(_STACK *st, int loc);
void *sk_delete_ptr(_STACK *st, void *p);
int sk_find(_STACK *st, void *data);
int sk_find_ex(_STACK *st, void *data);
int sk_push(_STACK *st, void *data);
int sk_unshift(_STACK *st, void *data);
void *sk_shift(_STACK *st);
void *sk_pop(_STACK *st);
void sk_zero(_STACK *st);
int (*sk_set_cmp_func(_STACK *sk, int (*c)(const void *, const void *)))
        (const void *, const void *);
_STACK *sk_dup(_STACK *st);
void sk_sort(_STACK *st);
int sk_is_sorted(const _STACK *st);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
  292643 Wed Aug 13 19:51:50 2014
new/usr/src/lib/openssl/include/openssl/sunw_prefix.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2014 Joyent, Inc.  All rights reserved.
 */

#ifndef _SUNW_PREFIX_H
#define _SUNW_PREFIX_H

#pragma redefine_extname        _CONF_add_string sunw__CONF_add_string
#pragma redefine_extname        _CONF_free_data sunw__CONF_free_data
#pragma redefine_extname        _CONF_get_section sunw__CONF_get_section
#pragma redefine_extname        _CONF_get_section_values sunw__CONF_get_section_
#pragma redefine_extname        _CONF_get_string sunw__CONF_get_string
#pragma redefine_extname        _CONF_new_data sunw__CONF_new_data
#pragma redefine_extname        _CONF_new_section sunw__CONF_new_section
#pragma redefine_extname        _des_crypt sunw__des_crypt
#pragma redefine_extname        _ossl_096_des_random_seed sunw__ossl_096_des_ran
#pragma redefine_extname        _ossl_old_crypt sunw__ossl_old_crypt
#pragma redefine_extname        _ossl_old_des_cbc_cksum sunw__ossl_old_des_cbc_c
#pragma redefine_extname        _ossl_old_des_cbc_encrypt sunw__ossl_old_des_cbc
#pragma redefine_extname        _ossl_old_des_cfb_encrypt sunw__ossl_old_des_cfb
#pragma redefine_extname        _ossl_old_des_cfb64_encrypt sunw__ossl_old_des_c
#pragma redefine_extname        _ossl_old_des_crypt sunw__ossl_old_des_crypt
#pragma redefine_extname        _ossl_old_des_decrypt3 sunw__ossl_old_des_decryp
#pragma redefine_extname        _ossl_old_des_ecb_encrypt sunw__ossl_old_des_ecb
#pragma redefine_extname        _ossl_old_des_ecb3_encrypt sunw__ossl_old_des_ec
#pragma redefine_extname        _ossl_old_des_ede3_cbc_encrypt sunw__ossl_old_de
#pragma redefine_extname        _ossl_old_des_ede3_cfb64_encrypt sunw__ossl_old_
#pragma redefine_extname        _ossl_old_des_ede3_ofb64_encrypt sunw__ossl_old_
#pragma redefine_extname        _ossl_old_des_enc_read sunw__ossl_old_des_enc_re
#pragma redefine_extname        _ossl_old_des_enc_write sunw__ossl_old_des_enc_w
#pragma redefine_extname        _ossl_old_des_encrypt sunw__ossl_old_des_encrypt
#pragma redefine_extname        _ossl_old_des_encrypt2 sunw__ossl_old_des_encryp
#pragma redefine_extname        _ossl_old_des_encrypt3 sunw__ossl_old_des_encryp
#pragma redefine_extname        _ossl_old_des_fcrypt sunw__ossl_old_des_fcrypt
#pragma redefine_extname        _ossl_old_des_is_weak_key sunw__ossl_old_des_is_
#pragma redefine_extname        _ossl_old_des_key_sched sunw__ossl_old_des_key_s
#pragma redefine_extname        _ossl_old_des_ncbc_encrypt sunw__ossl_old_des_nc
#pragma redefine_extname        _ossl_old_des_ofb_encrypt sunw__ossl_old_des_ofb
#pragma redefine_extname        _ossl_old_des_ofb64_encrypt sunw__ossl_old_des_o
#pragma redefine_extname        _ossl_old_des_options sunw__ossl_old_des_options
#pragma redefine_extname        _ossl_old_des_pcbc_encrypt sunw__ossl_old_des_pc
#pragma redefine_extname        _ossl_old_des_quad_cksum sunw__ossl_old_des_quad
#pragma redefine_extname        _ossl_old_des_random_key sunw__ossl_old_des_rand
#pragma redefine_extname        _ossl_old_des_random_seed sunw__ossl_old_des_ran
#pragma redefine_extname        _ossl_old_des_read_2passwords sunw__ossl_old_des
#pragma redefine_extname        _ossl_old_des_read_password sunw__ossl_old_des_r
#pragma redefine_extname        _ossl_old_des_read_pw sunw__ossl_old_des_read_pw
#pragma redefine_extname        _ossl_old_des_read_pw_string sunw__ossl_old_des_
#pragma redefine_extname        _ossl_old_des_set_key sunw__ossl_old_des_set_key
#pragma redefine_extname        _ossl_old_des_set_odd_parity sunw__ossl_old_des_
```



```
#pragma redefine_extname        _ossl_old_des_string_to_2keys sunw__ossl_old_des
#pragma redefine_extname        _ossl_old_des_string_to_key sunw__ossl_old_des_s
#pragma redefine_extname        _ossl_old_des_xcbc_encrypt sunw__ossl_old_des_xc
#pragma redefine_extname        _shadow_DES_rw_mode sunw__shadow_DES_rw_mode
#pragma redefine_extname        a2d_ASN1_OBJECT sunw_a2d_ASN1_OBJECT
#pragma redefine_extname        a2i_ASN1_ENUMERATED sunw_a2i_ASN1_ENUMERATED
#pragma redefine_extname        a2i_ASN1_INTEGER sunw_a2i_ASN1_INTEGER
#pragma redefine_extname        a2i_ASN1_STRING sunw_a2i_ASN1_STRING
#pragma redefine_extname        a2i_GENERAL_NAME sunw_a2i_GENERAL_NAME
#pragma redefine_extname        a2i_ipadd sunw_a2i_ipadd
#pragma redefine_extname        a2i_IPADDRESS sunw_a2i_IPADDRESS
#pragma redefine_extname        a2i_IPADDRESS_NC sunw_a2i_IPADDRESS_NC
#pragma redefine_extname        ACCESS_DESCRIPTION_free sunw_ACCESS_DESCRIPTION_
#pragma redefine_extname        ACCESS_DESCRIPTION_it sunw_ACCESS_DESCRIPTION_it
#pragma redefine_extname        ACCESS_DESCRIPTION_new sunw_ACCESS_DESCRIPTION_n
#pragma redefine_extname        AES_bi_ige_encrypt sunw_AES_bi_ige_encrypt
#pragma redefine_extname        AES_cbc_encrypt sunw_AES_cbc_encrypt
#pragma redefine_extname        AES_cfb1_encrypt sunw_AES_cfb1_encrypt
#pragma redefine_extname        AES_cfb128_encrypt sunw_AES_cfb128_encrypt
#pragma redefine_extname        AES_cfb8_encrypt sunw_AES_cfb8_encrypt
#pragma redefine_extname        AES_ctr128_encrypt sunw_AES_ctr128_encrypt
#pragma redefine_extname        AES_decrypt sunw_AES_decrypt
#pragma redefine_extname        AES_ecb_encrypt sunw_AES_ecb_encrypt
#pragma redefine_extname        AES_encrypt sunw_AES_encrypt
#pragma redefine_extname        AES_ige_encrypt sunw_AES_ige_encrypt
#pragma redefine_extname        AES_ofb128_encrypt sunw_AES_ofb128_encrypt
#pragma redefine_extname        AES_options sunw_AES_options
#pragma redefine_extname        AES_set_decrypt_key sunw_AES_set_decrypt_key
#pragma redefine_extname        AES_set_encrypt_key sunw_AES_set_encrypt_key
#pragma redefine_extname        AES_unwrap_key sunw_AES_unwrap_key
#pragma redefine_extname        AES_version sunw_AES_version
#pragma redefine_extname        AES_wrap_key sunw_AES_wrap_key
#pragma redefine_extname        aesni_cbc_encrypt sunw_aesni_cbc_encrypt
#pragma redefine_extname        aesni_cbc_sha1_enc sunw_aesni_cbc_sha1_enc
#pragma redefine_extname        aesni_ccm64_decrypt_blocks sunw_aesni_ccm64_decr
#pragma redefine_extname        aesni_ccm64_encrypt_blocks sunw_aesni_ccm64_encr
#pragma redefine_extname        aesni_ctr32_encrypt_blocks sunw_aesni_ctr32_encr
#pragma redefine_extname        aesni_decrypt sunw_aesni_decrypt
#pragma redefine_extname        aesni_ecb_encrypt sunw_aesni_ecb_encrypt
#pragma redefine_extname        aesni_encrypt sunw_aesni_encrypt
#pragma redefine_extname        aesni_set_decrypt_key sunw_aesni_set_decrypt_key
#pragma redefine_extname        aesni_set_encrypt_key sunw_aesni_set_encrypt_key
#pragma redefine_extname        aesni_xts_decrypt sunw_aesni_xts_decrypt
#pragma redefine_extname        aesni_xts_encrypt sunw_aesni_xts_encrypt
#pragma redefine_extname        asn1_add_error sunw_asn1_add_error
#pragma redefine_extname        ASN1_add_oid_module sunw_ASN1_add_oid_module
#pragma redefine_extname        ASN1_ANY_it sunw_ASN1_ANY_it
#pragma redefine_extname        ASN1_BIT_STRING_check sunw_ASN1_BIT_STRING_check
#pragma redefine_extname        ASN1_BIT_STRING_free sunw_ASN1_BIT_STRING_free
#pragma redefine_extname        ASN1_BIT_STRING_get_bit sunw_ASN1_BIT_STRING_get
#pragma redefine_extname        ASN1_BIT_STRING_it sunw_ASN1_BIT_STRING_it
#pragma redefine_extname        ASN1_BIT_STRING_name_print sunw_ASN1_BIT_STRING_
#pragma redefine_extname        ASN1_BIT_STRING_new sunw_ASN1_BIT_STRING_new
#pragma redefine_extname        ASN1_BIT_STRING_num_asc sunw_ASN1_BIT_STRING_num
#pragma redefine_extname        ASN1_BIT_STRING_set sunw_ASN1_BIT_STRING_set
#pragma redefine_extname        ASN1_BIT_STRING_set_asc sunw_ASN1_BIT_STRING_set
#pragma redefine_extname        ASN1_BIT_STRING_set_bit sunw_ASN1_BIT_STRING_set
#pragma redefine_extname        ASN1_BMPSTRING_free sunw_ASN1_BMPSTRING_free
#pragma redefine_extname        ASN1_BMPSTRING_it sunw_ASN1_BMPSTRING_it
#pragma redefine_extname        ASN1_BMPSTRING_new sunw_ASN1_BMPSTRING_new
#pragma redefine_extname        ASN1_bn_print sunw_ASN1_bn_print
#pragma redefine_extname        ASN1_BOOLEAN_it sunw_ASN1_BOOLEAN_it
#pragma redefine_extname        ASN1_check_infinite_end sunw_ASN1_check_infinite
#pragma redefine_extname        ASN1_const_check_infinite_end sunw_ASN1_const_ch
#pragma redefine_extname        asn1_const_Finish sunw_asn1_const_Finish
#pragma redefine_extname        ASN1_d2i_bio sunw_ASN1_d2i_bio
```

```
128 #pragma redefine_extname        ASN1_d2i_fp sunw_ASN1_d2i_fp
129 #pragma redefine_extname        ASN1_digest sunw_ASN1_digest
130 #pragma redefine_extname        asn1_do_adb sunw_asn1_do_adb
131 #pragma redefine_extname        asn1_do_lock sunw_asn1_do_lock
132 #pragma redefine_extname        ASN1_dup sunw_ASN1_dup
133 #pragma redefine_extname        asn1_enc_free sunw_asn1_enc_free
134 #pragma redefine_extname        asn1_enc_init sunw_asn1_enc_init
135 #pragma redefine_extname        asn1_enc_restore sunw_asn1_enc_restore
136 #pragma redefine_extname        asn1_enc_save sunw_asn1_enc_save
137 #pragma redefine_extname        ASN1_ENUMERATED_free sunw_ASN1_ENUMERATED_free
138 #pragma redefine_extname        ASN1_ENUMERATED_get sunw_ASN1_ENUMERATED_get
139 #pragma redefine_extname        ASN1_ENUMERATED_it sunw_ASN1_ENUMERATED_it
140 #pragma redefine_extname        ASN1_ENUMERATED_new sunw_ASN1_ENUMERATED_new
141 #pragma redefine_extname        ASN1_ENUMERATED_set sunw_ASN1_ENUMERATED_set
142 #pragma redefine_extname        ASN1_ENUMERATED_to_BN sunw_ASN1_ENUMERATED_to_BN
143 #pragma redefine_extname        asn1_ex_c2i sunw_asn1_ex_c2i
144 #pragma redefine_extname        asn1_ex_i2c sunw_asn1_ex_i2c
145 #pragma redefine_extname        ASN1_FBOOLEAN_it sunw_ASN1_FBOOLEAN_it
146 #pragma redefine_extname        asn1_Finish sunw_asn1_Finish
147 #pragma redefine_extname        ASN1_GENERALIZEDTIME_adj sunw_ASN1_GENERALIZEDTI
148 #pragma redefine_extname        ASN1_GENERALIZEDTIME_check sunw_ASN1_GENERALIZED
149 #pragma redefine_extname        ASN1_GENERALIZEDTIME_free sunw_ASN1_GENERALIZEDT
150 #pragma redefine_extname        ASN1_GENERALIZEDTIME_it sunw_ASN1_GENERALIZEDTIM
151 #pragma redefine_extname        ASN1_GENERALIZEDTIME_new sunw_ASN1_GENERALIZEDTI
152 #pragma redefine_extname        ASN1_GENERALIZEDTIME_print sunw_ASN1_GENERALIZED
153 #pragma redefine_extname        ASN1_GENERALIZEDTIME_set sunw_ASN1_GENERALIZEDTI
154 #pragma redefine_extname        ASN1_GENERALIZEDTIME_set_string sunw_ASN1_GENERA
155 #pragma redefine_extname        ASN1_GENERALSTRING_free sunw_ASN1_GENERALSTRING_
156 #pragma redefine_extname        ASN1_GENERALSTRING_it sunw_ASN1_GENERALSTRING_it
157 #pragma redefine_extname        ASN1_GENERALSTRING_new sunw_ASN1_GENERALSTRING_n
158 #pragma redefine_extname        ASN1_generate_nconf sunw_ASN1_generate_nconf
159 #pragma redefine_extname        ASN1_generate_v3 sunw_ASN1_generate_v3
160 #pragma redefine_extname        asn1_get_choice_selector sunw_asn1_get_choice_se
161 #pragma redefine_extname        asn1_get_field_ptr sunw_asn1_get_field_ptr
162 #pragma redefine_extname        ASN1_get_object sunw_ASN1_get_object
163 #pragma redefine_extname        asn1_GetSequence sunw_asn1_GetSequence
164 #pragma redefine_extname        ASN1_i2d_bio sunw_ASN1_i2d_bio
165 #pragma redefine_extname        ASN1_i2d_fp sunw_ASN1_i2d_fp
166 #pragma redefine_extname        ASN1_IA5STRING_free sunw_ASN1_IA5STRING_free
167 #pragma redefine_extname        ASN1_IA5STRING_it sunw_ASN1_IA5STRING_it
168 #pragma redefine_extname        ASN1_IA5STRING_new sunw_ASN1_IA5STRING_new
169 #pragma redefine_extname        ASN1_INTEGER_cmp sunw_ASN1_INTEGER_cmp
170 #pragma redefine_extname        ASN1_INTEGER_dup sunw_ASN1_INTEGER_dup
171 #pragma redefine_extname        ASN1_INTEGER_free sunw_ASN1_INTEGER_free
172 #pragma redefine_extname        ASN1_INTEGER_get sunw_ASN1_INTEGER_get
173 #pragma redefine_extname        ASN1_INTEGER_it sunw_ASN1_INTEGER_it
174 #pragma redefine_extname        ASN1_INTEGER_new sunw_ASN1_INTEGER_new
175 #pragma redefine_extname        ASN1_INTEGER_set sunw_ASN1_INTEGER_set
176 #pragma redefine_extname        ASN1_INTEGER_to_BN sunw_ASN1_INTEGER_to_BN
177 #pragma redefine_extname        ASN1_item_d2i sunw_ASN1_item_d2i
178 #pragma redefine_extname        ASN1_item_d2i_bio sunw_ASN1_item_d2i_bio
179 #pragma redefine_extname        ASN1_item_d2i_fp sunw_ASN1_item_d2i_fp
180 #pragma redefine_extname        ASN1_item_digest sunw_ASN1_item_digest
181 #pragma redefine_extname        ASN1_item_dup sunw_ASN1_item_dup
182 #pragma redefine_extname        ASN1_item_ex_d2i sunw_ASN1_item_ex_d2i
183 #pragma redefine_extname        ASN1_item_ex_free sunw_ASN1_item_ex_free
184 #pragma redefine_extname        ASN1_item_ex_i2d sunw_ASN1_item_ex_i2d
185 #pragma redefine_extname        ASN1_item_ex_new sunw_ASN1_item_ex_new
186 #pragma redefine_extname        ASN1_item_free sunw_ASN1_item_free
187 #pragma redefine_extname        ASN1_item_i2d sunw_ASN1_item_i2d
188 #pragma redefine_extname        ASN1_item_i2d_bio sunw_ASN1_item_i2d_bio
189 #pragma redefine_extname        ASN1_item_i2d_fp sunw_ASN1_item_i2d_fp
190 #pragma redefine_extname        ASN1_item_ndef_i2d sunw_ASN1_item_ndef_i2d
191 #pragma redefine_extname        ASN1_item_new sunw_ASN1_item_new
192 #pragma redefine_extname        ASN1_item_pack sunw_ASN1_item_pack
193 #pragma redefine_extname        ASN1_item_print sunw_ASN1_item_print
```



```
194 #pragma redefine_extname        ASN1_item_sign sunw_ASN1_item_sign
195 #pragma redefine_extname        ASN1_item_sign_ctx sunw_ASN1_item_sign_ctx
196 #pragma redefine_extname        ASN1_item_unpack sunw_ASN1_item_unpack
197 #pragma redefine_extname        ASN1_item_verify sunw_ASN1_item_verify
198 #pragma redefine_extname        ASN1_mbstring_copy sunw_ASN1_mbstring_copy
199 #pragma redefine_extname        ASN1_mbstring_ncopy sunw_ASN1_mbstring_ncopy
200 #pragma redefine_extname        ASN1_NULL_free sunw_ASN1_NULL_free
201 #pragma redefine_extname        ASN1_NULL_it sunw_ASN1_NULL_it
202 #pragma redefine_extname        ASN1_NULL_new sunw_ASN1_NULL_new
203 #pragma redefine_extname        ASN1_OBJECT_create sunw_ASN1_OBJECT_create
204 #pragma redefine_extname        ASN1_OBJECT_free sunw_ASN1_OBJECT_free
205 #pragma redefine_extname        ASN1_OBJECT_it sunw_ASN1_OBJECT_it
206 #pragma redefine_extname        ASN1_OBJECT_new sunw_ASN1_OBJECT_new
207 #pragma redefine_extname        ASN1_object_size sunw_ASN1_object_size
208 #pragma redefine_extname        ASN1_OCTET_STRING_cmp sunw_ASN1_OCTET_STRING_cmp
209 #pragma redefine_extname        ASN1_OCTET_STRING_dup sunw_ASN1_OCTET_STRING_dup
210 #pragma redefine_extname        ASN1_OCTET_STRING_free sunw_ASN1_OCTET_STRING_fr
211 #pragma redefine_extname        ASN1_OCTET_STRING_it sunw_ASN1_OCTET_STRING_it
212 #pragma redefine_extname        ASN1_OCTET_STRING_NDEF_it sunw_ASN1_OCTET_STRING
213 #pragma redefine_extname        ASN1_OCTET_STRING_new sunw_ASN1_OCTET_STRING_new
214 #pragma redefine_extname        ASN1_OCTET_STRING_set sunw_ASN1_OCTET_STRING_set
215 #pragma redefine_extname        ASN1_pack_string sunw_ASN1_pack_string
216 #pragma redefine_extname        ASN1_parse sunw_ASN1_parse
217 #pragma redefine_extname        ASN1_parse_dump sunw_ASN1_parse_dump
218 #pragma redefine_extname        ASN1_PCTX_free sunw_ASN1_PCTX_free
219 #pragma redefine_extname        ASN1_PCTX_get_cert_flags sunw_ASN1_PCTX_get_cert
220 #pragma redefine_extname        ASN1_PCTX_get_flags sunw_ASN1_PCTX_get_flags
221 #pragma redefine_extname        ASN1_PCTX_get_nm_flags sunw_ASN1_PCTX_get_nm_fla
222 #pragma redefine_extname        ASN1_PCTX_get_oid_flags sunw_ASN1_PCTX_get_oid_f
223 #pragma redefine_extname        ASN1_PCTX_get_str_flags sunw_ASN1_PCTX_get_str_f
224 #pragma redefine_extname        ASN1_PCTX_new sunw_ASN1_PCTX_new
225 #pragma redefine_extname        ASN1_PCTX_set_cert_flags sunw_ASN1_PCTX_set_cert
226 #pragma redefine_extname        ASN1_PCTX_set_flags sunw_ASN1_PCTX_set_flags
227 #pragma redefine_extname        ASN1_PCTX_set_nm_flags sunw_ASN1_PCTX_set_nm_fla
228 #pragma redefine_extname        ASN1_PCTX_set_oid_flags sunw_ASN1_PCTX_set_oid_f
229 #pragma redefine_extname        ASN1_PCTX_set_str_flags sunw_ASN1_PCTX_set_str_f
230 #pragma redefine_extname        ASN1_primitive_free sunw_ASN1_primitive_free
231 #pragma redefine_extname        ASN1_primitive_new sunw_ASN1_primitive_new
232 #pragma redefine_extname        ASN1_PRINTABLE_free sunw_ASN1_PRINTABLE_free
233 #pragma redefine_extname        ASN1_PRINTABLE_it sunw_ASN1_PRINTABLE_it
234 #pragma redefine_extname        ASN1_PRINTABLE_new sunw_ASN1_PRINTABLE_new
235 #pragma redefine_extname        ASN1_PRINTABLE_type sunw_ASN1_PRINTABLE_type
236 #pragma redefine_extname        ASN1_PRINTABLESTRING_free sunw_ASN1_PRINTABLESTR
237 #pragma redefine_extname        ASN1_PRINTABLESTRING_it sunw_ASN1_PRINTABLESTRIN
238 #pragma redefine_extname        ASN1_PRINTABLESTRING_new sunw_ASN1_PRINTABLESTRI
239 #pragma redefine_extname        ASN1_put_eoc sunw_ASN1_put_eoc
240 #pragma redefine_extname        ASN1_put_object sunw_ASN1_put_object
241 #pragma redefine_extname        ASN1_seq_pack sunw_ASN1_seq_pack
242 #pragma redefine_extname        ASN1_seq_unpack sunw_ASN1_seq_unpack
243 #pragma redefine_extname        ASN1_SEQUENCE_ANY_it sunw_ASN1_SEQUENCE_ANY_it
244 #pragma redefine_extname        ASN1_SEQUENCE_it sunw_ASN1_SEQUENCE_it
245 #pragma redefine_extname        ASN1_SET_ANY_it sunw_ASN1_SET_ANY_it
246 #pragma redefine_extname        asn1_set_choice_selector sunw_asn1_set_choice_se
247 #pragma redefine_extname        ASN1_sign sunw_ASN1_sign
248 #pragma redefine_extname        ASN1_STRING_cmp sunw_ASN1_STRING_cmp
249 #pragma redefine_extname        ASN1_STRING_copy sunw_ASN1_STRING_copy
250 #pragma redefine_extname        ASN1_STRING_data sunw_ASN1_STRING_data
251 #pragma redefine_extname        ASN1_STRING_dup sunw_ASN1_STRING_dup
252 #pragma redefine_extname        ASN1_STRING_free sunw_ASN1_STRING_free
253 #pragma redefine_extname        ASN1_STRING_get_default_mask sunw_ASN1_STRING_ge
254 #pragma redefine_extname        ASN1_STRING_length sunw_ASN1_STRING_length
255 #pragma redefine_extname        ASN1_STRING_length_set sunw_ASN1_STRING_length_s
256 #pragma redefine_extname        ASN1_STRING_new sunw_ASN1_STRING_new
257 #pragma redefine_extname        ASN1_STRING_print sunw_ASN1_STRING_print
258 #pragma redefine_extname        ASN1_STRING_print_ex sunw_ASN1_STRING_print_ex
259 #pragma redefine_extname        ASN1_STRING_print_ex_fp sunw_ASN1_STRING_print_e
```

```
260 #pragma redefine_extname         ASN1_STRING_set sunw_ASN1_STRING_set
261 #pragma redefine_extname         ASN1_STRING_set_by_NID sunw_ASN1_STRING_set_by_N
262 #pragma redefine_extname         ASN1_STRING_set_default_mask sunw_ASN1_STRING_se
263 #pragma redefine_extname         ASN1_STRING_set_default_mask_asc sunw_ASN1_STRIN
264 #pragma redefine_extname         ASN1_STRING_set0 sunw_ASN1_STRING_set0
265 #pragma redefine_extname         ASN1_STRING_TABLE_add sunw_ASN1_STRING_TABLE_add
266 #pragma redefine_extname         ASN1_STRING_TABLE_cleanup sunw_ASN1_STRING_TABLE
267 #pragma redefine_extname         ASN1_STRING_TABLE_get sunw_ASN1_STRING_TABLE_get
268 #pragma redefine_extname         ASN1_STRING_to_UTF8 sunw_ASN1_STRING_to_UTF8
269 #pragma redefine_extname         ASN1_STRING_type sunw_ASN1_STRING_type
270 #pragma redefine_extname         ASN1_STRING_type_new sunw_ASN1_STRING_type_new
271 #pragma redefine_extname         ASN1_T61STRING_free sunw_ASN1_T61STRING_free
272 #pragma redefine_extname         ASN1_T61STRING_it sunw_ASN1_T61STRING_it
273 #pragma redefine_extname         ASN1_T61STRING_new sunw_ASN1_T61STRING_new
274 #pragma redefine_extname         ASN1_tag2bit sunw_ASN1_tag2bit
275 #pragma redefine_extname         ASN1_tag2str sunw_ASN1_tag2str
276 #pragma redefine_extname         ASN1_TBOOLEAN_it sunw_ASN1_TBOOLEAN_it
277 #pragma redefine_extname         ASN1_template_d2i sunw_ASN1_template_d2i
278 #pragma redefine_extname         ASN1_template_free sunw_ASN1_template_free
279 #pragma redefine_extname         ASN1_template_i2d sunw_ASN1_template_i2d
280 #pragma redefine_extname         ASN1_template_new sunw_ASN1_template_new
281 #pragma redefine_extname         asn1_template_print_ctx sunw_asn1_template_print
282 #pragma redefine_extname         ASN1_TIME_adj sunw_ASN1_TIME_adj
283 #pragma redefine_extname         ASN1_TIME_check sunw_ASN1_TIME_check
284 #pragma redefine_extname         ASN1_TIME_free sunw_ASN1_TIME_free
285 #pragma redefine_extname         ASN1_TIME_it sunw_ASN1_TIME_it
286 #pragma redefine_extname         ASN1_TIME_new sunw_ASN1_TIME_new
287 #pragma redefine_extname         ASN1_TIME_print sunw_ASN1_TIME_print
288 #pragma redefine_extname         ASN1_TIME_set sunw_ASN1_TIME_set
289 #pragma redefine_extname         ASN1_TIME_set_string sunw_ASN1_TIME_set_string
290 #pragma redefine_extname         ASN1_TIME_to_generalizedtime sunw_ASN1_TIME_to_g
291 #pragma redefine_extname         ASN1_TYPE_cmp sunw_ASN1_TYPE_cmp
292 #pragma redefine_extname         ASN1_TYPE_free sunw_ASN1_TYPE_free
293 #pragma redefine_extname         ASN1_TYPE_get sunw_ASN1_TYPE_get
294 #pragma redefine_extname         ASN1_TYPE_get_int_octetstring sunw_ASN1_TYPE_get
295 #pragma redefine_extname         ASN1_TYPE_get_octetstring sunw_ASN1_TYPE_get_oct
296 #pragma redefine_extname         ASN1_TYPE_new sunw_ASN1_TYPE_new
297 #pragma redefine_extname         ASN1_TYPE_set sunw_ASN1_TYPE_set
298 #pragma redefine_extname         ASN1_TYPE_set_int_octetstring sunw_ASN1_TYPE_set
299 #pragma redefine_extname         ASN1_TYPE_set_octetstring sunw_ASN1_TYPE_set_oct
300 #pragma redefine_extname         ASN1_TYPE_set1 sunw_ASN1_TYPE_set1
301 #pragma redefine_extname         ASN1_UNIVERSALSTRING_free sunw_ASN1_UNIVERSALSTR
302 #pragma redefine_extname         ASN1_UNIVERSALSTRING_it sunw_ASN1_UNIVERSALSTRIN
303 #pragma redefine_extname         ASN1_UNIVERSALSTRING_new sunw_ASN1_UNIVERSALSTRI
304 #pragma redefine_extname         ASN1_UNIVERSALSTRING_to_string sunw_ASN1_UNIVERS
305 #pragma redefine_extname         ASN1_unpack_string sunw_ASN1_unpack_string
306 #pragma redefine_extname         ASN1_UTCTIME_adj sunw_ASN1_UTCTIME_adj
307 #pragma redefine_extname         ASN1_UTCTIME_check sunw_ASN1_UTCTIME_check
308 #pragma redefine_extname         ASN1_UTCTIME_cmp_time_t sunw_ASN1_UTCTIME_cmp_ti
309 #pragma redefine_extname         ASN1_UTCTIME_free sunw_ASN1_UTCTIME_free
310 #pragma redefine_extname         ASN1_UTCTIME_it sunw_ASN1_UTCTIME_it
311 #pragma redefine_extname         ASN1_UTCTIME_new sunw_ASN1_UTCTIME_new
312 #pragma redefine_extname         ASN1_UTCTIME_print sunw_ASN1_UTCTIME_print
313 #pragma redefine_extname         ASN1_UTCTIME_set sunw_ASN1_UTCTIME_set
314 #pragma redefine_extname         ASN1_UTCTIME_set_string sunw_ASN1_UTCTIME_set_st
315 #pragma redefine_extname         ASN1_UTF8STRING_free sunw_ASN1_UTF8STRING_free
316 #pragma redefine_extname         ASN1_UTF8STRING_it sunw_ASN1_UTF8STRING_it
317 #pragma redefine_extname         ASN1_UTF8STRING_new sunw_ASN1_UTF8STRING_new
318 #pragma redefine_extname         ASN1_verify sunw_ASN1_verify
319 #pragma redefine_extname         ASN1_version sunw_ASN1_version
320 #pragma redefine_extname         ASN1_VISIBLESTRING_free sunw_ASN1_VISIBLESTRING_
321 #pragma redefine_extname         ASN1_VISIBLESTRING_it sunw_ASN1_VISIBLESTRING_it
322 #pragma redefine_extname         ASN1_VISIBLESTRING_new sunw_ASN1_VISIBLESTRING_n
323 #pragma redefine_extname         AUTHORITY_INFO_ACCESS_free sunw_AUTHORITY_INFO_A
324 #pragma redefine_extname         AUTHORITY_INFO_ACCESS_it sunw_AUTHORITY_INFO_ACC
325 #pragma redefine_extname         AUTHORITY_INFO_ACCESS_new sunw_AUTHORITY_INFO_AC
```



```
326 #pragma redefine_extname         AUTHORITY_KEYID_free sunw_AUTHORITY_KEYID_free
327 #pragma redefine_extname         AUTHORITY_KEYID_it sunw_AUTHORITY_KEYID_it
328 #pragma redefine_extname         AUTHORITY_KEYID_new sunw_AUTHORITY_KEYID_new
329 #pragma redefine_extname         b2i_PrivateKey sunw_b2i_PrivateKey
330 #pragma redefine_extname         b2i_PrivateKey_bio sunw_b2i_PrivateKey_bio
331 #pragma redefine_extname         b2i_PublicKey sunw_b2i_PublicKey
332 #pragma redefine_extname         b2i_PublicKey_bio sunw_b2i_PublicKey_bio
333 #pragma redefine_extname         b2i_PVK_bio sunw_b2i_PVK_bio
334 #pragma redefine_extname         BASIC_CONSTRAINTS_free sunw_BASIC_CONSTRAINTS_fr
335 #pragma redefine_extname         BASIC_CONSTRAINTS_it sunw_BASIC_CONSTRAINTS_it
336 #pragma redefine_extname         BASIC_CONSTRAINTS_new sunw_BASIC_CONSTRAINTS_new
337 #pragma redefine_extname         BF_cbc_encrypt sunw_BF_cbc_encrypt
338 #pragma redefine_extname         BF_cfb64_encrypt sunw_BF_cfb64_encrypt
339 #pragma redefine_extname         BF_decrypt sunw_BF_decrypt
340 #pragma redefine_extname         BF_ecb_encrypt sunw_BF_ecb_encrypt
341 #pragma redefine_extname         BF_encrypt sunw_BF_encrypt
342 #pragma redefine_extname         BF_ofb64_encrypt sunw_BF_ofb64_encrypt
343 #pragma redefine_extname         BF_options sunw_BF_options
344 #pragma redefine_extname         BF_set_key sunw_BF_set_key
345 #pragma redefine_extname         BF_version sunw_BF_version
346 #pragma redefine_extname         BIGNUM_it sunw_BIGNUM_it
347 #pragma redefine_extname         BIO_accept sunw_BIO_accept
348 #pragma redefine_extname         BIO_asn1_get_prefix sunw_BIO_asn1_get_prefix
349 #pragma redefine_extname         BIO_asn1_get_suffix sunw_BIO_asn1_get_suffix
350 #pragma redefine_extname         BIO_asn1_set_prefix sunw_BIO_asn1_set_prefix
351 #pragma redefine_extname         BIO_asn1_set_suffix sunw_BIO_asn1_set_suffix
352 #pragma redefine_extname         BIO_callback_ctrl sunw_BIO_callback_ctrl
353 #pragma redefine_extname         BIO_clear_flags sunw_BIO_clear_flags
354 #pragma redefine_extname         BIO_CONNECT_free sunw_BIO_CONNECT_free
355 #pragma redefine_extname         BIO_CONNECT_new sunw_BIO_CONNECT_new
356 #pragma redefine_extname         BIO_copy_next_retry sunw_BIO_copy_next_retry
357 #pragma redefine_extname         BIO_ctrl sunw_BIO_ctrl
358 #pragma redefine_extname         BIO_ctrl_get_read_request sunw_BIO_ctrl_get_read
359 #pragma redefine_extname         BIO_ctrl_get_write_guarantee sunw_BIO_ctrl_get_w
360 #pragma redefine_extname         BIO_ctrl_pending sunw_BIO_ctrl_pending
361 #pragma redefine_extname         BIO_ctrl_reset_read_request sunw_BIO_ctrl_reset_
362 #pragma redefine_extname         BIO_ctrl_wpending sunw_BIO_ctrl_wpending
363 #pragma redefine_extname         BIO_debug_callback sunw_BIO_debug_callback
364 #pragma redefine_extname         BIO_dgram_non_fatal_error sunw_BIO_dgram_non_fat
365 #pragma redefine_extname         BIO_dump sunw_BIO_dump
366 #pragma redefine_extname         BIO_dump_cb sunw_BIO_dump_cb
367 #pragma redefine_extname         BIO_dump_fp sunw_BIO_dump_fp
368 #pragma redefine_extname         BIO_dump_indent sunw_BIO_dump_indent
369 #pragma redefine_extname         BIO_dump_indent_cb sunw_BIO_dump_indent_cb
370 #pragma redefine_extname         BIO_dump_indent_fp sunw_BIO_dump_indent_fp
371 #pragma redefine_extname         BIO_dup_chain sunw_BIO_dup_chain
372 #pragma redefine_extname         BIO_f_asn1 sunw_BIO_f_asn1
373 #pragma redefine_extname         BIO_f_base64 sunw_BIO_f_base64
374 #pragma redefine_extname         BIO_f_buffer sunw_BIO_f_buffer
375 #pragma redefine_extname         BIO_f_cipher sunw_BIO_f_cipher
376 #pragma redefine_extname         BIO_f_md sunw_BIO_f_md
377 #pragma redefine_extname         BIO_f_nbio_test sunw_BIO_f_nbio_test
378 #pragma redefine_extname         BIO_f_null sunw_BIO_f_null
379 #pragma redefine_extname         BIO_f_reliable sunw_BIO_f_reliable
380 #pragma redefine_extname         BIO_fd_non_fatal_error sunw_BIO_fd_non_fatal_err
381 #pragma redefine_extname         BIO_fd_should_retry sunw_BIO_fd_should_retry
382 #pragma redefine_extname         BIO_find_type sunw_BIO_find_type
383 #pragma redefine_extname         BIO_free sunw_BIO_free
384 #pragma redefine_extname         BIO_free_all sunw_BIO_free_all
385 #pragma redefine_extname         BIO_get_accept_socket sunw_BIO_get_accept_socket
386 #pragma redefine_extname         BIO_get_callback sunw_BIO_get_callback
387 #pragma redefine_extname         BIO_get_callback_arg sunw_BIO_get_callback_arg
388 #pragma redefine_extname         BIO_get_ex_data sunw_BIO_get_ex_data
389 #pragma redefine_extname         BIO_get_ex_new_index sunw_BIO_get_ex_new_index
390 #pragma redefine_extname         BIO_get_host_ip sunw_BIO_get_host_ip
391 #pragma redefine_extname         BIO_get_port sunw_BIO_get_port
```

```
392 #pragma redefine_extname        BIO_get_retry_BIO sunw_BIO_get_retry_BIO
393 #pragma redefine_extname        BIO_get_retry_reason sunw_BIO_get_retry_reason
394 #pragma redefine_extname        BIO_gethostbyname sunw_BIO_gethostbyname
395 #pragma redefine_extname        BIO_gets sunw_BIO_gets
396 #pragma redefine_extname        BIO_indent sunw_BIO_indent
397 #pragma redefine_extname        BIO_int_ctrl sunw_BIO_int_ctrl
398 #pragma redefine_extname        BIO_method_name sunw_BIO_method_name
399 #pragma redefine_extname        BIO_method_type sunw_BIO_method_type
400 #pragma redefine_extname        BIO_new sunw_BIO_new
401 #pragma redefine_extname        BIO_new_accept sunw_BIO_new_accept
402 #pragma redefine_extname        BIO_new_bio_pair sunw_BIO_new_bio_pair
403 #pragma redefine_extname        BIO_new_CMS sunw_BIO_new_CMS
404 #pragma redefine_extname        BIO_new_connect sunw_BIO_new_connect
405 #pragma redefine_extname        BIO_new_dgram sunw_BIO_new_dgram
406 #pragma redefine_extname        BIO_new_fd sunw_BIO_new_fd
407 #pragma redefine_extname        BIO_new_file sunw_BIO_new_file
408 #pragma redefine_extname        BIO_new_fp sunw_BIO_new_fp
409 #pragma redefine_extname        BIO_new_mem_buf sunw_BIO_new_mem_buf
410 #pragma redefine_extname        BIO_new_NDEF sunw_BIO_new_NDEF
411 #pragma redefine_extname        BIO_new_PKCS7 sunw_BIO_new_PKCS7
412 #pragma redefine_extname        BIO_new_socket sunw_BIO_new_socket
413 #pragma redefine_extname        BIO_next sunw_BIO_next
414 #pragma redefine_extname        BIO_nread sunw_BIO_nread
415 #pragma redefine_extname        BIO_nread0 sunw_BIO_nread0
416 #pragma redefine_extname        BIO_number_read sunw_BIO_number_read
417 #pragma redefine_extname        BIO_number_written sunw_BIO_number_written
418 #pragma redefine_extname        BIO_nwrite sunw_BIO_nwrite
419 #pragma redefine_extname        BIO_nwrite0 sunw_BIO_nwrite0
420 #pragma redefine_extname        BIO_pop sunw_BIO_pop
421 #pragma redefine_extname        BIO_printf sunw_BIO_printf
422 #pragma redefine_extname        BIO_ptr_ctrl sunw_BIO_ptr_ctrl
423 #pragma redefine_extname        BIO_push sunw_BIO_push
424 #pragma redefine_extname        BIO_puts sunw_BIO_puts
425 #pragma redefine_extname        BIO_read sunw_BIO_read
426 #pragma redefine_extname        BIO_s_accept sunw_BIO_s_accept
427 #pragma redefine_extname        BIO_s_bio sunw_BIO_s_bio
428 #pragma redefine_extname        BIO_s_connect sunw_BIO_s_connect
429 #pragma redefine_extname        BIO_s_datagram sunw_BIO_s_datagram
430 #pragma redefine_extname        BIO_s_fd sunw_BIO_s_fd
431 #pragma redefine_extname        BIO_s_file sunw_BIO_s_file
432 #pragma redefine_extname        BIO_s_log sunw_BIO_s_log
433 #pragma redefine_extname        BIO_s_mem sunw_BIO_s_mem
434 #pragma redefine_extname        BIO_s_null sunw_BIO_s_null
435 #pragma redefine_extname        BIO_s_socket sunw_BIO_s_socket
436 #pragma redefine_extname        BIO_set sunw_BIO_set
437 #pragma redefine_extname        BIO_set_callback sunw_BIO_set_callback
438 #pragma redefine_extname        BIO_set_callback_arg sunw_BIO_set_callback_arg
439 #pragma redefine_extname        BIO_set_cipher sunw_BIO_set_cipher
440 #pragma redefine_extname        BIO_set_ex_data sunw_BIO_set_ex_data
441 #pragma redefine_extname        BIO_set_flags sunw_BIO_set_flags
442 #pragma redefine_extname        BIO_set_tcp_ndelay sunw_BIO_set_tcp_ndelay
443 #pragma redefine_extname        BIO_snprintf sunw_BIO_snprintf
444 #pragma redefine_extname        BIO_sock_cleanup sunw_BIO_sock_cleanup
445 #pragma redefine_extname        BIO_sock_error sunw_BIO_sock_error
446 #pragma redefine_extname        BIO_sock_init sunw_BIO_sock_init
447 #pragma redefine_extname        BIO_sock_non_fatal_error sunw_BIO_sock_non_fatal
448 #pragma redefine_extname        BIO_sock_should_retry sunw_BIO_sock_should_retry
449 #pragma redefine_extname        BIO_socket_ioctl sunw_BIO_socket_ioctl
450 #pragma redefine_extname        BIO_socket_nbio sunw_BIO_socket_nbio
451 #pragma redefine_extname        BIO_test_flags sunw_BIO_test_flags
452 #pragma redefine_extname        BIO_vfree sunw_BIO_vfree
453 #pragma redefine_extname        BIO_vprintf sunw_BIO_vprintf
454 #pragma redefine_extname        BIO_vsnprintf sunw_BIO_vsnprintf
455 #pragma redefine_extname        BIO_write sunw_BIO_write
456 #pragma redefine_extname        BN_add sunw_BN_add
457 #pragma redefine_extname        bn_add_part_words sunw_bn_add_part_words
```



```
458 #pragma redefine_extname        BN_add_word sunw_BN_add_word
459 #pragma redefine_extname        bn_add_words sunw_bn_add_words
460 #pragma redefine_extname        BN_asc2bn sunw_BN_asc2bn
461 #pragma redefine_extname        BN_bin2bn sunw_BN_bin2bn
462 #pragma redefine_extname        BN_BLINDING_convert sunw_BN_BLINDING_convert
463 #pragma redefine_extname        BN_BLINDING_convert_ex sunw_BN_BLINDING_convert_
464 #pragma redefine_extname        BN_BLINDING_create_param sunw_BN_BLINDING_create
465 #pragma redefine_extname        BN_BLINDING_free sunw_BN_BLINDING_free
466 #pragma redefine_extname        BN_BLINDING_get_flags sunw_BN_BLINDING_get_flags
467 #pragma redefine_extname        BN_BLINDING_get_thread_id sunw_BN_BLINDING_get_t
468 #pragma redefine_extname        BN_BLINDING_invert sunw_BN_BLINDING_invert
469 #pragma redefine_extname        BN_BLINDING_invert_ex sunw_BN_BLINDING_invert_ex
470 #pragma redefine_extname        BN_BLINDING_new sunw_BN_BLINDING_new
471 #pragma redefine_extname        BN_BLINDING_set_flags sunw_BN_BLINDING_set_flags
472 #pragma redefine_extname        BN_BLINDING_set_thread_id sunw_BN_BLINDING_set_t
473 #pragma redefine_extname        BN_BLINDING_thread_id sunw_BN_BLINDING_thread_id
474 #pragma redefine_extname        BN_BLINDING_update sunw_BN_BLINDING_update
475 #pragma redefine_extname        BN_bn2bin sunw_BN_bn2bin
476 #pragma redefine_extname        BN_bn2dec sunw_BN_bn2dec
477 #pragma redefine_extname        BN_bn2hex sunw_BN_bn2hex
478 #pragma redefine_extname        BN_bn2mpi sunw_BN_bn2mpi
479 #pragma redefine_extname        BN_bntest_rand sunw_BN_bntest_rand
480 #pragma redefine_extname        BN_clear sunw_BN_clear
481 #pragma redefine_extname        BN_clear_bit sunw_BN_clear_bit
482 #pragma redefine_extname        BN_clear_free sunw_BN_clear_free
483 #pragma redefine_extname        BN_cmp sunw_BN_cmp
484 #pragma redefine_extname        bn_cmp_part_words sunw_bn_cmp_part_words
485 #pragma redefine_extname        bn_cmp_words sunw_bn_cmp_words
486 #pragma redefine_extname        BN_consttime_swap sunw_BN_consttime_swap
487 #pragma redefine_extname        BN_copy sunw_BN_copy
488 #pragma redefine_extname        BN_CTX_end sunw_BN_CTX_end
489 #pragma redefine_extname        BN_CTX_free sunw_BN_CTX_free
490 #pragma redefine_extname        BN_CTX_get sunw_BN_CTX_get
491 #pragma redefine_extname        BN_CTX_init sunw_BN_CTX_init
492 #pragma redefine_extname        BN_CTX_new sunw_BN_CTX_new
493 #pragma redefine_extname        BN_CTX_start sunw_BN_CTX_start
494 #pragma redefine_extname        BN_dec2bn sunw_BN_dec2bn
495 #pragma redefine_extname        BN_div sunw_BN_div
496 #pragma redefine_extname        BN_div_recp sunw_BN_div_recp
497 #pragma redefine_extname        BN_div_word sunw_BN_div_word
498 #pragma redefine_extname        bn_div_words sunw_bn_div_words
499 #pragma redefine_extname        BN_dup sunw_BN_dup
500 #pragma redefine_extname        bn_dup_expand sunw_bn_dup_expand
501 #pragma redefine_extname        BN_exp sunw_BN_exp
502 #pragma redefine_extname        bn_expand2 sunw_bn_expand2
503 #pragma redefine_extname        BN_free sunw_BN_free
504 #pragma redefine_extname        BN_from_montgomery sunw_BN_from_montgomery
505 #pragma redefine_extname        bn_gather5 sunw_bn_gather5
506 #pragma redefine_extname        BN_gcd sunw_BN_gcd
507 #pragma redefine_extname        BN_GENCB_call sunw_BN_GENCB_call
508 #pragma redefine_extname        BN_generate_prime sunw_BN_generate_prime
509 #pragma redefine_extname        BN_generate_prime_ex sunw_BN_generate_prime_ex
510 #pragma redefine_extname        BN_get_params sunw_BN_get_params
511 #pragma redefine_extname        BN_get_word sunw_BN_get_word
512 #pragma redefine_extname        BN_get0_nist_prime_192 sunw_BN_get0_nist_prime_1
513 #pragma redefine_extname        BN_get0_nist_prime_224 sunw_BN_get0_nist_prime_2
514 #pragma redefine_extname        BN_get0_nist_prime_256 sunw_BN_get0_nist_prime_2
515 #pragma redefine_extname        BN_get0_nist_prime_384 sunw_BN_get0_nist_prime_3
516 #pragma redefine_extname        BN_get0_nist_prime_521 sunw_BN_get0_nist_prime_5
517 #pragma redefine_extname        BN_GF2m_add sunw_BN_GF2m_add
518 #pragma redefine_extname        BN_GF2m_arr2poly sunw_BN_GF2m_arr2poly
519 #pragma redefine_extname        BN_GF2m_mod sunw_BN_GF2m_mod
520 #pragma redefine_extname        BN_GF2m_mod_arr sunw_BN_GF2m_mod_arr
521 #pragma redefine_extname        BN_GF2m_mod_div sunw_BN_GF2m_mod_div
522 #pragma redefine_extname        BN_GF2m_mod_div_arr sunw_BN_GF2m_mod_div_arr
523 #pragma redefine_extname        BN_GF2m_mod_exp sunw_BN_GF2m_mod_exp
```

```
524 #pragma redefine_extname         BN_GF2m_mod_exp_arr sunw_BN_GF2m_mod_exp_arr
525 #pragma redefine_extname         BN_GF2m_mod_inv sunw_BN_GF2m_mod_inv
526 #pragma redefine_extname         BN_GF2m_mod_inv_arr sunw_BN_GF2m_mod_inv_arr
527 #pragma redefine_extname         BN_GF2m_mod_mul sunw_BN_GF2m_mod_mul
528 #pragma redefine_extname         BN_GF2m_mod_mul_arr sunw_BN_GF2m_mod_mul_arr
529 #pragma redefine_extname         BN_GF2m_mod_solve_quad sunw_BN_GF2m_mod_solve_qu
530 #pragma redefine_extname         BN_GF2m_mod_solve_quad_arr sunw_BN_GF2m_mod_solv
531 #pragma redefine_extname         BN_GF2m_mod_sqr sunw_BN_GF2m_mod_sqr
532 #pragma redefine_extname         BN_GF2m_mod_sqr_arr sunw_BN_GF2m_mod_sqr_arr
533 #pragma redefine_extname         BN_GF2m_mod_sqrt sunw_BN_GF2m_mod_sqrt
534 #pragma redefine_extname         BN_GF2m_mod_sqrt_arr sunw_BN_GF2m_mod_sqrt_arr
535 #pragma redefine_extname         bn_GF2m_mul_2x2 sunw_bn_GF2m_mul_2x2
536 #pragma redefine_extname         BN_GF2m_poly2arr sunw_BN_GF2m_poly2arr
537 #pragma redefine_extname         BN_hex2bn sunw_BN_hex2bn
538 #pragma redefine_extname         BN_init sunw_BN_init
539 #pragma redefine_extname         BN_is_bit_set sunw_BN_is_bit_set
540 #pragma redefine_extname         BN_is_prime sunw_BN_is_prime
541 #pragma redefine_extname         BN_is_prime_ex sunw_BN_is_prime_ex
542 #pragma redefine_extname         BN_is_prime_fasttest sunw_BN_is_prime_fasttest
543 #pragma redefine_extname         BN_is_prime_fasttest_ex sunw_BN_is_prime_fasttes
544 #pragma redefine_extname         BN_kronecker sunw_BN_kronecker
545 #pragma redefine_extname         BN_lshift sunw_BN_lshift
546 #pragma redefine_extname         BN_lshift1 sunw_BN_lshift1
547 #pragma redefine_extname         BN_mask_bits sunw_BN_mask_bits
548 #pragma redefine_extname         BN_mod_add sunw_BN_mod_add
549 #pragma redefine_extname         BN_mod_add_quick sunw_BN_mod_add_quick
550 #pragma redefine_extname         BN_mod_exp sunw_BN_mod_exp
551 #pragma redefine_extname         BN_mod_exp_mont sunw_BN_mod_exp_mont
552 #pragma redefine_extname         BN_mod_exp_mont_consttime sunw_BN_mod_exp_mont_c
553 #pragma redefine_extname         BN_mod_exp_mont_word sunw_BN_mod_exp_mont_word
554 #pragma redefine_extname         BN_mod_exp_recp sunw_BN_mod_exp_recp
555 #pragma redefine_extname         BN_mod_exp_simple sunw_BN_mod_exp_simple
556 #pragma redefine_extname         BN_mod_exp2_mont sunw_BN_mod_exp2_mont
557 #pragma redefine_extname         BN_mod_inverse sunw_BN_mod_inverse
558 #pragma redefine_extname         BN_mod_lshift sunw_BN_mod_lshift
559 #pragma redefine_extname         BN_mod_lshift_quick sunw_BN_mod_lshift_quick
560 #pragma redefine_extname         BN_mod_lshift1 sunw_BN_mod_lshift1
561 #pragma redefine_extname         BN_mod_lshift1_quick sunw_BN_mod_lshift1_quick
562 #pragma redefine_extname         BN_mod_mul sunw_BN_mod_mul
563 #pragma redefine_extname         BN_mod_mul_montgomery sunw_BN_mod_mul_montgomery
564 #pragma redefine_extname         BN_mod_mul_reciprocal sunw_BN_mod_mul_reciprocal
565 #pragma redefine_extname         BN_mod_sqr sunw_BN_mod_sqr
566 #pragma redefine_extname         BN_mod_sqrt sunw_BN_mod_sqrt
567 #pragma redefine_extname         BN_mod_sub sunw_BN_mod_sub
568 #pragma redefine_extname         BN_mod_sub_quick sunw_BN_mod_sub_quick
569 #pragma redefine_extname         BN_mod_word sunw_BN_mod_word
570 #pragma redefine_extname         BN_MONT_CTX_copy sunw_BN_MONT_CTX_copy
571 #pragma redefine_extname         BN_MONT_CTX_free sunw_BN_MONT_CTX_free
572 #pragma redefine_extname         BN_MONT_CTX_init sunw_BN_MONT_CTX_init
573 #pragma redefine_extname         BN_MONT_CTX_new sunw_BN_MONT_CTX_new
574 #pragma redefine_extname         BN_MONT_CTX_set sunw_BN_MONT_CTX_set
575 #pragma redefine_extname         BN_MONT_CTX_set_locked sunw_BN_MONT_CTX_set_lock
576 #pragma redefine_extname         BN_mpi2bn sunw_BN_mpi2bn
577 #pragma redefine_extname         BN_mul sunw_BN_mul
578 #pragma redefine_extname         bn_mul_add_words sunw_bn_mul_add_words
579 #pragma redefine_extname         bn_mul_comba4 sunw_bn_mul_comba4
580 #pragma redefine_extname         bn_mul_comba8 sunw_bn_mul_comba8
581 #pragma redefine_extname         bn_mul_high sunw_bn_mul_high
582 #pragma redefine_extname         bn_mul_low_normal sunw_bn_mul_low_normal
583 #pragma redefine_extname         bn_mul_low_recursive sunw_bn_mul_low_recursive
584 #pragma redefine_extname         bn_mul_mont sunw_bn_mul_mont
585 #pragma redefine_extname         bn_mul_mont_gather5 sunw_bn_mul_mont_gather5
586 #pragma redefine_extname         bn_mul_normal sunw_bn_mul_normal
587 #pragma redefine_extname         bn_mul_part_recursive sunw_bn_mul_part_recursive
588 #pragma redefine_extname         bn_mul_recursive sunw_bn_mul_recursive
589 #pragma redefine_extname         BN_mul_word sunw_BN_mul_word
```



```
590 #pragma redefine_extname         bn_mul_words sunw_bn_mul_words
591 #pragma redefine_extname         BN_new sunw_BN_new
592 #pragma redefine_extname         BN_nist_mod_192 sunw_BN_nist_mod_192
593 #pragma redefine_extname         BN_nist_mod_224 sunw_BN_nist_mod_224
594 #pragma redefine_extname         BN_nist_mod_256 sunw_BN_nist_mod_256
595 #pragma redefine_extname         BN_nist_mod_384 sunw_BN_nist_mod_384
596 #pragma redefine_extname         BN_nist_mod_521 sunw_BN_nist_mod_521
597 #pragma redefine_extname         BN_nnmod sunw_BN_nnmod
598 #pragma redefine_extname         BN_num_bits sunw_BN_num_bits
599 #pragma redefine_extname         BN_num_bits_word sunw_BN_num_bits_word
600 #pragma redefine_extname         BN_options sunw_BN_options
601 #pragma redefine_extname         BN_print sunw_BN_print
602 #pragma redefine_extname         BN_print_fp sunw_BN_print_fp
603 #pragma redefine_extname         BN_pseudo_rand sunw_BN_pseudo_rand
604 #pragma redefine_extname         BN_pseudo_rand_range sunw_BN_pseudo_rand_range
605 #pragma redefine_extname         BN_rand sunw_BN_rand
606 #pragma redefine_extname         BN_rand_range sunw_BN_rand_range
607 #pragma redefine_extname         BN_reciprocal sunw_BN_reciprocal
608 #pragma redefine_extname         BN_RECP_CTX_free sunw_BN_RECP_CTX_free
609 #pragma redefine_extname         BN_RECP_CTX_init sunw_BN_RECP_CTX_init
610 #pragma redefine_extname         BN_RECP_CTX_new sunw_BN_RECP_CTX_new
611 #pragma redefine_extname         BN_RECP_CTX_set sunw_BN_RECP_CTX_set
612 #pragma redefine_extname         BN_rshift sunw_BN_rshift
613 #pragma redefine_extname         BN_rshift1 sunw_BN_rshift1
614 #pragma redefine_extname         bn_scatter5 sunw_bn_scatter5
615 #pragma redefine_extname         BN_set_bit sunw_BN_set_bit
616 #pragma redefine_extname         BN_set_negative sunw_BN_set_negative
617 #pragma redefine_extname         BN_set_params sunw_BN_set_params
618 #pragma redefine_extname         BN_set_word sunw_BN_set_word
619 #pragma redefine_extname         BN_sqr sunw_BN_sqr
620 #pragma redefine_extname         bn_sqr_comba4 sunw_bn_sqr_comba4
621 #pragma redefine_extname         bn_sqr_comba8 sunw_bn_sqr_comba8
622 #pragma redefine_extname         bn_sqr_normal sunw_bn_sqr_normal
623 #pragma redefine_extname         bn_sqr_recursive sunw_bn_sqr_recursive
624 #pragma redefine_extname         bn_sqr_words sunw_bn_sqr_words
625 #pragma redefine_extname         BN_sub sunw_BN_sub
626 #pragma redefine_extname         bn_sub_part_words sunw_bn_sub_part_words
627 #pragma redefine_extname         BN_sub_word sunw_BN_sub_word
628 #pragma redefine_extname         bn_sub_words sunw_bn_sub_words
629 #pragma redefine_extname         BN_swap sunw_BN_swap
630 #pragma redefine_extname         BN_to_ASN1_ENUMERATED sunw_BN_to_ASN1_ENUMERATED
631 #pragma redefine_extname         BN_to_ASN1_INTEGER sunw_BN_to_ASN1_INTEGER
632 #pragma redefine_extname         BN_uadd sunw_BN_uadd
633 #pragma redefine_extname         BN_ucmp sunw_BN_ucmp
634 #pragma redefine_extname         BN_usub sunw_BN_usub
635 #pragma redefine_extname         BN_value_one sunw_BN_value_one
636 #pragma redefine_extname         BN_version sunw_BN_version
637 #pragma redefine_extname         BN_X931_derive_prime_ex sunw_BN_X931_derive_prim
638 #pragma redefine_extname         BN_X931_generate_prime_ex sunw_BN_X931_generate_
639 #pragma redefine_extname         BN_X931_generate_Xpq sunw_BN_X931_generate_Xpq
640 #pragma redefine_extname         bsaes_cbc_encrypt sunw_bsaes_cbc_encrypt
641 #pragma redefine_extname         bsaes_ctr32_encrypt_blocks sunw_bsaes_ctr32_encr
642 #pragma redefine_extname         bsaes_xts_decrypt sunw_bsaes_xts_decrypt
643 #pragma redefine_extname         bsaes_xts_encrypt sunw_bsaes_xts_encrypt
644 #pragma redefine_extname         BUF_MEM_free sunw_BUF_MEM_free
645 #pragma redefine_extname         BUF_MEM_grow sunw_BUF_MEM_grow
646 #pragma redefine_extname         BUF_MEM_grow_clean sunw_BUF_MEM_grow_clean
647 #pragma redefine_extname         BUF_MEM_new sunw_BUF_MEM_new
648 #pragma redefine_extname         BUF_memdup sunw_BUF_memdup
649 #pragma redefine_extname         BUF_reverse sunw_BUF_reverse
650 #pragma redefine_extname         BUF_strdup sunw_BUF_strdup
651 #pragma redefine_extname         BUF_strlcat sunw_BUF_strlcat
652 #pragma redefine_extname         BUF_strlcpy sunw_BUF_strlcpy
653 #pragma redefine_extname         BUF_strndup sunw_BUF_strndup
654 #pragma redefine_extname         c2i_ASN1_BIT_STRING sunw_c2i_ASN1_BIT_STRING
655 #pragma redefine_extname         c2i_ASN1_INTEGER sunw_c2i_ASN1_INTEGER
```

```
656 #pragma redefine_extname         c2i_ASN1_OBJECT sunw_c2i_ASN1_OBJECT
657 #pragma redefine_extname         Camellia_cbc_encrypt sunw_Camellia_cbc_encrypt
658 #pragma redefine_extname         Camellia_cfb1_encrypt sunw_Camellia_cfb1_encrypt
659 #pragma redefine_extname         Camellia_cfb128_encrypt sunw_Camellia_cfb128_enc
660 #pragma redefine_extname         Camellia_cfb8_encrypt sunw_Camellia_cfb8_encrypt
661 #pragma redefine_extname         Camellia_ctr128_encrypt sunw_Camellia_ctr128_enc
662 #pragma redefine_extname         Camellia_decrypt sunw_Camellia_decrypt
663 #pragma redefine_extname         Camellia_DecryptBlock sunw_Camellia_DecryptBlock
664 #pragma redefine_extname         Camellia_DecryptBlock_Rounds sunw_Camellia_Decry
665 #pragma redefine_extname         Camellia_ecb_encrypt sunw_Camellia_ecb_encrypt
666 #pragma redefine_extname         Camellia_Ekeygen sunw_Camellia_Ekeygen
667 #pragma redefine_extname         Camellia_encrypt sunw_Camellia_encrypt
668 #pragma redefine_extname         Camellia_EncryptBlock sunw_Camellia_EncryptBlock
669 #pragma redefine_extname         Camellia_EncryptBlock_Rounds sunw_Camellia_Encry
670 #pragma redefine_extname         Camellia_ofb128_encrypt sunw_Camellia_ofb128_enc
671 #pragma redefine_extname         Camellia_set_key sunw_Camellia_set_key
672 #pragma redefine_extname         CAMELLIA_version sunw_CAMELLIA_version
673 #pragma redefine_extname         CAST_cbc_encrypt sunw_CAST_cbc_encrypt
674 #pragma redefine_extname         CAST_cfb64_encrypt sunw_CAST_cfb64_encrypt
675 #pragma redefine_extname         CAST_decrypt sunw_CAST_decrypt
676 #pragma redefine_extname         CAST_ecb_encrypt sunw_CAST_ecb_encrypt
677 #pragma redefine_extname         CAST_encrypt sunw_CAST_encrypt
678 #pragma redefine_extname         CAST_ofb64_encrypt sunw_CAST_ofb64_encrypt
679 #pragma redefine_extname         CAST_S_table0 sunw_CAST_S_table0
680 #pragma redefine_extname         CAST_S_table1 sunw_CAST_S_table1
681 #pragma redefine_extname         CAST_S_table2 sunw_CAST_S_table2
682 #pragma redefine_extname         CAST_S_table3 sunw_CAST_S_table3
683 #pragma redefine_extname         CAST_S_table4 sunw_CAST_S_table4
684 #pragma redefine_extname         CAST_S_table5 sunw_CAST_S_table5
685 #pragma redefine_extname         CAST_S_table6 sunw_CAST_S_table6
686 #pragma redefine_extname         CAST_S_table7 sunw_CAST_S_table7
687 #pragma redefine_extname         CAST_set_key sunw_CAST_set_key
688 #pragma redefine_extname         CAST_version sunw_CAST_version
689 #pragma redefine_extname         CBIGNUM_it sunw_CBIGNUM_it
690 #pragma redefine_extname         CERTIFICATEPOLICIES_free sunw_CERTIFICATEPOLICIE
691 #pragma redefine_extname         CERTIFICATEPOLICIES_it sunw_CERTIFICATEPOLICIES_
692 #pragma redefine_extname         CERTIFICATEPOLICIES_new sunw_CERTIFICATEPOLICIES
693 #pragma redefine_extname         check_defer sunw_check_defer
694 #pragma redefine_extname         cmac_asn1_meth sunw_cmac_asn1_meth
695 #pragma redefine_extname         CMAC_CTX_cleanup sunw_CMAC_CTX_cleanup
696 #pragma redefine_extname         CMAC_CTX_copy sunw_CMAC_CTX_copy
697 #pragma redefine_extname         CMAC_CTX_free sunw_CMAC_CTX_free
698 #pragma redefine_extname         CMAC_CTX_get0_cipher_ctx sunw_CMAC_CTX_get0_ciph
699 #pragma redefine_extname         CMAC_CTX_new sunw_CMAC_CTX_new
700 #pragma redefine_extname         CMAC_Final sunw_CMAC_Final
701 #pragma redefine_extname         CMAC_Init sunw_CMAC_Init
702 #pragma redefine_extname         cmac_pkey_meth sunw_cmac_pkey_meth
703 #pragma redefine_extname         CMAC_resume sunw_CMAC_resume
704 #pragma redefine_extname         CMAC_Update sunw_CMAC_Update
705 #pragma redefine_extname         CMS_add_simple_smimecap sunw_CMS_add_simple_smim
706 #pragma redefine_extname         CMS_add_smimecap sunw_CMS_add_smimecap
707 #pragma redefine_extname         CMS_add_standard_smimecap sunw_CMS_add_standard_
708 #pragma redefine_extname         CMS_add0_cert sunw_CMS_add0_cert
709 #pragma redefine_extname         CMS_add0_CertificateChoices sunw_CMS_add0_Certif
710 #pragma redefine_extname         CMS_add0_crl sunw_CMS_add0_crl
711 #pragma redefine_extname         CMS_add0_recipient_key sunw_CMS_add0_recipient_k
712 #pragma redefine_extname         CMS_add0_recipient_password sunw_CMS_add0_recipi
713 #pragma redefine_extname         CMS_add0_RevocationInfoChoice sunw_CMS_add0_Revo
714 #pragma redefine_extname         CMS_add1_cert sunw_CMS_add1_cert
715 #pragma redefine_extname         CMS_add1_crl sunw_CMS_add1_crl
716 #pragma redefine_extname         CMS_add1_ReceiptRequest sunw_CMS_add1_ReceiptReq
717 #pragma redefine_extname         CMS_add1_recipient_cert sunw_CMS_add1_recipient_
718 #pragma redefine_extname         CMS_add1_signer sunw_CMS_add1_signer
719 #pragma redefine_extname         CMS_Attributes_Sign_it sunw_CMS_Attributes_Sign_
720 #pragma redefine_extname         CMS_Attributes_Verify_it sunw_CMS_Attributes_Ver
721 #pragma redefine_extname         CMS_AuthenticatedData_it sunw_CMS_AuthenticatedD
```



```
722 #pragma redefine_extname         CMS_CertificateChoices_it sunw_CMS_CertificateCh
723 #pragma redefine_extname         CMS_compress sunw_CMS_compress
724 #pragma redefine_extname         CMS_CompressedData_it sunw_CMS_CompressedData_it
725 #pragma redefine_extname         cms_content_bio sunw_cms_content_bio
726 #pragma redefine_extname         CMS_ContentInfo_free sunw_CMS_ContentInfo_free
727 #pragma redefine_extname         CMS_ContentInfo_it sunw_CMS_ContentInfo_it
728 #pragma redefine_extname         CMS_ContentInfo_new sunw_CMS_ContentInfo_new
729 #pragma redefine_extname         CMS_ContentInfo_print_ctx sunw_CMS_ContentInfo_p
730 #pragma redefine_extname         CMS_data sunw_CMS_data
731 #pragma redefine_extname         cms_Data_create sunw_cms_Data_create
732 #pragma redefine_extname         CMS_data_create sunw_CMS_data_create
733 #pragma redefine_extname         CMS_dataFinal sunw_CMS_dataFinal
734 #pragma redefine_extname         CMS_dataInit sunw_CMS_dataInit
735 #pragma redefine_extname         CMS_decrypt sunw_CMS_decrypt
736 #pragma redefine_extname         CMS_decrypt_set1_key sunw_CMS_decrypt_set1_key
737 #pragma redefine_extname         CMS_decrypt_set1_password sunw_CMS_decrypt_set1_
738 #pragma redefine_extname         CMS_decrypt_set1_pkey sunw_CMS_decrypt_set1_pkey
739 #pragma redefine_extname         CMS_digest_create sunw_CMS_digest_create
740 #pragma redefine_extname         CMS_digest_verify sunw_CMS_digest_verify
741 #pragma redefine_extname         cms_DigestAlgorithm_find_ctx sunw_cms_DigestAlgo
742 #pragma redefine_extname         cms_DigestAlgorithm_init_bio sunw_cms_DigestAlgo
743 #pragma redefine_extname         cms_DigestAlgorithm_set sunw_cms_DigestAlgorithm
744 #pragma redefine_extname         cms_DigestedData_create sunw_cms_DigestedData_cr
745 #pragma redefine_extname         cms_DigestedData_do_final sunw_cms_DigestedData_
746 #pragma redefine_extname         cms_DigestedData_init_bio sunw_cms_DigestedData_
747 #pragma redefine_extname         CMS_DigestedData_it sunw_CMS_DigestedData_it
748 #pragma redefine_extname         CMS_EncapsulatedContentInfo_it sunw_CMS_Encapsul
749 #pragma redefine_extname         cms_encode_Receipt sunw_cms_encode_Receipt
750 #pragma redefine_extname         CMS_encrypt sunw_CMS_encrypt
751 #pragma redefine_extname         cms_EncryptedContent_init sunw_cms_EncryptedCont
752 #pragma redefine_extname         cms_EncryptedContent_init_bio sunw_cms_Encrypted
753 #pragma redefine_extname         CMS_EncryptedContentInfo_it sunw_CMS_EncryptedCo
754 #pragma redefine_extname         CMS_EncryptedData_decrypt sunw_CMS_EncryptedData
755 #pragma redefine_extname         CMS_EncryptedData_encrypt sunw_CMS_EncryptedData
756 #pragma redefine_extname         cms_EncryptedData_init_bio sunw_cms_EncryptedDat
757 #pragma redefine_extname         CMS_EncryptedData_it sunw_CMS_EncryptedData_it
758 #pragma redefine_extname         CMS_EncryptedData_set1_key sunw_CMS_EncryptedDat
759 #pragma redefine_extname         CMS_EnvelopedData_create sunw_CMS_EnvelopedData_
760 #pragma redefine_extname         cms_EnvelopedData_init_bio sunw_cms_EnvelopedDat
761 #pragma redefine_extname         CMS_EnvelopedData_it sunw_CMS_EnvelopedData_it
762 #pragma redefine_extname         CMS_final sunw_CMS_final
763 #pragma redefine_extname         CMS_get0_content sunw_CMS_get0_content
764 #pragma redefine_extname         CMS_get0_eContentType sunw_CMS_get0_eContentType
765 #pragma redefine_extname         cms_get0_enveloped sunw_cms_get0_enveloped
766 #pragma redefine_extname         CMS_get0_RecipientInfos sunw_CMS_get0_RecipientI
767 #pragma redefine_extname         CMS_get0_SignerInfos sunw_CMS_get0_SignerInfos
768 #pragma redefine_extname         CMS_get0_signers sunw_CMS_get0_signers
769 #pragma redefine_extname         CMS_get0_type sunw_CMS_get0_type
770 #pragma redefine_extname         CMS_get1_certs sunw_CMS_get1_certs
771 #pragma redefine_extname         CMS_get1_crls sunw_CMS_get1_crls
772 #pragma redefine_extname         CMS_get1_ReceiptRequest sunw_CMS_get1_ReceiptReq
773 #pragma redefine_extname         CMS_is_detached sunw_CMS_is_detached
774 #pragma redefine_extname         CMS_IssuerAndSerialNumber_it sunw_CMS_IssuerAndS
775 #pragma redefine_extname         CMS_KEKIdentifier_it sunw_CMS_KEKIdentifier_it
776 #pragma redefine_extname         CMS_KEKRecipientInfo_it sunw_CMS_KEKRecipientInf
777 #pragma redefine_extname         CMS_KeyAgreeRecipientIdentifier_it sunw_CMS_KeyA
778 #pragma redefine_extname         CMS_KeyAgreeRecipientInfo_it sunw_CMS_KeyAgreeRe
779 #pragma redefine_extname         CMS_KeyTransRecipientInfo_it sunw_CMS_KeyTransRe
780 #pragma redefine_extname         cms_msgSigDigest_add1 sunw_cms_msgSigDigest_add1
781 #pragma redefine_extname         CMS_OriginatorIdentifierOrKey_it sunw_CMS_Origin
782 #pragma redefine_extname         CMS_OriginatorInfo_it sunw_CMS_OriginatorInfo_it
783 #pragma redefine_extname         CMS_OriginatorPublicKey_it sunw_CMS_OriginatorPu
784 #pragma redefine_extname         CMS_OtherCertificateFormat_it sunw_CMS_OtherCert
785 #pragma redefine_extname         CMS_OtherKeyAttribute_it sunw_CMS_OtherKeyAttrib
786 #pragma redefine_extname         CMS_OtherRecipientInfo_it sunw_CMS_OtherRecipien
787 #pragma redefine_extname         CMS_OtherRevocationInfoFormat_it sunw_CMS_OtherR
```

```
788 #pragma redefine_extname         CMS_PasswordRecipientInfo_it sunw_CMS_PasswordRe
789 #pragma redefine_extname         CMS_Receipt_it sunw_CMS_Receipt_it
790 #pragma redefine_extname         cms_Receipt_verify sunw_cms_Receipt_verify
791 #pragma redefine_extname         CMS_ReceiptRequest_create0 sunw_CMS_ReceiptReque
792 #pragma redefine_extname         CMS_ReceiptRequest_free sunw_CMS_ReceiptRequest_
793 #pragma redefine_extname         CMS_ReceiptRequest_get0_values sunw_CMS_ReceiptR
794 #pragma redefine_extname         CMS_ReceiptRequest_it sunw_CMS_ReceiptRequest_it
795 #pragma redefine_extname         CMS_ReceiptRequest_new sunw_CMS_ReceiptRequest_n
796 #pragma redefine_extname         CMS_ReceiptsFrom_it sunw_CMS_ReceiptsFrom_it
797 #pragma redefine_extname         CMS_RecipientEncryptedKey_it sunw_CMS_RecipientE
798 #pragma redefine_extname         CMS_RecipientInfo_decrypt sunw_CMS_RecipientInfo
799 #pragma redefine_extname         CMS_RecipientInfo_it sunw_CMS_RecipientInfo_it
800 #pragma redefine_extname         CMS_RecipientInfo_kekri_get0_id sunw_CMS_Recipie
801 #pragma redefine_extname         CMS_RecipientInfo_kekri_id_cmp sunw_CMS_Recipien
802 #pragma redefine_extname         CMS_RecipientInfo_ktri_cert_cmp sunw_CMS_Recipie
803 #pragma redefine_extname         CMS_RecipientInfo_ktri_get0_algs sunw_CMS_Recipi
804 #pragma redefine_extname         CMS_RecipientInfo_ktri_get0_signer_id sunw_CMS_R
805 #pragma redefine_extname         cms_RecipientInfo_pwri_crypt sunw_cms_RecipientI
806 #pragma redefine_extname         CMS_RecipientInfo_set0_key sunw_CMS_RecipientInf
807 #pragma redefine_extname         CMS_RecipientInfo_set0_password sunw_CMS_Recipie
808 #pragma redefine_extname         CMS_RecipientInfo_set0_pkey sunw_CMS_RecipientIn
809 #pragma redefine_extname         CMS_RecipientInfo_type sunw_CMS_RecipientInfo_ty
810 #pragma redefine_extname         CMS_RecipientKeyIdentifier_it sunw_CMS_Recipient
811 #pragma redefine_extname         CMS_RevocationInfoChoice_it sunw_CMS_RevocationI
812 #pragma redefine_extname         CMS_set_detached sunw_CMS_set_detached
813 #pragma redefine_extname         CMS_set1_eContentType sunw_CMS_set1_eContentType
814 #pragma redefine_extname         cms_set1_SignerIdentifier sunw_cms_set1_SignerId
815 #pragma redefine_extname         CMS_set1_signers_certs sunw_CMS_set1_signers_cer
816 #pragma redefine_extname         CMS_sign sunw_CMS_sign
817 #pragma redefine_extname         CMS_sign_receipt sunw_CMS_sign_receipt
818 #pragma redefine_extname         CMS_signed_add1_attr sunw_CMS_signed_add1_attr
819 #pragma redefine_extname         CMS_signed_add1_attr_by_NID sunw_CMS_signed_add1
820 #pragma redefine_extname         CMS_signed_add1_attr_by_OBJ sunw_CMS_signed_add1
821 #pragma redefine_extname         CMS_signed_add1_attr_by_txt sunw_CMS_signed_add1
822 #pragma redefine_extname         CMS_signed_delete_attr sunw_CMS_signed_delete_at
823 #pragma redefine_extname         CMS_signed_get_attr sunw_CMS_signed_get_attr
824 #pragma redefine_extname         CMS_signed_get_attr_by_NID sunw_CMS_signed_get_a
825 #pragma redefine_extname         CMS_signed_get_attr_by_OBJ sunw_CMS_signed_get_a
826 #pragma redefine_extname         CMS_signed_get_attr_count sunw_CMS_signed_get_at
827 #pragma redefine_extname         CMS_signed_get0_data_by_OBJ sunw_CMS_signed_get0
828 #pragma redefine_extname         cms_SignedData_final sunw_cms_SignedData_final
829 #pragma redefine_extname         CMS_SignedData_init sunw_CMS_SignedData_init
830 #pragma redefine_extname         cms_SignedData_init_bio sunw_cms_SignedData_init
831 #pragma redefine_extname         CMS_SignedData_it sunw_CMS_SignedData_it
832 #pragma redefine_extname         cms_SignerIdentifier_cert_cmp sunw_cms_SignerIde
833 #pragma redefine_extname         cms_SignerIdentifier_get0_signer_id sunw_cms_Sig
834 #pragma redefine_extname         CMS_SignerIdentifier_it sunw_CMS_SignerIdentifie
835 #pragma redefine_extname         CMS_SignerInfo_cert_cmp sunw_CMS_SignerInfo_cert
836 #pragma redefine_extname         CMS_SignerInfo_get0_algs sunw_CMS_SignerInfo_get
837 #pragma redefine_extname         CMS_SignerInfo_get0_signer_id sunw_CMS_SignerInf
838 #pragma redefine_extname         CMS_SignerInfo_it sunw_CMS_SignerInfo_it
839 #pragma redefine_extname         CMS_SignerInfo_set1_signer_cert sunw_CMS_SignerI
840 #pragma redefine_extname         CMS_SignerInfo_sign sunw_CMS_SignerInfo_sign
841 #pragma redefine_extname         CMS_SignerInfo_verify sunw_CMS_SignerInfo_verify
842 #pragma redefine_extname         CMS_SignerInfo_verify_content sunw_CMS_SignerInf
843 #pragma redefine_extname         CMS_stream sunw_CMS_stream
844 #pragma redefine_extname         CMS_uncompress sunw_CMS_uncompress
845 #pragma redefine_extname         CMS_unsigned_add1_attr sunw_CMS_unsigned_add1_at
846 #pragma redefine_extname         CMS_unsigned_add1_attr_by_NID sunw_CMS_unsigned_
847 #pragma redefine_extname         CMS_unsigned_add1_attr_by_OBJ sunw_CMS_unsigned_
848 #pragma redefine_extname         CMS_unsigned_add1_attr_by_txt sunw_CMS_unsigned_
849 #pragma redefine_extname         CMS_unsigned_delete_attr sunw_CMS_unsigned_delet
850 #pragma redefine_extname         CMS_unsigned_get_attr sunw_CMS_unsigned_get_attr
851 #pragma redefine_extname         CMS_unsigned_get_attr_by_NID sunw_CMS_unsigned_g
852 #pragma redefine_extname         CMS_unsigned_get_attr_by_OBJ sunw_CMS_unsigned_g
853 #pragma redefine_extname         CMS_unsigned_get_attr_count sunw_CMS_unsigned_ge
```



```
854 #pragma redefine_extname         CMS_unsigned_get0_data_by_OBJ sunw_CMS_unsigned_
855 #pragma redefine_extname         CMS_verify sunw_CMS_verify
856 #pragma redefine_extname         CMS_verify_receipt sunw_CMS_verify_receipt
857 #pragma redefine_extname         COMP_compress_block sunw_COMP_compress_block
858 #pragma redefine_extname         COMP_CTX_free sunw_COMP_CTX_free
859 #pragma redefine_extname         COMP_CTX_new sunw_COMP_CTX_new
860 #pragma redefine_extname         COMP_expand_block sunw_COMP_expand_block
861 #pragma redefine_extname         COMP_rle sunw_COMP_rle
862 #pragma redefine_extname         COMP_zlib sunw_COMP_zlib
863 #pragma redefine_extname         COMP_zlib_cleanup sunw_COMP_zlib_cleanup
864 #pragma redefine_extname         CONF_def_version sunw_CONF_def_version
865 #pragma redefine_extname         CONF_dump_bio sunw_CONF_dump_bio
866 #pragma redefine_extname         CONF_dump_fp sunw_CONF_dump_fp
867 #pragma redefine_extname         CONF_free sunw_CONF_free
868 #pragma redefine_extname         CONF_get_number sunw_CONF_get_number
869 #pragma redefine_extname         CONF_get_section sunw_CONF_get_section
870 #pragma redefine_extname         CONF_get_string sunw_CONF_get_string
871 #pragma redefine_extname         CONF_get1_default_config_file sunw_CONF_get1_def
872 #pragma redefine_extname         CONF_imodule_get_flags sunw_CONF_imodule_get_fla
873 #pragma redefine_extname         CONF_imodule_get_module sunw_CONF_imodule_get_mo
874 #pragma redefine_extname         CONF_imodule_get_name sunw_CONF_imodule_get_name
875 #pragma redefine_extname         CONF_imodule_get_usr_data sunw_CONF_imodule_get_
876 #pragma redefine_extname         CONF_imodule_get_value sunw_CONF_imodule_get_val
877 #pragma redefine_extname         CONF_imodule_set_flags sunw_CONF_imodule_set_fla
878 #pragma redefine_extname         CONF_imodule_set_usr_data sunw_CONF_imodule_set_
879 #pragma redefine_extname         CONF_load sunw_CONF_load
880 #pragma redefine_extname         CONF_load_bio sunw_CONF_load_bio
881 #pragma redefine_extname         CONF_load_fp sunw_CONF_load_fp
882 #pragma redefine_extname         CONF_module_add sunw_CONF_module_add
883 #pragma redefine_extname         CONF_module_get_usr_data sunw_CONF_module_get_us
884 #pragma redefine_extname         CONF_module_set_usr_data sunw_CONF_module_set_us
885 #pragma redefine_extname         CONF_modules_finish sunw_CONF_modules_finish
886 #pragma redefine_extname         CONF_modules_free sunw_CONF_modules_free
887 #pragma redefine_extname         CONF_modules_load sunw_CONF_modules_load
888 #pragma redefine_extname         CONF_modules_load_file sunw_CONF_modules_load_fi
889 #pragma redefine_extname         CONF_modules_unload sunw_CONF_modules_unload
890 #pragma redefine_extname         CONF_parse_list sunw_CONF_parse_list
891 #pragma redefine_extname         CONF_set_default_method sunw_CONF_set_default_me
892 #pragma redefine_extname         CONF_set_nconf sunw_CONF_set_nconf
893 #pragma redefine_extname         CONF_version sunw_CONF_version
894 #pragma redefine_extname         CRL_DIST_POINTS_free sunw_CRL_DIST_POINTS_free
895 #pragma redefine_extname         CRL_DIST_POINTS_it sunw_CRL_DIST_POINTS_it
896 #pragma redefine_extname         CRL_DIST_POINTS_new sunw_CRL_DIST_POINTS_new
897 #pragma redefine_extname         CRYPTO_add_lock sunw_CRYPTO_add_lock
898 #pragma redefine_extname         CRYPTO_cbc128_decrypt sunw_CRYPTO_cbc128_decrypt
899 #pragma redefine_extname         CRYPTO_cbc128_encrypt sunw_CRYPTO_cbc128_encrypt
900 #pragma redefine_extname         CRYPTO_ccm128_aad sunw_CRYPTO_ccm128_aad
901 #pragma redefine_extname         CRYPTO_ccm128_decrypt sunw_CRYPTO_ccm128_decrypt
902 #pragma redefine_extname         CRYPTO_ccm128_decrypt_ccm64 sunw_CRYPTO_ccm128_d
903 #pragma redefine_extname         CRYPTO_ccm128_encrypt sunw_CRYPTO_ccm128_encrypt
904 #pragma redefine_extname         CRYPTO_ccm128_encrypt_ccm64 sunw_CRYPTO_ccm128_e
905 #pragma redefine_extname         CRYPTO_ccm128_init sunw_CRYPTO_ccm128_init
906 #pragma redefine_extname         CRYPTO_ccm128_setiv sunw_CRYPTO_ccm128_setiv
907 #pragma redefine_extname         CRYPTO_ccm128_tag sunw_CRYPTO_ccm128_tag
908 #pragma redefine_extname         CRYPTO_cfb128_1_encrypt sunw_CRYPTO_cfb128_1_enc
909 #pragma redefine_extname         CRYPTO_cfb128_8_encrypt sunw_CRYPTO_cfb128_8_enc
910 #pragma redefine_extname         CRYPTO_cfb128_encrypt sunw_CRYPTO_cfb128_encrypt
911 #pragma redefine_extname         CRYPTO_cleanup_all_ex_data sunw_CRYPTO_cleanup_a
912 #pragma redefine_extname         CRYPTO_ctr128_encrypt sunw_CRYPTO_ctr128_encrypt
913 #pragma redefine_extname         CRYPTO_ctr128_encrypt_ctr32 sunw_CRYPTO_ctr128_e
914 #pragma redefine_extname         CRYPTO_cts128_decrypt sunw_CRYPTO_cts128_decrypt
915 #pragma redefine_extname         CRYPTO_cts128_decrypt_block sunw_CRYPTO_cts128_d
916 #pragma redefine_extname         CRYPTO_cts128_encrypt sunw_CRYPTO_cts128_encrypt
917 #pragma redefine_extname         CRYPTO_cts128_encrypt_block sunw_CRYPTO_cts128_e
918 #pragma redefine_extname         CRYPTO_dbg_free sunw_CRYPTO_dbg_free
919 #pragma redefine_extname         CRYPTO_dbg_get_options sunw_CRYPTO_dbg_get_optio
```

```
#pragma redefine_extname        CRYPTO_dbg_malloc sunw_CRYPTO_dbg_malloc
#pragma redefine_extname        CRYPTO_dbg_realloc sunw_CRYPTO_dbg_realloc
#pragma redefine_extname        CRYPTO_dbg_set_options sunw_CRYPTO_dbg_set_optio
#pragma redefine_extname        CRYPTO_destroy_dynlockid sunw_CRYPTO_destroy_dyn
#pragma redefine_extname        CRYPTO_dup_ex_data sunw_CRYPTO_dup_ex_data
#pragma redefine_extname        CRYPTO_ex_data_new_class sunw_CRYPTO_ex_data_new
#pragma redefine_extname        CRYPTO_free sunw_CRYPTO_free
#pragma redefine_extname        CRYPTO_free_ex_data sunw_CRYPTO_free_ex_data
#pragma redefine_extname        CRYPTO_free_locked sunw_CRYPTO_free_locked
#pragma redefine_extname        CRYPTO_gcm128_aad sunw_CRYPTO_gcm128_aad
#pragma redefine_extname        CRYPTO_gcm128_decrypt sunw_CRYPTO_gcm128_decrypt
#pragma redefine_extname        CRYPTO_gcm128_decrypt_ctr32 sunw_CRYPTO_gcm128_d
#pragma redefine_extname        CRYPTO_gcm128_encrypt sunw_CRYPTO_gcm128_encrypt
#pragma redefine_extname        CRYPTO_gcm128_encrypt_ctr32 sunw_CRYPTO_gcm128_e
#pragma redefine_extname        CRYPTO_gcm128_finish sunw_CRYPTO_gcm128_finish
#pragma redefine_extname        CRYPTO_gcm128_init sunw_CRYPTO_gcm128_init
#pragma redefine_extname        CRYPTO_gcm128_new sunw_CRYPTO_gcm128_new
#pragma redefine_extname        CRYPTO_gcm128_release sunw_CRYPTO_gcm128_release
#pragma redefine_extname        CRYPTO_gcm128_setiv sunw_CRYPTO_gcm128_setiv
#pragma redefine_extname        CRYPTO_gcm128_tag sunw_CRYPTO_gcm128_tag
#pragma redefine_extname        CRYPTO_get_add_lock_callback sunw_CRYPTO_get_add
#pragma redefine_extname        CRYPTO_get_dynlock_create_callback sunw_CRYPTO_g
#pragma redefine_extname        CRYPTO_get_dynlock_destroy_callback sunw_CRYPTO_
#pragma redefine_extname        CRYPTO_get_dynlock_lock_callback sunw_CRYPTO_get
#pragma redefine_extname        CRYPTO_get_dynlock_value sunw_CRYPTO_get_dynlock
#pragma redefine_extname        CRYPTO_get_ex_data sunw_CRYPTO_get_ex_data
#pragma redefine_extname        CRYPTO_get_ex_data_implementation sunw_CRYPTO_ge
#pragma redefine_extname        CRYPTO_get_ex_new_index sunw_CRYPTO_get_ex_new_i
#pragma redefine_extname        CRYPTO_get_id_callback sunw_CRYPTO_get_id_callba
#pragma redefine_extname        CRYPTO_get_lock_name sunw_CRYPTO_get_lock_name
#pragma redefine_extname        CRYPTO_get_locked_mem_ex_functions sunw_CRYPTO_g
#pragma redefine_extname        CRYPTO_get_locked_mem_functions sunw_CRYPTO_get_
#pragma redefine_extname        CRYPTO_get_locking_callback sunw_CRYPTO_get_lock
#pragma redefine_extname        CRYPTO_get_mem_debug_functions sunw_CRYPTO_get_m
#pragma redefine_extname        CRYPTO_get_mem_debug_options sunw_CRYPTO_get_mem
#pragma redefine_extname        CRYPTO_get_mem_ex_functions sunw_CRYPTO_get_mem_
#pragma redefine_extname        CRYPTO_get_mem_functions sunw_CRYPTO_get_mem_fun
#pragma redefine_extname        CRYPTO_get_new_dynlockid sunw_CRYPTO_get_new_dyn
#pragma redefine_extname        CRYPTO_get_new_lockid sunw_CRYPTO_get_new_lockid
#pragma redefine_extname        CRYPTO_is_mem_check_on sunw_CRYPTO_is_mem_check_
#pragma redefine_extname        CRYPTO_lock sunw_CRYPTO_lock
#pragma redefine_extname        CRYPTO_malloc sunw_CRYPTO_malloc
#pragma redefine_extname        CRYPTO_malloc_locked sunw_CRYPTO_malloc_locked
#pragma redefine_extname        CRYPTO_mem_ctrl sunw_CRYPTO_mem_ctrl
#pragma redefine_extname        CRYPTO_mem_leaks sunw_CRYPTO_mem_leaks
#pragma redefine_extname        CRYPTO_mem_leaks_cb sunw_CRYPTO_mem_leaks_cb
#pragma redefine_extname        CRYPTO_mem_leaks_fp sunw_CRYPTO_mem_leaks_fp
#pragma redefine_extname        CRYPTO_memcmp sunw_CRYPTO_memcmp
#pragma redefine_extname        CRYPTO_new_ex_data sunw_CRYPTO_new_ex_data
#pragma redefine_extname        CRYPTO_nistcts128_decrypt sunw_CRYPTO_nistcts128
#pragma redefine_extname        CRYPTO_nistcts128_decrypt_block sunw_CRYPTO_nist
#pragma redefine_extname        CRYPTO_nistcts128_encrypt sunw_CRYPTO_nistcts128
#pragma redefine_extname        CRYPTO_nistcts128_encrypt_block sunw_CRYPTO_nist
#pragma redefine_extname        CRYPTO_num_locks sunw_CRYPTO_num_locks
#pragma redefine_extname        CRYPTO_ofb128_encrypt sunw_CRYPTO_ofb128_encrypt
#pragma redefine_extname        CRYPTO_pop_info sunw_CRYPTO_pop_info
#pragma redefine_extname        CRYPTO_push_info_ sunw_CRYPTO_push_info_
#pragma redefine_extname        CRYPTO_realloc sunw_CRYPTO_realloc
#pragma redefine_extname        CRYPTO_realloc_clean sunw_CRYPTO_realloc_clean
#pragma redefine_extname        CRYPTO_remalloc sunw_CRYPTO_remalloc
#pragma redefine_extname        CRYPTO_remove_all_info sunw_CRYPTO_remove_all_in
#pragma redefine_extname        CRYPTO_set_add_lock_callback sunw_CRYPTO_set_add
#pragma redefine_extname        CRYPTO_set_dynlock_create_callback sunw_CRYPTO_s
#pragma redefine_extname        CRYPTO_set_dynlock_destroy_callback sunw_CRYPTO_
#pragma redefine_extname        CRYPTO_set_dynlock_lock_callback sunw_CRYPTO_set
#pragma redefine_extname        CRYPTO_set_ex_data sunw_CRYPTO_set_ex_data
```



```
#pragma redefine_extname        CRYPTO_set_ex_data_implementation sunw_CRYPTO_se
#pragma redefine_extname        CRYPTO_set_id_callback sunw_CRYPTO_set_id_callba
#pragma redefine_extname        CRYPTO_set_locked_mem_ex_functions sunw_CRYPTO_s
#pragma redefine_extname        CRYPTO_set_locked_mem_functions sunw_CRYPTO_set_
#pragma redefine_extname        CRYPTO_set_locking_callback sunw_CRYPTO_set_lock
#pragma redefine_extname        CRYPTO_set_mem_debug_functions sunw_CRYPTO_set_m
#pragma redefine_extname        CRYPTO_set_mem_debug_options sunw_CRYPTO_set_mem
#pragma redefine_extname        CRYPTO_set_mem_ex_functions sunw_CRYPTO_set_mem_
#pragma redefine_extname        CRYPTO_set_mem_functions sunw_CRYPTO_set_mem_fun
#pragma redefine_extname        CRYPTO_strdup sunw_CRYPTO_strdup
#pragma redefine_extname        CRYPTO_thread_id sunw_CRYPTO_thread_id
#pragma redefine_extname        CRYPTO_THREADID_cmp sunw_CRYPTO_THREADID_cmp
#pragma redefine_extname        CRYPTO_THREADID_cpy sunw_CRYPTO_THREADID_cpy
#pragma redefine_extname        CRYPTO_THREADID_current sunw_CRYPTO_THREADID_cur
#pragma redefine_extname        CRYPTO_THREADID_get_callback sunw_CRYPTO_THREADI
#pragma redefine_extname        CRYPTO_THREADID_hash sunw_CRYPTO_THREADID_hash
#pragma redefine_extname        CRYPTO_THREADID_set_callback sunw_CRYPTO_THREADI
#pragma redefine_extname        CRYPTO_THREADID_set_numeric sunw_CRYPTO_THREADID
#pragma redefine_extname        CRYPTO_THREADID_set_pointer sunw_CRYPTO_THREADID
#pragma redefine_extname        CRYPTO_xts128_encrypt sunw_CRYPTO_xts128_encrypt
#pragma redefine_extname        d2i_ACCESS_DESCRIPTION sunw_d2i_ACCESS_DESCRIPTI
#pragma redefine_extname        d2i_ASN1_BIT_STRING sunw_d2i_ASN1_BIT_STRING
#pragma redefine_extname        d2i_ASN1_BMPSTRING sunw_d2i_ASN1_BMPSTRING
#pragma redefine_extname        d2i_ASN1_BOOLEAN sunw_d2i_ASN1_BOOLEAN
#pragma redefine_extname        d2i_ASN1_bytes sunw_d2i_ASN1_bytes
#pragma redefine_extname        d2i_ASN1_ENUMERATED sunw_d2i_ASN1_ENUMERATED
#pragma redefine_extname        d2i_ASN1_GENERALIZEDTIME sunw_d2i_ASN1_GENERALIZ
#pragma redefine_extname        d2i_ASN1_GENERALSTRING sunw_d2i_ASN1_GENERALSTRI
#pragma redefine_extname        d2i_ASN1_IA5STRING sunw_d2i_ASN1_IA5STRING
#pragma redefine_extname        d2i_ASN1_INTEGER sunw_d2i_ASN1_INTEGER
#pragma redefine_extname        d2i_ASN1_NULL sunw_d2i_ASN1_NULL
#pragma redefine_extname        d2i_ASN1_OBJECT sunw_d2i_ASN1_OBJECT
#pragma redefine_extname        d2i_ASN1_OCTET_STRING sunw_d2i_ASN1_OCTET_STRING
#pragma redefine_extname        d2i_ASN1_PRINTABLE sunw_d2i_ASN1_PRINTABLE
#pragma redefine_extname        d2i_ASN1_PRINTABLESTRING sunw_d2i_ASN1_PRINTABLE
#pragma redefine_extname        d2i_ASN1_SEQUENCE_ANY sunw_d2i_ASN1_SEQUENCE_ANY
#pragma redefine_extname        d2i_ASN1_SET sunw_d2i_ASN1_SET
#pragma redefine_extname        d2i_ASN1_SET_ANY sunw_d2i_ASN1_SET_ANY
#pragma redefine_extname        d2i_ASN1_T61STRING sunw_d2i_ASN1_T61STRING
#pragma redefine_extname        d2i_ASN1_TIME sunw_d2i_ASN1_TIME
#pragma redefine_extname        d2i_ASN1_TYPE sunw_d2i_ASN1_TYPE
#pragma redefine_extname        d2i_ASN1_type_bytes sunw_d2i_ASN1_type_bytes
#pragma redefine_extname        d2i_ASN1_UINTEGER sunw_d2i_ASN1_UINTEGER
#pragma redefine_extname        d2i_ASN1_UNIVERSALSTRING sunw_d2i_ASN1_UNIVERSAL
#pragma redefine_extname        d2i_ASN1_UTCTIME sunw_d2i_ASN1_UTCTIME
#pragma redefine_extname        d2i_ASN1_UTF8STRING sunw_d2i_ASN1_UTF8STRING
#pragma redefine_extname        d2i_ASN1_VISIBLESTRING sunw_d2i_ASN1_VISIBLESTRI
#pragma redefine_extname        d2i_AUTHORITY_INFO_ACCESS sunw_d2i_AUTHORITY_INF
#pragma redefine_extname        d2i_AUTHORITY_KEYID sunw_d2i_AUTHORITY_KEYID
#pragma redefine_extname        d2i_AutoPrivateKey sunw_d2i_AutoPrivateKey
#pragma redefine_extname        d2i_BASIC_CONSTRAINTS sunw_d2i_BASIC_CONSTRAINTS
#pragma redefine_extname        d2i_CERTIFICATEPOLICIES sunw_d2i_CERTIFICATEPOLI
#pragma redefine_extname        d2i_CMS_bio sunw_d2i_CMS_bio
#pragma redefine_extname        d2i_CMS_ContentInfo sunw_d2i_CMS_ContentInfo
#pragma redefine_extname        d2i_CMS_ReceiptRequest sunw_d2i_CMS_ReceiptReque
#pragma redefine_extname        d2i_CRL_DIST_POINTS sunw_d2i_CRL_DIST_POINTS
#pragma redefine_extname        d2i_DHparams sunw_d2i_DHparams
#pragma redefine_extname        d2i_DIRECTORYSTRING sunw_d2i_DIRECTORYSTRING
#pragma redefine_extname        d2i_DISPLAYTEXT sunw_d2i_DISPLAYTEXT
#pragma redefine_extname        d2i_DIST_POINT sunw_d2i_DIST_POINT
#pragma redefine_extname        d2i_DIST_POINT_NAME sunw_d2i_DIST_POINT_NAME
#pragma redefine_extname        d2i_DSA_PUBKEY sunw_d2i_DSA_PUBKEY
#pragma redefine_extname        d2i_DSA_PUBKEY_bio sunw_d2i_DSA_PUBKEY_bio
#pragma redefine_extname        d2i_DSA_PUBKEY_fp sunw_d2i_DSA_PUBKEY_fp
#pragma redefine_extname        d2i_DSA_SIG sunw_d2i_DSA_SIG
#pragma redefine_extname        d2i_DSAparams sunw_d2i_DSAparams
```

```
1052 #pragma redefine_extname        d2i_DSAPrivateKey sunw_d2i_DSAPrivateKey
1053 #pragma redefine_extname        d2i_DSAPrivateKey_bio sunw_d2i_DSAPrivateKey_bio
1054 #pragma redefine_extname        d2i_DSAPrivateKey_fp sunw_d2i_DSAPrivateKey_fp
1055 #pragma redefine_extname        d2i_DSAPublicKey sunw_d2i_DSAPublicKey
1056 #pragma redefine_extname        d2i_EDIPARTYNAME sunw_d2i_EDIPARTYNAME
1057 #pragma redefine_extname        d2i_ESS_CERT_ID sunw_d2i_ESS_CERT_ID
1058 #pragma redefine_extname        d2i_ESS_ISSUER_SERIAL sunw_d2i_ESS_ISSUER_SERIAL
1059 #pragma redefine_extname        d2i_ESS_SIGNING_CERT sunw_d2i_ESS_SIGNING_CERT
1060 #pragma redefine_extname        d2i_EXTENDED_KEY_USAGE sunw_d2i_EXTENDED_KEY_USA
1061 #pragma redefine_extname        d2i_GENERAL_NAME sunw_d2i_GENERAL_NAME
1062 #pragma redefine_extname        d2i_GENERAL_NAMES sunw_d2i_GENERAL_NAMES
1063 #pragma redefine_extname        d2i_ISSUING_DIST_POINT sunw_d2i_ISSUING_DIST_POI
1064 #pragma redefine_extname        d2i_KRB5_APREQ sunw_d2i_KRB5_APREQ
1065 #pragma redefine_extname        d2i_KRB5_APREQBODY sunw_d2i_KRB5_APREQBODY
1066 #pragma redefine_extname        d2i_KRB5_AUTHDATA sunw_d2i_KRB5_AUTHDATA
1067 #pragma redefine_extname        d2i_KRB5_AUTHENT sunw_d2i_KRB5_AUTHENT
1068 #pragma redefine_extname        d2i_KRB5_AUTHENTBODY sunw_d2i_KRB5_AUTHENTBODY
1069 #pragma redefine_extname        d2i_KRB5_CHECKSUM sunw_d2i_KRB5_CHECKSUM
1070 #pragma redefine_extname        d2i_KRB5_ENCDATA sunw_d2i_KRB5_ENCDATA
1071 #pragma redefine_extname        d2i_KRB5_ENCKEY sunw_d2i_KRB5_ENCKEY
1072 #pragma redefine_extname        d2i_KRB5_PRINCNAME sunw_d2i_KRB5_PRINCNAME
1073 #pragma redefine_extname        d2i_KRB5_TICKET sunw_d2i_KRB5_TICKET
1074 #pragma redefine_extname        d2i_KRB5_TKTBODY sunw_d2i_KRB5_TKTBODY
1075 #pragma redefine_extname        d2i_NETSCAPE_CERT_SEQUENCE sunw_d2i_NETSCAPE_CER
1076 #pragma redefine_extname        d2i_NETSCAPE_ENCRYPTED_PKEY sunw_d2i_NETSCAPE_EN
1077 #pragma redefine_extname        d2i_NETSCAPE_PKEY sunw_d2i_NETSCAPE_PKEY
1078 #pragma redefine_extname        d2i_Netscape_RSA sunw_d2i_Netscape_RSA
1079 #pragma redefine_extname        d2i_NETSCAPE_SPKAC sunw_d2i_NETSCAPE_SPKAC
1080 #pragma redefine_extname        d2i_NETSCAPE_SPKI sunw_d2i_NETSCAPE_SPKI
1081 #pragma redefine_extname        d2i_NETSCAPE_X509 sunw_d2i_NETSCAPE_X509
1082 #pragma redefine_extname        d2i_NOTICEREF sunw_d2i_NOTICEREF
1083 #pragma redefine_extname        d2i_OCSP_BASICRESP sunw_d2i_OCSP_BASICRESP
1084 #pragma redefine_extname        d2i_OCSP_CERTID sunw_d2i_OCSP_CERTID
1085 #pragma redefine_extname        d2i_OCSP_CERTSTATUS sunw_d2i_OCSP_CERTSTATUS
1086 #pragma redefine_extname        d2i_OCSP_CRLID sunw_d2i_OCSP_CRLID
1087 #pragma redefine_extname        d2i_OCSP_ONEREQ sunw_d2i_OCSP_ONEREQ
1088 #pragma redefine_extname        d2i_OCSP_REQINFO sunw_d2i_OCSP_REQINFO
1089 #pragma redefine_extname        d2i_OCSP_REQUEST sunw_d2i_OCSP_REQUEST
1090 #pragma redefine_extname        d2i_OCSP_RESPBYTES sunw_d2i_OCSP_RESPBYTES
1091 #pragma redefine_extname        d2i_OCSP_RESPDATA sunw_d2i_OCSP_RESPDATA
1092 #pragma redefine_extname        d2i_OCSP_RESPID sunw_d2i_OCSP_RESPID
1093 #pragma redefine_extname        d2i_OCSP_RESPONSE sunw_d2i_OCSP_RESPONSE
1094 #pragma redefine_extname        d2i_OCSP_REVOKEDINFO sunw_d2i_OCSP_REVOKEDINFO
1095 #pragma redefine_extname        d2i_OCSP_SERVICELOC sunw_d2i_OCSP_SERVICELOC
1096 #pragma redefine_extname        d2i_OCSP_SIGNATURE sunw_d2i_OCSP_SIGNATURE
1097 #pragma redefine_extname        d2i_OCSP_SINGLERESP sunw_d2i_OCSP_SINGLERESP
1098 #pragma redefine_extname        d2i_OTHERNAME sunw_d2i_OTHERNAME
1099 #pragma redefine_extname        d2i_PBE2PARAM sunw_d2i_PBE2PARAM
1100 #pragma redefine_extname        d2i_PBEPARAM sunw_d2i_PBEPARAM
1101 #pragma redefine_extname        d2i_PBKDF2PARAM sunw_d2i_PBKDF2PARAM
1102 #pragma redefine_extname        d2i_PKCS12 sunw_d2i_PKCS12
1103 #pragma redefine_extname        d2i_PKCS12_BAGS sunw_d2i_PKCS12_BAGS
1104 #pragma redefine_extname        d2i_PKCS12_bio sunw_d2i_PKCS12_bio
1105 #pragma redefine_extname        d2i_PKCS12_fp sunw_d2i_PKCS12_fp
1106 #pragma redefine_extname        d2i_PKCS12_MAC_DATA sunw_d2i_PKCS12_MAC_DATA
1107 #pragma redefine_extname        d2i_PKCS12_SAFEBAG sunw_d2i_PKCS12_SAFEBAG
1108 #pragma redefine_extname        d2i_PKCS7 sunw_d2i_PKCS7
1109 #pragma redefine_extname        d2i_PKCS7_bio sunw_d2i_PKCS7_bio
1110 #pragma redefine_extname        d2i_PKCS7_DIGEST sunw_d2i_PKCS7_DIGEST
1111 #pragma redefine_extname        d2i_PKCS7_ENC_CONTENT sunw_d2i_PKCS7_ENC_CONTENT
1112 #pragma redefine_extname        d2i_PKCS7_ENCRYPT sunw_d2i_PKCS7_ENCRYPT
1113 #pragma redefine_extname        d2i_PKCS7_ENVELOPE sunw_d2i_PKCS7_ENVELOPE
1114 #pragma redefine_extname        d2i_PKCS7_fp sunw_d2i_PKCS7_fp
1115 #pragma redefine_extname        d2i_PKCS7_ISSUER_AND_SERIAL sunw_d2i_PKCS7_ISSUE
1116 #pragma redefine_extname        d2i_PKCS7_RECIP_INFO sunw_d2i_PKCS7_RECIP_INFO
1117 #pragma redefine_extname        d2i_PKCS7_SIGN_ENVELOPE sunw_d2i_PKCS7_SIGN_ENVE
```



```
1118 #pragma redefine_extname        d2i_PKCS7_SIGNED sunw_d2i_PKCS7_SIGNED
1119 #pragma redefine_extname        d2i_PKCS7_SIGNER_INFO sunw_d2i_PKCS7_SIGNER_INFO
1120 #pragma redefine_extname        d2i_PKCS8_bio sunw_d2i_PKCS8_bio
1121 #pragma redefine_extname        d2i_PKCS8_fp sunw_d2i_PKCS8_fp
1122 #pragma redefine_extname        d2i_PKCS8_PRIV_KEY_INFO sunw_d2i_PKCS8_PRIV_KEY_
1123 #pragma redefine_extname        d2i_PKCS8_PRIV_KEY_INFO_bio sunw_d2i_PKCS8_PRIV_
1124 #pragma redefine_extname        d2i_PKCS8_PRIV_KEY_INFO_fp sunw_d2i_PKCS8_PRIV_K
1125 #pragma redefine_extname        d2i_PKCS8PrivateKey_bio sunw_d2i_PKCS8PrivateKey
1126 #pragma redefine_extname        d2i_PKCS8PrivateKey_fp sunw_d2i_PKCS8PrivateKey_
1127 #pragma redefine_extname        d2i_PKEY_USAGE_PERIOD sunw_d2i_PKEY_USAGE_PERIOD
1128 #pragma redefine_extname        d2i_POLICYINFO sunw_d2i_POLICYINFO
1129 #pragma redefine_extname        d2i_POLICYQUALINFO sunw_d2i_POLICYQUALINFO
1130 #pragma redefine_extname        d2i_PrivateKey sunw_d2i_PrivateKey
1131 #pragma redefine_extname        d2i_PrivateKey_bio sunw_d2i_PrivateKey_bio
1132 #pragma redefine_extname        d2i_PrivateKey_fp sunw_d2i_PrivateKey_fp
1133 #pragma redefine_extname        d2i_PROXY_CERT_INFO_EXTENSION sunw_d2i_PROXY_CER
1134 #pragma redefine_extname        d2i_PROXY_POLICY sunw_d2i_PROXY_POLICY
1135 #pragma redefine_extname        d2i_PUBKEY sunw_d2i_PUBKEY
1136 #pragma redefine_extname        d2i_PUBKEY_bio sunw_d2i_PUBKEY_bio
1137 #pragma redefine_extname        d2i_PUBKEY_fp sunw_d2i_PUBKEY_fp
1138 #pragma redefine_extname        d2i_PublicKey sunw_d2i_PublicKey
1139 #pragma redefine_extname        d2i_RSA_NET sunw_d2i_RSA_NET
1140 #pragma redefine_extname        d2i_RSA_PSS_PARAMS sunw_d2i_RSA_PSS_PARAMS
1141 #pragma redefine_extname        d2i_RSA_PUBKEY sunw_d2i_RSA_PUBKEY
1142 #pragma redefine_extname        d2i_RSA_PUBKEY_bio sunw_d2i_RSA_PUBKEY_bio
1143 #pragma redefine_extname        d2i_RSA_PUBKEY_fp sunw_d2i_RSA_PUBKEY_fp
1144 #pragma redefine_extname        d2i_RSAPrivateKey sunw_d2i_RSAPrivateKey
1145 #pragma redefine_extname        d2i_RSAPrivateKey_bio sunw_d2i_RSAPrivateKey_bio
1146 #pragma redefine_extname        d2i_RSAPrivateKey_fp sunw_d2i_RSAPrivateKey_fp
1147 #pragma redefine_extname        d2i_RSAPublicKey sunw_d2i_RSAPublicKey
1148 #pragma redefine_extname        d2i_RSAPublicKey_bio sunw_d2i_RSAPublicKey_bio
1149 #pragma redefine_extname        d2i_RSAPublicKey_fp sunw_d2i_RSAPublicKey_fp
1150 #pragma redefine_extname        d2i_SXNET sunw_d2i_SXNET
1151 #pragma redefine_extname        d2i_SXNETID sunw_d2i_SXNETID
1152 #pragma redefine_extname        d2i_TS_ACCURACY sunw_d2i_TS_ACCURACY
1153 #pragma redefine_extname        d2i_TS_MSG_IMPRINT sunw_d2i_TS_MSG_IMPRINT
1154 #pragma redefine_extname        d2i_TS_MSG_IMPRINT_bio sunw_d2i_TS_MSG_IMPRINT_b
1155 #pragma redefine_extname        d2i_TS_MSG_IMPRINT_fp sunw_d2i_TS_MSG_IMPRINT_fp
1156 #pragma redefine_extname        d2i_TS_REQ sunw_d2i_TS_REQ
1157 #pragma redefine_extname        d2i_TS_REQ_bio sunw_d2i_TS_REQ_bio
1158 #pragma redefine_extname        d2i_TS_REQ_fp sunw_d2i_TS_REQ_fp
1159 #pragma redefine_extname        d2i_TS_RESP sunw_d2i_TS_RESP
1160 #pragma redefine_extname        d2i_TS_RESP_bio sunw_d2i_TS_RESP_bio
1161 #pragma redefine_extname        d2i_TS_RESP_fp sunw_d2i_TS_RESP_fp
1162 #pragma redefine_extname        d2i_TS_STATUS_INFO sunw_d2i_TS_STATUS_INFO
1163 #pragma redefine_extname        d2i_TS_TST_INFO sunw_d2i_TS_TST_INFO
1164 #pragma redefine_extname        d2i_TS_TST_INFO_bio sunw_d2i_TS_TST_INFO_bio
1165 #pragma redefine_extname        d2i_TS_TST_INFO_fp sunw_d2i_TS_TST_INFO_fp
1166 #pragma redefine_extname        d2i_USERNOTICE sunw_d2i_USERNOTICE
1167 #pragma redefine_extname        d2i_X509 sunw_d2i_X509
1168 #pragma redefine_extname        d2i_X509_ALGOR sunw_d2i_X509_ALGOR
1169 #pragma redefine_extname        d2i_X509_ALGORS sunw_d2i_X509_ALGORS
1170 #pragma redefine_extname        d2i_X509_ATTRIBUTE sunw_d2i_X509_ATTRIBUTE
1171 #pragma redefine_extname        d2i_X509_AUX sunw_d2i_X509_AUX
1172 #pragma redefine_extname        d2i_X509_bio sunw_d2i_X509_bio
1173 #pragma redefine_extname        d2i_X509_CERT_AUX sunw_d2i_X509_CERT_AUX
1174 #pragma redefine_extname        d2i_X509_CERT_PAIR sunw_d2i_X509_CERT_PAIR
1175 #pragma redefine_extname        d2i_X509_CINF sunw_d2i_X509_CINF
1176 #pragma redefine_extname        d2i_X509_CRL sunw_d2i_X509_CRL
1177 #pragma redefine_extname        d2i_X509_CRL_bio sunw_d2i_X509_CRL_bio
1178 #pragma redefine_extname        d2i_X509_CRL_fp sunw_d2i_X509_CRL_fp
1179 #pragma redefine_extname        d2i_X509_CRL_INFO sunw_d2i_X509_CRL_INFO
1180 #pragma redefine_extname        d2i_X509_EXTENSION sunw_d2i_X509_EXTENSION
1181 #pragma redefine_extname        d2i_X509_EXTENSIONS sunw_d2i_X509_EXTENSIONS
1182 #pragma redefine_extname        d2i_X509_fp sunw_d2i_X509_fp
1183 #pragma redefine_extname        d2i_X509_NAME sunw_d2i_X509_NAME
```

```
1184 #pragma redefine_extname        d2i_X509_NAME_ENTRY sunw_d2i_X509_NAME_ENTRY
1185 #pragma redefine_extname        d2i_X509_PKEY sunw_d2i_X509_PKEY
1186 #pragma redefine_extname        d2i_X509_PUBKEY sunw_d2i_X509_PUBKEY
1187 #pragma redefine_extname        d2i_X509_REQ sunw_d2i_X509_REQ
1188 #pragma redefine_extname        d2i_X509_REQ_bio sunw_d2i_X509_REQ_bio
1189 #pragma redefine_extname        d2i_X509_REQ_fp sunw_d2i_X509_REQ_fp
1190 #pragma redefine_extname        d2i_X509_REQ_INFO sunw_d2i_X509_REQ_INFO
1191 #pragma redefine_extname        d2i_X509_REVOKED sunw_d2i_X509_REVOKED
1192 #pragma redefine_extname        d2i_X509_SIG sunw_d2i_X509_SIG
1193 #pragma redefine_extname        d2i_X509_VAL sunw_d2i_X509_VAL
1194 #pragma redefine_extname        default_pctx sunw_default_pctx
1195 #pragma redefine_extname        DES_cbc_cksum sunw_DES_cbc_cksum
1196 #pragma redefine_extname        DES_cbc_encrypt sunw_DES_cbc_encrypt
1197 #pragma redefine_extname        DES_cfb_encrypt sunw_DES_cfb_encrypt
1198 #pragma redefine_extname        DES_cfb64_encrypt sunw_DES_cfb64_encrypt
1199 #pragma redefine_extname        DES_check_key_parity sunw_DES_check_key_parity
1200 #pragma redefine_extname        DES_crypt sunw_DES_crypt
1201 #pragma redefine_extname        DES_decrypt3 sunw_DES_decrypt3
1202 #pragma redefine_extname        DES_ecb_encrypt sunw_DES_ecb_encrypt
1203 #pragma redefine_extname        DES_ecb3_encrypt sunw_DES_ecb3_encrypt
1204 #pragma redefine_extname        DES_ede3_cbc_encrypt sunw_DES_ede3_cbc_encrypt
1205 #pragma redefine_extname        DES_ede3_cbcm_encrypt sunw_DES_ede3_cbcm_encrypt
1206 #pragma redefine_extname        DES_ede3_cfb_encrypt sunw_DES_ede3_cfb_encrypt
1207 #pragma redefine_extname        DES_ede3_cfb64_encrypt sunw_DES_ede3_cfb64_encry
1208 #pragma redefine_extname        DES_ede3_ofb64_encrypt sunw_DES_ede3_ofb64_encry
1209 #pragma redefine_extname        DES_enc_read sunw_DES_enc_read
1210 #pragma redefine_extname        DES_enc_write sunw_DES_enc_write
1211 #pragma redefine_extname        DES_encrypt1 sunw_DES_encrypt1
1212 #pragma redefine_extname        DES_encrypt2 sunw_DES_encrypt2
1213 #pragma redefine_extname        DES_encrypt3 sunw_DES_encrypt3
1214 #pragma redefine_extname        DES_fcrypt sunw_DES_fcrypt
1215 #pragma redefine_extname        DES_is_weak_key sunw_DES_is_weak_key
1216 #pragma redefine_extname        DES_key_sched sunw_DES_key_sched
1217 #pragma redefine_extname        DES_ncbc_encrypt sunw_DES_ncbc_encrypt
1218 #pragma redefine_extname        DES_ofb_encrypt sunw_DES_ofb_encrypt
1219 #pragma redefine_extname        DES_ofb64_encrypt sunw_DES_ofb64_encrypt
1220 #pragma redefine_extname        DES_options sunw_DES_options
1221 #pragma redefine_extname        DES_pcbc_encrypt sunw_DES_pcbc_encrypt
1222 #pragma redefine_extname        DES_quad_cksum sunw_DES_quad_cksum
1223 #pragma redefine_extname        DES_random_key sunw_DES_random_key
1224 #pragma redefine_extname        DES_read_2passwords sunw_DES_read_2passwords
1225 #pragma redefine_extname        DES_read_password sunw_DES_read_password
1226 #pragma redefine_extname        DES_set_key sunw_DES_set_key
1227 #pragma redefine_extname        DES_set_key_checked sunw_DES_set_key_checked
1228 #pragma redefine_extname        DES_set_key_unchecked sunw_DES_set_key_unchecked
1229 #pragma redefine_extname        DES_set_odd_parity sunw_DES_set_odd_parity
1230 #pragma redefine_extname        DES_SPtrans sunw_DES_SPtrans
1231 #pragma redefine_extname        DES_string_to_2keys sunw_DES_string_to_2keys
1232 #pragma redefine_extname        DES_string_to_key sunw_DES_string_to_key
1233 #pragma redefine_extname        DES_xcbc_encrypt sunw_DES_xcbc_encrypt
1234 #pragma redefine_extname        dh_asn1_meth sunw_dh_asn1_meth
1235 #pragma redefine_extname        DH_check sunw_DH_check
1236 #pragma redefine_extname        DH_check_pub_key sunw_DH_check_pub_key
1237 #pragma redefine_extname        DH_compute_key sunw_DH_compute_key
1238 #pragma redefine_extname        DH_free sunw_DH_free
1239 #pragma redefine_extname        DH_generate_key sunw_DH_generate_key
1240 #pragma redefine_extname        DH_generate_parameters sunw_DH_generate_paramete
1241 #pragma redefine_extname        DH_generate_parameters_ex sunw_DH_generate_param
1242 #pragma redefine_extname        DH_get_default_method sunw_DH_get_default_method
1243 #pragma redefine_extname        DH_get_ex_data sunw_DH_get_ex_data
1244 #pragma redefine_extname        DH_get_ex_new_index sunw_DH_get_ex_new_index
1245 #pragma redefine_extname        DH_new sunw_DH_new
1246 #pragma redefine_extname        DH_new_method sunw_DH_new_method
1247 #pragma redefine_extname        DH_OpenSSL sunw_DH_OpenSSL
1248 #pragma redefine_extname        dh_pkey_meth sunw_dh_pkey_meth
1249 #pragma redefine_extname        DH_set_default_method sunw_DH_set_default_method
```



```
1250 #pragma redefine_extname        DH_set_ex_data sunw_DH_set_ex_data
1251 #pragma redefine_extname        DH_set_method sunw_DH_set_method
1252 #pragma redefine_extname        DH_size sunw_DH_size
1253 #pragma redefine_extname        DH_up_ref sunw_DH_up_ref
1254 #pragma redefine_extname        DH_version sunw_DH_version
1255 #pragma redefine_extname        DHparams_dup sunw_DHparams_dup
1256 #pragma redefine_extname        DHparams_it sunw_DHparams_it
1257 #pragma redefine_extname        DHparams_print sunw_DHparams_print
1258 #pragma redefine_extname        DHparams_print_fp sunw_DHparams_print_fp
1259 #pragma redefine_extname        DIRECTORYSTRING_free sunw_DIRECTORYSTRING_free
1260 #pragma redefine_extname        DIRECTORYSTRING_it sunw_DIRECTORYSTRING_it
1261 #pragma redefine_extname        DIRECTORYSTRING_new sunw_DIRECTORYSTRING_new
1262 #pragma redefine_extname        DISPLAYTEXT_free sunw_DISPLAYTEXT_free
1263 #pragma redefine_extname        DISPLAYTEXT_it sunw_DISPLAYTEXT_it
1264 #pragma redefine_extname        DISPLAYTEXT_new sunw_DISPLAYTEXT_new
1265 #pragma redefine_extname        DIST_POINT_free sunw_DIST_POINT_free
1266 #pragma redefine_extname        DIST_POINT_it sunw_DIST_POINT_it
1267 #pragma redefine_extname        DIST_POINT_NAME_free sunw_DIST_POINT_NAME_free
1268 #pragma redefine_extname        DIST_POINT_NAME_it sunw_DIST_POINT_NAME_it
1269 #pragma redefine_extname        DIST_POINT_NAME_new sunw_DIST_POINT_NAME_new
1270 #pragma redefine_extname        DIST_POINT_new sunw_DIST_POINT_new
1271 #pragma redefine_extname        DIST_POINT_set_dpname sunw_DIST_POINT_set_dpname
1272 #pragma redefine_extname        dsa_asn1_meths sunw_dsa_asn1_meths
1273 #pragma redefine_extname        dsa_builtin_paramgen sunw_dsa_builtin_paramgen
1274 #pragma redefine_extname        DSA_do_sign sunw_DSA_do_sign
1275 #pragma redefine_extname        DSA_do_verify sunw_DSA_do_verify
1276 #pragma redefine_extname        DSA_dup_DH sunw_DSA_dup_DH
1277 #pragma redefine_extname        DSA_free sunw_DSA_free
1278 #pragma redefine_extname        DSA_generate_key sunw_DSA_generate_key
1279 #pragma redefine_extname        DSA_generate_parameters sunw_DSA_generate_parame
1280 #pragma redefine_extname        DSA_generate_parameters_ex sunw_DSA_generate_par
1281 #pragma redefine_extname        DSA_get_default_method sunw_DSA_get_default_meth
1282 #pragma redefine_extname        DSA_get_ex_data sunw_DSA_get_ex_data
1283 #pragma redefine_extname        DSA_get_ex_new_index sunw_DSA_get_ex_new_index
1284 #pragma redefine_extname        DSA_new sunw_DSA_new
1285 #pragma redefine_extname        DSA_new_method sunw_DSA_new_method
1286 #pragma redefine_extname        DSA_OpenSSL sunw_DSA_OpenSSL
1287 #pragma redefine_extname        dsa_pkey_meth sunw_dsa_pkey_meth
1288 #pragma redefine_extname        DSA_print sunw_DSA_print
1289 #pragma redefine_extname        DSA_print_fp sunw_DSA_print_fp
1290 #pragma redefine_extname        dsa_pub_internal_it sunw_dsa_pub_internal_it
1291 #pragma redefine_extname        DSA_set_default_method sunw_DSA_set_default_meth
1292 #pragma redefine_extname        DSA_set_ex_data sunw_DSA_set_ex_data
1293 #pragma redefine_extname        DSA_set_method sunw_DSA_set_method
1294 #pragma redefine_extname        DSA_SIG_free sunw_DSA_SIG_free
1295 #pragma redefine_extname        DSA_SIG_it sunw_DSA_SIG_it
1296 #pragma redefine_extname        DSA_SIG_new sunw_DSA_SIG_new
1297 #pragma redefine_extname        DSA_sign sunw_DSA_sign
1298 #pragma redefine_extname        DSA_sign_setup sunw_DSA_sign_setup
1299 #pragma redefine_extname        DSA_size sunw_DSA_size
1300 #pragma redefine_extname        DSA_up_ref sunw_DSA_up_ref
1301 #pragma redefine_extname        DSA_verify sunw_DSA_verify
1302 #pragma redefine_extname        DSA_version sunw_DSA_version
1303 #pragma redefine_extname        DSAparams_dup sunw_DSAparams_dup
1304 #pragma redefine_extname        DSAparams_it sunw_DSAparams_it
1305 #pragma redefine_extname        DSAparams_print sunw_DSAparams_print
1306 #pragma redefine_extname        DSAparams_print_fp sunw_DSAparams_print_fp
1307 #pragma redefine_extname        DSAPrivateKey_it sunw_DSAPrivateKey_it
1308 #pragma redefine_extname        DSAPublicKey_it sunw_DSAPublicKey_it
1309 #pragma redefine_extname        DSO_bind_func sunw_DSO_bind_func
1310 #pragma redefine_extname        DSO_bind_var sunw_DSO_bind_var
1311 #pragma redefine_extname        DSO_convert_filename sunw_DSO_convert_filename
1312 #pragma redefine_extname        DSO_ctrl sunw_DSO_ctrl
1313 #pragma redefine_extname        DSO_flags sunw_DSO_flags
1314 #pragma redefine_extname        DSO_free sunw_DSO_free
1315 #pragma redefine_extname        DSO_get_default_method sunw_DSO_get_default_meth
```

```
1316 #pragma redefine_extname         DSO_get_filename sunw_DSO_get_filename
1317 #pragma redefine_extname         DSO_get_loaded_filename sunw_DSO_get_loaded_file
1318 #pragma redefine_extname         DSO_get_method sunw_DSO_get_method
1319 #pragma redefine_extname         DSO_global_lookup sunw_DSO_global_lookup
1320 #pragma redefine_extname         DSO_load sunw_DSO_load
1321 #pragma redefine_extname         DSO_merge sunw_DSO_merge
1322 #pragma redefine_extname         DSO_METHOD_beos sunw_DSO_METHOD_beos
1323 #pragma redefine_extname         DSO_METHOD_dl sunw_DSO_METHOD_dl
1324 #pragma redefine_extname         DSO_METHOD_dlfcn sunw_DSO_METHOD_dlfcn
1325 #pragma redefine_extname         DSO_METHOD_null sunw_DSO_METHOD_null
1326 #pragma redefine_extname         DSO_METHOD_openssl sunw_DSO_METHOD_openssl
1327 #pragma redefine_extname         DSO_METHOD_vms sunw_DSO_METHOD_vms
1328 #pragma redefine_extname         DSO_METHOD_win32 sunw_DSO_METHOD_win32
1329 #pragma redefine_extname         DSO_new sunw_DSO_new
1330 #pragma redefine_extname         DSO_new_method sunw_DSO_new_method
1331 #pragma redefine_extname         DSO_pathbyaddr sunw_DSO_pathbyaddr
1332 #pragma redefine_extname         DSO_set_default_method sunw_DSO_set_default_meth
1333 #pragma redefine_extname         DSO_set_filename sunw_DSO_set_filename
1334 #pragma redefine_extname         DSO_set_method sunw_DSO_set_method
1335 #pragma redefine_extname         DSO_set_name_converter sunw_DSO_set_name_convert
1336 #pragma redefine_extname         DSO_up_ref sunw_DSO_up_ref
1337 #pragma redefine_extname         EDIPARTYNAME_free sunw_EDIPARTYNAME_free
1338 #pragma redefine_extname         EDIPARTYNAME_it sunw_EDIPARTYNAME_it
1339 #pragma redefine_extname         EDIPARTYNAME_new sunw_EDIPARTYNAME_new
1340 #pragma redefine_extname         ENGINE_add sunw_ENGINE_add
1341 #pragma redefine_extname         ENGINE_add_conf_module sunw_ENGINE_add_conf_modu
1342 #pragma redefine_extname         ENGINE_by_id sunw_ENGINE_by_id
1343 #pragma redefine_extname         ENGINE_cleanup sunw_ENGINE_cleanup
1344 #pragma redefine_extname         engine_cleanup_add_first sunw_engine_cleanup_add
1345 #pragma redefine_extname         engine_cleanup_add_last sunw_engine_cleanup_add_
1346 #pragma redefine_extname         ENGINE_cmd_is_executable sunw_ENGINE_cmd_is_exec
1347 #pragma redefine_extname         ENGINE_ctrl sunw_ENGINE_ctrl
1348 #pragma redefine_extname         ENGINE_ctrl_cmd sunw_ENGINE_ctrl_cmd
1349 #pragma redefine_extname         ENGINE_ctrl_cmd_string sunw_ENGINE_ctrl_cmd_stri
1350 #pragma redefine_extname         ENGINE_finish sunw_ENGINE_finish
1351 #pragma redefine_extname         ENGINE_free sunw_ENGINE_free
1352 #pragma redefine_extname         engine_free_util sunw_engine_free_util
1353 #pragma redefine_extname         ENGINE_get_cipher sunw_ENGINE_get_cipher
1354 #pragma redefine_extname         ENGINE_get_cipher_engine sunw_ENGINE_get_cipher_
1355 #pragma redefine_extname         ENGINE_get_ciphers sunw_ENGINE_get_ciphers
1356 #pragma redefine_extname         ENGINE_get_cmd_defns sunw_ENGINE_get_cmd_defns
1357 #pragma redefine_extname         ENGINE_get_ctrl_function sunw_ENGINE_get_ctrl_fu
1358 #pragma redefine_extname         ENGINE_get_default_DH sunw_ENGINE_get_default_DH
1359 #pragma redefine_extname         ENGINE_get_default_DSA sunw_ENGINE_get_default_D
1360 #pragma redefine_extname         ENGINE_get_default_ECDH sunw_ENGINE_get_default_
1361 #pragma redefine_extname         ENGINE_get_default_ECDSA sunw_ENGINE_get_default
1362 #pragma redefine_extname         ENGINE_get_default_RAND sunw_ENGINE_get_default_
1363 #pragma redefine_extname         ENGINE_get_default_RSA sunw_ENGINE_get_default_R
1364 #pragma redefine_extname         ENGINE_get_destroy_function sunw_ENGINE_get_dest
1365 #pragma redefine_extname         ENGINE_get_DH sunw_ENGINE_get_DH
1366 #pragma redefine_extname         ENGINE_get_digest sunw_ENGINE_get_digest
1367 #pragma redefine_extname         ENGINE_get_digest_engine sunw_ENGINE_get_digest_
1368 #pragma redefine_extname         ENGINE_get_digests sunw_ENGINE_get_digests
1369 #pragma redefine_extname         ENGINE_get_DSA sunw_ENGINE_get_DSA
1370 #pragma redefine_extname         ENGINE_get_ECDH sunw_ENGINE_get_ECDH
1371 #pragma redefine_extname         ENGINE_get_ECDSA sunw_ENGINE_get_ECDSA
1372 #pragma redefine_extname         ENGINE_get_ex_data sunw_ENGINE_get_ex_data
1373 #pragma redefine_extname         ENGINE_get_ex_new_index sunw_ENGINE_get_ex_new_i
1374 #pragma redefine_extname         ENGINE_get_finish_function sunw_ENGINE_get_finis
1375 #pragma redefine_extname         ENGINE_get_first sunw_ENGINE_get_first
1376 #pragma redefine_extname         ENGINE_get_flags sunw_ENGINE_get_flags
1377 #pragma redefine_extname         ENGINE_get_id sunw_ENGINE_get_id
1378 #pragma redefine_extname         ENGINE_get_init_function sunw_ENGINE_get_init_fu
1379 #pragma redefine_extname         ENGINE_get_last sunw_ENGINE_get_last
1380 #pragma redefine_extname         ENGINE_get_load_privkey_function sunw_ENGINE_get
1381 #pragma redefine_extname         ENGINE_get_load_pubkey_function sunw_ENGINE_get_
```



```
1382 #pragma redefine_extname         ENGINE_get_name sunw_ENGINE_get_name
1383 #pragma redefine_extname         ENGINE_get_next sunw_ENGINE_get_next
1384 #pragma redefine_extname         ENGINE_get_pkey_asn1_meth sunw_ENGINE_get_pkey_a
1385 #pragma redefine_extname         ENGINE_get_pkey_asn1_meth_engine sunw_ENGINE_get
1386 #pragma redefine_extname         ENGINE_get_pkey_asn1_meth_str sunw_ENGINE_get_pk
1387 #pragma redefine_extname         ENGINE_get_pkey_asn1_meths sunw_ENGINE_get_pkey_
1388 #pragma redefine_extname         ENGINE_get_pkey_meth sunw_ENGINE_get_pkey_meth
1389 #pragma redefine_extname         ENGINE_get_pkey_meth_engine sunw_ENGINE_get_pkey
1390 #pragma redefine_extname         ENGINE_get_pkey_meths sunw_ENGINE_get_pkey_meths
1391 #pragma redefine_extname         ENGINE_get_prev sunw_ENGINE_get_prev
1392 #pragma redefine_extname         ENGINE_get_RAND sunw_ENGINE_get_RAND
1393 #pragma redefine_extname         ENGINE_get_RSA sunw_ENGINE_get_RSA
1394 #pragma redefine_extname         ENGINE_get_ssl_client_cert_function sunw_ENGINE_
1395 #pragma redefine_extname         ENGINE_get_static_state sunw_ENGINE_get_static_s
1396 #pragma redefine_extname         ENGINE_get_STORE sunw_ENGINE_get_STORE
1397 #pragma redefine_extname         ENGINE_get_table_flags sunw_ENGINE_get_table_fla
1398 #pragma redefine_extname         ENGINE_init sunw_ENGINE_init
1399 #pragma redefine_extname         ENGINE_load_builtin_engines sunw_ENGINE_load_bui
1400 #pragma redefine_extname         ENGINE_load_cryptodev sunw_ENGINE_load_cryptodev
1401 #pragma redefine_extname         ENGINE_load_dynamic sunw_ENGINE_load_dynamic
1402 #pragma redefine_extname         ENGINE_load_openssl sunw_ENGINE_load_openssl
1403 #pragma redefine_extname         ENGINE_load_pk11 sunw_ENGINE_load_pk11
1404 #pragma redefine_extname         ENGINE_load_private_key sunw_ENGINE_load_private
1405 #pragma redefine_extname         ENGINE_load_public_key sunw_ENGINE_load_public_k
1406 #pragma redefine_extname         ENGINE_load_rdrand sunw_ENGINE_load_rdrand
1407 #pragma redefine_extname         ENGINE_load_rsax sunw_ENGINE_load_rsax
1408 #pragma redefine_extname         ENGINE_load_ssl_client_cert sunw_ENGINE_load_ssl
1409 #pragma redefine_extname         ENGINE_new sunw_ENGINE_new
1410 #pragma redefine_extname         ENGINE_pkey_asn1_find_str sunw_ENGINE_pkey_asn1_
1411 #pragma redefine_extname         engine_pkey_asn1_meths_free sunw_engine_pkey_asn
1412 #pragma redefine_extname         engine_pkey_meths_free sunw_engine_pkey_meths_fr
1413 #pragma redefine_extname         ENGINE_register_all_ciphers sunw_ENGINE_register
1414 #pragma redefine_extname         ENGINE_register_all_complete sunw_ENGINE_registe
1415 #pragma redefine_extname         ENGINE_register_all_DH sunw_ENGINE_register_all_
1416 #pragma redefine_extname         ENGINE_register_all_digests sunw_ENGINE_register
1417 #pragma redefine_extname         ENGINE_register_all_DSA sunw_ENGINE_register_all
1418 #pragma redefine_extname         ENGINE_register_all_ECDH sunw_ENGINE_register_al
1419 #pragma redefine_extname         ENGINE_register_all_ECDSA sunw_ENGINE_register_a
1420 #pragma redefine_extname         ENGINE_register_all_pkey_asn1_meths sunw_ENGINE_
1421 #pragma redefine_extname         ENGINE_register_all_pkey_meths sunw_ENGINE_regis
1422 #pragma redefine_extname         ENGINE_register_all_RAND sunw_ENGINE_register_al
1423 #pragma redefine_extname         ENGINE_register_all_RSA sunw_ENGINE_register_all
1424 #pragma redefine_extname         ENGINE_register_all_STORE sunw_ENGINE_register_a
1425 #pragma redefine_extname         ENGINE_register_ciphers sunw_ENGINE_register_cip
1426 #pragma redefine_extname         ENGINE_register_complete sunw_ENGINE_register_co
1427 #pragma redefine_extname         ENGINE_register_DH sunw_ENGINE_register_DH
1428 #pragma redefine_extname         ENGINE_register_digests sunw_ENGINE_register_dig
1429 #pragma redefine_extname         ENGINE_register_DSA sunw_ENGINE_register_DSA
1430 #pragma redefine_extname         ENGINE_register_ECDH sunw_ENGINE_register_ECDH
1431 #pragma redefine_extname         ENGINE_register_ECDSA sunw_ENGINE_register_ECDSA
1432 #pragma redefine_extname         ENGINE_register_pkey_asn1_meths sunw_ENGINE_regi
1433 #pragma redefine_extname         ENGINE_register_pkey_meths sunw_ENGINE_register_
1434 #pragma redefine_extname         ENGINE_register_RAND sunw_ENGINE_register_RAND
1435 #pragma redefine_extname         ENGINE_register_RSA sunw_ENGINE_register_RSA
1436 #pragma redefine_extname         ENGINE_register_STORE sunw_ENGINE_register_STORE
1437 #pragma redefine_extname         ENGINE_remove sunw_ENGINE_remove
1438 #pragma redefine_extname         engine_set_all_null sunw_engine_set_all_null
1439 #pragma redefine_extname         ENGINE_set_ciphers sunw_ENGINE_set_ciphers
1440 #pragma redefine_extname         ENGINE_set_cmd_defns sunw_ENGINE_set_cmd_defns
1441 #pragma redefine_extname         ENGINE_set_ctrl_function sunw_ENGINE_set_ctrl_fu
1442 #pragma redefine_extname         ENGINE_set_default sunw_ENGINE_set_default
1443 #pragma redefine_extname         ENGINE_set_default_ciphers sunw_ENGINE_set_defau
1444 #pragma redefine_extname         ENGINE_set_default_DH sunw_ENGINE_set_default_DH
1445 #pragma redefine_extname         ENGINE_set_default_digests sunw_ENGINE_set_defau
1446 #pragma redefine_extname         ENGINE_set_default_DSA sunw_ENGINE_set_default_D
1447 #pragma redefine_extname         ENGINE_set_default_ECDH sunw_ENGINE_set_default_
```

```
#pragma redefine_extname        ENGINE_set_default_ECDSA sunw_ENGINE_set_default
#pragma redefine_extname        ENGINE_set_default_pkey_asn1_meths sunw_ENGINE_s
#pragma redefine_extname        ENGINE_set_default_pkey_meths sunw_ENGINE_set_de
#pragma redefine_extname        ENGINE_set_default_RAND sunw_ENGINE_set_default_
#pragma redefine_extname        ENGINE_set_default_RSA sunw_ENGINE_set_default_R
#pragma redefine_extname        ENGINE_set_default_string sunw_ENGINE_set_defaul
#pragma redefine_extname        ENGINE_set_destroy_function sunw_ENGINE_set_dest
#pragma redefine_extname        ENGINE_set_DH sunw_ENGINE_set_DH
#pragma redefine_extname        ENGINE_set_digests sunw_ENGINE_set_digests
#pragma redefine_extname        ENGINE_set_DSA sunw_ENGINE_set_DSA
#pragma redefine_extname        ENGINE_set_ECDH sunw_ENGINE_set_ECDH
#pragma redefine_extname        ENGINE_set_ECDSA sunw_ENGINE_set_ECDSA
#pragma redefine_extname        ENGINE_set_ex_data sunw_ENGINE_set_ex_data
#pragma redefine_extname        ENGINE_set_finish_function sunw_ENGINE_set_finis
#pragma redefine_extname        ENGINE_set_flags sunw_ENGINE_set_flags
#pragma redefine_extname        ENGINE_set_id sunw_ENGINE_set_id
#pragma redefine_extname        ENGINE_set_init_function sunw_ENGINE_set_init_fu
#pragma redefine_extname        ENGINE_set_load_privkey_function sunw_ENGINE_set
#pragma redefine_extname        ENGINE_set_load_pubkey_function sunw_ENGINE_set_
#pragma redefine_extname        ENGINE_set_load_ssl_client_cert_function sunw_EN
#pragma redefine_extname        ENGINE_set_name sunw_ENGINE_set_name
#pragma redefine_extname        ENGINE_set_pkey_asn1_meths sunw_ENGINE_set_pkey_
#pragma redefine_extname        ENGINE_set_pkey_meths sunw_ENGINE_set_pkey_meths
#pragma redefine_extname        ENGINE_set_RAND sunw_ENGINE_set_RAND
#pragma redefine_extname        ENGINE_set_RSA sunw_ENGINE_set_RSA
#pragma redefine_extname        ENGINE_set_STORE sunw_ENGINE_set_STORE
#pragma redefine_extname        ENGINE_set_table_flags sunw_ENGINE_set_table_fla
#pragma redefine_extname        engine_table_cleanup sunw_engine_table_cleanup
#pragma redefine_extname        engine_table_doall sunw_engine_table_doall
#pragma redefine_extname        engine_table_register sunw_engine_table_register
#pragma redefine_extname        engine_table_select sunw_engine_table_select
#pragma redefine_extname        engine_table_unregister sunw_engine_table_unregi
#pragma redefine_extname        engine_unlocked_finish sunw_engine_unlocked_fini
#pragma redefine_extname        engine_unlocked_init sunw_engine_unlocked_init
#pragma redefine_extname        ENGINE_unregister_ciphers sunw_ENGINE_unregister
#pragma redefine_extname        ENGINE_unregister_DH sunw_ENGINE_unregister_DH
#pragma redefine_extname        ENGINE_unregister_digests sunw_ENGINE_unregister
#pragma redefine_extname        ENGINE_unregister_DSA sunw_ENGINE_unregister_DSA
#pragma redefine_extname        ENGINE_unregister_ECDH sunw_ENGINE_unregister_EC
#pragma redefine_extname        ENGINE_unregister_ECDSA sunw_ENGINE_unregister_E
#pragma redefine_extname        ENGINE_unregister_pkey_asn1_meths sunw_ENGINE_un
#pragma redefine_extname        ENGINE_unregister_pkey_meths sunw_ENGINE_unregis
#pragma redefine_extname        ENGINE_unregister_RAND sunw_ENGINE_unregister_RA
#pragma redefine_extname        ENGINE_unregister_RSA sunw_ENGINE_unregister_RSA
#pragma redefine_extname        ENGINE_unregister_STORE sunw_ENGINE_unregister_S
#pragma redefine_extname        ENGINE_up_ref sunw_ENGINE_up_ref
#pragma redefine_extname        ERR_add_error_data sunw_ERR_add_error_data
#pragma redefine_extname        ERR_add_error_vdata sunw_ERR_add_error_vdata
#pragma redefine_extname        ERR_clear_error sunw_ERR_clear_error
#pragma redefine_extname        ERR_error_string sunw_ERR_error_string
#pragma redefine_extname        ERR_error_string_n sunw_ERR_error_string_n
#pragma redefine_extname        ERR_free_strings sunw_ERR_free_strings
#pragma redefine_extname        ERR_func_error_string sunw_ERR_func_error_string
#pragma redefine_extname        ERR_get_err_state_table sunw_ERR_get_err_state_t
#pragma redefine_extname        ERR_get_error sunw_ERR_get_error
#pragma redefine_extname        ERR_get_error_line sunw_ERR_get_error_line
#pragma redefine_extname        ERR_get_error_line_data sunw_ERR_get_error_line_
#pragma redefine_extname        ERR_get_implementation sunw_ERR_get_implementati
#pragma redefine_extname        ERR_get_next_error_library sunw_ERR_get_next_err
#pragma redefine_extname        ERR_get_state sunw_ERR_get_state
#pragma redefine_extname        ERR_get_string_table sunw_ERR_get_string_table
#pragma redefine_extname        ERR_lib_error_string sunw_ERR_lib_error_string
#pragma redefine_extname        ERR_load_ASN1_strings sunw_ERR_load_ASN1_strings
#pragma redefine_extname        ERR_load_BIO_strings sunw_ERR_load_BIO_strings
#pragma redefine_extname        ERR_load_BN_strings sunw_ERR_load_BN_strings
#pragma redefine_extname        ERR_load_BUF_strings sunw_ERR_load_BUF_strings
```



```
#pragma redefine_extname        ERR_load_CMS_strings sunw_ERR_load_CMS_strings
#pragma redefine_extname        ERR_load_COMP_strings sunw_ERR_load_COMP_strings
#pragma redefine_extname        ERR_load_CONF_strings sunw_ERR_load_CONF_strings
#pragma redefine_extname        ERR_load_crypto_strings sunw_ERR_load_crypto_str
#pragma redefine_extname        ERR_load_CRYPTO_strings sunw_ERR_load_CRYPTO_str
#pragma redefine_extname        ERR_load_DH_strings sunw_ERR_load_DH_strings
#pragma redefine_extname        ERR_load_DSA_strings sunw_ERR_load_DSA_strings
#pragma redefine_extname        ERR_load_DSO_strings sunw_ERR_load_DSO_strings
#pragma redefine_extname        ERR_load_ENGINE_strings sunw_ERR_load_ENGINE_str
#pragma redefine_extname        ERR_load_ERR_strings sunw_ERR_load_ERR_strings
#pragma redefine_extname        ERR_load_EVP_strings sunw_ERR_load_EVP_strings
#pragma redefine_extname        ERR_load_OBJ_strings sunw_ERR_load_OBJ_strings
#pragma redefine_extname        ERR_load_OCSP_strings sunw_ERR_load_OCSP_strings
#pragma redefine_extname        ERR_load_PEM_strings sunw_ERR_load_PEM_strings
#pragma redefine_extname        ERR_load_PKCS12_strings sunw_ERR_load_PKCS12_str
#pragma redefine_extname        ERR_load_PKCS7_strings sunw_ERR_load_PKCS7_strin
#pragma redefine_extname        ERR_load_RAND_strings sunw_ERR_load_RAND_strings
#pragma redefine_extname        ERR_load_RSA_strings sunw_ERR_load_RSA_strings
#pragma redefine_extname        ERR_load_strings sunw_ERR_load_strings
#pragma redefine_extname        ERR_load_TS_strings sunw_ERR_load_TS_strings
#pragma redefine_extname        ERR_load_UI_strings sunw_ERR_load_UI_strings
#pragma redefine_extname        ERR_load_X509_strings sunw_ERR_load_X509_strings
#pragma redefine_extname        ERR_load_X509V3_strings sunw_ERR_load_X509V3_str
#pragma redefine_extname        ERR_peek_error sunw_ERR_peek_error
#pragma redefine_extname        ERR_peek_error_line sunw_ERR_peek_error_line
#pragma redefine_extname        ERR_peek_error_line_data sunw_ERR_peek_error_lin
#pragma redefine_extname        ERR_peek_last_error sunw_ERR_peek_last_error
#pragma redefine_extname        ERR_peek_last_error_line sunw_ERR_peek_last_erro
#pragma redefine_extname        ERR_peek_last_error_line_data sunw_ERR_peek_last
#pragma redefine_extname        ERR_pk11_error sunw_ERR_pk11_error
#pragma redefine_extname        ERR_pop_to_mark sunw_ERR_pop_to_mark
#pragma redefine_extname        ERR_print_errors sunw_ERR_print_errors
#pragma redefine_extname        ERR_print_errors_cb sunw_ERR_print_errors_cb
#pragma redefine_extname        ERR_print_errors_fp sunw_ERR_print_errors_fp
#pragma redefine_extname        ERR_put_error sunw_ERR_put_error
#pragma redefine_extname        ERR_reason_error_string sunw_ERR_reason_error_st
#pragma redefine_extname        ERR_release_err_state_table sunw_ERR_release_err
#pragma redefine_extname        ERR_remove_state sunw_ERR_remove_state
#pragma redefine_extname        ERR_remove_thread_state sunw_ERR_remove_thread_s
#pragma redefine_extname        ERR_set_error_data sunw_ERR_set_error_data
#pragma redefine_extname        ERR_set_implementation sunw_ERR_set_implementati
#pragma redefine_extname        ERR_set_mark sunw_ERR_set_mark
#pragma redefine_extname        ERR_unload_strings sunw_ERR_unload_strings
#pragma redefine_extname        ESS_CERT_ID_dup sunw_ESS_CERT_ID_dup
#pragma redefine_extname        ESS_CERT_ID_free sunw_ESS_CERT_ID_free
#pragma redefine_extname        ESS_CERT_ID_it sunw_ESS_CERT_ID_it
#pragma redefine_extname        ESS_CERT_ID_new sunw_ESS_CERT_ID_new
#pragma redefine_extname        ESS_ISSUER_SERIAL_dup sunw_ESS_ISSUER_SERIAL_dup
#pragma redefine_extname        ESS_ISSUER_SERIAL_free sunw_ESS_ISSUER_SERIAL_fr
#pragma redefine_extname        ESS_ISSUER_SERIAL_it sunw_ESS_ISSUER_SERIAL_it
#pragma redefine_extname        ESS_ISSUER_SERIAL_new sunw_ESS_ISSUER_SERIAL_new
#pragma redefine_extname        ESS_SIGNING_CERT_dup sunw_ESS_SIGNING_CERT_dup
#pragma redefine_extname        ESS_SIGNING_CERT_free sunw_ESS_SIGNING_CERT_free
#pragma redefine_extname        ESS_SIGNING_CERT_it sunw_ESS_SIGNING_CERT_it
#pragma redefine_extname        ESS_SIGNING_CERT_new sunw_ESS_SIGNING_CERT_new
#pragma redefine_extname        EVP_add_alg_module sunw_EVP_add_alg_module
#pragma redefine_extname        EVP_add_cipher sunw_EVP_add_cipher
#pragma redefine_extname        EVP_add_digest sunw_EVP_add_digest
#pragma redefine_extname        EVP_aes_128_cbc sunw_EVP_aes_128_cbc
#pragma redefine_extname        EVP_aes_128_cbc_hmac_sha1 sunw_EVP_aes_128_cbc_h
#pragma redefine_extname        EVP_aes_128_ccm sunw_EVP_aes_128_ccm
#pragma redefine_extname        EVP_aes_128_cfb sunw_EVP_aes_128_cfb
#pragma redefine_extname        EVP_aes_128_cfb1 sunw_EVP_aes_128_cfb1
#pragma redefine_extname        EVP_aes_128_cfb128 sunw_EVP_aes_128_cfb128
#pragma redefine_extname        EVP_aes_128_cfb8 sunw_EVP_aes_128_cfb8
#pragma redefine_extname        EVP_aes_128_ctr sunw_EVP_aes_128_ctr
```

```
#pragma redefine_extname        EVP_aes_128_ecb sunw_EVP_aes_128_ecb
#pragma redefine_extname        EVP_aes_128_gcm sunw_EVP_aes_128_gcm
#pragma redefine_extname        EVP_aes_128_ofb sunw_EVP_aes_128_ofb
#pragma redefine_extname        EVP_aes_128_xts sunw_EVP_aes_128_xts
#pragma redefine_extname        EVP_aes_192_cbc sunw_EVP_aes_192_cbc
#pragma redefine_extname        EVP_aes_192_ccm sunw_EVP_aes_192_ccm
#pragma redefine_extname        EVP_aes_192_cfb sunw_EVP_aes_192_cfb
#pragma redefine_extname        EVP_aes_192_cfb1 sunw_EVP_aes_192_cfb1
#pragma redefine_extname        EVP_aes_192_cfb128 sunw_EVP_aes_192_cfb128
#pragma redefine_extname        EVP_aes_192_cfb8 sunw_EVP_aes_192_cfb8
#pragma redefine_extname        EVP_aes_192_ctr sunw_EVP_aes_192_ctr
#pragma redefine_extname        EVP_aes_192_ecb sunw_EVP_aes_192_ecb
#pragma redefine_extname        EVP_aes_192_gcm sunw_EVP_aes_192_gcm
#pragma redefine_extname        EVP_aes_192_ofb sunw_EVP_aes_192_ofb
#pragma redefine_extname        EVP_aes_256_cbc sunw_EVP_aes_256_cbc
#pragma redefine_extname        EVP_aes_256_cbc_hmac_sha1 sunw_EVP_aes_256_cbc_h
#pragma redefine_extname        EVP_aes_256_ccm sunw_EVP_aes_256_ccm
#pragma redefine_extname        EVP_aes_256_cfb sunw_EVP_aes_256_cfb
#pragma redefine_extname        EVP_aes_256_cfb1 sunw_EVP_aes_256_cfb1
#pragma redefine_extname        EVP_aes_256_cfb128 sunw_EVP_aes_256_cfb128
#pragma redefine_extname        EVP_aes_256_cfb8 sunw_EVP_aes_256_cfb8
#pragma redefine_extname        EVP_aes_256_ctr sunw_EVP_aes_256_ctr
#pragma redefine_extname        EVP_aes_256_ecb sunw_EVP_aes_256_ecb
#pragma redefine_extname        EVP_aes_256_gcm sunw_EVP_aes_256_gcm
#pragma redefine_extname        EVP_aes_256_ofb sunw_EVP_aes_256_ofb
#pragma redefine_extname        EVP_aes_256_xts sunw_EVP_aes_256_xts
#pragma redefine_extname        EVP_bf_cbc sunw_EVP_bf_cbc
#pragma redefine_extname        EVP_bf_cfb sunw_EVP_bf_cfb
#pragma redefine_extname        EVP_bf_cfb64 sunw_EVP_bf_cfb64
#pragma redefine_extname        EVP_bf_ecb sunw_EVP_bf_ecb
#pragma redefine_extname        EVP_bf_ofb sunw_EVP_bf_ofb
#pragma redefine_extname        EVP_BytesToKey sunw_EVP_BytesToKey
#pragma redefine_extname        EVP_camellia_128_cbc sunw_EVP_camellia_128_cbc
#pragma redefine_extname        EVP_camellia_128_cfb1 sunw_EVP_camellia_128_cfb1
#pragma redefine_extname        EVP_camellia_128_cfb128 sunw_EVP_camellia_128_cf
#pragma redefine_extname        EVP_camellia_128_cfb8 sunw_EVP_camellia_128_cfb8
#pragma redefine_extname        EVP_camellia_128_ecb sunw_EVP_camellia_128_ecb
#pragma redefine_extname        EVP_camellia_128_ofb sunw_EVP_camellia_128_ofb
#pragma redefine_extname        EVP_camellia_192_cbc sunw_EVP_camellia_192_cbc
#pragma redefine_extname        EVP_camellia_192_cfb1 sunw_EVP_camellia_192_cfb1
#pragma redefine_extname        EVP_camellia_192_cfb128 sunw_EVP_camellia_192_cf
#pragma redefine_extname        EVP_camellia_192_cfb8 sunw_EVP_camellia_192_cfb8
#pragma redefine_extname        EVP_camellia_192_ecb sunw_EVP_camellia_192_ecb
#pragma redefine_extname        EVP_camellia_192_ofb sunw_EVP_camellia_192_ofb
#pragma redefine_extname        EVP_camellia_256_cbc sunw_EVP_camellia_256_cbc
#pragma redefine_extname        EVP_camellia_256_cfb1 sunw_EVP_camellia_256_cfb1
#pragma redefine_extname        EVP_camellia_256_cfb128 sunw_EVP_camellia_256_cf
#pragma redefine_extname        EVP_camellia_256_cfb8 sunw_EVP_camellia_256_cfb8
#pragma redefine_extname        EVP_camellia_256_ecb sunw_EVP_camellia_256_ecb
#pragma redefine_extname        EVP_camellia_256_ofb sunw_EVP_camellia_256_ofb
#pragma redefine_extname        EVP_cast5_cbc sunw_EVP_cast5_cbc
#pragma redefine_extname        EVP_cast5_cfb sunw_EVP_cast5_cfb
#pragma redefine_extname        EVP_cast5_cfb64 sunw_EVP_cast5_cfb64
#pragma redefine_extname        EVP_cast5_ecb sunw_EVP_cast5_ecb
#pragma redefine_extname        EVP_cast5_ofb sunw_EVP_cast5_ofb
#pragma redefine_extname        EVP_Cipher sunw_EVP_Cipher
#pragma redefine_extname        EVP_CIPHER_asn1_to_param sunw_EVP_CIPHER_asn1_to
#pragma redefine_extname        EVP_CIPHER_block_size sunw_EVP_CIPHER_block_size
#pragma redefine_extname        EVP_CIPHER_CTX_block_size sunw_EVP_CIPHER_CTX_bl
#pragma redefine_extname        EVP_CIPHER_CTX_cipher sunw_EVP_CIPHER_CTX_cipher
#pragma redefine_extname        EVP_CIPHER_CTX_cleanup sunw_EVP_CIPHER_CTX_clean
#pragma redefine_extname        EVP_CIPHER_CTX_clear_flags sunw_EVP_CIPHER_CTX_c
#pragma redefine_extname        EVP_CIPHER_CTX_copy sunw_EVP_CIPHER_CTX_copy
#pragma redefine_extname        EVP_CIPHER_CTX_ctrl sunw_EVP_CIPHER_CTX_ctrl
#pragma redefine_extname        EVP_CIPHER_CTX_flags sunw_EVP_CIPHER_CTX_flags
#pragma redefine_extname        EVP_CIPHER_CTX_free sunw_EVP_CIPHER_CTX_free
```



```
#pragma redefine_extname        EVP_CIPHER_CTX_get_app_data sunw_EVP_CIPHER_CTX_
#pragma redefine_extname        EVP_CIPHER_CTX_init sunw_EVP_CIPHER_CTX_init
#pragma redefine_extname        EVP_CIPHER_CTX_iv_length sunw_EVP_CIPHER_CTX_iv_
#pragma redefine_extname        EVP_CIPHER_CTX_key_length sunw_EVP_CIPHER_CTX_ke
#pragma redefine_extname        EVP_CIPHER_CTX_new sunw_EVP_CIPHER_CTX_new
#pragma redefine_extname        EVP_CIPHER_CTX_nid sunw_EVP_CIPHER_CTX_nid
#pragma redefine_extname        EVP_CIPHER_CTX_rand_key sunw_EVP_CIPHER_CTX_rand
#pragma redefine_extname        EVP_CIPHER_CTX_set_app_data sunw_EVP_CIPHER_CTX_
#pragma redefine_extname        EVP_CIPHER_CTX_set_flags sunw_EVP_CIPHER_CTX_set
#pragma redefine_extname        EVP_CIPHER_CTX_set_key_length sunw_EVP_CIPHER_CT
#pragma redefine_extname        EVP_CIPHER_CTX_set_padding sunw_EVP_CIPHER_CTX_s
#pragma redefine_extname        EVP_CIPHER_CTX_test_flags sunw_EVP_CIPHER_CTX_te
#pragma redefine_extname        EVP_CIPHER_do_all sunw_EVP_CIPHER_do_all
#pragma redefine_extname        EVP_CIPHER_do_all_sorted sunw_EVP_CIPHER_do_all_
#pragma redefine_extname        EVP_CIPHER_flags sunw_EVP_CIPHER_flags
#pragma redefine_extname        EVP_CIPHER_get_asn1_iv sunw_EVP_CIPHER_get_asn1_
#pragma redefine_extname        EVP_CIPHER_iv_length sunw_EVP_CIPHER_iv_length
#pragma redefine_extname        EVP_CIPHER_key_length sunw_EVP_CIPHER_key_length
#pragma redefine_extname        EVP_CIPHER_nid sunw_EVP_CIPHER_nid
#pragma redefine_extname        EVP_CIPHER_param_to_asn1 sunw_EVP_CIPHER_param_t
#pragma redefine_extname        EVP_CIPHER_set_asn1_iv sunw_EVP_CIPHER_set_asn1_
#pragma redefine_extname        EVP_CIPHER_type sunw_EVP_CIPHER_type
#pragma redefine_extname        EVP_CipherFinal sunw_EVP_CipherFinal
#pragma redefine_extname        EVP_CipherFinal_ex sunw_EVP_CipherFinal_ex
#pragma redefine_extname        EVP_CipherInit sunw_EVP_CipherInit
#pragma redefine_extname        EVP_CipherInit_ex sunw_EVP_CipherInit_ex
#pragma redefine_extname        EVP_CipherUpdate sunw_EVP_CipherUpdate
#pragma redefine_extname        EVP_cleanup sunw_EVP_cleanup
#pragma redefine_extname        EVP_DecodeBlock sunw_EVP_DecodeBlock
#pragma redefine_extname        EVP_DecodeFinal sunw_EVP_DecodeFinal
#pragma redefine_extname        EVP_DecodeInit sunw_EVP_DecodeInit
#pragma redefine_extname        EVP_DecodeUpdate sunw_EVP_DecodeUpdate
#pragma redefine_extname        EVP_DecryptFinal sunw_EVP_DecryptFinal
#pragma redefine_extname        EVP_DecryptFinal_ex sunw_EVP_DecryptFinal_ex
#pragma redefine_extname        EVP_DecryptInit sunw_EVP_DecryptInit
#pragma redefine_extname        EVP_DecryptInit_ex sunw_EVP_DecryptInit_ex
#pragma redefine_extname        EVP_DecryptUpdate sunw_EVP_DecryptUpdate
#pragma redefine_extname        EVP_des_cbc sunw_EVP_des_cbc
#pragma redefine_extname        EVP_des_cfb sunw_EVP_des_cfb
#pragma redefine_extname        EVP_des_cfb1 sunw_EVP_des_cfb1
#pragma redefine_extname        EVP_des_cfb64 sunw_EVP_des_cfb64
#pragma redefine_extname        EVP_des_cfb8 sunw_EVP_des_cfb8
#pragma redefine_extname        EVP_des_ecb sunw_EVP_des_ecb
#pragma redefine_extname        EVP_des_ede sunw_EVP_des_ede
#pragma redefine_extname        EVP_des_ede_cbc sunw_EVP_des_ede_cbc
#pragma redefine_extname        EVP_des_ede_cfb sunw_EVP_des_ede_cfb
#pragma redefine_extname        EVP_des_ede_cfb64 sunw_EVP_des_ede_cfb64
#pragma redefine_extname        EVP_des_ede_ecb sunw_EVP_des_ede_ecb
#pragma redefine_extname        EVP_des_ede_ofb sunw_EVP_des_ede_ofb
#pragma redefine_extname        EVP_des_ede3 sunw_EVP_des_ede3
#pragma redefine_extname        EVP_des_ede3_cbc sunw_EVP_des_ede3_cbc
#pragma redefine_extname        EVP_des_ede3_cfb sunw_EVP_des_ede3_cfb
#pragma redefine_extname        EVP_des_ede3_cfb1 sunw_EVP_des_ede3_cfb1
#pragma redefine_extname        EVP_des_ede3_cfb64 sunw_EVP_des_ede3_cfb64
#pragma redefine_extname        EVP_des_ede3_cfb8 sunw_EVP_des_ede3_cfb8
#pragma redefine_extname        EVP_des_ede3_ecb sunw_EVP_des_ede3_ecb
#pragma redefine_extname        EVP_des_ede3_ofb sunw_EVP_des_ede3_ofb
#pragma redefine_extname        EVP_des_ofb sunw_EVP_des_ofb
#pragma redefine_extname        EVP_desx_cbc sunw_EVP_desx_cbc
#pragma redefine_extname        EVP_Digest sunw_EVP_Digest
#pragma redefine_extname        EVP_DigestFinal sunw_EVP_DigestFinal
#pragma redefine_extname        EVP_DigestFinal_ex sunw_EVP_DigestFinal_ex
#pragma redefine_extname        EVP_DigestInit sunw_EVP_DigestInit
#pragma redefine_extname        EVP_DigestInit_ex sunw_EVP_DigestInit_ex
#pragma redefine_extname        EVP_DigestSignFinal sunw_EVP_DigestSignFinal
#pragma redefine_extname        EVP_DigestSignInit sunw_EVP_DigestSignInit
```

```
#pragma redefine_extname        EVP_DigestUpdate sunw_EVP_DigestUpdate
#pragma redefine_extname        EVP_DigestVerifyFinal sunw_EVP_DigestVerifyFinal
#pragma redefine_extname        EVP_DigestVerifyInit sunw_EVP_DigestVerifyInit
#pragma redefine_extname        EVP_dss sunw_EVP_dss
#pragma redefine_extname        EVP_dss1 sunw_EVP_dss1
#pragma redefine_extname        EVP_ecdsa sunw_EVP_ecdsa
#pragma redefine_extname        EVP_enc_null sunw_EVP_enc_null
#pragma redefine_extname        EVP_EncodeBlock sunw_EVP_EncodeBlock
#pragma redefine_extname        EVP_EncodeFinal sunw_EVP_EncodeFinal
#pragma redefine_extname        EVP_EncodeInit sunw_EVP_EncodeInit
#pragma redefine_extname        EVP_EncodeUpdate sunw_EVP_EncodeUpdate
#pragma redefine_extname        EVP_EncryptFinal sunw_EVP_EncryptFinal
#pragma redefine_extname        EVP_EncryptFinal_ex sunw_EVP_EncryptFinal_ex
#pragma redefine_extname        EVP_EncryptInit sunw_EVP_EncryptInit
#pragma redefine_extname        EVP_EncryptInit_ex sunw_EVP_EncryptInit_ex
#pragma redefine_extname        EVP_EncryptUpdate sunw_EVP_EncryptUpdate
#pragma redefine_extname        EVP_get_cipherbyname sunw_EVP_get_cipherbyname
#pragma redefine_extname        EVP_get_digestbyname sunw_EVP_get_digestbyname
#pragma redefine_extname        EVP_get_pw_prompt sunw_EVP_get_pw_prompt
#pragma redefine_extname        EVP_MD_block_size sunw_EVP_MD_block_size
#pragma redefine_extname        EVP_MD_CTX_cleanup sunw_EVP_MD_CTX_cleanup
#pragma redefine_extname        EVP_MD_CTX_clear_flags sunw_EVP_MD_CTX_clear_fla
#pragma redefine_extname        EVP_MD_CTX_copy sunw_EVP_MD_CTX_copy
#pragma redefine_extname        EVP_MD_CTX_copy_ex sunw_EVP_MD_CTX_copy_ex
#pragma redefine_extname        EVP_MD_CTX_create sunw_EVP_MD_CTX_create
#pragma redefine_extname        EVP_MD_CTX_destroy sunw_EVP_MD_CTX_destroy
#pragma redefine_extname        EVP_MD_CTX_init sunw_EVP_MD_CTX_init
#pragma redefine_extname        EVP_MD_CTX_md sunw_EVP_MD_CTX_md
#pragma redefine_extname        EVP_MD_CTX_set_flags sunw_EVP_MD_CTX_set_flags
#pragma redefine_extname        EVP_MD_CTX_test_flags sunw_EVP_MD_CTX_test_flags
#pragma redefine_extname        EVP_MD_do_all sunw_EVP_MD_do_all
#pragma redefine_extname        EVP_MD_do_all_sorted sunw_EVP_MD_do_all_sorted
#pragma redefine_extname        EVP_MD_flags sunw_EVP_MD_flags
#pragma redefine_extname        EVP_md_null sunw_EVP_md_null
#pragma redefine_extname        EVP_MD_pkey_type sunw_EVP_MD_pkey_type
#pragma redefine_extname        EVP_MD_size sunw_EVP_MD_size
#pragma redefine_extname        EVP_MD_type sunw_EVP_MD_type
#pragma redefine_extname        EVP_md2 sunw_EVP_md2
#pragma redefine_extname        EVP_md4 sunw_EVP_md4
#pragma redefine_extname        EVP_md5 sunw_EVP_md5
#pragma redefine_extname        EVP_OpenFinal sunw_EVP_OpenFinal
#pragma redefine_extname        EVP_OpenInit sunw_EVP_OpenInit
#pragma redefine_extname        EVP_PBE_alg_add sunw_EVP_PBE_alg_add
#pragma redefine_extname        EVP_PBE_alg_add_type sunw_EVP_PBE_alg_add_type
#pragma redefine_extname        EVP_PBE_CipherInit sunw_EVP_PBE_CipherInit
#pragma redefine_extname        EVP_PBE_cleanup sunw_EVP_PBE_cleanup
#pragma redefine_extname        EVP_PBE_find sunw_EVP_PBE_find
#pragma redefine_extname        EVP_PKCS82PKEY sunw_EVP_PKCS82PKEY
#pragma redefine_extname        EVP_PKEY_add1_attr sunw_EVP_PKEY_add1_attr
#pragma redefine_extname        EVP_PKEY_add1_attr_by_NID sunw_EVP_PKEY_add1_att
#pragma redefine_extname        EVP_PKEY_add1_attr_by_OBJ sunw_EVP_PKEY_add1_att
#pragma redefine_extname        EVP_PKEY_add1_attr_by_txt sunw_EVP_PKEY_add1_att
#pragma redefine_extname        EVP_PKEY_asn1_add_alias sunw_EVP_PKEY_asn1_add_a
#pragma redefine_extname        EVP_PKEY_asn1_add0 sunw_EVP_PKEY_asn1_add0
#pragma redefine_extname        EVP_PKEY_asn1_copy sunw_EVP_PKEY_asn1_copy
#pragma redefine_extname        EVP_PKEY_asn1_find sunw_EVP_PKEY_asn1_find
#pragma redefine_extname        EVP_PKEY_asn1_find_str sunw_EVP_PKEY_asn1_find_s
#pragma redefine_extname        EVP_PKEY_asn1_free sunw_EVP_PKEY_asn1_free
#pragma redefine_extname        EVP_PKEY_asn1_get_count sunw_EVP_PKEY_asn1_get_c
#pragma redefine_extname        EVP_PKEY_asn1_get0 sunw_EVP_PKEY_asn1_get0
#pragma redefine_extname        EVP_PKEY_asn1_get0_info sunw_EVP_PKEY_asn1_get0_
#pragma redefine_extname        EVP_PKEY_asn1_new sunw_EVP_PKEY_asn1_new
#pragma redefine_extname        EVP_PKEY_asn1_set_ctrl sunw_EVP_PKEY_asn1_set_ct
#pragma redefine_extname        EVP_PKEY_asn1_set_free sunw_EVP_PKEY_asn1_set_fr
#pragma redefine_extname        EVP_PKEY_asn1_set_param sunw_EVP_PKEY_asn1_set_p
#pragma redefine_extname        EVP_PKEY_asn1_set_private sunw_EVP_PKEY_asn1_set
```



```
#pragma redefine_extname        EVP_PKEY_asn1_set_public sunw_EVP_PKEY_asn1_set_
#pragma redefine_extname        EVP_PKEY_assign sunw_EVP_PKEY_assign
#pragma redefine_extname        EVP_PKEY_base_id sunw_EVP_PKEY_base_id
#pragma redefine_extname        EVP_PKEY_bits sunw_EVP_PKEY_bits
#pragma redefine_extname        EVP_PKEY_cmp sunw_EVP_PKEY_cmp
#pragma redefine_extname        EVP_PKEY_cmp_parameters sunw_EVP_PKEY_cmp_parame
#pragma redefine_extname        EVP_PKEY_copy_parameters sunw_EVP_PKEY_copy_para
#pragma redefine_extname        EVP_PKEY_CTX_ctrl sunw_EVP_PKEY_CTX_ctrl
#pragma redefine_extname        EVP_PKEY_CTX_ctrl_str sunw_EVP_PKEY_CTX_ctrl_str
#pragma redefine_extname        EVP_PKEY_CTX_dup sunw_EVP_PKEY_CTX_dup
#pragma redefine_extname        EVP_PKEY_CTX_free sunw_EVP_PKEY_CTX_free
#pragma redefine_extname        EVP_PKEY_CTX_get_app_data sunw_EVP_PKEY_CTX_get_
#pragma redefine_extname        EVP_PKEY_CTX_get_cb sunw_EVP_PKEY_CTX_get_cb
#pragma redefine_extname        EVP_PKEY_CTX_get_data sunw_EVP_PKEY_CTX_get_data
#pragma redefine_extname        EVP_PKEY_CTX_get_keygen_info sunw_EVP_PKEY_CTX_g
#pragma redefine_extname        EVP_PKEY_CTX_get_operation sunw_EVP_PKEY_CTX_get
#pragma redefine_extname        EVP_PKEY_CTX_get0_peerkey sunw_EVP_PKEY_CTX_get0
#pragma redefine_extname        EVP_PKEY_CTX_get0_pkey sunw_EVP_PKEY_CTX_get0_pk
#pragma redefine_extname        EVP_PKEY_CTX_new sunw_EVP_PKEY_CTX_new
#pragma redefine_extname        EVP_PKEY_CTX_new_id sunw_EVP_PKEY_CTX_new_id
#pragma redefine_extname        EVP_PKEY_CTX_set_app_data sunw_EVP_PKEY_CTX_set_
#pragma redefine_extname        EVP_PKEY_CTX_set_cb sunw_EVP_PKEY_CTX_set_cb
#pragma redefine_extname        EVP_PKEY_CTX_set_data sunw_EVP_PKEY_CTX_set_data
#pragma redefine_extname        EVP_PKEY_CTX_set0_keygen_info sunw_EVP_PKEY_CTX_
#pragma redefine_extname        EVP_PKEY_decrypt sunw_EVP_PKEY_decrypt
#pragma redefine_extname        EVP_PKEY_decrypt_init sunw_EVP_PKEY_decrypt_init
#pragma redefine_extname        EVP_PKEY_decrypt_old sunw_EVP_PKEY_decrypt_old
#pragma redefine_extname        EVP_PKEY_delete_attr sunw_EVP_PKEY_delete_attr
#pragma redefine_extname        EVP_PKEY_derive sunw_EVP_PKEY_derive
#pragma redefine_extname        EVP_PKEY_derive_init sunw_EVP_PKEY_derive_init
#pragma redefine_extname        EVP_PKEY_derive_set_peer sunw_EVP_PKEY_derive_se
#pragma redefine_extname        EVP_PKEY_encrypt sunw_EVP_PKEY_encrypt
#pragma redefine_extname        EVP_PKEY_encrypt_init sunw_EVP_PKEY_encrypt_init
#pragma redefine_extname        EVP_PKEY_encrypt_old sunw_EVP_PKEY_encrypt_old
#pragma redefine_extname        EVP_PKEY_free sunw_EVP_PKEY_free
#pragma redefine_extname        EVP_PKEY_get_attr sunw_EVP_PKEY_get_attr
#pragma redefine_extname        EVP_PKEY_get_attr_by_NID sunw_EVP_PKEY_get_attr_
#pragma redefine_extname        EVP_PKEY_get_attr_by_OBJ sunw_EVP_PKEY_get_attr_
#pragma redefine_extname        EVP_PKEY_get_attr_count sunw_EVP_PKEY_get_attr_c
#pragma redefine_extname        EVP_PKEY_get_default_digest_nid sunw_EVP_PKEY_ge
#pragma redefine_extname        EVP_PKEY_get0 sunw_EVP_PKEY_get0
#pragma redefine_extname        EVP_PKEY_get0_asn1 sunw_EVP_PKEY_get0_asn1
#pragma redefine_extname        EVP_PKEY_get1_DH sunw_EVP_PKEY_get1_DH
#pragma redefine_extname        EVP_PKEY_get1_DSA sunw_EVP_PKEY_get1_DSA
#pragma redefine_extname        EVP_PKEY_get1_RSA sunw_EVP_PKEY_get1_RSA
#pragma redefine_extname        EVP_PKEY_id sunw_EVP_PKEY_id
#pragma redefine_extname        EVP_PKEY_keygen sunw_EVP_PKEY_keygen
#pragma redefine_extname        EVP_PKEY_keygen_init sunw_EVP_PKEY_keygen_init
#pragma redefine_extname        EVP_PKEY_meth_add0 sunw_EVP_PKEY_meth_add0
#pragma redefine_extname        EVP_PKEY_meth_copy sunw_EVP_PKEY_meth_copy
#pragma redefine_extname        EVP_PKEY_meth_find sunw_EVP_PKEY_meth_find
#pragma redefine_extname        EVP_PKEY_meth_free sunw_EVP_PKEY_meth_free
#pragma redefine_extname        EVP_PKEY_meth_get0_info sunw_EVP_PKEY_meth_get0_
#pragma redefine_extname        EVP_PKEY_meth_new sunw_EVP_PKEY_meth_new
#pragma redefine_extname        EVP_PKEY_meth_set_cleanup sunw_EVP_PKEY_meth_set
#pragma redefine_extname        EVP_PKEY_meth_set_copy sunw_EVP_PKEY_meth_set_co
#pragma redefine_extname        EVP_PKEY_meth_set_ctrl sunw_EVP_PKEY_meth_set_ct
#pragma redefine_extname        EVP_PKEY_meth_set_decrypt sunw_EVP_PKEY_meth_set
#pragma redefine_extname        EVP_PKEY_meth_set_derive sunw_EVP_PKEY_meth_set_
#pragma redefine_extname        EVP_PKEY_meth_set_encrypt sunw_EVP_PKEY_meth_set
#pragma redefine_extname        EVP_PKEY_meth_set_init sunw_EVP_PKEY_meth_set_in
#pragma redefine_extname        EVP_PKEY_meth_set_keygen sunw_EVP_PKEY_meth_set_
#pragma redefine_extname        EVP_PKEY_meth_set_paramgen sunw_EVP_PKEY_meth_se
#pragma redefine_extname        EVP_PKEY_meth_set_sign sunw_EVP_PKEY_meth_set_si
#pragma redefine_extname        EVP_PKEY_meth_set_signctx sunw_EVP_PKEY_meth_set
#pragma redefine_extname        EVP_PKEY_meth_set_verify sunw_EVP_PKEY_meth_set_
```

```
1844 #pragma redefine_extname        EVP_PKEY_meth_set_verify_recover sunw_EVP_PKEY_m
1845 #pragma redefine_extname        EVP_PKEY_meth_set_verifyctx sunw_EVP_PKEY_meth_s
1846 #pragma redefine_extname        EVP_PKEY_missing_parameters sunw_EVP_PKEY_missin
1847 #pragma redefine_extname        EVP_PKEY_new sunw_EVP_PKEY_new
1848 #pragma redefine_extname        EVP_PKEY_new_mac_key sunw_EVP_PKEY_new_mac_key
1849 #pragma redefine_extname        EVP_PKEY_paramgen sunw_EVP_PKEY_paramgen
1850 #pragma redefine_extname        EVP_PKEY_paramgen_init sunw_EVP_PKEY_paramgen_in
1851 #pragma redefine_extname        EVP_PKEY_print_params sunw_EVP_PKEY_print_params
1852 #pragma redefine_extname        EVP_PKEY_print_private sunw_EVP_PKEY_print_priva
1853 #pragma redefine_extname        EVP_PKEY_print_public sunw_EVP_PKEY_print_public
1854 #pragma redefine_extname        EVP_PKEY_save_parameters sunw_EVP_PKEY_save_para
1855 #pragma redefine_extname        evp_pkey_set_cb_translate sunw_evp_pkey_set_cb_t
1856 #pragma redefine_extname        EVP_PKEY_set_type sunw_EVP_PKEY_set_type
1857 #pragma redefine_extname        EVP_PKEY_set_type_str sunw_EVP_PKEY_set_type_str
1858 #pragma redefine_extname        EVP_PKEY_set1_DH sunw_EVP_PKEY_set1_DH
1859 #pragma redefine_extname        EVP_PKEY_set1_DSA sunw_EVP_PKEY_set1_DSA
1860 #pragma redefine_extname        EVP_PKEY_set1_RSA sunw_EVP_PKEY_set1_RSA
1861 #pragma redefine_extname        EVP_PKEY_sign sunw_EVP_PKEY_sign
1862 #pragma redefine_extname        EVP_PKEY_sign_init sunw_EVP_PKEY_sign_init
1863 #pragma redefine_extname        EVP_PKEY_size sunw_EVP_PKEY_size
1864 #pragma redefine_extname        EVP_PKEY_type sunw_EVP_PKEY_type
1865 #pragma redefine_extname        EVP_PKEY_verify sunw_EVP_PKEY_verify
1866 #pragma redefine_extname        EVP_PKEY_verify_init sunw_EVP_PKEY_verify_init
1867 #pragma redefine_extname        EVP_PKEY_verify_recover sunw_EVP_PKEY_verify_rec
1868 #pragma redefine_extname        EVP_PKEY_verify_recover_init sunw_EVP_PKEY_verif
1869 #pragma redefine_extname        EVP_PKEY2PKCS8 sunw_EVP_PKEY2PKCS8
1870 #pragma redefine_extname        EVP_PKEY2PKCS8_broken sunw_EVP_PKEY2PKCS8_broken
1871 #pragma redefine_extname        EVP_rc2_40_cbc sunw_EVP_rc2_40_cbc
1872 #pragma redefine_extname        EVP_rc2_64_cbc sunw_EVP_rc2_64_cbc
1873 #pragma redefine_extname        EVP_rc2_cbc sunw_EVP_rc2_cbc
1874 #pragma redefine_extname        EVP_rc2_cfb sunw_EVP_rc2_cfb
1875 #pragma redefine_extname        EVP_rc2_cfb64 sunw_EVP_rc2_cfb64
1876 #pragma redefine_extname        EVP_rc2_ecb sunw_EVP_rc2_ecb
1877 #pragma redefine_extname        EVP_rc2_ofb sunw_EVP_rc2_ofb
1878 #pragma redefine_extname        EVP_rc4 sunw_EVP_rc4
1879 #pragma redefine_extname        EVP_rc4_40 sunw_EVP_rc4_40
1880 #pragma redefine_extname        EVP_rc4_hmac_md5 sunw_EVP_rc4_hmac_md5
1881 #pragma redefine_extname        EVP_read_pw_string sunw_EVP_read_pw_string
1882 #pragma redefine_extname        EVP_read_pw_string_min sunw_EVP_read_pw_string_m
1883 #pragma redefine_extname        EVP_ripemd160 sunw_EVP_ripemd160
1884 #pragma redefine_extname        EVP_SealFinal sunw_EVP_SealFinal
1885 #pragma redefine_extname        EVP_SealInit sunw_EVP_SealInit
1886 #pragma redefine_extname        EVP_set_pw_prompt sunw_EVP_set_pw_prompt
1887 #pragma redefine_extname        EVP_sha sunw_EVP_sha
1888 #pragma redefine_extname        EVP_sha1 sunw_EVP_sha1
1889 #pragma redefine_extname        EVP_sha224 sunw_EVP_sha224
1890 #pragma redefine_extname        EVP_sha256 sunw_EVP_sha256
1891 #pragma redefine_extname        EVP_sha384 sunw_EVP_sha384
1892 #pragma redefine_extname        EVP_sha512 sunw_EVP_sha512
1893 #pragma redefine_extname        EVP_SignFinal sunw_EVP_SignFinal
1894 #pragma redefine_extname        EVP_VerifyFinal sunw_EVP_VerifyFinal
1895 #pragma redefine_extname        EVP_version sunw_EVP_version
1896 #pragma redefine_extname        EXTENDED_KEY_USAGE_free sunw_EXTENDED_KEY_USAGE_
1897 #pragma redefine_extname        EXTENDED_KEY_USAGE_it sunw_EXTENDED_KEY_USAGE_it
1898 #pragma redefine_extname        EXTENDED_KEY_USAGE_new sunw_EXTENDED_KEY_USAGE_n
1899 #pragma redefine_extname        fcrypt_body sunw_fcrypt_body
1900 #pragma redefine_extname        FIPS_mode sunw_FIPS_mode
1901 #pragma redefine_extname        FIPS_mode_set sunw_FIPS_mode_set
1902 #pragma redefine_extname        gcm_ghash_4bit sunw_gcm_ghash_4bit
1903 #pragma redefine_extname        gcm_ghash_4bit_mmx sunw_gcm_ghash_4bit_mmx
1904 #pragma redefine_extname        gcm_ghash_4bit_x86 sunw_gcm_ghash_4bit_x86
1905 #pragma redefine_extname        gcm_ghash_clmul sunw_gcm_ghash_clmul
1906 #pragma redefine_extname        gcm_gmult_4bit sunw_gcm_gmult_4bit
1907 #pragma redefine_extname        gcm_gmult_4bit_mmx sunw_gcm_gmult_4bit_mmx
1908 #pragma redefine_extname        gcm_gmult_4bit_x86 sunw_gcm_gmult_4bit_x86
1909 #pragma redefine_extname        gcm_gmult_clmul sunw_gcm_gmult_clmul
```



```
1910 #pragma redefine_extname        gcm_init_clmul sunw_gcm_init_clmul
1911 #pragma redefine_extname        GENERAL_NAME_cmp sunw_GENERAL_NAME_cmp
1912 #pragma redefine_extname        GENERAL_NAME_dup sunw_GENERAL_NAME_dup
1913 #pragma redefine_extname        GENERAL_NAME_free sunw_GENERAL_NAME_free
1914 #pragma redefine_extname        GENERAL_NAME_get0_otherName sunw_GENERAL_NAME_ge
1915 #pragma redefine_extname        GENERAL_NAME_get0_value sunw_GENERAL_NAME_get0_v
1916 #pragma redefine_extname        GENERAL_NAME_it sunw_GENERAL_NAME_it
1917 #pragma redefine_extname        GENERAL_NAME_new sunw_GENERAL_NAME_new
1918 #pragma redefine_extname        GENERAL_NAME_print sunw_GENERAL_NAME_print
1919 #pragma redefine_extname        GENERAL_NAME_set0_othername sunw_GENERAL_NAME_se
1920 #pragma redefine_extname        GENERAL_NAME_set0_value sunw_GENERAL_NAME_set0_v
1921 #pragma redefine_extname        GENERAL_NAMES_free sunw_GENERAL_NAMES_free
1922 #pragma redefine_extname        GENERAL_NAMES_it sunw_GENERAL_NAMES_it
1923 #pragma redefine_extname        GENERAL_NAMES_new sunw_GENERAL_NAMES_new
1924 #pragma redefine_extname        GENERAL_SUBTREE_free sunw_GENERAL_SUBTREE_free
1925 #pragma redefine_extname        GENERAL_SUBTREE_it sunw_GENERAL_SUBTREE_it
1926 #pragma redefine_extname        GENERAL_SUBTREE_new sunw_GENERAL_SUBTREE_new
1927 #pragma redefine_extname        get_rfc2409_prime_1024 sunw_get_rfc2409_prime_10
1928 #pragma redefine_extname        get_rfc2409_prime_768 sunw_get_rfc2409_prime_768
1929 #pragma redefine_extname        get_rfc3526_prime_1536 sunw_get_rfc3526_prime_15
1930 #pragma redefine_extname        get_rfc3526_prime_2048 sunw_get_rfc3526_prime_20
1931 #pragma redefine_extname        get_rfc3526_prime_3072 sunw_get_rfc3526_prime_30
1932 #pragma redefine_extname        get_rfc3526_prime_4096 sunw_get_rfc3526_prime_40
1933 #pragma redefine_extname        get_rfc3526_prime_6144 sunw_get_rfc3526_prime_61
1934 #pragma redefine_extname        get_rfc3526_prime_8192 sunw_get_rfc3526_prime_81
1935 #pragma redefine_extname        hex_to_string sunw_hex_to_string
1936 #pragma redefine_extname        HMAC sunw_HMAC
1937 #pragma redefine_extname        hmac_asn1_meth sunw_hmac_asn1_meth
1938 #pragma redefine_extname        HMAC_CTX_cleanup sunw_HMAC_CTX_cleanup
1939 #pragma redefine_extname        HMAC_CTX_copy sunw_HMAC_CTX_copy
1940 #pragma redefine_extname        HMAC_CTX_init sunw_HMAC_CTX_init
1941 #pragma redefine_extname        HMAC_CTX_set_flags sunw_HMAC_CTX_set_flags
1942 #pragma redefine_extname        HMAC_Final sunw_HMAC_Final
1943 #pragma redefine_extname        HMAC_Init sunw_HMAC_Init
1944 #pragma redefine_extname        HMAC_Init_ex sunw_HMAC_Init_ex
1945 #pragma redefine_extname        hmac_pkey_meth sunw_hmac_pkey_meth
1946 #pragma redefine_extname        HMAC_Update sunw_HMAC_Update
1947 #pragma redefine_extname        i2a_ACCESS_DESCRIPTION sunw_i2a_ACCESS_DESCRIPTI
1948 #pragma redefine_extname        i2a_ASN1_ENUMERATED sunw_i2a_ASN1_ENUMERATED
1949 #pragma redefine_extname        i2a_ASN1_INTEGER sunw_i2a_ASN1_INTEGER
1950 #pragma redefine_extname        i2a_ASN1_OBJECT sunw_i2a_ASN1_OBJECT
1951 #pragma redefine_extname        i2a_ASN1_STRING sunw_i2a_ASN1_STRING
1952 #pragma redefine_extname        i2b_PrivateKey_bio sunw_i2b_PrivateKey_bio
1953 #pragma redefine_extname        i2b_PublicKey_bio sunw_i2b_PublicKey_bio
1954 #pragma redefine_extname        i2b_PVK_bio sunw_i2b_PVK_bio
1955 #pragma redefine_extname        i2c_ASN1_BIT_STRING sunw_i2c_ASN1_BIT_STRING
1956 #pragma redefine_extname        i2c_ASN1_INTEGER sunw_i2c_ASN1_INTEGER
1957 #pragma redefine_extname        i2d_ACCESS_DESCRIPTION sunw_i2d_ACCESS_DESCRIPTI
1958 #pragma redefine_extname        i2d_ASN1_bio_stream sunw_i2d_ASN1_bio_stream
1959 #pragma redefine_extname        i2d_ASN1_BIT_STRING sunw_i2d_ASN1_BIT_STRING
1960 #pragma redefine_extname        i2d_ASN1_BMPSTRING sunw_i2d_ASN1_BMPSTRING
1961 #pragma redefine_extname        i2d_ASN1_BOOLEAN sunw_i2d_ASN1_BOOLEAN
1962 #pragma redefine_extname        i2d_ASN1_bytes sunw_i2d_ASN1_bytes
1963 #pragma redefine_extname        i2d_ASN1_ENUMERATED sunw_i2d_ASN1_ENUMERATED
1964 #pragma redefine_extname        i2d_ASN1_GENERALIZEDTIME sunw_i2d_ASN1_GENERALIZ
1965 #pragma redefine_extname        i2d_ASN1_GENERALSTRING sunw_i2d_ASN1_GENERALSTRI
1966 #pragma redefine_extname        i2d_ASN1_IA5STRING sunw_i2d_ASN1_IA5STRING
1967 #pragma redefine_extname        i2d_ASN1_INTEGER sunw_i2d_ASN1_INTEGER
1968 #pragma redefine_extname        i2d_ASN1_NULL sunw_i2d_ASN1_NULL
1969 #pragma redefine_extname        i2d_ASN1_OBJECT sunw_i2d_ASN1_OBJECT
1970 #pragma redefine_extname        i2d_ASN1_OCTET_STRING sunw_i2d_ASN1_OCTET_STRING
1971 #pragma redefine_extname        i2d_ASN1_PRINTABLE sunw_i2d_ASN1_PRINTABLE
1972 #pragma redefine_extname        i2d_ASN1_PRINTABLESTRING sunw_i2d_ASN1_PRINTABLE
1973 #pragma redefine_extname        i2d_ASN1_SEQUENCE_ANY sunw_i2d_ASN1_SEQUENCE_ANY
1974 #pragma redefine_extname        i2d_ASN1_SET sunw_i2d_ASN1_SET
1975 #pragma redefine_extname        i2d_ASN1_SET_ANY sunw_i2d_ASN1_SET_ANY
```

```
#pragma redefine_extname        i2d_ASN1_T61STRING sunw_i2d_ASN1_T61STRING
#pragma redefine_extname        i2d_ASN1_TIME sunw_i2d_ASN1_TIME
#pragma redefine_extname        i2d_ASN1_TYPE sunw_i2d_ASN1_TYPE
#pragma redefine_extname        i2d_ASN1_UNIVERSALSTRING sunw_i2d_ASN1_UNIVERSAL
#pragma redefine_extname        i2d_ASN1_UTCTIME sunw_i2d_ASN1_UTCTIME
#pragma redefine_extname        i2d_ASN1_UTF8STRING sunw_i2d_ASN1_UTF8STRING
#pragma redefine_extname        i2d_ASN1_VISIBLESTRING sunw_i2d_ASN1_VISIBLESTRI
#pragma redefine_extname        i2d_AUTHORITY_INFO_ACCESS sunw_i2d_AUTHORITY_INF
#pragma redefine_extname        i2d_AUTHORITY_KEYID sunw_i2d_AUTHORITY_KEYID
#pragma redefine_extname        i2d_BASIC_CONSTRAINTS sunw_i2d_BASIC_CONSTRAINTS
#pragma redefine_extname        i2d_CERTIFICATEPOLICIES sunw_i2d_CERTIFICATEPOLI
#pragma redefine_extname        i2d_CMS_bio sunw_i2d_CMS_bio
#pragma redefine_extname        i2d_CMS_bio_stream sunw_i2d_CMS_bio_stream
#pragma redefine_extname        i2d_CMS_ContentInfo sunw_i2d_CMS_ContentInfo
#pragma redefine_extname        i2d_CMS_ReceiptRequest sunw_i2d_CMS_ReceiptReque
#pragma redefine_extname        i2d_CRL_DIST_POINTS sunw_i2d_CRL_DIST_POINTS
#pragma redefine_extname        i2d_DHparams sunw_i2d_DHparams
#pragma redefine_extname        i2d_DIRECTORYSTRING sunw_i2d_DIRECTORYSTRING
#pragma redefine_extname        i2d_DISPLAYTEXT sunw_i2d_DISPLAYTEXT
#pragma redefine_extname        i2d_DIST_POINT sunw_i2d_DIST_POINT
#pragma redefine_extname        i2d_DIST_POINT_NAME sunw_i2d_DIST_POINT_NAME
#pragma redefine_extname        i2d_DSA_PUBKEY sunw_i2d_DSA_PUBKEY
#pragma redefine_extname        i2d_DSA_PUBKEY_bio sunw_i2d_DSA_PUBKEY_bio
#pragma redefine_extname        i2d_DSA_PUBKEY_fp sunw_i2d_DSA_PUBKEY_fp
#pragma redefine_extname        i2d_DSA_SIG sunw_i2d_DSA_SIG
#pragma redefine_extname        i2d_DSAparams sunw_i2d_DSAparams
#pragma redefine_extname        i2d_DSAPrivateKey sunw_i2d_DSAPrivateKey
#pragma redefine_extname        i2d_DSAPrivateKey_bio sunw_i2d_DSAPrivateKey_bio
#pragma redefine_extname        i2d_DSAPrivateKey_fp sunw_i2d_DSAPrivateKey_fp
#pragma redefine_extname        i2d_DSAPublicKey sunw_i2d_DSAPublicKey
#pragma redefine_extname        i2d_EDIPARTYNAME sunw_i2d_EDIPARTYNAME
#pragma redefine_extname        i2d_ESS_CERT_ID sunw_i2d_ESS_CERT_ID
#pragma redefine_extname        i2d_ESS_ISSUER_SERIAL sunw_i2d_ESS_ISSUER_SERIAL
#pragma redefine_extname        i2d_ESS_SIGNING_CERT sunw_i2d_ESS_SIGNING_CERT
#pragma redefine_extname        i2d_EXTENDED_KEY_USAGE sunw_i2d_EXTENDED_KEY_USA
#pragma redefine_extname        i2d_GENERAL_NAME sunw_i2d_GENERAL_NAME
#pragma redefine_extname        i2d_GENERAL_NAMES sunw_i2d_GENERAL_NAMES
#pragma redefine_extname        i2d_ISSUING_DIST_POINT sunw_i2d_ISSUING_DIST_POI
#pragma redefine_extname        i2d_KRB5_APREQ sunw_i2d_KRB5_APREQ
#pragma redefine_extname        i2d_KRB5_APREQBODY sunw_i2d_KRB5_APREQBODY
#pragma redefine_extname        i2d_KRB5_AUTHDATA sunw_i2d_KRB5_AUTHDATA
#pragma redefine_extname        i2d_KRB5_AUTHENT sunw_i2d_KRB5_AUTHENT
#pragma redefine_extname        i2d_KRB5_AUTHENTBODY sunw_i2d_KRB5_AUTHENTBODY
#pragma redefine_extname        i2d_KRB5_CHECKSUM sunw_i2d_KRB5_CHECKSUM
#pragma redefine_extname        i2d_KRB5_ENCDATA sunw_i2d_KRB5_ENCDATA
#pragma redefine_extname        i2d_KRB5_ENCKEY sunw_i2d_KRB5_ENCKEY
#pragma redefine_extname        i2d_KRB5_PRINCNAME sunw_i2d_KRB5_PRINCNAME
#pragma redefine_extname        i2d_KRB5_TICKET sunw_i2d_KRB5_TICKET
#pragma redefine_extname        i2d_KRB5_TKTBODY sunw_i2d_KRB5_TKTBODY
#pragma redefine_extname        i2d_NETSCAPE_CERT_SEQUENCE sunw_i2d_NETSCAPE_CER
#pragma redefine_extname        i2d_NETSCAPE_ENCRYPTED_PKEY sunw_i2d_NETSCAPE_EN
#pragma redefine_extname        i2d_NETSCAPE_PKEY sunw_i2d_NETSCAPE_PKEY
#pragma redefine_extname        i2d_Netscape_RSA sunw_i2d_Netscape_RSA
#pragma redefine_extname        i2d_NETSCAPE_SPKAC sunw_i2d_NETSCAPE_SPKAC
#pragma redefine_extname        i2d_NETSCAPE_SPKI sunw_i2d_NETSCAPE_SPKI
#pragma redefine_extname        i2d_NETSCAPE_X509 sunw_i2d_NETSCAPE_X509
#pragma redefine_extname        i2d_NOTICEREF sunw_i2d_NOTICEREF
#pragma redefine_extname        i2d_OCSP_BASICRESP sunw_i2d_OCSP_BASICRESP
#pragma redefine_extname        i2d_OCSP_CERTID sunw_i2d_OCSP_CERTID
#pragma redefine_extname        i2d_OCSP_CERTSTATUS sunw_i2d_OCSP_CERTSTATUS
#pragma redefine_extname        i2d_OCSP_CRLID sunw_i2d_OCSP_CRLID
#pragma redefine_extname        i2d_OCSP_ONEREQ sunw_i2d_OCSP_ONEREQ
#pragma redefine_extname        i2d_OCSP_REQINFO sunw_i2d_OCSP_REQINFO
#pragma redefine_extname        i2d_OCSP_REQUEST sunw_i2d_OCSP_REQUEST
#pragma redefine_extname        i2d_OCSP_RESPBYTES sunw_i2d_OCSP_RESPBYTES
#pragma redefine_extname        i2d_OCSP_RESPDATA sunw_i2d_OCSP_RESPDATA
```



```
#pragma redefine_extname        i2d_OCSP_RESPID sunw_i2d_OCSP_RESPID
#pragma redefine_extname        i2d_OCSP_RESPONSE sunw_i2d_OCSP_RESPONSE
#pragma redefine_extname        i2d_OCSP_REVOKEDINFO sunw_i2d_OCSP_REVOKEDINFO
#pragma redefine_extname        i2d_OCSP_SERVICELOC sunw_i2d_OCSP_SERVICELOC
#pragma redefine_extname        i2d_OCSP_SIGNATURE sunw_i2d_OCSP_SIGNATURE
#pragma redefine_extname        i2d_OCSP_SINGLERESP sunw_i2d_OCSP_SINGLERESP
#pragma redefine_extname        i2d_OTHERNAME sunw_i2d_OTHERNAME
#pragma redefine_extname        i2d_PBE2PARAM sunw_i2d_PBE2PARAM
#pragma redefine_extname        i2d_PBEPARAM sunw_i2d_PBEPARAM
#pragma redefine_extname        i2d_PBKDF2PARAM sunw_i2d_PBKDF2PARAM
#pragma redefine_extname        i2d_PKCS12 sunw_i2d_PKCS12
#pragma redefine_extname        i2d_PKCS12_BAGS sunw_i2d_PKCS12_BAGS
#pragma redefine_extname        i2d_PKCS12_bio sunw_i2d_PKCS12_bio
#pragma redefine_extname        i2d_PKCS12_fp sunw_i2d_PKCS12_fp
#pragma redefine_extname        i2d_PKCS12_MAC_DATA sunw_i2d_PKCS12_MAC_DATA
#pragma redefine_extname        i2d_PKCS12_SAFEBAG sunw_i2d_PKCS12_SAFEBAG
#pragma redefine_extname        i2d_PKCS7 sunw_i2d_PKCS7
#pragma redefine_extname        i2d_PKCS7_bio sunw_i2d_PKCS7_bio
#pragma redefine_extname        i2d_PKCS7_bio_stream sunw_i2d_PKCS7_bio_stream
#pragma redefine_extname        i2d_PKCS7_DIGEST sunw_i2d_PKCS7_DIGEST
#pragma redefine_extname        i2d_PKCS7_ENC_CONTENT sunw_i2d_PKCS7_ENC_CONTENT
#pragma redefine_extname        i2d_PKCS7_ENCRYPT sunw_i2d_PKCS7_ENCRYPT
#pragma redefine_extname        i2d_PKCS7_ENVELOPE sunw_i2d_PKCS7_ENVELOPE
#pragma redefine_extname        i2d_PKCS7_fp sunw_i2d_PKCS7_fp
#pragma redefine_extname        i2d_PKCS7_ISSUER_AND_SERIAL sunw_i2d_PKCS7_ISSUE
#pragma redefine_extname        i2d_PKCS7_NDEF sunw_i2d_PKCS7_NDEF
#pragma redefine_extname        i2d_PKCS7_RECIP_INFO sunw_i2d_PKCS7_RECIP_INFO
#pragma redefine_extname        i2d_PKCS7_SIGN_ENVELOPE sunw_i2d_PKCS7_SIGN_ENVE
#pragma redefine_extname        i2d_PKCS7_SIGNED sunw_i2d_PKCS7_SIGNED
#pragma redefine_extname        i2d_PKCS7_SIGNER_INFO sunw_i2d_PKCS7_SIGNER_INFO
#pragma redefine_extname        i2d_PKCS8_bio sunw_i2d_PKCS8_bio
#pragma redefine_extname        i2d_PKCS8_fp sunw_i2d_PKCS8_fp
#pragma redefine_extname        i2d_PKCS8_PRIV_KEY_INFO sunw_i2d_PKCS8_PRIV_KEY_
#pragma redefine_extname        i2d_PKCS8_PRIV_KEY_INFO_bio sunw_i2d_PKCS8_PRIV_
#pragma redefine_extname        i2d_PKCS8_PRIV_KEY_INFO_fp sunw_i2d_PKCS8_PRIV_K
#pragma redefine_extname        i2d_PKCS8PrivateKey_bio sunw_i2d_PKCS8PrivateKey
#pragma redefine_extname        i2d_PKCS8PrivateKey_fp sunw_i2d_PKCS8PrivateKey_
#pragma redefine_extname        i2d_PKCS8PrivateKey_nid_bio sunw_i2d_PKCS8Privat
#pragma redefine_extname        i2d_PKCS8PrivateKey_nid_fp sunw_i2d_PKCS8Private
#pragma redefine_extname        i2d_PKCS8PrivateKeyInfo_bio sunw_i2d_PKCS8Privat
#pragma redefine_extname        i2d_PKCS8PrivateKeyInfo_fp sunw_i2d_PKCS8Private
#pragma redefine_extname        i2d_PKEY_USAGE_PERIOD sunw_i2d_PKEY_USAGE_PERIOD
#pragma redefine_extname        i2d_POLICYINFO sunw_i2d_POLICYINFO
#pragma redefine_extname        i2d_POLICYQUALINFO sunw_i2d_POLICYQUALINFO
#pragma redefine_extname        i2d_PrivateKey sunw_i2d_PrivateKey
#pragma redefine_extname        i2d_PrivateKey_bio sunw_i2d_PrivateKey_bio
#pragma redefine_extname        i2d_PrivateKey_fp sunw_i2d_PrivateKey_fp
#pragma redefine_extname        i2d_PROXY_CERT_INFO_EXTENSION sunw_i2d_PROXY_CER
#pragma redefine_extname        i2d_PROXY_POLICY sunw_i2d_PROXY_POLICY
#pragma redefine_extname        i2d_PUBKEY sunw_i2d_PUBKEY
#pragma redefine_extname        i2d_PUBKEY_bio sunw_i2d_PUBKEY_bio
#pragma redefine_extname        i2d_PUBKEY_fp sunw_i2d_PUBKEY_fp
#pragma redefine_extname        i2d_PublicKey sunw_i2d_PublicKey
#pragma redefine_extname        i2d_RSA_NET sunw_i2d_RSA_NET
#pragma redefine_extname        i2d_RSA_PSS_PARAMS sunw_i2d_RSA_PSS_PARAMS
#pragma redefine_extname        i2d_RSA_PUBKEY sunw_i2d_RSA_PUBKEY
#pragma redefine_extname        i2d_RSA_PUBKEY_bio sunw_i2d_RSA_PUBKEY_bio
#pragma redefine_extname        i2d_RSA_PUBKEY_fp sunw_i2d_RSA_PUBKEY_fp
#pragma redefine_extname        i2d_RSAPrivateKey sunw_i2d_RSAPrivateKey
#pragma redefine_extname        i2d_RSAPrivateKey_bio sunw_i2d_RSAPrivateKey_bio
#pragma redefine_extname        i2d_RSAPrivateKey_fp sunw_i2d_RSAPrivateKey_fp
#pragma redefine_extname        i2d_RSAPublicKey sunw_i2d_RSAPublicKey
#pragma redefine_extname        i2d_RSAPublicKey_bio sunw_i2d_RSAPublicKey_bio
#pragma redefine_extname        i2d_RSAPublicKey_fp sunw_i2d_RSAPublicKey_fp
#pragma redefine_extname        i2d_SXNET sunw_i2d_SXNET
#pragma redefine_extname        i2d_SXNETID sunw_i2d_SXNETID
```

```
2108 #pragma redefine_extname        i2d_TS_ACCURACY sunw_i2d_TS_ACCURACY
2109 #pragma redefine_extname        i2d_TS_MSG_IMPRINT sunw_i2d_TS_MSG_IMPRINT
2110 #pragma redefine_extname        i2d_TS_MSG_IMPRINT_bio sunw_i2d_TS_MSG_IMPRINT_b
2111 #pragma redefine_extname        i2d_TS_MSG_IMPRINT_fp sunw_i2d_TS_MSG_IMPRINT_fp
2112 #pragma redefine_extname        i2d_TS_REQ sunw_i2d_TS_REQ
2113 #pragma redefine_extname        i2d_TS_REQ_bio sunw_i2d_TS_REQ_bio
2114 #pragma redefine_extname        i2d_TS_REQ_fp sunw_i2d_TS_REQ_fp
2115 #pragma redefine_extname        i2d_TS_RESP sunw_i2d_TS_RESP
2116 #pragma redefine_extname        i2d_TS_RESP_bio sunw_i2d_TS_RESP_bio
2117 #pragma redefine_extname        i2d_TS_RESP_fp sunw_i2d_TS_RESP_fp
2118 #pragma redefine_extname        i2d_TS_STATUS_INFO sunw_i2d_TS_STATUS_INFO
2119 #pragma redefine_extname        i2d_TS_TST_INFO sunw_i2d_TS_TST_INFO
2120 #pragma redefine_extname        i2d_TS_TST_INFO_bio sunw_i2d_TS_TST_INFO_bio
2121 #pragma redefine_extname        i2d_TS_TST_INFO_fp sunw_i2d_TS_TST_INFO_fp
2122 #pragma redefine_extname        i2d_USERNOTICE sunw_i2d_USERNOTICE
2123 #pragma redefine_extname        i2d_X509 sunw_i2d_X509
2124 #pragma redefine_extname        i2d_X509_ALGOR sunw_i2d_X509_ALGOR
2125 #pragma redefine_extname        i2d_X509_ALGORS sunw_i2d_X509_ALGORS
2126 #pragma redefine_extname        i2d_X509_ATTRIBUTE sunw_i2d_X509_ATTRIBUTE
2127 #pragma redefine_extname        i2d_X509_AUX sunw_i2d_X509_AUX
2128 #pragma redefine_extname        i2d_X509_bio sunw_i2d_X509_bio
2129 #pragma redefine_extname        i2d_X509_CERT_AUX sunw_i2d_X509_CERT_AUX
2130 #pragma redefine_extname        i2d_X509_CERT_PAIR sunw_i2d_X509_CERT_PAIR
2131 #pragma redefine_extname        i2d_X509_CINF sunw_i2d_X509_CINF
2132 #pragma redefine_extname        i2d_X509_CRL sunw_i2d_X509_CRL
2133 #pragma redefine_extname        i2d_X509_CRL_bio sunw_i2d_X509_CRL_bio
2134 #pragma redefine_extname        i2d_X509_CRL_fp sunw_i2d_X509_CRL_fp
2135 #pragma redefine_extname        i2d_X509_CRL_INFO sunw_i2d_X509_CRL_INFO
2136 #pragma redefine_extname        i2d_X509_EXTENSION sunw_i2d_X509_EXTENSION
2137 #pragma redefine_extname        i2d_X509_EXTENSIONS sunw_i2d_X509_EXTENSIONS
2138 #pragma redefine_extname        i2d_X509_fp sunw_i2d_X509_fp
2139 #pragma redefine_extname        i2d_X509_NAME sunw_i2d_X509_NAME
2140 #pragma redefine_extname        i2d_X509_NAME_ENTRY sunw_i2d_X509_NAME_ENTRY
2141 #pragma redefine_extname        i2d_X509_PKEY sunw_i2d_X509_PKEY
2142 #pragma redefine_extname        i2d_X509_PUBKEY sunw_i2d_X509_PUBKEY
2143 #pragma redefine_extname        i2d_X509_REQ sunw_i2d_X509_REQ
2144 #pragma redefine_extname        i2d_X509_REQ_bio sunw_i2d_X509_REQ_bio
2145 #pragma redefine_extname        i2d_X509_REQ_fp sunw_i2d_X509_REQ_fp
2146 #pragma redefine_extname        i2d_X509_REQ_INFO sunw_i2d_X509_REQ_INFO
2147 #pragma redefine_extname        i2d_X509_REVOKED sunw_i2d_X509_REVOKED
2148 #pragma redefine_extname        i2d_X509_SIG sunw_i2d_X509_SIG
2149 #pragma redefine_extname        i2d_X509_VAL sunw_i2d_X509_VAL
2150 #pragma redefine_extname        i2s_ASN1_ENUMERATED sunw_i2s_ASN1_ENUMERATED
2151 #pragma redefine_extname        i2s_ASN1_ENUMERATED_TABLE sunw_i2s_ASN1_ENUMERAT
2152 #pragma redefine_extname        i2s_ASN1_INTEGER sunw_i2s_ASN1_INTEGER
2153 #pragma redefine_extname        i2s_ASN1_OCTET_STRING sunw_i2s_ASN1_OCTET_STRING
2154 #pragma redefine_extname        i2t_ASN1_OBJECT sunw_i2t_ASN1_OBJECT
2155 #pragma redefine_extname        i2v_ASN1_BIT_STRING sunw_i2v_ASN1_BIT_STRING
2156 #pragma redefine_extname        i2v_GENERAL_NAME sunw_i2v_GENERAL_NAME
2157 #pragma redefine_extname        i2v_GENERAL_NAMES sunw_i2v_GENERAL_NAMES
2158 #pragma redefine_extname        int_rsa_verify sunw_int_rsa_verify
2159 #pragma redefine_extname        ISSUING_DIST_POINT_free sunw_ISSUING_DIST_POINT_
2160 #pragma redefine_extname        ISSUING_DIST_POINT_it sunw_ISSUING_DIST_POINT_it
2161 #pragma redefine_extname        ISSUING_DIST_POINT_new sunw_ISSUING_DIST_POINT_n
2162 #pragma redefine_extname        KRB5_APREQ_free sunw_KRB5_APREQ_free
2163 #pragma redefine_extname        KRB5_APREQ_it sunw_KRB5_APREQ_it
2164 #pragma redefine_extname        KRB5_APREQ_new sunw_KRB5_APREQ_new
2165 #pragma redefine_extname        KRB5_APREQBODY_free sunw_KRB5_APREQBODY_free
2166 #pragma redefine_extname        KRB5_APREQBODY_it sunw_KRB5_APREQBODY_it
2167 #pragma redefine_extname        KRB5_APREQBODY_new sunw_KRB5_APREQBODY_new
2168 #pragma redefine_extname        KRB5_AUTHDATA_free sunw_KRB5_AUTHDATA_free
2169 #pragma redefine_extname        KRB5_AUTHDATA_it sunw_KRB5_AUTHDATA_it
2170 #pragma redefine_extname        KRB5_AUTHDATA_new sunw_KRB5_AUTHDATA_new
2171 #pragma redefine_extname        KRB5_AUTHENT_free sunw_KRB5_AUTHENT_free
2172 #pragma redefine_extname        KRB5_AUTHENT_it sunw_KRB5_AUTHENT_it
2173 #pragma redefine_extname        KRB5_AUTHENT_new sunw_KRB5_AUTHENT_new
```



```
2174 #pragma redefine_extname        KRB5_AUTHENTBODY_free sunw_KRB5_AUTHENTBODY_free
2175 #pragma redefine_extname        KRB5_AUTHENTBODY_it sunw_KRB5_AUTHENTBODY_it
2176 #pragma redefine_extname        KRB5_AUTHENTBODY_new sunw_KRB5_AUTHENTBODY_new
2177 #pragma redefine_extname        KRB5_CHECKSUM_free sunw_KRB5_CHECKSUM_free
2178 #pragma redefine_extname        KRB5_CHECKSUM_it sunw_KRB5_CHECKSUM_it
2179 #pragma redefine_extname        KRB5_CHECKSUM_new sunw_KRB5_CHECKSUM_new
2180 #pragma redefine_extname        KRB5_ENCDATA_free sunw_KRB5_ENCDATA_free
2181 #pragma redefine_extname        KRB5_ENCDATA_it sunw_KRB5_ENCDATA_it
2182 #pragma redefine_extname        KRB5_ENCDATA_new sunw_KRB5_ENCDATA_new
2183 #pragma redefine_extname        KRB5_ENCKEY_free sunw_KRB5_ENCKEY_free
2184 #pragma redefine_extname        KRB5_ENCKEY_it sunw_KRB5_ENCKEY_it
2185 #pragma redefine_extname        KRB5_ENCKEY_new sunw_KRB5_ENCKEY_new
2186 #pragma redefine_extname        KRB5_PRINCNAME_free sunw_KRB5_PRINCNAME_free
2187 #pragma redefine_extname        KRB5_PRINCNAME_it sunw_KRB5_PRINCNAME_it
2188 #pragma redefine_extname        KRB5_PRINCNAME_new sunw_KRB5_PRINCNAME_new
2189 #pragma redefine_extname        KRB5_TICKET_free sunw_KRB5_TICKET_free
2190 #pragma redefine_extname        KRB5_TICKET_it sunw_KRB5_TICKET_it
2191 #pragma redefine_extname        KRB5_TICKET_new sunw_KRB5_TICKET_new
2192 #pragma redefine_extname        KRB5_TKTBODY_free sunw_KRB5_TKTBODY_free
2193 #pragma redefine_extname        KRB5_TKTBODY_it sunw_KRB5_TKTBODY_it
2194 #pragma redefine_extname        KRB5_TKTBODY_new sunw_KRB5_TKTBODY_new
2195 #pragma redefine_extname        level_add_node sunw_level_add_node
2196 #pragma redefine_extname        level_find_node sunw_level_find_node
2197 #pragma redefine_extname        lh_delete sunw_lh_delete
2198 #pragma redefine_extname        lh_doall sunw_lh_doall
2199 #pragma redefine_extname        lh_doall_arg sunw_lh_doall_arg
2200 #pragma redefine_extname        lh_free sunw_lh_free
2201 #pragma redefine_extname        lh_insert sunw_lh_insert
2202 #pragma redefine_extname        lh_new sunw_lh_new
2203 #pragma redefine_extname        lh_node_stats sunw_lh_node_stats
2204 #pragma redefine_extname        lh_node_stats_bio sunw_lh_node_stats_bio
2205 #pragma redefine_extname        lh_node_usage_stats sunw_lh_node_usage_stats
2206 #pragma redefine_extname        lh_node_usage_stats_bio sunw_lh_node_usage_stats
2207 #pragma redefine_extname        lh_num_items sunw_lh_num_items
2208 #pragma redefine_extname        lh_retrieve sunw_lh_retrieve
2209 #pragma redefine_extname        lh_stats sunw_lh_stats
2210 #pragma redefine_extname        lh_stats_bio sunw_lh_stats_bio
2211 #pragma redefine_extname        lh_strhash sunw_lh_strhash
2212 #pragma redefine_extname        lh_version sunw_lh_version
2213 #pragma redefine_extname        LONG_it sunw_LONG_it
2214 #pragma redefine_extname        MD2 sunw_MD2
2215 #pragma redefine_extname        MD2_Final sunw_MD2_Final
2216 #pragma redefine_extname        MD2_Init sunw_MD2_Init
2217 #pragma redefine_extname        MD2_options sunw_MD2_options
2218 #pragma redefine_extname        MD2_Update sunw_MD2_Update
2219 #pragma redefine_extname        MD2_version sunw_MD2_version
2220 #pragma redefine_extname        MD4 sunw_MD4
2221 #pragma redefine_extname        md4_block_data_order sunw_md4_block_data_order
2222 #pragma redefine_extname        MD4_Final sunw_MD4_Final
2223 #pragma redefine_extname        MD4_Init sunw_MD4_Init
2224 #pragma redefine_extname        MD4_Transform sunw_MD4_Transform
2225 #pragma redefine_extname        MD4_Update sunw_MD4_Update
2226 #pragma redefine_extname        MD4_version sunw_MD4_version
2227 #pragma redefine_extname        MD5 sunw_MD5
2228 #pragma redefine_extname        md5_block_asm_data_order sunw_md5_block_asm_data
2229 #pragma redefine_extname        MD5_Final sunw_MD5_Final
2230 #pragma redefine_extname        MD5_Init sunw_MD5_Init
2231 #pragma redefine_extname        MD5_Transform sunw_MD5_Transform
2232 #pragma redefine_extname        MD5_Update sunw_MD5_Update
2233 #pragma redefine_extname        MD5_version sunw_MD5_version
2234 #pragma redefine_extname        mod_exp_512 sunw_mod_exp_512
2235 #pragma redefine_extname        name_cmp sunw_name_cmp
2236 #pragma redefine_extname        NAME_CONSTRAINTS_check sunw_NAME_CONSTRAINTS_che
2237 #pragma redefine_extname        NAME_CONSTRAINTS_free sunw_NAME_CONSTRAINTS_free
2238 #pragma redefine_extname        NAME_CONSTRAINTS_it sunw_NAME_CONSTRAINTS_it
2239 #pragma redefine_extname        NAME_CONSTRAINTS_new sunw_NAME_CONSTRAINTS_new
```

```
2240 #pragma redefine_extname         NCONF_default sunw_NCONF_default
2241 #pragma redefine_extname         NCONF_dump_bio sunw_NCONF_dump_bio
2242 #pragma redefine_extname         NCONF_dump_fp sunw_NCONF_dump_fp
2243 #pragma redefine_extname         NCONF_free sunw_NCONF_free
2244 #pragma redefine_extname         NCONF_free_data sunw_NCONF_free_data
2245 #pragma redefine_extname         NCONF_get_number_e sunw_NCONF_get_number_e
2246 #pragma redefine_extname         NCONF_get_section sunw_NCONF_get_section
2247 #pragma redefine_extname         NCONF_get_string sunw_NCONF_get_string
2248 #pragma redefine_extname         NCONF_load sunw_NCONF_load
2249 #pragma redefine_extname         NCONF_load_bio sunw_NCONF_load_bio
2250 #pragma redefine_extname         NCONF_load_fp sunw_NCONF_load_fp
2251 #pragma redefine_extname         NCONF_new sunw_NCONF_new
2252 #pragma redefine_extname         NCONF_WIN32 sunw_NCONF_WIN32
2253 #pragma redefine_extname         NETSCAPE_CERT_SEQUENCE_free sunw_NETSCAPE_CERT_S
2254 #pragma redefine_extname         NETSCAPE_CERT_SEQUENCE_it sunw_NETSCAPE_CERT_SEQ
2255 #pragma redefine_extname         NETSCAPE_CERT_SEQUENCE_new sunw_NETSCAPE_CERT_SE
2256 #pragma redefine_extname         NETSCAPE_ENCRYPTED_PKEY_free sunw_NETSCAPE_ENCRY
2257 #pragma redefine_extname         NETSCAPE_ENCRYPTED_PKEY_it sunw_NETSCAPE_ENCRYPT
2258 #pragma redefine_extname         NETSCAPE_ENCRYPTED_PKEY_new sunw_NETSCAPE_ENCRYP
2259 #pragma redefine_extname         NETSCAPE_PKEY_free sunw_NETSCAPE_PKEY_free
2260 #pragma redefine_extname         NETSCAPE_PKEY_it sunw_NETSCAPE_PKEY_it
2261 #pragma redefine_extname         NETSCAPE_PKEY_new sunw_NETSCAPE_PKEY_new
2262 #pragma redefine_extname         NETSCAPE_SPKAC_free sunw_NETSCAPE_SPKAC_free
2263 #pragma redefine_extname         NETSCAPE_SPKAC_it sunw_NETSCAPE_SPKAC_it
2264 #pragma redefine_extname         NETSCAPE_SPKAC_new sunw_NETSCAPE_SPKAC_new
2265 #pragma redefine_extname         NETSCAPE_SPKI_b64_decode sunw_NETSCAPE_SPKI_b64_
2266 #pragma redefine_extname         NETSCAPE_SPKI_b64_encode sunw_NETSCAPE_SPKI_b64_
2267 #pragma redefine_extname         NETSCAPE_SPKI_free sunw_NETSCAPE_SPKI_free
2268 #pragma redefine_extname         NETSCAPE_SPKI_get_pubkey sunw_NETSCAPE_SPKI_get_
2269 #pragma redefine_extname         NETSCAPE_SPKI_it sunw_NETSCAPE_SPKI_it
2270 #pragma redefine_extname         NETSCAPE_SPKI_new sunw_NETSCAPE_SPKI_new
2271 #pragma redefine_extname         NETSCAPE_SPKI_print sunw_NETSCAPE_SPKI_print
2272 #pragma redefine_extname         NETSCAPE_SPKI_set_pubkey sunw_NETSCAPE_SPKI_set_
2273 #pragma redefine_extname         NETSCAPE_SPKI_sign sunw_NETSCAPE_SPKI_sign
2274 #pragma redefine_extname         NETSCAPE_SPKI_verify sunw_NETSCAPE_SPKI_verify
2275 #pragma redefine_extname         NETSCAPE_X509_free sunw_NETSCAPE_X509_free
2276 #pragma redefine_extname         NETSCAPE_X509_it sunw_NETSCAPE_X509_it
2277 #pragma redefine_extname         NETSCAPE_X509_new sunw_NETSCAPE_X509_new
2278 #pragma redefine_extname         NOTICEREF_free sunw_NOTICEREF_free
2279 #pragma redefine_extname         NOTICEREF_it sunw_NOTICEREF_it
2280 #pragma redefine_extname         NOTICEREF_new sunw_NOTICEREF_new
2281 #pragma redefine_extname         OBJ_add_object sunw_OBJ_add_object
2282 #pragma redefine_extname         OBJ_add_sigid sunw_OBJ_add_sigid
2283 #pragma redefine_extname         OBJ_bsearch_ sunw_OBJ_bsearch_
2284 #pragma redefine_extname         OBJ_bsearch_ex_ sunw_OBJ_bsearch_ex_
2285 #pragma redefine_extname         OBJ_cleanup sunw_OBJ_cleanup
2286 #pragma redefine_extname         OBJ_cmp sunw_OBJ_cmp
2287 #pragma redefine_extname         OBJ_create sunw_OBJ_create
2288 #pragma redefine_extname         OBJ_create_objects sunw_OBJ_create_objects
2289 #pragma redefine_extname         OBJ_dup sunw_OBJ_dup
2290 #pragma redefine_extname         OBJ_find_sigid_algs sunw_OBJ_find_sigid_algs
2291 #pragma redefine_extname         OBJ_find_sigid_by_algs sunw_OBJ_find_sigid_by_al
2292 #pragma redefine_extname         OBJ_ln2nid sunw_OBJ_ln2nid
2293 #pragma redefine_extname         OBJ_NAME_add sunw_OBJ_NAME_add
2294 #pragma redefine_extname         OBJ_NAME_cleanup sunw_OBJ_NAME_cleanup
2295 #pragma redefine_extname         OBJ_NAME_do_all sunw_OBJ_NAME_do_all
2296 #pragma redefine_extname         OBJ_NAME_do_all_sorted sunw_OBJ_NAME_do_all_sort
2297 #pragma redefine_extname         OBJ_NAME_get sunw_OBJ_NAME_get
2298 #pragma redefine_extname         OBJ_NAME_init sunw_OBJ_NAME_init
2299 #pragma redefine_extname         OBJ_NAME_new_index sunw_OBJ_NAME_new_index
2300 #pragma redefine_extname         OBJ_NAME_remove sunw_OBJ_NAME_remove
2301 #pragma redefine_extname         OBJ_new_nid sunw_OBJ_new_nid
2302 #pragma redefine_extname         OBJ_nid2ln sunw_OBJ_nid2ln
2303 #pragma redefine_extname         OBJ_nid2obj sunw_OBJ_nid2obj
2304 #pragma redefine_extname         OBJ_nid2sn sunw_OBJ_nid2sn
2305 #pragma redefine_extname         OBJ_obj2nid sunw_OBJ_obj2nid
```



```
2306 #pragma redefine_extname         OBJ_obj2txt sunw_OBJ_obj2txt
2307 #pragma redefine_extname         OBJ_sigid_free sunw_OBJ_sigid_free
2308 #pragma redefine_extname         OBJ_sn2nid sunw_OBJ_sn2nid
2309 #pragma redefine_extname         OBJ_txt2nid sunw_OBJ_txt2nid
2310 #pragma redefine_extname         OBJ_txt2obj sunw_OBJ_txt2obj
2311 #pragma redefine_extname         OCSP_accept_responses_new sunw_OCSP_accept_respo
2312 #pragma redefine_extname         OCSP_archive_cutoff_new sunw_OCSP_archive_cutoff
2313 #pragma redefine_extname         OCSP_basic_add1_cert sunw_OCSP_basic_add1_cert
2314 #pragma redefine_extname         OCSP_basic_add1_nonce sunw_OCSP_basic_add1_nonce
2315 #pragma redefine_extname         OCSP_basic_add1_status sunw_OCSP_basic_add1_stat
2316 #pragma redefine_extname         OCSP_basic_sign sunw_OCSP_basic_sign
2317 #pragma redefine_extname         OCSP_basic_verify sunw_OCSP_basic_verify
2318 #pragma redefine_extname         OCSP_BASICRESP_add_ext sunw_OCSP_BASICRESP_add_e
2319 #pragma redefine_extname         OCSP_BASICRESP_add1_ext_i2d sunw_OCSP_BASICRESP_
2320 #pragma redefine_extname         OCSP_BASICRESP_delete_ext sunw_OCSP_BASICRESP_de
2321 #pragma redefine_extname         OCSP_BASICRESP_free sunw_OCSP_BASICRESP_free
2322 #pragma redefine_extname         OCSP_BASICRESP_get_ext sunw_OCSP_BASICRESP_get_e
2323 #pragma redefine_extname         OCSP_BASICRESP_get_ext_by_critical sunw_OCSP_BAS
2324 #pragma redefine_extname         OCSP_BASICRESP_get_ext_by_NID sunw_OCSP_BASICRES
2325 #pragma redefine_extname         OCSP_BASICRESP_get_ext_by_OBJ sunw_OCSP_BASICRES
2326 #pragma redefine_extname         OCSP_BASICRESP_get_ext_count sunw_OCSP_BASICRESP
2327 #pragma redefine_extname         OCSP_BASICRESP_get1_ext_d2i sunw_OCSP_BASICRESP_
2328 #pragma redefine_extname         OCSP_BASICRESP_it sunw_OCSP_BASICRESP_it
2329 #pragma redefine_extname         OCSP_BASICRESP_new sunw_OCSP_BASICRESP_new
2330 #pragma redefine_extname         OCSP_cert_id_new sunw_OCSP_cert_id_new
2331 #pragma redefine_extname         OCSP_cert_status_str sunw_OCSP_cert_status_str
2332 #pragma redefine_extname         OCSP_cert_to_id sunw_OCSP_cert_to_id
2333 #pragma redefine_extname         OCSP_CERTID_dup sunw_OCSP_CERTID_dup
2334 #pragma redefine_extname         OCSP_CERTID_free sunw_OCSP_CERTID_free
2335 #pragma redefine_extname         OCSP_CERTID_it sunw_OCSP_CERTID_it
2336 #pragma redefine_extname         OCSP_CERTID_new sunw_OCSP_CERTID_new
2337 #pragma redefine_extname         OCSP_CERTSTATUS_free sunw_OCSP_CERTSTATUS_free
2338 #pragma redefine_extname         OCSP_CERTSTATUS_it sunw_OCSP_CERTSTATUS_it
2339 #pragma redefine_extname         OCSP_CERTSTATUS_new sunw_OCSP_CERTSTATUS_new
2340 #pragma redefine_extname         OCSP_check_nonce sunw_OCSP_check_nonce
2341 #pragma redefine_extname         OCSP_check_validity sunw_OCSP_check_validity
2342 #pragma redefine_extname         OCSP_copy_nonce sunw_OCSP_copy_nonce
2343 #pragma redefine_extname         OCSP_crl_reason_str sunw_OCSP_crl_reason_str
2344 #pragma redefine_extname         OCSP_CRLID_free sunw_OCSP_CRLID_free
2345 #pragma redefine_extname         OCSP_CRLID_it sunw_OCSP_CRLID_it
2346 #pragma redefine_extname         OCSP_crlID_new sunw_OCSP_crlID_new
2347 #pragma redefine_extname         OCSP_CRLID_new sunw_OCSP_CRLID_new
2348 #pragma redefine_extname         OCSP_id_cmp sunw_OCSP_id_cmp
2349 #pragma redefine_extname         OCSP_id_get0_info sunw_OCSP_id_get0_info
2350 #pragma redefine_extname         OCSP_id_issuer_cmp sunw_OCSP_id_issuer_cmp
2351 #pragma redefine_extname         OCSP_ONEREQ_add_ext sunw_OCSP_ONEREQ_add_ext
2352 #pragma redefine_extname         OCSP_ONEREQ_add1_ext_i2d sunw_OCSP_ONEREQ_add1_e
2353 #pragma redefine_extname         OCSP_ONEREQ_delete_ext sunw_OCSP_ONEREQ_delete_e
2354 #pragma redefine_extname         OCSP_ONEREQ_free sunw_OCSP_ONEREQ_free
2355 #pragma redefine_extname         OCSP_ONEREQ_get_ext sunw_OCSP_ONEREQ_get_ext
2356 #pragma redefine_extname         OCSP_ONEREQ_get_ext_by_critical sunw_OCSP_ONEREQ
2357 #pragma redefine_extname         OCSP_ONEREQ_get_ext_by_NID sunw_OCSP_ONEREQ_get_
2358 #pragma redefine_extname         OCSP_ONEREQ_get_ext_by_OBJ sunw_OCSP_ONEREQ_get_
2359 #pragma redefine_extname         OCSP_ONEREQ_get_ext_count sunw_OCSP_ONEREQ_get_e
2360 #pragma redefine_extname         OCSP_onereq_get0_id sunw_OCSP_onereq_get0_id
2361 #pragma redefine_extname         OCSP_ONEREQ_get1_ext_d2i sunw_OCSP_ONEREQ_get1_e
2362 #pragma redefine_extname         OCSP_ONEREQ_it sunw_OCSP_ONEREQ_it
2363 #pragma redefine_extname         OCSP_ONEREQ_new sunw_OCSP_ONEREQ_new
2364 #pragma redefine_extname         OCSP_parse_url sunw_OCSP_parse_url
2365 #pragma redefine_extname         OCSP_REQ_CTX_add1_header sunw_OCSP_REQ_CTX_add1_
2366 #pragma redefine_extname         OCSP_REQ_CTX_free sunw_OCSP_REQ_CTX_free
2367 #pragma redefine_extname         OCSP_REQ_CTX_set1_req sunw_OCSP_REQ_CTX_set1_req
2368 #pragma redefine_extname         OCSP_REQINFO_free sunw_OCSP_REQINFO_free
2369 #pragma redefine_extname         OCSP_REQINFO_it sunw_OCSP_REQINFO_it
2370 #pragma redefine_extname         OCSP_REQINFO_new sunw_OCSP_REQINFO_new
2371 #pragma redefine_extname         OCSP_REQUEST_add_ext sunw_OCSP_REQUEST_add_ext
```

```
2372 #pragma redefine_extname         OCSP_request_add0_id sunw_OCSP_request_add0_id
2373 #pragma redefine_extname         OCSP_request_add1_cert sunw_OCSP_request_add1_ce
2374 #pragma redefine_extname         OCSP_REQUEST_add1_ext_i2d sunw_OCSP_REQUEST_add1
2375 #pragma redefine_extname         OCSP_request_add1_nonce sunw_OCSP_request_add1_n
2376 #pragma redefine_extname         OCSP_REQUEST_delete_ext sunw_OCSP_REQUEST_delete
2377 #pragma redefine_extname         OCSP_REQUEST_free sunw_OCSP_REQUEST_free
2378 #pragma redefine_extname         OCSP_REQUEST_get_ext sunw_OCSP_REQUEST_get_ext
2379 #pragma redefine_extname         OCSP_REQUEST_get_ext_by_critical sunw_OCSP_REQUE
2380 #pragma redefine_extname         OCSP_REQUEST_get_ext_by_NID sunw_OCSP_REQUEST_ge
2381 #pragma redefine_extname         OCSP_REQUEST_get_ext_by_OBJ sunw_OCSP_REQUEST_ge
2382 #pragma redefine_extname         OCSP_REQUEST_get_ext_count sunw_OCSP_REQUEST_get
2383 #pragma redefine_extname         OCSP_REQUEST_get1_ext_d2i sunw_OCSP_REQUEST_get1
2384 #pragma redefine_extname         OCSP_request_is_signed sunw_OCSP_request_is_sign
2385 #pragma redefine_extname         OCSP_REQUEST_it sunw_OCSP_REQUEST_it
2386 #pragma redefine_extname         OCSP_REQUEST_new sunw_OCSP_REQUEST_new
2387 #pragma redefine_extname         OCSP_request_onereq_count sunw_OCSP_request_oner
2388 #pragma redefine_extname         OCSP_request_onereq_get0 sunw_OCSP_request_onere
2389 #pragma redefine_extname         OCSP_REQUEST_print sunw_OCSP_REQUEST_print
2390 #pragma redefine_extname         OCSP_request_set1_name sunw_OCSP_request_set1_na
2391 #pragma redefine_extname         OCSP_request_sign sunw_OCSP_request_sign
2392 #pragma redefine_extname         OCSP_request_verify sunw_OCSP_request_verify
2393 #pragma redefine_extname         OCSP_resp_count sunw_OCSP_resp_count
2394 #pragma redefine_extname         OCSP_resp_find sunw_OCSP_resp_find
2395 #pragma redefine_extname         OCSP_resp_find_status sunw_OCSP_resp_find_status
2396 #pragma redefine_extname         OCSP_resp_get0 sunw_OCSP_resp_get0
2397 #pragma redefine_extname         OCSP_RESPBYTES_free sunw_OCSP_RESPBYTES_free
2398 #pragma redefine_extname         OCSP_RESPBYTES_it sunw_OCSP_RESPBYTES_it
2399 #pragma redefine_extname         OCSP_RESPBYTES_new sunw_OCSP_RESPBYTES_new
2400 #pragma redefine_extname         OCSP_RESPDATA_free sunw_OCSP_RESPDATA_free
2401 #pragma redefine_extname         OCSP_RESPDATA_it sunw_OCSP_RESPDATA_it
2402 #pragma redefine_extname         OCSP_RESPDATA_new sunw_OCSP_RESPDATA_new
2403 #pragma redefine_extname         OCSP_RESPID_free sunw_OCSP_RESPID_free
2404 #pragma redefine_extname         OCSP_RESPID_it sunw_OCSP_RESPID_it
2405 #pragma redefine_extname         OCSP_RESPID_new sunw_OCSP_RESPID_new
2406 #pragma redefine_extname         OCSP_response_create sunw_OCSP_response_create
2407 #pragma redefine_extname         OCSP_RESPONSE_free sunw_OCSP_RESPONSE_free
2408 #pragma redefine_extname         OCSP_response_get1_basic sunw_OCSP_response_get1
2409 #pragma redefine_extname         OCSP_RESPONSE_it sunw_OCSP_RESPONSE_it
2410 #pragma redefine_extname         OCSP_RESPONSE_new sunw_OCSP_RESPONSE_new
2411 #pragma redefine_extname         OCSP_RESPONSE_print sunw_OCSP_RESPONSE_print
2412 #pragma redefine_extname         OCSP_response_status sunw_OCSP_response_status
2413 #pragma redefine_extname         OCSP_response_status_str sunw_OCSP_response_stat
2414 #pragma redefine_extname         OCSP_REVOKEDINFO_free sunw_OCSP_REVOKEDINFO_free
2415 #pragma redefine_extname         OCSP_REVOKEDINFO_it sunw_OCSP_REVOKEDINFO_it
2416 #pragma redefine_extname         OCSP_REVOKEDINFO_new sunw_OCSP_REVOKEDINFO_new
2417 #pragma redefine_extname         OCSP_sendreq_bio sunw_OCSP_sendreq_bio
2418 #pragma redefine_extname         OCSP_sendreq_nbio sunw_OCSP_sendreq_nbio
2419 #pragma redefine_extname         OCSP_sendreq_new sunw_OCSP_sendreq_new
2420 #pragma redefine_extname         OCSP_SERVICELOC_free sunw_OCSP_SERVICELOC_free
2421 #pragma redefine_extname         OCSP_SERVICELOC_it sunw_OCSP_SERVICELOC_it
2422 #pragma redefine_extname         OCSP_SERVICELOC_new sunw_OCSP_SERVICELOC_new
2423 #pragma redefine_extname         OCSP_SIGNATURE_free sunw_OCSP_SIGNATURE_free
2424 #pragma redefine_extname         OCSP_SIGNATURE_it sunw_OCSP_SIGNATURE_it
2425 #pragma redefine_extname         OCSP_SIGNATURE_new sunw_OCSP_SIGNATURE_new
2426 #pragma redefine_extname         OCSP_single_get0_status sunw_OCSP_single_get0_st
2427 #pragma redefine_extname         OCSP_SINGLERESP_add_ext sunw_OCSP_SINGLERESP_add
2428 #pragma redefine_extname         OCSP_SINGLERESP_add1_ext_i2d sunw_OCSP_SINGLERES
2429 #pragma redefine_extname         OCSP_SINGLERESP_delete_ext sunw_OCSP_SINGLERESP_
2430 #pragma redefine_extname         OCSP_SINGLERESP_free sunw_OCSP_SINGLERESP_free
2431 #pragma redefine_extname         OCSP_SINGLERESP_get_ext sunw_OCSP_SINGLERESP_get
2432 #pragma redefine_extname         OCSP_SINGLERESP_get_ext_by_critical sunw_OCSP_SI
2433 #pragma redefine_extname         OCSP_SINGLERESP_get_ext_by_NID sunw_OCSP_SINGLER
2434 #pragma redefine_extname         OCSP_SINGLERESP_get_ext_by_OBJ sunw_OCSP_SINGLER
2435 #pragma redefine_extname         OCSP_SINGLERESP_get_ext_count sunw_OCSP_SINGLERE
2436 #pragma redefine_extname         OCSP_SINGLERESP_get1_ext_d2i sunw_OCSP_SINGLERES
2437 #pragma redefine_extname         OCSP_SINGLERESP_it sunw_OCSP_SINGLERESP_it
```



```
2438 #pragma redefine_extname         OCSP_SINGLERESP_new sunw_OCSP_SINGLERESP_new
2439 #pragma redefine_extname         OCSP_url_svcloc_new sunw_OCSP_url_svcloc_new
2440 #pragma redefine_extname         OPENSSL_add_all_algorithms_conf sunw_OPENSSL_add
2441 #pragma redefine_extname         OPENSSL_add_all_algorithms_noconf sunw_OPENSSL_a
2442 #pragma redefine_extname         OpenSSL_add_all_ciphers sunw_OpenSSL_add_all_cip
2443 #pragma redefine_extname         OpenSSL_add_all_digests sunw_OpenSSL_add_all_dig
2444 #pragma redefine_extname         OPENSSL_asc2uni sunw_OPENSSL_asc2uni
2445 #pragma redefine_extname         OPENSSL_atomic_add sunw_OPENSSL_atomic_add
2446 #pragma redefine_extname         OPENSSL_cleanse sunw_OPENSSL_cleanse
2447 #pragma redefine_extname         OPENSSL_config sunw_OPENSSL_config
2448 #pragma redefine_extname         OPENSSL_cpuid_setup sunw_OPENSSL_cpuid_setup
2449 #pragma redefine_extname         OPENSSL_DIR_end sunw_OPENSSL_DIR_end
2450 #pragma redefine_extname         OPENSSL_DIR_read sunw_OPENSSL_DIR_read
2451 #pragma redefine_extname         OPENSSL_far_spin sunw_OPENSSL_far_spin
2452 #pragma redefine_extname         OPENSSL_gmtime sunw_OPENSSL_gmtime
2453 #pragma redefine_extname         OPENSSL_gmtime_adj sunw_OPENSSL_gmtime_adj
2454 #pragma redefine_extname         OPENSSL_ia32cap_P sunw_OPENSSL_ia32cap_P
2455 #pragma redefine_extname         OPENSSL_ia32_cpuid sunw_OPENSSL_ia32_cpuid
2456 #pragma redefine_extname         OPENSSL_ia32_rdrand sunw_OPENSSL_ia32_rdrand
2457 #pragma redefine_extname         OPENSSL_ia32cap_loc sunw_OPENSSL_ia32cap_loc
2458 #pragma redefine_extname         OPENSSL_indirect_call sunw_OPENSSL_indirect_call
2459 #pragma redefine_extname         OPENSSL_init sunw_OPENSSL_init
2460 #pragma redefine_extname         OPENSSL_instrument_halt sunw_OPENSSL_instrument_
2461 #pragma redefine_extname         OPENSSL_isservice sunw_OPENSSL_isservice
2462 #pragma redefine_extname         OPENSSL_issetugid sunw_OPENSSL_issetugid
2463 #pragma redefine_extname         OPENSSL_load_builtin_modules sunw_OPENSSL_load_b
2464 #pragma redefine_extname         OPENSSL_memcmp sunw_OPENSSL_memcmp
2465 #pragma redefine_extname         OPENSSL_no_config sunw_OPENSSL_no_config
2466 #pragma redefine_extname         OPENSSL_rdtsc sunw_OPENSSL_rdtsc
2467 #pragma redefine_extname         OPENSSL_showfatal sunw_OPENSSL_showfatal
2468 #pragma redefine_extname         OPENSSL_stderr sunw_OPENSSL_stderr
2469 #pragma redefine_extname         OPENSSL_strcasecmp sunw_OPENSSL_strcasecmp
2470 #pragma redefine_extname         OPENSSL_strncasecmp sunw_OPENSSL_strncasecmp
2471 #pragma redefine_extname         OPENSSL_uni2asc sunw_OPENSSL_uni2asc
2472 #pragma redefine_extname         OPENSSL_wipe_cpu sunw_OPENSSL_wipe_cpu
2473 #pragma redefine_extname         OpenSSLDie sunw_OpenSSLDie
2474 #pragma redefine_extname         OSSL_DES_version sunw_OSSL_DES_version
2475 #pragma redefine_extname         OSSL_libdes_version sunw_OSSL_libdes_version
2476 #pragma redefine_extname         OTHERNAME_cmp sunw_OTHERNAME_cmp
2477 #pragma redefine_extname         OTHERNAME_free sunw_OTHERNAME_free
2478 #pragma redefine_extname         OTHERNAME_it sunw_OTHERNAME_it
2479 #pragma redefine_extname         OTHERNAME_new sunw_OTHERNAME_new
2480 #pragma redefine_extname         PBE2PARAM_free sunw_PBE2PARAM_free
2481 #pragma redefine_extname         PBE2PARAM_it sunw_PBE2PARAM_it
2482 #pragma redefine_extname         PBE2PARAM_new sunw_PBE2PARAM_new
2483 #pragma redefine_extname         PBEPARAM_free sunw_PBEPARAM_free
2484 #pragma redefine_extname         PBEPARAM_it sunw_PBEPARAM_it
2485 #pragma redefine_extname         PBEPARAM_new sunw_PBEPARAM_new
2486 #pragma redefine_extname         PBKDF2PARAM_free sunw_PBKDF2PARAM_free
2487 #pragma redefine_extname         PBKDF2PARAM_it sunw_PBKDF2PARAM_it
2488 #pragma redefine_extname         PBKDF2PARAM_new sunw_PBKDF2PARAM_new
2489 #pragma redefine_extname         PEM_ASN1_read sunw_PEM_ASN1_read
2490 #pragma redefine_extname         PEM_ASN1_read_bio sunw_PEM_ASN1_read_bio
2491 #pragma redefine_extname         PEM_ASN1_write sunw_PEM_ASN1_write
2492 #pragma redefine_extname         PEM_ASN1_write_bio sunw_PEM_ASN1_write_bio
2493 #pragma redefine_extname         PEM_bytes_read_bio sunw_PEM_bytes_read_bio
2494 #pragma redefine_extname         pem_check_suffix sunw_pem_check_suffix
2495 #pragma redefine_extname         PEM_def_callback sunw_PEM_def_callback
2496 #pragma redefine_extname         PEM_dek_info sunw_PEM_dek_info
2497 #pragma redefine_extname         PEM_do_header sunw_PEM_do_header
2498 #pragma redefine_extname         PEM_get_EVP_CIPHER_INFO sunw_PEM_get_EVP_CIPHER_
2499 #pragma redefine_extname         PEM_proc_type sunw_PEM_proc_type
2500 #pragma redefine_extname         PEM_read sunw_PEM_read
2501 #pragma redefine_extname         PEM_read_bio sunw_PEM_read_bio
2502 #pragma redefine_extname         PEM_read_bio_CMS sunw_PEM_read_bio_CMS
2503 #pragma redefine_extname         PEM_read_bio_DHparams sunw_PEM_read_bio_DHparams
```

```
2504 #pragma redefine_extname         PEM_read_bio_DSA_PUBKEY sunw_PEM_read_bio_DSA_PU
2505 #pragma redefine_extname         PEM_read_bio_DSAparams sunw_PEM_read_bio_DSApara
2506 #pragma redefine_extname         PEM_read_bio_DSAPrivateKey sunw_PEM_read_bio_DSA
2507 #pragma redefine_extname         PEM_read_bio_NETSCAPE_CERT_SEQUENCE sunw_PEM_rea
2508 #pragma redefine_extname         PEM_read_bio_Parameters sunw_PEM_read_bio_Parame
2509 #pragma redefine_extname         PEM_read_bio_PKCS7 sunw_PEM_read_bio_PKCS7
2510 #pragma redefine_extname         PEM_read_bio_PKCS8 sunw_PEM_read_bio_PKCS8
2511 #pragma redefine_extname         PEM_read_bio_PKCS8_PRIV_KEY_INFO sunw_PEM_read_b
2512 #pragma redefine_extname         PEM_read_bio_PrivateKey sunw_PEM_read_bio_Privat
2513 #pragma redefine_extname         PEM_read_bio_PUBKEY sunw_PEM_read_bio_PUBKEY
2514 #pragma redefine_extname         PEM_read_bio_RSA_PUBKEY sunw_PEM_read_bio_RSA_PU
2515 #pragma redefine_extname         PEM_read_bio_RSAPrivateKey sunw_PEM_read_bio_RSA
2516 #pragma redefine_extname         PEM_read_bio_RSAPublicKey sunw_PEM_read_bio_RSAP
2517 #pragma redefine_extname         PEM_read_bio_X509 sunw_PEM_read_bio_X509
2518 #pragma redefine_extname         PEM_read_bio_X509_AUX sunw_PEM_read_bio_X509_AUX
2519 #pragma redefine_extname         PEM_read_bio_X509_CERT_PAIR sunw_PEM_read_bio_X5
2520 #pragma redefine_extname         PEM_read_bio_X509_CRL sunw_PEM_read_bio_X509_CRL
2521 #pragma redefine_extname         PEM_read_bio_X509_REQ sunw_PEM_read_bio_X509_REQ
2522 #pragma redefine_extname         PEM_read_CMS sunw_PEM_read_CMS
2523 #pragma redefine_extname         PEM_read_DHparams sunw_PEM_read_DHparams
2524 #pragma redefine_extname         PEM_read_DSA_PUBKEY sunw_PEM_read_DSA_PUBKEY
2525 #pragma redefine_extname         PEM_read_DSAparams sunw_PEM_read_DSAparams
2526 #pragma redefine_extname         PEM_read_DSAPrivateKey sunw_PEM_read_DSAPrivateK
2527 #pragma redefine_extname         PEM_read_NETSCAPE_CERT_SEQUENCE sunw_PEM_read_NE
2528 #pragma redefine_extname         PEM_read_PKCS7 sunw_PEM_read_PKCS7
2529 #pragma redefine_extname         PEM_read_PKCS8 sunw_PEM_read_PKCS8
2530 #pragma redefine_extname         PEM_read_PKCS8_PRIV_KEY_INFO sunw_PEM_read_PKCS8
2531 #pragma redefine_extname         PEM_read_PrivateKey sunw_PEM_read_PrivateKey
2532 #pragma redefine_extname         PEM_read_PUBKEY sunw_PEM_read_PUBKEY
2533 #pragma redefine_extname         PEM_read_RSA_PUBKEY sunw_PEM_read_RSA_PUBKEY
2534 #pragma redefine_extname         PEM_read_RSAPrivateKey sunw_PEM_read_RSAPrivateK
2535 #pragma redefine_extname         PEM_read_RSAPublicKey sunw_PEM_read_RSAPublicKey
2536 #pragma redefine_extname         PEM_read_X509 sunw_PEM_read_X509
2537 #pragma redefine_extname         PEM_read_X509_AUX sunw_PEM_read_X509_AUX
2538 #pragma redefine_extname         PEM_read_X509_CERT_PAIR sunw_PEM_read_X509_CERT_
2539 #pragma redefine_extname         PEM_read_X509_CRL sunw_PEM_read_X509_CRL
2540 #pragma redefine_extname         PEM_read_X509_REQ sunw_PEM_read_X509_REQ
2541 #pragma redefine_extname         PEM_SealFinal sunw_PEM_SealFinal
2542 #pragma redefine_extname         PEM_SealInit sunw_PEM_SealInit
2543 #pragma redefine_extname         PEM_SealUpdate sunw_PEM_SealUpdate
2544 #pragma redefine_extname         PEM_SignFinal sunw_PEM_SignFinal
2545 #pragma redefine_extname         PEM_SignInit sunw_PEM_SignInit
2546 #pragma redefine_extname         PEM_SignUpdate sunw_PEM_SignUpdate
2547 #pragma redefine_extname         PEM_version sunw_PEM_version
2548 #pragma redefine_extname         PEM_write sunw_PEM_write
2549 #pragma redefine_extname         PEM_write_bio sunw_PEM_write_bio
2550 #pragma redefine_extname         PEM_write_bio_ASN1_stream sunw_PEM_write_bio_ASN
2551 #pragma redefine_extname         PEM_write_bio_CMS sunw_PEM_write_bio_CMS
2552 #pragma redefine_extname         PEM_write_bio_CMS_stream sunw_PEM_write_bio_CMS_
2553 #pragma redefine_extname         PEM_write_bio_DHparams sunw_PEM_write_bio_DHpara
2554 #pragma redefine_extname         PEM_write_bio_DSA_PUBKEY sunw_PEM_write_bio_DSA_
2555 #pragma redefine_extname         PEM_write_bio_DSAparams sunw_PEM_write_bio_DSApa
2556 #pragma redefine_extname         PEM_write_bio_DSAPrivateKey sunw_PEM_write_bio_D
2557 #pragma redefine_extname         PEM_write_bio_NETSCAPE_CERT_SEQUENCE sunw_PEM_wr
2558 #pragma redefine_extname         PEM_write_bio_Parameters sunw_PEM_write_bio_Para
2559 #pragma redefine_extname         PEM_write_bio_PKCS7 sunw_PEM_write_bio_PKCS7
2560 #pragma redefine_extname         PEM_write_bio_PKCS7_stream sunw_PEM_write_bio_PK
2561 #pragma redefine_extname         PEM_write_bio_PKCS8 sunw_PEM_write_bio_PKCS8
2562 #pragma redefine_extname         PEM_write_bio_PKCS8_PRIV_KEY_INFO sunw_PEM_write
2563 #pragma redefine_extname         PEM_write_bio_PKCS8PrivateKey sunw_PEM_write_bio
2564 #pragma redefine_extname         PEM_write_bio_PKCS8PrivateKey_nid sunw_PEM_write
2565 #pragma redefine_extname         PEM_write_bio_PrivateKey sunw_PEM_write_bio_Priv
2566 #pragma redefine_extname         PEM_write_bio_PUBKEY sunw_PEM_write_bio_PUBKEY
2567 #pragma redefine_extname         PEM_write_bio_RSA_PUBKEY sunw_PEM_write_bio_RSA_
2568 #pragma redefine_extname         PEM_write_bio_RSAPrivateKey sunw_PEM_write_bio_R
2569 #pragma redefine_extname         PEM_write_bio_RSAPublicKey sunw_PEM_write_bio_RS
```



```
2570 #pragma redefine_extname         PEM_write_bio_X509 sunw_PEM_write_bio_X509
2571 #pragma redefine_extname         PEM_write_bio_X509_AUX sunw_PEM_write_bio_X509_A
2572 #pragma redefine_extname         PEM_write_bio_X509_CERT_PAIR sunw_PEM_write_bio_
2573 #pragma redefine_extname         PEM_write_bio_X509_CRL sunw_PEM_write_bio_X509_C
2574 #pragma redefine_extname         PEM_write_bio_X509_REQ sunw_PEM_write_bio_X509_R
2575 #pragma redefine_extname         PEM_write_bio_X509_REQ_NEW sunw_PEM_write_bio_X5
2576 #pragma redefine_extname         PEM_write_CMS sunw_PEM_write_CMS
2577 #pragma redefine_extname         PEM_write_DHparams sunw_PEM_write_DHparams
2578 #pragma redefine_extname         PEM_write_DSA_PUBKEY sunw_PEM_write_DSA_PUBKEY
2579 #pragma redefine_extname         PEM_write_DSAparams sunw_PEM_write_DSAparams
2580 #pragma redefine_extname         PEM_write_DSAPrivateKey sunw_PEM_write_DSAPrivat
2581 #pragma redefine_extname         PEM_write_NETSCAPE_CERT_SEQUENCE sunw_PEM_write_
2582 #pragma redefine_extname         PEM_write_PKCS7 sunw_PEM_write_PKCS7
2583 #pragma redefine_extname         PEM_write_PKCS8 sunw_PEM_write_PKCS8
2584 #pragma redefine_extname         PEM_write_PKCS8_PRIV_KEY_INFO sunw_PEM_write_PKC
2585 #pragma redefine_extname         PEM_write_PKCS8PrivateKey sunw_PEM_write_PKCS8Pr
2586 #pragma redefine_extname         PEM_write_PKCS8PrivateKey_nid sunw_PEM_write_PKC
2587 #pragma redefine_extname         PEM_write_PrivateKey sunw_PEM_write_PrivateKey
2588 #pragma redefine_extname         PEM_write_PUBKEY sunw_PEM_write_PUBKEY
2589 #pragma redefine_extname         PEM_write_RSA_PUBKEY sunw_PEM_write_RSA_PUBKEY
2590 #pragma redefine_extname         PEM_write_RSAPrivateKey sunw_PEM_write_RSAPrivat
2591 #pragma redefine_extname         PEM_write_RSAPublicKey sunw_PEM_write_RSAPublicK
2592 #pragma redefine_extname         PEM_write_X509 sunw_PEM_write_X509
2593 #pragma redefine_extname         PEM_write_X509_AUX sunw_PEM_write_X509_AUX
2594 #pragma redefine_extname         PEM_write_X509_CERT_PAIR sunw_PEM_write_X509_CER
2595 #pragma redefine_extname         PEM_write_X509_CRL sunw_PEM_write_X509_CRL
2596 #pragma redefine_extname         PEM_write_X509_REQ sunw_PEM_write_X509_REQ
2597 #pragma redefine_extname         PEM_write_X509_REQ_NEW sunw_PEM_write_X509_REQ_N
2598 #pragma redefine_extname         PEM_X509_INFO_read sunw_PEM_X509_INFO_read
2599 #pragma redefine_extname         PEM_X509_INFO_read_bio sunw_PEM_X509_INFO_read_b
2600 #pragma redefine_extname         PEM_X509_INFO_write_bio sunw_PEM_X509_INFO_write
2601 #pragma redefine_extname         pitem_free sunw_pitem_free
2602 #pragma redefine_extname         pitem_new sunw_pitem_new
2603 #pragma redefine_extname         PK11_DH sunw_PK11_DH
2604 #pragma redefine_extname         PK11_DSA sunw_PK11_DSA
2605 #pragma redefine_extname         PK11_RSA sunw_PK11_RSA
2606 #pragma redefine_extname         PK11err_add_data sunw_PK11err_add_data
2607 #pragma redefine_extname         pk11_active_add sunw_pk11_active_add
2608 #pragma redefine_extname         pk11_active_delete sunw_pk11_active_delete
2609 #pragma redefine_extname         pk11_active_remove sunw_pk11_active_remove
2610 #pragma redefine_extname         pk11_destroy_dh_key_objects sunw_pk11_destroy_dh
2611 #pragma redefine_extname         pk11_destroy_dh_object sunw_pk11_destroy_dh_obje
2612 #pragma redefine_extname         pk11_destroy_dsa_key_objects sunw_pk11_destroy_d
2613 #pragma redefine_extname         pk11_destroy_dsa_object_priv sunw_pk11_destroy_d
2614 #pragma redefine_extname         pk11_destroy_dsa_object_pub sunw_pk11_destroy_ds
2615 #pragma redefine_extname         pk11_destroy_rsa_key_objects sunw_pk11_destroy_r
2616 #pragma redefine_extname         pk11_destroy_rsa_object_priv sunw_pk11_destroy_r
2617 #pragma redefine_extname         pk11_destroy_rsa_object_pub sunw_pk11_destroy_rs
2618 #pragma redefine_extname         pk11_free_active_list sunw_pk11_free_active_list
2619 #pragma redefine_extname         pk11_get_session sunw_pk11_get_session
2620 #pragma redefine_extname         pk11_load_privkey sunw_pk11_load_privkey
2621 #pragma redefine_extname         pk11_load_pubkey sunw_pk11_load_pubkey
2622 #pragma redefine_extname         pk11_return_session sunw_pk11_return_session
2623 #pragma redefine_extname         PKCS1_MGF1 sunw_PKCS1_MGF1
2624 #pragma redefine_extname         PKCS12_add_cert sunw_PKCS12_add_cert
2625 #pragma redefine_extname         PKCS12_add_CSPName_asc sunw_PKCS12_add_CSPName_a
2626 #pragma redefine_extname         PKCS12_add_friendlyname_asc sunw_PKCS12_add_frie
2627 #pragma redefine_extname         PKCS12_add_friendlyname_uni sunw_PKCS12_add_frie
2628 #pragma redefine_extname         PKCS12_add_key sunw_PKCS12_add_key
2629 #pragma redefine_extname         PKCS12_add_localkeyid sunw_PKCS12_add_localkeyid
2630 #pragma redefine_extname         PKCS12_add_safe sunw_PKCS12_add_safe
2631 #pragma redefine_extname         PKCS12_add_safes sunw_PKCS12_add_safes
2632 #pragma redefine_extname         PKCS12_AUTHSAFES_it sunw_PKCS12_AUTHSAFES_it
2633 #pragma redefine_extname         PKCS12_BAGS_free sunw_PKCS12_BAGS_free
2634 #pragma redefine_extname         PKCS12_BAGS_it sunw_PKCS12_BAGS_it
2635 #pragma redefine_extname         PKCS12_BAGS_new sunw_PKCS12_BAGS_new
```

```
2636 #pragma redefine_extname         PKCS12_certbag2x509 sunw_PKCS12_certbag2x509
2637 #pragma redefine_extname         PKCS12_certbag2x509crl sunw_PKCS12_certbag2x509c
2638 #pragma redefine_extname         PKCS12_create sunw_PKCS12_create
2639 #pragma redefine_extname         PKCS12_decrypt_skey sunw_PKCS12_decrypt_skey
2640 #pragma redefine_extname         PKCS12_free sunw_PKCS12_free
2641 #pragma redefine_extname         PKCS12_gen_mac sunw_PKCS12_gen_mac
2642 #pragma redefine_extname         PKCS12_get_attr_gen sunw_PKCS12_get_attr_gen
2643 #pragma redefine_extname         PKCS12_get_friendlyname sunw_PKCS12_get_friendly
2644 #pragma redefine_extname         PKCS12_init sunw_PKCS12_init
2645 #pragma redefine_extname         PKCS12_it sunw_PKCS12_it
2646 #pragma redefine_extname         PKCS12_item_decrypt_d2i sunw_PKCS12_item_decrypt
2647 #pragma redefine_extname         PKCS12_item_i2d_encrypt sunw_PKCS12_item_i2d_enc
2648 #pragma redefine_extname         PKCS12_item_pack_safebag sunw_PKCS12_item_pack_s
2649 #pragma redefine_extname         PKCS12_key_gen_asc sunw_PKCS12_key_gen_asc
2650 #pragma redefine_extname         PKCS12_key_gen_uni sunw_PKCS12_key_gen_uni
2651 #pragma redefine_extname         PKCS12_MAC_DATA_free sunw_PKCS12_MAC_DATA_free
2652 #pragma redefine_extname         PKCS12_MAC_DATA_it sunw_PKCS12_MAC_DATA_it
2653 #pragma redefine_extname         PKCS12_MAC_DATA_new sunw_PKCS12_MAC_DATA_new
2654 #pragma redefine_extname         PKCS12_MAKE_KEYBAG sunw_PKCS12_MAKE_KEYBAG
2655 #pragma redefine_extname         PKCS12_MAKE_SHKEYBAG sunw_PKCS12_MAKE_SHKEYBAG
2656 #pragma redefine_extname         PKCS12_new sunw_PKCS12_new
2657 #pragma redefine_extname         PKCS12_newpass sunw_PKCS12_newpass
2658 #pragma redefine_extname         PKCS12_pack_authsafes sunw_PKCS12_pack_authsafes
2659 #pragma redefine_extname         PKCS12_pack_p7data sunw_PKCS12_pack_p7data
2660 #pragma redefine_extname         PKCS12_pack_p7encdata sunw_PKCS12_pack_p7encdata
2661 #pragma redefine_extname         PKCS12_parse sunw_PKCS12_parse
2662 #pragma redefine_extname         PKCS12_PBE_add sunw_PKCS12_PBE_add
2663 #pragma redefine_extname         PKCS12_pbe_crypt sunw_PKCS12_pbe_crypt
2664 #pragma redefine_extname         PKCS12_PBE_keyivgen sunw_PKCS12_PBE_keyivgen
2665 #pragma redefine_extname         PKCS12_SAFEBAG_free sunw_PKCS12_SAFEBAG_free
2666 #pragma redefine_extname         PKCS12_SAFEBAG_it sunw_PKCS12_SAFEBAG_it
2667 #pragma redefine_extname         PKCS12_SAFEBAG_new sunw_PKCS12_SAFEBAG_new
2668 #pragma redefine_extname         PKCS12_SAFEBAGS_it sunw_PKCS12_SAFEBAGS_it
2669 #pragma redefine_extname         PKCS12_set_mac sunw_PKCS12_set_mac
2670 #pragma redefine_extname         PKCS12_setup_mac sunw_PKCS12_setup_mac
2671 #pragma redefine_extname         PKCS12_unpack_authsafes sunw_PKCS12_unpack_auths
2672 #pragma redefine_extname         PKCS12_unpack_p7data sunw_PKCS12_unpack_p7data
2673 #pragma redefine_extname         PKCS12_unpack_p7encdata sunw_PKCS12_unpack_p7enc
2674 #pragma redefine_extname         PKCS12_verify_mac sunw_PKCS12_verify_mac
2675 #pragma redefine_extname         PKCS12_x5092certbag sunw_PKCS12_x5092certbag
2676 #pragma redefine_extname         PKCS12_x509crl2certbag sunw_PKCS12_x509crl2certb
2677 #pragma redefine_extname         PKCS5_PBE_add sunw_PKCS5_PBE_add
2678 #pragma redefine_extname         PKCS5_PBE_keyivgen sunw_PKCS5_PBE_keyivgen
2679 #pragma redefine_extname         PKCS5_pbe_set sunw_PKCS5_pbe_set
2680 #pragma redefine_extname         PKCS5_pbe_set0_algor sunw_PKCS5_pbe_set0_algor
2681 #pragma redefine_extname         PKCS5_pbe2_set sunw_PKCS5_pbe2_set
2682 #pragma redefine_extname         PKCS5_pbe2_set_iv sunw_PKCS5_pbe2_set_iv
2683 #pragma redefine_extname         PKCS5_PBKDF2_HMAC sunw_PKCS5_PBKDF2_HMAC
2684 #pragma redefine_extname         PKCS5_PBKDF2_HMAC_SHA1 sunw_PKCS5_PBKDF2_HMAC_SH
2685 #pragma redefine_extname         PKCS5_pbkdf2_set sunw_PKCS5_pbkdf2_set
2686 #pragma redefine_extname         PKCS5_v2_PBE_keyivgen sunw_PKCS5_v2_PBE_keyivgen
2687 #pragma redefine_extname         PKCS5_v2_PBKDF2_keyivgen sunw_PKCS5_v2_PBKDF2_ke
2688 #pragma redefine_extname         PKCS7_add_attrib_content_type sunw_PKCS7_add_att
2689 #pragma redefine_extname         PKCS7_add_attrib_smimecap sunw_PKCS7_add_attrib_
2690 #pragma redefine_extname         PKCS7_add_attribute sunw_PKCS7_add_attribute
2691 #pragma redefine_extname         PKCS7_add_certificate sunw_PKCS7_add_certificate
2692 #pragma redefine_extname         PKCS7_add_crl sunw_PKCS7_add_crl
2693 #pragma redefine_extname         PKCS7_add_recipient sunw_PKCS7_add_recipient
2694 #pragma redefine_extname         PKCS7_add_recipient_info sunw_PKCS7_add_recipien
2695 #pragma redefine_extname         PKCS7_add_signature sunw_PKCS7_add_signature
2696 #pragma redefine_extname         PKCS7_add_signed_attribute sunw_PKCS7_add_signed
2697 #pragma redefine_extname         PKCS7_add_signer sunw_PKCS7_add_signer
2698 #pragma redefine_extname         PKCS7_add0_attrib_signing_time sunw_PKCS7_add0_a
2699 #pragma redefine_extname         PKCS7_add1_attrib_digest sunw_PKCS7_add1_attrib_
2700 #pragma redefine_extname         PKCS7_ATTR_SIGN_it sunw_PKCS7_ATTR_SIGN_it
2701 #pragma redefine_extname         PKCS7_ATTR_VERIFY_it sunw_PKCS7_ATTR_VERIFY_it
```



```
2702 #pragma redefine_extname         PKCS7_cert_from_signer_info sunw_PKCS7_cert_from
2703 #pragma redefine_extname         PKCS7_content_new sunw_PKCS7_content_new
2704 #pragma redefine_extname         PKCS7_ctrl sunw_PKCS7_ctrl
2705 #pragma redefine_extname         PKCS7_dataDecode sunw_PKCS7_dataDecode
2706 #pragma redefine_extname         PKCS7_dataFinal sunw_PKCS7_dataFinal
2707 #pragma redefine_extname         PKCS7_dataInit sunw_PKCS7_dataInit
2708 #pragma redefine_extname         PKCS7_dataVerify sunw_PKCS7_dataVerify
2709 #pragma redefine_extname         PKCS7_decrypt sunw_PKCS7_decrypt
2710 #pragma redefine_extname         PKCS7_DIGEST_free sunw_PKCS7_DIGEST_free
2711 #pragma redefine_extname         PKCS7_digest_from_attributes sunw_PKCS7_digest_f
2712 #pragma redefine_extname         PKCS7_DIGEST_it sunw_PKCS7_DIGEST_it
2713 #pragma redefine_extname         PKCS7_DIGEST_new sunw_PKCS7_DIGEST_new
2714 #pragma redefine_extname         PKCS7_dup sunw_PKCS7_dup
2715 #pragma redefine_extname         PKCS7_ENC_CONTENT_free sunw_PKCS7_ENC_CONTENT_fr
2716 #pragma redefine_extname         PKCS7_ENC_CONTENT_it sunw_PKCS7_ENC_CONTENT_it
2717 #pragma redefine_extname         PKCS7_ENC_CONTENT_new sunw_PKCS7_ENC_CONTENT_new
2718 #pragma redefine_extname         PKCS7_encrypt sunw_PKCS7_encrypt
2719 #pragma redefine_extname         PKCS7_ENCRYPT_free sunw_PKCS7_ENCRYPT_free
2720 #pragma redefine_extname         PKCS7_ENCRYPT_it sunw_PKCS7_ENCRYPT_it
2721 #pragma redefine_extname         PKCS7_ENCRYPT_new sunw_PKCS7_ENCRYPT_new
2722 #pragma redefine_extname         PKCS7_ENVELOPE_free sunw_PKCS7_ENVELOPE_free
2723 #pragma redefine_extname         PKCS7_ENVELOPE_it sunw_PKCS7_ENVELOPE_it
2724 #pragma redefine_extname         PKCS7_ENVELOPE_new sunw_PKCS7_ENVELOPE_new
2725 #pragma redefine_extname         PKCS7_final sunw_PKCS7_final
2726 #pragma redefine_extname         PKCS7_free sunw_PKCS7_free
2727 #pragma redefine_extname         PKCS7_get_attribute sunw_PKCS7_get_attribute
2728 #pragma redefine_extname         PKCS7_get_issuer_and_serial sunw_PKCS7_get_issue
2729 #pragma redefine_extname         PKCS7_get_signed_attribute sunw_PKCS7_get_signed
2730 #pragma redefine_extname         PKCS7_get_signer_info sunw_PKCS7_get_signer_info
2731 #pragma redefine_extname         PKCS7_get_smimecap sunw_PKCS7_get_smimecap
2732 #pragma redefine_extname         PKCS7_get0_signers sunw_PKCS7_get0_signers
2733 #pragma redefine_extname         PKCS7_ISSUER_AND_SERIAL_digest sunw_PKCS7_ISSUER
2734 #pragma redefine_extname         PKCS7_ISSUER_AND_SERIAL_free sunw_PKCS7_ISSUER_A
2735 #pragma redefine_extname         PKCS7_ISSUER_AND_SERIAL_it sunw_PKCS7_ISSUER_AND
2736 #pragma redefine_extname         PKCS7_ISSUER_AND_SERIAL_new sunw_PKCS7_ISSUER_AN
2737 #pragma redefine_extname         PKCS7_it sunw_PKCS7_it
2738 #pragma redefine_extname         PKCS7_new sunw_PKCS7_new
2739 #pragma redefine_extname         PKCS7_print_ctx sunw_PKCS7_print_ctx
2740 #pragma redefine_extname         PKCS7_RECIP_INFO_free sunw_PKCS7_RECIP_INFO_free
2741 #pragma redefine_extname         PKCS7_RECIP_INFO_get0_alg sunw_PKCS7_RECIP_INFO_
2742 #pragma redefine_extname         PKCS7_RECIP_INFO_it sunw_PKCS7_RECIP_INFO_it
2743 #pragma redefine_extname         PKCS7_RECIP_INFO_new sunw_PKCS7_RECIP_INFO_new
2744 #pragma redefine_extname         PKCS7_RECIP_INFO_set sunw_PKCS7_RECIP_INFO_set
2745 #pragma redefine_extname         PKCS7_set_attributes sunw_PKCS7_set_attributes
2746 #pragma redefine_extname         PKCS7_set_cipher sunw_PKCS7_set_cipher
2747 #pragma redefine_extname         PKCS7_set_content sunw_PKCS7_set_content
2748 #pragma redefine_extname         PKCS7_set_digest sunw_PKCS7_set_digest
2749 #pragma redefine_extname         PKCS7_set_signed_attributes sunw_PKCS7_set_signe
2750 #pragma redefine_extname         PKCS7_set_type sunw_PKCS7_set_type
2751 #pragma redefine_extname         PKCS7_set0_type_other sunw_PKCS7_set0_type_other
2752 #pragma redefine_extname         PKCS7_sign sunw_PKCS7_sign
2753 #pragma redefine_extname         PKCS7_sign_add_signer sunw_PKCS7_sign_add_signer
2754 #pragma redefine_extname         PKCS7_SIGN_ENVELOPE_free sunw_PKCS7_SIGN_ENVELOP
2755 #pragma redefine_extname         PKCS7_SIGN_ENVELOPE_it sunw_PKCS7_SIGN_ENVELOPE_
2756 #pragma redefine_extname         PKCS7_SIGN_ENVELOPE_new sunw_PKCS7_SIGN_ENVELOPE
2757 #pragma redefine_extname         PKCS7_signatureVerify sunw_PKCS7_signatureVerify
2758 #pragma redefine_extname         PKCS7_SIGNED_free sunw_PKCS7_SIGNED_free
2759 #pragma redefine_extname         PKCS7_SIGNED_it sunw_PKCS7_SIGNED_it
2760 #pragma redefine_extname         PKCS7_SIGNED_new sunw_PKCS7_SIGNED_new
2761 #pragma redefine_extname         PKCS7_SIGNER_INFO_free sunw_PKCS7_SIGNER_INFO_fr
2762 #pragma redefine_extname         PKCS7_SIGNER_INFO_get0_algs sunw_PKCS7_SIGNER_IN
2763 #pragma redefine_extname         PKCS7_SIGNER_INFO_it sunw_PKCS7_SIGNER_INFO_it
2764 #pragma redefine_extname         PKCS7_SIGNER_INFO_new sunw_PKCS7_SIGNER_INFO_new
2765 #pragma redefine_extname         PKCS7_SIGNER_INFO_set sunw_PKCS7_SIGNER_INFO_set
2766 #pragma redefine_extname         PKCS7_SIGNER_INFO_sign sunw_PKCS7_SIGNER_INFO_si
2767 #pragma redefine_extname         PKCS7_simple_smimecap sunw_PKCS7_simple_smimecap
```

```
2768 #pragma redefine_extname        PKCS7_stream sunw_PKCS7_stream
2769 #pragma redefine_extname        PKCS7_to_TS_TST_INFO sunw_PKCS7_to_TS_TST_INFO
2770 #pragma redefine_extname        PKCS7_verify sunw_PKCS7_verify
2771 #pragma redefine_extname        PKCS8_add_keyusage sunw_PKCS8_add_keyusage
2772 #pragma redefine_extname        PKCS8_decrypt sunw_PKCS8_decrypt
2773 #pragma redefine_extname        PKCS8_encrypt sunw_PKCS8_encrypt
2774 #pragma redefine_extname        PKCS8_pkey_get0 sunw_PKCS8_pkey_get0
2775 #pragma redefine_extname        PKCS8_pkey_set0 sunw_PKCS8_pkey_set0
2776 #pragma redefine_extname        PKCS8_PRIV_KEY_INFO_free sunw_PKCS8_PRIV_KEY_INF
2777 #pragma redefine_extname        PKCS8_PRIV_KEY_INFO_it sunw_PKCS8_PRIV_KEY_INFO_
2778 #pragma redefine_extname        PKCS8_PRIV_KEY_INFO_new sunw_PKCS8_PRIV_KEY_INFO
2779 #pragma redefine_extname        PKCS8_set_broken sunw_PKCS8_set_broken
2780 #pragma redefine_extname        PKEY_USAGE_PERIOD_free sunw_PKEY_USAGE_PERIOD_fr
2781 #pragma redefine_extname        PKEY_USAGE_PERIOD_it sunw_PKEY_USAGE_PERIOD_it
2782 #pragma redefine_extname        PKEY_USAGE_PERIOD_new sunw_PKEY_USAGE_PERIOD_new
2783 #pragma redefine_extname        policy_cache_find_data sunw_policy_cache_find_da
2784 #pragma redefine_extname        policy_cache_free sunw_policy_cache_free
2785 #pragma redefine_extname        policy_cache_set sunw_policy_cache_set
2786 #pragma redefine_extname        policy_cache_set_mapping sunw_policy_cache_set_m
2787 #pragma redefine_extname        POLICY_CONSTRAINTS_free sunw_POLICY_CONSTRAINTS_
2788 #pragma redefine_extname        POLICY_CONSTRAINTS_it sunw_POLICY_CONSTRAINTS_it
2789 #pragma redefine_extname        POLICY_CONSTRAINTS_new sunw_POLICY_CONSTRAINTS_n
2790 #pragma redefine_extname        policy_data_free sunw_policy_data_free
2791 #pragma redefine_extname        policy_data_new sunw_policy_data_new
2792 #pragma redefine_extname        POLICY_MAPPING_free sunw_POLICY_MAPPING_free
2793 #pragma redefine_extname        POLICY_MAPPING_it sunw_POLICY_MAPPING_it
2794 #pragma redefine_extname        POLICY_MAPPING_new sunw_POLICY_MAPPING_new
2795 #pragma redefine_extname        POLICY_MAPPINGS_it sunw_POLICY_MAPPINGS_it
2796 #pragma redefine_extname        policy_node_cmp_new sunw_policy_node_cmp_new
2797 #pragma redefine_extname        policy_node_free sunw_policy_node_free
2798 #pragma redefine_extname        policy_node_match sunw_policy_node_match
2799 #pragma redefine_extname        POLICYINFO_free sunw_POLICYINFO_free
2800 #pragma redefine_extname        POLICYINFO_it sunw_POLICYINFO_it
2801 #pragma redefine_extname        POLICYINFO_new sunw_POLICYINFO_new
2802 #pragma redefine_extname        POLICYQUALINFO_free sunw_POLICYQUALINFO_free
2803 #pragma redefine_extname        POLICYQUALINFO_it sunw_POLICYQUALINFO_it
2804 #pragma redefine_extname        POLICYQUALINFO_new sunw_POLICYQUALINFO_new
2805 #pragma redefine_extname        pqueue_find sunw_pqueue_find
2806 #pragma redefine_extname        pqueue_free sunw_pqueue_free
2807 #pragma redefine_extname        pqueue_insert sunw_pqueue_insert
2808 #pragma redefine_extname        pqueue_iterator sunw_pqueue_iterator
2809 #pragma redefine_extname        pqueue_new sunw_pqueue_new
2810 #pragma redefine_extname        pqueue_next sunw_pqueue_next
2811 #pragma redefine_extname        pqueue_peek sunw_pqueue_peek
2812 #pragma redefine_extname        pqueue_pop sunw_pqueue_pop
2813 #pragma redefine_extname        pqueue_print sunw_pqueue_print
2814 #pragma redefine_extname        pqueue_size sunw_pqueue_size
2815 #pragma redefine_extname        private_AES_set_decrypt_key sunw_private_AES_set
2816 #pragma redefine_extname        private_AES_set_encrypt_key sunw_private_AES_set
2817 #pragma redefine_extname        private_Camellia_set_key sunw_private_Camellia_s
2818 #pragma redefine_extname        private_RC4_set_key sunw_private_RC4_set_key
2819 #pragma redefine_extname        PROXY_CERT_INFO_EXTENSION_free sunw_PROXY_CERT_I
2820 #pragma redefine_extname        PROXY_CERT_INFO_EXTENSION_it sunw_PROXY_CERT_INF
2821 #pragma redefine_extname        PROXY_CERT_INFO_EXTENSION_new sunw_PROXY_CERT_IN
2822 #pragma redefine_extname        PROXY_POLICY_free sunw_PROXY_POLICY_free
2823 #pragma redefine_extname        PROXY_POLICY_it sunw_PROXY_POLICY_it
2824 #pragma redefine_extname        PROXY_POLICY_new sunw_PROXY_POLICY_new
2825 #pragma redefine_extname        RAND_add sunw_RAND_add
2826 #pragma redefine_extname        RAND_bytes sunw_RAND_bytes
2827 #pragma redefine_extname        RAND_cleanup sunw_RAND_cleanup
2828 #pragma redefine_extname        RAND_egd sunw_RAND_egd
2829 #pragma redefine_extname        RAND_egd_bytes sunw_RAND_egd_bytes
2830 #pragma redefine_extname        RAND_file_name sunw_RAND_file_name
2831 #pragma redefine_extname        RAND_get_rand_method sunw_RAND_get_rand_method
2832 #pragma redefine_extname        RAND_load_file sunw_RAND_load_file
2833 #pragma redefine_extname        RAND_poll sunw_RAND_poll
```



```
2834 #pragma redefine_extname        RAND_pseudo_bytes sunw_RAND_pseudo_bytes
2835 #pragma redefine_extname        RAND_query_egd_bytes sunw_RAND_query_egd_bytes
2836 #pragma redefine_extname        RAND_seed sunw_RAND_seed
2837 #pragma redefine_extname        RAND_set_rand_engine sunw_RAND_set_rand_engine
2838 #pragma redefine_extname        RAND_set_rand_method sunw_RAND_set_rand_method
2839 #pragma redefine_extname        RAND_SSLeay sunw_RAND_SSLeay
2840 #pragma redefine_extname        rand_ssleay_meth sunw_rand_ssleay_meth
2841 #pragma redefine_extname        RAND_status sunw_RAND_status
2842 #pragma redefine_extname        RAND_version sunw_RAND_version
2843 #pragma redefine_extname        RAND_write_file sunw_RAND_write_file
2844 #pragma redefine_extname        RC2_cbc_encrypt sunw_RC2_cbc_encrypt
2845 #pragma redefine_extname        RC2_cfb64_encrypt sunw_RC2_cfb64_encrypt
2846 #pragma redefine_extname        RC2_decrypt sunw_RC2_decrypt
2847 #pragma redefine_extname        RC2_ecb_encrypt sunw_RC2_ecb_encrypt
2848 #pragma redefine_extname        RC2_encrypt sunw_RC2_encrypt
2849 #pragma redefine_extname        RC2_ofb64_encrypt sunw_RC2_ofb64_encrypt
2850 #pragma redefine_extname        RC2_set_key sunw_RC2_set_key
2851 #pragma redefine_extname        RC2_version sunw_RC2_version
2852 #pragma redefine_extname        RC4 sunw_RC4
2853 #pragma redefine_extname        rc4_md5_enc sunw_rc4_md5_enc
2854 #pragma redefine_extname        RC4_options sunw_RC4_options
2855 #pragma redefine_extname        RC4_set_key sunw_RC4_set_key
2856 #pragma redefine_extname        RIPEMD160 sunw_RIPEMD160
2857 #pragma redefine_extname        ripemd160_block_asm_data_order sunw_ripemd160_bl
2858 #pragma redefine_extname        ripemd160_block_data_order sunw_ripemd160_block_
2859 #pragma redefine_extname        RIPEMD160_Final sunw_RIPEMD160_Final
2860 #pragma redefine_extname        RIPEMD160_Init sunw_RIPEMD160_Init
2861 #pragma redefine_extname        RIPEMD160_Transform sunw_RIPEMD160_Transform
2862 #pragma redefine_extname        RIPEMD160_Update sunw_RIPEMD160_Update
2863 #pragma redefine_extname        RMD160_version sunw_RMD160_version
2864 #pragma redefine_extname        rsa_asn1_meths sunw_rsa_asn1_meths
2865 #pragma redefine_extname        RSA_blinding_off sunw_RSA_blinding_off
2866 #pragma redefine_extname        RSA_blinding_on sunw_RSA_blinding_on
2867 #pragma redefine_extname        RSA_check_key sunw_RSA_check_key
2868 #pragma redefine_extname        RSA_flags sunw_RSA_flags
2869 #pragma redefine_extname        RSA_free sunw_RSA_free
2870 #pragma redefine_extname        RSA_generate_key sunw_RSA_generate_key
2871 #pragma redefine_extname        RSA_generate_key_ex sunw_RSA_generate_key_ex
2872 #pragma redefine_extname        RSA_get_default_method sunw_RSA_get_default_meth
2873 #pragma redefine_extname        RSA_get_ex_data sunw_RSA_get_ex_data
2874 #pragma redefine_extname        RSA_get_ex_new_index sunw_RSA_get_ex_new_index
2875 #pragma redefine_extname        RSA_get_method sunw_RSA_get_method
2876 #pragma redefine_extname        RSA_memory_lock sunw_RSA_memory_lock
2877 #pragma redefine_extname        RSA_new sunw_RSA_new
2878 #pragma redefine_extname        RSA_new_method sunw_RSA_new_method
2879 #pragma redefine_extname        RSA_null_method sunw_RSA_null_method
2880 #pragma redefine_extname        RSA_padding_add_none sunw_RSA_padding_add_none
2881 #pragma redefine_extname        RSA_padding_add_PKCS1_OAEP sunw_RSA_padding_add_
2882 #pragma redefine_extname        RSA_padding_add_PKCS1_PSS sunw_RSA_padding_add_P
2883 #pragma redefine_extname        RSA_padding_add_PKCS1_PSS_mgf1 sunw_RSA_padding_
2884 #pragma redefine_extname        RSA_padding_add_PKCS1_type_1 sunw_RSA_padding_ad
2885 #pragma redefine_extname        RSA_padding_add_PKCS1_type_2 sunw_RSA_padding_ad
2886 #pragma redefine_extname        RSA_padding_add_SSLv23 sunw_RSA_padding_add_SSLv
2887 #pragma redefine_extname        RSA_padding_add_X931 sunw_RSA_padding_add_X931
2888 #pragma redefine_extname        RSA_padding_check_none sunw_RSA_padding_check_no
2889 #pragma redefine_extname        RSA_padding_check_PKCS1_OAEP sunw_RSA_padding_ch
2890 #pragma redefine_extname        RSA_padding_check_PKCS1_type_1 sunw_RSA_padding_
2891 #pragma redefine_extname        RSA_padding_check_PKCS1_type_2 sunw_RSA_padding_
2892 #pragma redefine_extname        RSA_padding_check_SSLv23 sunw_RSA_padding_check_
2893 #pragma redefine_extname        RSA_padding_check_X931 sunw_RSA_padding_check_X9
2894 #pragma redefine_extname        RSA_PKCS1_SSLeay sunw_RSA_PKCS1_SSLeay
2895 #pragma redefine_extname        rsa_pkey_meth sunw_rsa_pkey_meth
2896 #pragma redefine_extname        RSA_print sunw_RSA_print
2897 #pragma redefine_extname        RSA_print_fp sunw_RSA_print_fp
2898 #pragma redefine_extname        RSA_private_decrypt sunw_RSA_private_decrypt
2899 #pragma redefine_extname        RSA_private_encrypt sunw_RSA_private_encrypt
```

```
2900 #pragma redefine_extname         RSA_PSS_PARAMS_free sunw_RSA_PSS_PARAMS_free
2901 #pragma redefine_extname         RSA_PSS_PARAMS_it sunw_RSA_PSS_PARAMS_it
2902 #pragma redefine_extname         RSA_PSS_PARAMS_new sunw_RSA_PSS_PARAMS_new
2903 #pragma redefine_extname         RSA_public_decrypt sunw_RSA_public_decrypt
2904 #pragma redefine_extname         RSA_public_encrypt sunw_RSA_public_encrypt
2905 #pragma redefine_extname         RSA_set_default_method sunw_RSA_set_default_meth
2906 #pragma redefine_extname         RSA_set_ex_data sunw_RSA_set_ex_data
2907 #pragma redefine_extname         RSA_set_method sunw_RSA_set_method
2908 #pragma redefine_extname         RSA_setup_blinding sunw_RSA_setup_blinding
2909 #pragma redefine_extname         RSA_sign sunw_RSA_sign
2910 #pragma redefine_extname         RSA_sign_ASN1_OCTET_STRING sunw_RSA_sign_ASN1_OC
2911 #pragma redefine_extname         RSA_size sunw_RSA_size
2912 #pragma redefine_extname         RSA_up_ref sunw_RSA_up_ref
2913 #pragma redefine_extname         RSA_verify sunw_RSA_verify
2914 #pragma redefine_extname         RSA_verify_ASN1_OCTET_STRING sunw_RSA_verify_ASN
2915 #pragma redefine_extname         RSA_verify_PKCS1_PSS sunw_RSA_verify_PKCS1_PSS
2916 #pragma redefine_extname         RSA_verify_PKCS1_PSS_mgf1 sunw_RSA_verify_PKCS1_
2917 #pragma redefine_extname         RSA_version sunw_RSA_version
2918 #pragma redefine_extname         RSA_X931_hash_id sunw_RSA_X931_hash_id
2919 #pragma redefine_extname         RSAPrivateKey_dup sunw_RSAPrivateKey_dup
2920 #pragma redefine_extname         RSAPrivateKey_it sunw_RSAPrivateKey_it
2921 #pragma redefine_extname         RSAPublicKey_dup sunw_RSAPublicKey_dup
2922 #pragma redefine_extname         RSAPublicKey_it sunw_RSAPublicKey_it
2923 #pragma redefine_extname         s2i_ASN1_INTEGER sunw_s2i_ASN1_INTEGER
2924 #pragma redefine_extname         s2i_ASN1_OCTET_STRING sunw_s2i_ASN1_OCTET_STRING
2925 #pragma redefine_extname         SHA sunw_SHA
2926 #pragma redefine_extname         SHA_Final sunw_SHA_Final
2927 #pragma redefine_extname         SHA_Init sunw_SHA_Init
2928 #pragma redefine_extname         SHA_Transform sunw_SHA_Transform
2929 #pragma redefine_extname         SHA_Update sunw_SHA_Update
2930 #pragma redefine_extname         SHA_version sunw_SHA_version
2931 #pragma redefine_extname         SHA1 sunw_SHA1
2932 #pragma redefine_extname         sha1_block_data_order sunw_sha1_block_data_order
2933 #pragma redefine_extname         SHA1_Final sunw_SHA1_Final
2934 #pragma redefine_extname         SHA1_Init sunw_SHA1_Init
2935 #pragma redefine_extname         SHA1_Transform sunw_SHA1_Transform
2936 #pragma redefine_extname         SHA1_Update sunw_SHA1_Update
2937 #pragma redefine_extname         SHA1_version sunw_SHA1_version
2938 #pragma redefine_extname         SHA224 sunw_SHA224
2939 #pragma redefine_extname         SHA224_Final sunw_SHA224_Final
2940 #pragma redefine_extname         SHA224_Init sunw_SHA224_Init
2941 #pragma redefine_extname         SHA224_Update sunw_SHA224_Update
2942 #pragma redefine_extname         SHA256 sunw_SHA256
2943 #pragma redefine_extname         sha256_block_data_order sunw_sha256_block_data_o
2944 #pragma redefine_extname         SHA256_Final sunw_SHA256_Final
2945 #pragma redefine_extname         SHA256_Init sunw_SHA256_Init
2946 #pragma redefine_extname         SHA256_Transform sunw_SHA256_Transform
2947 #pragma redefine_extname         SHA256_Update sunw_SHA256_Update
2948 #pragma redefine_extname         SHA256_version sunw_SHA256_version
2949 #pragma redefine_extname         SHA384 sunw_SHA384
2950 #pragma redefine_extname         SHA384_Final sunw_SHA384_Final
2951 #pragma redefine_extname         SHA384_Init sunw_SHA384_Init
2952 #pragma redefine_extname         SHA384_Update sunw_SHA384_Update
2953 #pragma redefine_extname         SHA512 sunw_SHA512
2954 #pragma redefine_extname         sha512_block_data_order sunw_sha512_block_data_o
2955 #pragma redefine_extname         SHA512_Final sunw_SHA512_Final
2956 #pragma redefine_extname         SHA512_Init sunw_SHA512_Init
2957 #pragma redefine_extname         SHA512_Transform sunw_SHA512_Transform
2958 #pragma redefine_extname         SHA512_Update sunw_SHA512_Update
2959 #pragma redefine_extname         SHA512_version sunw_SHA512_version
2960 #pragma redefine_extname         sk_delete sunw_sk_delete
2961 #pragma redefine_extname         sk_delete_ptr sunw_sk_delete_ptr
2962 #pragma redefine_extname         sk_dup sunw_sk_dup
2963 #pragma redefine_extname         sk_find sunw_sk_find
2964 #pragma redefine_extname         sk_find_ex sunw_sk_find_ex
2965 #pragma redefine_extname         sk_free sunw_sk_free
```



```
2966 #pragma redefine_extname         sk_insert sunw_sk_insert
2967 #pragma redefine_extname         sk_is_sorted sunw_sk_is_sorted
2968 #pragma redefine_extname         sk_new sunw_sk_new
2969 #pragma redefine_extname         sk_new_null sunw_sk_new_null
2970 #pragma redefine_extname         sk_num sunw_sk_num
2971 #pragma redefine_extname         sk_pop sunw_sk_pop
2972 #pragma redefine_extname         sk_pop_free sunw_sk_pop_free
2973 #pragma redefine_extname         sk_push sunw_sk_push
2974 #pragma redefine_extname         sk_set sunw_sk_set
2975 #pragma redefine_extname         sk_set_cmp_func sunw_sk_set_cmp_func
2976 #pragma redefine_extname         sk_shift sunw_sk_shift
2977 #pragma redefine_extname         sk_sort sunw_sk_sort
2978 #pragma redefine_extname         sk_unshift sunw_sk_unshift
2979 #pragma redefine_extname         sk_value sunw_sk_value
2980 #pragma redefine_extname         sk_zero sunw_sk_zero
2981 #pragma redefine_extname         SMIME_crlf_copy sunw_SMIME_crlf_copy
2982 #pragma redefine_extname         SMIME_read_ASN1 sunw_SMIME_read_ASN1
2983 #pragma redefine_extname         SMIME_read_CMS sunw_SMIME_read_CMS
2984 #pragma redefine_extname         SMIME_read_PKCS7 sunw_SMIME_read_PKCS7
2985 #pragma redefine_extname         SMIME_text sunw_SMIME_text
2986 #pragma redefine_extname         SMIME_write_ASN1 sunw_SMIME_write_ASN1
2987 #pragma redefine_extname         SMIME_write_CMS sunw_SMIME_write_CMS
2988 #pragma redefine_extname         SMIME_write_PKCS7 sunw_SMIME_write_PKCS7
2989 #pragma redefine_extname         illumos_locking_setup sunw_illumos_locking_setup
2990 #pragma redefine_extname         SRP_Calc_A sunw_SRP_Calc_A
2991 #pragma redefine_extname         SRP_Calc_B sunw_SRP_Calc_B
2992 #pragma redefine_extname         SRP_Calc_client_key sunw_SRP_Calc_client_key
2993 #pragma redefine_extname         SRP_Calc_server_key sunw_SRP_Calc_server_key
2994 #pragma redefine_extname         SRP_Calc_u sunw_SRP_Calc_u
2995 #pragma redefine_extname         SRP_Calc_x sunw_SRP_Calc_x
2996 #pragma redefine_extname         SRP_check_known_gN_param sunw_SRP_check_known_gN
2997 #pragma redefine_extname         SRP_create_verifier sunw_SRP_create_verifier
2998 #pragma redefine_extname         SRP_create_verifier_BN sunw_SRP_create_verifier_
2999 #pragma redefine_extname         SRP_get_default_gN sunw_SRP_get_default_gN
3000 #pragma redefine_extname         SRP_VBASE_free sunw_SRP_VBASE_free
3001 #pragma redefine_extname         SRP_VBASE_get_by_user sunw_SRP_VBASE_get_by_user
3002 #pragma redefine_extname         SRP_VBASE_init sunw_SRP_VBASE_init
3003 #pragma redefine_extname         SRP_VBASE_new sunw_SRP_VBASE_new
3004 #pragma redefine_extname         SRP_Verify_A_mod_N sunw_SRP_Verify_A_mod_N
3005 #pragma redefine_extname         SRP_Verify_B_mod_N sunw_SRP_Verify_B_mod_N
3006 #pragma redefine_extname         SSLeay sunw_SSLeay
3007 #pragma redefine_extname         SSLeay_version sunw_SSLeay_version
3008 #pragma redefine_extname         STACK_version sunw_STACK_version
3009 #pragma redefine_extname         string_to_hex sunw_string_to_hex
3010 #pragma redefine_extname         SXNET_add_id_asc sunw_SXNET_add_id_asc
3011 #pragma redefine_extname         SXNET_add_id_INTEGER sunw_SXNET_add_id_INTEGER
3012 #pragma redefine_extname         SXNET_add_id_ulong sunw_SXNET_add_id_ulong
3013 #pragma redefine_extname         SXNET_free sunw_SXNET_free
3014 #pragma redefine_extname         SXNET_get_id_asc sunw_SXNET_get_id_asc
3015 #pragma redefine_extname         SXNET_get_id_INTEGER sunw_SXNET_get_id_INTEGER
3016 #pragma redefine_extname         SXNET_get_id_ulong sunw_SXNET_get_id_ulong
3017 #pragma redefine_extname         SXNET_it sunw_SXNET_it
3018 #pragma redefine_extname         SXNET_new sunw_SXNET_new
3019 #pragma redefine_extname         SXNETID_free sunw_SXNETID_free
3020 #pragma redefine_extname         SXNETID_it sunw_SXNETID_it
3021 #pragma redefine_extname         SXNETID_new sunw_SXNETID_new
3022 #pragma redefine_extname         tree_find_sk sunw_tree_find_sk
3023 #pragma redefine_extname         TS_ACCURACY_dup sunw_TS_ACCURACY_dup
3024 #pragma redefine_extname         TS_ACCURACY_free sunw_TS_ACCURACY_free
3025 #pragma redefine_extname         TS_ACCURACY_get_micros sunw_TS_ACCURACY_get_micr
3026 #pragma redefine_extname         TS_ACCURACY_get_millis sunw_TS_ACCURACY_get_mill
3027 #pragma redefine_extname         TS_ACCURACY_get_seconds sunw_TS_ACCURACY_get_sec
3028 #pragma redefine_extname         TS_ACCURACY_it sunw_TS_ACCURACY_it
3029 #pragma redefine_extname         TS_ACCURACY_new sunw_TS_ACCURACY_new
3030 #pragma redefine_extname         TS_ACCURACY_set_micros sunw_TS_ACCURACY_set_micr
3031 #pragma redefine_extname         TS_ACCURACY_set_millis sunw_TS_ACCURACY_set_mill
```

```
3032 #pragma redefine_extname        TS_ACCURACY_set_seconds sunw_TS_ACCURACY_set_sec
3033 #pragma redefine_extname        TS_ASN1_INTEGER_print_bio sunw_TS_ASN1_INTEGER_p
3034 #pragma redefine_extname        TS_CONF_get_tsa_section sunw_TS_CONF_get_tsa_sec
3035 #pragma redefine_extname        TS_CONF_load_cert sunw_TS_CONF_load_cert
3036 #pragma redefine_extname        TS_CONF_load_certs sunw_TS_CONF_load_certs
3037 #pragma redefine_extname        TS_CONF_load_key sunw_TS_CONF_load_key
3038 #pragma redefine_extname        TS_CONF_set_accuracy sunw_TS_CONF_set_accuracy
3039 #pragma redefine_extname        TS_CONF_set_certs sunw_TS_CONF_set_certs
3040 #pragma redefine_extname        TS_CONF_set_clock_precision_digits sunw_TS_CONF_
3041 #pragma redefine_extname        TS_CONF_set_crypto_device sunw_TS_CONF_set_crypt
3042 #pragma redefine_extname        TS_CONF_set_def_policy sunw_TS_CONF_set_def_poli
3043 #pragma redefine_extname        TS_CONF_set_default_engine sunw_TS_CONF_set_defa
3044 #pragma redefine_extname        TS_CONF_set_digests sunw_TS_CONF_set_digests
3045 #pragma redefine_extname        TS_CONF_set_ess_cert_id_chain sunw_TS_CONF_set_e
3046 #pragma redefine_extname        TS_CONF_set_ordering sunw_TS_CONF_set_ordering
3047 #pragma redefine_extname        TS_CONF_set_policies sunw_TS_CONF_set_policies
3048 #pragma redefine_extname        TS_CONF_set_serial sunw_TS_CONF_set_serial
3049 #pragma redefine_extname        TS_CONF_set_signer_cert sunw_TS_CONF_set_signer_
3050 #pragma redefine_extname        TS_CONF_set_signer_key sunw_TS_CONF_set_signer_k
3051 #pragma redefine_extname        TS_CONF_set_tsa_name sunw_TS_CONF_set_tsa_name
3052 #pragma redefine_extname        TS_ext_print_bio sunw_TS_ext_print_bio
3053 #pragma redefine_extname        TS_MSG_IMPRINT_dup sunw_TS_MSG_IMPRINT_dup
3054 #pragma redefine_extname        TS_MSG_IMPRINT_free sunw_TS_MSG_IMPRINT_free
3055 #pragma redefine_extname        TS_MSG_IMPRINT_get_algo sunw_TS_MSG_IMPRINT_get_
3056 #pragma redefine_extname        TS_MSG_IMPRINT_get_msg sunw_TS_MSG_IMPRINT_get_m
3057 #pragma redefine_extname        TS_MSG_IMPRINT_it sunw_TS_MSG_IMPRINT_it
3058 #pragma redefine_extname        TS_MSG_IMPRINT_new sunw_TS_MSG_IMPRINT_new
3059 #pragma redefine_extname        TS_MSG_IMPRINT_print_bio sunw_TS_MSG_IMPRINT_pri
3060 #pragma redefine_extname        TS_MSG_IMPRINT_set_algo sunw_TS_MSG_IMPRINT_set_
3061 #pragma redefine_extname        TS_MSG_IMPRINT_set_msg sunw_TS_MSG_IMPRINT_set_m
3062 #pragma redefine_extname        TS_OBJ_print_bio sunw_TS_OBJ_print_bio
3063 #pragma redefine_extname        TS_REQ_add_ext sunw_TS_REQ_add_ext
3064 #pragma redefine_extname        TS_REQ_delete_ext sunw_TS_REQ_delete_ext
3065 #pragma redefine_extname        TS_REQ_dup sunw_TS_REQ_dup
3066 #pragma redefine_extname        TS_REQ_ext_free sunw_TS_REQ_ext_free
3067 #pragma redefine_extname        TS_REQ_free sunw_TS_REQ_free
3068 #pragma redefine_extname        TS_REQ_get_cert_req sunw_TS_REQ_get_cert_req
3069 #pragma redefine_extname        TS_REQ_get_ext sunw_TS_REQ_get_ext
3070 #pragma redefine_extname        TS_REQ_get_ext_by_critical sunw_TS_REQ_get_ext_b
3071 #pragma redefine_extname        TS_REQ_get_ext_by_NID sunw_TS_REQ_get_ext_by_NID
3072 #pragma redefine_extname        TS_REQ_get_ext_by_OBJ sunw_TS_REQ_get_ext_by_OBJ
3073 #pragma redefine_extname        TS_REQ_get_ext_count sunw_TS_REQ_get_ext_count
3074 #pragma redefine_extname        TS_REQ_get_ext_d2i sunw_TS_REQ_get_ext_d2i
3075 #pragma redefine_extname        TS_REQ_get_exts sunw_TS_REQ_get_exts
3076 #pragma redefine_extname        TS_REQ_get_msg_imprint sunw_TS_REQ_get_msg_impri
3077 #pragma redefine_extname        TS_REQ_get_nonce sunw_TS_REQ_get_nonce
3078 #pragma redefine_extname        TS_REQ_get_policy_id sunw_TS_REQ_get_policy_id
3079 #pragma redefine_extname        TS_REQ_get_version sunw_TS_REQ_get_version
3080 #pragma redefine_extname        TS_REQ_it sunw_TS_REQ_it
3081 #pragma redefine_extname        TS_REQ_new sunw_TS_REQ_new
3082 #pragma redefine_extname        TS_REQ_print_bio sunw_TS_REQ_print_bio
3083 #pragma redefine_extname        TS_REQ_set_cert_req sunw_TS_REQ_set_cert_req
3084 #pragma redefine_extname        TS_REQ_set_msg_imprint sunw_TS_REQ_set_msg_impri
3085 #pragma redefine_extname        TS_REQ_set_nonce sunw_TS_REQ_set_nonce
3086 #pragma redefine_extname        TS_REQ_set_policy_id sunw_TS_REQ_set_policy_id
3087 #pragma redefine_extname        TS_REQ_set_version sunw_TS_REQ_set_version
3088 #pragma redefine_extname        TS_REQ_to_TS_VERIFY_CTX sunw_TS_REQ_to_TS_VERIFY
3089 #pragma redefine_extname        TS_RESP_create_response sunw_TS_RESP_create_resp
3090 #pragma redefine_extname        TS_RESP_CTX_add_failure_info sunw_TS_RESP_CTX_ad
3091 #pragma redefine_extname        TS_RESP_CTX_add_flags sunw_TS_RESP_CTX_add_flags
3092 #pragma redefine_extname        TS_RESP_CTX_add_md sunw_TS_RESP_CTX_add_md
3093 #pragma redefine_extname        TS_RESP_CTX_add_policy sunw_TS_RESP_CTX_add_poli
3094 #pragma redefine_extname        TS_RESP_CTX_free sunw_TS_RESP_CTX_free
3095 #pragma redefine_extname        TS_RESP_CTX_get_request sunw_TS_RESP_CTX_get_req
3096 #pragma redefine_extname        TS_RESP_CTX_get_tst_info sunw_TS_RESP_CTX_get_ts
3097 #pragma redefine_extname        TS_RESP_CTX_new sunw_TS_RESP_CTX_new
```



```
3098 #pragma redefine_extname        TS_RESP_CTX_set_accuracy sunw_TS_RESP_CTX_set_ac
3099 #pragma redefine_extname        TS_RESP_CTX_set_certs sunw_TS_RESP_CTX_set_certs
3100 #pragma redefine_extname        TS_RESP_CTX_set_clock_precision_digits sunw_TS_R
3101 #pragma redefine_extname        TS_RESP_CTX_set_def_policy sunw_TS_RESP_CTX_set_
3102 #pragma redefine_extname        TS_RESP_CTX_set_extension_cb sunw_TS_RESP_CTX_se
3103 #pragma redefine_extname        TS_RESP_CTX_set_serial_cb sunw_TS_RESP_CTX_set_s
3104 #pragma redefine_extname        TS_RESP_CTX_set_signer_cert sunw_TS_RESP_CTX_set
3105 #pragma redefine_extname        TS_RESP_CTX_set_signer_key sunw_TS_RESP_CTX_set_
3106 #pragma redefine_extname        TS_RESP_CTX_set_status_info sunw_TS_RESP_CTX_set
3107 #pragma redefine_extname        TS_RESP_CTX_set_status_info_cond sunw_TS_RESP_CT
3108 #pragma redefine_extname        TS_RESP_CTX_set_time_cb sunw_TS_RESP_CTX_set_tim
3109 #pragma redefine_extname        TS_RESP_dup sunw_TS_RESP_dup
3110 #pragma redefine_extname        TS_RESP_free sunw_TS_RESP_free
3111 #pragma redefine_extname        TS_RESP_get_status_info sunw_TS_RESP_get_status_
3112 #pragma redefine_extname        TS_RESP_get_token sunw_TS_RESP_get_token
3113 #pragma redefine_extname        TS_RESP_get_tst_info sunw_TS_RESP_get_tst_info
3114 #pragma redefine_extname        TS_RESP_it sunw_TS_RESP_it
3115 #pragma redefine_extname        TS_RESP_new sunw_TS_RESP_new
3116 #pragma redefine_extname        TS_RESP_print_bio sunw_TS_RESP_print_bio
3117 #pragma redefine_extname        TS_RESP_set_status_info sunw_TS_RESP_set_status_
3118 #pragma redefine_extname        TS_RESP_set_tst_info sunw_TS_RESP_set_tst_info
3119 #pragma redefine_extname        TS_RESP_verify_response sunw_TS_RESP_verify_resp
3120 #pragma redefine_extname        TS_RESP_verify_signature sunw_TS_RESP_verify_sig
3121 #pragma redefine_extname        TS_RESP_verify_token sunw_TS_RESP_verify_token
3122 #pragma redefine_extname        TS_STATUS_INFO_dup sunw_TS_STATUS_INFO_dup
3123 #pragma redefine_extname        TS_STATUS_INFO_free sunw_TS_STATUS_INFO_free
3124 #pragma redefine_extname        TS_STATUS_INFO_it sunw_TS_STATUS_INFO_it
3125 #pragma redefine_extname        TS_STATUS_INFO_new sunw_TS_STATUS_INFO_new
3126 #pragma redefine_extname        TS_STATUS_INFO_print_bio sunw_TS_STATUS_INFO_pri
3127 #pragma redefine_extname        TS_TST_INFO_add_ext sunw_TS_TST_INFO_add_ext
3128 #pragma redefine_extname        TS_TST_INFO_delete_ext sunw_TS_TST_INFO_delete_e
3129 #pragma redefine_extname        TS_TST_INFO_dup sunw_TS_TST_INFO_dup
3130 #pragma redefine_extname        TS_TST_INFO_ext_free sunw_TS_TST_INFO_ext_free
3131 #pragma redefine_extname        TS_TST_INFO_free sunw_TS_TST_INFO_free
3132 #pragma redefine_extname        TS_TST_INFO_get_accuracy sunw_TS_TST_INFO_get_ac
3133 #pragma redefine_extname        TS_TST_INFO_get_ext sunw_TS_TST_INFO_get_ext
3134 #pragma redefine_extname        TS_TST_INFO_get_ext_by_critical sunw_TS_TST_INFO
3135 #pragma redefine_extname        TS_TST_INFO_get_ext_by_NID sunw_TS_TST_INFO_get_
3136 #pragma redefine_extname        TS_TST_INFO_get_ext_by_OBJ sunw_TS_TST_INFO_get_
3137 #pragma redefine_extname        TS_TST_INFO_get_ext_count sunw_TS_TST_INFO_get_e
3138 #pragma redefine_extname        TS_TST_INFO_get_ext_d2i sunw_TS_TST_INFO_get_ext
3139 #pragma redefine_extname        TS_TST_INFO_get_exts sunw_TS_TST_INFO_get_exts
3140 #pragma redefine_extname        TS_TST_INFO_get_msg_imprint sunw_TS_TST_INFO_get
3141 #pragma redefine_extname        TS_TST_INFO_get_nonce sunw_TS_TST_INFO_get_nonce
3142 #pragma redefine_extname        TS_TST_INFO_get_ordering sunw_TS_TST_INFO_get_or
3143 #pragma redefine_extname        TS_TST_INFO_get_policy_id sunw_TS_TST_INFO_get_p
3144 #pragma redefine_extname        TS_TST_INFO_get_serial sunw_TS_TST_INFO_get_seri
3145 #pragma redefine_extname        TS_TST_INFO_get_time sunw_TS_TST_INFO_get_time
3146 #pragma redefine_extname        TS_TST_INFO_get_tsa sunw_TS_TST_INFO_get_tsa
3147 #pragma redefine_extname        TS_TST_INFO_get_version sunw_TS_TST_INFO_get_ver
3148 #pragma redefine_extname        TS_TST_INFO_it sunw_TS_TST_INFO_it
3149 #pragma redefine_extname        TS_TST_INFO_new sunw_TS_TST_INFO_new
3150 #pragma redefine_extname        TS_TST_INFO_print_bio sunw_TS_TST_INFO_print_bio
3151 #pragma redefine_extname        TS_TST_INFO_set_accuracy sunw_TS_TST_INFO_set_ac
3152 #pragma redefine_extname        TS_TST_INFO_set_msg_imprint sunw_TS_TST_INFO_set
3153 #pragma redefine_extname        TS_TST_INFO_set_nonce sunw_TS_TST_INFO_set_nonce
3154 #pragma redefine_extname        TS_TST_INFO_set_ordering sunw_TS_TST_INFO_set_or
3155 #pragma redefine_extname        TS_TST_INFO_set_policy_id sunw_TS_TST_INFO_set_p
3156 #pragma redefine_extname        TS_TST_INFO_set_serial sunw_TS_TST_INFO_set_seri
3157 #pragma redefine_extname        TS_TST_INFO_set_time sunw_TS_TST_INFO_set_time
3158 #pragma redefine_extname        TS_TST_INFO_set_tsa sunw_TS_TST_INFO_set_tsa
3159 #pragma redefine_extname        TS_TST_INFO_set_version sunw_TS_TST_INFO_set_ver
3160 #pragma redefine_extname        TS_VERIFY_CTX_cleanup sunw_TS_VERIFY_CTX_cleanup
3161 #pragma redefine_extname        TS_VERIFY_CTX_free sunw_TS_VERIFY_CTX_free
3162 #pragma redefine_extname        TS_VERIFY_CTX_init sunw_TS_VERIFY_CTX_init
3163 #pragma redefine_extname        TS_VERIFY_CTX_new sunw_TS_VERIFY_CTX_new
```

```
3164 #pragma redefine_extname        TS_X509_ALGOR_print_bio sunw_TS_X509_ALGOR_print
3165 #pragma redefine_extname        TXT_DB_create_index sunw_TXT_DB_create_index
3166 #pragma redefine_extname        TXT_DB_free sunw_TXT_DB_free
3167 #pragma redefine_extname        TXT_DB_get_by_index sunw_TXT_DB_get_by_index
3168 #pragma redefine_extname        TXT_DB_insert sunw_TXT_DB_insert
3169 #pragma redefine_extname        TXT_DB_read sunw_TXT_DB_read
3170 #pragma redefine_extname        TXT_DB_version sunw_TXT_DB_version
3171 #pragma redefine_extname        TXT_DB_write sunw_TXT_DB_write
3172 #pragma redefine_extname        UI_add_error_string sunw_UI_add_error_string
3173 #pragma redefine_extname        UI_add_info_string sunw_UI_add_info_string
3174 #pragma redefine_extname        UI_add_input_boolean sunw_UI_add_input_boolean
3175 #pragma redefine_extname        UI_add_input_string sunw_UI_add_input_string
3176 #pragma redefine_extname        UI_add_user_data sunw_UI_add_user_data
3177 #pragma redefine_extname        UI_add_verify_string sunw_UI_add_verify_string
3178 #pragma redefine_extname        UI_construct_prompt sunw_UI_construct_prompt
3179 #pragma redefine_extname        UI_create_method sunw_UI_create_method
3180 #pragma redefine_extname        UI_ctrl sunw_UI_ctrl
3181 #pragma redefine_extname        UI_destroy_method sunw_UI_destroy_method
3182 #pragma redefine_extname        UI_dup_error_string sunw_UI_dup_error_string
3183 #pragma redefine_extname        UI_dup_info_string sunw_UI_dup_info_string
3184 #pragma redefine_extname        UI_dup_input_boolean sunw_UI_dup_input_boolean
3185 #pragma redefine_extname        UI_dup_input_string sunw_UI_dup_input_string
3186 #pragma redefine_extname        UI_dup_verify_string sunw_UI_dup_verify_string
3187 #pragma redefine_extname        UI_free sunw_UI_free
3188 #pragma redefine_extname        UI_get_default_method sunw_UI_get_default_method
3189 #pragma redefine_extname        UI_get_ex_data sunw_UI_get_ex_data
3190 #pragma redefine_extname        UI_get_ex_new_index sunw_UI_get_ex_new_index
3191 #pragma redefine_extname        UI_get_input_flags sunw_UI_get_input_flags
3192 #pragma redefine_extname        UI_get_method sunw_UI_get_method
3193 #pragma redefine_extname        UI_get_result_maxsize sunw_UI_get_result_maxsize
3194 #pragma redefine_extname        UI_get_result_minsize sunw_UI_get_result_minsize
3195 #pragma redefine_extname        UI_get_string_type sunw_UI_get_string_type
3196 #pragma redefine_extname        UI_get0_action_string sunw_UI_get0_action_string
3197 #pragma redefine_extname        UI_get0_output_string sunw_UI_get0_output_string
3198 #pragma redefine_extname        UI_get0_result sunw_UI_get0_result
3199 #pragma redefine_extname        UI_get0_result_string sunw_UI_get0_result_string
3200 #pragma redefine_extname        UI_get0_test_string sunw_UI_get0_test_string
3201 #pragma redefine_extname        UI_get0_user_data sunw_UI_get0_user_data
3202 #pragma redefine_extname        UI_method_get_closer sunw_UI_method_get_closer
3203 #pragma redefine_extname        UI_method_get_flusher sunw_UI_method_get_flusher
3204 #pragma redefine_extname        UI_method_get_opener sunw_UI_method_get_opener
3205 #pragma redefine_extname        UI_method_get_prompt_constructor sunw_UI_method_
3206 #pragma redefine_extname        UI_method_get_reader sunw_UI_method_get_reader
3207 #pragma redefine_extname        UI_method_get_writer sunw_UI_method_get_writer
3208 #pragma redefine_extname        UI_method_set_closer sunw_UI_method_set_closer
3209 #pragma redefine_extname        UI_method_set_flusher sunw_UI_method_set_flusher
3210 #pragma redefine_extname        UI_method_set_opener sunw_UI_method_set_opener
3211 #pragma redefine_extname        UI_method_set_prompt_constructor sunw_UI_method_
3212 #pragma redefine_extname        UI_method_set_reader sunw_UI_method_set_reader
3213 #pragma redefine_extname        UI_method_set_writer sunw_UI_method_set_writer
3214 #pragma redefine_extname        UI_new sunw_UI_new
3215 #pragma redefine_extname        UI_new_method sunw_UI_new_method
3216 #pragma redefine_extname        UI_OpenSSL sunw_UI_OpenSSL
3217 #pragma redefine_extname        UI_process sunw_UI_process
3218 #pragma redefine_extname        UI_set_default_method sunw_UI_set_default_method
3219 #pragma redefine_extname        UI_set_ex_data sunw_UI_set_ex_data
3220 #pragma redefine_extname        UI_set_method sunw_UI_set_method
3221 #pragma redefine_extname        UI_set_result sunw_UI_set_result
3222 #pragma redefine_extname        UI_UTIL_read_pw sunw_UI_UTIL_read_pw
3223 #pragma redefine_extname        UI_UTIL_read_pw_string sunw_UI_UTIL_read_pw_stri
3224 #pragma redefine_extname        USERNOTICE_free sunw_USERNOTICE_free
3225 #pragma redefine_extname        USERNOTICE_it sunw_USERNOTICE_it
3226 #pragma redefine_extname        USERNOTICE_new sunw_USERNOTICE_new
3227 #pragma redefine_extname        UTF8_getc sunw_UTF8_getc
3228 #pragma redefine_extname        UTF8_putc sunw_UTF8_putc
3229 #pragma redefine_extname        v2i_ASN1_BIT_STRING sunw_v2i_ASN1_BIT_STRING
```



```
3230 #pragma redefine_extname        v2i_GENERAL_NAME sunw_v2i_GENERAL_NAME
3231 #pragma redefine_extname        v2i_GENERAL_NAME_ex sunw_v2i_GENERAL_NAME_ex
3232 #pragma redefine_extname        v2i_GENERAL_NAMES sunw_v2i_GENERAL_NAMES
3233 #pragma redefine_extname        v3_akey_id sunw_v3_akey_id
3234 #pragma redefine_extname        v3_alt sunw_v3_alt
3235 #pragma redefine_extname        v3_bcons sunw_v3_bcons
3236 #pragma redefine_extname        v3_cpols sunw_v3_cpols
3237 #pragma redefine_extname        v3_crl_hold sunw_v3_crl_hold
3238 #pragma redefine_extname        v3_crl_invdate sunw_v3_crl_invdate
3239 #pragma redefine_extname        v3_crl_num sunw_v3_crl_num
3240 #pragma redefine_extname        v3_crl_reason sunw_v3_crl_reason
3241 #pragma redefine_extname        v3_crld sunw_v3_crld
3242 #pragma redefine_extname        v3_delta_crl sunw_v3_delta_crl
3243 #pragma redefine_extname        v3_ext_ku sunw_v3_ext_ku
3244 #pragma redefine_extname        v3_freshest_crl sunw_v3_freshest_crl
3245 #pragma redefine_extname        v3_idp sunw_v3_idp
3246 #pragma redefine_extname        v3_info sunw_v3_info
3247 #pragma redefine_extname        v3_inhibit_anyp sunw_v3_inhibit_anyp
3248 #pragma redefine_extname        v3_key_usage sunw_v3_key_usage
3249 #pragma redefine_extname        v3_name_constraints sunw_v3_name_constraints
3250 #pragma redefine_extname        v3_ns_ia5_list sunw_v3_ns_ia5_list
3251 #pragma redefine_extname        v3_nscert sunw_v3_nscert
3252 #pragma redefine_extname        v3_ocsp_accresp sunw_v3_ocsp_accresp
3253 #pragma redefine_extname        v3_ocsp_acutoff sunw_v3_ocsp_acutoff
3254 #pragma redefine_extname        v3_ocsp_crlid sunw_v3_ocsp_crlid
3255 #pragma redefine_extname        v3_ocsp_nocheck sunw_v3_ocsp_nocheck
3256 #pragma redefine_extname        v3_ocsp_nonce sunw_v3_ocsp_nonce
3257 #pragma redefine_extname        v3_ocsp_serviceloc sunw_v3_ocsp_serviceloc
3258 #pragma redefine_extname        v3_pci sunw_v3_pci
3259 #pragma redefine_extname        v3_pkey_usage_period sunw_v3_pkey_usage_period
3260 #pragma redefine_extname        v3_policy_constraints sunw_v3_policy_constraints
3261 #pragma redefine_extname        v3_policy_mappings sunw_v3_policy_mappings
3262 #pragma redefine_extname        v3_sinfo sunw_v3_sinfo
3263 #pragma redefine_extname        v3_skey_id sunw_v3_skey_id
3264 #pragma redefine_extname        v3_sxnet sunw_v3_sxnet
3265 #pragma redefine_extname        vpaes_cbc_encrypt sunw_vpaes_cbc_encrypt
3266 #pragma redefine_extname        vpaes_decrypt sunw_vpaes_decrypt
3267 #pragma redefine_extname        vpaes_encrypt sunw_vpaes_encrypt
3268 #pragma redefine_extname        vpaes_set_decrypt_key sunw_vpaes_set_decrypt_key
3269 #pragma redefine_extname        vpaes_set_encrypt_key sunw_vpaes_set_encrypt_key
3270 #pragma redefine_extname        X509_add_ext sunw_X509_add_ext
3271 #pragma redefine_extname        X509_add1_ext_i2d sunw_X509_add1_ext_i2d
3272 #pragma redefine_extname        X509_add1_reject_object sunw_X509_add1_reject_ob
3273 #pragma redefine_extname        X509_add1_trust_object sunw_X509_add1_trust_obje
3274 #pragma redefine_extname        X509_ALGOR_dup sunw_X509_ALGOR_dup
3275 #pragma redefine_extname        X509_ALGOR_free sunw_X509_ALGOR_free
3276 #pragma redefine_extname        X509_ALGOR_get0 sunw_X509_ALGOR_get0
3277 #pragma redefine_extname        X509_ALGOR_it sunw_X509_ALGOR_it
3278 #pragma redefine_extname        X509_ALGOR_new sunw_X509_ALGOR_new
3279 #pragma redefine_extname        X509_ALGOR_set_md sunw_X509_ALGOR_set_md
3280 #pragma redefine_extname        X509_ALGOR_set0 sunw_X509_ALGOR_set0
3281 #pragma redefine_extname        X509_ALGORS_it sunw_X509_ALGORS_it
3282 #pragma redefine_extname        X509_alias_get0 sunw_X509_alias_get0
3283 #pragma redefine_extname        X509_alias_set1 sunw_X509_alias_set1
3284 #pragma redefine_extname        X509_ATTRIBUTE_count sunw_X509_ATTRIBUTE_count
3285 #pragma redefine_extname        X509_ATTRIBUTE_create sunw_X509_ATTRIBUTE_create
3286 #pragma redefine_extname        X509_ATTRIBUTE_create_by_NID sunw_X509_ATTRIBUTE
3287 #pragma redefine_extname        X509_ATTRIBUTE_create_by_OBJ sunw_X509_ATTRIBUTE
3288 #pragma redefine_extname        X509_ATTRIBUTE_create_by_txt sunw_X509_ATTRIBUTE
3289 #pragma redefine_extname        X509_ATTRIBUTE_dup sunw_X509_ATTRIBUTE_dup
3290 #pragma redefine_extname        X509_ATTRIBUTE_free sunw_X509_ATTRIBUTE_free
3291 #pragma redefine_extname        X509_ATTRIBUTE_get0_data sunw_X509_ATTRIBUTE_get
3292 #pragma redefine_extname        X509_ATTRIBUTE_get0_object sunw_X509_ATTRIBUTE_g
3293 #pragma redefine_extname        X509_ATTRIBUTE_get0_type sunw_X509_ATTRIBUTE_get
3294 #pragma redefine_extname        X509_ATTRIBUTE_it sunw_X509_ATTRIBUTE_it
3295 #pragma redefine_extname        X509_ATTRIBUTE_new sunw_X509_ATTRIBUTE_new
```

```
3296 #pragma redefine_extname         X509_ATTRIBUTE_SET_it sunw_X509_ATTRIBUTE_SET_it
3297 #pragma redefine_extname         X509_ATTRIBUTE_set1_data sunw_X509_ATTRIBUTE_set
3298 #pragma redefine_extname         X509_ATTRIBUTE_set1_object sunw_X509_ATTRIBUTE_s
3299 #pragma redefine_extname         X509_CERT_AUX_free sunw_X509_CERT_AUX_free
3300 #pragma redefine_extname         X509_CERT_AUX_it sunw_X509_CERT_AUX_it
3301 #pragma redefine_extname         X509_CERT_AUX_new sunw_X509_CERT_AUX_new
3302 #pragma redefine_extname         X509_CERT_AUX_print sunw_X509_CERT_AUX_print
3303 #pragma redefine_extname         X509_CERT_PAIR_free sunw_X509_CERT_PAIR_free
3304 #pragma redefine_extname         X509_CERT_PAIR_it sunw_X509_CERT_PAIR_it
3305 #pragma redefine_extname         X509_CERT_PAIR_new sunw_X509_CERT_PAIR_new
3306 #pragma redefine_extname         X509_certificate_type sunw_X509_certificate_type
3307 #pragma redefine_extname         X509_check_akid sunw_X509_check_akid
3308 #pragma redefine_extname         X509_check_ca sunw_X509_check_ca
3309 #pragma redefine_extname         X509_check_issued sunw_X509_check_issued
3310 #pragma redefine_extname         X509_check_private_key sunw_X509_check_private_k
3311 #pragma redefine_extname         X509_check_purpose sunw_X509_check_purpose
3312 #pragma redefine_extname         X509_check_trust sunw_X509_check_trust
3313 #pragma redefine_extname         X509_CINF_free sunw_X509_CINF_free
3314 #pragma redefine_extname         X509_CINF_it sunw_X509_CINF_it
3315 #pragma redefine_extname         X509_CINF_new sunw_X509_CINF_new
3316 #pragma redefine_extname         X509_cmp sunw_X509_cmp
3317 #pragma redefine_extname         X509_cmp_current_time sunw_X509_cmp_current_time
3318 #pragma redefine_extname         X509_cmp_time sunw_X509_cmp_time
3319 #pragma redefine_extname         X509_CRL_add_ext sunw_X509_CRL_add_ext
3320 #pragma redefine_extname         X509_CRL_add0_revoked sunw_X509_CRL_add0_revoked
3321 #pragma redefine_extname         X509_CRL_add1_ext_i2d sunw_X509_CRL_add1_ext_i2d
3322 #pragma redefine_extname         X509_CRL_cmp sunw_X509_CRL_cmp
3323 #pragma redefine_extname         X509_CRL_delete_ext sunw_X509_CRL_delete_ext
3324 #pragma redefine_extname         X509_CRL_digest sunw_X509_CRL_digest
3325 #pragma redefine_extname         X509_CRL_dup sunw_X509_CRL_dup
3326 #pragma redefine_extname         X509_CRL_free sunw_X509_CRL_free
3327 #pragma redefine_extname         X509_CRL_get_ext sunw_X509_CRL_get_ext
3328 #pragma redefine_extname         X509_CRL_get_ext_by_critical sunw_X509_CRL_get_e
3329 #pragma redefine_extname         X509_CRL_get_ext_by_NID sunw_X509_CRL_get_ext_by
3330 #pragma redefine_extname         X509_CRL_get_ext_by_OBJ sunw_X509_CRL_get_ext_by
3331 #pragma redefine_extname         X509_CRL_get_ext_count sunw_X509_CRL_get_ext_cou
3332 #pragma redefine_extname         X509_CRL_get_ext_d2i sunw_X509_CRL_get_ext_d2i
3333 #pragma redefine_extname         X509_CRL_get_meth_data sunw_X509_CRL_get_meth_da
3334 #pragma redefine_extname         X509_CRL_get0_by_cert sunw_X509_CRL_get0_by_cert
3335 #pragma redefine_extname         X509_CRL_get0_by_serial sunw_X509_CRL_get0_by_se
3336 #pragma redefine_extname         X509_CRL_INFO_free sunw_X509_CRL_INFO_free
3337 #pragma redefine_extname         X509_CRL_INFO_it sunw_X509_CRL_INFO_it
3338 #pragma redefine_extname         X509_CRL_INFO_new sunw_X509_CRL_INFO_new
3339 #pragma redefine_extname         X509_CRL_it sunw_X509_CRL_it
3340 #pragma redefine_extname         X509_CRL_match sunw_X509_CRL_match
3341 #pragma redefine_extname         X509_CRL_METHOD_free sunw_X509_CRL_METHOD_free
3342 #pragma redefine_extname         X509_CRL_METHOD_new sunw_X509_CRL_METHOD_new
3343 #pragma redefine_extname         X509_CRL_new sunw_X509_CRL_new
3344 #pragma redefine_extname         X509_CRL_print sunw_X509_CRL_print
3345 #pragma redefine_extname         X509_CRL_print_fp sunw_X509_CRL_print_fp
3346 #pragma redefine_extname         X509_CRL_set_default_method sunw_X509_CRL_set_de
3347 #pragma redefine_extname         X509_CRL_set_issuer_name sunw_X509_CRL_set_issue
3348 #pragma redefine_extname         X509_CRL_set_lastUpdate sunw_X509_CRL_set_lastUp
3349 #pragma redefine_extname         X509_CRL_set_meth_data sunw_X509_CRL_set_meth_da
3350 #pragma redefine_extname         X509_CRL_set_nextUpdate sunw_X509_CRL_set_nextUp
3351 #pragma redefine_extname         X509_CRL_set_version sunw_X509_CRL_set_version
3352 #pragma redefine_extname         X509_CRL_sign sunw_X509_CRL_sign
3353 #pragma redefine_extname         X509_CRL_sign_ctx sunw_X509_CRL_sign_ctx
3354 #pragma redefine_extname         X509_CRL_sort sunw_X509_CRL_sort
3355 #pragma redefine_extname         X509_CRL_verify sunw_X509_CRL_verify
3356 #pragma redefine_extname         X509_delete_ext sunw_X509_delete_ext
3357 #pragma redefine_extname         X509_digest sunw_X509_digest
3358 #pragma redefine_extname         x509_dir_lookup sunw_x509_dir_lookup
3359 #pragma redefine_extname         X509_dup sunw_X509_dup
3360 #pragma redefine_extname         X509_email_free sunw_X509_email_free
3361 #pragma redefine_extname         X509_EXTENSION_create_by_NID sunw_X509_EXTENSION
```



```
3362 #pragma redefine_extname         X509_EXTENSION_create_by_OBJ sunw_X509_EXTENSION
3363 #pragma redefine_extname         X509_EXTENSION_dup sunw_X509_EXTENSION_dup
3364 #pragma redefine_extname         X509_EXTENSION_free sunw_X509_EXTENSION_free
3365 #pragma redefine_extname         X509_EXTENSION_get_critical sunw_X509_EXTENSION_
3366 #pragma redefine_extname         X509_EXTENSION_get_data sunw_X509_EXTENSION_get_
3367 #pragma redefine_extname         X509_EXTENSION_get_object sunw_X509_EXTENSION_ge
3368 #pragma redefine_extname         X509_EXTENSION_it sunw_X509_EXTENSION_it
3369 #pragma redefine_extname         X509_EXTENSION_new sunw_X509_EXTENSION_new
3370 #pragma redefine_extname         X509_EXTENSION_set_critical sunw_X509_EXTENSION_
3371 #pragma redefine_extname         X509_EXTENSION_set_data sunw_X509_EXTENSION_set_
3372 #pragma redefine_extname         X509_EXTENSION_set_object sunw_X509_EXTENSION_se
3373 #pragma redefine_extname         X509_EXTENSIONS_it sunw_X509_EXTENSIONS_it
3374 #pragma redefine_extname         x509_file_lookup sunw_x509_file_lookup
3375 #pragma redefine_extname         X509_find_by_issuer_and_serial sunw_X509_find_by
3376 #pragma redefine_extname         X509_find_by_subject sunw_X509_find_by_subject
3377 #pragma redefine_extname         X509_free sunw_X509_free
3378 #pragma redefine_extname         X509_get_default_cert_area sunw_X509_get_default
3379 #pragma redefine_extname         X509_get_default_cert_dir sunw_X509_get_default_
3380 #pragma redefine_extname         X509_get_default_cert_dir_env sunw_X509_get_defa
3381 #pragma redefine_extname         X509_get_default_cert_file sunw_X509_get_default
3382 #pragma redefine_extname         X509_get_default_cert_file_env sunw_X509_get_def
3383 #pragma redefine_extname         X509_get_default_private_dir sunw_X509_get_defau
3384 #pragma redefine_extname         X509_get_ex_data sunw_X509_get_ex_data
3385 #pragma redefine_extname         X509_get_ex_new_index sunw_X509_get_ex_new_index
3386 #pragma redefine_extname         X509_get_ext sunw_X509_get_ext
3387 #pragma redefine_extname         X509_get_ext_by_critical sunw_X509_get_ext_by_cr
3388 #pragma redefine_extname         X509_get_ext_by_NID sunw_X509_get_ext_by_NID
3389 #pragma redefine_extname         X509_get_ext_by_OBJ sunw_X509_get_ext_by_OBJ
3390 #pragma redefine_extname         X509_get_ext_count sunw_X509_get_ext_count
3391 #pragma redefine_extname         X509_get_ext_d2i sunw_X509_get_ext_d2i
3392 #pragma redefine_extname         X509_get_issuer_name sunw_X509_get_issuer_name
3393 #pragma redefine_extname         X509_get_pubkey sunw_X509_get_pubkey
3394 #pragma redefine_extname         X509_get_pubkey_parameters sunw_X509_get_pubkey_
3395 #pragma redefine_extname         X509_get_serialNumber sunw_X509_get_serialNumber
3396 #pragma redefine_extname         X509_get_subject_name sunw_X509_get_subject_name
3397 #pragma redefine_extname         X509_get0_pubkey_bitstr sunw_X509_get0_pubkey_bi
3398 #pragma redefine_extname         X509_get1_email sunw_X509_get1_email
3399 #pragma redefine_extname         X509_get1_ocsp sunw_X509_get1_ocsp
3400 #pragma redefine_extname         X509_gmtime_adj sunw_X509_gmtime_adj
3401 #pragma redefine_extname         X509_INFO_free sunw_X509_INFO_free
3402 #pragma redefine_extname         X509_INFO_new sunw_X509_INFO_new
3403 #pragma redefine_extname         X509_issuer_and_serial_cmp sunw_X509_issuer_and_
3404 #pragma redefine_extname         X509_issuer_and_serial_hash sunw_X509_issuer_and
3405 #pragma redefine_extname         X509_issuer_name_cmp sunw_X509_issuer_name_cmp
3406 #pragma redefine_extname         X509_issuer_name_hash sunw_X509_issuer_name_hash
3407 #pragma redefine_extname         X509_issuer_name_hash_old sunw_X509_issuer_name_
3408 #pragma redefine_extname         X509_it sunw_X509_it
3409 #pragma redefine_extname         X509_keyid_get0 sunw_X509_keyid_get0
3410 #pragma redefine_extname         X509_keyid_set1 sunw_X509_keyid_set1
3411 #pragma redefine_extname         X509_load_cert_crl_file sunw_X509_load_cert_crl_
3412 #pragma redefine_extname         X509_load_cert_file sunw_X509_load_cert_file
3413 #pragma redefine_extname         X509_load_crl_file sunw_X509_load_crl_file
3414 #pragma redefine_extname         X509_LOOKUP_by_alias sunw_X509_LOOKUP_by_alias
3415 #pragma redefine_extname         X509_LOOKUP_by_fingerprint sunw_X509_LOOKUP_by_f
3416 #pragma redefine_extname         X509_LOOKUP_by_issuer_serial sunw_X509_LOOKUP_by
3417 #pragma redefine_extname         X509_LOOKUP_by_subject sunw_X509_LOOKUP_by_subje
3418 #pragma redefine_extname         X509_LOOKUP_ctrl sunw_X509_LOOKUP_ctrl
3419 #pragma redefine_extname         X509_LOOKUP_file sunw_X509_LOOKUP_file
3420 #pragma redefine_extname         X509_LOOKUP_free sunw_X509_LOOKUP_free
3421 #pragma redefine_extname         X509_LOOKUP_hash_dir sunw_X509_LOOKUP_hash_dir
3422 #pragma redefine_extname         X509_LOOKUP_init sunw_X509_LOOKUP_init
3423 #pragma redefine_extname         X509_LOOKUP_new sunw_X509_LOOKUP_new
3424 #pragma redefine_extname         X509_LOOKUP_shutdown sunw_X509_LOOKUP_shutdown
3425 #pragma redefine_extname         X509_NAME_add_entry sunw_X509_NAME_add_entry
3426 #pragma redefine_extname         X509_NAME_add_entry_by_NID sunw_X509_NAME_add_en
3427 #pragma redefine_extname         X509_NAME_add_entry_by_OBJ sunw_X509_NAME_add_en
```

```
3428 #pragma redefine_extname         X509_NAME_add_entry_by_txt sunw_X509_NAME_add_en
3429 #pragma redefine_extname         X509_NAME_cmp sunw_X509_NAME_cmp
3430 #pragma redefine_extname         X509_NAME_delete_entry sunw_X509_NAME_delete_ent
3431 #pragma redefine_extname         X509_NAME_digest sunw_X509_NAME_digest
3432 #pragma redefine_extname         X509_NAME_dup sunw_X509_NAME_dup
3433 #pragma redefine_extname         X509_NAME_ENTRIES_it sunw_X509_NAME_ENTRIES_it
3434 #pragma redefine_extname         X509_NAME_entry_count sunw_X509_NAME_entry_count
3435 #pragma redefine_extname         X509_NAME_ENTRY_create_by_NID sunw_X509_NAME_ENT
3436 #pragma redefine_extname         X509_NAME_ENTRY_create_by_OBJ sunw_X509_NAME_ENT
3437 #pragma redefine_extname         X509_NAME_ENTRY_create_by_txt sunw_X509_NAME_ENT
3438 #pragma redefine_extname         X509_NAME_ENTRY_dup sunw_X509_NAME_ENTRY_dup
3439 #pragma redefine_extname         X509_NAME_ENTRY_free sunw_X509_NAME_ENTRY_free
3440 #pragma redefine_extname         X509_NAME_ENTRY_get_data sunw_X509_NAME_ENTRY_ge
3441 #pragma redefine_extname         X509_NAME_ENTRY_get_object sunw_X509_NAME_ENTRY_
3442 #pragma redefine_extname         X509_NAME_ENTRY_it sunw_X509_NAME_ENTRY_it
3443 #pragma redefine_extname         X509_NAME_ENTRY_new sunw_X509_NAME_ENTRY_new
3444 #pragma redefine_extname         X509_NAME_ENTRY_set_data sunw_X509_NAME_ENTRY_se
3445 #pragma redefine_extname         X509_NAME_ENTRY_set_object sunw_X509_NAME_ENTRY_
3446 #pragma redefine_extname         x509_name_ff sunw_x509_name_ff
3447 #pragma redefine_extname         X509_NAME_free sunw_X509_NAME_free
3448 #pragma redefine_extname         X509_NAME_get_entry sunw_X509_NAME_get_entry
3449 #pragma redefine_extname         X509_NAME_get_index_by_NID sunw_X509_NAME_get_in
3450 #pragma redefine_extname         X509_NAME_get_index_by_OBJ sunw_X509_NAME_get_in
3451 #pragma redefine_extname         X509_NAME_get_text_by_NID sunw_X509_NAME_get_tex
3452 #pragma redefine_extname         X509_NAME_get_text_by_OBJ sunw_X509_NAME_get_tex
3453 #pragma redefine_extname         X509_NAME_hash sunw_X509_NAME_hash
3454 #pragma redefine_extname         X509_NAME_hash_old sunw_X509_NAME_hash_old
3455 #pragma redefine_extname         X509_NAME_INTERNAL_it sunw_X509_NAME_INTERNAL_it
3456 #pragma redefine_extname         X509_NAME_it sunw_X509_NAME_it
3457 #pragma redefine_extname         X509_NAME_new sunw_X509_NAME_new
3458 #pragma redefine_extname         X509_NAME_oneline sunw_X509_NAME_oneline
3459 #pragma redefine_extname         X509_NAME_print sunw_X509_NAME_print
3460 #pragma redefine_extname         X509_NAME_print_ex sunw_X509_NAME_print_ex
3461 #pragma redefine_extname         X509_NAME_print_ex_fp sunw_X509_NAME_print_ex_fp
3462 #pragma redefine_extname         X509_NAME_set sunw_X509_NAME_set
3463 #pragma redefine_extname         X509_new sunw_X509_new
3464 #pragma redefine_extname         X509_OBJECT_free_contents sunw_X509_OBJECT_free_
3465 #pragma redefine_extname         X509_OBJECT_idx_by_subject sunw_X509_OBJECT_idx_
3466 #pragma redefine_extname         X509_OBJECT_retrieve_by_subject sunw_X509_OBJECT
3467 #pragma redefine_extname         X509_OBJECT_retrieve_match sunw_X509_OBJECT_retr
3468 #pragma redefine_extname         X509_OBJECT_up_ref_count sunw_X509_OBJECT_up_ref
3469 #pragma redefine_extname         X509_ocspid_print sunw_X509_ocspid_print
3470 #pragma redefine_extname         X509_PKEY_free sunw_X509_PKEY_free
3471 #pragma redefine_extname         X509_PKEY_new sunw_X509_PKEY_new
3472 #pragma redefine_extname         X509_policy_check sunw_X509_policy_check
3473 #pragma redefine_extname         X509_policy_level_get0_node sunw_X509_policy_lev
3474 #pragma redefine_extname         X509_policy_level_node_count sunw_X509_policy_le
3475 #pragma redefine_extname         X509_policy_node_get0_parent sunw_X509_policy_no
3476 #pragma redefine_extname         X509_policy_node_get0_policy sunw_X509_policy_no
3477 #pragma redefine_extname         X509_policy_node_get0_qualifiers sunw_X509_polic
3478 #pragma redefine_extname         X509_POLICY_NODE_print sunw_X509_POLICY_NODE_pri
3479 #pragma redefine_extname         X509_policy_tree_free sunw_X509_policy_tree_free
3480 #pragma redefine_extname         X509_policy_tree_get0_level sunw_X509_policy_tre
3481 #pragma redefine_extname         X509_policy_tree_get0_policies sunw_X509_policy_
3482 #pragma redefine_extname         X509_policy_tree_get0_user_policies sunw_X509_po
3483 #pragma redefine_extname         X509_policy_tree_level_count sunw_X509_policy_tr
3484 #pragma redefine_extname         X509_print sunw_X509_print
3485 #pragma redefine_extname         X509_print_ex sunw_X509_print_ex
3486 #pragma redefine_extname         X509_print_ex_fp sunw_X509_print_ex_fp
3487 #pragma redefine_extname         X509_print_fp sunw_X509_print_fp
3488 #pragma redefine_extname         X509_pubkey_digest sunw_X509_pubkey_digest
3489 #pragma redefine_extname         X509_PUBKEY_free sunw_X509_PUBKEY_free
3490 #pragma redefine_extname         X509_PUBKEY_get sunw_X509_PUBKEY_get
3491 #pragma redefine_extname         X509_PUBKEY_get0_param sunw_X509_PUBKEY_get0_par
3492 #pragma redefine_extname         X509_PUBKEY_it sunw_X509_PUBKEY_it
3493 #pragma redefine_extname         X509_PUBKEY_new sunw_X509_PUBKEY_new
```



```
3494 #pragma redefine_extname         X509_PUBKEY_set sunw_X509_PUBKEY_set
3495 #pragma redefine_extname         X509_PUBKEY_set0_param sunw_X509_PUBKEY_set0_par
3496 #pragma redefine_extname         X509_PURPOSE_add sunw_X509_PURPOSE_add
3497 #pragma redefine_extname         X509_PURPOSE_cleanup sunw_X509_PURPOSE_cleanup
3498 #pragma redefine_extname         X509_PURPOSE_get_by_id sunw_X509_PURPOSE_get_by_
3499 #pragma redefine_extname         X509_PURPOSE_get_by_sname sunw_X509_PURPOSE_get_
3500 #pragma redefine_extname         X509_PURPOSE_get_count sunw_X509_PURPOSE_get_cou
3501 #pragma redefine_extname         X509_PURPOSE_get_id sunw_X509_PURPOSE_get_id
3502 #pragma redefine_extname         X509_PURPOSE_get_trust sunw_X509_PURPOSE_get_tru
3503 #pragma redefine_extname         X509_PURPOSE_get0 sunw_X509_PURPOSE_get0
3504 #pragma redefine_extname         X509_PURPOSE_get0_name sunw_X509_PURPOSE_get0_na
3505 #pragma redefine_extname         X509_PURPOSE_get0_sname sunw_X509_PURPOSE_get0_s
3506 #pragma redefine_extname         X509_PURPOSE_set sunw_X509_PURPOSE_set
3507 #pragma redefine_extname         X509_reject_clear sunw_X509_reject_clear
3508 #pragma redefine_extname         X509_REQ_add_extensions sunw_X509_REQ_add_extens
3509 #pragma redefine_extname         X509_REQ_add_extensions_nid sunw_X509_REQ_add_ex
3510 #pragma redefine_extname         X509_REQ_add1_attr sunw_X509_REQ_add1_attr
3511 #pragma redefine_extname         X509_REQ_add1_attr_by_NID sunw_X509_REQ_add1_att
3512 #pragma redefine_extname         X509_REQ_add1_attr_by_OBJ sunw_X509_REQ_add1_att
3513 #pragma redefine_extname         X509_REQ_add1_attr_by_txt sunw_X509_REQ_add1_att
3514 #pragma redefine_extname         X509_REQ_check_private_key sunw_X509_REQ_check_p
3515 #pragma redefine_extname         X509_REQ_delete_attr sunw_X509_REQ_delete_attr
3516 #pragma redefine_extname         X509_REQ_digest sunw_X509_REQ_digest
3517 #pragma redefine_extname         X509_REQ_dup sunw_X509_REQ_dup
3518 #pragma redefine_extname         X509_REQ_extension_nid sunw_X509_REQ_extension_n
3519 #pragma redefine_extname         X509_REQ_free sunw_X509_REQ_free
3520 #pragma redefine_extname         X509_REQ_get_attr sunw_X509_REQ_get_attr
3521 #pragma redefine_extname         X509_REQ_get_attr_by_NID sunw_X509_REQ_get_attr_
3522 #pragma redefine_extname         X509_REQ_get_attr_by_OBJ sunw_X509_REQ_get_attr_
3523 #pragma redefine_extname         X509_REQ_get_attr_count sunw_X509_REQ_get_attr_c
3524 #pragma redefine_extname         X509_REQ_get_extension_nids sunw_X509_REQ_get_ex
3525 #pragma redefine_extname         X509_REQ_get_extensions sunw_X509_REQ_get_extens
3526 #pragma redefine_extname         X509_REQ_get_pubkey sunw_X509_REQ_get_pubkey
3527 #pragma redefine_extname         X509_REQ_get1_email sunw_X509_REQ_get1_email
3528 #pragma redefine_extname         X509_REQ_INFO_free sunw_X509_REQ_INFO_free
3529 #pragma redefine_extname         X509_REQ_INFO_it sunw_X509_REQ_INFO_it
3530 #pragma redefine_extname         X509_REQ_INFO_new sunw_X509_REQ_INFO_new
3531 #pragma redefine_extname         X509_REQ_it sunw_X509_REQ_it
3532 #pragma redefine_extname         X509_REQ_new sunw_X509_REQ_new
3533 #pragma redefine_extname         X509_REQ_print sunw_X509_REQ_print
3534 #pragma redefine_extname         X509_REQ_print_ex sunw_X509_REQ_print_ex
3535 #pragma redefine_extname         X509_REQ_print_fp sunw_X509_REQ_print_fp
3536 #pragma redefine_extname         X509_REQ_set_extension_nids sunw_X509_REQ_set_ex
3537 #pragma redefine_extname         X509_REQ_set_pubkey sunw_X509_REQ_set_pubkey
3538 #pragma redefine_extname         X509_REQ_set_subject_name sunw_X509_REQ_set_subj
3539 #pragma redefine_extname         X509_REQ_set_version sunw_X509_REQ_set_version
3540 #pragma redefine_extname         X509_REQ_sign sunw_X509_REQ_sign
3541 #pragma redefine_extname         X509_REQ_sign_ctx sunw_X509_REQ_sign_ctx
3542 #pragma redefine_extname         X509_REQ_to_X509 sunw_X509_REQ_to_X509
3543 #pragma redefine_extname         X509_REQ_verify sunw_X509_REQ_verify
3544 #pragma redefine_extname         X509_REVOKED_add_ext sunw_X509_REVOKED_add_ext
3545 #pragma redefine_extname         X509_REVOKED_add1_ext_i2d sunw_X509_REVOKED_add1
3546 #pragma redefine_extname         X509_REVOKED_delete_ext sunw_X509_REVOKED_delete
3547 #pragma redefine_extname         X509_REVOKED_free sunw_X509_REVOKED_free
3548 #pragma redefine_extname         X509_REVOKED_get_ext sunw_X509_REVOKED_get_ext
3549 #pragma redefine_extname         X509_REVOKED_get_ext_by_critical sunw_X509_REVOK
3550 #pragma redefine_extname         X509_REVOKED_get_ext_by_NID sunw_X509_REVOKED_ge
3551 #pragma redefine_extname         X509_REVOKED_get_ext_by_OBJ sunw_X509_REVOKED_ge
3552 #pragma redefine_extname         X509_REVOKED_get_ext_count sunw_X509_REVOKED_get
3553 #pragma redefine_extname         X509_REVOKED_get_ext_d2i sunw_X509_REVOKED_get_e
3554 #pragma redefine_extname         X509_REVOKED_it sunw_X509_REVOKED_it
3555 #pragma redefine_extname         X509_REVOKED_new sunw_X509_REVOKED_new
3556 #pragma redefine_extname         X509_REVOKED_set_revocationDate sunw_X509_REVOKE
3557 #pragma redefine_extname         X509_REVOKED_set_serialNumber sunw_X509_REVOKED_
3558 #pragma redefine_extname         X509_set_ex_data sunw_X509_set_ex_data
3559 #pragma redefine_extname         X509_set_issuer_name sunw_X509_set_issuer_name
```

```
#pragma redefine_extname        X509_set_notAfter sunw_X509_set_notAfter
#pragma redefine_extname        X509_set_notBefore sunw_X509_set_notBefore
#pragma redefine_extname        X509_set_pubkey sunw_X509_set_pubkey
#pragma redefine_extname        X509_set_serialNumber sunw_X509_set_serialNumber
#pragma redefine_extname        X509_set_subject_name sunw_X509_set_subject_name
#pragma redefine_extname        X509_set_version sunw_X509_set_version
#pragma redefine_extname        X509_SIG_free sunw_X509_SIG_free
#pragma redefine_extname        X509_SIG_it sunw_X509_SIG_it
#pragma redefine_extname        X509_SIG_new sunw_X509_SIG_new
#pragma redefine_extname        X509_sign sunw_X509_sign
#pragma redefine_extname        X509_sign_ctx sunw_X509_sign_ctx
#pragma redefine_extname        X509_signature_dump sunw_X509_signature_dump
#pragma redefine_extname        X509_signature_print sunw_X509_signature_print
#pragma redefine_extname        X509_STORE_add_cert sunw_X509_STORE_add_cert
#pragma redefine_extname        X509_STORE_add_crl sunw_X509_STORE_add_crl
#pragma redefine_extname        X509_STORE_add_lookup sunw_X509_STORE_add_lookup
#pragma redefine_extname        X509_STORE_CTX_cleanup sunw_X509_STORE_CTX_clean
#pragma redefine_extname        X509_STORE_CTX_free sunw_X509_STORE_CTX_free
#pragma redefine_extname        X509_STORE_CTX_get_chain sunw_X509_STORE_CTX_get
#pragma redefine_extname        X509_STORE_CTX_get_current_cert sunw_X509_STORE_
#pragma redefine_extname        X509_STORE_CTX_get_error sunw_X509_STORE_CTX_get
#pragma redefine_extname        X509_STORE_CTX_get_error_depth sunw_X509_STORE_C
#pragma redefine_extname        X509_STORE_CTX_get_ex_data sunw_X509_STORE_CTX_g
#pragma redefine_extname        X509_STORE_CTX_get_ex_new_index sunw_X509_STORE_
#pragma redefine_extname        X509_STORE_CTX_get_explicit_policy sunw_X509_STO
#pragma redefine_extname        X509_STORE_CTX_get0_current_crl sunw_X509_STORE_
#pragma redefine_extname        X509_STORE_CTX_get0_current_issuer sunw_X509_STO
#pragma redefine_extname        X509_STORE_CTX_get0_param sunw_X509_STORE_CTX_ge
#pragma redefine_extname        X509_STORE_CTX_get0_parent_ctx sunw_X509_STORE_C
#pragma redefine_extname        X509_STORE_CTX_get0_policy_tree sunw_X509_STORE_
#pragma redefine_extname        X509_STORE_CTX_get1_chain sunw_X509_STORE_CTX_ge
#pragma redefine_extname        X509_STORE_CTX_get1_issuer sunw_X509_STORE_CTX_g
#pragma redefine_extname        X509_STORE_CTX_init sunw_X509_STORE_CTX_init
#pragma redefine_extname        X509_STORE_CTX_new sunw_X509_STORE_CTX_new
#pragma redefine_extname        X509_STORE_CTX_purpose_inherit sunw_X509_STORE_C
#pragma redefine_extname        X509_STORE_CTX_set_cert sunw_X509_STORE_CTX_set_
#pragma redefine_extname        X509_STORE_CTX_set_chain sunw_X509_STORE_CTX_set
#pragma redefine_extname        X509_STORE_CTX_set_default sunw_X509_STORE_CTX_s
#pragma redefine_extname        X509_STORE_CTX_set_depth sunw_X509_STORE_CTX_set
#pragma redefine_extname        X509_STORE_CTX_set_error sunw_X509_STORE_CTX_set
#pragma redefine_extname        X509_STORE_CTX_set_ex_data sunw_X509_STORE_CTX_s
#pragma redefine_extname        X509_STORE_CTX_set_flags sunw_X509_STORE_CTX_set
#pragma redefine_extname        X509_STORE_CTX_set_purpose sunw_X509_STORE_CTX_s
#pragma redefine_extname        X509_STORE_CTX_set_time sunw_X509_STORE_CTX_set_
#pragma redefine_extname        X509_STORE_CTX_set_trust sunw_X509_STORE_CTX_set
#pragma redefine_extname        X509_STORE_CTX_set_verify_cb sunw_X509_STORE_CTX
#pragma redefine_extname        X509_STORE_CTX_set0_crls sunw_X509_STORE_CTX_set
#pragma redefine_extname        X509_STORE_CTX_set0_param sunw_X509_STORE_CTX_se
#pragma redefine_extname        X509_STORE_CTX_trusted_stack sunw_X509_STORE_CTX
#pragma redefine_extname        X509_STORE_free sunw_X509_STORE_free
#pragma redefine_extname        X509_STORE_get_by_subject sunw_X509_STORE_get_by
#pragma redefine_extname        X509_STORE_get1_certs sunw_X509_STORE_get1_certs
#pragma redefine_extname        X509_STORE_get1_crls sunw_X509_STORE_get1_crls
#pragma redefine_extname        X509_STORE_load_locations sunw_X509_STORE_load_l
#pragma redefine_extname        X509_STORE_new sunw_X509_STORE_new
#pragma redefine_extname        X509_STORE_set_default_paths sunw_X509_STORE_set
#pragma redefine_extname        X509_STORE_set_depth sunw_X509_STORE_set_depth
#pragma redefine_extname        X509_STORE_set_flags sunw_X509_STORE_set_flags
#pragma redefine_extname        X509_STORE_set_purpose sunw_X509_STORE_set_purpo
#pragma redefine_extname        X509_STORE_set_trust sunw_X509_STORE_set_trust
#pragma redefine_extname        X509_STORE_set_verify_cb sunw_X509_STORE_set_ver
#pragma redefine_extname        X509_STORE_set1_param sunw_X509_STORE_set1_param
#pragma redefine_extname        X509_subject_name_cmp sunw_X509_subject_name_cmp
#pragma redefine_extname        X509_subject_name_hash sunw_X509_subject_name_ha
#pragma redefine_extname        X509_subject_name_hash_old sunw_X509_subject_nam
#pragma redefine_extname        X509_supported_extension sunw_X509_supported_ext
```



```
#pragma redefine_extname        X509_time_adj sunw_X509_time_adj
#pragma redefine_extname        X509_time_adj_ex sunw_X509_time_adj_ex
#pragma redefine_extname        X509_to_X509_REQ sunw_X509_to_X509_REQ
#pragma redefine_extname        X509_TRUST_add sunw_X509_TRUST_add
#pragma redefine_extname        X509_TRUST_cleanup sunw_X509_TRUST_cleanup
#pragma redefine_extname        X509_trust_clear sunw_X509_trust_clear
#pragma redefine_extname        X509_TRUST_get_by_id sunw_X509_TRUST_get_by_id
#pragma redefine_extname        X509_TRUST_get_count sunw_X509_TRUST_get_count
#pragma redefine_extname        X509_TRUST_get_flags sunw_X509_TRUST_get_flags
#pragma redefine_extname        X509_TRUST_get_trust sunw_X509_TRUST_get_trust
#pragma redefine_extname        X509_TRUST_get0 sunw_X509_TRUST_get0
#pragma redefine_extname        X509_TRUST_get0_name sunw_X509_TRUST_get0_name
#pragma redefine_extname        X509_TRUST_set sunw_X509_TRUST_set
#pragma redefine_extname        X509_TRUST_set_default sunw_X509_TRUST_set_defau
#pragma redefine_extname        X509_VAL_free sunw_X509_VAL_free
#pragma redefine_extname        X509_VAL_it sunw_X509_VAL_it
#pragma redefine_extname        X509_VAL_new sunw_X509_VAL_new
#pragma redefine_extname        X509_verify sunw_X509_verify
#pragma redefine_extname        X509_verify_cert sunw_X509_verify_cert
#pragma redefine_extname        X509_verify_cert_error_string sunw_X509_verify_c
#pragma redefine_extname        X509_VERIFY_PARAM_add0_policy sunw_X509_VERIFY_P
#pragma redefine_extname        X509_VERIFY_PARAM_add0_table sunw_X509_VERIFY_PA
#pragma redefine_extname        X509_VERIFY_PARAM_clear_flags sunw_X509_VERIFY_P
#pragma redefine_extname        X509_VERIFY_PARAM_free sunw_X509_VERIFY_PARAM_fr
#pragma redefine_extname        X509_VERIFY_PARAM_get_depth sunw_X509_VERIFY_PAR
#pragma redefine_extname        X509_VERIFY_PARAM_get_flags sunw_X509_VERIFY_PAR
#pragma redefine_extname        X509_VERIFY_PARAM_inherit sunw_X509_VERIFY_PARAM
#pragma redefine_extname        X509_VERIFY_PARAM_lookup sunw_X509_VERIFY_PARAM_
#pragma redefine_extname        X509_VERIFY_PARAM_new sunw_X509_VERIFY_PARAM_new
#pragma redefine_extname        X509_VERIFY_PARAM_set_depth sunw_X509_VERIFY_PAR
#pragma redefine_extname        X509_VERIFY_PARAM_set_flags sunw_X509_VERIFY_PAR
#pragma redefine_extname        X509_VERIFY_PARAM_set_purpose sunw_X509_VERIFY_P
#pragma redefine_extname        X509_VERIFY_PARAM_set_time sunw_X509_VERIFY_PARA
#pragma redefine_extname        X509_VERIFY_PARAM_set_trust sunw_X509_VERIFY_PAR
#pragma redefine_extname        X509_VERIFY_PARAM_set1 sunw_X509_VERIFY_PARAM_se
#pragma redefine_extname        X509_VERIFY_PARAM_set1_name sunw_X509_VERIFY_PAR
#pragma redefine_extname        X509_VERIFY_PARAM_set1_policies sunw_X509_VERIFY
#pragma redefine_extname        X509_VERIFY_PARAM_table_cleanup sunw_X509_VERIFY
#pragma redefine_extname        X509_version sunw_X509_version
#pragma redefine_extname        X509at_add1_attr sunw_X509at_add1_attr
#pragma redefine_extname        X509at_add1_attr_by_NID sunw_X509at_add1_attr_by
#pragma redefine_extname        X509at_add1_attr_by_OBJ sunw_X509at_add1_attr_by
#pragma redefine_extname        X509at_add1_attr_by_txt sunw_X509at_add1_attr_by
#pragma redefine_extname        X509at_delete_attr sunw_X509at_delete_attr
#pragma redefine_extname        X509at_get_attr sunw_X509at_get_attr
#pragma redefine_extname        X509at_get_attr_by_NID sunw_X509at_get_attr_by_N
#pragma redefine_extname        X509at_get_attr_by_OBJ sunw_X509at_get_attr_by_O
#pragma redefine_extname        X509at_get_attr_count sunw_X509at_get_attr_count
#pragma redefine_extname        X509at_get0_data_by_OBJ sunw_X509at_get0_data_by
#pragma redefine_extname        X509v3_add_ext sunw_X509v3_add_ext
#pragma redefine_extname        X509V3_add_standard_extensions sunw_X509V3_add_s
#pragma redefine_extname        X509V3_add_value sunw_X509V3_add_value
#pragma redefine_extname        X509V3_add_value_bool sunw_X509V3_add_value_bool
#pragma redefine_extname        X509V3_add_value_bool_nf sunw_X509V3_add_value_b
#pragma redefine_extname        X509V3_add_value_int sunw_X509V3_add_value_int
#pragma redefine_extname        X509V3_add_value_uchar sunw_X509V3_add_value_uch
#pragma redefine_extname        X509V3_add1_i2d sunw_X509V3_add1_i2d
#pragma redefine_extname        X509V3_conf_free sunw_X509V3_conf_free
#pragma redefine_extname        X509v3_delete_ext sunw_X509v3_delete_ext
#pragma redefine_extname        X509V3_EXT_add sunw_X509V3_EXT_add
#pragma redefine_extname        X509V3_EXT_add_alias sunw_X509V3_EXT_add_alias
#pragma redefine_extname        X509V3_EXT_add_conf sunw_X509V3_EXT_add_conf
#pragma redefine_extname        X509V3_EXT_add_list sunw_X509V3_EXT_add_list
#pragma redefine_extname        X509V3_EXT_add_nconf sunw_X509V3_EXT_add_nconf
#pragma redefine_extname        X509V3_EXT_add_nconf_sk sunw_X509V3_EXT_add_ncon
#pragma redefine_extname        X509V3_EXT_cleanup sunw_X509V3_EXT_cleanup
```

```
3692 #pragma redefine_extname         X509V3_EXT_conf sunw_X509V3_EXT_conf
3693 #pragma redefine_extname         X509V3_EXT_conf_nid sunw_X509V3_EXT_conf_nid
3694 #pragma redefine_extname         X509V3_EXT_CRL_add_conf sunw_X509V3_EXT_CRL_add_
3695 #pragma redefine_extname         X509V3_EXT_CRL_add_nconf sunw_X509V3_EXT_CRL_add
3696 #pragma redefine_extname         X509V3_EXT_d2i sunw_X509V3_EXT_d2i
3697 #pragma redefine_extname         X509V3_EXT_get sunw_X509V3_EXT_get
3698 #pragma redefine_extname         X509V3_EXT_get_nid sunw_X509V3_EXT_get_nid
3699 #pragma redefine_extname         X509V3_EXT_i2d sunw_X509V3_EXT_i2d
3700 #pragma redefine_extname         X509V3_EXT_nconf sunw_X509V3_EXT_nconf
3701 #pragma redefine_extname         X509V3_EXT_nconf_nid sunw_X509V3_EXT_nconf_nid
3702 #pragma redefine_extname         X509V3_EXT_print sunw_X509V3_EXT_print
3703 #pragma redefine_extname         X509V3_EXT_print_fp sunw_X509V3_EXT_print_fp
3704 #pragma redefine_extname         X509V3_EXT_REQ_add_conf sunw_X509V3_EXT_REQ_add_
3705 #pragma redefine_extname         X509V3_EXT_REQ_add_nconf sunw_X509V3_EXT_REQ_add
3706 #pragma redefine_extname         X509V3_EXT_val_prn sunw_X509V3_EXT_val_prn
3707 #pragma redefine_extname         X509V3_extensions_print sunw_X509V3_extensions_p
3708 #pragma redefine_extname         X509v3_get_d2i sunw_X509v3_get_d2i
3709 #pragma redefine_extname         X509v3_get_ext sunw_X509v3_get_ext
3710 #pragma redefine_extname         X509v3_get_ext_by_critical sunw_X509v3_get_ext_b
3711 #pragma redefine_extname         X509v3_get_ext_by_NID sunw_X509v3_get_ext_by_NID
3712 #pragma redefine_extname         X509v3_get_ext_by_OBJ sunw_X509v3_get_ext_by_OBJ
3713 #pragma redefine_extname         X509v3_get_ext_count sunw_X509v3_get_ext_count
3714 #pragma redefine_extname         X509V3_get_section sunw_X509V3_get_section
3715 #pragma redefine_extname         X509V3_get_string sunw_X509V3_get_string
3716 #pragma redefine_extname         X509V3_get_value_bool sunw_X509V3_get_value_bool
3717 #pragma redefine_extname         X509V3_get_value_int sunw_X509V3_get_value_int
3718 #pragma redefine_extname         X509V3_NAME_from_section sunw_X509V3_NAME_from_s
3719 #pragma redefine_extname         X509V3_parse_list sunw_X509V3_parse_list
3720 #pragma redefine_extname         X509V3_section_free sunw_X509V3_section_free
3721 #pragma redefine_extname         X509V3_set_conf_lhash sunw_X509V3_set_conf_lhash
3722 #pragma redefine_extname         X509V3_set_ctx sunw_X509V3_set_ctx
3723 #pragma redefine_extname         X509V3_set_nconf sunw_X509V3_set_nconf
3724 #pragma redefine_extname         X509V3_string_free sunw_X509V3_string_free
3725 #pragma redefine_extname         ZLONG_it sunw_ZLONG_it
3726 #pragma redefine_extname         BIO_f_ssl sunw_BIO_f_ssl
3727 #pragma redefine_extname         BIO_new_buffer_ssl_connect sunw_BIO_new_buffer_s
3728 #pragma redefine_extname         BIO_new_ssl sunw_BIO_new_ssl
3729 #pragma redefine_extname         BIO_new_ssl_connect sunw_BIO_new_ssl_connect
3730 #pragma redefine_extname         BIO_ssl_copy_session_id sunw_BIO_ssl_copy_sessio
3731 #pragma redefine_extname         BIO_ssl_shutdown sunw_BIO_ssl_shutdown
3732 #pragma redefine_extname         d2i_SSL_SESSION sunw_d2i_SSL_SESSION
3733 #pragma redefine_extname         do_dtls1_write sunw_do_dtls1_write
3734 #pragma redefine_extname         dtls1_accept sunw_dtls1_accept
3735 #pragma redefine_extname         dtls1_buffer_message sunw_dtls1_buffer_message
3736 #pragma redefine_extname         dtls1_check_timeout_num sunw_dtls1_check_timeout
3737 #pragma redefine_extname         dtls1_clear sunw_dtls1_clear
3738 #pragma redefine_extname         dtls1_clear_record_buffer sunw_dtls1_clear_recor
3739 #pragma redefine_extname         dtls1_client_hello sunw_dtls1_client_hello
3740 #pragma redefine_extname         dtls1_connect sunw_dtls1_connect
3741 #pragma redefine_extname         dtls1_ctrl sunw_dtls1_ctrl
3742 #pragma redefine_extname         dtls1_default_timeout sunw_dtls1_default_timeout
3743 #pragma redefine_extname         dtls1_dispatch_alert sunw_dtls1_dispatch_alert
3744 #pragma redefine_extname         dtls1_do_write sunw_dtls1_do_write
3745 #pragma redefine_extname         dtls1_double_timeout sunw_dtls1_double_timeout
3746 #pragma redefine_extname         dtls1_enc sunw_dtls1_enc
3747 #pragma redefine_extname         dtls1_free sunw_dtls1_free
3748 #pragma redefine_extname         dtls1_get_ccs_header sunw_dtls1_get_ccs_header
3749 #pragma redefine_extname         dtls1_get_cipher sunw_dtls1_get_cipher
3750 #pragma redefine_extname         dtls1_get_message sunw_dtls1_get_message
3751 #pragma redefine_extname         dtls1_get_message_header sunw_dtls1_get_message_
3752 #pragma redefine_extname         dtls1_get_queue_priority sunw_dtls1_get_queue_pr
3753 #pragma redefine_extname         dtls1_get_record sunw_dtls1_get_record
3754 #pragma redefine_extname         dtls1_get_timeout sunw_dtls1_get_timeout
3755 #pragma redefine_extname         dtls1_handle_timeout sunw_dtls1_handle_timeout
3756 #pragma redefine_extname         dtls1_heartbeat sunw_dtls1_heartbeat
3757 #pragma redefine_extname         dtls1_is_timer_expired sunw_dtls1_is_timer_expir
```



```
3758 #pragma redefine_extname         dtls1_listen sunw_dtls1_listen
3759 #pragma redefine_extname         dtls1_min_mtu sunw_dtls1_min_mtu
3760 #pragma redefine_extname         dtls1_new sunw_dtls1_new
3761 #pragma redefine_extname         dtls1_output_cert_chain sunw_dtls1_output_cert_c
3762 #pragma redefine_extname         dtls1_process_heartbeat sunw_dtls1_process_heart
3763 #pragma redefine_extname         dtls1_read_bytes sunw_dtls1_read_bytes
3764 #pragma redefine_extname         dtls1_read_failed sunw_dtls1_read_failed
3765 #pragma redefine_extname         dtls1_reset_seq_numbers sunw_dtls1_reset_seq_num
3766 #pragma redefine_extname         dtls1_retransmit_buffered_messages sunw_dtls1_re
3767 #pragma redefine_extname         dtls1_retransmit_message sunw_dtls1_retransmit_m
3768 #pragma redefine_extname         dtls1_send_certificate_request sunw_dtls1_send_c
3769 #pragma redefine_extname         dtls1_send_change_cipher_spec sunw_dtls1_send_ch
3770 #pragma redefine_extname         dtls1_send_client_certificate sunw_dtls1_send_cl
3771 #pragma redefine_extname         dtls1_send_client_key_exchange sunw_dtls1_send_c
3772 #pragma redefine_extname         dtls1_send_client_verify sunw_dtls1_send_client_
3773 #pragma redefine_extname         dtls1_send_finished sunw_dtls1_send_finished
3774 #pragma redefine_extname         dtls1_send_hello_request sunw_dtls1_send_hello_r
3775 #pragma redefine_extname         dtls1_send_newsession_ticket sunw_dtls1_send_new
3776 #pragma redefine_extname         dtls1_send_server_certificate sunw_dtls1_send_se
3777 #pragma redefine_extname         dtls1_send_server_done sunw_dtls1_send_server_do
3778 #pragma redefine_extname         dtls1_send_server_hello sunw_dtls1_send_server_h
3779 #pragma redefine_extname         dtls1_send_server_key_exchange sunw_dtls1_send_s
3780 #pragma redefine_extname         dtls1_set_message_header sunw_dtls1_set_message_
3781 #pragma redefine_extname         dtls1_shutdown sunw_dtls1_shutdown
3782 #pragma redefine_extname         dtls1_start_timer sunw_dtls1_start_timer
3783 #pragma redefine_extname         dtls1_stop_timer sunw_dtls1_stop_timer
3784 #pragma redefine_extname         dtls1_version_str sunw_dtls1_version_str
3785 #pragma redefine_extname         dtls1_write_app_data_bytes sunw_dtls1_write_app_
3786 #pragma redefine_extname         dtls1_write_bytes sunw_dtls1_write_bytes
3787 #pragma redefine_extname         DTLSv1_client_method sunw_DTLSv1_client_method
3788 #pragma redefine_extname         DTLSv1_enc_data sunw_DTLSv1_enc_data
3789 #pragma redefine_extname         DTLSv1_method sunw_DTLSv1_method
3790 #pragma redefine_extname         DTLSv1_server_method sunw_DTLSv1_server_method
3791 #pragma redefine_extname         ERR_load_SSL_strings sunw_ERR_load_SSL_strings
3792 #pragma redefine_extname         i2d_SSL_SESSION sunw_i2d_SSL_SESSION
3793 #pragma redefine_extname         n_ssl3_mac sunw_n_ssl3_mac
3794 #pragma redefine_extname         OBJ_bsearch_ssl_cipher_id sunw_OBJ_bsearch_ssl_c
3795 #pragma redefine_extname         PEM_read_bio_SSL_SESSION sunw_PEM_read_bio_SSL_S
3796 #pragma redefine_extname         PEM_read_SSL_SESSION sunw_PEM_read_SSL_SESSION
3797 #pragma redefine_extname         PEM_write_bio_SSL_SESSION sunw_PEM_write_bio_SSL
3798 #pragma redefine_extname         PEM_write_SSL_SESSION sunw_PEM_write_SSL_SESSION
3799 #pragma redefine_extname         SRP_Calc_A_param sunw_SRP_Calc_A_param
3800 #pragma redefine_extname         SRP_generate_client_master_secret sunw_SRP_gener
3801 #pragma redefine_extname         SRP_generate_server_master_secret sunw_SRP_gener
3802 #pragma redefine_extname         SSL_accept sunw_SSL_accept
3803 #pragma redefine_extname         SSL_add_client_CA sunw_SSL_add_client_CA
3804 #pragma redefine_extname         ssl_add_clienthello_renegotiate_ext sunw_ssl_add
3805 #pragma redefine_extname         ssl_add_clienthello_tlsext sunw_ssl_add_clienthe
3806 #pragma redefine_extname         ssl_add_clienthello_use_srtp_ext sunw_ssl_add_cl
3807 #pragma redefine_extname         SSL_add_dir_cert_subjects_to_stack sunw_SSL_add_
3808 #pragma redefine_extname         SSL_add_file_cert_subjects_to_stack sunw_SSL_add
3809 #pragma redefine_extname         ssl_add_serverhello_renegotiate_ext sunw_ssl_add
3810 #pragma redefine_extname         ssl_add_serverhello_tlsext sunw_ssl_add_serverhe
3811 #pragma redefine_extname         ssl_add_serverhello_use_srtp_ext sunw_ssl_add_se
3812 #pragma redefine_extname         SSL_alert_desc_string sunw_SSL_alert_desc_string
3813 #pragma redefine_extname         SSL_alert_desc_string_long sunw_SSL_alert_desc_s
3814 #pragma redefine_extname         SSL_alert_type_string sunw_SSL_alert_type_string
3815 #pragma redefine_extname         SSL_alert_type_string_long sunw_SSL_alert_type_s
3816 #pragma redefine_extname         ssl_bad_method sunw_ssl_bad_method
3817 #pragma redefine_extname         ssl_bytes_to_cipher_list sunw_ssl_bytes_to_ciphe
3818 #pragma redefine_extname         SSL_cache_hit sunw_SSL_cache_hit
3819 #pragma redefine_extname         SSL_callback_ctrl sunw_SSL_callback_ctrl
3820 #pragma redefine_extname         ssl_cert_dup sunw_ssl_cert_dup
3821 #pragma redefine_extname         ssl_cert_free sunw_ssl_cert_free
3822 #pragma redefine_extname         ssl_cert_inst sunw_ssl_cert_inst
3823 #pragma redefine_extname         ssl_cert_new sunw_ssl_cert_new
```

```
3824 #pragma redefine_extname         ssl_cert_type sunw_ssl_cert_type
3825 #pragma redefine_extname         ssl_check_clienthello_tlsext_early sunw_ssl_chec
3826 #pragma redefine_extname         ssl_check_clienthello_tlsext_late sunw_ssl_check
3827 #pragma redefine_extname         SSL_check_private_key sunw_SSL_check_private_key
3828 #pragma redefine_extname         ssl_check_serverhello_tlsext sunw_ssl_check_serv
3829 #pragma redefine_extname         SSL_CIPHER_description sunw_SSL_CIPHER_descripti
3830 #pragma redefine_extname         SSL_CIPHER_get_bits sunw_SSL_CIPHER_get_bits
3831 #pragma redefine_extname         ssl_cipher_get_evp sunw_ssl_cipher_get_evp
3832 #pragma redefine_extname         SSL_CIPHER_get_id sunw_SSL_CIPHER_get_id
3833 #pragma redefine_extname         SSL_CIPHER_get_name sunw_SSL_CIPHER_get_name
3834 #pragma redefine_extname         SSL_CIPHER_get_version sunw_SSL_CIPHER_get_versi
3835 #pragma redefine_extname         ssl_cipher_id_cmp sunw_ssl_cipher_id_cmp
3836 #pragma redefine_extname         ssl_cipher_list_to_bytes sunw_ssl_cipher_list_to
3837 #pragma redefine_extname         ssl_cipher_ptr_id_cmp sunw_ssl_cipher_ptr_id_cmp
3838 #pragma redefine_extname         SSL_clear sunw_SSL_clear
3839 #pragma redefine_extname         ssl_clear_bad_session sunw_ssl_clear_bad_session
3840 #pragma redefine_extname         ssl_clear_cipher_ctx sunw_ssl_clear_cipher_ctx
3841 #pragma redefine_extname         ssl_clear_hash_ctx sunw_ssl_clear_hash_ctx
3842 #pragma redefine_extname         SSL_COMP_add_compression_method sunw_SSL_COMP_ad
3843 #pragma redefine_extname         SSL_COMP_get_compression_methods sunw_SSL_COMP_g
3844 #pragma redefine_extname         SSL_COMP_get_name sunw_SSL_COMP_get_name
3845 #pragma redefine_extname         SSL_connect sunw_SSL_connect
3846 #pragma redefine_extname         SSL_copy_session_id sunw_SSL_copy_session_id
3847 #pragma redefine_extname         ssl_create_cipher_list sunw_ssl_create_cipher_li
3848 #pragma redefine_extname         SSL_ctrl sunw_SSL_ctrl
3849 #pragma redefine_extname         SSL_CTX_add_client_CA sunw_SSL_CTX_add_client_CA
3850 #pragma redefine_extname         SSL_CTX_add_session sunw_SSL_CTX_add_session
3851 #pragma redefine_extname         SSL_CTX_callback_ctrl sunw_SSL_CTX_callback_ctrl
3852 #pragma redefine_extname         SSL_CTX_check_private_key sunw_SSL_CTX_check_pri
3853 #pragma redefine_extname         SSL_CTX_ctrl sunw_SSL_CTX_ctrl
3854 #pragma redefine_extname         SSL_CTX_flush_sessions sunw_SSL_CTX_flush_sessio
3855 #pragma redefine_extname         SSL_CTX_free sunw_SSL_CTX_free
3856 #pragma redefine_extname         SSL_CTX_get_cert_store sunw_SSL_CTX_get_cert_sto
3857 #pragma redefine_extname         SSL_CTX_get_client_CA_list sunw_SSL_CTX_get_clie
3858 #pragma redefine_extname         SSL_CTX_get_client_cert_cb sunw_SSL_CTX_get_clie
3859 #pragma redefine_extname         SSL_CTX_get_ex_data sunw_SSL_CTX_get_ex_data
3860 #pragma redefine_extname         SSL_CTX_get_ex_new_index sunw_SSL_CTX_get_ex_new
3861 #pragma redefine_extname         SSL_CTX_get_info_callback sunw_SSL_CTX_get_info_
3862 #pragma redefine_extname         SSL_CTX_get_quiet_shutdown sunw_SSL_CTX_get_quie
3863 #pragma redefine_extname         SSL_CTX_get_timeout sunw_SSL_CTX_get_timeout
3864 #pragma redefine_extname         SSL_CTX_get_verify_callback sunw_SSL_CTX_get_ver
3865 #pragma redefine_extname         SSL_CTX_get_verify_depth sunw_SSL_CTX_get_verify
3866 #pragma redefine_extname         SSL_CTX_get_verify_mode sunw_SSL_CTX_get_verify_
3867 #pragma redefine_extname         SSL_CTX_load_verify_locations sunw_SSL_CTX_load_
3868 #pragma redefine_extname         SSL_CTX_new sunw_SSL_CTX_new
3869 #pragma redefine_extname         SSL_CTX_remove_session sunw_SSL_CTX_remove_sessi
3870 #pragma redefine_extname         SSL_CTX_sess_get_get_cb sunw_SSL_CTX_sess_get_ge
3871 #pragma redefine_extname         SSL_CTX_sess_get_new_cb sunw_SSL_CTX_sess_get_ne
3872 #pragma redefine_extname         SSL_CTX_sess_get_remove_cb sunw_SSL_CTX_sess_get
3873 #pragma redefine_extname         SSL_CTX_sess_set_get_cb sunw_SSL_CTX_sess_set_ge
3874 #pragma redefine_extname         SSL_CTX_sess_set_new_cb sunw_SSL_CTX_sess_set_ne
3875 #pragma redefine_extname         SSL_CTX_sess_set_remove_cb sunw_SSL_CTX_sess_set
3876 #pragma redefine_extname         SSL_CTX_sessions sunw_SSL_CTX_sessions
3877 #pragma redefine_extname         SSL_CTX_set_cert_store sunw_SSL_CTX_set_cert_sto
3878 #pragma redefine_extname         SSL_CTX_set_cert_verify_callback sunw_SSL_CTX_se
3879 #pragma redefine_extname         SSL_CTX_set_cipher_list sunw_SSL_CTX_set_cipher_
3880 #pragma redefine_extname         SSL_CTX_set_client_CA_list sunw_SSL_CTX_set_clie
3881 #pragma redefine_extname         SSL_CTX_set_client_cert_cb sunw_SSL_CTX_set_clie
3882 #pragma redefine_extname         SSL_CTX_set_client_cert_engine sunw_SSL_CTX_set_
3883 #pragma redefine_extname         SSL_CTX_set_cookie_generate_cb sunw_SSL_CTX_set_
3884 #pragma redefine_extname         SSL_CTX_set_cookie_verify_cb sunw_SSL_CTX_set_co
3885 #pragma redefine_extname         SSL_CTX_set_default_passwd_cb sunw_SSL_CTX_set_d
3886 #pragma redefine_extname         SSL_CTX_set_default_passwd_cb_userdata sunw_SSL_
3887 #pragma redefine_extname         SSL_CTX_set_default_verify_paths sunw_SSL_CTX_se
3888 #pragma redefine_extname         SSL_CTX_set_ex_data sunw_SSL_CTX_set_ex_data
3889 #pragma redefine_extname         SSL_CTX_set_generate_session_id sunw_SSL_CTX_set
```



```
3890 #pragma redefine_extname         SSL_CTX_set_info_callback sunw_SSL_CTX_set_info_
3891 #pragma redefine_extname         SSL_CTX_set_msg_callback sunw_SSL_CTX_set_msg_ca
3892 #pragma redefine_extname         SSL_CTX_set_next_proto_select_cb sunw_SSL_CTX_se
3893 #pragma redefine_extname         SSL_CTX_set_next_protos_advertised_cb sunw_SSL_C
3894 #pragma redefine_extname         SSL_CTX_set_psk_client_callback sunw_SSL_CTX_set
3895 #pragma redefine_extname         SSL_CTX_set_psk_server_callback sunw_SSL_CTX_set
3896 #pragma redefine_extname         SSL_CTX_set_purpose sunw_SSL_CTX_set_purpose
3897 #pragma redefine_extname         SSL_CTX_set_quiet_shutdown sunw_SSL_CTX_set_quie
3898 #pragma redefine_extname         SSL_CTX_set_session_id_context sunw_SSL_CTX_set_
3899 #pragma redefine_extname         SSL_CTX_set_srp_cb_arg sunw_SSL_CTX_set_srp_cb_a
3900 #pragma redefine_extname         SSL_CTX_set_srp_client_pwd_callback sunw_SSL_CTX
3901 #pragma redefine_extname         SSL_CTX_set_srp_password sunw_SSL_CTX_set_srp_pa
3902 #pragma redefine_extname         SSL_CTX_set_srp_strength sunw_SSL_CTX_set_srp_st
3903 #pragma redefine_extname         SSL_CTX_set_srp_username sunw_SSL_CTX_set_srp_us
3904 #pragma redefine_extname         SSL_CTX_set_srp_username_callback sunw_SSL_CTX_s
3905 #pragma redefine_extname         SSL_CTX_set_srp_verify_param_callback sunw_SSL_C
3906 #pragma redefine_extname         SSL_CTX_set_ssl_version sunw_SSL_CTX_set_ssl_ver
3907 #pragma redefine_extname         SSL_CTX_set_timeout sunw_SSL_CTX_set_timeout
3908 #pragma redefine_extname         SSL_CTX_set_tlsext_use_srtp sunw_SSL_CTX_set_tls
3909 #pragma redefine_extname         SSL_CTX_set_tmp_dh_callback sunw_SSL_CTX_set_tmp
3910 #pragma redefine_extname         SSL_CTX_set_tmp_rsa_callback sunw_SSL_CTX_set_tm
3911 #pragma redefine_extname         SSL_CTX_set_trust sunw_SSL_CTX_set_trust
3912 #pragma redefine_extname         SSL_CTX_set_verify sunw_SSL_CTX_set_verify
3913 #pragma redefine_extname         SSL_CTX_set_verify_depth sunw_SSL_CTX_set_verify
3914 #pragma redefine_extname         SSL_CTX_set1_param sunw_SSL_CTX_set1_param
3915 #pragma redefine_extname         SSL_CTX_SRP_CTX_free sunw_SSL_CTX_SRP_CTX_free
3916 #pragma redefine_extname         SSL_CTX_SRP_CTX_init sunw_SSL_CTX_SRP_CTX_init
3917 #pragma redefine_extname         SSL_CTX_use_certificate sunw_SSL_CTX_use_certifi
3918 #pragma redefine_extname         SSL_CTX_use_certificate_ASN1 sunw_SSL_CTX_use_ce
3919 #pragma redefine_extname         SSL_CTX_use_certificate_chain_file sunw_SSL_CTX_
3920 #pragma redefine_extname         SSL_CTX_use_certificate_file sunw_SSL_CTX_use_ce
3921 #pragma redefine_extname         SSL_CTX_use_PrivateKey sunw_SSL_CTX_use_PrivateK
3922 #pragma redefine_extname         SSL_CTX_use_PrivateKey_ASN1 sunw_SSL_CTX_use_Pri
3923 #pragma redefine_extname         SSL_CTX_use_PrivateKey_file sunw_SSL_CTX_use_Pri
3924 #pragma redefine_extname         SSL_CTX_use_psk_identity_hint sunw_SSL_CTX_use_p
3925 #pragma redefine_extname         SSL_CTX_use_RSAPrivateKey sunw_SSL_CTX_use_RSAPr
3926 #pragma redefine_extname         SSL_CTX_use_RSAPrivateKey_ASN1 sunw_SSL_CTX_use_
3927 #pragma redefine_extname         SSL_CTX_use_RSAPrivateKey_file sunw_SSL_CTX_use_
3928 #pragma redefine_extname         ssl_do_client_cert_cb sunw_ssl_do_client_cert_cb
3929 #pragma redefine_extname         SSL_do_handshake sunw_SSL_do_handshake
3930 #pragma redefine_extname         SSL_dup sunw_SSL_dup
3931 #pragma redefine_extname         SSL_dup_CA_list sunw_SSL_dup_CA_list
3932 #pragma redefine_extname         SSL_export_keying_material sunw_SSL_export_keyin
3933 #pragma redefine_extname         ssl_fill_hello_random sunw_ssl_fill_hello_random
3934 #pragma redefine_extname         SSL_free sunw_SSL_free
3935 #pragma redefine_extname         ssl_free_wbio_buffer sunw_ssl_free_wbio_buffer
3936 #pragma redefine_extname         ssl_get_algorithm2 sunw_ssl_get_algorithm2
3937 #pragma redefine_extname         SSL_get_certificate sunw_SSL_get_certificate
3938 #pragma redefine_extname         SSL_get_cipher_list sunw_SSL_get_cipher_list
3939 #pragma redefine_extname         SSL_get_ciphers sunw_SSL_get_ciphers
3940 #pragma redefine_extname         ssl_get_ciphers_by_id sunw_ssl_get_ciphers_by_id
3941 #pragma redefine_extname         SSL_get_client_CA_list sunw_SSL_get_client_CA_li
3942 #pragma redefine_extname         SSL_get_current_cipher sunw_SSL_get_current_ciph
3943 #pragma redefine_extname         SSL_get_current_compression sunw_SSL_get_current
3944 #pragma redefine_extname         SSL_get_current_expansion sunw_SSL_get_current_e
3945 #pragma redefine_extname         SSL_get_default_timeout sunw_SSL_get_default_tim
3946 #pragma redefine_extname         SSL_get_error sunw_SSL_get_error
3947 #pragma redefine_extname         SSL_get_ex_data sunw_SSL_get_ex_data
3948 #pragma redefine_extname         SSL_get_ex_data_X509_STORE_CTX_idx sunw_SSL_get_
3949 #pragma redefine_extname         SSL_get_ex_new_index sunw_SSL_get_ex_new_index
3950 #pragma redefine_extname         SSL_get_fd sunw_SSL_get_fd
3951 #pragma redefine_extname         SSL_get_finished sunw_SSL_get_finished
3952 #pragma redefine_extname         ssl_get_handshake_digest sunw_ssl_get_handshake_
3953 #pragma redefine_extname         SSL_get_info_callback sunw_SSL_get_info_callback
3954 #pragma redefine_extname         ssl_get_new_session sunw_ssl_get_new_session
3955 #pragma redefine_extname         SSL_get_peer_cert_chain sunw_SSL_get_peer_cert_c
```

```
3956 #pragma redefine_extname        SSL_get_peer_certificate sunw_SSL_get_peer_certi
3957 #pragma redefine_extname        SSL_get_peer_finished sunw_SSL_get_peer_finished
3958 #pragma redefine_extname        ssl_get_prev_session sunw_ssl_get_prev_session
3959 #pragma redefine_extname        SSL_get_privatekey sunw_SSL_get_privatekey
3960 #pragma redefine_extname        SSL_get_psk_identity sunw_SSL_get_psk_identity
3961 #pragma redefine_extname        SSL_get_psk_identity_hint sunw_SSL_get_psk_ident
3962 #pragma redefine_extname        SSL_get_quiet_shutdown sunw_SSL_get_quiet_shutdo
3963 #pragma redefine_extname        SSL_get_rbio sunw_SSL_get_rbio
3964 #pragma redefine_extname        SSL_get_read_ahead sunw_SSL_get_read_ahead
3965 #pragma redefine_extname        SSL_get_rfd sunw_SSL_get_rfd
3966 #pragma redefine_extname        SSL_get_selected_srtp_profile sunw_SSL_get_selec
3967 #pragma redefine_extname        ssl_get_server_send_cert sunw_ssl_get_server_sen
3968 #pragma redefine_extname        ssl_get_server_send_pkey sunw_ssl_get_server_sen
3969 #pragma redefine_extname        SSL_get_servername sunw_SSL_get_servername
3970 #pragma redefine_extname        SSL_get_servername_type sunw_SSL_get_servername_
3971 #pragma redefine_extname        SSL_get_session sunw_SSL_get_session
3972 #pragma redefine_extname        SSL_get_shared_ciphers sunw_SSL_get_shared_ciphe
3973 #pragma redefine_extname        SSL_get_shutdown sunw_SSL_get_shutdown
3974 #pragma redefine_extname        ssl_get_sign_pkey sunw_ssl_get_sign_pkey
3975 #pragma redefine_extname        SSL_get_srp_g sunw_SSL_get_srp_g
3976 #pragma redefine_extname        SSL_get_srp_N sunw_SSL_get_srp_N
3977 #pragma redefine_extname        SSL_get_srp_userinfo sunw_SSL_get_srp_userinfo
3978 #pragma redefine_extname        SSL_get_srp_username sunw_SSL_get_srp_username
3979 #pragma redefine_extname        SSL_get_srtp_profiles sunw_SSL_get_srtp_profiles
3980 #pragma redefine_extname        SSL_get_SSL_CTX sunw_SSL_get_SSL_CTX
3981 #pragma redefine_extname        SSL_get_ssl_method sunw_SSL_get_ssl_method
3982 #pragma redefine_extname        SSL_get_verify_callback sunw_SSL_get_verify_call
3983 #pragma redefine_extname        SSL_get_verify_depth sunw_SSL_get_verify_depth
3984 #pragma redefine_extname        SSL_get_verify_mode sunw_SSL_get_verify_mode
3985 #pragma redefine_extname        SSL_get_verify_result sunw_SSL_get_verify_result
3986 #pragma redefine_extname        SSL_get_version sunw_SSL_get_version
3987 #pragma redefine_extname        SSL_get_wbio sunw_SSL_get_wbio
3988 #pragma redefine_extname        SSL_get_wfd sunw_SSL_get_wfd
3989 #pragma redefine_extname        SSL_get0_next_proto_negotiated sunw_SSL_get0_nex
3990 #pragma redefine_extname        SSL_get1_session sunw_SSL_get1_session
3991 #pragma redefine_extname        SSL_has_matching_session_id sunw_SSL_has_matchin
3992 #pragma redefine_extname        ssl_init_wbio_buffer sunw_ssl_init_wbio_buffer
3993 #pragma redefine_extname        SSL_library_init sunw_SSL_library_init
3994 #pragma redefine_extname        ssl_load_ciphers sunw_ssl_load_ciphers
3995 #pragma redefine_extname        SSL_load_client_CA_file sunw_SSL_load_client_CA_
3996 #pragma redefine_extname        SSL_load_error_strings sunw_SSL_load_error_strin
3997 #pragma redefine_extname        SSL_new sunw_SSL_new
3998 #pragma redefine_extname        ssl_ok sunw_ssl_ok
3999 #pragma redefine_extname        ssl_parse_clienthello_renegotiate_ext sunw_ssl_p
4000 #pragma redefine_extname        ssl_parse_clienthello_tlsext sunw_ssl_parse_clie
4001 #pragma redefine_extname        ssl_parse_clienthello_use_srtp_ext sunw_ssl_pars
4002 #pragma redefine_extname        ssl_parse_serverhello_renegotiate_ext sunw_ssl_p
4003 #pragma redefine_extname        ssl_parse_serverhello_tlsext sunw_ssl_parse_serv
4004 #pragma redefine_extname        ssl_parse_serverhello_use_srtp_ext sunw_ssl_pars
4005 #pragma redefine_extname        SSL_peek sunw_SSL_peek
4006 #pragma redefine_extname        SSL_pending sunw_SSL_pending
4007 #pragma redefine_extname        ssl_prepare_clienthello_tlsext sunw_ssl_prepare_
4008 #pragma redefine_extname        ssl_prepare_serverhello_tlsext sunw_ssl_prepare_
4009 #pragma redefine_extname        SSL_read sunw_SSL_read
4010 #pragma redefine_extname        SSL_renegotiate sunw_SSL_renegotiate
4011 #pragma redefine_extname        SSL_renegotiate_abbreviated sunw_SSL_renegotiate
4012 #pragma redefine_extname        SSL_renegotiate_pending sunw_SSL_renegotiate_pen
4013 #pragma redefine_extname        ssl_replace_hash sunw_ssl_replace_hash
4014 #pragma redefine_extname        SSL_rstate_string sunw_SSL_rstate_string
4015 #pragma redefine_extname        SSL_rstate_string_long sunw_SSL_rstate_string_lo
4016 #pragma redefine_extname        SSL_select_next_proto sunw_SSL_select_next_proto
4017 #pragma redefine_extname        ssl_sess_cert_free sunw_ssl_sess_cert_free
4018 #pragma redefine_extname        ssl_sess_cert_new sunw_ssl_sess_cert_new
4019 #pragma redefine_extname        SSL_SESSION_free sunw_SSL_SESSION_free
4020 #pragma redefine_extname        SSL_SESSION_get_compress_id sunw_SSL_SESSION_get
4021 #pragma redefine_extname        SSL_SESSION_get_ex_data sunw_SSL_SESSION_get_ex_
```



```
4022 #pragma redefine_extname        SSL_SESSION_get_ex_new_index sunw_SSL_SESSION_ge
4023 #pragma redefine_extname        SSL_SESSION_get_id sunw_SSL_SESSION_get_id
4024 #pragma redefine_extname        SSL_SESSION_get_time sunw_SSL_SESSION_get_time
4025 #pragma redefine_extname        SSL_SESSION_get_timeout sunw_SSL_SESSION_get_tim
4026 #pragma redefine_extname        SSL_SESSION_get0_peer sunw_SSL_SESSION_get0_peer
4027 #pragma redefine_extname        SSL_SESSION_new sunw_SSL_SESSION_new
4028 #pragma redefine_extname        SSL_SESSION_print sunw_SSL_SESSION_print
4029 #pragma redefine_extname        SSL_SESSION_print_fp sunw_SSL_SESSION_print_fp
4030 #pragma redefine_extname        SSL_SESSION_set_ex_data sunw_SSL_SESSION_set_ex_
4031 #pragma redefine_extname        SSL_SESSION_set_time sunw_SSL_SESSION_set_time
4032 #pragma redefine_extname        SSL_SESSION_set_timeout sunw_SSL_SESSION_set_tim
4033 #pragma redefine_extname        SSL_SESSION_set1_id_context sunw_SSL_SESSION_set
4034 #pragma redefine_extname        SSL_set_accept_state sunw_SSL_set_accept_state
4035 #pragma redefine_extname        SSL_set_bio sunw_SSL_set_bio
4036 #pragma redefine_extname        ssl_set_cert_masks sunw_ssl_set_cert_masks
4037 #pragma redefine_extname        SSL_set_cipher_list sunw_SSL_set_cipher_list
4038 #pragma redefine_extname        SSL_set_client_CA_list sunw_SSL_set_client_CA_li
4039 #pragma redefine_extname        SSL_set_connect_state sunw_SSL_set_connect_state
4040 #pragma redefine_extname        SSL_set_debug sunw_SSL_set_debug
4041 #pragma redefine_extname        SSL_set_ex_data sunw_SSL_set_ex_data
4042 #pragma redefine_extname        SSL_set_fd sunw_SSL_set_fd
4043 #pragma redefine_extname        SSL_set_generate_session_id sunw_SSL_set_generat
4044 #pragma redefine_extname        SSL_set_info_callback sunw_SSL_set_info_callback
4045 #pragma redefine_extname        SSL_set_msg_callback sunw_SSL_set_msg_callback
4046 #pragma redefine_extname        ssl_set_peer_cert_type sunw_ssl_set_peer_cert_ty
4047 #pragma redefine_extname        SSL_set_psk_client_callback sunw_SSL_set_psk_cli
4048 #pragma redefine_extname        SSL_set_psk_server_callback sunw_SSL_set_psk_ser
4049 #pragma redefine_extname        SSL_set_purpose sunw_SSL_set_purpose
4050 #pragma redefine_extname        SSL_set_quiet_shutdown sunw_SSL_set_quiet_shutdo
4051 #pragma redefine_extname        SSL_set_read_ahead sunw_SSL_set_read_ahead
4052 #pragma redefine_extname        SSL_set_rfd sunw_SSL_set_rfd
4053 #pragma redefine_extname        SSL_set_session sunw_SSL_set_session
4054 #pragma redefine_extname        SSL_set_session_id_context sunw_SSL_set_session_
4055 #pragma redefine_extname        SSL_set_session_secret_cb sunw_SSL_set_session_s
4056 #pragma redefine_extname        SSL_set_session_ticket_ext sunw_SSL_set_session_
4057 #pragma redefine_extname        SSL_set_session_ticket_ext_cb sunw_SSL_set_sessi
4058 #pragma redefine_extname        SSL_set_shutdown sunw_SSL_set_shutdown
4059 #pragma redefine_extname        SSL_set_srp_server_param sunw_SSL_set_srp_server
4060 #pragma redefine_extname        SSL_set_srp_server_param_pw sunw_SSL_set_srp_ser
4061 #pragma redefine_extname        SSL_set_SSL_CTX sunw_SSL_set_SSL_CTX
4062 #pragma redefine_extname        SSL_set_ssl_method sunw_SSL_set_ssl_method
4063 #pragma redefine_extname        SSL_set_state sunw_SSL_set_state
4064 #pragma redefine_extname        SSL_set_tlsext_use_srtp sunw_SSL_set_tlsext_use_
4065 #pragma redefine_extname        SSL_set_tmp_dh_callback sunw_SSL_set_tmp_dh_call
4066 #pragma redefine_extname        SSL_set_tmp_rsa_callback sunw_SSL_set_tmp_rsa_ca
4067 #pragma redefine_extname        SSL_set_trust sunw_SSL_set_trust
4068 #pragma redefine_extname        SSL_set_verify sunw_SSL_set_verify
4069 #pragma redefine_extname        SSL_set_verify_depth sunw_SSL_set_verify_depth
4070 #pragma redefine_extname        SSL_set_verify_result sunw_SSL_set_verify_result
4071 #pragma redefine_extname        SSL_set_wfd sunw_SSL_set_wfd
4072 #pragma redefine_extname        SSL_set1_param sunw_SSL_set1_param
4073 #pragma redefine_extname        SSL_shutdown sunw_SSL_shutdown
4074 #pragma redefine_extname        SSL_SRP_CTX_free sunw_SSL_SRP_CTX_free
4075 #pragma redefine_extname        SSL_SRP_CTX_init sunw_SSL_SRP_CTX_init
4076 #pragma redefine_extname        SSL_srp_server_param_with_username sunw_SSL_srp_
4077 #pragma redefine_extname        SSL_state sunw_SSL_state
4078 #pragma redefine_extname        SSL_state_string sunw_SSL_state_string
4079 #pragma redefine_extname        SSL_state_string_long sunw_SSL_state_string_long
4080 #pragma redefine_extname        ssl_undefined_const_function sunw_ssl_undefined_
4081 #pragma redefine_extname        ssl_undefined_function sunw_ssl_undefined_functi
4082 #pragma redefine_extname        ssl_undefined_void_function sunw_ssl_undefined_v
4083 #pragma redefine_extname        ssl_update_cache sunw_ssl_update_cache
4084 #pragma redefine_extname        SSL_use_certificate sunw_SSL_use_certificate
4085 #pragma redefine_extname        SSL_use_certificate_ASN1 sunw_SSL_use_certificat
4086 #pragma redefine_extname        SSL_use_certificate_file sunw_SSL_use_certificat
4087 #pragma redefine_extname        SSL_use_PrivateKey sunw_SSL_use_PrivateKey
```

```
4088 #pragma redefine_extname         SSL_use_PrivateKey_ASN1 sunw_SSL_use_PrivateKey_
4089 #pragma redefine_extname         SSL_use_PrivateKey_file sunw_SSL_use_PrivateKey_
4090 #pragma redefine_extname         SSL_use_psk_identity_hint sunw_SSL_use_psk_ident
4091 #pragma redefine_extname         SSL_use_RSAPrivateKey sunw_SSL_use_RSAPrivateKey
4092 #pragma redefine_extname         SSL_use_RSAPrivateKey_ASN1 sunw_SSL_use_RSAPriva
4093 #pragma redefine_extname         SSL_use_RSAPrivateKey_file sunw_SSL_use_RSAPriva
4094 #pragma redefine_extname         ssl_verify_alarm_type sunw_ssl_verify_alarm_type
4095 #pragma redefine_extname         ssl_verify_cert_chain sunw_ssl_verify_cert_chain
4096 #pragma redefine_extname         SSL_version sunw_SSL_version
4097 #pragma redefine_extname         SSL_version_str sunw_SSL_version_str
4098 #pragma redefine_extname         SSL_want sunw_SSL_want
4099 #pragma redefine_extname         SSL_write sunw_SSL_write
4100 #pragma redefine_extname         ssl2_accept sunw_ssl2_accept
4101 #pragma redefine_extname         ssl2_callback_ctrl sunw_ssl2_callback_ctrl
4102 #pragma redefine_extname         ssl2_ciphers sunw_ssl2_ciphers
4103 #pragma redefine_extname         ssl2_clear sunw_ssl2_clear
4104 #pragma redefine_extname         ssl2_connect sunw_ssl2_connect
4105 #pragma redefine_extname         ssl2_ctrl sunw_ssl2_ctrl
4106 #pragma redefine_extname         ssl2_ctx_callback_ctrl sunw_ssl2_ctx_callback_ct
4107 #pragma redefine_extname         ssl2_ctx_ctrl sunw_ssl2_ctx_ctrl
4108 #pragma redefine_extname         ssl2_default_timeout sunw_ssl2_default_timeout
4109 #pragma redefine_extname         ssl2_do_write sunw_ssl2_do_write
4110 #pragma redefine_extname         ssl2_enc sunw_ssl2_enc
4111 #pragma redefine_extname         ssl2_enc_init sunw_ssl2_enc_init
4112 #pragma redefine_extname         ssl2_free sunw_ssl2_free
4113 #pragma redefine_extname         ssl2_generate_key_material sunw_ssl2_generate_ke
4114 #pragma redefine_extname         ssl2_get_cipher sunw_ssl2_get_cipher
4115 #pragma redefine_extname         ssl2_get_cipher_by_char sunw_ssl2_get_cipher_by_
4116 #pragma redefine_extname         ssl2_mac sunw_ssl2_mac
4117 #pragma redefine_extname         ssl2_new sunw_ssl2_new
4118 #pragma redefine_extname         ssl2_num_ciphers sunw_ssl2_num_ciphers
4119 #pragma redefine_extname         ssl2_part_read sunw_ssl2_part_read
4120 #pragma redefine_extname         ssl2_peek sunw_ssl2_peek
4121 #pragma redefine_extname         ssl2_pending sunw_ssl2_pending
4122 #pragma redefine_extname         ssl2_put_cipher_by_char sunw_ssl2_put_cipher_by_
4123 #pragma redefine_extname         ssl2_read sunw_ssl2_read
4124 #pragma redefine_extname         ssl2_return_error sunw_ssl2_return_error
4125 #pragma redefine_extname         ssl2_set_certificate sunw_ssl2_set_certificate
4126 #pragma redefine_extname         ssl2_shutdown sunw_ssl2_shutdown
4127 #pragma redefine_extname         ssl2_version_str sunw_ssl2_version_str
4128 #pragma redefine_extname         ssl2_write sunw_ssl2_write
4129 #pragma redefine_extname         ssl2_write_error sunw_ssl2_write_error
4130 #pragma redefine_extname         ssl23_accept sunw_ssl23_accept
4131 #pragma redefine_extname         ssl23_connect sunw_ssl23_connect
4132 #pragma redefine_extname         ssl23_default_timeout sunw_ssl23_default_timeout
4133 #pragma redefine_extname         ssl23_get_cipher sunw_ssl23_get_cipher
4134 #pragma redefine_extname         ssl23_get_cipher_by_char sunw_ssl23_get_cipher_b
4135 #pragma redefine_extname         ssl23_get_client_hello sunw_ssl23_get_client_hel
4136 #pragma redefine_extname         ssl23_num_ciphers sunw_ssl23_num_ciphers
4137 #pragma redefine_extname         ssl23_peek sunw_ssl23_peek
4138 #pragma redefine_extname         ssl23_put_cipher_by_char sunw_ssl23_put_cipher_b
4139 #pragma redefine_extname         ssl23_read sunw_ssl23_read
4140 #pragma redefine_extname         ssl23_read_bytes sunw_ssl23_read_bytes
4141 #pragma redefine_extname         ssl23_write sunw_ssl23_write
4142 #pragma redefine_extname         ssl23_write_bytes sunw_ssl23_write_bytes
4143 #pragma redefine_extname         ssl3_accept sunw_ssl3_accept
4144 #pragma redefine_extname         ssl3_alert_code sunw_ssl3_alert_code
4145 #pragma redefine_extname         ssl3_callback_ctrl sunw_ssl3_callback_ctrl
4146 #pragma redefine_extname         ssl3_cbc_copy_mac sunw_ssl3_cbc_copy_mac
4147 #pragma redefine_extname         ssl3_cbc_digest_record sunw_ssl3_cbc_digest_reco
4148 #pragma redefine_extname         ssl3_cbc_record_digest_supported sunw_ssl3_cbc_r
4149 #pragma redefine_extname         ssl3_cbc_remove_padding sunw_ssl3_cbc_remove_pad
4150 #pragma redefine_extname         ssl3_cert_verify_mac sunw_ssl3_cert_verify_mac
4151 #pragma redefine_extname         ssl3_change_cipher_state sunw_ssl3_change_cipher
4152 #pragma redefine_extname         ssl3_check_cert_and_algorithm sunw_ssl3_check_ce
4153 #pragma redefine_extname         ssl3_check_client_hello sunw_ssl3_check_client_h
```



```
4154 #pragma redefine_extname         ssl3_check_finished sunw_ssl3_check_finished
4155 #pragma redefine_extname         ssl3_choose_cipher sunw_ssl3_choose_cipher
4156 #pragma redefine_extname         ssl3_ciphers sunw_ssl3_ciphers
4157 #pragma redefine_extname         ssl3_cleanup_key_block sunw_ssl3_cleanup_key_blo
4158 #pragma redefine_extname         ssl3_clear sunw_ssl3_clear
4159 #pragma redefine_extname         ssl3_client_hello sunw_ssl3_client_hello
4160 #pragma redefine_extname         ssl3_comp_find sunw_ssl3_comp_find
4161 #pragma redefine_extname         ssl3_connect sunw_ssl3_connect
4162 #pragma redefine_extname         ssl3_ctrl sunw_ssl3_ctrl
4163 #pragma redefine_extname         ssl3_ctx_callback_ctrl sunw_ssl3_ctx_callback_ct
4164 #pragma redefine_extname         ssl3_ctx_ctrl sunw_ssl3_ctx_ctrl
4165 #pragma redefine_extname         ssl3_default_timeout sunw_ssl3_default_timeout
4166 #pragma redefine_extname         ssl3_digest_cached_records sunw_ssl3_digest_cach
4167 #pragma redefine_extname         ssl3_dispatch_alert sunw_ssl3_dispatch_alert
4168 #pragma redefine_extname         ssl3_do_change_cipher_spec sunw_ssl3_do_change_c
4169 #pragma redefine_extname         ssl3_do_compress sunw_ssl3_do_compress
4170 #pragma redefine_extname         ssl3_do_uncompress sunw_ssl3_do_uncompress
4171 #pragma redefine_extname         ssl3_do_write sunw_ssl3_do_write
4172 #pragma redefine_extname         ssl3_enc sunw_ssl3_enc
4173 #pragma redefine_extname         ssl3_final_finish_mac sunw_ssl3_final_finish_mac
4174 #pragma redefine_extname         ssl3_finish_mac sunw_ssl3_finish_mac
4175 #pragma redefine_extname         ssl3_free sunw_ssl3_free
4176 #pragma redefine_extname         ssl3_free_digest_list sunw_ssl3_free_digest_list
4177 #pragma redefine_extname         ssl3_generate_master_secret sunw_ssl3_generate_m
4178 #pragma redefine_extname         ssl3_get_cert_status sunw_ssl3_get_cert_status
4179 #pragma redefine_extname         ssl3_get_cert_verify sunw_ssl3_get_cert_verify
4180 #pragma redefine_extname         ssl3_get_certificate_request sunw_ssl3_get_certi
4181 #pragma redefine_extname         ssl3_get_cipher sunw_ssl3_get_cipher
4182 #pragma redefine_extname         ssl3_get_cipher_by_char sunw_ssl3_get_cipher_by_
4183 #pragma redefine_extname         ssl3_get_client_certificate sunw_ssl3_get_client
4184 #pragma redefine_extname         ssl3_get_client_hello sunw_ssl3_get_client_hello
4185 #pragma redefine_extname         ssl3_get_client_key_exchange sunw_ssl3_get_clien
4186 #pragma redefine_extname         ssl3_get_finished sunw_ssl3_get_finished
4187 #pragma redefine_extname         ssl3_get_key_exchange sunw_ssl3_get_key_exchange
4188 #pragma redefine_extname         ssl3_get_message sunw_ssl3_get_message
4189 #pragma redefine_extname         ssl3_get_new_session_ticket sunw_ssl3_get_new_se
4190 #pragma redefine_extname         ssl3_get_next_proto sunw_ssl3_get_next_proto
4191 #pragma redefine_extname         ssl3_get_req_cert_type sunw_ssl3_get_req_cert_ty
4192 #pragma redefine_extname         ssl3_get_server_certificate sunw_ssl3_get_server
4193 #pragma redefine_extname         ssl3_get_server_done sunw_ssl3_get_server_done
4194 #pragma redefine_extname         ssl3_get_server_hello sunw_ssl3_get_server_hello
4195 #pragma redefine_extname         ssl3_init_finished_mac sunw_ssl3_init_finished_m
4196 #pragma redefine_extname         ssl3_new sunw_ssl3_new
4197 #pragma redefine_extname         ssl3_num_ciphers sunw_ssl3_num_ciphers
4198 #pragma redefine_extname         ssl3_output_cert_chain sunw_ssl3_output_cert_cha
4199 #pragma redefine_extname         ssl3_peek sunw_ssl3_peek
4200 #pragma redefine_extname         ssl3_pending sunw_ssl3_pending
4201 #pragma redefine_extname         ssl3_put_cipher_by_char sunw_ssl3_put_cipher_by_
4202 #pragma redefine_extname         ssl3_read sunw_ssl3_read
4203 #pragma redefine_extname         ssl3_read_bytes sunw_ssl3_read_bytes
4204 #pragma redefine_extname         ssl3_read_n sunw_ssl3_read_n
4205 #pragma redefine_extname         ssl3_record_sequence_update sunw_ssl3_record_seq
4206 #pragma redefine_extname         ssl3_release_read_buffer sunw_ssl3_release_read_
4207 #pragma redefine_extname         ssl3_release_write_buffer sunw_ssl3_release_writ
4208 #pragma redefine_extname         ssl3_renegotiate sunw_ssl3_renegotiate
4209 #pragma redefine_extname         ssl3_renegotiate_check sunw_ssl3_renegotiate_che
4210 #pragma redefine_extname         ssl3_send_alert sunw_ssl3_send_alert
4211 #pragma redefine_extname         ssl3_send_cert_status sunw_ssl3_send_cert_status
4212 #pragma redefine_extname         ssl3_send_certificate_request sunw_ssl3_send_cer
4213 #pragma redefine_extname         ssl3_send_change_cipher_spec sunw_ssl3_send_chan
4214 #pragma redefine_extname         ssl3_send_client_certificate sunw_ssl3_send_clie
4215 #pragma redefine_extname         ssl3_send_client_key_exchange sunw_ssl3_send_cli
4216 #pragma redefine_extname         ssl3_send_client_verify sunw_ssl3_send_client_ve
4217 #pragma redefine_extname         ssl3_send_finished sunw_ssl3_send_finished
4218 #pragma redefine_extname         ssl3_send_hello_request sunw_ssl3_send_hello_req
4219 #pragma redefine_extname         ssl3_send_newsession_ticket sunw_ssl3_send_newse
```

```
4220 #pragma redefine_extname        ssl3_send_next_proto sunw_ssl3_send_next_proto
4221 #pragma redefine_extname        ssl3_send_server_certificate sunw_ssl3_send_serv
4222 #pragma redefine_extname        ssl3_send_server_done sunw_ssl3_send_server_done
4223 #pragma redefine_extname        ssl3_send_server_hello sunw_ssl3_send_server_hel
4224 #pragma redefine_extname        ssl3_send_server_key_exchange sunw_ssl3_send_ser
4225 #pragma redefine_extname        ssl3_setup_buffers sunw_ssl3_setup_buffers
4226 #pragma redefine_extname        ssl3_setup_key_block sunw_ssl3_setup_key_block
4227 #pragma redefine_extname        ssl3_setup_read_buffer sunw_ssl3_setup_read_buff
4228 #pragma redefine_extname        ssl3_setup_write_buffer sunw_ssl3_setup_write_bu
4229 #pragma redefine_extname        ssl3_shutdown sunw_ssl3_shutdown
4230 #pragma redefine_extname        ssl3_undef_enc_method sunw_ssl3_undef_enc_method
4231 #pragma redefine_extname        ssl3_version_str sunw_ssl3_version_str
4232 #pragma redefine_extname        ssl3_write sunw_ssl3_write
4233 #pragma redefine_extname        ssl3_write_bytes sunw_ssl3_write_bytes
4234 #pragma redefine_extname        ssl3_write_pending sunw_ssl3_write_pending
4235 #pragma redefine_extname        SSLv2_client_method sunw_SSLv2_client_method
4236 #pragma redefine_extname        SSLv2_method sunw_SSLv2_method
4237 #pragma redefine_extname        SSLv2_server_method sunw_SSLv2_server_method
4238 #pragma redefine_extname        SSLv23_client_method sunw_SSLv23_client_method
4239 #pragma redefine_extname        SSLv23_method sunw_SSLv23_method
4240 #pragma redefine_extname        SSLv23_server_method sunw_SSLv23_server_method
4241 #pragma redefine_extname        SSLv3_client_method sunw_SSLv3_client_method
4242 #pragma redefine_extname        SSLv3_enc_data sunw_SSLv3_enc_data
4243 #pragma redefine_extname        SSLv3_method sunw_SSLv3_method
4244 #pragma redefine_extname        SSLv3_server_method sunw_SSLv3_server_method
4245 #pragma redefine_extname        tls1_alert_code sunw_tls1_alert_code
4246 #pragma redefine_extname        tls1_cbc_remove_padding sunw_tls1_cbc_remove_pad
4247 #pragma redefine_extname        tls1_cert_verify_mac sunw_tls1_cert_verify_mac
4248 #pragma redefine_extname        tls1_change_cipher_state sunw_tls1_change_cipher
4249 #pragma redefine_extname        tls1_clear sunw_tls1_clear
4250 #pragma redefine_extname        tls1_default_timeout sunw_tls1_default_timeout
4251 #pragma redefine_extname        tls1_enc sunw_tls1_enc
4252 #pragma redefine_extname        tls1_export_keying_material sunw_tls1_export_key
4253 #pragma redefine_extname        tls1_final_finish_mac sunw_tls1_final_finish_mac
4254 #pragma redefine_extname        tls1_free sunw_tls1_free
4255 #pragma redefine_extname        tls1_generate_master_secret sunw_tls1_generate_m
4256 #pragma redefine_extname        tls1_heartbeat sunw_tls1_heartbeat
4257 #pragma redefine_extname        tls1_mac sunw_tls1_mac
4258 #pragma redefine_extname        tls1_new sunw_tls1_new
4259 #pragma redefine_extname        tls1_process_heartbeat sunw_tls1_process_heartbe
4260 #pragma redefine_extname        tls1_process_sigalgs sunw_tls1_process_sigalgs
4261 #pragma redefine_extname        tls1_process_ticket sunw_tls1_process_ticket
4262 #pragma redefine_extname        tls1_setup_key_block sunw_tls1_setup_key_block
4263 #pragma redefine_extname        tls1_version_str sunw_tls1_version_str
4264 #pragma redefine_extname        tls12_get_hash sunw_tls12_get_hash
4265 #pragma redefine_extname        tls12_get_req_sig_algs sunw_tls12_get_req_sig_al
4266 #pragma redefine_extname        tls12_get_sigandhash sunw_tls12_get_sigandhash
4267 #pragma redefine_extname        tls12_get_sigid sunw_tls12_get_sigid
4268 #pragma redefine_extname        TLSv1_1_client_method sunw_TLSv1_1_client_method
4269 #pragma redefine_extname        TLSv1_1_method sunw_TLSv1_1_method
4270 #pragma redefine_extname        TLSv1_1_server_method sunw_TLSv1_1_server_method
4271 #pragma redefine_extname        TLSv1_2_client_method sunw_TLSv1_2_client_method
4272 #pragma redefine_extname        TLSv1_2_method sunw_TLSv1_2_method
4273 #pragma redefine_extname        TLSv1_2_server_method sunw_TLSv1_2_server_method
4274 #pragma redefine_extname        TLSv1_client_method sunw_TLSv1_client_method
4275 #pragma redefine_extname        TLSv1_enc_data sunw_TLSv1_enc_data
4276 #pragma redefine_extname        TLSv1_method sunw_TLSv1_method
4277 #pragma redefine_extname        TLSv1_server_method sunw_TLSv1_server_method

4279 #endif /* _SUNW_PREFIX_H */
4280 #endif /* ! codereview */
```

```
**********************************************************
   24301 Wed Aug 13 19:51:50 2014
new/usr/src/lib/openssl/include/openssl/symhacks.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_SYMHACKS_H
#define HEADER_SYMHACKS_H

#include <openssl/e_os2.h>

/* Hacks to solve the problem with linkers incapable of handling very long
   symbol names.  In the case of VMS, the limit is 31 characters on VMS for
```



```
   VAX. */
/* Note that this affects util/libeay.num and util/ssleay.num...  you may
   change those manually, but that's not recommended, as those files are
   controlled centrally and updated on Unix, and the central definition
   may disagree with yours, which in turn may come with shareable library
   incompatibilities. */
#ifdef OPENSSL_SYS_VMS

/* Hack a long name in crypto/ex_data.c */
#undef CRYPTO_get_ex_data_implementation
#define CRYPTO_get_ex_data_implementation       CRYPTO_get_ex_data_impl
#undef CRYPTO_set_ex_data_implementation
#define CRYPTO_set_ex_data_implementation       CRYPTO_set_ex_data_impl

/* Hack a long name in crypto/asn1/a_mbstr.c */
#undef ASN1_STRING_set_default_mask_asc
#define ASN1_STRING_set_default_mask_asc        ASN1_STRING_set_def_mask_asc

#if 0 /* No longer needed, since safestack macro magic does the job */
/* Hack the names created with DECLARE_ASN1_SET_OF(PKCS7_SIGNER_INFO) */
#undef i2d_ASN1_SET_OF_PKCS7_SIGNER_INFO
#define i2d_ASN1_SET_OF_PKCS7_SIGNER_INFO       i2d_ASN1_SET_OF_PKCS7_SIGINF
#undef d2i_ASN1_SET_OF_PKCS7_SIGNER_INFO
#define d2i_ASN1_SET_OF_PKCS7_SIGNER_INFO       d2i_ASN1_SET_OF_PKCS7_SIGINF
#endif

#if 0 /* No longer needed, since safestack macro magic does the job */
/* Hack the names created with DECLARE_ASN1_SET_OF(PKCS7_RECIP_INFO) */
#undef i2d_ASN1_SET_OF_PKCS7_RECIP_INFO
#define i2d_ASN1_SET_OF_PKCS7_RECIP_INFO        i2d_ASN1_SET_OF_PKCS7_RECINF
#undef d2i_ASN1_SET_OF_PKCS7_RECIP_INFO
#define d2i_ASN1_SET_OF_PKCS7_RECIP_INFO        d2i_ASN1_SET_OF_PKCS7_RECINF
#endif

#if 0 /* No longer needed, since safestack macro magic does the job */
/* Hack the names created with DECLARE_ASN1_SET_OF(ACCESS_DESCRIPTION) */
#undef i2d_ASN1_SET_OF_ACCESS_DESCRIPTION
#define i2d_ASN1_SET_OF_ACCESS_DESCRIPTION      i2d_ASN1_SET_OF_ACC_DESC
#undef d2i_ASN1_SET_OF_ACCESS_DESCRIPTION
#define d2i_ASN1_SET_OF_ACCESS_DESCRIPTION      d2i_ASN1_SET_OF_ACC_DESC
#endif

/* Hack the names created with DECLARE_PEM_rw(NETSCAPE_CERT_SEQUENCE) */
#undef PEM_read_NETSCAPE_CERT_SEQUENCE
#define PEM_read_NETSCAPE_CERT_SEQUENCE         PEM_read_NS_CERT_SEQ
#undef PEM_write_NETSCAPE_CERT_SEQUENCE
#define PEM_write_NETSCAPE_CERT_SEQUENCE        PEM_write_NS_CERT_SEQ
#undef PEM_read_bio_NETSCAPE_CERT_SEQUENCE
#define PEM_read_bio_NETSCAPE_CERT_SEQUENCE     PEM_read_bio_NS_CERT_SEQ
#undef PEM_write_bio_NETSCAPE_CERT_SEQUENCE
#define PEM_write_bio_NETSCAPE_CERT_SEQUENCE    PEM_write_bio_NS_CERT_SEQ
#undef PEM_write_cb_bio_NETSCAPE_CERT_SEQUENCE
#define PEM_write_cb_bio_NETSCAPE_CERT_SEQUENCE PEM_write_cb_bio_NS_CERT_SEQ

/* Hack the names created with DECLARE_PEM_rw(PKCS8_PRIV_KEY_INFO) */
#undef PEM_read_PKCS8_PRIV_KEY_INFO
#define PEM_read_PKCS8_PRIV_KEY_INFO            PEM_read_P8_PRIV_KEY_INFO
#undef PEM_write_PKCS8_PRIV_KEY_INFO
#define PEM_write_PKCS8_PRIV_KEY_INFO           PEM_write_P8_PRIV_KEY_INFO
#undef PEM_read_bio_PKCS8_PRIV_KEY_INFO
#define PEM_read_bio_PKCS8_PRIV_KEY_INFO        PEM_read_bio_P8_PRIV_KEY_INFO
#undef PEM_write_bio_PKCS8_PRIV_KEY_INFO
#define PEM_write_bio_PKCS8_PRIV_KEY_INFO       PEM_write_bio_P8_PRIV_KEY_INFO
#undef PEM_write_cb_bio_PKCS8_PRIV_KEY_INFO
#define PEM_write_cb_bio_PKCS8_PRIV_KEY_INFO    PEM_wrt_cb_bio_P8_PRIV_KEY_INFO
```

```
/* Hack other PEM names */
#undef PEM_write_bio_PKCS8PrivateKey_nid
#define PEM_write_bio_PKCS8PrivateKey_nid       PEM_write_bio_PKCS8PrivKey_nid

/* Hack some long X509 names */
#undef X509_REVOKED_get_ext_by_critical
#define X509_REVOKED_get_ext_by_critical        X509_REVOKED_get_ext_by_critic
#undef X509_policy_tree_get0_user_policies
#define X509_policy_tree_get0_user_policies     X509_pcy_tree_get0_usr_policies
#undef X509_policy_node_get0_qualifiers
#define X509_policy_node_get0_qualifiers        X509_pcy_node_get0_qualifiers
#undef X509_STORE_CTX_get_explicit_policy
#define X509_STORE_CTX_get_explicit_policy      X509_STORE_CTX_get_expl_policy
#undef X509_STORE_CTX_get0_current_issuer
#define X509_STORE_CTX_get0_current_issuer      X509_STORE_CTX_get0_cur_issuer

/* Hack some long CRYPTO names */
#undef CRYPTO_set_dynlock_destroy_callback
#define CRYPTO_set_dynlock_destroy_callback     CRYPTO_set_dynlock_destroy_cb
#undef CRYPTO_set_dynlock_create_callback
#define CRYPTO_set_dynlock_create_callback      CRYPTO_set_dynlock_create_cb
#undef CRYPTO_set_dynlock_lock_callback
#define CRYPTO_set_dynlock_lock_callback        CRYPTO_set_dynlock_lock_cb
#undef CRYPTO_get_dynlock_lock_callback
#define CRYPTO_get_dynlock_lock_callback        CRYPTO_get_dynlock_lock_cb
#undef CRYPTO_get_dynlock_destroy_callback
#define CRYPTO_get_dynlock_destroy_callback     CRYPTO_get_dynlock_destroy_cb
#undef CRYPTO_get_dynlock_create_callback
#define CRYPTO_get_dynlock_create_callback      CRYPTO_get_dynlock_create_cb
#undef CRYPTO_set_locked_mem_ex_functions
#define CRYPTO_set_locked_mem_ex_functions      CRYPTO_set_locked_mem_ex_funcs
#undef CRYPTO_get_locked_mem_ex_functions
#define CRYPTO_get_locked_mem_ex_functions      CRYPTO_get_locked_mem_ex_funcs

/* Hack some long SSL names */
#undef SSL_CTX_set_default_verify_paths
#define SSL_CTX_set_default_verify_paths        SSL_CTX_set_def_verify_paths
#undef SSL_get_ex_data_X509_STORE_CTX_idx
#define SSL_get_ex_data_X509_STORE_CTX_idx      SSL_get_ex_d_X509_STORE_CTX_idx
#undef SSL_add_file_cert_subjects_to_stack
#define SSL_add_file_cert_subjects_to_stack     SSL_add_file_cert_subjs_to_stk
#undef SSL_add_dir_cert_subjects_to_stack
#define SSL_add_dir_cert_subjects_to_stack      SSL_add_dir_cert_subjs_to_stk
#undef SSL_CTX_use_certificate_chain_file
#define SSL_CTX_use_certificate_chain_file      SSL_CTX_use_cert_chain_file
#undef SSL_CTX_set_cert_verify_callback
#define SSL_CTX_set_cert_verify_callback        SSL_CTX_set_cert_verify_cb
#undef SSL_CTX_set_default_passwd_cb_userdata
#define SSL_CTX_set_default_passwd_cb_userdata  SSL_CTX_set_def_passwd_cb_ud
#undef SSL_COMP_get_compression_methods
#define SSL_COMP_get_compression_methods        SSL_COMP_get_compress_methods
#undef ssl_add_clienthello_renegotiate_ext
#define ssl_add_clienthello_renegotiate_ext     ssl_add_clienthello_reneg_ext
#undef ssl_add_serverhello_renegotiate_ext
#define ssl_add_serverhello_renegotiate_ext     ssl_add_serverhello_reneg_ext
#undef ssl_parse_clienthello_renegotiate_ext
#define ssl_parse_clienthello_renegotiate_ext   ssl_parse_clienthello_reneg_ext
#undef ssl_parse_serverhello_renegotiate_ext
#define ssl_parse_serverhello_renegotiate_ext   ssl_parse_serverhello_reneg_ext
#undef SSL_srp_server_param_with_username
#define SSL_srp_server_param_with_username      SSL_srp_server_param_with_un
#undef SSL_CTX_set_srp_client_pwd_callback
#define SSL_CTX_set_srp_client_pwd_callback     SSL_CTX_set_srp_client_pwd_cb
#undef SSL_CTX_set_srp_verify_param_callback
#define SSL_CTX_set_srp_verify_param_callback   SSL_CTX_set_srp_vfy_param_cb
#undef SSL_CTX_set_srp_username_callback
```



```
#define SSL_CTX_set_srp_username_callback       SSL_CTX_set_srp_un_cb
#undef ssl_add_clienthello_use_srtp_ext
#define ssl_add_clienthello_use_srtp_ext        ssl_add_clihello_use_srtp_ext
#undef ssl_add_serverhello_use_srtp_ext
#define ssl_add_serverhello_use_srtp_ext        ssl_add_serhello_use_srtp_ext
#undef ssl_parse_clienthello_use_srtp_ext
#define ssl_parse_clienthello_use_srtp_ext      ssl_parse_clihello_use_srtp_ext
#undef ssl_parse_serverhello_use_srtp_ext
#define ssl_parse_serverhello_use_srtp_ext      ssl_parse_serhello_use_srtp_ext
#undef SSL_CTX_set_next_protos_advertised_cb
#define SSL_CTX_set_next_protos_advertised_cb   SSL_CTX_set_next_protos_adv_cb
#undef SSL_CTX_set_next_proto_select_cb
#define SSL_CTX_set_next_proto_select_cb        SSL_CTX_set_next_proto_sel_cb
#undef ssl3_cbc_record_digest_supported
#define ssl3_cbc_record_digest_supported        ssl3_cbc_record_digest_support
#undef ssl_check_clienthello_tlsext_late
#define ssl_check_clienthello_tlsext_late       ssl_check_clihello_tlsext_late
#undef ssl_check_clienthello_tlsext_early
#define ssl_check_clienthello_tlsext_early      ssl_check_clihello_tlsext_early

/* Hack some long ENGINE names */
#undef ENGINE_get_default_BN_mod_exp_crt
#define ENGINE_get_default_BN_mod_exp_crt       ENGINE_get_def_BN_mod_exp_crt
#undef ENGINE_set_default_BN_mod_exp_crt
#define ENGINE_set_default_BN_mod_exp_crt       ENGINE_set_def_BN_mod_exp_crt
#undef ENGINE_set_load_privkey_function
#define ENGINE_set_load_privkey_function        ENGINE_set_load_privkey_fn
#undef ENGINE_get_load_privkey_function
#define ENGINE_get_load_privkey_function        ENGINE_get_load_privkey_fn
#undef ENGINE_unregister_pkey_asn1_meths
#define ENGINE_unregister_pkey_asn1_meths       ENGINE_unreg_pkey_asn1_meths
#undef ENGINE_register_all_pkey_asn1_meths
#define ENGINE_register_all_pkey_asn1_meths     ENGINE_reg_all_pkey_asn1_meths
#undef ENGINE_set_default_pkey_asn1_meths
#define ENGINE_set_default_pkey_asn1_meths      ENGINE_set_def_pkey_asn1_meths
#undef ENGINE_get_pkey_asn1_meth_engine
#define ENGINE_get_pkey_asn1_meth_engine        ENGINE_get_pkey_asn1_meth_eng
#undef ENGINE_set_load_ssl_client_cert_function
#define ENGINE_set_load_ssl_client_cert_function \
                                                ENGINE_set_ld_ssl_clnt_cert_fn
#undef ENGINE_get_ssl_client_cert_function
#define ENGINE_get_ssl_client_cert_function     ENGINE_get_ssl_client_cert_fn

/* Hack some long OCSP names */
#undef OCSP_REQUEST_get_ext_by_critical
#define OCSP_REQUEST_get_ext_by_critical        OCSP_REQUEST_get_ext_by_crit
#undef OCSP_BASICRESP_get_ext_by_critical
#define OCSP_BASICRESP_get_ext_by_critical      OCSP_BASICRESP_get_ext_by_crit
#undef OCSP_SINGLERESP_get_ext_by_critical
#define OCSP_SINGLERESP_get_ext_by_critical     OCSP_SINGLERESP_get_ext_by_crit

/* Hack some long DES names */
#undef _ossl_old_des_ede3_cfb64_encrypt
#define _ossl_old_des_ede3_cfb64_encrypt        _ossl_odes_ede3_cfb64_encrypt
#undef _ossl_old_des_ede3_ofb64_encrypt
#define _ossl_old_des_ede3_ofb64_encrypt        _ossl_odes_ede3_ofb64_encrypt

/* Hack some long EVP names */
#undef OPENSSL_add_all_algorithms_noconf
#define OPENSSL_add_all_algorithms_noconf       OPENSSL_add_all_algo_noconf
#undef OPENSSL_add_all_algorithms_conf
#define OPENSSL_add_all_algorithms_conf         OPENSSL_add_all_algo_conf
#undef EVP_PKEY_meth_set_verify_recover
#define EVP_PKEY_meth_set_verify_recover        EVP_PKEY_meth_set_vrfy_recover

/* Hack some long EC names */
```

```
#undef EC_GROUP_set_point_conversion_form
#define EC_GROUP_set_point_conversion_form      EC_GROUP_set_point_conv_form
#undef EC_GROUP_get_point_conversion_form
#define EC_GROUP_get_point_conversion_form      EC_GROUP_get_point_conv_form
#undef EC_GROUP_clear_free_all_extra_data
#define EC_GROUP_clear_free_all_extra_data      EC_GROUP_clr_free_all_xtra_data
#undef EC_KEY_set_public_key_affine_coordinates
#define EC_KEY_set_public_key_affine_coordinates \
                                                EC_KEY_set_pub_key_aff_coords
#undef EC_POINT_set_Jprojective_coordinates_GFp
#define EC_POINT_set_Jprojective_coordinates_GFp \
                                                EC_POINT_set_Jproj_coords_GFp
#undef EC_POINT_get_Jprojective_coordinates_GFp
#define EC_POINT_get_Jprojective_coordinates_GFp \
                                                EC_POINT_get_Jproj_coords_GFp
#undef EC_POINT_set_affine_coordinates_GFp
#define EC_POINT_set_affine_coordinates_GFp     EC_POINT_set_affine_coords_GFp
#undef EC_POINT_get_affine_coordinates_GFp
#define EC_POINT_get_affine_coordinates_GFp     EC_POINT_get_affine_coords_GFp
#undef EC_POINT_set_compressed_coordinates_GFp
#define EC_POINT_set_compressed_coordinates_GFp EC_POINT_set_compr_coords_GFp
#undef EC_POINT_set_affine_coordinates_GF2m
#define EC_POINT_set_affine_coordinates_GF2m    EC_POINT_set_affine_coords_GF2m
#undef EC_POINT_get_affine_coordinates_GF2m
#define EC_POINT_get_affine_coordinates_GF2m    EC_POINT_get_affine_coords_GF2m
#undef EC_POINT_set_compressed_coordinates_GF2m
#define EC_POINT_set_compressed_coordinates_GF2m \
                                                EC_POINT_set_compr_coords_GF2m
#undef ec_GF2m_simple_group_clear_finish
#define ec_GF2m_simple_group_clear_finish       ec_GF2m_simple_grp_clr_finish
#undef ec_GF2m_simple_group_check_discriminant
#define ec_GF2m_simple_group_check_discriminant ec_GF2m_simple_grp_chk_discrim
#undef ec_GF2m_simple_point_clear_finish
#define ec_GF2m_simple_point_clear_finish       ec_GF2m_simple_pt_clr_finish
#undef ec_GF2m_simple_point_set_to_infinity
#define ec_GF2m_simple_point_set_to_infinity    ec_GF2m_simple_pt_set_to_inf
#undef ec_GF2m_simple_points_make_affine
#define ec_GF2m_simple_points_make_affine       ec_GF2m_simple_pts_make_affine
#undef ec_GF2m_simple_point_set_affine_coordinates
#define ec_GF2m_simple_point_set_affine_coordinates \
                                                ec_GF2m_smp_pt_set_af_coords
#undef ec_GF2m_simple_point_get_affine_coordinates
#define ec_GF2m_simple_point_get_affine_coordinates \
                                                ec_GF2m_smp_pt_get_af_coords
#undef ec_GF2m_simple_set_compressed_coordinates
#define ec_GF2m_simple_set_compressed_coordinates \
                                                ec_GF2m_smp_set_compr_coords
#undef ec_GFp_simple_group_set_curve_GFp
#define ec_GFp_simple_group_set_curve_GFp       ec_GFp_simple_grp_set_curve_GFp
#undef ec_GFp_simple_group_get_curve_GFp
#define ec_GFp_simple_group_get_curve_GFp       ec_GFp_simple_grp_get_curve_GFp
#undef ec_GFp_simple_group_clear_finish
#define ec_GFp_simple_group_clear_finish        ec_GFp_simple_grp_clear_finish
#undef ec_GFp_simple_group_set_generator
#define ec_GFp_simple_group_set_generator       ec_GFp_simple_grp_set_generator
#undef ec_GFp_simple_group_get0_generator
#define ec_GFp_simple_group_get0_generator      ec_GFp_simple_grp_gt0_generator
#undef ec_GFp_simple_group_get_cofactor
#define ec_GFp_simple_group_get_cofactor        ec_GFp_simple_grp_get_cofactor
#undef ec_GFp_simple_point_clear_finish
#define ec_GFp_simple_point_clear_finish        ec_GFp_simple_pt_clear_finish
#undef ec_GFp_simple_point_set_to_infinity
#define ec_GFp_simple_point_set_to_infinity     ec_GFp_simple_pt_set_to_inf
#undef ec_GFp_simple_points_make_affine
#define ec_GFp_simple_points_make_affine        ec_GFp_simple_pts_make_affine
#undef ec_GFp_simple_set_Jprojective_coordinates_GFp
```



```
#define ec_GFp_simple_set_Jprojective_coordinates_GFp \
                                                ec_GFp_smp_set_Jproj_coords_GFp
#undef ec_GFp_simple_get_Jprojective_coordinates_GFp
#define ec_GFp_simple_get_Jprojective_coordinates_GFp \
                                                ec_GFp_smp_get_Jproj_coords_GFp
#undef ec_GFp_simple_point_set_affine_coordinates_GFp
#define ec_GFp_simple_point_set_affine_coordinates_GFp \
                                                ec_GFp_smp_pt_set_af_coords_GFp
#undef ec_GFp_simple_point_get_affine_coordinates_GFp
#define ec_GFp_simple_point_get_affine_coordinates_GFp \
                                                ec_GFp_smp_pt_get_af_coords_GFp
#undef ec_GFp_simple_set_compressed_coordinates_GFp
#define ec_GFp_simple_set_compressed_coordinates_GFp \
                                                ec_GFp_smp_set_compr_coords_GFp
#undef ec_GFp_simple_point_set_affine_coordinates
#define ec_GFp_simple_point_set_affine_coordinates \
                                                ec_GFp_smp_pt_set_af_coords
#undef ec_GFp_simple_point_get_affine_coordinates
#define ec_GFp_simple_point_get_affine_coordinates \
                                                ec_GFp_smp_pt_get_af_coords
#undef ec_GFp_simple_set_compressed_coordinates
#define ec_GFp_simple_set_compressed_coordinates \
                                                ec_GFp_smp_set_compr_coords
#undef ec_GFp_simple_group_check_discriminant
#define ec_GFp_simple_group_check_discriminant  ec_GFp_simple_grp_chk_discrim

/* Hack som long STORE names */
#undef STORE_method_set_initialise_function
#define STORE_method_set_initialise_function    STORE_meth_set_initialise_fn
#undef STORE_method_set_cleanup_function
#define STORE_method_set_cleanup_function       STORE_meth_set_cleanup_fn
#undef STORE_method_set_generate_function
#define STORE_method_set_generate_function      STORE_meth_set_generate_fn
#undef STORE_method_set_modify_function
#define STORE_method_set_modify_function        STORE_meth_set_modify_fn
#undef STORE_method_set_revoke_function
#define STORE_method_set_revoke_function        STORE_meth_set_revoke_fn
#undef STORE_method_set_delete_function
#define STORE_method_set_delete_function        STORE_meth_set_delete_fn
#undef STORE_method_set_list_start_function
#define STORE_method_set_list_start_function    STORE_meth_set_list_start_fn
#undef STORE_method_set_list_next_function
#define STORE_method_set_list_next_function     STORE_meth_set_list_next_fn
#undef STORE_method_set_list_end_function
#define STORE_method_set_list_end_function      STORE_meth_set_list_end_fn
#undef STORE_method_set_update_store_function
#define STORE_method_set_update_store_function  STORE_meth_set_update_store_fn
#undef STORE_method_set_lock_store_function
#define STORE_method_set_lock_store_function    STORE_meth_set_lock_store_fn
#undef STORE_method_set_unlock_store_function
#define STORE_method_set_unlock_store_function  STORE_meth_set_unlock_store_fn
#undef STORE_method_get_initialise_function
#define STORE_method_get_initialise_function    STORE_meth_get_initialise_fn
#undef STORE_method_get_cleanup_function
#define STORE_method_get_cleanup_function       STORE_meth_get_cleanup_fn
#undef STORE_method_get_generate_function
#define STORE_method_get_generate_function      STORE_meth_get_generate_fn
#undef STORE_method_get_modify_function
#define STORE_method_get_modify_function        STORE_meth_get_modify_fn
#undef STORE_method_get_revoke_function
#define STORE_method_get_revoke_function        STORE_meth_get_revoke_fn
#undef STORE_method_get_delete_function
#define STORE_method_get_delete_function        STORE_meth_get_delete_fn
#undef STORE_method_get_list_start_function
#define STORE_method_get_list_start_function    STORE_meth_get_list_start_fn
#undef STORE_method_get_list_next_function
```

```
#define STORE_method_get_list_next_function     STORE_meth_get_list_next_fn
#undef STORE_method_get_list_end_function
#define STORE_method_get_list_end_function      STORE_meth_get_list_end_fn
#undef STORE_method_get_update_store_function
#define STORE_method_get_update_store_function  STORE_meth_get_update_store_fn
#undef STORE_method_get_lock_store_function
#define STORE_method_get_lock_store_function    STORE_meth_get_lock_store_fn
#undef STORE_method_get_unlock_store_function
#define STORE_method_get_unlock_store_function  STORE_meth_get_unlock_store_fn

/* Hack some long TS names */
#undef TS_RESP_CTX_set_status_info_cond
#define TS_RESP_CTX_set_status_info_cond        TS_RESP_CTX_set_stat_info_cond
#undef TS_RESP_CTX_set_clock_precision_digits
#define TS_RESP_CTX_set_clock_precision_digits  TS_RESP_CTX_set_clk_prec_digits
#undef TS_CONF_set_clock_precision_digits
#define TS_CONF_set_clock_precision_digits      TS_CONF_set_clk_prec_digits

/* Hack some long CMS names */
#undef CMS_RecipientInfo_ktri_get0_algs
#define CMS_RecipientInfo_ktri_get0_algs        CMS_RecipInfo_ktri_get0_algs
#undef CMS_RecipientInfo_ktri_get0_signer_id
#define CMS_RecipientInfo_ktri_get0_signer_id   CMS_RecipInfo_ktri_get0_sigr_id
#undef CMS_OtherRevocationInfoFormat_it
#define CMS_OtherRevocationInfoFormat_it        CMS_OtherRevocInfoFormat_it
#undef CMS_KeyAgreeRecipientIdentifier_it
#define CMS_KeyAgreeRecipientIdentifier_it      CMS_KeyAgreeRecipIdentifier_it
#undef CMS_OriginatorIdentifierOrKey_it
#define CMS_OriginatorIdentifierOrKey_it        CMS_OriginatorIdOrKey_it
#undef cms_SignerIdentifier_get0_signer_id
#define cms_SignerIdentifier_get0_signer_id     cms_SignerId_get0_signer_id

/* Hack some long DTLS1 names */
#undef dtls1_retransmit_buffered_messages
#define dtls1_retransmit_buffered_messages      dtls1_retransmit_buffered_msgs

/* Hack some long SRP names */
#undef SRP_generate_server_master_secret
#define SRP_generate_server_master_secret       SRP_gen_server_master_secret
#undef SRP_generate_client_master_secret
#define SRP_generate_client_master_secret       SRP_gen_client_master_secret

/* Hack some long UI names */
#undef UI_method_get_prompt_constructor
#define UI_method_get_prompt_constructor        UI_method_get_prompt_constructr
#undef UI_method_set_prompt_constructor
#define UI_method_set_prompt_constructor        UI_method_set_prompt_constructr

#endif /* defined OPENSSL_SYS_VMS */


/* Case insensitive linking causes problems.... */
#if defined(OPENSSL_SYS_VMS) || defined(OPENSSL_SYS_OS2)
#undef ERR_load_CRYPTO_strings
#define ERR_load_CRYPTO_strings                 ERR_load_CRYPTOlib_strings
#undef OCSP_crlID_new
#define OCSP_crlID_new                          OCSP_crlID2_new

#undef d2i_ECPARAMETERS
#define d2i_ECPARAMETERS                        d2i_UC_ECPARAMETERS
#undef i2d_ECPARAMETERS
#define i2d_ECPARAMETERS                        i2d_UC_ECPARAMETERS
#undef d2i_ECPKPARAMETERS
#define d2i_ECPKPARAMETERS                      d2i_UC_ECPKPARAMETERS
#undef i2d_ECPKPARAMETERS
#define i2d_ECPKPARAMETERS                      i2d_UC_ECPKPARAMETERS
```



```
/* These functions do not seem to exist!  However, I'm paranoid...
   Original command in x509v3.h:
   These functions are being redefined in another directory,
   and clash when the linker is case-insensitive, so let's
   hide them a little, by giving them an extra 'o' at the
   beginning of the name... */
#undef X509v3_cleanup_extensions
#define X509v3_cleanup_extensions               oX509v3_cleanup_extensions
#undef X509v3_add_extension
#define X509v3_add_extension                    oX509v3_add_extension
#undef X509v3_add_netscape_extensions
#define X509v3_add_netscape_extensions          oX509v3_add_netscape_extensions
#undef X509v3_add_standard_extensions
#define X509v3_add_standard_extensions          oX509v3_add_standard_extensions

/* This one clashes with CMS_data_create */
#undef cms_Data_create
#define cms_Data_create                         priv_cms_Data_create

#endif


#endif /* ! defined HEADER_VMS_IDHACKS_H */
#endif /* ! codereview */
```

```
**********************************************************
   35284 Wed Aug 13 19:51:50 2014
new/usr/src/lib/openssl/include/openssl/tls1.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/tls1.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * Portions of the attached software ("Contribution") are developed by
 * SUN MICROSYSTEMS, INC., and are contributed to the OpenSSL project.
 *
 * The Contribution is licensed pursuant to the OpenSSL open source
 * license provided above.
 *
 * ECC cipher suite support in OpenSSL originally written by
 * Vipul Gupta and Sumit Gupta of Sun Microsystems Laboratories.
 *
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
```

```
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#ifndef HEADER_TLS1_H
#define HEADER_TLS1_H

#include <openssl/buffer.h>

#ifdef  __cplusplus
extern "C" {
#endif

#define TLS1_ALLOW_EXPERIMENTAL_CIPHERSUITES    0

#define TLS1_2_VERSION                  0x0303
#define TLS1_2_VERSION_MAJOR            0x03
#define TLS1_2_VERSION_MINOR            0x03

#define TLS1_1_VERSION                  0x0302
#define TLS1_1_VERSION_MAJOR            0x03
#define TLS1_1_VERSION_MINOR            0x02

#define TLS1_VERSION                    0x0301
#define TLS1_VERSION_MAJOR              0x03
#define TLS1_VERSION_MINOR              0x01

#define TLS1_get_version(s) \
                ((s->version >> 8) == TLS1_VERSION_MAJOR ? s->version : 0)

#define TLS1_get_client_version(s) \
                ((s->client_version >> 8) == TLS1_VERSION_MAJOR ? s->client_vers

#define TLS1_AD_DECRYPTION_FAILED       21
#define TLS1_AD_RECORD_OVERFLOW         22
#define TLS1_AD_UNKNOWN_CA              48      /* fatal */
#define TLS1_AD_ACCESS_DENIED           49      /* fatal */
#define TLS1_AD_DECODE_ERROR            50      /* fatal */
#define TLS1_AD_DECRYPT_ERROR           51
#define TLS1_AD_EXPORT_RESTRICTION      60      /* fatal */
#define TLS1_AD_PROTOCOL_VERSION        70      /* fatal */
#define TLS1_AD_INSUFFICIENT_SECURITY   71      /* fatal */
#define TLS1_AD_INTERNAL_ERROR          80      /* fatal */
#define TLS1_AD_USER_CANCELLED          90
#define TLS1_AD_NO_RENEGOTIATION        100
/* codes 110-114 are from RFC3546 */
#define TLS1_AD_UNSUPPORTED_EXTENSION   110
```



```
#define TLS1_AD_CERTIFICATE_UNOBTAINABLE 111
#define TLS1_AD_UNRECOGNIZED_NAME       112
#define TLS1_AD_BAD_CERTIFICATE_STATUS_RESPONSE 113
#define TLS1_AD_BAD_CERTIFICATE_HASH_VALUE 114
#define TLS1_AD_UNKNOWN_PSK_IDENTITY    115     /* fatal */

/* ExtensionType values from RFC3546 / RFC4366 / RFC6066 */
#define TLSEXT_TYPE_server_name                 0
#define TLSEXT_TYPE_max_fragment_length         1
#define TLSEXT_TYPE_client_certificate_url      2
#define TLSEXT_TYPE_trusted_ca_keys             3
#define TLSEXT_TYPE_truncated_hmac              4
#define TLSEXT_TYPE_status_request              5
/* ExtensionType values from RFC4681 */
#define TLSEXT_TYPE_user_mapping                6

/* ExtensionType values from RFC5878 */
#define TLSEXT_TYPE_client_authz                7
#define TLSEXT_TYPE_server_authz                8

/* ExtensionType values from RFC6091 */
#define TLSEXT_TYPE_cert_type           9

/* ExtensionType values from RFC4492 */
#define TLSEXT_TYPE_elliptic_curves             10
#define TLSEXT_TYPE_ec_point_formats            11

/* ExtensionType value from RFC5054 */
#define TLSEXT_TYPE_srp                         12

/* ExtensionType values from RFC5246 */
#define TLSEXT_TYPE_signature_algorithms        13

/* ExtensionType value from RFC5764 */
#define TLSEXT_TYPE_use_srtp    14

/* ExtensionType value from RFC5620 */
#define TLSEXT_TYPE_heartbeat   15

/* ExtensionType value for TLS padding extension.
 * http://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-va
 * http://tools.ietf.org/html/draft-agl-tls-padding-03
 */
#define TLSEXT_TYPE_padding     21

/* ExtensionType value from RFC4507 */
#define TLSEXT_TYPE_session_ticket              35

/* ExtensionType value from draft-rescorla-tls-opaque-prf-input-00.txt */
#if 0 /* will have to be provided externally for now ,
       * i.e. build with -DTLSEXT_TYPE_opaque_prf_input=38183
       * using whatever extension number you'd like to try */
# define TLSEXT_TYPE_opaque_prf_input           ?? */
#endif

/* Temporary extension type */
#define TLSEXT_TYPE_renegotiate                 0xff01

#ifndef OPENSSL_NO_NEXTPROTONEG
/* This is not an IANA defined extension number */
#define TLSEXT_TYPE_next_proto_neg              13172
#endif

/* NameType value from RFC 3546 */
#define TLSEXT_NAMETYPE_host_name 0
/* status request value from RFC 3546 */
```

```
#define TLSEXT_STATUSTYPE_ocsp 1

/* ECPointFormat values from draft-ietf-tls-ecc-12 */
#define TLSEXT_ECPOINTFORMAT_first                      0
#define TLSEXT_ECPOINTFORMAT_uncompressed               0
#define TLSEXT_ECPOINTFORMAT_ansiX962_compressed_prime  1
#define TLSEXT_ECPOINTFORMAT_ansiX962_compressed_char2  2
#define TLSEXT_ECPOINTFORMAT_last                       2

/* Signature and hash algorithms from RFC 5246 */

#define TLSEXT_signature_anonymous                      0
#define TLSEXT_signature_rsa                            1
#define TLSEXT_signature_dsa                            2
#define TLSEXT_signature_ecdsa                          3

#define TLSEXT_hash_none                                0
#define TLSEXT_hash_md5                                 1
#define TLSEXT_hash_sha1                                2
#define TLSEXT_hash_sha224                              3
#define TLSEXT_hash_sha256                              4
#define TLSEXT_hash_sha384                              5
#define TLSEXT_hash_sha512                              6

#ifndef OPENSSL_NO_TLSEXT

#define TLSEXT_MAXLEN_host_name 255

const char *SSL_get_servername(const SSL *s, const int type);
int SSL_get_servername_type(const SSL *s);
/* SSL_export_keying_material exports a value derived from the master secret,
 * as specified in RFC 5705. It writes |olen| bytes to |out| given a label and
 * optional context. (Since a zero length context is allowed, the |use_context|
 * flag controls whether a context is included.)
 *
 * It returns 1 on success and zero otherwise.
 */
int SSL_export_keying_material(SSL *s, unsigned char *out, size_t olen,
        const char *label, size_t llen, const unsigned char *p, size_t plen,
        int use_context);

#define SSL_set_tlsext_host_name(s,name) \
SSL_ctrl(s,SSL_CTRL_SET_TLSEXT_HOSTNAME,TLSEXT_NAMETYPE_host_name,(char *)name)

#define SSL_set_tlsext_debug_callback(ssl, cb) \
SSL_callback_ctrl(ssl,SSL_CTRL_SET_TLSEXT_DEBUG_CB,(void (*)(void))cb)

#define SSL_set_tlsext_debug_arg(ssl, arg) \
SSL_ctrl(ssl,SSL_CTRL_SET_TLSEXT_DEBUG_ARG,0, (void *)arg)

#define SSL_set_tlsext_status_type(ssl, type) \
SSL_ctrl(ssl,SSL_CTRL_SET_TLSEXT_STATUS_REQ_TYPE,type, NULL)

#define SSL_get_tlsext_status_exts(ssl, arg) \
SSL_ctrl(ssl,SSL_CTRL_GET_TLSEXT_STATUS_REQ_EXTS,0, (void *)arg)

#define SSL_set_tlsext_status_exts(ssl, arg) \
SSL_ctrl(ssl,SSL_CTRL_SET_TLSEXT_STATUS_REQ_EXTS,0, (void *)arg)

#define SSL_get_tlsext_status_ids(ssl, arg) \
SSL_ctrl(ssl,SSL_CTRL_GET_TLSEXT_STATUS_REQ_IDS,0, (void *)arg)

#define SSL_set_tlsext_status_ids(ssl, arg) \
SSL_ctrl(ssl,SSL_CTRL_SET_TLSEXT_STATUS_REQ_IDS,0, (void *)arg)

#define SSL_get_tlsext_status_ocsp_resp(ssl, arg) \
```



```
SSL_ctrl(ssl,SSL_CTRL_GET_TLSEXT_STATUS_REQ_OCSP_RESP,0, (void *)arg)

#define SSL_set_tlsext_status_ocsp_resp(ssl, arg, arglen) \
SSL_ctrl(ssl,SSL_CTRL_SET_TLSEXT_STATUS_REQ_OCSP_RESP,arglen, (void *)arg)

#define SSL_CTX_set_tlsext_servername_callback(ctx, cb) \
SSL_CTX_callback_ctrl(ctx,SSL_CTRL_SET_TLSEXT_SERVERNAME_CB,(void (*)(void))cb)

#define SSL_TLSEXT_ERR_OK 0
#define SSL_TLSEXT_ERR_ALERT_WARNING 1
#define SSL_TLSEXT_ERR_ALERT_FATAL 2
#define SSL_TLSEXT_ERR_NOACK 3

#define SSL_CTX_set_tlsext_servername_arg(ctx, arg) \
SSL_CTX_ctrl(ctx,SSL_CTRL_SET_TLSEXT_SERVERNAME_ARG,0, (void *)arg)

#define SSL_CTX_get_tlsext_ticket_keys(ctx, keys, keylen) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_GET_TLSEXT_TICKET_KEYS,(keylen),(keys))
#define SSL_CTX_set_tlsext_ticket_keys(ctx, keys, keylen) \
        SSL_CTX_ctrl((ctx),SSL_CTRL_SET_TLSEXT_TICKET_KEYS,(keylen),(keys))

#define SSL_CTX_set_tlsext_status_cb(ssl, cb) \
SSL_CTX_callback_ctrl(ssl,SSL_CTRL_SET_TLSEXT_STATUS_REQ_CB,(void (*)(void))cb)

#define SSL_CTX_set_tlsext_status_arg(ssl, arg) \
SSL_CTX_ctrl(ssl,SSL_CTRL_SET_TLSEXT_STATUS_REQ_CB_ARG,0, (void *)arg)

#define SSL_set_tlsext_opaque_prf_input(s, src, len) \
SSL_ctrl(s,SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT, len, src)
#define SSL_CTX_set_tlsext_opaque_prf_input_callback(ctx, cb) \
SSL_CTX_callback_ctrl(ctx,SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT_CB, (void (*)(voi
#define SSL_CTX_set_tlsext_opaque_prf_input_callback_arg(ctx, arg) \
SSL_CTX_ctrl(ctx,SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT_CB_ARG, 0, arg)

#define SSL_CTX_set_tlsext_ticket_key_cb(ssl, cb) \
SSL_CTX_callback_ctrl(ssl,SSL_CTRL_SET_TLSEXT_TICKET_KEY_CB,(void (*)(void))cb)

#ifndef OPENSSL_NO_HEARTBEATS
#define SSL_TLSEXT_HB_ENABLED                           0x01
#define SSL_TLSEXT_HB_DONT_SEND_REQUESTS        0x02
#define SSL_TLSEXT_HB_DONT_RECV_REQUESTS        0x04

#define SSL_get_tlsext_heartbeat_pending(ssl) \
        SSL_ctrl((ssl),SSL_CTRL_GET_TLS_EXT_HEARTBEAT_PENDING,0,NULL)
#define SSL_set_tlsext_heartbeat_no_requests(ssl, arg) \
        SSL_ctrl((ssl),SSL_CTRL_SET_TLS_EXT_HEARTBEAT_NO_REQUESTS,arg,NULL)
#endif
#endif

/* PSK ciphersuites from 4279 */
#define TLS1_CK_PSK_WITH_RC4_128_SHA                    0x0300008A
#define TLS1_CK_PSK_WITH_3DES_EDE_CBC_SHA               0x0300008B
#define TLS1_CK_PSK_WITH_AES_128_CBC_SHA                0x0300008C
#define TLS1_CK_PSK_WITH_AES_256_CBC_SHA                0x0300008D

/* Additional TLS ciphersuites from expired Internet Draft
 * draft-ietf-tls-56-bit-ciphersuites-01.txt
 * (available if TLS1_ALLOW_EXPERIMENTAL_CIPHERSUITES is defined, see
 * s3_lib.c).  We actually treat them like SSL 3.0 ciphers, which we probably
 * shouldn't.  Note that the first two are actually not in the IDs. */
#define TLS1_CK_RSA_EXPORT1024_WITH_RC4_56_MD5          0x03000060 /* not in ID
#define TLS1_CK_RSA_EXPORT1024_WITH_RC2_CBC_56_MD5      0x03000061 /* not in ID
#define TLS1_CK_RSA_EXPORT1024_WITH_DES_CBC_SHA         0x03000062
#define TLS1_CK_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA     0x03000063
#define TLS1_CK_RSA_EXPORT1024_WITH_RC4_56_SHA          0x03000064
#define TLS1_CK_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA      0x03000065
```

```
392 #define TLS1_CK_DHE_DSS_WITH_RC4_128_SHA                0x03000066

394 /* AES ciphersuites from RFC3268 */

396 #define TLS1_CK_RSA_WITH_AES_128_SHA                    0x0300002F
397 #define TLS1_CK_DH_DSS_WITH_AES_128_SHA                 0x03000030
398 #define TLS1_CK_DH_RSA_WITH_AES_128_SHA                 0x03000031
399 #define TLS1_CK_DHE_DSS_WITH_AES_128_SHA                0x03000032
400 #define TLS1_CK_DHE_RSA_WITH_AES_128_SHA                0x03000033
401 #define TLS1_CK_ADH_WITH_AES_128_SHA                    0x03000034

403 #define TLS1_CK_RSA_WITH_AES_256_SHA                    0x03000035
404 #define TLS1_CK_DH_DSS_WITH_AES_256_SHA                 0x03000036
405 #define TLS1_CK_DH_RSA_WITH_AES_256_SHA                 0x03000037
406 #define TLS1_CK_DHE_DSS_WITH_AES_256_SHA                0x03000038
407 #define TLS1_CK_DHE_RSA_WITH_AES_256_SHA                0x03000039
408 #define TLS1_CK_ADH_WITH_AES_256_SHA                    0x0300003A

410 /* TLS v1.2 ciphersuites */
411 #define TLS1_CK_RSA_WITH_NULL_SHA256                    0x0300003B
412 #define TLS1_CK_RSA_WITH_AES_128_SHA256                 0x0300003C
413 #define TLS1_CK_RSA_WITH_AES_256_SHA256                 0x0300003D
414 #define TLS1_CK_DH_DSS_WITH_AES_128_SHA256              0x0300003E
415 #define TLS1_CK_DH_RSA_WITH_AES_128_SHA256              0x0300003F
416 #define TLS1_CK_DHE_DSS_WITH_AES_128_SHA256             0x03000040

418 /* Camellia ciphersuites from RFC4132 */
419 #define TLS1_CK_RSA_WITH_CAMELLIA_128_CBC_SHA           0x03000041
420 #define TLS1_CK_DH_DSS_WITH_CAMELLIA_128_CBC_SHA        0x03000042
421 #define TLS1_CK_DH_RSA_WITH_CAMELLIA_128_CBC_SHA        0x03000043
422 #define TLS1_CK_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA       0x03000044
423 #define TLS1_CK_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA       0x03000045
424 #define TLS1_CK_ADH_WITH_CAMELLIA_128_CBC_SHA           0x03000046

426 /* TLS v1.2 ciphersuites */
427 #define TLS1_CK_DHE_RSA_WITH_AES_128_SHA256             0x03000067
428 #define TLS1_CK_DH_DSS_WITH_AES_256_SHA256              0x03000068
429 #define TLS1_CK_DH_RSA_WITH_AES_256_SHA256              0x03000069
430 #define TLS1_CK_DHE_DSS_WITH_AES_256_SHA256             0x0300006A
431 #define TLS1_CK_DHE_RSA_WITH_AES_256_SHA256             0x0300006B
432 #define TLS1_CK_ADH_WITH_AES_128_SHA256                 0x0300006C
433 #define TLS1_CK_ADH_WITH_AES_256_SHA256                 0x0300006D

435 /* Camellia ciphersuites from RFC4132 */
436 #define TLS1_CK_RSA_WITH_CAMELLIA_256_CBC_SHA           0x03000084
437 #define TLS1_CK_DH_DSS_WITH_CAMELLIA_256_CBC_SHA        0x03000085
438 #define TLS1_CK_DH_RSA_WITH_CAMELLIA_256_CBC_SHA        0x03000086
439 #define TLS1_CK_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA       0x03000087
440 #define TLS1_CK_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA       0x03000088
441 #define TLS1_CK_ADH_WITH_CAMELLIA_256_CBC_SHA           0x03000089

443 /* SEED ciphersuites from RFC4162 */
444 #define TLS1_CK_RSA_WITH_SEED_SHA                       0x03000096
445 #define TLS1_CK_DH_DSS_WITH_SEED_SHA                    0x03000097
446 #define TLS1_CK_DH_RSA_WITH_SEED_SHA                    0x03000098
447 #define TLS1_CK_DHE_DSS_WITH_SEED_SHA                   0x03000099
448 #define TLS1_CK_DHE_RSA_WITH_SEED_SHA                   0x0300009A
449 #define TLS1_CK_ADH_WITH_SEED_SHA                       0x0300009B

451 /* TLS v1.2 GCM ciphersuites from RFC5288 */
452 #define TLS1_CK_RSA_WITH_AES_128_GCM_SHA256             0x0300009C
453 #define TLS1_CK_RSA_WITH_AES_256_GCM_SHA384             0x0300009D
454 #define TLS1_CK_DHE_RSA_WITH_AES_128_GCM_SHA256         0x0300009E
455 #define TLS1_CK_DHE_RSA_WITH_AES_256_GCM_SHA384         0x0300009F
456 #define TLS1_CK_DH_RSA_WITH_AES_128_GCM_SHA256          0x030000A0
457 #define TLS1_CK_DH_RSA_WITH_AES_256_GCM_SHA384          0x030000A1
```



```
458 #define TLS1_CK_DHE_DSS_WITH_AES_128_GCM_SHA256         0x030000A2
459 #define TLS1_CK_DHE_DSS_WITH_AES_256_GCM_SHA384         0x030000A3
460 #define TLS1_CK_DH_DSS_WITH_AES_128_GCM_SHA256          0x030000A4
461 #define TLS1_CK_DH_DSS_WITH_AES_256_GCM_SHA384          0x030000A5
462 #define TLS1_CK_ADH_WITH_AES_128_GCM_SHA256             0x030000A6
463 #define TLS1_CK_ADH_WITH_AES_256_GCM_SHA384             0x030000A7

465 /* ECC ciphersuites from draft-ietf-tls-ecc-12.txt with changes soon to be in dr
466 #define TLS1_CK_ECDH_ECDSA_WITH_NULL_SHA                0x0300C001
467 #define TLS1_CK_ECDH_ECDSA_WITH_RC4_128_SHA             0x0300C002
468 #define TLS1_CK_ECDH_ECDSA_WITH_DES_192_CBC3_SHA        0x0300C003
469 #define TLS1_CK_ECDH_ECDSA_WITH_AES_128_CBC_SHA         0x0300C004
470 #define TLS1_CK_ECDH_ECDSA_WITH_AES_256_CBC_SHA         0x0300C005

472 #define TLS1_CK_ECDHE_ECDSA_WITH_NULL_SHA               0x0300C006
473 #define TLS1_CK_ECDHE_ECDSA_WITH_RC4_128_SHA            0x0300C007
474 #define TLS1_CK_ECDHE_ECDSA_WITH_DES_192_CBC3_SHA       0x0300C008
475 #define TLS1_CK_ECDHE_ECDSA_WITH_AES_128_CBC_SHA        0x0300C009
476 #define TLS1_CK_ECDHE_ECDSA_WITH_AES_256_CBC_SHA        0x0300C00A

478 #define TLS1_CK_ECDH_RSA_WITH_NULL_SHA                  0x0300C00B
479 #define TLS1_CK_ECDH_RSA_WITH_RC4_128_SHA               0x0300C00C
480 #define TLS1_CK_ECDH_RSA_WITH_DES_192_CBC3_SHA          0x0300C00D
481 #define TLS1_CK_ECDH_RSA_WITH_AES_128_CBC_SHA           0x0300C00E
482 #define TLS1_CK_ECDH_RSA_WITH_AES_256_CBC_SHA           0x0300C00F

484 #define TLS1_CK_ECDHE_RSA_WITH_NULL_SHA                 0x0300C010
485 #define TLS1_CK_ECDHE_RSA_WITH_RC4_128_SHA              0x0300C011
486 #define TLS1_CK_ECDHE_RSA_WITH_DES_192_CBC3_SHA         0x0300C012
487 #define TLS1_CK_ECDHE_RSA_WITH_AES_128_CBC_SHA          0x0300C013
488 #define TLS1_CK_ECDHE_RSA_WITH_AES_256_CBC_SHA          0x0300C014

490 #define TLS1_CK_ECDH_anon_WITH_NULL_SHA                 0x0300C015
491 #define TLS1_CK_ECDH_anon_WITH_RC4_128_SHA              0x0300C016
492 #define TLS1_CK_ECDH_anon_WITH_DES_192_CBC3_SHA         0x0300C017
493 #define TLS1_CK_ECDH_anon_WITH_AES_128_CBC_SHA          0x0300C018
494 #define TLS1_CK_ECDH_anon_WITH_AES_256_CBC_SHA          0x0300C019

496 /* SRP ciphersuites from RFC 5054 */
497 #define TLS1_CK_SRP_SHA_WITH_3DES_EDE_CBC_SHA           0x0300C01A
498 #define TLS1_CK_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA       0x0300C01B
499 #define TLS1_CK_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA       0x0300C01C
500 #define TLS1_CK_SRP_SHA_WITH_AES_128_CBC_SHA            0x0300C01D
501 #define TLS1_CK_SRP_SHA_RSA_WITH_AES_128_CBC_SHA        0x0300C01E
502 #define TLS1_CK_SRP_SHA_DSS_WITH_AES_128_CBC_SHA        0x0300C01F
503 #define TLS1_CK_SRP_SHA_WITH_AES_256_CBC_SHA            0x0300C020
504 #define TLS1_CK_SRP_SHA_RSA_WITH_AES_256_CBC_SHA        0x0300C021
505 #define TLS1_CK_SRP_SHA_DSS_WITH_AES_256_CBC_SHA        0x0300C022

507 /* ECDH HMAC based ciphersuites from RFC5289 */

509 #define TLS1_CK_ECDHE_ECDSA_WITH_AES_128_SHA256         0x0300C023
510 #define TLS1_CK_ECDHE_ECDSA_WITH_AES_256_SHA384         0x0300C024
511 #define TLS1_CK_ECDH_ECDSA_WITH_AES_128_SHA256          0x0300C025
512 #define TLS1_CK_ECDH_ECDSA_WITH_AES_256_SHA384          0x0300C026
513 #define TLS1_CK_ECDHE_RSA_WITH_AES_128_SHA256           0x0300C027
514 #define TLS1_CK_ECDHE_RSA_WITH_AES_256_SHA384           0x0300C028
515 #define TLS1_CK_ECDH_RSA_WITH_AES_128_SHA256            0x0300C029
516 #define TLS1_CK_ECDH_RSA_WITH_AES_256_SHA384            0x0300C02A

518 /* ECDH GCM based ciphersuites from RFC5289 */
519 #define TLS1_CK_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256     0x0300C02B
520 #define TLS1_CK_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384     0x0300C02C
521 #define TLS1_CK_ECDH_ECDSA_WITH_AES_128_GCM_SHA256      0x0300C02D
522 #define TLS1_CK_ECDH_ECDSA_WITH_AES_256_GCM_SHA384      0x0300C02E
523 #define TLS1_CK_ECDHE_RSA_WITH_AES_128_GCM_SHA256       0x0300C02F
```

```
524 #define TLS1_CK_ECDHE_RSA_WITH_AES_256_GCM_SHA384       0x0300C030
525 #define TLS1_CK_ECDH_RSA_WITH_AES_128_GCM_SHA256        0x0300C031
526 #define TLS1_CK_ECDH_RSA_WITH_AES_256_GCM_SHA384        0x0300C032

528 /* XXX
529  * Inconsistency alert:
530  * The OpenSSL names of ciphers with ephemeral DH here include the string
531  * "DHE", while elsewhere it has always been "EDH".
532  * (The alias for the list of all such ciphers also is "EDH".)
533  * The specifications speak of "EDH"; maybe we should allow both forms
534  * for everything. */
535 #define TLS1_TXT_RSA_EXPORT1024_WITH_RC4_56_MD5         "EXP1024-RC4-MD5"
536 #define TLS1_TXT_RSA_EXPORT1024_WITH_RC2_CBC_56_MD5     "EXP1024-RC2-CBC-MD5"
537 #define TLS1_TXT_RSA_EXPORT1024_WITH_DES_CBC_SHA        "EXP1024-DES-CBC-SHA"
538 #define TLS1_TXT_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA    "EXP1024-DHE-DSS-DES-CBC
539 #define TLS1_TXT_RSA_EXPORT1024_WITH_RC4_56_SHA         "EXP1024-RC4-SHA"
540 #define TLS1_TXT_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA     "EXP1024-DHE-DSS-RC4-SHA
541 #define TLS1_TXT_DHE_DSS_WITH_RC4_128_SHA               "DHE-DSS-RC4-SHA"

543 /* AES ciphersuites from RFC3268 */
544 #define TLS1_TXT_RSA_WITH_AES_128_SHA                   "AES128-SHA"
545 #define TLS1_TXT_DH_DSS_WITH_AES_128_SHA                "DH-DSS-AES128-SHA"
546 #define TLS1_TXT_DH_RSA_WITH_AES_128_SHA                "DH-RSA-AES128-SHA"
547 #define TLS1_TXT_DHE_DSS_WITH_AES_128_SHA               "DHE-DSS-AES128-SHA"
548 #define TLS1_TXT_DHE_RSA_WITH_AES_128_SHA               "DHE-RSA-AES128-SHA"
549 #define TLS1_TXT_ADH_WITH_AES_128_SHA                   "ADH-AES128-SHA"

551 #define TLS1_TXT_RSA_WITH_AES_256_SHA                   "AES256-SHA"
552 #define TLS1_TXT_DH_DSS_WITH_AES_256_SHA                "DH-DSS-AES256-SHA"
553 #define TLS1_TXT_DH_RSA_WITH_AES_256_SHA                "DH-RSA-AES256-SHA"
554 #define TLS1_TXT_DHE_DSS_WITH_AES_256_SHA               "DHE-DSS-AES256-SHA"
555 #define TLS1_TXT_DHE_RSA_WITH_AES_256_SHA               "DHE-RSA-AES256-SHA"
556 #define TLS1_TXT_ADH_WITH_AES_256_SHA                   "ADH-AES256-SHA"

558 /* ECC ciphersuites from draft-ietf-tls-ecc-01.txt (Mar 15, 2001) */
559 #define TLS1_TXT_ECDH_ECDSA_WITH_NULL_SHA               "ECDH-ECDSA-NULL-SHA"
560 #define TLS1_TXT_ECDH_ECDSA_WITH_RC4_128_SHA            "ECDH-ECDSA-RC4-SHA"
561 #define TLS1_TXT_ECDH_ECDSA_WITH_DES_192_CBC3_SHA       "ECDH-ECDSA-DES-CBC3-SHA
562 #define TLS1_TXT_ECDH_ECDSA_WITH_AES_128_CBC_SHA        "ECDH-ECDSA-AES128-SHA"
563 #define TLS1_TXT_ECDH_ECDSA_WITH_AES_256_CBC_SHA        "ECDH-ECDSA-AES256-SHA"

565 #define TLS1_TXT_ECDHE_ECDSA_WITH_NULL_SHA              "ECDHE-ECDSA-NULL-SHA"
566 #define TLS1_TXT_ECDHE_ECDSA_WITH_RC4_128_SHA           "ECDHE-ECDSA-RC4-SHA"
567 #define TLS1_TXT_ECDHE_ECDSA_WITH_DES_192_CBC3_SHA      "ECDHE-ECDSA-DES-CBC3-SH
568 #define TLS1_TXT_ECDHE_ECDSA_WITH_AES_128_CBC_SHA       "ECDHE-ECDSA-AES128-SHA"
569 #define TLS1_TXT_ECDHE_ECDSA_WITH_AES_256_CBC_SHA       "ECDHE-ECDSA-AES256-SHA"

571 #define TLS1_TXT_ECDH_RSA_WITH_NULL_SHA                 "ECDH-RSA-NULL-SHA"
572 #define TLS1_TXT_ECDH_RSA_WITH_RC4_128_SHA              "ECDH-RSA-RC4-SHA"
573 #define TLS1_TXT_ECDH_RSA_WITH_DES_192_CBC3_SHA         "ECDH-RSA-DES-CBC3-SHA"
574 #define TLS1_TXT_ECDH_RSA_WITH_AES_128_CBC_SHA          "ECDH-RSA-AES128-SHA"
575 #define TLS1_TXT_ECDH_RSA_WITH_AES_256_CBC_SHA          "ECDH-RSA-AES256-SHA"

577 #define TLS1_TXT_ECDHE_RSA_WITH_NULL_SHA                "ECDHE-RSA-NULL-SHA"
578 #define TLS1_TXT_ECDHE_RSA_WITH_RC4_128_SHA             "ECDHE-RSA-RC4-SHA"
579 #define TLS1_TXT_ECDHE_RSA_WITH_DES_192_CBC3_SHA        "ECDHE-RSA-DES-CBC3-SHA"
580 #define TLS1_TXT_ECDHE_RSA_WITH_AES_128_CBC_SHA         "ECDHE-RSA-AES128-SHA"
581 #define TLS1_TXT_ECDHE_RSA_WITH_AES_256_CBC_SHA         "ECDHE-RSA-AES256-SHA"

583 #define TLS1_TXT_ECDH_anon_WITH_NULL_SHA                "AECDH-NULL-SHA"
584 #define TLS1_TXT_ECDH_anon_WITH_RC4_128_SHA             "AECDH-RC4-SHA"
585 #define TLS1_TXT_ECDH_anon_WITH_DES_192_CBC3_SHA        "AECDH-DES-CBC3-SHA"
586 #define TLS1_TXT_ECDH_anon_WITH_AES_128_CBC_SHA         "AECDH-AES128-SHA"
587 #define TLS1_TXT_ECDH_anon_WITH_AES_256_CBC_SHA         "AECDH-AES256-SHA"

589 /* PSK ciphersuites from RFC 4279 */
```



```
590 #define TLS1_TXT_PSK_WITH_RC4_128_SHA                   "PSK-RC4-SHA"
591 #define TLS1_TXT_PSK_WITH_3DES_EDE_CBC_SHA              "PSK-3DES-EDE-CBC-SHA"
592 #define TLS1_TXT_PSK_WITH_AES_128_CBC_SHA               "PSK-AES128-CBC-SHA"
593 #define TLS1_TXT_PSK_WITH_AES_256_CBC_SHA               "PSK-AES256-CBC-SHA"

595 /* SRP ciphersuite from RFC 5054 */
596 #define TLS1_TXT_SRP_SHA_WITH_3DES_EDE_CBC_SHA          "SRP-3DES-EDE-CBC-SHA"
597 #define TLS1_TXT_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA      "SRP-RSA-3DES-EDE-CBC-SH
598 #define TLS1_TXT_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA      "SRP-DSS-3DES-EDE-CBC-SH
599 #define TLS1_TXT_SRP_SHA_WITH_AES_128_CBC_SHA           "SRP-AES-128-CBC-SHA"
600 #define TLS1_TXT_SRP_SHA_RSA_WITH_AES_128_CBC_SHA       "SRP-RSA-AES-128-CBC-SHA
601 #define TLS1_TXT_SRP_SHA_DSS_WITH_AES_128_CBC_SHA       "SRP-DSS-AES-128-CBC-SHA
602 #define TLS1_TXT_SRP_SHA_WITH_AES_256_CBC_SHA           "SRP-AES-256-CBC-SHA"
603 #define TLS1_TXT_SRP_SHA_RSA_WITH_AES_256_CBC_SHA       "SRP-RSA-AES-256-CBC-SHA
604 #define TLS1_TXT_SRP_SHA_DSS_WITH_AES_256_CBC_SHA       "SRP-DSS-AES-256-CBC-SHA

606 /* Camellia ciphersuites from RFC4132 */
607 #define TLS1_TXT_RSA_WITH_CAMELLIA_128_CBC_SHA          "CAMELLIA128-SHA"
608 #define TLS1_TXT_DH_DSS_WITH_CAMELLIA_128_CBC_SHA       "DH-DSS-CAMELLIA128-SHA"
609 #define TLS1_TXT_DH_RSA_WITH_CAMELLIA_128_CBC_SHA       "DH-RSA-CAMELLIA128-SHA"
610 #define TLS1_TXT_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA      "DHE-DSS-CAMELLIA128-SHA
611 #define TLS1_TXT_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA      "DHE-RSA-CAMELLIA128-SHA
612 #define TLS1_TXT_ADH_WITH_CAMELLIA_128_CBC_SHA          "ADH-CAMELLIA128-SHA"

614 #define TLS1_TXT_RSA_WITH_CAMELLIA_256_CBC_SHA          "CAMELLIA256-SHA"
615 #define TLS1_TXT_DH_DSS_WITH_CAMELLIA_256_CBC_SHA       "DH-DSS-CAMELLIA256-SHA"
616 #define TLS1_TXT_DH_RSA_WITH_CAMELLIA_256_CBC_SHA       "DH-RSA-CAMELLIA256-SHA"
617 #define TLS1_TXT_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA      "DHE-DSS-CAMELLIA256-SHA
618 #define TLS1_TXT_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA      "DHE-RSA-CAMELLIA256-SHA
619 #define TLS1_TXT_ADH_WITH_CAMELLIA_256_CBC_SHA          "ADH-CAMELLIA256-SHA"

621 /* SEED ciphersuites from RFC4162 */
622 #define TLS1_TXT_RSA_WITH_SEED_SHA                      "SEED-SHA"
623 #define TLS1_TXT_DH_DSS_WITH_SEED_SHA                   "DH-DSS-SEED-SHA"
624 #define TLS1_TXT_DH_RSA_WITH_SEED_SHA                   "DH-RSA-SEED-SHA"
625 #define TLS1_TXT_DHE_DSS_WITH_SEED_SHA                  "DHE-DSS-SEED-SHA"
626 #define TLS1_TXT_DHE_RSA_WITH_SEED_SHA                  "DHE-RSA-SEED-SHA"
627 #define TLS1_TXT_ADH_WITH_SEED_SHA                      "ADH-SEED-SHA"

629 /* TLS v1.2 ciphersuites */
630 #define TLS1_TXT_RSA_WITH_NULL_SHA256                   "NULL-SHA256"
631 #define TLS1_TXT_RSA_WITH_AES_128_SHA256                "AES128-SHA256"
632 #define TLS1_TXT_RSA_WITH_AES_256_SHA256                "AES256-SHA256"
633 #define TLS1_TXT_DH_DSS_WITH_AES_128_SHA256             "DH-DSS-AES128-SHA256"
634 #define TLS1_TXT_DH_RSA_WITH_AES_128_SHA256             "DH-RSA-AES128-SHA256"
635 #define TLS1_TXT_DHE_DSS_WITH_AES_128_SHA256            "DHE-DSS-AES128-SHA256"
636 #define TLS1_TXT_DHE_RSA_WITH_AES_128_SHA256            "DHE-RSA-AES128-SHA256"
637 #define TLS1_TXT_DH_DSS_WITH_AES_256_SHA256             "DH-DSS-AES256-SHA256"
638 #define TLS1_TXT_DH_RSA_WITH_AES_256_SHA256             "DH-RSA-AES256-SHA256"
639 #define TLS1_TXT_DHE_DSS_WITH_AES_256_SHA256            "DHE-DSS-AES256-SHA256"
640 #define TLS1_TXT_DHE_RSA_WITH_AES_256_SHA256            "DHE-RSA-AES256-SHA256"
641 #define TLS1_TXT_ADH_WITH_AES_128_SHA256                "ADH-AES128-SHA256"
642 #define TLS1_TXT_ADH_WITH_AES_256_SHA256                "ADH-AES256-SHA256"

644 /* TLS v1.2 GCM ciphersuites from RFC5288 */
645 #define TLS1_TXT_RSA_WITH_AES_128_GCM_SHA256            "AES128-GCM-SHA256"
646 #define TLS1_TXT_RSA_WITH_AES_256_GCM_SHA384            "AES256-GCM-SHA384"
647 #define TLS1_TXT_DHE_RSA_WITH_AES_128_GCM_SHA256        "DHE-RSA-AES128-GCM-SHA2
648 #define TLS1_TXT_DHE_RSA_WITH_AES_256_GCM_SHA384        "DHE-RSA-AES256-GCM-SHA3
649 #define TLS1_TXT_DH_RSA_WITH_AES_128_GCM_SHA256         "DH-RSA-AES128-GCM-SHA25
650 #define TLS1_TXT_DH_RSA_WITH_AES_256_GCM_SHA384         "DH-RSA-AES256-GCM-SHA38
651 #define TLS1_TXT_DHE_DSS_WITH_AES_128_GCM_SHA256        "DHE-DSS-AES128-GCM-SHA2
652 #define TLS1_TXT_DHE_DSS_WITH_AES_256_GCM_SHA384        "DHE-DSS-AES256-GCM-SHA3
653 #define TLS1_TXT_DH_DSS_WITH_AES_128_GCM_SHA256         "DH-DSS-AES128-GCM-SHA25
654 #define TLS1_TXT_DH_DSS_WITH_AES_256_GCM_SHA384         "DH-DSS-AES256-GCM-SHA38
655 #define TLS1_TXT_ADH_WITH_AES_128_GCM_SHA256            "ADH-AES128-GCM-SHA256"
```

```
#define TLS1_TXT_ADH_WITH_AES_256_GCM_SHA384            "ADH-AES256-GCM-SHA384"

/* ECDH HMAC based ciphersuites from RFC5289 */

#define TLS1_TXT_ECDHE_ECDSA_WITH_AES_128_SHA256    "ECDHE-ECDSA-AES128-SHA256"
#define TLS1_TXT_ECDHE_ECDSA_WITH_AES_256_SHA384    "ECDHE-ECDSA-AES256-SHA384"
#define TLS1_TXT_ECDH_ECDSA_WITH_AES_128_SHA256     "ECDH-ECDSA-AES128-SHA256"
#define TLS1_TXT_ECDH_ECDSA_WITH_AES_256_SHA384     "ECDH-ECDSA-AES256-SHA384"
#define TLS1_TXT_ECDHE_RSA_WITH_AES_128_SHA256      "ECDHE-RSA-AES128-SHA256"
#define TLS1_TXT_ECDHE_RSA_WITH_AES_256_SHA384      "ECDHE-RSA-AES256-SHA384"
#define TLS1_TXT_ECDH_RSA_WITH_AES_128_SHA256       "ECDH-RSA-AES128-SHA256"
#define TLS1_TXT_ECDH_RSA_WITH_AES_256_SHA384       "ECDH-RSA-AES256-SHA384"

/* ECDH GCM based ciphersuites from RFC5289 */
#define TLS1_TXT_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256    "ECDHE-ECDSA-AES128-GCM-
#define TLS1_TXT_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384    "ECDHE-ECDSA-AES256-GCM-
#define TLS1_TXT_ECDH_ECDSA_WITH_AES_128_GCM_SHA256     "ECDH-ECDSA-AES128-GCM-S
#define TLS1_TXT_ECDH_ECDSA_WITH_AES_256_GCM_SHA384     "ECDH-ECDSA-AES256-GCM-S
#define TLS1_TXT_ECDHE_RSA_WITH_AES_128_GCM_SHA256      "ECDHE-RSA-AES128-GCM-SH
#define TLS1_TXT_ECDHE_RSA_WITH_AES_256_GCM_SHA384      "ECDHE-RSA-AES256-GCM-SH
#define TLS1_TXT_ECDH_RSA_WITH_AES_128_GCM_SHA256       "ECDH-RSA-AES128-GCM-SHA
#define TLS1_TXT_ECDH_RSA_WITH_AES_256_GCM_SHA384       "ECDH-RSA-AES256-GCM-SHA

#define TLS_CT_RSA_SIGN                 1
#define TLS_CT_DSS_SIGN                 2
#define TLS_CT_RSA_FIXED_DH             3
#define TLS_CT_DSS_FIXED_DH             4
#define TLS_CT_ECDSA_SIGN               64
#define TLS_CT_RSA_FIXED_ECDH           65
#define TLS_CT_ECDSA_FIXED_ECDH         66
#define TLS_CT_GOST94_SIGN              21
#define TLS_CT_GOST01_SIGN              22
/* when correcting this number, correct also SSL3_CT_NUMBER in ssl3.h (see
 * comment there) */
#define TLS_CT_NUMBER                   9

#define TLS1_FINISH_MAC_LENGTH          12

#define TLS_MD_MAX_CONST_SIZE                   20
#define TLS_MD_CLIENT_FINISH_CONST              "client finished"
#define TLS_MD_CLIENT_FINISH_CONST_SIZE         15
#define TLS_MD_SERVER_FINISH_CONST              "server finished"
#define TLS_MD_SERVER_FINISH_CONST_SIZE         15
#define TLS_MD_SERVER_WRITE_KEY_CONST           "server write key"
#define TLS_MD_SERVER_WRITE_KEY_CONST_SIZE      16
#define TLS_MD_KEY_EXPANSION_CONST              "key expansion"
#define TLS_MD_KEY_EXPANSION_CONST_SIZE         13
#define TLS_MD_CLIENT_WRITE_KEY_CONST           "client write key"
#define TLS_MD_CLIENT_WRITE_KEY_CONST_SIZE      16
#define TLS_MD_SERVER_WRITE_KEY_CONST           "server write key"
#define TLS_MD_SERVER_WRITE_KEY_CONST_SIZE      16
#define TLS_MD_IV_BLOCK_CONST                   "IV block"
#define TLS_MD_IV_BLOCK_CONST_SIZE              8
#define TLS_MD_MASTER_SECRET_CONST              "master secret"
#define TLS_MD_MASTER_SECRET_CONST_SIZE         13

#ifdef CHARSET_EBCDIC
#undef TLS_MD_CLIENT_FINISH_CONST
#define TLS_MD_CLIENT_FINISH_CONST    "\x63\x6c\x69\x65\x6e\x74\x20\x66\x69\x6e\
#undef TLS_MD_SERVER_FINISH_CONST
#define TLS_MD_SERVER_FINISH_CONST    "\x73\x65\x72\x76\x65\x72\x20\x66\x69\x6e\
#undef TLS_MD_SERVER_WRITE_KEY_CONST
#define TLS_MD_SERVER_WRITE_KEY_CONST "\x73\x65\x72\x76\x65\x72\x20\x77\x72\x69\
#undef TLS_MD_KEY_EXPANSION_CONST
#define TLS_MD_KEY_EXPANSION_CONST    "\x6b\x65\x79\x20\x65\x78\x70\x61\x6e\x73\
#undef TLS_MD_CLIENT_WRITE_KEY_CONST
```



```
#define TLS_MD_CLIENT_WRITE_KEY_CONST "\x63\x6c\x69\x65\x6e\x74\x20\x77\x72\x69\
#undef TLS_MD_SERVER_WRITE_KEY_CONST
#define TLS_MD_SERVER_WRITE_KEY_CONST "\x73\x65\x72\x76\x65\x72\x20\x77\x72\x69\
#undef TLS_MD_IV_BLOCK_CONST
#define TLS_MD_IV_BLOCK_CONST         "\x49\x56\x20\x62\x6c\x6f\x63\x6b"  /*IV b
#undef TLS_MD_MASTER_SECRET_CONST
#define TLS_MD_MASTER_SECRET_CONST    "\x6d\x61\x73\x74\x65\x72\x20\x73\x65\x63\
#endif

/* TLS Session Ticket extension struct */
struct tls_session_ticket_ext_st
        {
        unsigned short length;
        void *data;
        };

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   30986 Wed Aug 13 19:51:50 2014
new/usr/src/lib/openssl/include/openssl/ts.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts.h */
/* Written by Zoltan Glozik (zglozik@opentsa.org) for the OpenSSL
 * project 2002, 2003, 2004.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_TS_H
#define HEADER_TS_H
```



```
#include <openssl/opensslconf.h>
#include <openssl/symhacks.h>
#ifndef OPENSSL_NO_BUFFER
#include <openssl/buffer.h>
#endif
#ifndef OPENSSL_NO_EVP
#include <openssl/evp.h>
#endif
#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/stack.h>
#include <openssl/asn1.h>
#include <openssl/safestack.h>

#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif

#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif

#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef WIN32
/* Under Win32 this is defined in wincrypt.h */
#undef X509_NAME
#endif

#include <openssl/x509.h>
#include <openssl/x509v3.h>

/*
MessageImprint ::= SEQUENCE  {
     hashAlgorithm                AlgorithmIdentifier,
     hashedMessage                OCTET STRING  }
*/

typedef struct TS_msg_imprint_st
        {
        X509_ALGOR *hash_algo;
        ASN1_OCTET_STRING *hashed_msg;
        } TS_MSG_IMPRINT;

/*
TimeStampReq ::= SEQUENCE  {
   version                  INTEGER  { v1(1) },
   messageImprint           MessageImprint,
     --a hash algorithm OID and the hash value of the data to be
     --time-stamped
   reqPolicy                TSAPolicyId                OPTIONAL,
   nonce                    INTEGER                    OPTIONAL,
   certReq                  BOOLEAN                    DEFAULT FALSE,
   extensions               [0] IMPLICIT Extensions    OPTIONAL  }
*/

typedef struct TS_req_st
        {
        ASN1_INTEGER *version;
```

```
128         TS_MSG_IMPRINT *msg_imprint;
129         ASN1_OBJECT *policy_id;         /* OPTIONAL */
130         ASN1_INTEGER *nonce;            /* OPTIONAL */
131         ASN1_BOOLEAN cert_req;          /* DEFAULT FALSE */
132         STACK_OF(X509_EXTENSION) *extensions;   /* [0] OPTIONAL */
133         } TS_REQ;

135 /*
136 Accuracy ::= SEQUENCE {
137                 seconds        INTEGER           OPTIONAL,
138                 millis     [0] INTEGER  (1..999) OPTIONAL,
139                 micros     [1] INTEGER  (1..999) OPTIONAL  }
140 */

142 typedef struct TS_accuracy_st
143         {
144         ASN1_INTEGER *seconds;
145         ASN1_INTEGER *millis;
146         ASN1_INTEGER *micros;
147         } TS_ACCURACY;

149 /*
150 TSTInfo ::= SEQUENCE  {
151     version                     INTEGER  { v1(1) },
152     policy                      TSAPolicyId,
153     messageImprint              MessageImprint,
154       -- MUST have the same value as the similar field in
155       -- TimeStampReq
156     serialNumber                INTEGER,
157      -- Time-Stamping users MUST be ready to accommodate integers
158      -- up to 160 bits.
159     genTime                     GeneralizedTime,
160     accuracy                    Accuracy                OPTIONAL,
161     ordering                    BOOLEAN             DEFAULT FALSE,
162     nonce                       INTEGER                 OPTIONAL,
163       -- MUST be present if the similar field was present
164       -- in TimeStampReq.  In that case it MUST have the same value.
165     tsa                         [0] GeneralName         OPTIONAL,
166     extensions                  [1] IMPLICIT Extensions  OPTIONAL   }
167 */

169 typedef struct TS_tst_info_st
170         {
171         ASN1_INTEGER *version;
172         ASN1_OBJECT *policy_id;
173         TS_MSG_IMPRINT *msg_imprint;
174         ASN1_INTEGER *serial;
175         ASN1_GENERALIZEDTIME *time;
176         TS_ACCURACY *accuracy;
177         ASN1_BOOLEAN ordering;
178         ASN1_INTEGER *nonce;
179         GENERAL_NAME *tsa;
180         STACK_OF(X509_EXTENSION) *extensions;
181         } TS_TST_INFO;

183 /*
184 PKIStatusInfo ::= SEQUENCE {
185     status        PKIStatus,
186     statusString  PKIFreeText     OPTIONAL,
187     failInfo      PKIFailureInfo  OPTIONAL  }

189 From RFC 1510 - section 3.1.1:
190 PKIFreeText ::= SEQUENCE SIZE (1..MAX) OF UTF8String
191         -- text encoded as UTF-8 String (note:  each UTF8String SHOULD
192         -- include an RFC 1766 language tag to indicate the language
193         -- of the contained text)
```



```
194 */

196 /* Possible values for status. See ts_resp_print.c && ts_resp_verify.c. */

198 #define TS_STATUS_GRANTED                       0
199 #define TS_STATUS_GRANTED_WITH_MODS             1
200 #define TS_STATUS_REJECTION                     2
201 #define TS_STATUS_WAITING                       3
202 #define TS_STATUS_REVOCATION_WARNING            4
203 #define TS_STATUS_REVOCATION_NOTIFICATION       5

205 /* Possible values for failure_info. See ts_resp_print.c && ts_resp_verify.c */

207 #define TS_INFO_BAD_ALG                 0
208 #define TS_INFO_BAD_REQUEST             2
209 #define TS_INFO_BAD_DATA_FORMAT         5
210 #define TS_INFO_TIME_NOT_AVAILABLE      14
211 #define TS_INFO_UNACCEPTED_POLICY       15
212 #define TS_INFO_UNACCEPTED_EXTENSION    16
213 #define TS_INFO_ADD_INFO_NOT_AVAILABLE  17
214 #define TS_INFO_SYSTEM_FAILURE          25

216 typedef struct TS_status_info_st
217         {
218         ASN1_INTEGER *status;
219         STACK_OF(ASN1_UTF8STRING) *text;
220         ASN1_BIT_STRING *failure_info;
221         } TS_STATUS_INFO;

223 DECLARE_STACK_OF(ASN1_UTF8STRING)
224 DECLARE_ASN1_SET_OF(ASN1_UTF8STRING)

226 /*
227 TimeStampResp ::= SEQUENCE  {
228      status                  PKIStatusInfo,
229     timeStampToken          TimeStampToken     OPTIONAL }
230 */

232 typedef struct TS_resp_st
233         {
234         TS_STATUS_INFO *status_info;
235         PKCS7 *token;
236         TS_TST_INFO *tst_info;
237         } TS_RESP;

239 /* The structure below would belong to the ESS component. */

241 /*
242 IssuerSerial ::= SEQUENCE {
243         issuer                  GeneralNames,
244         serialNumber            CertificateSerialNumber
245         }
246 */

248 typedef struct ESS_issuer_serial
249         {
250         STACK_OF(GENERAL_NAME)  *issuer;
251         ASN1_INTEGER            *serial;
252         } ESS_ISSUER_SERIAL;

254 /*
255 ESSCertID ::=  SEQUENCE {
256         certHash                Hash,
257         issuerSerial            IssuerSerial OPTIONAL
258 }
259 */
```

```
typedef struct ESS_cert_id
        {
        ASN1_OCTET_STRING *hash;        /* Always SHA-1 digest. */
        ESS_ISSUER_SERIAL *issuer_serial;
        } ESS_CERT_ID;

DECLARE_STACK_OF(ESS_CERT_ID)
DECLARE_ASN1_SET_OF(ESS_CERT_ID)

/*
SigningCertificate ::=  SEQUENCE {
       certs        SEQUENCE OF ESSCertID,
       policies     SEQUENCE OF PolicyInformation OPTIONAL
}
*/

typedef struct ESS_signing_cert
        {
        STACK_OF(ESS_CERT_ID) *cert_ids;
        STACK_OF(POLICYINFO) *policy_info;
        } ESS_SIGNING_CERT;


TS_REQ  *TS_REQ_new(void);
void    TS_REQ_free(TS_REQ *a);
int     i2d_TS_REQ(const TS_REQ *a, unsigned char **pp);
TS_REQ  *d2i_TS_REQ(TS_REQ **a, const unsigned char **pp, long length);

TS_REQ  *TS_REQ_dup(TS_REQ *a);

TS_REQ  *d2i_TS_REQ_fp(FILE *fp, TS_REQ **a);
int     i2d_TS_REQ_fp(FILE *fp, TS_REQ *a);
TS_REQ  *d2i_TS_REQ_bio(BIO *fp, TS_REQ **a);
int     i2d_TS_REQ_bio(BIO *fp, TS_REQ *a);

TS_MSG_IMPRINT  *TS_MSG_IMPRINT_new(void);
void            TS_MSG_IMPRINT_free(TS_MSG_IMPRINT *a);
int             i2d_TS_MSG_IMPRINT(const TS_MSG_IMPRINT *a, unsigned char **pp);
TS_MSG_IMPRINT  *d2i_TS_MSG_IMPRINT(TS_MSG_IMPRINT **a,
                                    const unsigned char **pp, long length);

TS_MSG_IMPRINT  *TS_MSG_IMPRINT_dup(TS_MSG_IMPRINT *a);

TS_MSG_IMPRINT  *d2i_TS_MSG_IMPRINT_fp(FILE *fp, TS_MSG_IMPRINT **a);
int             i2d_TS_MSG_IMPRINT_fp(FILE *fp, TS_MSG_IMPRINT *a);
TS_MSG_IMPRINT  *d2i_TS_MSG_IMPRINT_bio(BIO *fp, TS_MSG_IMPRINT **a);
int             i2d_TS_MSG_IMPRINT_bio(BIO *fp, TS_MSG_IMPRINT *a);

TS_RESP *TS_RESP_new(void);
void    TS_RESP_free(TS_RESP *a);
int     i2d_TS_RESP(const TS_RESP *a, unsigned char **pp);
TS_RESP *d2i_TS_RESP(TS_RESP **a, const unsigned char **pp, long length);
TS_TST_INFO *PKCS7_to_TS_TST_INFO(PKCS7 *token);
TS_RESP *TS_RESP_dup(TS_RESP *a);

TS_RESP *d2i_TS_RESP_fp(FILE *fp, TS_RESP **a);
int     i2d_TS_RESP_fp(FILE *fp, TS_RESP *a);
TS_RESP *d2i_TS_RESP_bio(BIO *fp, TS_RESP **a);
int     i2d_TS_RESP_bio(BIO *fp, TS_RESP *a);

TS_STATUS_INFO  *TS_STATUS_INFO_new(void);
void            TS_STATUS_INFO_free(TS_STATUS_INFO *a);
int             i2d_TS_STATUS_INFO(const TS_STATUS_INFO *a, unsigned char **pp);
TS_STATUS_INFO  *d2i_TS_STATUS_INFO(TS_STATUS_INFO **a,
                                    const unsigned char **pp, long length);
```



```
TS_STATUS_INFO  *TS_STATUS_INFO_dup(TS_STATUS_INFO *a);

TS_TST_INFO     *TS_TST_INFO_new(void);
void            TS_TST_INFO_free(TS_TST_INFO *a);
int             i2d_TS_TST_INFO(const TS_TST_INFO *a, unsigned char **pp);
TS_TST_INFO     *d2i_TS_TST_INFO(TS_TST_INFO **a, const unsigned char **pp,
                                 long length);
TS_TST_INFO     *TS_TST_INFO_dup(TS_TST_INFO *a);

TS_TST_INFO     *d2i_TS_TST_INFO_fp(FILE *fp, TS_TST_INFO **a);
int             i2d_TS_TST_INFO_fp(FILE *fp, TS_TST_INFO *a);
TS_TST_INFO     *d2i_TS_TST_INFO_bio(BIO *fp, TS_TST_INFO **a);
int             i2d_TS_TST_INFO_bio(BIO *fp, TS_TST_INFO *a);

TS_ACCURACY     *TS_ACCURACY_new(void);
void            TS_ACCURACY_free(TS_ACCURACY *a);
int             i2d_TS_ACCURACY(const TS_ACCURACY *a, unsigned char **pp);
TS_ACCURACY     *d2i_TS_ACCURACY(TS_ACCURACY **a, const unsigned char **pp,
                                 long length);
TS_ACCURACY     *TS_ACCURACY_dup(TS_ACCURACY *a);

ESS_ISSUER_SERIAL *ESS_ISSUER_SERIAL_new(void);
void              ESS_ISSUER_SERIAL_free(ESS_ISSUER_SERIAL *a);
int               i2d_ESS_ISSUER_SERIAL(const ESS_ISSUER_SERIAL *a,
                                        unsigned char **pp);
ESS_ISSUER_SERIAL *d2i_ESS_ISSUER_SERIAL(ESS_ISSUER_SERIAL **a,
                                         const unsigned char **pp, long length);
ESS_ISSUER_SERIAL *ESS_ISSUER_SERIAL_dup(ESS_ISSUER_SERIAL *a);

ESS_CERT_ID     *ESS_CERT_ID_new(void);
void            ESS_CERT_ID_free(ESS_CERT_ID *a);
int             i2d_ESS_CERT_ID(const ESS_CERT_ID *a, unsigned char **pp);
ESS_CERT_ID     *d2i_ESS_CERT_ID(ESS_CERT_ID **a, const unsigned char **pp,
                                 long length);
ESS_CERT_ID     *ESS_CERT_ID_dup(ESS_CERT_ID *a);

ESS_SIGNING_CERT *ESS_SIGNING_CERT_new(void);
void             ESS_SIGNING_CERT_free(ESS_SIGNING_CERT *a);
int              i2d_ESS_SIGNING_CERT(const ESS_SIGNING_CERT *a,
                                      unsigned char **pp);
ESS_SIGNING_CERT *d2i_ESS_SIGNING_CERT(ESS_SIGNING_CERT **a,
                                       const unsigned char **pp, long length);
ESS_SIGNING_CERT *ESS_SIGNING_CERT_dup(ESS_SIGNING_CERT *a);

void ERR_load_TS_strings(void);

int TS_REQ_set_version(TS_REQ *a, long version);
long TS_REQ_get_version(const TS_REQ *a);

int TS_REQ_set_msg_imprint(TS_REQ *a, TS_MSG_IMPRINT *msg_imprint);
TS_MSG_IMPRINT *TS_REQ_get_msg_imprint(TS_REQ *a);

int TS_MSG_IMPRINT_set_algo(TS_MSG_IMPRINT *a, X509_ALGOR *alg);
X509_ALGOR *TS_MSG_IMPRINT_get_algo(TS_MSG_IMPRINT *a);

int TS_MSG_IMPRINT_set_msg(TS_MSG_IMPRINT *a, unsigned char *d, int len);
ASN1_OCTET_STRING *TS_MSG_IMPRINT_get_msg(TS_MSG_IMPRINT *a);

int TS_REQ_set_policy_id(TS_REQ *a, ASN1_OBJECT *policy);
ASN1_OBJECT *TS_REQ_get_policy_id(TS_REQ *a);

int TS_REQ_set_nonce(TS_REQ *a, const ASN1_INTEGER *nonce);
const ASN1_INTEGER *TS_REQ_get_nonce(const TS_REQ *a);

int TS_REQ_set_cert_req(TS_REQ *a, int cert_req);
int TS_REQ_get_cert_req(const TS_REQ *a);
```

```
393 STACK_OF(X509_EXTENSION) *TS_REQ_get_exts(TS_REQ *a);
394 void TS_REQ_ext_free(TS_REQ *a);
395 int TS_REQ_get_ext_count(TS_REQ *a);
396 int TS_REQ_get_ext_by_NID(TS_REQ *a, int nid, int lastpos);
397 int TS_REQ_get_ext_by_OBJ(TS_REQ *a, ASN1_OBJECT *obj, int lastpos);
398 int TS_REQ_get_ext_by_critical(TS_REQ *a, int crit, int lastpos);
399 X509_EXTENSION *TS_REQ_get_ext(TS_REQ *a, int loc);
400 X509_EXTENSION *TS_REQ_delete_ext(TS_REQ *a, int loc);
401 int TS_REQ_add_ext(TS_REQ *a, X509_EXTENSION *ex, int loc);
402 void *TS_REQ_get_ext_d2i(TS_REQ *a, int nid, int *crit, int *idx);

404 /* Function declarations for TS_REQ defined in ts/ts_req_print.c */

406 int TS_REQ_print_bio(BIO *bio, TS_REQ *a);

408 /* Function declarations for TS_RESP defined in ts/ts_resp_utils.c */

410 int TS_RESP_set_status_info(TS_RESP *a, TS_STATUS_INFO *info);
411 TS_STATUS_INFO *TS_RESP_get_status_info(TS_RESP *a);

413 /* Caller loses ownership of PKCS7 and TS_TST_INFO objects. */
414 void TS_RESP_set_tst_info(TS_RESP *a, PKCS7 *p7, TS_TST_INFO *tst_info);
415 PKCS7 *TS_RESP_get_token(TS_RESP *a);
416 TS_TST_INFO *TS_RESP_get_tst_info(TS_RESP *a);

418 int TS_TST_INFO_set_version(TS_TST_INFO *a, long version);
419 long TS_TST_INFO_get_version(const TS_TST_INFO *a);

421 int TS_TST_INFO_set_policy_id(TS_TST_INFO *a, ASN1_OBJECT *policy_id);
422 ASN1_OBJECT *TS_TST_INFO_get_policy_id(TS_TST_INFO *a);

424 int TS_TST_INFO_set_msg_imprint(TS_TST_INFO *a, TS_MSG_IMPRINT *msg_imprint);
425 TS_MSG_IMPRINT *TS_TST_INFO_get_msg_imprint(TS_TST_INFO *a);

427 int TS_TST_INFO_set_serial(TS_TST_INFO *a, const ASN1_INTEGER *serial);
428 const ASN1_INTEGER *TS_TST_INFO_get_serial(const TS_TST_INFO *a);

430 int TS_TST_INFO_set_time(TS_TST_INFO *a, const ASN1_GENERALIZEDTIME *gtime);
431 const ASN1_GENERALIZEDTIME *TS_TST_INFO_get_time(const TS_TST_INFO *a);

433 int TS_TST_INFO_set_accuracy(TS_TST_INFO *a, TS_ACCURACY *accuracy);
434 TS_ACCURACY *TS_TST_INFO_get_accuracy(TS_TST_INFO *a);

436 int TS_ACCURACY_set_seconds(TS_ACCURACY *a, const ASN1_INTEGER *seconds);
437 const ASN1_INTEGER *TS_ACCURACY_get_seconds(const TS_ACCURACY *a);

439 int TS_ACCURACY_set_millis(TS_ACCURACY *a, const ASN1_INTEGER *millis);
440 const ASN1_INTEGER *TS_ACCURACY_get_millis(const TS_ACCURACY *a);

442 int TS_ACCURACY_set_micros(TS_ACCURACY *a, const ASN1_INTEGER *micros);
443 const ASN1_INTEGER *TS_ACCURACY_get_micros(const TS_ACCURACY *a);

445 int TS_TST_INFO_set_ordering(TS_TST_INFO *a, int ordering);
446 int TS_TST_INFO_get_ordering(const TS_TST_INFO *a);

448 int TS_TST_INFO_set_nonce(TS_TST_INFO *a, const ASN1_INTEGER *nonce);
449 const ASN1_INTEGER *TS_TST_INFO_get_nonce(const TS_TST_INFO *a);

451 int TS_TST_INFO_set_tsa(TS_TST_INFO *a, GENERAL_NAME *tsa);
452 GENERAL_NAME *TS_TST_INFO_get_tsa(TS_TST_INFO *a);

454 STACK_OF(X509_EXTENSION) *TS_TST_INFO_get_exts(TS_TST_INFO *a);
455 void TS_TST_INFO_ext_free(TS_TST_INFO *a);
456 int TS_TST_INFO_get_ext_count(TS_TST_INFO *a);
457 int TS_TST_INFO_get_ext_by_NID(TS_TST_INFO *a, int nid, int lastpos);
```



```
458 int TS_TST_INFO_get_ext_by_OBJ(TS_TST_INFO *a, ASN1_OBJECT *obj, int lastpos);
459 int TS_TST_INFO_get_ext_by_critical(TS_TST_INFO *a, int crit, int lastpos);
460 X509_EXTENSION *TS_TST_INFO_get_ext(TS_TST_INFO *a, int loc);
461 X509_EXTENSION *TS_TST_INFO_delete_ext(TS_TST_INFO *a, int loc);
462 int TS_TST_INFO_add_ext(TS_TST_INFO *a, X509_EXTENSION *ex, int loc);
463 void *TS_TST_INFO_get_ext_d2i(TS_TST_INFO *a, int nid, int *crit, int *idx);

465 /* Declarations related to response generation, defined in ts/ts_resp_sign.c. */

467 /* Optional flags for response generation. */

469 /* Don't include the TSA name in response. */
470 #define TS_TSA_NAME             0x01

472 /* Set ordering to true in response. */
473 #define TS_ORDERING             0x02

475 /*
476  * Include the signer certificate and the other specified certificates in
477  * the ESS signing certificate attribute beside the PKCS7 signed data.
478  * Only the signer certificates is included by default.
479  */
480 #define TS_ESS_CERT_ID_CHAIN    0x04

482 /* Forward declaration. */
483 struct TS_resp_ctx;

485 /* This must return a unique number less than 160 bits long. */
486 typedef ASN1_INTEGER *(*TS_serial_cb)(struct TS_resp_ctx *, void *);

488 /* This must return the seconds and microseconds since Jan 1, 1970 in
489    the sec and usec variables allocated by the caller.
490    Return non-zero for success and zero for failure. */
491 typedef int (*TS_time_cb)(struct TS_resp_ctx *, void *, long *sec, long *usec);

493 /* This must process the given extension.
494  * It can modify the TS_TST_INFO object of the context.
495  * Return values: !0 (processed), 0 (error, it must set the
496  * status info/failure info of the response).
497  */
498 typedef int (*TS_extension_cb)(struct TS_resp_ctx *, X509_EXTENSION *, void *);

500 typedef struct TS_resp_ctx
501         {
502         X509            *signer_cert;
503         EVP_PKEY        *signer_key;
504         STACK_OF(X509)  *certs; /* Certs to include in signed data. */
505         STACK_OF(ASN1_OBJECT)   *policies;      /* Acceptable policies. */
506         ASN1_OBJECT     *default_policy; /* It may appear in policies, too. */
507         STACK_OF(EVP_MD)        *mds;   /* Acceptable message digests. */
508         ASN1_INTEGER    *seconds;       /* accuracy, 0 means not specified. */
509         ASN1_INTEGER    *millis;        /* accuracy, 0 means not specified. */
510         ASN1_INTEGER    *micros;        /* accuracy, 0 means not specified. */
511         unsigned        clock_precision_digits; /* fraction of seconds in
512                                                    time stamp token. */
513         unsigned        flags;          /* Optional info, see values above. */

515         /* Callback functions. */
516         TS_serial_cb serial_cb;
517         void *serial_cb_data;   /* User data for serial_cb. */

519         TS_time_cb time_cb;
520         void *time_cb_data;     /* User data for time_cb. */

522         TS_extension_cb extension_cb;
523         void *extension_cb_data;        /* User data for extension_cb. */
```

```
525         /* These members are used only while creating the response. */
526         TS_REQ          *request;
527         TS_RESP         *response;
528         TS_TST_INFO     *tst_info;
529         } TS_RESP_CTX;

531 DECLARE_STACK_OF(EVP_MD)
532 DECLARE_ASN1_SET_OF(EVP_MD)

534 /* Creates a response context that can be used for generating responses. */
535 TS_RESP_CTX *TS_RESP_CTX_new(void);
536 void TS_RESP_CTX_free(TS_RESP_CTX *ctx);

538 /* This parameter must be set. */
539 int TS_RESP_CTX_set_signer_cert(TS_RESP_CTX *ctx, X509 *signer);

541 /* This parameter must be set. */
542 int TS_RESP_CTX_set_signer_key(TS_RESP_CTX *ctx, EVP_PKEY *key);

544 /* This parameter must be set. */
545 int TS_RESP_CTX_set_def_policy(TS_RESP_CTX *ctx, ASN1_OBJECT *def_policy);

547 /* No additional certs are included in the response by default. */
548 int TS_RESP_CTX_set_certs(TS_RESP_CTX *ctx, STACK_OF(X509) *certs);

550 /* Adds a new acceptable policy, only the default policy
551    is accepted by default. */
552 int TS_RESP_CTX_add_policy(TS_RESP_CTX *ctx, ASN1_OBJECT *policy);

554 /* Adds a new acceptable message digest. Note that no message digests
555    are accepted by default. The md argument is shared with the caller. */
556 int TS_RESP_CTX_add_md(TS_RESP_CTX *ctx, const EVP_MD *md);

558 /* Accuracy is not included by default. */
559 int TS_RESP_CTX_set_accuracy(TS_RESP_CTX *ctx,
560                              int secs, int millis, int micros);

562 /* Clock precision digits, i.e. the number of decimal digits:
563    '0' means sec, '3' msec, '6' usec, and so on. Default is 0. */
564 int TS_RESP_CTX_set_clock_precision_digits(TS_RESP_CTX *ctx,
565                                            unsigned clock_precision_digits);
566 /* At most we accept usec precision. */
567 #define TS_MAX_CLOCK_PRECISION_DIGITS   6

569 /* No flags are set by default. */
570 void TS_RESP_CTX_add_flags(TS_RESP_CTX *ctx, int flags);

572 /* Default callback always returns a constant. */
573 void TS_RESP_CTX_set_serial_cb(TS_RESP_CTX *ctx, TS_serial_cb cb, void *data);

575 /* Default callback uses the gettimeofday() and gmtime() system calls. */
576 void TS_RESP_CTX_set_time_cb(TS_RESP_CTX *ctx, TS_time_cb cb, void *data);

578 /* Default callback rejects all extensions. The extension callback is called
579  * when the TS_TST_INFO object is already set up and not signed yet. */
580 /* FIXME: extension handling is not tested yet. */
581 void TS_RESP_CTX_set_extension_cb(TS_RESP_CTX *ctx,
582                                   TS_extension_cb cb, void *data);

584 /* The following methods can be used in the callbacks. */
585 int TS_RESP_CTX_set_status_info(TS_RESP_CTX *ctx,
586                                 int status, const char *text);

588 /* Sets the status info only if it is still TS_STATUS_GRANTED. */
589 int TS_RESP_CTX_set_status_info_cond(TS_RESP_CTX *ctx,
```



```
590                                  int status, const char *text);

592 int TS_RESP_CTX_add_failure_info(TS_RESP_CTX *ctx, int failure);

594 /* The get methods below can be used in the extension callback. */
595 TS_REQ *TS_RESP_CTX_get_request(TS_RESP_CTX *ctx);

597 TS_TST_INFO *TS_RESP_CTX_get_tst_info(TS_RESP_CTX *ctx);

599 /*
600  * Creates the signed TS_TST_INFO and puts it in TS_RESP.
601  * In case of errors it sets the status info properly.
602  * Returns NULL only in case of memory allocation/fatal error.
603  */
604 TS_RESP *TS_RESP_create_response(TS_RESP_CTX *ctx, BIO *req_bio);

606 /*
607  * Declarations related to response verification,
608  * they are defined in ts/ts_resp_verify.c.
609  */

611 int TS_RESP_verify_signature(PKCS7 *token, STACK_OF(X509) *certs,
612                              X509_STORE *store, X509 **signer_out);

614 /* Context structure for the generic verify method. */

616 /* Verify the signer's certificate and the signature of the response. */
617 #define TS_VFY_SIGNATURE        (1u << 0)
618 /* Verify the version number of the response. */
619 #define TS_VFY_VERSION          (1u << 1)
620 /* Verify if the policy supplied by the user matches the policy of the TSA. */
621 #define TS_VFY_POLICY           (1u << 2)
622 /* Verify the message imprint provided by the user. This flag should not be
623    specified with TS_VFY_DATA. */
624 #define TS_VFY_IMPRINT          (1u << 3)
625 /* Verify the message imprint computed by the verify method from the user
626    provided data and the MD algorithm of the response. This flag should not be
627    specified with TS_VFY_IMPRINT. */
628 #define TS_VFY_DATA             (1u << 4)
629 /* Verify the nonce value. */
630 #define TS_VFY_NONCE            (1u << 5)
631 /* Verify if the TSA name field matches the signer certificate. */
632 #define TS_VFY_SIGNER           (1u << 6)
633 /* Verify if the TSA name field equals to the user provided name. */
634 #define TS_VFY_TSA_NAME         (1u << 7)

636 /* You can use the following convenience constants. */
637 #define TS_VFY_ALL_IMPRINT      (TS_VFY_SIGNATURE       \
638                                  | TS_VFY_VERSION       \
639                                  | TS_VFY_POLICY        \
640                                  | TS_VFY_IMPRINT       \
641                                  | TS_VFY_NONCE         \
642                                  | TS_VFY_SIGNER        \
643                                  | TS_VFY_TSA_NAME)
644 #define TS_VFY_ALL_DATA         (TS_VFY_SIGNATURE       \
645                                  | TS_VFY_VERSION       \
646                                  | TS_VFY_POLICY        \
647                                  | TS_VFY_DATA          \
648                                  | TS_VFY_NONCE         \
649                                  | TS_VFY_SIGNER        \
650                                  | TS_VFY_TSA_NAME)

652 typedef struct TS_verify_ctx
653         {
654         /* Set this to the union of TS_VFY_... flags you want to carry out. */
655         unsigned        flags;
```

```
657         /* Must be set only with TS_VFY_SIGNATURE. certs is optional. */
658         X509_STORE      *store;
659         STACK_OF(X509)  *certs;
660
661         /* Must be set only with TS_VFY_POLICY. */
662         ASN1_OBJECT     *policy;
663
664         /* Must be set only with TS_VFY_IMPRINT. If md_alg is NULL,
665            the algorithm from the response is used. */
666         X509_ALGOR      *md_alg;
667         unsigned char   *imprint;
668         unsigned        imprint_len;
669
670         /* Must be set only with TS_VFY_DATA. */
671         BIO             *data;
672
673         /* Must be set only with TS_VFY_TSA_NAME. */
674         ASN1_INTEGER    *nonce;
675
676         /* Must be set only with TS_VFY_TSA_NAME. */
677         GENERAL_NAME    *tsa_name;
678         } TS_VERIFY_CTX;
679
680 int TS_RESP_verify_response(TS_VERIFY_CTX *ctx, TS_RESP *response);
681 int TS_RESP_verify_token(TS_VERIFY_CTX *ctx, PKCS7 *token);
682
683 /*
684  * Declarations related to response verification context,
685  * they are defined in ts/ts_verify_ctx.c.
686  */
687
688 /* Set all fields to zero. */
689 TS_VERIFY_CTX *TS_VERIFY_CTX_new(void);
690 void TS_VERIFY_CTX_init(TS_VERIFY_CTX *ctx);
691 void TS_VERIFY_CTX_free(TS_VERIFY_CTX *ctx);
692 void TS_VERIFY_CTX_cleanup(TS_VERIFY_CTX *ctx);
693
694 /*
695  * If ctx is NULL, it allocates and returns a new object, otherwise
696  * it returns ctx. It initialises all the members as follows:
697  * flags = TS_VFY_ALL_IMPRINT & ~(TS_VFY_TSA_NAME | TS_VFY_SIGNATURE)
698  * certs = NULL
699  * store = NULL
700  * policy = policy from the request or NULL if absent (in this case
701  *      TS_VFY_POLICY is cleared from flags as well)
702  * md_alg = MD algorithm from request
703  * imprint, imprint_len = imprint from request
704  * data = NULL
705  * nonce, nonce_len = nonce from the request or NULL if absent (in this case
706  *      TS_VFY_NONCE is cleared from flags as well)
707  * tsa_name = NULL
708  * Important: after calling this method TS_VFY_SIGNATURE should be added!
709  */
710 TS_VERIFY_CTX *TS_REQ_to_TS_VERIFY_CTX(TS_REQ *req, TS_VERIFY_CTX *ctx);
711
712 /* Function declarations for TS_RESP defined in ts/ts_resp_print.c */
713
714 int TS_RESP_print_bio(BIO *bio, TS_RESP *a);
715 int TS_STATUS_INFO_print_bio(BIO *bio, TS_STATUS_INFO *a);
716 int TS_TST_INFO_print_bio(BIO *bio, TS_TST_INFO *a);
717
718 /* Common utility functions defined in ts/ts_lib.c */
719
720 int TS_ASN1_INTEGER_print_bio(BIO *bio, const ASN1_INTEGER *num);
721 int TS_OBJ_print_bio(BIO *bio, const ASN1_OBJECT *obj);
```



```
722 int TS_ext_print_bio(BIO *bio, const STACK_OF(X509_EXTENSION) *extensions);
723 int TS_X509_ALGOR_print_bio(BIO *bio, const X509_ALGOR *alg);
724 int TS_MSG_IMPRINT_print_bio(BIO *bio, TS_MSG_IMPRINT *msg);
725
726 /* Function declarations for handling configuration options,
727    defined in ts/ts_conf.c */
728
729 X509 *TS_CONF_load_cert(const char *file);
730 STACK_OF(X509) *TS_CONF_load_certs(const char *file);
731 EVP_PKEY *TS_CONF_load_key(const char *file, const char *pass);
732 const char *TS_CONF_get_tsa_section(CONF *conf, const char *section);
733 int TS_CONF_set_serial(CONF *conf, const char *section, TS_serial_cb cb,
734                        TS_RESP_CTX *ctx);
735 int TS_CONF_set_crypto_device(CONF *conf, const char *section,
736                               const char *device);
737 int TS_CONF_set_default_engine(const char *name);
738 int TS_CONF_set_signer_cert(CONF *conf, const char *section,
739                             const char *cert, TS_RESP_CTX *ctx);
740 int TS_CONF_set_certs(CONF *conf, const char *section, const char *certs,
741                       TS_RESP_CTX *ctx);
742 int TS_CONF_set_signer_key(CONF *conf, const char *section,
743                            const char *key, const char *pass, TS_RESP_CTX *ctx);
744 int TS_CONF_set_def_policy(CONF *conf, const char *section,
745                            const char *policy, TS_RESP_CTX *ctx);
746 int TS_CONF_set_policies(CONF *conf, const char *section, TS_RESP_CTX *ctx);
747 int TS_CONF_set_digests(CONF *conf, const char *section, TS_RESP_CTX *ctx);
748 int TS_CONF_set_accuracy(CONF *conf, const char *section, TS_RESP_CTX *ctx);
749 int TS_CONF_set_clock_precision_digits(CONF *conf, const char *section,
750                                        TS_RESP_CTX *ctx);
751 int TS_CONF_set_ordering(CONF *conf, const char *section, TS_RESP_CTX *ctx);
752 int TS_CONF_set_tsa_name(CONF *conf, const char *section, TS_RESP_CTX *ctx);
753 int TS_CONF_set_ess_cert_id_chain(CONF *conf, const char *section,
754                                   TS_RESP_CTX *ctx);
755
756 /* ---------------------------------------------------- */
757 /* BEGIN ERROR CODES */
758 /* The following lines are auto generated by the script mkerr.pl. Any changes
759  * made after this point may be overwritten when the script is next run.
760  */
761 void ERR_load_TS_strings(void);
762
763 /* Error codes for the TS functions. */
764
765 /* Function codes. */
766 #define TS_F_D2I_TS_RESP                                147
767 #define TS_F_DEF_SERIAL_CB                              110
768 #define TS_F_DEF_TIME_CB                                111
769 #define TS_F_ESS_ADD_SIGNING_CERT                       112
770 #define TS_F_ESS_CERT_ID_NEW_INIT                       113
771 #define TS_F_ESS_SIGNING_CERT_NEW_INIT                  114
772 #define TS_F_INT_TS_RESP_VERIFY_TOKEN                   149
773 #define TS_F_PKCS7_TO_TS_TST_INFO                       148
774 #define TS_F_TS_ACCURACY_SET_MICROS                     115
775 #define TS_F_TS_ACCURACY_SET_MILLIS                     116
776 #define TS_F_TS_ACCURACY_SET_SECONDS                    117
777 #define TS_F_TS_CHECK_IMPRINTS                          100
778 #define TS_F_TS_CHECK_NONCES                            101
779 #define TS_F_TS_CHECK_POLICY                            102
780 #define TS_F_TS_CHECK_SIGNING_CERTS                     103
781 #define TS_F_TS_CHECK_STATUS_INFO                       104
782 #define TS_F_TS_COMPUTE_IMPRINT                         145
783 #define TS_F_TS_CONF_SET_DEFAULT_ENGINE                 146
784 #define TS_F_TS_GET_STATUS_TEXT                         105
785 #define TS_F_TS_MSG_IMPRINT_SET_ALGO                    118
786 #define TS_F_TS_REQ_SET_MSG_IMPRINT                     119
787 #define TS_F_TS_REQ_SET_NONCE                           120
```

```
788 #define TS_F_TS_REQ_SET_POLICY_ID                       121
789 #define TS_F_TS_RESP_CREATE_RESPONSE                    122
790 #define TS_F_TS_RESP_CREATE_TST_INFO                    123
791 #define TS_F_TS_RESP_CTX_ADD_FAILURE_INFO               124
792 #define TS_F_TS_RESP_CTX_ADD_MD                         125
793 #define TS_F_TS_RESP_CTX_ADD_POLICY                     126
794 #define TS_F_TS_RESP_CTX_NEW                            127
795 #define TS_F_TS_RESP_CTX_SET_ACCURACY                   128
796 #define TS_F_TS_RESP_CTX_SET_CERTS                      129
797 #define TS_F_TS_RESP_CTX_SET_DEF_POLICY                 130
798 #define TS_F_TS_RESP_CTX_SET_SIGNER_CERT                131
799 #define TS_F_TS_RESP_CTX_SET_STATUS_INFO                132
800 #define TS_F_TS_RESP_GET_POLICY                         133
801 #define TS_F_TS_RESP_SET_GENTIME_WITH_PRECISION         134
802 #define TS_F_TS_RESP_SET_STATUS_INFO                    135
803 #define TS_F_TS_RESP_SET_TST_INFO                       150
804 #define TS_F_TS_RESP_SIGN                               136
805 #define TS_F_TS_RESP_VERIFY_SIGNATURE                   106
806 #define TS_F_TS_RESP_VERIFY_TOKEN                       107
807 #define TS_F_TS_TST_INFO_SET_ACCURACY                   137
808 #define TS_F_TS_TST_INFO_SET_MSG_IMPRINT                138
809 #define TS_F_TS_TST_INFO_SET_NONCE                      139
810 #define TS_F_TS_TST_INFO_SET_POLICY_ID                  140
811 #define TS_F_TS_TST_INFO_SET_SERIAL                     141
812 #define TS_F_TS_TST_INFO_SET_TIME                       142
813 #define TS_F_TS_TST_INFO_SET_TSA                        143
814 #define TS_F_TS_VERIFY                                  108
815 #define TS_F_TS_VERIFY_CERT                             109
816 #define TS_F_TS_VERIFY_CTX_NEW                          144

818 /* Reason codes. */
819 #define TS_R_BAD_PKCS7_TYPE                             132
820 #define TS_R_BAD_TYPE                                   133
821 #define TS_R_CERTIFICATE_VERIFY_ERROR                   100
822 #define TS_R_COULD_NOT_SET_ENGINE                       127
823 #define TS_R_COULD_NOT_SET_TIME                         115
824 #define TS_R_D2I_TS_RESP_INT_FAILED                     128
825 #define TS_R_DETACHED_CONTENT                           134
826 #define TS_R_ESS_ADD_SIGNING_CERT_ERROR                 116
827 #define TS_R_ESS_SIGNING_CERTIFICATE_ERROR              101
828 #define TS_R_INVALID_NULL_POINTER                       102
829 #define TS_R_INVALID_SIGNER_CERTIFICATE_PURPOSE         117
830 #define TS_R_MESSAGE_IMPRINT_MISMATCH                   103
831 #define TS_R_NONCE_MISMATCH                             104
832 #define TS_R_NONCE_NOT_RETURNED                         105
833 #define TS_R_NO_CONTENT                                 106
834 #define TS_R_NO_TIME_STAMP_TOKEN                        107
835 #define TS_R_PKCS7_ADD_SIGNATURE_ERROR                  118
836 #define TS_R_PKCS7_ADD_SIGNED_ATTR_ERROR                119
837 #define TS_R_PKCS7_TO_TS_TST_INFO_FAILED                129
838 #define TS_R_POLICY_MISMATCH                            108
839 #define TS_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE     120
840 #define TS_R_RESPONSE_SETUP_ERROR                       121
841 #define TS_R_SIGNATURE_FAILURE                          109
842 #define TS_R_THERE_MUST_BE_ONE_SIGNER                   110
843 #define TS_R_TIME_SYSCALL_ERROR                         122
844 #define TS_R_TOKEN_NOT_PRESENT                          130
845 #define TS_R_TOKEN_PRESENT                              131
846 #define TS_R_TSA_NAME_MISMATCH                          111
847 #define TS_R_TSA_UNTRUSTED                              112
848 #define TS_R_TST_INFO_SETUP_ERROR                       123
849 #define TS_R_TS_DATASIGN                                124
850 #define TS_R_UNACCEPTABLE_POLICY                        125
851 #define TS_R_UNSUPPORTED_MD_ALGORITHM                   126
852 #define TS_R_UNSUPPORTED_VERSION                        113
853 #define TS_R_WRONG_CONTENT_TYPE                         114
```



```
855 #ifdef  __cplusplus
856 }
857 #endif
858 #endif
859 #endif /* ! codereview */
```

```
**********************************************************
    4475 Wed Aug 13 19:51:50 2014
new/usr/src/lib/openssl/include/openssl/txt_db.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/txt_db/txt_db.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_TXT_DB_H
#define HEADER_TXT_DB_H
```



```
#include <openssl/opensslconf.h>
#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/stack.h>
#include <openssl/lhash.h>

#define DB_ERROR_OK                     0
#define DB_ERROR_MALLOC                 1
#define DB_ERROR_INDEX_CLASH            2
#define DB_ERROR_INDEX_OUT_OF_RANGE     3
#define DB_ERROR_NO_INDEX               4
#define DB_ERROR_INSERT_INDEX_CLASH     5

#ifdef  __cplusplus
extern "C" {
#endif

typedef OPENSSL_STRING *OPENSSL_PSTRING;
DECLARE_SPECIAL_STACK_OF(OPENSSL_PSTRING, OPENSSL_STRING)

typedef struct txt_db_st
        {
        int num_fields;
        STACK_OF(OPENSSL_PSTRING) *data;
        LHASH_OF(OPENSSL_STRING) **index;
        int (**qual)(OPENSSL_STRING *);
        long error;
        long arg1;
        long arg2;
        OPENSSL_STRING *arg_row;
        } TXT_DB;

#ifndef OPENSSL_NO_BIO
TXT_DB *TXT_DB_read(BIO *in, int num);
long TXT_DB_write(BIO *out, TXT_DB *db);
#else
TXT_DB *TXT_DB_read(char *in, int num);
long TXT_DB_write(char *out, TXT_DB *db);
#endif
int TXT_DB_create_index(TXT_DB *db,int field,int (*qual)(OPENSSL_STRING *),
                        LHASH_HASH_FN_TYPE hash, LHASH_COMP_FN_TYPE cmp);
void TXT_DB_free(TXT_DB *db);
OPENSSL_STRING *TXT_DB_get_by_index(TXT_DB *db, int idx, OPENSSL_STRING *value);
int TXT_DB_insert(TXT_DB *db, OPENSSL_STRING *value);

#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
   16655 Wed Aug 13 19:51:50 2014
new/usr/src/lib/openssl/include/openssl/ui.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ui/ui.h -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_UI_H
#define HEADER_UI_H
```



```
#ifndef OPENSSL_NO_DEPRECATED
#include <openssl/crypto.h>
#endif
#include <openssl/safestack.h>
#include <openssl/ossl_typ.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* Declared already in ossl_typ.h */
/* typedef struct ui_st UI; */
/* typedef struct ui_method_st UI_METHOD; */


/* All the following functions return -1 or NULL on error and in some cases
   (UI_process()) -2 if interrupted or in some other way cancelled.
   When everything is fine, they return 0, a positive value or a non-NULL
   pointer, all depending on their purpose. */

/* Creators and destructor.   */
UI *UI_new(void);
UI *UI_new_method(const UI_METHOD *method);
void UI_free(UI *ui);

/* The following functions are used to add strings to be printed and prompt
   strings to prompt for data.  The names are UI_{add,dup}_<function>_string
   and UI_{add,dup}_input_boolean.

   UI_{add,dup}_<function>_string have the following meanings:
        add     add a text or prompt string.  The pointers given to these
                functions are used verbatim, no copying is done.
        dup     make a copy of the text or prompt string, then add the copy
                to the collection of strings in the user interface.
        <function>
                The function is a name for the functionality that the given
                string shall be used for.  It can be one of:
                        input   use the string as data prompt.
                        verify  use the string as verification prompt.  This
                                is used to verify a previous input.
                        info    use the string for informational output.
                        error   use the string for error output.
   Honestly, there's currently no difference between info and error for the
   moment.

   UI_{add,dup}_input_boolean have the same semantics for "add" and "dup",
   and are typically used when one wants to prompt for a yes/no response.


   All of the functions in this group take a UI and a prompt string.
   The string input and verify addition functions also take a flag argument,
   a buffer for the result to end up with, a minimum input size and a maximum
   input size (the result buffer MUST be large enough to be able to contain
   the maximum number of characters).  Additionally, the verify addition
   functions takes another buffer to compare the result against.
   The boolean input functions take an action description string (which should
   be safe to ignore if the expected user action is obvious, for example with
   a dialog box with an OK button and a Cancel button), a string of acceptable
   characters to mean OK and to mean Cancel.  The two last strings are checked
   to make sure they don't have common characters.  Additionally, the same
   flag argument as for the string input is taken, as well as a result buffer.
   The result buffer is required to be at least one byte long.  Depending on
   the answer, the first character from the OK or the Cancel character strings
   will be stored in the first byte of the result buffer.  No NUL will be
   added, so the result is *not* a string.
```

```
   On success, the all return an index of the added information.  That index
   is usefull when retrieving results with UI_get0_result(). */
int UI_add_input_string(UI *ui, const char *prompt, int flags,
        char *result_buf, int minsize, int maxsize);
int UI_dup_input_string(UI *ui, const char *prompt, int flags,
        char *result_buf, int minsize, int maxsize);
int UI_add_verify_string(UI *ui, const char *prompt, int flags,
        char *result_buf, int minsize, int maxsize, const char *test_buf);
int UI_dup_verify_string(UI *ui, const char *prompt, int flags,
        char *result_buf, int minsize, int maxsize, const char *test_buf);
int UI_add_input_boolean(UI *ui, const char *prompt, const char *action_desc,
        const char *ok_chars, const char *cancel_chars,
        int flags, char *result_buf);
int UI_dup_input_boolean(UI *ui, const char *prompt, const char *action_desc,
        const char *ok_chars, const char *cancel_chars,
        int flags, char *result_buf);
int UI_add_info_string(UI *ui, const char *text);
int UI_dup_info_string(UI *ui, const char *text);
int UI_add_error_string(UI *ui, const char *text);
int UI_dup_error_string(UI *ui, const char *text);

/* These are the possible flags.  They can be or'ed together. */
/* Use to have echoing of input */
#define UI_INPUT_FLAG_ECHO              0x01
/* Use a default password.  Where that password is found is completely
   up to the application, it might for example be in the user data set
   with UI_add_user_data().  It is not recommended to have more than
   one input in each UI being marked with this flag, or the application
   might get confused. */
#define UI_INPUT_FLAG_DEFAULT_PWD       0x02

/* The user of these routines may want to define flags of their own.  The core
   UI won't look at those, but will pass them on to the method routines.  They
   must use higher bits so they don't get confused with the UI bits above.
   UI_INPUT_FLAG_USER_BASE tells which is the lowest bit to use.  A good
   example of use is this:

        #define MY_UI_FLAG1     (0x01 << UI_INPUT_FLAG_USER_BASE)

*/
#define UI_INPUT_FLAG_USER_BASE 16


/* The following function helps construct a prompt.  object_desc is a
   textual short description of the object, for example "pass phrase",
   and object_name is the name of the object (might be a card name or
   a file name.
   The returned string shall always be allocated on the heap with
   OPENSSL_malloc(), and need to be free'd with OPENSSL_free().

   If the ui_method doesn't contain a pointer to a user-defined prompt
   constructor, a default string is built, looking like this:

        "Enter {object_desc} for {object_name}:"

   So, if object_desc has the value "pass phrase" and object_name has
   the value "foo.key", the resulting string is:

        "Enter pass phrase for foo.key:"
*/
char *UI_construct_prompt(UI *ui_method,
        const char *object_desc, const char *object_name);


/* The following function is used to store a pointer to user-specific data.
   Any previous such pointer will be returned and replaced.
```



```
   For callback purposes, this function makes a lot more sense than using
   ex_data, since the latter requires that different parts of OpenSSL or
   applications share the same ex_data index.

   Note that the UI_OpenSSL() method completely ignores the user data.
   Other methods may not, however.  */
void *UI_add_user_data(UI *ui, void *user_data);
/* We need a user data retrieving function as well.  */
void *UI_get0_user_data(UI *ui);

/* Return the result associated with a prompt given with the index i. */
const char *UI_get0_result(UI *ui, int i);

/* When all strings have been added, process the whole thing. */
int UI_process(UI *ui);

/* Give a user interface parametrised control commands.  This can be used to
   send down an integer, a data pointer or a function pointer, as well as
   be used to get information from a UI. */
int UI_ctrl(UI *ui, int cmd, long i, void *p, void (*f)(void));

/* The commands */
/* Use UI_CONTROL_PRINT_ERRORS with the value 1 to have UI_process print the
   OpenSSL error stack before printing any info or added error messages and
   before any prompting. */
#define UI_CTRL_PRINT_ERRORS            1
/* Check if a UI_process() is possible to do again with the same instance of
   a user interface.  This makes UI_ctrl() return 1 if it is redoable, and 0
   if not. */
#define UI_CTRL_IS_REDOABLE             2


/* Some methods may use extra data */
#define UI_set_app_data(s,arg)          UI_set_ex_data(s,0,arg)
#define UI_get_app_data(s)              UI_get_ex_data(s,0)
int UI_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
        CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
int UI_set_ex_data(UI *r,int idx,void *arg);
void *UI_get_ex_data(UI *r, int idx);

/* Use specific methods instead of the built-in one */
void UI_set_default_method(const UI_METHOD *meth);
const UI_METHOD *UI_get_default_method(void);
const UI_METHOD *UI_get_method(UI *ui);
const UI_METHOD *UI_set_method(UI *ui, const UI_METHOD *meth);

/* The method with all the built-in thingies */
UI_METHOD *UI_OpenSSL(void);


/* ---------- For method writers ---------- */
/* A method contains a number of functions that implement the low level
   of the User Interface.  The functions are:

        an opener       This function starts a session, maybe by opening
                        a channel to a tty, or by opening a window.
        a writer        This function is called to write a given string,
                        maybe to the tty, maybe as a field label in a
                        window.
        a flusher       This function is called to flush everything that
                        has been output so far.  It can be used to actually
                        display a dialog box after it has been built.
        a reader        This function is called to read a given prompt,
                        maybe from the tty, maybe from a field in a
                        window.  Note that it's called wth all string
```

```
                        structures, not only the prompt ones, so it must
                        check such things itself.
        a closer        This function closes the session, maybe by closing
                        the channel to the tty, or closing the window.

   All these functions are expected to return:

        0       on error.
        1       on success.
        -1      on out-of-band events, for example if some prompting has
                been canceled (by pressing Ctrl-C, for example).  This is
                only checked when returned by the flusher or the reader.

   The way this is used, the opener is first called, then the writer for all
   strings, then the flusher, then the reader for all strings and finally the
   closer.  Note that if you want to prompt from a terminal or other command
   line interface, the best is to have the reader also write the prompts
   instead of having the writer do it.  If you want to prompt from a dialog
   box, the writer can be used to build up the contents of the box, and the
   flusher to actually display the box and run the event loop until all data
   has been given, after which the reader only grabs the given data and puts
   them back into the UI strings.

   All method functions take a UI as argument.  Additionally, the writer and
   the reader take a UI_STRING.
*/

/* The UI_STRING type is the data structure that contains all the needed info
   about a string or a prompt, including test data for a verification prompt.
*/
typedef struct ui_string_st UI_STRING;
DECLARE_STACK_OF(UI_STRING)

/* The different types of strings that are currently supported.
   This is only needed by method authors. */
enum UI_string_types
        {
        UIT_NONE=0,
        UIT_PROMPT,             /* Prompt for a string */
        UIT_VERIFY,             /* Prompt for a string and verify */
        UIT_BOOLEAN,            /* Prompt for a yes/no response */
        UIT_INFO,               /* Send info to the user */
        UIT_ERROR               /* Send an error message to the user */
        };

/* Create and manipulate methods */
UI_METHOD *UI_create_method(char *name);
void UI_destroy_method(UI_METHOD *ui_method);
int UI_method_set_opener(UI_METHOD *method, int (*opener)(UI *ui));
int UI_method_set_writer(UI_METHOD *method, int (*writer)(UI *ui, UI_STRING *uis
int UI_method_set_flusher(UI_METHOD *method, int (*flusher)(UI *ui));
int UI_method_set_reader(UI_METHOD *method, int (*reader)(UI *ui, UI_STRING *uis
int UI_method_set_closer(UI_METHOD *method, int (*closer)(UI *ui));
int UI_method_set_prompt_constructor(UI_METHOD *method, char *(*prompt_construct
int (*UI_method_get_opener(UI_METHOD *method))(UI*);
int (*UI_method_get_writer(UI_METHOD *method))(UI*,UI_STRING*);
int (*UI_method_get_flusher(UI_METHOD *method))(UI*);
int (*UI_method_get_reader(UI_METHOD *method))(UI*,UI_STRING*);
int (*UI_method_get_closer(UI_METHOD *method))(UI*);
char * (*UI_method_get_prompt_constructor(UI_METHOD *method))(UI*, const char*,

/* The following functions are helpers for method writers to access relevant
   data from a UI_STRING. */

/* Return type of the UI_STRING */
enum UI_string_types UI_get_string_type(UI_STRING *uis);
```



```
/* Return input flags of the UI_STRING */
int UI_get_input_flags(UI_STRING *uis);
/* Return the actual string to output (the prompt, info or error) */
const char *UI_get0_output_string(UI_STRING *uis);
/* Return the optional action string to output (the boolean promtp instruction)
const char *UI_get0_action_string(UI_STRING *uis);
/* Return the result of a prompt */
const char *UI_get0_result_string(UI_STRING *uis);
/* Return the string to test the result against.  Only useful with verifies. */
const char *UI_get0_test_string(UI_STRING *uis);
/* Return the required minimum size of the result */
int UI_get_result_minsize(UI_STRING *uis);
/* Return the required maximum size of the result */
int UI_get_result_maxsize(UI_STRING *uis);
/* Set the result of a UI_STRING. */
int UI_set_result(UI *ui, UI_STRING *uis, const char *result);


/* A couple of popular utility functions */
int UI_UTIL_read_pw_string(char *buf,int length,const char *prompt,int verify);
int UI_UTIL_read_pw(char *buf,char *buff,int size,const char *prompt,int verify)


/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_UI_strings(void);

/* Error codes for the UI functions. */

/* Function codes. */
#define UI_F_GENERAL_ALLOCATE_BOOLEAN                   108
#define UI_F_GENERAL_ALLOCATE_PROMPT                    109
#define UI_F_GENERAL_ALLOCATE_STRING                    100
#define UI_F_UI_CTRL                                    111
#define UI_F_UI_DUP_ERROR_STRING                        101
#define UI_F_UI_DUP_INFO_STRING                         102
#define UI_F_UI_DUP_INPUT_BOOLEAN                       110
#define UI_F_UI_DUP_INPUT_STRING                        103
#define UI_F_UI_DUP_VERIFY_STRING                       106
#define UI_F_UI_GET0_RESULT                             107
#define UI_F_UI_NEW_METHOD                              104
#define UI_F_UI_SET_RESULT                              105

/* Reason codes. */
#define UI_R_COMMON_OK_AND_CANCEL_CHARACTERS            104
#define UI_R_INDEX_TOO_LARGE                            102
#define UI_R_INDEX_TOO_SMALL                            103
#define UI_R_NO_RESULT_BUFFER                           105
#define UI_R_RESULT_TOO_LARGE                           100
#define UI_R_RESULT_TOO_SMALL                           101
#define UI_R_UNKNOWN_CONTROL_COMMAND                    106

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3438 Wed Aug 13 19:51:51 2014
new/usr/src/lib/openssl/include/openssl/ui_compat.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ui/ui.h -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_UI_COMPAT_H
#define HEADER_UI_COMPAT_H
```



```
#include <openssl/opensslconf.h>
#include <openssl/ui.h>

#ifdef  __cplusplus
extern "C" {
#endif

/* The following functions were previously part of the DES section,
   and are provided here for backward compatibility reasons. */

#define des_read_pw_string(b,l,p,v) \
        _ossl_old_des_read_pw_string((b),(l),(p),(v))
#define des_read_pw(b,bf,s,p,v) \
        _ossl_old_des_read_pw((b),(bf),(s),(p),(v))

int _ossl_old_des_read_pw_string(char *buf,int length,const char *prompt,int ver
int _ossl_old_des_read_pw(char *buf,char *buff,int size,const char *prompt,int v

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   45028 Wed Aug 13 19:51:51 2014
new/usr/src/lib/openssl/include/openssl/x509.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
```



```
 */

#ifndef HEADER_X509_H
#define HEADER_X509_H

#include <openssl/e_os2.h>
#include <openssl/symhacks.h>
#ifndef OPENSSL_NO_BUFFER
#include <openssl/buffer.h>
#endif
#ifndef OPENSSL_NO_EVP
#include <openssl/evp.h>
#endif
#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/stack.h>
#include <openssl/asn1.h>
#include <openssl/safestack.h>

#ifndef OPENSSL_NO_EC
#include <openssl/ec.h>
#endif

#ifndef OPENSSL_NO_ECDSA
#include <openssl/ecdsa.h>
#endif

#ifndef OPENSSL_NO_ECDH
#include <openssl/ecdh.h>
#endif

#ifndef OPENSSL_NO_DEPRECATED
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#endif

#ifndef OPENSSL_NO_SHA
#include <openssl/sha.h>
#endif
#include <openssl/ossl_typ.h>

#ifdef  __cplusplus
extern "C" {
#endif

#ifdef OPENSSL_SYS_WIN32
/* Under Win32 these are defined in wincrypt.h */
#undef X509_NAME
#undef X509_CERT_PAIR
#undef X509_EXTENSIONS
#endif

#define X509_FILETYPE_PEM       1
#define X509_FILETYPE_ASN1      2
#define X509_FILETYPE_DEFAULT   3

#define X509v3_KU_DIGITAL_SIGNATURE     0x0080
#define X509v3_KU_NON_REPUDIATION       0x0040
```

```
#define X509v3_KU_KEY_ENCIPHERMENT      0x0020
#define X509v3_KU_DATA_ENCIPHERMENT     0x0010
#define X509v3_KU_KEY_AGREEMENT         0x0008
#define X509v3_KU_KEY_CERT_SIGN         0x0004
#define X509v3_KU_CRL_SIGN              0x0002
#define X509v3_KU_ENCIPHER_ONLY         0x0001
#define X509v3_KU_DECIPHER_ONLY         0x8000
#define X509v3_KU_UNDEF                 0xffff

typedef struct X509_objects_st
        {
        int nid;
        int (*a2i)(void);
        int (*i2a)(void);
        } X509_OBJECTS;

struct X509_algor_st
        {
        ASN1_OBJECT *algorithm;
        ASN1_TYPE *parameter;
        } /* X509_ALGOR */;

DECLARE_ASN1_SET_OF(X509_ALGOR)

typedef STACK_OF(X509_ALGOR) X509_ALGORS;

typedef struct X509_val_st
        {
        ASN1_TIME *notBefore;
        ASN1_TIME *notAfter;
        } X509_VAL;

struct X509_pubkey_st
        {
        X509_ALGOR *algor;
        ASN1_BIT_STRING *public_key;
        EVP_PKEY *pkey;
        };

typedef struct X509_sig_st
        {
        X509_ALGOR *algor;
        ASN1_OCTET_STRING *digest;
        } X509_SIG;

typedef struct X509_name_entry_st
        {
        ASN1_OBJECT *object;
        ASN1_STRING *value;
        int set;
        int size;       /* temp variable */
        } X509_NAME_ENTRY;

DECLARE_STACK_OF(X509_NAME_ENTRY)
DECLARE_ASN1_SET_OF(X509_NAME_ENTRY)

/* we always keep X509_NAMEs in 2 forms. */
struct X509_name_st
        {
        STACK_OF(X509_NAME_ENTRY) *entries;
        int modified;   /* true if 'bytes' needs to be built */
#ifndef OPENSSL_NO_BUFFER
        BUF_MEM *bytes;
#else
        char *bytes;
#endif
```



```
/*      unsigned long hash; Keep the hash around for lookups */
        unsigned char *canon_enc;
        int canon_enclen;
        } /* X509_NAME */;

DECLARE_STACK_OF(X509_NAME)

#define X509_EX_V_NETSCAPE_HACK         0x8000
#define X509_EX_V_INIT                  0x0001
typedef struct X509_extension_st
        {
        ASN1_OBJECT *object;
        ASN1_BOOLEAN critical;
        ASN1_OCTET_STRING *value;
        } X509_EXTENSION;

typedef STACK_OF(X509_EXTENSION) X509_EXTENSIONS;

DECLARE_STACK_OF(X509_EXTENSION)
DECLARE_ASN1_SET_OF(X509_EXTENSION)

/* a sequence of these are used */
typedef struct x509_attributes_st
        {
        ASN1_OBJECT *object;
        int single; /* 0 for a set, 1 for a single item (which is wrong) */
        union   {
                char            *ptr;
/* 0 */         STACK_OF(ASN1_TYPE) *set;
/* 1 */         ASN1_TYPE       *single;
                } value;
        } X509_ATTRIBUTE;

DECLARE_STACK_OF(X509_ATTRIBUTE)
DECLARE_ASN1_SET_OF(X509_ATTRIBUTE)


typedef struct X509_req_info_st
        {
        ASN1_ENCODING enc;
        ASN1_INTEGER *version;
        X509_NAME *subject;
        X509_PUBKEY *pubkey;
        /*  d=2 hl=2 l=  0 cons: cont: 00 */
        STACK_OF(X509_ATTRIBUTE) *attributes; /* [ 0 ] */
        } X509_REQ_INFO;

typedef struct X509_req_st
        {
        X509_REQ_INFO *req_info;
        X509_ALGOR *sig_alg;
        ASN1_BIT_STRING *signature;
        int references;
        } X509_REQ;

typedef struct x509_cinf_st
        {
        ASN1_INTEGER *version;          /* [ 0 ] default of v1 */
        ASN1_INTEGER *serialNumber;
        X509_ALGOR *signature;
        X509_NAME *issuer;
        X509_VAL *validity;
        X509_NAME *subject;
        X509_PUBKEY *key;
        ASN1_BIT_STRING *issuerUID;             /* [ 1 ] optional in v2 */
        ASN1_BIT_STRING *subjectUID;            /* [ 2 ] optional in v2 */
```

```
260         STACK_OF(X509_EXTENSION) *extensions;   /* [ 3 ] optional in v3 */
261         ASN1_ENCODING enc;
262         } X509_CINF;

264 /* This stuff is certificate "auxiliary info"
265  * it contains details which are useful in certificate
266  * stores and databases. When used this is tagged onto
267  * the end of the certificate itself
268  */

270 typedef struct x509_cert_aux_st
271         {
272         STACK_OF(ASN1_OBJECT) *trust;           /* trusted uses */
273         STACK_OF(ASN1_OBJECT) *reject;          /* rejected uses */
274         ASN1_UTF8STRING *alias;                 /* "friendly name" */
275         ASN1_OCTET_STRING *keyid;               /* key id of private key */
276         STACK_OF(X509_ALGOR) *other;            /* other unspecified info */
277         } X509_CERT_AUX;

279 struct x509_st
280         {
281         X509_CINF *cert_info;
282         X509_ALGOR *sig_alg;
283         ASN1_BIT_STRING *signature;
284         int valid;
285         int references;
286         char *name;
287         CRYPTO_EX_DATA ex_data;
288         /* These contain copies of various extension values */
289         long ex_pathlen;
290         long ex_pcpathlen;
291         unsigned long ex_flags;
292         unsigned long ex_kusage;
293         unsigned long ex_xkusage;
294         unsigned long ex_nscert;
295         ASN1_OCTET_STRING *skid;
296         AUTHORITY_KEYID *akid;
297         X509_POLICY_CACHE *policy_cache;
298         STACK_OF(DIST_POINT) *crldp;
299         STACK_OF(GENERAL_NAME) *altname;
300         NAME_CONSTRAINTS *nc;
301 #ifndef OPENSSL_NO_RFC3779
302         STACK_OF(IPAddressFamily) *rfc3779_addr;
303         struct ASIdentifiers_st *rfc3779_asid;
304 #endif
305 #ifndef OPENSSL_NO_SHA
306         unsigned char sha1_hash[SHA_DIGEST_LENGTH];
307 #endif
308         X509_CERT_AUX *aux;
309         } /* X509 */;

311 DECLARE_STACK_OF(X509)
312 DECLARE_ASN1_SET_OF(X509)

314 /* This is used for a table of trust checking functions */

316 typedef struct x509_trust_st {
317         int trust;
318         int flags;
319         int (*check_trust)(struct x509_trust_st *, X509 *, int);
320         char *name;
321         int arg1;
322         void *arg2;
323 } X509_TRUST;

325 DECLARE_STACK_OF(X509_TRUST)
```



```
327 typedef struct x509_cert_pair_st {
328         X509 *forward;
329         X509 *reverse;
330 } X509_CERT_PAIR;

332 /* standard trust ids */

334 #define X509_TRUST_DEFAULT      -1      /* Only valid in purpose settings */

336 #define X509_TRUST_COMPAT       1
337 #define X509_TRUST_SSL_CLIENT   2
338 #define X509_TRUST_SSL_SERVER   3
339 #define X509_TRUST_EMAIL        4
340 #define X509_TRUST_OBJECT_SIGN  5
341 #define X509_TRUST_OCSP_SIGN    6
342 #define X509_TRUST_OCSP_REQUEST 7
343 #define X509_TRUST_TSA          8

345 /* Keep these up to date! */
346 #define X509_TRUST_MIN          1
347 #define X509_TRUST_MAX          8


350 /* trust_flags values */
351 #define X509_TRUST_DYNAMIC      1
352 #define X509_TRUST_DYNAMIC_NAME 2

354 /* check_trust return codes */

356 #define X509_TRUST_TRUSTED      1
357 #define X509_TRUST_REJECTED     2
358 #define X509_TRUST_UNTRUSTED    3

360 /* Flags for X509_print_ex() */

362 #define X509_FLAG_COMPAT                0
363 #define X509_FLAG_NO_HEADER             1L
364 #define X509_FLAG_NO_VERSION            (1L << 1)
365 #define X509_FLAG_NO_SERIAL             (1L << 2)
366 #define X509_FLAG_NO_SIGNAME            (1L << 3)
367 #define X509_FLAG_NO_ISSUER             (1L << 4)
368 #define X509_FLAG_NO_VALIDITY           (1L << 5)
369 #define X509_FLAG_NO_SUBJECT            (1L << 6)
370 #define X509_FLAG_NO_PUBKEY             (1L << 7)
371 #define X509_FLAG_NO_EXTENSIONS         (1L << 8)
372 #define X509_FLAG_NO_SIGDUMP            (1L << 9)
373 #define X509_FLAG_NO_AUX                (1L << 10)
374 #define X509_FLAG_NO_ATTRIBUTES         (1L << 11)

376 /* Flags specific to X509_NAME_print_ex() */

378 /* The field separator information */

380 #define XN_FLAG_SEP_MASK        (0xf << 16)

382 #define XN_FLAG_COMPAT          0               /* Traditional SSLeay: use old X
383 #define XN_FLAG_SEP_COMMA_PLUS  (1 << 16)       /* RFC2253 ,+ */
384 #define XN_FLAG_SEP_CPLUS_SPC   (2 << 16)       /* ,+ spaced: more readable */
385 #define XN_FLAG_SEP_SPLUS_SPC   (3 << 16)       /* ;+ spaced */
386 #define XN_FLAG_SEP_MULTILINE   (4 << 16)       /* One line per field */

388 #define XN_FLAG_DN_REV          (1 << 20)       /* Reverse DN order */

390 /* How the field name is shown */
```

```
392 #define XN_FLAG_FN_MASK         (0x3 << 21)

394 #define XN_FLAG_FN_SN           0               /* Object short name */
395 #define XN_FLAG_FN_LN           (1 << 21)       /* Object long name */
396 #define XN_FLAG_FN_OID          (2 << 21)       /* Always use OIDs */
397 #define XN_FLAG_FN_NONE         (3 << 21)       /* No field names */

399 #define XN_FLAG_SPC_EQ          (1 << 23)       /* Put spaces round '=' */

401 /* This determines if we dump fields we don't recognise:
402  * RFC2253 requires this.
403  */

405 #define XN_FLAG_DUMP_UNKNOWN_FIELDS (1 << 24)

407 #define XN_FLAG_FN_ALIGN        (1 << 25)       /* Align field names to 20 chara

409 /* Complete set of RFC2253 flags */

411 #define XN_FLAG_RFC2253 (ASN1_STRFLGS_RFC2253 | \
412                         XN_FLAG_SEP_COMMA_PLUS | \
413                         XN_FLAG_DN_REV | \
414                         XN_FLAG_FN_SN | \
415                         XN_FLAG_DUMP_UNKNOWN_FIELDS)

417 /* readable oneline form */

419 #define XN_FLAG_ONELINE (ASN1_STRFLGS_RFC2253 | \
420                         ASN1_STRFLGS_ESC_QUOTE | \
421                         XN_FLAG_SEP_CPLUS_SPC | \
422                         XN_FLAG_SPC_EQ | \
423                         XN_FLAG_FN_SN)

425 /* readable multiline form */

427 #define XN_FLAG_MULTILINE (ASN1_STRFLGS_ESC_CTRL | \
428                         ASN1_STRFLGS_ESC_MSB | \
429                         XN_FLAG_SEP_MULTILINE | \
430                         XN_FLAG_SPC_EQ | \
431                         XN_FLAG_FN_LN | \
432                         XN_FLAG_FN_ALIGN)

434 struct x509_revoked_st
435         {
436         ASN1_INTEGER *serialNumber;
437         ASN1_TIME *revocationDate;
438         STACK_OF(X509_EXTENSION) /* optional */ *extensions;
439         /* Set up if indirect CRL */
440         STACK_OF(GENERAL_NAME) *issuer;
441         /* Revocation reason */
442         int reason;
443         int sequence; /* load sequence */
444         };

446 DECLARE_STACK_OF(X509_REVOKED)
447 DECLARE_ASN1_SET_OF(X509_REVOKED)

449 typedef struct X509_crl_info_st
450         {
451         ASN1_INTEGER *version;
452         X509_ALGOR *sig_alg;
453         X509_NAME *issuer;
454         ASN1_TIME *lastUpdate;
455         ASN1_TIME *nextUpdate;
456         STACK_OF(X509_REVOKED) *revoked;
457         STACK_OF(X509_EXTENSION) /* [0] */ *extensions;
```



```
458         ASN1_ENCODING enc;
459         } X509_CRL_INFO;

461 struct X509_crl_st
462         {
463         /* actual signature */
464         X509_CRL_INFO *crl;
465         X509_ALGOR *sig_alg;
466         ASN1_BIT_STRING *signature;
467         int references;
468         int flags;
469         /* Copies of various extensions */
470         AUTHORITY_KEYID *akid;
471         ISSUING_DIST_POINT *idp;
472         /* Convenient breakdown of IDP */
473         int idp_flags;
474         int idp_reasons;
475         /* CRL and base CRL numbers for delta processing */
476         ASN1_INTEGER *crl_number;
477         ASN1_INTEGER *base_crl_number;
478 #ifndef OPENSSL_NO_SHA
479         unsigned char sha1_hash[SHA_DIGEST_LENGTH];
480 #endif
481         STACK_OF(GENERAL_NAMES) *issuers;
482         const X509_CRL_METHOD *meth;
483         void *meth_data;
484         } /* X509_CRL */;

486 DECLARE_STACK_OF(X509_CRL)
487 DECLARE_ASN1_SET_OF(X509_CRL)

489 typedef struct private_key_st
490         {
491         int version;
492         /* The PKCS#8 data types */
493         X509_ALGOR *enc_algor;
494         ASN1_OCTET_STRING *enc_pkey;    /* encrypted pub key */

496         /* When decrypted, the following will not be NULL */
497         EVP_PKEY *dec_pkey;

499         /* used to encrypt and decrypt */
500         int key_length;
501         char *key_data;
502         int key_free;   /* true if we should auto free key_data */

504         /* expanded version of 'enc_algor' */
505         EVP_CIPHER_INFO cipher;

507         int references;
508         } X509_PKEY;

510 #ifndef OPENSSL_NO_EVP
511 typedef struct X509_info_st
512         {
513         X509 *x509;
514         X509_CRL *crl;
515         X509_PKEY *x_pkey;

517         EVP_CIPHER_INFO enc_cipher;
518         int enc_len;
519         char *enc_data;

521         int references;
522         } X509_INFO;
```

```
524 DECLARE_STACK_OF(X509_INFO)
525 #endif

527 /* The next 2 structures and their 8 routines were sent to me by
528  * Pat Richard <patr@x509.com> and are used to manipulate
529  * Netscapes spki structures - useful if you are writing a CA web page
530  */
531 typedef struct Netscape_spkac_st
532         {
533         X509_PUBKEY *pubkey;
534         ASN1_IA5STRING *challenge;      /* challenge sent in atlas >= PR2 */
535         } NETSCAPE_SPKAC;

537 typedef struct Netscape_spki_st
538         {
539         NETSCAPE_SPKAC *spkac;  /* signed public key and challenge */
540         X509_ALGOR *sig_algor;
541         ASN1_BIT_STRING *signature;
542         } NETSCAPE_SPKI;

544 /* Netscape certificate sequence structure */
545 typedef struct Netscape_certificate_sequence
546         {
547         ASN1_OBJECT *type;
548         STACK_OF(X509) *certs;
549         } NETSCAPE_CERT_SEQUENCE;

551 /* Unused (and iv length is wrong)
552 typedef struct CBCParameter_st
553         {
554         unsigned char iv[8];
555         } CBC_PARAM;
556 */

558 /* Password based encryption structure */

560 typedef struct PBEPARAM_st {
561 ASN1_OCTET_STRING *salt;
562 ASN1_INTEGER *iter;
563 } PBEPARAM;

565 /* Password based encryption V2 structures */

567 typedef struct PBE2PARAM_st {
568 X509_ALGOR *keyfunc;
569 X509_ALGOR *encryption;
570 } PBE2PARAM;

572 typedef struct PBKDF2PARAM_st {
573 ASN1_TYPE *salt;        /* Usually OCTET STRING but could be anything */
574 ASN1_INTEGER *iter;
575 ASN1_INTEGER *keylength;
576 X509_ALGOR *prf;
577 } PBKDF2PARAM;


580 /* PKCS#8 private key info structure */

582 struct pkcs8_priv_key_info_st
583         {
584         int broken;     /* Flag for various broken formats */
585 #define PKCS8_OK                0
586 #define PKCS8_NO_OCTET          1
587 #define PKCS8_EMBEDDED_PARAM    2
588 #define PKCS8_NS_DB             3
589 #define PKCS8_NEG_PRIVKEY       4
```



```
590         ASN1_INTEGER *version;
591         X509_ALGOR *pkeyalg;
592         ASN1_TYPE *pkey; /* Should be OCTET STRING but some are broken */
593         STACK_OF(X509_ATTRIBUTE) *attributes;
594         };

596 #ifdef  __cplusplus
597 }
598 #endif

600 #include <openssl/x509_vfy.h>
601 #include <openssl/pkcs7.h>

603 #ifdef  __cplusplus
604 extern "C" {
605 #endif

607 #define X509_EXT_PACK_UNKNOWN   1
608 #define X509_EXT_PACK_STRING    2

610 #define         X509_get_version(x) ASN1_INTEGER_get((x)->cert_info->version)
611 /* #define      X509_get_serialNumber(x) ((x)->cert_info->serialNumber) */
612 #define         X509_get_notBefore(x) ((x)->cert_info->validity->notBefore)
613 #define         X509_get_notAfter(x) ((x)->cert_info->validity->notAfter)
614 #define         X509_extract_key(x)     X509_get_pubkey(x) /*****/
615 #define         X509_REQ_get_version(x) ASN1_INTEGER_get((x)->req_info->version)
616 #define         X509_REQ_get_subject_name(x) ((x)->req_info->subject)
617 #define         X509_REQ_extract_key(a) X509_REQ_get_pubkey(a)
618 #define         X509_name_cmp(a,b)      X509_NAME_cmp((a),(b))
619 #define         X509_get_signature_type(x) EVP_PKEY_type(OBJ_obj2nid((x)->sig_al

621 #define         X509_CRL_get_version(x) ASN1_INTEGER_get((x)->crl->version)
622 #define         X509_CRL_get_lastUpdate(x) ((x)->crl->lastUpdate)
623 #define         X509_CRL_get_nextUpdate(x) ((x)->crl->nextUpdate)
624 #define         X509_CRL_get_issuer(x) ((x)->crl->issuer)
625 #define         X509_CRL_get_REVOKED(x) ((x)->crl->revoked)

627 void X509_CRL_set_default_method(const X509_CRL_METHOD *meth);
628 X509_CRL_METHOD *X509_CRL_METHOD_new(
629         int (*crl_init)(X509_CRL *crl),
630         int (*crl_free)(X509_CRL *crl),
631         int (*crl_lookup)(X509_CRL *crl, X509_REVOKED **ret,
632                           ASN1_INTEGER *ser, X509_NAME *issuer),
633         int (*crl_verify)(X509_CRL *crl, EVP_PKEY *pk));
634 void X509_CRL_METHOD_free(X509_CRL_METHOD *m);

636 void X509_CRL_set_meth_data(X509_CRL *crl, void *dat);
637 void *X509_CRL_get_meth_data(X509_CRL *crl);

639 /* This one is only used so that a binary form can output, as in
640  * i2d_X509_NAME(X509_get_X509_PUBKEY(x),&buf) */
641 #define         X509_get_X509_PUBKEY(x) ((x)->cert_info->key)


644 const char *X509_verify_cert_error_string(long n);

646 #ifndef OPENSSL_NO_EVP
647 int X509_verify(X509 *a, EVP_PKEY *r);

649 int X509_REQ_verify(X509_REQ *a, EVP_PKEY *r);
650 int X509_CRL_verify(X509_CRL *a, EVP_PKEY *r);
651 int NETSCAPE_SPKI_verify(NETSCAPE_SPKI *a, EVP_PKEY *r);

653 NETSCAPE_SPKI * NETSCAPE_SPKI_b64_decode(const char *str, int len);
654 char * NETSCAPE_SPKI_b64_encode(NETSCAPE_SPKI *x);
655 EVP_PKEY *NETSCAPE_SPKI_get_pubkey(NETSCAPE_SPKI *x);
```

```
656 int NETSCAPE_SPKI_set_pubkey(NETSCAPE_SPKI *x, EVP_PKEY *pkey);

658 int NETSCAPE_SPKI_print(BIO *out, NETSCAPE_SPKI *spki);

660 int X509_signature_dump(BIO *bp,const ASN1_STRING *sig, int indent);
661 int X509_signature_print(BIO *bp,X509_ALGOR *alg, ASN1_STRING *sig);

663 int X509_sign(X509 *x, EVP_PKEY *pkey, const EVP_MD *md);
664 int X509_sign_ctx(X509 *x, EVP_MD_CTX *ctx);
665 int X509_REQ_sign(X509_REQ *x, EVP_PKEY *pkey, const EVP_MD *md);
666 int X509_REQ_sign_ctx(X509_REQ *x, EVP_MD_CTX *ctx);
667 int X509_CRL_sign(X509_CRL *x, EVP_PKEY *pkey, const EVP_MD *md);
668 int X509_CRL_sign_ctx(X509_CRL *x, EVP_MD_CTX *ctx);
669 int NETSCAPE_SPKI_sign(NETSCAPE_SPKI *x, EVP_PKEY *pkey, const EVP_MD *md);

671 int X509_pubkey_digest(const X509 *data,const EVP_MD *type,
672                 unsigned char *md, unsigned int *len);
673 int X509_digest(const X509 *data,const EVP_MD *type,
674                 unsigned char *md, unsigned int *len);
675 int X509_CRL_digest(const X509_CRL *data,const EVP_MD *type,
676                 unsigned char *md, unsigned int *len);
677 int X509_REQ_digest(const X509_REQ *data,const EVP_MD *type,
678                 unsigned char *md, unsigned int *len);
679 int X509_NAME_digest(const X509_NAME *data,const EVP_MD *type,
680                 unsigned char *md, unsigned int *len);
681 #endif

683 #ifndef OPENSSL_NO_FP_API
684 X509 *d2i_X509_fp(FILE *fp, X509 **x509);
685 int i2d_X509_fp(FILE *fp,X509 *x509);
686 X509_CRL *d2i_X509_CRL_fp(FILE *fp,X509_CRL **crl);
687 int i2d_X509_CRL_fp(FILE *fp,X509_CRL *crl);
688 X509_REQ *d2i_X509_REQ_fp(FILE *fp,X509_REQ **req);
689 int i2d_X509_REQ_fp(FILE *fp,X509_REQ *req);
690 #ifndef OPENSSL_NO_RSA
691 RSA *d2i_RSAPrivateKey_fp(FILE *fp,RSA **rsa);
692 int i2d_RSAPrivateKey_fp(FILE *fp,RSA *rsa);
693 RSA *d2i_RSAPublicKey_fp(FILE *fp,RSA **rsa);
694 int i2d_RSAPublicKey_fp(FILE *fp,RSA *rsa);
695 RSA *d2i_RSA_PUBKEY_fp(FILE *fp,RSA **rsa);
696 int i2d_RSA_PUBKEY_fp(FILE *fp,RSA *rsa);
697 #endif
698 #ifndef OPENSSL_NO_DSA
699 DSA *d2i_DSA_PUBKEY_fp(FILE *fp, DSA **dsa);
700 int i2d_DSA_PUBKEY_fp(FILE *fp, DSA *dsa);
701 DSA *d2i_DSAPrivateKey_fp(FILE *fp, DSA **dsa);
702 int i2d_DSAPrivateKey_fp(FILE *fp, DSA *dsa);
703 #endif
704 #ifndef OPENSSL_NO_EC
705 EC_KEY *d2i_EC_PUBKEY_fp(FILE *fp, EC_KEY **eckey);
706 int   i2d_EC_PUBKEY_fp(FILE *fp, EC_KEY *eckey);
707 EC_KEY *d2i_ECPrivateKey_fp(FILE *fp, EC_KEY **eckey);
708 int   i2d_ECPrivateKey_fp(FILE *fp, EC_KEY *eckey);
709 #endif
710 X509_SIG *d2i_PKCS8_fp(FILE *fp,X509_SIG **p8);
711 int i2d_PKCS8_fp(FILE *fp,X509_SIG *p8);
712 PKCS8_PRIV_KEY_INFO *d2i_PKCS8_PRIV_KEY_INFO_fp(FILE *fp,
713                                                 PKCS8_PRIV_KEY_INFO **p8inf);
714 int i2d_PKCS8_PRIV_KEY_INFO_fp(FILE *fp,PKCS8_PRIV_KEY_INFO *p8inf);
715 int i2d_PKCS8PrivateKeyInfo_fp(FILE *fp, EVP_PKEY *key);
716 int i2d_PrivateKey_fp(FILE *fp, EVP_PKEY *pkey);
717 EVP_PKEY *d2i_PrivateKey_fp(FILE *fp, EVP_PKEY **a);
718 int i2d_PUBKEY_fp(FILE *fp, EVP_PKEY *pkey);
719 EVP_PKEY *d2i_PUBKEY_fp(FILE *fp, EVP_PKEY **a);
720 #endif
```



```
722 #ifndef OPENSSL_NO_BIO
723 X509 *d2i_X509_bio(BIO *bp,X509 **x509);
724 int i2d_X509_bio(BIO *bp,X509 *x509);
725 X509_CRL *d2i_X509_CRL_bio(BIO *bp,X509_CRL **crl);
726 int i2d_X509_CRL_bio(BIO *bp,X509_CRL *crl);
727 X509_REQ *d2i_X509_REQ_bio(BIO *bp,X509_REQ **req);
728 int i2d_X509_REQ_bio(BIO *bp,X509_REQ *req);
729 #ifndef OPENSSL_NO_RSA
730 RSA *d2i_RSAPrivateKey_bio(BIO *bp,RSA **rsa);
731 int i2d_RSAPrivateKey_bio(BIO *bp,RSA *rsa);
732 RSA *d2i_RSAPublicKey_bio(BIO *bp,RSA **rsa);
733 int i2d_RSAPublicKey_bio(BIO *bp,RSA *rsa);
734 RSA *d2i_RSA_PUBKEY_bio(BIO *bp,RSA **rsa);
735 int i2d_RSA_PUBKEY_bio(BIO *bp,RSA *rsa);
736 #endif
737 #ifndef OPENSSL_NO_DSA
738 DSA *d2i_DSA_PUBKEY_bio(BIO *bp, DSA **dsa);
739 int i2d_DSA_PUBKEY_bio(BIO *bp, DSA *dsa);
740 DSA *d2i_DSAPrivateKey_bio(BIO *bp, DSA **dsa);
741 int i2d_DSAPrivateKey_bio(BIO *bp, DSA *dsa);
742 #endif
743 #ifndef OPENSSL_NO_EC
744 EC_KEY *d2i_EC_PUBKEY_bio(BIO *bp, EC_KEY **eckey);
745 int   i2d_EC_PUBKEY_bio(BIO *bp, EC_KEY *eckey);
746 EC_KEY *d2i_ECPrivateKey_bio(BIO *bp, EC_KEY **eckey);
747 int   i2d_ECPrivateKey_bio(BIO *bp, EC_KEY *eckey);
748 #endif
749 X509_SIG *d2i_PKCS8_bio(BIO *bp,X509_SIG **p8);
750 int i2d_PKCS8_bio(BIO *bp,X509_SIG *p8);
751 PKCS8_PRIV_KEY_INFO *d2i_PKCS8_PRIV_KEY_INFO_bio(BIO *bp,
752                                                 PKCS8_PRIV_KEY_INFO **p8inf);
753 int i2d_PKCS8_PRIV_KEY_INFO_bio(BIO *bp,PKCS8_PRIV_KEY_INFO *p8inf);
754 int i2d_PKCS8PrivateKeyInfo_bio(BIO *bp, EVP_PKEY *key);
755 int i2d_PrivateKey_bio(BIO *bp, EVP_PKEY *pkey);
756 EVP_PKEY *d2i_PrivateKey_bio(BIO *bp, EVP_PKEY **a);
757 int i2d_PUBKEY_bio(BIO *bp, EVP_PKEY *pkey);
758 EVP_PKEY *d2i_PUBKEY_bio(BIO *bp, EVP_PKEY **a);
759 #endif

761 X509 *X509_dup(X509 *x509);
762 X509_ATTRIBUTE *X509_ATTRIBUTE_dup(X509_ATTRIBUTE *xa);
763 X509_EXTENSION *X509_EXTENSION_dup(X509_EXTENSION *ex);
764 X509_CRL *X509_CRL_dup(X509_CRL *crl);
765 X509_REQ *X509_REQ_dup(X509_REQ *req);
766 X509_ALGOR *X509_ALGOR_dup(X509_ALGOR *xn);
767 int X509_ALGOR_set0(X509_ALGOR *alg, ASN1_OBJECT *aobj, int ptype, void *pval);
768 void X509_ALGOR_get0(ASN1_OBJECT **paobj, int *pptype, void **ppval,
769                                                 X509_ALGOR *algor);
770 void X509_ALGOR_set_md(X509_ALGOR *alg, const EVP_MD *md);

772 X509_NAME *X509_NAME_dup(X509_NAME *xn);
773 X509_NAME_ENTRY *X509_NAME_ENTRY_dup(X509_NAME_ENTRY *ne);

775 int             X509_cmp_time(const ASN1_TIME *s, time_t *t);
776 int             X509_cmp_current_time(const ASN1_TIME *s);
777 ASN1_TIME *     X509_time_adj(ASN1_TIME *s, long adj, time_t *t);
778 ASN1_TIME *     X509_time_adj_ex(ASN1_TIME *s,
779                                 int offset_day, long offset_sec, time_t *t);
780 ASN1_TIME *     X509_gmtime_adj(ASN1_TIME *s, long adj);

782 const char *    X509_get_default_cert_area(void );
783 const char *    X509_get_default_cert_dir(void );
784 const char *    X509_get_default_cert_file(void );
785 const char *    X509_get_default_cert_dir_env(void );
786 const char *    X509_get_default_cert_file_env(void );
787 const char *    X509_get_default_private_dir(void );
```

```
X509_REQ *      X509_to_X509_REQ(X509 *x, EVP_PKEY *pkey, const EVP_MD *md);
X509 *          X509_REQ_to_X509(X509_REQ *r, int days,EVP_PKEY *pkey);

DECLARE_ASN1_FUNCTIONS(X509_ALGOR)
DECLARE_ASN1_ENCODE_FUNCTIONS(X509_ALGORS, X509_ALGORS, X509_ALGORS)
DECLARE_ASN1_FUNCTIONS(X509_VAL)

DECLARE_ASN1_FUNCTIONS(X509_PUBKEY)

int             X509_PUBKEY_set(X509_PUBKEY **x, EVP_PKEY *pkey);
EVP_PKEY *      X509_PUBKEY_get(X509_PUBKEY *key);
int             X509_get_pubkey_parameters(EVP_PKEY *pkey,
                                           STACK_OF(X509) *chain);
int             i2d_PUBKEY(EVP_PKEY *a,unsigned char **pp);
EVP_PKEY *      d2i_PUBKEY(EVP_PKEY **a,const unsigned char **pp,
                        long length);
#ifndef OPENSSL_NO_RSA
int             i2d_RSA_PUBKEY(RSA *a,unsigned char **pp);
RSA *           d2i_RSA_PUBKEY(RSA **a,const unsigned char **pp,
                        long length);
#endif
#ifndef OPENSSL_NO_DSA
int             i2d_DSA_PUBKEY(DSA *a,unsigned char **pp);
DSA *           d2i_DSA_PUBKEY(DSA **a,const unsigned char **pp,
                        long length);
#endif
#ifndef OPENSSL_NO_EC
int             i2d_EC_PUBKEY(EC_KEY *a, unsigned char **pp);
EC_KEY          *d2i_EC_PUBKEY(EC_KEY **a, const unsigned char **pp,
                        long length);
#endif

DECLARE_ASN1_FUNCTIONS(X509_SIG)
DECLARE_ASN1_FUNCTIONS(X509_REQ_INFO)
DECLARE_ASN1_FUNCTIONS(X509_REQ)

DECLARE_ASN1_FUNCTIONS(X509_ATTRIBUTE)
X509_ATTRIBUTE *X509_ATTRIBUTE_create(int nid, int atrtype, void *value);

DECLARE_ASN1_FUNCTIONS(X509_EXTENSION)
DECLARE_ASN1_ENCODE_FUNCTIONS(X509_EXTENSIONS, X509_EXTENSIONS, X509_EXTENSIONS)

DECLARE_ASN1_FUNCTIONS(X509_NAME_ENTRY)

DECLARE_ASN1_FUNCTIONS(X509_NAME)

int             X509_NAME_set(X509_NAME **xn, X509_NAME *name);

DECLARE_ASN1_FUNCTIONS(X509_CINF)

DECLARE_ASN1_FUNCTIONS(X509)
DECLARE_ASN1_FUNCTIONS(X509_CERT_AUX)

DECLARE_ASN1_FUNCTIONS(X509_CERT_PAIR)

int X509_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
             CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
int X509_set_ex_data(X509 *r, int idx, void *arg);
void *X509_get_ex_data(X509 *r, int idx);
int             i2d_X509_AUX(X509 *a,unsigned char **pp);
X509 *          d2i_X509_AUX(X509 **a,const unsigned char **pp,long length);

int X509_alias_set1(X509 *x, unsigned char *name, int len);
int X509_keyid_set1(X509 *x, unsigned char *id, int len);
unsigned char * X509_alias_get0(X509 *x, int *len);
```



```
unsigned char * X509_keyid_get0(X509 *x, int *len);
int (*X509_TRUST_set_default(int (*trust)(int , X509 *, int)))(int, X509 *, int)
int X509_TRUST_set(int *t, int trust);
int X509_add1_trust_object(X509 *x, ASN1_OBJECT *obj);
int X509_add1_reject_object(X509 *x, ASN1_OBJECT *obj);
void X509_trust_clear(X509 *x);
void X509_reject_clear(X509 *x);

DECLARE_ASN1_FUNCTIONS(X509_REVOKED)
DECLARE_ASN1_FUNCTIONS(X509_CRL_INFO)
DECLARE_ASN1_FUNCTIONS(X509_CRL)

int X509_CRL_add0_revoked(X509_CRL *crl, X509_REVOKED *rev);
int X509_CRL_get0_by_serial(X509_CRL *crl,
                X509_REVOKED **ret, ASN1_INTEGER *serial);
int X509_CRL_get0_by_cert(X509_CRL *crl, X509_REVOKED **ret, X509 *x);

X509_PKEY *     X509_PKEY_new(void );
void            X509_PKEY_free(X509_PKEY *a);
int             i2d_X509_PKEY(X509_PKEY *a,unsigned char **pp);
X509_PKEY *     d2i_X509_PKEY(X509_PKEY **a,const unsigned char **pp,long length

DECLARE_ASN1_FUNCTIONS(NETSCAPE_SPKI)
DECLARE_ASN1_FUNCTIONS(NETSCAPE_SPKAC)
DECLARE_ASN1_FUNCTIONS(NETSCAPE_CERT_SEQUENCE)

#ifndef OPENSSL_NO_EVP
X509_INFO *     X509_INFO_new(void);
void            X509_INFO_free(X509_INFO *a);
char *          X509_NAME_oneline(X509_NAME *a,char *buf,int size);

int ASN1_verify(i2d_of_void *i2d, X509_ALGOR *algor1,
                ASN1_BIT_STRING *signature,char *data,EVP_PKEY *pkey);

int ASN1_digest(i2d_of_void *i2d,const EVP_MD *type,char *data,
                unsigned char *md,unsigned int *len);

int ASN1_sign(i2d_of_void *i2d, X509_ALGOR *algor1,
              X509_ALGOR *algor2, ASN1_BIT_STRING *signature,
              char *data,EVP_PKEY *pkey, const EVP_MD *type);

int ASN1_item_digest(const ASN1_ITEM *it,const EVP_MD *type,void *data,
        unsigned char *md,unsigned int *len);

int ASN1_item_verify(const ASN1_ITEM *it, X509_ALGOR *algor1,
        ASN1_BIT_STRING *signature,void *data,EVP_PKEY *pkey);

int ASN1_item_sign(const ASN1_ITEM *it, X509_ALGOR *algor1, X509_ALGOR *algor2,
        ASN1_BIT_STRING *signature,
        void *data, EVP_PKEY *pkey, const EVP_MD *type);
int ASN1_item_sign_ctx(const ASN1_ITEM *it,
                X509_ALGOR *algor1, X509_ALGOR *algor2,
                ASN1_BIT_STRING *signature, void *asn, EVP_MD_CTX *ctx);
#endif

int             X509_set_version(X509 *x,long version);
int             X509_set_serialNumber(X509 *x, ASN1_INTEGER *serial);
ASN1_INTEGER *  X509_get_serialNumber(X509 *x);
int             X509_set_issuer_name(X509 *x, X509_NAME *name);
X509_NAME *     X509_get_issuer_name(X509 *a);
int             X509_set_subject_name(X509 *x, X509_NAME *name);
X509_NAME *     X509_get_subject_name(X509 *a);
int             X509_set_notBefore(X509 *x, const ASN1_TIME *tm);
int             X509_set_notAfter(X509 *x, const ASN1_TIME *tm);
int             X509_set_pubkey(X509 *x, EVP_PKEY *pkey);
EVP_PKEY *      X509_get_pubkey(X509 *x);
```

```
ASN1_BIT_STRING * X509_get0_pubkey_bitstr(const X509 *x);
int             X509_certificate_type(X509 *x,EVP_PKEY *pubkey /* optional */);

int             X509_REQ_set_version(X509_REQ *x,long version);
int             X509_REQ_set_subject_name(X509_REQ *req,X509_NAME *name);
int             X509_REQ_set_pubkey(X509_REQ *x, EVP_PKEY *pkey);
EVP_PKEY *      X509_REQ_get_pubkey(X509_REQ *req);
int             X509_REQ_extension_nid(int nid);
int *           X509_REQ_get_extension_nids(void);
void            X509_REQ_set_extension_nids(int *nids);
STACK_OF(X509_EXTENSION) *X509_REQ_get_extensions(X509_REQ *req);
int X509_REQ_add_extensions_nid(X509_REQ *req, STACK_OF(X509_EXTENSION) *exts,
                                int nid);
int X509_REQ_add_extensions(X509_REQ *req, STACK_OF(X509_EXTENSION) *exts);
int X509_REQ_get_attr_count(const X509_REQ *req);
int X509_REQ_get_attr_by_NID(const X509_REQ *req, int nid,
                             int lastpos);
int X509_REQ_get_attr_by_OBJ(const X509_REQ *req, ASN1_OBJECT *obj,
                             int lastpos);
X509_ATTRIBUTE *X509_REQ_get_attr(const X509_REQ *req, int loc);
X509_ATTRIBUTE *X509_REQ_delete_attr(X509_REQ *req, int loc);
int X509_REQ_add1_attr(X509_REQ *req, X509_ATTRIBUTE *attr);
int X509_REQ_add1_attr_by_OBJ(X509_REQ *req,
                              const ASN1_OBJECT *obj, int type,
                              const unsigned char *bytes, int len);
int X509_REQ_add1_attr_by_NID(X509_REQ *req,
                              int nid, int type,
                              const unsigned char *bytes, int len);
int X509_REQ_add1_attr_by_txt(X509_REQ *req,
                              const char *attrname, int type,
                              const unsigned char *bytes, int len);

int X509_CRL_set_version(X509_CRL *x, long version);
int X509_CRL_set_issuer_name(X509_CRL *x, X509_NAME *name);
int X509_CRL_set_lastUpdate(X509_CRL *x, const ASN1_TIME *tm);
int X509_CRL_set_nextUpdate(X509_CRL *x, const ASN1_TIME *tm);
int X509_CRL_sort(X509_CRL *crl);

int X509_REVOKED_set_serialNumber(X509_REVOKED *x, ASN1_INTEGER *serial);
int X509_REVOKED_set_revocationDate(X509_REVOKED *r, ASN1_TIME *tm);

int             X509_REQ_check_private_key(X509_REQ *x509,EVP_PKEY *pkey);

int             X509_check_private_key(X509 *x509,EVP_PKEY *pkey);

int             X509_issuer_and_serial_cmp(const X509 *a, const X509 *b);
unsigned long   X509_issuer_and_serial_hash(X509 *a);

int             X509_issuer_name_cmp(const X509 *a, const X509 *b);
unsigned long   X509_issuer_name_hash(X509 *a);

int             X509_subject_name_cmp(const X509 *a, const X509 *b);
unsigned long   X509_subject_name_hash(X509 *x);

#ifndef OPENSSL_NO_MD5
unsigned long   X509_issuer_name_hash_old(X509 *a);
unsigned long   X509_subject_name_hash_old(X509 *x);
#endif

int             X509_cmp(const X509 *a, const X509 *b);
int             X509_NAME_cmp(const X509_NAME *a, const X509_NAME *b);
unsigned long   X509_NAME_hash(X509_NAME *x);
unsigned long   X509_NAME_hash_old(X509_NAME *x);

int             X509_CRL_cmp(const X509_CRL *a, const X509_CRL *b);
int             X509_CRL_match(const X509_CRL *a, const X509_CRL *b);
```



```
#ifndef OPENSSL_NO_FP_API
int             X509_print_ex_fp(FILE *bp,X509 *x, unsigned long nmflag, unsigne
int             X509_print_fp(FILE *bp,X509 *x);
int             X509_CRL_print_fp(FILE *bp,X509_CRL *x);
int             X509_REQ_print_fp(FILE *bp,X509_REQ *req);
int X509_NAME_print_ex_fp(FILE *fp, X509_NAME *nm, int indent, unsigned long fla
#endif

#ifndef OPENSSL_NO_BIO
int             X509_NAME_print(BIO *bp, X509_NAME *name, int obase);
int X509_NAME_print_ex(BIO *out, X509_NAME *nm, int indent, unsigned long flags)
int             X509_print_ex(BIO *bp,X509 *x, unsigned long nmflag, unsigned lo
int             X509_print(BIO *bp,X509 *x);
int             X509_ocspid_print(BIO *bp,X509 *x);
int             X509_CERT_AUX_print(BIO *bp,X509_CERT_AUX *x, int indent);
int             X509_CRL_print(BIO *bp,X509_CRL *x);
int             X509_REQ_print_ex(BIO *bp, X509_REQ *x, unsigned long nmflag, un
int             X509_REQ_print(BIO *bp,X509_REQ *req);
#endif

int             X509_NAME_entry_count(X509_NAME *name);
int             X509_NAME_get_text_by_NID(X509_NAME *name, int nid,
                        char *buf,int len);
int             X509_NAME_get_text_by_OBJ(X509_NAME *name, ASN1_OBJECT *obj,
                        char *buf,int len);

/* NOTE: you should be passsing -1, not 0 as lastpos.  The functions that use
 * lastpos, search after that position on. */
int             X509_NAME_get_index_by_NID(X509_NAME *name,int nid,int lastpos);
int             X509_NAME_get_index_by_OBJ(X509_NAME *name,ASN1_OBJECT *obj,
                        int lastpos);
X509_NAME_ENTRY *X509_NAME_get_entry(X509_NAME *name, int loc);
X509_NAME_ENTRY *X509_NAME_delete_entry(X509_NAME *name, int loc);
int             X509_NAME_add_entry(X509_NAME *name,X509_NAME_ENTRY *ne,
                        int loc, int set);
int X509_NAME_add_entry_by_OBJ(X509_NAME *name, ASN1_OBJECT *obj, int type,
                        unsigned char *bytes, int len, int loc, int set);
int X509_NAME_add_entry_by_NID(X509_NAME *name, int nid, int type,
                        unsigned char *bytes, int len, int loc, int set);
X509_NAME_ENTRY *X509_NAME_ENTRY_create_by_txt(X509_NAME_ENTRY **ne,
                const char *field, int type, const unsigned char *bytes, int len
X509_NAME_ENTRY *X509_NAME_ENTRY_create_by_NID(X509_NAME_ENTRY **ne, int nid,
                        int type,unsigned char *bytes, int len);
int X509_NAME_add_entry_by_txt(X509_NAME *name, const char *field, int type,
                        const unsigned char *bytes, int len, int loc, int set);
X509_NAME_ENTRY *X509_NAME_ENTRY_create_by_OBJ(X509_NAME_ENTRY **ne,
                        ASN1_OBJECT *obj, int type,const unsigned char *bytes,
                        int len);
int             X509_NAME_ENTRY_set_object(X509_NAME_ENTRY *ne,
                        ASN1_OBJECT *obj);
int             X509_NAME_ENTRY_set_data(X509_NAME_ENTRY *ne, int type,
                        const unsigned char *bytes, int len);
ASN1_OBJECT *   X509_NAME_ENTRY_get_object(X509_NAME_ENTRY *ne);
ASN1_STRING *   X509_NAME_ENTRY_get_data(X509_NAME_ENTRY *ne);

int             X509v3_get_ext_count(const STACK_OF(X509_EXTENSION) *x);
int             X509v3_get_ext_by_NID(const STACK_OF(X509_EXTENSION) *x,
                                      int nid, int lastpos);
int             X509v3_get_ext_by_OBJ(const STACK_OF(X509_EXTENSION) *x,
                                      ASN1_OBJECT *obj,int lastpos);
int             X509v3_get_ext_by_critical(const STACK_OF(X509_EXTENSION) *x,
                                           int crit, int lastpos);
X509_EXTENSION *X509v3_get_ext(const STACK_OF(X509_EXTENSION) *x, int loc);
X509_EXTENSION *X509v3_delete_ext(STACK_OF(X509_EXTENSION) *x, int loc);
STACK_OF(X509_EXTENSION) *X509v3_add_ext(STACK_OF(X509_EXTENSION) **x,
                                         X509_EXTENSION *ex, int loc);
```

```
int             X509_get_ext_count(X509 *x);
int             X509_get_ext_by_NID(X509 *x, int nid, int lastpos);
int             X509_get_ext_by_OBJ(X509 *x,ASN1_OBJECT *obj,int lastpos);
int             X509_get_ext_by_critical(X509 *x, int crit, int lastpos);
X509_EXTENSION *X509_get_ext(X509 *x, int loc);
X509_EXTENSION *X509_delete_ext(X509 *x, int loc);
int             X509_add_ext(X509 *x, X509_EXTENSION *ex, int loc);
void    *       X509_get_ext_d2i(X509 *x, int nid, int *crit, int *idx);
int             X509_add1_ext_i2d(X509 *x, int nid, void *value, int crit,
                                                        unsigned long flags);

int             X509_CRL_get_ext_count(X509_CRL *x);
int             X509_CRL_get_ext_by_NID(X509_CRL *x, int nid, int lastpos);
int             X509_CRL_get_ext_by_OBJ(X509_CRL *x,ASN1_OBJECT *obj,int lastpos
int             X509_CRL_get_ext_by_critical(X509_CRL *x, int crit, int lastpos)
X509_EXTENSION *X509_CRL_get_ext(X509_CRL *x, int loc);
X509_EXTENSION *X509_CRL_delete_ext(X509_CRL *x, int loc);
int             X509_CRL_add_ext(X509_CRL *x, X509_EXTENSION *ex, int loc);
void    *       X509_CRL_get_ext_d2i(X509_CRL *x, int nid, int *crit, int *idx);
int             X509_CRL_add1_ext_i2d(X509_CRL *x, int nid, void *value, int cri
                                                        unsigned long flags);

int             X509_REVOKED_get_ext_count(X509_REVOKED *x);
int             X509_REVOKED_get_ext_by_NID(X509_REVOKED *x, int nid, int lastpo
int             X509_REVOKED_get_ext_by_OBJ(X509_REVOKED *x,ASN1_OBJECT *obj,int
int             X509_REVOKED_get_ext_by_critical(X509_REVOKED *x, int crit, int
X509_EXTENSION *X509_REVOKED_get_ext(X509_REVOKED *x, int loc);
X509_EXTENSION *X509_REVOKED_delete_ext(X509_REVOKED *x, int loc);
int             X509_REVOKED_add_ext(X509_REVOKED *x, X509_EXTENSION *ex, int lo
void    *       X509_REVOKED_get_ext_d2i(X509_REVOKED *x, int nid, int *crit, in
int             X509_REVOKED_add1_ext_i2d(X509_REVOKED *x, int nid, void *value,
                                                        unsigned long flags);

X509_EXTENSION *X509_EXTENSION_create_by_NID(X509_EXTENSION **ex,
                        int nid, int crit, ASN1_OCTET_STRING *data);
X509_EXTENSION *X509_EXTENSION_create_by_OBJ(X509_EXTENSION **ex,
                        ASN1_OBJECT *obj,int crit,ASN1_OCTET_STRING *data);
int             X509_EXTENSION_set_object(X509_EXTENSION *ex,ASN1_OBJECT *obj);
int             X509_EXTENSION_set_critical(X509_EXTENSION *ex, int crit);
int             X509_EXTENSION_set_data(X509_EXTENSION *ex,
                        ASN1_OCTET_STRING *data);
ASN1_OBJECT *   X509_EXTENSION_get_object(X509_EXTENSION *ex);
ASN1_OCTET_STRING *X509_EXTENSION_get_data(X509_EXTENSION *ne);
int             X509_EXTENSION_get_critical(X509_EXTENSION *ex);

int X509at_get_attr_count(const STACK_OF(X509_ATTRIBUTE) *x);
int X509at_get_attr_by_NID(const STACK_OF(X509_ATTRIBUTE) *x, int nid,
                           int lastpos);
int X509at_get_attr_by_OBJ(const STACK_OF(X509_ATTRIBUTE) *sk, ASN1_OBJECT *obj,
                           int lastpos);
X509_ATTRIBUTE *X509at_get_attr(const STACK_OF(X509_ATTRIBUTE) *x, int loc);
X509_ATTRIBUTE *X509at_delete_attr(STACK_OF(X509_ATTRIBUTE) *x, int loc);
STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr(STACK_OF(X509_ATTRIBUTE) **x,
                                        X509_ATTRIBUTE *attr);
STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr_by_OBJ(STACK_OF(X509_ATTRIBUTE) **x,
                        const ASN1_OBJECT *obj, int type,
                        const unsigned char *bytes, int len);
STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr_by_NID(STACK_OF(X509_ATTRIBUTE) **x,
                        int nid, int type,
                        const unsigned char *bytes, int len);
STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr_by_txt(STACK_OF(X509_ATTRIBUTE) **x,
                        const char *attrname, int type,
                        const unsigned char *bytes, int len);
void *X509at_get0_data_by_OBJ(STACK_OF(X509_ATTRIBUTE) *x,
                                ASN1_OBJECT *obj, int lastpos, int type);
```



```
X509_ATTRIBUTE *X509_ATTRIBUTE_create_by_NID(X509_ATTRIBUTE **attr, int nid,
             int atrtype, const void *data, int len);
X509_ATTRIBUTE *X509_ATTRIBUTE_create_by_OBJ(X509_ATTRIBUTE **attr,
             const ASN1_OBJECT *obj, int atrtype, const void *data, int len);
X509_ATTRIBUTE *X509_ATTRIBUTE_create_by_txt(X509_ATTRIBUTE **attr,
                const char *atrname, int type, const unsigned char *bytes, int l
int X509_ATTRIBUTE_set1_object(X509_ATTRIBUTE *attr, const ASN1_OBJECT *obj);
int X509_ATTRIBUTE_set1_data(X509_ATTRIBUTE *attr, int attrtype, const void *dat
void *X509_ATTRIBUTE_get0_data(X509_ATTRIBUTE *attr, int idx,
                                        int atrtype, void *data);
int X509_ATTRIBUTE_count(X509_ATTRIBUTE *attr);
ASN1_OBJECT *X509_ATTRIBUTE_get0_object(X509_ATTRIBUTE *attr);
ASN1_TYPE *X509_ATTRIBUTE_get0_type(X509_ATTRIBUTE *attr, int idx);

int EVP_PKEY_get_attr_count(const EVP_PKEY *key);
int EVP_PKEY_get_attr_by_NID(const EVP_PKEY *key, int nid,
                             int lastpos);
int EVP_PKEY_get_attr_by_OBJ(const EVP_PKEY *key, ASN1_OBJECT *obj,
                             int lastpos);
X509_ATTRIBUTE *EVP_PKEY_get_attr(const EVP_PKEY *key, int loc);
X509_ATTRIBUTE *EVP_PKEY_delete_attr(EVP_PKEY *key, int loc);
int EVP_PKEY_add1_attr(EVP_PKEY *key, X509_ATTRIBUTE *attr);
int EVP_PKEY_add1_attr_by_OBJ(EVP_PKEY *key,
                          const ASN1_OBJECT *obj, int type,
                          const unsigned char *bytes, int len);
int EVP_PKEY_add1_attr_by_NID(EVP_PKEY *key,
                          int nid, int type,
                          const unsigned char *bytes, int len);
int EVP_PKEY_add1_attr_by_txt(EVP_PKEY *key,
                          const char *attrname, int type,
                          const unsigned char *bytes, int len);

int             X509_verify_cert(X509_STORE_CTX *ctx);

/* lookup a cert from a X509 STACK */
X509 *X509_find_by_issuer_and_serial(STACK_OF(X509) *sk,X509_NAME *name,
                                     ASN1_INTEGER *serial);
X509 *X509_find_by_subject(STACK_OF(X509) *sk,X509_NAME *name);

DECLARE_ASN1_FUNCTIONS(PBEPARAM)
DECLARE_ASN1_FUNCTIONS(PBE2PARAM)
DECLARE_ASN1_FUNCTIONS(PBKDF2PARAM)

int PKCS5_pbe_set0_algor(X509_ALGOR *algor, int alg, int iter,
                                 const unsigned char *salt, int saltlen);

X509_ALGOR *PKCS5_pbe_set(int alg, int iter,
                                 const unsigned char *salt, int saltlen);
X509_ALGOR *PKCS5_pbe2_set(const EVP_CIPHER *cipher, int iter,
                                   unsigned char *salt, int saltlen);
X509_ALGOR *PKCS5_pbe2_set_iv(const EVP_CIPHER *cipher, int iter,
                                  unsigned char *salt, int saltlen,
                                  unsigned char *aiv, int prf_nid);

X509_ALGOR *PKCS5_pbkdf2_set(int iter, unsigned char *salt, int saltlen,
                                 int prf_nid, int keylen);

/* PKCS#8 utilities */

DECLARE_ASN1_FUNCTIONS(PKCS8_PRIV_KEY_INFO)

EVP_PKEY *EVP_PKCS82PKEY(PKCS8_PRIV_KEY_INFO *p8);
PKCS8_PRIV_KEY_INFO *EVP_PKEY2PKCS8(EVP_PKEY *pkey);
PKCS8_PRIV_KEY_INFO *EVP_PKEY2PKCS8_broken(EVP_PKEY *pkey, int broken);
PKCS8_PRIV_KEY_INFO *PKCS8_set_broken(PKCS8_PRIV_KEY_INFO *p8, int broken);
```

```
int PKCS8_pkey_set0(PKCS8_PRIV_KEY_INFO *priv, ASN1_OBJECT *aobj,
                        int version, int ptype, void *pval,
                                unsigned char *penc, int penclen);
int PKCS8_pkey_get0(ASN1_OBJECT **ppkalg,
                const unsigned char **pk, int *ppklen,
                X509_ALGOR **pa,
                PKCS8_PRIV_KEY_INFO *p8);

int X509_PUBKEY_set0_param(X509_PUBKEY *pub, ASN1_OBJECT *aobj,
                                    int ptype, void *pval,
                                    unsigned char *penc, int penclen);
int X509_PUBKEY_get0_param(ASN1_OBJECT **ppkalg,
                const unsigned char **pk, int *ppklen,
                X509_ALGOR **pa,
                X509_PUBKEY *pub);

int X509_check_trust(X509 *x, int id, int flags);
int X509_TRUST_get_count(void);
X509_TRUST * X509_TRUST_get0(int idx);
int X509_TRUST_get_by_id(int id);
int X509_TRUST_add(int id, int flags, int (*ck)(X509_TRUST *, X509 *, int),
                                    char *name, int arg1, void *arg2);
void X509_TRUST_cleanup(void);
int X509_TRUST_get_flags(X509_TRUST *xp);
char *X509_TRUST_get0_name(X509_TRUST *xp);
int X509_TRUST_get_trust(X509_TRUST *xp);

/* BEGIN ERROR CODES */
/* The following lines are auto generated by the script mkerr.pl. Any changes
 * made after this point may be overwritten when the script is next run.
 */
void ERR_load_X509_strings(void);

/* Error codes for the X509 functions. */

/* Function codes. */
#define X509_F_ADD_CERT_DIR                             100
#define X509_F_BY_FILE_CTRL                             101
#define X509_F_CHECK_POLICY                             145
#define X509_F_DIR_CTRL                                 102
#define X509_F_GET_CERT_BY_SUBJECT                      103
#define X509_F_NETSCAPE_SPKI_B64_DECODE                 129
#define X509_F_NETSCAPE_SPKI_B64_ENCODE                 130
#define X509_F_X509AT_ADD1_ATTR                         135
#define X509_F_X509V3_ADD_EXT                           104
#define X509_F_X509_ATTRIBUTE_CREATE_BY_NID             136
#define X509_F_X509_ATTRIBUTE_CREATE_BY_OBJ             137
#define X509_F_X509_ATTRIBUTE_CREATE_BY_TXT             140
#define X509_F_X509_ATTRIBUTE_GET0_DATA                 139
#define X509_F_X509_ATTRIBUTE_SET1_DATA                 138
#define X509_F_X509_CHECK_PRIVATE_KEY                   128
#define X509_F_X509_CRL_PRINT_FP                        147
#define X509_F_X509_EXTENSION_CREATE_BY_NID             108
#define X509_F_X509_EXTENSION_CREATE_BY_OBJ             109
#define X509_F_X509_GET_PUBKEY_PARAMETERS               110
#define X509_F_X509_LOAD_CERT_CRL_FILE                  132
#define X509_F_X509_LOAD_CERT_FILE                      111
#define X509_F_X509_LOAD_CRL_FILE                       112
#define X509_F_X509_NAME_ADD_ENTRY                      113
#define X509_F_X509_NAME_ENTRY_CREATE_BY_NID            114
#define X509_F_X509_NAME_ENTRY_CREATE_BY_TXT            131
#define X509_F_X509_NAME_ENTRY_SET_OBJECT               115
#define X509_F_X509_NAME_ONELINE                        116
#define X509_F_X509_NAME_PRINT                          117
#define X509_F_X509_PRINT_EX_FP                         118
#define X509_F_X509_PUBKEY_GET                          119
```



```
#define X509_F_X509_PUBKEY_SET                          120
#define X509_F_X509_REQ_CHECK_PRIVATE_KEY               144
#define X509_F_X509_REQ_PRINT_EX                        121
#define X509_F_X509_REQ_PRINT_FP                        122
#define X509_F_X509_REQ_TO_X509                         123
#define X509_F_X509_STORE_ADD_CERT                      124
#define X509_F_X509_STORE_ADD_CRL                       125
#define X509_F_X509_STORE_CTX_GET1_ISSUER               146
#define X509_F_X509_STORE_CTX_INIT                      143
#define X509_F_X509_STORE_CTX_NEW                       142
#define X509_F_X509_STORE_CTX_PURPOSE_INHERIT           134
#define X509_F_X509_TO_X509_REQ                         126
#define X509_F_X509_TRUST_ADD                           133
#define X509_F_X509_TRUST_SET                           141
#define X509_F_X509_VERIFY_CERT                         127

/* Reason codes. */
#define X509_R_BAD_X509_FILETYPE                        100
#define X509_R_BASE64_DECODE_ERROR                      118
#define X509_R_CANT_CHECK_DH_KEY                        114
#define X509_R_CERT_ALREADY_IN_HASH_TABLE               101
#define X509_R_ERR_ASN1_LIB                             102
#define X509_R_INVALID_DIRECTORY                        113
#define X509_R_INVALID_FIELD_NAME                       119
#define X509_R_INVALID_TRUST                            123
#define X509_R_KEY_TYPE_MISMATCH                        115
#define X509_R_KEY_VALUES_MISMATCH                      116
#define X509_R_LOADING_CERT_DIR                         103
#define X509_R_LOADING_DEFAULTS                         104
#define X509_R_METHOD_NOT_SUPPORTED                     124
#define X509_R_NO_CERT_SET_FOR_US_TO_VERIFY             105
#define X509_R_PUBLIC_KEY_DECODE_ERROR                  125
#define X509_R_PUBLIC_KEY_ENCODE_ERROR                  126
#define X509_R_SHOULD_RETRY                             106
#define X509_R_UNABLE_TO_FIND_PARAMETERS_IN_CHAIN       107
#define X509_R_UNABLE_TO_GET_CERTS_PUBLIC_KEY           108
#define X509_R_UNKNOWN_KEY_TYPE                         117
#define X509_R_UNKNOWN_NID                              109
#define X509_R_UNKNOWN_PURPOSE_ID                       121
#define X509_R_UNKNOWN_TRUST_ID                         120
#define X509_R_UNSUPPORTED_ALGORITHM                    111
#define X509_R_WRONG_LOOKUP_TYPE                        112
#define X509_R_WRONG_TYPE                               122

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
   22388 Wed Aug 13 19:51:51 2014
new/usr/src/lib/openssl/include/openssl/x509_vfy.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_vfy.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef HEADER_X509_H
#include <openssl/x509.h>
/* openssl/x509.h ends up #include-ing this file at about the only
```



```
 * appropriate moment. */
#endif

#ifndef HEADER_X509_VFY_H
#define HEADER_X509_VFY_H

#include <openssl/opensslconf.h>
#ifndef OPENSSL_NO_LHASH
#include <openssl/lhash.h>
#endif
#include <openssl/bio.h>
#include <openssl/crypto.h>
#include <openssl/symhacks.h>

#ifdef  __cplusplus
extern "C" {
#endif

#if 0
/* Outer object */
typedef struct x509_hash_dir_st
        {
        int num_dirs;
        char **dirs;
        int *dirs_type;
        int num_dirs_alloced;
        } X509_HASH_DIR_CTX;
#endif

typedef struct x509_file_st
        {
        int num_paths;  /* number of paths to files or directories */
        int num_alloced;
        char **paths;   /* the list of paths or directories */
        int *path_type;
        } X509_CERT_FILE_CTX;

/*******************************/
/*
SSL_CTX -> X509_STORE
                -> X509_LOOKUP
                        ->X509_LOOKUP_METHOD
                -> X509_LOOKUP
                        ->X509_LOOKUP_METHOD

SSL     -> X509_STORE_CTX
                ->X509_STORE

The X509_STORE holds the tables etc for verification stuff.
A X509_STORE_CTX is used while validating a single certificate.
The X509_STORE has X509_LOOKUPs for looking up certs.
The X509_STORE then calls a function to actually verify the
certificate chain.
*/

#define X509_LU_RETRY           -1
#define X509_LU_FAIL            0
#define X509_LU_X509            1
#define X509_LU_CRL             2
#define X509_LU_PKEY            3

typedef struct x509_object_st
        {
        /* one of the above types */
        int type;
        union   {
```

```
128                 char *ptr;
129                 X509 *x509;
130                 X509_CRL *crl;
131                 EVP_PKEY *pkey;
132                 } data;
133         } X509_OBJECT;

135 typedef struct x509_lookup_st X509_LOOKUP;

137 DECLARE_STACK_OF(X509_LOOKUP)
138 DECLARE_STACK_OF(X509_OBJECT)

140 /* This is a static that defines the function interface */
141 typedef struct x509_lookup_method_st
142         {
143         const char *name;
144         int (*new_item)(X509_LOOKUP *ctx);
145         void (*free)(X509_LOOKUP *ctx);
146         int (*init)(X509_LOOKUP *ctx);
147         int (*shutdown)(X509_LOOKUP *ctx);
148         int (*ctrl)(X509_LOOKUP *ctx,int cmd,const char *argc,long argl,
149                     char **ret);
150         int (*get_by_subject)(X509_LOOKUP *ctx,int type,X509_NAME *name,
151                               X509_OBJECT *ret);
152         int (*get_by_issuer_serial)(X509_LOOKUP *ctx,int type,X509_NAME *name,
153                                     ASN1_INTEGER *serial,X509_OBJECT *ret);
154         int (*get_by_fingerprint)(X509_LOOKUP *ctx,int type,
155                                   unsigned char *bytes,int len,
156                                   X509_OBJECT *ret);
157         int (*get_by_alias)(X509_LOOKUP *ctx,int type,char *str,int len,
158                             X509_OBJECT *ret);
159         } X509_LOOKUP_METHOD;

161 /* This structure hold all parameters associated with a verify operation
162  * by including an X509_VERIFY_PARAM structure in related structures the
163  * parameters used can be customized
164  */

166 typedef struct X509_VERIFY_PARAM_st
167         {
168         char *name;
169         time_t check_time;      /* Time to use */
170         unsigned long inh_flags; /* Inheritance flags */
171         unsigned long flags;    /* Various verify flags */
172         int purpose;            /* purpose to check untrusted certificates */
173         int trust;              /* trust setting to check */
174         int depth;              /* Verify depth */
175         STACK_OF(ASN1_OBJECT) *policies;        /* Permissible policies */
176         } X509_VERIFY_PARAM;

178 DECLARE_STACK_OF(X509_VERIFY_PARAM)

180 /* This is used to hold everything.  It is used for all certificate
181  * validation.  Once we have a certificate chain, the 'verify'
182  * function is then called to actually check the cert chain. */
183 struct x509_store_st
184         {
185         /* The following is a cache of trusted certs */
186         int cache;      /* if true, stash any hits */
187         STACK_OF(X509_OBJECT) *objs;    /* Cache of all objects */

189         /* These are external lookup methods */
190         STACK_OF(X509_LOOKUP) *get_cert_methods;

192         X509_VERIFY_PARAM *param;
```



```
194         /* Callbacks for various operations */
195         int (*verify)(X509_STORE_CTX *ctx);     /* called to verify a certificat
196         int (*verify_cb)(int ok,X509_STORE_CTX *ctx);   /* error callback */
197         int (*get_issuer)(X509 **issuer, X509_STORE_CTX *ctx, X509 *x); /* get i
198         int (*check_issued)(X509_STORE_CTX *ctx, X509 *x, X509 *issuer); /* chec
199         int (*check_revocation)(X509_STORE_CTX *ctx); /* Check revocation status
200         int (*get_crl)(X509_STORE_CTX *ctx, X509_CRL **crl, X509 *x); /* retriev
201         int (*check_crl)(X509_STORE_CTX *ctx, X509_CRL *crl); /* Check CRL valid
202         int (*cert_crl)(X509_STORE_CTX *ctx, X509_CRL *crl, X509 *x); /* Check c
203         STACK_OF(X509) * (*lookup_certs)(X509_STORE_CTX *ctx, X509_NAME *nm);
204         STACK_OF(X509_CRL) * (*lookup_crls)(X509_STORE_CTX *ctx, X509_NAME *nm);
205         int (*cleanup)(X509_STORE_CTX *ctx);

207         CRYPTO_EX_DATA ex_data;
208         int references;
209         } /* X509_STORE */;

211 int X509_STORE_set_depth(X509_STORE *store, int depth);

213 #define X509_STORE_set_verify_cb_func(ctx,func) ((ctx)->verify_cb=(func))
214 #define X509_STORE_set_verify_func(ctx,func)    ((ctx)->verify=(func))

216 /* This is the functions plus an instance of the local variables. */
217 struct x509_lookup_st
218         {
219         int init;                       /* have we been started */
220         int skip;                       /* don't use us. */
221         X509_LOOKUP_METHOD *method;     /* the functions */
222         char *method_data;              /* method data */

224         X509_STORE *store_ctx;  /* who owns us */
225         } /* X509_LOOKUP */;

227 /* This is a used when verifying cert chains.  Since the
228  * gathering of the cert chain can take some time (and have to be
229  * 'retried', this needs to be kept and passed around. */
230 struct x509_store_ctx_st      /* X509_STORE_CTX */
231         {
232         X509_STORE *ctx;
233         int current_method;     /* used when looking up certs */

235         /* The following are set by the caller */
236         X509 *cert;             /* The cert to check */
237         STACK_OF(X509) *untrusted;      /* chain of X509s - untrusted - passed i
238         STACK_OF(X509_CRL) *crls;       /* set of CRLs passed in */

240         X509_VERIFY_PARAM *param;
241         void *other_ctx;        /* Other info for use with get_issuer() */

243         /* Callbacks for various operations */
244         int (*verify)(X509_STORE_CTX *ctx);     /* called to verify a certificat
245         int (*verify_cb)(int ok,X509_STORE_CTX *ctx);           /* error callbac
246         int (*get_issuer)(X509 **issuer, X509_STORE_CTX *ctx, X509 *x); /* get i
247         int (*check_issued)(X509_STORE_CTX *ctx, X509 *x, X509 *issuer); /* chec
248         int (*check_revocation)(X509_STORE_CTX *ctx); /* Check revocation status
249         int (*get_crl)(X509_STORE_CTX *ctx, X509_CRL **crl, X509 *x); /* retriev
250         int (*check_crl)(X509_STORE_CTX *ctx, X509_CRL *crl); /* Check CRL valid
251         int (*cert_crl)(X509_STORE_CTX *ctx, X509_CRL *crl, X509 *x); /* Check c
252         int (*check_policy)(X509_STORE_CTX *ctx);
253         STACK_OF(X509) * (*lookup_certs)(X509_STORE_CTX *ctx, X509_NAME *nm);
254         STACK_OF(X509_CRL) * (*lookup_crls)(X509_STORE_CTX *ctx, X509_NAME *nm);
255         int (*cleanup)(X509_STORE_CTX *ctx);

257         /* The following is built up */
258         int valid;              /* if 0, rebuild chain */
259         int last_untrusted;     /* index of last untrusted cert */
```

```
260         STACK_OF(X509) *chain;          /* chain of X509s - built up and trusted
261         X509_POLICY_TREE *tree; /* Valid policy tree */

263         int explicit_policy;    /* Require explicit policy value */

265         /* When something goes wrong, this is why */
266         int error_depth;
267         int error;
268         X509 *current_cert;
269         X509 *current_issuer;   /* cert currently being tested as valid issuer *
270         X509_CRL *current_crl;  /* current CRL */

272         int current_crl_score;  /* score of current CRL */
273         unsigned int current_reasons;  /* Reason mask */

275         X509_STORE_CTX *parent; /* For CRL path validation: parent context */

277         CRYPTO_EX_DATA ex_data;
278         } /* X509_STORE_CTX */;

280 void X509_STORE_CTX_set_depth(X509_STORE_CTX *ctx, int depth);

282 #define X509_STORE_CTX_set_app_data(ctx,data) \
283         X509_STORE_CTX_set_ex_data(ctx,0,data)
284 #define X509_STORE_CTX_get_app_data(ctx) \
285         X509_STORE_CTX_get_ex_data(ctx,0)

287 #define X509_L_FILE_LOAD        1
288 #define X509_L_ADD_DIR          2

290 #define X509_LOOKUP_load_file(x,name,type) \
291                 X509_LOOKUP_ctrl((x),X509_L_FILE_LOAD,(name),(long)(type),NULL)

293 #define X509_LOOKUP_add_dir(x,name,type) \
294                 X509_LOOKUP_ctrl((x),X509_L_ADD_DIR,(name),(long)(type),NULL)

296 #define         X509_V_OK                                       0
297 /* illegal error (for uninitialized values, to avoid X509_V_OK): 1 */

299 #define         X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT            2
300 #define         X509_V_ERR_UNABLE_TO_GET_CRL                    3
301 #define         X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE     4
302 #define         X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE      5
303 #define         X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY   6
304 #define         X509_V_ERR_CERT_SIGNATURE_FAILURE               7
305 #define         X509_V_ERR_CRL_SIGNATURE_FAILURE                8
306 #define         X509_V_ERR_CERT_NOT_YET_VALID                   9
307 #define         X509_V_ERR_CERT_HAS_EXPIRED                     10
308 #define         X509_V_ERR_CRL_NOT_YET_VALID                    11
309 #define         X509_V_ERR_CRL_HAS_EXPIRED                      12
310 #define         X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD       13
311 #define         X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD        14
312 #define         X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD       15
313 #define         X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD       16
314 #define         X509_V_ERR_OUT_OF_MEM                           17
315 #define         X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT          18
316 #define         X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN            19
317 #define         X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY    20
318 #define         X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE      21
319 #define         X509_V_ERR_CERT_CHAIN_TOO_LONG                  22
320 #define         X509_V_ERR_CERT_REVOKED                         23
321 #define         X509_V_ERR_INVALID_CA                           24
322 #define         X509_V_ERR_PATH_LENGTH_EXCEEDED                 25
323 #define         X509_V_ERR_INVALID_PURPOSE                      26
324 #define         X509_V_ERR_CERT_UNTRUSTED                       27
325 #define         X509_V_ERR_CERT_REJECTED                        28
```



```
326 /* These are 'informational' when looking for issuer cert */
327 #define         X509_V_ERR_SUBJECT_ISSUER_MISMATCH              29
328 #define         X509_V_ERR_AKID_SKID_MISMATCH                   30
329 #define         X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH          31
330 #define         X509_V_ERR_KEYUSAGE_NO_CERTSIGN                 32

332 #define         X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER             33
333 #define         X509_V_ERR_UNHANDLED_CRITICAL_EXTENSION         34
334 #define         X509_V_ERR_KEYUSAGE_NO_CRL_SIGN                 35
335 #define         X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION     36
336 #define         X509_V_ERR_INVALID_NON_CA                       37
337 #define         X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED           38
338 #define         X509_V_ERR_KEYUSAGE_NO_DIGITAL_SIGNATURE        39
339 #define         X509_V_ERR_PROXY_CERTIFICATES_NOT_ALLOWED       40

341 #define         X509_V_ERR_INVALID_EXTENSION                    41
342 #define         X509_V_ERR_INVALID_POLICY_EXTENSION             42
343 #define         X509_V_ERR_NO_EXPLICIT_POLICY                   43
344 #define         X509_V_ERR_DIFFERENT_CRL_SCOPE                  44
345 #define         X509_V_ERR_UNSUPPORTED_EXTENSION_FEATURE        45

347 #define         X509_V_ERR_UNNESTED_RESOURCE                    46

349 #define         X509_V_ERR_PERMITTED_VIOLATION                  47
350 #define         X509_V_ERR_EXCLUDED_VIOLATION                   48
351 #define         X509_V_ERR_SUBTREE_MINMAX                       49
352 #define         X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE          51
353 #define         X509_V_ERR_UNSUPPORTED_CONSTRAINT_SYNTAX        52
354 #define         X509_V_ERR_UNSUPPORTED_NAME_SYNTAX              53
355 #define         X509_V_ERR_CRL_PATH_VALIDATION_ERROR            54

357 /* The application is not happy */
358 #define         X509_V_ERR_APPLICATION_VERIFICATION             50

360 /* Certificate verify flags */

362 /* Send issuer+subject checks to verify_cb */
363 #define X509_V_FLAG_CB_ISSUER_CHECK             0x1
364 /* Use check time instead of current time */
365 #define X509_V_FLAG_USE_CHECK_TIME              0x2
366 /* Lookup CRLs */
367 #define X509_V_FLAG_CRL_CHECK                   0x4
368 /* Lookup CRLs for whole chain */
369 #define X509_V_FLAG_CRL_CHECK_ALL               0x8
370 /* Ignore unhandled critical extensions */
371 #define X509_V_FLAG_IGNORE_CRITICAL             0x10
372 /* Disable workarounds for broken certificates */
373 #define X509_V_FLAG_X509_STRICT                 0x20
374 /* Enable proxy certificate validation */
375 #define X509_V_FLAG_ALLOW_PROXY_CERTS           0x40
376 /* Enable policy checking */
377 #define X509_V_FLAG_POLICY_CHECK                0x80
378 /* Policy variable require-explicit-policy */
379 #define X509_V_FLAG_EXPLICIT_POLICY             0x100
380 /* Policy variable inhibit-any-policy */
381 #define X509_V_FLAG_INHIBIT_ANY                 0x200
382 /* Policy variable inhibit-policy-mapping */
383 #define X509_V_FLAG_INHIBIT_MAP                 0x400
384 /* Notify callback that policy is OK */
385 #define X509_V_FLAG_NOTIFY_POLICY               0x800
386 /* Extended CRL features such as indirect CRLs, alternate CRL signing keys */
387 #define X509_V_FLAG_EXTENDED_CRL_SUPPORT        0x1000
388 /* Delta CRL support */
389 #define X509_V_FLAG_USE_DELTAS                  0x2000
390 /* Check selfsigned CA signature */
391 #define X509_V_FLAG_CHECK_SS_SIGNATURE          0x4000
```

```
#define X509_VP_FLAG_DEFAULT                    0x1
#define X509_VP_FLAG_OVERWRITE                  0x2
#define X509_VP_FLAG_RESET_FLAGS                0x4
#define X509_VP_FLAG_LOCKED                     0x8
#define X509_VP_FLAG_ONCE                       0x10

/* Internal use: mask of policy related options */
#define X509_V_FLAG_POLICY_MASK (X509_V_FLAG_POLICY_CHECK \
                                 | X509_V_FLAG_EXPLICIT_POLICY \
                                 | X509_V_FLAG_INHIBIT_ANY \
                                 | X509_V_FLAG_INHIBIT_MAP)

int X509_OBJECT_idx_by_subject(STACK_OF(X509_OBJECT) *h, int type,
               X509_NAME *name);
X509_OBJECT *X509_OBJECT_retrieve_by_subject(STACK_OF(X509_OBJECT) *h,int type,X
X509_OBJECT *X509_OBJECT_retrieve_match(STACK_OF(X509_OBJECT) *h, X509_OBJECT *x
void X509_OBJECT_up_ref_count(X509_OBJECT *a);
void X509_OBJECT_free_contents(X509_OBJECT *a);
X509_STORE *X509_STORE_new(void );
void X509_STORE_free(X509_STORE *v);

STACK_OF(X509)* X509_STORE_get1_certs(X509_STORE_CTX *st, X509_NAME *nm);
STACK_OF(X509_CRL)* X509_STORE_get1_crls(X509_STORE_CTX *st, X509_NAME *nm);
int X509_STORE_set_flags(X509_STORE *ctx, unsigned long flags);
int X509_STORE_set_purpose(X509_STORE *ctx, int purpose);
int X509_STORE_set_trust(X509_STORE *ctx, int trust);
int X509_STORE_set1_param(X509_STORE *ctx, X509_VERIFY_PARAM *pm);

void X509_STORE_set_verify_cb(X509_STORE *ctx,
                              int (*verify_cb)(int, X509_STORE_CTX *));

X509_STORE_CTX *X509_STORE_CTX_new(void);

int X509_STORE_CTX_get1_issuer(X509 **issuer, X509_STORE_CTX *ctx, X509 *x);

void X509_STORE_CTX_free(X509_STORE_CTX *ctx);
int X509_STORE_CTX_init(X509_STORE_CTX *ctx, X509_STORE *store,
                        X509 *x509, STACK_OF(X509) *chain);
void X509_STORE_CTX_trusted_stack(X509_STORE_CTX *ctx, STACK_OF(X509) *sk);
void X509_STORE_CTX_cleanup(X509_STORE_CTX *ctx);

X509_LOOKUP *X509_STORE_add_lookup(X509_STORE *v, X509_LOOKUP_METHOD *m);

X509_LOOKUP_METHOD *X509_LOOKUP_hash_dir(void);
X509_LOOKUP_METHOD *X509_LOOKUP_file(void);

int X509_STORE_add_cert(X509_STORE *ctx, X509 *x);
int X509_STORE_add_crl(X509_STORE *ctx, X509_CRL *x);

int X509_STORE_get_by_subject(X509_STORE_CTX *vs,int type,X509_NAME *name,
        X509_OBJECT *ret);

int X509_LOOKUP_ctrl(X509_LOOKUP *ctx, int cmd, const char *argc,
        long argl, char **ret);

#ifndef OPENSSL_NO_STDIO
int X509_load_cert_file(X509_LOOKUP *ctx, const char *file, int type);
int X509_load_crl_file(X509_LOOKUP *ctx, const char *file, int type);
int X509_load_cert_crl_file(X509_LOOKUP *ctx, const char *file, int type);
#endif


X509_LOOKUP *X509_LOOKUP_new(X509_LOOKUP_METHOD *method);
void X509_LOOKUP_free(X509_LOOKUP *ctx);
```



```
int X509_LOOKUP_init(X509_LOOKUP *ctx);
int X509_LOOKUP_by_subject(X509_LOOKUP *ctx, int type, X509_NAME *name,
        X509_OBJECT *ret);
int X509_LOOKUP_by_issuer_serial(X509_LOOKUP *ctx, int type, X509_NAME *name,
        ASN1_INTEGER *serial, X509_OBJECT *ret);
int X509_LOOKUP_by_fingerprint(X509_LOOKUP *ctx, int type,
        unsigned char *bytes, int len, X509_OBJECT *ret);
int X509_LOOKUP_by_alias(X509_LOOKUP *ctx, int type, char *str,
        int len, X509_OBJECT *ret);
int X509_LOOKUP_shutdown(X509_LOOKUP *ctx);

#ifndef OPENSSL_NO_STDIO
int     X509_STORE_load_locations (X509_STORE *ctx,
                const char *file, const char *dir);
int     X509_STORE_set_default_paths(X509_STORE *ctx);
#endif

int X509_STORE_CTX_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_fu
        CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func);
int     X509_STORE_CTX_set_ex_data(X509_STORE_CTX *ctx,int idx,void *data);
void *  X509_STORE_CTX_get_ex_data(X509_STORE_CTX *ctx,int idx);
int     X509_STORE_CTX_get_error(X509_STORE_CTX *ctx);
void    X509_STORE_CTX_set_error(X509_STORE_CTX *ctx,int s);
int     X509_STORE_CTX_get_error_depth(X509_STORE_CTX *ctx);
X509 *  X509_STORE_CTX_get_current_cert(X509_STORE_CTX *ctx);
X509 *X509_STORE_CTX_get0_current_issuer(X509_STORE_CTX *ctx);
X509_CRL *X509_STORE_CTX_get0_current_crl(X509_STORE_CTX *ctx);
X509_STORE_CTX *X509_STORE_CTX_get0_parent_ctx(X509_STORE_CTX *ctx);
STACK_OF(X509) *X509_STORE_CTX_get_chain(X509_STORE_CTX *ctx);
STACK_OF(X509) *X509_STORE_CTX_get1_chain(X509_STORE_CTX *ctx);
void    X509_STORE_CTX_set_cert(X509_STORE_CTX *c,X509 *x);
void    X509_STORE_CTX_set_chain(X509_STORE_CTX *c,STACK_OF(X509) *sk);
void    X509_STORE_CTX_set0_crls(X509_STORE_CTX *c,STACK_OF(X509_CRL) *sk);
int X509_STORE_CTX_set_purpose(X509_STORE_CTX *ctx, int purpose);
int X509_STORE_CTX_set_trust(X509_STORE_CTX *ctx, int trust);
int X509_STORE_CTX_purpose_inherit(X509_STORE_CTX *ctx, int def_purpose,
                                   int purpose, int trust);
void X509_STORE_CTX_set_flags(X509_STORE_CTX *ctx, unsigned long flags);
void X509_STORE_CTX_set_time(X509_STORE_CTX *ctx, unsigned long flags,
                                                        time_t t);
void X509_STORE_CTX_set_verify_cb(X509_STORE_CTX *ctx,
                                  int (*verify_cb)(int, X509_STORE_CTX *));

X509_POLICY_TREE *X509_STORE_CTX_get0_policy_tree(X509_STORE_CTX *ctx);
int X509_STORE_CTX_get_explicit_policy(X509_STORE_CTX *ctx);

X509_VERIFY_PARAM *X509_STORE_CTX_get0_param(X509_STORE_CTX *ctx);
void X509_STORE_CTX_set0_param(X509_STORE_CTX *ctx, X509_VERIFY_PARAM *param);
int X509_STORE_CTX_set_default(X509_STORE_CTX *ctx, const char *name);

/* X509_VERIFY_PARAM functions */

X509_VERIFY_PARAM *X509_VERIFY_PARAM_new(void);
void X509_VERIFY_PARAM_free(X509_VERIFY_PARAM *param);
int X509_VERIFY_PARAM_inherit(X509_VERIFY_PARAM *to,
                                                const X509_VERIFY_PARAM *from);
int X509_VERIFY_PARAM_set1(X509_VERIFY_PARAM *to,
                                                 const X509_VERIFY_PARAM *from);
int X509_VERIFY_PARAM_set1_name(X509_VERIFY_PARAM *param, const char *name);
int X509_VERIFY_PARAM_set_flags(X509_VERIFY_PARAM *param, unsigned long flags);
int X509_VERIFY_PARAM_clear_flags(X509_VERIFY_PARAM *param,
                                                        unsigned long flags);
unsigned long X509_VERIFY_PARAM_get_flags(X509_VERIFY_PARAM *param);
int X509_VERIFY_PARAM_set_purpose(X509_VERIFY_PARAM *param, int purpose);
int X509_VERIFY_PARAM_set_trust(X509_VERIFY_PARAM *param, int trust);
void X509_VERIFY_PARAM_set_depth(X509_VERIFY_PARAM *param, int depth);
```

```
524 void X509_VERIFY_PARAM_set_time(X509_VERIFY_PARAM *param, time_t t);
525 int X509_VERIFY_PARAM_add0_policy(X509_VERIFY_PARAM *param,
526                                   ASN1_OBJECT *policy);
527 int X509_VERIFY_PARAM_set1_policies(X509_VERIFY_PARAM *param,
528                                     STACK_OF(ASN1_OBJECT) *policies);
529 int X509_VERIFY_PARAM_get_depth(const X509_VERIFY_PARAM *param);

531 int X509_VERIFY_PARAM_add0_table(X509_VERIFY_PARAM *param);
532 const X509_VERIFY_PARAM *X509_VERIFY_PARAM_lookup(const char *name);
533 void X509_VERIFY_PARAM_table_cleanup(void);

535 int X509_policy_check(X509_POLICY_TREE **ptree, int *pexplicit_policy,
536                       STACK_OF(X509) *certs,
537                       STACK_OF(ASN1_OBJECT) *policy_oids,
538                       unsigned int flags);

540 void X509_policy_tree_free(X509_POLICY_TREE *tree);

542 int X509_policy_tree_level_count(const X509_POLICY_TREE *tree);
543 X509_POLICY_LEVEL *
544         X509_policy_tree_get0_level(const X509_POLICY_TREE *tree, int i);

546 STACK_OF(X509_POLICY_NODE) *
547         X509_policy_tree_get0_policies(const X509_POLICY_TREE *tree);

549 STACK_OF(X509_POLICY_NODE) *
550         X509_policy_tree_get0_user_policies(const X509_POLICY_TREE *tree);

552 int X509_policy_level_node_count(X509_POLICY_LEVEL *level);

554 X509_POLICY_NODE *X509_policy_level_get0_node(X509_POLICY_LEVEL *level, int i);

556 const ASN1_OBJECT *X509_policy_node_get0_policy(const X509_POLICY_NODE *node);

558 STACK_OF(POLICYQUALINFO) *
559         X509_policy_node_get0_qualifiers(const X509_POLICY_NODE *node);
560 const X509_POLICY_NODE *
561         X509_policy_node_get0_parent(const X509_POLICY_NODE *node);

563 #ifdef  __cplusplus
564 }
565 #endif
566 #endif
567 #endif /* ! codereview */
```

```
**********************************************************
   32707 Wed Aug 13 19:51:51 2014
new/usr/src/lib/openssl/include/openssl/x509v3.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x509v3.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#ifndef HEADER_X509V3_H
#define HEADER_X509V3_H

#include <openssl/bio.h>
```



```
#include <openssl/x509.h>
#include <openssl/conf.h>

#ifdef __cplusplus
extern "C" {
#endif

/* Forward reference */
struct v3_ext_method;
struct v3_ext_ctx;

/* Useful typedefs */

typedef void * (*X509V3_EXT_NEW)(void);
typedef void (*X509V3_EXT_FREE)(void *);
typedef void * (*X509V3_EXT_D2I)(void *, const unsigned char ** , long);
typedef int (*X509V3_EXT_I2D)(void *, unsigned char **);
typedef STACK_OF(CONF_VALUE) *
  (*X509V3_EXT_I2V)(const struct v3_ext_method *method, void *ext,
                    STACK_OF(CONF_VALUE) *extlist);
typedef void * (*X509V3_EXT_V2I)(const struct v3_ext_method *method,
                                 struct v3_ext_ctx *ctx,
                                 STACK_OF(CONF_VALUE) *values);
typedef char * (*X509V3_EXT_I2S)(const struct v3_ext_method *method, void *ext);
typedef void * (*X509V3_EXT_S2I)(const struct v3_ext_method *method,
                                 struct v3_ext_ctx *ctx, const char *str);
typedef int (*X509V3_EXT_I2R)(const struct v3_ext_method *method, void *ext,
                              BIO *out, int indent);
typedef void * (*X509V3_EXT_R2I)(const struct v3_ext_method *method,
                                 struct v3_ext_ctx *ctx, const char *str);

/* V3 extension structure */

struct v3_ext_method {
int ext_nid;
int ext_flags;
/* If this is set the following four fields are ignored */
ASN1_ITEM_EXP *it;
/* Old style ASN1 calls */
X509V3_EXT_NEW ext_new;
X509V3_EXT_FREE ext_free;
X509V3_EXT_D2I d2i;
X509V3_EXT_I2D i2d;

/* The following pair is used for string extensions */
X509V3_EXT_I2S i2s;
X509V3_EXT_S2I s2i;

/* The following pair is used for multi-valued extensions */
X509V3_EXT_I2V i2v;
X509V3_EXT_V2I v2i;

/* The following are used for raw extensions */
X509V3_EXT_I2R i2r;
X509V3_EXT_R2I r2i;

void *usr_data; /* Any extension specific data */
};

typedef struct X509V3_CONF_METHOD_st {
char * (*get_string)(void *db, char *section, char *value);
STACK_OF(CONF_VALUE) * (*get_section)(void *db, char *section);
void (*free_string)(void *db, char * string);
void (*free_section)(void *db, STACK_OF(CONF_VALUE) *section);
} X509V3_CONF_METHOD;
```

```
/* Context specific info */
struct v3_ext_ctx {
#define CTX_TEST 0x1
int flags;
X509 *issuer_cert;
X509 *subject_cert;
X509_REQ *subject_req;
X509_CRL *crl;
X509V3_CONF_METHOD *db_meth;
void *db;
/* Maybe more here */
};

typedef struct v3_ext_method X509V3_EXT_METHOD;

DECLARE_STACK_OF(X509V3_EXT_METHOD)

/* ext_flags values */
#define X509V3_EXT_DYNAMIC      0x1
#define X509V3_EXT_CTX_DEP      0x2
#define X509V3_EXT_MULTILINE    0x4

typedef BIT_STRING_BITNAME ENUMERATED_NAMES;

typedef struct BASIC_CONSTRAINTS_st {
int ca;
ASN1_INTEGER *pathlen;
} BASIC_CONSTRAINTS;


typedef struct PKEY_USAGE_PERIOD_st {
ASN1_GENERALIZEDTIME *notBefore;
ASN1_GENERALIZEDTIME *notAfter;
} PKEY_USAGE_PERIOD;

typedef struct otherName_st {
ASN1_OBJECT *type_id;
ASN1_TYPE *value;
} OTHERNAME;

typedef struct EDIPartyName_st {
        ASN1_STRING *nameAssigner;
        ASN1_STRING *partyName;
} EDIPARTYNAME;

typedef struct GENERAL_NAME_st {

#define GEN_OTHERNAME   0
#define GEN_EMAIL       1
#define GEN_DNS         2
#define GEN_X400        3
#define GEN_DIRNAME     4
#define GEN_EDIPARTY    5
#define GEN_URI         6
#define GEN_IPADD       7
#define GEN_RID         8

int type;
union {
        char *ptr;
        OTHERNAME *otherName; /* otherName */
        ASN1_IA5STRING *rfc822Name;
        ASN1_IA5STRING *dNSName;
        ASN1_TYPE *x400Address;
        X509_NAME *directoryName;
        EDIPARTYNAME *ediPartyName;
```



```
        ASN1_IA5STRING *uniformResourceIdentifier;
        ASN1_OCTET_STRING *iPAddress;
        ASN1_OBJECT *registeredID;

        /* Old names */
        ASN1_OCTET_STRING *ip; /* iPAddress */
        X509_NAME *dirn;                /* dirn */
        ASN1_IA5STRING *ia5;/* rfc822Name, dNSName, uniformResourceIdentifier */
        ASN1_OBJECT *rid; /* registeredID */
        ASN1_TYPE *other; /* x400Address */
} d;
} GENERAL_NAME;

typedef STACK_OF(GENERAL_NAME) GENERAL_NAMES;

typedef struct ACCESS_DESCRIPTION_st {
        ASN1_OBJECT *method;
        GENERAL_NAME *location;
} ACCESS_DESCRIPTION;

typedef STACK_OF(ACCESS_DESCRIPTION) AUTHORITY_INFO_ACCESS;

typedef STACK_OF(ASN1_OBJECT) EXTENDED_KEY_USAGE;

DECLARE_STACK_OF(GENERAL_NAME)
DECLARE_ASN1_SET_OF(GENERAL_NAME)

DECLARE_STACK_OF(ACCESS_DESCRIPTION)
DECLARE_ASN1_SET_OF(ACCESS_DESCRIPTION)

typedef struct DIST_POINT_NAME_st {
int type;
union {
        GENERAL_NAMES *fullname;
        STACK_OF(X509_NAME_ENTRY) *relativename;
} name;
/* If relativename then this contains the full distribution point name */
X509_NAME *dpname;
} DIST_POINT_NAME;
/* All existing reasons */
#define CRLDP_ALL_REASONS       0x807f

#define CRL_REASON_NONE                         -1
#define CRL_REASON_UNSPECIFIED                  0
#define CRL_REASON_KEY_COMPROMISE               1
#define CRL_REASON_CA_COMPROMISE                2
#define CRL_REASON_AFFILIATION_CHANGED          3
#define CRL_REASON_SUPERSEDED                   4
#define CRL_REASON_CESSATION_OF_OPERATION       5
#define CRL_REASON_CERTIFICATE_HOLD             6
#define CRL_REASON_REMOVE_FROM_CRL              8
#define CRL_REASON_PRIVILEGE_WITHDRAWN          9
#define CRL_REASON_AA_COMPROMISE                10

struct DIST_POINT_st {
DIST_POINT_NAME *distpoint;
ASN1_BIT_STRING *reasons;
GENERAL_NAMES *CRLissuer;
int dp_reasons;
};

typedef STACK_OF(DIST_POINT) CRL_DIST_POINTS;

DECLARE_STACK_OF(DIST_POINT)
DECLARE_ASN1_SET_OF(DIST_POINT)
```

```
260 struct AUTHORITY_KEYID_st {
261 ASN1_OCTET_STRING *keyid;
262 GENERAL_NAMES *issuer;
263 ASN1_INTEGER *serial;
264 };

266 /* Strong extranet structures */

268 typedef struct SXNET_ID_st {
269         ASN1_INTEGER *zone;
270         ASN1_OCTET_STRING *user;
271 } SXNETID;

273 DECLARE_STACK_OF(SXNETID)
274 DECLARE_ASN1_SET_OF(SXNETID)

276 typedef struct SXNET_st {
277         ASN1_INTEGER *version;
278         STACK_OF(SXNETID) *ids;
279 } SXNET;

281 typedef struct NOTICEREF_st {
282         ASN1_STRING *organization;
283         STACK_OF(ASN1_INTEGER) *noticenos;
284 } NOTICEREF;

286 typedef struct USERNOTICE_st {
287         NOTICEREF *noticeref;
288         ASN1_STRING *exptext;
289 } USERNOTICE;

291 typedef struct POLICYQUALINFO_st {
292         ASN1_OBJECT *pqualid;
293         union {
294                 ASN1_IA5STRING *cpsuri;
295                 USERNOTICE *usernotice;
296                 ASN1_TYPE *other;
297         } d;
298 } POLICYQUALINFO;

300 DECLARE_STACK_OF(POLICYQUALINFO)
301 DECLARE_ASN1_SET_OF(POLICYQUALINFO)

303 typedef struct POLICYINFO_st {
304         ASN1_OBJECT *policyid;
305         STACK_OF(POLICYQUALINFO) *qualifiers;
306 } POLICYINFO;

308 typedef STACK_OF(POLICYINFO) CERTIFICATEPOLICIES;

310 DECLARE_STACK_OF(POLICYINFO)
311 DECLARE_ASN1_SET_OF(POLICYINFO)

313 typedef struct POLICY_MAPPING_st {
314         ASN1_OBJECT *issuerDomainPolicy;
315         ASN1_OBJECT *subjectDomainPolicy;
316 } POLICY_MAPPING;

318 DECLARE_STACK_OF(POLICY_MAPPING)

320 typedef STACK_OF(POLICY_MAPPING) POLICY_MAPPINGS;

322 typedef struct GENERAL_SUBTREE_st {
323         GENERAL_NAME *base;
324         ASN1_INTEGER *minimum;
325         ASN1_INTEGER *maximum;
```



```
326 } GENERAL_SUBTREE;

328 DECLARE_STACK_OF(GENERAL_SUBTREE)

330 struct NAME_CONSTRAINTS_st {
331         STACK_OF(GENERAL_SUBTREE) *permittedSubtrees;
332         STACK_OF(GENERAL_SUBTREE) *excludedSubtrees;
333 };

335 typedef struct POLICY_CONSTRAINTS_st {
336         ASN1_INTEGER *requireExplicitPolicy;
337         ASN1_INTEGER *inhibitPolicyMapping;
338 } POLICY_CONSTRAINTS;

340 /* Proxy certificate structures, see RFC 3820 */
341 typedef struct PROXY_POLICY_st
342         {
343         ASN1_OBJECT *policyLanguage;
344         ASN1_OCTET_STRING *policy;
345         } PROXY_POLICY;

347 typedef struct PROXY_CERT_INFO_EXTENSION_st
348         {
349         ASN1_INTEGER *pcPathLengthConstraint;
350         PROXY_POLICY *proxyPolicy;
351         } PROXY_CERT_INFO_EXTENSION;

353 DECLARE_ASN1_FUNCTIONS(PROXY_POLICY)
354 DECLARE_ASN1_FUNCTIONS(PROXY_CERT_INFO_EXTENSION)

356 struct ISSUING_DIST_POINT_st
357         {
358         DIST_POINT_NAME *distpoint;
359         int onlyuser;
360         int onlyCA;
361         ASN1_BIT_STRING *onlysomereasons;
362         int indirectCRL;
363         int onlyattr;
364         };

366 /* Values in idp_flags field */
367 /* IDP present */
368 #define IDP_PRESENT     0x1
369 /* IDP values inconsistent */
370 #define IDP_INVALID     0x2
371 /* onlyuser true */
372 #define IDP_ONLYUSER    0x4
373 /* onlyCA true */
374 #define IDP_ONLYCA      0x8
375 /* onlyattr true */
376 #define IDP_ONLYATTR    0x10
377 /* indirectCRL true */
378 #define IDP_INDIRECT    0x20
379 /* onlysomereasons present */
380 #define IDP_REASONS     0x40

382 #define X509V3_conf_err(val) ERR_add_error_data(6, "section:", val->section, \
383 ",name:", val->name, ",value:", val->value);

385 #define X509V3_set_ctx_test(ctx) \
386                         X509V3_set_ctx(ctx, NULL, NULL, NULL, NULL, CTX_TEST)
387 #define X509V3_set_ctx_nodb(ctx) (ctx)->db = NULL;

389 #define EXT_BITSTRING(nid, table) { nid, 0, ASN1_ITEM_ref(ASN1_BIT_STRING), \
390                         0,0,0,0, \
391                         0,0, \
```

```
                        (X509V3_EXT_I2V)i2v_ASN1_BIT_STRING, \
                        (X509V3_EXT_V2I)v2i_ASN1_BIT_STRING, \
                        NULL, NULL, \
                        table}

#define EXT_IA5STRING(nid) { nid, 0, ASN1_ITEM_ref(ASN1_IA5STRING), \
                        0,0,0,0, \
                        (X509V3_EXT_I2S)i2s_ASN1_IA5STRING, \
                        (X509V3_EXT_S2I)s2i_ASN1_IA5STRING, \
                        0,0,0,0, \
                        NULL}

#define EXT_END { -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}


/* X509_PURPOSE stuff */

#define EXFLAG_BCONS            0x1
#define EXFLAG_KUSAGE           0x2
#define EXFLAG_XKUSAGE          0x4
#define EXFLAG_NSCERT           0x8

#define EXFLAG_CA               0x10
/* Really self issued not necessarily self signed */
#define EXFLAG_SI               0x20
#define EXFLAG_SS               0x20
#define EXFLAG_V1               0x40
#define EXFLAG_INVALID          0x80
#define EXFLAG_SET              0x100
#define EXFLAG_CRITICAL         0x200
#define EXFLAG_PROXY            0x400

#define EXFLAG_INVALID_POLICY   0x800
#define EXFLAG_FRESHEST         0x1000

#define KU_DIGITAL_SIGNATURE    0x0080
#define KU_NON_REPUDIATION      0x0040
#define KU_KEY_ENCIPHERMENT     0x0020
#define KU_DATA_ENCIPHERMENT    0x0010
#define KU_KEY_AGREEMENT        0x0008
#define KU_KEY_CERT_SIGN        0x0004
#define KU_CRL_SIGN             0x0002
#define KU_ENCIPHER_ONLY        0x0001
#define KU_DECIPHER_ONLY        0x8000

#define NS_SSL_CLIENT           0x80
#define NS_SSL_SERVER           0x40
#define NS_SMIME                0x20
#define NS_OBJSIGN              0x10
#define NS_SSL_CA               0x04
#define NS_SMIME_CA             0x02
#define NS_OBJSIGN_CA           0x01
#define NS_ANY_CA               (NS_SSL_CA|NS_SMIME_CA|NS_OBJSIGN_CA)

#define XKU_SSL_SERVER          0x1
#define XKU_SSL_CLIENT          0x2
#define XKU_SMIME               0x4
#define XKU_CODE_SIGN           0x8
#define XKU_SGC                 0x10
#define XKU_OCSP_SIGN           0x20
#define XKU_TIMESTAMP           0x40
#define XKU_DVCS                0x80

#define X509_PURPOSE_DYNAMIC    0x1
#define X509_PURPOSE_DYNAMIC_NAME       0x2
```



```
typedef struct x509_purpose_st {
        int purpose;
        int trust;              /* Default trust ID */
        int flags;
        int (*check_purpose)(const struct x509_purpose_st *,
                             const X509 *, int);
        char *name;
        char *sname;
        void *usr_data;
} X509_PURPOSE;

#define X509_PURPOSE_SSL_CLIENT         1
#define X509_PURPOSE_SSL_SERVER         2
#define X509_PURPOSE_NS_SSL_SERVER      3
#define X509_PURPOSE_SMIME_SIGN         4
#define X509_PURPOSE_SMIME_ENCRYPT      5
#define X509_PURPOSE_CRL_SIGN           6
#define X509_PURPOSE_ANY                7
#define X509_PURPOSE_OCSP_HELPER        8
#define X509_PURPOSE_TIMESTAMP_SIGN     9

#define X509_PURPOSE_MIN                1
#define X509_PURPOSE_MAX                9

/* Flags for X509V3_EXT_print() */

#define X509V3_EXT_UNKNOWN_MASK         (0xfL << 16)
/* Return error for unknown extensions */
#define X509V3_EXT_DEFAULT              0
/* Print error for unknown extensions */
#define X509V3_EXT_ERROR_UNKNOWN        (1L << 16)
/* ASN1 parse unknown extensions */
#define X509V3_EXT_PARSE_UNKNOWN        (2L << 16)
/* BIO_dump unknown extensions */
#define X509V3_EXT_DUMP_UNKNOWN         (3L << 16)

/* Flags for X509V3_add1_i2d */

#define X509V3_ADD_OP_MASK              0xfL
#define X509V3_ADD_DEFAULT              0L
#define X509V3_ADD_APPEND               1L
#define X509V3_ADD_REPLACE              2L
#define X509V3_ADD_REPLACE_EXISTING     3L
#define X509V3_ADD_KEEP_EXISTING        4L
#define X509V3_ADD_DELETE               5L
#define X509V3_ADD_SILENT               0x10

DECLARE_STACK_OF(X509_PURPOSE)

DECLARE_ASN1_FUNCTIONS(BASIC_CONSTRAINTS)

DECLARE_ASN1_FUNCTIONS(SXNET)
DECLARE_ASN1_FUNCTIONS(SXNETID)

int SXNET_add_id_asc(SXNET **psx, char *zone, char *user, int userlen);
int SXNET_add_id_ulong(SXNET **psx, unsigned long lzone, char *user, int userlen
int SXNET_add_id_INTEGER(SXNET **psx, ASN1_INTEGER *izone, char *user, int userl

ASN1_OCTET_STRING *SXNET_get_id_asc(SXNET *sx, char *zone);
ASN1_OCTET_STRING *SXNET_get_id_ulong(SXNET *sx, unsigned long lzone);
ASN1_OCTET_STRING *SXNET_get_id_INTEGER(SXNET *sx, ASN1_INTEGER *zone);

DECLARE_ASN1_FUNCTIONS(AUTHORITY_KEYID)

DECLARE_ASN1_FUNCTIONS(PKEY_USAGE_PERIOD)
```

```
524 DECLARE_ASN1_FUNCTIONS(GENERAL_NAME)
525 GENERAL_NAME *GENERAL_NAME_dup(GENERAL_NAME *a);
526 int GENERAL_NAME_cmp(GENERAL_NAME *a, GENERAL_NAME *b);



530 ASN1_BIT_STRING *v2i_ASN1_BIT_STRING(X509V3_EXT_METHOD *method,
531                                 X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval);
532 STACK_OF(CONF_VALUE) *i2v_ASN1_BIT_STRING(X509V3_EXT_METHOD *method,
533                                 ASN1_BIT_STRING *bits,
534                                 STACK_OF(CONF_VALUE) *extlist);

536 STACK_OF(CONF_VALUE) *i2v_GENERAL_NAME(X509V3_EXT_METHOD *method, GENERAL_NAME *
537 int GENERAL_NAME_print(BIO *out, GENERAL_NAME *gen);

539 DECLARE_ASN1_FUNCTIONS(GENERAL_NAMES)

541 STACK_OF(CONF_VALUE) *i2v_GENERAL_NAMES(X509V3_EXT_METHOD *method,
542                 GENERAL_NAMES *gen, STACK_OF(CONF_VALUE) *extlist);
543 GENERAL_NAMES *v2i_GENERAL_NAMES(const X509V3_EXT_METHOD *method,
544                                  X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval);

546 DECLARE_ASN1_FUNCTIONS(OTHERNAME)
547 DECLARE_ASN1_FUNCTIONS(EDIPARTYNAME)
548 int OTHERNAME_cmp(OTHERNAME *a, OTHERNAME *b);
549 void GENERAL_NAME_set0_value(GENERAL_NAME *a, int type, void *value);
550 void *GENERAL_NAME_get0_value(GENERAL_NAME *a, int *ptype);
551 int GENERAL_NAME_set0_othername(GENERAL_NAME *gen,
552                                 ASN1_OBJECT *oid, ASN1_TYPE *value);
553 int GENERAL_NAME_get0_otherName(GENERAL_NAME *gen,
554                                 ASN1_OBJECT **poid, ASN1_TYPE **pvalue);

556 char *i2s_ASN1_OCTET_STRING(X509V3_EXT_METHOD *method, ASN1_OCTET_STRING *ia5);
557 ASN1_OCTET_STRING *s2i_ASN1_OCTET_STRING(X509V3_EXT_METHOD *method, X509V3_CTX *

559 DECLARE_ASN1_FUNCTIONS(EXTENDED_KEY_USAGE)
560 int i2a_ACCESS_DESCRIPTION(BIO *bp, ACCESS_DESCRIPTION* a);

562 DECLARE_ASN1_FUNCTIONS(CERTIFICATEPOLICIES)
563 DECLARE_ASN1_FUNCTIONS(POLICYINFO)
564 DECLARE_ASN1_FUNCTIONS(POLICYQUALINFO)
565 DECLARE_ASN1_FUNCTIONS(USERNOTICE)
566 DECLARE_ASN1_FUNCTIONS(NOTICEREF)

568 DECLARE_ASN1_FUNCTIONS(CRL_DIST_POINTS)
569 DECLARE_ASN1_FUNCTIONS(DIST_POINT)
570 DECLARE_ASN1_FUNCTIONS(DIST_POINT_NAME)
571 DECLARE_ASN1_FUNCTIONS(ISSUING_DIST_POINT)

573 int DIST_POINT_set_dpname(DIST_POINT_NAME *dpn, X509_NAME *iname);

575 int NAME_CONSTRAINTS_check(X509 *x, NAME_CONSTRAINTS *nc);

577 DECLARE_ASN1_FUNCTIONS(ACCESS_DESCRIPTION)
578 DECLARE_ASN1_FUNCTIONS(AUTHORITY_INFO_ACCESS)

580 DECLARE_ASN1_ITEM(POLICY_MAPPING)
581 DECLARE_ASN1_ALLOC_FUNCTIONS(POLICY_MAPPING)
582 DECLARE_ASN1_ITEM(POLICY_MAPPINGS)

584 DECLARE_ASN1_ITEM(GENERAL_SUBTREE)
585 DECLARE_ASN1_ALLOC_FUNCTIONS(GENERAL_SUBTREE)

587 DECLARE_ASN1_ITEM(NAME_CONSTRAINTS)
588 DECLARE_ASN1_ALLOC_FUNCTIONS(NAME_CONSTRAINTS)
```



```
590 DECLARE_ASN1_ALLOC_FUNCTIONS(POLICY_CONSTRAINTS)
591 DECLARE_ASN1_ITEM(POLICY_CONSTRAINTS)

593 GENERAL_NAME *a2i_GENERAL_NAME(GENERAL_NAME *out,
594                                 const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
595                                 int gen_type, char *value, int is_nc);

597 #ifdef HEADER_CONF_H
598 GENERAL_NAME *v2i_GENERAL_NAME(const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
599                                 CONF_VALUE *cnf);
600 GENERAL_NAME *v2i_GENERAL_NAME_ex(GENERAL_NAME *out,
601                                   const X509V3_EXT_METHOD *method,
602                                   X509V3_CTX *ctx, CONF_VALUE *cnf, int is_nc);
603 void X509V3_conf_free(CONF_VALUE *val);

605 X509_EXTENSION *X509V3_EXT_nconf_nid(CONF *conf, X509V3_CTX *ctx, int ext_nid, c
606 X509_EXTENSION *X509V3_EXT_nconf(CONF *conf, X509V3_CTX *ctx, char *name, char *
607 int X509V3_EXT_add_nconf_sk(CONF *conf, X509V3_CTX *ctx, char *section, STACK_OF
608 int X509V3_EXT_add_nconf(CONF *conf, X509V3_CTX *ctx, char *section, X509 *cert)
609 int X509V3_EXT_REQ_add_nconf(CONF *conf, X509V3_CTX *ctx, char *section, X509_RE
610 int X509V3_EXT_CRL_add_nconf(CONF *conf, X509V3_CTX *ctx, char *section, X509_CR

612 X509_EXTENSION *X509V3_EXT_conf_nid(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
613                                     int ext_nid, char *value);
614 X509_EXTENSION *X509V3_EXT_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
615                                  char *name, char *value);
616 int X509V3_EXT_add_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
617                         char *section, X509 *cert);
618 int X509V3_EXT_REQ_add_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
619                             char *section, X509_REQ *req);
620 int X509V3_EXT_CRL_add_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
621                             char *section, X509_CRL *crl);

623 int X509V3_add_value_bool_nf(char *name, int asn1_bool,
624                              STACK_OF(CONF_VALUE) **extlist);
625 int X509V3_get_value_bool(CONF_VALUE *value, int *asn1_bool);
626 int X509V3_get_value_int(CONF_VALUE *value, ASN1_INTEGER **aint);
627 void X509V3_set_nconf(X509V3_CTX *ctx, CONF *conf);
628 void X509V3_set_conf_lhash(X509V3_CTX *ctx, LHASH_OF(CONF_VALUE) *lhash);
629 #endif

631 char * X509V3_get_string(X509V3_CTX *ctx, char *name, char *section);
632 STACK_OF(CONF_VALUE) * X509V3_get_section(X509V3_CTX *ctx, char *section);
633 void X509V3_string_free(X509V3_CTX *ctx, char *str);
634 void X509V3_section_free( X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *section);
635 void X509V3_set_ctx(X509V3_CTX *ctx, X509 *issuer, X509 *subject,
636                                   X509_REQ *req, X509_CRL *crl, int flags);

638 int X509V3_add_value(const char *name, const char *value,
639                                                 STACK_OF(CONF_VALUE) **extlist);
640 int X509V3_add_value_uchar(const char *name, const unsigned char *value,
641                                                 STACK_OF(CONF_VALUE) **extlist);
642 int X509V3_add_value_bool(const char *name, int asn1_bool,
643                                                 STACK_OF(CONF_VALUE) **extlist);
644 int X509V3_add_value_int(const char *name, ASN1_INTEGER *aint,
645                                                 STACK_OF(CONF_VALUE) **extlist);
646 char * i2s_ASN1_INTEGER(X509V3_EXT_METHOD *meth, ASN1_INTEGER *aint);
647 ASN1_INTEGER * s2i_ASN1_INTEGER(X509V3_EXT_METHOD *meth, char *value);
648 char * i2s_ASN1_ENUMERATED(X509V3_EXT_METHOD *meth, ASN1_ENUMERATED *aint);
649 char * i2s_ASN1_ENUMERATED_TABLE(X509V3_EXT_METHOD *meth, ASN1_ENUMERATED *aint)
650 int X509V3_EXT_add(X509V3_EXT_METHOD *ext);
651 int X509V3_EXT_add_list(X509V3_EXT_METHOD *extlist);
652 int X509V3_EXT_add_alias(int nid_to, int nid_from);
653 void X509V3_EXT_cleanup(void);

655 const X509V3_EXT_METHOD *X509V3_EXT_get(X509_EXTENSION *ext);
```

```
656 const X509V3_EXT_METHOD *X509V3_EXT_get_nid(int nid);
657 int X509V3_add_standard_extensions(void);
658 STACK_OF(CONF_VALUE) *X509V3_parse_list(const char *line);
659 void *X509V3_EXT_d2i(X509_EXTENSION *ext);
660 void *X509V3_get_d2i(STACK_OF(X509_EXTENSION) *x, int nid, int *crit, int *idx);


663 X509_EXTENSION *X509V3_EXT_i2d(int ext_nid, int crit, void *ext_struc);
664 int X509V3_add1_i2d(STACK_OF(X509_EXTENSION) **x, int nid, void *value, int crit

666 char *hex_to_string(const unsigned char *buffer, long len);
667 unsigned char *string_to_hex(const char *str, long *len);
668 int name_cmp(const char *name, const char *cmp);

670 void X509V3_EXT_val_prn(BIO *out, STACK_OF(CONF_VALUE) *val, int indent,
671                                                      int ml);
672 int X509V3_EXT_print(BIO *out, X509_EXTENSION *ext, unsigned long flag, int inde
673 int X509V3_EXT_print_fp(FILE *out, X509_EXTENSION *ext, int flag, int indent);

675 int X509V3_extensions_print(BIO *out, char *title, STACK_OF(X509_EXTENSION) *ext

677 int X509_check_ca(X509 *x);
678 int X509_check_purpose(X509 *x, int id, int ca);
679 int X509_supported_extension(X509_EXTENSION *ex);
680 int X509_PURPOSE_set(int *p, int purpose);
681 int X509_check_issued(X509 *issuer, X509 *subject);
682 int X509_check_akid(X509 *issuer, AUTHORITY_KEYID *akid);
683 int X509_PURPOSE_get_count(void);
684 X509_PURPOSE * X509_PURPOSE_get0(int idx);
685 int X509_PURPOSE_get_by_sname(char *sname);
686 int X509_PURPOSE_get_by_id(int id);
687 int X509_PURPOSE_add(int id, int trust, int flags,
688                      int (*ck)(const X509_PURPOSE *, const X509 *, int),
689                                char *name, char *sname, void *arg);
690 char *X509_PURPOSE_get0_name(X509_PURPOSE *xp);
691 char *X509_PURPOSE_get0_sname(X509_PURPOSE *xp);
692 int X509_PURPOSE_get_trust(X509_PURPOSE *xp);
693 void X509_PURPOSE_cleanup(void);
694 int X509_PURPOSE_get_id(X509_PURPOSE *);

696 STACK_OF(OPENSSL_STRING) *X509_get1_email(X509 *x);
697 STACK_OF(OPENSSL_STRING) *X509_REQ_get1_email(X509_REQ *x);
698 void X509_email_free(STACK_OF(OPENSSL_STRING) *sk);
699 STACK_OF(OPENSSL_STRING) *X509_get1_ocsp(X509 *x);

701 ASN1_OCTET_STRING *a2i_IPADDRESS(const char *ipasc);
702 ASN1_OCTET_STRING *a2i_IPADDRESS_NC(const char *ipasc);
703 int a2i_ipadd(unsigned char *ipout, const char *ipasc);
704 int X509V3_NAME_from_section(X509_NAME *nm, STACK_OF(CONF_VALUE)*dn_sk,
705                                             unsigned long chtype);

707 void X509_POLICY_NODE_print(BIO *out, X509_POLICY_NODE *node, int indent);
708 DECLARE_STACK_OF(X509_POLICY_NODE)

710 #ifndef OPENSSL_NO_RFC3779

712 typedef struct ASRange_st {
713   ASN1_INTEGER *min, *max;
714 } ASRange;

716 #define ASIdOrRange_id          0
717 #define ASIdOrRange_range       1

719 typedef struct ASIdOrRange_st {
720   int type;
721   union {
```



```
722     ASN1_INTEGER *id;
723     ASRange      *range;
724   } u;
725 } ASIdOrRange;

727 typedef STACK_OF(ASIdOrRange) ASIdOrRanges;
728 DECLARE_STACK_OF(ASIdOrRange)

730 #define ASIdentifierChoice_inherit              0
731 #define ASIdentifierChoice_asIdsOrRanges        1

733 typedef struct ASIdentifierChoice_st {
734   int type;
735   union {
736     ASN1_NULL    *inherit;
737     ASIdOrRanges *asIdsOrRanges;
738   } u;
739 } ASIdentifierChoice;

741 typedef struct ASIdentifiers_st {
742   ASIdentifierChoice *asnum, *rdi;
743 } ASIdentifiers;

745 DECLARE_ASN1_FUNCTIONS(ASRange)
746 DECLARE_ASN1_FUNCTIONS(ASIdOrRange)
747 DECLARE_ASN1_FUNCTIONS(ASIdentifierChoice)
748 DECLARE_ASN1_FUNCTIONS(ASIdentifiers)


751 typedef struct IPAddressRange_st {
752   ASN1_BIT_STRING       *min, *max;
753 } IPAddressRange;

755 #define IPAddressOrRange_addressPrefix  0
756 #define IPAddressOrRange_addressRange   1

758 typedef struct IPAddressOrRange_st {
759   int type;
760   union {
761     ASN1_BIT_STRING     *addressPrefix;
762     IPAddressRange      *addressRange;
763   } u;
764 } IPAddressOrRange;

766 typedef STACK_OF(IPAddressOrRange) IPAddressOrRanges;
767 DECLARE_STACK_OF(IPAddressOrRange)

769 #define IPAddressChoice_inherit                 0
770 #define IPAddressChoice_addressesOrRanges       1

772 typedef struct IPAddressChoice_st {
773   int type;
774   union {
775     ASN1_NULL           *inherit;
776     IPAddressOrRanges   *addressesOrRanges;
777   } u;
778 } IPAddressChoice;

780 typedef struct IPAddressFamily_st {
781   ASN1_OCTET_STRING     *addressFamily;
782   IPAddressChoice       *ipAddressChoice;
783 } IPAddressFamily;

785 typedef STACK_OF(IPAddressFamily) IPAddrBlocks;
786 DECLARE_STACK_OF(IPAddressFamily)
```

```
788 DECLARE_ASN1_FUNCTIONS(IPAddressRange)
789 DECLARE_ASN1_FUNCTIONS(IPAddressOrRange)
790 DECLARE_ASN1_FUNCTIONS(IPAddressChoice)
791 DECLARE_ASN1_FUNCTIONS(IPAddressFamily)

793 /*
794  * API tag for elements of the ASIdentifer SEQUENCE.
795  */
796 #define V3_ASID_ASNUM   0
797 #define V3_ASID_RDI     1

799 /*
800  * AFI values, assigned by IANA.  It'd be nice to make the AFI
801  * handling code totally generic, but there are too many little things
802  * that would need to be defined for other address families for it to
803  * be worth the trouble.
804  */
805 #define IANA_AFI_IPV4   1
806 #define IANA_AFI_IPV6   2

808 /*
809  * Utilities to construct and extract values from RFC3779 extensions,
810  * since some of the encodings (particularly for IP address prefixes
811  * and ranges) are a bit tedious to work with directly.
812  */
813 int v3_asid_add_inherit(ASIdentifiers *asid, int which);
814 int v3_asid_add_id_or_range(ASIdentifiers *asid, int which,
815                             ASN1_INTEGER *min, ASN1_INTEGER *max);
816 int v3_addr_add_inherit(IPAddrBlocks *addr,
817                         const unsigned afi, const unsigned *safi);
818 int v3_addr_add_prefix(IPAddrBlocks *addr,
819                        const unsigned afi, const unsigned *safi,
820                        unsigned char *a, const int prefixlen);
821 int v3_addr_add_range(IPAddrBlocks *addr,
822                       const unsigned afi, const unsigned *safi,
823                       unsigned char *min, unsigned char *max);
824 unsigned v3_addr_get_afi(const IPAddressFamily *f);
825 int v3_addr_get_range(IPAddressOrRange *aor, const unsigned afi,
826                       unsigned char *min, unsigned char *max,
827                       const int length);

829 /*
830  * Canonical forms.
831  */
832 int v3_asid_is_canonical(ASIdentifiers *asid);
833 int v3_addr_is_canonical(IPAddrBlocks *addr);
834 int v3_asid_canonize(ASIdentifiers *asid);
835 int v3_addr_canonize(IPAddrBlocks *addr);

837 /*
838  * Tests for inheritance and containment.
839  */
840 int v3_asid_inherits(ASIdentifiers *asid);
841 int v3_addr_inherits(IPAddrBlocks *addr);
842 int v3_asid_subset(ASIdentifiers *a, ASIdentifiers *b);
843 int v3_addr_subset(IPAddrBlocks *a, IPAddrBlocks *b);

845 /*
846  * Check whether RFC 3779 extensions nest properly in chains.
847  */
848 int v3_asid_validate_path(X509_STORE_CTX *);
849 int v3_addr_validate_path(X509_STORE_CTX *);
850 int v3_asid_validate_resource_set(STACK_OF(X509) *chain,
851                                   ASIdentifiers *ext,
852                                   int allow_inheritance);
853 int v3_addr_validate_resource_set(STACK_OF(X509) *chain,
```



```
854                                   IPAddrBlocks *ext,
855                                   int allow_inheritance);

857 #endif /* OPENSSL_NO_RFC3779 */

859 /* BEGIN ERROR CODES */
860 /* The following lines are auto generated by the script mkerr.pl. Any changes
861  * made after this point may be overwritten when the script is next run.
862  */
863 void ERR_load_X509V3_strings(void);

865 /* Error codes for the X509V3 functions. */

867 /* Function codes. */
868 #define X509V3_F_A2I_GENERAL_NAME                       164
869 #define X509V3_F_ASIDENTIFIERCHOICE_CANONIZE            161
870 #define X509V3_F_ASIDENTIFIERCHOICE_IS_CANONICAL        162
871 #define X509V3_F_COPY_EMAIL                             122
872 #define X509V3_F_COPY_ISSUER                            123
873 #define X509V3_F_DO_DIRNAME                             144
874 #define X509V3_F_DO_EXT_CONF                            124
875 #define X509V3_F_DO_EXT_I2D                             135
876 #define X509V3_F_DO_EXT_NCONF                           151
877 #define X509V3_F_DO_I2V_NAME_CONSTRAINTS                148
878 #define X509V3_F_GNAMES_FROM_SECTNAME                   156
879 #define X509V3_F_HEX_TO_STRING                          111
880 #define X509V3_F_I2S_ASN1_ENUMERATED                    121
881 #define X509V3_F_I2S_ASN1_IA5STRING                     149
882 #define X509V3_F_I2S_ASN1_INTEGER                       120
883 #define X509V3_F_I2V_AUTHORITY_INFO_ACCESS              138
884 #define X509V3_F_NOTICE_SECTION                         132
885 #define X509V3_F_NREF_NOS                               133
886 #define X509V3_F_POLICY_SECTION                         131
887 #define X509V3_F_PROCESS_PCI_VALUE                      150
888 #define X509V3_F_R2I_CERTPOL                            130
889 #define X509V3_F_R2I_PCI                                155
890 #define X509V3_F_S2I_ASN1_IA5STRING                     100
891 #define X509V3_F_S2I_ASN1_INTEGER                       108
892 #define X509V3_F_S2I_ASN1_OCTET_STRING                  112
893 #define X509V3_F_S2I_ASN1_SKEY_ID                       114
894 #define X509V3_F_S2I_SKEY_ID                            115
895 #define X509V3_F_SET_DIST_POINT_NAME                    158
896 #define X509V3_F_STRING_TO_HEX                          113
897 #define X509V3_F_SXNET_ADD_ID_ASC                       125
898 #define X509V3_F_SXNET_ADD_ID_INTEGER                   126
899 #define X509V3_F_SXNET_ADD_ID_ULONG                     127
900 #define X509V3_F_SXNET_GET_ID_ASC                       128
901 #define X509V3_F_SXNET_GET_ID_ULONG                     129
902 #define X509V3_F_V2I_ASIDENTIFIERS                      163
903 #define X509V3_F_V2I_ASN1_BIT_STRING                    101
904 #define X509V3_F_V2I_AUTHORITY_INFO_ACCESS              139
905 #define X509V3_F_V2I_AUTHORITY_KEYID                    119
906 #define X509V3_F_V2I_BASIC_CONSTRAINTS                  102
907 #define X509V3_F_V2I_CRLD                               134
908 #define X509V3_F_V2I_EXTENDED_KEY_USAGE                 103
909 #define X509V3_F_V2I_GENERAL_NAMES                      118
910 #define X509V3_F_V2I_GENERAL_NAME_EX                    117
911 #define X509V3_F_V2I_IDP                                157
912 #define X509V3_F_V2I_IPADDRBLOCKS                       159
913 #define X509V3_F_V2I_ISSUER_ALT                         153
914 #define X509V3_F_V2I_NAME_CONSTRAINTS                   147
915 #define X509V3_F_V2I_POLICY_CONSTRAINTS                 146
916 #define X509V3_F_V2I_POLICY_MAPPINGS                    145
917 #define X509V3_F_V2I_SUBJECT_ALT                        154
918 #define X509V3_F_V3_ADDR_VALIDATE_PATH_INTERNAL         160
919 #define X509V3_F_V3_GENERIC_EXTENSION                   116
```

```
#define X509V3_F_X509V3_ADD1_I2D                        140
#define X509V3_F_X509V3_ADD_VALUE                       105
#define X509V3_F_X509V3_EXT_ADD                         104
#define X509V3_F_X509V3_EXT_ADD_ALIAS                   106
#define X509V3_F_X509V3_EXT_CONF                        107
#define X509V3_F_X509V3_EXT_I2D                         136
#define X509V3_F_X509V3_EXT_NCONF                       152
#define X509V3_F_X509V3_GET_SECTION                     142
#define X509V3_F_X509V3_GET_STRING                      143
#define X509V3_F_X509V3_GET_VALUE_BOOL                  110
#define X509V3_F_X509V3_PARSE_LIST                      109
#define X509V3_F_X509_PURPOSE_ADD                       137
#define X509V3_F_X509_PURPOSE_SET                       141

/* Reason codes. */
#define X509V3_R_BAD_IP_ADDRESS                         118
#define X509V3_R_BAD_OBJECT                             119
#define X509V3_R_BN_DEC2BN_ERROR                        100
#define X509V3_R_BN_TO_ASN1_INTEGER_ERROR               101
#define X509V3_R_DIRNAME_ERROR                          149
#define X509V3_R_DISTPOINT_ALREADY_SET                  160
#define X509V3_R_DUPLICATE_ZONE_ID                      133
#define X509V3_R_ERROR_CONVERTING_ZONE                  131
#define X509V3_R_ERROR_CREATING_EXTENSION               144
#define X509V3_R_ERROR_IN_EXTENSION                     128
#define X509V3_R_EXPECTED_A_SECTION_NAME                137
#define X509V3_R_EXTENSION_EXISTS                       145
#define X509V3_R_EXTENSION_NAME_ERROR                   115
#define X509V3_R_EXTENSION_NOT_FOUND                    102
#define X509V3_R_EXTENSION_SETTING_NOT_SUPPORTED        103
#define X509V3_R_EXTENSION_VALUE_ERROR                  116
#define X509V3_R_ILLEGAL_EMPTY_EXTENSION                151
#define X509V3_R_ILLEGAL_HEX_DIGIT                      113
#define X509V3_R_INCORRECT_POLICY_SYNTAX_TAG            152
#define X509V3_R_INVALID_MULTIPLE_RDNS                  161
#define X509V3_R_INVALID_ASNUMBER                       162
#define X509V3_R_INVALID_ASRANGE                        163
#define X509V3_R_INVALID_BOOLEAN_STRING                 104
#define X509V3_R_INVALID_EXTENSION_STRING               105
#define X509V3_R_INVALID_INHERITANCE                    165
#define X509V3_R_INVALID_IPADDRESS                      166
#define X509V3_R_INVALID_NAME                           106
#define X509V3_R_INVALID_NULL_ARGUMENT                  107
#define X509V3_R_INVALID_NULL_NAME                      108
#define X509V3_R_INVALID_NULL_VALUE                     109
#define X509V3_R_INVALID_NUMBER                         140
#define X509V3_R_INVALID_NUMBERS                        141
#define X509V3_R_INVALID_OBJECT_IDENTIFIER              110
#define X509V3_R_INVALID_OPTION                         138
#define X509V3_R_INVALID_POLICY_IDENTIFIER              134
#define X509V3_R_INVALID_PROXY_POLICY_SETTING           153
#define X509V3_R_INVALID_PURPOSE                        146
#define X509V3_R_INVALID_SAFI                           164
#define X509V3_R_INVALID_SECTION                        135
#define X509V3_R_INVALID_SYNTAX                         143
#define X509V3_R_ISSUER_DECODE_ERROR                    126
#define X509V3_R_MISSING_VALUE                          124
#define X509V3_R_NEED_ORGANIZATION_AND_NUMBERS          142
#define X509V3_R_NO_CONFIG_DATABASE                     136
#define X509V3_R_NO_ISSUER_CERTIFICATE                  121
#define X509V3_R_NO_ISSUER_DETAILS                      127
#define X509V3_R_NO_POLICY_IDENTIFIER                   139
#define X509V3_R_NO_PROXY_CERT_POLICY_LANGUAGE_DEFINED  154
#define X509V3_R_NO_PUBLIC_KEY                          114
#define X509V3_R_NO_SUBJECT_DETAILS                     125
#define X509V3_R_ODD_NUMBER_OF_DIGITS                   112
```



```
#define X509V3_R_OPERATION_NOT_DEFINED                  148
#define X509V3_R_OTHERNAME_ERROR                        147
#define X509V3_R_POLICY_LANGUAGE_ALREADY_DEFINED        155
#define X509V3_R_POLICY_PATH_LENGTH                     156
#define X509V3_R_POLICY_PATH_LENGTH_ALREADY_DEFINED     157
#define X509V3_R_POLICY_SYNTAX_NOT_CURRENTLY_SUPPORTED  158
#define X509V3_R_POLICY_WHEN_PROXY_LANGUAGE_REQUIRES_NO_POLICY 159
#define X509V3_R_SECTION_NOT_FOUND                      150
#define X509V3_R_UNABLE_TO_GET_ISSUER_DETAILS           122
#define X509V3_R_UNABLE_TO_GET_ISSUER_KEYID             123
#define X509V3_R_UNKNOWN_BIT_STRING_ARGUMENT            111
#define X509V3_R_UNKNOWN_EXTENSION                      129
#define X509V3_R_UNKNOWN_EXTENSION_NAME                 130
#define X509V3_R_UNKNOWN_OPTION                         120
#define X509V3_R_UNSUPPORTED_OPTION                     117
#define X509V3_R_UNSUPPORTED_TYPE                       167
#define X509V3_R_USER_TOO_LONG                          132

#ifdef  __cplusplus
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    7000 Wed Aug 13 19:51:51 2014
new/usr/src/lib/openssl/include/pcy_int.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pcy_int.h */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


typedef struct X509_POLICY_DATA_st X509_POLICY_DATA;
```



```
DECLARE_STACK_OF(X509_POLICY_DATA)

/* Internal structures */

/* This structure and the field names correspond to the Policy 'node' of
 * RFC3280. NB this structure contains no pointers to parent or child
 * data: X509_POLICY_NODE contains that. This means that the main policy data
 * can be kept static and cached with the certificate.
 */

struct X509_POLICY_DATA_st
        {
        unsigned int flags;
        /* Policy OID and qualifiers for this data */
        ASN1_OBJECT *valid_policy;
        STACK_OF(POLICYQUALINFO) *qualifier_set;
        STACK_OF(ASN1_OBJECT) *expected_policy_set;
        };

/* X509_POLICY_DATA flags values */

/* This flag indicates the structure has been mapped using a policy mapping
 * extension. If policy mapping is not active its references get deleted.
 */

#define POLICY_DATA_FLAG_MAPPED                 0x1

/* This flag indicates the data doesn't correspond to a policy in Certificate
 * Policies: it has been mapped to any policy.
 */

#define POLICY_DATA_FLAG_MAPPED_ANY             0x2

/* AND with flags to see if any mapping has occurred */

#define POLICY_DATA_FLAG_MAP_MASK               0x3

/* qualifiers are shared and shouldn't be freed */

#define POLICY_DATA_FLAG_SHARED_QUALIFIERS      0x4

/* Parent node is an extra node and should be freed */

#define POLICY_DATA_FLAG_EXTRA_NODE             0x8

/* Corresponding CertificatePolicies is critical */

#define POLICY_DATA_FLAG_CRITICAL               0x10

/* This structure is cached with a certificate */

struct X509_POLICY_CACHE_st {
        /* anyPolicy data or NULL if no anyPolicy */
        X509_POLICY_DATA *anyPolicy;
        /* other policy data */
        STACK_OF(X509_POLICY_DATA) *data;
        /* If InhibitAnyPolicy present this is its value or -1 if absent. */
        long any_skip;
        /* If policyConstraints and requireExplicitPolicy present this is its
         * value or -1 if absent.
         */
        long explicit_skip;
        /* If policyConstraints and policyMapping present this is its
         * value or -1 if absent.
         */
        long map_skip;
```

```
128         };

130 /*#define POLICY_CACHE_FLAG_CRITICAL            POLICY_DATA_FLAG_CRITICAL*/

132 /* This structure represents the relationship between nodes */

134 struct X509_POLICY_NODE_st
135         {
136         /* node data this refers to */
137         const X509_POLICY_DATA *data;
138         /* Parent node */
139         X509_POLICY_NODE *parent;
140         /* Number of child nodes */
141         int nchild;
142         };

144 struct X509_POLICY_LEVEL_st
145         {
146         /* Cert for this level */
147         X509 *cert;
148         /* nodes at this level */
149         STACK_OF(X509_POLICY_NODE) *nodes;
150         /* anyPolicy node */
151         X509_POLICY_NODE *anyPolicy;
152         /* Extra data */
153         /*STACK_OF(X509_POLICY_DATA) *extra_data;*/
154         unsigned int flags;
155         };

157 struct X509_POLICY_TREE_st
158         {
159         /* This is the tree 'level' data */
160         X509_POLICY_LEVEL *levels;
161         int nlevel;
162         /* Extra policy data when additional nodes (not from the certificate)
163          * are required.
164          */
165         STACK_OF(X509_POLICY_DATA) *extra_data;
166         /* This is the authority constained policy set */
167         STACK_OF(X509_POLICY_NODE) *auth_policies;
168         STACK_OF(X509_POLICY_NODE) *user_policies;
169         unsigned int flags;
170         };

172 /* Set if anyPolicy present in user policies */
173 #define POLICY_FLAG_ANY_POLICY          0x2

175 /* Useful macros */

177 #define node_data_critical(data) (data->flags & POLICY_DATA_FLAG_CRITICAL)
178 #define node_critical(node) node_data_critical(node->data)

180 /* Internal functions */

182 X509_POLICY_DATA *policy_data_new(POLICYINFO *policy, const ASN1_OBJECT *id,
183                                                         int crit);
184 void policy_data_free(X509_POLICY_DATA *data);

186 X509_POLICY_DATA *policy_cache_find_data(const X509_POLICY_CACHE *cache,
187                                                 const ASN1_OBJECT *id);
188 int policy_cache_set_mapping(X509 *x, POLICY_MAPPINGS *maps);


191 STACK_OF(X509_POLICY_NODE) *policy_node_cmp_new(void);

193 void policy_cache_init(void);
```



```
195 void policy_cache_free(X509_POLICY_CACHE *cache);

197 X509_POLICY_NODE *level_find_node(const X509_POLICY_LEVEL *level,
198                                   const X509_POLICY_NODE *parent,
199                                   const ASN1_OBJECT *id);

201 X509_POLICY_NODE *tree_find_sk(STACK_OF(X509_POLICY_NODE) *sk,
202                                         const ASN1_OBJECT *id);

204 X509_POLICY_NODE *level_add_node(X509_POLICY_LEVEL *level,
205                         const X509_POLICY_DATA *data,
206                         X509_POLICY_NODE *parent,
207                         X509_POLICY_TREE *tree);
208 void policy_node_free(X509_POLICY_NODE *node);
209 int policy_node_match(const X509_POLICY_LEVEL *lvl,
210                       const X509_POLICY_NODE *node, const ASN1_OBJECT *oid);

212 const X509_POLICY_CACHE *policy_cache_set(X509 *x);
213 #endif /* ! codereview */
```

```
**********************************************************
    9275 Wed Aug 13 19:51:51 2014
new/usr/src/lib/openssl/include/pkcs11.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pkcs11.h include file for PKCS #11. */
/* $Revision: 1.4 $ */

/* License to copy and use this software is granted provided that it is
 * identified as "RSA Security Inc. PKCS #11 Cryptographic Token Interface
 * (Cryptoki)" in all material mentioning or referencing this software.

 * License is also granted to make and use derivative works provided that
 * such works are identified as "derived from the RSA Security Inc. PKCS #11
 * Cryptographic Token Interface (Cryptoki)" in all material mentioning or
 * referencing the derived work.

 * RSA Security Inc. makes no representations concerning either the
 * merchantability of this software or the suitability of this software for
 * any particular purpose. It is provided "as is" without express or implied
 * warranty of any kind.
 */

#ifndef _PKCS11_H_
#define _PKCS11_H_ 1

#ifdef __cplusplus
extern "C" {
#endif

/* Before including this file (pkcs11.h) (or pkcs11t.h by
 * itself), 6 platform-specific macros must be defined.  These
 * macros are described below, and typical definitions for them
 * are also given.  Be advised that these definitions can depend
 * on both the platform and the compiler used (and possibly also
 * on whether a Cryptoki library is linked statically or
 * dynamically).
 *
 * In addition to defining these 6 macros, the packing convention
 * for Cryptoki structures should be set.  The Cryptoki
 * convention on packing is that structures should be 1-byte
 * aligned.
 *
 * If you're using Microsoft Developer Studio 5.0 to produce
 * Win32 stuff, this might be done by using the following
 * preprocessor directive before including pkcs11.h or pkcs11t.h:
 *
 * #pragma pack(push, cryptoki, 1)
 *
 * and using the following preprocessor directive after including
 * pkcs11.h or pkcs11t.h:
 *
 * #pragma pack(pop, cryptoki)
 *
 * If you're using an earlier version of Microsoft Developer
 * Studio to produce Win16 stuff, this might be done by using
 * the following preprocessor directive before including
 * pkcs11.h or pkcs11t.h:
 *
 * #pragma pack(1)
 *
 * In a UNIX environment, you're on your own for this.  You might
 * not need to do (or be able to do!) anything.
 *
 *
 * Now for the macros:
```



```
 *
 *
 * 1. CK_PTR: The indirection string for making a pointer to an
 * object.  It can be used like this:
 *
 * typedef CK_BYTE CK_PTR CK_BYTE_PTR;
 *
 * If you're using Microsoft Developer Studio 5.0 to produce
 * Win32 stuff, it might be defined by:
 *
 * #define CK_PTR *
 *
 * If you're using an earlier version of Microsoft Developer
 * Studio to produce Win16 stuff, it might be defined by:
 *
 * #define CK_PTR far *
 *
 * In a typical UNIX environment, it might be defined by:
 *
 * #define CK_PTR *
 *
 *
 * 2. CK_DEFINE_FUNCTION(returnType, name): A macro which makes
 * an exportable Cryptoki library function definition out of a
 * return type and a function name.  It should be used in the
 * following fashion to define the exposed Cryptoki functions in
 * a Cryptoki library:
 *
 * CK_DEFINE_FUNCTION(CK_RV, C_Initialize)(
 *   CK_VOID_PTR pReserved
 * )
 * {
 *   ...
 * }
 *
 * If you're using Microsoft Developer Studio 5.0 to define a
 * function in a Win32 Cryptoki .dll, it might be defined by:
 *
 * #define CK_DEFINE_FUNCTION(returnType, name) \
 *   returnType __declspec(dllexport) name
 *
 * If you're using an earlier version of Microsoft Developer
 * Studio to define a function in a Win16 Cryptoki .dll, it
 * might be defined by:
 *
 * #define CK_DEFINE_FUNCTION(returnType, name) \
 *   returnType __export _far _pascal name
 *
 * In a UNIX environment, it might be defined by:
 *
 * #define CK_DEFINE_FUNCTION(returnType, name) \
 *   returnType name
 *
 *
 * 3. CK_DECLARE_FUNCTION(returnType, name): A macro which makes
 * an importable Cryptoki library function declaration out of a
 * return type and a function name.  It should be used in the
 * following fashion:
 *
 * extern CK_DECLARE_FUNCTION(CK_RV, C_Initialize)(
 *   CK_VOID_PTR pReserved
 * );
 *
 * If you're using Microsoft Developer Studio 5.0 to declare a
 * function in a Win32 Cryptoki .dll, it might be defined by:
 *
```

```
 * #define CK_DECLARE_FUNCTION(returnType, name) \
 *   returnType __declspec(dllimport) name
 *
 * If you're using an earlier version of Microsoft Developer
 * Studio to declare a function in a Win16 Cryptoki .dll, it
 * might be defined by:
 *
 * #define CK_DECLARE_FUNCTION(returnType, name) \
 *   returnType __export _far _pascal name
 *
 * In a UNIX environment, it might be defined by:
 *
 * #define CK_DECLARE_FUNCTION(returnType, name) \
 *   returnType name
 *
 *
 * 4. CK_DECLARE_FUNCTION_POINTER(returnType, name): A macro
 * which makes a Cryptoki API function pointer declaration or
 * function pointer type declaration out of a return type and a
 * function name.  It should be used in the following fashion:
 *
 * // Define funcPtr to be a pointer to a Cryptoki API function
 * // taking arguments args and returning CK_RV.
 * CK_DECLARE_FUNCTION_POINTER(CK_RV, funcPtr)(args);
 *
 * or
 *
 * // Define funcPtrType to be the type of a pointer to a
 * // Cryptoki API function taking arguments args and returning
 * // CK_RV, and then define funcPtr to be a variable of type
 * // funcPtrType.
 * typedef CK_DECLARE_FUNCTION_POINTER(CK_RV, funcPtrType)(args);
 * funcPtrType funcPtr;
 *
 * If you're using Microsoft Developer Studio 5.0 to access
 * functions in a Win32 Cryptoki .dll, in might be defined by:
 *
 * #define CK_DECLARE_FUNCTION_POINTER(returnType, name) \
 *   returnType __declspec(dllimport) (* name)
 *
 * If you're using an earlier version of Microsoft Developer
 * Studio to access functions in a Win16 Cryptoki .dll, it might
 * be defined by:
 *
 * #define CK_DECLARE_FUNCTION_POINTER(returnType, name) \
 *   returnType __export _far _pascal (* name)
 *
 * In a UNIX environment, it might be defined by:
 *
 * #define CK_DECLARE_FUNCTION_POINTER(returnType, name) \
 *   returnType (* name)
 *
 *
 * 5. CK_CALLBACK_FUNCTION(returnType, name): A macro which makes
 * a function pointer type for an application callback out of
 * a return type for the callback and a name for the callback.
 * It should be used in the following fashion:
 *
 * CK_CALLBACK_FUNCTION(CK_RV, myCallback)(args);
 *
 * to declare a function pointer, myCallback, to a callback
 * which takes arguments args and returns a CK_RV.  It can also
 * be used like this:
 *
 * typedef CK_CALLBACK_FUNCTION(CK_RV, myCallbackType)(args);
 * myCallbackType myCallback;
```



```
 *
 * If you're using Microsoft Developer Studio 5.0 to do Win32
 * Cryptoki development, it might be defined by:
 *
 * #define CK_CALLBACK_FUNCTION(returnType, name) \
 *   returnType (* name)
 *
 * If you're using an earlier version of Microsoft Developer
 * Studio to do Win16 development, it might be defined by:
 *
 * #define CK_CALLBACK_FUNCTION(returnType, name) \
 *   returnType _far _pascal (* name)
 *
 * In a UNIX environment, it might be defined by:
 *
 * #define CK_CALLBACK_FUNCTION(returnType, name) \
 *   returnType (* name)
 *
 *
 * 6. NULL_PTR: This macro is the value of a NULL pointer.
 *
 * In any ANSI/ISO C environment (and in many others as well),
 * this should best be defined by
 *
 * #ifndef NULL_PTR
 * #define NULL_PTR 0
 * #endif
 */


/* All the various Cryptoki types and #define'd values are in the
 * file pkcs11t.h. */
#include "pkcs11t.h"

#define __PASTE(x,y)      x##y


/* ==============================================================
 * Define the "extern" form of all the entry points.
 * ==============================================================
 */

#define CK_NEED_ARG_LIST  1
#define CK_PKCS11_FUNCTION_INFO(name) \
  extern CK_DECLARE_FUNCTION(CK_RV, name)

/* pkcs11f.h has all the information about the Cryptoki
 * function prototypes. */
#include "pkcs11f.h"

#undef CK_NEED_ARG_LIST
#undef CK_PKCS11_FUNCTION_INFO


/* ==============================================================
 * Define the typedef form of all the entry points.  That is, for
 * each Cryptoki function C_XXX, define a type CK_C_XXX which is
 * a pointer to that kind of function.
 * ==============================================================
 */

#define CK_NEED_ARG_LIST  1
#define CK_PKCS11_FUNCTION_INFO(name) \
  typedef CK_DECLARE_FUNCTION_POINTER(CK_RV, __PASTE(CK_,name))

/* pkcs11f.h has all the information about the Cryptoki
```

```
 * function prototypes. */
#include "pkcs11f.h"

#undef CK_NEED_ARG_LIST
#undef CK_PKCS11_FUNCTION_INFO


/* ==============================================================
 * Define structed vector of entry points.  A CK_FUNCTION_LIST
 * contains a CK_VERSION indicating a library's Cryptoki version
 * and then a whole slew of function pointers to the routines in
 * the library.  This type was declared, but not defined, in
 * pkcs11t.h.
 * ==============================================================
 */

#define CK_PKCS11_FUNCTION_INFO(name) \
  __PASTE(CK_,name) name;

struct CK_FUNCTION_LIST {

  CK_VERSION    version;  /* Cryptoki version */

/* Pile all the function pointers into the CK_FUNCTION_LIST. */
/* pkcs11f.h has all the information about the Cryptoki
 * function prototypes. */
#include "pkcs11f.h"

};

#undef CK_PKCS11_FUNCTION_INFO


#undef __PASTE

#ifdef __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
   28338 Wed Aug 13 19:51:52 2014
new/usr/src/lib/openssl/include/pkcs11f.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pkcs11f.h include file for PKCS #11. */
/* $Revision: 1.4 $ */

/* License to copy and use this software is granted provided that it is
 * identified as "RSA Security Inc. PKCS #11 Cryptographic Token Interface
 * (Cryptoki)" in all material mentioning or referencing this software.

 * License is also granted to make and use derivative works provided that
 * such works are identified as "derived from the RSA Security Inc. PKCS #11
 * Cryptographic Token Interface (Cryptoki)" in all material mentioning or
 * referencing the derived work.

 * RSA Security Inc. makes no representations concerning either the
 * merchantability of this software or the suitability of this software for
 * any particular purpose. It is provided "as is" without express or implied
 * warranty of any kind.
 */

/* This header file contains pretty much everything about all the */
/* Cryptoki function prototypes.  Because this information is */
/* used for more than just declaring function prototypes, the */
/* order of the functions appearing herein is important, and */
/* should not be altered. */

/* General-purpose */

/* C_Initialize initializes the Cryptoki library. */
CK_PKCS11_FUNCTION_INFO(C_Initialize)
#ifdef CK_NEED_ARG_LIST
(
  CK_VOID_PTR   pInitArgs  /* if this is not NULL_PTR, it gets
                            * cast to CK_C_INITIALIZE_ARGS_PTR
                            * and dereferenced */
);
#endif


/* C_Finalize indicates that an application is done with the
 * Cryptoki library. */
CK_PKCS11_FUNCTION_INFO(C_Finalize)
#ifdef CK_NEED_ARG_LIST
(
  CK_VOID_PTR   pReserved  /* reserved.  Should be NULL_PTR */
);
#endif


/* C_GetInfo returns general information about Cryptoki. */
CK_PKCS11_FUNCTION_INFO(C_GetInfo)
#ifdef CK_NEED_ARG_LIST
(
  CK_INFO_PTR   pInfo  /* location that receives information */
);
#endif


/* C_GetFunctionList returns the function list. */
CK_PKCS11_FUNCTION_INFO(C_GetFunctionList)
#ifdef CK_NEED_ARG_LIST
(
  CK_FUNCTION_LIST_PTR_PTR ppFunctionList  /* receives pointer to
```



```
                                           * function list */
);
#endif



/* Slot and token management */

/* C_GetSlotList obtains a list of slots in the system. */
CK_PKCS11_FUNCTION_INFO(C_GetSlotList)
#ifdef CK_NEED_ARG_LIST
(
  CK_BBOOL       tokenPresent,  /* only slots with tokens? */
  CK_SLOT_ID_PTR pSlotList,     /* receives array of slot IDs */
  CK_ULONG_PTR   pulCount       /* receives number of slots */
);
#endif


/* C_GetSlotInfo obtains information about a particular slot in
 * the system. */
CK_PKCS11_FUNCTION_INFO(C_GetSlotInfo)
#ifdef CK_NEED_ARG_LIST
(
  CK_SLOT_ID       slotID,  /* the ID of the slot */
  CK_SLOT_INFO_PTR pInfo    /* receives the slot information */
);
#endif


/* C_GetTokenInfo obtains information about a particular token
 * in the system. */
CK_PKCS11_FUNCTION_INFO(C_GetTokenInfo)
#ifdef CK_NEED_ARG_LIST
(
  CK_SLOT_ID        slotID,  /* ID of the token's slot */
  CK_TOKEN_INFO_PTR pInfo    /* receives the token information */
);
#endif


/* C_GetMechanismList obtains a list of mechanism types
 * supported by a token. */
CK_PKCS11_FUNCTION_INFO(C_GetMechanismList)
#ifdef CK_NEED_ARG_LIST
(
  CK_SLOT_ID            slotID,          /* ID of token's slot */
  CK_MECHANISM_TYPE_PTR pMechanismList,  /* gets mech. array */
  CK_ULONG_PTR          pulCount         /* gets # of mechs. */
);
#endif


/* C_GetMechanismInfo obtains information about a particular
 * mechanism possibly supported by a token. */
CK_PKCS11_FUNCTION_INFO(C_GetMechanismInfo)
#ifdef CK_NEED_ARG_LIST
(
  CK_SLOT_ID            slotID,  /* ID of the token's slot */
  CK_MECHANISM_TYPE     type,    /* type of mechanism */
  CK_MECHANISM_INFO_PTR pInfo    /* receives mechanism info */
);
#endif


/* C_InitToken initializes a token. */
```

```
128 CK_PKCS11_FUNCTION_INFO(C_InitToken)
129 #ifdef CK_NEED_ARG_LIST
130 /* pLabel changed from CK_CHAR_PTR to CK_UTF8CHAR_PTR for v2.10 */
131 (
132   CK_SLOT_ID      slotID,    /* ID of the token's slot */
133   CK_UTF8CHAR_PTR pPin,      /* the SO's initial PIN */
134   CK_ULONG        ulPinLen,  /* length in bytes of the PIN */
135   CK_UTF8CHAR_PTR pLabel     /* 32-byte token label (blank padded) */
136 );
137 #endif


140 /* C_InitPIN initializes the normal user's PIN. */
141 CK_PKCS11_FUNCTION_INFO(C_InitPIN)
142 #ifdef CK_NEED_ARG_LIST
143 (
144   CK_SESSION_HANDLE hSession,  /* the session's handle */
145   CK_UTF8CHAR_PTR   pPin,      /* the normal user's PIN */
146   CK_ULONG          ulPinLen   /* length in bytes of the PIN */
147 );
148 #endif


151 /* C_SetPIN modifies the PIN of the user who is logged in. */
152 CK_PKCS11_FUNCTION_INFO(C_SetPIN)
153 #ifdef CK_NEED_ARG_LIST
154 (
155   CK_SESSION_HANDLE hSession,  /* the session's handle */
156   CK_UTF8CHAR_PTR   pOldPin,   /* the old PIN */
157   CK_ULONG          ulOldLen,  /* length of the old PIN */
158   CK_UTF8CHAR_PTR   pNewPin,   /* the new PIN */
159   CK_ULONG          ulNewLen   /* length of the new PIN */
160 );
161 #endif



165 /* Session management */

167 /* C_OpenSession opens a session between an application and a
168  * token. */
169 CK_PKCS11_FUNCTION_INFO(C_OpenSession)
170 #ifdef CK_NEED_ARG_LIST
171 (
172   CK_SLOT_ID            slotID,        /* the slot's ID */
173   CK_FLAGS              flags,         /* from CK_SESSION_INFO */
174   CK_VOID_PTR           pApplication,  /* passed to callback */
175   CK_NOTIFY             Notify,        /* callback function */
176   CK_SESSION_HANDLE_PTR phSession      /* gets session handle */
177 );
178 #endif


181 /* C_CloseSession closes a session between an application and a
182  * token. */
183 CK_PKCS11_FUNCTION_INFO(C_CloseSession)
184 #ifdef CK_NEED_ARG_LIST
185 (
186   CK_SESSION_HANDLE hSession  /* the session's handle */
187 );
188 #endif


191 /* C_CloseAllSessions closes all sessions with a token. */
192 CK_PKCS11_FUNCTION_INFO(C_CloseAllSessions)
193 #ifdef CK_NEED_ARG_LIST
```



```
194 (
195   CK_SLOT_ID     slotID  /* the token's slot */
196 );
197 #endif


200 /* C_GetSessionInfo obtains information about the session. */
201 CK_PKCS11_FUNCTION_INFO(C_GetSessionInfo)
202 #ifdef CK_NEED_ARG_LIST
203 (
204   CK_SESSION_HANDLE   hSession,  /* the session's handle */
205   CK_SESSION_INFO_PTR pInfo      /* receives session info */
206 );
207 #endif


210 /* C_GetOperationState obtains the state of the cryptographic operation
211  * in a session. */
212 CK_PKCS11_FUNCTION_INFO(C_GetOperationState)
213 #ifdef CK_NEED_ARG_LIST
214 (
215   CK_SESSION_HANDLE hSession,             /* session's handle */
216   CK_BYTE_PTR       pOperationState,      /* gets state */
217   CK_ULONG_PTR      pulOperationStateLen  /* gets state length */
218 );
219 #endif


222 /* C_SetOperationState restores the state of the cryptographic
223  * operation in a session. */
224 CK_PKCS11_FUNCTION_INFO(C_SetOperationState)
225 #ifdef CK_NEED_ARG_LIST
226 (
227   CK_SESSION_HANDLE hSession,            /* session's handle */
228   CK_BYTE_PTR      pOperationState,      /* holds state */
229   CK_ULONG         ulOperationStateLen,  /* holds state length */
230   CK_OBJECT_HANDLE hEncryptionKey,       /* en/decryption key */
231   CK_OBJECT_HANDLE hAuthenticationKey    /* sign/verify key */
232 );
233 #endif


236 /* C_Login logs a user into a token. */
237 CK_PKCS11_FUNCTION_INFO(C_Login)
238 #ifdef CK_NEED_ARG_LIST
239 (
240   CK_SESSION_HANDLE hSession,  /* the session's handle */
241   CK_USER_TYPE      userType,  /* the user type */
242   CK_UTF8CHAR_PTR   pPin,      /* the user's PIN */
243   CK_ULONG          ulPinLen   /* the length of the PIN */
244 );
245 #endif


248 /* C_Logout logs a user out from a token. */
249 CK_PKCS11_FUNCTION_INFO(C_Logout)
250 #ifdef CK_NEED_ARG_LIST
251 (
252   CK_SESSION_HANDLE hSession  /* the session's handle */
253 );
254 #endif



258 /* Object management */
```

```
260 /* C_CreateObject creates a new object. */
261 CK_PKCS11_FUNCTION_INFO(C_CreateObject)
262 #ifdef CK_NEED_ARG_LIST
263 (
264   CK_SESSION_HANDLE hSession,    /* the session's handle */
265   CK_ATTRIBUTE_PTR  pTemplate,   /* the object's template */
266   CK_ULONG          ulCount,     /* attributes in template */
267   CK_OBJECT_HANDLE_PTR phObject  /* gets new object's handle. */
268 );
269 #endif


272 /* C_CopyObject copies an object, creating a new object for the
273  * copy. */
274 CK_PKCS11_FUNCTION_INFO(C_CopyObject)
275 #ifdef CK_NEED_ARG_LIST
276 (
277   CK_SESSION_HANDLE    hSession,    /* the session's handle */
278   CK_OBJECT_HANDLE     hObject,     /* the object's handle */
279   CK_ATTRIBUTE_PTR     pTemplate,   /* template for new object */
280   CK_ULONG             ulCount,     /* attributes in template */
281   CK_OBJECT_HANDLE_PTR phNewObject  /* receives handle of copy */
282 );
283 #endif


286 /* C_DestroyObject destroys an object. */
287 CK_PKCS11_FUNCTION_INFO(C_DestroyObject)
288 #ifdef CK_NEED_ARG_LIST
289 (
290   CK_SESSION_HANDLE hSession,  /* the session's handle */
291   CK_OBJECT_HANDLE  hObject    /* the object's handle */
292 );
293 #endif


296 /* C_GetObjectSize gets the size of an object in bytes. */
297 CK_PKCS11_FUNCTION_INFO(C_GetObjectSize)
298 #ifdef CK_NEED_ARG_LIST
299 (
300   CK_SESSION_HANDLE hSession,  /* the session's handle */
301   CK_OBJECT_HANDLE  hObject,   /* the object's handle */
302   CK_ULONG_PTR      pulSize    /* receives size of object */
303 );
304 #endif


307 /* C_GetAttributeValue obtains the value of one or more object
308  * attributes. */
309 CK_PKCS11_FUNCTION_INFO(C_GetAttributeValue)
310 #ifdef CK_NEED_ARG_LIST
311 (
312   CK_SESSION_HANDLE hSession,   /* the session's handle */
313   CK_OBJECT_HANDLE  hObject,    /* the object's handle */
314   CK_ATTRIBUTE_PTR  pTemplate,  /* specifies attrs; gets vals */
315   CK_ULONG          ulCount     /* attributes in template */
316 );
317 #endif


320 /* C_SetAttributeValue modifies the value of one or more object
321  * attributes */
322 CK_PKCS11_FUNCTION_INFO(C_SetAttributeValue)
323 #ifdef CK_NEED_ARG_LIST
324 (
325   CK_SESSION_HANDLE hSession,   /* the session's handle */
```



```
326   CK_OBJECT_HANDLE  hObject,    /* the object's handle */
327   CK_ATTRIBUTE_PTR  pTemplate,  /* specifies attrs and values */
328   CK_ULONG          ulCount     /* attributes in template */
329 );
330 #endif


333 /* C_FindObjectsInit initializes a search for token and session
334  * objects that match a template. */
335 CK_PKCS11_FUNCTION_INFO(C_FindObjectsInit)
336 #ifdef CK_NEED_ARG_LIST
337 (
338   CK_SESSION_HANDLE hSession,   /* the session's handle */
339   CK_ATTRIBUTE_PTR  pTemplate,  /* attribute values to match */
340   CK_ULONG          ulCount     /* attrs in search template */
341 );
342 #endif


345 /* C_FindObjects continues a search for token and session
346  * objects that match a template, obtaining additional object
347  * handles. */
348 CK_PKCS11_FUNCTION_INFO(C_FindObjects)
349 #ifdef CK_NEED_ARG_LIST
350 (
351  CK_SESSION_HANDLE    hSession,          /* session's handle */
352  CK_OBJECT_HANDLE_PTR phObject,          /* gets obj. handles */
353  CK_ULONG             ulMaxObjectCount,  /* max handles to get */
354  CK_ULONG_PTR         pulObjectCount     /* actual # returned */
355 );
356 #endif


359 /* C_FindObjectsFinal finishes a search for token and session
360  * objects. */
361 CK_PKCS11_FUNCTION_INFO(C_FindObjectsFinal)
362 #ifdef CK_NEED_ARG_LIST
363 (
364   CK_SESSION_HANDLE hSession  /* the session's handle */
365 );
366 #endif



370 /* Encryption and decryption */

372 /* C_EncryptInit initializes an encryption operation. */
373 CK_PKCS11_FUNCTION_INFO(C_EncryptInit)
374 #ifdef CK_NEED_ARG_LIST
375 (
376   CK_SESSION_HANDLE hSession,    /* the session's handle */
377   CK_MECHANISM_PTR  pMechanism,  /* the encryption mechanism */
378   CK_OBJECT_HANDLE  hKey         /* handle of encryption key */
379 );
380 #endif


383 /* C_Encrypt encrypts single-part data. */
384 CK_PKCS11_FUNCTION_INFO(C_Encrypt)
385 #ifdef CK_NEED_ARG_LIST
386 (
387   CK_SESSION_HANDLE hSession,            /* session's handle */
388   CK_BYTE_PTR       pData,               /* the plaintext data */
389   CK_ULONG          ulDataLen,           /* bytes of plaintext */
390   CK_BYTE_PTR       pEncryptedData,      /* gets ciphertext */
391   CK_ULONG_PTR      pulEncryptedDataLen  /* gets c-text size */
```

```
392 );
393 #endif


396 /* C_EncryptUpdate continues a multiple-part encryption
397  * operation. */
398 CK_PKCS11_FUNCTION_INFO(C_EncryptUpdate)
399 #ifdef CK_NEED_ARG_LIST
400 (
401   CK_SESSION_HANDLE hSession,           /* session's handle */
402   CK_BYTE_PTR       pPart,              /* the plaintext data */
403   CK_ULONG          ulPartLen,          /* plaintext data len */
404   CK_BYTE_PTR       pEncryptedPart,     /* gets ciphertext */
405   CK_ULONG_PTR      pulEncryptedPartLen /* gets c-text size */
406 );
407 #endif


410 /* C_EncryptFinal finishes a multiple-part encryption
411  * operation. */
412 CK_PKCS11_FUNCTION_INFO(C_EncryptFinal)
413 #ifdef CK_NEED_ARG_LIST
414 (
415   CK_SESSION_HANDLE hSession,                /* session handle */
416   CK_BYTE_PTR       pLastEncryptedPart,      /* last c-text */
417   CK_ULONG_PTR      pulLastEncryptedPartLen  /* gets last size */
418 );
419 #endif


422 /* C_DecryptInit initializes a decryption operation. */
423 CK_PKCS11_FUNCTION_INFO(C_DecryptInit)
424 #ifdef CK_NEED_ARG_LIST
425 (
426   CK_SESSION_HANDLE hSession,    /* the session's handle */
427   CK_MECHANISM_PTR  pMechanism,  /* the decryption mechanism */
428   CK_OBJECT_HANDLE  hKey         /* handle of decryption key */
429 );
430 #endif


433 /* C_Decrypt decrypts encrypted data in a single part. */
434 CK_PKCS11_FUNCTION_INFO(C_Decrypt)
435 #ifdef CK_NEED_ARG_LIST
436 (
437   CK_SESSION_HANDLE hSession,           /* session's handle */
438   CK_BYTE_PTR       pEncryptedData,     /* ciphertext */
439   CK_ULONG          ulEncryptedDataLen, /* ciphertext length */
440   CK_BYTE_PTR       pData,              /* gets plaintext */
441   CK_ULONG_PTR      pulDataLen          /* gets p-text size */
442 );
443 #endif


446 /* C_DecryptUpdate continues a multiple-part decryption
447  * operation. */
448 CK_PKCS11_FUNCTION_INFO(C_DecryptUpdate)
449 #ifdef CK_NEED_ARG_LIST
450 (
451   CK_SESSION_HANDLE hSession,            /* session's handle */
452   CK_BYTE_PTR       pEncryptedPart,      /* encrypted data */
453   CK_ULONG          ulEncryptedPartLen,  /* input length */
454   CK_BYTE_PTR       pPart,               /* gets plaintext */
455   CK_ULONG_PTR      pulPartLen           /* p-text size */
456 );
457 #endif
```



```


460 /* C_DecryptFinal finishes a multiple-part decryption
461  * operation. */
462 CK_PKCS11_FUNCTION_INFO(C_DecryptFinal)
463 #ifdef CK_NEED_ARG_LIST
464 (
465   CK_SESSION_HANDLE hSession,       /* the session's handle */
466   CK_BYTE_PTR       pLastPart,      /* gets plaintext */
467   CK_ULONG_PTR      pulLastPartLen  /* p-text size */
468 );
469 #endif



473 /* Message digesting */

475 /* C_DigestInit initializes a message-digesting operation. */
476 CK_PKCS11_FUNCTION_INFO(C_DigestInit)
477 #ifdef CK_NEED_ARG_LIST
478 (
479   CK_SESSION_HANDLE hSession,   /* the session's handle */
480   CK_MECHANISM_PTR  pMechanism  /* the digesting mechanism */
481 );
482 #endif


485 /* C_Digest digests data in a single part. */
486 CK_PKCS11_FUNCTION_INFO(C_Digest)
487 #ifdef CK_NEED_ARG_LIST
488 (
489   CK_SESSION_HANDLE hSession,     /* the session's handle */
490   CK_BYTE_PTR       pData,        /* data to be digested */
491   CK_ULONG          ulDataLen,    /* bytes of data to digest */
492   CK_BYTE_PTR       pDigest,      /* gets the message digest */
493   CK_ULONG_PTR      pulDigestLen  /* gets digest length */
494 );
495 #endif


498 /* C_DigestUpdate continues a multiple-part message-digesting
499  * operation. */
500 CK_PKCS11_FUNCTION_INFO(C_DigestUpdate)
501 #ifdef CK_NEED_ARG_LIST
502 (
503   CK_SESSION_HANDLE hSession,  /* the session's handle */
504   CK_BYTE_PTR       pPart,     /* data to be digested */
505   CK_ULONG          ulPartLen  /* bytes of data to be digested */
506 );
507 #endif


510 /* C_DigestKey continues a multi-part message-digesting
511  * operation, by digesting the value of a secret key as part of
512  * the data already digested. */
513 CK_PKCS11_FUNCTION_INFO(C_DigestKey)
514 #ifdef CK_NEED_ARG_LIST
515 (
516   CK_SESSION_HANDLE hSession,  /* the session's handle */
517   CK_OBJECT_HANDLE  hKey       /* secret key to digest */
518 );
519 #endif


522 /* C_DigestFinal finishes a multiple-part message-digesting
523  * operation. */
```

```
524 CK_PKCS11_FUNCTION_INFO(C_DigestFinal)
525 #ifdef CK_NEED_ARG_LIST
526 (
527   CK_SESSION_HANDLE hSession,     /* the session's handle */
528   CK_BYTE_PTR       pDigest,      /* gets the message digest */
529   CK_ULONG_PTR      pulDigestLen  /* gets byte count of digest */
530 );
531 #endif



535 /* Signing and MACing */

537 /* C_SignInit initializes a signature (private key encryption)
538  * operation, where the signature is (will be) an appendix to
539  * the data, and plaintext cannot be recovered from the
540  *signature. */
541 CK_PKCS11_FUNCTION_INFO(C_SignInit)
542 #ifdef CK_NEED_ARG_LIST
543 (
544   CK_SESSION_HANDLE hSession,    /* the session's handle */
545   CK_MECHANISM_PTR  pMechanism,  /* the signature mechanism */
546   CK_OBJECT_HANDLE  hKey         /* handle of signature key */
547 );
548 #endif


551 /* C_Sign signs (encrypts with private key) data in a single
552  * part, where the signature is (will be) an appendix to the
553  * data, and plaintext cannot be recovered from the signature. */
554 CK_PKCS11_FUNCTION_INFO(C_Sign)
555 #ifdef CK_NEED_ARG_LIST
556 (
557   CK_SESSION_HANDLE hSession,        /* the session's handle */
558   CK_BYTE_PTR       pData,           /* the data to sign */
559   CK_ULONG          ulDataLen,       /* count of bytes to sign */
560   CK_BYTE_PTR       pSignature,      /* gets the signature */
561   CK_ULONG_PTR      pulSignatureLen  /* gets signature length */
562 );
563 #endif


566 /* C_SignUpdate continues a multiple-part signature operation,
567  * where the signature is (will be) an appendix to the data,
568  * and plaintext cannot be recovered from the signature. */
569 CK_PKCS11_FUNCTION_INFO(C_SignUpdate)
570 #ifdef CK_NEED_ARG_LIST
571 (
572   CK_SESSION_HANDLE hSession,  /* the session's handle */
573   CK_BYTE_PTR       pPart,     /* the data to sign */
574   CK_ULONG          ulPartLen  /* count of bytes to sign */
575 );
576 #endif


579 /* C_SignFinal finishes a multiple-part signature operation,
580  * returning the signature. */
581 CK_PKCS11_FUNCTION_INFO(C_SignFinal)
582 #ifdef CK_NEED_ARG_LIST
583 (
584   CK_SESSION_HANDLE hSession,        /* the session's handle */
585   CK_BYTE_PTR       pSignature,      /* gets the signature */
586   CK_ULONG_PTR      pulSignatureLen  /* gets signature length */
587 );
588 #endif
```



```
591 /* C_SignRecoverInit initializes a signature operation, where
592  * the data can be recovered from the signature. */
593 CK_PKCS11_FUNCTION_INFO(C_SignRecoverInit)
594 #ifdef CK_NEED_ARG_LIST
595 (
596   CK_SESSION_HANDLE hSession,   /* the session's handle */
597   CK_MECHANISM_PTR  pMechanism, /* the signature mechanism */
598   CK_OBJECT_HANDLE  hKey        /* handle of the signature key */
599 );
600 #endif


603 /* C_SignRecover signs data in a single operation, where the
604  * data can be recovered from the signature. */
605 CK_PKCS11_FUNCTION_INFO(C_SignRecover)
606 #ifdef CK_NEED_ARG_LIST
607 (
608   CK_SESSION_HANDLE hSession,        /* the session's handle */
609   CK_BYTE_PTR       pData,           /* the data to sign */
610   CK_ULONG          ulDataLen,       /* count of bytes to sign */
611   CK_BYTE_PTR       pSignature,      /* gets the signature */
612   CK_ULONG_PTR      pulSignatureLen  /* gets signature length */
613 );
614 #endif



618 /* Verifying signatures and MACs */

620 /* C_VerifyInit initializes a verification operation, where the
621  * signature is an appendix to the data, and plaintext cannot
622  *  cannot be recovered from the signature (e.g. DSA). */
623 CK_PKCS11_FUNCTION_INFO(C_VerifyInit)
624 #ifdef CK_NEED_ARG_LIST
625 (
626   CK_SESSION_HANDLE hSession,    /* the session's handle */
627   CK_MECHANISM_PTR  pMechanism,  /* the verification mechanism */
628   CK_OBJECT_HANDLE  hKey         /* verification key */
629 );
630 #endif


633 /* C_Verify verifies a signature in a single-part operation,
634  * where the signature is an appendix to the data, and plaintext
635  * cannot be recovered from the signature. */
636 CK_PKCS11_FUNCTION_INFO(C_Verify)
637 #ifdef CK_NEED_ARG_LIST
638 (
639   CK_SESSION_HANDLE hSession,       /* the session's handle */
640   CK_BYTE_PTR       pData,          /* signed data */
641   CK_ULONG          ulDataLen,      /* length of signed data */
642   CK_BYTE_PTR       pSignature,     /* signature */
643   CK_ULONG          ulSignatureLen  /* signature length*/
644 );
645 #endif


648 /* C_VerifyUpdate continues a multiple-part verification
649  * operation, where the signature is an appendix to the data,
650  * and plaintext cannot be recovered from the signature. */
651 CK_PKCS11_FUNCTION_INFO(C_VerifyUpdate)
652 #ifdef CK_NEED_ARG_LIST
653 (
654   CK_SESSION_HANDLE hSession,  /* the session's handle */
655   CK_BYTE_PTR       pPart,     /* signed data */
```

```
656   CK_ULONG          ulPartLen  /* length of signed data */
657 );
658 #endif


661 /* C_VerifyFinal finishes a multiple-part verification
662  * operation, checking the signature. */
663 CK_PKCS11_FUNCTION_INFO(C_VerifyFinal)
664 #ifdef CK_NEED_ARG_LIST
665 (
666   CK_SESSION_HANDLE hSession,       /* the session's handle */
667   CK_BYTE_PTR       pSignature,     /* signature to verify */
668   CK_ULONG          ulSignatureLen  /* signature length */
669 );
670 #endif


673 /* C_VerifyRecoverInit initializes a signature verification
674  * operation, where the data is recovered from the signature. */
675 CK_PKCS11_FUNCTION_INFO(C_VerifyRecoverInit)
676 #ifdef CK_NEED_ARG_LIST
677 (
678   CK_SESSION_HANDLE hSession,    /* the session's handle */
679   CK_MECHANISM_PTR  pMechanism,  /* the verification mechanism */
680   CK_OBJECT_HANDLE  hKey         /* verification key */
681 );
682 #endif


685 /* C_VerifyRecover verifies a signature in a single-part
686  * operation, where the data is recovered from the signature. */
687 CK_PKCS11_FUNCTION_INFO(C_VerifyRecover)
688 #ifdef CK_NEED_ARG_LIST
689 (
690   CK_SESSION_HANDLE hSession,        /* the session's handle */
691   CK_BYTE_PTR       pSignature,      /* signature to verify */
692   CK_ULONG          ulSignatureLen,  /* signature length */
693   CK_BYTE_PTR       pData,           /* gets signed data */
694   CK_ULONG_PTR      pulDataLen       /* gets signed data len */
695 );
696 #endif



700 /* Dual-function cryptographic operations */

702 /* C_DigestEncryptUpdate continues a multiple-part digesting
703  * and encryption operation. */
704 CK_PKCS11_FUNCTION_INFO(C_DigestEncryptUpdate)
705 #ifdef CK_NEED_ARG_LIST
706 (
707   CK_SESSION_HANDLE hSession,            /* session's handle */
708   CK_BYTE_PTR       pPart,               /* the plaintext data */
709   CK_ULONG          ulPartLen,           /* plaintext length */
710   CK_BYTE_PTR       pEncryptedPart,      /* gets ciphertext */
711   CK_ULONG_PTR      pulEncryptedPartLen  /* gets c-text length */
712 );
713 #endif


716 /* C_DecryptDigestUpdate continues a multiple-part decryption and
717  * digesting operation. */
718 CK_PKCS11_FUNCTION_INFO(C_DecryptDigestUpdate)
719 #ifdef CK_NEED_ARG_LIST
720 (
721   CK_SESSION_HANDLE hSession,            /* session's handle */
```



```
722   CK_BYTE_PTR       pEncryptedPart,      /* ciphertext */
723   CK_ULONG          ulEncryptedPartLen,  /* ciphertext length */
724   CK_BYTE_PTR       pPart,               /* gets plaintext */
725   CK_ULONG_PTR      pulPartLen           /* gets plaintext len */
726 );
727 #endif


730 /* C_SignEncryptUpdate continues a multiple-part signing and
731  * encryption operation. */
732 CK_PKCS11_FUNCTION_INFO(C_SignEncryptUpdate)
733 #ifdef CK_NEED_ARG_LIST
734 (
735   CK_SESSION_HANDLE hSession,            /* session's handle */
736   CK_BYTE_PTR       pPart,               /* the plaintext data */
737   CK_ULONG          ulPartLen,           /* plaintext length */
738   CK_BYTE_PTR       pEncryptedPart,      /* gets ciphertext */
739   CK_ULONG_PTR      pulEncryptedPartLen  /* gets c-text length */
740 );
741 #endif


744 /* C_DecryptVerifyUpdate continues a multiple-part decryption and
745  * verify operation. */
746 CK_PKCS11_FUNCTION_INFO(C_DecryptVerifyUpdate)
747 #ifdef CK_NEED_ARG_LIST
748 (
749   CK_SESSION_HANDLE hSession,            /* session's handle */
750   CK_BYTE_PTR       pEncryptedPart,      /* ciphertext */
751   CK_ULONG          ulEncryptedPartLen,  /* ciphertext length */
752   CK_BYTE_PTR       pPart,               /* gets plaintext */
753   CK_ULONG_PTR      pulPartLen           /* gets p-text length */
754 );
755 #endif



759 /* Key management */

761 /* C_GenerateKey generates a secret key, creating a new key
762  * object. */
763 CK_PKCS11_FUNCTION_INFO(C_GenerateKey)
764 #ifdef CK_NEED_ARG_LIST
765 (
766   CK_SESSION_HANDLE    hSession,    /* the session's handle */
767   CK_MECHANISM_PTR     pMechanism,  /* key generation mech. */
768   CK_ATTRIBUTE_PTR     pTemplate,   /* template for new key */
769   CK_ULONG             ulCount,     /* # of attrs in template */
770   CK_OBJECT_HANDLE_PTR phKey        /* gets handle of new key */
771 );
772 #endif


775 /* C_GenerateKeyPair generates a public-key/private-key pair,
776  * creating new key objects. */
777 CK_PKCS11_FUNCTION_INFO(C_GenerateKeyPair)
778 #ifdef CK_NEED_ARG_LIST
779 (
780   CK_SESSION_HANDLE    hSession,                    /* session
781                                                     * handle */
782   CK_MECHANISM_PTR     pMechanism,                  /* key-gen
783                                                     * mech. */
784   CK_ATTRIBUTE_PTR     pPublicKeyTemplate,          /* template
785                                                     * for pub.
786                                                     * key */
787   CK_ULONG             ulPublicKeyAttributeCount,   /* # pub.
```

```
788                                                 * attrs. */
789   CK_ATTRIBUTE_PTR     pPrivateKeyTemplate,     /* template
790                                                 * for priv.
791                                                 * key */
792   CK_ULONG             ulPrivateKeyAttributeCount,  /* # priv.
793                                                 * attrs. */
794   CK_OBJECT_HANDLE_PTR phPublicKey,             /* gets pub.
795                                                 * key
796                                                 * handle */
797   CK_OBJECT_HANDLE_PTR phPrivateKey             /* gets
798                                                 * priv. key
799                                                 * handle */
800 );
801 #endif


804 /* C_WrapKey wraps (i.e., encrypts) a key. */
805 CK_PKCS11_FUNCTION_INFO(C_WrapKey)
806 #ifdef CK_NEED_ARG_LIST
807 (
808   CK_SESSION_HANDLE hSession,        /* the session's handle */
809   CK_MECHANISM_PTR  pMechanism,      /* the wrapping mechanism */
810   CK_OBJECT_HANDLE  hWrappingKey,    /* wrapping key */
811   CK_OBJECT_HANDLE  hKey,            /* key to be wrapped */
812   CK_BYTE_PTR       pWrappedKey,     /* gets wrapped key */
813   CK_ULONG_PTR      pulWrappedKeyLen /* gets wrapped key size */
814 );
815 #endif


818 /* C_UnwrapKey unwraps (decrypts) a wrapped key, creating a new
819  * key object. */
820 CK_PKCS11_FUNCTION_INFO(C_UnwrapKey)
821 #ifdef CK_NEED_ARG_LIST
822 (
823   CK_SESSION_HANDLE    hSession,          /* session's handle */
824   CK_MECHANISM_PTR     pMechanism,        /* unwrapping mech. */
825   CK_OBJECT_HANDLE     hUnwrappingKey,    /* unwrapping key */
826   CK_BYTE_PTR          pWrappedKey,       /* the wrapped key */
827   CK_ULONG             ulWrappedKeyLen,   /* wrapped key len */
828   CK_ATTRIBUTE_PTR     pTemplate,         /* new key template */
829   CK_ULONG             ulAttributeCount,  /* template length */
830   CK_OBJECT_HANDLE_PTR phKey              /* gets new handle */
831 );
832 #endif


835 /* C_DeriveKey derives a key from a base key, creating a new key
836  * object. */
837 CK_PKCS11_FUNCTION_INFO(C_DeriveKey)
838 #ifdef CK_NEED_ARG_LIST
839 (
840   CK_SESSION_HANDLE    hSession,          /* session's handle */
841   CK_MECHANISM_PTR     pMechanism,        /* key deriv. mech. */
842   CK_OBJECT_HANDLE     hBaseKey,          /* base key */
843   CK_ATTRIBUTE_PTR     pTemplate,         /* new key template */
844   CK_ULONG             ulAttributeCount,  /* template length */
845   CK_OBJECT_HANDLE_PTR phKey              /* gets new handle */
846 );
847 #endif



851 /* Random number generation */

853 /* C_SeedRandom mixes additional seed material into the token's
```



```
854  * random number generator. */
855 CK_PKCS11_FUNCTION_INFO(C_SeedRandom)
856 #ifdef CK_NEED_ARG_LIST
857 (
858   CK_SESSION_HANDLE hSession,  /* the session's handle */
859   CK_BYTE_PTR       pSeed,     /* the seed material */
860   CK_ULONG          ulSeedLen  /* length of seed material */
861 );
862 #endif


865 /* C_GenerateRandom generates random data. */
866 CK_PKCS11_FUNCTION_INFO(C_GenerateRandom)
867 #ifdef CK_NEED_ARG_LIST
868 (
869   CK_SESSION_HANDLE hSession,    /* the session's handle */
870   CK_BYTE_PTR       RandomData,  /* receives the random data */
871   CK_ULONG          ulRandomLen  /* # of bytes to generate */
872 );
873 #endif



877 /* Parallel function management */

879 /* C_GetFunctionStatus is a legacy function; it obtains an
880  * updated status of a function running in parallel with an
881  * application. */
882 CK_PKCS11_FUNCTION_INFO(C_GetFunctionStatus)
883 #ifdef CK_NEED_ARG_LIST
884 (
885   CK_SESSION_HANDLE hSession  /* the session's handle */
886 );
887 #endif


890 /* C_CancelFunction is a legacy function; it cancels a function
891  * running in parallel. */
892 CK_PKCS11_FUNCTION_INFO(C_CancelFunction)
893 #ifdef CK_NEED_ARG_LIST
894 (
895   CK_SESSION_HANDLE hSession  /* the session's handle */
896 );
897 #endif



901 /* Functions added in for Cryptoki Version 2.01 or later */

903 /* C_WaitForSlotEvent waits for a slot event (token insertion,
904  * removal, etc.) to occur. */
905 CK_PKCS11_FUNCTION_INFO(C_WaitForSlotEvent)
906 #ifdef CK_NEED_ARG_LIST
907 (
908   CK_FLAGS flags,        /* blocking/nonblocking flag */
909   CK_SLOT_ID_PTR pSlot,  /* location that receives the slot ID */
910   CK_VOID_PTR pRserved   /* reserved.  Should be NULL_PTR */
911 );
912 #endif
913 #endif /* ! codereview */
```

```
**********************************************************
   69057 Wed Aug 13 19:51:52 2014
new/usr/src/lib/openssl/include/pkcs11t.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pkcs11t.h include file for PKCS #11. */
/* $Revision: 1.10 $ */

/* License to copy and use this software is granted provided that it is
 * identified as "RSA Security Inc. PKCS #11 Cryptographic Token Interface
 * (Cryptoki)" in all material mentioning or referencing this software.

 * License is also granted to make and use derivative works provided that
 * such works are identified as "derived from the RSA Security Inc. PKCS #11
 * Cryptographic Token Interface (Cryptoki)" in all material mentioning or
 * referencing the derived work.

 * RSA Security Inc. makes no representations concerning either the
 * merchantability of this software or the suitability of this software for
 * any particular purpose. It is provided "as is" without express or implied
 * warranty of any kind.
 */

/* See top of pkcs11.h for information about the macros that
 * must be defined and the structure-packing conventions that
 * must be set before including this file. */

#ifndef _PKCS11T_H_
#define _PKCS11T_H_ 1

#define CRYPTOKI_VERSION_MAJOR 2
#define CRYPTOKI_VERSION_MINOR 20
#define CRYPTOKI_VERSION_AMENDMENT 3

#define CK_TRUE 1
#define CK_FALSE 0

#ifndef CK_DISABLE_TRUE_FALSE
#ifndef FALSE
#define FALSE CK_FALSE
#endif

#ifndef TRUE
#define TRUE CK_TRUE
#endif
#endif

/* an unsigned 8-bit value */
typedef unsigned char     CK_BYTE;

/* an unsigned 8-bit character */
typedef CK_BYTE           CK_CHAR;

/* an 8-bit UTF-8 character */
typedef CK_BYTE           CK_UTF8CHAR;

/* a BYTE-sized Boolean flag */
typedef CK_BYTE           CK_BBOOL;

/* an unsigned value, at least 32 bits long */
typedef unsigned long int CK_ULONG;

/* a signed value, the same size as a CK_ULONG */
/* CK_LONG is new for v2.0 */
typedef long int          CK_LONG;
```



```
/* at least 32 bits; each bit is a Boolean flag */
typedef CK_ULONG          CK_FLAGS;


/* some special values for certain CK_ULONG variables */
#define CK_UNAVAILABLE_INFORMATION (~0UL)
#define CK_EFFECTIVELY_INFINITE    0


typedef CK_BYTE     CK_PTR   CK_BYTE_PTR;
typedef CK_CHAR     CK_PTR   CK_CHAR_PTR;
typedef CK_UTF8CHAR CK_PTR   CK_UTF8CHAR_PTR;
typedef CK_ULONG    CK_PTR   CK_ULONG_PTR;
typedef void        CK_PTR   CK_VOID_PTR;

/* Pointer to a CK_VOID_PTR-- i.e., pointer to pointer to void */
typedef CK_VOID_PTR CK_PTR CK_VOID_PTR_PTR;


/* The following value is always invalid if used as a session */
/* handle or object handle */
#define CK_INVALID_HANDLE 0


typedef struct CK_VERSION {
  CK_BYTE       major;  /* integer portion of version number */
  CK_BYTE       minor;  /* 1/100ths portion of version number */
} CK_VERSION;

typedef CK_VERSION CK_PTR CK_VERSION_PTR;


typedef struct CK_INFO {
  /* manufacturerID and libraryDecription have been changed from
   * CK_CHAR to CK_UTF8CHAR for v2.10 */
  CK_VERSION    cryptokiVersion;     /* Cryptoki interface ver */
  CK_UTF8CHAR   manufacturerID[32];  /* blank padded */
  CK_FLAGS      flags;               /* must be zero */

  /* libraryDescription and libraryVersion are new for v2.0 */
  CK_UTF8CHAR   libraryDescription[32];  /* blank padded */
  CK_VERSION    libraryVersion;          /* version of library */
} CK_INFO;

typedef CK_INFO CK_PTR    CK_INFO_PTR;


/* CK_NOTIFICATION enumerates the types of notifications that
 * Cryptoki provides to an application */
/* CK_NOTIFICATION has been changed from an enum to a CK_ULONG
 * for v2.0 */
typedef CK_ULONG CK_NOTIFICATION;
#define CKN_SURRENDER       0

/* The following notification is new for PKCS #11 v2.20 amendment 3 */
#define CKN_OTP_CHANGED     1


typedef CK_ULONG          CK_SLOT_ID;

typedef CK_SLOT_ID CK_PTR CK_SLOT_ID_PTR;


/* CK_SLOT_INFO provides information about a slot */
typedef struct CK_SLOT_INFO {
  /* slotDescription and manufacturerID have been changed from
```

```
128    * CK_CHAR to CK_UTF8CHAR for v2.10 */
129   CK_UTF8CHAR   slotDescription[64];  /* blank padded */
130   CK_UTF8CHAR   manufacturerID[32];   /* blank padded */
131   CK_FLAGS      flags;

133   /* hardwareVersion and firmwareVersion are new for v2.0 */
134   CK_VERSION    hardwareVersion;  /* version of hardware */
135   CK_VERSION    firmwareVersion;  /* version of firmware */
136 } CK_SLOT_INFO;

138 /* flags: bit flags that provide capabilities of the slot
139  *      Bit Flag              Mask        Meaning
140  */
141 #define CKF_TOKEN_PRESENT     0x00000001  /* a token is there */
142 #define CKF_REMOVABLE_DEVICE  0x00000002  /* removable devices*/
143 #define CKF_HW_SLOT           0x00000004  /* hardware slot */

145 typedef CK_SLOT_INFO CK_PTR CK_SLOT_INFO_PTR;


148 /* CK_TOKEN_INFO provides information about a token */
149 typedef struct CK_TOKEN_INFO {
150   /* label, manufacturerID, and model have been changed from
151    * CK_CHAR to CK_UTF8CHAR for v2.10 */
152   CK_UTF8CHAR   label[32];           /* blank padded */
153   CK_UTF8CHAR   manufacturerID[32];  /* blank padded */
154   CK_UTF8CHAR   model[16];           /* blank padded */
155   CK_CHAR       serialNumber[16];    /* blank padded */
156   CK_FLAGS      flags;               /* see below */

158   /* ulMaxSessionCount, ulSessionCount, ulMaxRwSessionCount,
159    * ulRwSessionCount, ulMaxPinLen, and ulMinPinLen have all been
160    * changed from CK_USHORT to CK_ULONG for v2.0 */
161   CK_ULONG      ulMaxSessionCount;     /* max open sessions */
162   CK_ULONG      ulSessionCount;        /* sess. now open */
163   CK_ULONG      ulMaxRwSessionCount;   /* max R/W sessions */
164   CK_ULONG      ulRwSessionCount;      /* R/W sess. now open */
165   CK_ULONG      ulMaxPinLen;           /* in bytes */
166   CK_ULONG      ulMinPinLen;           /* in bytes */
167   CK_ULONG      ulTotalPublicMemory;   /* in bytes */
168   CK_ULONG      ulFreePublicMemory;    /* in bytes */
169   CK_ULONG      ulTotalPrivateMemory;  /* in bytes */
170   CK_ULONG      ulFreePrivateMemory;   /* in bytes */

172   /* hardwareVersion, firmwareVersion, and time are new for
173    * v2.0 */
174   CK_VERSION    hardwareVersion;       /* version of hardware */
175   CK_VERSION    firmwareVersion;       /* version of firmware */
176   CK_CHAR       utcTime[16];           /* time */
177 } CK_TOKEN_INFO;

179 /* The flags parameter is defined as follows:
180  *      Bit Flag                    Mask        Meaning
181  */
182 #define CKF_RNG                     0x00000001  /* has random #
183                                                  * generator */
184 #define CKF_WRITE_PROTECTED         0x00000002  /* token is
185                                                  * write-
186                                                  * protected */
187 #define CKF_LOGIN_REQUIRED          0x00000004  /* user must
188                                                  * login */
189 #define CKF_USER_PIN_INITIALIZED    0x00000008  /* normal user's
190                                                  * PIN is set */

192 /* CKF_RESTORE_KEY_NOT_NEEDED is new for v2.0.  If it is set,
193  * that means that *every* time the state of cryptographic
```



```
194  * operations of a session is successfully saved, all keys
195  * needed to continue those operations are stored in the state */
196 #define CKF_RESTORE_KEY_NOT_NEEDED  0x00000020

198 /* CKF_CLOCK_ON_TOKEN is new for v2.0.  If it is set, that means
199  * that the token has some sort of clock.  The time on that
200  * clock is returned in the token info structure */
201 #define CKF_CLOCK_ON_TOKEN          0x00000040

203 /* CKF_PROTECTED_AUTHENTICATION_PATH is new for v2.0.  If it is
204  * set, that means that there is some way for the user to login
205  * without sending a PIN through the Cryptoki library itself */
206 #define CKF_PROTECTED_AUTHENTICATION_PATH 0x00000100

208 /* CKF_DUAL_CRYPTO_OPERATIONS is new for v2.0.  If it is true,
209  * that means that a single session with the token can perform
210  * dual simultaneous cryptographic operations (digest and
211  * encrypt; decrypt and digest; sign and encrypt; and decrypt
212  * and sign) */
213 #define CKF_DUAL_CRYPTO_OPERATIONS  0x00000200

215 /* CKF_TOKEN_INITIALIZED if new for v2.10. If it is true, the
216  * token has been initialized using C_InitializeToken or an
217  * equivalent mechanism outside the scope of PKCS #11.
218  * Calling C_InitializeToken when this flag is set will cause
219  * the token to be reinitialized. */
220 #define CKF_TOKEN_INITIALIZED       0x00000400

222 /* CKF_SECONDARY_AUTHENTICATION if new for v2.10. If it is
223  * true, the token supports secondary authentication for
224  * private key objects. This flag is deprecated in v2.11 and
225    onwards. */
226 #define CKF_SECONDARY_AUTHENTICATION  0x00000800

228 /* CKF_USER_PIN_COUNT_LOW if new for v2.10. If it is true, an
229  * incorrect user login PIN has been entered at least once
230  * since the last successful authentication. */
231 #define CKF_USER_PIN_COUNT_LOW       0x00010000

233 /* CKF_USER_PIN_FINAL_TRY if new for v2.10. If it is true,
234  * supplying an incorrect user PIN will it to become locked. */
235 #define CKF_USER_PIN_FINAL_TRY       0x00020000

237 /* CKF_USER_PIN_LOCKED if new for v2.10. If it is true, the
238  * user PIN has been locked. User login to the token is not
239  * possible. */
240 #define CKF_USER_PIN_LOCKED          0x00040000

242 /* CKF_USER_PIN_TO_BE_CHANGED if new for v2.10. If it is true,
243  * the user PIN value is the default value set by token
244  * initialization or manufacturing, or the PIN has been
245  * expired by the card. */
246 #define CKF_USER_PIN_TO_BE_CHANGED   0x00080000

248 /* CKF_SO_PIN_COUNT_LOW if new for v2.10. If it is true, an
249  * incorrect SO login PIN has been entered at least once since
250  * the last successful authentication. */
251 #define CKF_SO_PIN_COUNT_LOW         0x00100000

253 /* CKF_SO_PIN_FINAL_TRY if new for v2.10. If it is true,
254  * supplying an incorrect SO PIN will it to become locked. */
255 #define CKF_SO_PIN_FINAL_TRY         0x00200000

257 /* CKF_SO_PIN_LOCKED if new for v2.10. If it is true, the SO
258  * PIN has been locked. SO login to the token is not possible.
259  */
```

```
260 #define CKF_SO_PIN_LOCKED            0x00400000

262 /* CKF_SO_PIN_TO_BE_CHANGED if new for v2.10. If it is true,
263  * the SO PIN value is the default value set by token
264  * initialization or manufacturing, or the PIN has been
265  * expired by the card. */
266 #define CKF_SO_PIN_TO_BE_CHANGED     0x00800000

268 typedef CK_TOKEN_INFO CK_PTR CK_TOKEN_INFO_PTR;


271 /* CK_SESSION_HANDLE is a Cryptoki-assigned value that
272  * identifies a session */
273 typedef CK_ULONG          CK_SESSION_HANDLE;

275 typedef CK_SESSION_HANDLE CK_PTR CK_SESSION_HANDLE_PTR;


278 /* CK_USER_TYPE enumerates the types of Cryptoki users */
279 /* CK_USER_TYPE has been changed from an enum to a CK_ULONG for
280  * v2.0 */
281 typedef CK_ULONG          CK_USER_TYPE;
282 /* Security Officer */
283 #define CKU_SO    0
284 /* Normal user */
285 #define CKU_USER  1
286 /* Context specific (added in v2.20) */
287 #define CKU_CONTEXT_SPECIFIC   2

289 /* CK_STATE enumerates the session states */
290 /* CK_STATE has been changed from an enum to a CK_ULONG for
291  * v2.0 */
292 typedef CK_ULONG          CK_STATE;
293 #define CKS_RO_PUBLIC_SESSION  0
294 #define CKS_RO_USER_FUNCTIONS  1
295 #define CKS_RW_PUBLIC_SESSION  2
296 #define CKS_RW_USER_FUNCTIONS  3
297 #define CKS_RW_SO_FUNCTIONS    4


300 /* CK_SESSION_INFO provides information about a session */
301 typedef struct CK_SESSION_INFO {
302   CK_SLOT_ID    slotID;
303   CK_STATE      state;
304   CK_FLAGS      flags;          /* see below */

306   /* ulDeviceError was changed from CK_USHORT to CK_ULONG for
307    * v2.0 */
308   CK_ULONG      ulDeviceError;  /* device-dependent error code */
309 } CK_SESSION_INFO;

311 /* The flags are defined in the following table:
312  *      Bit Flag                Mask        Meaning
313  */
314 #define CKF_RW_SESSION          0x00000002  /* session is r/w */
315 #define CKF_SERIAL_SESSION      0x00000004  /* no parallel */

317 typedef CK_SESSION_INFO CK_PTR CK_SESSION_INFO_PTR;


320 /* CK_OBJECT_HANDLE is a token-specific identifier for an
321  * object  */
322 typedef CK_ULONG          CK_OBJECT_HANDLE;

324 typedef CK_OBJECT_HANDLE CK_PTR CK_OBJECT_HANDLE_PTR;
```



```
327 /* CK_OBJECT_CLASS is a value that identifies the classes (or
328  * types) of objects that Cryptoki recognizes.  It is defined
329  * as follows: */
330 /* CK_OBJECT_CLASS was changed from CK_USHORT to CK_ULONG for
331  * v2.0 */
332 typedef CK_ULONG          CK_OBJECT_CLASS;

334 /* The following classes of objects are defined: */
335 /* CKO_HW_FEATURE is new for v2.10 */
336 /* CKO_DOMAIN_PARAMETERS is new for v2.11 */
337 /* CKO_MECHANISM is new for v2.20 */
338 #define CKO_DATA                0x00000000
339 #define CKO_CERTIFICATE         0x00000001
340 #define CKO_PUBLIC_KEY          0x00000002
341 #define CKO_PRIVATE_KEY         0x00000003
342 #define CKO_SECRET_KEY          0x00000004
343 #define CKO_HW_FEATURE          0x00000005
344 #define CKO_DOMAIN_PARAMETERS 0x00000006
345 #define CKO_MECHANISM           0x00000007

347 /* CKO_OTP_KEY is new for PKCS #11 v2.20 amendment 1 */
348 #define CKO_OTP_KEY             0x00000008

350 #define CKO_VENDOR_DEFINED    0x80000000

352 typedef CK_OBJECT_CLASS CK_PTR CK_OBJECT_CLASS_PTR;

354 /* CK_HW_FEATURE_TYPE is new for v2.10. CK_HW_FEATURE_TYPE is a
355  * value that identifies the hardware feature type of an object
356  * with CK_OBJECT_CLASS equal to CKO_HW_FEATURE. */
357 typedef CK_ULONG          CK_HW_FEATURE_TYPE;

359 /* The following hardware feature types are defined */
360 /* CKH_USER_INTERFACE is new for v2.20 */
361 #define CKH_MONOTONIC_COUNTER  0x00000001
362 #define CKH_CLOCK              0x00000002
363 #define CKH_USER_INTERFACE     0x00000003
364 #define CKH_VENDOR_DEFINED     0x80000000

366 /* CK_KEY_TYPE is a value that identifies a key type */
367 /* CK_KEY_TYPE was changed from CK_USHORT to CK_ULONG for v2.0 */
368 typedef CK_ULONG          CK_KEY_TYPE;

370 /* the following key types are defined: */
371 #define CKK_RSA                 0x00000000
372 #define CKK_DSA                 0x00000001
373 #define CKK_DH                  0x00000002

375 /* CKK_ECDSA and CKK_KEA are new for v2.0 */
376 /* CKK_ECDSA is deprecated in v2.11, CKK_EC is preferred. */
377 #define CKK_ECDSA               0x00000003
378 #define CKK_EC                  0x00000003
379 #define CKK_X9_42_DH            0x00000004
380 #define CKK_KEA                 0x00000005

382 #define CKK_GENERIC_SECRET  0x00000010
383 #define CKK_RC2                 0x00000011
384 #define CKK_RC4                 0x00000012
385 #define CKK_DES                 0x00000013
386 #define CKK_DES2                0x00000014
387 #define CKK_DES3                0x00000015

389 /* all these key types are new for v2.0 */
390 #define CKK_CAST                0x00000016
391 #define CKK_CAST3               0x00000017
```

```
/* CKK_CAST5 is deprecated in v2.11, CKK_CAST128 is preferred. */
#define CKK_CAST5           0x00000018
#define CKK_CAST128         0x00000018
#define CKK_RC5             0x00000019
#define CKK_IDEA            0x0000001A
#define CKK_SKIPJACK        0x0000001B
#define CKK_BATON           0x0000001C
#define CKK_JUNIPER         0x0000001D
#define CKK_CDMF            0x0000001E
#define CKK_AES             0x0000001F

/* BlowFish and TwoFish are new for v2.20 */
#define CKK_BLOWFISH        0x00000020
#define CKK_TWOFISH         0x00000021

/* SecurID, HOTP, and ACTI are new for PKCS #11 v2.20 amendment 1 */
#define CKK_SECURID         0x00000022
#define CKK_HOTP            0x00000023
#define CKK_ACTI            0x00000024

/* Camellia is new for PKCS #11 v2.20 amendment 3 */
#define CKK_CAMELLIA                    0x00000025
/* ARIA is new for PKCS #11 v2.20 amendment 3 */
#define CKK_ARIA                        0x00000026


#define CKK_VENDOR_DEFINED  0x80000000


/* CK_CERTIFICATE_TYPE is a value that identifies a certificate
 * type */
/* CK_CERTIFICATE_TYPE was changed from CK_USHORT to CK_ULONG
 * for v2.0 */
typedef CK_ULONG          CK_CERTIFICATE_TYPE;

/* The following certificate types are defined: */
/* CKC_X_509_ATTR_CERT is new for v2.10 */
/* CKC_WTLS is new for v2.20 */
#define CKC_X_509           0x00000000
#define CKC_X_509_ATTR_CERT 0x00000001
#define CKC_WTLS            0x00000002
#define CKC_VENDOR_DEFINED  0x80000000


/* CK_ATTRIBUTE_TYPE is a value that identifies an attribute
 * type */
/* CK_ATTRIBUTE_TYPE was changed from CK_USHORT to CK_ULONG for
 * v2.0 */
typedef CK_ULONG          CK_ATTRIBUTE_TYPE;

/* The CKF_ARRAY_ATTRIBUTE flag identifies an attribute which
   consists of an array of values. */
#define CKF_ARRAY_ATTRIBUTE    0x40000000

/* The following OTP-related defines are new for PKCS #11 v2.20 amendment 1
   and relates to the CKA_OTP_FORMAT attribute */
#define CK_OTP_FORMAT_DECIMAL      0
#define CK_OTP_FORMAT_HEXADECIMAL  1
#define CK_OTP_FORMAT_ALPHANUMERIC 2
#define CK_OTP_FORMAT_BINARY       3

/* The following OTP-related defines are new for PKCS #11 v2.20 amendment 1
   and relates to the CKA_OTP_..._REQUIREMENT attributes */
#define CK_OTP_PARAM_IGNORED       0
#define CK_OTP_PARAM_OPTIONAL      1
#define CK_OTP_PARAM_MANDATORY     2
```



```
/* The following attribute types are defined: */
#define CKA_CLASS               0x00000000
#define CKA_TOKEN               0x00000001
#define CKA_PRIVATE             0x00000002
#define CKA_LABEL               0x00000003
#define CKA_APPLICATION         0x00000010
#define CKA_VALUE               0x00000011

/* CKA_OBJECT_ID is new for v2.10 */
#define CKA_OBJECT_ID           0x00000012

#define CKA_CERTIFICATE_TYPE    0x00000080
#define CKA_ISSUER              0x00000081
#define CKA_SERIAL_NUMBER       0x00000082

/* CKA_AC_ISSUER, CKA_OWNER, and CKA_ATTR_TYPES are new
 * for v2.10 */
#define CKA_AC_ISSUER           0x00000083
#define CKA_OWNER               0x00000084
#define CKA_ATTR_TYPES          0x00000085

/* CKA_TRUSTED is new for v2.11 */
#define CKA_TRUSTED             0x00000086

/* CKA_CERTIFICATE_CATEGORY ...
 * CKA_CHECK_VALUE are new for v2.20 */
#define CKA_CERTIFICATE_CATEGORY        0x00000087
#define CKA_JAVA_MIDP_SECURITY_DOMAIN   0x00000088
#define CKA_URL                         0x00000089
#define CKA_HASH_OF_SUBJECT_PUBLIC_KEY  0x0000008A
#define CKA_HASH_OF_ISSUER_PUBLIC_KEY   0x0000008B
#define CKA_CHECK_VALUE                 0x00000090

#define CKA_KEY_TYPE            0x00000100
#define CKA_SUBJECT             0x00000101
#define CKA_ID                  0x00000102
#define CKA_SENSITIVE           0x00000103
#define CKA_ENCRYPT             0x00000104
#define CKA_DECRYPT             0x00000105
#define CKA_WRAP                0x00000106
#define CKA_UNWRAP              0x00000107
#define CKA_SIGN                0x00000108
#define CKA_SIGN_RECOVER        0x00000109
#define CKA_VERIFY              0x0000010A
#define CKA_VERIFY_RECOVER      0x0000010B
#define CKA_DERIVE              0x0000010C
#define CKA_START_DATE          0x00000110
#define CKA_END_DATE            0x00000111
#define CKA_MODULUS             0x00000120
#define CKA_MODULUS_BITS        0x00000121
#define CKA_PUBLIC_EXPONENT     0x00000122
#define CKA_PRIVATE_EXPONENT    0x00000123
#define CKA_PRIME_1             0x00000124
#define CKA_PRIME_2             0x00000125
#define CKA_EXPONENT_1          0x00000126
#define CKA_EXPONENT_2          0x00000127
#define CKA_COEFFICIENT         0x00000128
#define CKA_PRIME               0x00000130
#define CKA_SUBPRIME            0x00000131
#define CKA_BASE                0x00000132

/* CKA_PRIME_BITS and CKA_SUB_PRIME_BITS are new for v2.11 */
#define CKA_PRIME_BITS          0x00000133
#define CKA_SUBPRIME_BITS       0x00000134
#define CKA_SUB_PRIME_BITS      CKA_SUBPRIME_BITS
```

```
524 /* (To retain backwards-compatibility) */

526 #define CKA_VALUE_BITS          0x00000160
527 #define CKA_VALUE_LEN           0x00000161

529 /* CKA_EXTRACTABLE, CKA_LOCAL, CKA_NEVER_EXTRACTABLE,
530  * CKA_ALWAYS_SENSITIVE, CKA_MODIFIABLE, CKA_ECDSA_PARAMS,
531  * and CKA_EC_POINT are new for v2.0 */
532 #define CKA_EXTRACTABLE         0x00000162
533 #define CKA_LOCAL               0x00000163
534 #define CKA_NEVER_EXTRACTABLE   0x00000164
535 #define CKA_ALWAYS_SENSITIVE    0x00000165

537 /* CKA_KEY_GEN_MECHANISM is new for v2.11 */
538 #define CKA_KEY_GEN_MECHANISM   0x00000166

540 #define CKA_MODIFIABLE          0x00000170

542 /* CKA_ECDSA_PARAMS is deprecated in v2.11,
543  * CKA_EC_PARAMS is preferred. */
544 #define CKA_ECDSA_PARAMS        0x00000180
545 #define CKA_EC_PARAMS           0x00000180

547 #define CKA_EC_POINT            0x00000181

549 /* CKA_SECONDARY_AUTH, CKA_AUTH_PIN_FLAGS,
550  * are new for v2.10. Deprecated in v2.11 and onwards. */
551 #define CKA_SECONDARY_AUTH      0x00000200
552 #define CKA_AUTH_PIN_FLAGS      0x00000201

554 /* CKA_ALWAYS_AUTHENTICATE ...
555  * CKA_UNWRAP_TEMPLATE are new for v2.20 */
556 #define CKA_ALWAYS_AUTHENTICATE  0x00000202

558 #define CKA_WRAP_WITH_TRUSTED    0x00000210
559 #define CKA_WRAP_TEMPLATE        (CKF_ARRAY_ATTRIBUTE|0x00000211)
560 #define CKA_UNWRAP_TEMPLATE      (CKF_ARRAY_ATTRIBUTE|0x00000212)

562 /* CKA_OTP... atttributes are new for PKCS #11 v2.20 amendment 3. */
563 #define CKA_OTP_FORMAT                0x00000220
564 #define CKA_OTP_LENGTH                0x00000221
565 #define CKA_OTP_TIME_INTERVAL         0x00000222
566 #define CKA_OTP_USER_FRIENDLY_MODE    0x00000223
567 #define CKA_OTP_CHALLENGE_REQUIREMENT 0x00000224
568 #define CKA_OTP_TIME_REQUIREMENT      0x00000225
569 #define CKA_OTP_COUNTER_REQUIREMENT   0x00000226
570 #define CKA_OTP_PIN_REQUIREMENT       0x00000227
571 #define CKA_OTP_COUNTER               0x0000022E
572 #define CKA_OTP_TIME                  0x0000022F
573 #define CKA_OTP_USER_IDENTIFIER       0x0000022A
574 #define CKA_OTP_SERVICE_IDENTIFIER    0x0000022B
575 #define CKA_OTP_SERVICE_LOGO          0x0000022C
576 #define CKA_OTP_SERVICE_LOGO_TYPE     0x0000022D


579 /* CKA_HW_FEATURE_TYPE, CKA_RESET_ON_INIT, and CKA_HAS_RESET
580  * are new for v2.10 */
581 #define CKA_HW_FEATURE_TYPE    0x00000300
582 #define CKA_RESET_ON_INIT      0x00000301
583 #define CKA_HAS_RESET          0x00000302

585 /* The following attributes are new for v2.20 */
586 #define CKA_PIXEL_X                    0x00000400
587 #define CKA_PIXEL_Y                    0x00000401
588 #define CKA_RESOLUTION                 0x00000402
589 #define CKA_CHAR_ROWS                  0x00000403
```



```
590 #define CKA_CHAR_COLUMNS               0x00000404
591 #define CKA_COLOR                      0x00000405
592 #define CKA_BITS_PER_PIXEL             0x00000406
593 #define CKA_CHAR_SETS                  0x00000480
594 #define CKA_ENCODING_METHODS           0x00000481
595 #define CKA_MIME_TYPES                 0x00000482
596 #define CKA_MECHANISM_TYPE             0x00000500
597 #define CKA_REQUIRED_CMS_ATTRIBUTES    0x00000501
598 #define CKA_DEFAULT_CMS_ATTRIBUTES     0x00000502
599 #define CKA_SUPPORTED_CMS_ATTRIBUTES   0x00000503
600 #define CKA_ALLOWED_MECHANISMS         (CKF_ARRAY_ATTRIBUTE|0x00000600)

602 #define CKA_VENDOR_DEFINED     0x80000000

604 /* CK_ATTRIBUTE is a structure that includes the type, length
605  * and value of an attribute */
606 typedef struct CK_ATTRIBUTE {
607   CK_ATTRIBUTE_TYPE type;
608   CK_VOID_PTR       pValue;

610   /* ulValueLen went from CK_USHORT to CK_ULONG for v2.0 */
611   CK_ULONG          ulValueLen;  /* in bytes */
612 } CK_ATTRIBUTE;

614 typedef CK_ATTRIBUTE CK_PTR CK_ATTRIBUTE_PTR;


617 /* CK_DATE is a structure that defines a date */
618 typedef struct CK_DATE{
619   CK_CHAR       year[4];   /* the year ("1900" - "9999") */
620   CK_CHAR       month[2];  /* the month ("01" - "12") */
621   CK_CHAR       day[2];    /* the day   ("01" - "31") */
622 } CK_DATE;


625 /* CK_MECHANISM_TYPE is a value that identifies a mechanism
626  * type */
627 /* CK_MECHANISM_TYPE was changed from CK_USHORT to CK_ULONG for
628  * v2.0 */
629 typedef CK_ULONG          CK_MECHANISM_TYPE;

631 /* the following mechanism types are defined: */
632 #define CKM_RSA_PKCS_KEY_PAIR_GEN      0x00000000
633 #define CKM_RSA_PKCS                   0x00000001
634 #define CKM_RSA_9796                   0x00000002
635 #define CKM_RSA_X_509                  0x00000003

637 /* CKM_MD2_RSA_PKCS, CKM_MD5_RSA_PKCS, and CKM_SHA1_RSA_PKCS
638  * are new for v2.0.  They are mechanisms which hash and sign */
639 #define CKM_MD2_RSA_PKCS               0x00000004
640 #define CKM_MD5_RSA_PKCS               0x00000005
641 #define CKM_SHA1_RSA_PKCS              0x00000006

643 /* CKM_RIPEMD128_RSA_PKCS, CKM_RIPEMD160_RSA_PKCS, and
644  * CKM_RSA_PKCS_OAEP are new for v2.10 */
645 #define CKM_RIPEMD128_RSA_PKCS         0x00000007
646 #define CKM_RIPEMD160_RSA_PKCS         0x00000008
647 #define CKM_RSA_PKCS_OAEP              0x00000009

649 /* CKM_RSA_X9_31_KEY_PAIR_GEN, CKM_RSA_X9_31, CKM_SHA1_RSA_X9_31,
650  * CKM_RSA_PKCS_PSS, and CKM_SHA1_RSA_PKCS_PSS are new for v2.11 */
651 #define CKM_RSA_X9_31_KEY_PAIR_GEN     0x0000000A
652 #define CKM_RSA_X9_31                  0x0000000B
653 #define CKM_SHA1_RSA_X9_31             0x0000000C
654 #define CKM_RSA_PKCS_PSS               0x0000000D
655 #define CKM_SHA1_RSA_PKCS_PSS          0x0000000E
```

```
#define CKM_DSA_KEY_PAIR_GEN           0x00000010
#define CKM_DSA                        0x00000011
#define CKM_DSA_SHA1                   0x00000012
#define CKM_DH_PKCS_KEY_PAIR_GEN       0x00000020
#define CKM_DH_PKCS_DERIVE             0x00000021

/* CKM_X9_42_DH_KEY_PAIR_GEN, CKM_X9_42_DH_DERIVE,
 * CKM_X9_42_DH_HYBRID_DERIVE, and CKM_X9_42_MQV_DERIVE are new for
 * v2.11 */
#define CKM_X9_42_DH_KEY_PAIR_GEN      0x00000030
#define CKM_X9_42_DH_DERIVE            0x00000031
#define CKM_X9_42_DH_HYBRID_DERIVE     0x00000032
#define CKM_X9_42_MQV_DERIVE           0x00000033

/* CKM_SHA256/384/512 are new for v2.20 */
#define CKM_SHA256_RSA_PKCS            0x00000040
#define CKM_SHA384_RSA_PKCS            0x00000041
#define CKM_SHA512_RSA_PKCS            0x00000042
#define CKM_SHA256_RSA_PKCS_PSS        0x00000043
#define CKM_SHA384_RSA_PKCS_PSS        0x00000044
#define CKM_SHA512_RSA_PKCS_PSS        0x00000045

/* SHA-224 RSA mechanisms are new for PKCS #11 v2.20 amendment 3 */
#define CKM_SHA224_RSA_PKCS            0x00000046
#define CKM_SHA224_RSA_PKCS_PSS        0x00000047

#define CKM_RC2_KEY_GEN                0x00000100
#define CKM_RC2_ECB                    0x00000101
#define CKM_RC2_CBC                    0x00000102
#define CKM_RC2_MAC                    0x00000103

/* CKM_RC2_MAC_GENERAL and CKM_RC2_CBC_PAD are new for v2.0 */
#define CKM_RC2_MAC_GENERAL            0x00000104
#define CKM_RC2_CBC_PAD                0x00000105

#define CKM_RC4_KEY_GEN                0x00000110
#define CKM_RC4                        0x00000111
#define CKM_DES_KEY_GEN                0x00000120
#define CKM_DES_ECB                    0x00000121
#define CKM_DES_CBC                    0x00000122
#define CKM_DES_MAC                    0x00000123

/* CKM_DES_MAC_GENERAL and CKM_DES_CBC_PAD are new for v2.0 */
#define CKM_DES_MAC_GENERAL            0x00000124
#define CKM_DES_CBC_PAD                0x00000125

#define CKM_DES2_KEY_GEN               0x00000130
#define CKM_DES3_KEY_GEN               0x00000131
#define CKM_DES3_ECB                   0x00000132
#define CKM_DES3_CBC                   0x00000133
#define CKM_DES3_MAC                   0x00000134

/* CKM_DES3_MAC_GENERAL, CKM_DES3_CBC_PAD, CKM_CDMF_KEY_GEN,
 * CKM_CDMF_ECB, CKM_CDMF_CBC, CKM_CDMF_MAC,
 * CKM_CDMF_MAC_GENERAL, and CKM_CDMF_CBC_PAD are new for v2.0 */
#define CKM_DES3_MAC_GENERAL           0x00000135
#define CKM_DES3_CBC_PAD               0x00000136
#define CKM_CDMF_KEY_GEN               0x00000140
#define CKM_CDMF_ECB                   0x00000141
#define CKM_CDMF_CBC                   0x00000142
#define CKM_CDMF_MAC                   0x00000143
#define CKM_CDMF_MAC_GENERAL           0x00000144
#define CKM_CDMF_CBC_PAD               0x00000145

/* the following four DES mechanisms are new for v2.20 */
```



```
#define CKM_DES_OFB64                  0x00000150
#define CKM_DES_OFB8                   0x00000151
#define CKM_DES_CFB64                  0x00000152
#define CKM_DES_CFB8                   0x00000153

#define CKM_MD2                        0x00000200

/* CKM_MD2_HMAC and CKM_MD2_HMAC_GENERAL are new for v2.0 */
#define CKM_MD2_HMAC                   0x00000201
#define CKM_MD2_HMAC_GENERAL           0x00000202

#define CKM_MD5                        0x00000210

/* CKM_MD5_HMAC and CKM_MD5_HMAC_GENERAL are new for v2.0 */
#define CKM_MD5_HMAC                   0x00000211
#define CKM_MD5_HMAC_GENERAL           0x00000212

#define CKM_SHA_1                      0x00000220

/* CKM_SHA_1_HMAC and CKM_SHA_1_HMAC_GENERAL are new for v2.0 */
#define CKM_SHA_1_HMAC                 0x00000221
#define CKM_SHA_1_HMAC_GENERAL         0x00000222

/* CKM_RIPEMD128, CKM_RIPEMD128_HMAC,
 * CKM_RIPEMD128_HMAC_GENERAL, CKM_RIPEMD160, CKM_RIPEMD160_HMAC,
 * and CKM_RIPEMD160_HMAC_GENERAL are new for v2.10 */
#define CKM_RIPEMD128                  0x00000230
#define CKM_RIPEMD128_HMAC             0x00000231
#define CKM_RIPEMD128_HMAC_GENERAL     0x00000232
#define CKM_RIPEMD160                  0x00000240
#define CKM_RIPEMD160_HMAC             0x00000241
#define CKM_RIPEMD160_HMAC_GENERAL     0x00000242

/* CKM_SHA256/384/512 are new for v2.20 */
#define CKM_SHA256                     0x00000250
#define CKM_SHA256_HMAC                0x00000251
#define CKM_SHA256_HMAC_GENERAL        0x00000252

/* SHA-224 is new for PKCS #11 v2.20 amendment 3 */
#define CKM_SHA224                     0x00000255
#define CKM_SHA224_HMAC                0x00000256
#define CKM_SHA224_HMAC_GENERAL        0x00000257

#define CKM_SHA384                     0x00000260
#define CKM_SHA384_HMAC                0x00000261
#define CKM_SHA384_HMAC_GENERAL        0x00000262
#define CKM_SHA512                     0x00000270
#define CKM_SHA512_HMAC                0x00000271
#define CKM_SHA512_HMAC_GENERAL        0x00000272

/* SecurID is new for PKCS #11 v2.20 amendment 1 */
#define CKM_SECURID_KEY_GEN            0x00000280
#define CKM_SECURID                    0x00000282

/* HOTP is new for PKCS #11 v2.20 amendment 1 */
#define CKM_HOTP_KEY_GEN    0x00000290
#define CKM_HOTP            0x00000291

/* ACTI is new for PKCS #11 v2.20 amendment 1 */
#define CKM_ACTI            0x000002A0
#define CKM_ACTI_KEY_GEN    0x000002A1

/* All of the following mechanisms are new for v2.0 */
/* Note that CAST128 and CAST5 are the same algorithm */
#define CKM_CAST_KEY_GEN               0x00000300
#define CKM_CAST_ECB                   0x00000301
```

```
788 #define CKM_CAST_CBC                    0x00000302
789 #define CKM_CAST_MAC                    0x00000303
790 #define CKM_CAST_MAC_GENERAL            0x00000304
791 #define CKM_CAST_CBC_PAD                0x00000305
792 #define CKM_CAST3_KEY_GEN               0x00000310
793 #define CKM_CAST3_ECB                   0x00000311
794 #define CKM_CAST3_CBC                   0x00000312
795 #define CKM_CAST3_MAC                   0x00000313
796 #define CKM_CAST3_MAC_GENERAL           0x00000314
797 #define CKM_CAST3_CBC_PAD               0x00000315
798 #define CKM_CAST5_KEY_GEN               0x00000320
799 #define CKM_CAST128_KEY_GEN             0x00000320
800 #define CKM_CAST5_ECB                   0x00000321
801 #define CKM_CAST128_ECB                 0x00000321
802 #define CKM_CAST5_CBC                   0x00000322
803 #define CKM_CAST128_CBC                 0x00000322
804 #define CKM_CAST5_MAC                   0x00000323
805 #define CKM_CAST128_MAC                 0x00000323
806 #define CKM_CAST5_MAC_GENERAL           0x00000324
807 #define CKM_CAST128_MAC_GENERAL         0x00000324
808 #define CKM_CAST5_CBC_PAD               0x00000325
809 #define CKM_CAST128_CBC_PAD             0x00000325
810 #define CKM_RC5_KEY_GEN                 0x00000330
811 #define CKM_RC5_ECB                     0x00000331
812 #define CKM_RC5_CBC                     0x00000332
813 #define CKM_RC5_MAC                     0x00000333
814 #define CKM_RC5_MAC_GENERAL             0x00000334
815 #define CKM_RC5_CBC_PAD                 0x00000335
816 #define CKM_IDEA_KEY_GEN                0x00000340
817 #define CKM_IDEA_ECB                    0x00000341
818 #define CKM_IDEA_CBC                    0x00000342
819 #define CKM_IDEA_MAC                    0x00000343
820 #define CKM_IDEA_MAC_GENERAL            0x00000344
821 #define CKM_IDEA_CBC_PAD                0x00000345
822 #define CKM_GENERIC_SECRET_KEY_GEN      0x00000350
823 #define CKM_CONCATENATE_BASE_AND_KEY    0x00000360
824 #define CKM_CONCATENATE_BASE_AND_DATA   0x00000362
825 #define CKM_CONCATENATE_DATA_AND_BASE   0x00000363
826 #define CKM_XOR_BASE_AND_DATA           0x00000364
827 #define CKM_EXTRACT_KEY_FROM_KEY        0x00000365
828 #define CKM_SSL3_PRE_MASTER_KEY_GEN     0x00000370
829 #define CKM_SSL3_MASTER_KEY_DERIVE      0x00000371
830 #define CKM_SSL3_KEY_AND_MAC_DERIVE     0x00000372

832 /* CKM_SSL3_MASTER_KEY_DERIVE_DH, CKM_TLS_PRE_MASTER_KEY_GEN,
833  * CKM_TLS_MASTER_KEY_DERIVE, CKM_TLS_KEY_AND_MAC_DERIVE, and
834  * CKM_TLS_MASTER_KEY_DERIVE_DH are new for v2.11 */
835 #define CKM_SSL3_MASTER_KEY_DERIVE_DH   0x00000373
836 #define CKM_TLS_PRE_MASTER_KEY_GEN      0x00000374
837 #define CKM_TLS_MASTER_KEY_DERIVE       0x00000375
838 #define CKM_TLS_KEY_AND_MAC_DERIVE      0x00000376
839 #define CKM_TLS_MASTER_KEY_DERIVE_DH    0x00000377

841 /* CKM_TLS_PRF is new for v2.20 */
842 #define CKM_TLS_PRF                     0x00000378

844 #define CKM_SSL3_MD5_MAC                0x00000380
845 #define CKM_SSL3_SHA1_MAC               0x00000381
846 #define CKM_MD5_KEY_DERIVATION          0x00000390
847 #define CKM_MD2_KEY_DERIVATION          0x00000391
848 #define CKM_SHA1_KEY_DERIVATION         0x00000392

850 /* CKM_SHA256/384/512 are new for v2.20 */
851 #define CKM_SHA256_KEY_DERIVATION       0x00000393
852 #define CKM_SHA384_KEY_DERIVATION       0x00000394
853 #define CKM_SHA512_KEY_DERIVATION       0x00000395
```



```
855 /* SHA-224 key derivation is new for PKCS #11 v2.20 amendment 3 */
856 #define CKM_SHA224_KEY_DERIVATION       0x00000396

858 #define CKM_PBE_MD2_DES_CBC             0x000003A0
859 #define CKM_PBE_MD5_DES_CBC             0x000003A1
860 #define CKM_PBE_MD5_CAST_CBC            0x000003A2
861 #define CKM_PBE_MD5_CAST3_CBC           0x000003A3
862 #define CKM_PBE_MD5_CAST5_CBC           0x000003A4
863 #define CKM_PBE_MD5_CAST128_CBC         0x000003A4
864 #define CKM_PBE_SHA1_CAST5_CBC          0x000003A5
865 #define CKM_PBE_SHA1_CAST128_CBC        0x000003A5
866 #define CKM_PBE_SHA1_RC4_128            0x000003A6
867 #define CKM_PBE_SHA1_RC4_40             0x000003A7
868 #define CKM_PBE_SHA1_DES3_EDE_CBC       0x000003A8
869 #define CKM_PBE_SHA1_DES2_EDE_CBC       0x000003A9
870 #define CKM_PBE_SHA1_RC2_128_CBC        0x000003AA
871 #define CKM_PBE_SHA1_RC2_40_CBC         0x000003AB

873 /* CKM_PKCS5_PBKD2 is new for v2.10 */
874 #define CKM_PKCS5_PBKD2                 0x000003B0

876 #define CKM_PBA_SHA1_WITH_SHA1_HMAC     0x000003C0

878 /* WTLS mechanisms are new for v2.20 */
879 #define CKM_WTLS_PRE_MASTER_KEY_GEN         0x000003D0
880 #define CKM_WTLS_MASTER_KEY_DERIVE          0x000003D1
881 #define CKM_WTLS_MASTER_KEY_DERIVE_DH_ECC   0x000003D2
882 #define CKM_WTLS_PRF                        0x000003D3
883 #define CKM_WTLS_SERVER_KEY_AND_MAC_DERIVE  0x000003D4
884 #define CKM_WTLS_CLIENT_KEY_AND_MAC_DERIVE  0x000003D5

886 #define CKM_KEY_WRAP_LYNKS              0x00000400
887 #define CKM_KEY_WRAP_SET_OAEP           0x00000401

889 /* CKM_CMS_SIG is new for v2.20 */
890 #define CKM_CMS_SIG                     0x00000500

892 /* CKM_KIP mechanisms are new for PKCS #11 v2.20 amendment 2 */
893 #define CKM_KIP_DERIVE                  0x00000510
894 #define CKM_KIP_WRAP                    0x00000511
895 #define CKM_KIP_MAC                     0x00000512

897 /* Camellia is new for PKCS #11 v2.20 amendment 3 */
898 #define CKM_CAMELLIA_KEY_GEN            0x00000550
899 #define CKM_CAMELLIA_ECB                0x00000551
900 #define CKM_CAMELLIA_CBC                0x00000552
901 #define CKM_CAMELLIA_MAC                0x00000553
902 #define CKM_CAMELLIA_MAC_GENERAL        0x00000554
903 #define CKM_CAMELLIA_CBC_PAD            0x00000555
904 #define CKM_CAMELLIA_ECB_ENCRYPT_DATA   0x00000556
905 #define CKM_CAMELLIA_CBC_ENCRYPT_DATA   0x00000557
906 #define CKM_CAMELLIA_CTR                0x00000558

908 /* ARIA is new for PKCS #11 v2.20 amendment 3 */
909 #define CKM_ARIA_KEY_GEN                0x00000560
910 #define CKM_ARIA_ECB                    0x00000561
911 #define CKM_ARIA_CBC                    0x00000562
912 #define CKM_ARIA_MAC                    0x00000563
913 #define CKM_ARIA_MAC_GENERAL            0x00000564
914 #define CKM_ARIA_CBC_PAD                0x00000565
915 #define CKM_ARIA_ECB_ENCRYPT_DATA       0x00000566
916 #define CKM_ARIA_CBC_ENCRYPT_DATA       0x00000567

918 /* Fortezza mechanisms */
919 #define CKM_SKIPJACK_KEY_GEN            0x00001000
```

```
#define CKM_SKIPJACK_ECB64              0x00001001
#define CKM_SKIPJACK_CBC64              0x00001002
#define CKM_SKIPJACK_OFB64              0x00001003
#define CKM_SKIPJACK_CFB64              0x00001004
#define CKM_SKIPJACK_CFB32              0x00001005
#define CKM_SKIPJACK_CFB16              0x00001006
#define CKM_SKIPJACK_CFB8               0x00001007
#define CKM_SKIPJACK_WRAP               0x00001008
#define CKM_SKIPJACK_PRIVATE_WRAP       0x00001009
#define CKM_SKIPJACK_RELAYX             0x0000100a
#define CKM_KEA_KEY_PAIR_GEN            0x00001010
#define CKM_KEA_KEY_DERIVE              0x00001011
#define CKM_FORTEZZA_TIMESTAMP          0x00001020
#define CKM_BATON_KEY_GEN               0x00001030
#define CKM_BATON_ECB128                0x00001031
#define CKM_BATON_ECB96                 0x00001032
#define CKM_BATON_CBC128                0x00001033
#define CKM_BATON_COUNTER               0x00001034
#define CKM_BATON_SHUFFLE               0x00001035
#define CKM_BATON_WRAP                  0x00001036

/* CKM_ECDSA_KEY_PAIR_GEN is deprecated in v2.11,
 * CKM_EC_KEY_PAIR_GEN is preferred */
#define CKM_ECDSA_KEY_PAIR_GEN          0x00001040
#define CKM_EC_KEY_PAIR_GEN             0x00001040

#define CKM_ECDSA                       0x00001041
#define CKM_ECDSA_SHA1                  0x00001042

/* CKM_ECDH1_DERIVE, CKM_ECDH1_COFACTOR_DERIVE, and CKM_ECMQV_DERIVE
 * are new for v2.11 */
#define CKM_ECDH1_DERIVE                0x00001050
#define CKM_ECDH1_COFACTOR_DERIVE       0x00001051
#define CKM_ECMQV_DERIVE                0x00001052

#define CKM_JUNIPER_KEY_GEN             0x00001060
#define CKM_JUNIPER_ECB128              0x00001061
#define CKM_JUNIPER_CBC128              0x00001062
#define CKM_JUNIPER_COUNTER             0x00001063
#define CKM_JUNIPER_SHUFFLE             0x00001064
#define CKM_JUNIPER_WRAP                0x00001065
#define CKM_FASTHASH                    0x00001070

/* CKM_AES_KEY_GEN, CKM_AES_ECB, CKM_AES_CBC, CKM_AES_MAC,
 * CKM_AES_MAC_GENERAL, CKM_AES_CBC_PAD, CKM_DSA_PARAMETER_GEN,
 * CKM_DH_PKCS_PARAMETER_GEN, and CKM_X9_42_DH_PARAMETER_GEN are
 * new for v2.11 */
#define CKM_AES_KEY_GEN                 0x00001080
#define CKM_AES_ECB                     0x00001081
#define CKM_AES_CBC                     0x00001082
#define CKM_AES_MAC                     0x00001083
#define CKM_AES_MAC_GENERAL             0x00001084
#define CKM_AES_CBC_PAD                 0x00001085

/* AES counter mode is new for PKCS #11 v2.20 amendment 3 */
#define CKM_AES_CTR                     0x00001086

/* BlowFish and TwoFish are new for v2.20 */
#define CKM_BLOWFISH_KEY_GEN            0x00001090
#define CKM_BLOWFISH_CBC                0x00001091
#define CKM_TWOFISH_KEY_GEN             0x00001092
#define CKM_TWOFISH_CBC                 0x00001093


/* CKM_xxx_ENCRYPT_DATA mechanisms are new for v2.20 */
#define CKM_DES_ECB_ENCRYPT_DATA        0x00001100
```



```
#define CKM_DES_CBC_ENCRYPT_DATA        0x00001101
#define CKM_DES3_ECB_ENCRYPT_DATA       0x00001102
#define CKM_DES3_CBC_ENCRYPT_DATA       0x00001103
#define CKM_AES_ECB_ENCRYPT_DATA        0x00001104
#define CKM_AES_CBC_ENCRYPT_DATA        0x00001105

#define CKM_DSA_PARAMETER_GEN           0x00002000
#define CKM_DH_PKCS_PARAMETER_GEN       0x00002001
#define CKM_X9_42_DH_PARAMETER_GEN      0x00002002

#define CKM_VENDOR_DEFINED              0x80000000

typedef CK_MECHANISM_TYPE CK_PTR CK_MECHANISM_TYPE_PTR;


/* CK_MECHANISM is a structure that specifies a particular
 * mechanism  */
typedef struct CK_MECHANISM {
  CK_MECHANISM_TYPE mechanism;
  CK_VOID_PTR       pParameter;

  /* ulParameterLen was changed from CK_USHORT to CK_ULONG for
   * v2.0 */
  CK_ULONG          ulParameterLen;  /* in bytes */
} CK_MECHANISM;

typedef CK_MECHANISM CK_PTR CK_MECHANISM_PTR;


/* CK_MECHANISM_INFO provides information about a particular
 * mechanism */
typedef struct CK_MECHANISM_INFO {
    CK_ULONG    ulMinKeySize;
    CK_ULONG    ulMaxKeySize;
    CK_FLAGS    flags;
} CK_MECHANISM_INFO;

/* The flags are defined as follows:
 *      Bit Flag               Mask        Meaning */
#define CKF_HW                  0x00000001  /* performed by HW */

/* The flags CKF_ENCRYPT, CKF_DECRYPT, CKF_DIGEST, CKF_SIGN,
 * CKG_SIGN_RECOVER, CKF_VERIFY, CKF_VERIFY_RECOVER,
 * CKF_GENERATE, CKF_GENERATE_KEY_PAIR, CKF_WRAP, CKF_UNWRAP,
 * and CKF_DERIVE are new for v2.0.  They specify whether or not
 * a mechanism can be used for a particular task */
#define CKF_ENCRYPT             0x00000100
#define CKF_DECRYPT             0x00000200
#define CKF_DIGEST              0x00000400
#define CKF_SIGN                0x00000800
#define CKF_SIGN_RECOVER        0x00001000
#define CKF_VERIFY              0x00002000
#define CKF_VERIFY_RECOVER      0x00004000
#define CKF_GENERATE            0x00008000
#define CKF_GENERATE_KEY_PAIR   0x00010000
#define CKF_WRAP                0x00020000
#define CKF_UNWRAP              0x00040000
#define CKF_DERIVE              0x00080000

/* CKF_EC_F_P, CKF_EC_F_2M, CKF_EC_ECPARAMETERS, CKF_EC_NAMEDCURVE,
 * CKF_EC_UNCOMPRESS, and CKF_EC_COMPRESS are new for v2.11. They
 * describe a token's EC capabilities not available in mechanism
 * information. */
#define CKF_EC_F_P              0x00100000
#define CKF_EC_F_2M             0x00200000
#define CKF_EC_ECPARAMETERS     0x00400000
```

```
#define CKF_EC_NAMEDCURVE      0x00800000
#define CKF_EC_UNCOMPRESS      0x01000000
#define CKF_EC_COMPRESS        0x02000000

#define CKF_EXTENSION          0x80000000 /* FALSE for this version */

typedef CK_MECHANISM_INFO CK_PTR CK_MECHANISM_INFO_PTR;


/* CK_RV is a value that identifies the return value of a
 * Cryptoki function */
/* CK_RV was changed from CK_USHORT to CK_ULONG for v2.0 */
typedef CK_ULONG          CK_RV;

#define CKR_OK                                 0x00000000
#define CKR_CANCEL                             0x00000001
#define CKR_HOST_MEMORY                        0x00000002
#define CKR_SLOT_ID_INVALID                    0x00000003

/* CKR_FLAGS_INVALID was removed for v2.0 */

/* CKR_GENERAL_ERROR and CKR_FUNCTION_FAILED are new for v2.0 */
#define CKR_GENERAL_ERROR                      0x00000005
#define CKR_FUNCTION_FAILED                    0x00000006

/* CKR_ARGUMENTS_BAD, CKR_NO_EVENT, CKR_NEED_TO_CREATE_THREADS,
 * and CKR_CANT_LOCK are new for v2.01 */
#define CKR_ARGUMENTS_BAD                      0x00000007
#define CKR_NO_EVENT                           0x00000008
#define CKR_NEED_TO_CREATE_THREADS             0x00000009
#define CKR_CANT_LOCK                          0x0000000A

#define CKR_ATTRIBUTE_READ_ONLY                0x00000010
#define CKR_ATTRIBUTE_SENSITIVE                0x00000011
#define CKR_ATTRIBUTE_TYPE_INVALID             0x00000012
#define CKR_ATTRIBUTE_VALUE_INVALID            0x00000013
#define CKR_DATA_INVALID                       0x00000020
#define CKR_DATA_LEN_RANGE                     0x00000021
#define CKR_DEVICE_ERROR                       0x00000030
#define CKR_DEVICE_MEMORY                      0x00000031
#define CKR_DEVICE_REMOVED                     0x00000032
#define CKR_ENCRYPTED_DATA_INVALID             0x00000040
#define CKR_ENCRYPTED_DATA_LEN_RANGE           0x00000041
#define CKR_FUNCTION_CANCELED                  0x00000050
#define CKR_FUNCTION_NOT_PARALLEL              0x00000051

/* CKR_FUNCTION_NOT_SUPPORTED is new for v2.0 */
#define CKR_FUNCTION_NOT_SUPPORTED             0x00000054

#define CKR_KEY_HANDLE_INVALID                 0x00000060

/* CKR_KEY_SENSITIVE was removed for v2.0 */

#define CKR_KEY_SIZE_RANGE                     0x00000062
#define CKR_KEY_TYPE_INCONSISTENT              0x00000063

/* CKR_KEY_NOT_NEEDED, CKR_KEY_CHANGED, CKR_KEY_NEEDED,
 * CKR_KEY_INDIGESTIBLE, CKR_KEY_FUNCTION_NOT_PERMITTED,
 * CKR_KEY_NOT_WRAPPABLE, and CKR_KEY_UNEXTRACTABLE are new for
 * v2.0 */
#define CKR_KEY_NOT_NEEDED                     0x00000064
#define CKR_KEY_CHANGED                        0x00000065
#define CKR_KEY_NEEDED                         0x00000066
#define CKR_KEY_INDIGESTIBLE                   0x00000067
#define CKR_KEY_FUNCTION_NOT_PERMITTED         0x00000068
#define CKR_KEY_NOT_WRAPPABLE                  0x00000069
```



```
#define CKR_KEY_UNEXTRACTABLE                  0x0000006A

#define CKR_MECHANISM_INVALID                  0x00000070
#define CKR_MECHANISM_PARAM_INVALID            0x00000071

/* CKR_OBJECT_CLASS_INCONSISTENT and CKR_OBJECT_CLASS_INVALID
 * were removed for v2.0 */
#define CKR_OBJECT_HANDLE_INVALID              0x00000082
#define CKR_OPERATION_ACTIVE                   0x00000090
#define CKR_OPERATION_NOT_INITIALIZED          0x00000091
#define CKR_PIN_INCORRECT                      0x000000A0
#define CKR_PIN_INVALID                        0x000000A1
#define CKR_PIN_LEN_RANGE                      0x000000A2

/* CKR_PIN_EXPIRED and CKR_PIN_LOCKED are new for v2.0 */
#define CKR_PIN_EXPIRED                        0x000000A3
#define CKR_PIN_LOCKED                         0x000000A4

#define CKR_SESSION_CLOSED                     0x000000B0
#define CKR_SESSION_COUNT                      0x000000B1
#define CKR_SESSION_HANDLE_INVALID             0x000000B3
#define CKR_SESSION_PARALLEL_NOT_SUPPORTED     0x000000B4
#define CKR_SESSION_READ_ONLY                  0x000000B5
#define CKR_SESSION_EXISTS                     0x000000B6

/* CKR_SESSION_READ_ONLY_EXISTS and
 * CKR_SESSION_READ_WRITE_SO_EXISTS are new for v2.0 */
#define CKR_SESSION_READ_ONLY_EXISTS           0x000000B7
#define CKR_SESSION_READ_WRITE_SO_EXISTS       0x000000B8

#define CKR_SIGNATURE_INVALID                  0x000000C0
#define CKR_SIGNATURE_LEN_RANGE                0x000000C1
#define CKR_TEMPLATE_INCOMPLETE                0x000000D0
#define CKR_TEMPLATE_INCONSISTENT              0x000000D1
#define CKR_TOKEN_NOT_PRESENT                  0x000000E0
#define CKR_TOKEN_NOT_RECOGNIZED               0x000000E1
#define CKR_TOKEN_WRITE_PROTECTED              0x000000E2
#define CKR_UNWRAPPING_KEY_HANDLE_INVALID      0x000000F0
#define CKR_UNWRAPPING_KEY_SIZE_RANGE          0x000000F1
#define CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT   0x000000F2
#define CKR_USER_ALREADY_LOGGED_IN             0x00000100
#define CKR_USER_NOT_LOGGED_IN                 0x00000101
#define CKR_USER_PIN_NOT_INITIALIZED           0x00000102
#define CKR_USER_TYPE_INVALID                  0x00000103

/* CKR_USER_ANOTHER_ALREADY_LOGGED_IN and CKR_USER_TOO_MANY_TYPES
 * are new to v2.01 */
#define CKR_USER_ANOTHER_ALREADY_LOGGED_IN     0x00000104
#define CKR_USER_TOO_MANY_TYPES                0x00000105

#define CKR_WRAPPED_KEY_INVALID                0x00000110
#define CKR_WRAPPED_KEY_LEN_RANGE              0x00000112
#define CKR_WRAPPING_KEY_HANDLE_INVALID        0x00000113
#define CKR_WRAPPING_KEY_SIZE_RANGE            0x00000114
#define CKR_WRAPPING_KEY_TYPE_INCONSISTENT     0x00000115
#define CKR_RANDOM_SEED_NOT_SUPPORTED          0x00000120

/* These are new to v2.0 */
#define CKR_RANDOM_NO_RNG                      0x00000121

/* These are new to v2.11 */
#define CKR_DOMAIN_PARAMS_INVALID              0x00000130

/* These are new to v2.0 */
#define CKR_BUFFER_TOO_SMALL                   0x00000150
#define CKR_SAVED_STATE_INVALID                0x00000160
```

```
#define CKR_INFORMATION_SENSITIVE              0x00000170
#define CKR_STATE_UNSAVEABLE                   0x00000180

/* These are new to v2.01 */
#define CKR_CRYPTOKI_NOT_INITIALIZED           0x00000190
#define CKR_CRYPTOKI_ALREADY_INITIALIZED       0x00000191
#define CKR_MUTEX_BAD                          0x000001A0
#define CKR_MUTEX_NOT_LOCKED                   0x000001A1

/* The following return values are new for PKCS #11 v2.20 amendment 3 */
#define CKR_NEW_PIN_MODE                       0x000001B0
#define CKR_NEXT_OTP                           0x000001B1

/* This is new to v2.20 */
#define CKR_FUNCTION_REJECTED                  0x00000200

#define CKR_VENDOR_DEFINED                     0x80000000


/* CK_NOTIFY is an application callback that processes events */
typedef CK_CALLBACK_FUNCTION(CK_RV, CK_NOTIFY)(
  CK_SESSION_HANDLE hSession,     /* the session's handle */
  CK_NOTIFICATION   event,
  CK_VOID_PTR       pApplication  /* passed to C_OpenSession */
);


/* CK_FUNCTION_LIST is a structure holding a Cryptoki spec
 * version and pointers of appropriate types to all the
 * Cryptoki functions */
/* CK_FUNCTION_LIST is new for v2.0 */
typedef struct CK_FUNCTION_LIST CK_FUNCTION_LIST;

typedef CK_FUNCTION_LIST CK_PTR CK_FUNCTION_LIST_PTR;

typedef CK_FUNCTION_LIST_PTR CK_PTR CK_FUNCTION_LIST_PTR_PTR;


/* CK_CREATEMUTEX is an application callback for creating a
 * mutex object */
typedef CK_CALLBACK_FUNCTION(CK_RV, CK_CREATEMUTEX)(
  CK_VOID_PTR_PTR ppMutex  /* location to receive ptr to mutex */
);


/* CK_DESTROYMUTEX is an application callback for destroying a
 * mutex object */
typedef CK_CALLBACK_FUNCTION(CK_RV, CK_DESTROYMUTEX)(
  CK_VOID_PTR pMutex  /* pointer to mutex */
);


/* CK_LOCKMUTEX is an application callback for locking a mutex */
typedef CK_CALLBACK_FUNCTION(CK_RV, CK_LOCKMUTEX)(
  CK_VOID_PTR pMutex  /* pointer to mutex */
);


/* CK_UNLOCKMUTEX is an application callback for unlocking a
 * mutex */
typedef CK_CALLBACK_FUNCTION(CK_RV, CK_UNLOCKMUTEX)(
  CK_VOID_PTR pMutex  /* pointer to mutex */
);


/* CK_C_INITIALIZE_ARGS provides the optional arguments to
```



```
 * C_Initialize */
typedef struct CK_C_INITIALIZE_ARGS {
  CK_CREATEMUTEX CreateMutex;
  CK_DESTROYMUTEX DestroyMutex;
  CK_LOCKMUTEX LockMutex;
  CK_UNLOCKMUTEX UnlockMutex;
  CK_FLAGS flags;
  CK_VOID_PTR pReserved;
} CK_C_INITIALIZE_ARGS;

/* flags: bit flags that provide capabilities of the slot
 *      Bit Flag                          Mask       Meaning
 */
#define CKF_LIBRARY_CANT_CREATE_OS_THREADS 0x00000001
#define CKF_OS_LOCKING_OK                  0x00000002

typedef CK_C_INITIALIZE_ARGS CK_PTR CK_C_INITIALIZE_ARGS_PTR;


/* additional flags for parameters to functions */

/* CKF_DONT_BLOCK is for the function C_WaitForSlotEvent */
#define CKF_DONT_BLOCK     1

/* CK_RSA_PKCS_OAEP_MGF_TYPE is new for v2.10.
 * CK_RSA_PKCS_OAEP_MGF_TYPE  is used to indicate the Message
 * Generation Function (MGF) applied to a message block when
 * formatting a message block for the PKCS #1 OAEP encryption
 * scheme. */
typedef CK_ULONG CK_RSA_PKCS_MGF_TYPE;

typedef CK_RSA_PKCS_MGF_TYPE CK_PTR CK_RSA_PKCS_MGF_TYPE_PTR;

/* The following MGFs are defined */
/* CKG_MGF1_SHA256, CKG_MGF1_SHA384, and CKG_MGF1_SHA512
 * are new for v2.20 */
#define CKG_MGF1_SHA1         0x00000001
#define CKG_MGF1_SHA256       0x00000002
#define CKG_MGF1_SHA384       0x00000003
#define CKG_MGF1_SHA512       0x00000004
/* SHA-224 is new for PKCS #11 v2.20 amendment 3 */
#define CKG_MGF1_SHA224       0x00000005

/* CK_RSA_PKCS_OAEP_SOURCE_TYPE is new for v2.10.
 * CK_RSA_PKCS_OAEP_SOURCE_TYPE  is used to indicate the source
 * of the encoding parameter when formatting a message block
 * for the PKCS #1 OAEP encryption scheme. */
typedef CK_ULONG CK_RSA_PKCS_OAEP_SOURCE_TYPE;

typedef CK_RSA_PKCS_OAEP_SOURCE_TYPE CK_PTR CK_RSA_PKCS_OAEP_SOURCE_TYPE_PTR;

/* The following encoding parameter sources are defined */
#define CKZ_DATA_SPECIFIED    0x00000001

/* CK_RSA_PKCS_OAEP_PARAMS is new for v2.10.
 * CK_RSA_PKCS_OAEP_PARAMS provides the parameters to the
 * CKM_RSA_PKCS_OAEP mechanism. */
typedef struct CK_RSA_PKCS_OAEP_PARAMS {
        CK_MECHANISM_TYPE hashAlg;
        CK_RSA_PKCS_MGF_TYPE mgf;
        CK_RSA_PKCS_OAEP_SOURCE_TYPE source;
        CK_VOID_PTR pSourceData;
        CK_ULONG ulSourceDataLen;
} CK_RSA_PKCS_OAEP_PARAMS;

typedef CK_RSA_PKCS_OAEP_PARAMS CK_PTR CK_RSA_PKCS_OAEP_PARAMS_PTR;
```

```
/* CK_RSA_PKCS_PSS_PARAMS is new for v2.11.
 * CK_RSA_PKCS_PSS_PARAMS provides the parameters to the
 * CKM_RSA_PKCS_PSS mechanism(s). */
typedef struct CK_RSA_PKCS_PSS_PARAMS {
        CK_MECHANISM_TYPE    hashAlg;
        CK_RSA_PKCS_MGF_TYPE mgf;
        CK_ULONG             sLen;
} CK_RSA_PKCS_PSS_PARAMS;

typedef CK_RSA_PKCS_PSS_PARAMS CK_PTR CK_RSA_PKCS_PSS_PARAMS_PTR;

/* CK_EC_KDF_TYPE is new for v2.11. */
typedef CK_ULONG CK_EC_KDF_TYPE;

/* The following EC Key Derivation Functions are defined */
#define CKD_NULL                0x00000001
#define CKD_SHA1_KDF            0x00000002

/* CK_ECDH1_DERIVE_PARAMS is new for v2.11.
 * CK_ECDH1_DERIVE_PARAMS provides the parameters to the
 * CKM_ECDH1_DERIVE and CKM_ECDH1_COFACTOR_DERIVE mechanisms,
 * where each party contributes one key pair.
 */
typedef struct CK_ECDH1_DERIVE_PARAMS {
  CK_EC_KDF_TYPE kdf;
  CK_ULONG ulSharedDataLen;
  CK_BYTE_PTR pSharedData;
  CK_ULONG ulPublicDataLen;
  CK_BYTE_PTR pPublicData;
} CK_ECDH1_DERIVE_PARAMS;

typedef CK_ECDH1_DERIVE_PARAMS CK_PTR CK_ECDH1_DERIVE_PARAMS_PTR;


/* CK_ECDH2_DERIVE_PARAMS is new for v2.11.
 * CK_ECDH2_DERIVE_PARAMS provides the parameters to the
 * CKM_ECMQV_DERIVE mechanism, where each party contributes two key pairs. */
typedef struct CK_ECDH2_DERIVE_PARAMS {
  CK_EC_KDF_TYPE kdf;
  CK_ULONG ulSharedDataLen;
  CK_BYTE_PTR pSharedData;
  CK_ULONG ulPublicDataLen;
  CK_BYTE_PTR pPublicData;
  CK_ULONG ulPrivateDataLen;
  CK_OBJECT_HANDLE hPrivateData;
  CK_ULONG ulPublicDataLen2;
  CK_BYTE_PTR pPublicData2;
} CK_ECDH2_DERIVE_PARAMS;

typedef CK_ECDH2_DERIVE_PARAMS CK_PTR CK_ECDH2_DERIVE_PARAMS_PTR;

typedef struct CK_ECMQV_DERIVE_PARAMS {
  CK_EC_KDF_TYPE kdf;
  CK_ULONG ulSharedDataLen;
  CK_BYTE_PTR pSharedData;
  CK_ULONG ulPublicDataLen;
  CK_BYTE_PTR pPublicData;
  CK_ULONG ulPrivateDataLen;
  CK_OBJECT_HANDLE hPrivateData;
  CK_ULONG ulPublicDataLen2;
  CK_BYTE_PTR pPublicData2;
  CK_OBJECT_HANDLE publicKey;
} CK_ECMQV_DERIVE_PARAMS;

typedef CK_ECMQV_DERIVE_PARAMS CK_PTR CK_ECMQV_DERIVE_PARAMS_PTR;
```



```
/* Typedefs and defines for the CKM_X9_42_DH_KEY_PAIR_GEN and the
 * CKM_X9_42_DH_PARAMETER_GEN mechanisms (new for PKCS #11 v2.11) */
typedef CK_ULONG CK_X9_42_DH_KDF_TYPE;
typedef CK_X9_42_DH_KDF_TYPE CK_PTR CK_X9_42_DH_KDF_TYPE_PTR;

/* The following X9.42 DH key derivation functions are defined
   (besides CKD_NULL already defined : */
#define CKD_SHA1_KDF_ASN1       0x00000003
#define CKD_SHA1_KDF_CONCATENATE 0x00000004

/* CK_X9_42_DH1_DERIVE_PARAMS is new for v2.11.
 * CK_X9_42_DH1_DERIVE_PARAMS provides the parameters to the
 * CKM_X9_42_DH_DERIVE key derivation mechanism, where each party
 * contributes one key pair */
typedef struct CK_X9_42_DH1_DERIVE_PARAMS {
  CK_X9_42_DH_KDF_TYPE kdf;
  CK_ULONG ulOtherInfoLen;
  CK_BYTE_PTR pOtherInfo;
  CK_ULONG ulPublicDataLen;
  CK_BYTE_PTR pPublicData;
} CK_X9_42_DH1_DERIVE_PARAMS;

typedef struct CK_X9_42_DH1_DERIVE_PARAMS CK_PTR CK_X9_42_DH1_DERIVE_PARAMS_PTR;

/* CK_X9_42_DH2_DERIVE_PARAMS is new for v2.11.
 * CK_X9_42_DH2_DERIVE_PARAMS provides the parameters to the
 * CKM_X9_42_DH_HYBRID_DERIVE and CKM_X9_42_MQV_DERIVE key derivation
 * mechanisms, where each party contributes two key pairs */
typedef struct CK_X9_42_DH2_DERIVE_PARAMS {
  CK_X9_42_DH_KDF_TYPE kdf;
  CK_ULONG ulOtherInfoLen;
  CK_BYTE_PTR pOtherInfo;
  CK_ULONG ulPublicDataLen;
  CK_BYTE_PTR pPublicData;
  CK_ULONG ulPrivateDataLen;
  CK_OBJECT_HANDLE hPrivateData;
  CK_ULONG ulPublicDataLen2;
  CK_BYTE_PTR pPublicData2;
} CK_X9_42_DH2_DERIVE_PARAMS;

typedef CK_X9_42_DH2_DERIVE_PARAMS CK_PTR CK_X9_42_DH2_DERIVE_PARAMS_PTR;

typedef struct CK_X9_42_MQV_DERIVE_PARAMS {
  CK_X9_42_DH_KDF_TYPE kdf;
  CK_ULONG ulOtherInfoLen;
  CK_BYTE_PTR pOtherInfo;
  CK_ULONG ulPublicDataLen;
  CK_BYTE_PTR pPublicData;
  CK_ULONG ulPrivateDataLen;
  CK_OBJECT_HANDLE hPrivateData;
  CK_ULONG ulPublicDataLen2;
  CK_BYTE_PTR pPublicData2;
  CK_OBJECT_HANDLE publicKey;
} CK_X9_42_MQV_DERIVE_PARAMS;

typedef CK_X9_42_MQV_DERIVE_PARAMS CK_PTR CK_X9_42_MQV_DERIVE_PARAMS_PTR;

/* CK_KEA_DERIVE_PARAMS provides the parameters to the
 * CKM_KEA_DERIVE mechanism */
/* CK_KEA_DERIVE_PARAMS is new for v2.0 */
typedef struct CK_KEA_DERIVE_PARAMS {
  CK_BBOOL      isSender;
  CK_ULONG      ulRandomLen;
  CK_BYTE_PTR   pRandomA;
  CK_BYTE_PTR   pRandomB;
```

```
  CK_ULONG      ulPublicDataLen;
  CK_BYTE_PTR   pPublicData;
} CK_KEA_DERIVE_PARAMS;

typedef CK_KEA_DERIVE_PARAMS CK_PTR CK_KEA_DERIVE_PARAMS_PTR;


/* CK_RC2_PARAMS provides the parameters to the CKM_RC2_ECB and
 * CKM_RC2_MAC mechanisms.  An instance of CK_RC2_PARAMS just
 * holds the effective keysize */
typedef CK_ULONG          CK_RC2_PARAMS;

typedef CK_RC2_PARAMS CK_PTR CK_RC2_PARAMS_PTR;


/* CK_RC2_CBC_PARAMS provides the parameters to the CKM_RC2_CBC
 * mechanism */
typedef struct CK_RC2_CBC_PARAMS {
  /* ulEffectiveBits was changed from CK_USHORT to CK_ULONG for
   * v2.0 */
  CK_ULONG      ulEffectiveBits;  /* effective bits (1-1024) */

  CK_BYTE       iv[8];            /* IV for CBC mode */
} CK_RC2_CBC_PARAMS;

typedef CK_RC2_CBC_PARAMS CK_PTR CK_RC2_CBC_PARAMS_PTR;


/* CK_RC2_MAC_GENERAL_PARAMS provides the parameters for the
 * CKM_RC2_MAC_GENERAL mechanism */
/* CK_RC2_MAC_GENERAL_PARAMS is new for v2.0 */
typedef struct CK_RC2_MAC_GENERAL_PARAMS {
  CK_ULONG      ulEffectiveBits;  /* effective bits (1-1024) */
  CK_ULONG      ulMacLength;      /* Length of MAC in bytes */
} CK_RC2_MAC_GENERAL_PARAMS;

typedef CK_RC2_MAC_GENERAL_PARAMS CK_PTR \
  CK_RC2_MAC_GENERAL_PARAMS_PTR;


/* CK_RC5_PARAMS provides the parameters to the CKM_RC5_ECB and
 * CKM_RC5_MAC mechanisms */
/* CK_RC5_PARAMS is new for v2.0 */
typedef struct CK_RC5_PARAMS {
  CK_ULONG      ulWordsize;  /* wordsize in bits */
  CK_ULONG      ulRounds;    /* number of rounds */
} CK_RC5_PARAMS;

typedef CK_RC5_PARAMS CK_PTR CK_RC5_PARAMS_PTR;


/* CK_RC5_CBC_PARAMS provides the parameters to the CKM_RC5_CBC
 * mechanism */
/* CK_RC5_CBC_PARAMS is new for v2.0 */
typedef struct CK_RC5_CBC_PARAMS {
  CK_ULONG      ulWordsize;  /* wordsize in bits */
  CK_ULONG      ulRounds;    /* number of rounds */
  CK_BYTE_PTR   pIv;         /* pointer to IV */
  CK_ULONG      ulIvLen;     /* length of IV in bytes */
} CK_RC5_CBC_PARAMS;

typedef CK_RC5_CBC_PARAMS CK_PTR CK_RC5_CBC_PARAMS_PTR;


/* CK_RC5_MAC_GENERAL_PARAMS provides the parameters for the
 * CKM_RC5_MAC_GENERAL mechanism */
```



```
/* CK_RC5_MAC_GENERAL_PARAMS is new for v2.0 */
typedef struct CK_RC5_MAC_GENERAL_PARAMS {
  CK_ULONG      ulWordsize;   /* wordsize in bits */
  CK_ULONG      ulRounds;     /* number of rounds */
  CK_ULONG      ulMacLength;  /* Length of MAC in bytes */
} CK_RC5_MAC_GENERAL_PARAMS;

typedef CK_RC5_MAC_GENERAL_PARAMS CK_PTR \
  CK_RC5_MAC_GENERAL_PARAMS_PTR;


/* CK_MAC_GENERAL_PARAMS provides the parameters to most block
 * ciphers' MAC_GENERAL mechanisms.  Its value is the length of
 * the MAC */
/* CK_MAC_GENERAL_PARAMS is new for v2.0 */
typedef CK_ULONG          CK_MAC_GENERAL_PARAMS;

typedef CK_MAC_GENERAL_PARAMS CK_PTR CK_MAC_GENERAL_PARAMS_PTR;

/* CK_DES/AES_ECB/CBC_ENCRYPT_DATA_PARAMS are new for v2.20 */
typedef struct CK_DES_CBC_ENCRYPT_DATA_PARAMS {
  CK_BYTE      iv[8];
  CK_BYTE_PTR  pData;
  CK_ULONG     length;
} CK_DES_CBC_ENCRYPT_DATA_PARAMS;

typedef CK_DES_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_DES_CBC_ENCRYPT_DATA_PARAMS_PTR

typedef struct CK_AES_CBC_ENCRYPT_DATA_PARAMS {
  CK_BYTE      iv[16];
  CK_BYTE_PTR  pData;
  CK_ULONG     length;
} CK_AES_CBC_ENCRYPT_DATA_PARAMS;

typedef CK_AES_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_AES_CBC_ENCRYPT_DATA_PARAMS_PTR

/* CK_SKIPJACK_PRIVATE_WRAP_PARAMS provides the parameters to the
 * CKM_SKIPJACK_PRIVATE_WRAP mechanism */
/* CK_SKIPJACK_PRIVATE_WRAP_PARAMS is new for v2.0 */
typedef struct CK_SKIPJACK_PRIVATE_WRAP_PARAMS {
  CK_ULONG      ulPasswordLen;
  CK_BYTE_PTR   pPassword;
  CK_ULONG      ulPublicDataLen;
  CK_BYTE_PTR   pPublicData;
  CK_ULONG      ulPAndGLen;
  CK_ULONG      ulQLen;
  CK_ULONG      ulRandomLen;
  CK_BYTE_PTR   pRandomA;
  CK_BYTE_PTR   pPrimeP;
  CK_BYTE_PTR   pBaseG;
  CK_BYTE_PTR   pSubprimeQ;
} CK_SKIPJACK_PRIVATE_WRAP_PARAMS;

typedef CK_SKIPJACK_PRIVATE_WRAP_PARAMS CK_PTR \
  CK_SKIPJACK_PRIVATE_WRAP_PTR;


/* CK_SKIPJACK_RELAYX_PARAMS provides the parameters to the
 * CKM_SKIPJACK_RELAYX mechanism */
/* CK_SKIPJACK_RELAYX_PARAMS is new for v2.0 */
typedef struct CK_SKIPJACK_RELAYX_PARAMS {
  CK_ULONG      ulOldWrappedXLen;
  CK_BYTE_PTR   pOldWrappedX;
  CK_ULONG      ulOldPasswordLen;
  CK_BYTE_PTR   pOldPassword;
  CK_ULONG      ulOldPublicDataLen;
```

```
1580   CK_BYTE_PTR   pOldPublicData;
1581   CK_ULONG      ulOldRandomLen;
1582   CK_BYTE_PTR   pOldRandomA;
1583   CK_ULONG      ulNewPasswordLen;
1584   CK_BYTE_PTR   pNewPassword;
1585   CK_ULONG      ulNewPublicDataLen;
1586   CK_BYTE_PTR   pNewPublicData;
1587   CK_ULONG      ulNewRandomLen;
1588   CK_BYTE_PTR   pNewRandomA;
1589 } CK_SKIPJACK_RELAYX_PARAMS;

1591 typedef CK_SKIPJACK_RELAYX_PARAMS CK_PTR \
1592   CK_SKIPJACK_RELAYX_PARAMS_PTR;


1595 typedef struct CK_PBE_PARAMS {
1596   CK_BYTE_PTR      pInitVector;
1597   CK_UTF8CHAR_PTR  pPassword;
1598   CK_ULONG         ulPasswordLen;
1599   CK_BYTE_PTR      pSalt;
1600   CK_ULONG         ulSaltLen;
1601   CK_ULONG         ulIteration;
1602 } CK_PBE_PARAMS;

1604 typedef CK_PBE_PARAMS CK_PTR CK_PBE_PARAMS_PTR;


1607 /* CK_KEY_WRAP_SET_OAEP_PARAMS provides the parameters to the
1608  * CKM_KEY_WRAP_SET_OAEP mechanism */
1609 /* CK_KEY_WRAP_SET_OAEP_PARAMS is new for v2.0 */
1610 typedef struct CK_KEY_WRAP_SET_OAEP_PARAMS {
1611   CK_BYTE       bBC;     /* block contents byte */
1612   CK_BYTE_PTR   pX;      /* extra data */
1613   CK_ULONG      ulXLen;  /* length of extra data in bytes */
1614 } CK_KEY_WRAP_SET_OAEP_PARAMS;

1616 typedef CK_KEY_WRAP_SET_OAEP_PARAMS CK_PTR \
1617   CK_KEY_WRAP_SET_OAEP_PARAMS_PTR;


1620 typedef struct CK_SSL3_RANDOM_DATA {
1621   CK_BYTE_PTR  pClientRandom;
1622   CK_ULONG     ulClientRandomLen;
1623   CK_BYTE_PTR  pServerRandom;
1624   CK_ULONG     ulServerRandomLen;
1625 } CK_SSL3_RANDOM_DATA;


1628 typedef struct CK_SSL3_MASTER_KEY_DERIVE_PARAMS {
1629   CK_SSL3_RANDOM_DATA RandomInfo;
1630   CK_VERSION_PTR pVersion;
1631 } CK_SSL3_MASTER_KEY_DERIVE_PARAMS;

1633 typedef struct CK_SSL3_MASTER_KEY_DERIVE_PARAMS CK_PTR \
1634   CK_SSL3_MASTER_KEY_DERIVE_PARAMS_PTR;


1637 typedef struct CK_SSL3_KEY_MAT_OUT {
1638   CK_OBJECT_HANDLE hClientMacSecret;
1639   CK_OBJECT_HANDLE hServerMacSecret;
1640   CK_OBJECT_HANDLE hClientKey;
1641   CK_OBJECT_HANDLE hServerKey;
1642   CK_BYTE_PTR      pIVClient;
1643   CK_BYTE_PTR      pIVServer;
1644 } CK_SSL3_KEY_MAT_OUT;
```



```
1646 typedef CK_SSL3_KEY_MAT_OUT CK_PTR CK_SSL3_KEY_MAT_OUT_PTR;


1649 typedef struct CK_SSL3_KEY_MAT_PARAMS {
1650   CK_ULONG               ulMacSizeInBits;
1651   CK_ULONG               ulKeySizeInBits;
1652   CK_ULONG               ulIVSizeInBits;
1653   CK_BBOOL               bIsExport;
1654   CK_SSL3_RANDOM_DATA    RandomInfo;
1655   CK_SSL3_KEY_MAT_OUT_PTR pReturnedKeyMaterial;
1656 } CK_SSL3_KEY_MAT_PARAMS;

1658 typedef CK_SSL3_KEY_MAT_PARAMS CK_PTR CK_SSL3_KEY_MAT_PARAMS_PTR;

1660 /* CK_TLS_PRF_PARAMS is new for version 2.20 */
1661 typedef struct CK_TLS_PRF_PARAMS {
1662   CK_BYTE_PTR  pSeed;
1663   CK_ULONG     ulSeedLen;
1664   CK_BYTE_PTR  pLabel;
1665   CK_ULONG     ulLabelLen;
1666   CK_BYTE_PTR  pOutput;
1667   CK_ULONG_PTR pulOutputLen;
1668 } CK_TLS_PRF_PARAMS;

1670 typedef CK_TLS_PRF_PARAMS CK_PTR CK_TLS_PRF_PARAMS_PTR;

1672 /* WTLS is new for version 2.20 */
1673 typedef struct CK_WTLS_RANDOM_DATA {
1674   CK_BYTE_PTR pClientRandom;
1675   CK_ULONG    ulClientRandomLen;
1676   CK_BYTE_PTR pServerRandom;
1677   CK_ULONG    ulServerRandomLen;
1678 } CK_WTLS_RANDOM_DATA;

1680 typedef CK_WTLS_RANDOM_DATA CK_PTR CK_WTLS_RANDOM_DATA_PTR;

1682 typedef struct CK_WTLS_MASTER_KEY_DERIVE_PARAMS {
1683   CK_MECHANISM_TYPE   DigestMechanism;
1684   CK_WTLS_RANDOM_DATA RandomInfo;
1685   CK_BYTE_PTR         pVersion;
1686 } CK_WTLS_MASTER_KEY_DERIVE_PARAMS;

1688 typedef CK_WTLS_MASTER_KEY_DERIVE_PARAMS CK_PTR \
1689   CK_WTLS_MASTER_KEY_DERIVE_PARAMS_PTR;

1691 typedef struct CK_WTLS_PRF_PARAMS {
1692   CK_MECHANISM_TYPE DigestMechanism;
1693   CK_BYTE_PTR       pSeed;
1694   CK_ULONG          ulSeedLen;
1695   CK_BYTE_PTR       pLabel;
1696   CK_ULONG          ulLabelLen;
1697   CK_BYTE_PTR       pOutput;
1698   CK_ULONG_PTR      pulOutputLen;
1699 } CK_WTLS_PRF_PARAMS;

1701 typedef CK_WTLS_PRF_PARAMS CK_PTR CK_WTLS_PRF_PARAMS_PTR;

1703 typedef struct CK_WTLS_KEY_MAT_OUT {
1704   CK_OBJECT_HANDLE hMacSecret;
1705   CK_OBJECT_HANDLE hKey;
1706   CK_BYTE_PTR      pIV;
1707 } CK_WTLS_KEY_MAT_OUT;

1709 typedef CK_WTLS_KEY_MAT_OUT CK_PTR CK_WTLS_KEY_MAT_OUT_PTR;

1711 typedef struct CK_WTLS_KEY_MAT_PARAMS {
```

```
  CK_MECHANISM_TYPE       DigestMechanism;
  CK_ULONG                ulMacSizeInBits;
  CK_ULONG                ulKeySizeInBits;
  CK_ULONG                ulIVSizeInBits;
  CK_ULONG                ulSequenceNumber;
  CK_BBOOL                bIsExport;
  CK_WTLS_RANDOM_DATA     RandomInfo;
  CK_WTLS_KEY_MAT_OUT_PTR pReturnedKeyMaterial;
} CK_WTLS_KEY_MAT_PARAMS;

typedef CK_WTLS_KEY_MAT_PARAMS CK_PTR CK_WTLS_KEY_MAT_PARAMS_PTR;

/* CMS is new for version 2.20 */
typedef struct CK_CMS_SIG_PARAMS {
  CK_OBJECT_HANDLE      certificateHandle;
  CK_MECHANISM_PTR      pSigningMechanism;
  CK_MECHANISM_PTR      pDigestMechanism;
  CK_UTF8CHAR_PTR       pContentType;
  CK_BYTE_PTR           pRequestedAttributes;
  CK_ULONG              ulRequestedAttributesLen;
  CK_BYTE_PTR           pRequiredAttributes;
  CK_ULONG              ulRequiredAttributesLen;
} CK_CMS_SIG_PARAMS;

typedef CK_CMS_SIG_PARAMS CK_PTR CK_CMS_SIG_PARAMS_PTR;

typedef struct CK_KEY_DERIVATION_STRING_DATA {
  CK_BYTE_PTR pData;
  CK_ULONG    ulLen;
} CK_KEY_DERIVATION_STRING_DATA;

typedef CK_KEY_DERIVATION_STRING_DATA CK_PTR \
  CK_KEY_DERIVATION_STRING_DATA_PTR;


/* The CK_EXTRACT_PARAMS is used for the
 * CKM_EXTRACT_KEY_FROM_KEY mechanism.  It specifies which bit
 * of the base key should be used as the first bit of the
 * derived key */
/* CK_EXTRACT_PARAMS is new for v2.0 */
typedef CK_ULONG CK_EXTRACT_PARAMS;

typedef CK_EXTRACT_PARAMS CK_PTR CK_EXTRACT_PARAMS_PTR;

/* CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE is new for v2.10.
 * CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE is used to
 * indicate the Pseudo-Random Function (PRF) used to generate
 * key bits using PKCS #5 PBKDF2. */
typedef CK_ULONG CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE;

typedef CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE CK_PTR CK_PKCS5_PBKD2_PSEUDO_

/* The following PRFs are defined in PKCS #5 v2.0. */
#define CKP_PKCS5_PBKD2_HMAC_SHA1 0x00000001


/* CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE is new for v2.10.
 * CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE is used to indicate the
 * source of the salt value when deriving a key using PKCS #5
 * PBKDF2. */
typedef CK_ULONG CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE;

typedef CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE CK_PTR CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE

/* The following salt value sources are defined in PKCS #5 v2.0. */
#define CKZ_SALT_SPECIFIED        0x00000001
```



```
/* CK_PKCS5_PBKD2_PARAMS is new for v2.10.
 * CK_PKCS5_PBKD2_PARAMS is a structure that provides the
 * parameters to the CKM_PKCS5_PBKD2 mechanism. */
typedef struct CK_PKCS5_PBKD2_PARAMS {
        CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE           saltSource;
        CK_VOID_PTR                                pSaltSourceData;
        CK_ULONG                                   ulSaltSourceDataLen;
        CK_ULONG                                   iterations;
        CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE prf;
        CK_VOID_PTR                                pPrfData;
        CK_ULONG                                   ulPrfDataLen;
        CK_UTF8CHAR_PTR                            pPassword;
        CK_ULONG_PTR                               ulPasswordLen;
} CK_PKCS5_PBKD2_PARAMS;

typedef CK_PKCS5_PBKD2_PARAMS CK_PTR CK_PKCS5_PBKD2_PARAMS_PTR;

/* All CK_OTP structs are new for PKCS #11 v2.20 amendment 3 */

typedef CK_ULONG CK_OTP_PARAM_TYPE;
typedef CK_OTP_PARAM_TYPE CK_PARAM_TYPE; /* B/w compatibility */

typedef struct CK_OTP_PARAM {
    CK_OTP_PARAM_TYPE type;
    CK_VOID_PTR pValue;
    CK_ULONG ulValueLen;
} CK_OTP_PARAM;

typedef CK_OTP_PARAM CK_PTR CK_OTP_PARAM_PTR;

typedef struct CK_OTP_PARAMS {
    CK_OTP_PARAM_PTR pParams;
    CK_ULONG ulCount;
} CK_OTP_PARAMS;

typedef CK_OTP_PARAMS CK_PTR CK_OTP_PARAMS_PTR;

typedef struct CK_OTP_SIGNATURE_INFO {
    CK_OTP_PARAM_PTR pParams;
    CK_ULONG ulCount;
} CK_OTP_SIGNATURE_INFO;

typedef CK_OTP_SIGNATURE_INFO CK_PTR CK_OTP_SIGNATURE_INFO_PTR;

/* The following OTP-related defines are new for PKCS #11 v2.20 amendment 1 */
#define CK_OTP_VALUE         0
#define CK_OTP_PIN           1
#define CK_OTP_CHALLENGE     2
#define CK_OTP_TIME          3
#define CK_OTP_COUNTER       4
#define CK_OTP_FLAGS         5
#define CK_OTP_OUTPUT_LENGTH 6
#define CK_OTP_OUTPUT_FORMAT 7

/* The following OTP-related defines are new for PKCS #11 v2.20 amendment 1 */
#define CKF_NEXT_OTP         0x00000001
#define CKF_EXCLUDE_TIME     0x00000002
#define CKF_EXCLUDE_COUNTER  0x00000004
#define CKF_EXCLUDE_CHALLENGE 0x00000008
#define CKF_EXCLUDE_PIN      0x00000010
#define CKF_USER_FRIENDLY_OTP 0x00000020

/* CK_KIP_PARAMS is new for PKCS #11 v2.20 amendment 2 */
typedef struct CK_KIP_PARAMS {
    CK_MECHANISM_PTR  pMechanism;
```

```
1844     CK_OBJECT_HANDLE  hKey;
1845     CK_BYTE_PTR       pSeed;
1846     CK_ULONG          ulSeedLen;
1847 } CK_KIP_PARAMS;

1849 typedef CK_KIP_PARAMS CK_PTR CK_KIP_PARAMS_PTR;

1851 /* CK_AES_CTR_PARAMS is new for PKCS #11 v2.20 amendment 3 */
1852 typedef struct CK_AES_CTR_PARAMS {
1853     CK_ULONG ulCounterBits;
1854     CK_BYTE cb[16];
1855 } CK_AES_CTR_PARAMS;

1857 typedef CK_AES_CTR_PARAMS CK_PTR CK_AES_CTR_PARAMS_PTR;

1859 /* CK_CAMELLIA_CTR_PARAMS is new for PKCS #11 v2.20 amendment 3 */
1860 typedef struct CK_CAMELLIA_CTR_PARAMS {
1861     CK_ULONG ulCounterBits;
1862     CK_BYTE cb[16];
1863 } CK_CAMELLIA_CTR_PARAMS;

1865 typedef CK_CAMELLIA_CTR_PARAMS CK_PTR CK_CAMELLIA_CTR_PARAMS_PTR;

1867 /* CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS is new for PKCS #11 v2.20 amendment 3 */
1868 typedef struct CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS {
1869     CK_BYTE      iv[16];
1870     CK_BYTE_PTR  pData;
1871     CK_ULONG     length;
1872 } CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS;

1874 typedef CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_CAMELLIA_CBC_ENCRYPT_DATA_

1876 /* CK_ARIA_CBC_ENCRYPT_DATA_PARAMS is new for PKCS #11 v2.20 amendment 3 */
1877 typedef struct CK_ARIA_CBC_ENCRYPT_DATA_PARAMS {
1878     CK_BYTE      iv[16];
1879     CK_BYTE_PTR  pData;
1880     CK_ULONG     length;
1881 } CK_ARIA_CBC_ENCRYPT_DATA_PARAMS;

1883 typedef CK_ARIA_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_ARIA_CBC_ENCRYPT_DATA_PARAMS_P

1885 #endif
1886 #endif /* ! codereview */
```

```
**********************************************************
    7461 Wed Aug 13 19:51:52 2014
new/usr/src/lib/openssl/include/rand_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/rand/rand_lcl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_RAND_LCL_H
#define HEADER_RAND_LCL_H

#define ENTROPY_NEEDED 32  /* require 256 bits = 32 bytes of randomness */


#if !defined(USE_MD5_RAND) && !defined(USE_SHA1_RAND) && !defined(USE_MDC2_RAND)
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA1)
#define USE_SHA1_RAND
#elif !defined(OPENSSL_NO_MD5)
#define USE_MD5_RAND
#elif !defined(OPENSSL_NO_MDC2) && !defined(OPENSSL_NO_DES)
#define USE_MDC2_RAND
#elif !defined(OPENSSL_NO_MD2)
#define USE_MD2_RAND
#else
```

```
#error No message digest algorithm available
#endif
#endif

#include <openssl/evp.h>
#define MD_Update(a,b,c)        EVP_DigestUpdate(a,b,c)
#define MD_Final(a,b)           EVP_DigestFinal_ex(a,b,NULL)
#if defined(USE_MD5_RAND)
#include <openssl/md5.h>
#define MD_DIGEST_LENGTH        MD5_DIGEST_LENGTH
#define MD_Init(a)              EVP_DigestInit_ex(a,EVP_md5(), NULL)
#define MD(a,b,c)               EVP_Digest(a,b,c,NULL,EVP_md5(), NULL)
#elif defined(USE_SHA1_RAND)
#include <openssl/sha.h>
#define MD_DIGEST_LENGTH        SHA_DIGEST_LENGTH
#define MD_Init(a)              EVP_DigestInit_ex(a,EVP_sha1(), NULL)
#define MD(a,b,c)               EVP_Digest(a,b,c,NULL,EVP_sha1(), NULL)
#elif defined(USE_MDC2_RAND)
#include <openssl/mdc2.h>
#define MD_DIGEST_LENGTH        MDC2_DIGEST_LENGTH
#define MD_Init(a)              EVP_DigestInit_ex(a,EVP_mdc2(), NULL)
#define MD(a,b,c)               EVP_Digest(a,b,c,NULL,EVP_mdc2(), NULL)
#elif defined(USE_MD2_RAND)
#include <openssl/md2.h>
#define MD_DIGEST_LENGTH        MD2_DIGEST_LENGTH
#define MD_Init(a)              EVP_DigestInit_ex(a,EVP_md2(), NULL)
#define MD(a,b,c)               EVP_Digest(a,b,c,NULL,EVP_md2(), NULL)
#endif

int ssleay_rand_bytes(unsigned char *buf, int num, int pseudo, int lock);

#endif
#endif /* ! codereview */
```

```
**********************************************************
    6669 Wed Aug 13 19:51:52 2014
new/usr/src/lib/openssl/include/rc2_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/rc2/rc2_locl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#undef c2l
#define c2l(c,l)        (l =((unsigned long)(*((c)++)))    , \
                         l|=((unsigned long)(*((c)++)))<< 8L, \
```



```
                         l|=((unsigned long)(*((c)++)))<<16L, \
                         l|=((unsigned long)(*((c)++)))<<24L)

/* NOTE - c is not incremented as per c2l */
#undef c2ln
#define c2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))<<24L; \
                        case 7: l2|=((unsigned long)(*(--(c))))<<16L; \
                        case 6: l2|=((unsigned long)(*(--(c))))<< 8L; \
                        case 5: l2|=((unsigned long)(*(--(c))));     \
                        case 4: l1 =((unsigned long)(*(--(c))))<<24L; \
                        case 3: l1|=((unsigned long)(*(--(c))))<<16L; \
                        case 2: l1|=((unsigned long)(*(--(c))))<< 8L; \
                        case 1: l1|=((unsigned long)(*(--(c))));     \
                                } \
                        }

#undef l2c
#define l2c(l,c)        (*((c)++)=(unsigned char)(((l)     )&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24L)&0xff))

/* NOTE - c is not incremented as per l2c */
#undef l2cn
#define l2cn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)>>24L)&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>>16L)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>> 8L)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)     )&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)>>24L)&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>>16L)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>> 8L)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)     )&0xff); \
                                } \
                        }

/* NOTE - c is not incremented as per n2l */
#define n2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))    ; \
                        case 7: l2|=((unsigned long)(*(--(c))))<< 8; \
                        case 6: l2|=((unsigned long)(*(--(c))))<<16; \
                        case 5: l2|=((unsigned long)(*(--(c))))<<24; \
                        case 4: l1 =((unsigned long)(*(--(c))))    ; \
                        case 3: l1|=((unsigned long)(*(--(c))))<< 8; \
                        case 2: l1|=((unsigned long)(*(--(c))))<<16; \
                        case 1: l1|=((unsigned long)(*(--(c))))<<24; \
                                } \
                        }

/* NOTE - c is not incremented as per l2n */
#define l2nn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)    )&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>> 8)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>>16)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)>>24)&0xff); \
```

```
128                         case 4: *(--(c))=(unsigned char)(((l1)    )&0xff); \
129                         case 3: *(--(c))=(unsigned char)(((l1)>> 8)&0xff); \
130                         case 2: *(--(c))=(unsigned char)(((l1)>>16)&0xff); \
131                         case 1: *(--(c))=(unsigned char)(((l1)>>24)&0xff); \
132                                 } \
133                         }

135 #undef n2l
136 #define n2l(c,l)        (l =((unsigned long)(*((c)++)))<<24L, \
137                          l|=((unsigned long)(*((c)++)))<<16L, \
138                          l|=((unsigned long)(*((c)++)))<< 8L, \
139                          l|=((unsigned long)(*((c)++))))

141 #undef l2n
142 #define l2n(l,c)        (*((c)++)=(unsigned char)(((l)>>24L)&0xff), \
143                          *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
144                          *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
145                          *((c)++)=(unsigned char)(((l)     )&0xff))

147 #define C_RC2(n) \
148         t=(x0+(x1& ~x3)+(x2&x3)+ *(p0++))&0xffff; \
149         x0=(t<<1)|(t>>15); \
150         t=(x1+(x2& ~x0)+(x3&x0)+ *(p0++))&0xffff; \
151         x1=(t<<2)|(t>>14); \
152         t=(x2+(x3& ~x1)+(x0&x1)+ *(p0++))&0xffff; \
153         x2=(t<<3)|(t>>13); \
154         t=(x3+(x0& ~x2)+(x1&x2)+ *(p0++))&0xffff; \
155         x3=(t<<5)|(t>>11);
156 #endif /* ! codereview */
```

```
**********************************************************
     114 Wed Aug 13 19:51:52 2014
new/usr/src/lib/openssl/include/rc4_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #ifndef HEADER_RC4_LOCL_H
   2 #define HEADER_RC4_LOCL_H
   3 #include <openssl/opensslconf.h>
   4 #include <cryptlib.h>
   5 #endif
   6 #endif /* ! codereview */
```

```
**********************************************************
    7911 Wed Aug 13 19:51:52 2014
new/usr/src/lib/openssl/include/rc5_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc5/rc5_locl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdlib.h>

#undef c2l
```



```
#define c2l(c,l)        (l =((unsigned long)(*((c)++)))    , \
                         l|=((unsigned long)(*((c)++)))<< 8L, \
                         l|=((unsigned long)(*((c)++)))<<16L, \
                         l|=((unsigned long)(*((c)++)))<<24L)

/* NOTE - c is not incremented as per c2l */
#undef c2ln
#define c2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))<<24L; \
                        case 7: l2|=((unsigned long)(*(--(c))))<<16L; \
                        case 6: l2|=((unsigned long)(*(--(c))))<< 8L; \
                        case 5: l2|=((unsigned long)(*(--(c))));     \
                        case 4: l1 =((unsigned long)(*(--(c))))<<24L; \
                        case 3: l1|=((unsigned long)(*(--(c))))<<16L; \
                        case 2: l1|=((unsigned long)(*(--(c))))<< 8L; \
                        case 1: l1|=((unsigned long)(*(--(c))));     \
                                } \
                        }

#undef l2c
#define l2c(l,c)        (*((c)++)=(unsigned char)(((l)     )&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24L)&0xff))

/* NOTE - c is not incremented as per l2c */
#undef l2cn
#define l2cn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)>>24L)&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>>16L)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>> 8L)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)     )&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)>>24L)&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>>16L)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>> 8L)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)     )&0xff); \
                                } \
                        }

/* NOTE - c is not incremented as per n2l */
#define n2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))    ; \
                        case 7: l2|=((unsigned long)(*(--(c))))<< 8; \
                        case 6: l2|=((unsigned long)(*(--(c))))<<16; \
                        case 5: l2|=((unsigned long)(*(--(c))))<<24; \
                        case 4: l1 =((unsigned long)(*(--(c))))    ; \
                        case 3: l1|=((unsigned long)(*(--(c))))<< 8; \
                        case 2: l1|=((unsigned long)(*(--(c))))<<16; \
                        case 1: l1|=((unsigned long)(*(--(c))))<<24; \
                                } \
                        }

/* NOTE - c is not incremented as per l2n */
#define l2nn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)    )&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>> 8)&0xff); \
```

```
128                         case 6: *(--(c))=(unsigned char)(((l2)>>16)&0xff); \
129                         case 5: *(--(c))=(unsigned char)(((l2)>>24)&0xff); \
130                         case 4: *(--(c))=(unsigned char)(((l1)    )&0xff); \
131                         case 3: *(--(c))=(unsigned char)(((l1)>> 8)&0xff); \
132                         case 2: *(--(c))=(unsigned char)(((l1)>>16)&0xff); \
133                         case 1: *(--(c))=(unsigned char)(((l1)>>24)&0xff); \
134                                 } \
135                         }

137 #undef n2l
138 #define n2l(c,l)        (l =((unsigned long)(*((c)++)))<<24L, \
139                          l|=((unsigned long)(*((c)++)))<<16L, \
140                          l|=((unsigned long)(*((c)++)))<< 8L, \
141                          l|=((unsigned long)(*((c)++))))

143 #undef l2n
144 #define l2n(l,c)        (*((c)++)=(unsigned char)(((l)>>24L)&0xff), \
145                          *((c)++)=(unsigned char)(((l)>>16L)&0xff), \
146                          *((c)++)=(unsigned char)(((l)>> 8L)&0xff), \
147                          *((c)++)=(unsigned char)(((l)     )&0xff))

149 #if (defined(OPENSSL_SYS_WIN32) && defined(_MSC_VER)) || defined(__ICC)
150 #define ROTATE_l32(a,n)     _lrotl(a,n)
151 #define ROTATE_r32(a,n)     _lrotr(a,n)
152 #elif defined(__GNUC__) && __GNUC__>=2 && !defined(__STRICT_ANSI__) && !defined(
153 # if defined(__i386) || defined(__i386__) || defined(__x86_64) || defined(__x86_
154 #  define ROTATE_l32(a,n)       ({ register unsigned int ret;   \
155                                   asm ("roll %%cl,%0"     \
156                                         : "=r"(ret)     \
157                                         : "c"(n),"0"((unsigned int)(a))
158                                         : "cc");        \
159                                   ret;                  \
160                                })
161 #  define ROTATE_r32(a,n)       ({ register unsigned int ret;   \
162                                   asm ("rorl %%cl,%0"     \
163                                         : "=r"(ret)     \
164                                         : "c"(n),"0"((unsigned int)(a))
165                                         : "cc");        \
166                                   ret;                  \
167                                })
168 # endif
169 #endif
170 #ifndef ROTATE_l32
171 #define ROTATE_l32(a,n)     (((a)<<(n&0x1f))|(((a)&0xffffffff)>>(32-(n&0x1f))))
172 #endif
173 #ifndef ROTATE_r32
174 #define ROTATE_r32(a,n)     (((a)<<(32-(n&0x1f)))|(((a)&0xffffffff)>>(n&0x1f)))
175 #endif

177 #define RC5_32_MASK     0xffffffffL

179 #define RC5_16_P        0xB7E1
180 #define RC5_16_Q        0x9E37
181 #define RC5_32_P        0xB7E15163L
182 #define RC5_32_Q        0x9E3779B9L
183 #define RC5_64_P        0xB7E151628AED2A6BLL
184 #define RC5_64_Q        0x9E3779B97F4A7C15LL

186 #define E_RC5_32(a,b,s,n) \
187         a^=b; \
188         a=ROTATE_l32(a,b); \
189         a+=s[n]; \
190         a&=RC5_32_MASK; \
191         b^=a; \
192         b=ROTATE_l32(b,a); \
193         b+=s[n+1]; \
```



```
194         b&=RC5_32_MASK;

196 #define D_RC5_32(a,b,s,n) \
197         b-=s[n+1]; \
198         b&=RC5_32_MASK; \
199         b=ROTATE_r32(b,a); \
200         b^=a; \
201         a-=s[n]; \
202         a&=RC5_32_MASK; \
203         a=ROTATE_r32(a,b); \
204         a^=b;
205 #endif /* ! codereview */
```

```
**********************************************************
    5712 Wed Aug 13 19:51:53 2014
new/usr/src/lib/openssl/include/rmd_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ripemd/rmd_locl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdlib.h>
#include <string.h>
#include <openssl/opensslconf.h>
```



```
#include <openssl/ripemd.h>

#ifndef RIPEMD160_LONG_LOG2
#define RIPEMD160_LONG_LOG2 2 /* default to 32 bits */
#endif

/*
 * DO EXAMINE COMMENTS IN crypto/md5/md5_locl.h & crypto/md5/md5_dgst.c
 * FOR EXPLANATIONS ON FOLLOWING "CODE."
 *                                      <appro@fy.chalmers.se>
 */
#ifdef RMD160_ASM
# if defined(__i386) || defined(__i386__) || defined(_M_IX86) || defined(__INTEL
#  define ripemd160_block_data_order ripemd160_block_asm_data_order
# endif
#endif

void ripemd160_block_data_order (RIPEMD160_CTX *c, const void *p,size_t num);

#define DATA_ORDER_IS_LITTLE_ENDIAN

#define HASH_LONG               RIPEMD160_LONG
#define HASH_CTX                RIPEMD160_CTX
#define HASH_CBLOCK             RIPEMD160_CBLOCK
#define HASH_UPDATE             RIPEMD160_Update
#define HASH_TRANSFORM          RIPEMD160_Transform
#define HASH_FINAL              RIPEMD160_Final
#define HASH_MAKE_STRING(c,s)   do {    \
        unsigned long ll;               \
        ll=(c)->A; (void)HOST_l2c(ll,(s));      \
        ll=(c)->B; (void)HOST_l2c(ll,(s));      \
        ll=(c)->C; (void)HOST_l2c(ll,(s));      \
        ll=(c)->D; (void)HOST_l2c(ll,(s));      \
        ll=(c)->E; (void)HOST_l2c(ll,(s));      \
        } while (0)
#define HASH_BLOCK_DATA_ORDER   ripemd160_block_data_order

#include "md32_common.h"

#if 0
#define F1(x,y,z)        ((x)^(y)^(z))
#define F2(x,y,z)       (((x)&(y))|((~x)&z))
#define F3(x,y,z)       (((x)|(~y))^(z))
#define F4(x,y,z)       (((x)&(z))|((y)&(~(z))))
#define F5(x,y,z)        ((x)^((y)|(~(z))))
#else
/*
 * Transformed F2 and F4 are courtesy of Wei Dai <weidai@eskimo.com>
 */
#define F1(x,y,z)       ((x) ^ (y) ^ (z))
#define F2(x,y,z)       ((((y) ^ (z)) & (x)) ^ (z))
#define F3(x,y,z)       (((~(y)) | (x)) ^ (z))
#define F4(x,y,z)       ((((x) ^ (y)) & (z)) ^ (y))
#define F5(x,y,z)       (((~(z)) | (y)) ^ (x))
#endif

#define RIPEMD160_A     0x67452301L
#define RIPEMD160_B     0xEFCDAB89L
#define RIPEMD160_C     0x98BADCFEL
#define RIPEMD160_D     0x10325476L
#define RIPEMD160_E     0xC3D2E1F0L

#include "rmdconst.h"

#define RIP1(a,b,c,d,e,w,s) { \
        a+=F1(b,c,d)+X(w); \
```

```
128         a=ROTATE(a,s)+e; \
129         c=ROTATE(c,10); }

131 #define RIP2(a,b,c,d,e,w,s,K) { \
132         a+=F2(b,c,d)+X(w)+K; \
133         a=ROTATE(a,s)+e; \
134         c=ROTATE(c,10); }

136 #define RIP3(a,b,c,d,e,w,s,K) { \
137         a+=F3(b,c,d)+X(w)+K; \
138         a=ROTATE(a,s)+e; \
139         c=ROTATE(c,10); }

141 #define RIP4(a,b,c,d,e,w,s,K) { \
142         a+=F4(b,c,d)+X(w)+K; \
143         a=ROTATE(a,s)+e; \
144         c=ROTATE(c,10); }

146 #define RIP5(a,b,c,d,e,w,s,K) { \
147         a+=F5(b,c,d)+X(w)+K; \
148         a=ROTATE(a,s)+e; \
149         c=ROTATE(c,10); }
150 #endif /* ! codereview */
```

```
**********************************************************
    8565 Wed Aug 13 19:51:53 2014
new/usr/src/lib/openssl/include/rmdconst.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/ripemd/rmdconst.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
#define KL0 0x00000000L
#define KL1 0x5A827999L
#define KL2 0x6ED9EBA1L
#define KL3 0x8F1BBCDCL
```



```
#define KL4 0xA953FD4EL

#define KR0 0x50A28BE6L
#define KR1 0x5C4DD124L
#define KR2 0x6D703EF3L
#define KR3 0x7A6D76E9L
#define KR4 0x00000000L

#define WL00  0
#define SL00 11
#define WL01  1
#define SL01 14
#define WL02  2
#define SL02 15
#define WL03  3
#define SL03 12
#define WL04  4
#define SL04  5
#define WL05  5
#define SL05  8
#define WL06  6
#define SL06  7
#define WL07  7
#define SL07  9
#define WL08  8
#define SL08 11
#define WL09  9
#define SL09 13
#define WL10 10
#define SL10 14
#define WL11 11
#define SL11 15
#define WL12 12
#define SL12  6
#define WL13 13
#define SL13  7
#define WL14 14
#define SL14  9
#define WL15 15
#define SL15  8

#define WL16  7
#define SL16  7
#define WL17  4
#define SL17  6
#define WL18 13
#define SL18  8
#define WL19  1
#define SL19 13
#define WL20 10
#define SL20 11
#define WL21  6
#define SL21  9
#define WL22 15
#define SL22  7
#define WL23  3
#define SL23 15
#define WL24 12
#define SL24  7
#define WL25  0
#define SL25 12
#define WL26  9
#define SL26 15
#define WL27  5
#define SL27  9
#define WL28  2
```

```
#define SL28 11
#define WL29 14
#define SL29  7
#define WL30 11
#define SL30 13
#define WL31  8
#define SL31 12

#define WL32  3
#define SL32 11
#define WL33 10
#define SL33 13
#define WL34 14
#define SL34  6
#define WL35  4
#define SL35  7
#define WL36  9
#define SL36 14
#define WL37 15
#define SL37  9
#define WL38  8
#define SL38 13
#define WL39  1
#define SL39 15
#define WL40  2
#define SL40 14
#define WL41  7
#define SL41  8
#define WL42  0
#define SL42 13
#define WL43  6
#define SL43  6
#define WL44 13
#define SL44  5
#define WL45 11
#define SL45 12
#define WL46  5
#define SL46  7
#define WL47 12
#define SL47  5

#define WL48  1
#define SL48 11
#define WL49  9
#define SL49 12
#define WL50 11
#define SL50 14
#define WL51 10
#define SL51 15
#define WL52  0
#define SL52 14
#define WL53  8
#define SL53 15
#define WL54 12
#define SL54  9
#define WL55  4
#define SL55  8
#define WL56 13
#define SL56  9
#define WL57  3
#define SL57 14
#define WL58  7
#define SL58  5
#define WL59 15
#define SL59  6
#define WL60 14
```



```
#define SL60  8
#define WL61  5
#define SL61  6
#define WL62  6
#define SL62  5
#define WL63  2
#define SL63 12

#define WL64  4
#define SL64  9
#define WL65  0
#define SL65 15
#define WL66  5
#define SL66  5
#define WL67  9
#define SL67 11
#define WL68  7
#define SL68  6
#define WL69 12
#define SL69  8
#define WL70  2
#define SL70 13
#define WL71 10
#define SL71 12
#define WL72 14
#define SL72  5
#define WL73  1
#define SL73 12
#define WL74  3
#define SL74 13
#define WL75  8
#define SL75 14
#define WL76 11
#define SL76 11
#define WL77  6
#define SL77  8
#define WL78 15
#define SL78  5
#define WL79 13
#define SL79  6

#define WR00  5
#define SR00  8
#define WR01 14
#define SR01  9
#define WR02  7
#define SR02  9
#define WR03  0
#define SR03 11
#define WR04  9
#define SR04 13
#define WR05  2
#define SR05 15
#define WR06 11
#define SR06 15
#define WR07  4
#define SR07  5
#define WR08 13
#define SR08  7
#define WR09  6
#define SR09  7
#define WR10 15
#define SR10  8
#define WR11  8
#define SR11 11
#define WR12  1
```

```
260 #define SR12 14
261 #define WR13 10
262 #define SR13 14
263 #define WR14  3
264 #define SR14 12
265 #define WR15 12
266 #define SR15  6

268 #define WR16  6
269 #define SR16  9
270 #define WR17 11
271 #define SR17 13
272 #define WR18  3
273 #define SR18 15
274 #define WR19  7
275 #define SR19  7
276 #define WR20  0
277 #define SR20 12
278 #define WR21 13
279 #define SR21  8
280 #define WR22  5
281 #define SR22  9
282 #define WR23 10
283 #define SR23 11
284 #define WR24 14
285 #define SR24  7
286 #define WR25 15
287 #define SR25  7
288 #define WR26  8
289 #define SR26 12
290 #define WR27 12
291 #define SR27  7
292 #define WR28  4
293 #define SR28  6
294 #define WR29  9
295 #define SR29 15
296 #define WR30  1
297 #define SR30 13
298 #define WR31  2
299 #define SR31 11

301 #define WR32 15
302 #define SR32  9
303 #define WR33  5
304 #define SR33  7
305 #define WR34  1
306 #define SR34 15
307 #define WR35  3
308 #define SR35 11
309 #define WR36  7
310 #define SR36  8
311 #define WR37 14
312 #define SR37  6
313 #define WR38  6
314 #define SR38  6
315 #define WR39  9
316 #define SR39 14
317 #define WR40 11
318 #define SR40 12
319 #define WR41  8
320 #define SR41 13
321 #define WR42 12
322 #define SR42  5
323 #define WR43  2
324 #define SR43 14
325 #define WR44 10
```



```
326 #define SR44 13
327 #define WR45  0
328 #define SR45 13
329 #define WR46  4
330 #define SR46  7
331 #define WR47 13
332 #define SR47  5

334 #define WR48  8
335 #define SR48 15
336 #define WR49  6
337 #define SR49  5
338 #define WR50  4
339 #define SR50  8
340 #define WR51  1
341 #define SR51 11
342 #define WR52  3
343 #define SR52 14
344 #define WR53 11
345 #define SR53 14
346 #define WR54 15
347 #define SR54  6
348 #define WR55  0
349 #define SR55 14
350 #define WR56  5
351 #define SR56  6
352 #define WR57 12
353 #define SR57  9
354 #define WR58  2
355 #define SR58 12
356 #define WR59 13
357 #define SR59  9
358 #define WR60  9
359 #define SR60 12
360 #define WR61  7
361 #define SR61  5
362 #define WR62 10
363 #define SR62 15
364 #define WR63 14
365 #define SR63  8

367 #define WR64 12
368 #define SR64  8
369 #define WR65 15
370 #define SR65  5
371 #define WR66 10
372 #define SR66 12
373 #define WR67  4
374 #define SR67  9
375 #define WR68  1
376 #define SR68 12
377 #define WR69  5
378 #define SR69  5
379 #define WR70  8
380 #define SR70 14
381 #define WR71  7
382 #define SR71  6
383 #define WR72  6
384 #define SR72  8
385 #define WR73  2
386 #define SR73 13
387 #define WR74 13
388 #define SR74  6
389 #define WR75 14
390 #define SR75  5
391 #define WR76  0
```

```
392 #define SR76 15
393 #define WR77  3
394 #define SR77 13
395 #define WR78  9
396 #define SR78 11
397 #define WR79 11
398 #define SR79 11
399 #endif /* ! codereview */
```

```
**********************************************************
    5602 Wed Aug 13 19:51:53 2014
new/usr/src/lib/openssl/include/rpc_des.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/des/rpc_des.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/*  @(#)des.h   2.2 88/08/10 4.0 RPCSRC; from 2.7 88/02/08 SMI  */
/*
 * Sun RPC is a product of Sun Microsystems, Inc. and is provided for
```



```
 * unrestricted use provided that this legend is included on all tape
 * media and as a part of the software program in whole or part.  Users
 * may copy or modify Sun RPC without charge, but are not authorized
 * to license or distribute it to anyone else except as part of a product or
 * program developed by the user.
 *
 * SUN RPC IS PROVIDED AS IS WITH NO WARRANTIES OF ANY KIND INCLUDING THE
 * WARRANTIES OF DESIGN, MERCHANTIBILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.
 *
 * Sun RPC is provided with no support and without any obligation on the
 * part of Sun Microsystems, Inc. to assist in its use, correction,
 * modification or enhancement.
 *
 * SUN MICROSYSTEMS, INC. SHALL HAVE NO LIABILITY WITH RESPECT TO THE
 * INFRINGEMENT OF COPYRIGHTS, TRADE SECRETS OR ANY PATENTS BY SUN RPC
 * OR ANY PART THEREOF.
 *
 * In no event will Sun Microsystems, Inc. be liable for any lost revenue
 * or profits or other special, indirect and consequential damages, even if
 * Sun has been advised of the possibility of such damages.
 *
 * Sun Microsystems, Inc.
 * 2550 Garcia Avenue
 * Mountain View, California  94043
 */
/*
 * Generic DES driver interface
 * Keep this file hardware independent!
 * Copyright (c) 1986 by Sun Microsystems, Inc.
 */

#define DES_MAXLEN      65536   /* maximum # of bytes to encrypt  */
#define DES_QUICKLEN    16      /* maximum # of bytes to encrypt quickly */

#ifdef HEADER_DES_H
#undef ENCRYPT
#undef DECRYPT
#endif

enum desdir { ENCRYPT, DECRYPT };
enum desmode { CBC, ECB };

/*
 * parameters to ioctl call
 */
struct desparams {
        unsigned char des_key[8];       /* key (with low bit parity) */
        enum desdir des_dir;    /* direction */
        enum desmode des_mode;  /* mode */
        unsigned char des_ivec[8];      /* input vector */
        unsigned des_len;       /* number of bytes to crypt */
        union {
                unsigned char UDES_data[DES_QUICKLEN];
                unsigned char *UDES_buf;
        } UDES;
#       define des_data UDES.UDES_data  /* direct data here if quick */
#       define des_buf  UDES.UDES_buf   /* otherwise, pointer to data */
};

/*
 * Encrypt an arbitrary sized buffer
 */
#define DESIOCBLOCK     _IOWR('d', 6, struct desparams)

/*
```

```
 * Encrypt of small amount of data, quickly
 */
#define DESIOCQUICK     _IOWR('d', 7, struct desparams)
#endif /* ! codereview */
```

```
**********************************************************
     178 Wed Aug 13 19:51:53 2014
new/usr/src/lib/openssl/include/rsa_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
extern int int_rsa_verify(int dtype, const unsigned char *m, unsigned int m_len,
                unsigned char *rm, size_t *prm_len,
                const unsigned char *sigbuf, size_t siglen,
                RSA *rsa);
#endif /* ! codereview */
```

```
**********************************************************
    4436 Wed Aug 13 19:51:53 2014
new/usr/src/lib/openssl/include/seed_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Copyright (c) 2007 KISA(Korea Information Security Agency). All rights reserv
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Neither the name of author nor the names of its contributors may
 *    be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY AUTHOR AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 */
#ifndef HEADER_SEED_LOCL_H
#define HEADER_SEED_LOCL_H

#include "openssl/e_os2.h"
#include <openssl/seed.h>


#ifdef SEED_LONG /* need 32-bit type */
typedef unsigned long seed_word;
#else
typedef unsigned int seed_word;
#endif


#ifdef  __cplusplus
extern "C" {
#endif

#define G_FUNC(v)       \
        SS[0][(unsigned char)      (v) & 0xff] ^ SS[1][(unsigned char) ((v)>>8)
        SS[2][(unsigned char)((v)>>16) & 0xff] ^ SS[3][(unsigned char)((v)>>24)

#define char2word(c, i)  \
        (i) = ((((seed_word)(c)[0]) << 24) | (((seed_word)(c)[1]) << 16) | (((se

#define word2char(l, c)  \
        *((c)+0) = (unsigned char)((l)>>24) & 0xff; \
        *((c)+1) = (unsigned char)((l)>>16) & 0xff; \
        *((c)+2) = (unsigned char)((l)>> 8) & 0xff; \
        *((c)+3) = (unsigned char)((l))     & 0xff

#define KEYSCHEDULE_UPDATE0(T0, T1, X1, X2, X3, X4, KC)  \
        (T0) = (X3);                                     \
        (X3) = (((X3)<<8) ^ ((X4)>>24)) & 0xffffffff;    \
        (X4) = (((X4)<<8) ^ ((T0)>>24)) & 0xffffffff;    \
        (T0) = ((X1) + (X3) - (KC))     & 0xffffffff;    \
```



```
        (T1) = ((X2) + (KC) - (X4))     & 0xffffffff

#define KEYSCHEDULE_UPDATE1(T0, T1, X1, X2, X3, X4, KC)  \
        (T0) = (X1);                                     \
        (X1) = (((X1)>>8) ^ ((X2)<<24)) & 0xffffffff;    \
        (X2) = (((X2)>>8) ^ ((T0)<<24)) & 0xffffffff;    \
        (T0) = ((X1) + (X3) - (KC))     & 0xffffffff;    \
        (T1) = ((X2) + (KC) - (X4))     & 0xffffffff

#define KEYUPDATE_TEMP(T0, T1, K)   \
        (K)[0] = G_FUNC((T0));      \
        (K)[1] = G_FUNC((T1))

#define XOR_SEEDBLOCK(DST, SRC)      \
        ((DST))[0] ^= ((SRC))[0];    \
        ((DST))[1] ^= ((SRC))[1];    \
        ((DST))[2] ^= ((SRC))[2];    \
        ((DST))[3] ^= ((SRC))[3]

#define MOV_SEEDBLOCK(DST, SRC)      \
        ((DST))[0] = ((SRC))[0];     \
        ((DST))[1] = ((SRC))[1];     \
        ((DST))[2] = ((SRC))[2];     \
        ((DST))[3] = ((SRC))[3]

# define CHAR2WORD(C, I)              \
        char2word((C),    (I)[0]);    \
        char2word((C+4),  (I)[1]);    \
        char2word((C+8),  (I)[2]);    \
        char2word((C+12), (I)[3])

# define WORD2CHAR(I, C)              \
        word2char((I)[0], (C));       \
        word2char((I)[1], (C+4));     \
        word2char((I)[2], (C+8));     \
        word2char((I)[3], (C+12))

# define E_SEED(T0, T1, X1, X2, X3, X4, rbase)   \
        (T0) = (X3) ^ (ks->data)[(rbase)];       \
        (T1) = (X4) ^ (ks->data)[(rbase)+1];     \
        (T1) ^= (T0);                            \
        (T1) = G_FUNC((T1));                     \
        (T0) = ((T0) + (T1)) & 0xffffffff;       \
        (T0) = G_FUNC((T0));                     \
        (T1) = ((T1) + (T0)) & 0xffffffff;       \
        (T1) = G_FUNC((T1));                     \
        (T0) = ((T0) + (T1)) & 0xffffffff;       \
        (X1) ^= (T0);                            \
        (X2) ^= (T1)

#ifdef  __cplusplus
}
#endif

#endif /* HEADER_SEED_LOCL_H */
#endif /* ! codereview */
```

```
**********************************************************
   15705 Wed Aug 13 19:51:53 2014
new/usr/src/lib/openssl/include/sha_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha_locl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdlib.h>
#include <string.h>
```



```
#include <openssl/opensslconf.h>
#include <openssl/sha.h>

#define DATA_ORDER_IS_BIG_ENDIAN

#define HASH_LONG               SHA_LONG
#define HASH_CTX                SHA_CTX
#define HASH_CBLOCK             SHA_CBLOCK
#define HASH_MAKE_STRING(c,s)   do {    \
        unsigned long ll;               \
        ll=(c)->h0; (void)HOST_l2c(ll,(s));     \
        ll=(c)->h1; (void)HOST_l2c(ll,(s));     \
        ll=(c)->h2; (void)HOST_l2c(ll,(s));     \
        ll=(c)->h3; (void)HOST_l2c(ll,(s));     \
        ll=(c)->h4; (void)HOST_l2c(ll,(s));     \
        } while (0)

#if defined(SHA_0)

# define HASH_UPDATE                    SHA_Update
# define HASH_TRANSFORM                 SHA_Transform
# define HASH_FINAL                     SHA_Final
# define HASH_INIT                      SHA_Init
# define HASH_BLOCK_DATA_ORDER          sha_block_data_order
# define Xupdate(a,ix,ia,ib,ic,id)      (ix=(a)=(ia^ib^ic^id))

static void sha_block_data_order (SHA_CTX *c, const void *p,size_t num);

#elif defined(SHA_1)

# define HASH_UPDATE                    SHA1_Update
# define HASH_TRANSFORM                 SHA1_Transform
# define HASH_FINAL                     SHA1_Final
# define HASH_INIT                      SHA1_Init
# define HASH_BLOCK_DATA_ORDER          sha1_block_data_order
# if defined(__MWERKS__) && defined(__MC68K__)
   /* Metrowerks for Motorola fails otherwise:-( <appro@fy.chalmers.se> */
#  define Xupdate(a,ix,ia,ib,ic,id)     do { (a)=(ia^ib^ic^id);         \
                                             ix=(a)=ROTATE((a),1);      \
                                        } while (0)
# else
#  define Xupdate(a,ix,ia,ib,ic,id)     ( (a)=(ia^ib^ic^id),    \
                                          ix=(a)=ROTATE((a),1)  \
                                        )
# endif

#ifndef SHA1_ASM
static
#endif
void sha1_block_data_order (SHA_CTX *c, const void *p,size_t num);

#else
# error "Either SHA_0 or SHA_1 must be defined."
#endif

#include "md32_common.h"

#define INIT_DATA_h0 0x67452301UL
#define INIT_DATA_h1 0xefcdab89UL
#define INIT_DATA_h2 0x98badcfeUL
#define INIT_DATA_h3 0x10325476UL
#define INIT_DATA_h4 0xc3d2e1f0UL

#ifdef SHA_0
fips_md_init(SHA)
#else
```

```
fips_md_init_ctx(SHA1, SHA)
#endif
        {
        memset (c,0,sizeof(*c));
        c->h0=INIT_DATA_h0;
        c->h1=INIT_DATA_h1;
        c->h2=INIT_DATA_h2;
        c->h3=INIT_DATA_h3;
        c->h4=INIT_DATA_h4;
        return 1;
        }

#define K_00_19 0x5a827999UL
#define K_20_39 0x6ed9eba1UL
#define K_40_59 0x8f1bbcdcUL
#define K_60_79 0xca62c1d6UL

/* As  pointed out by Wei Dai <weidai@eskimo.com>, F() below can be
 * simplified to the code in F_00_19.  Wei attributes these optimisations
 * to Peter Gutmann's SHS code, and he attributes it to Rich Schroeppel.
 * #define F(x,y,z) (((x) & (y))  |  ((~(x)) & (z)))
 * I've just become aware of another tweak to be made, again from Wei Dai,
 * in F_40_59, (x&a)|(y&a) -> (x|y)&a
 */
#define F_00_19(b,c,d)  ((((c) ^ (d)) & (b)) ^ (d))
#define F_20_39(b,c,d)  ((b) ^ (c) ^ (d))
#define F_40_59(b,c,d)  (((b) & (c)) | (((b)|(c)) & (d)))
#define F_60_79(b,c,d)  F_20_39(b,c,d)

#ifndef OPENSSL_SMALL_FOOTPRINT

#define BODY_00_15(i,a,b,c,d,e,f,xi) \
        (f)=xi+(e)+K_00_19+ROTATE((a),5)+F_00_19((b),(c),(d)); \
        (b)=ROTATE((b),30);

#define BODY_16_19(i,a,b,c,d,e,f,xi,xa,xb,xc,xd) \
        Xupdate(f,xi,xa,xb,xc,xd); \
        (f)+=(e)+K_00_19+ROTATE((a),5)+F_00_19((b),(c),(d)); \
        (b)=ROTATE((b),30);

#define BODY_20_31(i,a,b,c,d,e,f,xi,xa,xb,xc,xd) \
        Xupdate(f,xi,xa,xb,xc,xd); \
        (f)+=(e)+K_20_39+ROTATE((a),5)+F_20_39((b),(c),(d)); \
        (b)=ROTATE((b),30);

#define BODY_32_39(i,a,b,c,d,e,f,xa,xb,xc,xd) \
        Xupdate(f,xa,xa,xb,xc,xd); \
        (f)+=(e)+K_20_39+ROTATE((a),5)+F_20_39((b),(c),(d)); \
        (b)=ROTATE((b),30);

#define BODY_40_59(i,a,b,c,d,e,f,xa,xb,xc,xd) \
        Xupdate(f,xa,xa,xb,xc,xd); \
        (f)+=(e)+K_40_59+ROTATE((a),5)+F_40_59((b),(c),(d)); \
        (b)=ROTATE((b),30);

#define BODY_60_79(i,a,b,c,d,e,f,xa,xb,xc,xd) \
        Xupdate(f,xa,xa,xb,xc,xd); \
        (f)=xa+(e)+K_60_79+ROTATE((a),5)+F_60_79((b),(c),(d)); \
        (b)=ROTATE((b),30);

#ifdef X
#undef X
#endif
#ifndef MD32_XARRAY
  /*
   * Originally X was an array. As it's automatic it's natural
```



```
   * to expect RISC compiler to accomodate at least part of it in
   * the register bank, isn't it? Unfortunately not all compilers
   * "find" this expectation reasonable:-( On order to make such
   * compilers generate better code I replace X[] with a bunch of
   * X0, X1, etc. See the function body below...
   *                                      <appro@fy.chalmers.se>
   */
# define X(i)   XX##i
#else
  /*
   * However! Some compilers (most notably HP C) get overwhelmed by
   * that many local variables so that we have to have the way to
   * fall down to the original behavior.
   */
# define X(i)   XX[i]
#endif

#if !defined(SHA_1) || !defined(SHA1_ASM)
static void HASH_BLOCK_DATA_ORDER (SHA_CTX *c, const void *p, size_t num)
        {
        const unsigned char *data=p;
        register unsigned MD32_REG_T A,B,C,D,E,T,l;
#ifndef MD32_XARRAY
        unsigned MD32_REG_T     XX0, XX1, XX2, XX3, XX4, XX5, XX6, XX7,
                                XX8, XX9,XX10,XX11,XX12,XX13,XX14,XX15;
#else
        SHA_LONG        XX[16];
#endif

        A=c->h0;
        B=c->h1;
        C=c->h2;
        D=c->h3;
        E=c->h4;

        for (;;)
                {
        const union { long one; char little; } is_endian = {1};

        if (!is_endian.little && sizeof(SHA_LONG)==4 && ((size_t)p%4)==0)
                {
                const SHA_LONG *W=(const SHA_LONG *)data;

                X( 0) = W[0];                           X( 1) = W[ 1];
                BODY_00_15( 0,A,B,C,D,E,T,X( 0));       X( 2) = W[ 2];
                BODY_00_15( 1,T,A,B,C,D,E,X( 1));       X( 3) = W[ 3];
                BODY_00_15( 2,E,T,A,B,C,D,X( 2));       X( 4) = W[ 4];
                BODY_00_15( 3,D,E,T,A,B,C,X( 3));       X( 5) = W[ 5];
                BODY_00_15( 4,C,D,E,T,A,B,X( 4));       X( 6) = W[ 6];
                BODY_00_15( 5,B,C,D,E,T,A,X( 5));       X( 7) = W[ 7];
                BODY_00_15( 6,A,B,C,D,E,T,X( 6));       X( 8) = W[ 8];
                BODY_00_15( 7,T,A,B,C,D,E,X( 7));       X( 9) = W[ 9];
                BODY_00_15( 8,E,T,A,B,C,D,X( 8));       X(10) = W[10];
                BODY_00_15( 9,D,E,T,A,B,C,X( 9));       X(11) = W[11];
                BODY_00_15(10,C,D,E,T,A,B,X(10));       X(12) = W[12];
                BODY_00_15(11,B,C,D,E,T,A,X(11));       X(13) = W[13];
                BODY_00_15(12,A,B,C,D,E,T,X(12));       X(14) = W[14];
                BODY_00_15(13,T,A,B,C,D,E,X(13));       X(15) = W[15];
                BODY_00_15(14,E,T,A,B,C,D,X(14));
                BODY_00_15(15,D,E,T,A,B,C,X(15));

                data += SHA_CBLOCK;
                }
        else
                {
                (void)HOST_c2l(data,l); X( 0)=l;        (void)HOST_c2l(data,l);
```

```
260                 BODY_00_15( 0,A,B,C,D,E,T,X( 0));       (void)HOST_c2l(data,l);
261                 BODY_00_15( 1,T,A,B,C,D,E,X( 1));       (void)HOST_c2l(data,l);
262                 BODY_00_15( 2,E,T,A,B,C,D,X( 2));       (void)HOST_c2l(data,l);
263                 BODY_00_15( 3,D,E,T,A,B,C,X( 3));       (void)HOST_c2l(data,l);
264                 BODY_00_15( 4,C,D,E,T,A,B,X( 4));       (void)HOST_c2l(data,l);
265                 BODY_00_15( 5,B,C,D,E,T,A,X( 5));       (void)HOST_c2l(data,l);
266                 BODY_00_15( 6,A,B,C,D,E,T,X( 6));       (void)HOST_c2l(data,l);
267                 BODY_00_15( 7,T,A,B,C,D,E,X( 7));       (void)HOST_c2l(data,l);
268                 BODY_00_15( 8,E,T,A,B,C,D,X( 8));       (void)HOST_c2l(data,l);
269                 BODY_00_15( 9,D,E,T,A,B,C,X( 9));       (void)HOST_c2l(data,l);
270                 BODY_00_15(10,C,D,E,T,A,B,X(10));       (void)HOST_c2l(data,l);
271                 BODY_00_15(11,B,C,D,E,T,A,X(11));       (void)HOST_c2l(data,l);
272                 BODY_00_15(12,A,B,C,D,E,T,X(12));       (void)HOST_c2l(data,l);
273                 BODY_00_15(13,T,A,B,C,D,E,X(13));       (void)HOST_c2l(data,l);
274                 BODY_00_15(14,E,T,A,B,C,D,X(14));
275                 BODY_00_15(15,D,E,T,A,B,C,X(15));
276                 }

278         BODY_16_19(16,C,D,E,T,A,B,X( 0),X( 0),X( 2),X( 8),X(13));
279         BODY_16_19(17,B,C,D,E,T,A,X( 1),X( 1),X( 3),X( 9),X(14));
280         BODY_16_19(18,A,B,C,D,E,T,X( 2),X( 2),X( 4),X(10),X(15));
281         BODY_16_19(19,T,A,B,C,D,E,X( 3),X( 3),X( 5),X(11),X( 0));

283         BODY_20_31(20,E,T,A,B,C,D,X( 4),X( 4),X( 6),X(12),X( 1));
284         BODY_20_31(21,D,E,T,A,B,C,X( 5),X( 5),X( 7),X(13),X( 2));
285         BODY_20_31(22,C,D,E,T,A,B,X( 6),X( 6),X( 8),X(14),X( 3));
286         BODY_20_31(23,B,C,D,E,T,A,X( 7),X( 7),X( 9),X(15),X( 4));
287         BODY_20_31(24,A,B,C,D,E,T,X( 8),X( 8),X(10),X( 0),X( 5));
288         BODY_20_31(25,T,A,B,C,D,E,X( 9),X( 9),X(11),X( 1),X( 6));
289         BODY_20_31(26,E,T,A,B,C,D,X(10),X(10),X(12),X( 2),X( 7));
290         BODY_20_31(27,D,E,T,A,B,C,X(11),X(11),X(13),X( 3),X( 8));
291         BODY_20_31(28,C,D,E,T,A,B,X(12),X(12),X(14),X( 4),X( 9));
292         BODY_20_31(29,B,C,D,E,T,A,X(13),X(13),X(15),X( 5),X(10));
293         BODY_20_31(30,A,B,C,D,E,T,X(14),X(14),X( 0),X( 6),X(11));
294         BODY_20_31(31,T,A,B,C,D,E,X(15),X(15),X( 1),X( 7),X(12));

296         BODY_32_39(32,E,T,A,B,C,D,X( 0),X( 2),X( 8),X(13));
297         BODY_32_39(33,D,E,T,A,B,C,X( 1),X( 3),X( 9),X(14));
298         BODY_32_39(34,C,D,E,T,A,B,X( 2),X( 4),X(10),X(15));
299         BODY_32_39(35,B,C,D,E,T,A,X( 3),X( 5),X(11),X( 0));
300         BODY_32_39(36,A,B,C,D,E,T,X( 4),X( 6),X(12),X( 1));
301         BODY_32_39(37,T,A,B,C,D,E,X( 5),X( 7),X(13),X( 2));
302         BODY_32_39(38,E,T,A,B,C,D,X( 6),X( 8),X(14),X( 3));
303         BODY_32_39(39,D,E,T,A,B,C,X( 7),X( 9),X(15),X( 4));

305         BODY_40_59(40,C,D,E,T,A,B,X( 8),X(10),X( 0),X( 5));
306         BODY_40_59(41,B,C,D,E,T,A,X( 9),X(11),X( 1),X( 6));
307         BODY_40_59(42,A,B,C,D,E,T,X(10),X(12),X( 2),X( 7));
308         BODY_40_59(43,T,A,B,C,D,E,X(11),X(13),X( 3),X( 8));
309         BODY_40_59(44,E,T,A,B,C,D,X(12),X(14),X( 4),X( 9));
310         BODY_40_59(45,D,E,T,A,B,C,X(13),X(15),X( 5),X(10));
311         BODY_40_59(46,C,D,E,T,A,B,X(14),X( 0),X( 6),X(11));
312         BODY_40_59(47,B,C,D,E,T,A,X(15),X( 1),X( 7),X(12));
313         BODY_40_59(48,A,B,C,D,E,T,X( 0),X( 2),X( 8),X(13));
314         BODY_40_59(49,T,A,B,C,D,E,X( 1),X( 3),X( 9),X(14));
315         BODY_40_59(50,E,T,A,B,C,D,X( 2),X( 4),X(10),X(15));
316         BODY_40_59(51,D,E,T,A,B,C,X( 3),X( 5),X(11),X( 0));
317         BODY_40_59(52,C,D,E,T,A,B,X( 4),X( 6),X(12),X( 1));
318         BODY_40_59(53,B,C,D,E,T,A,X( 5),X( 7),X(13),X( 2));
319         BODY_40_59(54,A,B,C,D,E,T,X( 6),X( 8),X(14),X( 3));
320         BODY_40_59(55,T,A,B,C,D,E,X( 7),X( 9),X(15),X( 4));
321         BODY_40_59(56,E,T,A,B,C,D,X( 8),X(10),X( 0),X( 5));
322         BODY_40_59(57,D,E,T,A,B,C,X( 9),X(11),X( 1),X( 6));
323         BODY_40_59(58,C,D,E,T,A,B,X(10),X(12),X( 2),X( 7));
324         BODY_40_59(59,B,C,D,E,T,A,X(11),X(13),X( 3),X( 8));
```



```
326         BODY_60_79(60,A,B,C,D,E,T,X(12),X(14),X( 4),X( 9));
327         BODY_60_79(61,T,A,B,C,D,E,X(13),X(15),X( 5),X(10));
328         BODY_60_79(62,E,T,A,B,C,D,X(14),X( 0),X( 6),X(11));
329         BODY_60_79(63,D,E,T,A,B,C,X(15),X( 1),X( 7),X(12));
330         BODY_60_79(64,C,D,E,T,A,B,X( 0),X( 2),X( 8),X(13));
331         BODY_60_79(65,B,C,D,E,T,A,X( 1),X( 3),X( 9),X(14));
332         BODY_60_79(66,A,B,C,D,E,T,X( 2),X( 4),X(10),X(15));
333         BODY_60_79(67,T,A,B,C,D,E,X( 3),X( 5),X(11),X( 0));
334         BODY_60_79(68,E,T,A,B,C,D,X( 4),X( 6),X(12),X( 1));
335         BODY_60_79(69,D,E,T,A,B,C,X( 5),X( 7),X(13),X( 2));
336         BODY_60_79(70,C,D,E,T,A,B,X( 6),X( 8),X(14),X( 3));
337         BODY_60_79(71,B,C,D,E,T,A,X( 7),X( 9),X(15),X( 4));
338         BODY_60_79(72,A,B,C,D,E,T,X( 8),X(10),X( 0),X( 5));
339         BODY_60_79(73,T,A,B,C,D,E,X( 9),X(11),X( 1),X( 6));
340         BODY_60_79(74,E,T,A,B,C,D,X(10),X(12),X( 2),X( 7));
341         BODY_60_79(75,D,E,T,A,B,C,X(11),X(13),X( 3),X( 8));
342         BODY_60_79(76,C,D,E,T,A,B,X(12),X(14),X( 4),X( 9));
343         BODY_60_79(77,B,C,D,E,T,A,X(13),X(15),X( 5),X(10));
344         BODY_60_79(78,A,B,C,D,E,T,X(14),X( 0),X( 6),X(11));
345         BODY_60_79(79,T,A,B,C,D,E,X(15),X( 1),X( 7),X(12));

347         c->h0=(c->h0+E)&0xffffffffL;
348         c->h1=(c->h1+T)&0xffffffffL;
349         c->h2=(c->h2+A)&0xffffffffL;
350         c->h3=(c->h3+B)&0xffffffffL;
351         c->h4=(c->h4+C)&0xffffffffL;

353         if (--num == 0) break;

355         A=c->h0;
356         B=c->h1;
357         C=c->h2;
358         D=c->h3;
359         E=c->h4;

361                         }
362         }
363 #endif

365 #else   /* OPENSSL_SMALL_FOOTPRINT */

367 #define BODY_00_15(xi)           do {   \
368         T=E+K_00_19+F_00_19(B,C,D);     \
369         E=D, D=C, C=ROTATE(B,30), B=A;  \
370         A=ROTATE(A,5)+T+xi;         } while(0)

372 #define BODY_16_19(xa,xb,xc,xd)  do {   \
373         Xupdate(T,xa,xa,xb,xc,xd);      \
374         T+=E+K_00_19+F_00_19(B,C,D);    \
375         E=D, D=C, C=ROTATE(B,30), B=A;  \
376         A=ROTATE(A,5)+T;            } while(0)

378 #define BODY_20_39(xa,xb,xc,xd)  do {   \
379         Xupdate(T,xa,xa,xb,xc,xd);      \
380         T+=E+K_20_39+F_20_39(B,C,D);    \
381         E=D, D=C, C=ROTATE(B,30), B=A;  \
382         A=ROTATE(A,5)+T;            } while(0)

384 #define BODY_40_59(xa,xb,xc,xd)  do {   \
385         Xupdate(T,xa,xa,xb,xc,xd);      \
386         T+=E+K_40_59+F_40_59(B,C,D);    \
387         E=D, D=C, C=ROTATE(B,30), B=A;  \
388         A=ROTATE(A,5)+T;            } while(0)

390 #define BODY_60_79(xa,xb,xc,xd)  do {   \
391         Xupdate(T,xa,xa,xb,xc,xd);      \
```

```
392         T=E+K_60_79+F_60_79(B,C,D);     \
393         E=D, D=C, C=ROTATE(B,30), B=A;  \
394         A=ROTATE(A,5)+T+xa;         } while(0)

396 #if !defined(SHA_1) || !defined(SHA1_ASM)
397 static void HASH_BLOCK_DATA_ORDER (SHA_CTX *c, const void *p, size_t num)
398         {
399         const unsigned char *data=p;
400         register unsigned MD32_REG_T A,B,C,D,E,T,l;
401         int i;
402         SHA_LONG        X[16];

404         A=c->h0;
405         B=c->h1;
406         C=c->h2;
407         D=c->h3;
408         E=c->h4;

410         for (;;)
411                 {
412         for (i=0;i<16;i++)
413         { HOST_c2l(data,l); X[i]=l; BODY_00_15(X[i]); }
414         for (i=0;i<4;i++)
415         { BODY_16_19(X[i],       X[i+2],      X[i+8],     X[(i+13)&15]); }
416         for (;i<24;i++)
417         { BODY_20_39(X[i&15],    X[(i+2)&15], X[(i+8)&15],X[(i+13)&15]); }
418         for (i=0;i<20;i++)
419         { BODY_40_59(X[(i+8)&15],X[(i+10)&15],X[i&15],    X[(i+5)&15]);  }
420         for (i=4;i<24;i++)
421         { BODY_60_79(X[(i+8)&15],X[(i+10)&15],X[i&15],    X[(i+5)&15]);  }

423         c->h0=(c->h0+A)&0xffffffffL;
424         c->h1=(c->h1+B)&0xffffffffL;
425         c->h2=(c->h2+C)&0xffffffffL;
426         c->h3=(c->h3+D)&0xffffffffL;
427         c->h4=(c->h4+E)&0xffffffffL;

429         if (--num == 0) break;

431         A=c->h0;
432         B=c->h1;
433         C=c->h2;
434         D=c->h3;
435         E=c->h4;

437                 }
438         }
439 #endif

441 #endif
442 #endif /* ! codereview */
```

```
**********************************************************
   10048 Wed Aug 13 19:51:54 2014
new/usr/src/lib/openssl/include/spr.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/spr.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

OPENSSL_GLOBAL const DES_LONG DES_SPtrans[8][64]={
{
/* nibble 0 */
```



```
0x02080800L, 0x00080000L, 0x02000002L, 0x02080802L,
0x02000000L, 0x00080802L, 0x00080002L, 0x02000002L,
0x00080802L, 0x02080800L, 0x02080000L, 0x00000802L,
0x02000802L, 0x02000000L, 0x00000000L, 0x00080002L,
0x00080000L, 0x00000002L, 0x02000800L, 0x00080800L,
0x02080802L, 0x02080000L, 0x00000802L, 0x02000800L,
0x00000002L, 0x00000800L, 0x00080800L, 0x02080002L,
0x00000800L, 0x02000802L, 0x02080002L, 0x00000000L,
0x00000000L, 0x02080802L, 0x02000800L, 0x00080002L,
0x02080800L, 0x00080000L, 0x00000802L, 0x02000800L,
0x02080002L, 0x00000800L, 0x00080800L, 0x02000002L,
0x00080802L, 0x00000002L, 0x02000002L, 0x02080000L,
0x02080802L, 0x00080800L, 0x02080000L, 0x02000802L,
0x02000000L, 0x00000802L, 0x00080002L, 0x00000000L,
0x00080000L, 0x02000000L, 0x02000802L, 0x02080800L,
0x00000002L, 0x02080002L, 0x00000800L, 0x00080802L,
},{
/* nibble 1 */
0x40108010L, 0x00000000L, 0x00108000L, 0x40100000L,
0x40000010L, 0x00008010L, 0x40008000L, 0x00108000L,
0x00008000L, 0x40100010L, 0x00000010L, 0x40008000L,
0x00100010L, 0x40108000L, 0x40100000L, 0x00000010L,
0x00100000L, 0x40008010L, 0x40100010L, 0x00008000L,
0x00108010L, 0x40000000L, 0x00000000L, 0x00100010L,
0x40008010L, 0x00108010L, 0x40108000L, 0x40000010L,
0x40000000L, 0x00100000L, 0x00008010L, 0x40108010L,
0x00100010L, 0x40108000L, 0x40008000L, 0x00108010L,
0x40108010L, 0x00100010L, 0x40000010L, 0x00000000L,
0x40000000L, 0x00008010L, 0x00100000L, 0x40100010L,
0x00008000L, 0x40000000L, 0x00108010L, 0x40008010L,
0x40108000L, 0x00008000L, 0x00000000L, 0x40000010L,
0x00000010L, 0x40108010L, 0x00108000L, 0x40100000L,
0x40100010L, 0x00100000L, 0x00008010L, 0x40008000L,
0x40008010L, 0x00000010L, 0x40100000L, 0x00108000L,
},{
/* nibble 2 */
0x04000001L, 0x04040100L, 0x00000100L, 0x04000101L,
0x00040001L, 0x04000000L, 0x04000101L, 0x00040100L,
0x04000100L, 0x00040000L, 0x04040000L, 0x00000001L,
0x04040101L, 0x00000101L, 0x00000001L, 0x04040001L,
0x00000000L, 0x00040001L, 0x04040100L, 0x00000100L,
0x00000101L, 0x04040101L, 0x00040000L, 0x04000001L,
0x04040001L, 0x04000100L, 0x00040101L, 0x04040000L,
0x00040100L, 0x00000000L, 0x04000000L, 0x00040101L,
0x04040100L, 0x00000100L, 0x00000001L, 0x00040000L,
0x00000101L, 0x00040001L, 0x04040000L, 0x04000101L,
0x00000000L, 0x04040100L, 0x00040100L, 0x04040001L,
0x00040001L, 0x04000000L, 0x04040101L, 0x00000001L,
0x00040101L, 0x04000001L, 0x04000000L, 0x04040101L,
0x00040000L, 0x04000100L, 0x04000101L, 0x00040100L,
0x04000100L, 0x00000000L, 0x04040001L, 0x00000101L,
0x04000001L, 0x00040101L, 0x00000100L, 0x04040000L,
},{
/* nibble 3 */
0x00401008L, 0x10001000L, 0x00000008L, 0x10401008L,
0x00000000L, 0x10400000L, 0x10001008L, 0x00400008L,
0x10401000L, 0x10000008L, 0x10000000L, 0x00001008L,
0x10000008L, 0x00401008L, 0x00400000L, 0x10000000L,
0x10400008L, 0x00401000L, 0x00001000L, 0x00000008L,
0x00401000L, 0x10001008L, 0x10400000L, 0x00001000L,
0x00001008L, 0x00000000L, 0x00400008L, 0x10401000L,
0x10001000L, 0x10400008L, 0x10401008L, 0x00400000L,
0x10400008L, 0x00001008L, 0x00400000L, 0x10000008L,
0x00401000L, 0x10001000L, 0x00000008L, 0x10400000L,
0x10001008L, 0x00000000L, 0x00001000L, 0x00400008L,
0x00000000L, 0x10400008L, 0x10401000L, 0x00001000L,
```

```
128 0x10000000L, 0x10401008L, 0x00401008L, 0x00400000L,
129 0x10401008L, 0x00000008L, 0x10001000L, 0x00401008L,
130 0x00400008L, 0x00401000L, 0x10400000L, 0x10001008L,
131 0x00001008L, 0x10000000L, 0x10000008L, 0x10401000L,
132 },{
133 /* nibble 4 */
134 0x08000000L, 0x00010000L, 0x00000400L, 0x08010420L,
135 0x08010020L, 0x08000400L, 0x00010420L, 0x08010000L,
136 0x00010000L, 0x00000020L, 0x08000020L, 0x00010400L,
137 0x08000420L, 0x08010020L, 0x08010400L, 0x00000000L,
138 0x00010400L, 0x08000000L, 0x00010020L, 0x00000420L,
139 0x08000400L, 0x00010420L, 0x00000000L, 0x08000020L,
140 0x00000020L, 0x08000420L, 0x08010420L, 0x00010020L,
141 0x08010000L, 0x00000400L, 0x00000420L, 0x08010400L,
142 0x08010400L, 0x08000420L, 0x00010020L, 0x08010000L,
143 0x00010000L, 0x00000020L, 0x08000020L, 0x08000400L,
144 0x08000000L, 0x00010400L, 0x08010420L, 0x00000000L,
145 0x00010420L, 0x08000000L, 0x00000400L, 0x00010020L,
146 0x08000420L, 0x00000400L, 0x00000000L, 0x08010420L,
147 0x08010020L, 0x08010400L, 0x00000420L, 0x00010000L,
148 0x00010400L, 0x08010020L, 0x08000400L, 0x00000420L,
149 0x00000020L, 0x00010420L, 0x08010000L, 0x08000020L,
150 },{
151 /* nibble 5 */
152 0x80000040L, 0x00200040L, 0x00000000L, 0x80202000L,
153 0x00200040L, 0x00002000L, 0x80002040L, 0x00200000L,
154 0x00002040L, 0x80202040L, 0x00202000L, 0x80000000L,
155 0x80002000L, 0x80000040L, 0x80200000L, 0x00202040L,
156 0x00200000L, 0x80002040L, 0x80200040L, 0x00000000L,
157 0x00002000L, 0x00000040L, 0x80202000L, 0x80200040L,
158 0x80202040L, 0x80200000L, 0x80000000L, 0x00002040L,
159 0x00000040L, 0x00202000L, 0x00202040L, 0x80002000L,
160 0x00002040L, 0x80000000L, 0x80002000L, 0x00202040L,
161 0x80202000L, 0x00200040L, 0x00000000L, 0x80002000L,
162 0x80000000L, 0x00002000L, 0x80200040L, 0x00200000L,
163 0x00200040L, 0x80202040L, 0x00202000L, 0x00000040L,
164 0x80202040L, 0x00202000L, 0x00200000L, 0x80002040L,
165 0x80000040L, 0x80200000L, 0x00202040L, 0x00000000L,
166 0x00002000L, 0x80000040L, 0x80002040L, 0x80202000L,
167 0x80200000L, 0x00002040L, 0x00000040L, 0x80200040L,
168 },{
169 /* nibble 6 */
170 0x00004000L, 0x00000200L, 0x01000200L, 0x01000004L,
171 0x01004204L, 0x00004004L, 0x00004200L, 0x00000000L,
172 0x01000000L, 0x01000204L, 0x00000204L, 0x01004000L,
173 0x00000004L, 0x01004200L, 0x01004000L, 0x00000204L,
174 0x01000204L, 0x00004000L, 0x00004004L, 0x01004204L,
175 0x00000000L, 0x01000200L, 0x01000004L, 0x00004200L,
176 0x01004004L, 0x00004204L, 0x01004200L, 0x00000004L,
177 0x00004204L, 0x01004004L, 0x00000200L, 0x01000000L,
178 0x00004204L, 0x01004000L, 0x01004004L, 0x00000204L,
179 0x00004000L, 0x00000200L, 0x01000000L, 0x01004004L,
180 0x01000204L, 0x00004204L, 0x00004200L, 0x00000000L,
181 0x00000200L, 0x01000004L, 0x00000004L, 0x01000200L,
182 0x00000000L, 0x01000204L, 0x01000200L, 0x00004200L,
183 0x00000204L, 0x00004000L, 0x01004204L, 0x01000000L,
184 0x01004200L, 0x00000004L, 0x00004004L, 0x01004204L,
185 0x01000004L, 0x01004200L, 0x01004000L, 0x00004004L,
186 },{
187 /* nibble 7 */
188 0x20800080L, 0x20820000L, 0x00020080L, 0x00000000L,
189 0x20020000L, 0x00800080L, 0x20800000L, 0x20820080L,
190 0x00000080L, 0x20000000L, 0x00820000L, 0x00020080L,
191 0x00820080L, 0x20020080L, 0x20000080L, 0x20800000L,
192 0x00020000L, 0x00820080L, 0x00800080L, 0x20020000L,
193 0x20820080L, 0x20000080L, 0x00000000L, 0x00820000L,
```



```
194 0x20000000L, 0x00800000L, 0x20020080L, 0x20800080L,
195 0x00800000L, 0x00020000L, 0x20820000L, 0x00000080L,
196 0x00800000L, 0x00020000L, 0x20000080L, 0x20820080L,
197 0x00020080L, 0x20000000L, 0x00000000L, 0x00820000L,
198 0x20800080L, 0x20020080L, 0x20020000L, 0x00800080L,
199 0x20820000L, 0x00000080L, 0x00800080L, 0x20020000L,
200 0x20820080L, 0x00800000L, 0x20800000L, 0x20000080L,
201 0x00820000L, 0x00020080L, 0x20020080L, 0x20800000L,
202 0x00000080L, 0x20820000L, 0x00820080L, 0x00000000L,
203 0x20000000L, 0x20800080L, 0x00020000L, 0x00820080L,
204 }};
205 #endif /* ! codereview */
```

```
**********************************************************
   18750 Wed Aug 13 19:51:54 2014
new/usr/src/lib/openssl/include/srp_grps.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* start of generated data */

static BN_ULONG bn_group_1024_value[] = {
        bn_pack4(0x9FC6,0x1D2F,0xC0EB,0x06E3),
        bn_pack4(0xFD51,0x38FE,0x8376,0x435B),
        bn_pack4(0x2FD4,0xCBF4,0x976E,0xAA9A),
        bn_pack4(0x68ED,0xBC3C,0x0572,0x6CC0),
        bn_pack4(0xC529,0xF566,0x660E,0x57EC),
        bn_pack4(0x8255,0x9B29,0x7BCF,0x1885),
        bn_pack4(0xCE8E,0xF4AD,0x69B1,0x5D49),
        bn_pack4(0x5DC7,0xD7B4,0x6154,0xD6B6),
        bn_pack4(0x8E49,0x5C1D,0x6089,0xDAD1),
        bn_pack4(0xE0D5,0xD8E2,0x50B9,0x8BE4),
        bn_pack4(0x383B,0x4813,0xD692,0xC6E0),
        bn_pack4(0xD674,0xDF74,0x96EA,0x81D3),
        bn_pack4(0x9EA2,0x314C,0x9C25,0x6576),
        bn_pack4(0x6072,0x6187,0x75FF,0x3C0B),
        bn_pack4(0x9C33,0xF80A,0xFA8F,0xC5E8),
        bn_pack4(0xEEAF,0x0AB9,0xADB3,0x8DD6)
};
static BIGNUM bn_group_1024 = {
        bn_group_1024_value,
        (sizeof bn_group_1024_value)/sizeof(BN_ULONG),
        (sizeof bn_group_1024_value)/sizeof(BN_ULONG),
        0,
        BN_FLG_STATIC_DATA
};

static BN_ULONG bn_group_1536_value[] = {
        bn_pack4(0xCF76,0xE3FE,0xD135,0xF9BB),
        bn_pack4(0x1518,0x0F93,0x499A,0x234D),
        bn_pack4(0x8CE7,0xA28C,0x2442,0xC6F3),
        bn_pack4(0x5A02,0x1FFF,0x5E91,0x479E),
        bn_pack4(0x7F8A,0x2FE9,0xB8B5,0x292E),
        bn_pack4(0x837C,0x264A,0xE3A9,0xBEB8),
        bn_pack4(0xE442,0x734A,0xF7CC,0xB7AE),
        bn_pack4(0x6577,0x2E43,0x7D6C,0x7F8C),
        bn_pack4(0xDB2F,0xD53D,0x24B7,0xC486),
        bn_pack4(0x6EDF,0x0195,0x3934,0x9627),
        bn_pack4(0x158B,0xFD3E,0x2B9C,0x8CF5),
        bn_pack4(0x764E,0x3F4B,0x53DD,0x9DA1),
        bn_pack4(0x4754,0x8381,0xDBC5,0xB1FC),
        bn_pack4(0x9B60,0x9E0B,0xE3BA,0xB63D),
        bn_pack4(0x8134,0xB1C8,0xB979,0x8914),
        bn_pack4(0xDF02,0x8A7C,0xEC67,0xF0D0),
        bn_pack4(0x80B6,0x55BB,0x9A22,0xE8DC),
        bn_pack4(0x1558,0x903B,0xA0D0,0xF843),
        bn_pack4(0x51C6,0xA94B,0xE460,0x7A29),
        bn_pack4(0x5F4F,0x5F55,0x6E27,0xCBDE),
        bn_pack4(0xBEEE,0xA961,0x4B19,0xCC4D),
        bn_pack4(0xDBA5,0x1DF4,0x99AC,0x4C80),
        bn_pack4(0xB1F1,0x2A86,0x17A4,0x7BBB),
        bn_pack4(0x9DEF,0x3CAF,0xB939,0x277A)
};
static BIGNUM bn_group_1536 = {
        bn_group_1536_value,
        (sizeof bn_group_1536_value)/sizeof(BN_ULONG),
        (sizeof bn_group_1536_value)/sizeof(BN_ULONG),
        0,
        BN_FLG_STATIC_DATA
};
```



```
static BN_ULONG bn_group_2048_value[] = {
        bn_pack4(0x0FA7,0x111F,0x9E4A,0xFF73),
        bn_pack4(0x9B65,0xE372,0xFCD6,0x8EF2),
        bn_pack4(0x35DE,0x236D,0x525F,0x5475),
        bn_pack4(0x94B5,0xC803,0xD89F,0x7AE4),
        bn_pack4(0x71AE,0x35F8,0xE9DB,0xFBB6),
        bn_pack4(0x2A56,0x98F3,0xA8D0,0xC382),
        bn_pack4(0x9CCC,0x041C,0x7BC3,0x08D8),
        bn_pack4(0xAF87,0x4E73,0x03CE,0x5329),
        bn_pack4(0x6160,0x2790,0x04E5,0x7AE6),
        bn_pack4(0x032C,0xFBDB,0xF52F,0xB378),
        bn_pack4(0x5EA7,0x7A27,0x75D2,0xECFA),
        bn_pack4(0x5445,0x23B5,0x24B0,0xD57D),
        bn_pack4(0x5B9D,0x32E6,0x88F8,0x7748),
        bn_pack4(0xF1D2,0xB907,0x8717,0x461A),
        bn_pack4(0x76BD,0x207A,0x436C,0x6481),
        bn_pack4(0xCA97,0xB43A,0x23FB,0x8016),
        bn_pack4(0x1D28,0x1E44,0x6B14,0x773B),
        bn_pack4(0x7359,0xD041,0xD5C3,0x3EA7),
        bn_pack4(0xA80D,0x740A,0xDBF4,0xFF74),
        bn_pack4(0x55F9,0x7993,0xEC97,0x5EEA),
        bn_pack4(0x2918,0xA996,0x2F0B,0x93B8),
        bn_pack4(0x661A,0x05FB,0xD5FA,0xAAE8),
        bn_pack4(0xCF60,0x9517,0x9A16,0x3AB3),
        bn_pack4(0xE808,0x3969,0xEDB7,0x67B0),
        bn_pack4(0xCD7F,0x48A9,0xDA04,0xFD50),
        bn_pack4(0xD523,0x12AB,0x4B03,0x310D),
        bn_pack4(0x8193,0xE075,0x7767,0xA13D),
        bn_pack4(0xA373,0x29CB,0xB4A0,0x99ED),
        bn_pack4(0xFC31,0x9294,0x3DB5,0x6050),
        bn_pack4(0xAF72,0xB665,0x1987,0xEE07),
        bn_pack4(0xF166,0xDE5E,0x1389,0x582F),
        bn_pack4(0xAC6B,0xDB41,0x324A,0x9A9B)
};
static BIGNUM bn_group_2048 = {
        bn_group_2048_value,
        (sizeof bn_group_2048_value)/sizeof(BN_ULONG),
        (sizeof bn_group_2048_value)/sizeof(BN_ULONG),
        0,
        BN_FLG_STATIC_DATA
};

static BN_ULONG bn_group_3072_value[] = {
        bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF),
        bn_pack4(0x4B82,0xD120,0xA93A,0xD2CA),
        bn_pack4(0x43DB,0x5BFC,0xE0FD,0x108E),
        bn_pack4(0x08E2,0x4FA0,0x74E5,0xAB31),
        bn_pack4(0x7709,0x88C0,0xBAD9,0x46E2),
        bn_pack4(0xBBE1,0x1757,0x7A61,0x5D6C),
        bn_pack4(0x521F,0x2B18,0x177B,0x200C),
        bn_pack4(0xD876,0x0273,0x3EC8,0x6A64),
        bn_pack4(0xF12F,0xFA06,0xD98A,0x0864),
        bn_pack4(0xCEE3,0xD226,0x1AD2,0xEE6B),
        bn_pack4(0x1E8C,0x94E0,0x4A25,0x619D),
        bn_pack4(0xABF5,0xAE8C,0xDB09,0x33D7),
        bn_pack4(0xB397,0x0F85,0xA6E1,0xE4C7),
        bn_pack4(0x8AEA,0x7157,0x5D06,0x0C7D),
        bn_pack4(0xECFB,0x8504,0x58DB,0xEF0A),
        bn_pack4(0xA855,0x21AB,0xDF1C,0xBA64),
        bn_pack4(0xAD33,0x170D,0x0450,0x7A33),
        bn_pack4(0x1572,0x8E5A,0x8AAA,0xC42D),
        bn_pack4(0x15D2,0x2618,0x98FA,0x0510),
        bn_pack4(0x3995,0x497C,0xEA95,0x6AE5),
        bn_pack4(0xDE2B,0xCBF6,0x9558,0x1718),
        bn_pack4(0xB5C5,0x5DF0,0x6F4C,0x52C9),
```

```
128         bn_pack4(0x9B27,0x83A2,0xEC07,0xA28F),
129         bn_pack4(0xE39E,0x772C,0x180E,0x8603),
130         bn_pack4(0x3290,0x5E46,0x2E36,0xCE3B),
131         bn_pack4(0xF174,0x6C08,0xCA18,0x217C),
132         bn_pack4(0x670C,0x354E,0x4ABC,0x9804),
133         bn_pack4(0x9ED5,0x2907,0x7096,0x966D),
134         bn_pack4(0x1C62,0xF356,0x2085,0x52BB),
135         bn_pack4(0x8365,0x5D23,0xDCA3,0xAD96),
136         bn_pack4(0x6916,0x3FA8,0xFD24,0xCF5F),
137         bn_pack4(0x98DA,0x4836,0x1C55,0xD39A),
138         bn_pack4(0xC200,0x7CB8,0xA163,0xBF05),
139         bn_pack4(0x4928,0x6651,0xECE4,0x5B3D),
140         bn_pack4(0xAE9F,0x2411,0x7C4B,0x1FE6),
141         bn_pack4(0xEE38,0x6BFB,0x5A89,0x9FA5),
142         bn_pack4(0x0BFF,0x5CB6,0xF406,0xB7ED),
143         bn_pack4(0xF44C,0x42E9,0xA637,0xED6B),
144         bn_pack4(0xE485,0xB576,0x625E,0x7EC6),
145         bn_pack4(0x4FE1,0x356D,0x6D51,0xC245),
146         bn_pack4(0x302B,0x0A6D,0xF25F,0x1437),
147         bn_pack4(0xEF95,0x19B3,0xCD3A,0x431B),
148         bn_pack4(0x514A,0x0879,0x8E34,0x04DD),
149         bn_pack4(0x020B,0xBEA6,0x3B13,0x9B22),
150         bn_pack4(0x2902,0x4E08,0x8A67,0xCC74),
151         bn_pack4(0xC4C6,0x628B,0x80DC,0x1CD1),
152         bn_pack4(0xC90F,0xDAA2,0x2168,0xC234),
153         bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF)
154 };
155 static BIGNUM bn_group_3072 = {
156         bn_group_3072_value,
157         (sizeof bn_group_3072_value)/sizeof(BN_ULONG),
158         (sizeof bn_group_3072_value)/sizeof(BN_ULONG),
159         0,
160         BN_FLG_STATIC_DATA
161 };

163 static BN_ULONG bn_group_4096_value[] = {
164         bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF),
165         bn_pack4(0x4DF4,0x35C9,0x3406,0x3199),
166         bn_pack4(0x86FF,0xB7DC,0x90A6,0xC08F),
167         bn_pack4(0x93B4,0xEA98,0x8D8F,0xDDC1),
168         bn_pack4(0xD006,0x9127,0xD5B0,0x5AA9),
169         bn_pack4(0xB81B,0xDD76,0x2170,0x481C),
170         bn_pack4(0x1F61,0x2970,0xCEE2,0xD7AF),
171         bn_pack4(0x233B,0xA186,0x515B,0xE7ED),
172         bn_pack4(0x99B2,0x964F,0xA090,0xC3A2),
173         bn_pack4(0x287C,0x5947,0x4E6B,0xC05D),
174         bn_pack4(0x2E8E,0xFC14,0x1FBE,0xCAA6),
175         bn_pack4(0xDBBB,0xC2DB,0x04DE,0x8EF9),
176         bn_pack4(0x2583,0xE9CA,0x2AD4,0x4CE8),
177         bn_pack4(0x1A94,0x6834,0xB615,0x0BDA),
178         bn_pack4(0x99C3,0x2718,0x6AF4,0xE23C),
179         bn_pack4(0x8871,0x9A10,0xBDBA,0x5B26),
180         bn_pack4(0x1A72,0x3C12,0xA787,0xE6D7),
181         bn_pack4(0x4B82,0xD120,0xA921,0x0801),
182         bn_pack4(0x43DB,0x5BFC,0xE0FD,0x108E),
183         bn_pack4(0x08E2,0x4FA0,0x74E5,0xAB31),
184         bn_pack4(0x7709,0x88C0,0xBAD9,0x46E2),
185         bn_pack4(0xBBE1,0x1757,0x7A61,0x5D6C),
186         bn_pack4(0x521F,0x2B18,0x177B,0x200C),
187         bn_pack4(0xD876,0x0273,0x3EC8,0x6A64),
188         bn_pack4(0xF12F,0xFA06,0xD98A,0x0864),
189         bn_pack4(0xCEE3,0xD226,0x1AD2,0xEE6B),
190         bn_pack4(0x1E8C,0x94E0,0x4A25,0x619D),
191         bn_pack4(0xABF5,0xAE8C,0xDB09,0x33D7),
192         bn_pack4(0xB397,0x0F85,0xA6E1,0xE4C7),
193         bn_pack4(0x8AEA,0x7157,0x5D06,0x0C7D),
```



```
194         bn_pack4(0xECFB,0x8504,0x58DB,0xEF0A),
195         bn_pack4(0xA855,0x21AB,0xDF1C,0xBA64),
196         bn_pack4(0xAD33,0x170D,0x0450,0x7A33),
197         bn_pack4(0x1572,0x8E5A,0x8AAA,0xC42D),
198         bn_pack4(0x15D2,0x2618,0x98FA,0x0510),
199         bn_pack4(0x3995,0x497C,0xEA95,0x6AE5),
200         bn_pack4(0xDE2B,0xCBF6,0x9558,0x1718),
201         bn_pack4(0xB5C5,0x5DF0,0x6F4C,0x52C9),
202         bn_pack4(0x9B27,0x83A2,0xEC07,0xA28F),
203         bn_pack4(0xE39E,0x772C,0x180E,0x8603),
204         bn_pack4(0x3290,0x5E46,0x2E36,0xCE3B),
205         bn_pack4(0xF174,0x6C08,0xCA18,0x217C),
206         bn_pack4(0x670C,0x354E,0x4ABC,0x9804),
207         bn_pack4(0x9ED5,0x2907,0x7096,0x966D),
208         bn_pack4(0x1C62,0xF356,0x2085,0x52BB),
209         bn_pack4(0x8365,0x5D23,0xDCA3,0xAD96),
210         bn_pack4(0x6916,0x3FA8,0xFD24,0xCF5F),
211         bn_pack4(0x98DA,0x4836,0x1C55,0xD39A),
212         bn_pack4(0xC200,0x7CB8,0xA163,0xBF05),
213         bn_pack4(0x4928,0x6651,0xECE4,0x5B3D),
214         bn_pack4(0xAE9F,0x2411,0x7C4B,0x1FE6),
215         bn_pack4(0xEE38,0x6BFB,0x5A89,0x9FA5),
216         bn_pack4(0x0BFF,0x5CB6,0xF406,0xB7ED),
217         bn_pack4(0xF44C,0x42E9,0xA637,0xED6B),
218         bn_pack4(0xE485,0xB576,0x625E,0x7EC6),
219         bn_pack4(0x4FE1,0x356D,0x6D51,0xC245),
220         bn_pack4(0x302B,0x0A6D,0xF25F,0x1437),
221         bn_pack4(0xEF95,0x19B3,0xCD3A,0x431B),
222         bn_pack4(0x514A,0x0879,0x8E34,0x04DD),
223         bn_pack4(0x020B,0xBEA6,0x3B13,0x9B22),
224         bn_pack4(0x2902,0x4E08,0x8A67,0xCC74),
225         bn_pack4(0xC4C6,0x628B,0x80DC,0x1CD1),
226         bn_pack4(0xC90F,0xDAA2,0x2168,0xC234),
227         bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF)
228 };
229 static BIGNUM bn_group_4096 = {
230         bn_group_4096_value,
231         (sizeof bn_group_4096_value)/sizeof(BN_ULONG),
232         (sizeof bn_group_4096_value)/sizeof(BN_ULONG),
233         0,
234         BN_FLG_STATIC_DATA
235 };

237 static BN_ULONG bn_group_6144_value[] = {
238         bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF),
239         bn_pack4(0xE694,0xF91E,0x6DCC,0x4024),
240         bn_pack4(0x12BF,0x2D5B,0x0B74,0x74D6),
241         bn_pack4(0x043E,0x8F66,0x3F48,0x60EE),
242         bn_pack4(0x387F,0xE8D7,0x6E3C,0x0468),
243         bn_pack4(0xDA56,0xC9EC,0x2EF2,0x9632),
244         bn_pack4(0xEB19,0xCCB1,0xA313,0xD55C),
245         bn_pack4(0xF550,0xAA3D,0x8A1F,0xBFF0),
246         bn_pack4(0x06A1,0xD58B,0xB7C5,0xDA76),
247         bn_pack4(0xA797,0x15EE,0xF29B,0xE328),
248         bn_pack4(0x14CC,0x5ED2,0x0F80,0x37E0),
249         bn_pack4(0xCC8F,0x6D7E,0xBF48,0xE1D8),
250         bn_pack4(0x4BD4,0x07B2,0x2B41,0x54AA),
251         bn_pack4(0x0F1D,0x45B7,0xFF58,0x5AC5),
252         bn_pack4(0x23A9,0x7A7E,0x36CC,0x88BE),
253         bn_pack4(0x59E7,0xC97F,0xBEC7,0xE8F3),
254         bn_pack4(0xB5A8,0x4031,0x900B,0x1C9E),
255         bn_pack4(0xD55E,0x702F,0x4698,0x0C82),
256         bn_pack4(0xF482,0xD7CE,0x6E74,0xFEF6),
257         bn_pack4(0xF032,0xEA15,0xD172,0x1D03),
258         bn_pack4(0x5983,0xCA01,0xC64B,0x92EC),
259         bn_pack4(0x6FB8,0xF401,0x378C,0xD2BF),
```

```
260         bn_pack4(0x3320,0x5151,0x2BD7,0xAF42),
261         bn_pack4(0xDB7F,0x1447,0xE6CC,0x254B),
262         bn_pack4(0x44CE,0x6CBA,0xCED4,0xBB1B),
263         bn_pack4(0xDA3E,0xDBEB,0xCF9B,0x14ED),
264         bn_pack4(0x1797,0x27B0,0x865A,0x8918),
265         bn_pack4(0xB06A,0x53ED,0x9027,0xD831),
266         bn_pack4(0xE5DB,0x382F,0x4130,0x01AE),
267         bn_pack4(0xF8FF,0x9406,0xAD9E,0x530E),
268         bn_pack4(0xC975,0x1E76,0x3DBA,0x37BD),
269         bn_pack4(0xC1D4,0xDCB2,0x6026,0x46DE),
270         bn_pack4(0x36C3,0xFAB4,0xD27C,0x7026),
271         bn_pack4(0x4DF4,0x35C9,0x3402,0x8492),
272         bn_pack4(0x86FF,0xB7DC,0x90A6,0xC08F),
273         bn_pack4(0x93B4,0xEA98,0x8D8F,0xDDC1),
274         bn_pack4(0xD006,0x9127,0xD5B0,0x5AA9),
275         bn_pack4(0xB81B,0xDD76,0x2170,0x481C),
276         bn_pack4(0x1F61,0x2970,0xCEE2,0xD7AF),
277         bn_pack4(0x233B,0xA186,0x515B,0xE7ED),
278         bn_pack4(0x99B2,0x964F,0xA090,0xC3A2),
279         bn_pack4(0x287C,0x5947,0x4E6B,0xC05D),
280         bn_pack4(0x2E8E,0xFC14,0x1FBE,0xCAA6),
281         bn_pack4(0xDBBB,0xC2DB,0x04DE,0x8EF9),
282         bn_pack4(0x2583,0xE9CA,0x2AD4,0x4CE8),
283         bn_pack4(0x1A94,0x6834,0xB615,0x0BDA),
284         bn_pack4(0x99C3,0x2718,0x6AF4,0xE23C),
285         bn_pack4(0x8871,0x9A10,0xBDBA,0x5B26),
286         bn_pack4(0x1A72,0x3C12,0xA787,0xE6D7),
287         bn_pack4(0x4B82,0xD120,0xA921,0x0801),
288         bn_pack4(0x43DB,0x5BFC,0xE0FD,0x108E),
289         bn_pack4(0x08E2,0x4FA0,0x74E5,0xAB31),
290         bn_pack4(0x7709,0x88C0,0xBAD9,0x46E2),
291         bn_pack4(0xBBE1,0x1757,0x7A61,0x5D6C),
292         bn_pack4(0x521F,0x2B18,0x177B,0x200C),
293         bn_pack4(0xD876,0x0273,0x3EC8,0x6A64),
294         bn_pack4(0xF12F,0xFA06,0xD98A,0x0864),
295         bn_pack4(0xCEE3,0xD226,0x1AD2,0xEE6B),
296         bn_pack4(0x1E8C,0x94E0,0x4A25,0x619D),
297         bn_pack4(0xABF5,0xAE8C,0xDB09,0x33D7),
298         bn_pack4(0xB397,0x0F85,0xA6E1,0xE4C7),
299         bn_pack4(0x8AEA,0x7157,0x5D06,0x0C7D),
300         bn_pack4(0xECFB,0x8504,0x58DB,0xEF0A),
301         bn_pack4(0xA855,0x21AB,0xDF1C,0xBA64),
302         bn_pack4(0xAD33,0x170D,0x0450,0x7A33),
303         bn_pack4(0x1572,0x8E5A,0x8AAA,0xC42D),
304         bn_pack4(0x15D2,0x2618,0x98FA,0x0510),
305         bn_pack4(0x3995,0x497C,0xEA95,0x6AE5),
306         bn_pack4(0xDE2B,0xCBF6,0x9558,0x1718),
307         bn_pack4(0xB5C5,0x5DF0,0x6F4C,0x52C9),
308         bn_pack4(0x9B27,0x83A2,0xEC07,0xA28F),
309         bn_pack4(0xE39E,0x772C,0x180E,0x8603),
310         bn_pack4(0x3290,0x5E46,0x2E36,0xCE3B),
311         bn_pack4(0xF174,0x6C08,0xCA18,0x217C),
312         bn_pack4(0x670C,0x354E,0x4ABC,0x9804),
313         bn_pack4(0x9ED5,0x2907,0x7096,0x966D),
314         bn_pack4(0x1C62,0xF356,0x2085,0x52BB),
315         bn_pack4(0x8365,0x5D23,0xDCA3,0xAD96),
316         bn_pack4(0x6916,0x3FA8,0xFD24,0xCF5F),
317         bn_pack4(0x98DA,0x4836,0x1C55,0xD39A),
318         bn_pack4(0xC200,0x7CB8,0xA163,0xBF05),
319         bn_pack4(0x4928,0x6651,0xECE4,0x5B3D),
320         bn_pack4(0xAE9F,0x2411,0x7C4B,0x1FE6),
321         bn_pack4(0xEE38,0x6BFB,0x5A89,0x9FA5),
322         bn_pack4(0x0BFF,0x5CB6,0xF406,0xB7ED),
323         bn_pack4(0xF44C,0x42E9,0xA637,0xED6B),
324         bn_pack4(0xE485,0xB576,0x625E,0x7EC6),
325         bn_pack4(0x4FE1,0x356D,0x6D51,0xC245),
```



```
326         bn_pack4(0x302B,0x0A6D,0xF25F,0x1437),
327         bn_pack4(0xEF95,0x19B3,0xCD3A,0x431B),
328         bn_pack4(0x514A,0x0879,0x8E34,0x04DD),
329         bn_pack4(0x020B,0xBEA6,0x3B13,0x9B22),
330         bn_pack4(0x2902,0x4E08,0x8A67,0xCC74),
331         bn_pack4(0xC4C6,0x628B,0x80DC,0x1CD1),
332         bn_pack4(0xC90F,0xDAA2,0x2168,0xC234),
333         bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF)
334 };
335 static BIGNUM bn_group_6144 = {
336         bn_group_6144_value,
337         (sizeof bn_group_6144_value)/sizeof(BN_ULONG),
338         (sizeof bn_group_6144_value)/sizeof(BN_ULONG),
339         0,
340         BN_FLG_STATIC_DATA
341 };

343 static BN_ULONG bn_group_8192_value[] = {
344         bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF),
345         bn_pack4(0x60C9,0x80DD,0x98ED,0xD3DF),
346         bn_pack4(0xC81F,0x56E8,0x80B9,0x6E71),
347         bn_pack4(0x9E30,0x50E2,0x7656,0x94DF),
348         bn_pack4(0x9558,0xE447,0x5677,0xE9AA),
349         bn_pack4(0xC919,0x0DA6,0xFC02,0x6E47),
350         bn_pack4(0x889A,0x002E,0xD5EE,0x382B),
351         bn_pack4(0x4009,0x438B,0x481C,0x6CD7),
352         bn_pack4(0x3590,0x46F4,0xEB87,0x9F92),
353         bn_pack4(0xFAF3,0x6BC3,0x1ECF,0xA268),
354         bn_pack4(0xB1D5,0x10BD,0x7EE7,0x4D73),
355         bn_pack4(0xF9AB,0x4819,0x5DED,0x7EA1),
356         bn_pack4(0x64F3,0x1CC5,0x0846,0x851D),
357         bn_pack4(0x4597,0xE899,0xA025,0x5DC1),
358         bn_pack4(0xDF31,0x0EE0,0x74AB,0x6A36),
359         bn_pack4(0x6D2A,0x13F8,0x3F44,0xF82D),
360         bn_pack4(0x062B,0x3CF5,0xB3A2,0x78A6),
361         bn_pack4(0x7968,0x3303,0xED5B,0xDD3A),
362         bn_pack4(0xFA9D,0x4B7F,0xA2C0,0x87E8),
363         bn_pack4(0x4BCB,0xC886,0x2F83,0x85DD),
364         bn_pack4(0x3473,0xFC64,0x6CEA,0x306B),
365         bn_pack4(0x13EB,0x57A8,0x1A23,0xF0C7),
366         bn_pack4(0x2222,0x2E04,0xA403,0x7C07),
367         bn_pack4(0xE3FD,0xB8BE,0xFC84,0x8AD9),
368         bn_pack4(0x238F,0x16CB,0xE39D,0x652D),
369         bn_pack4(0x3423,0xB474,0x2BF1,0xC978),
370         bn_pack4(0x3AAB,0x639C,0x5AE4,0xF568),
371         bn_pack4(0x2576,0xF693,0x6BA4,0x2466),
372         bn_pack4(0x741F,0xA7BF,0x8AFC,0x47ED),
373         bn_pack4(0x3BC8,0x32B6,0x8D9D,0xD300),
374         bn_pack4(0xD8BE,0xC4D0,0x73B9,0x31BA),
375         bn_pack4(0x3877,0x7CB6,0xA932,0xDF8C),
376         bn_pack4(0x74A3,0x926F,0x12FE,0xE5E4),
377         bn_pack4(0xE694,0xF91E,0x6DBE,0x1159),
378         bn_pack4(0x12BF,0x2D5B,0x0B74,0x74D6),
379         bn_pack4(0x043E,0x8F66,0x3F48,0x60EE),
380         bn_pack4(0x387F,0xE8D7,0x6E3C,0x0468),
381         bn_pack4(0xDA56,0xC9EC,0x2EF2,0x9632),
382         bn_pack4(0xEB19,0xCCB1,0xA313,0xD55C),
383         bn_pack4(0xF550,0xAA3D,0x8A1F,0xBFF0),
384         bn_pack4(0x06A1,0xD58B,0xB7C5,0xDA76),
385         bn_pack4(0xA797,0x15EE,0xF29B,0xE328),
386         bn_pack4(0x14CC,0x5ED2,0x0F80,0x37E0),
387         bn_pack4(0xCC8F,0x6D7E,0xBF48,0xE1D8),
388         bn_pack4(0x4BD4,0x07B2,0x2B41,0x54AA),
389         bn_pack4(0x0F1D,0x45B7,0xFF58,0x5AC5),
390         bn_pack4(0x23A9,0x7A7E,0x36CC,0x88BE),
391         bn_pack4(0x59E7,0xC97F,0xBEC7,0xE8F3),
```

```
392         bn_pack4(0xB5A8,0x4031,0x900B,0x1C9E),
393         bn_pack4(0xD55E,0x702F,0x4698,0x0C82),
394         bn_pack4(0xF482,0xD7CE,0x6E74,0xFEF6),
395         bn_pack4(0xF032,0xEA15,0xD172,0x1D03),
396         bn_pack4(0x5983,0xCA01,0xC64B,0x92EC),
397         bn_pack4(0x6FB8,0xF401,0x378C,0xD2BF),
398         bn_pack4(0x3320,0x5151,0x2BD7,0xAF42),
399         bn_pack4(0xDB7F,0x1447,0xE6CC,0x254B),
400         bn_pack4(0x44CE,0x6CBA,0xCED4,0xBB1B),
401         bn_pack4(0xDA3E,0xDBEB,0xCF9B,0x14ED),
402         bn_pack4(0x1797,0x27B0,0x865A,0x8918),
403         bn_pack4(0xB06A,0x53ED,0x9027,0xD831),
404         bn_pack4(0xE5DB,0x382F,0x4130,0x01AE),
405         bn_pack4(0xF8FF,0x9406,0xAD9E,0x530E),
406         bn_pack4(0xC975,0x1E76,0x3DBA,0x37BD),
407         bn_pack4(0xC1D4,0xDCB2,0x6026,0x46DE),
408         bn_pack4(0x36C3,0xFAB4,0xD27C,0x7026),
409         bn_pack4(0x4DF4,0x35C9,0x3402,0x8492),
410         bn_pack4(0x86FF,0xB7DC,0x90A6,0xC08F),
411         bn_pack4(0x93B4,0xEA98,0x8D8F,0xDDC1),
412         bn_pack4(0xD006,0x9127,0xD5B0,0x5AA9),
413         bn_pack4(0xB81B,0xDD76,0x2170,0x481C),
414         bn_pack4(0x1F61,0x2970,0xCEE2,0xD7AF),
415         bn_pack4(0x233B,0xA186,0x515B,0xE7ED),
416         bn_pack4(0x99B2,0x964F,0xA090,0xC3A2),
417         bn_pack4(0x287C,0x5947,0x4E6B,0xC05D),
418         bn_pack4(0x2E8E,0xFC14,0x1FBE,0xCAA6),
419         bn_pack4(0xDBBB,0xC2DB,0x04DE,0x8EF9),
420         bn_pack4(0x2583,0xE9CA,0x2AD4,0x4CE8),
421         bn_pack4(0x1A94,0x6834,0xB615,0x0BDA),
422         bn_pack4(0x99C3,0x2718,0x6AF4,0xE23C),
423         bn_pack4(0x8871,0x9A10,0xBDBA,0x5B26),
424         bn_pack4(0x1A72,0x3C12,0xA787,0xE6D7),
425         bn_pack4(0x4B82,0xD120,0xA921,0x0801),
426         bn_pack4(0x43DB,0x5BFC,0xE0FD,0x108E),
427         bn_pack4(0x08E2,0x4FA0,0x74E5,0xAB31),
428         bn_pack4(0x7709,0x88C0,0xBAD9,0x46E2),
429         bn_pack4(0xBBE1,0x1757,0x7A61,0x5D6C),
430         bn_pack4(0x521F,0x2B18,0x177B,0x200C),
431         bn_pack4(0xD876,0x0273,0x3EC8,0x6A64),
432         bn_pack4(0xF12F,0xFA06,0xD98A,0x0864),
433         bn_pack4(0xCEE3,0xD226,0x1AD2,0xEE6B),
434         bn_pack4(0x1E8C,0x94E0,0x4A25,0x619D),
435         bn_pack4(0xABF5,0xAE8C,0xDB09,0x33D7),
436         bn_pack4(0xB397,0x0F85,0xA6E1,0xE4C7),
437         bn_pack4(0x8AEA,0x7157,0x5D06,0x0C7D),
438         bn_pack4(0xECFB,0x8504,0x58DB,0xEF0A),
439         bn_pack4(0xA855,0x21AB,0xDF1C,0xBA64),
440         bn_pack4(0xAD33,0x170D,0x0450,0x7A33),
441         bn_pack4(0x1572,0x8E5A,0x8AAA,0xC42D),
442         bn_pack4(0x15D2,0x2618,0x98FA,0x0510),
443         bn_pack4(0x3995,0x497C,0xEA95,0x6AE5),
444         bn_pack4(0xDE2B,0xCBF6,0x9558,0x1718),
445         bn_pack4(0xB5C5,0x5DF0,0x6F4C,0x52C9),
446         bn_pack4(0x9B27,0x83A2,0xEC07,0xA28F),
447         bn_pack4(0xE39E,0x772C,0x180E,0x8603),
448         bn_pack4(0x3290,0x5E46,0x2E36,0xCE3B),
449         bn_pack4(0xF174,0x6C08,0xCA18,0x217C),
450         bn_pack4(0x670C,0x354E,0x4ABC,0x9804),
451         bn_pack4(0x9ED5,0x2907,0x7096,0x966D),
452         bn_pack4(0x1C62,0xF356,0x2085,0x52BB),
453         bn_pack4(0x8365,0x5D23,0xDCA3,0xAD96),
454         bn_pack4(0x6916,0x3FA8,0xFD24,0xCF5F),
455         bn_pack4(0x98DA,0x4836,0x1C55,0xD39A),
456         bn_pack4(0xC200,0x7CB8,0xA163,0xBF05),
457         bn_pack4(0x4928,0x6651,0xECE4,0x5B3D),
```



```
458         bn_pack4(0xAE9F,0x2411,0x7C4B,0x1FE6),
459         bn_pack4(0xEE38,0x6BFB,0x5A89,0x9FA5),
460         bn_pack4(0x0BFF,0x5CB6,0xF406,0xB7ED),
461         bn_pack4(0xF44C,0x42E9,0xA637,0xED6B),
462         bn_pack4(0xE485,0xB576,0x625E,0x7EC6),
463         bn_pack4(0x4FE1,0x356D,0x6D51,0xC245),
464         bn_pack4(0x302B,0x0A6D,0xF25F,0x1437),
465         bn_pack4(0xEF95,0x19B3,0xCD3A,0x431B),
466         bn_pack4(0x514A,0x0879,0x8E34,0x04DD),
467         bn_pack4(0x020B,0xBEA6,0x3B13,0x9B22),
468         bn_pack4(0x2902,0x4E08,0x8A67,0xCC74),
469         bn_pack4(0xC4C6,0x628B,0x80DC,0x1CD1),
470         bn_pack4(0xC90F,0xDAA2,0x2168,0xC234),
471         bn_pack4(0xFFFF,0xFFFF,0xFFFF,0xFFFF)
472 };
473 static BIGNUM bn_group_8192 = {
474         bn_group_8192_value,
475         (sizeof bn_group_8192_value)/sizeof(BN_ULONG),
476         (sizeof bn_group_8192_value)/sizeof(BN_ULONG),
477         0,
478         BN_FLG_STATIC_DATA
479 };

481 static BN_ULONG bn_generator_19_value[] = {19} ;
482 static BIGNUM bn_generator_19 = {
483         bn_generator_19_value,
484         1,
485         1,
486         0,
487         BN_FLG_STATIC_DATA
488 };
489 static BN_ULONG bn_generator_5_value[] = {5} ;
490 static BIGNUM bn_generator_5 = {
491         bn_generator_5_value,
492         1,
493         1,
494         0,
495         BN_FLG_STATIC_DATA
496 };
497 static BN_ULONG bn_generator_2_value[] = {2} ;
498 static BIGNUM bn_generator_2 = {
499         bn_generator_2_value,
500         1,
501         1,
502         0,
503         BN_FLG_STATIC_DATA
504 };

506 static SRP_gN knowngN[] = {
507         {"8192",&bn_generator_19 , &bn_group_8192},
508         {"6144",&bn_generator_5 , &bn_group_6144},
509         {"4096",&bn_generator_5 , &bn_group_4096},
510         {"3072",&bn_generator_5 , &bn_group_3072},
511         {"2048",&bn_generator_2 , &bn_group_2048},
512         {"1536",&bn_generator_2 , &bn_group_1536},
513         {"1024",&bn_generator_2 , &bn_group_1024},
514 };
515 #define KNOWN_GN_NUMBER sizeof(knowngN) / sizeof(SRP_gN)

517 /* end of generated data */
518 #endif /* ! codereview */
```

```
**********************************************************
    3118 Wed Aug 13 19:51:54 2014
new/usr/src/lib/openssl/include/srp_lcl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/srp/srp_lcl.h */
/* Written by Peter Sylvester (peter.sylvester@edelweb.fr)
 * for the EdelKey project and contributed to the OpenSSL project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#ifndef HEADER_SRP_LCL_H
#define HEADER_SRP_LCL_H

#include <openssl/srp.h>
```



```
#include <openssl/sha.h>

#if 0
#define srp_bn_print(a) {fprintf(stderr, #a "="); BN_print_fp(stderr,a); \
   fprintf(stderr,"\n");}
#else
#define   srp_bn_print(a)
#endif



#ifdef  __cplusplus
extern "C" {
#endif



#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
   42949 Wed Aug 13 19:51:54 2014
new/usr/src/lib/openssl/include/ssl_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_locl.h */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECC cipher suite support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
```

```
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#ifndef HEADER_SSL_LOCL_H
#define HEADER_SSL_LOCL_H
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <errno.h>

#include "e_os.h"

#include <openssl/buffer.h>
#ifndef OPENSSL_NO_COMP
#include <openssl/comp.h>
#endif
#include <openssl/bio.h>
#include <openssl/stack.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#include <openssl/err.h>
#include <openssl/ssl.h>
#include <openssl/symhacks.h>

#ifdef OPENSSL_BUILD_SHLIBSSL
# undef OPENSSL_EXTERN
# define OPENSSL_EXTERN OPENSSL_EXPORT
#endif

#undef PKCS1_CHECK

#define c2l(c,l)        (l = ((unsigned long)(*((c)++)))      , \
                         l|=(((unsigned long)(*((c)++)))<< 8), \
                         l|=(((unsigned long)(*((c)++)))<<16), \
                         l|=(((unsigned long)(*((c)++)))<<24))

/* NOTE - c is not incremented as per c2l */
#define c2ln(c,l1,l2,n) { \
                        c+=n; \
                        l1=l2=0; \
                        switch (n) { \
                        case 8: l2 =((unsigned long)(*(--(c))))<<24; \
                        case 7: l2|=((unsigned long)(*(--(c))))<<16; \
                        case 6: l2|=((unsigned long)(*(--(c))))<< 8; \
                        case 5: l2|=((unsigned long)(*(--(c))));     \
                        case 4: l1 =((unsigned long)(*(--(c))))<<24; \
                        case 3: l1|=((unsigned long)(*(--(c))))<<16; \
                        case 2: l1|=((unsigned long)(*(--(c))))<< 8; \
                        case 1: l1|=((unsigned long)(*(--(c))));     \
                                } \
```



```
                        }

#define l2c(l,c)        (*((c)++)=(unsigned char)(((l)    )&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24)&0xff))

#define n2l(c,l)        (l =((unsigned long)(*((c)++)))<<24, \
                         l|=((unsigned long)(*((c)++)))<<16, \
                         l|=((unsigned long)(*((c)++)))<< 8, \
                         l|=((unsigned long)(*((c)++))))

#define l2n(l,c)        (*((c)++)=(unsigned char)(((l)>>24)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16)&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8)&0xff), \
                         *((c)++)=(unsigned char)(((l)    )&0xff))

#define l2n6(l,c)       (*((c)++)=(unsigned char)(((l)>>40)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>32)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16)&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8)&0xff), \
                         *((c)++)=(unsigned char)(((l)    )&0xff))

#define l2n8(l,c)       (*((c)++)=(unsigned char)(((l)>>56)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>48)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>40)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>32)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>24)&0xff), \
                         *((c)++)=(unsigned char)(((l)>>16)&0xff), \
                         *((c)++)=(unsigned char)(((l)>> 8)&0xff), \
                         *((c)++)=(unsigned char)(((l)    )&0xff))

#define n2l6(c,l)       (l =((BN_ULLONG)(*((c)++)))<<40, \
                         l|=((BN_ULLONG)(*((c)++)))<<32, \
                         l|=((BN_ULLONG)(*((c)++)))<<24, \
                         l|=((BN_ULLONG)(*((c)++)))<<16, \
                         l|=((BN_ULLONG)(*((c)++)))<< 8, \
                         l|=((BN_ULLONG)(*((c)++))))

/* NOTE - c is not incremented as per l2c */
#define l2cn(l1,l2,c,n) { \
                        c+=n; \
                        switch (n) { \
                        case 8: *(--(c))=(unsigned char)(((l2)>>24)&0xff); \
                        case 7: *(--(c))=(unsigned char)(((l2)>>16)&0xff); \
                        case 6: *(--(c))=(unsigned char)(((l2)>> 8)&0xff); \
                        case 5: *(--(c))=(unsigned char)(((l2)    )&0xff); \
                        case 4: *(--(c))=(unsigned char)(((l1)>>24)&0xff); \
                        case 3: *(--(c))=(unsigned char)(((l1)>>16)&0xff); \
                        case 2: *(--(c))=(unsigned char)(((l1)>> 8)&0xff); \
                        case 1: *(--(c))=(unsigned char)(((l1)    )&0xff); \
                                } \
                        }

#define n2s(c,s)        ((s=(((unsigned int)(c[0]))<< 8)| \
                            (((unsigned int)(c[1]))    )),c+=2)
#define s2n(s,c)        ((c[0]=(unsigned char)(((s)>> 8)&0xff), \
                          c[1]=(unsigned char)(((s)    )&0xff)),c+=2)

#define n2l3(c,l)       ((l =(((unsigned long)(c[0]))<<16)| \
                             (((unsigned long)(c[1]))<< 8)| \
                             (((unsigned long)(c[2]))    )),c+=3)

#define l2n3(l,c)       ((c[0]=(unsigned char)(((l)>>16)&0xff), \
                          c[1]=(unsigned char)(((l)>> 8)&0xff), \
```

```
260                         c[2]=(unsigned char)(((l)    )&0xff)),c+=3)

262 /* LOCAL STUFF */

264 #define SSL_DECRYPT     0
265 #define SSL_ENCRYPT     1

267 #define TWO_BYTE_BIT    0x80
268 #define SEC_ESC_BIT     0x40
269 #define TWO_BYTE_MASK   0x7fff
270 #define THREE_BYTE_MASK 0x3fff

272 #define INC32(a)        ((a)=((a)+1)&0xffffffffL)
273 #define DEC32(a)        ((a)=((a)-1)&0xffffffffL)
274 #define MAX_MAC_SIZE    20 /* up from 16 for SSLv3 */

276 /*
277  * Define the Bitmasks for SSL_CIPHER.algorithms.
278  * This bits are used packed as dense as possible. If new methods/ciphers
279  * etc will be added, the bits a likely to change, so this information
280  * is for internal library use only, even though SSL_CIPHER.algorithms
281  * can be publicly accessed.
282  * Use the according functions for cipher management instead.
283  *
284  * The bit mask handling in the selection and sorting scheme in
285  * ssl_create_cipher_list() has only limited capabilities, reflecting
286  * that the different entities within are mutually exclusive:
287  * ONLY ONE BIT PER MASK CAN BE SET AT A TIME.
288  */

290 /* Bits for algorithm_mkey (key exchange algorithm) */
291 #define SSL_kRSA                0x00000001L /* RSA key exchange */
292 #define SSL_kDHr                0x00000002L /* DH cert, RSA CA cert */ /* no suc
293 #define SSL_kDHd                0x00000004L /* DH cert, DSA CA cert */ /* no suc
294 #define SSL_kEDH                0x00000008L /* tmp DH key no DH cert */
295 #define SSL_kKRB5               0x00000010L /* Kerberos5 key exchange */
296 #define SSL_kECDHr              0x00000020L /* ECDH cert, RSA CA cert */
297 #define SSL_kECDHe              0x00000040L /* ECDH cert, ECDSA CA cert */
298 #define SSL_kEECDH              0x00000080L /* ephemeral ECDH */
299 #define SSL_kPSK                0x00000100L /* PSK */
300 #define SSL_kGOST       0x00000200L /* GOST key exchange */
301 #define SSL_kSRP        0x00000400L /* SRP */

303 /* Bits for algorithm_auth (server authentication) */
304 #define SSL_aRSA                0x00000001L /* RSA auth */
305 #define SSL_aDSS                0x00000002L /* DSS auth */
306 #define SSL_aNULL               0x00000004L /* no auth (i.e. use ADH or AECDH) *
307 #define SSL_aDH                 0x00000008L /* Fixed DH auth (kDHd or kDHr) */ /
308 #define SSL_aECDH               0x00000010L /* Fixed ECDH auth (kECDHe or kECDHr
309 #define SSL_aKRB5               0x00000020L /* KRB5 auth */
310 #define SSL_aECDSA              0x00000040L /* ECDSA auth*/
311 #define SSL_aPSK                0x00000080L /* PSK auth */
312 #define SSL_aGOST94                          0x00000100L /* GOST R 34.10-94 s
313 #define SSL_aGOST01                  0x00000200L /* GOST R 34.10-2001 signatu
314 #define SSL_aSRP                0x00000400L /* SRP auth */


317 /* Bits for algorithm_enc (symmetric encryption) */
318 #define SSL_DES                 0x00000001L
319 #define SSL_3DES                0x00000002L
320 #define SSL_RC4                 0x00000004L
321 #define SSL_RC2                 0x00000008L
322 #define SSL_IDEA                0x00000010L
323 #define SSL_eNULL               0x00000020L
324 #define SSL_AES128              0x00000040L
325 #define SSL_AES256              0x00000080L
```



```
326 #define SSL_CAMELLIA128         0x00000100L
327 #define SSL_CAMELLIA256         0x00000200L
328 #define SSL_eGOST2814789CNT     0x00000400L
329 #define SSL_SEED                0x00000800L
330 #define SSL_AES128GCM           0x00001000L
331 #define SSL_AES256GCM           0x00002000L

333 #define SSL_AES                 (SSL_AES128|SSL_AES256|SSL_AES128GCM|SSL_AES256G
334 #define SSL_CAMELLIA            (SSL_CAMELLIA128|SSL_CAMELLIA256)


337 /* Bits for algorithm_mac (symmetric authentication) */

339 #define SSL_MD5                 0x00000001L
340 #define SSL_SHA1                0x00000002L
341 #define SSL_GOST94      0x00000004L
342 #define SSL_GOST89MAC   0x00000008L
343 #define SSL_SHA256              0x00000010L
344 #define SSL_SHA384              0x00000020L
345 /* Not a real MAC, just an indication it is part of cipher */
346 #define SSL_AEAD                0x00000040L

348 /* Bits for algorithm_ssl (protocol version) */
349 #define SSL_SSLV2               0x00000001L
350 #define SSL_SSLV3               0x00000002L
351 #define SSL_TLSV1               SSL_SSLV3       /* for now */
352 #define SSL_TLSV1_2             0x00000004L


355 /* Bits for algorithm2 (handshake digests and other extra flags) */

357 #define SSL_HANDSHAKE_MAC_MD5 0x10
358 #define SSL_HANDSHAKE_MAC_SHA 0x20
359 #define SSL_HANDSHAKE_MAC_GOST94 0x40
360 #define SSL_HANDSHAKE_MAC_SHA256 0x80
361 #define SSL_HANDSHAKE_MAC_SHA384 0x100
362 #define SSL_HANDSHAKE_MAC_DEFAULT (SSL_HANDSHAKE_MAC_MD5 | SSL_HANDSHAKE_MAC_SHA

364 /* When adding new digest in the ssl_ciph.c and increment SSM_MD_NUM_IDX
365  * make sure to update this constant too */
366 #define SSL_MAX_DIGEST 6

368 #define TLS1_PRF_DGST_MASK      (0xff << TLS1_PRF_DGST_SHIFT)

370 #define TLS1_PRF_DGST_SHIFT 10
371 #define TLS1_PRF_MD5 (SSL_HANDSHAKE_MAC_MD5 << TLS1_PRF_DGST_SHIFT)
372 #define TLS1_PRF_SHA1 (SSL_HANDSHAKE_MAC_SHA << TLS1_PRF_DGST_SHIFT)
373 #define TLS1_PRF_SHA256 (SSL_HANDSHAKE_MAC_SHA256 << TLS1_PRF_DGST_SHIFT)
374 #define TLS1_PRF_SHA384 (SSL_HANDSHAKE_MAC_SHA384 << TLS1_PRF_DGST_SHIFT)
375 #define TLS1_PRF_GOST94 (SSL_HANDSHAKE_MAC_GOST94 << TLS1_PRF_DGST_SHIFT)
376 #define TLS1_PRF (TLS1_PRF_MD5 | TLS1_PRF_SHA1)

378 /* Stream MAC for GOST ciphersuites from cryptopro draft
379  * (currently this also goes into algorithm2) */
380 #define TLS1_STREAM_MAC 0x04



384 /*
385  * Export and cipher strength information. For each cipher we have to decide
386  * whether it is exportable or not. This information is likely to change
387  * over time, since the export control rules are no static technical issue.
388  *
389  * Independent of the export flag the cipher strength is sorted into classes.
390  * SSL_EXP40 was denoting the 40bit US export limit of past times, which now
391  * is at 56bit (SSL_EXP56). If the exportable cipher class is going to change
```

```
 * again (eg. to 64bit) the use of "SSL_EXP*" becomes blurred even more,
 * since SSL_EXP64 could be similar to SSL_LOW.
 * For this reason SSL_MICRO and SSL_MINI macros are included to widen the
 * namespace of SSL_LOW-SSL_HIGH to lower values. As development of speed
 * and ciphers goes, another extension to SSL_SUPER and/or SSL_ULTRA would
 * be possible.
 */
#define SSL_EXP_MASK            0x00000003L
#define SSL_STRONG_MASK         0x000001fcL

#define SSL_NOT_EXP             0x00000001L
#define SSL_EXPORT              0x00000002L

#define SSL_STRONG_NONE         0x00000004L
#define SSL_EXP40               0x00000008L
#define SSL_MICRO               (SSL_EXP40)
#define SSL_EXP56               0x00000010L
#define SSL_MINI                (SSL_EXP56)
#define SSL_LOW                 0x00000020L
#define SSL_MEDIUM              0x00000040L
#define SSL_HIGH                0x00000080L
#define SSL_FIPS                0x00000100L

/* we have used 000001ff - 23 bits left to go */

/*
 * Macros to check the export status and cipher strength for export ciphers.
 * Even though the macros for EXPORT and EXPORT40/56 have similar names,
 * their meaning is different:
 * *_EXPORT macros check the 'exportable' status.
 * *_EXPORT40/56 macros are used to check whether a certain cipher strength
 *          is given.
 * Since the SSL_IS_EXPORT* and SSL_EXPORT* macros depend on the correct
 * algorithm structure element to be passed (algorithms, algo_strength) and no
 * typechecking can be done as they are all of type unsigned long, their
 * direct usage is discouraged.
 * Use the SSL_C_* macros instead.
 */
#define SSL_IS_EXPORT(a)        ((a)&SSL_EXPORT)
#define SSL_IS_EXPORT56(a)      ((a)&SSL_EXP56)
#define SSL_IS_EXPORT40(a)      ((a)&SSL_EXP40)
#define SSL_C_IS_EXPORT(c)      SSL_IS_EXPORT((c)->algo_strength)
#define SSL_C_IS_EXPORT56(c)    SSL_IS_EXPORT56((c)->algo_strength)
#define SSL_C_IS_EXPORT40(c)    SSL_IS_EXPORT40((c)->algo_strength)

#define SSL_EXPORT_KEYLENGTH(a,s)       (SSL_IS_EXPORT40(s) ? 5 : \
                                 (a) == SSL_DES ? 8 : 7)
#define SSL_EXPORT_PKEYLENGTH(a) (SSL_IS_EXPORT40(a) ? 512 : 1024)
#define SSL_C_EXPORT_KEYLENGTH(c)       SSL_EXPORT_KEYLENGTH((c)->algorithm_enc,
                                (c)->algo_strength)
#define SSL_C_EXPORT_PKEYLENGTH(c)      SSL_EXPORT_PKEYLENGTH((c)->algo_strength




/* Mostly for SSLv3 */
#define SSL_PKEY_RSA_ENC        0
#define SSL_PKEY_RSA_SIGN       1
#define SSL_PKEY_DSA_SIGN       2
#define SSL_PKEY_DH_RSA         3
#define SSL_PKEY_DH_DSA         4
#define SSL_PKEY_ECC            5
#define SSL_PKEY_GOST94         6
#define SSL_PKEY_GOST01         7
#define SSL_PKEY_NUM            8
```



```
/* SSL_kRSA <- RSA_ENC | (RSA_TMP & RSA_SIGN) |
 *          <- (EXPORT & (RSA_ENC | RSA_TMP) & RSA_SIGN)
 * SSL_kDH  <- DH_ENC & (RSA_ENC | RSA_SIGN | DSA_SIGN)
 * SSL_kEDH <- RSA_ENC | RSA_SIGN | DSA_SIGN
 * SSL_aRSA <- RSA_ENC | RSA_SIGN
 * SSL_aDSS <- DSA_SIGN
 */

/*
#define CERT_INVALID            0
#define CERT_PUBLIC_KEY         1
#define CERT_PRIVATE_KEY        2
*/

#ifndef OPENSSL_NO_EC
/* From ECC-TLS draft, used in encoding the curve type in
 * ECParameters
 */
#define EXPLICIT_PRIME_CURVE_TYPE  1
#define EXPLICIT_CHAR2_CURVE_TYPE  2
#define NAMED_CURVE_TYPE           3
#endif  /* OPENSSL_NO_EC */

typedef struct cert_pkey_st
        {
        X509 *x509;
        EVP_PKEY *privatekey;
        /* Digest to use when signing */
        const EVP_MD *digest;
        } CERT_PKEY;

typedef struct cert_st
        {
        /* Current active set */
        CERT_PKEY *key; /* ALWAYS points to an element of the pkeys array
                         * Probably it would make more sense to store
                         * an index, not a pointer. */

        /* The following masks are for the key and auth
         * algorithms that are supported by the certs below */
        int valid;
        unsigned long mask_k;
        unsigned long mask_a;
        unsigned long export_mask_k;
        unsigned long export_mask_a;
#ifndef OPENSSL_NO_RSA
        RSA *rsa_tmp;
        RSA *(*rsa_tmp_cb)(SSL *ssl,int is_export,int keysize);
#endif
#ifndef OPENSSL_NO_DH
        DH *dh_tmp;
        DH *(*dh_tmp_cb)(SSL *ssl,int is_export,int keysize);
#endif
#ifndef OPENSSL_NO_ECDH
        EC_KEY *ecdh_tmp;
        /* Callback for generating ephemeral ECDH keys */
        EC_KEY *(*ecdh_tmp_cb)(SSL *ssl,int is_export,int keysize);
#endif

        CERT_PKEY pkeys[SSL_PKEY_NUM];

        int references; /* >1 only if SSL_copy_session_id is used */
        } CERT;


typedef struct sess_cert_st
```

```
524         {
525         STACK_OF(X509) *cert_chain; /* as received from peer (not for SSL2) */

527         /* The 'peer_...' members are used only by clients. */
528         int peer_cert_type;

530         CERT_PKEY *peer_key; /* points to an element of peer_pkeys (never NULL!) */
531         CERT_PKEY peer_pkeys[SSL_PKEY_NUM];
532         /* Obviously we don't have the private keys of these,
533          * so maybe we shouldn't even use the CERT_PKEY type here. */

535 #ifndef OPENSSL_NO_RSA
536         RSA *peer_rsa_tmp; /* not used for SSL 2 */
537 #endif
538 #ifndef OPENSSL_NO_DH
539         DH *peer_dh_tmp; /* not used for SSL 2 */
540 #endif
541 #ifndef OPENSSL_NO_ECDH
542         EC_KEY *peer_ecdh_tmp;
543 #endif

545         int references; /* actually always 1 at the moment */
546         } SESS_CERT;


549 /*#define MAC_DEBUG     */

551 /*#define ERR_DEBUG     */
552 /*#define ABORT_DEBUG   */
553 /*#define PKT_DEBUG 1   */
554 /*#define DES_DEBUG     */
555 /*#define DES_OFB_DEBUG */
556 /*#define SSL_DEBUG     */
557 /*#define RSA_DEBUG     */
558 /*#define IDEA_DEBUG    */

560 #define FP_ICC  (int (*)(const void *,const void *))
561 #define ssl_put_cipher_by_char(ssl,ciph,ptr) \
562                 ((ssl)->method->put_cipher_by_char((ciph),(ptr)))
563 #define ssl_get_cipher_by_char(ssl,ptr) \
564                 ((ssl)->method->get_cipher_by_char(ptr))

566 /* This is for the SSLv3/TLSv1.0 differences in crypto/hash stuff
567  * It is a bit of a mess of functions, but hell, think of it as
568  * an opaque structure :-) */
569 typedef struct ssl3_enc_method
570         {
571         int (*enc)(SSL *, int);
572         int (*mac)(SSL *, unsigned char *, int);
573         int (*setup_key_block)(SSL *);
574         int (*generate_master_secret)(SSL *, unsigned char *, unsigned char *, i
575         int (*change_cipher_state)(SSL *, int);
576         int (*final_finish_mac)(SSL *,  const char *, int, unsigned char *);
577         int finish_mac_length;
578         int (*cert_verify_mac)(SSL *, int, unsigned char *);
579         const char *client_finished_label;
580         int client_finished_label_len;
581         const char *server_finished_label;
582         int server_finished_label_len;
583         int (*alert_value)(int);
584         int (*export_keying_material)(SSL *, unsigned char *, size_t,
585                                       const char *, size_t,
586                                       const unsigned char *, size_t,
587                                       int use_context);
588         } SSL3_ENC_METHOD;
```



```
590 #ifndef OPENSSL_NO_COMP
591 /* Used for holding the relevant compression methods loaded into SSL_CTX */
592 typedef struct ssl3_comp_st
593         {
594         int comp_id;    /* The identifier byte for this compression type */
595         char *name;     /* Text name used for the compression type */
596         COMP_METHOD *method; /* The method :-) */
597         } SSL3_COMP;
598 #endif

600 #ifndef OPENSSL_NO_BUF_FREELISTS
601 typedef struct ssl3_buf_freelist_st
602         {
603         size_t chunklen;
604         unsigned int len;
605         struct ssl3_buf_freelist_entry_st *head;
606         } SSL3_BUF_FREELIST;

608 typedef struct ssl3_buf_freelist_entry_st
609         {
610         struct ssl3_buf_freelist_entry_st *next;
611         } SSL3_BUF_FREELIST_ENTRY;
612 #endif

614 extern SSL3_ENC_METHOD ssl3_undef_enc_method;
615 OPENSSL_EXTERN const SSL_CIPHER ssl2_ciphers[];
616 OPENSSL_EXTERN SSL_CIPHER ssl3_ciphers[];


619 SSL_METHOD *ssl_bad_method(int ver);

621 extern SSL3_ENC_METHOD TLSv1_enc_data;
622 extern SSL3_ENC_METHOD SSLv3_enc_data;
623 extern SSL3_ENC_METHOD DTLSv1_enc_data;

625 #define SSL_IS_DTLS(s) (s->method->version == DTLS1_VERSION)

627 #define IMPLEMENT_tls_meth_func(version, func_name, s_accept, s_connect, \
628                                 s_get_meth) \
629 const SSL_METHOD *func_name(void)  \
630         { \
631         static const SSL_METHOD func_name##_data= { \
632                 version, \
633                 tls1_new, \
634                 tls1_clear, \
635                 tls1_free, \
636                 s_accept, \
637                 s_connect, \
638                 ssl3_read, \
639                 ssl3_peek, \
640                 ssl3_write, \
641                 ssl3_shutdown, \
642                 ssl3_renegotiate, \
643                 ssl3_renegotiate_check, \
644                 ssl3_get_message, \
645                 ssl3_read_bytes, \
646                 ssl3_write_bytes, \
647                 ssl3_dispatch_alert, \
648                 ssl3_ctrl, \
649                 ssl3_ctx_ctrl, \
650                 ssl3_get_cipher_by_char, \
651                 ssl3_put_cipher_by_char, \
652                 ssl3_pending, \
653                 ssl3_num_ciphers, \
654                 ssl3_get_cipher, \
655                 s_get_meth, \
```

```
656                 tls1_default_timeout, \
657                 &TLSv1_enc_data, \
658                 ssl_undefined_void_function, \
659                 ssl3_callback_ctrl, \
660                 ssl3_ctx_callback_ctrl, \
661         }; \
662         return &func_name##_data; \
663         }

665 #define IMPLEMENT_ssl3_meth_func(func_name, s_accept, s_connect, s_get_meth) \
666 const SSL_METHOD *func_name(void)  \
667         { \
668         static const SSL_METHOD func_name##_data= { \
669                 SSL3_VERSION, \
670                 ssl3_new, \
671                 ssl3_clear, \
672                 ssl3_free, \
673                 s_accept, \
674                 s_connect, \
675                 ssl3_read, \
676                 ssl3_peek, \
677                 ssl3_write, \
678                 ssl3_shutdown, \
679                 ssl3_renegotiate, \
680                 ssl3_renegotiate_check, \
681                 ssl3_get_message, \
682                 ssl3_read_bytes, \
683                 ssl3_write_bytes, \
684                 ssl3_dispatch_alert, \
685                 ssl3_ctrl, \
686                 ssl3_ctx_ctrl, \
687                 ssl3_get_cipher_by_char, \
688                 ssl3_put_cipher_by_char, \
689                 ssl3_pending, \
690                 ssl3_num_ciphers, \
691                 ssl3_get_cipher, \
692                 s_get_meth, \
693                 ssl3_default_timeout, \
694                 &SSLv3_enc_data, \
695                 ssl_undefined_void_function, \
696                 ssl3_callback_ctrl, \
697                 ssl3_ctx_callback_ctrl, \
698         }; \
699         return &func_name##_data; \
700         }

702 #define IMPLEMENT_ssl23_meth_func(func_name, s_accept, s_connect, s_get_meth) \
703 const SSL_METHOD *func_name(void)  \
704         { \
705         static const SSL_METHOD func_name##_data= { \
706         TLS1_2_VERSION, \
707         tls1_new, \
708         tls1_clear, \
709         tls1_free, \
710         s_accept, \
711         s_connect, \
712         ssl23_read, \
713         ssl23_peek, \
714         ssl23_write, \
715         ssl_undefined_function, \
716         ssl_undefined_function, \
717         ssl_ok, \
718         ssl3_get_message, \
719         ssl3_read_bytes, \
720         ssl3_write_bytes, \
721         ssl3_dispatch_alert, \
```



```
722         ssl3_ctrl, \
723         ssl3_ctx_ctrl, \
724         ssl23_get_cipher_by_char, \
725         ssl23_put_cipher_by_char, \
726         ssl_undefined_const_function, \
727         ssl23_num_ciphers, \
728         ssl23_get_cipher, \
729         s_get_meth, \
730         ssl23_default_timeout, \
731         &ssl3_undef_enc_method, \
732         ssl_undefined_void_function, \
733         ssl3_callback_ctrl, \
734         ssl3_ctx_callback_ctrl, \
735         }; \
736         return &func_name##_data; \
737         }

739 #define IMPLEMENT_ssl2_meth_func(func_name, s_accept, s_connect, s_get_meth) \
740 const SSL_METHOD *func_name(void)  \
741         { \
742         static const SSL_METHOD func_name##_data= { \
743                 SSL2_VERSION, \
744                 ssl2_new,       /* local */ \
745                 ssl2_clear,     /* local */ \
746                 ssl2_free,      /* local */ \
747                 s_accept, \
748                 s_connect, \
749                 ssl2_read, \
750                 ssl2_peek, \
751                 ssl2_write, \
752                 ssl2_shutdown, \
753                 ssl_ok, /* NULL - renegotiate */ \
754                 ssl_ok, /* NULL - check renegotiate */ \
755                 NULL, /* NULL - ssl_get_message */ \
756                 NULL, /* NULL - ssl_get_record */ \
757                 NULL, /* NULL - ssl_write_bytes */ \
758                 NULL, /* NULL - dispatch_alert */ \
759                 ssl2_ctrl,      /* local */ \
760                 ssl2_ctx_ctrl,  /* local */ \
761                 ssl2_get_cipher_by_char, \
762                 ssl2_put_cipher_by_char, \
763                 ssl2_pending, \
764                 ssl2_num_ciphers, \
765                 ssl2_get_cipher, \
766                 s_get_meth, \
767                 ssl2_default_timeout, \
768                 &ssl3_undef_enc_method, \
769                 ssl_undefined_void_function, \
770                 ssl2_callback_ctrl,     /* local */ \
771                 ssl2_ctx_callback_ctrl, /* local */ \
772         }; \
773         return &func_name##_data; \
774         }

776 #define IMPLEMENT_dtls1_meth_func(func_name, s_accept, s_connect, s_get_meth) \
777 const SSL_METHOD *func_name(void)  \
778         { \
779         static const SSL_METHOD func_name##_data= { \
780                 DTLS1_VERSION, \
781                 dtls1_new, \
782                 dtls1_clear, \
783                 dtls1_free, \
784                 s_accept, \
785                 s_connect, \
786                 ssl3_read, \
787                 ssl3_peek, \
```

```
788                 ssl3_write, \
789                 dtls1_shutdown, \
790                 ssl3_renegotiate, \
791                 ssl3_renegotiate_check, \
792                 dtls1_get_message, \
793                 dtls1_read_bytes, \
794                 dtls1_write_app_data_bytes, \
795                 dtls1_dispatch_alert, \
796                 dtls1_ctrl, \
797                 ssl3_ctx_ctrl, \
798                 ssl3_get_cipher_by_char, \
799                 ssl3_put_cipher_by_char, \
800                 ssl3_pending, \
801                 ssl3_num_ciphers, \
802                 dtls1_get_cipher, \
803                 s_get_meth, \
804                 dtls1_default_timeout, \
805                 &DTLSv1_enc_data, \
806                 ssl_undefined_void_function, \
807                 ssl3_callback_ctrl, \
808                 ssl3_ctx_callback_ctrl, \
809         }; \
810         return &func_name##_data; \
811         }

813 struct openssl_ssl_test_functions
814         {
815         int (*p_ssl_init_wbio_buffer)(SSL *s, int push);
816         int (*p_ssl3_setup_buffers)(SSL *s);
817         int (*p_tls1_process_heartbeat)(SSL *s);
818         int (*p_dtls1_process_heartbeat)(SSL *s);
819         };

821 #ifndef OPENSSL_UNIT_TEST

823 void ssl_clear_cipher_ctx(SSL *s);
824 int ssl_clear_bad_session(SSL *s);
825 CERT *ssl_cert_new(void);
826 CERT *ssl_cert_dup(CERT *cert);
827 int ssl_cert_inst(CERT **o);
828 void ssl_cert_free(CERT *c);
829 SESS_CERT *ssl_sess_cert_new(void);
830 void ssl_sess_cert_free(SESS_CERT *sc);
831 int ssl_set_peer_cert_type(SESS_CERT *c, int type);
832 int ssl_get_new_session(SSL *s, int session);
833 int ssl_get_prev_session(SSL *s, unsigned char *session,int len, const unsigned
834 int ssl_cipher_id_cmp(const SSL_CIPHER *a,const SSL_CIPHER *b);
835 DECLARE_OBJ_BSEARCH_GLOBAL_CMP_FN(SSL_CIPHER, SSL_CIPHER,
836                                   ssl_cipher_id);
837 int ssl_cipher_ptr_id_cmp(const SSL_CIPHER * const *ap,
838                           const SSL_CIPHER * const *bp);
839 STACK_OF(SSL_CIPHER) *ssl_bytes_to_cipher_list(SSL *s,unsigned char *p,int num,
840                                                 STACK_OF(SSL_CIPHER) **skp);
841 int ssl_cipher_list_to_bytes(SSL *s,STACK_OF(SSL_CIPHER) *sk,unsigned char *p,
842                              int (*put_cb)(const SSL_CIPHER *, unsigned char *))
843 STACK_OF(SSL_CIPHER) *ssl_create_cipher_list(const SSL_METHOD *meth,
844                                               STACK_OF(SSL_CIPHER) **pref,
845                                               STACK_OF(SSL_CIPHER) **sorted,
846                                               const char *rule_str);
847 void ssl_update_cache(SSL *s, int mode);
848 int ssl_cipher_get_evp(const SSL_SESSION *s,const EVP_CIPHER **enc,
849                        const EVP_MD **md,int *mac_pkey_type,int *mac_secret_size
850 int ssl_get_handshake_digest(int i,long *mask,const EVP_MD **md);
851 int ssl_verify_cert_chain(SSL *s,STACK_OF(X509) *sk);
852 int ssl_undefined_function(SSL *s);
853 int ssl_undefined_void_function(void);
```



```
854 int ssl_undefined_const_function(const SSL *s);
855 CERT_PKEY *ssl_get_server_send_pkey(const SSL *s);
856 X509 *ssl_get_server_send_cert(const SSL *);
857 EVP_PKEY *ssl_get_sign_pkey(SSL *s,const SSL_CIPHER *c, const EVP_MD **pmd);
858 int ssl_cert_type(X509 *x,EVP_PKEY *pkey);
859 void ssl_set_cert_masks(CERT *c, const SSL_CIPHER *cipher);
860 STACK_OF(SSL_CIPHER) *ssl_get_ciphers_by_id(SSL *s);
861 int ssl_verify_alarm_type(long type);
862 void ssl_load_ciphers(void);
863 int ssl_fill_hello_random(SSL *s, int server, unsigned char *field, int len);

865 int ssl2_enc_init(SSL *s, int client);
866 int ssl2_generate_key_material(SSL *s);
867 void ssl2_enc(SSL *s,int send_data);
868 void ssl2_mac(SSL *s,unsigned char *mac,int send_data);
869 const SSL_CIPHER *ssl2_get_cipher_by_char(const unsigned char *p);
870 int ssl2_put_cipher_by_char(const SSL_CIPHER *c,unsigned char *p);
871 int ssl2_part_read(SSL *s, unsigned long f, int i);
872 int ssl2_do_write(SSL *s);
873 int ssl2_set_certificate(SSL *s, int type, int len, const unsigned char *data);
874 void ssl2_return_error(SSL *s,int reason);
875 void ssl2_write_error(SSL *s);
876 int ssl2_num_ciphers(void);
877 const SSL_CIPHER *ssl2_get_cipher(unsigned int u);
878 int     ssl2_new(SSL *s);
879 void    ssl2_free(SSL *s);
880 int     ssl2_accept(SSL *s);
881 int     ssl2_connect(SSL *s);
882 int     ssl2_read(SSL *s, void *buf, int len);
883 int     ssl2_peek(SSL *s, void *buf, int len);
884 int     ssl2_write(SSL *s, const void *buf, int len);
885 int     ssl2_shutdown(SSL *s);
886 void    ssl2_clear(SSL *s);
887 long    ssl2_ctrl(SSL *s,int cmd, long larg, void *parg);
888 long    ssl2_ctx_ctrl(SSL_CTX *s,int cmd, long larg, void *parg);
889 long    ssl2_callback_ctrl(SSL *s,int cmd, void (*fp)(void));
890 long    ssl2_ctx_callback_ctrl(SSL_CTX *s,int cmd, void (*fp)(void));
891 int     ssl2_pending(const SSL *s);
892 long    ssl2_default_timeout(void );

894 const SSL_CIPHER *ssl3_get_cipher_by_char(const unsigned char *p);
895 int ssl3_put_cipher_by_char(const SSL_CIPHER *c,unsigned char *p);
896 void ssl3_init_finished_mac(SSL *s);
897 int ssl3_send_server_certificate(SSL *s);
898 int ssl3_send_newsession_ticket(SSL *s);
899 int ssl3_send_cert_status(SSL *s);
900 int ssl3_get_finished(SSL *s,int state_a,int state_b);
901 int ssl3_setup_key_block(SSL *s);
902 int ssl3_send_change_cipher_spec(SSL *s,int state_a,int state_b);
903 int ssl3_change_cipher_state(SSL *s,int which);
904 void ssl3_cleanup_key_block(SSL *s);
905 int ssl3_do_write(SSL *s,int type);
906 int ssl3_send_alert(SSL *s,int level, int desc);
907 int ssl3_generate_master_secret(SSL *s, unsigned char *out,
908         unsigned char *p, int len);
909 int ssl3_get_req_cert_type(SSL *s,unsigned char *p);
910 long ssl3_get_message(SSL *s, int st1, int stn, int mt, long max, int *ok);
911 int ssl3_send_finished(SSL *s, int a, int b, const char *sender,int slen);
912 int ssl3_num_ciphers(void);
913 const SSL_CIPHER *ssl3_get_cipher(unsigned int u);
914 int ssl3_renegotiate(SSL *ssl);
915 int ssl3_renegotiate_check(SSL *ssl);
916 int ssl3_dispatch_alert(SSL *s);
917 int ssl3_read_bytes(SSL *s, int type, unsigned char *buf, int len, int peek);
918 int ssl3_write_bytes(SSL *s, int type, const void *buf, int len);
919 int ssl3_final_finish_mac(SSL *s, const char *sender, int slen,unsigned char *p)
```

```
int ssl3_cert_verify_mac(SSL *s, int md_nid, unsigned char *p);
void ssl3_finish_mac(SSL *s, const unsigned char *buf, int len);
int ssl3_enc(SSL *s, int send_data);
int n_ssl3_mac(SSL *ssl, unsigned char *md, int send_data);
void ssl3_free_digest_list(SSL *s);
unsigned long ssl3_output_cert_chain(SSL *s, X509 *x);
SSL_CIPHER *ssl3_choose_cipher(SSL *ssl,STACK_OF(SSL_CIPHER) *clnt,
                               STACK_OF(SSL_CIPHER) *srvr);
int     ssl3_setup_buffers(SSL *s);
int     ssl3_setup_read_buffer(SSL *s);
int     ssl3_setup_write_buffer(SSL *s);
int     ssl3_release_read_buffer(SSL *s);
int     ssl3_release_write_buffer(SSL *s);
int     ssl3_digest_cached_records(SSL *s);
int     ssl3_new(SSL *s);
void    ssl3_free(SSL *s);
int     ssl3_accept(SSL *s);
int     ssl3_connect(SSL *s);
int     ssl3_read(SSL *s, void *buf, int len);
int     ssl3_peek(SSL *s, void *buf, int len);
int     ssl3_write(SSL *s, const void *buf, int len);
int     ssl3_shutdown(SSL *s);
void    ssl3_clear(SSL *s);
long    ssl3_ctrl(SSL *s,int cmd, long larg, void *parg);
long    ssl3_ctx_ctrl(SSL_CTX *s,int cmd, long larg, void *parg);
long    ssl3_callback_ctrl(SSL *s,int cmd, void (*fp)(void));
long    ssl3_ctx_callback_ctrl(SSL_CTX *s,int cmd, void (*fp)(void));
int     ssl3_pending(const SSL *s);

void ssl3_record_sequence_update(unsigned char *seq);
int ssl3_do_change_cipher_spec(SSL *ssl);
long ssl3_default_timeout(void );

int ssl23_num_ciphers(void );
const SSL_CIPHER *ssl23_get_cipher(unsigned int u);
int ssl23_read(SSL *s, void *buf, int len);
int ssl23_peek(SSL *s, void *buf, int len);
int ssl23_write(SSL *s, const void *buf, int len);
int ssl23_put_cipher_by_char(const SSL_CIPHER *c, unsigned char *p);
const SSL_CIPHER *ssl23_get_cipher_by_char(const unsigned char *p);
long ssl23_default_timeout(void );

long tls1_default_timeout(void);
int dtls1_do_write(SSL *s,int type);
int ssl3_read_n(SSL *s, int n, int max, int extend);
int dtls1_read_bytes(SSL *s, int type, unsigned char *buf, int len, int peek);
int ssl3_do_compress(SSL *ssl);
int ssl3_do_uncompress(SSL *ssl);
int ssl3_write_pending(SSL *s, int type, const unsigned char *buf,
        unsigned int len);
unsigned char *dtls1_set_message_header(SSL *s,
        unsigned char *p, unsigned char mt,     unsigned long len,
        unsigned long frag_off, unsigned long frag_len);

int dtls1_write_app_data_bytes(SSL *s, int type, const void *buf, int len);
int dtls1_write_bytes(SSL *s, int type, const void *buf, int len);

int dtls1_send_change_cipher_spec(SSL *s, int a, int b);
int dtls1_send_finished(SSL *s, int a, int b, const char *sender, int slen);
unsigned long dtls1_output_cert_chain(SSL *s, X509 *x);
int dtls1_read_failed(SSL *s, int code);
int dtls1_buffer_message(SSL *s, int ccs);
int dtls1_retransmit_message(SSL *s, unsigned short seq,
        unsigned long frag_off, int *found);
int dtls1_get_queue_priority(unsigned short seq, int is_ccs);
int dtls1_retransmit_buffered_messages(SSL *s);
```



```
void dtls1_clear_record_buffer(SSL *s);
void dtls1_get_message_header(unsigned char *data, struct hm_header_st *msg_hdr)
void dtls1_get_ccs_header(unsigned char *data, struct ccs_header_st *ccs_hdr);
void dtls1_reset_seq_numbers(SSL *s, int rw);
long dtls1_default_timeout(void);
struct timeval* dtls1_get_timeout(SSL *s, struct timeval* timeleft);
int dtls1_check_timeout_num(SSL *s);
int dtls1_handle_timeout(SSL *s);
const SSL_CIPHER *dtls1_get_cipher(unsigned int u);
void dtls1_start_timer(SSL *s);
void dtls1_stop_timer(SSL *s);
int dtls1_is_timer_expired(SSL *s);
void dtls1_double_timeout(SSL *s);
int dtls1_send_newsession_ticket(SSL *s);
unsigned int dtls1_min_mtu(void);

/* some client-only functions */
int ssl3_client_hello(SSL *s);
int ssl3_get_server_hello(SSL *s);
int ssl3_get_certificate_request(SSL *s);
int ssl3_get_new_session_ticket(SSL *s);
int ssl3_get_cert_status(SSL *s);
int ssl3_get_server_done(SSL *s);
int ssl3_send_client_verify(SSL *s);
int ssl3_send_client_certificate(SSL *s);
int ssl_do_client_cert_cb(SSL *s, X509 **px509, EVP_PKEY **ppkey);
int ssl3_send_client_key_exchange(SSL *s);
int ssl3_get_key_exchange(SSL *s);
int ssl3_get_server_certificate(SSL *s);
int ssl3_check_cert_and_algorithm(SSL *s);
#ifndef OPENSSL_NO_TLSEXT
int ssl3_check_finished(SSL *s);
# ifndef OPENSSL_NO_NEXTPROTONEG
int ssl3_send_next_proto(SSL *s);
# endif
#endif

int dtls1_client_hello(SSL *s);
int dtls1_send_client_certificate(SSL *s);
int dtls1_send_client_key_exchange(SSL *s);
int dtls1_send_client_verify(SSL *s);

/* some server-only functions */
int ssl3_get_client_hello(SSL *s);
int ssl3_send_server_hello(SSL *s);
int ssl3_send_hello_request(SSL *s);
int ssl3_send_server_key_exchange(SSL *s);
int ssl3_send_certificate_request(SSL *s);
int ssl3_send_server_done(SSL *s);
int ssl3_check_client_hello(SSL *s);
int ssl3_get_client_certificate(SSL *s);
int ssl3_get_client_key_exchange(SSL *s);
int ssl3_get_cert_verify(SSL *s);
#ifndef OPENSSL_NO_NEXTPROTONEG
int ssl3_get_next_proto(SSL *s);
#endif

int dtls1_send_hello_request(SSL *s);
int dtls1_send_server_hello(SSL *s);
int dtls1_send_server_certificate(SSL *s);
int dtls1_send_server_key_exchange(SSL *s);
int dtls1_send_certificate_request(SSL *s);
int dtls1_send_server_done(SSL *s);
```

```
1052 int ssl23_accept(SSL *s);
1053 int ssl23_connect(SSL *s);
1054 int ssl23_read_bytes(SSL *s, int n);
1055 int ssl23_write_bytes(SSL *s);

1057 int tls1_new(SSL *s);
1058 void tls1_free(SSL *s);
1059 void tls1_clear(SSL *s);
1060 long tls1_ctrl(SSL *s,int cmd, long larg, void *parg);
1061 long tls1_callback_ctrl(SSL *s,int cmd, void (*fp)(void));

1063 int dtls1_new(SSL *s);
1064 int     dtls1_accept(SSL *s);
1065 int     dtls1_connect(SSL *s);
1066 void dtls1_free(SSL *s);
1067 void dtls1_clear(SSL *s);
1068 long dtls1_ctrl(SSL *s,int cmd, long larg, void *parg);
1069 int dtls1_shutdown(SSL *s);

1071 long dtls1_get_message(SSL *s, int st1, int stn, int mt, long max, int *ok);
1072 int dtls1_get_record(SSL *s);
1073 int do_dtls1_write(SSL *s, int type, const unsigned char *buf,
1074         unsigned int len, int create_empty_fragement);
1075 int dtls1_dispatch_alert(SSL *s);
1076 int dtls1_enc(SSL *s, int snd);

1078 int ssl_init_wbio_buffer(SSL *s, int push);
1079 void ssl_free_wbio_buffer(SSL *s);

1081 int tls1_change_cipher_state(SSL *s, int which);
1082 int tls1_setup_key_block(SSL *s);
1083 int tls1_enc(SSL *s, int snd);
1084 int tls1_final_finish_mac(SSL *s,
1085         const char *str, int slen, unsigned char *p);
1086 int tls1_cert_verify_mac(SSL *s, int md_nid, unsigned char *p);
1087 int tls1_mac(SSL *ssl, unsigned char *md, int snd);
1088 int tls1_generate_master_secret(SSL *s, unsigned char *out,
1089         unsigned char *p, int len);
1090 int tls1_export_keying_material(SSL *s, unsigned char *out, size_t olen,
1091         const char *label, size_t llen,
1092         const unsigned char *p, size_t plen, int use_context);
1093 int tls1_alert_code(int code);
1094 int ssl3_alert_code(int code);
1095 int ssl_ok(SSL *s);

1097 #ifndef OPENSSL_NO_ECDH
1098 int ssl_check_srvr_ecc_cert_and_alg(X509 *x, SSL *s);
1099 #endif

1101 SSL_COMP *ssl3_comp_find(STACK_OF(SSL_COMP) *sk, int n);

1103 #ifndef OPENSSL_NO_EC
1104 int tls1_ec_curve_id2nid(int curve_id);
1105 int tls1_ec_nid2curve_id(int nid);
1106 #endif /* OPENSSL_NO_EC */

1108 #ifndef OPENSSL_NO_TLSEXT
1109 unsigned char *ssl_add_clienthello_tlsext(SSL *s, unsigned char *buf, unsigned c
1110 unsigned char *ssl_add_serverhello_tlsext(SSL *s, unsigned char *buf, unsigned c
1111 int ssl_parse_clienthello_tlsext(SSL *s, unsigned char **data, unsigned char *d,
1112 int ssl_parse_serverhello_tlsext(SSL *s, unsigned char **data, unsigned char *d,
1113 int ssl_prepare_clienthello_tlsext(SSL *s);
1114 int ssl_prepare_serverhello_tlsext(SSL *s);
1115 int ssl_check_clienthello_tlsext_early(SSL *s);
1116 int ssl_check_clienthello_tlsext_late(SSL *s);
1117 int ssl_check_serverhello_tlsext(SSL *s);
```



```
1119 #ifndef OPENSSL_NO_HEARTBEATS
1120 int tls1_heartbeat(SSL *s);
1121 int dtls1_heartbeat(SSL *s);
1122 int tls1_process_heartbeat(SSL *s);
1123 int dtls1_process_heartbeat(SSL *s);
1124 #endif

1126 #ifdef OPENSSL_NO_SHA256
1127 #define tlsext_tick_md  EVP_sha1
1128 #else
1129 #define tlsext_tick_md  EVP_sha256
1130 #endif
1131 int tls1_process_ticket(SSL *s, unsigned char *session_id, int len,
1132                                 const unsigned char *limit, SSL_SESSION **ret);

1134 int tls12_get_sigandhash(unsigned char *p, const EVP_PKEY *pk,
1135                                 const EVP_MD *md);
1136 int tls12_get_sigid(const EVP_PKEY *pk);
1137 const EVP_MD *tls12_get_hash(unsigned char hash_alg);

1139 #endif
1140 EVP_MD_CTX* ssl_replace_hash(EVP_MD_CTX **hash,const EVP_MD *md) ;
1141 void ssl_clear_hash_ctx(EVP_MD_CTX **hash);
1142 int ssl_add_serverhello_renegotiate_ext(SSL *s, unsigned char *p, int *len,
1143                                         int maxlen);
1144 int ssl_parse_serverhello_renegotiate_ext(SSL *s, unsigned char *d, int len,
1145                                           int *al);
1146 int ssl_add_clienthello_renegotiate_ext(SSL *s, unsigned char *p, int *len,
1147                                         int maxlen);
1148 int ssl_parse_clienthello_renegotiate_ext(SSL *s, unsigned char *d, int len,
1149                                           int *al);
1150 long ssl_get_algorithm2(SSL *s);
1151 int tls1_process_sigalgs(SSL *s, const unsigned char *data, int dsize);
1152 int tls12_get_req_sig_algs(SSL *s, unsigned char *p);

1154 int ssl_add_clienthello_use_srtp_ext(SSL *s, unsigned char *p, int *len, int max
1155 int ssl_parse_clienthello_use_srtp_ext(SSL *s, unsigned char *d, int len,int *al
1156 int ssl_add_serverhello_use_srtp_ext(SSL *s, unsigned char *p, int *len, int max
1157 int ssl_parse_serverhello_use_srtp_ext(SSL *s, unsigned char *d, int len,int *al

1159 /* s3_cbc.c */
1160 void ssl3_cbc_copy_mac(unsigned char* out,
1161                        const SSL3_RECORD *rec,
1162                        unsigned md_size,unsigned orig_len);
1163 int ssl3_cbc_remove_padding(const SSL* s,
1164                             SSL3_RECORD *rec,
1165                             unsigned block_size,
1166                             unsigned mac_size);
1167 int tls1_cbc_remove_padding(const SSL* s,
1168                             SSL3_RECORD *rec,
1169                             unsigned block_size,
1170                             unsigned mac_size);
1171 char ssl3_cbc_record_digest_supported(const EVP_MD_CTX *ctx);
1172 void ssl3_cbc_digest_record(
1173         const EVP_MD_CTX *ctx,
1174         unsigned char* md_out,
1175         size_t* md_out_size,
1176         const unsigned char header[13],
1177         const unsigned char *data,
1178         size_t data_plus_mac_size,
1179         size_t data_plus_mac_plus_padding_size,
1180         const unsigned char *mac_secret,
1181         unsigned mac_secret_length,
1182         char is_sslv3);
```

```
1184 void tls_fips_digest_extra(
1185         const EVP_CIPHER_CTX *cipher_ctx, EVP_MD_CTX *mac_ctx,
1186         const unsigned char *data, size_t data_len, size_t orig_len);

1188 int srp_verify_server_param(SSL *s, int *al);

1190 #else

1192 #define ssl_init_wbio_buffer SSL_test_functions()->p_ssl_init_wbio_buffer
1193 #define ssl3_setup_buffers SSL_test_functions()->p_ssl3_setup_buffers
1194 #define tls1_process_heartbeat SSL_test_functions()->p_tls1_process_heartbeat
1195 #define dtls1_process_heartbeat SSL_test_functions()->p_dtls1_process_heartbeat

1197 #endif
1198 #endif
1199 #endif /* ! codereview */
```

```
**********************************************************
    4793 Wed Aug 13 19:51:54 2014
new/usr/src/lib/openssl/include/str_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/store/str_locl.h -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2003.
 */
/* ====================================================================
 * Copyright (c) 2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_STORE_LOCL_H
#define HEADER_STORE_LOCL_H
```



```
#include <openssl/crypto.h>
#include <openssl/store.h>

#ifdef  __cplusplus
extern "C" {
#endif

struct store_method_st
        {
        char *name;

        /* All the functions return a positive integer or non-NULL for success
           and 0, a negative integer or NULL for failure */

        /* Initialise the STORE with private data */
        STORE_INITIALISE_FUNC_PTR init;
        /* Initialise the STORE with private data */
        STORE_CLEANUP_FUNC_PTR clean;
        /* Generate an object of a given type */
        STORE_GENERATE_OBJECT_FUNC_PTR generate_object;
        /* Get an object of a given type.  This function isn't really very
           useful since the listing functions (below) can be used for the
           same purpose and are much more general. */
        STORE_GET_OBJECT_FUNC_PTR get_object;
        /* Store an object of a given type. */
        STORE_STORE_OBJECT_FUNC_PTR store_object;
        /* Modify the attributes bound to an object of a given type. */
        STORE_MODIFY_OBJECT_FUNC_PTR modify_object;
        /* Revoke an object of a given type. */
        STORE_HANDLE_OBJECT_FUNC_PTR revoke_object;
        /* Delete an object of a given type. */
        STORE_HANDLE_OBJECT_FUNC_PTR delete_object;
        /* List a bunch of objects of a given type and with the associated
           attributes. */
        STORE_START_OBJECT_FUNC_PTR list_object_start;
        STORE_NEXT_OBJECT_FUNC_PTR list_object_next;
        STORE_END_OBJECT_FUNC_PTR list_object_end;
        STORE_END_OBJECT_FUNC_PTR list_object_endp;
        /* Store-level function to make any necessary update operations. */
        STORE_GENERIC_FUNC_PTR update_store;
        /* Store-level function to get exclusive access to the store. */
        STORE_GENERIC_FUNC_PTR lock_store;
        /* Store-level function to release exclusive access to the store. */
        STORE_GENERIC_FUNC_PTR unlock_store;

        /* Generic control function */
        STORE_CTRL_FUNC_PTR ctrl;
        };

struct store_st
        {
        const STORE_METHOD *meth;
        /* functional reference if 'meth' is ENGINE-provided */
        ENGINE *engine;

        CRYPTO_EX_DATA ex_data;
        int references;
        };
#ifdef  __cplusplus
}
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    5291 Wed Aug 13 19:51:54 2014
new/usr/src/lib/openssl/include/ui_locl.h
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ui/ui.h -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef HEADER_UI_LOCL_H
#define HEADER_UI_LOCL_H
```



```
#include <openssl/ui.h>
#include <openssl/crypto.h>

#ifdef _
#undef _
#endif

struct ui_method_st
        {
        char *name;

        /* All the functions return 1 or non-NULL for success and 0 or NULL
           for failure */

        /* Open whatever channel for this, be it the console, an X window
           or whatever.
           This function should use the ex_data structure to save
           intermediate data. */
        int (*ui_open_session)(UI *ui);

        int (*ui_write_string)(UI *ui, UI_STRING *uis);

        /* Flush the output.  If a GUI dialog box is used, this function can
           be used to actually display it. */
        int (*ui_flush)(UI *ui);

        int (*ui_read_string)(UI *ui, UI_STRING *uis);

        int (*ui_close_session)(UI *ui);

        /* Construct a prompt in a user-defined manner.  object_desc is a
           textual short description of the object, for example "pass phrase",
           and object_name is the name of the object (might be a card name or
           a file name.
           The returned string shall always be allocated on the heap with
           OPENSSL_malloc(), and need to be free'd with OPENSSL_free(). */
        char *(*ui_construct_prompt)(UI *ui, const char *object_desc,
                const char *object_name);
        };

struct ui_string_st
        {
        enum UI_string_types type; /* Input */
        const char *out_string; /* Input */
        int input_flags;        /* Flags from the user */

        /* The following parameters are completely irrelevant for UIT_INFO,
           and can therefore be set to 0 or NULL */
        char *result_buf;       /* Input and Output: If not NULL, user-defined
                                   with size in result_maxsize.  Otherwise, it
                                   may be allocated by the UI routine, meaning
                                   result_minsize is going to be overwritten.*/
        union
                {
                struct
                        {
                        int result_minsize;     /* Input: minimum required
                                                   size of the result.
                                                */
                        int result_maxsize;     /* Input: maximum permitted
                                                   size of the result */

                        const char *test_buf;   /* Input: test string to verify
                                                   against */
                        } string_data;
                struct
```

```
128                         {
129                         const char *action_desc; /* Input */
130                         const char *ok_chars; /* Input */
131                         const char *cancel_chars; /* Input */
132                         } boolean_data;
133                 } _;

135 #define OUT_STRING_FREEABLE 0x01
136         int flags;              /* flags for internal use */
137         };

139 struct ui_st
140         {
141         const UI_METHOD *meth;
142         STACK_OF(UI_STRING) *strings; /* We might want to prompt for more
143                                          than one thing at a time, and
144                                          with different echoing status.  */
145         void *user_data;
146         CRYPTO_EX_DATA ex_data;

148 #define UI_FLAG_REDOABLE        0x0001
149 #define UI_FLAG_PRINT_ERRORS    0x0100
150         int flags;
151         };

153 #endif
154 #endif /* ! codereview */
```

****
3680 Wed Aug 13 19:51:55 2014
new/usr/src/lib/openssl/libsunw_crypto/LPdir_unix.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* $LP: LPlib/source/LPdir_unix.c,v 1.11 2004/09/23 22:07:22 _cvs_levitte Exp $
/*
 * Copyright (c) 2004, Richard Levitte <richard@levitte.org>
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * ''AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
 * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
 * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

#include <stddef.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>
#ifndef LPDIR_H
#include "LPdir.h"
#endif

/* The POSIXly macro for the maximum number of characters in a file path
   is NAME_MAX.  However, some operating systems use PATH_MAX instead.
   Therefore, it seems natural to first check for PATH_MAX and use that,
   and if it doesn't exist, use NAME_MAX. */
#if defined(PATH_MAX)
# define LP_ENTRY_SIZE PATH_MAX
#elif defined(NAME_MAX)
# define LP_ENTRY_SIZE NAME_MAX
#endif

/* Of course, there's the possibility that neither PATH_MAX nor NAME_MAX
   exist.  It's also possible that NAME_MAX exists but is define to a
   very small value (HP-UX offers 14), so we need to check if we got a
   result, and if it meets a minimum standard, and create or change it
   if not. */
#if !defined(LP_ENTRY_SIZE) || LP_ENTRY_SIZE<255
# undef LP_ENTRY_SIZE
# define LP_ENTRY_SIZE 255
#endif

struct LP_dir_context_st
{
  DIR *dir;
```



```
  char entry_name[LP_ENTRY_SIZE+1];
};

const char *LP_find_file(LP_DIR_CTX **ctx, const char *directory)
{
  struct dirent *direntry = NULL;

  if (ctx == NULL || directory == NULL)
    {
      errno = EINVAL;
      return 0;
    }

  errno = 0;
  if (*ctx == NULL)
    {
      *ctx = (LP_DIR_CTX *)malloc(sizeof(LP_DIR_CTX));
      if (*ctx == NULL)
        {
          errno = ENOMEM;
          return 0;
        }
      memset(*ctx, '\0', sizeof(LP_DIR_CTX));

      (*ctx)->dir = opendir(directory);
      if ((*ctx)->dir == NULL)
        {
          int save_errno = errno; /* Probably not needed, but I'm paranoid */
          free(*ctx);
          *ctx = NULL;
          errno = save_errno;
          return 0;
        }
    }

  direntry = readdir((*ctx)->dir);
  if (direntry == NULL)
    {
      return 0;
    }

  strncpy((*ctx)->entry_name, direntry->d_name, sizeof((*ctx)->entry_name) - 1);
  (*ctx)->entry_name[sizeof((*ctx)->entry_name) - 1] = '\0';
  return (*ctx)->entry_name;
}

int LP_find_file_end(LP_DIR_CTX **ctx)
{
  if (ctx != NULL && *ctx != NULL)
    {
      int ret = closedir((*ctx)->dir);

      free(*ctx);
      switch (ret)
        {
        case 0:
          return 1;
        case -1:
          return 0;
        default:
          break;
        }
    }
  errno = EINVAL;
  return 0;
}
```

```
128 #endif /* ! codereview */
```

```
**********************************************************
    1319 Wed Aug 13 19:51:55 2014
new/usr/src/lib/openssl/libsunw_crypto/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#

include $(SRC)/lib/Makefile.lib

SUBDIRS = .WAIT $(MACH) $(BUILD64) $(MACH64)

# conditional assignments
all :=          TARGET= all
install :=      TARGET= install
clean :=        TARGET= clean
clobber :=      TARGET= clobber
lint :=         TARGET= lint
_msg :=         TARGET= _msg

.KEEP_STATE:

all install clean clobber lint: $(SUBDIRS)

_msg check:

$(MACH) $(MACH64): FRC
        @cd $@; pwd; $(MAKE) $(TARGET)

FRC:

#endif /* ! codereview */
```

```
**********************************************************
   15756 Wed Aug 13 19:51:55 2014
new/usr/src/lib/openssl/libsunw_crypto/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2014 Alexander Pyhalov
# Use is subject to license terms.
#

LIBRARY=        libsunw_crypto.a
VERS=           .1

COMMON_OBJECTS =        cpt_err.o  \
                        cryptlib.o \
                        cversion.o \
                        ebcdic.o   \
                        ex_data.o  \
                        fips_ers.o \
                        mem_dbg.o  \
                        mem.o      \
                        o_dir.o    \
                        o_fips.o   \
                        o_init.o   \
                        o_str.o    \
                        o_time.o   \
                        uid.o
OBJECTS +=      $(COMMON_OBJECTS)

# aes/*
AES_OBJECTS =   aes_cfb.o  \
                aes_ctr.o  \
                aes_ecb.o  \
                aes_ige.o  \
                aes_misc.o \
                aes_ofb.o  \
                aes_wrap.o
OBJECTS +=      $(AES_OBJECTS)

# asn1/*
ASN1_OBJECTS = a_bitstr.o  \
                a_bool.o    \
                a_bytes.o   \
                a_d2i_fp.o  \
                a_digest.o  \
```



```
                a_dup.o     \
                a_enum.o    \
                a_gentm.o   \
                a_i2d_fp.o  \
                a_int.o     \
                a_mbstr.o   \
                a_object.o  \
                a_octet.o   \
                a_print.o   \
                a_set.o     \
                a_sign.o    \
                a_strex.o   \
                a_strnid.o  \
                a_time.o    \
                a_type.o    \
                a_utctm.o   \
                a_utf8.o    \
                a_verify.o  \
                ameth_lib.o \
                asn1_err.o  \
                asn1_gen.o  \
                asn1_lib.o  \
                asn1_par.o  \
                asn_mime.o  \
                asn_moid.o  \
                asn_pack.o  \
                bio_asn1.o  \
                bio_ndef.o  \
                d2i_pr.o    \
                d2i_pu.o    \
                evp_asn1.o  \
                f_enum.o    \
                f_int.o     \
                f_string.o  \
                i2d_pr.o    \
                i2d_pu.o    \
                n_pkey.o    \
                nsseq.o     \
                p5_pbe.o    \
                p5_pbev2.o  \
                p8_pkey.o   \
                t_bitst.o   \
                t_crl.o     \
                t_pkey.o    \
                t_req.o     \
                t_spki.o    \
                t_x509.o    \
                t_x509a.o   \
                tasn_dec.o  \
                tasn_enc.o  \
                tasn_fre.o  \
                tasn_new.o  \
                tasn_prn.o  \
                tasn_typ.o  \
                tasn_utl.o  \
                x_algor.o   \
                x_attrib.o  \
                x_bignum.o  \
                x_crl.o     \
                x_exten.o   \
                x_info.o    \
                x_long.o    \
                x_name.o    \
                x_nx509.o   \
                x_pkey.o    \
                x_pubkey.o  \
```

```
128                 x_req.o     \
129                 x_sig.o     \
130                 x_spki.o    \
131                 x_val.o     \
132                 x_x509.o    \
133                 x_x509a.o
134 OBJECTS += $(ASN1_OBJECTS)

136 # bf/*
137 BF_OBJECTS =    bf_cfb64.o \
138                 bf_ecb.o   \
139                 bf_ofb64.o \
140                 bf_skey.o
141 OBJECTS += $(BF_OBJECTS)

143 # bio/*
144 BIO_OBJECTS =   b_dump.o        \
145                 b_print.o       \
146                 b_sock.o        \
147                 bf_buff.o       \
148                 bf_nbio.o       \
149                 bf_null.o       \
150                 bio_cb.o        \
151                 bio_err.o       \
152                 bio_lib.o       \
153                 bss_acpt.o      \
154                 bss_bio.o       \
155                 bss_conn.o      \
156                 bss_dgram.o     \
157                 bss_fd.o        \
158                 bss_file.o      \
159                 bss_log.o       \
160                 bss_mem.o       \
161                 bss_null.o      \
162                 bss_sock.o
163 OBJECTS += $(BIO_OBJECTS)

165 # bn/*
166 BN_OBJECTS =    bn_add.o        \
167                 bn_blind.o      \
168                 bn_const.o      \
169                 bn_ctx.o        \
170                 bn_depr.o       \
171                 bn_div.o        \
172                 bn_err.o        \
173                 bn_exp.o        \
174                 bn_exp2.o       \
175                 bn_gcd.o        \
176                 bn_gf2m.o       \
177                 bn_kron.o       \
178                 bn_lib.o        \
179                 bn_mod.o        \
180                 bn_mont.o       \
181                 bn_mpi.o        \
182                 bn_mul.o        \
183                 bn_nist.o       \
184                 bn_prime.o      \
185                 bn_print.o      \
186                 bn_rand.o       \
187                 bn_recp.o       \
188                 bn_shift.o      \
189                 bn_sqr.o        \
190                 bn_sqrt.o       \
191                 bn_word.o       \
192                 bn_x931p.o
193 OBJECTS += $(BN_OBJECTS)
```



```
195 # buffer/*
196 BUFFER_OBJECTS =        buf_err.o       \
197                         buf_str.o       \
198                         buffer.o
199 OBJECTS += $(BUFFER_OBJECTS)

201 # camellia/*
202 CAMELLIA_OBJECTS =      cmll_cfb.o      \
203                         cmll_ctr.o      \
204                         cmll_ecb.o      \
205                         cmll_ofb.o      \
206                         cmll_utl.o
207 OBJECTS += $(CAMELLIA_OBJECTS)

209 # cast/*
210 CAST_OBJECTS =  c_cfb64.o       \
211                 c_ecb.o         \
212                 c_enc.o         \
213                 c_ofb64.o       \
214                 c_skey.o
215 OBJECTS += $(CAST_OBJECTS)

217 # cmac/*
218 CMAC_OBJECTS =  cm_ameth.o      \
219                 cm_pmeth.o      \
220                 cmac.o
221 OBJECTS += $(CMAC_OBJECTS)

223 # cms/*
224 CMS_OBJECTS =   cms_asn1.o      \
225                 cms_att.o       \
226                 cms_cd.o        \
227                 cms_dd.o        \
228                 cms_enc.o       \
229                 cms_env.o       \
230                 cms_err.o       \
231                 cms_ess.o       \
232                 cms_io.o        \
233                 cms_lib.o       \
234                 cms_pwri.o      \
235                 cms_sd.o        \
236                 cms_smime.o
237 OBJECTS += $(CMS_OBJECTS)

239 # comp/*
240 COMP_OBJECTS =  c_rle.o         \
241                 c_zlib.o        \
242                 comp_err.o      \
243                 comp_lib.o
244 OBJECTS += $(COMP_OBJECTS)

246 # conf/*
247 CONF_OBJECTS =  conf_api.o      \
248                 conf_def.o      \
249                 conf_err.o      \
250                 conf_lib.o      \
251                 conf_mall.o     \
252                 conf_mod.o      \
253                 conf_sap.o
254 OBJECTS += $(CONF_OBJECTS)

256 # des/*
257 DES_OBJECTS =   cbc_cksm.o      \
258                 cbc_enc.o       \
259                 cfb_enc.o       \
```

```
260                 cfb64ede.o      \
261                 cfb64enc.o      \
262                 des_old.o       \
263                 des_old2.o      \
264                 ecb_enc.o       \
265                 ecb3_enc.o      \
266                 ede_cbcm_enc.o  \
267                 enc_read.o      \
268                 enc_writ.o      \
269                 fcrypt.o        \
270                 ofb_enc.o       \
271                 ofb64ede.o      \
272                 ofb64enc.o      \
273                 pcbc_enc.o      \
274                 qud_cksm.o      \
275                 rand_key.o      \
276                 read2pwd.o      \
277                 rpc_enc.o       \
278                 set_key.o       \
279                 str2key.o       \
280                 xcbc_enc.o
281 OBJECTS += $(DES_OBJECTS)

283 # dh/*
284 DH_OBJECTS =    dh_ameth.o      \
285                 dh_asn1.o       \
286                 dh_check.o      \
287                 dh_depr.o       \
288                 dh_err.o        \
289                 dh_gen.o        \
290                 dh_key.o        \
291                 dh_lib.o        \
292                 dh_pmeth.o      \
293                 dh_prn.o
294 OBJECTS += $(DH_OBJECTS)

296 # dsa/*
297 DSA_OBJECTS =   dsa_ameth.o     \
298                 dsa_asn1.o      \
299                 dsa_depr.o      \
300                 dsa_err.o       \
301                 dsa_gen.o       \
302                 dsa_key.o       \
303                 dsa_lib.o       \
304                 dsa_ossl.o      \
305                 dsa_pmeth.o     \
306                 dsa_prn.o       \
307                 dsa_sign.o      \
308                 dsa_vrf.o
309 OBJECTS += $(DSA_OBJECTS)

311 # dso/*
312 DSO_OBJECTS =   dso_beos.o      \
313                 dso_dl.o        \
314                 dso_dlfcn.o     \
315                 dso_err.o       \
316                 dso_lib.o       \
317                 dso_null.o      \
318                 dso_openssl.o   \
319                 dso_vms.o       \
320                 dso_win32.o
321 OBJECTS += $(DSO_OBJECTS)

323 # engine/*
324 ENGINE_OBJECTS =        eng_all.o       \
325                         eng_cnf.o       \
```



```
326                         eng_cryptodev.o \
327                         eng_ctrl.o      \
328                         eng_dyn.o       \
329                         eng_err.o       \
330                         eng_fat.o       \
331                         eng_init.o      \
332                         eng_lib.o       \
333                         eng_list.o      \
334                         eng_openssl.o   \
335                         eng_pkey.o      \
336                         eng_rdrand.o    \
337                         eng_rsax.o      \
338                         eng_table.o     \
339                         hw_pk11.o       \
340                         hw_pk11_pub.o   \
341                         tb_asnmth.o     \
342                         tb_cipher.o     \
343                         tb_dh.o \
344                         tb_digest.o     \
345                         tb_dsa.o        \
346                         tb_ecdh.o       \
347                         tb_ecdsa.o      \
348                         tb_pkmeth.o     \
349                         tb_rand.o       \
350                         tb_rsa.o        \
351                         tb_store.o
352 OBJECTS += $(ENGINE_OBJECTS)

354 # err/*
355 ERR_OBJECTS =   err_all.o       \
356                 err_prn.o       \
357                 err.o
358 OBJECTS += $(ERR_OBJECTS)

360 # evp/*
361 EVP_OBJECTS =   bio_b64.o               \
362                 bio_enc.o               \
363                 bio_md.o                \
364                 bio_ok.o                \
365                 c_all.o                 \
366                 c_allc.o                \
367                 c_alld.o                \
368                 digest.o                \
369                 e_aes.o                 \
370                 e_aes_cbc_hmac_sha1.o   \
371                 e_bf.o                  \
372                 e_camellia.o            \
373                 e_cast.o                \
374                 e_des.o                 \
375                 e_des3.o                \
376                 e_idea.o                \
377                 e_null.o                \
378                 e_old.o                 \
379                 e_rc2.o                 \
380                 e_rc4.o                 \
381                 e_rc4_hmac_md5.o        \
382                 e_rc5.o                 \
383                 e_seed.o                \
384                 e_xcbc_d.o              \
385                 encode.o                \
386                 evp_acnf.o              \
387                 evp_cnf.o               \
388                 evp_enc.o               \
389                 evp_err.o               \
390                 evp_fips.o              \
391                 evp_key.o               \
```

```
392                 evp_lib.o               \
393                 evp_pbe.o               \
394                 evp_pkey.o              \
395                 m_dss.o                 \
396                 m_dss1.o                \
397                 m_ecdsa.o               \
398                 m_md2.o                 \
399                 m_md4.o                 \
400                 m_md5.o                 \
401                 m_mdc2.o                \
402                 m_null.o                \
403                 m_ripemd.o              \
404                 m_sha.o                 \
405                 m_sha1.o                \
406                 m_sigver.o              \
407                 m_wp.o                  \
408                 names.o                 \
409                 p5_crpt.o               \
410                 p5_crpt2.o              \
411                 p_dec.o                 \
412                 p_enc.o                 \
413                 p_lib.o                 \
414                 p_open.o                \
415                 p_seal.o                \
416                 p_sign.o                \
417                 p_verify.o              \
418                 pmeth_fn.o              \
419                 pmeth_gn.o              \
420                 pmeth_lib.o
421 OBJECTS += $(EVP_OBJECTS)

423 # hmac/*
424 HMAC_OBJECTS =  hm_ameth.o      \
425                 hm_pmeth.o      \
426                 hmac.o
427 OBJECTS += $(HMAC_OBJECTS)

429 # krb5/*
430 KRB5_OBJECTS =  krb5_asn.o
431 OBJECTS += $(KRB5_OBJECTS)

433 # lhash/*
434 LHASH_OBJECTS = lh_stats.o      \
435                 lhash.o
436 OBJECTS += $(LHASH_OBJECTS)

438 # md2/*
439 MD2_OBJECTS =   md2_dgst.o      \
440                 md2_one.o
441 OBJECTS += $(MD2_OBJECTS)

443 # md4/*
444 MD4_OBJECTS =   md4_dgst.o      \
445                 md4_one.o
446 OBJECTS += $(MD4_OBJECTS)

448 # md5/*
449 MD5_OBJECTS =   md5_dgst.o      \
450                 md5_one.o
451 OBJECTS += $(MD5_OBJECTS)

453 # modes/*
454 MODES_OBJECTS = cbc128.o        \
455                 ccm128.o        \
456                 cfb128.o        \
457                 ctr128.o        \
```



```
458                 cts128.o        \
459                 gcm128.o        \
460                 ofb128.o        \
461                 xts128.o
462 OBJECTS += $(MODES_OBJECTS)

464 # objects/*
465 OBJECTS_OBJECTS =       o_names.o       \
466                         obj_dat.o       \
467                         obj_err.o       \
468                         obj_lib.o       \
469                         obj_xref.o
470 OBJECTS += $(OBJECTS_OBJECTS)

472 # ocsp/*
473 OCSP_OBJECTS =  ocsp_asn.o      \
474                 ocsp_cl.o       \
475                 ocsp_err.o      \
476                 ocsp_ext.o      \
477                 ocsp_ht.o       \
478                 ocsp_lib.o      \
479                 ocsp_prn.o      \
480                 ocsp_srv.o      \
481                 ocsp_vfy.o
482 OBJECTS += $(OCSP_OBJECTS)

484 # pem/*
485 PEM_OBJECTS =   pem_all.o       \
486                 pem_err.o       \
487                 pem_info.o      \
488                 pem_lib.o       \
489                 pem_oth.o       \
490                 pem_pk8.o       \
491                 pem_pkey.o      \
492                 pem_seal.o      \
493                 pem_sign.o      \
494                 pem_x509.o      \
495                 pem_xaux.o      \
496                 pvkfmt.o
497 OBJECTS += $(PEM_OBJECTS)

499 # pkcs12/*
500 PKCS12_OBJECTS =        p12_add.o       \
501                         p12_asn.o       \
502                         p12_attr.o      \
503                         p12_crpt.o      \
504                         p12_crt.o       \
505                         p12_decr.o      \
506                         p12_init.o      \
507                         p12_key.o       \
508                         p12_kiss.o      \
509                         p12_mutl.o      \
510                         p12_npas.o      \
511                         p12_p8d.o       \
512                         p12_p8e.o       \
513                         p12_utl.o       \
514                         pk12err.o
515 OBJECTS += $(PKCS12_OBJECTS)

517 # pkcs7/*
518 PKCS7_OBJECTS = bio_pk7.o       \
519                 pk7_asn1.o      \
520                 pk7_attr.o      \
521                 pk7_doit.o      \
522                 pk7_lib.o       \
523                 pk7_mime.o      \
```

```
524                 pk7_smime.o     \
525                 pkcs7err.o
526 OBJECTS += $(PKCS7_OBJECTS)

528 # pqueue/*
529 PQUEUE_OBJECTS = pqueue.o
530 OBJECTS += $(PQUEUE_OBJECTS)

532 # rand/*
533 RAND_OBJECTS =  md_rand.o       \
534                 rand_egd.o      \
535                 rand_err.o      \
536                 rand_lib.o      \
537                 rand_nw.o       \
538                 rand_os2.o      \
539                 rand_unix.o     \
540                 rand_win.o      \
541                 randfile.o
542 OBJECTS += $(RAND_OBJECTS)

544 # rc2/*
545 RC2_OBJECTS =   rc2_cbc.o       \
546                 rc2_ecb.o       \
547                 rc2_skey.o      \
548                 rc2cfb64.o      \
549                 rc2ofb64.o
550 OBJECTS += $(RC2_OBJECTS)

552 # rc4/*
553 RC4_OBJECTS =   rc4_utl.o
554 OBJECTS += $(RC4_OBJECTS)

556 # ripemd/*
557 RIPEMD_OBJECTS =        rmd_dgst.o      \
558                         rmd_one.o
559 OBJECTS += $(RIPEMD_OBJECTS)

561 # rsa/*
562 RSA_OBJECTS =   rsa_ameth.o     \
563                 rsa_asn1.o      \
564                 rsa_chk.o       \
565                 rsa_crpt.o      \
566                 rsa_depr.o      \
567                 rsa_eay.o       \
568                 rsa_err.o       \
569                 rsa_gen.o       \
570                 rsa_lib.o       \
571                 rsa_none.o      \
572                 rsa_null.o      \
573                 rsa_oaep.o      \
574                 rsa_pk1.o       \
575                 rsa_pmeth.o     \
576                 rsa_prn.o       \
577                 rsa_pss.o       \
578                 rsa_saos.o      \
579                 rsa_sign.o      \
580                 rsa_ssl.o       \
581                 rsa_x931.o
582 OBJECTS += $(RSA_OBJECTS)

584 # sha/*
585 SHA_OBJECTS =   sha1_one.o      \
586                 sha1dgst.o      \
587                 sha256.o        \
588                 sha512.o        \
589                 sha_dgst.o      \
```



```
590                 sha_one.o
591 OBJECTS += $(SHA_OBJECTS)

593 # srp/*
594 SRP_OBJECTS =   srp_lib.o       \
595                 srp_vfy.o
596 OBJECTS += $(SRP_OBJECTS)

598 # stack/*
599 STACK_OBJECTS = stack.o
600 OBJECTS += $(STACK_OBJECTS)

602 # ts/*
603 TS_OBJECTS =    ts_asn1.o       \
604                 ts_conf.o       \
605                 ts_err.o        \
606                 ts_lib.o        \
607                 ts_req_print.o  \
608                 ts_req_utils.o  \
609                 ts_rsp_print.o  \
610                 ts_rsp_sign.o   \
611                 ts_rsp_utils.o  \
612                 ts_rsp_verify.o \
613                 ts_verify_ctx.o
614 OBJECTS += $(TS_OBJECTS)

616 # txt_db/*
617 TXT_DB_OBJECTS =        txt_db.o
618 OBJECTS += $(TXT_DB_OBJECTS)

620 # ui/*
621 UI_OBJECTS =    ui_compat.o     \
622                 ui_err.o        \
623                 ui_lib.o        \
624                 ui_openssl.o    \
625                 ui_util.o
626 OBJECTS += $(UI_OBJECTS)

628 # x509/*
629 X509_OBJECTS =  by_dir.o        \
630                 by_file.o       \
631                 x_all.o         \
632                 x509_att.o      \
633                 x509_cmp.o      \
634                 x509_d2.o       \
635                 x509_def.o      \
636                 x509_err.o      \
637                 x509_ext.o      \
638                 x509_lu.o       \
639                 x509_obj.o      \
640                 x509_r2x.o      \
641                 x509_req.o      \
642                 x509_set.o      \
643                 x509_trs.o      \
644                 x509_txt.o      \
645                 x509_v3.o       \
646                 x509_vfy.o      \
647                 x509_vpm.o      \
648                 x509cset.o      \
649                 x509name.o      \
650                 x509rset.o      \
651                 x509spki.o      \
652                 x509type.o
653 OBJECTS += $(X509_OBJECTS)

655 # x509v3/*
```

```
X509V3_OBJECTS =        pcy_cache.o     \
                        pcy_data.o      \
                        pcy_lib.o       \
                        pcy_map.o       \
                        pcy_node.o      \
                        pcy_tree.o      \
                        v3_addr.o       \
                        v3_akey.o       \
                        v3_akeya.o      \
                        v3_alt.o        \
                        v3_asid.o       \
                        v3_bcons.o      \
                        v3_bitst.o      \
                        v3_conf.o       \
                        v3_cpols.o      \
                        v3_crld.o       \
                        v3_enum.o       \
                        v3_extku.o      \
                        v3_genn.o       \
                        v3_ia5.o        \
                        v3_info.o       \
                        v3_int.o        \
                        v3_lib.o        \
                        v3_ncons.o      \
                        v3_ocsp.o       \
                        v3_pci.o        \
                        v3_pcia.o       \
                        v3_pcons.o      \
                        v3_pku.o        \
                        v3_pmaps.o      \
                        v3_prn.o        \
                        v3_purp.o       \
                        v3_skey.o       \
                        v3_sxnet.o      \
                        v3_utl.o        \
                        v3err.o
OBJECTS += $(X509V3_OBJECTS)

# include library definitions
include $(SRC)/lib/Makefile.lib

CLOBBERFILES += $(LIBLINKS)

LIBS =          $(DYNLIB)

LDLIBS += -lsocket -lnsl -lc

LINTFLAGS =     -uxn
LINTFLAGS64 =   $(LINTFLAGS) -m64
LINTOUT=        lint.out
LINTSRC =       $(LINTLIB:%.ln=%)
ROOTLINTDIR =   $(ROOTLIBDIR)
ROOTLINT =      $(LINTSRC:%=$(ROOTLINTDIR)/%)

CPPFLAGS +=     -I.. \
                -I$(SRC)/lib/openssl/include

CPPFLAGS +=     -D_REENTRANT
CPPFLAGS +=     -DOPENSSL_THREADS
CPPFLAGS +=     -DOPENSSL_PIC
CPPFLAGS +=     -DDSO_DLFCN
CPPFLAGS +=     -DHAVE_DLFCN_H
CPPFLAGS +=     -DSOLARIS_OPENSSL
CPPFLAGS +=     -DNO_WINDOWS_BRAINDEATH
CPPFLAGS +=     -DOPENSSL_BN_ASM_GF2m
CPPFLAGS +=     -DSHA1_ASM
```



```
CPPFLAGS +=     -DSHA256_ASM
CPPFLAGS +=     -DSHA512_ASM
CPPFLAGS +=     -DMD5_ASM
CPPFLAGS +=     -DAES_ASM
CPPFLAGS +=     -DVPAES_ASM
CPPFLAGS +=     -DGHASH_ASM
CPPFLAGS +=     -DVPAES_ASM
CPPFLAGS +=     -DOPENSSL_BN_ASM_MONT

CFLAGS +=       $(CCVERBOSE)

CERRWARN +=     -_gcc=-Wno-switch
CERRWARN +=     -erroff=E_CONST_PROMOTED_UNSIGNED_LONG
CERRWARN +=     -erroff=E_END_OF_LOOP_CODE_NOT_REACHED

$(LINTLIB) :=   LINTFLAGS = -nvx -I..
$(LINTLIB) :=   LINTFLAGS64 = -nvx -m64 -I..

BUILD.perl =    $(PERL)  $< elf $(PERL_CPPFLAGS) > $@

.KEEP_STATE:

all : $(LIBS)

lint :  lintcheck

# include library targets
include $(SRC)/lib/Makefile.targ

pics/%.o:       ../%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../aes/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../asn1/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../bf/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../bio/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../bn/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../buffer/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../camellia/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../cast/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

pics/%.o:       ../cmac/%.c
```

```
788         $(COMPILE.c) -o $@ $<
789         $(POST_PROCESS_O)

791 pics/%.o:       ../cms/%.c
792         $(COMPILE.c) -o $@ $<
793         $(POST_PROCESS_O)

795 pics/%.o:       ../comp/%.c
796         $(COMPILE.c) -o $@ $<
797         $(POST_PROCESS_O)

799 pics/%.o:       ../conf/%.c
800         $(COMPILE.c) -o $@ $<
801         $(POST_PROCESS_O)

803 pics/%.o:       ../des/%.c
804         $(COMPILE.c) -o $@ $<
805         $(POST_PROCESS_O)

807 pics/%.o:       ../dh/%.c
808         $(COMPILE.c) -o $@ $<
809         $(POST_PROCESS_O)

811 pics/%.o:       ../dsa/%.c
812         $(COMPILE.c) -o $@ $<
813         $(POST_PROCESS_O)

815 pics/%.o:       ../dso/%.c
816         $(COMPILE.c) -o $@ $<
817         $(POST_PROCESS_O)

819 pics/%.o:       ../engine/%.c
820         $(COMPILE.c) -o $@ $<
821         $(POST_PROCESS_O)

823 pics/%.o:       ../err/%.c
824         $(COMPILE.c) -o $@ $<
825         $(POST_PROCESS_O)

827 pics/%.o:       ../evp/%.c
828         $(COMPILE.c) -o $@ $<
829         $(POST_PROCESS_O)

831 pics/%.o:       ../hmac/%.c
832         $(COMPILE.c) -o $@ $<
833         $(POST_PROCESS_O)

835 pics/%.o:       ../krb5/%.c
836         $(COMPILE.c) -o $@ $<
837         $(POST_PROCESS_O)

839 pics/%.o:       ../lhash/%.c
840         $(COMPILE.c) -o $@ $<
841         $(POST_PROCESS_O)

843 pics/%.o:       ../md2/%.c
844         $(COMPILE.c) -o $@ $<
845         $(POST_PROCESS_O)

847 pics/%.o:       ../md4/%.c
848         $(COMPILE.c) -o $@ $<
849         $(POST_PROCESS_O)

851 pics/%.o:       ../md5/%.c
852         $(COMPILE.c) -o $@ $<
853         $(POST_PROCESS_O)
```



```
855 pics/%.o:       ../modes/%.c
856         $(COMPILE.c) -o $@ $<
857         $(POST_PROCESS_O)

859 pics/%.o:       ../objects/%.c
860         $(COMPILE.c) -o $@ $<
861         $(POST_PROCESS_O)

863 pics/%.o:       ../ocsp/%.c
864         $(COMPILE.c) -o $@ $<
865         $(POST_PROCESS_O)

867 pics/%.o:       ../pem/%.c
868         $(COMPILE.c) -o $@ $<
869         $(POST_PROCESS_O)

871 pics/%.o:       ../pkcs12/%.c
872         $(COMPILE.c) -o $@ $<
873         $(POST_PROCESS_O)

875 pics/%.o:       ../pkcs7/%.c
876         $(COMPILE.c) -o $@ $<
877         $(POST_PROCESS_O)

879 pics/%.o:       ../pqueue/%.c
880         $(COMPILE.c) -o $@ $<
881         $(POST_PROCESS_O)

883 pics/%.o:       ../rand/%.c
884         $(COMPILE.c) -o $@ $<
885         $(POST_PROCESS_O)

887 pics/%.o:       ../rc2/%.c
888         $(COMPILE.c) -o $@ $<
889         $(POST_PROCESS_O)

891 pics/%.o:       ../rc4/%.c
892         $(COMPILE.c) -o $@ $<
893         $(POST_PROCESS_O)

895 pics/%.o:       ../ripemd/%.c
896         $(COMPILE.c) -o $@ $<
897         $(POST_PROCESS_O)

899 pics/%.o:       ../rsa/%.c
900         $(COMPILE.c) -o $@ $<
901         $(POST_PROCESS_O)

903 pics/%.o:       ../sha/%.c
904         $(COMPILE.c) -o $@ $<
905         $(POST_PROCESS_O)

907 pics/%.o:       ../srp/%.c
908         $(COMPILE.c) -o $@ $<
909         $(POST_PROCESS_O)

911 pics/%.o:       ../stack/%.c
912         $(COMPILE.c) -o $@ $<
913         $(POST_PROCESS_O)

915 pics/%.o:       ../ts/%.c
916         $(COMPILE.c) -o $@ $<
917         $(POST_PROCESS_O)

919 pics/%.o:       ../txt_db/%.c
```

```
920         $(COMPILE.c) -o $@ $<
921         $(POST_PROCESS_O)

923 pics/%.o:       ../ui/%.c
924         $(COMPILE.c) -o $@ $<
925         $(POST_PROCESS_O)

927 pics/%.o:       ../x509/%.c
928         $(COMPILE.c) -o $@ $<
929         $(POST_PROCESS_O)

931 pics/%.o:       ../x509v3/%.c
932         $(COMPILE.c) -o $@ $<
933         $(POST_PROCESS_O)

935 pics/%.o:       %.s
936         $(COMPILE.c) -o $@ $<

938 %.s:    ../pl/%.pl
939         $(BUILD.perl)

941 $(ROOTLINTDIR)/%: ../%
942         $(INS.file)
943 #endif /* ! codereview */
```

```
**********************************************************
    3586 Wed Aug 13 19:51:55 2014
new/usr/src/lib/openssl/libsunw_crypto/aes/aes_cfb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes_cfb.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2002-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/aes.h>
#include <openssl/modes.h>

/* The input and output encrypted as though 128bit cfb mode is being
 * used.  The extra state information to record how much of the
 * 128bit block we have used is contained in *num;
 */

void AES_cfb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
```



```
        unsigned char *ivec, int *num, const int enc) {

        CRYPTO_cfb128_encrypt(in,out,length,key,ivec,num,enc,(block128_f)AES_enc
}

/* N.B. This expects the input to be packed, MS bit first */
void AES_cfb1_encrypt(const unsigned char *in, unsigned char *out,
                      size_t length, const AES_KEY *key,
                      unsigned char *ivec, int *num, const int enc)
    {
    CRYPTO_cfb128_1_encrypt(in,out,length,key,ivec,num,enc,(block128_f)AES_encry
    }

void AES_cfb8_encrypt(const unsigned char *in, unsigned char *out,
                      size_t length, const AES_KEY *key,
                      unsigned char *ivec, int *num, const int enc)
    {
    CRYPTO_cfb128_8_encrypt(in,out,length,key,ivec,num,enc,(block128_f)AES_encry
    }
#endif /* ! codereview */
```

```
**********************************************************
    2879 Wed Aug 13 19:51:55 2014
new/usr/src/lib/openssl/libsunw_crypto/aes/aes_ctr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes_ctr.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/aes.h>
#include <openssl/modes.h>

void AES_ctr128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t length, const AES_KEY *key,
                        unsigned char ivec[AES_BLOCK_SIZE],
                        unsigned char ecount_buf[AES_BLOCK_SIZE],
                        unsigned int *num) {
        CRYPTO_ctr128_encrypt(in,out,length,key,ivec,ecount_buf,num,(block128_f)
}
```



```
#endif /* ! codereview */
```

```
**********************************************************
    2947 Wed Aug 13 19:51:55 2014
new/usr/src/lib/openssl/libsunw_crypto/aes/aes_ecb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes_ecb.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#ifndef AES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>

#include <openssl/aes.h>
#include "aes_locl.h"
```



```
void AES_ecb_encrypt(const unsigned char *in, unsigned char *out,
                     const AES_KEY *key, const int enc) {

        assert(in && out && key);
        assert((AES_ENCRYPT == enc)||(AES_DECRYPT == enc));

        if (AES_ENCRYPT == enc)
                AES_encrypt(in, out, key);
        else
                AES_decrypt(in, out, key);
}
#endif /* ! codereview */
```

```
**********************************************************
    9661 Wed Aug 13 19:51:56 2014
new/usr/src/lib/openssl/libsunw_crypto/aes/aes_ige.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes_ige.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include "cryptlib.h"

#include <openssl/aes.h>
#include "aes_locl.h"

#define N_WORDS (AES_BLOCK_SIZE / sizeof(unsigned long))
typedef struct {
        unsigned long data[N_WORDS];
} aes_block_t;
```



```
/* XXX: probably some better way to do this */
#if defined(__i386__) || defined(__x86_64__)
#define UNALIGNED_MEMOPS_ARE_FAST 1
#else
#define UNALIGNED_MEMOPS_ARE_FAST 0
#endif

#if UNALIGNED_MEMOPS_ARE_FAST
#define load_block(d, s)        (d) = *(const aes_block_t *)(s)
#define store_block(d, s)       *(aes_block_t *)(d) = (s)
#else
#define load_block(d, s)        memcpy((d).data, (s), AES_BLOCK_SIZE)
#define store_block(d, s)       memcpy((d), (s).data, AES_BLOCK_SIZE)
#endif

/* N.B. The IV for this mode is _twice_ the block size */

void AES_ige_encrypt(const unsigned char *in, unsigned char *out,
                                        size_t length, const AES_KEY *key,
                                        unsigned char *ivec, const int enc)
        {
        size_t n;
        size_t len = length;

        OPENSSL_assert(in && out && key && ivec);
        OPENSSL_assert((AES_ENCRYPT == enc)||(AES_DECRYPT == enc));
        OPENSSL_assert((length%AES_BLOCK_SIZE) == 0);

        len = length / AES_BLOCK_SIZE;

        if (AES_ENCRYPT == enc)
                {
                if (in != out &&
                    (UNALIGNED_MEMOPS_ARE_FAST || ((size_t)in|(size_t)out|(size_
                        {
                        aes_block_t *ivp = (aes_block_t *)ivec;
                        aes_block_t *iv2p = (aes_block_t *)(ivec + AES_BLOCK_SIZ

                        while (len)
                                {
                                aes_block_t *inp = (aes_block_t *)in;
                                aes_block_t *outp = (aes_block_t *)out;

                                for(n=0 ; n < N_WORDS; ++n)
                                        outp->data[n] = inp->data[n] ^ ivp->data
                                AES_encrypt((unsigned char *)outp->data, (unsign
                                for(n=0 ; n < N_WORDS; ++n)
                                        outp->data[n] ^= iv2p->data[n];
                                ivp = outp;
                                iv2p = inp;
                                --len;
                                in += AES_BLOCK_SIZE;
                                out += AES_BLOCK_SIZE;
                                }
                        memcpy(ivec, ivp->data, AES_BLOCK_SIZE);
                        memcpy(ivec + AES_BLOCK_SIZE, iv2p->data, AES_BLOCK_SIZE
                        }
                else
                        {
                        aes_block_t tmp, tmp2;
                        aes_block_t iv;
                        aes_block_t iv2;

                        load_block(iv, ivec);
                        load_block(iv2, ivec + AES_BLOCK_SIZE);
```

```
128                         while (len)
129                                 {
130                                 load_block(tmp, in);
131                                 for(n=0 ; n < N_WORDS; ++n)
132                                         tmp2.data[n] = tmp.data[n] ^ iv.data[n];
133                                 AES_encrypt((unsigned char *)tmp2.data, (unsigne
134                                 for(n=0 ; n < N_WORDS; ++n)
135                                         tmp2.data[n] ^= iv2.data[n];
136                                 store_block(out, tmp2);
137                                 iv = tmp2;
138                                 iv2 = tmp;
139                                 --len;
140                                 in += AES_BLOCK_SIZE;
141                                 out += AES_BLOCK_SIZE;
142                                 }
143                         memcpy(ivec, iv.data, AES_BLOCK_SIZE);
144                         memcpy(ivec + AES_BLOCK_SIZE, iv2.data, AES_BLOCK_SIZE);
145                         }
146                 }
147         else
148                 {
149                 if (in != out &&
150                     (UNALIGNED_MEMOPS_ARE_FAST || ((size_t)in|(size_t)out|(size_
151                         {
152                         aes_block_t *ivp = (aes_block_t *)ivec;
153                         aes_block_t *iv2p = (aes_block_t *)(ivec + AES_BLOCK_SIZ
154
155                         while (len)
156                                 {
157                                 aes_block_t tmp;
158                                 aes_block_t *inp = (aes_block_t *)in;
159                                 aes_block_t *outp = (aes_block_t *)out;
160
161                                 for(n=0 ; n < N_WORDS; ++n)
162                                         tmp.data[n] = inp->data[n] ^ iv2p->data[
163                                 AES_decrypt((unsigned char *)tmp.data, (unsigned
164                                 for(n=0 ; n < N_WORDS; ++n)
165                                         outp->data[n] ^= ivp->data[n];
166                                 ivp = inp;
167                                 iv2p = outp;
168                                 --len;
169                                 in += AES_BLOCK_SIZE;
170                                 out += AES_BLOCK_SIZE;
171                                 }
172                         memcpy(ivec, ivp->data, AES_BLOCK_SIZE);
173                         memcpy(ivec + AES_BLOCK_SIZE, iv2p->data, AES_BLOCK_SIZE
174                         }
175                 else
176                         {
177                         aes_block_t tmp, tmp2;
178                         aes_block_t iv;
179                         aes_block_t iv2;
180
181                         load_block(iv, ivec);
182                         load_block(iv2, ivec + AES_BLOCK_SIZE);
183
184                         while (len)
185                                 {
186                                 load_block(tmp, in);
187                                 tmp2 = tmp;
188                                 for(n=0 ; n < N_WORDS; ++n)
189                                         tmp.data[n] ^= iv2.data[n];
190                                 AES_decrypt((unsigned char *)tmp.data, (unsigned
191                                 for(n=0 ; n < N_WORDS; ++n)
192                                         tmp.data[n] ^= iv.data[n];
193                                 store_block(out, tmp);
```



```
194                                 iv = tmp2;
195                                 iv2 = tmp;
196                                 --len;
197                                 in += AES_BLOCK_SIZE;
198                                 out += AES_BLOCK_SIZE;
199                                 }
200                         memcpy(ivec, iv.data, AES_BLOCK_SIZE);
201                         memcpy(ivec + AES_BLOCK_SIZE, iv2.data, AES_BLOCK_SIZE);
202                         }
203                 }
204         }
205
206 /*
207  * Note that its effectively impossible to do biIGE in anything other
208  * than a single pass, so no provision is made for chaining.
209  */
210
211 /* N.B. The IV for this mode is _four times_ the block size */
212
213 void AES_bi_ige_encrypt(const unsigned char *in, unsigned char *out,
214                                                 size_t length, const AES_KEY *ke
215                                                 const AES_KEY *key2, const unsig
216                                                 const int enc)
217         {
218         size_t n;
219         size_t len = length;
220         unsigned char tmp[AES_BLOCK_SIZE];
221         unsigned char tmp2[AES_BLOCK_SIZE];
222         unsigned char tmp3[AES_BLOCK_SIZE];
223         unsigned char prev[AES_BLOCK_SIZE];
224         const unsigned char *iv;
225         const unsigned char *iv2;
226
227         OPENSSL_assert(in && out && key && ivec);
228         OPENSSL_assert((AES_ENCRYPT == enc)||(AES_DECRYPT == enc));
229         OPENSSL_assert((length%AES_BLOCK_SIZE) == 0);
230
231         if (AES_ENCRYPT == enc)
232                 {
233                 /* XXX: Do a separate case for when in != out (strictly should
234                    check for overlap, too) */
235
236                 /* First the forward pass */
237                 iv = ivec;
238                 iv2 = ivec + AES_BLOCK_SIZE;
239                 while (len >= AES_BLOCK_SIZE)
240                         {
241                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
242                                 out[n] = in[n] ^ iv[n];
243                         AES_encrypt(out, out, key);
244                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
245                                 out[n] ^= iv2[n];
246                         iv = out;
247                         memcpy(prev, in, AES_BLOCK_SIZE);
248                         iv2 = prev;
249                         len -= AES_BLOCK_SIZE;
250                         in += AES_BLOCK_SIZE;
251                         out += AES_BLOCK_SIZE;
252                         }
253
254                 /* And now backwards */
255                 iv = ivec + AES_BLOCK_SIZE*2;
256                 iv2 = ivec + AES_BLOCK_SIZE*3;
257                 len = length;
258                 while(len >= AES_BLOCK_SIZE)
259                         {
```

```
260                         out -= AES_BLOCK_SIZE;
261                         /* XXX: reduce copies by alternating between buffers */
262                         memcpy(tmp, out, AES_BLOCK_SIZE);
263                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
264                                 out[n] ^= iv[n];
265                         /*                          hexdump(stdout, "out ^ iv", out,
266                         AES_encrypt(out, out, key);
267                         /*                          hexdump(stdout,"enc", out, AES_B
268                         /*                          hexdump(stdout,"iv2", iv2, AES_B
269                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
270                                 out[n] ^= iv2[n];
271                         /*                          hexdump(stdout,"out", out, AES_B
272                         iv = out;
273                         memcpy(prev, tmp, AES_BLOCK_SIZE);
274                         iv2 = prev;
275                         len -= AES_BLOCK_SIZE;
276                         }
277                 }
278         else
279                 {
280                 /* First backwards */
281                 iv = ivec + AES_BLOCK_SIZE*2;
282                 iv2 = ivec + AES_BLOCK_SIZE*3;
283                 in += length;
284                 out += length;
285                 while (len >= AES_BLOCK_SIZE)
286                         {
287                         in -= AES_BLOCK_SIZE;
288                         out -= AES_BLOCK_SIZE;
289                         memcpy(tmp, in, AES_BLOCK_SIZE);
290                         memcpy(tmp2, in, AES_BLOCK_SIZE);
291                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
292                                 tmp[n] ^= iv2[n];
293                         AES_decrypt(tmp, out, key);
294                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
295                                 out[n] ^= iv[n];
296                         memcpy(tmp3, tmp2, AES_BLOCK_SIZE);
297                         iv = tmp3;
298                         iv2 = out;
299                         len -= AES_BLOCK_SIZE;
300                         }

302                 /* And now forwards */
303                 iv = ivec;
304                 iv2 = ivec + AES_BLOCK_SIZE;
305                 len = length;
306                 while (len >= AES_BLOCK_SIZE)
307                         {
308                         memcpy(tmp, out, AES_BLOCK_SIZE);
309                         memcpy(tmp2, out, AES_BLOCK_SIZE);
310                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
311                                 tmp[n] ^= iv2[n];
312                         AES_decrypt(tmp, out, key);
313                         for(n=0 ; n < AES_BLOCK_SIZE ; ++n)
314                                 out[n] ^= iv[n];
315                         memcpy(tmp3, tmp2, AES_BLOCK_SIZE);
316                         iv = tmp3;
317                         iv2 = out;
318                         len -= AES_BLOCK_SIZE;
319                         in += AES_BLOCK_SIZE;
320                         out += AES_BLOCK_SIZE;
321                         }
322                 }
323         }
324 #endif /* ! codereview */
```

```
**********************************************************
    3287 Wed Aug 13 19:51:56 2014
new/usr/src/lib/openssl/libsunw_crypto/aes/aes_misc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes_misc.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/opensslv.h>
#include <openssl/crypto.h>
#include <openssl/aes.h>
#include "aes_locl.h"

const char AES_version[]="AES" OPENSSL_VERSION_PTEXT;

const char *AES_options(void) {
#ifdef FULL_UNROLL
        return "aes(full)";
```



```
#else
        return "aes(partial)";
#endif
}

/* FIPS wrapper functions to block low level AES calls in FIPS mode */

int AES_set_encrypt_key(const unsigned char *userKey, const int bits,
                        AES_KEY *key)
        {
#ifdef OPENSSL_FIPS
        fips_cipher_abort(AES);
#endif
        return private_AES_set_encrypt_key(userKey, bits, key);
        }

int AES_set_decrypt_key(const unsigned char *userKey, const int bits,
                        AES_KEY *key)
        {
#ifdef OPENSSL_FIPS
        fips_cipher_abort(AES);
#endif
        return private_AES_set_decrypt_key(userKey, bits, key);
        }
#endif /* ! codereview */
```

```
**********************************************************
    2792 Wed Aug 13 19:51:56 2014
new/usr/src/lib/openssl/libsunw_crypto/aes/aes_ofb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes_ofb.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2002-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/aes.h>
#include <openssl/modes.h>

void AES_ofb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const AES_KEY *key,
        unsigned char *ivec, int *num)
{
        CRYPTO_ofb128_encrypt(in,out,length,key,ivec,num,(block128_f)AES_encrypt
}
#endif /* ! codereview */
```

```
**********************************************************
    7670 Wed Aug 13 19:51:56 2014
new/usr/src/lib/openssl/libsunw_crypto/aes/aes_wrap.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/aes/aes_wrap.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/aes.h>
#include <openssl/bio.h>

static const unsigned char default_iv[] = {
  0xA6, 0xA6, 0xA6, 0xA6, 0xA6, 0xA6, 0xA6, 0xA6,
};
```



```
int AES_wrap_key(AES_KEY *key, const unsigned char *iv,
                unsigned char *out,
                const unsigned char *in, unsigned int inlen)
        {
        unsigned char *A, B[16], *R;
        unsigned int i, j, t;
        if ((inlen & 0x7) || (inlen < 8))
                return -1;
        A = B;
        t = 1;
        memcpy(out + 8, in, inlen);
        if (!iv)
                iv = default_iv;

        memcpy(A, iv, 8);

        for (j = 0; j < 6; j++)
                {
                R = out + 8;
                for (i = 0; i < inlen; i += 8, t++, R += 8)
                        {
                        memcpy(B + 8, R, 8);
                        AES_encrypt(B, B, key);
                        A[7] ^= (unsigned char)(t & 0xff);
                        if (t > 0xff)
                                {
                                A[6] ^= (unsigned char)((t >> 8) & 0xff);
                                A[5] ^= (unsigned char)((t >> 16) & 0xff);
                                A[4] ^= (unsigned char)((t >> 24) & 0xff);
                                }
                        memcpy(R, B + 8, 8);
                        }
                }
        memcpy(out, A, 8);
        return inlen + 8;
        }

int AES_unwrap_key(AES_KEY *key, const unsigned char *iv,
                unsigned char *out,
                const unsigned char *in, unsigned int inlen)
        {
        unsigned char *A, B[16], *R;
        unsigned int i, j, t;
        inlen -= 8;
        if (inlen & 0x7)
                return -1;
        if (inlen < 8)
                return -1;
        A = B;
        t =  6 * (inlen >> 3);
        memcpy(A, in, 8);
        memcpy(out, in + 8, inlen);
        for (j = 0; j < 6; j++)
                {
                R = out + inlen - 8;
                for (i = 0; i < inlen; i += 8, t--, R -= 8)
                        {
                        A[7] ^= (unsigned char)(t & 0xff);
                        if (t > 0xff)
                                {
                                A[6] ^= (unsigned char)((t >> 8) & 0xff);
                                A[5] ^= (unsigned char)((t >> 16) & 0xff);
                                A[4] ^= (unsigned char)((t >> 24) & 0xff);
                                }
                        memcpy(B + 8, R, 8);
                        AES_decrypt(B, B, key);
```

```
128                         memcpy(R, B + 8, 8);
129                         }
130                 }
131         if (!iv)
132                 iv = default_iv;
133         if (memcmp(A, iv, 8))
134                 {
135                 OPENSSL_cleanse(out, inlen);
136                 return 0;
137                 }
138         return inlen;
139         }

141 #ifdef AES_WRAP_TEST

143 int AES_wrap_unwrap_test(const unsigned char *kek, int keybits,
144                          const unsigned char *iv,
145                          const unsigned char *eout,
146                          const unsigned char *key, int keylen)
147         {
148         unsigned char *otmp = NULL, *ptmp = NULL;
149         int r, ret = 0;
150         AES_KEY wctx;
151         otmp = OPENSSL_malloc(keylen + 8);
152         ptmp = OPENSSL_malloc(keylen);
153         if (!otmp || !ptmp)
154                 return 0;
155         if (AES_set_encrypt_key(kek, keybits, &wctx))
156                 goto err;
157         r = AES_wrap_key(&wctx, iv, otmp, key, keylen);
158         if (r <= 0)
159                 goto err;

161         if (eout && memcmp(eout, otmp, keylen))
162                 goto err;

164         if (AES_set_decrypt_key(kek, keybits, &wctx))
165                 goto err;
166         r = AES_unwrap_key(&wctx, iv, ptmp, otmp, r);

168         if (memcmp(key, ptmp, keylen))
169                 goto err;

171         ret = 1;

173         err:
174         if (otmp)
175                 OPENSSL_free(otmp);
176         if (ptmp)
177                 OPENSSL_free(ptmp);

179         return ret;

181         }



185 int main(int argc, char **argv)
186 {

188 static const unsigned char kek[] = {
189   0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
190   0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
191   0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
192   0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f
193 };
```



```
195 static const unsigned char key[] = {
196   0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
197   0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff,
198   0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
199   0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f
200 };

202 static const unsigned char e1[] = {
203   0x1f, 0xa6, 0x8b, 0x0a, 0x81, 0x12, 0xb4, 0x47,
204   0xae, 0xf3, 0x4b, 0xd8, 0xfb, 0x5a, 0x7b, 0x82,
205   0x9d, 0x3e, 0x86, 0x23, 0x71, 0xd2, 0xcf, 0xe5
206 };

208 static const unsigned char e2[] = {
209   0x96, 0x77, 0x8b, 0x25, 0xae, 0x6c, 0xa4, 0x35,
210   0xf9, 0x2b, 0x5b, 0x97, 0xc0, 0x50, 0xae, 0xd2,
211   0x46, 0x8a, 0xb8, 0xa1, 0x7a, 0xd8, 0x4e, 0x5d
212 };

214 static const unsigned char e3[] = {
215   0x64, 0xe8, 0xc3, 0xf9, 0xce, 0x0f, 0x5b, 0xa2,
216   0x63, 0xe9, 0x77, 0x79, 0x05, 0x81, 0x8a, 0x2a,
217   0x93, 0xc8, 0x19, 0x1e, 0x7d, 0x6e, 0x8a, 0xe7
218 };

220 static const unsigned char e4[] = {
221   0x03, 0x1d, 0x33, 0x26, 0x4e, 0x15, 0xd3, 0x32,
222   0x68, 0xf2, 0x4e, 0xc2, 0x60, 0x74, 0x3e, 0xdc,
223   0xe1, 0xc6, 0xc7, 0xdd, 0xee, 0x72, 0x5a, 0x93,
224   0x6b, 0xa8, 0x14, 0x91, 0x5c, 0x67, 0x62, 0xd2
225 };

227 static const unsigned char e5[] = {
228   0xa8, 0xf9, 0xbc, 0x16, 0x12, 0xc6, 0x8b, 0x3f,
229   0xf6, 0xe6, 0xf4, 0xfb, 0xe3, 0x0e, 0x71, 0xe4,
230   0x76, 0x9c, 0x8b, 0x80, 0xa3, 0x2c, 0xb8, 0x95,
231   0x8c, 0xd5, 0xd1, 0x7d, 0x6b, 0x25, 0x4d, 0xa1
232 };

234 static const unsigned char e6[] = {
235   0x28, 0xc9, 0xf4, 0x04, 0xc4, 0xb8, 0x10, 0xf4,
236   0xcb, 0xcc, 0xb3, 0x5c, 0xfb, 0x87, 0xf8, 0x26,
237   0x3f, 0x57, 0x86, 0xe2, 0xd8, 0x0e, 0xd3, 0x26,
238   0xcb, 0xc7, 0xf0, 0xe7, 0x1a, 0x99, 0xf4, 0x3b,
239   0xfb, 0x98, 0x8b, 0x9b, 0x7a, 0x02, 0xdd, 0x21
240 };

242         AES_KEY wctx, xctx;
243         int ret;
244         ret = AES_wrap_unwrap_test(kek, 128, NULL, e1, key, 16);
245         fprintf(stderr, "Key test result %d\n", ret);
246         ret = AES_wrap_unwrap_test(kek, 192, NULL, e2, key, 16);
247         fprintf(stderr, "Key test result %d\n", ret);
248         ret = AES_wrap_unwrap_test(kek, 256, NULL, e3, key, 16);
249         fprintf(stderr, "Key test result %d\n", ret);
250         ret = AES_wrap_unwrap_test(kek, 192, NULL, e4, key, 24);
251         fprintf(stderr, "Key test result %d\n", ret);
252         ret = AES_wrap_unwrap_test(kek, 256, NULL, e5, key, 24);
253         fprintf(stderr, "Key test result %d\n", ret);
254         ret = AES_wrap_unwrap_test(kek, 256, NULL, e6, key, 32);
255         fprintf(stderr, "Key test result %d\n", ret);
256 }


259 #endif
```

```
260 #endif /* ! codereview */
```

```
**********************************************************
    2175 Wed Aug 13 19:51:56 2014
new/usr/src/lib/openssl/libsunw_crypto/amd64/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2014 Alexander Pyhalov
# Use is subject to license terms.

.KEEP_STATE:

# aes/*.s
ASM_SOURCES =   x86_64cpuid.s \
                aes-x86_64.s \
                aesni-sha1-x86_64.s \
                aesni-x86_64.s \
                bsaes-x86_64.s \
                vpaes-x86_64.s \
                modexp512-x86_64.s \
                x86_64-gf2m.s \
                x86_64-mont.s \
                x86_64-mont5.s \
                cmll-x86_64.s \
                md5-x86_64.s \
                ghash-x86_64.s \
                rc4-md5-x86_64.s \
                rc4-x86_64.s \
                sha1-x86_64.s \
                sha256-x86_64.s \
                sha512-x86_64.s

OBJECTS +=      $(ASM_SOURCES:%.s=%.o)

OBJECTS64 =     bf_enc.o \
                x86_64-gcc.o \
                cmll_misc.o \
                des_enc.o \
                fcrypt_b.o

OBJECTS +=      $(OBJECTS64)

CLEANFILES +=   $(ASM_SOURCES)

include ../Makefile.com
include ../../../Makefile.lib.64
```



```
CPPFLAGS += -DL_ENDIAN
CPPFLAGS += -DOPENSSL_IA32_SSE2
CPPFLAGS += -DOPENSSL_BN_ASM_MONT5
CPPFLAGS += -DBSAES_ASM
CPPFLAGS += -DPK11_LIB_LOCATION=\"/usr/lib/64/libpkcs11.so.1\"

all:  $(ROOTLIBDIR64) $(LIBS) $(LIBLINKS)

$(LIBLINKS):    FRC
        $(RM) $@; $(SYMLINK) $(DYNLIB) $@

$(ROOTLIBDIR64):
        $(INS.dir)

# sha512-x86_64.pl generates both sha512-x86_64.s and sha256-x86_64.s
# sha256-x86_64.s is generated by default
sha256-x86_64.s:        ../pl/sha512-x86_64.pl
        $(PERL)  ../pl/sha512-x86_64.pl elf $@

sha512-x86_64.s:        ../pl/sha512-x86_64.pl
        $(PERL)  ../pl/sha512-x86_64.pl elf $@

install: all $(ROOTLIBS64) $(ROOTLINKS64)

FRC:
#endif /* ! codereview */
```

```
**********************************************************
    7272 Wed Aug 13 19:51:56 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_bitstr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_bitstr.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
int ASN1_BIT_STRING_set(ASN1_BIT_STRING *x, unsigned char *d, int len)
{ return M_ASN1_BIT_STRING_set(x, d, len); }

int i2c_ASN1_BIT_STRING(ASN1_BIT_STRING *a, unsigned char **pp)
        {
        int ret,j,bits,len;
        unsigned char *p,*d;

        if (a == NULL) return(0);

        len=a->length;

        if (len > 0)
                {
                if (a->flags & ASN1_STRING_FLAG_BITS_LEFT)
                        {
                        bits=(int)a->flags&0x07;
                        }
                else
                        {
                        for ( ; len > 0; len--)
                                {
                                if (a->data[len-1]) break;
                                }
                        j=a->data[len-1];
                        if      (j & 0x01) bits=0;
                        else if (j & 0x02) bits=1;
                        else if (j & 0x04) bits=2;
                        else if (j & 0x08) bits=3;
                        else if (j & 0x10) bits=4;
                        else if (j & 0x20) bits=5;
                        else if (j & 0x40) bits=6;
                        else if (j & 0x80) bits=7;
                        else bits=0; /* should not happen */
                        }
                }
        else
                bits=0;

        ret=1+len;
        if (pp == NULL) return(ret);

        p= *pp;

        *(p++)=(unsigned char)bits;
        d=a->data;
        memcpy(p,d,len);
        p+=len;
        if (len > 0) p[-1]&=(0xff<<bits);
        *pp=p;
        return(ret);
        }

ASN1_BIT_STRING *c2i_ASN1_BIT_STRING(ASN1_BIT_STRING **a,
        const unsigned char **pp, long len)
        {
        ASN1_BIT_STRING *ret=NULL;
        const unsigned char *p;
        unsigned char *s;
        int i;

        if (len < 1)
                {
                i=ASN1_R_STRING_TOO_SHORT;
                goto err;
```

```
128                 }

130         if ((a == NULL) || ((*a) == NULL))
131                 {
132                 if ((ret=M_ASN1_BIT_STRING_new()) == NULL) return(NULL);
133                 }
134         else
135                 ret=(*a);

137         p= *pp;
138         i= *(p++);
139         /* We do this to preserve the settings.  If we modify
140          * the settings, via the _set_bit function, we will recalculate
141          * on output */
142         ret->flags&= ~(ASN1_STRING_FLAG_BITS_LEFT|0x07); /* clear */
143         ret->flags|=(ASN1_STRING_FLAG_BITS_LEFT|(i&0x07)); /* set */

145         if (len-- > 1) /* using one because of the bits left byte */
146                 {
147                 s=(unsigned char *)OPENSSL_malloc((int)len);
148                 if (s == NULL)
149                         {
150                         i=ERR_R_MALLOC_FAILURE;
151                         goto err;
152                         }
153                 memcpy(s,p,(int)len);
154                 s[len-1]&=(0xff<<i);
155                 p+=len;
156                 }
157         else
158                 s=NULL;

160         ret->length=(int)len;
161         if (ret->data != NULL) OPENSSL_free(ret->data);
162         ret->data=s;
163         ret->type=V_ASN1_BIT_STRING;
164         if (a != NULL) (*a)=ret;
165         *pp=p;
166         return(ret);
167 err:
168         ASN1err(ASN1_F_C2I_ASN1_BIT_STRING,i);
169         if ((ret != NULL) && ((a == NULL) || (*a != ret)))
170                 M_ASN1_BIT_STRING_free(ret);
171         return(NULL);
172         }

174 /* These next 2 functions from Goetz Babin-Ebell <babinebell@trustcenter.de>
175  */
176 int ASN1_BIT_STRING_set_bit(ASN1_BIT_STRING *a, int n, int value)
177         {
178         int w,v,iv;
179         unsigned char *c;

181         w=n/8;
182         v=1<<(7-(n&0x07));
183         iv= ~v;
184         if (!value) v=0;

186         if (a == NULL)
187                 return 0;

189         a->flags&= ~(ASN1_STRING_FLAG_BITS_LEFT|0x07); /* clear, set on write */

191         if ((a->length < (w+1)) || (a->data == NULL))
192                 {
193                 if (!value) return(1); /* Don't need to set */
```



```
194                 if (a->data == NULL)
195                         c=(unsigned char *)OPENSSL_malloc(w+1);
196                 else
197                         c=(unsigned char *)OPENSSL_realloc_clean(a->data,
198                                                                 a->length,
199                                                                 w+1);
200                 if (c == NULL)
201                         {
202                         ASN1err(ASN1_F_ASN1_BIT_STRING_SET_BIT,ERR_R_MALLOC_FAIL
203                         return 0;
204                         }
205                 if (w+1-a->length > 0) memset(c+a->length, 0, w+1-a->length);
206                 a->data=c;
207                 a->length=w+1;
208         }
209         a->data[w]=((a->data[w])&iv)|v;
210         while ((a->length > 0) && (a->data[a->length-1] == 0))
211                 a->length--;
212         return(1);
213         }

215 int ASN1_BIT_STRING_get_bit(ASN1_BIT_STRING *a, int n)
216         {
217         int w,v;

219         w=n/8;
220         v=1<<(7-(n&0x07));
221         if ((a == NULL) || (a->length < (w+1)) || (a->data == NULL))
222                 return(0);
223         return((a->data[w]&v) != 0);
224         }

226 /*
227  * Checks if the given bit string contains only bits specified by
228  * the flags vector. Returns 0 if there is at least one bit set in 'a'
229  * which is not specified in 'flags', 1 otherwise.
230  * 'len' is the length of 'flags'.
231  */
232 int ASN1_BIT_STRING_check(ASN1_BIT_STRING *a,
233                           unsigned char *flags, int flags_len)
234         {
235         int i, ok;
236         /* Check if there is one bit set at all. */
237         if (!a || !a->data) return 1;

239         /* Check each byte of the internal representation of the bit string. */
240         ok = 1;
241         for (i = 0; i < a->length && ok; ++i)
242                 {
243                 unsigned char mask = i < flags_len ? ~flags[i] : 0xff;
244                 /* We are done if there is an unneeded bit set. */
245                 ok = (a->data[i] & mask) == 0;
246                 }
247         return ok;
248         }
249 #endif /* ! codereview */
```

```
**********************************************************
    4086 Wed Aug 13 19:51:57 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_bool.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_bool.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
int i2d_ASN1_BOOLEAN(int a, unsigned char **pp)
        {
        int r;
        unsigned char *p;

        r=ASN1_object_size(0,1,V_ASN1_BOOLEAN);
        if (pp == NULL) return(r);
        p= *pp;

        ASN1_put_object(&p,0,1,V_ASN1_BOOLEAN,V_ASN1_UNIVERSAL);
        *(p++)= (unsigned char)a;
        *pp=p;
        return(r);
        }

int d2i_ASN1_BOOLEAN(int *a, const unsigned char **pp, long length)
        {
        int ret= -1;
        const unsigned char *p;
        long len;
        int inf,tag,xclass;
        int i=0;

        p= *pp;
        inf=ASN1_get_object(&p,&len,&tag,&xclass,length);
        if (inf & 0x80)
                {
                i=ASN1_R_BAD_OBJECT_HEADER;
                goto err;
                }

        if (tag != V_ASN1_BOOLEAN)
                {
                i=ASN1_R_EXPECTING_A_BOOLEAN;
                goto err;
                }

        if (len != 1)
                {
                i=ASN1_R_BOOLEAN_IS_WRONG_LENGTH;
                goto err;
                }
        ret= (int)*(p++);
        if (a != NULL) (*a)=ret;
        *pp=p;
        return(ret);
err:
        ASN1err(ASN1_F_D2I_ASN1_BOOLEAN,i);
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    7907 Wed Aug 13 19:51:57 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_bytes.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_bytes.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
static int asn1_collate_primitive(ASN1_STRING *a, ASN1_const_CTX *c);
/* type is a 'bitmap' of acceptable string types.
 */
ASN1_STRING *d2i_ASN1_type_bytes(ASN1_STRING **a, const unsigned char **pp,
             long length, int type)
        {
        ASN1_STRING *ret=NULL;
        const unsigned char *p;
        unsigned char *s;
        long len;
        int inf,tag,xclass;
        int i=0;

        p= *pp;
        inf=ASN1_get_object(&p,&len,&tag,&xclass,length);
        if (inf & 0x80) goto err;

        if (tag >= 32)
                {
                i=ASN1_R_TAG_VALUE_TOO_HIGH;
                goto err;
                }
        if (!(ASN1_tag2bit(tag) & type))
                {
                i=ASN1_R_WRONG_TYPE;
                goto err;
                }

        /* If a bit-string, exit early */
        if (tag == V_ASN1_BIT_STRING)
                return(d2i_ASN1_BIT_STRING(a,pp,length));

        if ((a == NULL) || ((*a) == NULL))
                {
                if ((ret=ASN1_STRING_new()) == NULL) return(NULL);
                }
        else
                ret=(*a);

        if (len != 0)
                {
                s=(unsigned char *)OPENSSL_malloc((int)len+1);
                if (s == NULL)
                        {
                        i=ERR_R_MALLOC_FAILURE;
                        goto err;
                        }
                memcpy(s,p,(int)len);
                s[len]='\0';
                p+=len;
                }
        else
                s=NULL;

        if (ret->data != NULL) OPENSSL_free(ret->data);
        ret->length=(int)len;
        ret->data=s;
        ret->type=tag;
        if (a != NULL) (*a)=ret;
        *pp=p;
        return(ret);
err:
        ASN1err(ASN1_F_D2I_ASN1_TYPE_BYTES,i);
        if ((ret != NULL) && ((a == NULL) || (*a != ret)))
                ASN1_STRING_free(ret);
```

```
128         return(NULL);
129         }

131 int i2d_ASN1_bytes(ASN1_STRING *a, unsigned char **pp, int tag, int xclass)
132         {
133         int ret,r,constructed;
134         unsigned char *p;

136         if (a == NULL)  return(0);

138         if (tag == V_ASN1_BIT_STRING)
139                 return(i2d_ASN1_BIT_STRING(a,pp));

141         ret=a->length;
142         r=ASN1_object_size(0,ret,tag);
143         if (pp == NULL) return(r);
144         p= *pp;

146         if ((tag == V_ASN1_SEQUENCE) || (tag == V_ASN1_SET))
147                 constructed=1;
148         else
149                 constructed=0;
150         ASN1_put_object(&p,constructed,ret,tag,xclass);
151         memcpy(p,a->data,a->length);
152         p+=a->length;
153         *pp= p;
154         return(r);
155         }

157 ASN1_STRING *d2i_ASN1_bytes(ASN1_STRING **a, const unsigned char **pp,
158              long length, int Ptag, int Pclass)
159         {
160         ASN1_STRING *ret=NULL;
161         const unsigned char *p;
162         unsigned char *s;
163         long len;
164         int inf,tag,xclass;
165         int i=0;

167         if ((a == NULL) || ((*a) == NULL))
168                 {
169                 if ((ret=ASN1_STRING_new()) == NULL) return(NULL);
170                 }
171         else
172                 ret=(*a);

174         p= *pp;
175         inf=ASN1_get_object(&p,&len,&tag,&xclass,length);
176         if (inf & 0x80)
177                 {
178                 i=ASN1_R_BAD_OBJECT_HEADER;
179                 goto err;
180                 }

182         if (tag != Ptag)
183                 {
184                 i=ASN1_R_WRONG_TAG;
185                 goto err;
186                 }

188         if (inf & V_ASN1_CONSTRUCTED)
189                 {
190                 ASN1_const_CTX c;

192                 c.pp=pp;
193                 c.p=p;
```



```
194                 c.inf=inf;
195                 c.slen=len;
196                 c.tag=Ptag;
197                 c.xclass=Pclass;
198                 c.max=(length == 0)?0:(p+length);
199                 if (!asn1_collate_primitive(ret,&c))
200                         goto err;
201                 else
202                         {
203                         p=c.p;
204                         }
205                 }
206         else
207                 {
208                 if (len != 0)
209                         {
210                         if ((ret->length < len) || (ret->data == NULL))
211                                 {
212                                 if (ret->data != NULL) OPENSSL_free(ret->data);
213                                 s=(unsigned char *)OPENSSL_malloc((int)len + 1);
214                                 if (s == NULL)
215                                         {
216                                         i=ERR_R_MALLOC_FAILURE;
217                                         goto err;
218                                         }
219                                 }
220                         else
221                                 s=ret->data;
222                         memcpy(s,p,(int)len);
223                         s[len] = '\0';
224                         p+=len;
225                         }
226                 else
227                         {
228                         s=NULL;
229                         if (ret->data != NULL) OPENSSL_free(ret->data);
230                         }

232                 ret->length=(int)len;
233                 ret->data=s;
234                 ret->type=Ptag;
235                 }

237         if (a != NULL) (*a)=ret;
238         *pp=p;
239         return(ret);
240 err:
241         if ((ret != NULL) && ((a == NULL) || (*a != ret)))
242                 ASN1_STRING_free(ret);
243         ASN1err(ASN1_F_D2I_ASN1_BYTES,i);
244         return(NULL);
245         }


248 /* We are about to parse 0..n d2i_ASN1_bytes objects, we are to collapse
249  * them into the one structure that is then returned */
250 /* There have been a few bug fixes for this function from
251  * Paul Keogh <paul.keogh@sse.ie>, many thanks to him */
252 static int asn1_collate_primitive(ASN1_STRING *a, ASN1_const_CTX *c)
253         {
254         ASN1_STRING *os=NULL;
255         BUF_MEM b;
256         int num;

258         b.length=0;
259         b.max=0;
```

```
260         b.data=NULL;

262         if (a == NULL)
263                 {
264                 c->error=ERR_R_PASSED_NULL_PARAMETER;
265                 goto err;
266                 }

268         num=0;
269         for (;;)
270                 {
271                 if (c->inf & 1)
272                         {
273                         c->eos=ASN1_const_check_infinite_end(&c->p,
274                                 (long)(c->max-c->p));
275                         if (c->eos) break;
276                         }
277                 else
278                         {
279                         if (c->slen <= 0) break;
280                         }

282                 c->q=c->p;
283                 if (d2i_ASN1_bytes(&os,&c->p,c->max-c->p,c->tag,c->xclass)
284                         == NULL)
285                         {
286                         c->error=ERR_R_ASN1_LIB;
287                         goto err;
288                         }

290                 if (!BUF_MEM_grow_clean(&b,num+os->length))
291                         {
292                         c->error=ERR_R_BUF_LIB;
293                         goto err;
294                         }
295                 memcpy(&(b.data[num]),os->data,os->length);
296                 if (!(c->inf & 1))
297                         c->slen-=(c->p-c->q);
298                 num+=os->length;
299                 }

301         if (!asn1_const_Finish(c)) goto err;

303         a->length=num;
304         if (a->data != NULL) OPENSSL_free(a->data);
305         a->data=(unsigned char *)b.data;
306         if (os != NULL) ASN1_STRING_free(os);
307         return(1);
308 err:
309         ASN1err(ASN1_F_ASN1_COLLATE_PRIMITIVE,c->error);
310         if (os != NULL) ASN1_STRING_free(os);
311         if (b.data != NULL) OPENSSL_free(b.data);
312         return(0);
313         }
314 #endif /* ! codereview */
```

```
**********************************************************
    7617 Wed Aug 13 19:51:57 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_d2i_fp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_d2i_fp.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <limits.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include <openssl/asn1_mac.h>

static int asn1_d2i_read_bio(BIO *in, BUF_MEM **pb);

#ifndef NO_OLD_ASN1
#ifndef OPENSSL_NO_FP_API

void *ASN1_d2i_fp(void *(*xnew)(void), d2i_of_void *d2i, FILE *in, void **x)
        {
        BIO *b;
        void *ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                ASN1err(ASN1_F_ASN1_D2I_FP,ERR_R_BUF_LIB);
                return(NULL);
                }
        BIO_set_fp(b,in,BIO_NOCLOSE);
        ret=ASN1_d2i_bio(xnew,d2i,b,x);
        BIO_free(b);
        return(ret);
        }
#endif

void *ASN1_d2i_bio(void *(*xnew)(void), d2i_of_void *d2i, BIO *in, void **x)
        {
        BUF_MEM *b = NULL;
        const unsigned char *p;
        void *ret=NULL;
        int len;

        len = asn1_d2i_read_bio(in, &b);
        if(len < 0) goto err;

        p=(unsigned char *)b->data;
        ret=d2i(x,&p,len);
err:
        if (b != NULL) BUF_MEM_free(b);
        return(ret);
        }

#endif

void *ASN1_item_d2i_bio(const ASN1_ITEM *it, BIO *in, void *x)
        {
        BUF_MEM *b = NULL;
        const unsigned char *p;
        void *ret=NULL;
        int len;

        len = asn1_d2i_read_bio(in, &b);
        if(len < 0) goto err;

        p=(const unsigned char *)b->data;
        ret=ASN1_item_d2i(x,&p,len, it);
err:
        if (b != NULL) BUF_MEM_free(b);
        return(ret);
        }

#ifndef OPENSSL_NO_FP_API
void *ASN1_item_d2i_fp(const ASN1_ITEM *it, FILE *in, void *x)
        {
        BIO *b;
        char *ret;
```

```
129         if ((b=BIO_new(BIO_s_file())) == NULL)
130                 {
131                 ASN1err(ASN1_F_ASN1_ITEM_D2I_FP,ERR_R_BUF_LIB);
132                 return(NULL);
133                 }
134         BIO_set_fp(b,in,BIO_NOCLOSE);
135         ret=ASN1_item_d2i_bio(it,b,x);
136         BIO_free(b);
137         return(ret);
138         }
139 #endif

141 #define HEADER_SIZE   8
142 static int asn1_d2i_read_bio(BIO *in, BUF_MEM **pb)
143         {
144         BUF_MEM *b;
145         unsigned char *p;
146         int i;
147         ASN1_const_CTX c;
148         size_t want=HEADER_SIZE;
149         int eos=0;
150         size_t off=0;
151         size_t len=0;

153         b=BUF_MEM_new();
154         if (b == NULL)
155                 {
156                 ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ERR_R_MALLOC_FAILURE);
157                 return -1;
158                 }

160         ERR_clear_error();
161         for (;;)
162                 {
163                 if (want >= (len-off))
164                         {
165                         want-=(len-off);

167                         if (len + want < len || !BUF_MEM_grow_clean(b,len+want))
168                                 {
169                                 ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ERR_R_MALLOC_FA
170                                 goto err;
171                                 }
172                         i=BIO_read(in,&(b->data[len]),want);
173                         if ((i < 0) && ((len-off) == 0))
174                                 {
175                                 ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ASN1_R_NOT_ENOU
176                                 goto err;
177                                 }
178                         if (i > 0)
179                                 {
180                                 if (len+i < len)
181                                         {
182                                         ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ASN1_R_
183                                         goto err;
184                                         }
185                                 len+=i;
186                                 }
187                         }
188                 /* else data already loaded */

190                 p=(unsigned char *)&(b->data[off]);
191                 c.p=p;
192                 c.inf=ASN1_get_object(&(c.p),&(c.slen),&(c.tag),&(c.xclass),
193                         len-off);
```



```
194                 if (c.inf & 0x80)
195                         {
196                         unsigned long e;

198                         e=ERR_GET_REASON(ERR_peek_error());
199                         if (e != ASN1_R_TOO_LONG)
200                                 goto err;
201                         else
202                                 ERR_clear_error(); /* clear error */
203                         }
204                 i=c.p-p;/* header length */
205                 off+=i; /* end of data */

207                 if (c.inf & 1)
208                         {
209                         /* no data body so go round again */
210                         eos++;
211                         if (eos < 0)
212                                 {
213                                 ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ASN1_R_HEADER_T
214                                 goto err;
215                                 }
216                         want=HEADER_SIZE;
217                         }
218                 else if (eos && (c.slen == 0) && (c.tag == V_ASN1_EOC))
219                         {
220                         /* eos value, so go back and read another header */
221                         eos--;
222                         if (eos <= 0)
223                                 break;
224                         else
225                                 want=HEADER_SIZE;
226                         }
227                 else
228                         {
229                         /* suck in c.slen bytes of data */
230                         want=c.slen;
231                         if (want > (len-off))
232                                 {
233                                 want-=(len-off);
234                                 if (want > INT_MAX /* BIO_read takes an int leng
235                                         len+want < len)
236                                         {
237                                         ASN1err(ASN1_F_ASN1_D2I_READ_BIO
238                                         goto err;
239                                         }
240                                 if (!BUF_MEM_grow_clean(b,len+want))
241                                         {
242                                         ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ERR_R_M
243                                         goto err;
244                                         }
245                                 while (want > 0)
246                                         {
247                                         i=BIO_read(in,&(b->data[len]),want);
248                                         if (i <= 0)
249                                                 {
250                                                 ASN1err(ASN1_F_ASN1_D2I_READ_BIO
251                                                     ASN1_R_NOT_ENOUGH_DATA);
252                                                 goto err;
253                                                 }
254                                         /* This can't overflow because
255                                          * |len+want| didn't overflow. */
256                                         len+=i;
257                                         want-=i;
258                                         }
259                                 }
```

```
260                         if (off + c.slen < off)
261                                 {
262                                 ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ASN1_R_TOO_LONG
263                                 goto err;
264                                 }
265                         off+=c.slen;
266                         if (eos <= 0)
267                                 {
268                                 break;
269                                 }
270                         else
271                                 want=HEADER_SIZE;
272                         }
273                 }

275         if (off > INT_MAX)
276                 {
277                 ASN1err(ASN1_F_ASN1_D2I_READ_BIO,ASN1_R_TOO_LONG);
278                 goto err;
279                 }

281         *pb = b;
282         return off;
283 err:
284         if (b != NULL) BUF_MEM_free(b);
285         return -1;
286         }
287 #endif /* ! codereview */
```

```
**********************************************************
    4154 Wed Aug 13 19:51:57 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_digest.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_digest.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
```



```
#include "cryptlib.h"

#ifndef NO_SYS_TYPES_H
# include <sys/types.h>
#endif

#include <openssl/err.h>
#include <openssl/evp.h>
#include <openssl/buffer.h>
#include <openssl/x509.h>

#ifndef NO_ASN1_OLD

int ASN1_digest(i2d_of_void *i2d, const EVP_MD *type, char *data,
                unsigned char *md, unsigned int *len)
        {
        int i;
        unsigned char *str,*p;

        i=i2d(data,NULL);
        if ((str=(unsigned char *)OPENSSL_malloc(i)) == NULL)
                {
                ASN1err(ASN1_F_ASN1_DIGEST,ERR_R_MALLOC_FAILURE);
                return(0);
                }
        p=str;
        i2d(data,&p);

        if (!EVP_Digest(str, i, md, len, type, NULL))
                return 0;
        OPENSSL_free(str);
        return(1);
        }

#endif


int ASN1_item_digest(const ASN1_ITEM *it, const EVP_MD *type, void *asn,
                unsigned char *md, unsigned int *len)
        {
        int i;
        unsigned char *str = NULL;

        i=ASN1_item_i2d(asn,&str, it);
        if (!str) return(0);

        if (!EVP_Digest(str, i, md, len, type, NULL))
                return 0;
        OPENSSL_free(str);
        return(1);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4243 Wed Aug 13 19:51:57 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_dup.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_dup.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#ifndef NO_OLD_ASN1

void *ASN1_dup(i2d_of_void *i2d, d2i_of_void *d2i, void *x)
        {
        unsigned char *b,*p;
        const unsigned char *p2;
        int i;
        char *ret;

        if (x == NULL) return(NULL);

        i=i2d(x,NULL);
        b=OPENSSL_malloc(i+10);
        if (b == NULL)
                { ASN1err(ASN1_F_ASN1_DUP,ERR_R_MALLOC_FAILURE); return(NULL); }
        p= b;
        i=i2d(x,&p);
        p2= b;
        ret=d2i(NULL,&p2,i);
        OPENSSL_free(b);
        return(ret);
        }

#endif

/* ASN1_ITEM version of dup: this follows the model above except we don't need
 * to allocate the buffer. At some point this could be rewritten to directly dup
 * the underlying structure instead of doing and encode and decode.
 */

void *ASN1_item_dup(const ASN1_ITEM *it, void *x)
        {
        unsigned char *b = NULL;
        const unsigned char *p;
        long i;
        void *ret;

        if (x == NULL) return(NULL);

        i=ASN1_item_i2d(x,&b,it);
        if (b == NULL)
                { ASN1err(ASN1_F_ASN1_ITEM_DUP,ERR_R_MALLOC_FAILURE); return(NUL
        p= b;
        ret=ASN1_item_d2i(NULL,&p,i, it);
        OPENSSL_free(b);
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    5609 Wed Aug 13 19:51:57 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_enum.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_enum.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#include <openssl/bn.h>

/*
 * Code for ENUMERATED type: identical to INTEGER apart from a different tag.
 * for comments on encoding see a_int.c
 */

int ASN1_ENUMERATED_set(ASN1_ENUMERATED *a, long v)
        {
        int j,k;
        unsigned int i;
        unsigned char buf[sizeof(long)+1];
        long d;

        a->type=V_ASN1_ENUMERATED;
        if (a->length < (int)(sizeof(long)+1))
                {
                if (a->data != NULL)
                        OPENSSL_free(a->data);
                if ((a->data=(unsigned char *)OPENSSL_malloc(sizeof(long)+1)) !=
                        memset((char *)a->data,0,sizeof(long)+1);
                }
        if (a->data == NULL)
                {
                ASN1err(ASN1_F_ASN1_ENUMERATED_SET,ERR_R_MALLOC_FAILURE);
                return(0);
                }
        d=v;
        if (d < 0)
                {
                d= -d;
                a->type=V_ASN1_NEG_ENUMERATED;
                }

        for (i=0; i<sizeof(long); i++)
                {
                if (d == 0) break;
                buf[i]=(int)d&0xff;
                d>>=8;
                }
        j=0;
        for (k=i-1; k >=0; k--)
                a->data[j++]=buf[k];
        a->length=j;
        return(1);
        }

long ASN1_ENUMERATED_get(ASN1_ENUMERATED *a)
        {
        int neg=0,i;
        long r=0;

        if (a == NULL) return(0L);
        i=a->type;
        if (i == V_ASN1_NEG_ENUMERATED)
                neg=1;
        else if (i != V_ASN1_ENUMERATED)
                return -1;

        if (a->length > (int)sizeof(long))
                {
                /* hmm... a bit ugly */
                return(0xffffffffL);
                }
        if (a->data == NULL)
                return 0;
```

```
129         for (i=0; i<a->length; i++)
130                 {
131                 r<<=8;
132                 r|=(unsigned char)a->data[i];
133                 }
134         if (neg) r= -r;
135         return(r);
136         }

138 ASN1_ENUMERATED *BN_to_ASN1_ENUMERATED(BIGNUM *bn, ASN1_ENUMERATED *ai)
139         {
140         ASN1_ENUMERATED *ret;
141         int len,j;

143         if (ai == NULL)
144                 ret=M_ASN1_ENUMERATED_new();
145         else
146                 ret=ai;
147         if (ret == NULL)
148                 {
149                 ASN1err(ASN1_F_BN_TO_ASN1_ENUMERATED,ERR_R_NESTED_ASN1_ERROR);
150                 goto err;
151                 }
152         if(BN_is_negative(bn)) ret->type = V_ASN1_NEG_ENUMERATED;
153         else ret->type=V_ASN1_ENUMERATED;
154         j=BN_num_bits(bn);
155         len=((j == 0)?0:((j/8)+1));
156         if (ret->length < len+4)
157                 {
158                 unsigned char *new_data=OPENSSL_realloc(ret->data, len+4);
159                 if (!new_data)
160                         {
161                         ASN1err(ASN1_F_BN_TO_ASN1_ENUMERATED,ERR_R_MALLOC_FAILUR
162                         goto err;
163                         }
164                 ret->data=new_data;
165                 }

167         ret->length=BN_bn2bin(bn,ret->data);
168         return(ret);
169 err:
170         if (ret != ai) M_ASN1_ENUMERATED_free(ret);
171         return(NULL);
172         }

174 BIGNUM *ASN1_ENUMERATED_to_BN(ASN1_ENUMERATED *ai, BIGNUM *bn)
175         {
176         BIGNUM *ret;

178         if ((ret=BN_bin2bn(ai->data,ai->length,bn)) == NULL)
179                 ASN1err(ASN1_F_ASN1_ENUMERATED_TO_BN,ASN1_R_BN_LIB);
180         else if(ai->type == V_ASN1_NEG_ENUMERATED) BN_set_negative(ret,1);
181         return(ret);
182         }
183 #endif /* ! codereview */
```

```
**********************************************************
    7759 Wed Aug 13 19:51:58 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_gentm.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_gentm.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* GENERALIZEDTIME implementation, written by Steve Henson. Based on UTCTIME */

#include <stdio.h>
```



```
#include <time.h>
#include "cryptlib.h"
#include "o_time.h"
#include <openssl/asn1.h>

#if 0

int i2d_ASN1_GENERALIZEDTIME(ASN1_GENERALIZEDTIME *a, unsigned char **pp)
        {
#ifdef CHARSET_EBCDIC
        /* KLUDGE! We convert to ascii before writing DER */
        int len;
        char tmp[24];
        ASN1_STRING tmpstr = *(ASN1_STRING *)a;

        len = tmpstr.length;
        ebcdic2ascii(tmp, tmpstr.data, (len >= sizeof tmp) ? sizeof tmp : len);
        tmpstr.data = tmp;

        a = (ASN1_GENERALIZEDTIME *) &tmpstr;
#endif
        return(i2d_ASN1_bytes((ASN1_STRING *)a,pp,
                V_ASN1_GENERALIZEDTIME,V_ASN1_UNIVERSAL));
        }


ASN1_GENERALIZEDTIME *d2i_ASN1_GENERALIZEDTIME(ASN1_GENERALIZEDTIME **a,
             unsigned char **pp, long length)
        {
        ASN1_GENERALIZEDTIME *ret=NULL;

        ret=(ASN1_GENERALIZEDTIME *)d2i_ASN1_bytes((ASN1_STRING **)a,pp,length,
                V_ASN1_GENERALIZEDTIME,V_ASN1_UNIVERSAL);
        if (ret == NULL)
                {
                ASN1err(ASN1_F_D2I_ASN1_GENERALIZEDTIME,ERR_R_NESTED_ASN1_ERROR);
                return(NULL);
                }
#ifdef CHARSET_EBCDIC
        ascii2ebcdic(ret->data, ret->data, ret->length);
#endif
        if (!ASN1_GENERALIZEDTIME_check(ret))
                {
                ASN1err(ASN1_F_D2I_ASN1_GENERALIZEDTIME,ASN1_R_INVALID_TIME_FORM
                goto err;
                }

        return(ret);
err:
        if ((ret != NULL) && ((a == NULL) || (*a != ret)))
                M_ASN1_GENERALIZEDTIME_free(ret);
        return(NULL);
        }

#endif

int ASN1_GENERALIZEDTIME_check(ASN1_GENERALIZEDTIME *d)
        {
        static const int min[9]={ 0, 0, 1, 1, 0, 0, 0, 0, 0};
        static const int max[9]={99, 99,12,31,23,59,59,12,59};
        char *a;
        int n,i,l,o;

        if (d->type != V_ASN1_GENERALIZEDTIME) return(0);
        l=d->length;
        a=(char *)d->data;
```

```
128         o=0;
129         /* GENERALIZEDTIME is similar to UTCTIME except the year is
130          * represented as YYYY. This stuff treats everything as a two digit
131          * field so make first two fields 00 to 99
132          */
133         if (l < 13) goto err;
134         for (i=0; i<7; i++)
135                 {
136                 if ((i == 6) && ((a[o] == 'Z') ||
137                         (a[o] == '+') || (a[o] == '-')))
138                         { i++; break; }
139                 if ((a[o] < '0') || (a[o] > '9')) goto err;
140                 n= a[o]-'0';
141                 if (++o > l) goto err;
142
143                 if ((a[o] < '0') || (a[o] > '9')) goto err;
144                 n=(n*10)+ a[o]-'0';
145                 if (++o > l) goto err;
146
147                 if ((n < min[i]) || (n > max[i])) goto err;
148                 }
149         /* Optional fractional seconds: decimal point followed by one
150          * or more digits.
151          */
152         if (a[o] == '.')
153                 {
154                 if (++o > l) goto err;
155                 i = o;
156                 while ((a[o] >= '0') && (a[o] <= '9') && (o <= l))
157                         o++;
158                 /* Must have at least one digit after decimal point */
159                 if (i == o) goto err;
160                 }
161
162         if (a[o] == 'Z')
163                 o++;
164         else if ((a[o] == '+') || (a[o] == '-'))
165                 {
166                 o++;
167                 if (o+4 > l) goto err;
168                 for (i=7; i<9; i++)
169                         {
170                         if ((a[o] < '0') || (a[o] > '9')) goto err;
171                         n= a[o]-'0';
172                         o++;
173                         if ((a[o] < '0') || (a[o] > '9')) goto err;
174                         n=(n*10)+ a[o]-'0';
175                         if ((n < min[i]) || (n > max[i])) goto err;
176                         o++;
177                         }
178                 }
179         else
180                 {
181                 /* Missing time zone information. */
182                 goto err;
183                 }
184         return(o == l);
185 err:
186         return(0);
187         }
188
189 int ASN1_GENERALIZEDTIME_set_string(ASN1_GENERALIZEDTIME *s, const char *str)
190         {
191         ASN1_GENERALIZEDTIME t;
192
193         t.type=V_ASN1_GENERALIZEDTIME;
```



```
194         t.length=strlen(str);
195         t.data=(unsigned char *)str;
196         if (ASN1_GENERALIZEDTIME_check(&t))
197                 {
198                 if (s != NULL)
199                         {
200                         if (!ASN1_STRING_set((ASN1_STRING *)s,
201                                 (unsigned char *)str,t.length))
202                                 return 0;
203                         s->type=V_ASN1_GENERALIZEDTIME;
204                         }
205                 return(1);
206                 }
207         else
208                 return(0);
209         }
210
211 ASN1_GENERALIZEDTIME *ASN1_GENERALIZEDTIME_set(ASN1_GENERALIZEDTIME *s,
212              time_t t)
213         {
214                 return ASN1_GENERALIZEDTIME_adj(s, t, 0, 0);
215         }
216
217 ASN1_GENERALIZEDTIME *ASN1_GENERALIZEDTIME_adj(ASN1_GENERALIZEDTIME *s,
218              time_t t, int offset_day, long offset_sec)
219         {
220         char *p;
221         struct tm *ts;
222         struct tm data;
223         size_t len = 20;
224
225         if (s == NULL)
226                 s=M_ASN1_GENERALIZEDTIME_new();
227         if (s == NULL)
228                 return(NULL);
229
230         ts=OPENSSL_gmtime(&t, &data);
231         if (ts == NULL)
232                 return(NULL);
233
234         if (offset_day || offset_sec)
235                 {
236                 if (!OPENSSL_gmtime_adj(ts, offset_day, offset_sec))
237                         return NULL;
238                 }
239
240         p=(char *)s->data;
241         if ((p == NULL) || ((size_t)s->length < len))
242                 {
243                 p=OPENSSL_malloc(len);
244                 if (p == NULL)
245                         {
246                         ASN1err(ASN1_F_ASN1_GENERALIZEDTIME_ADJ,
247                                 ERR_R_MALLOC_FAILURE);
248                         return(NULL);
249                         }
250                 if (s->data != NULL)
251                         OPENSSL_free(s->data);
252                 s->data=(unsigned char *)p;
253                 }
254
255         BIO_snprintf(p,len,"%04d%02d%02d%02d%02d%02dZ",ts->tm_year + 1900,
256                      ts->tm_mon+1,ts->tm_mday,ts->tm_hour,ts->tm_min,ts->tm_sec)
257         s->length=strlen(p);
258         s->type=V_ASN1_GENERALIZEDTIME;
259 #ifdef CHARSET_EBCDIC_not
```

```
260         ebcdic2ascii(s->data, s->data, s->length);
261 #endif
262         return(s);
263         }
264 #endif /* ! codereview */
```

```
**********************************************************
    4962 Wed Aug 13 19:51:58 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_i2d_fp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_i2d_fp.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/asn1.h>

#ifndef NO_OLD_ASN1

#ifndef OPENSSL_NO_FP_API
int ASN1_i2d_fp(i2d_of_void *i2d, FILE *out, void *x)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                ASN1err(ASN1_F_ASN1_I2D_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,out,BIO_NOCLOSE);
        ret=ASN1_i2d_bio(i2d,b,x);
        BIO_free(b);
        return(ret);
        }
#endif

int ASN1_i2d_bio(i2d_of_void *i2d, BIO *out, unsigned char *x)
        {
        char *b;
        unsigned char *p;
        int i,j=0,n,ret=1;

        n=i2d(x,NULL);
        b=(char *)OPENSSL_malloc(n);
        if (b == NULL)
                {
                ASN1err(ASN1_F_ASN1_I2D_BIO,ERR_R_MALLOC_FAILURE);
                return(0);
                }

        p=(unsigned char *)b;
        i2d(x,&p);

        for (;;)
                {
                i=BIO_write(out,&(b[j]),n);
                if (i == n) break;
                if (i <= 0)
                        {
                        ret=0;
                        break;
                        }
                j+=i;
                n-=i;
                }
        OPENSSL_free(b);
        return(ret);
        }

#endif

#ifndef OPENSSL_NO_FP_API
int ASN1_item_i2d_fp(const ASN1_ITEM *it, FILE *out, void *x)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                ASN1err(ASN1_F_ASN1_ITEM_I2D_FP,ERR_R_BUF_LIB);
```

```
128                 return(0);
129                 }
130         BIO_set_fp(b,out,BIO_NOCLOSE);
131         ret=ASN1_item_i2d_bio(it,b,x);
132         BIO_free(b);
133         return(ret);
134         }
135 #endif

137 int ASN1_item_i2d_bio(const ASN1_ITEM *it, BIO *out, void *x)
138         {
139         unsigned char *b = NULL;
140         int i,j=0,n,ret=1;

142         n = ASN1_item_i2d(x, &b, it);
143         if (b == NULL)
144                 {
145                 ASN1err(ASN1_F_ASN1_ITEM_I2D_BIO,ERR_R_MALLOC_FAILURE);
146                 return(0);
147                 }

149         for (;;)
150                 {
151                 i=BIO_write(out,&(b[j]),n);
152                 if (i == n) break;
153                 if (i <= 0)
154                         {
155                         ret=0;
156                         break;
157                         }
158                 j+=i;
159                 n-=i;
160                 }
161         OPENSSL_free(b);
162         return(ret);
163         }
164 #endif /* ! codereview */
```

```
**********************************************************
   11888 Wed Aug 13 19:51:58 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_int.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_int.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#include <openssl/bn.h>

ASN1_INTEGER *ASN1_INTEGER_dup(const ASN1_INTEGER *x)
{ return M_ASN1_INTEGER_dup(x);}

int ASN1_INTEGER_cmp(const ASN1_INTEGER *x, const ASN1_INTEGER *y)
        {
        int neg, ret;
        /* Compare signs */
        neg = x->type & V_ASN1_NEG;
        if (neg != (y->type & V_ASN1_NEG))
                {
                if (neg)
                        return -1;
                else
                        return 1;
                }

        ret = ASN1_STRING_cmp(x, y);

        if (neg)
                return -ret;
        else
                return ret;
        }


/*
 * This converts an ASN1 INTEGER into its content encoding.
 * The internal representation is an ASN1_STRING whose data is a big endian
 * representation of the value, ignoring the sign. The sign is determined by
 * the type: V_ASN1_INTEGER for positive and V_ASN1_NEG_INTEGER for negative.
 *
 * Positive integers are no problem: they are almost the same as the DER
 * encoding, except if the first byte is >= 0x80 we need to add a zero pad.
 *
 * Negative integers are a bit trickier...
 * The DER representation of negative integers is in 2s complement form.
 * The internal form is converted by complementing each octet and finally
 * adding one to the result. This can be done less messily with a little trick.
 * If the internal form has trailing zeroes then they will become FF by the
 * complement and 0 by the add one (due to carry) so just copy as many trailing
 * zeros to the destination as there are in the source. The carry will add one
 * to the last none zero octet: so complement this octet and add one and finally
 * complement any left over until you get to the start of the string.
 *
 * Padding is a little trickier too. If the first bytes is > 0x80 then we pad
 * with 0xff. However if the first byte is 0x80 and one of the following bytes
 * is non-zero we pad with 0xff. The reason for this distinction is that 0x80
 * followed by optional zeros isn't padded.
 */

int i2c_ASN1_INTEGER(ASN1_INTEGER *a, unsigned char **pp)
        {
        int pad=0,ret,i,neg;
        unsigned char *p,*n,pb=0;

        if (a == NULL) return(0);
        neg=a->type & V_ASN1_NEG;
        if (a->length == 0)
                ret=1;
        else
                {
                ret=a->length;
                i=a->data[0];
                if (!neg && (i > 127)) {
```

```
128                         pad=1;
129                         pb=0;
130                 } else if(neg) {
131                         if(i>128) {
132                                 pad=1;
133                                 pb=0xFF;
134                         } else if(i == 128) {
135                         /*
136                          * Special case: if any other bytes non zero we pad:
137                          * otherwise we don't.
138                          */
139                                 for(i = 1; i < a->length; i++) if(a->data[i]) {
140                                                 pad=1;
141                                                 pb=0xFF;
142                                                 break;
143                                 }
144                         }
145                 }
146                 ret+=pad;
147                 }
148         if (pp == NULL) return(ret);
149         p= *pp;
150
151         if (pad) *(p++)=pb;
152         if (a->length == 0) *(p++)=0;
153         else if (!neg) memcpy(p,a->data,(unsigned int)a->length);
154         else {
155                 /* Begin at the end of the encoding */
156                 n=a->data + a->length - 1;
157                 p += a->length - 1;
158                 i = a->length;
159                 /* Copy zeros to destination as long as source is zero */
160                 while(!*n) {
161                         *(p--) = 0;
162                         n--;
163                         i--;
164                 }
165                 /* Complement and increment next octet */
166                 *(p--) = ((*(n--)) ^ 0xff) + 1;
167                 i--;
168                 /* Complement any octets left */
169                 for(;i > 0; i--) *(p--) = *(n--) ^ 0xff;
170         }
171
172         *pp+=ret;
173         return(ret);
174         }
175
176 /* Convert just ASN1 INTEGER content octets to ASN1_INTEGER structure */
177
178 ASN1_INTEGER *c2i_ASN1_INTEGER(ASN1_INTEGER **a, const unsigned char **pp,
179              long len)
180         {
181         ASN1_INTEGER *ret=NULL;
182         const unsigned char *p, *pend;
183         unsigned char *to,*s;
184         int i;
185
186         if ((a == NULL) || ((*a) == NULL))
187                 {
188                 if ((ret=M_ASN1_INTEGER_new()) == NULL) return(NULL);
189                 ret->type=V_ASN1_INTEGER;
190                 }
191         else
192                 ret=(*a);
```



```
193
194         p= *pp;
195         pend = p + len;
196
197         /* We must OPENSSL_malloc stuff, even for 0 bytes otherwise it
198          * signifies a missing NULL parameter. */
199         s=(unsigned char *)OPENSSL_malloc((int)len+1);
200         if (s == NULL)
201                 {
202                 i=ERR_R_MALLOC_FAILURE;
203                 goto err;
204                 }
205         to=s;
206         if(!len) {
207                 /* Strictly speaking this is an illegal INTEGER but we
208                  * tolerate it.
209                  */
210                 ret->type=V_ASN1_INTEGER;
211         } else if (*p & 0x80) /* a negative number */
212                 {
213                 ret->type=V_ASN1_NEG_INTEGER;
214                 if ((*p == 0xff) && (len != 1)) {
215                         p++;
216                         len--;
217                 }
218                 i = len;
219                 p += i - 1;
220                 to += i - 1;
221                 while((!*p) && i) {
222                         *(to--) = 0;
223                         i--;
224                         p--;
225                 }
226                 /* Special case: if all zeros then the number will be of
227                  * the form FF followed by n zero bytes: this corresponds to
228                  * 1 followed by n zero bytes. We've already written n zeros
229                  * so we just append an extra one and set the first byte to
230                  * a 1. This is treated separately because it is the only case
231                  * where the number of bytes is larger than len.
232                  */
233                 if(!i) {
234                         *s = 1;
235                         s[len] = 0;
236                         len++;
237                 } else {
238                         *(to--) = (*(p--) ^ 0xff) + 1;
239                         i--;
240                         for(;i > 0; i--) *(to--) = *(p--) ^ 0xff;
241                 }
242         } else {
243                 ret->type=V_ASN1_INTEGER;
244                 if ((*p == 0) && (len != 1))
245                         {
246                         p++;
247                         len--;
248                         }
249                 memcpy(s,p,(int)len);
250         }
251
252         if (ret->data != NULL) OPENSSL_free(ret->data);
253         ret->data=s;
254         ret->length=(int)len;
255         if (a != NULL) (*a)=ret;
256         *pp=pend;
257         return(ret);
258 err:
259         ASN1err(ASN1_F_C2I_ASN1_INTEGER,i);
```

```
260         if ((ret != NULL) && ((a == NULL) || (*a != ret)))
261                 M_ASN1_INTEGER_free(ret);
262         return(NULL);
263         }


266 /* This is a version of d2i_ASN1_INTEGER that ignores the sign bit of
267  * ASN1 integers: some broken software can encode a positive INTEGER
268  * with its MSB set as negative (it doesn't add a padding zero).
269  */

271 ASN1_INTEGER *d2i_ASN1_UINTEGER(ASN1_INTEGER **a, const unsigned char **pp,
272              long length)
273         {
274         ASN1_INTEGER *ret=NULL;
275         const unsigned char *p;
276         unsigned char *s;
277         long len;
278         int inf,tag,xclass;
279         int i;

281         if ((a == NULL) || ((*a) == NULL))
282                 {
283                 if ((ret=M_ASN1_INTEGER_new()) == NULL) return(NULL);
284                 ret->type=V_ASN1_INTEGER;
285                 }
286         else
287                 ret=(*a);

289         p= *pp;
290         inf=ASN1_get_object(&p,&len,&tag,&xclass,length);
291         if (inf & 0x80)
292                 {
293                 i=ASN1_R_BAD_OBJECT_HEADER;
294                 goto err;
295                 }

297         if (tag != V_ASN1_INTEGER)
298                 {
299                 i=ASN1_R_EXPECTING_AN_INTEGER;
300                 goto err;
301                 }

303         /* We must OPENSSL_malloc stuff, even for 0 bytes otherwise it
304          * signifies a missing NULL parameter. */
305         s=(unsigned char *)OPENSSL_malloc((int)len+1);
306         if (s == NULL)
307                 {
308                 i=ERR_R_MALLOC_FAILURE;
309                 goto err;
310                 }
311         ret->type=V_ASN1_INTEGER;
312         if(len) {
313                 if ((*p == 0) && (len != 1))
314                         {
315                         p++;
316                         len--;
317                         }
318                 memcpy(s,p,(int)len);
319                 p+=len;
320         }

322         if (ret->data != NULL) OPENSSL_free(ret->data);
323         ret->data=s;
324         ret->length=(int)len;
325         if (a != NULL) (*a)=ret;
```



```
326         *pp=p;
327         return(ret);
328 err:
329         ASN1err(ASN1_F_D2I_ASN1_UINTEGER,i);
330         if ((ret != NULL) && ((a == NULL) || (*a != ret)))
331                 M_ASN1_INTEGER_free(ret);
332         return(NULL);
333         }

335 int ASN1_INTEGER_set(ASN1_INTEGER *a, long v)
336         {
337         int j,k;
338         unsigned int i;
339         unsigned char buf[sizeof(long)+1];
340         long d;

342         a->type=V_ASN1_INTEGER;
343         if (a->length < (int)(sizeof(long)+1))
344                 {
345                 if (a->data != NULL)
346                         OPENSSL_free(a->data);
347                 if ((a->data=(unsigned char *)OPENSSL_malloc(sizeof(long)+1)) !=
348                         memset((char *)a->data,0,sizeof(long)+1);
349                 }
350         if (a->data == NULL)
351                 {
352                 ASN1err(ASN1_F_ASN1_INTEGER_SET,ERR_R_MALLOC_FAILURE);
353                 return(0);
354                 }
355         d=v;
356         if (d < 0)
357                 {
358                 d= -d;
359                 a->type=V_ASN1_NEG_INTEGER;
360                 }

362         for (i=0; i<sizeof(long); i++)
363                 {
364                 if (d == 0) break;
365                 buf[i]=(int)d&0xff;
366                 d>>=8;
367                 }
368         j=0;
369         for (k=i-1; k >=0; k--)
370                 a->data[j++]=buf[k];
371         a->length=j;
372         return(1);
373         }

375 long ASN1_INTEGER_get(const ASN1_INTEGER *a)
376         {
377         int neg=0,i;
378         long r=0;

380         if (a == NULL) return(0L);
381         i=a->type;
382         if (i == V_ASN1_NEG_INTEGER)
383                 neg=1;
384         else if (i != V_ASN1_INTEGER)
385                 return -1;

387         if (a->length > (int)sizeof(long))
388                 {
389                 /* hmm... a bit ugly, return all ones */
390                 return -1;
391                 }
```

```
392         if (a->data == NULL)
393                 return 0;

395         for (i=0; i<a->length; i++)
396                 {
397                 r<<=8;
398                 r|=(unsigned char)a->data[i];
399                 }
400         if (neg) r= -r;
401         return(r);
402         }

404 ASN1_INTEGER *BN_to_ASN1_INTEGER(const BIGNUM *bn, ASN1_INTEGER *ai)
405         {
406         ASN1_INTEGER *ret;
407         int len,j;

409         if (ai == NULL)
410                 ret=M_ASN1_INTEGER_new();
411         else
412                 ret=ai;
413         if (ret == NULL)
414                 {
415                 ASN1err(ASN1_F_BN_TO_ASN1_INTEGER,ERR_R_NESTED_ASN1_ERROR);
416                 goto err;
417                 }
418         if (BN_is_negative(bn))
419                 ret->type = V_ASN1_NEG_INTEGER;
420         else ret->type=V_ASN1_INTEGER;
421         j=BN_num_bits(bn);
422         len=((j == 0)?0:((j/8)+1));
423         if (ret->length < len+4)
424                 {
425                 unsigned char *new_data=OPENSSL_realloc(ret->data, len+4);
426                 if (!new_data)
427                         {
428                         ASN1err(ASN1_F_BN_TO_ASN1_INTEGER,ERR_R_MALLOC_FAILURE);
429                         goto err;
430                         }
431                 ret->data=new_data;
432                 }
433         ret->length=BN_bn2bin(bn,ret->data);
434         /* Correct zero case */
435         if(!ret->length)
436                 {
437                 ret->data[0] = 0;
438                 ret->length = 1;
439                 }
440         return(ret);
441 err:
442         if (ret != ai) M_ASN1_INTEGER_free(ret);
443         return(NULL);
444         }

446 BIGNUM *ASN1_INTEGER_to_BN(const ASN1_INTEGER *ai, BIGNUM *bn)
447         {
448         BIGNUM *ret;

450         if ((ret=BN_bin2bn(ai->data,ai->length,bn)) == NULL)
451                 ASN1err(ASN1_F_ASN1_INTEGER_TO_BN,ASN1_R_BN_LIB);
452         else if(ai->type == V_ASN1_NEG_INTEGER)
453                 BN_set_negative(ret, 1);
454         return(ret);
455         }

457 IMPLEMENT_STACK_OF(ASN1_INTEGER)
```



```
458 IMPLEMENT_ASN1_SET_OF(ASN1_INTEGER)
459 #endif /* ! codereview */
```

```
**********************************************************
   11422 Wed Aug 13 19:51:58 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_mbstr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* a_mbstr.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>

static int traverse_string(const unsigned char *p, int len, int inform,
                           int (*rfunc)(unsigned long value, void *in), void *arg);
static int in_utf8(unsigned long value, void *arg);
static int out_utf8(unsigned long value, void *arg);
static int type_str(unsigned long value, void *arg);
static int cpy_asc(unsigned long value, void *arg);
static int cpy_bmp(unsigned long value, void *arg);
static int cpy_univ(unsigned long value, void *arg);
static int cpy_utf8(unsigned long value, void *arg);
static int is_printable(unsigned long value);

/* These functions take a string in UTF8, ASCII or multibyte form and
 * a mask of permissible ASN1 string types. It then works out the minimal
 * type (using the order Printable < IA5 < T61 < BMP < Universal < UTF8)
 * and creates a string of the correct type with the supplied data.
 * Yes this is horrible: it has to be :-(
 * The 'ncopy' form checks minimum and maximum size limits too.
 */

int ASN1_mbstring_copy(ASN1_STRING **out, const unsigned char *in, int len,
                                        int inform, unsigned long mask)
{
        return ASN1_mbstring_ncopy(out, in, len, inform, mask, 0, 0);
}

int ASN1_mbstring_ncopy(ASN1_STRING **out, const unsigned char *in, int len,
                                        int inform, unsigned long mask,
                                        long minsize, long maxsize)
{
        int str_type;
        int ret;
        char free_out;
        int outform, outlen = 0;
        ASN1_STRING *dest;
        unsigned char *p;
        int nchar;
        char strbuf[32];
        int (*cpyfunc)(unsigned long,void *) = NULL;
        if(len == -1) len = strlen((const char *)in);
        if(!mask) mask = DIRSTRING_TYPE;

        /* First do a string check and work out the number of characters */
        switch(inform) {

                case MBSTRING_BMP:
                if(len & 1) {
                        ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY,
                                        ASN1_R_INVALID_BMPSTRING_LENGTH);
                        return -1;
                }
                nchar = len >> 1;
                break;

                case MBSTRING_UNIV:
                if(len & 3) {
                        ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY,
                                        ASN1_R_INVALID_UNIVERSALSTRING_LENGTH);
                        return -1;
                }
                nchar = len >> 2;
                break;

                case MBSTRING_UTF8:
                nchar = 0;
```

```
128                 /* This counts the characters and does utf8 syntax checking */
129                 ret = traverse_string(in, len, MBSTRING_UTF8, in_utf8, &nchar);
130                 if(ret < 0) {
131                         ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY,
132                                                 ASN1_R_INVALID_UTF8STRING);
133                         return -1;
134                 }
135                 break;
136
137                 case MBSTRING_ASC:
138                 nchar = len;
139                 break;
140
141                 default:
142                 ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY, ASN1_R_UNKNOWN_FORMAT);
143                 return -1;
144         }
145
146         if((minsize > 0) && (nchar < minsize)) {
147                 ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY, ASN1_R_STRING_TOO_SHORT);
148                 BIO_snprintf(strbuf, sizeof strbuf, "%ld", minsize);
149                 ERR_add_error_data(2, "minsize=", strbuf);
150                 return -1;
151         }
152
153         if((maxsize > 0) && (nchar > maxsize)) {
154                 ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY, ASN1_R_STRING_TOO_LONG);
155                 BIO_snprintf(strbuf, sizeof strbuf, "%ld", maxsize);
156                 ERR_add_error_data(2, "maxsize=", strbuf);
157                 return -1;
158         }
159
160         /* Now work out minimal type (if any) */
161         if(traverse_string(in, len, inform, type_str, &mask) < 0) {
162                 ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY, ASN1_R_ILLEGAL_CHARACTERS);
163                 return -1;
164         }
165
166
167         /* Now work out output format and string type */
168         outform = MBSTRING_ASC;
169         if(mask & B_ASN1_PRINTABLESTRING) str_type = V_ASN1_PRINTABLESTRING;
170         else if(mask & B_ASN1_IA5STRING) str_type = V_ASN1_IA5STRING;
171         else if(mask & B_ASN1_T61STRING) str_type = V_ASN1_T61STRING;
172         else if(mask & B_ASN1_BMPSTRING) {
173                 str_type = V_ASN1_BMPSTRING;
174                 outform = MBSTRING_BMP;
175         } else if(mask & B_ASN1_UNIVERSALSTRING) {
176                 str_type = V_ASN1_UNIVERSALSTRING;
177                 outform = MBSTRING_UNIV;
178         } else {
179                 str_type = V_ASN1_UTF8STRING;
180                 outform = MBSTRING_UTF8;
181         }
182         if(!out) return str_type;
183         if(*out) {
184                 free_out = 0;
185                 dest = *out;
186                 if(dest->data) {
187                         dest->length = 0;
188                         OPENSSL_free(dest->data);
189                         dest->data = NULL;
190                 }
191                 dest->type = str_type;
192         } else {
193                 free_out = 1;
```



```
194                 dest = ASN1_STRING_type_new(str_type);
195                 if(!dest) {
196                         ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY,
197                                                         ERR_R_MALLOC_FAILURE);
198                         return -1;
199                 }
200                 *out = dest;
201         }
202         /* If both the same type just copy across */
203         if(inform == outform) {
204                 if(!ASN1_STRING_set(dest, in, len)) {
205                         ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY,ERR_R_MALLOC_FAILURE)
206                         return -1;
207                 }
208                 return str_type;
209         }
210
211         /* Work out how much space the destination will need */
212         switch(outform) {
213                 case MBSTRING_ASC:
214                 outlen = nchar;
215                 cpyfunc = cpy_asc;
216                 break;
217
218                 case MBSTRING_BMP:
219                 outlen = nchar << 1;
220                 cpyfunc = cpy_bmp;
221                 break;
222
223                 case MBSTRING_UNIV:
224                 outlen = nchar << 2;
225                 cpyfunc = cpy_univ;
226                 break;
227
228                 case MBSTRING_UTF8:
229                 outlen = 0;
230                 traverse_string(in, len, inform, out_utf8, &outlen);
231                 cpyfunc = cpy_utf8;
232                 break;
233         }
234         if(!(p = OPENSSL_malloc(outlen + 1))) {
235                 if(free_out) ASN1_STRING_free(dest);
236                 ASN1err(ASN1_F_ASN1_MBSTRING_NCOPY,ERR_R_MALLOC_FAILURE);
237                 return -1;
238         }
239         dest->length = outlen;
240         dest->data = p;
241         p[outlen] = 0;
242         traverse_string(in, len, inform, cpyfunc, &p);
243         return str_type;
244 }
245
246 /* This function traverses a string and passes the value of each character
247  * to an optional function along with a void * argument.
248  */
249
250 static int traverse_string(const unsigned char *p, int len, int inform,
251                            int (*rfunc)(unsigned long value, void *in), void *arg)
252 {
253         unsigned long value;
254         int ret;
255         while(len) {
256                 if(inform == MBSTRING_ASC) {
257                         value = *p++;
258                         len--;
259                 } else if(inform == MBSTRING_BMP) {
```

```
260                         value = *p++ << 8;
261                         value |= *p++;
262                         len -= 2;
263                 } else if(inform == MBSTRING_UNIV) {
264                         value = ((unsigned long)*p++) << 24;
265                         value |= ((unsigned long)*p++) << 16;
266                         value |= *p++ << 8;
267                         value |= *p++;
268                         len -= 4;
269                 } else {
270                         ret = UTF8_getc(p, len, &value);
271                         if(ret < 0) return -1;
272                         len -= ret;
273                         p += ret;
274                 }
275                 if(rfunc) {
276                         ret = rfunc(value, arg);
277                         if(ret <= 0) return ret;
278                 }
279         }
280         return 1;
281 }

283 /* Various utility functions for traverse_string */

285 /* Just count number of characters */

287 static int in_utf8(unsigned long value, void *arg)
288 {
289         int *nchar;
290         nchar = arg;
291         (*nchar)++;
292         return 1;
293 }

295 /* Determine size of output as a UTF8 String */

297 static int out_utf8(unsigned long value, void *arg)
298 {
299         int *outlen;
300         outlen = arg;
301         *outlen += UTF8_putc(NULL, -1, value);
302         return 1;
303 }

305 /* Determine the "type" of a string: check each character against a
306  * supplied "mask".
307  */

309 static int type_str(unsigned long value, void *arg)
310 {
311         unsigned long types;
312         types = *((unsigned long *)arg);
313         if((types & B_ASN1_PRINTABLESTRING) && !is_printable(value))
314                                         types &= ~B_ASN1_PRINTABLESTRING;
315         if((types & B_ASN1_IA5STRING) && (value > 127))
316                                         types &= ~B_ASN1_IA5STRING;
317         if((types & B_ASN1_T61STRING) && (value > 0xff))
318                                         types &= ~B_ASN1_T61STRING;
319         if((types & B_ASN1_BMPSTRING) && (value > 0xffff))
320                                         types &= ~B_ASN1_BMPSTRING;
321         if(!types) return -1;
322         *((unsigned long *)arg) = types;
323         return 1;
324 }
```



```
326 /* Copy one byte per character ASCII like strings */

328 static int cpy_asc(unsigned long value, void *arg)
329 {
330         unsigned char **p, *q;
331         p = arg;
332         q = *p;
333         *q = (unsigned char) value;
334         (*p)++;
335         return 1;
336 }

338 /* Copy two byte per character BMPStrings */

340 static int cpy_bmp(unsigned long value, void *arg)
341 {
342         unsigned char **p, *q;
343         p = arg;
344         q = *p;
345         *q++ = (unsigned char) ((value >> 8) & 0xff);
346         *q = (unsigned char) (value & 0xff);
347         *p += 2;
348         return 1;
349 }

351 /* Copy four byte per character UniversalStrings */

353 static int cpy_univ(unsigned long value, void *arg)
354 {
355         unsigned char **p, *q;
356         p = arg;
357         q = *p;
358         *q++ = (unsigned char) ((value >> 24) & 0xff);
359         *q++ = (unsigned char) ((value >> 16) & 0xff);
360         *q++ = (unsigned char) ((value >> 8) & 0xff);
361         *q = (unsigned char) (value & 0xff);
362         *p += 4;
363         return 1;
364 }

366 /* Copy to a UTF8String */

368 static int cpy_utf8(unsigned long value, void *arg)
369 {
370         unsigned char **p;
371         int ret;
372         p = arg;
373         /* We already know there is enough room so pass 0xff as the length */
374         ret = UTF8_putc(*p, 0xff, value);
375         *p += ret;
376         return 1;
377 }

379 /* Return 1 if the character is permitted in a PrintableString */
380 static int is_printable(unsigned long value)
381 {
382         int ch;
383         if(value > 0x7f) return 0;
384         ch = (int) value;
385         /* Note: we can't use 'isalnum' because certain accented
386          * characters may count as alphanumeric in some environments.
387          */
388 #ifndef CHARSET_EBCDIC
389         if((ch >= 'a') && (ch <= 'z')) return 1;
390         if((ch >= 'A') && (ch <= 'Z')) return 1;
391         if((ch >= '0') && (ch <= '9')) return 1;
```

```
392         if ((ch == ' ') || strchr("'()+,-./:=?", ch)) return 1;
393 #else /*CHARSET_EBCDIC*/
394         if((ch >= os_toascii['a']) && (ch <= os_toascii['z'])) return 1;
395         if((ch >= os_toascii['A']) && (ch <= os_toascii['Z'])) return 1;
396         if((ch >= os_toascii['0']) && (ch <= os_toascii['9'])) return 1;
397         if ((ch == os_toascii[' ']) || strchr("'()+,-./:=?", os_toebcdic[ch])) r
398 #endif /*CHARSET_EBCDIC*/
399         return 0;
400 }
401 #endif /* ! codereview */
```

```
**********************************************************
   10388 Wed Aug 13 19:51:58 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_object.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_object.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <limits.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include <openssl/asn1.h>
#include <openssl/objects.h>
#include <openssl/bn.h>

int i2d_ASN1_OBJECT(ASN1_OBJECT *a, unsigned char **pp)
        {
        unsigned char *p;
        int objsize;

        if ((a == NULL) || (a->data == NULL)) return(0);

        objsize = ASN1_object_size(0,a->length,V_ASN1_OBJECT);
        if (pp == NULL) return objsize;

        p= *pp;
        ASN1_put_object(&p,0,a->length,V_ASN1_OBJECT,V_ASN1_UNIVERSAL);
        memcpy(p,a->data,a->length);
        p+=a->length;

        *pp=p;
        return(objsize);
        }

int a2d_ASN1_OBJECT(unsigned char *out, int olen, const char *buf, int num)
        {
        int i,first,len=0,c, use_bn;
        char ftmp[24], *tmp = ftmp;
        int tmpsize = sizeof ftmp;
        const char *p;
        unsigned long l;
        BIGNUM *bl = NULL;

        if (num == 0)
                return(0);
        else if (num == -1)
                num=strlen(buf);

        p=buf;
        c= *(p++);
        num--;
        if ((c >= '0') && (c <= '2'))
                {
                first= c-'0';
                }
        else
                {
                ASN1err(ASN1_F_A2D_ASN1_OBJECT,ASN1_R_FIRST_NUM_TOO_LARGE);
                goto err;
                }

        if (num <= 0)
                {
                ASN1err(ASN1_F_A2D_ASN1_OBJECT,ASN1_R_MISSING_SECOND_NUMBER);
                goto err;
                }
        c= *(p++);
        num--;
        for (;;)
                {
                if (num <= 0) break;
                if ((c != '.') && (c != ' '))
                        {
                        ASN1err(ASN1_F_A2D_ASN1_OBJECT,ASN1_R_INVALID_SEPARATOR)
                        goto err;
                        }
```

```
128                 l=0;
129                 use_bn = 0;
130                 for (;;)
131                         {
132                         if (num <= 0) break;
133                         num--;
134                         c= *(p++);
135                         if ((c == ' ') || (c == '.'))
136                                 break;
137                         if ((c < '0') || (c > '9'))
138                                 {
139                                 ASN1err(ASN1_F_A2D_ASN1_OBJECT,ASN1_R_INVALID_DI
140                                 goto err;
141                                 }
142                         if (!use_bn && l >= ((ULONG_MAX - 80) / 10L))
143                                 {
144                                 use_bn = 1;
145                                 if (!bl)
146                                         bl = BN_new();
147                                 if (!bl || !BN_set_word(bl, l))
148                                         goto err;
149                                 }
150                         if (use_bn)
151                                 {
152                                 if (!BN_mul_word(bl, 10L)
153                                         || !BN_add_word(bl, c-'0'))
154                                         goto err;
155                                 }
156                         else
157                                 l=l*10L+(long)(c-'0');
158                         }
159                 if (len == 0)
160                         {
161                         if ((first < 2) && (l >= 40))
162                                 {
163                                 ASN1err(ASN1_F_A2D_ASN1_OBJECT,ASN1_R_SECOND_NUM
164                                 goto err;
165                                 }
166                         if (use_bn)
167                                 {
168                                 if (!BN_add_word(bl, first * 40))
169                                         goto err;
170                                 }
171                         else
172                                 l+=(long)first*40;
173                         }
174                 i=0;
175                 if (use_bn)
176                         {
177                         int blsize;
178                         blsize = BN_num_bits(bl);
179                         blsize = (blsize + 6)/7;
180                         if (blsize > tmpsize)
181                                 {
182                                 if (tmp != ftmp)
183                                         OPENSSL_free(tmp);
184                                 tmpsize = blsize + 32;
185                                 tmp = OPENSSL_malloc(tmpsize);
186                                 if (!tmp)
187                                         goto err;
188                                 }
189                         while(blsize--)
190                                 tmp[i++] = (unsigned char)BN_div_word(bl, 0x80L)
191                         }
192                 else
193                         {
```



```
195                         for (;;)
196                                 {
197                                 tmp[i++]=(unsigned char)l&0x7f;
198                                 l>>=7L;
199                                 if (l == 0L) break;
200                                 }

202                         }
203                 if (out != NULL)
204                         {
205                         if (len+i > olen)
206                                 {
207                                 ASN1err(ASN1_F_A2D_ASN1_OBJECT,ASN1_R_BUFFER_TOO
208                                 goto err;
209                                 }
210                         while (--i > 0)
211                                 out[len++]=tmp[i]|0x80;
212                         out[len++]=tmp[0];
213                         }
214                 else
215                         len+=i;
216                 }
217         if (tmp != ftmp)
218                 OPENSSL_free(tmp);
219         if (bl)
220                 BN_free(bl);
221         return(len);
222 err:
223         if (tmp != ftmp)
224                 OPENSSL_free(tmp);
225         if (bl)
226                 BN_free(bl);
227         return(0);
228         }

230 int i2t_ASN1_OBJECT(char *buf, int buf_len, ASN1_OBJECT *a)
231 {
232         return OBJ_obj2txt(buf, buf_len, a, 0);
233 }

235 int i2a_ASN1_OBJECT(BIO *bp, ASN1_OBJECT *a)
236         {
237         char buf[80], *p = buf;
238         int i;

240         if ((a == NULL) || (a->data == NULL))
241                 return(BIO_write(bp,"NULL",4));
242         i=i2t_ASN1_OBJECT(buf,sizeof buf,a);
243         if (i > (int)(sizeof(buf) - 1))
244                 {
245                 p = OPENSSL_malloc(i + 1);
246                 if (!p)
247                         return -1;
248                 i2t_ASN1_OBJECT(p,i + 1,a);
249                 }
250         if (i <= 0)
251                 return BIO_write(bp, "<INVALID>", 9);
252         BIO_write(bp,p,i);
253         if (p != buf)
254                 OPENSSL_free(p);
255         return(i);
256         }

258 ASN1_OBJECT *d2i_ASN1_OBJECT(ASN1_OBJECT **a, const unsigned char **pp,
259              long length)
```

```
{
        const unsigned char *p;
        long len;
        int tag,xclass;
        int inf,i;
        ASN1_OBJECT *ret = NULL;
        p= *pp;
        inf=ASN1_get_object(&p,&len,&tag,&xclass,length);
        if (inf & 0x80)
                {
                i=ASN1_R_BAD_OBJECT_HEADER;
                goto err;
                }

        if (tag != V_ASN1_OBJECT)
                {
                i=ASN1_R_EXPECTING_AN_OBJECT;
                goto err;
                }
        ret = c2i_ASN1_OBJECT(a, &p, len);
        if(ret) *pp = p;
        return ret;
err:
        ASN1err(ASN1_F_D2I_ASN1_OBJECT,i);
        return(NULL);
}

ASN1_OBJECT *c2i_ASN1_OBJECT(ASN1_OBJECT **a, const unsigned char **pp,
             long len)
        {
        ASN1_OBJECT *ret=NULL;
        const unsigned char *p;
        unsigned char *data;
        int i, length;

        /* Sanity check OID encoding.
         * Need at least one content octet.
         * MSB must be clear in the last octet.
         * can't have leading 0x80 in subidentifiers, see: X.690 8.19.2
         */
        if (len <= 0 || len > INT_MAX || pp == NULL || (p = *pp) == NULL ||
            p[len - 1] & 0x80)
                {
                ASN1err(ASN1_F_C2I_ASN1_OBJECT,ASN1_R_INVALID_OBJECT_ENCODING);
                return NULL;
                }
        /* Now 0 < len <= INT_MAX, so the cast is safe. */
        length = (int)len;
        for (i = 0; i < length; i++, p++)
                {
                if (*p == 0x80 && (!i || !(p[-1] & 0x80)))
                        {
                        ASN1err(ASN1_F_C2I_ASN1_OBJECT,ASN1_R_INVALID_OBJECT_ENC
                        return NULL;
                        }
                }

        /* only the ASN1_OBJECTs from the 'table' will have values
         * for ->sn or ->ln */
        if ((a == NULL) || ((*a) == NULL) ||
                !((*a)->flags & ASN1_OBJECT_FLAG_DYNAMIC))
                {
                if ((ret=ASN1_OBJECT_new()) == NULL) return(NULL);
                }
        else    ret=(*a);
```



```
        p= *pp;
        /* detach data from object */
        data = (unsigned char *)ret->data;
        ret->data = NULL;
        /* once detached we can change it */
        if ((data == NULL) || (ret->length < length))
                {
                ret->length=0;
                if (data != NULL) OPENSSL_free(data);
                data=(unsigned char *)OPENSSL_malloc(length);
                if (data == NULL)
                        { i=ERR_R_MALLOC_FAILURE; goto err; }
                ret->flags|=ASN1_OBJECT_FLAG_DYNAMIC_DATA;
                }
        memcpy(data,p,length);
        /* reattach data to object, after which it remains const */
        ret->data  =data;
        ret->length=length;
        ret->sn=NULL;
        ret->ln=NULL;
        /* ret->flags=ASN1_OBJECT_FLAG_DYNAMIC; we know it is dynamic */
        p+=length;

        if (a != NULL) (*a)=ret;
        *pp=p;
        return(ret);
err:
        ASN1err(ASN1_F_C2I_ASN1_OBJECT,i);
        if ((ret != NULL) && ((a == NULL) || (*a != ret)))
                ASN1_OBJECT_free(ret);
        return(NULL);
        }

ASN1_OBJECT *ASN1_OBJECT_new(void)
        {
        ASN1_OBJECT *ret;

        ret=(ASN1_OBJECT *)OPENSSL_malloc(sizeof(ASN1_OBJECT));
        if (ret == NULL)
                {
                ASN1err(ASN1_F_ASN1_OBJECT_NEW,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }
        ret->length=0;
        ret->data=NULL;
        ret->nid=0;
        ret->sn=NULL;
        ret->ln=NULL;
        ret->flags=ASN1_OBJECT_FLAG_DYNAMIC;
        return(ret);
        }

void ASN1_OBJECT_free(ASN1_OBJECT *a)
        {
        if (a == NULL) return;
        if (a->flags & ASN1_OBJECT_FLAG_DYNAMIC_STRINGS)
                {
#ifndef CONST_STRICT /* disable purely for compile-time strict const checking. D
                if (a->sn != NULL) OPENSSL_free((void *)a->sn);
                if (a->ln != NULL) OPENSSL_free((void *)a->ln);
#endif
                a->sn=a->ln=NULL;
                }
        if (a->flags & ASN1_OBJECT_FLAG_DYNAMIC_DATA)
                {
                if (a->data != NULL) OPENSSL_free((void *)a->data);
```

```
392                 a->data=NULL;
393                 a->length=0;
394                 }
395         if (a->flags & ASN1_OBJECT_FLAG_DYNAMIC)
396                 OPENSSL_free(a);
397         }

399 ASN1_OBJECT *ASN1_OBJECT_create(int nid, unsigned char *data, int len,
400              const char *sn, const char *ln)
401         {
402         ASN1_OBJECT o;

404         o.sn=sn;
405         o.ln=ln;
406         o.data=data;
407         o.nid=nid;
408         o.length=len;
409         o.flags=ASN1_OBJECT_FLAG_DYNAMIC|ASN1_OBJECT_FLAG_DYNAMIC_STRINGS|
410                 ASN1_OBJECT_FLAG_DYNAMIC_DATA;
411         return(OBJ_dup(&o));
412         }

414 IMPLEMENT_STACK_OF(ASN1_OBJECT)
415 IMPLEMENT_ASN1_SET_OF(ASN1_OBJECT)
416 #endif /* ! codereview */
```

```
**********************************************************
    3622 Wed Aug 13 19:51:58 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_octet.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_octet.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
ASN1_OCTET_STRING *ASN1_OCTET_STRING_dup(const ASN1_OCTET_STRING *x)
{ return M_ASN1_OCTET_STRING_dup(x); }

int ASN1_OCTET_STRING_cmp(const ASN1_OCTET_STRING *a, const ASN1_OCTET_STRING *b
{ return M_ASN1_OCTET_STRING_cmp(a, b); }

int ASN1_OCTET_STRING_set(ASN1_OCTET_STRING *x, const unsigned char *d, int len)
{ return M_ASN1_OCTET_STRING_set(x, d, len); }
#endif /* ! codereview */
```

```
**********************************************************
    4592 Wed Aug 13 19:51:59 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_print.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_print.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
int ASN1_PRINTABLE_type(const unsigned char *s, int len)
        {
        int c;
        int ia5=0;
        int t61=0;

        if (len <= 0) len= -1;
        if (s == NULL) return(V_ASN1_PRINTABLESTRING);

        while ((*s) && (len-- != 0))
                {
                c= *(s++);
#ifndef CHARSET_EBCDIC
                if (!(  ((c >= 'a') && (c <= 'z')) ||
                        ((c >= 'A') && (c <= 'Z')) ||
                        (c == ' ') ||
                        ((c >= '0') && (c <= '9')) ||
                        (c == '') || (c == '\'') ||
                        (c == '(') || (c == ')') ||
                        (c == '+') || (c == ',') ||
                        (c == '-') || (c == '.') ||
                        (c == '/') || (c == ':') ||
                        (c == '=') || (c == '?')))
                        ia5=1;
                if (c&0x80)
                        t61=1;
#else
                if (!isalnum(c) && (c != ' ') &&
                    strchr("'()+,-./:=?", c) == NULL)
                        ia5=1;
                if (os_toascii[c] & 0x80)
                        t61=1;
#endif
                }
        if (t61) return(V_ASN1_T61STRING);
        if (ia5) return(V_ASN1_IA5STRING);
        return(V_ASN1_PRINTABLESTRING);
        }

int ASN1_UNIVERSALSTRING_to_string(ASN1_UNIVERSALSTRING *s)
        {
        int i;
        unsigned char *p;

        if (s->type != V_ASN1_UNIVERSALSTRING) return(0);
        if ((s->length%4) != 0) return(0);
        p=s->data;
        for (i=0; i<s->length; i+=4)
                {
                if ((p[0] != '\0') || (p[1] != '\0') || (p[2] != '\0'))
                        break;
                else
                        p+=4;
                }
        if (i < s->length) return(0);
        p=s->data;
        for (i=3; i<s->length; i+=4)
                {
                *(p++)=s->data[i];
                }
        *(p)='\0';
        s->length/=4;
        s->type=ASN1_PRINTABLE_type(s->data,s->length);
        return(1);
        }
```

```
128 #endif /* ! codereview */
```

```
**********************************************************
    7715 Wed Aug 13 19:51:59 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_set.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_set.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1_mac.h>
```



```
#ifndef NO_ASN1_OLD

typedef struct
    {
    unsigned char *pbData;
    int cbData;
    } MYBLOB;

/* SetBlobCmp
 * This function compares two elements of SET_OF block
 */
static int SetBlobCmp(const void *elem1, const void *elem2 )
    {
    const MYBLOB *b1 = (const MYBLOB *)elem1;
    const MYBLOB *b2 = (const MYBLOB *)elem2;
    int r;

    r = memcmp(b1->pbData, b2->pbData,
               b1->cbData < b2->cbData ? b1->cbData : b2->cbData);
    if(r != 0)
        return r;
    return b1->cbData-b2->cbData;
    }

/* int is_set:  if TRUE, then sort the contents (i.e. it isn't a SEQUENCE)    */
int i2d_ASN1_SET(STACK_OF(OPENSSL_BLOCK) *a, unsigned char **pp,
                 i2d_of_void *i2d, int ex_tag, int ex_class,
                 int is_set)
        {
        int ret=0,r;
        int i;
        unsigned char *p;
        unsigned char *pStart, *pTempMem;
        MYBLOB *rgSetBlob;
        int totSize;

        if (a == NULL) return(0);
        for (i=sk_OPENSSL_BLOCK_num(a)-1; i>=0; i--)
                ret+=i2d(sk_OPENSSL_BLOCK_value(a,i),NULL);
        r=ASN1_object_size(1,ret,ex_tag);
        if (pp == NULL) return(r);

        p= *pp;
        ASN1_put_object(&p,1,ret,ex_tag,ex_class);

/* Modified by gp@nsj.co.jp */
        /* And then again by Ben */
        /* And again by Steve */

        if(!is_set || (sk_OPENSSL_BLOCK_num(a) < 2))
                {
                for (i=0; i<sk_OPENSSL_BLOCK_num(a); i++)
                        i2d(sk_OPENSSL_BLOCK_value(a,i),&p);

                *pp=p;
                return(r);
                }

        pStart  = p; /* Catch the beg of Setblobs*/
                /* In this array we will store the SET blobs */
                rgSetBlob = OPENSSL_malloc(sk_OPENSSL_BLOCK_num(a) * sizeof(MYBL
                if (rgSetBlob == NULL)
                        {
                        ASN1err(ASN1_F_I2D_ASN1_SET,ERR_R_MALLOC_FAILURE);
                        return(0);
```

```
128                         }

130         for (i=0; i<sk_OPENSSL_BLOCK_num(a); i++)
131                 {
132                 rgSetBlob[i].pbData = p;  /* catch each set encode blob */
133                 i2d(sk_OPENSSL_BLOCK_value(a,i),&p);
134                 rgSetBlob[i].cbData = p - rgSetBlob[i].pbData; /* Length of this
135 SetBlob
136 */
137                 }
138         *pp=p;
139         totSize = p - pStart; /* This is the total size of all set blobs */

141  /* Now we have to sort the blobs. I am using a simple algo.
142     *Sort ptrs *Copy to temp-mem *Copy from temp-mem to user-mem*/
143         qsort( rgSetBlob, sk_OPENSSL_BLOCK_num(a), sizeof(MYBLOB), SetBlobCmp);
144                 if (!(pTempMem = OPENSSL_malloc(totSize)))
145                         {
146                         ASN1err(ASN1_F_I2D_ASN1_SET,ERR_R_MALLOC_FAILURE);
147                         return(0);
148                         }

150 /* Copy to temp mem */
151         p = pTempMem;
152         for(i=0; i<sk_OPENSSL_BLOCK_num(a); ++i)
153                 {
154                 memcpy(p, rgSetBlob[i].pbData, rgSetBlob[i].cbData);
155                 p += rgSetBlob[i].cbData;
156                 }

158 /* Copy back to user mem*/
159         memcpy(pStart, pTempMem, totSize);
160         OPENSSL_free(pTempMem);
161         OPENSSL_free(rgSetBlob);

163         return(r);
164         }

166 STACK_OF(OPENSSL_BLOCK) *d2i_ASN1_SET(STACK_OF(OPENSSL_BLOCK) **a,
167                                const unsigned char **pp,
168                                long length, d2i_of_void *d2i,
169                                void (*free_func)(OPENSSL_BLOCK), int ex_tag,
170                                int ex_class)
171         {
172         ASN1_const_CTX c;
173         STACK_OF(OPENSSL_BLOCK) *ret=NULL;

175         if ((a == NULL) || ((*a) == NULL))
176                 {
177                 if ((ret=sk_OPENSSL_BLOCK_new_null()) == NULL)
178                         {
179                         ASN1err(ASN1_F_D2I_ASN1_SET,ERR_R_MALLOC_FAILURE);
180                         goto err;
181                         }
182                 }
183         else
184                 ret=(*a);

186         c.p= *pp;
187         c.max=(length == 0)?0:(c.p+length);

189         c.inf=ASN1_get_object(&c.p,&c.slen,&c.tag,&c.xclass,c.max-c.p);
190         if (c.inf & 0x80) goto err;
191         if (ex_class != c.xclass)
192                 {
193                 ASN1err(ASN1_F_D2I_ASN1_SET,ASN1_R_BAD_CLASS);
```



```
194                 goto err;
195                 }
196         if (ex_tag != c.tag)
197                 {
198                 ASN1err(ASN1_F_D2I_ASN1_SET,ASN1_R_BAD_TAG);
199                 goto err;
200                 }
201         if ((c.slen+c.p) > c.max)
202                 {
203                 ASN1err(ASN1_F_D2I_ASN1_SET,ASN1_R_LENGTH_ERROR);
204                 goto err;
205                 }
206         /* check for infinite constructed - it can be as long
207          * as the amount of data passed to us */
208         if (c.inf == (V_ASN1_CONSTRUCTED+1))
209                 c.slen=length+ *pp-c.p;
210         c.max=c.p+c.slen;

212         while (c.p < c.max)
213                 {
214                 char *s;

216                 if (M_ASN1_D2I_end_sequence()) break;
217                 /* XXX: This was called with 4 arguments, incorrectly, it seems
218                    if ((s=func(NULL,&c.p,c.slen,c.max-c.p)) == NULL) */
219                 if ((s=d2i(NULL,&c.p,c.slen)) == NULL)
220                         {
221                         ASN1err(ASN1_F_D2I_ASN1_SET,ASN1_R_ERROR_PARSING_SET_ELE
222                         asn1_add_error(*pp,(int)(c.p- *pp));
223                         goto err;
224                         }
225                 if (!sk_OPENSSL_BLOCK_push(ret,s)) goto err;
226                 }
227         if (a != NULL) (*a)=ret;
228         *pp=c.p;
229         return(ret);
230 err:
231         if ((ret != NULL) && ((a == NULL) || (*a != ret)))
232                 {
233                 if (free_func != NULL)
234                         sk_OPENSSL_BLOCK_pop_free(ret,free_func);
235                 else
236                         sk_OPENSSL_BLOCK_free(ret);
237                 }
238         return(NULL);
239         }

241 #endif
242 #endif /* ! codereview */
```

```
**********************************************************
   11409 Wed Aug 13 19:51:59 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_sign.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_sign.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <time.h>

#include "cryptlib.h"

#ifndef NO_SYS_TYPES_H
# include <sys/types.h>
#endif

#include <openssl/bn.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/objects.h>
#include <openssl/buffer.h>
#include "asn1_locl.h"
```

```
#ifndef NO_ASN1_OLD

int ASN1_sign(i2d_of_void *i2d, X509_ALGOR *algor1, X509_ALGOR *algor2,
              ASN1_BIT_STRING *signature, char *data, EVP_PKEY *pkey,
              const EVP_MD *type)
        {
        EVP_MD_CTX ctx;
        unsigned char *p,*buf_in=NULL,*buf_out=NULL;
        int i,inl=0,outl=0,outll=0;
        X509_ALGOR *a;

        EVP_MD_CTX_init(&ctx);
        for (i=0; i<2; i++)
                {
                if (i == 0)
                        a=algor1;
                else
                        a=algor2;
                if (a == NULL) continue;
                if (type->pkey_type == NID_dsaWithSHA1)
                        {
                        /* special case: RFC 2459 tells us to omit 'parameters'
                         * with id-dsa-with-sha1 */
                        ASN1_TYPE_free(a->parameter);
                        a->parameter = NULL;
                        }
                else if ((a->parameter == NULL) ||
                        (a->parameter->type != V_ASN1_NULL))
                        {
                        ASN1_TYPE_free(a->parameter);
                        if ((a->parameter=ASN1_TYPE_new()) == NULL) goto err;
                        a->parameter->type=V_ASN1_NULL;
                        }
                ASN1_OBJECT_free(a->algorithm);
                a->algorithm=OBJ_nid2obj(type->pkey_type);
                if (a->algorithm == NULL)
                        {
                        ASN1err(ASN1_F_ASN1_SIGN,ASN1_R_UNKNOWN_OBJECT_TYPE);
                        goto err;
                        }
                if (a->algorithm->length == 0)
                        {
                        ASN1err(ASN1_F_ASN1_SIGN,ASN1_R_THE_ASN1_OBJECT_IDENTIFI
                        goto err;
                        }
                }
        inl=i2d(data,NULL);
        buf_in=(unsigned char *)OPENSSL_malloc((unsigned int)inl);
        outl=outl=EVP_PKEY_size(pkey);
        buf_out=(unsigned char *)OPENSSL_malloc((unsigned int)outl);
        if ((buf_in == NULL) || (buf_out == NULL))
                {
                outl=0;
                ASN1err(ASN1_F_ASN1_SIGN,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        p=buf_in;

        i2d(data,&p);
        if (!EVP_SignInit_ex(&ctx,type, NULL)
                || !EVP_SignUpdate(&ctx,(unsigned char *)buf_in,inl)
                || !EVP_SignFinal(&ctx,(unsigned char *)buf_out,
                        (unsigned int *)&outl,pkey))
                {
                outl=0;
                ASN1err(ASN1_F_ASN1_SIGN,ERR_R_EVP_LIB);
```



```
                goto err;
                }
        if (signature->data != NULL) OPENSSL_free(signature->data);
        signature->data=buf_out;
        buf_out=NULL;
        signature->length=outl;
        /* In the interests of compatibility, I'll make sure that
         * the bit string has a 'not-used bits' value of 0
         */
        signature->flags&= ~(ASN1_STRING_FLAG_BITS_LEFT|0x07);
        signature->flags|=ASN1_STRING_FLAG_BITS_LEFT;
err:
        EVP_MD_CTX_cleanup(&ctx);
        if (buf_in != NULL)
                { OPENSSL_cleanse((char *)buf_in,(unsigned int)inl); OPENSSL_fre
        if (buf_out != NULL)
                { OPENSSL_cleanse((char *)buf_out,outll); OPENSSL_free(buf_out);
        return(outl);
        }

#endif

int ASN1_item_sign(const ASN1_ITEM *it, X509_ALGOR *algor1, X509_ALGOR *algor2,
             ASN1_BIT_STRING *signature, void *asn, EVP_PKEY *pkey,
             const EVP_MD *type)
        {
        EVP_MD_CTX ctx;
        EVP_MD_CTX_init(&ctx);
        if (!EVP_DigestSignInit(&ctx, NULL, type, NULL, pkey))
                {
                EVP_MD_CTX_cleanup(&ctx);
                return 0;
                }
        return ASN1_item_sign_ctx(it, algor1, algor2, signature, asn, &ctx);
        }


int ASN1_item_sign_ctx(const ASN1_ITEM *it,
                X509_ALGOR *algor1, X509_ALGOR *algor2,
                ASN1_BIT_STRING *signature, void *asn, EVP_MD_CTX *ctx)
        {
        const EVP_MD *type;
        EVP_PKEY *pkey;
        unsigned char *buf_in=NULL,*buf_out=NULL;
        size_t inl=0,outl=0,outll=0;
        int signid, paramtype;
        int rv;

        type = EVP_MD_CTX_md(ctx);
        pkey = EVP_PKEY_CTX_get0_pkey(ctx->pctx);

        if (!type || !pkey)
                {
                ASN1err(ASN1_F_ASN1_ITEM_SIGN_CTX, ASN1_R_CONTEXT_NOT_INITIALISE
                return 0;
                }

        if (pkey->ameth->item_sign)
                {
                rv = pkey->ameth->item_sign(ctx, it, asn, algor1, algor2,
                                                    signature);
                if (rv == 1)
                        outl = signature->length;
                /* Return value meanings:
                 * <=0: error.
                 *   1: method does everything.
```

```
260                  *   2: carry on as normal.
261                  *   3: ASN1 method sets algorithm identifiers: just sign.
262                  */
263                 if (rv <= 0)
264                         ASN1err(ASN1_F_ASN1_ITEM_SIGN_CTX, ERR_R_EVP_LIB);
265                 if (rv <= 1)
266                         goto err;
267                 }
268         else
269                 rv = 2;

271         if (rv == 2)
272                 {
273                 if (type->flags & EVP_MD_FLAG_PKEY_METHOD_SIGNATURE)
274                         {
275                         if (!pkey->ameth ||
276                                 !OBJ_find_sigid_by_algs(&signid,
277                                                         EVP_MD_nid(type),
278                                                         pkey->ameth->pkey_id))
279                                 {
280                                 ASN1err(ASN1_F_ASN1_ITEM_SIGN_CTX,
281                                         ASN1_R_DIGEST_AND_KEY_TYPE_NOT_SUPPORTED
282                                 return 0;
283                                 }
284                         }
285                 else
286                         signid = type->pkey_type;

288                 if (pkey->ameth->pkey_flags & ASN1_PKEY_SIGPARAM_NULL)
289                         paramtype = V_ASN1_NULL;
290                 else
291                         paramtype = V_ASN1_UNDEF;

293                 if (algor1)
294                         X509_ALGOR_set0(algor1, OBJ_nid2obj(signid), paramtype,
295                 if (algor2)
296                         X509_ALGOR_set0(algor2, OBJ_nid2obj(signid), paramtype,

298                 }

300         inl=ASN1_item_i2d(asn,&buf_in, it);
301         outll=outl=EVP_PKEY_size(pkey);
302         buf_out=OPENSSL_malloc((unsigned int)outl);
303         if ((buf_in == NULL) || (buf_out == NULL))
304                 {
305                 outl=0;
306                 ASN1err(ASN1_F_ASN1_ITEM_SIGN_CTX,ERR_R_MALLOC_FAILURE);
307                 goto err;
308                 }

310         if (!EVP_DigestSignUpdate(ctx, buf_in, inl)
311                 || !EVP_DigestSignFinal(ctx, buf_out, &outl))
312                 {
313                 outl=0;
314                 ASN1err(ASN1_F_ASN1_ITEM_SIGN_CTX,ERR_R_EVP_LIB);
315                 goto err;
316                 }
317         if (signature->data != NULL) OPENSSL_free(signature->data);
318         signature->data=buf_out;
319         buf_out=NULL;
320         signature->length=outl;
321         /* In the interests of compatibility, I'll make sure that
322          * the bit string has a 'not-used bits' value of 0
323          */
324         signature->flags&= ~(ASN1_STRING_FLAG_BITS_LEFT|0x07);
325         signature->flags|=ASN1_STRING_FLAG_BITS_LEFT;
```



```
326 err:
327         EVP_MD_CTX_cleanup(ctx);
328         if (buf_in != NULL)
329                 { OPENSSL_cleanse((char *)buf_in,(unsigned int)inl); OPENSSL_fre
330         if (buf_out != NULL)
331                 { OPENSSL_cleanse((char *)buf_out,outll); OPENSSL_free(buf_out);
332         return(outl);
333         }
334 #endif /* ! codereview */
```

```
**********************************************************
   15830 Wed Aug 13 19:51:59 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_strex.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* a_strex.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <string.h>
#include "cryptlib.h"
```



```
#include <openssl/crypto.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>

#include "charmap.h"

/* ASN1_STRING_print_ex() and X509_NAME_print_ex().
 * Enhanced string and name printing routines handling
 * multibyte characters, RFC2253 and a host of other
 * options.
 */


#define CHARTYPE_BS_ESC         (ASN1_STRFLGS_ESC_2253 | CHARTYPE_FIRST_ESC_2253

#define ESC_FLAGS (ASN1_STRFLGS_ESC_2253 | \
                   ASN1_STRFLGS_ESC_QUOTE | \
                   ASN1_STRFLGS_ESC_CTRL | \
                   ASN1_STRFLGS_ESC_MSB)


/* Three IO functions for sending data to memory, a BIO and
 * and a FILE pointer.
 */
#if 0                           /* never used */
static int send_mem_chars(void *arg, const void *buf, int len)
{
        unsigned char **out = arg;
        if(!out) return 1;
        memcpy(*out, buf, len);
        *out += len;
        return 1;
}
#endif

static int send_bio_chars(void *arg, const void *buf, int len)
{
        if(!arg) return 1;
        if(BIO_write(arg, buf, len) != len) return 0;
        return 1;
}

static int send_fp_chars(void *arg, const void *buf, int len)
{
        if(!arg) return 1;
        if(fwrite(buf, 1, len, arg) != (unsigned int)len) return 0;
        return 1;
}

typedef int char_io(void *arg, const void *buf, int len);

/* This function handles display of
 * strings, one character at a time.
 * It is passed an unsigned long for each
 * character because it could come from 2 or even
 * 4 byte forms.
 */

static int do_esc_char(unsigned long c, unsigned char flags, char *do_quotes, ch
{
        unsigned char chflgs, chtmp;
        char tmphex[HEX_SIZE(long)+3];

        if(c > 0xffffffffL)
                return -1;
        if(c > 0xffff) {
```

```
128                 BIO_snprintf(tmphex, sizeof tmphex, "\\W%08lX", c);
129                 if(!io_ch(arg, tmphex, 10)) return -1;
130                 return 10;
131         }
132         if(c > 0xff) {
133                 BIO_snprintf(tmphex, sizeof tmphex, "\\U%04lX", c);
134                 if(!io_ch(arg, tmphex, 6)) return -1;
135                 return 6;
136         }
137         chtmp = (unsigned char)c;
138         if(chtmp > 0x7f) chflgs = flags & ASN1_STRFLGS_ESC_MSB;
139         else chflgs = char_type[chtmp] & flags;
140         if(chflgs & CHARTYPE_BS_ESC) {
141                 /* If we don't escape with quotes, signal we need quotes */
142                 if(chflgs & ASN1_STRFLGS_ESC_QUOTE) {
143                         if(do_quotes) *do_quotes = 1;
144                         if(!io_ch(arg, &chtmp, 1)) return -1;
145                         return 1;
146                 }
147                 if(!io_ch(arg, "\\", 1)) return -1;
148                 if(!io_ch(arg, &chtmp, 1)) return -1;
149                 return 2;
150         }
151         if(chflgs & (ASN1_STRFLGS_ESC_CTRL|ASN1_STRFLGS_ESC_MSB)) {
152                 BIO_snprintf(tmphex, 11, "\\%02X", chtmp);
153                 if(!io_ch(arg, tmphex, 3)) return -1;
154                 return 3;
155         }
156         /* If we get this far and do any escaping at all must escape
157          * the escape character itself: backslash.
158          */
159         if (chtmp == '\\' && flags & ESC_FLAGS) {
160                 if(!io_ch(arg, "\\\\", 2)) return -1;
161                 return 2;
162         }
163         if(!io_ch(arg, &chtmp, 1)) return -1;
164         return 1;
165 }

167 #define BUF_TYPE_WIDTH_MASK     0x7
168 #define BUF_TYPE_CONVUTF8       0x8

170 /* This function sends each character in a buffer to
171  * do_esc_char(). It interprets the content formats
172  * and converts to or from UTF8 as appropriate.
173  */

175 static int do_buf(unsigned char *buf, int buflen,
176                         int type, unsigned char flags, char *quotes, char_io *io
177 {
178         int i, outlen, len;
179         unsigned char orflags, *p, *q;
180         unsigned long c;
181         p = buf;
182         q = buf + buflen;
183         outlen = 0;
184         while(p != q) {
185                 if(p == buf && flags & ASN1_STRFLGS_ESC_2253) orflags = CHARTYPE
186                 else orflags = 0;
187                 switch(type & BUF_TYPE_WIDTH_MASK) {
188                         case 4:
189                         c = ((unsigned long)*p++) << 24;
190                         c |= ((unsigned long)*p++) << 16;
191                         c |= ((unsigned long)*p++) << 8;
192                         c |= *p++;
193                         break;
```



```
195                         case 2:
196                         c = ((unsigned long)*p++) << 8;
197                         c |= *p++;
198                         break;

200                         case 1:
201                         c = *p++;
202                         break;

204                         case 0:
205                         i = UTF8_getc(p, buflen, &c);
206                         if(i < 0) return -1;    /* Invalid UTF8String */
207                         p += i;
208                         break;
209                         default:
210                         return -1;      /* invalid width */
211                 }
212                 if (p == q && flags & ASN1_STRFLGS_ESC_2253) orflags = CHARTYPE_
213                 if(type & BUF_TYPE_CONVUTF8) {
214                         unsigned char utfbuf[6];
215                         int utflen;
216                         utflen = UTF8_putc(utfbuf, sizeof utfbuf, c);
217                         for(i = 0; i < utflen; i++) {
218                                 /* We don't need to worry about setting orflags
219                                  * because if utflen==1 its value will be correc
220                                  * otherwise each character will be > 0x7f and s
221                                  * character will never be escaped on first and
222                                  */
223                                 len = do_esc_char(utfbuf[i], (unsigned char)(fla
224                                 if(len < 0) return -1;
225                                 outlen += len;
226                         }
227                 } else {
228                         len = do_esc_char(c, (unsigned char)(flags | orflags), q
229                         if(len < 0) return -1;
230                         outlen += len;
231                 }
232         }
233         return outlen;
234 }

236 /* This function hex dumps a buffer of characters */

238 static int do_hex_dump(char_io *io_ch, void *arg, unsigned char *buf, int buflen
239 {
240         static const char hexdig[] = "0123456789ABCDEF";
241         unsigned char *p, *q;
242         char hextmp[2];
243         if(arg) {
244                 p = buf;
245                 q = buf + buflen;
246                 while(p != q) {
247                         hextmp[0] = hexdig[*p >> 4];
248                         hextmp[1] = hexdig[*p & 0xf];
249                         if(!io_ch(arg, hextmp, 2)) return -1;
250                         p++;
251                 }
252         }
253         return buflen << 1;
254 }

256 /* "dump" a string. This is done when the type is unknown,
257  * or the flags request it. We can either dump the content
258  * octets or the entire DER encoding. This uses the RFC2253
259  * #01234 format.
```

```
 */

static int do_dump(unsigned long lflags, char_io *io_ch, void *arg, ASN1_STRING
{
        /* Placing the ASN1_STRING in a temp ASN1_TYPE allows
         * the DER encoding to readily obtained
         */
        ASN1_TYPE t;
        unsigned char *der_buf, *p;
        int outlen, der_len;

        if(!io_ch(arg, "#", 1)) return -1;
        /* If we don't dump DER encoding just dump content octets */
        if(!(lflags & ASN1_STRFLGS_DUMP_DER)) {
                outlen = do_hex_dump(io_ch, arg, str->data, str->length);
                if(outlen < 0) return -1;
                return outlen + 1;
        }
        t.type = str->type;
        t.value.ptr = (char *)str;
        der_len = i2d_ASN1_TYPE(&t, NULL);
        der_buf = OPENSSL_malloc(der_len);
        if(!der_buf) return -1;
        p = der_buf;
        i2d_ASN1_TYPE(&t, &p);
        outlen = do_hex_dump(io_ch, arg, der_buf, der_len);
        OPENSSL_free(der_buf);
        if(outlen < 0) return -1;
        return outlen + 1;
}

/* Lookup table to convert tags to character widths,
 * 0 = UTF8 encoded, -1 is used for non string types
 * otherwise it is the number of bytes per character
 */

static const signed char tag2nbyte[] = {
        -1, -1, -1, -1, -1,     /* 0-4 */
        -1, -1, -1, -1, -1,     /* 5-9 */
        -1, -1, 0, -1,          /* 10-13 */
        -1, -1, -1, -1,         /* 15-17 */
        -1, 1, 1,               /* 18-20 */
        -1, 1, 1, 1,            /* 21-24 */
        -1, 1, -1,              /* 25-27 */
        4, -1, 2                /* 28-30 */
};

/* This is the main function, print out an
 * ASN1_STRING taking note of various escape
 * and display options. Returns number of
 * characters written or -1 if an error
 * occurred.
 */

static int do_print_ex(char_io *io_ch, void *arg, unsigned long lflags, ASN1_STR
{
        int outlen, len;
        int type;
        char quotes;
        unsigned char flags;
        quotes = 0;
        /* Keep a copy of escape flags */
        flags = (unsigned char)(lflags & ESC_FLAGS);

        type = str->type;
```



```
        outlen = 0;


        if(lflags & ASN1_STRFLGS_SHOW_TYPE) {
                const char *tagname;
                tagname = ASN1_tag2str(type);
                outlen += strlen(tagname);
                if(!io_ch(arg, tagname, outlen) || !io_ch(arg, ":", 1)) return -
                outlen++;
        }

        /* Decide what to do with type, either dump content or display it */

        /* Dump everything */
        if(lflags & ASN1_STRFLGS_DUMP_ALL) type = -1;
        /* Ignore the string type */
        else if(lflags & ASN1_STRFLGS_IGNORE_TYPE) type = 1;
        else {
                /* Else determine width based on type */
                if((type > 0) && (type < 31)) type = tag2nbyte[type];
                else type = -1;
                if((type == -1) && !(lflags & ASN1_STRFLGS_DUMP_UNKNOWN)) type =
        }

        if(type == -1) {
                len = do_dump(lflags, io_ch, arg, str);
                if(len < 0) return -1;
                outlen += len;
                return outlen;
        }

        if(lflags & ASN1_STRFLGS_UTF8_CONVERT) {
                /* Note: if string is UTF8 and we want
                 * to convert to UTF8 then we just interpret
                 * it as 1 byte per character to avoid converting
                 * twice.
                 */
                if(!type) type = 1;
                else type |= BUF_TYPE_CONVUTF8;
        }

        len = do_buf(str->data, str->length, type, flags, &quotes, io_ch, NULL);
        if(len < 0) return -1;
        outlen += len;
        if(quotes) outlen += 2;
        if(!arg) return outlen;
        if(quotes && !io_ch(arg, "\"", 1)) return -1;
        if(do_buf(str->data, str->length, type, flags, NULL, io_ch, arg) < 0)
                return -1;
        if(quotes && !io_ch(arg, "\"", 1)) return -1;
        return outlen;
}

/* Used for line indenting: print 'indent' spaces */

static int do_indent(char_io *io_ch, void *arg, int indent)
{
        int i;
        for(i = 0; i < indent; i++)
                        if(!io_ch(arg, " ", 1)) return 0;
        return 1;
}

#define FN_WIDTH_LN     25
#define FN_WIDTH_SN     10
```

```
392 static int do_name_ex(char_io *io_ch, void *arg, X509_NAME *n,
393                                 int indent, unsigned long flags)
394 {
395         int i, prev = -1, orflags, cnt;
396         int fn_opt, fn_nid;
397         ASN1_OBJECT *fn;
398         ASN1_STRING *val;
399         X509_NAME_ENTRY *ent;
400         char objtmp[80];
401         const char *objbuf;
402         int outlen, len;
403         char *sep_dn, *sep_mv, *sep_eq;
404         int sep_dn_len, sep_mv_len, sep_eq_len;
405         if(indent < 0) indent = 0;
406         outlen = indent;
407         if(!do_indent(io_ch, arg, indent)) return -1;
408         switch (flags & XN_FLAG_SEP_MASK)
409         {
410                 case XN_FLAG_SEP_MULTILINE:
411                 sep_dn = "\n";
412                 sep_dn_len = 1;
413                 sep_mv = " + ";
414                 sep_mv_len = 3;
415                 break;

417                 case XN_FLAG_SEP_COMMA_PLUS:
418                 sep_dn = ",";
419                 sep_dn_len = 1;
420                 sep_mv = "+";
421                 sep_mv_len = 1;
422                 indent = 0;
423                 break;

425                 case XN_FLAG_SEP_CPLUS_SPC:
426                 sep_dn = ", ";
427                 sep_dn_len = 2;
428                 sep_mv = " + ";
429                 sep_mv_len = 3;
430                 indent = 0;
431                 break;

433                 case XN_FLAG_SEP_SPLUS_SPC:
434                 sep_dn = "; ";
435                 sep_dn_len = 2;
436                 sep_mv = " + ";
437                 sep_mv_len = 3;
438                 indent = 0;
439                 break;

441                 default:
442                 return -1;
443         }

445         if(flags & XN_FLAG_SPC_EQ) {
446                 sep_eq = " = ";
447                 sep_eq_len = 3;
448         } else {
449                 sep_eq = "=";
450                 sep_eq_len = 1;
451         }

453         fn_opt = flags & XN_FLAG_FN_MASK;

455         cnt = X509_NAME_entry_count(n);
456         for(i = 0; i < cnt; i++) {
457                 if(flags & XN_FLAG_DN_REV)
```



```
458                         ent = X509_NAME_get_entry(n, cnt - i - 1);
459                 else ent = X509_NAME_get_entry(n, i);
460                 if(prev != -1) {
461                         if(prev == ent->set) {
462                                 if(!io_ch(arg, sep_mv, sep_mv_len)) return -1;
463                                 outlen += sep_mv_len;
464                         } else {
465                                 if(!io_ch(arg, sep_dn, sep_dn_len)) return -1;
466                                 outlen += sep_dn_len;
467                                 if(!do_indent(io_ch, arg, indent)) return -1;
468                                 outlen += indent;
469                         }
470                 }
471                 prev = ent->set;
472                 fn = X509_NAME_ENTRY_get_object(ent);
473                 val = X509_NAME_ENTRY_get_data(ent);
474                 fn_nid = OBJ_obj2nid(fn);
475                 if(fn_opt != XN_FLAG_FN_NONE) {
476                         int objlen, fld_len;
477                         if((fn_opt == XN_FLAG_FN_OID) || (fn_nid==NID_undef) ) {
478                                 OBJ_obj2txt(objtmp, sizeof objtmp, fn, 1);
479                                 fld_len = 0; /* XXX: what should this be? */
480                                 objbuf = objtmp;
481                         } else {
482                                 if(fn_opt == XN_FLAG_FN_SN) {
483                                         fld_len = FN_WIDTH_SN;
484                                         objbuf = OBJ_nid2sn(fn_nid);
485                                 } else if(fn_opt == XN_FLAG_FN_LN) {
486                                         fld_len = FN_WIDTH_LN;
487                                         objbuf = OBJ_nid2ln(fn_nid);
488                                 } else {
489                                         fld_len = 0; /* XXX: what should this be
490                                         objbuf = "";
491                                 }
492                         }
493                         objlen = strlen(objbuf);
494                         if(!io_ch(arg, objbuf, objlen)) return -1;
495                         if ((objlen < fld_len) && (flags & XN_FLAG_FN_ALIGN)) {
496                                 if (!do_indent(io_ch, arg, fld_len - objlen)) re
497                                 outlen += fld_len - objlen;
498                         }
499                         if(!io_ch(arg, sep_eq, sep_eq_len)) return -1;
500                         outlen += objlen + sep_eq_len;
501                 }
502                 /* If the field name is unknown then fix up the DER dump
503                  * flag. We might want to limit this further so it will
504                  * DER dump on anything other than a few 'standard' fields.
505                  */
506                 if((fn_nid == NID_undef) && (flags & XN_FLAG_DUMP_UNKNOWN_FIELDS
507                                         orflags = ASN1_STRFLGS_DUMP_ALL;
508                 else orflags = 0;

510                 len = do_print_ex(io_ch, arg, flags | orflags, val);
511                 if(len < 0) return -1;
512                 outlen += len;
513         }
514         return outlen;
515 }

517 /* Wrappers round the main functions */

519 int X509_NAME_print_ex(BIO *out, X509_NAME *nm, int indent, unsigned long flags)
520 {
521         if(flags == XN_FLAG_COMPAT)
522                 return X509_NAME_print(out, nm, indent);
523         return do_name_ex(send_bio_chars, out, nm, indent, flags);
```

```
524 }

526 #ifndef OPENSSL_NO_FP_API
527 int X509_NAME_print_ex_fp(FILE *fp, X509_NAME *nm, int indent, unsigned long fla
528 {
529         if(flags == XN_FLAG_COMPAT)
530                 {
531                 BIO *btmp;
532                 int ret;
533                 btmp = BIO_new_fp(fp, BIO_NOCLOSE);
534                 if(!btmp) return -1;
535                 ret = X509_NAME_print(btmp, nm, indent);
536                 BIO_free(btmp);
537                 return ret;
538                 }
539         return do_name_ex(send_fp_chars, fp, nm, indent, flags);
540 }
541 #endif

543 int ASN1_STRING_print_ex(BIO *out, ASN1_STRING *str, unsigned long flags)
544 {
545         return do_print_ex(send_bio_chars, out, flags, str);
546 }

548 #ifndef OPENSSL_NO_FP_API
549 int ASN1_STRING_print_ex_fp(FILE *fp, ASN1_STRING *str, unsigned long flags)
550 {
551         return do_print_ex(send_fp_chars, fp, flags, str);
552 }
553 #endif

555 /* Utility function: convert any string type to UTF8, returns number of bytes
556  * in output string or a negative error code
557  */

559 int ASN1_STRING_to_UTF8(unsigned char **out, ASN1_STRING *in)
560 {
561         ASN1_STRING stmp, *str = &stmp;
562         int mbflag, type, ret;
563         if(!in) return -1;
564         type = in->type;
565         if((type < 0) || (type > 30)) return -1;
566         mbflag = tag2nbyte[type];
567         if(mbflag == -1) return -1;
568         mbflag |= MBSTRING_FLAG;
569         stmp.data = NULL;
570         stmp.length = 0;
571         ret = ASN1_mbstring_copy(&str, in->data, in->length, mbflag, B_ASN1_UTF8
572         if(ret < 0) return ret;
573         *out = stmp.data;
574         return stmp.length;
575 }
576 #endif /* ! codereview */
```

```
**********************************************************
    9395 Wed Aug 13 19:51:59 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_strnid.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* a_strnid.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/objects.h>


static STACK_OF(ASN1_STRING_TABLE) *stable = NULL;
static void st_free(ASN1_STRING_TABLE *tbl);
static int sk_table_cmp(const ASN1_STRING_TABLE * const *a,
                        const ASN1_STRING_TABLE * const *b);


/* This is the global mask for the mbstring functions: this is use to
 * mask out certain types (such as BMPString and UTF8String) because
 * certain software (e.g. Netscape) has problems with them.
 */

static unsigned long global_mask = B_ASN1_UTF8STRING;

void ASN1_STRING_set_default_mask(unsigned long mask)
{
        global_mask = mask;
}

unsigned long ASN1_STRING_get_default_mask(void)
{
        return global_mask;
}

/* This function sets the default to various "flavours" of configuration.
 * based on an ASCII string. Currently this is:
 * MASK:XXXX : a numerical mask value.
 * nobmp : Don't use BMPStrings (just Printable, T61).
 * pkix : PKIX recommendation in RFC2459.
 * utf8only : only use UTF8Strings (RFC2459 recommendation for 2004).
 * default:   the default value, Printable, T61, BMP.
 */

int ASN1_STRING_set_default_mask_asc(const char *p)
{
        unsigned long mask;
        char *end;
        if(!strncmp(p, "MASK:", 5)) {
                if(!p[5]) return 0;
                mask = strtoul(p + 5, &end, 0);
                if(*end) return 0;
        } else if(!strcmp(p, "nombstr"))
                        mask = ~((unsigned long)(B_ASN1_BMPSTRING|B_ASN1_UTF8ST
        else if(!strcmp(p, "pkix"))
                        mask = ~((unsigned long)B_ASN1_T61STRING);
        else if(!strcmp(p, "utf8only")) mask = B_ASN1_UTF8STRING;
        else if(!strcmp(p, "default"))
            mask = 0xFFFFFFFFL;
        else return 0;
        ASN1_STRING_set_default_mask(mask);
        return 1;
}

/* The following function generates an ASN1_STRING based on limits in a table.
 * Frequently the types and length of an ASN1_STRING are restricted by a
 * corresponding OID. For example certificates and certificate requests.
 */

ASN1_STRING *ASN1_STRING_set_by_NID(ASN1_STRING **out, const unsigned char *in,
                                    int inlen, int inform, int nid)
{
        ASN1_STRING_TABLE *tbl;
        ASN1_STRING *str = NULL;
```

```
128         unsigned long mask;
129         int ret;
130         if(!out) out = &str;
131         tbl = ASN1_STRING_TABLE_get(nid);
132         if(tbl) {
133                 mask = tbl->mask;
134                 if(!(tbl->flags & STABLE_NO_MASK)) mask &= global_mask;
135                 ret = ASN1_mbstring_ncopy(out, in, inlen, inform, mask,
136                                           tbl->minsize, tbl->maxsize);
137         } else ret = ASN1_mbstring_copy(out, in, inlen, inform, DIRSTRING_TYPE &
138         if(ret <= 0) return NULL;
139         return *out;
140 }

142 /* Now the tables and helper functions for the string table:
143  */

145 /* size limits: this stuff is taken straight from RFC3280 */

147 #define ub_name                         32768
148 #define ub_common_name                  64
149 #define ub_locality_name                128
150 #define ub_state_name                   128
151 #define ub_organization_name            64
152 #define ub_organization_unit_name       64
153 #define ub_title                        64
154 #define ub_email_address                128
155 #define ub_serial_number                64


158 /* This table must be kept in NID order */

160 static const ASN1_STRING_TABLE tbl_standard[] = {
161 {NID_commonName,                1, ub_common_name, DIRSTRING_TYPE, 0},
162 {NID_countryName,               2, 2, B_ASN1_PRINTABLESTRING, STABLE_NO_MASK},
163 {NID_localityName,              1, ub_locality_name, DIRSTRING_TYPE, 0},
164 {NID_stateOrProvinceName,       1, ub_state_name, DIRSTRING_TYPE, 0},
165 {NID_organizationName,          1, ub_organization_name, DIRSTRING_TYPE, 0},
166 {NID_organizationalUnitName,    1, ub_organization_unit_name, DIRSTRING_TYPE, 0}
167 {NID_pkcs9_emailAddress,        1, ub_email_address, B_ASN1_IA5STRING, STABLE_NO
168 {NID_pkcs9_unstructuredName,    1, -1, PKCS9STRING_TYPE, 0},
169 {NID_pkcs9_challengePassword,   1, -1, PKCS9STRING_TYPE, 0},
170 {NID_pkcs9_unstructuredAddress, 1, -1, DIRSTRING_TYPE, 0},
171 {NID_givenName,                 1, ub_name, DIRSTRING_TYPE, 0},
172 {NID_surname,                   1, ub_name, DIRSTRING_TYPE, 0},
173 {NID_initials,                  1, ub_name, DIRSTRING_TYPE, 0},
174 {NID_serialNumber,              1, ub_serial_number, B_ASN1_PRINTABLESTRING, STA
175 {NID_friendlyName,              -1, -1, B_ASN1_BMPSTRING, STABLE_NO_MASK},
176 {NID_name,                      1, ub_name, DIRSTRING_TYPE, 0},
177 {NID_dnQualifier,               -1, -1, B_ASN1_PRINTABLESTRING, STABLE_NO_MASK},
178 {NID_domainComponent,           1, -1, B_ASN1_IA5STRING, STABLE_NO_MASK},
179 {NID_ms_csp_name,               -1, -1, B_ASN1_BMPSTRING, STABLE_NO_MASK}
180 };

182 static int sk_table_cmp(const ASN1_STRING_TABLE * const *a,
183                         const ASN1_STRING_TABLE * const *b)
184 {
185         return (*a)->nid - (*b)->nid;
186 }

188 DECLARE_OBJ_BSEARCH_CMP_FN(ASN1_STRING_TABLE, ASN1_STRING_TABLE, table);

190 static int table_cmp(const ASN1_STRING_TABLE *a, const ASN1_STRING_TABLE *b)
191 {
192         return a->nid - b->nid;
193 }
```



```
195 IMPLEMENT_OBJ_BSEARCH_CMP_FN(ASN1_STRING_TABLE, ASN1_STRING_TABLE, table);

197 ASN1_STRING_TABLE *ASN1_STRING_TABLE_get(int nid)
198 {
199         int idx;
200         ASN1_STRING_TABLE *ttmp;
201         ASN1_STRING_TABLE fnd;
202         fnd.nid = nid;
203         ttmp = OBJ_bsearch_table(&fnd, tbl_standard,
204                                  sizeof(tbl_standard)/sizeof(ASN1_STRING_TABLE));
205         if(ttmp) return ttmp;
206         if(!stable) return NULL;
207         idx = sk_ASN1_STRING_TABLE_find(stable, &fnd);
208         if(idx < 0) return NULL;
209         return sk_ASN1_STRING_TABLE_value(stable, idx);
210 }

212 int ASN1_STRING_TABLE_add(int nid,
213                  long minsize, long maxsize, unsigned long mask,
214                                   unsigned long flags)
215 {
216         ASN1_STRING_TABLE *tmp;
217         char new_nid = 0;
218         flags &= ~STABLE_FLAGS_MALLOC;
219         if(!stable) stable = sk_ASN1_STRING_TABLE_new(sk_table_cmp);
220         if(!stable) {
221                 ASN1err(ASN1_F_ASN1_STRING_TABLE_ADD, ERR_R_MALLOC_FAILURE);
222                 return 0;
223         }
224         if(!(tmp = ASN1_STRING_TABLE_get(nid))) {
225                 tmp = OPENSSL_malloc(sizeof(ASN1_STRING_TABLE));
226                 if(!tmp) {
227                         ASN1err(ASN1_F_ASN1_STRING_TABLE_ADD,
228                                                         ERR_R_MALLOC_FAILURE);
229                         return 0;
230                 }
231                 tmp->flags = flags | STABLE_FLAGS_MALLOC;
232                 tmp->nid = nid;
233                 new_nid = 1;
234         } else tmp->flags = (tmp->flags & STABLE_FLAGS_MALLOC) | flags;
235         if(minsize != -1) tmp->minsize = minsize;
236         if(maxsize != -1) tmp->maxsize = maxsize;
237         tmp->mask = mask;
238         if(new_nid) sk_ASN1_STRING_TABLE_push(stable, tmp);
239         return 1;
240 }

242 void ASN1_STRING_TABLE_cleanup(void)
243 {
244         STACK_OF(ASN1_STRING_TABLE) *tmp;
245         tmp = stable;
246         if(!tmp) return;
247         stable = NULL;
248         sk_ASN1_STRING_TABLE_pop_free(tmp, st_free);
249 }

251 static void st_free(ASN1_STRING_TABLE *tbl)
252 {
253         if(tbl->flags & STABLE_FLAGS_MALLOC) OPENSSL_free(tbl);
254 }


257 IMPLEMENT_STACK_OF(ASN1_STRING_TABLE)

259 #ifdef STRING_TABLE_TEST
```

```
main()
{
        ASN1_STRING_TABLE *tmp;
        int i, last_nid = -1;

        for (tmp = tbl_standard, i = 0;
                i < sizeof(tbl_standard)/sizeof(ASN1_STRING_TABLE); i++, tmp++)
                {
                        if (tmp->nid < last_nid)
                                {
                                last_nid = 0;
                                break;
                                }
                        last_nid = tmp->nid;
                }

        if (last_nid != 0)
                {
                printf("Table order OK\n");
                exit(0);
                }

        for (tmp = tbl_standard, i = 0;
                i < sizeof(tbl_standard)/sizeof(ASN1_STRING_TABLE); i++, tmp++)
                        printf("Index %d, NID %d, Name=%s\n", i, tmp->nid,
                                                        OBJ_nid2ln(tmp->nid));

}

#endif
#endif /* ! codereview */
```

```
**********************************************************
    5900 Wed Aug 13 19:51:59 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_time.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/asn1/a_time.c */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


/* This is an implementation of the ASN1 Time structure which is:
 *    Time ::= CHOICE {
 *      utcTime        UTCTime,
 *      generalTime    GeneralizedTime }
 * written by Steve Henson.
```



```
 */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
#include "o_time.h"
#include <openssl/asn1t.h>

IMPLEMENT_ASN1_MSTRING(ASN1_TIME, B_ASN1_TIME)

IMPLEMENT_ASN1_FUNCTIONS(ASN1_TIME)

#if 0
int i2d_ASN1_TIME(ASN1_TIME *a, unsigned char **pp)
        {
#ifdef CHARSET_EBCDIC
        /* KLUDGE! We convert to ascii before writing DER */
        char tmp[24];
        ASN1_STRING tmpstr;

        if(a->type == V_ASN1_UTCTIME || a->type == V_ASN1_GENERALIZEDTIME) {
            int len;

            tmpstr = *(ASN1_STRING *)a;
            len = tmpstr.length;
            ebcdic2ascii(tmp, tmpstr.data, (len >= sizeof tmp) ? sizeof tmp : le
            tmpstr.data = tmp;
            a = (ASN1_GENERALIZEDTIME *) &tmpstr;
        }
#endif
        if(a->type == V_ASN1_UTCTIME || a->type == V_ASN1_GENERALIZEDTIME)
                                    return(i2d_ASN1_bytes((ASN1_STRING *)a,pp,
                                        a->type ,V_ASN1_UNIVERSAL));
        ASN1err(ASN1_F_I2D_ASN1_TIME,ASN1_R_EXPECTING_A_TIME);
        return -1;
        }
#endif


ASN1_TIME *ASN1_TIME_set(ASN1_TIME *s, time_t t)
        {
        return ASN1_TIME_adj(s, t, 0, 0);
        }

ASN1_TIME *ASN1_TIME_adj(ASN1_TIME *s, time_t t,
                                    int offset_day, long offset_sec)
        {
        struct tm *ts;
        struct tm data;

        ts=OPENSSL_gmtime(&t,&data);
        if (ts == NULL)
                {
                ASN1err(ASN1_F_ASN1_TIME_ADJ, ASN1_R_ERROR_GETTING_TIME);
                return NULL;
                }
        if (offset_day || offset_sec)
                {
                if (!OPENSSL_gmtime_adj(ts, offset_day, offset_sec))
                        return NULL;
                }
        if((ts->tm_year >= 50) && (ts->tm_year < 150))
                        return ASN1_UTCTIME_adj(s, t, offset_day, offset_sec);
        return ASN1_GENERALIZEDTIME_adj(s, t, offset_day, offset_sec);
        }
```

```
int ASN1_TIME_check(ASN1_TIME *t)
        {
        if (t->type == V_ASN1_GENERALIZEDTIME)
                return ASN1_GENERALIZEDTIME_check(t);
        else if (t->type == V_ASN1_UTCTIME)
                return ASN1_UTCTIME_check(t);
        return 0;
        }

/* Convert an ASN1_TIME structure to GeneralizedTime */
ASN1_GENERALIZEDTIME *ASN1_TIME_to_generalizedtime(ASN1_TIME *t, ASN1_GENERALIZE
        {
        ASN1_GENERALIZEDTIME *ret;
        char *str;
        int newlen;

        if (!ASN1_TIME_check(t)) return NULL;

        if (!out || !*out)
                {
                if (!(ret = ASN1_GENERALIZEDTIME_new ()))
                        return NULL;
                if (out) *out = ret;
                }
        else ret = *out;

        /* If already GeneralizedTime just copy across */
        if (t->type == V_ASN1_GENERALIZEDTIME)
                {
                if(!ASN1_STRING_set(ret, t->data, t->length))
                        return NULL;
                return ret;
                }

        /* grow the string */
        if (!ASN1_STRING_set(ret, NULL, t->length + 2))
                return NULL;
        /* ASN1_STRING_set() allocated 'len + 1' bytes. */
        newlen = t->length + 2 + 1;
        str = (char *)ret->data;
        /* Work out the century and prepend */
        if (t->data[0] >= '5') BUF_strlcpy(str, "19", newlen);
        else BUF_strlcpy(str, "20", newlen);

        BUF_strlcat(str, (char *)t->data, newlen);

        return ret;
        }

int ASN1_TIME_set_string(ASN1_TIME *s, const char *str)
        {
        ASN1_TIME t;

        t.length = strlen(str);
        t.data = (unsigned char *)str;
        t.flags = 0;

        t.type = V_ASN1_UTCTIME;

        if (!ASN1_TIME_check(&t))
                {
                t.type = V_ASN1_GENERALIZEDTIME;
                if (!ASN1_TIME_check(&t))
                        return 0;
                }
```



```
        if (s && !ASN1_STRING_copy((ASN1_STRING *)s, (ASN1_STRING *)&t))
                        return 0;

        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    5279 Wed Aug 13 19:52:00 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_type.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_type.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/objects.h>

int ASN1_TYPE_get(ASN1_TYPE *a)
        {
        if ((a->value.ptr != NULL) || (a->type == V_ASN1_NULL))
                return(a->type);
        else
                return(0);
        }

void ASN1_TYPE_set(ASN1_TYPE *a, int type, void *value)
        {
        if (a->value.ptr != NULL)
                {
                ASN1_TYPE **tmp_a = &a;
                ASN1_primitive_free((ASN1_VALUE **)tmp_a, NULL);
                }
        a->type=type;
        if (type == V_ASN1_BOOLEAN)
                a->value.boolean = value ? 0xff : 0;
        else
                a->value.ptr=value;
        }

int ASN1_TYPE_set1(ASN1_TYPE *a, int type, const void *value)
        {
        if (!value || (type == V_ASN1_BOOLEAN))
                {
                void *p = (void *)value;
                ASN1_TYPE_set(a, type, p);
                }
        else if (type == V_ASN1_OBJECT)
                {
                ASN1_OBJECT *odup;
                odup = OBJ_dup(value);
                if (!odup)
                        return 0;
                ASN1_TYPE_set(a, type, odup);
                }
        else
                {
                ASN1_STRING *sdup;
                sdup = ASN1_STRING_dup(value);
                if (!sdup)
                        return 0;
                ASN1_TYPE_set(a, type, sdup);
                }
        return 1;
        }

IMPLEMENT_STACK_OF(ASN1_TYPE)
IMPLEMENT_ASN1_SET_OF(ASN1_TYPE)

/* Returns 0 if they are equal, != 0 otherwise. */
int ASN1_TYPE_cmp(ASN1_TYPE *a, ASN1_TYPE *b)
        {
        int result = -1;

        if (!a || !b || a->type != b->type) return -1;

        switch (a->type)
                {
        case V_ASN1_OBJECT:
                result = OBJ_cmp(a->value.object, b->value.object);
                break;
        case V_ASN1_NULL:
```

```
128                 result = 0;     /* They do not have content. */
129                 break;
130         case V_ASN1_INTEGER:
131         case V_ASN1_NEG_INTEGER:
132         case V_ASN1_ENUMERATED:
133         case V_ASN1_NEG_ENUMERATED:
134         case V_ASN1_BIT_STRING:
135         case V_ASN1_OCTET_STRING:
136         case V_ASN1_SEQUENCE:
137         case V_ASN1_SET:
138         case V_ASN1_NUMERICSTRING:
139         case V_ASN1_PRINTABLESTRING:
140         case V_ASN1_T61STRING:
141         case V_ASN1_VIDEOTEXSTRING:
142         case V_ASN1_IA5STRING:
143         case V_ASN1_UTCTIME:
144         case V_ASN1_GENERALIZEDTIME:
145         case V_ASN1_GRAPHICSTRING:
146         case V_ASN1_VISIBLESTRING:
147         case V_ASN1_GENERALSTRING:
148         case V_ASN1_UNIVERSALSTRING:
149         case V_ASN1_BMPSTRING:
150         case V_ASN1_UTF8STRING:
151         case V_ASN1_OTHER:
152         default:
153                 result = ASN1_STRING_cmp((ASN1_STRING *) a->value.ptr,
154                                          (ASN1_STRING *) b->value.ptr);
155                 break;
156                 }

158         return result;
159         }
160 #endif /* ! codereview */
```

```
**********************************************************
    9013 Wed Aug 13 19:52:00 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_utctm.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_utctm.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
```



```
#include "o_time.h"
#include <openssl/asn1.h>

#if 0
int i2d_ASN1_UTCTIME(ASN1_UTCTIME *a, unsigned char **pp)
	{
#ifndef CHARSET_EBCDIC
	return(i2d_ASN1_bytes((ASN1_STRING *)a,pp,
		V_ASN1_UTCTIME,V_ASN1_UNIVERSAL));
#else
	/* KLUDGE! We convert to ascii before writing DER */
	int len;
	char tmp[24];
	ASN1_STRING x = *(ASN1_STRING *)a;

	len = x.length;
	ebcdic2ascii(tmp, x.data, (len >= sizeof tmp) ? sizeof tmp : len);
	x.data = tmp;
	return i2d_ASN1_bytes(&x, pp, V_ASN1_UTCTIME,V_ASN1_UNIVERSAL);
#endif
	}


ASN1_UTCTIME *d2i_ASN1_UTCTIME(ASN1_UTCTIME **a, unsigned char **pp,
	     long length)
	{
	ASN1_UTCTIME *ret=NULL;

	ret=(ASN1_UTCTIME *)d2i_ASN1_bytes((ASN1_STRING **)a,pp,length,
		V_ASN1_UTCTIME,V_ASN1_UNIVERSAL);
	if (ret == NULL)
		{
		ASN1err(ASN1_F_D2I_ASN1_UTCTIME,ERR_R_NESTED_ASN1_ERROR);
		return(NULL);
		}
#ifdef CHARSET_EBCDIC
	ascii2ebcdic(ret->data, ret->data, ret->length);
#endif
	if (!ASN1_UTCTIME_check(ret))
		{
		ASN1err(ASN1_F_D2I_ASN1_UTCTIME,ASN1_R_INVALID_TIME_FORMAT);
		goto err;
		}

	return(ret);
err:
	if ((ret != NULL) && ((a == NULL) || (*a != ret)))
		M_ASN1_UTCTIME_free(ret);
	return(NULL);
	}

#endif

int ASN1_UTCTIME_check(ASN1_UTCTIME *d)
	{
	static const int min[8]={ 0, 1, 1, 0, 0, 0, 0, 0};
	static const int max[8]={99,12,31,23,59,59,12,59};
	char *a;
	int n,i,l,o;

	if (d->type != V_ASN1_UTCTIME) return(0);
	l=d->length;
	a=(char *)d->data;
	o=0;

	if (l < 11) goto err;
```

```
128         for (i=0; i<6; i++)
129                 {
130                 if ((i == 5) && ((a[o] == 'Z') ||
131                         (a[o] == '+') || (a[o] == '-')))
132                         { i++; break; }
133                 if ((a[o] < '0') || (a[o] > '9')) goto err;
134                 n= a[o]-'0';
135                 if (++o > l) goto err;
136
137                 if ((a[o] < '0') || (a[o] > '9')) goto err;
138                 n=(n*10)+ a[o]-'0';
139                 if (++o > l) goto err;
140
141                 if ((n < min[i]) || (n > max[i])) goto err;
142                 }
143         if (a[o] == 'Z')
144                 o++;
145         else if ((a[o] == '+') || (a[o] == '-'))
146                 {
147                 o++;
148                 if (o+4 > l) goto err;
149                 for (i=6; i<8; i++)
150                         {
151                         if ((a[o] < '0') || (a[o] > '9')) goto err;
152                         n= a[o]-'0';
153                         o++;
154                         if ((a[o] < '0') || (a[o] > '9')) goto err;
155                         n=(n*10)+ a[o]-'0';
156                         if ((n < min[i]) || (n > max[i])) goto err;
157                         o++;
158                         }
159                 }
160         return(o == l);
161 err:
162         return(0);
163         }
164
165 int ASN1_UTCTIME_set_string(ASN1_UTCTIME *s, const char *str)
166         {
167         ASN1_UTCTIME t;
168
169         t.type=V_ASN1_UTCTIME;
170         t.length=strlen(str);
171         t.data=(unsigned char *)str;
172         if (ASN1_UTCTIME_check(&t))
173                 {
174                 if (s != NULL)
175                         {
176                         if (!ASN1_STRING_set((ASN1_STRING *)s,
177                                 (unsigned char *)str,t.length))
178                                 return 0;
179                         s->type = V_ASN1_UTCTIME;
180                         }
181                 return(1);
182                 }
183         else
184                 return(0);
185         }
186
187 ASN1_UTCTIME *ASN1_UTCTIME_set(ASN1_UTCTIME *s, time_t t)
188         {
189         return ASN1_UTCTIME_adj(s, t, 0, 0);
190         }
191
192 ASN1_UTCTIME *ASN1_UTCTIME_adj(ASN1_UTCTIME *s, time_t t,
193                                 int offset_day, long offset_sec)
```



```
194         {
195         char *p;
196         struct tm *ts;
197         struct tm data;
198         size_t len = 20;
199         int free_s = 0;
200
201         if (s == NULL)
202                 {
203                 free_s = 1;
204                 s=M_ASN1_UTCTIME_new();
205                 }
206         if (s == NULL)
207                 goto err;
208
209
210         ts=OPENSSL_gmtime(&t, &data);
211         if (ts == NULL)
212                 goto err;
213
214         if (offset_day || offset_sec)
215                 {
216                 if (!OPENSSL_gmtime_adj(ts, offset_day, offset_sec))
217                         goto err;
218                 }
219
220         if((ts->tm_year < 50) || (ts->tm_year >= 150))
221                 goto err;
222
223         p=(char *)s->data;
224         if ((p == NULL) || ((size_t)s->length < len))
225                 {
226                 p=OPENSSL_malloc(len);
227                 if (p == NULL)
228                         {
229                         ASN1err(ASN1_F_ASN1_UTCTIME_ADJ,ERR_R_MALLOC_FAILURE);
230                         goto err;
231                         }
232                 if (s->data != NULL)
233                         OPENSSL_free(s->data);
234                 s->data=(unsigned char *)p;
235                 }
236
237         BIO_snprintf(p,len,"%02d%02d%02d%02d%02d%02dZ",ts->tm_year%100,
238                     ts->tm_mon+1,ts->tm_mday,ts->tm_hour,ts->tm_min,ts->tm_sec)
239         s->length=strlen(p);
240         s->type=V_ASN1_UTCTIME;
241 #ifdef CHARSET_EBCDIC_not
242         ebcdic2ascii(s->data, s->data, s->length);
243 #endif
244         return(s);
245         err:
246         if (free_s && s)
247                 M_ASN1_UTCTIME_free(s);
248         return NULL;
249         }
250
251
252 int ASN1_UTCTIME_cmp_time_t(const ASN1_UTCTIME *s, time_t t)
253         {
254         struct tm *tm;
255         struct tm data;
256         int offset;
257         int year;
258
259 #define g2(p) (((p)[0]-'0')*10+(p)[1]-'0')
```

```
261         if (s->data[12] == 'Z')
262                 offset=0;
263         else
264                 {
265                 offset = g2(s->data+13)*60+g2(s->data+15);
266                 if (s->data[12] == '-')
267                         offset = -offset;
268                 }

270         t -= offset*60; /* FIXME: may overflow in extreme cases */

272         tm = OPENSSL_gmtime(&t, &data);
273         /* NB: -1, 0, 1 already valid return values so use -2 to
274          * indicate error.
275          */
276         if (tm == NULL)
277                 return -2;

279 #define return_cmp(a,b) if ((a)<(b)) return -1; else if ((a)>(b)) return 1
280         year = g2(s->data);
281         if (year < 50)
282                 year += 100;
283         return_cmp(year,             tm->tm_year);
284         return_cmp(g2(s->data+2) - 1, tm->tm_mon);
285         return_cmp(g2(s->data+4),     tm->tm_mday);
286         return_cmp(g2(s->data+6),     tm->tm_hour);
287         return_cmp(g2(s->data+8),     tm->tm_min);
288         return_cmp(g2(s->data+10),    tm->tm_sec);
289 #undef g2
290 #undef return_cmp

292         return 0;
293         }


296 #if 0
297 time_t ASN1_UTCTIME_get(const ASN1_UTCTIME *s)
298         {
299         struct tm tm;
300         int offset;

302         memset(&tm,'\0',sizeof tm);

304 #define g2(p) (((p)[0]-'0')*10+(p)[1]-'0')
305         tm.tm_year=g2(s->data);
306         if(tm.tm_year < 50)
307                 tm.tm_year+=100;
308         tm.tm_mon=g2(s->data+2)-1;
309         tm.tm_mday=g2(s->data+4);
310         tm.tm_hour=g2(s->data+6);
311         tm.tm_min=g2(s->data+8);
312         tm.tm_sec=g2(s->data+10);
313         if(s->data[12] == 'Z')
314                 offset=0;
315         else
316                 {
317                 offset=g2(s->data+13)*60+g2(s->data+15);
318                 if(s->data[12] == '-')
319                         offset= -offset;
320                 }
321 #undef g2

323         return mktime(&tm)-offset*60; /* FIXME: mktime assumes the current timez
324                                        * instead of UTC, and unless we rewrite O
325                                        * in Lisp we cannot locally change the ti
```



```
326                                        * without possibly interfering with other
327                                        * of the program. timegm, which uses UTC,
328                                        * non-standard.
329                                        * Also time_t is inappropriate for genera
330                                        * UTC times because it may a 32 bit type.
331         }
332 #endif
333 #endif /* ! codereview */
```

```
**********************************************************
    7858 Wed Aug 13 19:52:00 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_utf8.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_utf8.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
/* UTF8 utilities */

/* This parses a UTF8 string one character at a time. It is passed a pointer
 * to the string and the length of the string. It sets 'value' to the value of
 * the current character. It returns the number of characters read or a
 * negative error code:
 * -1 = string too short
 * -2 = illegal character
 * -3 = subsequent characters not of the form 10xxxxxx
 * -4 = character encoded incorrectly (not minimal length).
 */

int UTF8_getc(const unsigned char *str, int len, unsigned long *val)
{
        const unsigned char *p;
        unsigned long value;
        int ret;
        if(len <= 0) return 0;
        p = str;

        /* Check syntax and work out the encoded value (if correct) */
        if((*p & 0x80) == 0) {
                value = *p++ & 0x7f;
                ret = 1;
        } else if((*p & 0xe0) == 0xc0) {
                if(len < 2) return -1;
                if((p[1] & 0xc0) != 0x80) return -3;
                value = (*p++ & 0x1f) << 6;
                value |= *p++ & 0x3f;
                if(value < 0x80) return -4;
                ret = 2;
        } else if((*p & 0xf0) == 0xe0) {
                if(len < 3) return -1;
                if( ((p[1] & 0xc0) != 0x80)
                   || ((p[2] & 0xc0) != 0x80) ) return -3;
                value = (*p++ & 0xf) << 12;
                value |= (*p++ & 0x3f) << 6;
                value |= *p++ & 0x3f;
                if(value < 0x800) return -4;
                ret = 3;
        } else if((*p & 0xf8) == 0xf0) {
                if(len < 4) return -1;
                if( ((p[1] & 0xc0) != 0x80)
                   || ((p[2] & 0xc0) != 0x80)
                   || ((p[3] & 0xc0) != 0x80) ) return -3;
                value = ((unsigned long)(*p++ & 0x7)) << 18;
                value |= (*p++ & 0x3f) << 12;
                value |= (*p++ & 0x3f) << 6;
                value |= *p++ & 0x3f;
                if(value < 0x10000) return -4;
                ret = 4;
        } else if((*p & 0xfc) == 0xf8) {
                if(len < 5) return -1;
                if( ((p[1] & 0xc0) != 0x80)
                   || ((p[2] & 0xc0) != 0x80)
                   || ((p[3] & 0xc0) != 0x80)
                   || ((p[4] & 0xc0) != 0x80) ) return -3;
                value = ((unsigned long)(*p++ & 0x3)) << 24;
                value |= ((unsigned long)(*p++ & 0x3f)) << 18;
                value |= ((unsigned long)(*p++ & 0x3f)) << 12;
                value |= (*p++ & 0x3f) << 6;
                value |= *p++ & 0x3f;
                if(value < 0x200000) return -4;
                ret = 5;
```

```
128         } else if((*p & 0xfe) == 0xfc) {
129                 if(len < 6) return -1;
130                 if( ((p[1] & 0xc0) != 0x80)
131                    || ((p[2] & 0xc0) != 0x80)
132                    || ((p[3] & 0xc0) != 0x80)
133                    || ((p[4] & 0xc0) != 0x80)
134                    || ((p[5] & 0xc0) != 0x80) ) return -3;
135                 value = ((unsigned long)(*p++ & 0x1)) << 30;
136                 value |= ((unsigned long)(*p++ & 0x3f)) << 24;
137                 value |= ((unsigned long)(*p++ & 0x3f)) << 18;
138                 value |= ((unsigned long)(*p++ & 0x3f)) << 12;
139                 value |= (*p++ & 0x3f) << 6;
140                 value |= *p++ & 0x3f;
141                 if(value < 0x4000000) return -4;
142                 ret = 6;
143         } else return -2;
144         *val = value;
145         return ret;
146 }

148 /* This takes a character 'value' and writes the UTF8 encoded value in
149  * 'str' where 'str' is a buffer containing 'len' characters. Returns
150  * the number of characters written or -1 if 'len' is too small. 'str' can
151  * be set to NULL in which case it just returns the number of characters.
152  * It will need at most 6 characters.
153  */

155 int UTF8_putc(unsigned char *str, int len, unsigned long value)
156 {
157         if(!str) len = 6;       /* Maximum we will need */
158         else if(len <= 0) return -1;
159         if(value < 0x80) {
160                 if(str) *str = (unsigned char)value;
161                 return 1;
162         }
163         if(value < 0x800) {
164                 if(len < 2) return -1;
165                 if(str) {
166                         *str++ = (unsigned char)(((value >> 6) & 0x1f) | 0xc0);
167                         *str = (unsigned char)((value & 0x3f) | 0x80);
168                 }
169                 return 2;
170         }
171         if(value < 0x10000) {
172                 if(len < 3) return -1;
173                 if(str) {
174                         *str++ = (unsigned char)(((value >> 12) & 0xf) | 0xe0);
175                         *str++ = (unsigned char)(((value >> 6) & 0x3f) | 0x80);
176                         *str = (unsigned char)((value & 0x3f) | 0x80);
177                 }
178                 return 3;
179         }
180         if(value < 0x200000) {
181                 if(len < 4) return -1;
182                 if(str) {
183                         *str++ = (unsigned char)(((value >> 18) & 0x7) | 0xf0);
184                         *str++ = (unsigned char)(((value >> 12) & 0x3f) | 0x80);
185                         *str++ = (unsigned char)(((value >> 6) & 0x3f) | 0x80);
186                         *str = (unsigned char)((value & 0x3f) | 0x80);
187                 }
188                 return 4;
189         }
190         if(value < 0x4000000) {
191                 if(len < 5) return -1;
192                 if(str) {
193                         *str++ = (unsigned char)(((value >> 24) & 0x3) | 0xf8);
```



```
194                         *str++ = (unsigned char)(((value >> 18) & 0x3f) | 0x80);
195                         *str++ = (unsigned char)(((value >> 12) & 0x3f) | 0x80);
196                         *str++ = (unsigned char)(((value >> 6) & 0x3f) | 0x80);
197                         *str = (unsigned char)((value & 0x3f) | 0x80);
198                 }
199                 return 5;
200         }
201         if(len < 6) return -1;
202         if(str) {
203                 *str++ = (unsigned char)(((value >> 30) & 0x1) | 0xfc);
204                 *str++ = (unsigned char)(((value >> 24) & 0x3f) | 0x80);
205                 *str++ = (unsigned char)(((value >> 18) & 0x3f) | 0x80);
206                 *str++ = (unsigned char)(((value >> 12) & 0x3f) | 0x80);
207                 *str++ = (unsigned char)(((value >> 6) & 0x3f) | 0x80);
208                 *str = (unsigned char)((value & 0x3f) | 0x80);
209         }
210         return 6;
211 }
212 #endif /* ! codereview */
```

```
**********************************************************
    6925 Wed Aug 13 19:52:00 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/a_verify.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/a_verify.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
```



```
#include "cryptlib.h"
#include "asn1_locl.h"

#ifndef NO_SYS_TYPES_H
# include <sys/types.h>
#endif

#include <openssl/bn.h>
#include <openssl/x509.h>
#include <openssl/objects.h>
#include <openssl/buffer.h>
#include <openssl/evp.h>

#ifndef NO_ASN1_OLD

int ASN1_verify(i2d_of_void *i2d, X509_ALGOR *a, ASN1_BIT_STRING *signature,
                char *data, EVP_PKEY *pkey)
        {
        EVP_MD_CTX ctx;
        const EVP_MD *type;
        unsigned char *p,*buf_in=NULL;
        int ret= -1,i,inl;

        EVP_MD_CTX_init(&ctx);
        i=OBJ_obj2nid(a->algorithm);
        type=EVP_get_digestbyname(OBJ_nid2sn(i));
        if (type == NULL)
                {
                ASN1err(ASN1_F_ASN1_VERIFY,ASN1_R_UNKNOWN_MESSAGE_DIGEST_ALGORIT
                goto err;
                }

        inl=i2d(data,NULL);
        buf_in=OPENSSL_malloc((unsigned int)inl);
        if (buf_in == NULL)
                {
                ASN1err(ASN1_F_ASN1_VERIFY,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        p=buf_in;

        i2d(data,&p);
        if (!EVP_VerifyInit_ex(&ctx,type, NULL)
                || !EVP_VerifyUpdate(&ctx,(unsigned char *)buf_in,inl))
                {
                ASN1err(ASN1_F_ASN1_VERIFY,ERR_R_EVP_LIB);
                ret=0;
                goto err;
                }

        OPENSSL_cleanse(buf_in,(unsigned int)inl);
        OPENSSL_free(buf_in);

        if (EVP_VerifyFinal(&ctx,(unsigned char *)signature->data,
                        (unsigned int)signature->length,pkey) <= 0)
                {
                ASN1err(ASN1_F_ASN1_VERIFY,ERR_R_EVP_LIB);
                ret=0;
                goto err;
                }
        /* we don't need to zero the 'ctx' because we just checked
         * public information */
        /* memset(&ctx,0,sizeof(ctx)); */
        ret=1;
err:
        EVP_MD_CTX_cleanup(&ctx);
```

```
128         return(ret);
129         }

131 #endif


134 int ASN1_item_verify(const ASN1_ITEM *it, X509_ALGOR *a,
135                 ASN1_BIT_STRING *signature, void *asn, EVP_PKEY *pkey)
136         {
137         EVP_MD_CTX ctx;
138         unsigned char *buf_in=NULL;
139         int ret= -1,inl;

141         int mdnid, pknid;

143         if (!pkey)
144                 {
145                 ASN1err(ASN1_F_ASN1_ITEM_VERIFY, ERR_R_PASSED_NULL_PARAMETER);
146                 return -1;
147                 }

149         EVP_MD_CTX_init(&ctx);

151         /* Convert signature OID into digest and public key OIDs */
152         if (!OBJ_find_sigid_algs(OBJ_obj2nid(a->algorithm), &mdnid, &pknid))
153                 {
154                 ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ASN1_R_UNKNOWN_SIGNATURE_ALGORIT
155                 goto err;
156                 }
157         if (mdnid == NID_undef)
158                 {
159                 if (!pkey->ameth || !pkey->ameth->item_verify)
160                         {
161                         ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ASN1_R_UNKNOWN_SIGNATURE
162                         goto err;
163                         }
164                 ret = pkey->ameth->item_verify(&ctx, it, asn, a,
165                                                         signature, pkey);
166                 /* Return value of 2 means carry on, anything else means we
167                  * exit straight away: either a fatal error of the underlying
168                  * verification routine handles all verification.
169                  */
170                 if (ret != 2)
171                         goto err;
172                 ret = -1;
173                 }
174         else
175                 {
176                 const EVP_MD *type;
177                 type=EVP_get_digestbynid(mdnid);
178                 if (type == NULL)
179                         {
180                         ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ASN1_R_UNKNOWN_MESSAGE_D
181                         goto err;
182                         }

184                 /* Check public key OID matches public key type */
185                 if (EVP_PKEY_type(pknid) != pkey->ameth->pkey_id)
186                         {
187                         ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ASN1_R_WRONG_PUBLIC_KEY_
188                         goto err;
189                         }

191                 if (!EVP_DigestVerifyInit(&ctx, NULL, type, NULL, pkey))
192                         {
193                         ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ERR_R_EVP_LIB);
```



```
194                         ret=0;
195                         goto err;
196                         }

198                 }

200         inl = ASN1_item_i2d(asn, &buf_in, it);

202         if (buf_in == NULL)
203                 {
204                 ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ERR_R_MALLOC_FAILURE);
205                 goto err;
206                 }

208         if (!EVP_DigestVerifyUpdate(&ctx,buf_in,inl))
209                 {
210                 ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ERR_R_EVP_LIB);
211                 ret=0;
212                 goto err;
213                 }

215         OPENSSL_cleanse(buf_in,(unsigned int)inl);
216         OPENSSL_free(buf_in);

218         if (EVP_DigestVerifyFinal(&ctx,signature->data,
219                         (size_t)signature->length) <= 0)
220                 {
221                 ASN1err(ASN1_F_ASN1_ITEM_VERIFY,ERR_R_EVP_LIB);
222                 ret=0;
223                 goto err;
224                 }
225         /* we don't need to zero the 'ctx' because we just checked
226          * public information */
227         /* memset(&ctx,0,sizeof(ctx)); */
228         ret=1;
229 err:
230         EVP_MD_CTX_cleanup(&ctx);
231         return(ret);
232         }
233 #endif /* ! codereview */
```

```
**********************************************************
   12309 Wed Aug 13 19:52:00 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/ameth_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/x509.h>
```



```
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include "asn1_locl.h"

extern const EVP_PKEY_ASN1_METHOD rsa_asn1_meths[];
extern const EVP_PKEY_ASN1_METHOD dsa_asn1_meths[];
extern const EVP_PKEY_ASN1_METHOD dh_asn1_meth;
extern const EVP_PKEY_ASN1_METHOD eckey_asn1_meth;
extern const EVP_PKEY_ASN1_METHOD hmac_asn1_meth;
extern const EVP_PKEY_ASN1_METHOD cmac_asn1_meth;

/* Keep this sorted in type order !! */
static const EVP_PKEY_ASN1_METHOD *standard_methods[] =
        {
#ifndef OPENSSL_NO_RSA
        &rsa_asn1_meths[0],
        &rsa_asn1_meths[1],
#endif
#ifndef OPENSSL_NO_DH
        &dh_asn1_meth,
#endif
#ifndef OPENSSL_NO_DSA
        &dsa_asn1_meths[0],
        &dsa_asn1_meths[1],
        &dsa_asn1_meths[2],
        &dsa_asn1_meths[3],
        &dsa_asn1_meths[4],
#endif
#ifndef OPENSSL_NO_EC
        &eckey_asn1_meth,
#endif
        &hmac_asn1_meth,
        &cmac_asn1_meth
        };

typedef int sk_cmp_fn_type(const char * const *a, const char * const *b);
DECLARE_STACK_OF(EVP_PKEY_ASN1_METHOD)
static STACK_OF(EVP_PKEY_ASN1_METHOD) *app_methods = NULL;



#ifdef TEST
void main()
        {
        int i;
        for (i = 0;
                i < sizeof(standard_methods)/sizeof(EVP_PKEY_ASN1_METHOD *);
                i++)
                fprintf(stderr, "Number %d id=%d (%s)\n", i,
                        standard_methods[i]->pkey_id,
                        OBJ_nid2sn(standard_methods[i]->pkey_id));
        }
#endif

DECLARE_OBJ_BSEARCH_CMP_FN(const EVP_PKEY_ASN1_METHOD *,
                           const EVP_PKEY_ASN1_METHOD *, ameth);

static int ameth_cmp(const EVP_PKEY_ASN1_METHOD * const *a,
                     const EVP_PKEY_ASN1_METHOD * const *b)
        {
        return ((*a)->pkey_id - (*b)->pkey_id);
        }

IMPLEMENT_OBJ_BSEARCH_CMP_FN(const EVP_PKEY_ASN1_METHOD *,
                             const EVP_PKEY_ASN1_METHOD *, ameth);
```

```
int EVP_PKEY_asn1_get_count(void)
        {
        int num = sizeof(standard_methods)/sizeof(EVP_PKEY_ASN1_METHOD *);
        if (app_methods)
                num += sk_EVP_PKEY_ASN1_METHOD_num(app_methods);
        return num;
        }

const EVP_PKEY_ASN1_METHOD *EVP_PKEY_asn1_get0(int idx)
        {
        int num = sizeof(standard_methods)/sizeof(EVP_PKEY_ASN1_METHOD *);
        if (idx < 0)
                return NULL;
        if (idx < num)
                return standard_methods[idx];
        idx -= num;
        return sk_EVP_PKEY_ASN1_METHOD_value(app_methods, idx);
        }

static const EVP_PKEY_ASN1_METHOD *pkey_asn1_find(int type)
        {
        EVP_PKEY_ASN1_METHOD tmp;
        const EVP_PKEY_ASN1_METHOD *t = &tmp, **ret;
        tmp.pkey_id = type;
        if (app_methods)
                {
                int idx;
                idx = sk_EVP_PKEY_ASN1_METHOD_find(app_methods, &tmp);
                if (idx >= 0)
                        return sk_EVP_PKEY_ASN1_METHOD_value(app_methods, idx);
                }
        ret = OBJ_bsearch_ameth(&t, standard_methods,
                          sizeof(standard_methods)
                          /sizeof(EVP_PKEY_ASN1_METHOD *));
        if (!ret || !*ret)
                return NULL;
        return *ret;
        }

/* Find an implementation of an ASN1 algorithm. If 'pe' is not NULL
 * also search through engines and set *pe to a functional reference
 * to the engine implementing 'type' or NULL if no engine implements
 * it.
 */

const EVP_PKEY_ASN1_METHOD *EVP_PKEY_asn1_find(ENGINE **pe, int type)
        {
        const EVP_PKEY_ASN1_METHOD *t;

        for (;;)
                {
                t = pkey_asn1_find(type);
                if (!t || !(t->pkey_flags & ASN1_PKEY_ALIAS))
                        break;
                type = t->pkey_base_id;
                }
        if (pe)
                {
#ifndef OPENSSL_NO_ENGINE
                ENGINE *e;
                /* type will contain the final unaliased type */
                e = ENGINE_get_pkey_asn1_meth_engine(type);
                if (e)
                        {
                        *pe = e;
```



```
                        return ENGINE_get_pkey_asn1_meth(e, type);
                        }
#endif
                *pe = NULL;
                }
        return t;
        }

const EVP_PKEY_ASN1_METHOD *EVP_PKEY_asn1_find_str(ENGINE **pe,
                                        const char *str, int len)
        {
        int i;
        const EVP_PKEY_ASN1_METHOD *ameth;
        if (len == -1)
                len = strlen(str);
        if (pe)
                {
#ifndef OPENSSL_NO_ENGINE
                ENGINE *e;
                ameth = ENGINE_pkey_asn1_find_str(&e, str, len);
                if (ameth)
                        {
                        /* Convert structural into
                         * functional reference
                         */
                        if (!ENGINE_init(e))
                                ameth = NULL;
                        ENGINE_free(e);
                        *pe = e;
                        return ameth;
                        }
#endif
                *pe = NULL;
                }
        for (i = 0; i < EVP_PKEY_asn1_get_count(); i++)
                {
                ameth = EVP_PKEY_asn1_get0(i);
                if (ameth->pkey_flags & ASN1_PKEY_ALIAS)
                        continue;
                if (((int)strlen(ameth->pem_str) == len) &&
                        !strncasecmp(ameth->pem_str, str, len))
                        return ameth;
                }
        return NULL;
        }

int EVP_PKEY_asn1_add0(const EVP_PKEY_ASN1_METHOD *ameth)
        {
        if (app_methods == NULL)
                {
                app_methods = sk_EVP_PKEY_ASN1_METHOD_new(ameth_cmp);
                if (!app_methods)
                        return 0;
                }
        if (!sk_EVP_PKEY_ASN1_METHOD_push(app_methods, ameth))
                return 0;
        sk_EVP_PKEY_ASN1_METHOD_sort(app_methods);
        return 1;
        }

int EVP_PKEY_asn1_add_alias(int to, int from)
        {
        EVP_PKEY_ASN1_METHOD *ameth;
        ameth = EVP_PKEY_asn1_new(from, ASN1_PKEY_ALIAS, NULL, NULL);
        if (!ameth)
                return 0;
```

```
260         ameth->pkey_base_id = to;
261         if (!EVP_PKEY_asn1_add0(ameth))
262                 {
263                 EVP_PKEY_asn1_free(ameth);
264                 return 0;
265                 }
266         return 1;
267         }

269 int EVP_PKEY_asn1_get0_info(int *ppkey_id, int *ppkey_base_id, int *ppkey_flags,
270                              const char **pinfo, const char **ppem_str,
271                                      const EVP_PKEY_ASN1_METHOD *ameth)
272         {
273         if (!ameth)
274                 return 0;
275         if (ppkey_id)
276                 *ppkey_id = ameth->pkey_id;
277         if (ppkey_base_id)
278                 *ppkey_base_id = ameth->pkey_base_id;
279         if (ppkey_flags)
280                 *ppkey_flags = ameth->pkey_flags;
281         if (pinfo)
282                 *pinfo = ameth->info;
283         if (ppem_str)
284                 *ppem_str = ameth->pem_str;
285         return 1;
286         }

288 const EVP_PKEY_ASN1_METHOD* EVP_PKEY_get0_asn1(EVP_PKEY *pkey)
289         {
290         return pkey->ameth;
291         }

293 EVP_PKEY_ASN1_METHOD* EVP_PKEY_asn1_new(int id, int flags,
294                                         const char *pem_str, const char *info)
295         {
296         EVP_PKEY_ASN1_METHOD *ameth;
297         ameth = OPENSSL_malloc(sizeof(EVP_PKEY_ASN1_METHOD));
298         if (!ameth)
299                 return NULL;

301         memset(ameth, 0, sizeof(EVP_PKEY_ASN1_METHOD));

303         ameth->pkey_id = id;
304         ameth->pkey_base_id = id;
305         ameth->pkey_flags = flags | ASN1_PKEY_DYNAMIC;

307         if (info)
308                 {
309                 ameth->info = BUF_strdup(info);
310                 if (!ameth->info)
311                         goto err;
312                 }
313         else
314                 ameth->info = NULL;

316         if (pem_str)
317                 {
318                 ameth->pem_str = BUF_strdup(pem_str);
319                 if (!ameth->pem_str)
320                         goto err;
321                 }
322         else
323                 ameth->pem_str = NULL;

325         ameth->pub_decode = 0;
```



```
326         ameth->pub_encode = 0;
327         ameth->pub_cmp = 0;
328         ameth->pub_print = 0;

330         ameth->priv_decode = 0;
331         ameth->priv_encode = 0;
332         ameth->priv_print = 0;

334         ameth->old_priv_encode = 0;
335         ameth->old_priv_decode = 0;

337         ameth->item_verify = 0;
338         ameth->item_sign = 0;

340         ameth->pkey_size = 0;
341         ameth->pkey_bits = 0;

343         ameth->param_decode = 0;
344         ameth->param_encode = 0;
345         ameth->param_missing = 0;
346         ameth->param_copy = 0;
347         ameth->param_cmp = 0;
348         ameth->param_print = 0;

350         ameth->pkey_free = 0;
351         ameth->pkey_ctrl = 0;

353         return ameth;

355         err:

357         EVP_PKEY_asn1_free(ameth);
358         return NULL;

360         }

362 void EVP_PKEY_asn1_copy(EVP_PKEY_ASN1_METHOD *dst,
363                         const EVP_PKEY_ASN1_METHOD *src)
364         {

366         dst->pub_decode = src->pub_decode;
367         dst->pub_encode = src->pub_encode;
368         dst->pub_cmp = src->pub_cmp;
369         dst->pub_print = src->pub_print;

371         dst->priv_decode = src->priv_decode;
372         dst->priv_encode = src->priv_encode;
373         dst->priv_print = src->priv_print;

375         dst->old_priv_encode = src->old_priv_encode;
376         dst->old_priv_decode = src->old_priv_decode;

378         dst->pkey_size = src->pkey_size;
379         dst->pkey_bits = src->pkey_bits;

381         dst->param_decode = src->param_decode;
382         dst->param_encode = src->param_encode;
383         dst->param_missing = src->param_missing;
384         dst->param_copy = src->param_copy;
385         dst->param_cmp = src->param_cmp;
386         dst->param_print = src->param_print;

388         dst->pkey_free = src->pkey_free;
389         dst->pkey_ctrl = src->pkey_ctrl;

391         dst->item_sign = src->item_sign;
```

```
392         dst->item_verify = src->item_verify;

394         }

396 void EVP_PKEY_asn1_free(EVP_PKEY_ASN1_METHOD *ameth)
397         {
398         if (ameth && (ameth->pkey_flags & ASN1_PKEY_DYNAMIC))
399                 {
400                 if (ameth->pem_str)
401                         OPENSSL_free(ameth->pem_str);
402                 if (ameth->info)
403                         OPENSSL_free(ameth->info);
404                 OPENSSL_free(ameth);
405                 }
406         }

408 void EVP_PKEY_asn1_set_public(EVP_PKEY_ASN1_METHOD *ameth,
409                 int (*pub_decode)(EVP_PKEY *pk, X509_PUBKEY *pub),
410                 int (*pub_encode)(X509_PUBKEY *pub, const EVP_PKEY *pk),
411                 int (*pub_cmp)(const EVP_PKEY *a, const EVP_PKEY *b),
412                 int (*pub_print)(BIO *out, const EVP_PKEY *pkey, int indent,
413                                                 ASN1_PCTX *pctx),
414                 int (*pkey_size)(const EVP_PKEY *pk),
415                 int (*pkey_bits)(const EVP_PKEY *pk))
416         {
417         ameth->pub_decode = pub_decode;
418         ameth->pub_encode = pub_encode;
419         ameth->pub_cmp = pub_cmp;
420         ameth->pub_print = pub_print;
421         ameth->pkey_size = pkey_size;
422         ameth->pkey_bits = pkey_bits;
423         }

425 void EVP_PKEY_asn1_set_private(EVP_PKEY_ASN1_METHOD *ameth,
426                 int (*priv_decode)(EVP_PKEY *pk, PKCS8_PRIV_KEY_INFO *p8inf),
427                 int (*priv_encode)(PKCS8_PRIV_KEY_INFO *p8, const EVP_PKEY *pk),
428                 int (*priv_print)(BIO *out, const EVP_PKEY *pkey, int indent,
429                                                 ASN1_PCTX *pctx))
430         {
431         ameth->priv_decode = priv_decode;
432         ameth->priv_encode = priv_encode;
433         ameth->priv_print = priv_print;
434         }

436 void EVP_PKEY_asn1_set_param(EVP_PKEY_ASN1_METHOD *ameth,
437                 int (*param_decode)(EVP_PKEY *pkey,
438                                 const unsigned char **pder, int derlen),
439                 int (*param_encode)(const EVP_PKEY *pkey, unsigned char **pder),
440                 int (*param_missing)(const EVP_PKEY *pk),
441                 int (*param_copy)(EVP_PKEY *to, const EVP_PKEY *from),
442                 int (*param_cmp)(const EVP_PKEY *a, const EVP_PKEY *b),
443                 int (*param_print)(BIO *out, const EVP_PKEY *pkey, int indent,
444                                                 ASN1_PCTX *pctx))
445         {
446         ameth->param_decode = param_decode;
447         ameth->param_encode = param_encode;
448         ameth->param_missing = param_missing;
449         ameth->param_copy = param_copy;
450         ameth->param_cmp = param_cmp;
451         ameth->param_print = param_print;
452         }

454 void EVP_PKEY_asn1_set_free(EVP_PKEY_ASN1_METHOD *ameth,
455                 void (*pkey_free)(EVP_PKEY *pkey))
456         {
457         ameth->pkey_free = pkey_free;
```



```
458         }

460 void EVP_PKEY_asn1_set_ctrl(EVP_PKEY_ASN1_METHOD *ameth,
461                 int (*pkey_ctrl)(EVP_PKEY *pkey, int op,
462                                                 long arg1, void *arg2))
463         {
464         ameth->pkey_ctrl = pkey_ctrl;
465         }
466 #endif /* ! codereview */
```

```
**********************************************************
   18219 Wed Aug 13 19:52:00 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/asn1_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/asn1_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/asn1.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_ASN1,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_ASN1,0,reason)

static ERR_STRING_DATA ASN1_str_functs[]=
        {
{ERR_FUNC(ASN1_F_A2D_ASN1_OBJECT),      "a2d_ASN1_OBJECT"},
{ERR_FUNC(ASN1_F_A2I_ASN1_ENUMERATED),  "a2i_ASN1_ENUMERATED"},
{ERR_FUNC(ASN1_F_A2I_ASN1_INTEGER),     "a2i_ASN1_INTEGER"},
{ERR_FUNC(ASN1_F_A2I_ASN1_STRING),      "a2i_ASN1_STRING"},
{ERR_FUNC(ASN1_F_APPEND_EXP),   "APPEND_EXP"},
{ERR_FUNC(ASN1_F_ASN1_BIT_STRING_SET_BIT),      "ASN1_BIT_STRING_set_bit"},
{ERR_FUNC(ASN1_F_ASN1_CB),      "ASN1_CB"},
{ERR_FUNC(ASN1_F_ASN1_CHECK_TLEN),      "ASN1_CHECK_TLEN"},
{ERR_FUNC(ASN1_F_ASN1_COLLATE_PRIMITIVE),       "ASN1_COLLATE_PRIMITIVE"},
{ERR_FUNC(ASN1_F_ASN1_COLLECT), "ASN1_COLLECT"},
{ERR_FUNC(ASN1_F_ASN1_D2I_EX_PRIMITIVE),        "ASN1_D2I_EX_PRIMITIVE"},
{ERR_FUNC(ASN1_F_ASN1_D2I_FP),  "ASN1_d2i_fp"},
{ERR_FUNC(ASN1_F_ASN1_D2I_READ_BIO),    "ASN1_D2I_READ_BIO"},
{ERR_FUNC(ASN1_F_ASN1_DIGEST),  "ASN1_digest"},
{ERR_FUNC(ASN1_F_ASN1_DO_ADB),  "ASN1_DO_ADB"},
{ERR_FUNC(ASN1_F_ASN1_DUP),     "ASN1_dup"},
{ERR_FUNC(ASN1_F_ASN1_ENUMERATED_SET),  "ASN1_ENUMERATED_set"},
{ERR_FUNC(ASN1_F_ASN1_ENUMERATED_TO_BN),        "ASN1_ENUMERATED_to_BN"},
{ERR_FUNC(ASN1_F_ASN1_EX_C2I),  "ASN1_EX_C2I"},
{ERR_FUNC(ASN1_F_ASN1_FIND_END),        "ASN1_FIND_END"},
{ERR_FUNC(ASN1_F_ASN1_GENERALIZEDTIME_ADJ),     "ASN1_GENERALIZEDTIME_adj"},
{ERR_FUNC(ASN1_F_ASN1_GENERALIZEDTIME_SET),     "ASN1_GENERALIZEDTIME_set"},
{ERR_FUNC(ASN1_F_ASN1_GENERATE_V3),     "ASN1_generate_v3"},
{ERR_FUNC(ASN1_F_ASN1_GET_OBJECT),      "ASN1_get_object"},
{ERR_FUNC(ASN1_F_ASN1_HEADER_NEW),      "ASN1_HEADER_NEW"},
{ERR_FUNC(ASN1_F_ASN1_I2D_BIO), "ASN1_i2d_bio"},
{ERR_FUNC(ASN1_F_ASN1_I2D_FP),  "ASN1_i2d_fp"},
{ERR_FUNC(ASN1_F_ASN1_INTEGER_SET),     "ASN1_INTEGER_set"},
{ERR_FUNC(ASN1_F_ASN1_INTEGER_TO_BN),   "ASN1_INTEGER_to_BN"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_D2I_FP),     "ASN1_item_d2i_fp"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_DUP),        "ASN1_item_dup"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_EX_COMBINE_NEW),     "ASN1_ITEM_EX_COMBINE_NEW"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_EX_D2I),     "ASN1_ITEM_EX_D2I"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_I2D_BIO),    "ASN1_item_i2d_bio"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_I2D_FP),     "ASN1_item_i2d_fp"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_PACK),       "ASN1_item_pack"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_SIGN),       "ASN1_item_sign"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_SIGN_CTX),   "ASN1_item_sign_ctx"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_UNPACK),     "ASN1_item_unpack"},
{ERR_FUNC(ASN1_F_ASN1_ITEM_VERIFY),     "ASN1_item_verify"},
{ERR_FUNC(ASN1_F_ASN1_MBSTRING_NCOPY),  "ASN1_mbstring_ncopy"},
{ERR_FUNC(ASN1_F_ASN1_OBJECT_NEW),      "ASN1_OBJECT_new"},
{ERR_FUNC(ASN1_F_ASN1_OUTPUT_DATA),     "ASN1_OUTPUT_DATA"},
{ERR_FUNC(ASN1_F_ASN1_PACK_STRING),     "ASN1_pack_string"},
{ERR_FUNC(ASN1_F_ASN1_PCTX_NEW),        "ASN1_PCTX_new"},
{ERR_FUNC(ASN1_F_ASN1_PKCS5_PBE_SET),   "ASN1_PKCS5_PBE_SET"},
{ERR_FUNC(ASN1_F_ASN1_SEQ_PACK),        "ASN1_seq_pack"},
{ERR_FUNC(ASN1_F_ASN1_SEQ_UNPACK),      "ASN1_seq_unpack"},
{ERR_FUNC(ASN1_F_ASN1_SIGN),    "ASN1_sign"},
{ERR_FUNC(ASN1_F_ASN1_STR2TYPE),        "ASN1_STR2TYPE"},
{ERR_FUNC(ASN1_F_ASN1_STRING_SET),      "ASN1_STRING_set"},
{ERR_FUNC(ASN1_F_ASN1_STRING_TABLE_ADD),        "ASN1_STRING_TABLE_add"},
{ERR_FUNC(ASN1_F_ASN1_STRING_TYPE_NEW), "ASN1_STRING_type_new"},
{ERR_FUNC(ASN1_F_ASN1_TEMPLATE_EX_D2I), "ASN1_TEMPLATE_EX_D2I"},
{ERR_FUNC(ASN1_F_ASN1_TEMPLATE_NEW),    "ASN1_TEMPLATE_NEW"},
```

```
{ERR_FUNC(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I),      "ASN1_TEMPLATE_NOEXP_D2I"},
{ERR_FUNC(ASN1_F_ASN1_TIME_ADJ),        "ASN1_TIME_adj"},
{ERR_FUNC(ASN1_F_ASN1_TIME_SET),        "ASN1_TIME_set"},
{ERR_FUNC(ASN1_F_ASN1_TYPE_GET_INT_OCTETSTRING),        "ASN1_TYPE_get_int_octet
{ERR_FUNC(ASN1_F_ASN1_TYPE_GET_OCTETSTRING),    "ASN1_TYPE_get_octetstring"},
{ERR_FUNC(ASN1_F_ASN1_UNPACK_STRING),   "ASN1_unpack_string"},
{ERR_FUNC(ASN1_F_ASN1_UTCTIME_ADJ),     "ASN1_UTCTIME_adj"},
{ERR_FUNC(ASN1_F_ASN1_UTCTIME_SET),     "ASN1_UTCTIME_set"},
{ERR_FUNC(ASN1_F_ASN1_VERIFY),  "ASN1_verify"},
{ERR_FUNC(ASN1_F_B64_READ_ASN1),        "B64_READ_ASN1"},
{ERR_FUNC(ASN1_F_B64_WRITE_ASN1),       "B64_WRITE_ASN1"},
{ERR_FUNC(ASN1_F_BIO_NEW_NDEF), "BIO_new_NDEF"},
{ERR_FUNC(ASN1_F_BITSTR_CB),    "BITSTR_CB"},
{ERR_FUNC(ASN1_F_BN_TO_ASN1_ENUMERATED),        "BN_to_ASN1_ENUMERATED"},
{ERR_FUNC(ASN1_F_BN_TO_ASN1_INTEGER),   "BN_to_ASN1_INTEGER"},
{ERR_FUNC(ASN1_F_C2I_ASN1_BIT_STRING),  "c2i_ASN1_BIT_STRING"},
{ERR_FUNC(ASN1_F_C2I_ASN1_INTEGER),     "c2i_ASN1_INTEGER"},
{ERR_FUNC(ASN1_F_C2I_ASN1_OBJECT),      "c2i_ASN1_OBJECT"},
{ERR_FUNC(ASN1_F_COLLECT_DATA), "COLLECT_DATA"},
{ERR_FUNC(ASN1_F_D2I_ASN1_BIT_STRING),  "D2I_ASN1_BIT_STRING"},
{ERR_FUNC(ASN1_F_D2I_ASN1_BOOLEAN),     "d2i_ASN1_BOOLEAN"},
{ERR_FUNC(ASN1_F_D2I_ASN1_BYTES),       "d2i_ASN1_bytes"},
{ERR_FUNC(ASN1_F_D2I_ASN1_GENERALIZEDTIME),     "D2I_ASN1_GENERALIZEDTIME"},
{ERR_FUNC(ASN1_F_D2I_ASN1_HEADER),      "D2I_ASN1_HEADER"},
{ERR_FUNC(ASN1_F_D2I_ASN1_INTEGER),     "D2I_ASN1_INTEGER"},
{ERR_FUNC(ASN1_F_D2I_ASN1_OBJECT),      "d2i_ASN1_OBJECT"},
{ERR_FUNC(ASN1_F_D2I_ASN1_SET), "d2i_ASN1_SET"},
{ERR_FUNC(ASN1_F_D2I_ASN1_TYPE_BYTES),  "d2i_ASN1_type_bytes"},
{ERR_FUNC(ASN1_F_D2I_ASN1_UINTEGER),    "d2i_ASN1_UINTEGER"},
{ERR_FUNC(ASN1_F_D2I_ASN1_UTCTIME),     "D2I_ASN1_UTCTIME"},
{ERR_FUNC(ASN1_F_D2I_AUTOPRIVATEKEY),   "d2i_AutoPrivateKey"},
{ERR_FUNC(ASN1_F_D2I_NETSCAPE_RSA),     "d2i_Netscape_RSA"},
{ERR_FUNC(ASN1_F_D2I_NETSCAPE_RSA_2),   "D2I_NETSCAPE_RSA_2"},
{ERR_FUNC(ASN1_F_D2I_PRIVATEKEY),       "d2i_PrivateKey"},
{ERR_FUNC(ASN1_F_D2I_PUBLICKEY),        "d2i_PublicKey"},
{ERR_FUNC(ASN1_F_D2I_RSA_NET),  "d2i_RSA_NET"},
{ERR_FUNC(ASN1_F_D2I_RSA_NET_2),        "D2I_RSA_NET_2"},
{ERR_FUNC(ASN1_F_D2I_X509),     "D2I_X509"},
{ERR_FUNC(ASN1_F_D2I_X509_CINF),        "D2I_X509_CINF"},
{ERR_FUNC(ASN1_F_D2I_X509_PKEY),        "d2i_X509_PKEY"},
{ERR_FUNC(ASN1_F_I2D_ASN1_BIO_STREAM),  "i2d_ASN1_bio_stream"},
{ERR_FUNC(ASN1_F_I2D_ASN1_SET), "i2d_ASN1_SET"},
{ERR_FUNC(ASN1_F_I2D_ASN1_TIME),        "I2D_ASN1_TIME"},
{ERR_FUNC(ASN1_F_I2D_DSA_PUBKEY),       "i2d_DSA_PUBKEY"},
{ERR_FUNC(ASN1_F_I2D_EC_PUBKEY),        "i2d_EC_PUBKEY"},
{ERR_FUNC(ASN1_F_I2D_PRIVATEKEY),       "i2d_PrivateKey"},
{ERR_FUNC(ASN1_F_I2D_PUBLICKEY),        "i2d_PublicKey"},
{ERR_FUNC(ASN1_F_I2D_RSA_NET),  "i2d_RSA_NET"},
{ERR_FUNC(ASN1_F_I2D_RSA_PUBKEY),       "i2d_RSA_PUBKEY"},
{ERR_FUNC(ASN1_F_LONG_C2I),     "LONG_C2I"},
{ERR_FUNC(ASN1_F_OID_MODULE_INIT),      "OID_MODULE_INIT"},
{ERR_FUNC(ASN1_F_PARSE_TAGGING),        "PARSE_TAGGING"},
{ERR_FUNC(ASN1_F_PKCS5_PBE2_SET_IV),    "PKCS5_pbe2_set_iv"},
{ERR_FUNC(ASN1_F_PKCS5_PBE_SET),        "PKCS5_pbe_set"},
{ERR_FUNC(ASN1_F_PKCS5_PBE_SET0_ALGOR), "PKCS5_pbe_set0_algor"},
{ERR_FUNC(ASN1_F_PKCS5_PBKDF2_SET),     "PKCS5_pbkdf2_set"},
{ERR_FUNC(ASN1_F_SMIME_READ_ASN1),      "SMIME_read_ASN1"},
{ERR_FUNC(ASN1_F_SMIME_TEXT),   "SMIME_text"},
{ERR_FUNC(ASN1_F_X509_CINF_NEW),        "X509_CINF_NEW"},
{ERR_FUNC(ASN1_F_X509_CRL_ADD0_REVOKED),        "X509_CRL_add0_revoked"},
{ERR_FUNC(ASN1_F_X509_INFO_NEW),        "X509_INFO_new"},
{ERR_FUNC(ASN1_F_X509_NAME_ENCODE),     "X509_NAME_ENCODE"},
{ERR_FUNC(ASN1_F_X509_NAME_EX_D2I),     "X509_NAME_EX_D2I"},
{ERR_FUNC(ASN1_F_X509_NAME_EX_NEW),     "X509_NAME_EX_NEW"},
{ERR_FUNC(ASN1_F_X509_NEW),     "X509_NEW"},
{ERR_FUNC(ASN1_F_X509_PKEY_NEW),        "X509_PKEY_new"},
```



```
{0,NULL}
        };

static ERR_STRING_DATA ASN1_str_reasons[]=
        {
{ERR_REASON(ASN1_R_ADDING_OBJECT)        ,"adding object"},
{ERR_REASON(ASN1_R_ASN1_PARSE_ERROR)     ,"asn1 parse error"},
{ERR_REASON(ASN1_R_ASN1_SIG_PARSE_ERROR) ,"asn1 sig parse error"},
{ERR_REASON(ASN1_R_AUX_ERROR)            ,"aux error"},
{ERR_REASON(ASN1_R_BAD_CLASS)            ,"bad class"},
{ERR_REASON(ASN1_R_BAD_OBJECT_HEADER)    ,"bad object header"},
{ERR_REASON(ASN1_R_BAD_PASSWORD_READ)    ,"bad password read"},
{ERR_REASON(ASN1_R_BAD_TAG)              ,"bad tag"},
{ERR_REASON(ASN1_R_BMPSTRING_IS_WRONG_LENGTH),"bmpstring is wrong length"},
{ERR_REASON(ASN1_R_BN_LIB)               ,"bn lib"},
{ERR_REASON(ASN1_R_BOOLEAN_IS_WRONG_LENGTH),"boolean is wrong length"},
{ERR_REASON(ASN1_R_BUFFER_TOO_SMALL)     ,"buffer too small"},
{ERR_REASON(ASN1_R_CIPHER_HAS_NO_OBJECT_IDENTIFIER),"cipher has no object identi
{ERR_REASON(ASN1_R_CONTEXT_NOT_INITIALISED),"context not initialised"},
{ERR_REASON(ASN1_R_DATA_IS_WRONG)        ,"data is wrong"},
{ERR_REASON(ASN1_R_DECODE_ERROR)         ,"decode error"},
{ERR_REASON(ASN1_R_DECODING_ERROR)       ,"decoding error"},
{ERR_REASON(ASN1_R_DEPTH_EXCEEDED)       ,"depth exceeded"},
{ERR_REASON(ASN1_R_DIGEST_AND_KEY_TYPE_NOT_SUPPORTED),"digest and key type not s
{ERR_REASON(ASN1_R_ENCODE_ERROR)         ,"encode error"},
{ERR_REASON(ASN1_R_ERROR_GETTING_TIME)   ,"error getting time"},
{ERR_REASON(ASN1_R_ERROR_LOADING_SECTION),"error loading section"},
{ERR_REASON(ASN1_R_ERROR_PARSING_SET_ELEMENT),"error parsing set element"},
{ERR_REASON(ASN1_R_ERROR_SETTING_CIPHER_PARAMS),"error setting cipher params"},
{ERR_REASON(ASN1_R_EXPECTING_AN_INTEGER) ,"expecting an integer"},
{ERR_REASON(ASN1_R_EXPECTING_AN_OBJECT)  ,"expecting an object"},
{ERR_REASON(ASN1_R_EXPECTING_A_BOOLEAN)  ,"expecting a boolean"},
{ERR_REASON(ASN1_R_EXPECTING_A_TIME)     ,"expecting a time"},
{ERR_REASON(ASN1_R_EXPLICIT_LENGTH_MISMATCH),"explicit length mismatch"},
{ERR_REASON(ASN1_R_EXPLICIT_TAG_NOT_CONSTRUCTED),"explicit tag not constructed"}
{ERR_REASON(ASN1_R_FIELD_MISSING)        ,"field missing"},
{ERR_REASON(ASN1_R_FIRST_NUM_TOO_LARGE)  ,"first num too large"},
{ERR_REASON(ASN1_R_HEADER_TOO_LONG)      ,"header too long"},
{ERR_REASON(ASN1_R_ILLEGAL_BITSTRING_FORMAT),"illegal bitstring format"},
{ERR_REASON(ASN1_R_ILLEGAL_BOOLEAN)      ,"illegal boolean"},
{ERR_REASON(ASN1_R_ILLEGAL_CHARACTERS)   ,"illegal characters"},
{ERR_REASON(ASN1_R_ILLEGAL_FORMAT)       ,"illegal format"},
{ERR_REASON(ASN1_R_ILLEGAL_HEX)          ,"illegal hex"},
{ERR_REASON(ASN1_R_ILLEGAL_IMPLICIT_TAG) ,"illegal implicit tag"},
{ERR_REASON(ASN1_R_ILLEGAL_INTEGER)      ,"illegal integer"},
{ERR_REASON(ASN1_R_ILLEGAL_NESTED_TAGGING),"illegal nested tagging"},
{ERR_REASON(ASN1_R_ILLEGAL_NULL)         ,"illegal null"},
{ERR_REASON(ASN1_R_ILLEGAL_NULL_VALUE)   ,"illegal null value"},
{ERR_REASON(ASN1_R_ILLEGAL_OBJECT)       ,"illegal object"},
{ERR_REASON(ASN1_R_ILLEGAL_OPTIONAL_ANY) ,"illegal optional any"},
{ERR_REASON(ASN1_R_ILLEGAL_OPTIONS_ON_ITEM_TEMPLATE),"illegal options on item te
{ERR_REASON(ASN1_R_ILLEGAL_TAGGED_ANY)   ,"illegal tagged any"},
{ERR_REASON(ASN1_R_ILLEGAL_TIME_VALUE)   ,"illegal time value"},
{ERR_REASON(ASN1_R_INTEGER_NOT_ASCII_FORMAT),"integer not ascii format"},
{ERR_REASON(ASN1_R_INTEGER_TOO_LARGE_FOR_LONG),"integer too large for long"},
{ERR_REASON(ASN1_R_INVALID_BMPSTRING_LENGTH),"invalid bmpstring length"},
{ERR_REASON(ASN1_R_INVALID_DIGIT)        ,"invalid digit"},
{ERR_REASON(ASN1_R_INVALID_MIME_TYPE)    ,"invalid mime type"},
{ERR_REASON(ASN1_R_INVALID_MODIFIER)     ,"invalid modifier"},
{ERR_REASON(ASN1_R_INVALID_NUMBER)       ,"invalid number"},
{ERR_REASON(ASN1_R_INVALID_OBJECT_ENCODING),"invalid object encoding"},
{ERR_REASON(ASN1_R_INVALID_SEPARATOR)    ,"invalid separator"},
{ERR_REASON(ASN1_R_INVALID_TIME_FORMAT)  ,"invalid time format"},
{ERR_REASON(ASN1_R_INVALID_UNIVERSALSTRING_LENGTH),"invalid universalstring leng
{ERR_REASON(ASN1_R_INVALID_UTF8STRING)   ,"invalid utf8string"},
{ERR_REASON(ASN1_R_IV_TOO_LARGE)         ,"iv too large"},
```

```
{ERR_REASON(ASN1_R_LENGTH_ERROR)         ,"length error"},
{ERR_REASON(ASN1_R_LIST_ERROR)           ,"list error"},
{ERR_REASON(ASN1_R_MIME_NO_CONTENT_TYPE) ,"mime no content type"},
{ERR_REASON(ASN1_R_MIME_PARSE_ERROR)     ,"mime parse error"},
{ERR_REASON(ASN1_R_MIME_SIG_PARSE_ERROR) ,"mime sig parse error"},
{ERR_REASON(ASN1_R_MISSING_EOC)          ,"missing eoc"},
{ERR_REASON(ASN1_R_MISSING_SECOND_NUMBER),"missing second number"},
{ERR_REASON(ASN1_R_MISSING_VALUE)        ,"missing value"},
{ERR_REASON(ASN1_R_MSTRING_NOT_UNIVERSAL),"mstring not universal"},
{ERR_REASON(ASN1_R_MSTRING_WRONG_TAG)    ,"mstring wrong tag"},
{ERR_REASON(ASN1_R_NESTED_ASN1_STRING)   ,"nested asn1 string"},
{ERR_REASON(ASN1_R_NON_HEX_CHARACTERS)   ,"non hex characters"},
{ERR_REASON(ASN1_R_NOT_ASCII_FORMAT)     ,"not ascii format"},
{ERR_REASON(ASN1_R_NOT_ENOUGH_DATA)      ,"not enough data"},
{ERR_REASON(ASN1_R_NO_CONTENT_TYPE)      ,"no content type"},
{ERR_REASON(ASN1_R_NO_DEFAULT_DIGEST)    ,"no default digest"},
{ERR_REASON(ASN1_R_NO_MATCHING_CHOICE_TYPE),"no matching choice type"},
{ERR_REASON(ASN1_R_NO_MULTIPART_BODY_FAILURE),"no multipart body failure"},
{ERR_REASON(ASN1_R_NO_MULTIPART_BOUNDARY),"no multipart boundary"},
{ERR_REASON(ASN1_R_NO_SIG_CONTENT_TYPE)  ,"no sig content type"},
{ERR_REASON(ASN1_R_NULL_IS_WRONG_LENGTH) ,"null is wrong length"},
{ERR_REASON(ASN1_R_OBJECT_NOT_ASCII_FORMAT),"object not ascii format"},
{ERR_REASON(ASN1_R_ODD_NUMBER_OF_CHARS)  ,"odd number of chars"},
{ERR_REASON(ASN1_R_PRIVATE_KEY_HEADER_MISSING),"private key header missing"},
{ERR_REASON(ASN1_R_SECOND_NUMBER_TOO_LARGE),"second number too large"},
{ERR_REASON(ASN1_R_SEQUENCE_LENGTH_MISMATCH),"sequence length mismatch"},
{ERR_REASON(ASN1_R_SEQUENCE_NOT_CONSTRUCTED),"sequence not constructed"},
{ERR_REASON(ASN1_R_SEQUENCE_OR_SET_NEEDS_CONFIG),"sequence or set needs config"}
{ERR_REASON(ASN1_R_SHORT_LINE)           ,"short line"},
{ERR_REASON(ASN1_R_SIG_INVALID_MIME_TYPE),"sig invalid mime type"},
{ERR_REASON(ASN1_R_STREAMING_NOT_SUPPORTED),"streaming not supported"},
{ERR_REASON(ASN1_R_STRING_TOO_LONG)      ,"string too long"},
{ERR_REASON(ASN1_R_STRING_TOO_SHORT)     ,"string too short"},
{ERR_REASON(ASN1_R_TAG_VALUE_TOO_HIGH)   ,"tag value too high"},
{ERR_REASON(ASN1_R_THE_ASN1_OBJECT_IDENTIFIER_IS_NOT_KNOWN_FOR_THIS_MD),"the asn
{ERR_REASON(ASN1_R_TIME_NOT_ASCII_FORMAT),"time not ascii format"},
{ERR_REASON(ASN1_R_TOO_LONG)             ,"too long"},
{ERR_REASON(ASN1_R_TYPE_NOT_CONSTRUCTED) ,"type not constructed"},
{ERR_REASON(ASN1_R_UNABLE_TO_DECODE_RSA_KEY),"unable to decode rsa key"},
{ERR_REASON(ASN1_R_UNABLE_TO_DECODE_RSA_PRIVATE_KEY),"unable to decode rsa priva
{ERR_REASON(ASN1_R_UNEXPECTED_EOC)       ,"unexpected eoc"},
{ERR_REASON(ASN1_R_UNIVERSALSTRING_IS_WRONG_LENGTH),"universalstring is wrong le
{ERR_REASON(ASN1_R_UNKNOWN_FORMAT)       ,"unknown format"},
{ERR_REASON(ASN1_R_UNKNOWN_MESSAGE_DIGEST_ALGORITHM),"unknown message digest alg
{ERR_REASON(ASN1_R_UNKNOWN_OBJECT_TYPE)  ,"unknown object type"},
{ERR_REASON(ASN1_R_UNKNOWN_PUBLIC_KEY_TYPE),"unknown public key type"},
{ERR_REASON(ASN1_R_UNKNOWN_SIGNATURE_ALGORITHM),"unknown signature algorithm"},
{ERR_REASON(ASN1_R_UNKNOWN_TAG)          ,"unknown tag"},
{ERR_REASON(ASN1_R_UNKOWN_FORMAT)        ,"unknown format"},
{ERR_REASON(ASN1_R_UNSUPPORTED_ANY_DEFINED_BY_TYPE),"unsupported any defined by
{ERR_REASON(ASN1_R_UNSUPPORTED_CIPHER)   ,"unsupported cipher"},
{ERR_REASON(ASN1_R_UNSUPPORTED_ENCRYPTION_ALGORITHM),"unsupported encryption alg
{ERR_REASON(ASN1_R_UNSUPPORTED_PUBLIC_KEY_TYPE),"unsupported public key type"},
{ERR_REASON(ASN1_R_UNSUPPORTED_TYPE)     ,"unsupported type"},
{ERR_REASON(ASN1_R_WRONG_PUBLIC_KEY_TYPE),"wrong public key type"},
{ERR_REASON(ASN1_R_WRONG_TAG)            ,"wrong tag"},
{ERR_REASON(ASN1_R_WRONG_TYPE)           ,"wrong type"},
{0,NULL}
        };

#endif

void ERR_load_ASN1_strings(void)
        {
#ifndef OPENSSL_NO_ERR
```



```
        if (ERR_func_error_string(ASN1_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,ASN1_str_functs);
                ERR_load_strings(0,ASN1_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
   21079 Wed Aug 13 19:52:01 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/asn1_gen.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* asn1_gen.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/asn1.h>
#include <openssl/x509v3.h>
```



```
#define ASN1_GEN_FLAG           0x10000
#define ASN1_GEN_FLAG_IMP       (ASN1_GEN_FLAG|1)
#define ASN1_GEN_FLAG_EXP       (ASN1_GEN_FLAG|2)
#define ASN1_GEN_FLAG_TAG       (ASN1_GEN_FLAG|3)
#define ASN1_GEN_FLAG_BITWRAP   (ASN1_GEN_FLAG|4)
#define ASN1_GEN_FLAG_OCTWRAP   (ASN1_GEN_FLAG|5)
#define ASN1_GEN_FLAG_SEQWRAP   (ASN1_GEN_FLAG|6)
#define ASN1_GEN_FLAG_SETWRAP   (ASN1_GEN_FLAG|7)
#define ASN1_GEN_FLAG_FORMAT    (ASN1_GEN_FLAG|8)

#define ASN1_GEN_STR(str,val)   {str, sizeof(str) - 1, val}

#define ASN1_FLAG_EXP_MAX       20

/* Input formats */

/* ASCII: default */
#define ASN1_GEN_FORMAT_ASCII   1
/* UTF8 */
#define ASN1_GEN_FORMAT_UTF8    2
/* Hex */
#define ASN1_GEN_FORMAT_HEX     3
/* List of bits */
#define ASN1_GEN_FORMAT_BITLIST 4


struct tag_name_st
        {
        const char *strnam;
        int len;
        int tag;
        };

typedef struct
        {
        int exp_tag;
        int exp_class;
        int exp_constructed;
        int exp_pad;
        long exp_len;
        } tag_exp_type;

typedef struct
        {
        int imp_tag;
        int imp_class;
        int utype;
        int format;
        const char *str;
        tag_exp_type exp_list[ASN1_FLAG_EXP_MAX];
        int exp_count;
        } tag_exp_arg;

static int bitstr_cb(const char *elem, int len, void *bitstr);
static int asn1_cb(const char *elem, int len, void *bitstr);
static int append_exp(tag_exp_arg *arg, int exp_tag, int exp_class, int exp_cons
static int parse_tagging(const char *vstart, int vlen, int *ptag, int *pclass);
static ASN1_TYPE *asn1_multi(int utype, const char *section, X509V3_CTX *cnf);
static ASN1_TYPE *asn1_str2type(const char *str, int format, int utype);
static int asn1_str2tag(const char *tagstr, int len);

ASN1_TYPE *ASN1_generate_nconf(char *str, CONF *nconf)
        {
        X509V3_CTX cnf;
```

```
128         if (!nconf)
129                 return ASN1_generate_v3(str, NULL);

131         X509V3_set_nconf(&cnf, nconf);
132         return ASN1_generate_v3(str, &cnf);
133         }

135 ASN1_TYPE *ASN1_generate_v3(char *str, X509V3_CTX *cnf)
136         {
137         ASN1_TYPE *ret;
138         tag_exp_arg asn1_tags;
139         tag_exp_type *etmp;

141         int i, len;

143         unsigned char *orig_der = NULL, *new_der = NULL;
144         const unsigned char *cpy_start;
145         unsigned char *p;
146         const unsigned char *cp;
147         int cpy_len;
148         long hdr_len;
149         int hdr_constructed = 0, hdr_tag, hdr_class;
150         int r;

152         asn1_tags.imp_tag = -1;
153         asn1_tags.imp_class = -1;
154         asn1_tags.format = ASN1_GEN_FORMAT_ASCII;
155         asn1_tags.exp_count = 0;
156         if (CONF_parse_list(str, ',', 1, asn1_cb, &asn1_tags) != 0)
157                 return NULL;

159         if ((asn1_tags.utype == V_ASN1_SEQUENCE) || (asn1_tags.utype == V_ASN1_S
160                 {
161                 if (!cnf)
162                         {
163                         ASN1err(ASN1_F_ASN1_GENERATE_V3, ASN1_R_SEQUENCE_OR_SET_
164                         return NULL;
165                         }
166                 ret = asn1_multi(asn1_tags.utype, asn1_tags.str, cnf);
167                 }
168         else
169                 ret = asn1_str2type(asn1_tags.str, asn1_tags.format, asn1_tags.u

171         if (!ret)
172                 return NULL;

174         /* If no tagging return base type */
175         if ((asn1_tags.imp_tag == -1) && (asn1_tags.exp_count == 0))
176                 return ret;

178         /* Generate the encoding */
179         cpy_len = i2d_ASN1_TYPE(ret, &orig_der);
180         ASN1_TYPE_free(ret);
181         ret = NULL;
182         /* Set point to start copying for modified encoding */
183         cpy_start = orig_der;

185         /* Do we need IMPLICIT tagging? */
186         if (asn1_tags.imp_tag != -1)
187                 {
188                 /* If IMPLICIT we will replace the underlying tag */
189                 /* Skip existing tag+len */
190                 r = ASN1_get_object(&cpy_start, &hdr_len, &hdr_tag, &hdr_class,
191                 if (r & 0x80)
192                         goto err;
193                 /* Update copy length */
```



```
194                 cpy_len -= cpy_start - orig_der;
195                 /* For IMPLICIT tagging the length should match the
196                  * original length and constructed flag should be
197                  * consistent.
198                  */
199                 if (r & 0x1)
200                         {
201                         /* Indefinite length constructed */
202                         hdr_constructed = 2;
203                         hdr_len = 0;
204                         }
205                 else
206                         /* Just retain constructed flag */
207                         hdr_constructed = r & V_ASN1_CONSTRUCTED;
208                 /* Work out new length with IMPLICIT tag: ignore constructed
209                  * because it will mess up if indefinite length
210                  */
211                 len = ASN1_object_size(0, hdr_len, asn1_tags.imp_tag);
212                 }
213         else
214                 len = cpy_len;

216         /* Work out length in any EXPLICIT, starting from end */

218         for(i = 0, etmp = asn1_tags.exp_list + asn1_tags.exp_count - 1; i < asn1
219                 {
220                 /* Content length: number of content octets + any padding */
221                 len += etmp->exp_pad;
222                 etmp->exp_len = len;
223                 /* Total object length: length including new header */
224                 len = ASN1_object_size(0, len, etmp->exp_tag);
225                 }

227         /* Allocate buffer for new encoding */

229         new_der = OPENSSL_malloc(len);
230         if (!new_der)
231                 goto err;

233         /* Generate tagged encoding */

235         p = new_der;

237         /* Output explicit tags first */

239         for (i = 0, etmp = asn1_tags.exp_list; i < asn1_tags.exp_count; i++, etm
240                 {
241                 ASN1_put_object(&p, etmp->exp_constructed, etmp->exp_len,
242                                         etmp->exp_tag, etmp->exp_class);
243                 if (etmp->exp_pad)
244                         *p++ = 0;
245                 }

247         /* If IMPLICIT, output tag */

249         if (asn1_tags.imp_tag != -1)
250                 {
251                 if (asn1_tags.imp_class == V_ASN1_UNIVERSAL
252                     && (asn1_tags.imp_tag == V_ASN1_SEQUENCE
253                      || asn1_tags.imp_tag == V_ASN1_SET) )
254                         hdr_constructed = V_ASN1_CONSTRUCTED;
255                 ASN1_put_object(&p, hdr_constructed, hdr_len,
256                                         asn1_tags.imp_tag, asn1_tags.imp_class);
257                 }

259         /* Copy across original encoding */
```

```
260         memcpy(p, cpy_start, cpy_len);

262         cp = new_der;

264         /* Obtain new ASN1_TYPE structure */
265         ret = d2i_ASN1_TYPE(NULL, &cp, len);

267         err:
268         if (orig_der)
269                 OPENSSL_free(orig_der);
270         if (new_der)
271                 OPENSSL_free(new_der);

273         return ret;

275         }

277 static int asn1_cb(const char *elem, int len, void *bitstr)
278         {
279         tag_exp_arg *arg = bitstr;
280         int i;
281         int utype;
282         int vlen = 0;
283         const char *p, *vstart = NULL;

285         int tmp_tag, tmp_class;

287         for(i = 0, p = elem; i < len; p++, i++)
288                 {
289                 /* Look for the ':' in name value pairs */
290                 if (*p == ':')
291                         {
292                         vstart = p + 1;
293                         vlen = len - (vstart - elem);
294                         len = p - elem;
295                         break;
296                         }
297                 }

299         utype = asn1_str2tag(elem, len);

301         if (utype == -1)
302                 {
303                 ASN1err(ASN1_F_ASN1_CB, ASN1_R_UNKNOWN_TAG);
304                 ERR_add_error_data(2, "tag=", elem);
305                 return -1;
306                 }

308         /* If this is not a modifier mark end of string and exit */
309         if (!(utype & ASN1_GEN_FLAG))
310                 {
311                 arg->utype = utype;
312                 arg->str = vstart;
313                 /* If no value and not end of string, error */
314                 if (!vstart && elem[len])
315                         {
316                         ASN1err(ASN1_F_ASN1_CB, ASN1_R_MISSING_VALUE);
317                         return -1;
318                         }
319                 return 0;
320                 }

322         switch(utype)
323                 {

325                 case ASN1_GEN_FLAG_IMP:
```



```
326                 /* Check for illegal multiple IMPLICIT tagging */
327                 if (arg->imp_tag != -1)
328                         {
329                         ASN1err(ASN1_F_ASN1_CB, ASN1_R_ILLEGAL_NESTED_TAGGING);
330                         return -1;
331                         }
332                 if (!parse_tagging(vstart, vlen, &arg->imp_tag, &arg->imp_class)
333                         return -1;
334                 break;

336                 case ASN1_GEN_FLAG_EXP:

338                 if (!parse_tagging(vstart, vlen, &tmp_tag, &tmp_class))
339                         return -1;
340                 if (!append_exp(arg, tmp_tag, tmp_class, 1, 0, 0))
341                         return -1;
342                 break;

344                 case ASN1_GEN_FLAG_SEQWRAP:
345                 if (!append_exp(arg, V_ASN1_SEQUENCE, V_ASN1_UNIVERSAL, 1, 0, 1)
346                         return -1;
347                 break;

349                 case ASN1_GEN_FLAG_SETWRAP:
350                 if (!append_exp(arg, V_ASN1_SET, V_ASN1_UNIVERSAL, 1, 0, 1))
351                         return -1;
352                 break;

354                 case ASN1_GEN_FLAG_BITWRAP:
355                 if (!append_exp(arg, V_ASN1_BIT_STRING, V_ASN1_UNIVERSAL, 0, 1,
356                         return -1;
357                 break;

359                 case ASN1_GEN_FLAG_OCTWRAP:
360                 if (!append_exp(arg, V_ASN1_OCTET_STRING, V_ASN1_UNIVERSAL, 0, 0
361                         return -1;
362                 break;

364                 case ASN1_GEN_FLAG_FORMAT:
365                 if (!strncmp(vstart, "ASCII", 5))
366                         arg->format = ASN1_GEN_FORMAT_ASCII;
367                 else if (!strncmp(vstart, "UTF8", 4))
368                         arg->format = ASN1_GEN_FORMAT_UTF8;
369                 else if (!strncmp(vstart, "HEX", 3))
370                         arg->format = ASN1_GEN_FORMAT_HEX;
371                 else if (!strncmp(vstart, "BITLIST", 3))
372                         arg->format = ASN1_GEN_FORMAT_BITLIST;
373                 else
374                         {
375                         ASN1err(ASN1_F_ASN1_CB, ASN1_R_UNKOWN_FORMAT);
376                         return -1;
377                         }
378                 break;

380                 }

382         return 1;

384         }

386 static int parse_tagging(const char *vstart, int vlen, int *ptag, int *pclass)
387         {
388         char erch[2];
389         long tag_num;
390         char *eptr;
391         if (!vstart)
```

```
392                 return 0;
393         tag_num = strtoul(vstart, &eptr, 10);
394         /* Check we haven't gone past max length: should be impossible */
395         if (eptr && *eptr && (eptr > vstart + vlen))
396                 return 0;
397         if (tag_num < 0)
398                 {
399                 ASN1err(ASN1_F_PARSE_TAGGING, ASN1_R_INVALID_NUMBER);
400                 return 0;
401                 }
402         *ptag = tag_num;
403         /* If we have non numeric characters, parse them */
404         if (eptr)
405                 vlen -= eptr - vstart;
406         else
407                 vlen = 0;
408         if (vlen)
409                 {
410                 switch (*eptr)
411                         {

413                         case 'U':
414                         *pclass = V_ASN1_UNIVERSAL;
415                         break;

417                         case 'A':
418                         *pclass = V_ASN1_APPLICATION;
419                         break;

421                         case 'P':
422                         *pclass = V_ASN1_PRIVATE;
423                         break;

425                         case 'C':
426                         *pclass = V_ASN1_CONTEXT_SPECIFIC;
427                         break;

429                         default:
430                         erch[0] = *eptr;
431                         erch[1] = 0;
432                         ASN1err(ASN1_F_PARSE_TAGGING, ASN1_R_INVALID_MODIFIER);
433                         ERR_add_error_data(2, "Char=", erch);
434                         return 0;
435                         break;

437                         }
438                 }
439         else
440                 *pclass = V_ASN1_CONTEXT_SPECIFIC;

442         return 1;

444         }

446 /* Handle multiple types: SET and SEQUENCE */

448 static ASN1_TYPE *asn1_multi(int utype, const char *section, X509V3_CTX *cnf)
449         {
450         ASN1_TYPE *ret = NULL;
451         STACK_OF(ASN1_TYPE) *sk = NULL;
452         STACK_OF(CONF_VALUE) *sect = NULL;
453         unsigned char *der = NULL;
454         int derlen;
455         int i;
456         sk = sk_ASN1_TYPE_new_null();
457         if (!sk)
```



```
458                 goto bad;
459         if (section)
460                 {
461                 if (!cnf)
462                         goto bad;
463                 sect = X509V3_get_section(cnf, (char *)section);
464                 if (!sect)
465                         goto bad;
466                 for (i = 0; i < sk_CONF_VALUE_num(sect); i++)
467                         {
468                         ASN1_TYPE *typ = ASN1_generate_v3(sk_CONF_VALUE_value(se
469                         if (!typ)
470                                 goto bad;
471                         if (!sk_ASN1_TYPE_push(sk, typ))
472                                 goto bad;
473                         }
474                 }

476         /* Now we has a STACK of the components, convert to the correct form */

478         if (utype == V_ASN1_SET)
479                 derlen = i2d_ASN1_SET_ANY(sk, &der);
480         else
481                 derlen = i2d_ASN1_SEQUENCE_ANY(sk, &der);

483         if (derlen < 0)
484                 goto bad;

486         if (!(ret = ASN1_TYPE_new()))
487                 goto bad;

489         if (!(ret->value.asn1_string = ASN1_STRING_type_new(utype)))
490                 goto bad;

492         ret->type = utype;

494         ret->value.asn1_string->data = der;
495         ret->value.asn1_string->length = derlen;

497         der = NULL;

499         bad:

501         if (der)
502                 OPENSSL_free(der);

504         if (sk)
505                 sk_ASN1_TYPE_pop_free(sk, ASN1_TYPE_free);
506         if (sect)
507                 X509V3_section_free(cnf, sect);

509         return ret;
510         }

512 static int append_exp(tag_exp_arg *arg, int exp_tag, int exp_class, int exp_cons
513         {
514         tag_exp_type *exp_tmp;
515         /* Can only have IMPLICIT if permitted */
516         if ((arg->imp_tag != -1) && !imp_ok)
517                 {
518                 ASN1err(ASN1_F_APPEND_EXP, ASN1_R_ILLEGAL_IMPLICIT_TAG);
519                 return 0;
520                 }

522         if (arg->exp_count == ASN1_FLAG_EXP_MAX)
523                 {
```

```
524                 ASN1err(ASN1_F_APPEND_EXP, ASN1_R_DEPTH_EXCEEDED);
525                 return 0;
526                 }

528         exp_tmp = &arg->exp_list[arg->exp_count++];

530         /* If IMPLICIT set tag to implicit value then
531          * reset implicit tag since it has been used.
532          */
533         if (arg->imp_tag != -1)
534                 {
535                 exp_tmp->exp_tag = arg->imp_tag;
536                 exp_tmp->exp_class = arg->imp_class;
537                 arg->imp_tag = -1;
538                 arg->imp_class = -1;
539                 }
540         else
541                 {
542                 exp_tmp->exp_tag = exp_tag;
543                 exp_tmp->exp_class = exp_class;
544                 }
545         exp_tmp->exp_constructed = exp_constructed;
546         exp_tmp->exp_pad = exp_pad;

548         return 1;
549         }


552 static int asn1_str2tag(const char *tagstr, int len)
553         {
554         unsigned int i;
555         static const struct tag_name_st *tntmp, tnst [] = {
556                 ASN1_GEN_STR("BOOL", V_ASN1_BOOLEAN),
557                 ASN1_GEN_STR("BOOLEAN", V_ASN1_BOOLEAN),
558                 ASN1_GEN_STR("NULL", V_ASN1_NULL),
559                 ASN1_GEN_STR("INT", V_ASN1_INTEGER),
560                 ASN1_GEN_STR("INTEGER", V_ASN1_INTEGER),
561                 ASN1_GEN_STR("ENUM", V_ASN1_ENUMERATED),
562                 ASN1_GEN_STR("ENUMERATED", V_ASN1_ENUMERATED),
563                 ASN1_GEN_STR("OID", V_ASN1_OBJECT),
564                 ASN1_GEN_STR("OBJECT", V_ASN1_OBJECT),
565                 ASN1_GEN_STR("UTCTIME", V_ASN1_UTCTIME),
566                 ASN1_GEN_STR("UTC", V_ASN1_UTCTIME),
567                 ASN1_GEN_STR("GENERALIZEDTIME", V_ASN1_GENERALIZEDTIME),
568                 ASN1_GEN_STR("GENTIME", V_ASN1_GENERALIZEDTIME),
569                 ASN1_GEN_STR("OCT", V_ASN1_OCTET_STRING),
570                 ASN1_GEN_STR("OCTETSTRING", V_ASN1_OCTET_STRING),
571                 ASN1_GEN_STR("BITSTR", V_ASN1_BIT_STRING),
572                 ASN1_GEN_STR("BITSTRING", V_ASN1_BIT_STRING),
573                 ASN1_GEN_STR("UNIVERSALSTRING", V_ASN1_UNIVERSALSTRING),
574                 ASN1_GEN_STR("UNIV", V_ASN1_UNIVERSALSTRING),
575                 ASN1_GEN_STR("IA5", V_ASN1_IA5STRING),
576                 ASN1_GEN_STR("IA5STRING", V_ASN1_IA5STRING),
577                 ASN1_GEN_STR("UTF8", V_ASN1_UTF8STRING),
578                 ASN1_GEN_STR("UTF8String", V_ASN1_UTF8STRING),
579                 ASN1_GEN_STR("BMP", V_ASN1_BMPSTRING),
580                 ASN1_GEN_STR("BMPSTRING", V_ASN1_BMPSTRING),
581                 ASN1_GEN_STR("VISIBLESTRING", V_ASN1_VISIBLESTRING),
582                 ASN1_GEN_STR("VISIBLE", V_ASN1_VISIBLESTRING),
583                 ASN1_GEN_STR("PRINTABLESTRING", V_ASN1_PRINTABLESTRING),
584                 ASN1_GEN_STR("PRINTABLE", V_ASN1_PRINTABLESTRING),
585                 ASN1_GEN_STR("T61", V_ASN1_T61STRING),
586                 ASN1_GEN_STR("T61STRING", V_ASN1_T61STRING),
587                 ASN1_GEN_STR("TELETEXSTRING", V_ASN1_T61STRING),
588                 ASN1_GEN_STR("GeneralString", V_ASN1_GENERALSTRING),
589                 ASN1_GEN_STR("GENSTR", V_ASN1_GENERALSTRING),
```



```
590                 ASN1_GEN_STR("NUMERIC", V_ASN1_NUMERICSTRING),
591                 ASN1_GEN_STR("NUMERICSTRING", V_ASN1_NUMERICSTRING),

593                 /* Special cases */
594                 ASN1_GEN_STR("SEQUENCE", V_ASN1_SEQUENCE),
595                 ASN1_GEN_STR("SEQ", V_ASN1_SEQUENCE),
596                 ASN1_GEN_STR("SET", V_ASN1_SET),
597                 /* type modifiers */
598                 /* Explicit tag */
599                 ASN1_GEN_STR("EXP", ASN1_GEN_FLAG_EXP),
600                 ASN1_GEN_STR("EXPLICIT", ASN1_GEN_FLAG_EXP),
601                 /* Implicit tag */
602                 ASN1_GEN_STR("IMP", ASN1_GEN_FLAG_IMP),
603                 ASN1_GEN_STR("IMPLICIT", ASN1_GEN_FLAG_IMP),
604                 /* OCTET STRING wrapper */
605                 ASN1_GEN_STR("OCTWRAP", ASN1_GEN_FLAG_OCTWRAP),
606                 /* SEQUENCE wrapper */
607                 ASN1_GEN_STR("SEQWRAP", ASN1_GEN_FLAG_SEQWRAP),
608                 /* SET wrapper */
609                 ASN1_GEN_STR("SETWRAP", ASN1_GEN_FLAG_SETWRAP),
610                 /* BIT STRING wrapper */
611                 ASN1_GEN_STR("BITWRAP", ASN1_GEN_FLAG_BITWRAP),
612                 ASN1_GEN_STR("FORM", ASN1_GEN_FLAG_FORMAT),
613                 ASN1_GEN_STR("FORMAT", ASN1_GEN_FLAG_FORMAT),
614         };

616         if (len == -1)
617                 len = strlen(tagstr);

619         tntmp = tnst;
620         for (i = 0; i < sizeof(tnst) / sizeof(struct tag_name_st); i++, tntmp++)
621                 {
622                 if ((len == tntmp->len) && !strncmp(tntmp->strnam, tagstr, len))
623                         return tntmp->tag;
624                 }

626         return -1;
627         }

629 static ASN1_TYPE *asn1_str2type(const char *str, int format, int utype)
630         {
631         ASN1_TYPE *atmp = NULL;

633         CONF_VALUE vtmp;

635         unsigned char *rdata;
636         long rdlen;

638         int no_unused = 1;

640         if (!(atmp = ASN1_TYPE_new()))
641                 {
642                 ASN1err(ASN1_F_ASN1_STR2TYPE, ERR_R_MALLOC_FAILURE);
643                 return NULL;
644                 }

646         if (!str)
647                 str = "";

649         switch(utype)
650                 {

652                 case V_ASN1_NULL:
653                 if (str && *str)
654                         {
655                         ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_NULL_VALUE)
```

```
656                     goto bad_form;
657                     }
658             break;

660             case V_ASN1_BOOLEAN:
661             if (format != ASN1_GEN_FORMAT_ASCII)
662                     {
663                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_NOT_ASCII_FORMAT);
664                     goto bad_form;
665                     }
666             vtmp.name = NULL;
667             vtmp.section = NULL;
668             vtmp.value = (char *)str;
669             if (!X509V3_get_value_bool(&vtmp, &atmp->value.boolean))
670                     {
671                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_BOOLEAN);
672                     goto bad_str;
673                     }
674             break;

676             case V_ASN1_INTEGER:
677             case V_ASN1_ENUMERATED:
678             if (format != ASN1_GEN_FORMAT_ASCII)
679                     {
680                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_INTEGER_NOT_ASCII_F
681                     goto bad_form;
682                     }
683             if (!(atmp->value.integer = s2i_ASN1_INTEGER(NULL, (char *)str))
684                     {
685                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_INTEGER);
686                     goto bad_str;
687                     }
688             break;

690             case V_ASN1_OBJECT:
691             if (format != ASN1_GEN_FORMAT_ASCII)
692                     {
693                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_OBJECT_NOT_ASCII_FO
694                     goto bad_form;
695                     }
696             if (!(atmp->value.object = OBJ_txt2obj(str, 0)))
697                     {
698                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_OBJECT);
699                     goto bad_str;
700                     }
701             break;

703             case V_ASN1_UTCTIME:
704             case V_ASN1_GENERALIZEDTIME:
705             if (format != ASN1_GEN_FORMAT_ASCII)
706                     {
707                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_TIME_NOT_ASCII_FORM
708                     goto bad_form;
709                     }
710             if (!(atmp->value.asn1_string = ASN1_STRING_new()))
711                     {
712                     ASN1err(ASN1_F_ASN1_STR2TYPE, ERR_R_MALLOC_FAILURE);
713                     goto bad_str;
714                     }
715             if (!ASN1_STRING_set(atmp->value.asn1_string, str, -1))
716                     {
717                     ASN1err(ASN1_F_ASN1_STR2TYPE, ERR_R_MALLOC_FAILURE);
718                     goto bad_str;
719                     }
720             atmp->value.asn1_string->type = utype;
721             if (!ASN1_TIME_check(atmp->value.asn1_string))
```



```
722                     {
723                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_TIME_VALUE)
724                     goto bad_str;
725                     }

727             break;

729             case V_ASN1_BMPSTRING:
730             case V_ASN1_PRINTABLESTRING:
731             case V_ASN1_IA5STRING:
732             case V_ASN1_T61STRING:
733             case V_ASN1_UTF8STRING:
734             case V_ASN1_VISIBLESTRING:
735             case V_ASN1_UNIVERSALSTRING:
736             case V_ASN1_GENERALSTRING:
737             case V_ASN1_NUMERICSTRING:

739             if (format == ASN1_GEN_FORMAT_ASCII)
740                     format = MBSTRING_ASC;
741             else if (format == ASN1_GEN_FORMAT_UTF8)
742                     format = MBSTRING_UTF8;
743             else
744                     {
745                     ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_FORMAT);
746                     goto bad_form;
747                     }


750             if (ASN1_mbstring_copy(&atmp->value.asn1_string, (unsigned char
751                                             -1, format, ASN1_tag2bit(utype))
752                     {
753                     ASN1err(ASN1_F_ASN1_STR2TYPE, ERR_R_MALLOC_FAILURE);
754                     goto bad_str;
755                     }


758             break;

760             case V_ASN1_BIT_STRING:

762             case V_ASN1_OCTET_STRING:

764             if (!(atmp->value.asn1_string = ASN1_STRING_new()))
765                     {
766                     ASN1err(ASN1_F_ASN1_STR2TYPE, ERR_R_MALLOC_FAILURE);
767                     goto bad_form;
768                     }

770             if (format == ASN1_GEN_FORMAT_HEX)
771                     {

773                     if (!(rdata = string_to_hex((char *)str, &rdlen)))
774                             {
775                             ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_HEX
776                             goto bad_str;
777                             }

779                     atmp->value.asn1_string->data = rdata;
780                     atmp->value.asn1_string->length = rdlen;
781                     atmp->value.asn1_string->type = utype;

783                     }
784             else if (format == ASN1_GEN_FORMAT_ASCII)
785                     ASN1_STRING_set(atmp->value.asn1_string, str, -1);
786             else if ((format == ASN1_GEN_FORMAT_BITLIST) && (utype == V_ASN1
787                     {
```

```
788                         if (!CONF_parse_list(str, ',', 1, bitstr_cb, atmp->value
789                                 {
790                                 ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_LIST_ERROR)
791                                 goto bad_str;
792                                 }
793                         no_unused = 0;

795                         }
796                 else
797                         {
798                         ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_ILLEGAL_BITSTRING_F
799                         goto bad_form;
800                         }

802                 if ((utype == V_ASN1_BIT_STRING) && no_unused)
803                         {
804                         atmp->value.asn1_string->flags
805                                 &= ~(ASN1_STRING_FLAG_BITS_LEFT|0x07);
806                         atmp->value.asn1_string->flags
807                                 |= ASN1_STRING_FLAG_BITS_LEFT;
808                         }


811                 break;

813                 default:
814                 ASN1err(ASN1_F_ASN1_STR2TYPE, ASN1_R_UNSUPPORTED_TYPE);
815                 goto bad_str;
816                 break;
817                 }


820         atmp->type = utype;
821         return atmp;


824         bad_str:
825         ERR_add_error_data(2, "string=", str);
826         bad_form:

828         ASN1_TYPE_free(atmp);
829         return NULL;

831         }

833 static int bitstr_cb(const char *elem, int len, void *bitstr)
834         {
835         long bitnum;
836         char *eptr;
837         if (!elem)
838                 return 0;
839         bitnum = strtoul(elem, &eptr, 10);
840         if (eptr && *eptr && (eptr != elem + len))
841                 return 0;
842         if (bitnum < 0)
843                 {
844                 ASN1err(ASN1_F_BITSTR_CB, ASN1_R_INVALID_NUMBER);
845                 return 0;
846                 }
847         if (!ASN1_BIT_STRING_set_bit(bitstr, bitnum, 1))
848                 {
849                 ASN1err(ASN1_F_BITSTR_CB, ERR_R_MALLOC_FAILURE);
850                 return 0;
851                 }
852         return 1;
853         }
```



```
854 #endif /* ! codereview */
```

```
**********************************************************
   11077 Wed Aug 13 19:52:01 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/asn1_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/asn1_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <limits.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/asn1_mac.h>

static int asn1_get_length(const unsigned char **pp,int *inf,long *rl,int max);
static void asn1_put_length(unsigned char **pp, int length);
const char ASN1_version[]="ASN.1" OPENSSL_VERSION_PTEXT;

static int _asn1_check_infinite_end(const unsigned char **p, long len)
        {
        /* If there is 0 or 1 byte left, the length check should pick
         * things up */
        if (len <= 0)
                return(1);
        else if ((len >= 2) && ((*p)[0] == 0) && ((*p)[1] == 0))
                {
                (*p)+=2;
                return(1);
                }
        return(0);
        }

int ASN1_check_infinite_end(unsigned char **p, long len)
        {
        return _asn1_check_infinite_end((const unsigned char **)p, len);
        }

int ASN1_const_check_infinite_end(const unsigned char **p, long len)
        {
        return _asn1_check_infinite_end(p, len);
        }


int ASN1_get_object(const unsigned char **pp, long *plength, int *ptag,
        int *pclass, long omax)
        {
        int i,ret;
        long l;
        const unsigned char *p= *pp;
        int tag,xclass,inf;
        long max=omax;

        if (!max) goto err;
        ret=(*p&V_ASN1_CONSTRUCTED);
        xclass=(*p&V_ASN1_PRIVATE);
        i= *p&V_ASN1_PRIMITIVE_TAG;
        if (i == V_ASN1_PRIMITIVE_TAG)
                {               /* high-tag */
                p++;
                if (--max == 0) goto err;
                l=0;
                while (*p&0x80)
                        {
                        l<<=7L;
                        l|= *(p++)&0x7f;
                        if (--max == 0) goto err;
                        if (l > (INT_MAX >> 7L)) goto err;
                        }
                l<<=7L;
                l|= *(p++)&0x7f;
                tag=(int)l;
                if (--max == 0) goto err;
                }
        else
                {
                tag=i;
                p++;
```

```
128                 if (--max == 0) goto err;
129                 }
130         *ptag=tag;
131         *pclass=xclass;
132         if (!asn1_get_length(&p,&inf,plength,(int)max)) goto err;
133
134         if (inf && !(ret & V_ASN1_CONSTRUCTED))
135                 goto err;
136
137 #if 0
138         fprintf(stderr,"p=%d + *plength=%ld > omax=%ld + *pp=%d  (%d > %d)\n",
139                 (int)p,*plength,omax,(int)*pp,(int)(p+ *plength),
140                 (int)(omax+ *pp));
141
142 #endif
143         if (*plength > (omax - (p - *pp)))
144                 {
145                 ASN1err(ASN1_F_ASN1_GET_OBJECT,ASN1_R_TOO_LONG);
146                 /* Set this so that even if things are not long enough
147                  * the values are set correctly */
148                 ret|=0x80;
149                 }
150         *pp=p;
151         return(ret|inf);
152 err:
153         ASN1err(ASN1_F_ASN1_GET_OBJECT,ASN1_R_HEADER_TOO_LONG);
154         return(0x80);
155         }
156
157 static int asn1_get_length(const unsigned char **pp, int *inf, long *rl, int max
158         {
159         const unsigned char *p= *pp;
160         unsigned long ret=0;
161         unsigned int i;
162
163         if (max-- < 1) return(0);
164         if (*p == 0x80)
165                 {
166                 *inf=1;
167                 ret=0;
168                 p++;
169                 }
170         else
171                 {
172                 *inf=0;
173                 i= *p&0x7f;
174                 if (*(p++) & 0x80)
175                         {
176                         if (i > sizeof(long))
177                                 return 0;
178                         if (max-- == 0) return(0);
179                         while (i-- > 0)
180                                 {
181                                 ret<<=8L;
182                                 ret|= *(p++);
183                                 if (max-- == 0) return(0);
184                                 }
185                         }
186                 else
187                         ret=i;
188                 }
189         if (ret > LONG_MAX)
190                 return 0;
191         *pp=p;
192         *rl=(long)ret;
193         return(1);
```



```
194         }
195
196 /* class 0 is constructed
197  * constructed == 2 for indefinite length constructed */
198 void ASN1_put_object(unsigned char **pp, int constructed, int length, int tag,
199                      int xclass)
200         {
201         unsigned char *p= *pp;
202         int i, ttag;
203
204         i=(constructed)?V_ASN1_CONSTRUCTED:0;
205         i|=(xclass&V_ASN1_PRIVATE);
206         if (tag < 31)
207                 *(p++)=i|(tag&V_ASN1_PRIMITIVE_TAG);
208         else
209                 {
210                 *(p++)=i|V_ASN1_PRIMITIVE_TAG;
211                 for(i = 0, ttag = tag; ttag > 0; i++) ttag >>=7;
212                 ttag = i;
213                 while(i-- > 0)
214                         {
215                         p[i] = tag & 0x7f;
216                         if(i != (ttag - 1)) p[i] |= 0x80;
217                         tag >>= 7;
218                         }
219                 p += ttag;
220                 }
221         if (constructed == 2)
222                 *(p++)=0x80;
223         else
224                 asn1_put_length(&p,length);
225         *pp=p;
226         }
227
228 int ASN1_put_eoc(unsigned char **pp)
229         {
230         unsigned char *p = *pp;
231         *p++ = 0;
232         *p++ = 0;
233         *pp = p;
234         return 2;
235         }
236
237 static void asn1_put_length(unsigned char **pp, int length)
238         {
239         unsigned char *p= *pp;
240         int i,l;
241         if (length <= 127)
242                 *(p++)=(unsigned char)length;
243         else
244                 {
245                 l=length;
246                 for (i=0; l > 0; i++)
247                         l>>=8;
248                 *(p++)=i|0x80;
249                 l=i;
250                 while (i-- > 0)
251                         {
252                         p[i]=length&0xff;
253                         length>>=8;
254                         }
255                 p+=l;
256                 }
257         *pp=p;
258         }
```

```
260 int ASN1_object_size(int constructed, int length, int tag)
261         {
262         int ret;

264         ret=length;
265         ret++;
266         if (tag >= 31)
267                 {
268                 while (tag > 0)
269                         {
270                         tag>>=7;
271                         ret++;
272                         }
273                 }
274         if (constructed == 2)
275                 return ret + 3;
276         ret++;
277         if (length > 127)
278                 {
279                 while (length > 0)
280                         {
281                         length>>=8;
282                         ret++;
283                         }
284                 }
285         return(ret);
286         }

288 static int _asn1_Finish(ASN1_const_CTX *c)
289         {
290         if ((c->inf == (1|V_ASN1_CONSTRUCTED)) && (!c->eos))
291                 {
292                 if (!ASN1_const_check_infinite_end(&c->p,c->slen))
293                         {
294                         c->error=ERR_R_MISSING_ASN1_EOS;
295                         return(0);
296                         }
297                 }
298         if (    ((c->slen != 0) && !(c->inf & 1)) ||
299                 ((c->slen < 0) && (c->inf & 1)))
300                 {
301                 c->error=ERR_R_ASN1_LENGTH_MISMATCH;
302                 return(0);
303                 }
304         return(1);
305         }

307 int asn1_Finish(ASN1_CTX *c)
308         {
309         return _asn1_Finish((ASN1_const_CTX *)c);
310         }

312 int asn1_const_Finish(ASN1_const_CTX *c)
313         {
314         return _asn1_Finish(c);
315         }

317 int asn1_GetSequence(ASN1_const_CTX *c, long *length)
318         {
319         const unsigned char *q;

321         q=c->p;
322         c->inf=ASN1_get_object(&(c->p),&(c->slen),&(c->tag),&(c->xclass),
323                 *length);
324         if (c->inf & 0x80)
325                 {
```



```
326                 c->error=ERR_R_BAD_GET_ASN1_OBJECT_CALL;
327                 return(0);
328                 }
329         if (c->tag != V_ASN1_SEQUENCE)
330                 {
331                 c->error=ERR_R_EXPECTING_AN_ASN1_SEQUENCE;
332                 return(0);
333                 }
334         (*length)-=(c->p-q);
335         if (c->max && (*length < 0))
336                 {
337                 c->error=ERR_R_ASN1_LENGTH_MISMATCH;
338                 return(0);
339                 }
340         if (c->inf == (1|V_ASN1_CONSTRUCTED))
341                 c->slen= *length+ *(c->pp)-c->p;
342         c->eos=0;
343         return(1);
344         }

346 int ASN1_STRING_copy(ASN1_STRING *dst, const ASN1_STRING *str)
347         {
348         if (str == NULL)
349                 return 0;
350         dst->type = str->type;
351         if (!ASN1_STRING_set(dst,str->data,str->length))
352                 return 0;
353         dst->flags = str->flags;
354         return 1;
355         }

357 ASN1_STRING *ASN1_STRING_dup(const ASN1_STRING *str)
358         {
359         ASN1_STRING *ret;
360         if (!str)
361                 return NULL;
362         ret=ASN1_STRING_new();
363         if (!ret)
364                 return NULL;
365         if (!ASN1_STRING_copy(ret,str))
366                 {
367                 ASN1_STRING_free(ret);
368                 return NULL;
369                 }
370         return ret;
371         }

373 int ASN1_STRING_set(ASN1_STRING *str, const void *_data, int len)
374         {
375         unsigned char *c;
376         const char *data=_data;

378         if (len < 0)
379                 {
380                 if (data == NULL)
381                         return(0);
382                 else
383                         len=strlen(data);
384                 }
385         if ((str->length < len) || (str->data == NULL))
386                 {
387                 c=str->data;
388                 if (c == NULL)
389                         str->data=OPENSSL_malloc(len+1);
390                 else
391                         str->data=OPENSSL_realloc(c,len+1);
```

```
393                 if (str->data == NULL)
394                         {
395                         ASN1err(ASN1_F_ASN1_STRING_SET,ERR_R_MALLOC_FAILURE);
396                         str->data=c;
397                         return(0);
398                         }
399                 }
400         str->length=len;
401         if (data != NULL)
402                 {
403                 memcpy(str->data,data,len);
404                 /* an allowance for strings :-) */
405                 str->data[len]='\0';
406                 }
407         return(1);
408         }

410 void ASN1_STRING_set0(ASN1_STRING *str, void *data, int len)
411         {
412         if (str->data)
413                 OPENSSL_free(str->data);
414         str->data = data;
415         str->length = len;
416         }

418 ASN1_STRING *ASN1_STRING_new(void)
419         {
420         return(ASN1_STRING_type_new(V_ASN1_OCTET_STRING));
421         }


424 ASN1_STRING *ASN1_STRING_type_new(int type)
425         {
426         ASN1_STRING *ret;

428         ret=(ASN1_STRING *)OPENSSL_malloc(sizeof(ASN1_STRING));
429         if (ret == NULL)
430                 {
431                 ASN1err(ASN1_F_ASN1_STRING_TYPE_NEW,ERR_R_MALLOC_FAILURE);
432                 return(NULL);
433                 }
434         ret->length=0;
435         ret->type=type;
436         ret->data=NULL;
437         ret->flags=0;
438         return(ret);
439         }

441 void ASN1_STRING_free(ASN1_STRING *a)
442         {
443         if (a == NULL) return;
444         if (a->data && !(a->flags & ASN1_STRING_FLAG_NDEF))
445                 OPENSSL_free(a->data);
446         OPENSSL_free(a);
447         }

449 int ASN1_STRING_cmp(const ASN1_STRING *a, const ASN1_STRING *b)
450         {
451         int i;

453         i=(a->length-b->length);
454         if (i == 0)
455                 {
456                 i=memcmp(a->data,b->data,a->length);
457                 if (i == 0)
```



```
458                         return(a->type-b->type);
459                 else
460                         return(i);
461                 }
462         else
463                 return(i);
464         }

466 void asn1_add_error(const unsigned char *address, int offset)
467         {
468         char buf1[DECIMAL_SIZE(address)+1],buf2[DECIMAL_SIZE(offset)+1];

470         BIO_snprintf(buf1,sizeof buf1,"%lu",(unsigned long)address);
471         BIO_snprintf(buf2,sizeof buf2,"%d",offset);
472         ERR_add_error_data(4,"address=",buf1," offset=",buf2);
473         }

475 int ASN1_STRING_length(const ASN1_STRING *x)
476 { return M_ASN1_STRING_length(x); }

478 void ASN1_STRING_length_set(ASN1_STRING *x, int len)
479 { M_ASN1_STRING_length_set(x, len); return; }

481 int ASN1_STRING_type(ASN1_STRING *x)
482 { return M_ASN1_STRING_type(x); }

484 unsigned char * ASN1_STRING_data(ASN1_STRING *x)
485 { return M_ASN1_STRING_data(x); }
486 #endif /* ! codereview */
```

```
**********************************************************
   11655 Wed Aug 13 19:52:01 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/asn1_par.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/asn1_par.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/objects.h>
#include <openssl/asn1.h>

static int asn1_print_info(BIO *bp, int tag, int xclass,int constructed,
        int indent);
static int asn1_parse2(BIO *bp, const unsigned char **pp, long length,
        int offset, int depth, int indent, int dump);
static int asn1_print_info(BIO *bp, int tag, int xclass, int constructed,
             int indent)
        {
        static const char fmt[]="%-18s";
        char str[128];
        const char *p;

        if (constructed & V_ASN1_CONSTRUCTED)
                p="cons: ";
        else
                p="prim: ";
        if (BIO_write(bp,p,6) < 6) goto err;
        BIO_indent(bp,indent,128);

        p=str;
        if ((xclass & V_ASN1_PRIVATE) == V_ASN1_PRIVATE)
                BIO_snprintf(str,sizeof str,"priv [ %d ] ",tag);
        else if ((xclass & V_ASN1_CONTEXT_SPECIFIC) == V_ASN1_CONTEXT_SPECIFIC)
                BIO_snprintf(str,sizeof str,"cont [ %d ]",tag);
        else if ((xclass & V_ASN1_APPLICATION) == V_ASN1_APPLICATION)
                BIO_snprintf(str,sizeof str,"appl [ %d ]",tag);
        else if (tag > 30)
                BIO_snprintf(str,sizeof str,"<ASN1 %d>",tag);
        else
                p = ASN1_tag2str(tag);

        if (BIO_printf(bp,fmt,p) <= 0)
                goto err;
        return(1);
err:
        return(0);
        }

int ASN1_parse(BIO *bp, const unsigned char *pp, long len, int indent)
        {
        return(asn1_parse2(bp,&pp,len,0,0,indent,0));
        }

int ASN1_parse_dump(BIO *bp, const unsigned char *pp, long len, int indent, int
        {
        return(asn1_parse2(bp,&pp,len,0,0,indent,dump));
        }

static int asn1_parse2(BIO *bp, const unsigned char **pp, long length, int offse
             int depth, int indent, int dump)
        {
        const unsigned char *p,*ep,*tot,*op,*opp;
        long len;
        int tag,xclass,ret=0;
        int nl,hl,j,r;
        ASN1_OBJECT *o=NULL;
        ASN1_OCTET_STRING *os=NULL;
        /* ASN1_BMPSTRING *bmp=NULL;*/
        int dump_indent;

#if 0
        dump_indent = indent;
#else
        dump_indent = 6;        /* Because we know BIO_dump_indent() */
```

```
#endif
        p= *pp;
        tot=p+length;
        op=p-1;
        while ((p < tot) && (op < p))
                {
                op=p;
                j=ASN1_get_object(&p,&len,&tag,&xclass,length);
#ifdef LINT
                j=j;
#endif
                if (j & 0x80)
                        {
                        if (BIO_write(bp,"Error in encoding\n",18) <= 0)
                                goto end;
                        ret=0;
                        goto end;
                        }
                hl=(p-op);
                length-=hl;
                /* if j == 0x21 it is a constructed indefinite length object */
                if (BIO_printf(bp,"%5ld:",(long)offset+(long)(op- *pp))
                        <= 0) goto end;

                if (j != (V_ASN1_CONSTRUCTED | 1))
                        {
                        if (BIO_printf(bp,"d=%-2d hl=%ld l=%4ld ",
                                depth,(long)hl,len) <= 0)
                                goto end;
                        }
                else
                        {
                        if (BIO_printf(bp,"d=%-2d hl=%ld l=inf  ",
                                depth,(long)hl) <= 0)
                                goto end;
                        }
                if (!asn1_print_info(bp,tag,xclass,j,(indent)?depth:0))
                        goto end;
                if (j & V_ASN1_CONSTRUCTED)
                        {
                        ep=p+len;
                        if (BIO_write(bp,"\n",1) <= 0) goto end;
                        if (len > length)
                                {
                                BIO_printf(bp,
                                        "length is greater than %ld\n",length);
                                ret=0;
                                goto end;
                                }
                        if ((j == 0x21) && (len == 0))
                                {
                                for (;;)
                                        {
                                        r=asn1_parse2(bp,&p,(long)(tot-p),
                                                offset+(p - *pp),depth+1,
                                                indent,dump);
                                        if (r == 0) { ret=0; goto end; }
                                        if ((r == 2) || (p >= tot)) break;
                                        }
                                }
                        else
                                while (p < ep)
                                        {
                                        r=asn1_parse2(bp,&p,(long)len,
                                                offset+(p - *pp),depth+1,
                                                indent,dump);
```



```
                                        if (r == 0) { ret=0; goto end; }
                                        }
                        }
                else if (xclass != 0)
                        {
                        p+=len;
                        if (BIO_write(bp,"\n",1) <= 0) goto end;
                        }
                else
                        {
                        nl=0;
                        if (    (tag == V_ASN1_PRINTABLESTRING) ||
                                (tag == V_ASN1_T61STRING) ||
                                (tag == V_ASN1_IA5STRING) ||
                                (tag == V_ASN1_VISIBLESTRING) ||
                                (tag == V_ASN1_NUMERICSTRING) ||
                                (tag == V_ASN1_UTF8STRING) ||
                                (tag == V_ASN1_UTCTIME) ||
                                (tag == V_ASN1_GENERALIZEDTIME))
                                {
                                if (BIO_write(bp,":",1) <= 0) goto end;
                                if ((len > 0) &&
                                        BIO_write(bp,(const char *)p,(int)len)
                                        != (int)len)
                                        goto end;
                                }
                        else if (tag == V_ASN1_OBJECT)
                                {
                                opp=op;
                                if (d2i_ASN1_OBJECT(&o,&opp,len+hl) != NULL)
                                        {
                                        if (BIO_write(bp,":",1) <= 0) goto end;
                                        i2a_ASN1_OBJECT(bp,o);
                                        }
                                else
                                        {
                                        if (BIO_write(bp,":BAD OBJECT",11) <= 0)
                                                goto end;
                                        }
                                }
                        else if (tag == V_ASN1_BOOLEAN)
                                {
                                int ii;

                                opp=op;
                                ii=d2i_ASN1_BOOLEAN(NULL,&opp,len+hl);
                                if (ii < 0)
                                        {
                                        if (BIO_write(bp,"Bad boolean\n",12) <=
                                                goto end;
                                        }
                                BIO_printf(bp,":%d",ii);
                                }
                        else if (tag == V_ASN1_BMPSTRING)
                                {
                                /* do the BMP thang */
                                }
                        else if (tag == V_ASN1_OCTET_STRING)
                                {
                                int i,printable=1;

                                opp=op;
                                os=d2i_ASN1_OCTET_STRING(NULL,&opp,len+hl);
                                if (os != NULL && os->length > 0)
                                        {
                                        opp = os->data;
```

```
260                                     /* testing whether the octet string is
261                                      * printable */
262                                     for (i=0; i<os->length; i++)
263                                             {
264                                             if ((   (opp[i] < ' ') &&
265                                                     (opp[i] != '\n') &&
266                                                     (opp[i] != '\r') &&
267                                                     (opp[i] != '\t')) ||
268                                                     (opp[i] > '~'))
269                                                     {
270                                                     printable=0;
271                                                     break;
272                                                     }
273                                             }
274                                     if (printable)
275                                     /* printable string */
276                                             {
277                                             if (BIO_write(bp,":",1) <= 0)
278                                                     goto end;
279                                             if (BIO_write(bp,(const char *)o
280                                                     os->length) <= 0)
281                                                     goto end;
282                                             }
283                                     else if (!dump)
284                                     /* not printable => print octet string
285                                      * as hex dump */
286                                             {
287                                             if (BIO_write(bp,"[HEX DUMP]:",1
288                                                     goto end;
289                                             for (i=0; i<os->length; i++)
290                                                     {
291                                                     if (BIO_printf(bp,"%02X"
292                                                             , opp[i]) <= 0)
293                                                             goto end;
294                                                     }
295                                             }
296                                     else
297                                     /* print the normal dump */
298                                             {
299                                             if (!nl)
300                                                     {
301                                                     if (BIO_write(bp,"\n",1)
302                                                             goto end;
303                                                     }
304                                             if (BIO_dump_indent(bp,
305                                                     (const char *)opp,
306                                                     ((dump == -1 || dump >
307                                                     os->length)?os->length:d
308                                                     dump_indent) <= 0)
309                                                     goto end;
310                                             nl=1;
311                                             }
312                                     }
313                             if (os != NULL)
314                                     {
315                                     M_ASN1_OCTET_STRING_free(os);
316                                     os=NULL;
317                                     }
318                             }
319                     else if (tag == V_ASN1_INTEGER)
320                             {
321                             ASN1_INTEGER *bs;
322                             int i;

324                             opp=op;
325                             bs=d2i_ASN1_INTEGER(NULL,&opp,len+hl);
```



```
326                             if (bs != NULL)
327                                     {
328                                     if (BIO_write(bp,":",1) <= 0) goto end;
329                                     if (bs->type == V_ASN1_NEG_INTEGER)
330                                             if (BIO_write(bp,"-",1) <= 0)
331                                                     goto end;
332                                     for (i=0; i<bs->length; i++)
333                                             {
334                                             if (BIO_printf(bp,"%02X",
335                                                     bs->data[i]) <= 0)
336                                                     goto end;
337                                             }
338                                     if (bs->length == 0)
339                                             {
340                                             if (BIO_write(bp,"00",2) <= 0)
341                                                     goto end;
342                                             }
343                                     }
344                             else
345                                     {
346                                     if (BIO_write(bp,"BAD INTEGER",11) <= 0)
347                                             goto end;
348                                     }
349                             M_ASN1_INTEGER_free(bs);
350                             }
351                     else if (tag == V_ASN1_ENUMERATED)
352                             {
353                             ASN1_ENUMERATED *bs;
354                             int i;

356                             opp=op;
357                             bs=d2i_ASN1_ENUMERATED(NULL,&opp,len+hl);
358                             if (bs != NULL)
359                                     {
360                                     if (BIO_write(bp,":",1) <= 0) goto end;
361                                     if (bs->type == V_ASN1_NEG_ENUMERATED)
362                                             if (BIO_write(bp,"-",1) <= 0)
363                                                     goto end;
364                                     for (i=0; i<bs->length; i++)
365                                             {
366                                             if (BIO_printf(bp,"%02X",
367                                                     bs->data[i]) <= 0)
368                                                     goto end;
369                                             }
370                                     if (bs->length == 0)
371                                             {
372                                             if (BIO_write(bp,"00",2) <= 0)
373                                                     goto end;
374                                             }
375                                     }
376                             else
377                                     {
378                                     if (BIO_write(bp,"BAD ENUMERATED",11) <=
379                                             goto end;
380                                     }
381                             M_ASN1_ENUMERATED_free(bs);
382                             }
383                     else if (len > 0 && dump)
384                             {
385                             if (!nl)
386                                     {
387                                     if (BIO_write(bp,"\n",1) <= 0)
388                                             goto end;
389                                     }
390                             if (BIO_dump_indent(bp,(const char *)p,
391                                     ((dump == -1 || dump > len)?len:dump),
```

```
392                                         dump_indent) <= 0)
393                                         goto end;
394                                 nl=1;
395                                 }
396
397                         if (!nl)
398                                 {
399                                 if (BIO_write(bp,"\n",1) <= 0) goto end;
400                                 }
401                         p+=len;
402                         if ((tag == V_ASN1_EOC) && (xclass == 0))
403                                 {
404                                 ret=2; /* End of sequence */
405                                 goto end;
406                                 }
407                         }
408                 length-=len;
409                 }
410         ret=1;
411 end:
412         if (o != NULL) ASN1_OBJECT_free(o);
413         if (os != NULL) M_ASN1_OCTET_STRING_free(os);
414         *pp=p;
415         return(ret);
416         }
417
418 const char *ASN1_tag2str(int tag)
419 {
420         static const char * const tag2str[] = {
421          "EOC", "BOOLEAN", "INTEGER", "BIT STRING", "OCTET STRING", /* 0-4 */
422          "NULL", "OBJECT", "OBJECT DESCRIPTOR", "EXTERNAL", "REAL", /* 5-9 */
423          "ENUMERATED", "<ASN1 11>", "UTF8STRING", "<ASN1 13>",      /* 10-13 */
424         "<ASN1 14>", "<ASN1 15>", "SEQUENCE", "SET",                /* 15-17 */
425         "NUMERICSTRING", "PRINTABLESTRING", "T61STRING",            /* 18-20 */
426         "VIDEOTEXSTRING", "IA5STRING", "UTCTIME","GENERALIZEDTIME", /* 21-24 */
427         "GRAPHICSTRING", "VISIBLESTRING", "GENERALSTRING",          /* 25-27 */
428         "UNIVERSALSTRING", "<ASN1 29>", "BMPSTRING"                 /* 28-30 */
429         };
430
431         if((tag == V_ASN1_NEG_INTEGER) || (tag == V_ASN1_NEG_ENUMERATED))
432                                                 tag &= ~0x100;
433
434         if(tag < 0 || tag > 30) return "(unknown)";
435         return tag2str[tag];
436 }
437 #endif /* ! codereview */
```

```
**********************************************************
   24670 Wed Aug 13 19:52:01 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/asn_mime.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* asn_mime.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 1999-2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
```



```
#include "asn1_locl.h"

/* Generalised MIME like utilities for streaming ASN1. Although many
 * have a PKCS7/CMS like flavour others are more general purpose.
 */

/* MIME format structures
 * Note that all are translated to lower case apart from
 * parameter values. Quotes are stripped off
 */

typedef struct {
char *param_name;                       /* Param name e.g. "micalg" */
char *param_value;                      /* Param value e.g. "sha1" */
} MIME_PARAM;

DECLARE_STACK_OF(MIME_PARAM)
IMPLEMENT_STACK_OF(MIME_PARAM)

typedef struct {
char *name;                             /* Name of line e.g. "content-type" */
char *value;                            /* Value of line e.g. "text/plain" */
STACK_OF(MIME_PARAM) *params;           /* Zero or more parameters */
} MIME_HEADER;

DECLARE_STACK_OF(MIME_HEADER)
IMPLEMENT_STACK_OF(MIME_HEADER)

static int asn1_output_data(BIO *out, BIO *data, ASN1_VALUE *val, int flags,
                                                const ASN1_ITEM *it);
static char * strip_ends(char *name);
static char * strip_start(char *name);
static char * strip_end(char *name);
static MIME_HEADER *mime_hdr_new(char *name, char *value);
static int mime_hdr_addparam(MIME_HEADER *mhdr, char *name, char *value);
static STACK_OF(MIME_HEADER) *mime_parse_hdr(BIO *bio);
static int mime_hdr_cmp(const MIME_HEADER * const *a,
                        const MIME_HEADER * const *b);
static int mime_param_cmp(const MIME_PARAM * const *a,
                        const MIME_PARAM * const *b);
static void mime_param_free(MIME_PARAM *param);
static int mime_bound_check(char *line, int linelen, char *bound, int blen);
static int multi_split(BIO *bio, char *bound, STACK_OF(BIO) **ret);
static int strip_eol(char *linebuf, int *plen);
static MIME_HEADER *mime_hdr_find(STACK_OF(MIME_HEADER) *hdrs, char *name);
static MIME_PARAM *mime_param_find(MIME_HEADER *hdr, char *name);
static void mime_hdr_free(MIME_HEADER *hdr);

#define MAX_SMLEN 1024
#define mime_debug(x) /* x */

/* Output an ASN1 structure in BER format streaming if necessary */

int i2d_ASN1_bio_stream(BIO *out, ASN1_VALUE *val, BIO *in, int flags,
                                const ASN1_ITEM *it)
        {
        /* If streaming create stream BIO and copy all content through it */
        if (flags & SMIME_STREAM)
                {
                BIO *bio, *tbio;
                bio = BIO_new_NDEF(out, val, it);
                if (!bio)
                        {
                        ASN1err(ASN1_F_I2D_ASN1_BIO_STREAM,ERR_R_MALLOC_FAILURE);
                        return 0;
                        }
```

```
128                 SMIME_crlf_copy(in, bio, flags);
129                 (void)BIO_flush(bio);
130                 /* Free up successive BIOs until we hit the old output BIO */
131                 do
132                         {
133                         tbio = BIO_pop(bio);
134                         BIO_free(bio);
135                         bio = tbio;
136                         } while (bio != out);
137                 }
138         /* else just write out ASN1 structure which will have all content
139          * stored internally
140          */
141         else
142                 ASN1_item_i2d_bio(it, out, val);
143         return 1;
144         }

146 /* Base 64 read and write of ASN1 structure */

148 static int B64_write_ASN1(BIO *out, ASN1_VALUE *val, BIO *in, int flags,
149                                 const ASN1_ITEM *it)
150         {
151         BIO *b64;
152         int r;
153         b64 = BIO_new(BIO_f_base64());
154         if(!b64)
155                 {
156                 ASN1err(ASN1_F_B64_WRITE_ASN1,ERR_R_MALLOC_FAILURE);
157                 return 0;
158                 }
159         /* prepend the b64 BIO so all data is base64 encoded.
160          */
161         out = BIO_push(b64, out);
162         r = i2d_ASN1_bio_stream(out, val, in, flags, it);
163         (void)BIO_flush(out);
164         BIO_pop(out);
165         BIO_free(b64);
166         return r;
167         }

169 /* Streaming ASN1 PEM write */

171 int PEM_write_bio_ASN1_stream(BIO *out, ASN1_VALUE *val, BIO *in, int flags,
172                                 const char *hdr,
173                                 const ASN1_ITEM *it)
174         {
175         int r;
176         BIO_printf(out, "-----BEGIN %s-----\n", hdr);
177         r = B64_write_ASN1(out, val, in, flags, it);
178         BIO_printf(out, "-----END %s-----\n", hdr);
179         return r;
180         }

182 static ASN1_VALUE *b64_read_asn1(BIO *bio, const ASN1_ITEM *it)
183 {
184         BIO *b64;
185         ASN1_VALUE *val;
186         if(!(b64 = BIO_new(BIO_f_base64()))) {
187                 ASN1err(ASN1_F_B64_READ_ASN1,ERR_R_MALLOC_FAILURE);
188                 return 0;
189         }
190         bio = BIO_push(b64, bio);
191         val = ASN1_item_d2i_bio(it, bio, NULL);
192         if(!val)
193                 ASN1err(ASN1_F_B64_READ_ASN1,ASN1_R_DECODE_ERROR);
```



```
194         (void)BIO_flush(bio);
195         bio = BIO_pop(bio);
196         BIO_free(b64);
197         return val;
198 }

200 /* Generate the MIME "micalg" parameter from RFC3851, RFC4490 */

202 static int asn1_write_micalg(BIO *out, STACK_OF(X509_ALGOR) *mdalgs)
203         {
204         const EVP_MD *md;
205         int i, have_unknown = 0, write_comma, ret = 0, md_nid;
206         have_unknown = 0;
207         write_comma = 0;
208         for (i = 0; i < sk_X509_ALGOR_num(mdalgs); i++)
209                 {
210                 if (write_comma)
211                         BIO_write(out, ",", 1);
212                 write_comma = 1;
213                 md_nid = OBJ_obj2nid(sk_X509_ALGOR_value(mdalgs, i)->algorithm);
214                 md = EVP_get_digestbynid(md_nid);
215                 if (md && md->md_ctrl)
216                         {
217                         int rv;
218                         char *micstr;
219                         rv = md->md_ctrl(NULL, EVP_MD_CTRL_MICALG, 0, &micstr);
220                         if (rv > 0)
221                                 {
222                                 BIO_puts(out, micstr);
223                                 OPENSSL_free(micstr);
224                                 continue;
225                                 }
226                         if (rv != -2)
227                                 goto err;
228                         }
229                 switch(md_nid)
230                         {
231                         case NID_sha1:
232                         BIO_puts(out, "sha1");
233                         break;

235                         case NID_md5:
236                         BIO_puts(out, "md5");
237                         break;

239                         case NID_sha256:
240                         BIO_puts(out, "sha-256");
241                         break;

243                         case NID_sha384:
244                         BIO_puts(out, "sha-384");
245                         break;

247                         case NID_sha512:
248                         BIO_puts(out, "sha-512");
249                         break;

251                         case NID_id_GostR3411_94:
252                         BIO_puts(out, "gostr3411-94");
253                                 goto err;
254                         break;

256                         default:
257                         if (have_unknown)
258                                 write_comma = 0;
259                         else
```

```
260                                 {
261                                 BIO_puts(out, "unknown");
262                                 have_unknown = 1;
263                                 }
264                         break;

266                         }
267                 }

269         ret = 1;
270         err:

272         return ret;

274         }

276 /* SMIME sender */

278 int SMIME_write_ASN1(BIO *bio, ASN1_VALUE *val, BIO *data, int flags,
279                                 int ctype_nid, int econt_nid,
280                                 STACK_OF(X509_ALGOR) *mdalgs,
281                                 const ASN1_ITEM *it)
282 {
283         char bound[33], c;
284         int i;
285         const char *mime_prefix, *mime_eol, *cname = "smime.p7m";
286         const char *msg_type=NULL;
287         if (flags & SMIME_OLDMIME)
288                 mime_prefix = "application/x-pkcs7-";
289         else
290                 mime_prefix = "application/pkcs7-";

292         if (flags & SMIME_CRLFEOL)
293                 mime_eol = "\r\n";
294         else
295                 mime_eol = "\n";
296         if((flags & SMIME_DETACHED) && data) {
297         /* We want multipart/signed */
298                 /* Generate a random boundary */
299                 RAND_pseudo_bytes((unsigned char *)bound, 32);
300                 for(i = 0; i < 32; i++) {
301                         c = bound[i] & 0xf;
302                         if(c < 10) c += '0';
303                         else c += 'A' - 10;
304                         bound[i] = c;
305                 }
306                 bound[32] = 0;
307                 BIO_printf(bio, "MIME-Version: 1.0%s", mime_eol);
308                 BIO_printf(bio, "Content-Type: multipart/signed;");
309                 BIO_printf(bio, " protocol=\"%ssignature\";", mime_prefix);
310                 BIO_puts(bio, " micalg=\"");
311                 asn1_write_micalg(bio, mdalgs);
312                 BIO_printf(bio, "\"; boundary=\"----%s\"%s%s",
313                                                 bound, mime_eol, mime_eol);
314                 BIO_printf(bio, "This is an S/MIME signed message%s%s",
315                                                 mime_eol, mime_eol);
316                 /* Now write out the first part */
317                 BIO_printf(bio, "------%s%s", bound, mime_eol);
318                 if (!asn1_output_data(bio, data, val, flags, it))
319                         return 0;
320                 BIO_printf(bio, "%s------%s%s", mime_eol, bound, mime_eol);

322                 /* Headers for signature */

324                 BIO_printf(bio, "Content-Type: %ssignature;", mime_prefix);
325                 BIO_printf(bio, " name=\"smime.p7s\"%s", mime_eol);
```



```
326                 BIO_printf(bio, "Content-Transfer-Encoding: base64%s",
327                                                         mime_eol);
328                 BIO_printf(bio, "Content-Disposition: attachment;");
329                 BIO_printf(bio, " filename=\"smime.p7s\"%s%s",
330                                                         mime_eol, mime_eol);
331                 B64_write_ASN1(bio, val, NULL, 0, it);
332                 BIO_printf(bio,"%s------%s--%s%s", mime_eol, bound,
333                                                         mime_eol, mime_eol);
334                 return 1;
335         }

337         /* Determine smime-type header */

339         if (ctype_nid == NID_pkcs7_enveloped)
340                 msg_type = "enveloped-data";
341         else if (ctype_nid == NID_pkcs7_signed)
342                 {
343                 if (econt_nid == NID_id_smime_ct_receipt)
344                         msg_type = "signed-receipt";
345                 else if (sk_X509_ALGOR_num(mdalgs) >= 0)
346                         msg_type = "signed-data";
347                 else
348                         msg_type = "certs-only";
349                 }
350         else if (ctype_nid == NID_id_smime_ct_compressedData)
351                 {
352                 msg_type = "compressed-data";
353                 cname = "smime.p7z";
354                 }
355         /* MIME headers */
356         BIO_printf(bio, "MIME-Version: 1.0%s", mime_eol);
357         BIO_printf(bio, "Content-Disposition: attachment;");
358         BIO_printf(bio, " filename=\"%s\"%s", cname, mime_eol);
359         BIO_printf(bio, "Content-Type: %smime;", mime_prefix);
360         if (msg_type)
361                 BIO_printf(bio, " smime-type=%s;", msg_type);
362         BIO_printf(bio, " name=\"%s\"%s", cname, mime_eol);
363         BIO_printf(bio, "Content-Transfer-Encoding: base64%s%s",
364                                                 mime_eol, mime_eol);
365         if (!B64_write_ASN1(bio, val, data, flags, it))
366                 return 0;
367         BIO_printf(bio, "%s", mime_eol);
368         return 1;
369 }

371 /* Handle output of ASN1 data */


374 static int asn1_output_data(BIO *out, BIO *data, ASN1_VALUE *val, int flags,
375                                         const ASN1_ITEM *it)
376         {
377         BIO *tmpbio;
378         const ASN1_AUX *aux = it->funcs;
379         ASN1_STREAM_ARG sarg;
380         int rv = 1;

382         /* If data is not deteched or resigning then the output BIO is
383          * already set up to finalise when it is written through.
384          */
385         if (!(flags & SMIME_DETACHED) || (flags & PKCS7_REUSE_DIGEST))
386                 {
387                 SMIME_crlf_copy(data, out, flags);
388                 return 1;
389                 }

391         if (!aux || !aux->asn1_cb)
```

```
392                 {
393                 ASN1err(ASN1_F_ASN1_OUTPUT_DATA,
394                                         ASN1_R_STREAMING_NOT_SUPPORTED);
395                 return 0;
396                 }

398         sarg.out = out;
399         sarg.ndef_bio = NULL;
400         sarg.boundary = NULL;

402         /* Let ASN1 code prepend any needed BIOs */

404         if (aux->asn1_cb(ASN1_OP_DETACHED_PRE, &val, it, &sarg) <= 0)
405                 return 0;

407         /* Copy data across, passing through filter BIOs for processing */
408         SMIME_crlf_copy(data, sarg.ndef_bio, flags);

410         /* Finalize structure */
411         if (aux->asn1_cb(ASN1_OP_DETACHED_POST, &val, it, &sarg) <= 0)
412                 rv = 0;

414         /* Now remove any digests prepended to the BIO */

416         while (sarg.ndef_bio != out)
417                 {
418                 tmpbio = BIO_pop(sarg.ndef_bio);
419                 BIO_free(sarg.ndef_bio);
420                 sarg.ndef_bio = tmpbio;
421                 }

423         return rv;

425         }

427 /* SMIME reader: handle multipart/signed and opaque signing.
428  * in multipart case the content is placed in a memory BIO
429  * pointed to by "bcont". In opaque this is set to NULL
430  */

432 ASN1_VALUE *SMIME_read_ASN1(BIO *bio, BIO **bcont, const ASN1_ITEM *it)
433 {
434         BIO *asnin;
435         STACK_OF(MIME_HEADER) *headers = NULL;
436         STACK_OF(BIO) *parts = NULL;
437         MIME_HEADER *hdr;
438         MIME_PARAM *prm;
439         ASN1_VALUE *val;
440         int ret;

442         if(bcont) *bcont = NULL;

444         if (!(headers = mime_parse_hdr(bio))) {
445                 ASN1err(ASN1_F_SMIME_READ_ASN1,ASN1_R_MIME_PARSE_ERROR);
446                 return NULL;
447         }

449         if(!(hdr = mime_hdr_find(headers, "content-type")) || !hdr->value) {
450                 sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
451                 ASN1err(ASN1_F_SMIME_READ_ASN1, ASN1_R_NO_CONTENT_TYPE);
452                 return NULL;
453         }

455         /* Handle multipart/signed */

457         if(!strcmp(hdr->value, "multipart/signed")) {
```



```
458                 /* Split into two parts */
459                 prm = mime_param_find(hdr, "boundary");
460                 if(!prm || !prm->param_value) {
461                         sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
462                         ASN1err(ASN1_F_SMIME_READ_ASN1, ASN1_R_NO_MULTIPART_BOUN
463                         return NULL;
464                 }
465                 ret = multi_split(bio, prm->param_value, &parts);
466                 sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
467                 if(!ret || (sk_BIO_num(parts) != 2) ) {
468                         ASN1err(ASN1_F_SMIME_READ_ASN1, ASN1_R_NO_MULTIPART_BODY
469                         sk_BIO_pop_free(parts, BIO_vfree);
470                         return NULL;
471                 }

473                 /* Parse the signature piece */
474                 asnin = sk_BIO_value(parts, 1);

476                 if (!(headers = mime_parse_hdr(asnin))) {
477                         ASN1err(ASN1_F_SMIME_READ_ASN1,ASN1_R_MIME_SIG_PARSE_ERR
478                         sk_BIO_pop_free(parts, BIO_vfree);
479                         return NULL;
480                 }

482                 /* Get content type */

484                 if(!(hdr = mime_hdr_find(headers, "content-type")) ||
485                                                                 !hdr->value) {
486                         sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
487                         ASN1err(ASN1_F_SMIME_READ_ASN1, ASN1_R_NO_SIG_CONTENT_TY
488                         return NULL;
489                 }

491                 if(strcmp(hdr->value, "application/x-pkcs7-signature") &&
492                         strcmp(hdr->value, "application/pkcs7-signature")) {
493                         ASN1err(ASN1_F_SMIME_READ_ASN1,ASN1_R_SIG_INVALID_MIME_T
494                         ERR_add_error_data(2, "type: ", hdr->value);
495                         sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
496                         sk_BIO_pop_free(parts, BIO_vfree);
497                         return NULL;
498                 }
499                 sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
500                 /* Read in ASN1 */
501                 if(!(val = b64_read_asn1(asnin, it))) {
502                         ASN1err(ASN1_F_SMIME_READ_ASN1,ASN1_R_ASN1_SIG_PARSE_ERR
503                         sk_BIO_pop_free(parts, BIO_vfree);
504                         return NULL;
505                 }

507                 if(bcont) {
508                         *bcont = sk_BIO_value(parts, 0);
509                         BIO_free(asnin);
510                         sk_BIO_free(parts);
511                 } else sk_BIO_pop_free(parts, BIO_vfree);
512                 return val;
513         }

515         /* OK, if not multipart/signed try opaque signature */

517         if (strcmp (hdr->value, "application/x-pkcs7-mime") &&
518             strcmp (hdr->value, "application/pkcs7-mime")) {
519                 ASN1err(ASN1_F_SMIME_READ_ASN1,ASN1_R_INVALID_MIME_TYPE);
520                 ERR_add_error_data(2, "type: ", hdr->value);
521                 sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
522                 return NULL;
523         }
```

```
525         sk_MIME_HEADER_pop_free(headers, mime_hdr_free);

527         if(!(val = b64_read_asn1(bio, it))) {
528                 ASN1err(ASN1_F_SMIME_READ_ASN1, ASN1_R_ASN1_PARSE_ERROR);
529                 return NULL;
530         }
531         return val;

533 }

535 /* Copy text from one BIO to another making the output CRLF at EOL */
536 int SMIME_crlf_copy(BIO *in, BIO *out, int flags)
537 {
538         BIO *bf;
539         char eol;
540         int len;
541         char linebuf[MAX_SMLEN];
542         /* Buffer output so we don't write one line at a time. This is
543          * useful when streaming as we don't end up with one OCTET STRING
544          * per line.
545          */
546         bf = BIO_new(BIO_f_buffer());
547         if (!bf)
548                 return 0;
549         out = BIO_push(bf, out);
550         if(flags & SMIME_BINARY)
551                 {
552                 while((len = BIO_read(in, linebuf, MAX_SMLEN)) > 0)
553                                                 BIO_write(out, linebuf, len);
554                 }
555         else
556                 {
557                 if(flags & SMIME_TEXT)
558                         BIO_printf(out, "Content-Type: text/plain\r\n\r\n");
559                 while ((len = BIO_gets(in, linebuf, MAX_SMLEN)) > 0)
560                         {
561                         eol = strip_eol(linebuf, &len);
562                         if (len)
563                                 BIO_write(out, linebuf, len);
564                         if(eol) BIO_write(out, "\r\n", 2);
565                         }
566                 }
567         (void)BIO_flush(out);
568         BIO_pop(out);
569         BIO_free(bf);
570         return 1;
571 }

573 /* Strip off headers if they are text/plain */
574 int SMIME_text(BIO *in, BIO *out)
575 {
576         char iobuf[4096];
577         int len;
578         STACK_OF(MIME_HEADER) *headers;
579         MIME_HEADER *hdr;

581         if (!(headers = mime_parse_hdr(in))) {
582                 ASN1err(ASN1_F_SMIME_TEXT,ASN1_R_MIME_PARSE_ERROR);
583                 return 0;
584         }
585         if(!(hdr = mime_hdr_find(headers, "content-type")) || !hdr->value) {
586                 ASN1err(ASN1_F_SMIME_TEXT,ASN1_R_MIME_NO_CONTENT_TYPE);
587                 sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
588                 return 0;
589         }
```



```
590         if (strcmp (hdr->value, "text/plain")) {
591                 ASN1err(ASN1_F_SMIME_TEXT,ASN1_R_INVALID_MIME_TYPE);
592                 ERR_add_error_data(2, "type: ", hdr->value);
593                 sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
594                 return 0;
595         }
596         sk_MIME_HEADER_pop_free(headers, mime_hdr_free);
597         while ((len = BIO_read(in, iobuf, sizeof(iobuf))) > 0)
598                                                 BIO_write(out, iobuf, len);
599         if (len < 0)
600                 return 0;
601         return 1;
602 }

604 /* Split a multipart/XXX message body into component parts: result is
605  * canonical parts in a STACK of bios
606  */

608 static int multi_split(BIO *bio, char *bound, STACK_OF(BIO) **ret)
609 {
610         char linebuf[MAX_SMLEN];
611         int len, blen;
612         int eol = 0, next_eol = 0;
613         BIO *bpart = NULL;
614         STACK_OF(BIO) *parts;
615         char state, part, first;

617         blen = strlen(bound);
618         part = 0;
619         state = 0;
620         first = 1;
621         parts = sk_BIO_new_null();
622         *ret = parts;
623         while ((len = BIO_gets(bio, linebuf, MAX_SMLEN)) > 0) {
624                 state = mime_bound_check(linebuf, len, bound, blen);
625                 if(state == 1) {
626                         first = 1;
627                         part++;
628                 } else if(state == 2) {
629                         sk_BIO_push(parts, bpart);
630                         return 1;
631                 } else if(part) {
632                         /* Strip CR+LF from linebuf */
633                         next_eol = strip_eol(linebuf, &len);
634                         if(first) {
635                                 first = 0;
636                                 if(bpart) sk_BIO_push(parts, bpart);
637                                 bpart = BIO_new(BIO_s_mem());
638                                 BIO_set_mem_eof_return(bpart, 0);
639                         } else if (eol)
640                                 BIO_write(bpart, "\r\n", 2);
641                         eol = next_eol;
642                         if (len)
643                                 BIO_write(bpart, linebuf, len);
644                 }
645         }
646         return 0;
647 }

649 /* This is the big one: parse MIME header lines up to message body */

651 #define MIME_INVALID    0
652 #define MIME_START      1
653 #define MIME_TYPE       2
654 #define MIME_NAME       3
655 #define MIME_VALUE      4
```

```
#define MIME_QUOTE      5
#define MIME_COMMENT    6


static STACK_OF(MIME_HEADER) *mime_parse_hdr(BIO *bio)
{
        char *p, *q, c;
        char *ntmp;
        char linebuf[MAX_SMLEN];
        MIME_HEADER *mhdr = NULL;
        STACK_OF(MIME_HEADER) *headers;
        int len, state, save_state = 0;

        headers = sk_MIME_HEADER_new(mime_hdr_cmp);
        if (!headers)
                return NULL;
        while ((len = BIO_gets(bio, linebuf, MAX_SMLEN)) > 0) {
        /* If whitespace at line start then continuation line */
        if(mhdr && isspace((unsigned char)linebuf[0])) state = MIME_NAME;
        else state = MIME_START;
        ntmp = NULL;
        /* Go through all characters */
        for(p = linebuf, q = linebuf; (c = *p) && (c!='\r') && (c!='\n'); p++) {

        /* State machine to handle MIME headers
         * if this looks horrible that's because it *is*
         */

                switch(state) {
                        case MIME_START:
                        if(c == ':') {
                                state = MIME_TYPE;
                                *p = 0;
                                ntmp = strip_ends(q);
                                q = p + 1;
                        }
                        break;

                        case MIME_TYPE:
                        if(c == ';') {
                                mime_debug("Found End Value\n");
                                *p = 0;
                                mhdr = mime_hdr_new(ntmp, strip_ends(q));
                                sk_MIME_HEADER_push(headers, mhdr);
                                ntmp = NULL;
                                q = p + 1;
                                state = MIME_NAME;
                        } else if(c == '(') {
                                save_state = state;
                                state = MIME_COMMENT;
                        }
                        break;

                        case MIME_COMMENT:
                        if(c == ')') {
                                state = save_state;
                        }
                        break;

                        case MIME_NAME:
                        if(c == '=') {
                                state = MIME_VALUE;
                                *p = 0;
                                ntmp = strip_ends(q);
                                q = p + 1;
                        }
```



```
                        break ;

                        case MIME_VALUE:
                        if(c == ';') {
                                state = MIME_NAME;
                                *p = 0;
                                mime_hdr_addparam(mhdr, ntmp, strip_ends(q));
                                ntmp = NULL;
                                q = p + 1;
                        } else if (c == '"') {
                                mime_debug("Found Quote\n");
                                state = MIME_QUOTE;
                        } else if(c == '(') {
                                save_state = state;
                                state = MIME_COMMENT;
                        }
                        break;

                        case MIME_QUOTE:
                        if(c == '"') {
                                mime_debug("Found Match Quote\n");
                                state = MIME_VALUE;
                        }
                        break;
                }
        }

        if(state == MIME_TYPE) {
                mhdr = mime_hdr_new(ntmp, strip_ends(q));
                sk_MIME_HEADER_push(headers, mhdr);
        } else if(state == MIME_VALUE)
                         mime_hdr_addparam(mhdr, ntmp, strip_ends(q));
        if(p == linebuf) break; /* Blank line means end of headers */
}

return headers;

}

static char *strip_ends(char *name)
{
        return strip_end(strip_start(name));
}

/* Strip a parameter of whitespace from start of param */
static char *strip_start(char *name)
{
        char *p, c;
        /* Look for first non white space or quote */
        for(p = name; (c = *p) ;p++) {
                if(c == '"') {
                        /* Next char is start of string if non null */
                        if(p[1]) return p + 1;
                        /* Else null string */
                        return NULL;
                }
                if(!isspace((unsigned char)c)) return p;
        }
        return NULL;
}

/* As above but strip from end of string : maybe should handle brackets? */
static char *strip_end(char *name)
{
        char *p, c;
        if(!name) return NULL;
```

```
788         /* Look for first non white space or quote */
789         for(p = name + strlen(name) - 1; p >= name ;p--) {
790                 c = *p;
791                 if(c == '"') {
792                         if(p - 1 == name) return NULL;
793                         *p = 0;
794                         return name;
795                 }
796                 if(isspace((unsigned char)c)) *p = 0;
797                 else return name;
798         }
799         return NULL;
800 }
801
802 static MIME_HEADER *mime_hdr_new(char *name, char *value)
803 {
804         MIME_HEADER *mhdr;
805         char *tmpname, *tmpval, *p;
806         int c;
807         if(name) {
808                 if(!(tmpname = BUF_strdup(name))) return NULL;
809                 for(p = tmpname ; *p; p++) {
810                         c = (unsigned char)*p;
811                         if(isupper(c)) {
812                                 c = tolower(c);
813                                 *p = c;
814                         }
815                 }
816         } else tmpname = NULL;
817         if(value) {
818                 if(!(tmpval = BUF_strdup(value))) return NULL;
819                 for(p = tmpval ; *p; p++) {
820                         c = (unsigned char)*p;
821                         if(isupper(c)) {
822                                 c = tolower(c);
823                                 *p = c;
824                         }
825                 }
826         } else tmpval = NULL;
827         mhdr = (MIME_HEADER *) OPENSSL_malloc(sizeof(MIME_HEADER));
828         if(!mhdr) return NULL;
829         mhdr->name = tmpname;
830         mhdr->value = tmpval;
831         if(!(mhdr->params = sk_MIME_PARAM_new(mime_param_cmp))) return NULL;
832         return mhdr;
833 }
834
835 static int mime_hdr_addparam(MIME_HEADER *mhdr, char *name, char *value)
836 {
837         char *tmpname, *tmpval, *p;
838         int c;
839         MIME_PARAM *mparam;
840         if(name) {
841                 tmpname = BUF_strdup(name);
842                 if(!tmpname) return 0;
843                 for(p = tmpname ; *p; p++) {
844                         c = (unsigned char)*p;
845                         if(isupper(c)) {
846                                 c = tolower(c);
847                                 *p = c;
848                         }
849                 }
850         } else tmpname = NULL;
851         if(value) {
852                 tmpval = BUF_strdup(value);
853                 if(!tmpval) return 0;
```



```
854         } else tmpval = NULL;
855         /* Parameter values are case sensitive so leave as is */
856         mparam = (MIME_PARAM *) OPENSSL_malloc(sizeof(MIME_PARAM));
857         if(!mparam) return 0;
858         mparam->param_name = tmpname;
859         mparam->param_value = tmpval;
860         sk_MIME_PARAM_push(mhdr->params, mparam);
861         return 1;
862 }
863
864 static int mime_hdr_cmp(const MIME_HEADER * const *a,
865                         const MIME_HEADER * const *b)
866 {
867         if (!(*a)->name || !(*b)->name)
868                 return !!(*a)->name - !!(*b)->name;
869
870         return(strcmp((*a)->name, (*b)->name));
871 }
872
873 static int mime_param_cmp(const MIME_PARAM * const *a,
874                         const MIME_PARAM * const *b)
875 {
876         if (!(*a)->param_name || !(*b)->param_name)
877                 return !!(*a)->param_name - !!(*b)->param_name;
878         return(strcmp((*a)->param_name, (*b)->param_name));
879 }
880
881 /* Find a header with a given name (if possible) */
882
883 static MIME_HEADER *mime_hdr_find(STACK_OF(MIME_HEADER) *hdrs, char *name)
884 {
885         MIME_HEADER htmp;
886         int idx;
887         htmp.name = name;
888         idx = sk_MIME_HEADER_find(hdrs, &htmp);
889         if(idx < 0) return NULL;
890         return sk_MIME_HEADER_value(hdrs, idx);
891 }
892
893 static MIME_PARAM *mime_param_find(MIME_HEADER *hdr, char *name)
894 {
895         MIME_PARAM param;
896         int idx;
897         param.param_name = name;
898         idx = sk_MIME_PARAM_find(hdr->params, &param);
899         if(idx < 0) return NULL;
900         return sk_MIME_PARAM_value(hdr->params, idx);
901 }
902
903 static void mime_hdr_free(MIME_HEADER *hdr)
904 {
905         if(hdr->name) OPENSSL_free(hdr->name);
906         if(hdr->value) OPENSSL_free(hdr->value);
907         if(hdr->params) sk_MIME_PARAM_pop_free(hdr->params, mime_param_free);
908         OPENSSL_free(hdr);
909 }
910
911 static void mime_param_free(MIME_PARAM *param)
912 {
913         if(param->param_name) OPENSSL_free(param->param_name);
914         if(param->param_value) OPENSSL_free(param->param_value);
915         OPENSSL_free(param);
916 }
917
918 /* Check for a multipart boundary. Returns:
919  * 0 : no boundary
```

```
 * 1 : part boundary
 * 2 : final boundary
 */
static int mime_bound_check(char *line, int linelen, char *bound, int blen)
{
        if(linelen == -1) linelen = strlen(line);
        if(blen == -1) blen = strlen(bound);
        /* Quickly eliminate if line length too short */
        if(blen + 2 > linelen) return 0;
        /* Check for part boundary */
        if(!strncmp(line, "--", 2) && !strncmp(line + 2, bound, blen)) {
                if(!strncmp(line + blen + 2, "--", 2)) return 2;
                else return 1;
        }
        return 0;
}

static int strip_eol(char *linebuf, int *plen)
        {
        int len = *plen;
        char *p, c;
        int is_eol = 0;
        p = linebuf + len - 1;
        for (p = linebuf + len - 1; len > 0; len--, p--)
                {
                c = *p;
                if (c == '\n')
                        is_eol = 1;
                else if (c != '\r')
                        break;
                }
        *plen = len;
        return is_eol;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4611 Wed Aug 13 19:52:01 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/asn_moid.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* asn_moid.c */
/* Written by Stephen Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <ctype.h>
#include <openssl/crypto.h>
```



```
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/dso.h>
#include <openssl/x509.h>

/* Simple ASN1 OID module: add all objects in a given section */

static int do_create(char *value, char *name);

static int oid_module_init(CONF_IMODULE *md, const CONF *cnf)
        {
        int i;
        const char *oid_section;
        STACK_OF(CONF_VALUE) *sktmp;
        CONF_VALUE *oval;
        oid_section = CONF_imodule_get_value(md);
        if(!(sktmp = NCONF_get_section(cnf, oid_section)))
                {
                ASN1err(ASN1_F_OID_MODULE_INIT, ASN1_R_ERROR_LOADING_SECTION);
                return 0;
                }
        for(i = 0; i < sk_CONF_VALUE_num(sktmp); i++)
                {
                oval = sk_CONF_VALUE_value(sktmp, i);
                if(!do_create(oval->value, oval->name))
                        {
                        ASN1err(ASN1_F_OID_MODULE_INIT, ASN1_R_ADDING_OBJECT);
                        return 0;
                        }
                }
        return 1;
        }

static void oid_module_finish(CONF_IMODULE *md)
        {
        OBJ_cleanup();
        }

void ASN1_add_oid_module(void)
        {
        CONF_module_add("oid_section", oid_module_init, oid_module_finish);
        }

/* Create an OID based on a name value pair. Accept two formats.
 * shortname = 1.2.3.4
 * shortname = some long name, 1.2.3.4
 */


static int do_create(char *value, char *name)
        {
        int nid;
        ASN1_OBJECT *oid;
        char *ln, *ostr, *p, *lntmp;
        p = strrchr(value, ',');
        if (!p)
                {
                ln = name;
                ostr = value;
                }
        else
                {
                ln = NULL;
                ostr = p + 1;
                if (!*ostr)
                        return 0;
```

```
128                 while(isspace((unsigned char)*ostr)) ostr++;
129                 }

131         nid = OBJ_create(ostr, name, ln);

133         if (nid == NID_undef)
134                 return 0;

136         if (p)
137                 {
138                 ln = value;
139                 while(isspace((unsigned char)*ln)) ln++;
140                 p--;
141                 while(isspace((unsigned char)*p))
142                         {
143                         if (p == ln)
144                                 return 0;
145                         p--;
146                         }
147                 p++;
148                 lntmp = OPENSSL_malloc((p - ln) + 1);
149                 if (lntmp == NULL)
150                         return 0;
151                 memcpy(lntmp, ln, p - ln);
152                 lntmp[p - ln] = 0;
153                 oid = OBJ_nid2obj(nid);
154                 oid->ln = lntmp;
155                 }

157         return 1;
158         }
159 #endif /* ! codereview */
```

```
**********************************************************
    6030 Wed Aug 13 19:52:01 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/asn_pack.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* asn_pack.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#ifndef NO_ASN1_OLD

/* ASN1 packing and unpacking functions */

/* Turn an ASN1 encoded SEQUENCE OF into a STACK of structures */

STACK_OF(OPENSSL_BLOCK) *ASN1_seq_unpack(const unsigned char *buf, int len,
                        d2i_of_void *d2i, void (*free_func)(OPENSSL_BLOCK))
{
    STACK_OF(OPENSSL_BLOCK) *sk;
    const unsigned char *pbuf;
    pbuf =  buf;
    if (!(sk = d2i_ASN1_SET(NULL, &pbuf, len, d2i, free_func,
                                        V_ASN1_SEQUENCE, V_ASN1_UNIVERSAL)))
                ASN1err(ASN1_F_ASN1_SEQ_UNPACK,ASN1_R_DECODE_ERROR);
    return sk;
}

/* Turn a STACK structures into an ASN1 encoded SEQUENCE OF structure in a
 * OPENSSL_malloc'ed buffer
 */

unsigned char *ASN1_seq_pack(STACK_OF(OPENSSL_BLOCK) *safes, i2d_of_void *i2d,
                             unsigned char **buf, int *len)
{
        int safelen;
        unsigned char *safe, *p;
        if (!(safelen = i2d_ASN1_SET(safes, NULL, i2d, V_ASN1_SEQUENCE,
                                                V_ASN1_UNIVERSAL, IS_SEQUENCE))) {
                ASN1err(ASN1_F_ASN1_SEQ_PACK,ASN1_R_ENCODE_ERROR);
                return NULL;
        }
        if (!(safe = OPENSSL_malloc (safelen))) {
                ASN1err(ASN1_F_ASN1_SEQ_PACK,ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        p = safe;
        i2d_ASN1_SET(safes, &p, i2d, V_ASN1_SEQUENCE, V_ASN1_UNIVERSAL,
                                                                IS_SEQUENCE);
        if (len) *len = safelen;
        if (buf) *buf = safe;
        return safe;
}

/* Extract an ASN1 object from an ASN1_STRING */

void *ASN1_unpack_string(ASN1_STRING *oct, d2i_of_void *d2i)
{
        const unsigned char *p;
        char *ret;

        p = oct->data;
        if(!(ret = d2i(NULL, &p, oct->length)))
                ASN1err(ASN1_F_ASN1_UNPACK_STRING,ASN1_R_DECODE_ERROR);
        return ret;
}

/* Pack an ASN1 object into an ASN1_STRING */

ASN1_STRING *ASN1_pack_string(void *obj, i2d_of_void *i2d, ASN1_STRING **oct)
{
        unsigned char *p;
        ASN1_STRING *octmp;

        if (!oct || !*oct) {
```

```
128                 if (!(octmp = ASN1_STRING_new ())) {
129                         ASN1err(ASN1_F_ASN1_PACK_STRING,ERR_R_MALLOC_FAILURE);
130                         return NULL;
131                 }
132                 if (oct) *oct = octmp;
133         } else octmp = *oct;

135         if (!(octmp->length = i2d(obj, NULL))) {
136                 ASN1err(ASN1_F_ASN1_PACK_STRING,ASN1_R_ENCODE_ERROR);
137                 goto err;
138         }
139         if (!(p = OPENSSL_malloc (octmp->length))) {
140                 ASN1err(ASN1_F_ASN1_PACK_STRING,ERR_R_MALLOC_FAILURE);
141                 goto err;
142         }
143         octmp->data = p;
144         i2d (obj, &p);
145         return octmp;
146         err:
147         if (!oct || !*oct)
148                 {
149                 ASN1_STRING_free(octmp);
150                 if (oct)
151                         *oct = NULL;
152                 }
153         return NULL;
154 }

156 #endif

158 /* ASN1_ITEM versions of the above */

160 ASN1_STRING *ASN1_item_pack(void *obj, const ASN1_ITEM *it, ASN1_STRING **oct)
161 {
162         ASN1_STRING *octmp;

164         if (!oct || !*oct) {
165                 if (!(octmp = ASN1_STRING_new ())) {
166                         ASN1err(ASN1_F_ASN1_ITEM_PACK,ERR_R_MALLOC_FAILURE);
167                         return NULL;
168                 }
169                 if (oct) *oct = octmp;
170         } else octmp = *oct;

172         if(octmp->data) {
173                 OPENSSL_free(octmp->data);
174                 octmp->data = NULL;
175         }

177         if (!(octmp->length = ASN1_item_i2d(obj, &octmp->data, it))) {
178                 ASN1err(ASN1_F_ASN1_ITEM_PACK,ASN1_R_ENCODE_ERROR);
179                 return NULL;
180         }
181         if (!octmp->data) {
182                 ASN1err(ASN1_F_ASN1_ITEM_PACK,ERR_R_MALLOC_FAILURE);
183                 return NULL;
184         }
185         return octmp;
186 }

188 /* Extract an ASN1 object from an ASN1_STRING */

190 void *ASN1_item_unpack(ASN1_STRING *oct, const ASN1_ITEM *it)
191 {
192         const unsigned char *p;
193         void *ret;
```



```
195         p = oct->data;
196         if(!(ret = ASN1_item_d2i(NULL, &p, oct->length, it)))
197                 ASN1err(ASN1_F_ASN1_ITEM_UNPACK,ASN1_R_DECODE_ERROR);
198         return ret;
199 }
200 #endif /* ! codereview */
```

```
**********************************************************
   11782 Wed Aug 13 19:52:02 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/bio_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* bio_asn1.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* Experimental ASN1 BIO. When written through the data is converted
 * to an ASN1 string type: default is OCTET STRING. Additional functions
 * can be provided to add prefix and suffix data.
```



```
 */

#include <string.h>
#include <openssl/bio.h>
#include <openssl/asn1.h>

/* Must be large enough for biggest tag+length */
#define DEFAULT_ASN1_BUF_SIZE 20

typedef enum
        {
        ASN1_STATE_START,
        ASN1_STATE_PRE_COPY,
        ASN1_STATE_HEADER,
        ASN1_STATE_HEADER_COPY,
        ASN1_STATE_DATA_COPY,
        ASN1_STATE_POST_COPY,
        ASN1_STATE_DONE
        } asn1_bio_state_t;

typedef struct BIO_ASN1_EX_FUNCS_st
        {
        asn1_ps_func    *ex_func;
        asn1_ps_func    *ex_free_func;
        } BIO_ASN1_EX_FUNCS;

typedef struct BIO_ASN1_BUF_CTX_t
        {
        /* Internal state */
        asn1_bio_state_t state;
        /* Internal buffer */
        unsigned char *buf;
        /* Size of buffer */
        int bufsize;
        /* Current position in buffer */
        int bufpos;
        /* Current buffer length */
        int buflen;
        /* Amount of data to copy */
        int copylen;
        /* Class and tag to use */
        int asn1_class, asn1_tag;
        asn1_ps_func *prefix, *prefix_free, *suffix, *suffix_free;
        /* Extra buffer for prefix and suffix data */
        unsigned char *ex_buf;
        int ex_len;
        int ex_pos;
        void *ex_arg;
        } BIO_ASN1_BUF_CTX;


static int asn1_bio_write(BIO *h, const char *buf,int num);
static int asn1_bio_read(BIO *h, char *buf, int size);
static int asn1_bio_puts(BIO *h, const char *str);
static int asn1_bio_gets(BIO *h, char *str, int size);
static long asn1_bio_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int asn1_bio_new(BIO *h);
static int asn1_bio_free(BIO *data);
static long asn1_bio_callback_ctrl(BIO *h, int cmd, bio_info_cb *fp);

static int asn1_bio_init(BIO_ASN1_BUF_CTX *ctx, int size);
static int asn1_bio_flush_ex(BIO *b, BIO_ASN1_BUF_CTX *ctx,
                             asn1_ps_func *cleanup, asn1_bio_state_t next);
static int asn1_bio_setup_ex(BIO *b, BIO_ASN1_BUF_CTX *ctx,
                             asn1_ps_func *setup,
                             asn1_bio_state_t ex_state,
```

```
128                                 asn1_bio_state_t other_state);

130 static BIO_METHOD methods_asn1=
131         {
132         BIO_TYPE_ASN1,
133         "asn1",
134         asn1_bio_write,
135         asn1_bio_read,
136         asn1_bio_puts,
137         asn1_bio_gets,
138         asn1_bio_ctrl,
139         asn1_bio_new,
140         asn1_bio_free,
141         asn1_bio_callback_ctrl,
142         };

144 BIO_METHOD *BIO_f_asn1(void)
145         {
146         return(&methods_asn1);
147         }


150 static int asn1_bio_new(BIO *b)
151         {
152         BIO_ASN1_BUF_CTX *ctx;
153         ctx = OPENSSL_malloc(sizeof(BIO_ASN1_BUF_CTX));
154         if (!ctx)
155                 return 0;
156         if (!asn1_bio_init(ctx, DEFAULT_ASN1_BUF_SIZE))
157                 {
158                 OPENSSL_free(ctx);
159                 return 0;
160                 }
161         b->init = 1;
162         b->ptr = (char *)ctx;
163         b->flags = 0;
164         return 1;
165         }

167 static int asn1_bio_init(BIO_ASN1_BUF_CTX *ctx, int size)
168         {
169         ctx->buf = OPENSSL_malloc(size);
170         if (!ctx->buf)
171                 return 0;
172         ctx->bufsize = size;
173         ctx->bufpos = 0;
174         ctx->buflen = 0;
175         ctx->copylen = 0;
176         ctx->asn1_class = V_ASN1_UNIVERSAL;
177         ctx->asn1_tag = V_ASN1_OCTET_STRING;
178         ctx->ex_buf = 0;
179         ctx->ex_pos = 0;
180         ctx->ex_len = 0;
181         ctx->state = ASN1_STATE_START;
182         return 1;
183         }

185 static int asn1_bio_free(BIO *b)
186         {
187         BIO_ASN1_BUF_CTX *ctx;
188         ctx = (BIO_ASN1_BUF_CTX *) b->ptr;
189         if (ctx == NULL)
190                 return 0;
191         if (ctx->buf)
192                 OPENSSL_free(ctx->buf);
193         OPENSSL_free(ctx);
```



```
194         b->init = 0;
195         b->ptr = NULL;
196         b->flags = 0;
197         return 1;
198         }

200 static int asn1_bio_write(BIO *b, const char *in , int inl)
201         {
202         BIO_ASN1_BUF_CTX *ctx;
203         int wrmax, wrlen, ret;
204         unsigned char *p;
205         if (!in || (inl < 0) || (b->next_bio == NULL))
206                 return 0;
207         ctx = (BIO_ASN1_BUF_CTX *) b->ptr;
208         if (ctx == NULL)
209                 return 0;

211         wrlen = 0;
212         ret = -1;

214         for(;;)
215                 {
216                 switch (ctx->state)
217                         {

219                         /* Setup prefix data, call it */
220                         case ASN1_STATE_START:
221                         if (!asn1_bio_setup_ex(b, ctx, ctx->prefix,
222                                 ASN1_STATE_PRE_COPY, ASN1_STATE_HEADER))
223                                 return 0;
224                         break;

226                         /* Copy any pre data first */
227                         case ASN1_STATE_PRE_COPY:

229                         ret = asn1_bio_flush_ex(b, ctx, ctx->prefix_free,
230                                                 ASN1_STATE_HEADER);

232                         if (ret <= 0)
233                                 goto done;

235                         break;

237                         case ASN1_STATE_HEADER:
238                         ctx->buflen =
239                                 ASN1_object_size(0, inl, ctx->asn1_tag) - inl;
240                         OPENSSL_assert(ctx->buflen <= ctx->bufsize);
241                         p = ctx->buf;
242                         ASN1_put_object(&p, 0, inl,
243                                         ctx->asn1_tag, ctx->asn1_class);
244                         ctx->copylen = inl;
245                         ctx->state = ASN1_STATE_HEADER_COPY;

247                         break;

249                         case ASN1_STATE_HEADER_COPY:
250                         ret = BIO_write(b->next_bio,
251                                         ctx->buf + ctx->bufpos, ctx->buflen);
252                         if (ret <= 0)
253                                 goto done;

255                         ctx->buflen -= ret;
256                         if (ctx->buflen)
257                                 ctx->bufpos += ret;
258                         else
259                                 {
```

```
260                                 ctx->bufpos = 0;
261                                 ctx->state = ASN1_STATE_DATA_COPY;
262                                 }

264                         break;

266                         case ASN1_STATE_DATA_COPY:

268                         if (inl > ctx->copylen)
269                                 wrmax = ctx->copylen;
270                         else
271                                 wrmax = inl;
272                         ret = BIO_write(b->next_bio, in, wrmax);
273                         if (ret <= 0)
274                                 break;
275                         wrlen += ret;
276                         ctx->copylen -= ret;
277                         in += ret;
278                         inl -= ret;

280                         if (ctx->copylen == 0)
281                                 ctx->state = ASN1_STATE_HEADER;

283                         if (inl == 0)
284                                 goto done;

286                         break;

288                         default:
289                         BIO_clear_retry_flags(b);
290                         return 0;

292                         }

294                 }

296         done:
297         BIO_clear_retry_flags(b);
298         BIO_copy_next_retry(b);

300         return (wrlen > 0) ? wrlen : ret;

302         }

304 static int asn1_bio_flush_ex(BIO *b, BIO_ASN1_BUF_CTX *ctx,
305                              asn1_ps_func *cleanup, asn1_bio_state_t next)
306         {
307         int ret;
308         if (ctx->ex_len <= 0)
309                 return 1;
310         for(;;)
311                 {
312                 ret = BIO_write(b->next_bio, ctx->ex_buf + ctx->ex_pos,
313                                                                 ctx->ex_len);
314                 if (ret <= 0)
315                         break;
316                 ctx->ex_len -= ret;
317                 if (ctx->ex_len > 0)
318                         ctx->ex_pos += ret;
319                 else
320                         {
321                         if(cleanup)
322                                 cleanup(b, &ctx->ex_buf, &ctx->ex_len,
323                                                                 &ctx->ex_arg);
324                         ctx->state = next;
325                         ctx->ex_pos = 0;
```



```
326                         break;
327                         }
328                 }
329         return ret;
330         }

332 static int asn1_bio_setup_ex(BIO *b, BIO_ASN1_BUF_CTX *ctx,
333                              asn1_ps_func *setup,
334                              asn1_bio_state_t ex_state,
335                              asn1_bio_state_t other_state)
336         {
337         if (setup && !setup(b, &ctx->ex_buf, &ctx->ex_len, &ctx->ex_arg))
338                 {
339                 BIO_clear_retry_flags(b);
340                 return 0;
341                 }
342         if (ctx->ex_len > 0)
343                 ctx->state = ex_state;
344         else
345                 ctx->state = other_state;
346         return 1;
347         }

349 static int asn1_bio_read(BIO *b, char *in , int inl)
350         {
351         if (!b->next_bio)
352                 return 0;
353         return BIO_read(b->next_bio, in , inl);
354         }

356 static int asn1_bio_puts(BIO *b, const char *str)
357         {
358         return asn1_bio_write(b, str, strlen(str));
359         }

361 static int asn1_bio_gets(BIO *b, char *str, int size)
362         {
363         if (!b->next_bio)
364                 return 0;
365         return BIO_gets(b->next_bio, str , size);
366         }

368 static long asn1_bio_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
369         {
370         if (b->next_bio == NULL) return(0);
371         return BIO_callback_ctrl(b->next_bio,cmd,fp);
372         }

374 static long asn1_bio_ctrl(BIO *b, int cmd, long arg1, void *arg2)
375         {
376         BIO_ASN1_BUF_CTX *ctx;
377         BIO_ASN1_EX_FUNCS *ex_func;
378         long ret = 1;
379         ctx = (BIO_ASN1_BUF_CTX *) b->ptr;
380         if (ctx == NULL)
381                 return 0;
382         switch(cmd)
383                 {

385                 case BIO_C_SET_PREFIX:
386                 ex_func = arg2;
387                 ctx->prefix  = ex_func->ex_func;
388                 ctx->prefix_free  = ex_func->ex_free_func;
389                 break;

391                 case BIO_C_GET_PREFIX:
```

```
392                 ex_func = arg2;
393                 ex_func->ex_func = ctx->prefix;
394                 ex_func->ex_free_func = ctx->prefix_free;
395                 break;

397                 case BIO_C_SET_SUFFIX:
398                 ex_func = arg2;
399                 ctx->suffix  = ex_func->ex_func;
400                 ctx->suffix_free  = ex_func->ex_free_func;
401                 break;

403                 case BIO_C_GET_SUFFIX:
404                 ex_func = arg2;
405                 ex_func->ex_func = ctx->suffix;
406                 ex_func->ex_free_func = ctx->suffix_free;
407                 break;

409                 case BIO_C_SET_EX_ARG:
410                 ctx->ex_arg = arg2;
411                 break;

413                 case BIO_C_GET_EX_ARG:
414                 *(void **)arg2 = ctx->ex_arg;
415                 break;

417                 case BIO_CTRL_FLUSH:
418                 if (!b->next_bio)
419                         return 0;

421                 /* Call post function if possible */
422                 if (ctx->state == ASN1_STATE_HEADER)
423                         {
424                         if (!asn1_bio_setup_ex(b, ctx, ctx->suffix,
425                                 ASN1_STATE_POST_COPY, ASN1_STATE_DONE))
426                                 return 0;
427                         }

429                 if (ctx->state == ASN1_STATE_POST_COPY)
430                         {
431                         ret = asn1_bio_flush_ex(b, ctx, ctx->suffix_free,
432                                                 ASN1_STATE_DONE);
433                         if (ret <= 0)
434                                 return ret;
435                         }

437                 if (ctx->state == ASN1_STATE_DONE)
438                         return BIO_ctrl(b->next_bio, cmd, arg1, arg2);
439                 else
440                         {
441                         BIO_clear_retry_flags(b);
442                         return 0;
443                         }
444                 break;


447                 default:
448                 if (!b->next_bio)
449                         return 0;
450                 return BIO_ctrl(b->next_bio, cmd, arg1, arg2);

452                 }

454         return ret;
455         }

457 static int asn1_bio_set_ex(BIO *b, int cmd,
```



```
458                 asn1_ps_func *ex_func, asn1_ps_func *ex_free_func)
459         {
460         BIO_ASN1_EX_FUNCS extmp;
461         extmp.ex_func = ex_func;
462         extmp.ex_free_func = ex_free_func;
463         return BIO_ctrl(b, cmd, 0, &extmp);
464         }

466 static int asn1_bio_get_ex(BIO *b, int cmd,
467                 asn1_ps_func **ex_func, asn1_ps_func **ex_free_func)
468         {
469         BIO_ASN1_EX_FUNCS extmp;
470         int ret;
471         ret = BIO_ctrl(b, cmd, 0, &extmp);
472         if (ret > 0)
473                 {
474                 *ex_func = extmp.ex_func;
475                 *ex_free_func = extmp.ex_free_func;
476                 }
477         return ret;
478         }

480 int BIO_asn1_set_prefix(BIO *b, asn1_ps_func *prefix, asn1_ps_func *prefix_free)
481         {
482         return asn1_bio_set_ex(b, BIO_C_SET_PREFIX, prefix, prefix_free);
483         }

485 int BIO_asn1_get_prefix(BIO *b, asn1_ps_func **pprefix, asn1_ps_func **pprefix_f
486         {
487         return asn1_bio_get_ex(b, BIO_C_GET_PREFIX, pprefix, pprefix_free);
488         }

490 int BIO_asn1_set_suffix(BIO *b, asn1_ps_func *suffix, asn1_ps_func *suffix_free)
491         {
492         return asn1_bio_set_ex(b, BIO_C_SET_SUFFIX, suffix, suffix_free);
493         }

495 int BIO_asn1_get_suffix(BIO *b, asn1_ps_func **psuffix, asn1_ps_func **psuffix_f
496         {
497         return asn1_bio_get_ex(b, BIO_C_GET_SUFFIX, psuffix, psuffix_free);
498         }
499 #endif /* ! codereview */
```

```
**********************************************************
    6971 Wed Aug 13 19:52:02 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/bio_ndef.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* bio_ndef.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/bio.h>
#include <openssl/err.h>

#include <stdio.h>
```



```
/* Experimental NDEF ASN1 BIO support routines */

/* The usage is quite simple, initialize an ASN1 structure,
 * get a BIO from it then any data written through the BIO
 * will end up translated to approptiate format on the fly.
 * The data is streamed out and does *not* need to be
 * all held in memory at once.
 *
 * When the BIO is flushed the output is finalized and any
 * signatures etc written out.
 *
 * The BIO is a 'proper' BIO and can handle non blocking I/O
 * correctly.
 *
 * The usage is simple. The implementation is *not*...
 */

/* BIO support data stored in the ASN1 BIO ex_arg */

typedef struct ndef_aux_st
        {
        /* ASN1 structure this BIO refers to */
        ASN1_VALUE *val;
        const ASN1_ITEM *it;
        /* Top of the BIO chain */
        BIO *ndef_bio;
        /* Output BIO */
        BIO *out;
        /* Boundary where content is inserted */
        unsigned char **boundary;
        /* DER buffer start */
        unsigned char *derbuf;
        } NDEF_SUPPORT;

static int ndef_prefix(BIO *b, unsigned char **pbuf, int *plen, void *parg);
static int ndef_prefix_free(BIO *b, unsigned char **pbuf, int *plen, void *parg)
static int ndef_suffix(BIO *b, unsigned char **pbuf, int *plen, void *parg);
static int ndef_suffix_free(BIO *b, unsigned char **pbuf, int *plen, void *parg)

BIO *BIO_new_NDEF(BIO *out, ASN1_VALUE *val, const ASN1_ITEM *it)
        {
        NDEF_SUPPORT *ndef_aux = NULL;
        BIO *asn_bio = NULL;
        const ASN1_AUX *aux = it->funcs;
        ASN1_STREAM_ARG sarg;

        if (!aux || !aux->asn1_cb)
                {
                ASN1err(ASN1_F_BIO_NEW_NDEF, ASN1_R_STREAMING_NOT_SUPPORTED);
                return NULL;
                }
        ndef_aux = OPENSSL_malloc(sizeof(NDEF_SUPPORT));
        asn_bio = BIO_new(BIO_f_asn1());

        /* ASN1 bio needs to be next to output BIO */

        out = BIO_push(asn_bio, out);

        if (!ndef_aux || !asn_bio || !out)
                goto err;

        BIO_asn1_set_prefix(asn_bio, ndef_prefix, ndef_prefix_free);
        BIO_asn1_set_suffix(asn_bio, ndef_suffix, ndef_suffix_free);

        /* Now let callback prepend any digest, cipher etc BIOs
         * ASN1 structure needs.
```

```
128          */
129
130         sarg.out = out;
131         sarg.ndef_bio = NULL;
132         sarg.boundary = NULL;
133
134         if (aux->asn1_cb(ASN1_OP_STREAM_PRE, &val, it, &sarg) <= 0)
135                 goto err;
136
137         ndef_aux->val = val;
138         ndef_aux->it = it;
139         ndef_aux->ndef_bio = sarg.ndef_bio;
140         ndef_aux->boundary = sarg.boundary;
141         ndef_aux->out = out;
142
143         BIO_ctrl(asn_bio, BIO_C_SET_EX_ARG, 0, ndef_aux);
144
145         return sarg.ndef_bio;
146
147         err:
148         if (asn_bio)
149                 BIO_free(asn_bio);
150         if (ndef_aux)
151                 OPENSSL_free(ndef_aux);
152         return NULL;
153         }
154
155 static int ndef_prefix(BIO *b, unsigned char **pbuf, int *plen, void *parg)
156         {
157         NDEF_SUPPORT *ndef_aux;
158         unsigned char *p;
159         int derlen;
160
161         if (!parg)
162                 return 0;
163
164         ndef_aux = *(NDEF_SUPPORT **)parg;
165
166         derlen = ASN1_item_ndef_i2d(ndef_aux->val, NULL, ndef_aux->it);
167         p = OPENSSL_malloc(derlen);
168         ndef_aux->derbuf = p;
169         *pbuf = p;
170         derlen = ASN1_item_ndef_i2d(ndef_aux->val, &p, ndef_aux->it);
171
172         if (!*ndef_aux->boundary)
173                 return 0;
174
175         *plen = *ndef_aux->boundary - *pbuf;
176
177         return 1;
178         }
179
180 static int ndef_prefix_free(BIO *b, unsigned char **pbuf, int *plen, void *parg)
181         {
182         NDEF_SUPPORT *ndef_aux;
183
184         if (!parg)
185                 return 0;
186
187         ndef_aux = *(NDEF_SUPPORT **)parg;
188
189         if (ndef_aux->derbuf)
190                 OPENSSL_free(ndef_aux->derbuf);
191
192         ndef_aux->derbuf = NULL;
193         *pbuf = NULL;
```



```
194         *plen = 0;
195         return 1;
196         }
197
198 static int ndef_suffix_free(BIO *b, unsigned char **pbuf, int *plen, void *parg)
199         {
200         NDEF_SUPPORT **pndef_aux = (NDEF_SUPPORT **)parg;
201         if (!ndef_prefix_free(b, pbuf, plen, parg))
202                 return 0;
203         OPENSSL_free(*pndef_aux);
204         *pndef_aux = NULL;
205         return 1;
206         }
207
208 static int ndef_suffix(BIO *b, unsigned char **pbuf, int *plen, void *parg)
209         {
210         NDEF_SUPPORT *ndef_aux;
211         unsigned char *p;
212         int derlen;
213         const ASN1_AUX *aux;
214         ASN1_STREAM_ARG sarg;
215
216         if (!parg)
217                 return 0;
218
219         ndef_aux = *(NDEF_SUPPORT **)parg;
220
221         aux = ndef_aux->it->funcs;
222
223         /* Finalize structures */
224         sarg.ndef_bio = ndef_aux->ndef_bio;
225         sarg.out = ndef_aux->out;
226         sarg.boundary = ndef_aux->boundary;
227         if (aux->asn1_cb(ASN1_OP_STREAM_POST,
228                                 &ndef_aux->val, ndef_aux->it, &sarg) <= 0)
229                 return 0;
230
231         derlen = ASN1_item_ndef_i2d(ndef_aux->val, NULL, ndef_aux->it);
232         p = OPENSSL_malloc(derlen);
233         ndef_aux->derbuf = p;
234         *pbuf = p;
235         derlen = ASN1_item_ndef_i2d(ndef_aux->val, &p, ndef_aux->it);
236
237         if (!*ndef_aux->boundary)
238                 return 0;
239         *pbuf = *ndef_aux->boundary;
240         *plen = derlen - (*ndef_aux->boundary - ndef_aux->derbuf);
241
242         return 1;
243         }
244 #endif /* ! codereview */
```

```
**********************************************************
    5848 Wed Aug 13 19:52:02 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/d2i_pr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/d2i_pr.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/evp.h>
#include <openssl/objects.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include <openssl/x509.h>
#include <openssl/asn1.h>
#include "asn1_locl.h"

EVP_PKEY *d2i_PrivateKey(int type, EVP_PKEY **a, const unsigned char **pp,
             long length)
        {
        EVP_PKEY *ret;

        if ((a == NULL) || (*a == NULL))
                {
                if ((ret=EVP_PKEY_new()) == NULL)
                        {
                        ASN1err(ASN1_F_D2I_PRIVATEKEY,ERR_R_EVP_LIB);
                        return(NULL);
                        }
                }
        else
                {
                ret= *a;
#ifndef OPENSSL_NO_ENGINE
                if (ret->engine)
                        {
                        ENGINE_finish(ret->engine);
                        ret->engine = NULL;
                        }
#endif
                }

        if (!EVP_PKEY_set_type(ret, type))
                {
                ASN1err(ASN1_F_D2I_PRIVATEKEY,ASN1_R_UNKNOWN_PUBLIC_KEY_TYPE);
                goto err;
                }

        if (!ret->ameth->old_priv_decode ||
                        !ret->ameth->old_priv_decode(ret, pp, length))
                {
                if (ret->ameth->priv_decode)
                        {
                        PKCS8_PRIV_KEY_INFO *p8=NULL;
                        p8=d2i_PKCS8_PRIV_KEY_INFO(NULL,pp,length);
                        if (!p8) goto err;
                        EVP_PKEY_free(ret);
                        ret = EVP_PKCS82PKEY(p8);
                        PKCS8_PRIV_KEY_INFO_free(p8);

                        }
                else
                        {
                        ASN1err(ASN1_F_D2I_PRIVATEKEY,ERR_R_ASN1_LIB);
                        goto err;
                        }
                }
        if (a != NULL) (*a)=ret;
        return(ret);
err:
        if ((ret != NULL) && ((a == NULL) || (*a != ret))) EVP_PKEY_free(ret);
        return(NULL);
        }
```

```
128 /* This works like d2i_PrivateKey() except it automatically works out the type *

130 EVP_PKEY *d2i_AutoPrivateKey(EVP_PKEY **a, const unsigned char **pp,
131              long length)
132 {
133         STACK_OF(ASN1_TYPE) *inkey;
134         const unsigned char *p;
135         int keytype;
136         p = *pp;
137         /* Dirty trick: read in the ASN1 data into a STACK_OF(ASN1_TYPE):
138          * by analyzing it we can determine the passed structure: this
139          * assumes the input is surrounded by an ASN1 SEQUENCE.
140          */
141         inkey = d2i_ASN1_SEQUENCE_ANY(NULL, &p, length);
142         /* Since we only need to discern "traditional format" RSA and DSA
143          * keys we can just count the elements.
144          */
145         if(sk_ASN1_TYPE_num(inkey) == 6)
146                 keytype = EVP_PKEY_DSA;
147         else if (sk_ASN1_TYPE_num(inkey) == 4)
148                 keytype = EVP_PKEY_EC;
149         else if (sk_ASN1_TYPE_num(inkey) == 3)
150                 { /* This seems to be PKCS8, not traditional format */
151                         PKCS8_PRIV_KEY_INFO *p8 = d2i_PKCS8_PRIV_KEY_INFO(NULL,p
152                         EVP_PKEY *ret;

154                         sk_ASN1_TYPE_pop_free(inkey, ASN1_TYPE_free);
155                         if (!p8)
156                                 {
157                                 ASN1err(ASN1_F_D2I_AUTOPRIVATEKEY,ASN1_R_UNSUPPO
158                                 return NULL;
159                                 }
160                         ret = EVP_PKCS82PKEY(p8);
161                         PKCS8_PRIV_KEY_INFO_free(p8);
162                         if (a) {
163                                 *a = ret;
164                         }
165                         return ret;
166                 }
167         else keytype = EVP_PKEY_RSA;
168         sk_ASN1_TYPE_pop_free(inkey, ASN1_TYPE_free);
169         return d2i_PrivateKey(keytype, a, pp, length);
170 }
171 #endif /* ! codereview */
```

```
**********************************************************
    4847 Wed Aug 13 19:52:02 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/d2i_pu.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/d2i_pu.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/asn1.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_EC
#include <openssl/ec.h>
#endif

EVP_PKEY *d2i_PublicKey(int type, EVP_PKEY **a, const unsigned char **pp,
             long length)
        {
        EVP_PKEY *ret;

        if ((a == NULL) || (*a == NULL))
                {
                if ((ret=EVP_PKEY_new()) == NULL)
                        {
                        ASN1err(ASN1_F_D2I_PUBLICKEY,ERR_R_EVP_LIB);
                        return(NULL);
                        }
                }
        else    ret= *a;

        if (!EVP_PKEY_set_type(ret, type))
                {
                ASN1err(ASN1_F_D2I_PUBLICKEY,ERR_R_EVP_LIB);
                goto err;
                }

        switch (EVP_PKEY_id(ret))
                {
#ifndef OPENSSL_NO_RSA
        case EVP_PKEY_RSA:
                if ((ret->pkey.rsa=d2i_RSAPublicKey(NULL,
                        (const unsigned char **)pp,length)) == NULL) /* TMP UGLY
                        {
                        ASN1err(ASN1_F_D2I_PUBLICKEY,ERR_R_ASN1_LIB);
                        goto err;
                        }
                break;
#endif
#ifndef OPENSSL_NO_DSA
        case EVP_PKEY_DSA:
                if (!d2i_DSAPublicKey(&(ret->pkey.dsa),
                        (const unsigned char **)pp,length)) /* TMP UGLY CAST */
                        {
                        ASN1err(ASN1_F_D2I_PUBLICKEY,ERR_R_ASN1_LIB);
                        goto err;
                        }
                break;
#endif
#ifndef OPENSSL_NO_EC
        case EVP_PKEY_EC:
                if (!o2i_ECPublicKey(&(ret->pkey.ec),
                                (const unsigned char **)pp, length))
                        {
                        ASN1err(ASN1_F_D2I_PUBLICKEY, ERR_R_ASN1_LIB);
                        goto err;
                        }
        break;
#endif
```

```
128         default:
129                 ASN1err(ASN1_F_D2I_PUBLICKEY,ASN1_R_UNKNOWN_PUBLIC_KEY_TYPE);
130                 goto err;
131                 /* break; */
132                 }
133         if (a != NULL) (*a)=ret;
134         return(ret);
135 err:
136         if ((ret != NULL) && ((a == NULL) || (*a != ret))) EVP_PKEY_free(ret);
137         return(NULL);
138         }
139 #endif /* ! codereview */
```

```
**********************************************************
    6285 Wed Aug 13 19:52:02 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/evp_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/evp_asn1.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#include <openssl/asn1_mac.h>

int ASN1_TYPE_set_octetstring(ASN1_TYPE *a, unsigned char *data, int len)
        {
        ASN1_STRING *os;

        if ((os=M_ASN1_OCTET_STRING_new()) == NULL) return(0);
        if (!M_ASN1_OCTET_STRING_set(os,data,len))
                {
                M_ASN1_OCTET_STRING_free(os);
                return 0;
                }
        ASN1_TYPE_set(a,V_ASN1_OCTET_STRING,os);
        return(1);
        }

/* int max_len:  for returned value    */
int ASN1_TYPE_get_octetstring(ASN1_TYPE *a, unsigned char *data,
             int max_len)
        {
        int ret,num;
        unsigned char *p;

        if ((a->type != V_ASN1_OCTET_STRING) || (a->value.octet_string == NULL))
                {
                ASN1err(ASN1_F_ASN1_TYPE_GET_OCTETSTRING,ASN1_R_DATA_IS_WRONG);
                return(-1);
                }
        p=M_ASN1_STRING_data(a->value.octet_string);
        ret=M_ASN1_STRING_length(a->value.octet_string);
        if (ret < max_len)
                num=ret;
        else
                num=max_len;
        memcpy(data,p,num);
        return(ret);
        }

int ASN1_TYPE_set_int_octetstring(ASN1_TYPE *a, long num, unsigned char *data,
             int len)
        {
        int n,size;
        ASN1_OCTET_STRING os,*osp;
        ASN1_INTEGER in;
        unsigned char *p;
        unsigned char buf[32]; /* when they have 256bit longs,
                                * I'll be in trouble */
        in.data=buf;
        in.length=32;
        os.data=data;
        os.type=V_ASN1_OCTET_STRING;
        os.length=len;
        ASN1_INTEGER_set(&in,num);
        n =  i2d_ASN1_INTEGER(&in,NULL);
        n+=M_i2d_ASN1_OCTET_STRING(&os,NULL);

        size=ASN1_object_size(1,n,V_ASN1_SEQUENCE);

        if ((osp=ASN1_STRING_new()) == NULL) return(0);
        /* Grow the 'string' */
        if (!ASN1_STRING_set(osp,NULL,size))
                {
                ASN1_STRING_free(osp);
                return(0);
                }
```

```
128         M_ASN1_STRING_length_set(osp, size);
129         p=M_ASN1_STRING_data(osp);

131         ASN1_put_object(&p,1,n,V_ASN1_SEQUENCE,V_ASN1_UNIVERSAL);
132           i2d_ASN1_INTEGER(&in,&p);
133         M_i2d_ASN1_OCTET_STRING(&os,&p);

135         ASN1_TYPE_set(a,V_ASN1_SEQUENCE,osp);
136         return(1);
137         }

139 /* we return the actual length..., num may be missing, in which
140  * case, set it to zero */
141 /* int max_len:  for returned value    */
142 int ASN1_TYPE_get_int_octetstring(ASN1_TYPE *a, long *num, unsigned char *data,
143              int max_len)
144         {
145         int ret= -1,n;
146         ASN1_INTEGER *ai=NULL;
147         ASN1_OCTET_STRING *os=NULL;
148         const unsigned char *p;
149         long length;
150         ASN1_const_CTX c;

152         if ((a->type != V_ASN1_SEQUENCE) || (a->value.sequence == NULL))
153                 {
154                 goto err;
155                 }
156         p=M_ASN1_STRING_data(a->value.sequence);
157         length=M_ASN1_STRING_length(a->value.sequence);

159         c.pp= &p;
160         c.p=p;
161         c.max=p+length;
162         c.error=ASN1_R_DATA_IS_WRONG;

164         M_ASN1_D2I_start_sequence();
165         c.q=c.p;
166         if ((ai=d2i_ASN1_INTEGER(NULL,&c.p,c.slen)) == NULL) goto err;
167         c.slen-=(c.p-c.q);
168         c.q=c.p;
169         if ((os=d2i_ASN1_OCTET_STRING(NULL,&c.p,c.slen)) == NULL) goto err;
170         c.slen-=(c.p-c.q);
171         if (!M_ASN1_D2I_end_sequence()) goto err;

173         if (num != NULL)
174                 *num=ASN1_INTEGER_get(ai);

176         ret=M_ASN1_STRING_length(os);
177         if (max_len > ret)
178                 n=ret;
179         else
180                 n=max_len;

182         if (data != NULL)
183                 memcpy(data,M_ASN1_STRING_data(os),n);
184         if (0)
185                 {
186 err:
187                 ASN1err(ASN1_F_ASN1_TYPE_GET_INT_OCTETSTRING,ASN1_R_DATA_IS_WRON
188                 }
189         if (os != NULL) M_ASN1_OCTET_STRING_free(os);
190         if (ai != NULL) M_ASN1_INTEGER_free(ai);
191         return(ret);
192         }
193 #endif /* ! codereview */
```

```
**********************************************************
    5960 Wed Aug 13 19:52:02 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/f_enum.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/f_enum.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/asn1.h>

/* Based on a_int.c: equivalent ENUMERATED functions */

int i2a_ASN1_ENUMERATED(BIO *bp, ASN1_ENUMERATED *a)
        {
        int i,n=0;
        static const char *h="0123456789ABCDEF";
        char buf[2];

        if (a == NULL) return(0);

        if (a->length == 0)
                {
                if (BIO_write(bp,"00",2) != 2) goto err;
                n=2;
                }
        else
                {
                for (i=0; i<a->length; i++)
                        {
                        if ((i != 0) && (i%35 == 0))
                                {
                                if (BIO_write(bp,"\\\n",2) != 2) goto err;
                                n+=2;
                                }
                        buf[0]=h[((unsigned char)a->data[i]>>4)&0x0f];
                        buf[1]=h[((unsigned char)a->data[i]   )&0x0f];
                        if (BIO_write(bp,buf,2) != 2) goto err;
                        n+=2;
                        }
                }
        return(n);
err:
        return(-1);
        }

int a2i_ASN1_ENUMERATED(BIO *bp, ASN1_ENUMERATED *bs, char *buf, int size)
        {
        int ret=0;
        int i,j,k,m,n,again,bufsize;
        unsigned char *s=NULL,*sp;
        unsigned char *bufp;
        int num=0,slen=0,first=1;

        bs->type=V_ASN1_ENUMERATED;

        bufsize=BIO_gets(bp,buf,size);
        for (;;)
                {
                if (bufsize < 1) goto err_sl;
                i=bufsize;
                if (buf[i-1] == '\n') buf[--i]='\0';
                if (i == 0) goto err_sl;
                if (buf[i-1] == '\r') buf[--i]='\0';
                if (i == 0) goto err_sl;
                again=(buf[i-1] == '\\');

                for (j=0; j<i; j++)
                        {
                        if (!(  ((buf[j] >= '0') && (buf[j] <= '9')) ||
                                ((buf[j] >= 'a') && (buf[j] <= 'f')) ||
                                ((buf[j] >= 'A') && (buf[j] <= 'F'))))
                                {
                                i=j;
                                break;
```

```
128                                 }
129                         }
130                 buf[i]='\0';
131                 /* We have now cleared all the crap off the end of the
132                  * line */
133                 if (i < 2) goto err_sl;

135                 bufp=(unsigned char *)buf;
136                 if (first)
137                         {
138                         first=0;
139                         if ((bufp[0] == '0') && (buf[1] == '0'))
140                                 {
141                                 bufp+=2;
142                                 i-=2;
143                                 }
144                         }
145                 k=0;
146                 i-=again;
147                 if (i%2 != 0)
148                         {
149                         ASN1err(ASN1_F_A2I_ASN1_ENUMERATED,ASN1_R_ODD_NUMBER_OF_
150                         goto err;
151                         }
152                 i/=2;
153                 if (num+i > slen)
154                         {
155                         if (s == NULL)
156                                 sp=(unsigned char *)OPENSSL_malloc(
157                                         (unsigned int)num+i*2);
158                         else
159                                 sp=(unsigned char *)OPENSSL_realloc(s,
160                                         (unsigned int)num+i*2);
161                         if (sp == NULL)
162                                 {
163                                 ASN1err(ASN1_F_A2I_ASN1_ENUMERATED,ERR_R_MALLOC_
164                                 if (s != NULL) OPENSSL_free(s);
165                                 goto err;
166                                 }
167                         s=sp;
168                         slen=num+i*2;
169                         }
170                 for (j=0; j<i; j++,k+=2)
171                         {
172                         for (n=0; n<2; n++)
173                                 {
174                                 m=bufp[k+n];
175                                 if ((m >= '0') && (m <= '9'))
176                                         m-='0';
177                                 else if ((m >= 'a') && (m <= 'f'))
178                                         m=m-'a'+10;
179                                 else if ((m >= 'A') && (m <= 'F'))
180                                         m=m-'A'+10;
181                                 else
182                                         {
183                                         ASN1err(ASN1_F_A2I_ASN1_ENUMERATED,ASN1_
184                                         goto err;
185                                         }
186                                 s[num+j]<<=4;
187                                 s[num+j]|=m;
188                                 }
189                         }
190                 num+=i;
191                 if (again)
192                         bufsize=BIO_gets(bp,buf,size);
193                 else
```



```
194                                 break;
195                         }
196                 bs->length=num;
197                 bs->data=s;
198                 ret=1;
199         err:
200                 if (0)
201                         {
202         err_sl:
203                         ASN1err(ASN1_F_A2I_ASN1_ENUMERATED,ASN1_R_SHORT_LINE);
204                         }
205                 return(ret);
206                 }
207         #endif /* ! codereview */
```

```
**********************************************************
    6215 Wed Aug 13 19:52:03 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/f_int.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/f_int.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/asn1.h>

int i2a_ASN1_INTEGER(BIO *bp, ASN1_INTEGER *a)
        {
        int i,n=0;
        static const char *h="0123456789ABCDEF";
        char buf[2];

        if (a == NULL) return(0);

        if (a->type & V_ASN1_NEG)
                {
                if (BIO_write(bp, "-", 1) != 1) goto err;
                n = 1;
                }

        if (a->length == 0)
                {
                if (BIO_write(bp,"00",2) != 2) goto err;
                n += 2;
                }
        else
                {
                for (i=0; i<a->length; i++)
                        {
                        if ((i != 0) && (i%35 == 0))
                                {
                                if (BIO_write(bp,"\\\n",2) != 2) goto err;
                                n+=2;
                                }
                        buf[0]=h[((unsigned char)a->data[i]>>4)&0x0f];
                        buf[1]=h[((unsigned char)a->data[i]   )&0x0f];
                        if (BIO_write(bp,buf,2) != 2) goto err;
                        n+=2;
                        }
                }
        return(n);
err:
        return(-1);
        }

int a2i_ASN1_INTEGER(BIO *bp, ASN1_INTEGER *bs, char *buf, int size)
        {
        int ret=0;
        int i,j,k,m,n,again,bufsize;
        unsigned char *s=NULL,*sp;
        unsigned char *bufp;
        int num=0,slen=0,first=1;

        bs->type=V_ASN1_INTEGER;

        bufsize=BIO_gets(bp,buf,size);
        for (;;)
                {
                if (bufsize < 1) goto err_sl;
                i=bufsize;
                if (buf[i-1] == '\n') buf[--i]='\0';
                if (i == 0) goto err_sl;
                if (buf[i-1] == '\r') buf[--i]='\0';
                if (i == 0) goto err_sl;
                again=(buf[i-1] == '\\');

                for (j=0; j<i; j++)
                        {
#ifndef CHARSET_EBCDIC
                        if (!(  ((buf[j] >= '0') && (buf[j] <= '9')) ||
```

```
128                                 ((buf[j] >= 'a') && (buf[j] <= 'f')) ||
129                                 ((buf[j] >= 'A') && (buf[j] <= 'F'))))
130 #else
131                         /* This #ifdef is not strictly necessary, since
132                          * the characters A...F a...f 0...9 are contiguous
133                          * (yes, even in EBCDIC - but not the whole alphabet).
134                          * Nevertheless, isxdigit() is faster.
135                          */
136                         if (!isxdigit(buf[j]))
137 #endif
138                                 {
139                                 i=j;
140                                 break;
141                                 }
142                         }
143                 buf[i]='\0';
144                 /* We have now cleared all the crap off the end of the
145                  * line */
146                 if (i < 2) goto err_sl;
147
148                 bufp=(unsigned char *)buf;
149                 if (first)
150                         {
151                         first=0;
152                         if ((bufp[0] == '0') && (buf[1] == '0'))
153                                 {
154                                 bufp+=2;
155                                 i-=2;
156                                 }
157                         }
158                 k=0;
159                 i-=again;
160                 if (i%2 != 0)
161                         {
162                         ASN1err(ASN1_F_A2I_ASN1_INTEGER,ASN1_R_ODD_NUMBER_OF_CHA
163                         goto err;
164                         }
165                 i/=2;
166                 if (num+i > slen)
167                         {
168                         if (s == NULL)
169                                 sp=(unsigned char *)OPENSSL_malloc(
170                                         (unsigned int)num+i*2);
171                         else
172                                 sp=OPENSSL_realloc_clean(s,slen,num+i*2);
173                         if (sp == NULL)
174                                 {
175                                 ASN1err(ASN1_F_A2I_ASN1_INTEGER,ERR_R_MALLOC_FAI
176                                 if (s != NULL) OPENSSL_free(s);
177                                 goto err;
178                                 }
179                         s=sp;
180                         slen=num+i*2;
181                         }
182                 for (j=0; j<i; j++,k+=2)
183                         {
184                         for (n=0; n<2; n++)
185                                 {
186                                 m=bufp[k+n];
187                                 if ((m >= '0') && (m <= '9'))
188                                         m-='0';
189                                 else if ((m >= 'a') && (m <= 'f'))
190                                         m=m-'a'+10;
191                                 else if ((m >= 'A') && (m <= 'F'))
192                                         m=m-'A'+10;
193                                 else
```



```
194                                         {
195                                         ASN1err(ASN1_F_A2I_ASN1_INTEGER,ASN1_R_N
196                                         goto err;
197                                         }
198                                 s[num+j]<<=4;
199                                 s[num+j]|=m;
200                                 }
201                         }
202                 num+=i;
203                 if (again)
204                         bufsize=BIO_gets(bp,buf,size);
205                 else
206                         break;
207                 }
208         bs->length=num;
209         bs->data=s;
210         ret=1;
211 err:
212         if (0)
213                 {
214 err_sl:
215                 ASN1err(ASN1_F_A2I_ASN1_INTEGER,ASN1_R_SHORT_LINE);
216                 }
217         return(ret);
218         }
219 #endif /* ! codereview */
```

```
**********************************************************
    6074 Wed Aug 13 19:52:03 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/f_string.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/f_string.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/asn1.h>

int i2a_ASN1_STRING(BIO *bp, ASN1_STRING *a, int type)
        {
        int i,n=0;
        static const char *h="0123456789ABCDEF";
        char buf[2];

        if (a == NULL) return(0);

        if (a->length == 0)
                {
                if (BIO_write(bp,"0",1) != 1) goto err;
                n=1;
                }
        else
                {
                for (i=0; i<a->length; i++)
                        {
                        if ((i != 0) && (i%35 == 0))
                                {
                                if (BIO_write(bp,"\\\n",2) != 2) goto err;
                                n+=2;
                                }
                        buf[0]=h[((unsigned char)a->data[i]>>4)&0x0f];
                        buf[1]=h[((unsigned char)a->data[i]   )&0x0f];
                        if (BIO_write(bp,buf,2) != 2) goto err;
                        n+=2;
                        }
                }
        return(n);
err:
        return(-1);
        }

int a2i_ASN1_STRING(BIO *bp, ASN1_STRING *bs, char *buf, int size)
        {
        int ret=0;
        int i,j,k,m,n,again,bufsize;
        unsigned char *s=NULL,*sp;
        unsigned char *bufp;
        int num=0,slen=0,first=1;

        bufsize=BIO_gets(bp,buf,size);
        for (;;)
                {
                if (bufsize < 1)
                        {
                        if (first)
                                break;
                        else
                                goto err_sl;
                        }
                first=0;

                i=bufsize;
                if (buf[i-1] == '\n') buf[--i]='\0';
                if (i == 0) goto err_sl;
                if (buf[i-1] == '\r') buf[--i]='\0';
                if (i == 0) goto err_sl;
                again=(buf[i-1] == '\\');

                for (j=i-1; j>0; j--)
                        {
#ifndef CHARSET_EBCDIC
                        if (!(  ((buf[j] >= '0') && (buf[j] <= '9')) ||
```

```
128                                 ((buf[j] >= 'a') && (buf[j] <= 'f')) ||
129                                 ((buf[j] >= 'A') && (buf[j] <= 'F'))))
130 #else
131                         /* This #ifdef is not strictly necessary, since
132                          * the characters A...F a...f 0...9 are contiguous
133                          * (yes, even in EBCDIC - but not the whole alphabet).
134                          * Nevertheless, isxdigit() is faster.
135                          */
136                         if (!isxdigit(buf[j]))
137 #endif
138                                 {
139                                 i=j;
140                                 break;
141                                 }
142                         }
143                 buf[i]='\0';
144                 /* We have now cleared all the crap off the end of the
145                  * line */
146                 if (i < 2) goto err_sl;

148                 bufp=(unsigned char *)buf;

150                 k=0;
151                 i-=again;
152                 if (i%2 != 0)
153                         {
154                         ASN1err(ASN1_F_A2I_ASN1_STRING,ASN1_R_ODD_NUMBER_OF_CHAR
155                         goto err;
156                         }
157                 i/=2;
158                 if (num+i > slen)
159                         {
160                         if (s == NULL)
161                                 sp=(unsigned char *)OPENSSL_malloc(
162                                         (unsigned int)num+i*2);
163                         else
164                                 sp=(unsigned char *)OPENSSL_realloc(s,
165                                         (unsigned int)num+i*2);
166                         if (sp == NULL)
167                                 {
168                                 ASN1err(ASN1_F_A2I_ASN1_STRING,ERR_R_MALLOC_FAIL
169                                 if (s != NULL) OPENSSL_free(s);
170                                 goto err;
171                                 }
172                         s=sp;
173                         slen=num+i*2;
174                         }
175                 for (j=0; j<i; j++,k+=2)
176                         {
177                         for (n=0; n<2; n++)
178                                 {
179                                 m=bufp[k+n];
180                                 if ((m >= '0') && (m <= '9'))
181                                         m-='0';
182                                 else if ((m >= 'a') && (m <= 'f'))
183                                         m=m-'a'+10;
184                                 else if ((m >= 'A') && (m <= 'F'))
185                                         m=m-'A'+10;
186                                 else
187                                         {
188                                         ASN1err(ASN1_F_A2I_ASN1_STRING,ASN1_R_NO
189                                         goto err;
190                                         }
191                                 s[num+j]<<=4;
192                                 s[num+j]|=m;
193                                 }
```



```
194                                 }
195                         num+=i;
196                         if (again)
197                                 bufsize=BIO_gets(bp,buf,size);
198                         else
199                                 break;
200                         }
201                 bs->length=num;
202                 bs->data=s;
203                 ret=1;
204 err:
205                 if (0)
206                         {
207 err_sl:
208                         ASN1err(ASN1_F_A2I_ASN1_STRING,ASN1_R_SHORT_LINE);
209                         }
210                 return(ret);
211                 }
212 #endif /* ! codereview */
```

```
**********************************************************
    3723 Wed Aug 13 19:52:03 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/i2d_pr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/i2d_pr.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/x509.h>
#include "asn1_locl.h"

int i2d_PrivateKey(EVP_PKEY *a, unsigned char **pp)
        {
        if (a->ameth && a->ameth->old_priv_encode)
                {
                return a->ameth->old_priv_encode(a, pp);
                }
        if (a->ameth && a->ameth->priv_encode) {
                PKCS8_PRIV_KEY_INFO *p8 = EVP_PKEY2PKCS8(a);
                int ret = i2d_PKCS8_PRIV_KEY_INFO(p8,pp);
                PKCS8_PRIV_KEY_INFO_free(p8);
                return ret;
        }
        ASN1err(ASN1_F_I2D_PRIVATEKEY,ASN1_R_UNSUPPORTED_PUBLIC_KEY_TYPE);
        return(-1);
        }
#endif /* ! codereview */
```

```
**********************************************************
    3928 Wed Aug 13 19:52:03 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/i2d_pu.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/i2d_pu.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/evp.h>
#include <openssl/objects.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_EC
#include <openssl/ec.h>
#endif

int i2d_PublicKey(EVP_PKEY *a, unsigned char **pp)
        {
        switch (a->type)
                {
#ifndef OPENSSL_NO_RSA
        case EVP_PKEY_RSA:
                return(i2d_RSAPublicKey(a->pkey.rsa,pp));
#endif
#ifndef OPENSSL_NO_DSA
        case EVP_PKEY_DSA:
                return(i2d_DSAPublicKey(a->pkey.dsa,pp));
#endif
#ifndef OPENSSL_NO_EC
        case EVP_PKEY_EC:
                return(i2o_ECPublicKey(a->pkey.ec, pp));
#endif
        default:
                ASN1err(ASN1_F_I2D_PUBLICKEY,ASN1_R_UNSUPPORTED_PUBLIC_KEY_TYPE)
                return(-1);
                }
        }
#endif /* ! codereview */
```

```
**********************************************************
   10575 Wed Aug 13 19:52:03 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/n_pkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/n_pkey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#ifndef OPENSSL_NO_RSA
```



```
#include <openssl/rsa.h>
#include <openssl/objects.h>
#include <openssl/asn1t.h>
#include <openssl/asn1_mac.h>
#include <openssl/evp.h>
#include <openssl/x509.h>


#ifndef OPENSSL_NO_RC4

typedef struct netscape_pkey_st
        {
        long version;
        X509_ALGOR *algor;
        ASN1_OCTET_STRING *private_key;
        } NETSCAPE_PKEY;

typedef struct netscape_encrypted_pkey_st
        {
        ASN1_OCTET_STRING *os;
        /* This is the same structure as DigestInfo so use it:
         * although this isn't really anything to do with
         * digests.
         */
        X509_SIG *enckey;
        } NETSCAPE_ENCRYPTED_PKEY;


ASN1_BROKEN_SEQUENCE(NETSCAPE_ENCRYPTED_PKEY) = {
        ASN1_SIMPLE(NETSCAPE_ENCRYPTED_PKEY, os, ASN1_OCTET_STRING),
        ASN1_SIMPLE(NETSCAPE_ENCRYPTED_PKEY, enckey, X509_SIG)
} ASN1_BROKEN_SEQUENCE_END(NETSCAPE_ENCRYPTED_PKEY)

DECLARE_ASN1_FUNCTIONS_const(NETSCAPE_ENCRYPTED_PKEY)
DECLARE_ASN1_ENCODE_FUNCTIONS_const(NETSCAPE_ENCRYPTED_PKEY,NETSCAPE_ENCRYPTED_P
IMPLEMENT_ASN1_FUNCTIONS_const(NETSCAPE_ENCRYPTED_PKEY)

ASN1_SEQUENCE(NETSCAPE_PKEY) = {
        ASN1_SIMPLE(NETSCAPE_PKEY, version, LONG),
        ASN1_SIMPLE(NETSCAPE_PKEY, algor, X509_ALGOR),
        ASN1_SIMPLE(NETSCAPE_PKEY, private_key, ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END(NETSCAPE_PKEY)

DECLARE_ASN1_FUNCTIONS_const(NETSCAPE_PKEY)
DECLARE_ASN1_ENCODE_FUNCTIONS_const(NETSCAPE_PKEY,NETSCAPE_PKEY)
IMPLEMENT_ASN1_FUNCTIONS_const(NETSCAPE_PKEY)

static RSA *d2i_RSA_NET_2(RSA **a, ASN1_OCTET_STRING *os,
                          int (*cb)(char *buf, int len, const char *prompt,
                                    int verify),
                          int sgckey);

int i2d_Netscape_RSA(const RSA *a, unsigned char **pp,
                     int (*cb)(char *buf, int len, const char *prompt,
                               int verify))
{
        return i2d_RSA_NET(a, pp, cb, 0);
}

int i2d_RSA_NET(const RSA *a, unsigned char **pp,
                int (*cb)(char *buf, int len, const char *prompt, int verify),
                int sgckey)
        {
        int i, j, ret = 0;
        int rsalen, pkeylen, olen;
        NETSCAPE_PKEY *pkey = NULL;
```

```
128         NETSCAPE_ENCRYPTED_PKEY *enckey = NULL;
129         unsigned char buf[256],*zz;
130         unsigned char key[EVP_MAX_KEY_LENGTH];
131         EVP_CIPHER_CTX ctx;
132         EVP_CIPHER_CTX_init(&ctx);

134         if (a == NULL) return(0);

136         if ((pkey=NETSCAPE_PKEY_new()) == NULL) goto err;
137         if ((enckey=NETSCAPE_ENCRYPTED_PKEY_new()) == NULL) goto err;
138         pkey->version = 0;

140         pkey->algor->algorithm=OBJ_nid2obj(NID_rsaEncryption);
141         if ((pkey->algor->parameter=ASN1_TYPE_new()) == NULL) goto err;
142         pkey->algor->parameter->type=V_ASN1_NULL;

144         rsalen = i2d_RSAPrivateKey(a, NULL);

146         /* Fake some octet strings just for the initial length
147          * calculation.
148          */

150         pkey->private_key->length=rsalen;

152         pkeylen=i2d_NETSCAPE_PKEY(pkey,NULL);

154         enckey->enckey->digest->length = pkeylen;

156         enckey->os->length = 11;        /* "private-key" */

158         enckey->enckey->algor->algorithm=OBJ_nid2obj(NID_rc4);
159         if ((enckey->enckey->algor->parameter=ASN1_TYPE_new()) == NULL) goto err
160         enckey->enckey->algor->parameter->type=V_ASN1_NULL;

162         if (pp == NULL)
163                 {
164                 olen = i2d_NETSCAPE_ENCRYPTED_PKEY(enckey, NULL);
165                 NETSCAPE_PKEY_free(pkey);
166                 NETSCAPE_ENCRYPTED_PKEY_free(enckey);
167                 return olen;
168                 }


171         /* Since its RC4 encrypted length is actual length */
172         if ((zz=(unsigned char *)OPENSSL_malloc(rsalen)) == NULL)
173                 {
174                 ASN1err(ASN1_F_I2D_RSA_NET,ERR_R_MALLOC_FAILURE);
175                 goto err;
176                 }

178         pkey->private_key->data = zz;
179         /* Write out private key encoding */
180         i2d_RSAPrivateKey(a,&zz);

182         if ((zz=OPENSSL_malloc(pkeylen)) == NULL)
183                 {
184                 ASN1err(ASN1_F_I2D_RSA_NET,ERR_R_MALLOC_FAILURE);
185                 goto err;
186                 }

188         if (!ASN1_STRING_set(enckey->os, "private-key", -1))
189                 {
190                 ASN1err(ASN1_F_I2D_RSA_NET,ERR_R_MALLOC_FAILURE);
191                 goto err;
192                 }
193         enckey->enckey->digest->data = zz;
```



```
194         i2d_NETSCAPE_PKEY(pkey,&zz);

196         /* Wipe the private key encoding */
197         OPENSSL_cleanse(pkey->private_key->data, rsalen);

199         if (cb == NULL)
200                 cb=EVP_read_pw_string;
201         i=cb((char *)buf,256,"Enter Private Key password:",1);
202         if (i != 0)
203                 {
204                 ASN1err(ASN1_F_I2D_RSA_NET,ASN1_R_BAD_PASSWORD_READ);
205                 goto err;
206                 }
207         i = strlen((char *)buf);
208         /* If the key is used for SGC the algorithm is modified a little. */
209         if(sgckey) {
210                 if (!EVP_Digest(buf, i, buf, NULL, EVP_md5(), NULL))
211                         goto err;
212                 memcpy(buf + 16, "SGCKEYSALT", 10);
213                 i = 26;
214         }

216         if (!EVP_BytesToKey(EVP_rc4(),EVP_md5(),NULL,buf,i,1,key,NULL))
217                 goto err;
218         OPENSSL_cleanse(buf,256);

220         /* Encrypt private key in place */
221         zz = enckey->enckey->digest->data;
222         if (!EVP_EncryptInit_ex(&ctx,EVP_rc4(),NULL,key,NULL))
223                 goto err;
224         if (!EVP_EncryptUpdate(&ctx,zz,&i,zz,pkeylen))
225                 goto err;
226         if (!EVP_EncryptFinal_ex(&ctx,zz + i,&j))
227                 goto err;

229         ret = i2d_NETSCAPE_ENCRYPTED_PKEY(enckey, pp);
230 err:
231         EVP_CIPHER_CTX_cleanup(&ctx);
232         NETSCAPE_ENCRYPTED_PKEY_free(enckey);
233         NETSCAPE_PKEY_free(pkey);
234         return(ret);
235         }


238 RSA *d2i_Netscape_RSA(RSA **a, const unsigned char **pp, long length,
239                       int (*cb)(char *buf, int len, const char *prompt,
240                                 int verify))
241 {
242         return d2i_RSA_NET(a, pp, length, cb, 0);
243 }

245 RSA *d2i_RSA_NET(RSA **a, const unsigned char **pp, long length,
246                  int (*cb)(char *buf, int len, const char *prompt, int verify),
247                  int sgckey)
248         {
249         RSA *ret=NULL;
250         const unsigned char *p;
251         NETSCAPE_ENCRYPTED_PKEY *enckey = NULL;

253         p = *pp;

255         enckey = d2i_NETSCAPE_ENCRYPTED_PKEY(NULL, &p, length);
256         if(!enckey) {
257                 ASN1err(ASN1_F_D2I_RSA_NET,ASN1_R_DECODING_ERROR);
258                 return NULL;
259         }
```

```
261         if ((enckey->os->length != 11) || (strncmp("private-key",
262                 (char *)enckey->os->data,11) != 0))
263                 {
264                 ASN1err(ASN1_F_D2I_RSA_NET,ASN1_R_PRIVATE_KEY_HEADER_MISSING);
265                 NETSCAPE_ENCRYPTED_PKEY_free(enckey);
266                 return NULL;
267                 }
268         if (OBJ_obj2nid(enckey->enckey->algor->algorithm) != NID_rc4)
269                 {
270                 ASN1err(ASN1_F_D2I_RSA_NET,ASN1_R_UNSUPPORTED_ENCRYPTION_ALGORIT
271                 goto err;
272         }
273         if (cb == NULL)
274                 cb=EVP_read_pw_string;
275         if ((ret=d2i_RSA_NET_2(a, enckey->enckey->digest,cb, sgckey)) == NULL) g

277         *pp = p;

279         err:
280         NETSCAPE_ENCRYPTED_PKEY_free(enckey);
281         return ret;

283         }

285 static RSA *d2i_RSA_NET_2(RSA **a, ASN1_OCTET_STRING *os,
286                           int (*cb)(char *buf, int len, const char *prompt,
287                                     int verify), int sgckey)
288         {
289         NETSCAPE_PKEY *pkey=NULL;
290         RSA *ret=NULL;
291         int i,j;
292         unsigned char buf[256];
293         const unsigned char *zz;
294         unsigned char key[EVP_MAX_KEY_LENGTH];
295         EVP_CIPHER_CTX ctx;
296         EVP_CIPHER_CTX_init(&ctx);

298         i=cb((char *)buf,256,"Enter Private Key password:",0);
299         if (i != 0)
300                 {
301                 ASN1err(ASN1_F_D2I_RSA_NET_2,ASN1_R_BAD_PASSWORD_READ);
302                 goto err;
303                 }

305         i = strlen((char *)buf);
306         if(sgckey){
307                 if (!EVP_Digest(buf, i, buf, NULL, EVP_md5(), NULL))
308                         goto err;
309                 memcpy(buf + 16, "SGCKEYSALT", 10);
310                 i = 26;
311         }

313         if (!EVP_BytesToKey(EVP_rc4(),EVP_md5(),NULL,buf,i,1,key,NULL))
314                 goto err;
315         OPENSSL_cleanse(buf,256);

317         if (!EVP_DecryptInit_ex(&ctx,EVP_rc4(),NULL, key,NULL))
318                 goto err;
319         if (!EVP_DecryptUpdate(&ctx,os->data,&i,os->data,os->length))
320                 goto err;
321         if (!EVP_DecryptFinal_ex(&ctx,&(os->data[i]),&j))
322                 goto err;
323         os->length=i+j;

325         zz=os->data;
```



```
327         if ((pkey=d2i_NETSCAPE_PKEY(NULL,&zz,os->length)) == NULL)
328                 {
329                 ASN1err(ASN1_F_D2I_RSA_NET_2,ASN1_R_UNABLE_TO_DECODE_RSA_PRIVATE
330                 goto err;
331                 }

333         zz=pkey->private_key->data;
334         if ((ret=d2i_RSAPrivateKey(a,&zz,pkey->private_key->length)) == NULL)
335                 {
336                 ASN1err(ASN1_F_D2I_RSA_NET_2,ASN1_R_UNABLE_TO_DECODE_RSA_KEY);
337                 goto err;
338                 }
339 err:
340         EVP_CIPHER_CTX_cleanup(&ctx);
341         NETSCAPE_PKEY_free(pkey);
342         return(ret);
343         }

345 #endif /* OPENSSL_NO_RC4 */

347 #else /* !OPENSSL_NO_RSA */

349 # if PEDANTIC
350 static void *dummy=&dummy;
351 # endif

353 #endif
354 #endif /* ! codereview */
```

```
**********************************************************
    3488 Wed Aug 13 19:52:03 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/nsseq.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* nsseq.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>
#include <openssl/objects.h>

static int nsseq_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                        void *exarg)
{
        if(operation == ASN1_OP_NEW_POST) {
                NETSCAPE_CERT_SEQUENCE *nsseq;
                nsseq = (NETSCAPE_CERT_SEQUENCE *)*pval;
                nsseq->type = OBJ_nid2obj(NID_netscape_cert_sequence);
        }
        return 1;
}

/* Netscape certificate sequence structure */

ASN1_SEQUENCE_cb(NETSCAPE_CERT_SEQUENCE, nsseq_cb) = {
        ASN1_SIMPLE(NETSCAPE_CERT_SEQUENCE, type, ASN1_OBJECT),
        ASN1_EXP_SEQUENCE_OF_OPT(NETSCAPE_CERT_SEQUENCE, certs, X509, 0)
} ASN1_SEQUENCE_END_cb(NETSCAPE_CERT_SEQUENCE, NETSCAPE_CERT_SEQUENCE)

IMPLEMENT_ASN1_FUNCTIONS(NETSCAPE_CERT_SEQUENCE)
#endif /* ! codereview */
```

```
**********************************************************
    4707 Wed Aug 13 19:52:04 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/p5_pbe.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p5_pbe.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>
#include <openssl/rand.h>

/* PKCS#5 password based encryption structure */

ASN1_SEQUENCE(PBEPARAM) = {
        ASN1_SIMPLE(PBEPARAM, salt, ASN1_OCTET_STRING),
        ASN1_SIMPLE(PBEPARAM, iter, ASN1_INTEGER)
} ASN1_SEQUENCE_END(PBEPARAM)

IMPLEMENT_ASN1_FUNCTIONS(PBEPARAM)


/* Set an algorithm identifier for a PKCS#5 PBE algorithm */

int PKCS5_pbe_set0_algor(X509_ALGOR *algor, int alg, int iter,
                                const unsigned char *salt, int saltlen)
        {
        PBEPARAM *pbe=NULL;
        ASN1_STRING *pbe_str=NULL;
        unsigned char *sstr;

        pbe = PBEPARAM_new();
        if (!pbe)
                {
                ASN1err(ASN1_F_PKCS5_PBE_SET0_ALGOR,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        if(iter <= 0)
                iter = PKCS5_DEFAULT_ITER;
        if (!ASN1_INTEGER_set(pbe->iter, iter))
                {
                ASN1err(ASN1_F_PKCS5_PBE_SET0_ALGOR,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        if (!saltlen)
                saltlen = PKCS5_SALT_LEN;
        if (!ASN1_STRING_set(pbe->salt, NULL, saltlen))
                {
                ASN1err(ASN1_F_PKCS5_PBE_SET0_ALGOR,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        sstr = ASN1_STRING_data(pbe->salt);
        if (salt)
                memcpy(sstr, salt, saltlen);
        else if (RAND_pseudo_bytes(sstr, saltlen) < 0)
                goto err;

        if(!ASN1_item_pack(pbe, ASN1_ITEM_rptr(PBEPARAM), &pbe_str))
                {
                ASN1err(ASN1_F_PKCS5_PBE_SET0_ALGOR,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        PBEPARAM_free(pbe);
        pbe = NULL;

        if (X509_ALGOR_set0(algor, OBJ_nid2obj(alg), V_ASN1_SEQUENCE, pbe_str))
                return 1;

err:
        if (pbe != NULL)
                PBEPARAM_free(pbe);
        if (pbe_str != NULL)
                ASN1_STRING_free(pbe_str);
        return 0;
```

```
128         }

130 /* Return an algorithm identifier for a PKCS#5 PBE algorithm */

132 X509_ALGOR *PKCS5_pbe_set(int alg, int iter,
133                           const unsigned char *salt, int saltlen)
134         {
135         X509_ALGOR *ret;
136         ret = X509_ALGOR_new();
137         if (!ret)
138                 {
139                 ASN1err(ASN1_F_PKCS5_PBE_SET,ERR_R_MALLOC_FAILURE);
140                 return NULL;
141                 }

143         if (PKCS5_pbe_set0_algor(ret, alg, iter, salt, saltlen))
144                 return ret;

146         X509_ALGOR_free(ret);
147         return NULL;
148         }
149 #endif /* ! codereview */
```

```
**********************************************************
    8017 Wed Aug 13 19:52:04 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/p5_pbev2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p5_pbev2.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999-2004.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>
#include <openssl/rand.h>

/* PKCS#5 v2.0 password based encryption structures */

ASN1_SEQUENCE(PBE2PARAM) = {
        ASN1_SIMPLE(PBE2PARAM, keyfunc, X509_ALGOR),
        ASN1_SIMPLE(PBE2PARAM, encryption, X509_ALGOR)
} ASN1_SEQUENCE_END(PBE2PARAM)

IMPLEMENT_ASN1_FUNCTIONS(PBE2PARAM)

ASN1_SEQUENCE(PBKDF2PARAM) = {
        ASN1_SIMPLE(PBKDF2PARAM, salt, ASN1_ANY),
        ASN1_SIMPLE(PBKDF2PARAM, iter, ASN1_INTEGER),
        ASN1_OPT(PBKDF2PARAM, keylength, ASN1_INTEGER),
        ASN1_OPT(PBKDF2PARAM, prf, X509_ALGOR)
} ASN1_SEQUENCE_END(PBKDF2PARAM)

IMPLEMENT_ASN1_FUNCTIONS(PBKDF2PARAM)

/* Return an algorithm identifier for a PKCS#5 v2.0 PBE algorithm:
 * yes I know this is horrible!
 *
 * Extended version to allow application supplied PRF NID and IV.
 */

X509_ALGOR *PKCS5_pbe2_set_iv(const EVP_CIPHER *cipher, int iter,
                              unsigned char *salt, int saltlen,
                              unsigned char *aiv, int prf_nid)
{
        X509_ALGOR *scheme = NULL, *kalg = NULL, *ret = NULL;
        int alg_nid, keylen;
        EVP_CIPHER_CTX ctx;
        unsigned char iv[EVP_MAX_IV_LENGTH];
        PBE2PARAM *pbe2 = NULL;
        ASN1_OBJECT *obj;

        alg_nid = EVP_CIPHER_type(cipher);
        if(alg_nid == NID_undef) {
                ASN1err(ASN1_F_PKCS5_PBE2_SET_IV,
                                ASN1_R_CIPHER_HAS_NO_OBJECT_IDENTIFIER);
                goto err;
        }
        obj = OBJ_nid2obj(alg_nid);

        if(!(pbe2 = PBE2PARAM_new())) goto merr;

        /* Setup the AlgorithmIdentifier for the encryption scheme */
        scheme = pbe2->encryption;

        scheme->algorithm = obj;
        if(!(scheme->parameter = ASN1_TYPE_new())) goto merr;

        /* Create random IV */
        if (EVP_CIPHER_iv_length(cipher))
                {
                if (aiv)
                        memcpy(iv, aiv, EVP_CIPHER_iv_length(cipher));
                else if (RAND_pseudo_bytes(iv, EVP_CIPHER_iv_length(cipher)) < 0
                        goto err;
                }

        EVP_CIPHER_CTX_init(&ctx);

        /* Dummy cipherinit to just setup the IV, and PRF */
```

```
128         if (!EVP_CipherInit_ex(&ctx, cipher, NULL, NULL, iv, 0))
129                 goto err;
130         if(EVP_CIPHER_param_to_asn1(&ctx, scheme->parameter) < 0) {
131                 ASN1err(ASN1_F_PKCS5_PBE2_SET_IV,
132                                         ASN1_R_ERROR_SETTING_CIPHER_PARAMS);
133                 EVP_CIPHER_CTX_cleanup(&ctx);
134                 goto err;
135         }
136         /* If prf NID unspecified see if cipher has a preference.
137          * An error is OK here: just means use default PRF.
138          */
139         if ((prf_nid == -1) &&
140         EVP_CIPHER_CTX_ctrl(&ctx, EVP_CTRL_PBE_PRF_NID, 0, &prf_nid) <= 0)
141                 {
142                 ERR_clear_error();
143                 prf_nid = NID_hmacWithSHA1;
144                 }
145         EVP_CIPHER_CTX_cleanup(&ctx);

147         /* If its RC2 then we'd better setup the key length */

149         if(alg_nid == NID_rc2_cbc)
150                 keylen = EVP_CIPHER_key_length(cipher);
151         else
152                 keylen = -1;

154         /* Setup keyfunc */

156         X509_ALGOR_free(pbe2->keyfunc);

158         pbe2->keyfunc = PKCS5_pbkdf2_set(iter, salt, saltlen, prf_nid, keylen);

160         if (!pbe2->keyfunc)
161                 goto merr;

163         /* Now set up top level AlgorithmIdentifier */

165         if(!(ret = X509_ALGOR_new())) goto merr;
166         if(!(ret->parameter = ASN1_TYPE_new())) goto merr;

168         ret->algorithm = OBJ_nid2obj(NID_pbes2);

170         /* Encode PBE2PARAM into parameter */

172         if(!ASN1_item_pack(pbe2, ASN1_ITEM_rptr(PBE2PARAM),
173                                 &ret->parameter->value.sequence)) goto merr;
174         ret->parameter->type = V_ASN1_SEQUENCE;

176         PBE2PARAM_free(pbe2);
177         pbe2 = NULL;

179         return ret;

181         merr:
182         ASN1err(ASN1_F_PKCS5_PBE2_SET_IV,ERR_R_MALLOC_FAILURE);

184         err:
185         PBE2PARAM_free(pbe2);
186         /* Note 'scheme' is freed as part of pbe2 */
187         X509_ALGOR_free(kalg);
188         X509_ALGOR_free(ret);

190         return NULL;

192 }
```



```
194 X509_ALGOR *PKCS5_pbe2_set(const EVP_CIPHER *cipher, int iter,
195                                 unsigned char *salt, int saltlen)
196         {
197         return PKCS5_pbe2_set_iv(cipher, iter, salt, saltlen, NULL, -1);
198         }

200 X509_ALGOR *PKCS5_pbkdf2_set(int iter, unsigned char *salt, int saltlen,
201                                 int prf_nid, int keylen)
202         {
203         X509_ALGOR *keyfunc = NULL;
204         PBKDF2PARAM *kdf = NULL;
205         ASN1_OCTET_STRING *osalt = NULL;

207         if(!(kdf = PBKDF2PARAM_new()))
208                 goto merr;
209         if(!(osalt = M_ASN1_OCTET_STRING_new()))
210                 goto merr;

212         kdf->salt->value.octet_string = osalt;
213         kdf->salt->type = V_ASN1_OCTET_STRING;

215         if (!saltlen)
216                 saltlen = PKCS5_SALT_LEN;
217         if (!(osalt->data = OPENSSL_malloc (saltlen)))
218                 goto merr;

220         osalt->length = saltlen;

222         if (salt)
223                 memcpy (osalt->data, salt, saltlen);
224         else if (RAND_pseudo_bytes (osalt->data, saltlen) < 0)
225                 goto merr;

227         if(iter <= 0)
228                 iter = PKCS5_DEFAULT_ITER;

230         if(!ASN1_INTEGER_set(kdf->iter, iter))
231                 goto merr;

233         /* If have a key len set it up */

235         if(keylen > 0)
236                 {
237                 if(!(kdf->keylength = M_ASN1_INTEGER_new()))
238                         goto merr;
239                 if(!ASN1_INTEGER_set (kdf->keylength, keylen))
240                         goto merr;
241                 }

243         /* prf can stay NULL if we are using hmacWithSHA1 */
244         if (prf_nid > 0 && prf_nid != NID_hmacWithSHA1)
245                 {
246                 kdf->prf = X509_ALGOR_new();
247                 if (!kdf->prf)
248                         goto merr;
249                 X509_ALGOR_set0(kdf->prf, OBJ_nid2obj(prf_nid),
250                                         V_ASN1_NULL, NULL);
251                 }

253         /* Finally setup the keyfunc structure */

255         keyfunc = X509_ALGOR_new();
256         if (!keyfunc)
257                 goto merr;

259         keyfunc->algorithm = OBJ_nid2obj(NID_id_pbkdf2);
```

```
261         /* Encode PBKDF2PARAM into parameter of pbe2 */

263         if(!(keyfunc->parameter = ASN1_TYPE_new()))
264                 goto merr;

266         if(!ASN1_item_pack(kdf, ASN1_ITEM_rptr(PBKDF2PARAM),
267                            &keyfunc->parameter->value.sequence))
268                 goto merr;
269         keyfunc->parameter->type = V_ASN1_SEQUENCE;

271         PBKDF2PARAM_free(kdf);
272         return keyfunc;

274         merr:
275         ASN1err(ASN1_F_PKCS5_PBKDF2_SET,ERR_R_MALLOC_FAILURE);
276         PBKDF2PARAM_free(kdf);
277         X509_ALGOR_free(keyfunc);
278         return NULL;
279         }
280 #endif /* ! codereview */
```

```
**********************************************************
    5101 Wed Aug 13 19:52:04 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/p8_pkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p8_pkey.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>

/* Minor tweak to operation: zero private key data */
static int pkey_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                        void *exarg)
{
        /* Since the structure must still be valid use ASN1_OP_FREE_PRE */
        if(operation == ASN1_OP_FREE_PRE) {
                PKCS8_PRIV_KEY_INFO *key = (PKCS8_PRIV_KEY_INFO *)*pval;
                if (key->pkey->value.octet_string)
                OPENSSL_cleanse(key->pkey->value.octet_string->data,
                        key->pkey->value.octet_string->length);
        }
        return 1;
}

ASN1_SEQUENCE_cb(PKCS8_PRIV_KEY_INFO, pkey_cb) = {
        ASN1_SIMPLE(PKCS8_PRIV_KEY_INFO, version, ASN1_INTEGER),
        ASN1_SIMPLE(PKCS8_PRIV_KEY_INFO, pkeyalg, X509_ALGOR),
        ASN1_SIMPLE(PKCS8_PRIV_KEY_INFO, pkey, ASN1_ANY),
        ASN1_IMP_SET_OF_OPT(PKCS8_PRIV_KEY_INFO, attributes, X509_ATTRIBUTE, 0)
} ASN1_SEQUENCE_END_cb(PKCS8_PRIV_KEY_INFO, PKCS8_PRIV_KEY_INFO)

IMPLEMENT_ASN1_FUNCTIONS(PKCS8_PRIV_KEY_INFO)

int PKCS8_pkey_set0(PKCS8_PRIV_KEY_INFO *priv, ASN1_OBJECT *aobj,
                                        int version,
                                        int ptype, void *pval,
                                        unsigned char *penc, int penclen)
        {
        unsigned char **ppenc = NULL;
        if (version >= 0)
                {
                if (!ASN1_INTEGER_set(priv->version, version))
                        return 0;
                }
        if (penc)
                {
                int pmtype;
                ASN1_OCTET_STRING *oct;
                oct = ASN1_OCTET_STRING_new();
                if (!oct)
                        return 0;
                oct->data = penc;
                ppenc = &oct->data;
                oct->length = penclen;
                if (priv->broken == PKCS8_NO_OCTET)
                        pmtype = V_ASN1_SEQUENCE;
                else
                        pmtype = V_ASN1_OCTET_STRING;
                ASN1_TYPE_set(priv->pkey, pmtype, oct);
                }
        if (!X509_ALGOR_set0(priv->pkeyalg, aobj, ptype, pval))
                {
                /* If call fails do not swallow 'enc' */
                if (ppenc)
                        *ppenc = NULL;
                return 0;
                }
        return 1;
        }

int PKCS8_pkey_get0(ASN1_OBJECT **ppkalg,
                const unsigned char **pk, int *ppklen,
                X509_ALGOR **pa,
                PKCS8_PRIV_KEY_INFO *p8)
```

```
128         {
129         if (ppkalg)
130                 *ppkalg = p8->pkeyalg->algorithm;
131         if(p8->pkey->type == V_ASN1_OCTET_STRING)
132                 {
133                 p8->broken = PKCS8_OK;
134                 if (pk)
135                         {
136                         *pk = p8->pkey->value.octet_string->data;
137                         *ppklen = p8->pkey->value.octet_string->length;
138                         }
139                 }
140         else if (p8->pkey->type == V_ASN1_SEQUENCE)
141                 {
142                 p8->broken = PKCS8_NO_OCTET;
143                 if (pk)
144                         {
145                         *pk = p8->pkey->value.sequence->data;
146                         *ppklen = p8->pkey->value.sequence->length;
147                         }
148                 }
149         else
150                 return 0;
151         if (pa)
152                 *pa = p8->pkeyalg;
153         return 1;
154         }
155 #endif /* ! codereview */
```

```
**********************************************************
    3745 Wed Aug 13 19:52:04 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/t_bitst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* t_bitst.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/x509v3.h>

int ASN1_BIT_STRING_name_print(BIO *out, ASN1_BIT_STRING *bs,
                               BIT_STRING_BITNAME *tbl, int indent)
{
        BIT_STRING_BITNAME *bnam;
        char first = 1;
        BIO_printf(out, "%*s", indent, "");
        for(bnam = tbl; bnam->lname; bnam++) {
                if(ASN1_BIT_STRING_get_bit(bs, bnam->bitnum)) {
                        if(!first) BIO_puts(out, ", ");
                        BIO_puts(out, bnam->lname);
                        first = 0;
                }
        }
        BIO_puts(out, "\n");
        return 1;
}

int ASN1_BIT_STRING_set_asc(ASN1_BIT_STRING *bs, char *name, int value,
                               BIT_STRING_BITNAME *tbl)
{
        int bitnum;
        bitnum = ASN1_BIT_STRING_num_asc(name, tbl);
        if(bitnum < 0) return 0;
        if(bs) {
                if(!ASN1_BIT_STRING_set_bit(bs, bitnum, value))
                        return 0;
        }
        return 1;
}

int ASN1_BIT_STRING_num_asc(char *name, BIT_STRING_BITNAME *tbl)
{
        BIT_STRING_BITNAME *bnam;
        for(bnam = tbl; bnam->lname; bnam++) {
                if(!strcmp(bnam->sname, name) ||
                        !strcmp(bnam->lname, name) ) return bnam->bitnum;
        }
        return -1;
}
#endif /* ! codereview */
```

```
**********************************************************
    4714 Wed Aug 13 19:52:04 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/t_crl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* t_crl.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/bn.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

#ifndef OPENSSL_NO_FP_API
int X509_CRL_print_fp(FILE *fp, X509_CRL *x)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                X509err(X509_F_X509_CRL_PRINT_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=X509_CRL_print(b, x);
        BIO_free(b);
        return(ret);
        }
#endif

int X509_CRL_print(BIO *out, X509_CRL *x)
{
        STACK_OF(X509_REVOKED) *rev;
        X509_REVOKED *r;
        long l;
        int i;
        char *p;

        BIO_printf(out, "Certificate Revocation List (CRL):\n");
        l = X509_CRL_get_version(x);
        BIO_printf(out, "%8sVersion %lu (0x%lx)\n", "", l+1, l);
        i = OBJ_obj2nid(x->sig_alg->algorithm);
        X509_signature_print(out, x->sig_alg, NULL);
        p=X509_NAME_oneline(X509_CRL_get_issuer(x),NULL,0);
        BIO_printf(out,"%8sIssuer: %s\n","",p);
        OPENSSL_free(p);
        BIO_printf(out,"%8sLast Update: ","");
        ASN1_TIME_print(out,X509_CRL_get_lastUpdate(x));
        BIO_printf(out,"\n%8sNext Update: ","");
        if (X509_CRL_get_nextUpdate(x))
                ASN1_TIME_print(out,X509_CRL_get_nextUpdate(x));
        else BIO_printf(out,"NONE");
        BIO_printf(out,"\n");

        X509V3_extensions_print(out, "CRL extensions",
                                                x->crl->extensions, 0, 8);

        rev = X509_CRL_get_REVOKED(x);

        if(sk_X509_REVOKED_num(rev) > 0)
            BIO_printf(out, "Revoked Certificates:\n");
        else BIO_printf(out, "No Revoked Certificates.\n");

        for(i = 0; i < sk_X509_REVOKED_num(rev); i++) {
                r = sk_X509_REVOKED_value(rev, i);
                BIO_printf(out,"    Serial Number: ");
                i2a_ASN1_INTEGER(out,r->serialNumber);
                BIO_printf(out,"\n        Revocation Date: ");
                ASN1_TIME_print(out,r->revocationDate);
                BIO_printf(out,"\n");
                X509V3_extensions_print(out, "CRL entry extensions",
                                                r->extensions, 0, 8);
        }
```

```
128         X509_signature_print(out, x->sig_alg, x->signature);

130         return 1;

132 }
133 #endif /* ! codereview */
```

```
**********************************************************
    4307 Wed Aug 13 19:52:04 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/t_pkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/t_pkey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/buffer.h>
#include <openssl/bn.h>

int ASN1_bn_print(BIO *bp, const char *number, const BIGNUM *num,
                        unsigned char *buf, int off)
        {
        int n,i;
        const char *neg;

        if (num == NULL) return(1);
        neg = (BN_is_negative(num))?"-":"";
        if(!BIO_indent(bp,off,128))
                return 0;
        if (BN_is_zero(num))
                {
                if (BIO_printf(bp, "%s 0\n", number) <= 0)
                        return 0;
                return 1;
                }

        if (BN_num_bytes(num) <= BN_BYTES)
                {
                if (BIO_printf(bp,"%s %s%lu (%s0x%lx)\n",number,neg,
                        (unsigned long)num->d[0],neg,(unsigned long)num->d[0])
                        <= 0) return(0);
                }
        else
                {
                buf[0]=0;
                if (BIO_printf(bp,"%s%s",number,
                        (neg[0] == '-')?" (Negative)":"") <= 0)
                        return(0);
                n=BN_bn2bin(num,&buf[1]);

                if (buf[1] & 0x80)
                        n++;
                else    buf++;

                for (i=0; i<n; i++)
                        {
                        if ((i%15) == 0)
                                {
                                if(BIO_puts(bp,"\n") <= 0
                                   || !BIO_indent(bp,off+4,128))
                                    return 0;
                                }
                        if (BIO_printf(bp,"%02x%s",buf[i],((i+1) == n)?"":":")
                                <= 0) return(0);
                        }
                if (BIO_write(bp,"\n",1) <= 0) return(0);
                }
        return(1);
        }
#endif /* ! codereview */
```

```
**********************************************************
    8130 Wed Aug 13 19:52:05 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/t_req.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/t_req.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/bn.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif

#ifndef OPENSSL_NO_FP_API
int X509_REQ_print_fp(FILE *fp, X509_REQ *x)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                X509err(X509_F_X509_REQ_PRINT_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=X509_REQ_print(b, x);
        BIO_free(b);
        return(ret);
        }
#endif

int X509_REQ_print_ex(BIO *bp, X509_REQ *x, unsigned long nmflags, unsigned long
        {
        unsigned long l;
        int i;
        const char *neg;
        X509_REQ_INFO *ri;
        EVP_PKEY *pkey;
        STACK_OF(X509_ATTRIBUTE) *sk;
        STACK_OF(X509_EXTENSION) *exts;
        char mlch = ' ';
        int nmindent = 0;

        if((nmflags & XN_FLAG_SEP_MASK) == XN_FLAG_SEP_MULTILINE) {
                mlch = '\n';
                nmindent = 12;
        }

        if(nmflags == X509_FLAG_COMPAT)
                nmindent = 16;


        ri=x->req_info;
        if(!(cflag & X509_FLAG_NO_HEADER))
                {
                if (BIO_write(bp,"Certificate Request:\n",21) <= 0) goto err;
                if (BIO_write(bp,"    Data:\n",10) <= 0) goto err;
                }
        if(!(cflag & X509_FLAG_NO_VERSION))
                {
                neg=(ri->version->type == V_ASN1_NEG_INTEGER)?"-":"";
                l=0;
                for (i=0; i<ri->version->length; i++)
                        { l<<=8; l+=ri->version->data[i]; }
                if(BIO_printf(bp,"%8sVersion: %s%lu (%s0x%lx)\n","",neg,l,neg,
                              l) <= 0)
                    goto err;
                }
```

```
128         if(!(cflag & X509_FLAG_NO_SUBJECT))
129                 {
130                 if (BIO_printf(bp,"        Subject:%c",mlch) <= 0) goto err;
131                 if (X509_NAME_print_ex(bp,ri->subject,nmindent, nmflags) < 0) go
132                 if (BIO_write(bp,"\n",1) <= 0) goto err;
133                 }
134         if(!(cflag & X509_FLAG_NO_PUBKEY))
135                 {
136                 if (BIO_write(bp,"        Subject Public Key Info:\n",33) <= 0)
137                         goto err;
138                 if (BIO_printf(bp,"%12sPublic Key Algorithm: ","") <= 0)
139                         goto err;
140                 if (i2a_ASN1_OBJECT(bp, ri->pubkey->algor->algorithm) <= 0)
141                         goto err;
142                 if (BIO_puts(bp, "\n") <= 0)
143                         goto err;
144
145                 pkey=X509_REQ_get_pubkey(x);
146                 if (pkey == NULL)
147                         {
148                         BIO_printf(bp,"%12sUnable to load Public Key\n","");
149                         ERR_print_errors(bp);
150                         }
151                 else
152                         {
153                         EVP_PKEY_print_public(bp, pkey, 16, NULL);
154                         EVP_PKEY_free(pkey);
155                         }
156                 }
157
158         if(!(cflag & X509_FLAG_NO_ATTRIBUTES))
159                 {
160                 /* may not be */
161                 if(BIO_printf(bp,"%8sAttributes:\n","") <= 0)
162                     goto err;
163
164                 sk=x->req_info->attributes;
165                 if (sk_X509_ATTRIBUTE_num(sk) == 0)
166                         {
167                         if(BIO_printf(bp,"%12sa0:00\n","") <= 0)
168                             goto err;
169                         }
170                 else
171                         {
172                         for (i=0; i<sk_X509_ATTRIBUTE_num(sk); i++)
173                                 {
174                                 ASN1_TYPE *at;
175                                 X509_ATTRIBUTE *a;
176                                 ASN1_BIT_STRING *bs=NULL;
177                                 ASN1_TYPE *t;
178                                 int j,type=0,count=1,ii=0;
179
180                                 a=sk_X509_ATTRIBUTE_value(sk,i);
181                                 if(X509_REQ_extension_nid(OBJ_obj2nid(a->object)
182                                                                         continue
183                                 if(BIO_printf(bp,"%12s","") <= 0)
184                                     goto err;
185                                 if ((j=i2a_ASN1_OBJECT(bp,a->object)) > 0)
186                                 {
187                                 if (a->single)
188                                         {
189                                         t=a->value.single;
190                                         type=t->type;
191                                         bs=t->value.bit_string;
192                                         }
193                                 else
```



```
194                                         {
195                                         ii=0;
196                                         count=sk_ASN1_TYPE_num(a->value.set);
197 get_next:
198                                         at=sk_ASN1_TYPE_value(a->value.set,ii);
199                                         type=at->type;
200                                         bs=at->value.asn1_string;
201                                         }
202                                 }
203                                 for (j=25-j; j>0; j--)
204                                         if (BIO_write(bp," ",1) != 1) goto err;
205                                 if (BIO_puts(bp,":") <= 0) goto err;
206                                 if (    (type == V_ASN1_PRINTABLESTRING) ||
207                                         (type == V_ASN1_T61STRING) ||
208                                         (type == V_ASN1_IA5STRING))
209                                         {
210                                         if (BIO_write(bp,(char *)bs->data,bs->le
211                                                 != bs->length)
212                                                 goto err;
213                                         BIO_puts(bp,"\n");
214                                         }
215                                 else
216                                         {
217                                         BIO_puts(bp,"unable to print attribute\n
218                                         }
219                                 if (++ii < count) goto get_next;
220                                 }
221                         }
222                 }
223         if(!(cflag & X509_FLAG_NO_EXTENSIONS))
224                 {
225                 exts = X509_REQ_get_extensions(x);
226                 if(exts)
227                         {
228                         BIO_printf(bp,"%8sRequested Extensions:\n","");
229                         for (i=0; i<sk_X509_EXTENSION_num(exts); i++)
230                                 {
231                                 ASN1_OBJECT *obj;
232                                 X509_EXTENSION *ex;
233                                 int j;
234                                 ex=sk_X509_EXTENSION_value(exts, i);
235                                 if (BIO_printf(bp,"%12s","") <= 0) goto err;
236                                 obj=X509_EXTENSION_get_object(ex);
237                                 i2a_ASN1_OBJECT(bp,obj);
238                                 j=X509_EXTENSION_get_critical(ex);
239                                 if (BIO_printf(bp,": %s\n",j?"critical":"") <= 0
240                                         goto err;
241                                 if(!X509V3_EXT_print(bp, ex, cflag, 16))
242                                         {
243                                         BIO_printf(bp, "%16s", "");
244                                         M_ASN1_OCTET_STRING_print(bp,ex->value);
245                                         }
246                                 if (BIO_write(bp,"\n",1) <= 0) goto err;
247                                 }
248                         sk_X509_EXTENSION_pop_free(exts, X509_EXTENSION_free);
249                         }
250                 }
251
252         if(!(cflag & X509_FLAG_NO_SIGDUMP))
253                 {
254                 if(!X509_signature_print(bp, x->sig_alg, x->signature)) goto err
255                 }
256
257         return(1);
258 err:
259         X509err(X509_F_X509_REQ_PRINT_EX,ERR_R_BUF_LIB);
```

```
260         return(0);
261         }

263 int X509_REQ_print(BIO *bp, X509_REQ *x)
264         {
265         return X509_REQ_print_ex(bp, x, XN_FLAG_COMPAT, X509_FLAG_COMPAT);
266         }
267 #endif /* ! codereview */
```

```
**********************************************************
    3977 Wed Aug 13 19:52:05 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/t_spki.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* t_spki.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
```



```
#include <openssl/asn1.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#include <openssl/bn.h>

/* Print out an SPKI */

int NETSCAPE_SPKI_print(BIO *out, NETSCAPE_SPKI *spki)
{
        EVP_PKEY *pkey;
        ASN1_IA5STRING *chal;
        int i, n;
        char *s;
        BIO_printf(out, "Netscape SPKI:\n");
        i=OBJ_obj2nid(spki->spkac->pubkey->algor->algorithm);
        BIO_printf(out,"  Public Key Algorithm: %s\n",
                                (i == NID_undef)?"UNKNOWN":OBJ_nid2ln(i));
        pkey = X509_PUBKEY_get(spki->spkac->pubkey);
        if(!pkey) BIO_printf(out, "  Unable to load public key\n");
        else
                {
                EVP_PKEY_print_public(out, pkey, 4, NULL);
                EVP_PKEY_free(pkey);
                }
        chal = spki->spkac->challenge;
        if(chal->length)
                BIO_printf(out, "  Challenge String: %s\n", chal->data);
        i=OBJ_obj2nid(spki->sig_algor->algorithm);
        BIO_printf(out,"  Signature Algorithm: %s",
                                (i == NID_undef)?"UNKNOWN":OBJ_nid2ln(i));

        n=spki->signature->length;
        s=(char *)spki->signature->data;
        for (i=0; i<n; i++)
                {
                if ((i%18) == 0) BIO_write(out,"\n      ",7);
                BIO_printf(out,"%02x%s",(unsigned char)s[i],
                                                ((i+1) == n)?"":":");
                }
        BIO_write(out,"\n",1);
        return 1;
}
#endif /* ! codereview */
```

```
**********************************************************
   13738 Wed Aug 13 19:52:05 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/t_x509.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/t_x509.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/bn.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_EC
#include <openssl/ec.h>
#endif
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include "asn1_locl.h"

#ifndef OPENSSL_NO_FP_API
int X509_print_fp(FILE *fp, X509 *x)
        {
        return X509_print_ex_fp(fp, x, XN_FLAG_COMPAT, X509_FLAG_COMPAT);
        }

int X509_print_ex_fp(FILE *fp, X509 *x, unsigned long nmflag, unsigned long cfla
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                X509err(X509_F_X509_PRINT_EX_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=X509_print_ex(b, x, nmflag, cflag);
        BIO_free(b);
        return(ret);
        }
#endif

int X509_print(BIO *bp, X509 *x)
{
        return X509_print_ex(bp, x, XN_FLAG_COMPAT, X509_FLAG_COMPAT);
}

int X509_print_ex(BIO *bp, X509 *x, unsigned long nmflags, unsigned long cflag)
        {
        long l;
        int ret=0,i;
        char *m=NULL,mlch = ' ';
        int nmindent = 0;
        X509_CINF *ci;
        ASN1_INTEGER *bs;
        EVP_PKEY *pkey=NULL;
        const char *neg;

        if((nmflags & XN_FLAG_SEP_MASK) == XN_FLAG_SEP_MULTILINE) {
                        mlch = '\n';
                        nmindent = 12;
        }

        if(nmflags == X509_FLAG_COMPAT)
                nmindent = 16;

        ci=x->cert_info;
        if(!(cflag & X509_FLAG_NO_HEADER))
                {
                if (BIO_write(bp,"Certificate:\n",13) <= 0) goto err;
```

```
128                 if (BIO_write(bp,"    Data:\n",10) <= 0) goto err;
129                 }
130         if(!(cflag & X509_FLAG_NO_VERSION))
131                 {
132                 l=X509_get_version(x);
133                 if (BIO_printf(bp,"%8sVersion: %lu (0x%lx)\n","",l+1,l) <= 0) go
134                 }
135         if(!(cflag & X509_FLAG_NO_SERIAL))
136                 {
138                 if (BIO_write(bp,"        Serial Number:",22) <= 0) goto err;
140                 bs=X509_get_serialNumber(x);
141                 if (bs->length <= (int)sizeof(long))
142                         {
143                         l=ASN1_INTEGER_get(bs);
144                         if (bs->type == V_ASN1_NEG_INTEGER)
145                                 {
146                                 l= -l;
147                                 neg="-";
148                                 }
149                         else
150                                 neg="";
151                         if (BIO_printf(bp," %s%lu (%s0x%lx)\n",neg,l,neg,l) <= 0
152                                 goto err;
153                         }
154                 else
155                         {
156                         neg=(bs->type == V_ASN1_NEG_INTEGER)?" (Negative)":"";
157                         if (BIO_printf(bp,"\n%12s%s","",neg) <= 0) goto err;
159                         for (i=0; i<bs->length; i++)
160                                 {
161                                 if (BIO_printf(bp,"%02x%c",bs->data[i],
162                                         ((i+1 == bs->length)?'\n':':')) <= 0)
163                                         goto err;
164                                 }
165                         }
167                 }
169         if(!(cflag & X509_FLAG_NO_SIGNAME))
170                 {
171                 if(X509_signature_print(bp, x->sig_alg, NULL) <= 0)
172                         goto err;
173 #if 0
174                 if (BIO_printf(bp,"%8sSignature Algorithm: ","") <= 0)
175                         goto err;
176                 if (i2a_ASN1_OBJECT(bp, ci->signature->algorithm) <= 0)
177                         goto err;
178                 if (BIO_puts(bp, "\n") <= 0)
179                         goto err;
180 #endif
181                 }
183         if(!(cflag & X509_FLAG_NO_ISSUER))
184                 {
185                 if (BIO_printf(bp,"        Issuer:%c",mlch) <= 0) goto err;
186                 if (X509_NAME_print_ex(bp,X509_get_issuer_name(x),nmindent, nmfl
187                 if (BIO_write(bp,"\n",1) <= 0) goto err;
188                 }
189         if(!(cflag & X509_FLAG_NO_VALIDITY))
190                 {
191                 if (BIO_write(bp,"        Validity\n",17) <= 0) goto err;
192                 if (BIO_write(bp,"            Not Before: ",24) <= 0) goto err;
193                 if (!ASN1_TIME_print(bp,X509_get_notBefore(x))) goto err;
```



```
194                 if (BIO_write(bp,"\n            Not After : ",25) <= 0) goto err
195                 if (!ASN1_TIME_print(bp,X509_get_notAfter(x))) goto err;
196                 if (BIO_write(bp,"\n",1) <= 0) goto err;
197                 }
198         if(!(cflag & X509_FLAG_NO_SUBJECT))
199                 {
200                 if (BIO_printf(bp,"        Subject:%c",mlch) <= 0) goto err;
201                 if (X509_NAME_print_ex(bp,X509_get_subject_name(x),nmindent, nmf
202                 if (BIO_write(bp,"\n",1) <= 0) goto err;
203                 }
204         if(!(cflag & X509_FLAG_NO_PUBKEY))
205                 {
206                 if (BIO_write(bp,"        Subject Public Key Info:\n",33) <= 0)
207                         goto err;
208                 if (BIO_printf(bp,"%12sPublic Key Algorithm: ","") <= 0)
209                         goto err;
210                 if (i2a_ASN1_OBJECT(bp, ci->key->algor->algorithm) <= 0)
211                         goto err;
212                 if (BIO_puts(bp, "\n") <= 0)
213                         goto err;
215                 pkey=X509_get_pubkey(x);
216                 if (pkey == NULL)
217                         {
218                         BIO_printf(bp,"%12sUnable to load Public Key\n","");
219                         ERR_print_errors(bp);
220                         }
221                 else
222                         {
223                         EVP_PKEY_print_public(bp, pkey, 16, NULL);
224                         EVP_PKEY_free(pkey);
225                         }
226                 }
228         if (!(cflag & X509_FLAG_NO_EXTENSIONS))
229                 X509V3_extensions_print(bp, "X509v3 extensions",
230                                         ci->extensions, cflag, 8);
232         if(!(cflag & X509_FLAG_NO_SIGDUMP))
233                 {
234                 if(X509_signature_print(bp, x->sig_alg, x->signature) <= 0) goto
235                 }
236         if(!(cflag & X509_FLAG_NO_AUX))
237                 {
238                 if (!X509_CERT_AUX_print(bp, x->aux, 0)) goto err;
239                 }
240         ret=1;
241 err:
242         if (m != NULL) OPENSSL_free(m);
243         return(ret);
244         }
246 int X509_ocspid_print (BIO *bp, X509 *x)
247         {
248         unsigned char *der=NULL ;
249         unsigned char *dertmp;
250         int derlen;
251         int i;
252         unsigned char SHA1md[SHA_DIGEST_LENGTH];
254         /* display the hash of the subject as it would appear
255            in OCSP requests */
256         if (BIO_printf(bp,"        Subject OCSP hash: ") <= 0)
257                 goto err;
258         derlen = i2d_X509_NAME(x->cert_info->subject, NULL);
259         if ((der = dertmp = (unsigned char *)OPENSSL_malloc (derlen)) == NULL)
```

```
260                 goto err;
261         i2d_X509_NAME(x->cert_info->subject, &dertmp);

263         if (!EVP_Digest(der, derlen, SHA1md, NULL, EVP_sha1(), NULL))
264                 goto err;
265         for (i=0; i < SHA_DIGEST_LENGTH; i++)
266                 {
267                 if (BIO_printf(bp,"%02X",SHA1md[i]) <= 0) goto err;
268                 }
269         OPENSSL_free (der);
270         der=NULL;

272         /* display the hash of the public key as it would appear
273            in OCSP requests */
274         if (BIO_printf(bp,"\n        Public key OCSP hash: ") <= 0)
275                 goto err;

277         if (!EVP_Digest(x->cert_info->key->public_key->data,
278                         x->cert_info->key->public_key->length,
279                         SHA1md, NULL, EVP_sha1(), NULL))
280                 goto err;
281         for (i=0; i < SHA_DIGEST_LENGTH; i++)
282                 {
283                 if (BIO_printf(bp,"%02X",SHA1md[i]) <= 0)
284                         goto err;
285                 }
286         BIO_printf(bp,"\n");

288         return (1);
289 err:
290         if (der != NULL) OPENSSL_free(der);
291         return(0);
292         }

294 int X509_signature_dump(BIO *bp, const ASN1_STRING *sig, int indent)
295 {
296         const unsigned char *s;
297         int i, n;

299         n=sig->length;
300         s=sig->data;
301         for (i=0; i<n; i++)
302                 {
303                 if ((i%18) == 0)
304                         {
305                         if (BIO_write(bp,"\n",1) <= 0) return 0;
306                         if (BIO_indent(bp, indent, indent) <= 0) return 0;
307                         }
308                         if (BIO_printf(bp,"%02x%s",s[i],
309                                 ((i+1) == n)?"":":") <= 0) return 0;
310                 }
311         if (BIO_write(bp,"\n",1) != 1) return 0;

313         return 1;
314 }

316 int X509_signature_print(BIO *bp, X509_ALGOR *sigalg, ASN1_STRING *sig)
317 {
318         int sig_nid;
319         if (BIO_puts(bp,"    Signature Algorithm: ") <= 0) return 0;
320         if (i2a_ASN1_OBJECT(bp, sigalg->algorithm) <= 0) return 0;

322         sig_nid = OBJ_obj2nid(sigalg->algorithm);
323         if (sig_nid != NID_undef)
324                 {
325                 int pkey_nid, dig_nid;
```



```
326                 const EVP_PKEY_ASN1_METHOD *ameth;
327                 if (OBJ_find_sigid_algs(sig_nid, &dig_nid, &pkey_nid))
328                         {
329                         ameth = EVP_PKEY_asn1_find(NULL, pkey_nid);
330                         if (ameth && ameth->sig_print)
331                                 return ameth->sig_print(bp, sigalg, sig, 9, 0);
332                         }
333                 }
334         if (sig)
335                 return X509_signature_dump(bp, sig, 9);
336         else if (BIO_puts(bp, "\n") <= 0)
337                 return 0;
338         return 1;
339 }

341 int ASN1_STRING_print(BIO *bp, const ASN1_STRING *v)
342         {
343         int i,n;
344         char buf[80];
345         const char *p;

347         if (v == NULL) return(0);
348         n=0;
349         p=(const char *)v->data;
350         for (i=0; i<v->length; i++)
351                 {
352                 if ((p[i] > '~') || ((p[i] < ' ') &&
353                         (p[i] != '\n') && (p[i] != '\r')))
354                         buf[n]='.';
355                 else
356                         buf[n]=p[i];
357                 n++;
358                 if (n >= 80)
359                         {
360                         if (BIO_write(bp,buf,n) <= 0)
361                                 return(0);
362                         n=0;
363                         }
364                 }
365         if (n > 0)
366                 if (BIO_write(bp,buf,n) <= 0)
367                         return(0);
368         return(1);
369         }

371 int ASN1_TIME_print(BIO *bp, const ASN1_TIME *tm)
372 {
373         if(tm->type == V_ASN1_UTCTIME) return ASN1_UTCTIME_print(bp, tm);
374         if(tm->type == V_ASN1_GENERALIZEDTIME)
375                                 return ASN1_GENERALIZEDTIME_print(bp, tm);
376         BIO_write(bp,"Bad time value",14);
377         return(0);
378 }

380 static const char *mon[12]=
381     {
382     "Jan","Feb","Mar","Apr","May","Jun",
383     "Jul","Aug","Sep","Oct","Nov","Dec"
384     };

386 int ASN1_GENERALIZEDTIME_print(BIO *bp, const ASN1_GENERALIZEDTIME *tm)
387         {
388         char *v;
389         int gmt=0;
390         int i;
391         int y=0,M=0,d=0,h=0,m=0,s=0;
```

```
392         char *f = NULL;
393         int f_len = 0;

395         i=tm->length;
396         v=(char *)tm->data;

398         if (i < 12) goto err;
399         if (v[i-1] == 'Z') gmt=1;
400         for (i=0; i<12; i++)
401                 if ((v[i] > '9') || (v[i] < '0')) goto err;
402         y= (v[0]-'0')*1000+(v[1]-'0')*100 + (v[2]-'0')*10+(v[3]-'0');
403         M= (v[4]-'0')*10+(v[5]-'0');
404         if ((M > 12) || (M < 1)) goto err;
405         d= (v[6]-'0')*10+(v[7]-'0');
406         h= (v[8]-'0')*10+(v[9]-'0');
407         m=  (v[10]-'0')*10+(v[11]-'0');
408         if (tm->length >= 14 &&
409             (v[12] >= '0') && (v[12] <= '9') &&
410             (v[13] >= '0') && (v[13] <= '9'))
411                 {
412                 s=  (v[12]-'0')*10+(v[13]-'0');
413                 /* Check for fractions of seconds. */
414                 if (tm->length >= 15 && v[14] == '.')
415                         {
416                         int l = tm->length;
417                         f = &v[14];     /* The decimal point. */
418                         f_len = 1;
419                         while (14 + f_len < l && f[f_len] >= '0' && f[f_len] <=
420                                 ++f_len;
421                         }
422                 }

424         if (BIO_printf(bp,"%s %2d %02d:%02d:%02d%.*s %d%s",
425                 mon[M-1],d,h,m,s,f_len,f,y,(gmt)?" GMT":"") <= 0)
426                 return(0);
427         else
428                 return(1);
429 err:
430         BIO_write(bp,"Bad time value",14);
431         return(0);
432         }

434 int ASN1_UTCTIME_print(BIO *bp, const ASN1_UTCTIME *tm)
435         {
436         const char *v;
437         int gmt=0;
438         int i;
439         int y=0,M=0,d=0,h=0,m=0,s=0;

441         i=tm->length;
442         v=(const char *)tm->data;

444         if (i < 10) goto err;
445         if (v[i-1] == 'Z') gmt=1;
446         for (i=0; i<10; i++)
447                 if ((v[i] > '9') || (v[i] < '0')) goto err;
448         y= (v[0]-'0')*10+(v[1]-'0');
449         if (y < 50) y+=100;
450         M= (v[2]-'0')*10+(v[3]-'0');
451         if ((M > 12) || (M < 1)) goto err;
452         d= (v[4]-'0')*10+(v[5]-'0');
453         h= (v[6]-'0')*10+(v[7]-'0');
454         m=  (v[8]-'0')*10+(v[9]-'0');
455         if (tm->length >=12 &&
456             (v[10] >= '0') && (v[10] <= '9') &&
457             (v[11] >= '0') && (v[11] <= '9'))
```



```
458                 s=  (v[10]-'0')*10+(v[11]-'0');

460         if (BIO_printf(bp,"%s %2d %02d:%02d:%02d %d%s",
461                 mon[M-1],d,h,m,s,y+1900,(gmt)?" GMT":"") <= 0)
462                 return(0);
463         else
464                 return(1);
465 err:
466         BIO_write(bp,"Bad time value",14);
467         return(0);
468         }

470 int X509_NAME_print(BIO *bp, X509_NAME *name, int obase)
471         {
472         char *s,*c,*b;
473         int ret=0,l,i;

475         l=80-2-obase;

477         b=X509_NAME_oneline(name,NULL,0);
478         if (!b)
479                 return 0;
480         if (!*b)
481                 {
482                 OPENSSL_free(b);
483                 return 1;
484                 }
485         s=b+1; /* skip the first slash */

487         c=s;
488         for (;;)
489                 {
490 #ifndef CHARSET_EBCDIC
491                 if (    ((*s == '/') &&
492                         ((s[1] >= 'A') && (s[1] <= 'Z') && (
493                                 (s[2] == '=') ||
494                                 ((s[2] >= 'A') && (s[2] <= 'Z') &&
495                                 (s[3] == '='))
496                          ))) ||
497                         (*s == '\0'))
498 #else
499                 if (    ((*s == '/') &&
500                         (isupper(s[1]) && (
501                                 (s[2] == '=') ||
502                                 (isupper(s[2]) &&
503                                 (s[3] == '='))
504                          ))) ||
505                         (*s == '\0'))
506 #endif
507                         {
508                         i=s-c;
509                         if (BIO_write(bp,c,i) != i) goto err;
510                         c=s+1;  /* skip following slash */
511                         if (*s != '\0')
512                                 {
513                                 if (BIO_write(bp,", ",2) != 2) goto err;
514                                 }
515                         l--;
516                         }
517                 if (*s == '\0') break;
518                 s++;
519                 l--;
520                 }

522         ret=1;
523         if (0)
```

```
524                 {
525 err:
526                 X509err(X509_F_X509_NAME_PRINT,ERR_R_BUF_LIB);
527                 }
528         OPENSSL_free(b);
529         return(ret);
530         }
531 #endif /* ! codereview */
```

```
**********************************************************
    4166 Wed Aug 13 19:52:05 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/t_x509a.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* t_x509a.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/asn1.h>
#include <openssl/x509.h>

/* X509_CERT_AUX and string set routines
 */

int X509_CERT_AUX_print(BIO *out, X509_CERT_AUX *aux, int indent)
{
        char oidstr[80], first;
        int i;
        if(!aux) return 1;
        if(aux->trust) {
                first = 1;
                BIO_printf(out, "%*sTrusted Uses:\n%*s",
                                                indent, "", indent + 2, "");
                for(i = 0; i < sk_ASN1_OBJECT_num(aux->trust); i++) {
                        if(!first) BIO_puts(out, ", ");
                        else first = 0;
                        OBJ_obj2txt(oidstr, sizeof oidstr,
                                sk_ASN1_OBJECT_value(aux->trust, i), 0);
                        BIO_puts(out, oidstr);
                }
                BIO_puts(out, "\n");
        } else BIO_printf(out, "%*sNo Trusted Uses.\n", indent, "");
        if(aux->reject) {
                first = 1;
                BIO_printf(out, "%*sRejected Uses:\n%*s",
                                                indent, "", indent + 2, "");
                for(i = 0; i < sk_ASN1_OBJECT_num(aux->reject); i++) {
                        if(!first) BIO_puts(out, ", ");
                        else first = 0;
                        OBJ_obj2txt(oidstr, sizeof oidstr,
                                sk_ASN1_OBJECT_value(aux->reject, i), 0);
                        BIO_puts(out, oidstr);
                }
                BIO_puts(out, "\n");
        } else BIO_printf(out, "%*sNo Rejected Uses.\n", indent, "");
        if(aux->alias) BIO_printf(out, "%*sAlias: %s\n", indent, "",
                                                aux->alias->data);
        if(aux->keyid) {
                BIO_printf(out, "%*sKey Id: ", indent, "");
                for(i = 0; i < aux->keyid->length; i++)
                        BIO_printf(out, "%s%02X",
                                i ? ":" : "",
                                aux->keyid->data[i]);
                BIO_write(out,"\n",1);
        }
        return 1;
}
#endif /* ! codereview */
```

```
**********************************************************
   32072 Wed Aug 13 19:52:05 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/tasn_dec.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* tasn_dec.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stddef.h>
#include <string.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/objects.h>
#include <openssl/buffer.h>
#include <openssl/err.h>

static int asn1_check_eoc(const unsigned char **in, long len);
static int asn1_find_end(const unsigned char **in, long len, char inf);

static int asn1_collect(BUF_MEM *buf, const unsigned char **in, long len,
                        char inf, int tag, int aclass, int depth);

static int collect_data(BUF_MEM *buf, const unsigned char **p, long plen);

static int asn1_check_tlen(long *olen, int *otag, unsigned char *oclass,
                                char *inf, char *cst,
                                const unsigned char **in, long len,
                                int exptag, int expclass, char opt,
                                ASN1_TLC *ctx);

static int asn1_template_ex_d2i(ASN1_VALUE **pval,
                                const unsigned char **in, long len,
                                const ASN1_TEMPLATE *tt, char opt,
                                ASN1_TLC *ctx);
static int asn1_template_noexp_d2i(ASN1_VALUE **val,
                                const unsigned char **in, long len,
                                const ASN1_TEMPLATE *tt, char opt,
                                ASN1_TLC *ctx);
static int asn1_d2i_ex_primitive(ASN1_VALUE **pval,
                                const unsigned char **in, long len,
                                const ASN1_ITEM *it,
                                int tag, int aclass, char opt, ASN1_TLC *ctx);

/* Table to convert tags to bit values, used for MSTRING type */
static const unsigned long tag2bit[32] = {
0,      0,      0,      B_ASN1_BIT_STRING,      /* tags  0 -  3 */
B_ASN1_OCTET_STRING,    0,      0,              B_ASN1_UNKNOWN,/* tags  4- 7 */
B_ASN1_UNKNOWN, B_ASN1_UNKNOWN, B_ASN1_UNKNOWN, B_ASN1_UNKNOWN,/* tags  8-11 */
B_ASN1_UTF8STRING,B_ASN1_UNKNOWN,B_ASN1_UNKNOWN,B_ASN1_UNKNOWN,/* tags 12-15 */
B_ASN1_SEQUENCE,0,B_ASN1_NUMERICSTRING,B_ASN1_PRINTABLESTRING, /* tags 16-19 */
B_ASN1_T61STRING,B_ASN1_VIDEOTEXSTRING,B_ASN1_IA5STRING,       /* tags 20-22 */
B_ASN1_UTCTIME, B_ASN1_GENERALIZEDTIME,                        /* tags 23-24 */
B_ASN1_GRAPHICSTRING,B_ASN1_ISO64STRING,B_ASN1_GENERALSTRING,  /* tags 25-27 */
B_ASN1_UNIVERSALSTRING,B_ASN1_UNKNOWN,B_ASN1_BMPSTRING,B_ASN1_UNKNOWN, /* tags 2
        };

unsigned long ASN1_tag2bit(int tag)
        {
        if ((tag < 0) || (tag > 30)) return 0;
        return tag2bit[tag];
        }

/* Macro to initialize and invalidate the cache */

#define asn1_tlc_clear(c)       if (c) (c)->valid = 0
/* Version to avoid compiler warning about 'c' always non-NULL */
#define asn1_tlc_clear_nc(c)    (c)->valid = 0

/* Decode an ASN1 item, this currently behaves just
 * like a standard 'd2i' function. 'in' points to
 * a buffer to read the data from, in future we will
 * have more advanced versions that can input data
 * a piece at a time and this will simply be a special
 * case.
 */
```

```
ASN1_VALUE *ASN1_item_d2i(ASN1_VALUE **pval,
                const unsigned char **in, long len, const ASN1_ITEM *it)
        {
        ASN1_TLC c;
        ASN1_VALUE *ptmpval = NULL;
        if (!pval)
                pval = &ptmpval;
        asn1_tlc_clear_nc(&c);
        if (ASN1_item_ex_d2i(pval, in, len, it, -1, 0, 0, &c) > 0)
                return *pval;
        return NULL;
        }

int ASN1_template_d2i(ASN1_VALUE **pval,
                const unsigned char **in, long len, const ASN1_TEMPLATE *tt)
        {
        ASN1_TLC c;
        asn1_tlc_clear_nc(&c);
        return asn1_template_ex_d2i(pval, in, len, tt, 0, &c);
        }


/* Decode an item, taking care of IMPLICIT tagging, if any.
 * If 'opt' set and tag mismatch return -1 to handle OPTIONAL
 */

int ASN1_item_ex_d2i(ASN1_VALUE **pval, const unsigned char **in, long len,
                        const ASN1_ITEM *it,
                        int tag, int aclass, char opt, ASN1_TLC *ctx)
        {
        const ASN1_TEMPLATE *tt, *errtt = NULL;
        const ASN1_COMPAT_FUNCS *cf;
        const ASN1_EXTERN_FUNCS *ef;
        const ASN1_AUX *aux = it->funcs;
        ASN1_aux_cb *asn1_cb;
        const unsigned char *p = NULL, *q;
        unsigned char *wp=NULL; /* BIG FAT WARNING!  BREAKS CONST WHERE USED */
        unsigned char imphack = 0, oclass;
        char seq_eoc, seq_nolen, cst, isopt;
        long tmplen;
        int i;
        int otag;
        int ret = 0;
        ASN1_VALUE **pchptr, *ptmpval;
        if (!pval)
                return 0;
        if (aux && aux->asn1_cb)
                asn1_cb = aux->asn1_cb;
        else asn1_cb = 0;

        switch(it->itype)
                {
                case ASN1_ITYPE_PRIMITIVE:
                if (it->templates)
                        {
                        /* tagging or OPTIONAL is currently illegal on an item
                         * template because the flags can't get passed down.
                         * In practice this isn't a problem: we include the
                         * relevant flags from the item template in the
                         * template itself.
                         */
                        if ((tag != -1) || opt)
                                {
                                ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
                                ASN1_R_ILLEGAL_OPTIONS_ON_ITEM_TEMPLATE);
                                goto err;
```



```
                                }
                        return asn1_template_ex_d2i(pval, in, len,
                                        it->templates, opt, ctx);
                }
                return asn1_d2i_ex_primitive(pval, in, len, it,
                                                tag, aclass, opt, ctx);
                break;

                case ASN1_ITYPE_MSTRING:
                p = *in;
                /* Just read in tag and class */
                ret = asn1_check_tlen(NULL, &otag, &oclass, NULL, NULL,
                                                &p, len, -1, 0, 1, ctx);
                if (!ret)
                        {
                        ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
                                        ERR_R_NESTED_ASN1_ERROR);
                        goto err;
                        }

                /* Must be UNIVERSAL class */
                if (oclass != V_ASN1_UNIVERSAL)
                        {
                        /* If OPTIONAL, assume this is OK */
                        if (opt) return -1;
                        ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
                                        ASN1_R_MSTRING_NOT_UNIVERSAL);
                        goto err;
                        }
                /* Check tag matches bit map */
                if (!(ASN1_tag2bit(otag) & it->utype))
                        {
                        /* If OPTIONAL, assume this is OK */
                        if (opt)
                                return -1;
                        ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
                                        ASN1_R_MSTRING_WRONG_TAG);
                        goto err;
                        }
                return asn1_d2i_ex_primitive(pval, in, len,
                                                it, otag, 0, 0, ctx);

                case ASN1_ITYPE_EXTERN:
                /* Use new style d2i */
                ef = it->funcs;
                return ef->asn1_ex_d2i(pval, in, len,
                                                it, tag, aclass, opt, ctx);

                case ASN1_ITYPE_COMPAT:
                /* we must resort to old style evil hackery */
                cf = it->funcs;

                /* If OPTIONAL see if it is there */
                if (opt)
                        {
                        int exptag;
                        p = *in;
                        if (tag == -1)
                                exptag = it->utype;
                        else exptag = tag;
                        /* Don't care about anything other than presence
                         * of expected tag */

                        ret = asn1_check_tlen(NULL, NULL, NULL, NULL, NULL,
                                        &p, len, exptag, aclass, 1, ctx);
                        if (!ret)
```

```
260                             {
261                             ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
262                                     ERR_R_NESTED_ASN1_ERROR);
263                             goto err;
264                             }
265                     if (ret == -1)
266                             return -1;
267                     }

269                 /* This is the old style evil hack IMPLICIT handling:
270                  * since the underlying code is expecting a tag and
271                  * class other than the one present we change the
272                  * buffer temporarily then change it back afterwards.
273                  * This doesn't and never did work for tags > 30.
274                  *
275                  * Yes this is *horrible* but it is only needed for
276                  * old style d2i which will hopefully not be around
277                  * for much longer.
278                  * FIXME: should copy the buffer then modify it so
279                  * the input buffer can be const: we should *always*
280                  * copy because the old style d2i might modify the
281                  * buffer.
282                  */

284                 if (tag != -1)
285                         {
286                         wp = *(unsigned char **)in;
287                         imphack = *wp;
288                         if (p == NULL)
289                                 {
290                                 ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
291                                         ERR_R_NESTED_ASN1_ERROR);
292                                 goto err;
293                                 }
294                         *wp = (unsigned char)((*p & V_ASN1_CONSTRUCTED)
295                                                         | it->utype);
296                         }

298                 ptmpval = cf->asn1_d2i(pval, in, len);

300                 if (tag != -1)
301                         *wp = imphack;

303                 if (ptmpval)
304                         return 1;

306                 ASN1err(ASN1_F_ASN1_ITEM_EX_D2I, ERR_R_NESTED_ASN1_ERROR);
307                 goto err;


310                 case ASN1_ITYPE_CHOICE:
311                 if (asn1_cb && !asn1_cb(ASN1_OP_D2I_PRE, pval, it, NULL))
312                         goto auxerr;

314                 /* Allocate structure */
315                 if (!*pval && !ASN1_item_ex_new(pval, it))
316                         {
317                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
318                                         ERR_R_NESTED_ASN1_ERROR);
319                         goto err;
320                         }
321                 /* CHOICE type, try each possibility in turn */
322                 p = *in;
323                 for (i = 0, tt=it->templates; i < it->tcount; i++, tt++)
324                         {
325                         pchptr = asn1_get_field_ptr(pval, tt);
```



```
326                         /* We mark field as OPTIONAL so its absence
327                          * can be recognised.
328                          */
329                         ret = asn1_template_ex_d2i(pchptr, &p, len, tt, 1, ctx);
330                         /* If field not present, try the next one */
331                         if (ret == -1)
332                                 continue;
333                         /* If positive return, read OK, break loop */
334                         if (ret > 0)
335                                 break;
336                         /* Otherwise must be an ASN1 parsing error */
337                         errtt = tt;
338                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
339                                         ERR_R_NESTED_ASN1_ERROR);
340                         goto err;
341                         }

343                 /* Did we fall off the end without reading anything? */
344                 if (i == it->tcount)
345                         {
346                         /* If OPTIONAL, this is OK */
347                         if (opt)
348                                 {
349                                 /* Free and zero it */
350                                 ASN1_item_ex_free(pval, it);
351                                 return -1;
352                                 }
353                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
354                                         ASN1_R_NO_MATCHING_CHOICE_TYPE);
355                         goto err;
356                         }

358                 asn1_set_choice_selector(pval, i, it);
359                 *in = p;
360                 if (asn1_cb && !asn1_cb(ASN1_OP_D2I_POST, pval, it, NULL))
361                                 goto auxerr;
362                 return 1;

364                 case ASN1_ITYPE_NDEF_SEQUENCE:
365                 case ASN1_ITYPE_SEQUENCE:
366                 p = *in;
367                 tmplen = len;

369                 /* If no IMPLICIT tagging set to SEQUENCE, UNIVERSAL */
370                 if (tag == -1)
371                         {
372                         tag = V_ASN1_SEQUENCE;
373                         aclass = V_ASN1_UNIVERSAL;
374                         }
375                 /* Get SEQUENCE length and update len, p */
376                 ret = asn1_check_tlen(&len, NULL, NULL, &seq_eoc, &cst,
377                                         &p, len, tag, aclass, opt, ctx);
378                 if (!ret)
379                         {
380                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
381                                         ERR_R_NESTED_ASN1_ERROR);
382                         goto err;
383                         }
384                 else if (ret == -1)
385                         return -1;
386                 if (aux && (aux->flags & ASN1_AFLG_BROKEN))
387                         {
388                         len = tmplen - (p - *in);
389                         seq_nolen = 1;
390                         }
391                 /* If indefinite we don't do a length check */
```

```
392                 else seq_nolen = seq_eoc;
393                 if (!cst)
394                         {
395                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
396                                 ASN1_R_SEQUENCE_NOT_CONSTRUCTED);
397                         goto err;
398                         }
399
400                 if (!*pval && !ASN1_item_ex_new(pval, it))
401                         {
402                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
403                                 ERR_R_NESTED_ASN1_ERROR);
404                         goto err;
405                         }
406
407                 if (asn1_cb && !asn1_cb(ASN1_OP_D2I_PRE, pval, it, NULL))
408                                 goto auxerr;
409
410                 /* Get each field entry */
411                 for (i = 0, tt = it->templates; i < it->tcount; i++, tt++)
412                         {
413                         const ASN1_TEMPLATE *seqtt;
414                         ASN1_VALUE **pseqval;
415                         seqtt = asn1_do_adb(pval, tt, 1);
416                         if (!seqtt)
417                                 goto err;
418                         pseqval = asn1_get_field_ptr(pval, seqtt);
419                         /* Have we ran out of data? */
420                         if (!len)
421                                 break;
422                         q = p;
423                         if (asn1_check_eoc(&p, len))
424                                 {
425                                 if (!seq_eoc)
426                                         {
427                                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
428                                                 ASN1_R_UNEXPECTED_EOC);
429                                         goto err;
430                                         }
431                                 len -= p - q;
432                                 seq_eoc = 0;
433                                 q = p;
434                                 break;
435                                 }
436                         /* This determines the OPTIONAL flag value. The field
437                          * cannot be omitted if it is the last of a SEQUENCE
438                          * and there is still data to be read. This isn't
439                          * strictly necessary but it increases efficiency in
440                          * some cases.
441                          */
442                         if (i == (it->tcount - 1))
443                                 isopt = 0;
444                         else isopt = (char)(seqtt->flags & ASN1_TFLG_OPTIONAL);
445                         /* attempt to read in field, allowing each to be
446                          * OPTIONAL */
447
448                         ret = asn1_template_ex_d2i(pseqval, &p, len,
449                                                 seqtt, isopt, ctx);
450                         if (!ret)
451                                 {
452                                 errtt = seqtt;
453                                 goto err;
454                                 }
455                         else if (ret == -1)
456                                 {
457                                 /* OPTIONAL component absent.
```



```
458                                  * Free and zero the field.
459                                  */
460                                 ASN1_template_free(pseqval, seqtt);
461                                 continue;
462                                 }
463                         /* Update length */
464                         len -= p - q;
465                         }
466
467                 /* Check for EOC if expecting one */
468                 if (seq_eoc && !asn1_check_eoc(&p, len))
469                         {
470                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I, ASN1_R_MISSING_EOC);
471                         goto err;
472                         }
473                 /* Check all data read */
474                 if (!seq_nolen && len)
475                         {
476                         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
477                                         ASN1_R_SEQUENCE_LENGTH_MISMATCH);
478                         goto err;
479                         }
480
481                 /* If we get here we've got no more data in the SEQUENCE,
482                  * however we may not have read all fields so check all
483                  * remaining are OPTIONAL and clear any that are.
484                  */
485                 for (; i < it->tcount; tt++, i++)
486                         {
487                         const ASN1_TEMPLATE *seqtt;
488                         seqtt = asn1_do_adb(pval, tt, 1);
489                         if (!seqtt)
490                                 goto err;
491                         if (seqtt->flags & ASN1_TFLG_OPTIONAL)
492                                 {
493                                 ASN1_VALUE **pseqval;
494                                 pseqval = asn1_get_field_ptr(pval, seqtt);
495                                 ASN1_template_free(pseqval, seqtt);
496                                 }
497                         else
498                                 {
499                                 errtt = seqtt;
500                                 ASN1err(ASN1_F_ASN1_ITEM_EX_D2I,
501                                                         ASN1_R_FIELD_MISSING);
502                                 goto err;
503                                 }
504                         }
505                 /* Save encoding */
506                 if (!asn1_enc_save(pval, *in, p - *in, it))
507                         goto auxerr;
508                 *in = p;
509                 if (asn1_cb && !asn1_cb(ASN1_OP_D2I_POST, pval, it, NULL))
510                         goto auxerr;
511                 return 1;
512
513                 default:
514                 return 0;
515                 }
516         auxerr:
517         ASN1err(ASN1_F_ASN1_ITEM_EX_D2I, ASN1_R_AUX_ERROR);
518         err:
519         ASN1_item_ex_free(pval, it);
520         if (errtt)
521                 ERR_add_error_data(4, "Field=", errtt->field_name,
522                                         ", Type=", it->sname);
523         else
```

```
524                 ERR_add_error_data(2, "Type=", it->sname);
525         return 0;
526         }

528 /* Templates are handled with two separate functions.
529  * One handles any EXPLICIT tag and the other handles the rest.
530  */

532 static int asn1_template_ex_d2i(ASN1_VALUE **val,
533                                 const unsigned char **in, long inlen,
534                                 const ASN1_TEMPLATE *tt, char opt,
535                                                         ASN1_TLC *ctx)
536         {
537         int flags, aclass;
538         int ret;
539         long len;
540         const unsigned char *p, *q;
541         char exp_eoc;
542         if (!val)
543                 return 0;
544         flags = tt->flags;
545         aclass = flags & ASN1_TFLG_TAG_CLASS;

547         p = *in;

549         /* Check if EXPLICIT tag expected */
550         if (flags & ASN1_TFLG_EXPTAG)
551                 {
552                 char cst;
553                 /* Need to work out amount of data available to the inner
554                  * content and where it starts: so read in EXPLICIT header to
555                  * get the info.
556                  */
557                 ret = asn1_check_tlen(&len, NULL, NULL, &exp_eoc, &cst,
558                                       &p, inlen, tt->tag, aclass, opt, ctx);
559                 q = p;
560                 if (!ret)
561                         {
562                         ASN1err(ASN1_F_ASN1_TEMPLATE_EX_D2I,
563                                       ERR_R_NESTED_ASN1_ERROR);
564                         return 0;
565                         }
566                 else if (ret == -1)
567                         return -1;
568                 if (!cst)
569                         {
570                         ASN1err(ASN1_F_ASN1_TEMPLATE_EX_D2I,
571                                       ASN1_R_EXPLICIT_TAG_NOT_CONSTRUCTED);
572                         return 0;
573                         }
574                 /* We've found the field so it can't be OPTIONAL now */
575                 ret = asn1_template_noexp_d2i(val, &p, len, tt, 0, ctx);
576                 if (!ret)
577                         {
578                         ASN1err(ASN1_F_ASN1_TEMPLATE_EX_D2I,
579                                       ERR_R_NESTED_ASN1_ERROR);
580                         return 0;
581                         }
582                 /* We read the field in OK so update length */
583                 len -= p - q;
584                 if (exp_eoc)
585                         {
586                         /* If NDEF we must have an EOC here */
587                         if (!asn1_check_eoc(&p, len))
588                                 {
589                                 ASN1err(ASN1_F_ASN1_TEMPLATE_EX_D2I,
```



```
590                                                 ASN1_R_MISSING_EOC);
591                                 goto err;
592                                 }
593                         }
594                 else
595                         {
596                         /* Otherwise we must hit the EXPLICIT tag end or its
597                          * an error */
598                         if (len)
599                                 {
600                                 ASN1err(ASN1_F_ASN1_TEMPLATE_EX_D2I,
601                                         ASN1_R_EXPLICIT_LENGTH_MISMATCH);
602                                 goto err;
603                                 }
604                         }
605                 }
606                 else
607                         return asn1_template_noexp_d2i(val, in, inlen,
608                                                        tt, opt, ctx);

610         *in = p;
611         return 1;

613         err:
614         ASN1_template_free(val, tt);
615         return 0;
616         }

618 static int asn1_template_noexp_d2i(ASN1_VALUE **val,
619                                 const unsigned char **in, long len,
620                                 const ASN1_TEMPLATE *tt, char opt,
621                                 ASN1_TLC *ctx)
622         {
623         int flags, aclass;
624         int ret;
625         const unsigned char *p, *q;
626         if (!val)
627                 return 0;
628         flags = tt->flags;
629         aclass = flags & ASN1_TFLG_TAG_CLASS;

631         p = *in;
632         q = p;

634         if (flags & ASN1_TFLG_SK_MASK)
635                 {
636                 /* SET OF, SEQUENCE OF */
637                 int sktag, skaclass;
638                 char sk_eoc;
639                 /* First work out expected inner tag value */
640                 if (flags & ASN1_TFLG_IMPTAG)
641                         {
642                         sktag = tt->tag;
643                         skaclass = aclass;
644                         }
645                 else
646                         {
647                         skaclass = V_ASN1_UNIVERSAL;
648                         if (flags & ASN1_TFLG_SET_OF)
649                                 sktag = V_ASN1_SET;
650                         else
651                                 sktag = V_ASN1_SEQUENCE;
652                         }
653                 /* Get the tag */
654                 ret = asn1_check_tlen(&len, NULL, NULL, &sk_eoc, NULL,
655                                       &p, len, sktag, skaclass, opt, ctx);
```

```
656                 if (!ret)
657                         {
658                         ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I,
659                                         ERR_R_NESTED_ASN1_ERROR);
660                         return 0;
661                         }
662                 else if (ret == -1)
663                         return -1;
664                 if (!*val)
665                         *val = (ASN1_VALUE *)sk_new_null();
666                 else
667                         {
668                         /* We've got a valid STACK: free up any items present */
669                         STACK_OF(ASN1_VALUE) *sktmp
670                             = (STACK_OF(ASN1_VALUE) *)*val;
671                         ASN1_VALUE *vtmp;
672                         while(sk_ASN1_VALUE_num(sktmp) > 0)
673                                 {
674                                 vtmp = sk_ASN1_VALUE_pop(sktmp);
675                                 ASN1_item_ex_free(&vtmp,
676                                         ASN1_ITEM_ptr(tt->item));
677                                 }
678                         }

680                 if (!*val)
681                         {
682                         ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I,
683                                         ERR_R_MALLOC_FAILURE);
684                         goto err;
685                         }

687                 /* Read as many items as we can */
688                 while(len > 0)
689                         {
690                         ASN1_VALUE *skfield;
691                         q = p;
692                         /* See if EOC found */
693                         if (asn1_check_eoc(&p, len))
694                                 {
695                                 if (!sk_eoc)
696                                         {
697                                         ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I,
698                                                         ASN1_R_UNEXPECTED_EOC);
699                                         goto err;
700                                         }
701                                 len -= p - q;
702                                 sk_eoc = 0;
703                                 break;
704                                 }
705                         skfield = NULL;
706                         if (!ASN1_item_ex_d2i(&skfield, &p, len,
707                                         ASN1_ITEM_ptr(tt->item),
708                                         -1, 0, 0, ctx))
709                                 {
710                                 ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I,
711                                         ERR_R_NESTED_ASN1_ERROR);
712                                 goto err;
713                                 }
714                         len -= p - q;
715                         if (!sk_ASN1_VALUE_push((STACK_OF(ASN1_VALUE) *)*val,
716                                         skfield))
717                                 {
718                                 ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I,
719                                         ERR_R_MALLOC_FAILURE);
720                                 goto err;
721                                 }
```



```
722                         }
723                 if (sk_eoc)
724                         {
725                         ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I, ASN1_R_MISSING_E
726                         goto err;
727                         }
728                 }
729         else if (flags & ASN1_TFLG_IMPTAG)
730                 {
731                 /* IMPLICIT tagging */
732                 ret = ASN1_item_ex_d2i(val, &p, len,
733                         ASN1_ITEM_ptr(tt->item), tt->tag, aclass, opt, ctx);
734                 if (!ret)
735                         {
736                         ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I,
737                                                 ERR_R_NESTED_ASN1_ERROR);
738                         goto err;
739                         }
740                 else if (ret == -1)
741                         return -1;
742                 }
743         else
744                 {
745                 /* Nothing special */
746                 ret = ASN1_item_ex_d2i(val, &p, len, ASN1_ITEM_ptr(tt->item),
747                                                         -1, 0, opt, ctx);
748                 if (!ret)
749                         {
750                         ASN1err(ASN1_F_ASN1_TEMPLATE_NOEXP_D2I,
751                                                 ERR_R_NESTED_ASN1_ERROR);
752                         goto err;
753                         }
754                 else if (ret == -1)
755                         return -1;
756                 }

758         *in = p;
759         return 1;

761         err:
762         ASN1_template_free(val, tt);
763         return 0;
764         }

766 static int asn1_d2i_ex_primitive(ASN1_VALUE **pval,
767                                 const unsigned char **in, long inlen,
768                                 const ASN1_ITEM *it,
769                                 int tag, int aclass, char opt, ASN1_TLC *ctx)
770         {
771         int ret = 0, utype;
772         long plen;
773         char cst, inf, free_cont = 0;
774         const unsigned char *p;
775         BUF_MEM buf;
776         const unsigned char *cont = NULL;
777         long len;
778         if (!pval)
779                 {
780                 ASN1err(ASN1_F_ASN1_D2I_EX_PRIMITIVE, ASN1_R_ILLEGAL_NULL);
781                 return 0; /* Should never happen */
782                 }

784         if (it->itype == ASN1_ITYPE_MSTRING)
785                 {
786                 utype = tag;
787                 tag = -1;
```

```
788                 }
789         else
790                 utype = it->utype;

792         if (utype == V_ASN1_ANY)
793                 {
794                 /* If type is ANY need to figure out type from tag */
795                 unsigned char oclass;
796                 if (tag >= 0)
797                         {
798                         ASN1err(ASN1_F_ASN1_D2I_EX_PRIMITIVE,
799                                         ASN1_R_ILLEGAL_TAGGED_ANY);
800                         return 0;
801                         }
802                 if (opt)
803                         {
804                         ASN1err(ASN1_F_ASN1_D2I_EX_PRIMITIVE,
805                                         ASN1_R_ILLEGAL_OPTIONAL_ANY);
806                         return 0;
807                         }
808                 p = *in;
809                 ret = asn1_check_tlen(NULL, &utype, &oclass, NULL, NULL,
810                                         &p, inlen, -1, 0, 0, ctx);
811                 if (!ret)
812                         {
813                         ASN1err(ASN1_F_ASN1_D2I_EX_PRIMITIVE,
814                                         ERR_R_NESTED_ASN1_ERROR);
815                         return 0;
816                         }
817                 if (oclass != V_ASN1_UNIVERSAL)
818                         utype = V_ASN1_OTHER;
819                 }
820         if (tag == -1)
821                 {
822                 tag = utype;
823                 aclass = V_ASN1_UNIVERSAL;
824                 }
825         p = *in;
826         /* Check header */
827         ret = asn1_check_tlen(&plen, NULL, NULL, &inf, &cst,
828                                 &p, inlen, tag, aclass, opt, ctx);
829         if (!ret)
830                 {
831                 ASN1err(ASN1_F_ASN1_D2I_EX_PRIMITIVE, ERR_R_NESTED_ASN1_ERROR);
832                 return 0;
833                 }
834         else if (ret == -1)
835                 return -1;
836         ret = 0;
837         /* SEQUENCE, SET and "OTHER" are left in encoded form */
838         if ((utype == V_ASN1_SEQUENCE)
839                 || (utype == V_ASN1_SET) || (utype == V_ASN1_OTHER))
840                 {
841                 /* Clear context cache for type OTHER because the auto clear
842                  * when we have a exact match wont work
843                  */
844                 if (utype == V_ASN1_OTHER)
845                         {
846                         asn1_tlc_clear(ctx);
847                         }
848                 /* SEQUENCE and SET must be constructed */
849                 else if (!cst)
850                         {
851                         ASN1err(ASN1_F_ASN1_D2I_EX_PRIMITIVE,
852                                 ASN1_R_TYPE_NOT_CONSTRUCTED);
853                         return 0;
```



```
854                         }

856                 cont = *in;
857                 /* If indefinite length constructed find the real end */
858                 if (inf)
859                         {
860                         if (!asn1_find_end(&p, plen, inf))
861                                 goto err;
862                         len = p - cont;
863                         }
864                 else
865                         {
866                         len = p - cont + plen;
867                         p += plen;
868                         buf.data = NULL;
869                         }
870                 }
871         else if (cst)
872                 {
873                 buf.length = 0;
874                 buf.max = 0;
875                 buf.data = NULL;
876                 /* Should really check the internal tags are correct but
877                  * some things may get this wrong. The relevant specs
878                  * say that constructed string types should be OCTET STRINGs
879                  * internally irrespective of the type. So instead just check
880                  * for UNIVERSAL class and ignore the tag.
881                  */
882                 if (!asn1_collect(&buf, &p, plen, inf, -1, V_ASN1_UNIVERSAL, 0))
883                         {
884                         free_cont = 1;
885                         goto err;
886                         }
887                 len = buf.length;
888                 /* Append a final null to string */
889                 if (!BUF_MEM_grow_clean(&buf, len + 1))
890                         {
891                         ASN1err(ASN1_F_ASN1_D2I_EX_PRIMITIVE,
892                                                 ERR_R_MALLOC_FAILURE);
893                         return 0;
894                         }
895                 buf.data[len] = 0;
896                 cont = (const unsigned char *)buf.data;
897                 free_cont = 1;
898                 }
899         else
900                 {
901                 cont = p;
902                 len = plen;
903                 p += plen;
904                 }

906         /* We now have content length and type: translate into a structure */
907         if (!asn1_ex_c2i(pval, cont, len, utype, &free_cont, it))
908                 goto err;

910         *in = p;
911         ret = 1;
912         err:
913         if (free_cont && buf.data) OPENSSL_free(buf.data);
914         return ret;
915         }

917 /* Translate ASN1 content octets into a structure */

919 int asn1_ex_c2i(ASN1_VALUE **pval, const unsigned char *cont, int len,
```

```
920                         int utype, char *free_cont, const ASN1_ITEM *it)
921         {
922         ASN1_VALUE **opval = NULL;
923         ASN1_STRING *stmp;
924         ASN1_TYPE *typ = NULL;
925         int ret = 0;
926         const ASN1_PRIMITIVE_FUNCS *pf;
927         ASN1_INTEGER **tint;
928         pf = it->funcs;

930         if (pf && pf->prim_c2i)
931                 return pf->prim_c2i(pval, cont, len, utype, free_cont, it);
932         /* If ANY type clear type and set pointer to internal value */
933         if (it->utype == V_ASN1_ANY)
934                 {
935                 if (!*pval)
936                         {
937                         typ = ASN1_TYPE_new();
938                         if (typ == NULL)
939                                 goto err;
940                         *pval = (ASN1_VALUE *)typ;
941                         }
942                 else
943                         typ = (ASN1_TYPE *)*pval;

945                 if (utype != typ->type)
946                         ASN1_TYPE_set(typ, utype, NULL);
947                 opval = pval;
948                 pval = &typ->value.asn1_value;
949                 }
950         switch(utype)
951                 {
952                 case V_ASN1_OBJECT:
953                 if (!c2i_ASN1_OBJECT((ASN1_OBJECT **)pval, &cont, len))
954                         goto err;
955                 break;

957                 case V_ASN1_NULL:
958                 if (len)
959                         {
960                         ASN1err(ASN1_F_ASN1_EX_C2I,
961                                                 ASN1_R_NULL_IS_WRONG_LENGTH);
962                         goto err;
963                         }
964                 *pval = (ASN1_VALUE *)1;
965                 break;

967                 case V_ASN1_BOOLEAN:
968                 if (len != 1)
969                         {
970                         ASN1err(ASN1_F_ASN1_EX_C2I,
971                                                 ASN1_R_BOOLEAN_IS_WRONG_LENGTH);
972                         goto err;
973                         }
974                 else
975                         {
976                         ASN1_BOOLEAN *tbool;
977                         tbool = (ASN1_BOOLEAN *)pval;
978                         *tbool = *cont;
979                         }
980                 break;

982                 case V_ASN1_BIT_STRING:
983                 if (!c2i_ASN1_BIT_STRING((ASN1_BIT_STRING **)pval, &cont, len))
984                         goto err;
985                 break;
```



```
987                 case V_ASN1_INTEGER:
988                 case V_ASN1_NEG_INTEGER:
989                 case V_ASN1_ENUMERATED:
990                 case V_ASN1_NEG_ENUMERATED:
991                 tint = (ASN1_INTEGER **)pval;
992                 if (!c2i_ASN1_INTEGER(tint, &cont, len))
993                         goto err;
994                 /* Fixup type to match the expected form */
995                 (*tint)->type = utype | ((*tint)->type & V_ASN1_NEG);
996                 break;

998                 case V_ASN1_OCTET_STRING:
999                 case V_ASN1_NUMERICSTRING:
1000                case V_ASN1_PRINTABLESTRING:
1001                case V_ASN1_T61STRING:
1002                case V_ASN1_VIDEOTEXSTRING:
1003                case V_ASN1_IA5STRING:
1004                case V_ASN1_UTCTIME:
1005                case V_ASN1_GENERALIZEDTIME:
1006                case V_ASN1_GRAPHICSTRING:
1007                case V_ASN1_VISIBLESTRING:
1008                case V_ASN1_GENERALSTRING:
1009                case V_ASN1_UNIVERSALSTRING:
1010                case V_ASN1_BMPSTRING:
1011                case V_ASN1_UTF8STRING:
1012                case V_ASN1_OTHER:
1013                case V_ASN1_SET:
1014                case V_ASN1_SEQUENCE:
1015                default:
1016                if (utype == V_ASN1_BMPSTRING && (len & 1))
1017                        {
1018                        ASN1err(ASN1_F_ASN1_EX_C2I,
1019                                        ASN1_R_BMPSTRING_IS_WRONG_LENGTH);
1020                        goto err;
1021                        }
1022                if (utype == V_ASN1_UNIVERSALSTRING && (len & 3))
1023                        {
1024                        ASN1err(ASN1_F_ASN1_EX_C2I,
1025                                        ASN1_R_UNIVERSALSTRING_IS_WRONG_LENGTH);
1026                        goto err;
1027                        }
1028                /* All based on ASN1_STRING and handled the same */
1029                if (!*pval)
1030                        {
1031                        stmp = ASN1_STRING_type_new(utype);
1032                        if (!stmp)
1033                                {
1034                                ASN1err(ASN1_F_ASN1_EX_C2I,
1035                                                        ERR_R_MALLOC_FAILURE);
1036                                goto err;
1037                                }
1038                        *pval = (ASN1_VALUE *)stmp;
1039                        }
1040                else
1041                        {
1042                        stmp = (ASN1_STRING *)*pval;
1043                        stmp->type = utype;
1044                        }
1045                /* If we've already allocated a buffer use it */
1046                if (*free_cont)
1047                        {
1048                        if (stmp->data)
1049                                OPENSSL_free(stmp->data);
1050                        stmp->data = (unsigned char *)cont; /* UGLY CAST! RL */
1051                        stmp->length = len;
```

```
1052                        *free_cont = 0;
1053                        }
1054                else
1055                        {
1056                        if (!ASN1_STRING_set(stmp, cont, len))
1057                                {
1058                                ASN1err(ASN1_F_ASN1_EX_C2I,
1059                                                        ERR_R_MALLOC_FAILURE);
1060                                ASN1_STRING_free(stmp);
1061                                *pval = NULL;
1062                                goto err;
1063                                }
1064                        }
1065                break;
1066                }
1067        /* If ASN1_ANY and NULL type fix up value */
1068        if (typ && (utype == V_ASN1_NULL))
1069                 typ->value.ptr = NULL;

1071        ret = 1;
1072        err:
1073        if (!ret)
1074                {
1075                ASN1_TYPE_free(typ);
1076                if (opval)
1077                        *opval = NULL;
1078                }
1079        return ret;
1080        }


1083 /* This function finds the end of an ASN1 structure when passed its maximum
1084  * length, whether it is indefinite length and a pointer to the content.
1085  * This is more efficient than calling asn1_collect because it does not
1086  * recurse on each indefinite length header.
1087  */

1089 static int asn1_find_end(const unsigned char **in, long len, char inf)
1090        {
1091        int expected_eoc;
1092        long plen;
1093        const unsigned char *p = *in, *q;
1094        /* If not indefinite length constructed just add length */
1095        if (inf == 0)
1096                {
1097                *in += len;
1098                return 1;
1099                }
1100        expected_eoc = 1;
1101        /* Indefinite length constructed form. Find the end when enough EOCs
1102         * are found. If more indefinite length constructed headers
1103         * are encountered increment the expected eoc count otherwise just
1104         * skip to the end of the data.
1105         */
1106        while (len > 0)
1107                {
1108                if(asn1_check_eoc(&p, len))
1109                        {
1110                        expected_eoc--;
1111                        if (expected_eoc == 0)
1112                                break;
1113                        len -= 2;
1114                        continue;
1115                        }
1116                q = p;
1117                /* Just read in a header: only care about the length */
```



```
1118                if(!asn1_check_tlen(&plen, NULL, NULL, &inf, NULL, &p, len,
1119                                -1, 0, 0, NULL))
1120                        {
1121                ASN1err(ASN1_F_ASN1_FIND_END, ERR_R_NESTED_ASN1_ERROR);
1122                        return 0;
1123                        }
1124                if (inf)
1125                        expected_eoc++;
1126                else
1127                        p += plen;
1128                len -= p - q;
1129                }
1130        if (expected_eoc)
1131                {
1132                ASN1err(ASN1_F_ASN1_FIND_END, ASN1_R_MISSING_EOC);
1133                return 0;
1134                }
1135        *in = p;
1136        return 1;
1137        }
1138 /* This function collects the asn1 data from a constructred string
1139  * type into a buffer. The values of 'in' and 'len' should refer
1140  * to the contents of the constructed type and 'inf' should be set
1141  * if it is indefinite length.
1142  */

1144 #ifndef ASN1_MAX_STRING_NEST
1145 /* This determines how many levels of recursion are permitted in ASN1
1146  * string types. If it is not limited stack overflows can occur. If set
1147  * to zero no recursion is allowed at all. Although zero should be adequate
1148  * examples exist that require a value of 1. So 5 should be more than enough.
1149  */
1150 #define ASN1_MAX_STRING_NEST 5
1151 #endif


1154 static int asn1_collect(BUF_MEM *buf, const unsigned char **in, long len,
1155                        char inf, int tag, int aclass, int depth)
1156        {
1157        const unsigned char *p, *q;
1158        long plen;
1159        char cst, ininf;
1160        p = *in;
1161        inf &= 1;
1162        /* If no buffer and not indefinite length constructed just pass over
1163         * the encoded data */
1164        if (!buf && !inf)
1165                {
1166                *in += len;
1167                return 1;
1168                }
1169        while(len > 0)
1170                {
1171                q = p;
1172                /* Check for EOC */
1173                if (asn1_check_eoc(&p, len))
1174                        {
1175                        /* EOC is illegal outside indefinite length
1176                         * constructed form */
1177                        if (!inf)
1178                                {
1179                                ASN1err(ASN1_F_ASN1_COLLECT,
1180                                        ASN1_R_UNEXPECTED_EOC);
1181                                return 0;
1182                                }
1183                        inf = 0;
```

```
                        break;
                        }

                if (!asn1_check_tlen(&plen, NULL, NULL, &ininf, &cst, &p,
                                     len, tag, aclass, 0, NULL))
                        {
                        ASN1err(ASN1_F_ASN1_COLLECT, ERR_R_NESTED_ASN1_ERROR);
                        return 0;
                        }

                /* If indefinite length constructed update max length */
                if (cst)
                        {
                        if (depth >= ASN1_MAX_STRING_NEST)
                                {
                                ASN1err(ASN1_F_ASN1_COLLECT,
                                        ASN1_R_NESTED_ASN1_STRING);
                                return 0;
                                }
                        if (!asn1_collect(buf, &p, plen, ininf, tag, aclass,
                                          depth + 1))
                                return 0;
                        }
                else if (plen && !collect_data(buf, &p, plen))
                        return 0;
                len -= p - q;
                }
        if (inf)
                {
                ASN1err(ASN1_F_ASN1_COLLECT, ASN1_R_MISSING_EOC);
                return 0;
                }
        *in = p;
        return 1;
        }

static int collect_data(BUF_MEM *buf, const unsigned char **p, long plen)
        {
        int len;
        if (buf)
                {
                len = buf->length;
                if (!BUF_MEM_grow_clean(buf, len + plen))
                        {
                        ASN1err(ASN1_F_COLLECT_DATA, ERR_R_MALLOC_FAILURE);
                        return 0;
                        }
                memcpy(buf->data + len, *p, plen);
                }
        *p += plen;
        return 1;
        }

/* Check for ASN1 EOC and swallow it if found */

static int asn1_check_eoc(const unsigned char **in, long len)
        {
        const unsigned char *p;
        if (len < 2) return 0;
        p = *in;
        if (!p[0] && !p[1])
                {
                *in += 2;
                return 1;
                }
        return 0;
```



```
        }

/* Check an ASN1 tag and length: a bit like ASN1_get_object
 * but it sets the length for indefinite length constructed
 * form, we don't know the exact length but we can set an
 * upper bound to the amount of data available minus the
 * header length just read.
 */

static int asn1_check_tlen(long *olen, int *otag, unsigned char *oclass,
                                char *inf, char *cst,
                                const unsigned char **in, long len,
                                int exptag, int expclass, char opt,
                                ASN1_TLC *ctx)
        {
        int i;
        int ptag, pclass;
        long plen;
        const unsigned char *p, *q;
        p = *in;
        q = p;

        if (ctx && ctx->valid)
                {
                i = ctx->ret;
                plen = ctx->plen;
                pclass = ctx->pclass;
                ptag = ctx->ptag;
                p += ctx->hdrlen;
                }
        else
                {
                i = ASN1_get_object(&p, &plen, &ptag, &pclass, len);
                if (ctx)
                        {
                        ctx->ret = i;
                        ctx->plen = plen;
                        ctx->pclass = pclass;
                        ctx->ptag = ptag;
                        ctx->hdrlen = p - q;
                        ctx->valid = 1;
                        /* If definite length, and no error, length +
                         * header can't exceed total amount of data available.
                         */
                        if (!(i & 0x81) && ((plen + ctx->hdrlen) > len))
                                {
                                ASN1err(ASN1_F_ASN1_CHECK_TLEN,
                                                ASN1_R_TOO_LONG);
                                asn1_tlc_clear(ctx);
                                return 0;
                                }
                        }
                }

        if (i & 0x80)
                {
                ASN1err(ASN1_F_ASN1_CHECK_TLEN, ASN1_R_BAD_OBJECT_HEADER);
                asn1_tlc_clear(ctx);
                return 0;
                }
        if (exptag >= 0)
                {
                if ((exptag != ptag) || (expclass != pclass))
                        {
                        /* If type is OPTIONAL, not an error:
                         * indicate missing type.
```

```
1316                         */
1317                        if (opt) return -1;
1318                        asn1_tlc_clear(ctx);
1319                        ASN1err(ASN1_F_ASN1_CHECK_TLEN, ASN1_R_WRONG_TAG);
1320                        return 0;
1321                        }
1322                /* We have a tag and class match:
1323                 * assume we are going to do something with it */
1324                asn1_tlc_clear(ctx);
1325                }

1327        if (i & 1)
1328                plen = len - (p - q);

1330        if (inf)
1331                *inf = i & 1;

1333        if (cst)
1334                *cst = i & V_ASN1_CONSTRUCTED;

1336        if (olen)
1337                *olen = plen;

1339        if (oclass)
1340                *oclass = pclass;

1342        if (otag)
1343                *otag = ptag;

1345        *in = p;
1346        return 1;
1347        }
1348 #endif /* ! codereview */
```

```
**********************************************************
   18140 Wed Aug 13 19:52:05 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/tasn_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* tasn_enc.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stddef.h>
#include <string.h>
```



```
#include "cryptlib.h"
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/objects.h>

static int asn1_i2d_ex_primitive(ASN1_VALUE **pval, unsigned char **out,
                                 const ASN1_ITEM *it,
                                 int tag, int aclass);
static int asn1_set_seq_out(STACK_OF(ASN1_VALUE) *sk, unsigned char **out,
                            int skcontlen, const ASN1_ITEM *item,
                            int do_sort, int iclass);
static int asn1_template_ex_i2d(ASN1_VALUE **pval, unsigned char **out,
                                const ASN1_TEMPLATE *tt,
                                int tag, int aclass);
static int asn1_item_flags_i2d(ASN1_VALUE *val, unsigned char **out,
                               const ASN1_ITEM *it, int flags);

/* Top level i2d equivalents: the 'ndef' variant instructs the encoder
 * to use indefinite length constructed encoding, where appropriate
 */

int ASN1_item_ndef_i2d(ASN1_VALUE *val, unsigned char **out,
                                        const ASN1_ITEM *it)
        {
        return asn1_item_flags_i2d(val, out, it, ASN1_TFLG_NDEF);
        }

int ASN1_item_i2d(ASN1_VALUE *val, unsigned char **out, const ASN1_ITEM *it)
        {
        return asn1_item_flags_i2d(val, out, it, 0);
        }

/* Encode an ASN1 item, this is use by the
 * standard 'i2d' function. 'out' points to
 * a buffer to output the data to.
 *
 * The new i2d has one additional feature. If the output
 * buffer is NULL (i.e. *out == NULL) then a buffer is
 * allocated and populated with the encoding.
 */

static int asn1_item_flags_i2d(ASN1_VALUE *val, unsigned char **out,
                               const ASN1_ITEM *it, int flags)
        {
        if (out && !*out)
                {
                unsigned char *p, *buf;
                int len;
                len = ASN1_item_ex_i2d(&val, NULL, it, -1, flags);
                if (len <= 0)
                        return len;
                buf = OPENSSL_malloc(len);
                if (!buf)
                        return -1;
                p = buf;
                ASN1_item_ex_i2d(&val, &p, it, -1, flags);
                *out = buf;
                return len;
                }

        return ASN1_item_ex_i2d(&val, out, it, -1, flags);
        }

/* Encode an item, taking care of IMPLICIT tagging (if any).
 * This function performs the normal item handling: it can be
 * used in external types.
```

```
 */

int ASN1_item_ex_i2d(ASN1_VALUE **pval, unsigned char **out,
                     const ASN1_ITEM *it, int tag, int aclass)
        {
        const ASN1_TEMPLATE *tt = NULL;
        unsigned char *p = NULL;
        int i, seqcontlen, seqlen, ndef = 1;
        const ASN1_COMPAT_FUNCS *cf;
        const ASN1_EXTERN_FUNCS *ef;
        const ASN1_AUX *aux = it->funcs;
        ASN1_aux_cb *asn1_cb = 0;

        if ((it->itype != ASN1_ITYPE_PRIMITIVE) && !*pval)
                return 0;

        if (aux && aux->asn1_cb)
                 asn1_cb = aux->asn1_cb;

        switch(it->itype)
                {

                case ASN1_ITYPE_PRIMITIVE:
                if (it->templates)
                        return asn1_template_ex_i2d(pval, out, it->templates,
                                                                tag, aclass);
                return asn1_i2d_ex_primitive(pval, out, it, tag, aclass);
                break;

                case ASN1_ITYPE_MSTRING:
                return asn1_i2d_ex_primitive(pval, out, it, -1, aclass);

                case ASN1_ITYPE_CHOICE:
                if (asn1_cb && !asn1_cb(ASN1_OP_I2D_PRE, pval, it, NULL))
                                return 0;
                i = asn1_get_choice_selector(pval, it);
                if ((i >= 0) && (i < it->tcount))
                        {
                        ASN1_VALUE **pchval;
                        const ASN1_TEMPLATE *chtt;
                        chtt = it->templates + i;
                        pchval = asn1_get_field_ptr(pval, chtt);
                        return asn1_template_ex_i2d(pchval, out, chtt,
                                                                -1, aclass);
                        }
                /* Fixme: error condition if selector out of range */
                if (asn1_cb && !asn1_cb(ASN1_OP_I2D_POST, pval, it, NULL))
                                return 0;
                break;

                case ASN1_ITYPE_EXTERN:
                /* If new style i2d it does all the work */
                ef = it->funcs;
                return ef->asn1_ex_i2d(pval, out, it, tag, aclass);

                case ASN1_ITYPE_COMPAT:
                /* old style hackery... */
                cf = it->funcs;
                if (out)
                        p = *out;
                i = cf->asn1_i2d(*pval, out);
                /* Fixup for IMPLICIT tag: note this messes up for tags > 30,
                 * but so did the old code. Tags > 30 are very rare anyway.
                 */
                if (out && (tag != -1))
                        *p = aclass | tag | (*p & V_ASN1_CONSTRUCTED);
```



```
                return i;

                case ASN1_ITYPE_NDEF_SEQUENCE:
                /* Use indefinite length constructed if requested */
                if (aclass & ASN1_TFLG_NDEF) ndef = 2;
                /* fall through */

                case ASN1_ITYPE_SEQUENCE:
                i = asn1_enc_restore(&seqcontlen, out, pval, it);
                /* An error occurred */
                if (i < 0)
                        return 0;
                /* We have a valid cached encoding... */
                if (i > 0)
                        return seqcontlen;
                /* Otherwise carry on */
                seqcontlen = 0;
                /* If no IMPLICIT tagging set to SEQUENCE, UNIVERSAL */
                if (tag == -1)
                        {
                        tag = V_ASN1_SEQUENCE;
                        /* Retain any other flags in aclass */
                        aclass = (aclass & ~ASN1_TFLG_TAG_CLASS)
                                        | V_ASN1_UNIVERSAL;
                        }
                if (asn1_cb && !asn1_cb(ASN1_OP_I2D_PRE, pval, it, NULL))
                                return 0;
                /* First work out sequence content length */
                for (i = 0, tt = it->templates; i < it->tcount; tt++, i++)
                        {
                        const ASN1_TEMPLATE *seqtt;
                        ASN1_VALUE **pseqval;
                        seqtt = asn1_do_adb(pval, tt, 1);
                        if (!seqtt)
                                return 0;
                        pseqval = asn1_get_field_ptr(pval, seqtt);
                        /* FIXME: check for errors in enhanced version */
                        seqcontlen += asn1_template_ex_i2d(pseqval, NULL, seqtt,
                                                                -1, aclass);
                        }

                seqlen = ASN1_object_size(ndef, seqcontlen, tag);
                if (!out)
                        return seqlen;
                /* Output SEQUENCE header */
                ASN1_put_object(out, ndef, seqcontlen, tag, aclass);
                for (i = 0, tt = it->templates; i < it->tcount; tt++, i++)
                        {
                        const ASN1_TEMPLATE *seqtt;
                        ASN1_VALUE **pseqval;
                        seqtt = asn1_do_adb(pval, tt, 1);
                        if (!seqtt)
                                return 0;
                        pseqval = asn1_get_field_ptr(pval, seqtt);
                        /* FIXME: check for errors in enhanced version */
                        asn1_template_ex_i2d(pseqval, out, seqtt, -1, aclass);
                        }
                if (ndef == 2)
                        ASN1_put_eoc(out);
                if (asn1_cb && !asn1_cb(ASN1_OP_I2D_POST, pval, it, NULL))
                                return 0;
                return seqlen;

                default:
                return 0;
```

```
260                 }
261         return 0;
262         }

264 int ASN1_template_i2d(ASN1_VALUE **pval, unsigned char **out,
265                                                         const ASN1_TEMPLATE *tt)
266         {
267         return asn1_template_ex_i2d(pval, out, tt, -1, 0);
268         }

270 static int asn1_template_ex_i2d(ASN1_VALUE **pval, unsigned char **out,
271                                 const ASN1_TEMPLATE *tt, int tag, int iclass)
272         {
273         int i, ret, flags, ttag, tclass, ndef;
274         flags = tt->flags;
275         /* Work out tag and class to use: tagging may come
276          * either from the template or the arguments, not both
277          * because this would create ambiguity. Additionally
278          * the iclass argument may contain some additional flags
279          * which should be noted and passed down to other levels.
280          */
281         if (flags & ASN1_TFLG_TAG_MASK)
282                 {
283                 /* Error if argument and template tagging */
284                 if (tag != -1)
285                         /* FIXME: error code here */
286                         return -1;
287                 /* Get tagging from template */
288                 ttag = tt->tag;
289                 tclass = flags & ASN1_TFLG_TAG_CLASS;
290                 }
291         else if (tag != -1)
292                 {
293                 /* No template tagging, get from arguments */
294                 ttag = tag;
295                 tclass = iclass & ASN1_TFLG_TAG_CLASS;
296                 }
297         else
298                 {
299                 ttag = -1;
300                 tclass = 0;
301                 }
302         /*
303          * Remove any class mask from iflag.
304          */
305         iclass &= ~ASN1_TFLG_TAG_CLASS;

307         /* At this point 'ttag' contains the outer tag to use,
308          * 'tclass' is the class and iclass is any flags passed
309          * to this function.
310          */

312         /* if template and arguments require ndef, use it */
313         if ((flags & ASN1_TFLG_NDEF) && (iclass & ASN1_TFLG_NDEF))
314                 ndef = 2;
315         else ndef = 1;

317         if (flags & ASN1_TFLG_SK_MASK)
318                 {
319                 /* SET OF, SEQUENCE OF */
320                 STACK_OF(ASN1_VALUE) *sk = (STACK_OF(ASN1_VALUE) *)*pval;
321                 int isset, sktag, skaclass;
322                 int skcontlen, sklen;
323                 ASN1_VALUE *skitem;

325                 if (!*pval)
```



```
326                         return 0;

328                 if (flags & ASN1_TFLG_SET_OF)
329                         {
330                         isset = 1;
331                         /* 2 means we reorder */
332                         if (flags & ASN1_TFLG_SEQUENCE_OF)
333                                 isset = 2;
334                         }
335                 else isset = 0;

337                 /* Work out inner tag value: if EXPLICIT
338                  * or no tagging use underlying type.
339                  */
340                 if ((ttag != -1) && !(flags & ASN1_TFLG_EXPTAG))
341                         {
342                         sktag = ttag;
343                         skaclass = tclass;
344                         }
345                 else
346                         {
347                         skaclass = V_ASN1_UNIVERSAL;
348                         if (isset)
349                                 sktag = V_ASN1_SET;
350                         else sktag = V_ASN1_SEQUENCE;
351                         }

353                 /* Determine total length of items */
354                 skcontlen = 0;
355                 for (i = 0; i < sk_ASN1_VALUE_num(sk); i++)
356                         {
357                         skitem = sk_ASN1_VALUE_value(sk, i);
358                         skcontlen += ASN1_item_ex_i2d(&skitem, NULL,
359                                                 ASN1_ITEM_ptr(tt->item),
360                                                         -1, iclass);
361                         }
362                 sklen = ASN1_object_size(ndef, skcontlen, sktag);
363                 /* If EXPLICIT need length of surrounding tag */
364                 if (flags & ASN1_TFLG_EXPTAG)
365                         ret = ASN1_object_size(ndef, sklen, ttag);
366                 else ret = sklen;

368                 if (!out)
369                         return ret;

371                 /* Now encode this lot... */
372                 /* EXPLICIT tag */
373                 if (flags & ASN1_TFLG_EXPTAG)
374                         ASN1_put_object(out, ndef, sklen, ttag, tclass);
375                 /* SET or SEQUENCE and IMPLICIT tag */
376                 ASN1_put_object(out, ndef, skcontlen, sktag, skaclass);
377                 /* And the stuff itself */
378                 asn1_set_seq_out(sk, out, skcontlen, ASN1_ITEM_ptr(tt->item),
379                                                                 isset, iclass);
380                 if (ndef == 2)
381                         {
382                         ASN1_put_eoc(out);
383                         if (flags & ASN1_TFLG_EXPTAG)
384                                 ASN1_put_eoc(out);
385                         }

387                 return ret;
388                 }

390         if (flags & ASN1_TFLG_EXPTAG)
391                 {
```

```
392                 /* EXPLICIT tagging */
393                 /* Find length of tagged item */
394                 i = ASN1_item_ex_i2d(pval, NULL, ASN1_ITEM_ptr(tt->item),
395                                                                 -1, iclass);
396                 if (!i)
397                         return 0;
398                 /* Find length of EXPLICIT tag */
399                 ret = ASN1_object_size(ndef, i, ttag);
400                 if (out)
401                         {
402                         /* Output tag and item */
403                         ASN1_put_object(out, ndef, i, ttag, tclass);
404                         ASN1_item_ex_i2d(pval, out, ASN1_ITEM_ptr(tt->item),
405                                                                 -1, iclass);
406                         if (ndef == 2)
407                                 ASN1_put_eoc(out);
408                         }
409                 return ret;
410                 }

412         /* Either normal or IMPLICIT tagging: combine class and flags */
413         return ASN1_item_ex_i2d(pval, out, ASN1_ITEM_ptr(tt->item),
414                                                 ttag, tclass | iclass);

416 }

418 /* Temporary structure used to hold DER encoding of items for SET OF */

420 typedef struct {
421         unsigned char *data;
422         int length;
423         ASN1_VALUE *field;
424 } DER_ENC;

426 static int der_cmp(const void *a, const void *b)
427         {
428         const DER_ENC *d1 = a, *d2 = b;
429         int cmplen, i;
430         cmplen = (d1->length < d2->length) ? d1->length : d2->length;
431         i = memcmp(d1->data, d2->data, cmplen);
432         if (i)
433                 return i;
434         return d1->length - d2->length;
435         }

437 /* Output the content octets of SET OF or SEQUENCE OF */

439 static int asn1_set_seq_out(STACK_OF(ASN1_VALUE) *sk, unsigned char **out,
440                                         int skcontlen, const ASN1_ITEM *item,
441                                         int do_sort, int iclass)
442         {
443         int i;
444         ASN1_VALUE *skitem;
445         unsigned char *tmpdat = NULL, *p = NULL;
446         DER_ENC *derlst = NULL, *tder;
447         if (do_sort)
448                 {
449                 /* Don't need to sort less than 2 items */
450                 if (sk_ASN1_VALUE_num(sk) < 2)
451                         do_sort = 0;
452                 else
453                         {
454                         derlst = OPENSSL_malloc(sk_ASN1_VALUE_num(sk)
455                                                 * sizeof(*derlst));
456                         if (!derlst)
457                                 return 0;
```



```
458                         tmpdat = OPENSSL_malloc(skcontlen);
459                         if (!tmpdat)
460                                 {
461                                 OPENSSL_free(derlst);
462                                 return 0;
463                                 }
464                         }
465                 }
466         /* If not sorting just output each item */
467         if (!do_sort)
468                 {
469                 for (i = 0; i < sk_ASN1_VALUE_num(sk); i++)
470                         {
471                         skitem = sk_ASN1_VALUE_value(sk, i);
472                         ASN1_item_ex_i2d(&skitem, out, item, -1, iclass);
473                         }
474                 return 1;
475                 }
476         p = tmpdat;

478         /* Doing sort: build up a list of each member's DER encoding */
479         for (i = 0, tder = derlst; i < sk_ASN1_VALUE_num(sk); i++, tder++)
480                 {
481                 skitem = sk_ASN1_VALUE_value(sk, i);
482                 tder->data = p;
483                 tder->length = ASN1_item_ex_i2d(&skitem, &p, item, -1, iclass);
484                 tder->field = skitem;
485                 }

487         /* Now sort them */
488         qsort(derlst, sk_ASN1_VALUE_num(sk), sizeof(*derlst), der_cmp);
489         /* Output sorted DER encoding */
490         p = *out;
491         for (i = 0, tder = derlst; i < sk_ASN1_VALUE_num(sk); i++, tder++)
492                 {
493                 memcpy(p, tder->data, tder->length);
494                 p += tder->length;
495                 }
496         *out = p;
497         /* If do_sort is 2 then reorder the STACK */
498         if (do_sort == 2)
499                 {
500                 for (i = 0, tder = derlst; i < sk_ASN1_VALUE_num(sk);
501                                                         i++, tder++)
502                         (void)sk_ASN1_VALUE_set(sk, i, tder->field);
503                 }
504         OPENSSL_free(derlst);
505         OPENSSL_free(tmpdat);
506         return 1;
507         }

509 static int asn1_i2d_ex_primitive(ASN1_VALUE **pval, unsigned char **out,
510                                         const ASN1_ITEM *it, int tag, int aclass)
511         {
512         int len;
513         int utype;
514         int usetag;
515         int ndef = 0;

517         utype = it->utype;

519         /* Get length of content octets and maybe find
520          * out the underlying type.
521          */

523         len = asn1_ex_i2c(pval, NULL, &utype, it);
```

```
525         /* If SEQUENCE, SET or OTHER then header is
526          * included in pseudo content octets so don't
527          * include tag+length. We need to check here
528          * because the call to asn1_ex_i2c() could change
529          * utype.
530          */
531         if ((utype == V_ASN1_SEQUENCE) || (utype == V_ASN1_SET) ||
532            (utype == V_ASN1_OTHER))
533                 usetag = 0;
534         else usetag = 1;

536         /* -1 means omit type */

538         if (len == -1)
539                 return 0;

541         /* -2 return is special meaning use ndef */
542         if (len == -2)
543                 {
544                 ndef = 2;
545                 len = 0;
546                 }

548         /* If not implicitly tagged get tag from underlying type */
549         if (tag == -1) tag = utype;

551         /* Output tag+length followed by content octets */
552         if (out)
553                 {
554                 if (usetag)
555                         ASN1_put_object(out, ndef, len, tag, aclass);
556                 asn1_ex_i2c(pval, *out, &utype, it);
557                 if (ndef)
558                         ASN1_put_eoc(out);
559                 else
560                         *out += len;
561                 }

563         if (usetag)
564                 return ASN1_object_size(ndef, len, tag);
565         return len;
566         }

568 /* Produce content octets from a structure */

570 int asn1_ex_i2c(ASN1_VALUE **pval, unsigned char *cout, int *putype,
571                                 const ASN1_ITEM *it)
572         {
573         ASN1_BOOLEAN *tbool = NULL;
574         ASN1_STRING *strtmp;
575         ASN1_OBJECT *otmp;
576         int utype;
577         const unsigned char *cont;
578         unsigned char c;
579         int len;
580         const ASN1_PRIMITIVE_FUNCS *pf;
581         pf = it->funcs;
582         if (pf && pf->prim_i2c)
583                 return pf->prim_i2c(pval, cout, putype, it);

585         /* Should type be omitted? */
586         if ((it->itype != ASN1_ITYPE_PRIMITIVE)
587                 || (it->utype != V_ASN1_BOOLEAN))
588                 {
589                 if (!*pval) return -1;
```



```
590                 }

592         if (it->itype == ASN1_ITYPE_MSTRING)
593                 {
594                 /* If MSTRING type set the underlying type */
595                 strtmp = (ASN1_STRING *)*pval;
596                 utype = strtmp->type;
597                 *putype = utype;
598                 }
599         else if (it->utype == V_ASN1_ANY)
600                 {
601                 /* If ANY set type and pointer to value */
602                 ASN1_TYPE *typ;
603                 typ = (ASN1_TYPE *)*pval;
604                 utype = typ->type;
605                 *putype = utype;
606                 pval = &typ->value.asn1_value;
607                 }
608         else utype = *putype;

610         switch(utype)
611                 {
612                 case V_ASN1_OBJECT:
613                 otmp = (ASN1_OBJECT *)*pval;
614                 cont = otmp->data;
615                 len = otmp->length;
616                 break;

618                 case V_ASN1_NULL:
619                 cont = NULL;
620                 len = 0;
621                 break;

623                 case V_ASN1_BOOLEAN:
624                 tbool = (ASN1_BOOLEAN *)pval;
625                 if (*tbool == -1)
626                         return -1;
627                 if (it->utype != V_ASN1_ANY)
628                         {
629                         /* Default handling if value == size field then omit */
630                         if (*tbool && (it->size > 0))
631                                 return -1;
632                         if (!*tbool && !it->size)
633                                 return -1;
634                         }
635                 c = (unsigned char)*tbool;
636                 cont = &c;
637                 len = 1;
638                 break;

640                 case V_ASN1_BIT_STRING:
641                 return i2c_ASN1_BIT_STRING((ASN1_BIT_STRING *)*pval,
642                                                         cout ? &cout : NULL);
643                 break;

645                 case V_ASN1_INTEGER:
646                 case V_ASN1_NEG_INTEGER:
647                 case V_ASN1_ENUMERATED:
648                 case V_ASN1_NEG_ENUMERATED:
649                 /* These are all have the same content format
650                  * as ASN1_INTEGER
651                  */
652                 return i2c_ASN1_INTEGER((ASN1_INTEGER *)*pval,
653                                                         cout ? &cout : NULL);
654                 break;
```

```
656                 case V_ASN1_OCTET_STRING:
657                 case V_ASN1_NUMERICSTRING:
658                 case V_ASN1_PRINTABLESTRING:
659                 case V_ASN1_T61STRING:
660                 case V_ASN1_VIDEOTEXSTRING:
661                 case V_ASN1_IA5STRING:
662                 case V_ASN1_UTCTIME:
663                 case V_ASN1_GENERALIZEDTIME:
664                 case V_ASN1_GRAPHICSTRING:
665                 case V_ASN1_VISIBLESTRING:
666                 case V_ASN1_GENERALSTRING:
667                 case V_ASN1_UNIVERSALSTRING:
668                 case V_ASN1_BMPSTRING:
669                 case V_ASN1_UTF8STRING:
670                 case V_ASN1_SEQUENCE:
671                 case V_ASN1_SET:
672                 default:
673                 /* All based on ASN1_STRING and handled the same */
674                 strtmp = (ASN1_STRING *)*pval;
675                 /* Special handling for NDEF */
676                 if ((it->size == ASN1_TFLG_NDEF)
677                         && (strtmp->flags & ASN1_STRING_FLAG_NDEF))
678                         {
679                         if (cout)
680                                 {
681                                 strtmp->data = cout;
682                                 strtmp->length = 0;
683                                 }
684                         /* Special return code */
685                         return -2;
686                         }
687                 cont = strtmp->data;
688                 len = strtmp->length;

690                 break;

692                 }
693         if (cout && len)
694                 memcpy(cout, cont, len);
695         return len;
696         }
697 #endif /* ! codereview */
```

```
**********************************************************
    6920 Wed Aug 13 19:52:06 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/tasn_fre.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* tasn_fre.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stddef.h>
#include <openssl/asn1.h>
```



```
#include <openssl/asn1t.h>
#include <openssl/objects.h>

static void asn1_item_combine_free(ASN1_VALUE **pval, const ASN1_ITEM *it, int c

/* Free up an ASN1 structure */

void ASN1_item_free(ASN1_VALUE *val, const ASN1_ITEM *it)
        {
        asn1_item_combine_free(&val, it, 0);
        }

void ASN1_item_ex_free(ASN1_VALUE **pval, const ASN1_ITEM *it)
        {
        asn1_item_combine_free(pval, it, 0);
        }

static void asn1_item_combine_free(ASN1_VALUE **pval, const ASN1_ITEM *it, int c
        {
        const ASN1_TEMPLATE *tt = NULL, *seqtt;
        const ASN1_EXTERN_FUNCS *ef;
        const ASN1_COMPAT_FUNCS *cf;
        const ASN1_AUX *aux = it->funcs;
        ASN1_aux_cb *asn1_cb;
        int i;
        if (!pval)
                return;
        if ((it->itype != ASN1_ITYPE_PRIMITIVE) && !*pval)
                return;
        if (aux && aux->asn1_cb)
                asn1_cb = aux->asn1_cb;
        else
                asn1_cb = 0;

        switch(it->itype)
                {

                case ASN1_ITYPE_PRIMITIVE:
                if (it->templates)
                        ASN1_template_free(pval, it->templates);
                else
                        ASN1_primitive_free(pval, it);
                break;

                case ASN1_ITYPE_MSTRING:
                ASN1_primitive_free(pval, it);
                break;

                case ASN1_ITYPE_CHOICE:
                if (asn1_cb)
                        {
                        i = asn1_cb(ASN1_OP_FREE_PRE, pval, it, NULL);
                        if (i == 2)
                                return;
                        }
                i = asn1_get_choice_selector(pval, it);
                if ((i >= 0) && (i < it->tcount))
                        {
                        ASN1_VALUE **pchval;
                        tt = it->templates + i;
                        pchval = asn1_get_field_ptr(pval, tt);
                        ASN1_template_free(pchval, tt);
                        }
                if (asn1_cb)
                        asn1_cb(ASN1_OP_FREE_POST, pval, it, NULL);
                if (!combine)
```

```
128                     {
129                     OPENSSL_free(*pval);
130                     *pval = NULL;
131                     }
132             break;

134             case ASN1_ITYPE_COMPAT:
135             cf = it->funcs;
136             if (cf && cf->asn1_free)
137                     cf->asn1_free(*pval);
138             break;

140             case ASN1_ITYPE_EXTERN:
141             ef = it->funcs;
142             if (ef && ef->asn1_ex_free)
143                     ef->asn1_ex_free(pval, it);
144             break;

146             case ASN1_ITYPE_NDEF_SEQUENCE:
147             case ASN1_ITYPE_SEQUENCE:
148             if (asn1_do_lock(pval, -1, it) > 0)
149                     return;
150             if (asn1_cb)
151                     {
152                     i = asn1_cb(ASN1_OP_FREE_PRE, pval, it, NULL);
153                     if (i == 2)
154                             return;
155                     }
156             asn1_enc_free(pval, it);
157             /* If we free up as normal we will invalidate any
158              * ANY DEFINED BY field and we wont be able to
159              * determine the type of the field it defines. So
160              * free up in reverse order.
161              */
162             tt = it->templates + it->tcount - 1;
163             for (i = 0; i < it->tcount; tt--, i++)
164                     {
165                     ASN1_VALUE **pseqval;
166                     seqtt = asn1_do_adb(pval, tt, 0);
167                     if (!seqtt)
168                             continue;
169                     pseqval = asn1_get_field_ptr(pval, seqtt);
170                     ASN1_template_free(pseqval, seqtt);
171                     }
172             if (asn1_cb)
173                     asn1_cb(ASN1_OP_FREE_POST, pval, it, NULL);
174             if (!combine)
175                     {
176                     OPENSSL_free(*pval);
177                     *pval = NULL;
178                     }
179             break;
180             }
181     }

183 void ASN1_template_free(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt)
184     {
185     int i;
186     if (tt->flags & ASN1_TFLG_SK_MASK)
187             {
188             STACK_OF(ASN1_VALUE) *sk = (STACK_OF(ASN1_VALUE) *)*pval;
189             for (i = 0; i < sk_ASN1_VALUE_num(sk); i++)
190                     {
191                     ASN1_VALUE *vtmp;
192                     vtmp = sk_ASN1_VALUE_value(sk, i);
193                     asn1_item_combine_free(&vtmp, ASN1_ITEM_ptr(tt->item),
```



```
194                                                                     0);
195                     }
196             sk_ASN1_VALUE_free(sk);
197             *pval = NULL;
198             }
199     else
200             asn1_item_combine_free(pval, ASN1_ITEM_ptr(tt->item),
201                                                 tt->flags & ASN1_TFLG_COMBINE);
202     }

204 void ASN1_primitive_free(ASN1_VALUE **pval, const ASN1_ITEM *it)
205     {
206     int utype;
207     if (it)
208             {
209             const ASN1_PRIMITIVE_FUNCS *pf;
210             pf = it->funcs;
211             if (pf && pf->prim_free)
212                     {
213                     pf->prim_free(pval, it);
214                     return;
215                     }
216             }
217     /* Special case: if 'it' is NULL free contents of ASN1_TYPE */
218     if (!it)
219             {
220             ASN1_TYPE *typ = (ASN1_TYPE *)*pval;
221             utype = typ->type;
222             pval = &typ->value.asn1_value;
223             if (!*pval)
224                     return;
225             }
226     else if (it->itype == ASN1_ITYPE_MSTRING)
227             {
228             utype = -1;
229             if (!*pval)
230                     return;
231             }
232     else
233             {
234             utype = it->utype;
235             if ((utype != V_ASN1_BOOLEAN) && !*pval)
236                     return;
237             }

239     switch(utype)
240             {
241             case V_ASN1_OBJECT:
242             ASN1_OBJECT_free((ASN1_OBJECT *)*pval);
243             break;

245             case V_ASN1_BOOLEAN:
246             if (it)
247                     *(ASN1_BOOLEAN *)pval = it->size;
248             else
249                     *(ASN1_BOOLEAN *)pval = -1;
250             return;

252             case V_ASN1_NULL:
253             break;

255             case V_ASN1_ANY:
256             ASN1_primitive_free(pval, NULL);
257             OPENSSL_free(*pval);
258             break;
```

```
260                 default:
261                 ASN1_STRING_free((ASN1_STRING *)*pval);
262                 *pval = NULL;
263                 break;
264                 }
265         *pval = NULL;
266         }
267 #endif /* ! codereview */
```

```
**********************************************************
    9498 Wed Aug 13 19:52:06 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/tasn_new.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* tasn_new.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stddef.h>
#include <openssl/asn1.h>
```



```
#include <openssl/objects.h>
#include <openssl/err.h>
#include <openssl/asn1t.h>
#include <string.h>

static int asn1_item_ex_combine_new(ASN1_VALUE **pval, const ASN1_ITEM *it,
                                    int combine);
static void asn1_item_clear(ASN1_VALUE **pval, const ASN1_ITEM *it);
static void asn1_template_clear(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt);
static void asn1_primitive_clear(ASN1_VALUE **pval, const ASN1_ITEM *it);

ASN1_VALUE *ASN1_item_new(const ASN1_ITEM *it)
        {
        ASN1_VALUE *ret = NULL;
        if (ASN1_item_ex_new(&ret, it) > 0)
                return ret;
        return NULL;
        }

/* Allocate an ASN1 structure */

int ASN1_item_ex_new(ASN1_VALUE **pval, const ASN1_ITEM *it)
        {
        return asn1_item_ex_combine_new(pval, it, 0);
        }

static int asn1_item_ex_combine_new(ASN1_VALUE **pval, const ASN1_ITEM *it,
                                    int combine)
        {
        const ASN1_TEMPLATE *tt = NULL;
        const ASN1_COMPAT_FUNCS *cf;
        const ASN1_EXTERN_FUNCS *ef;
        const ASN1_AUX *aux = it->funcs;
        ASN1_aux_cb *asn1_cb;
        ASN1_VALUE **pseqval;
        int i;
        if (aux && aux->asn1_cb)
                asn1_cb = aux->asn1_cb;
        else
                asn1_cb = 0;

        if (!combine) *pval = NULL;

#ifdef CRYPTO_MDEBUG
        if (it->sname)
                CRYPTO_push_info(it->sname);
#endif

        switch(it->itype)
                {

                case ASN1_ITYPE_EXTERN:
                ef = it->funcs;
                if (ef && ef->asn1_ex_new)
                        {
                        if (!ef->asn1_ex_new(pval, it))
                                goto memerr;
                        }
                break;

                case ASN1_ITYPE_COMPAT:
                cf = it->funcs;
                if (cf && cf->asn1_new) {
                        *pval = cf->asn1_new();
                        if (!*pval)
                                goto memerr;
```

```
128                 }
129                 break;

131                 case ASN1_ITYPE_PRIMITIVE:
132                 if (it->templates)
133                         {
134                         if (!ASN1_template_new(pval, it->templates))
135                                 goto memerr;
136                         }
137                 else if (!ASN1_primitive_new(pval, it))
138                                 goto memerr;
139                 break;

141                 case ASN1_ITYPE_MSTRING:
142                 if (!ASN1_primitive_new(pval, it))
143                                 goto memerr;
144                 break;

146                 case ASN1_ITYPE_CHOICE:
147                 if (asn1_cb)
148                         {
149                         i = asn1_cb(ASN1_OP_NEW_PRE, pval, it, NULL);
150                         if (!i)
151                                 goto auxerr;
152                         if (i==2)
153                                 {
154 #ifdef CRYPTO_MDEBUG
155                                 if (it->sname)
156                                         CRYPTO_pop_info();
157 #endif
158                                 return 1;
159                                 }
160                         }
161                 if (!combine)
162                         {
163                         *pval = OPENSSL_malloc(it->size);
164                         if (!*pval)
165                                 goto memerr;
166                         memset(*pval, 0, it->size);
167                         }
168                 asn1_set_choice_selector(pval, -1, it);
169                 if (asn1_cb && !asn1_cb(ASN1_OP_NEW_POST, pval, it, NULL))
170                                 goto auxerr;
171                 break;

173                 case ASN1_ITYPE_NDEF_SEQUENCE:
174                 case ASN1_ITYPE_SEQUENCE:
175                 if (asn1_cb)
176                         {
177                         i = asn1_cb(ASN1_OP_NEW_PRE, pval, it, NULL);
178                         if (!i)
179                                 goto auxerr;
180                         if (i==2)
181                                 {
182 #ifdef CRYPTO_MDEBUG
183                                 if (it->sname)
184                                         CRYPTO_pop_info();
185 #endif
186                                 return 1;
187                                 }
188                         }
189                 if (!combine)
190                         {
191                         *pval = OPENSSL_malloc(it->size);
192                         if (!*pval)
193                                 goto memerr;
```



```
194                         memset(*pval, 0, it->size);
195                         asn1_do_lock(pval, 0, it);
196                         asn1_enc_init(pval, it);
197                         }
198                 for (i = 0, tt = it->templates; i < it->tcount; tt++, i++)
199                         {
200                         pseqval = asn1_get_field_ptr(pval, tt);
201                         if (!ASN1_template_new(pseqval, tt))
202                                 goto memerr;
203                         }
204                 if (asn1_cb && !asn1_cb(ASN1_OP_NEW_POST, pval, it, NULL))
205                                 goto auxerr;
206                 break;
207         }
208 #ifdef CRYPTO_MDEBUG
209         if (it->sname) CRYPTO_pop_info();
210 #endif
211         return 1;

213         memerr:
214         ASN1err(ASN1_F_ASN1_ITEM_EX_COMBINE_NEW, ERR_R_MALLOC_FAILURE);
215 #ifdef CRYPTO_MDEBUG
216         if (it->sname) CRYPTO_pop_info();
217 #endif
218         return 0;

220         auxerr:
221         ASN1err(ASN1_F_ASN1_ITEM_EX_COMBINE_NEW, ASN1_R_AUX_ERROR);
222         ASN1_item_ex_free(pval, it);
223 #ifdef CRYPTO_MDEBUG
224         if (it->sname) CRYPTO_pop_info();
225 #endif
226         return 0;

228         }

230 static void asn1_item_clear(ASN1_VALUE **pval, const ASN1_ITEM *it)
231         {
232         const ASN1_EXTERN_FUNCS *ef;

234         switch(it->itype)
235                 {

237                 case ASN1_ITYPE_EXTERN:
238                 ef = it->funcs;
239                 if (ef && ef->asn1_ex_clear)
240                         ef->asn1_ex_clear(pval, it);
241                 else *pval = NULL;
242                 break;


245                 case ASN1_ITYPE_PRIMITIVE:
246                 if (it->templates)
247                         asn1_template_clear(pval, it->templates);
248                 else
249                         asn1_primitive_clear(pval, it);
250                 break;

252                 case ASN1_ITYPE_MSTRING:
253                 asn1_primitive_clear(pval, it);
254                 break;

256                 case ASN1_ITYPE_COMPAT:
257                 case ASN1_ITYPE_CHOICE:
258                 case ASN1_ITYPE_SEQUENCE:
259                 case ASN1_ITYPE_NDEF_SEQUENCE:
```

```
260                 *pval = NULL;
261                 break;
262                 }
263         }


266 int ASN1_template_new(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt)
267         {
268         const ASN1_ITEM *it = ASN1_ITEM_ptr(tt->item);
269         int ret;
270         if (tt->flags & ASN1_TFLG_OPTIONAL)
271                 {
272                 asn1_template_clear(pval, tt);
273                 return 1;
274                 }
275         /* If ANY DEFINED BY nothing to do */

277         if (tt->flags & ASN1_TFLG_ADB_MASK)
278                 {
279                 *pval = NULL;
280                 return 1;
281                 }
282 #ifdef CRYPTO_MDEBUG
283         if (tt->field_name)
284                 CRYPTO_push_info(tt->field_name);
285 #endif
286         /* If SET OF or SEQUENCE OF, its a STACK */
287         if (tt->flags & ASN1_TFLG_SK_MASK)
288                 {
289                 STACK_OF(ASN1_VALUE) *skval;
290                 skval = sk_ASN1_VALUE_new_null();
291                 if (!skval)
292                         {
293                         ASN1err(ASN1_F_ASN1_TEMPLATE_NEW, ERR_R_MALLOC_FAILURE);
294                         ret = 0;
295                         goto done;
296                         }
297                 *pval = (ASN1_VALUE *)skval;
298                 ret = 1;
299                 goto done;
300                 }
301         /* Otherwise pass it back to the item routine */
302         ret = asn1_item_ex_combine_new(pval, it, tt->flags & ASN1_TFLG_COMBINE);
303         done:
304 #ifdef CRYPTO_MDEBUG
305         if (it->sname)
306                 CRYPTO_pop_info();
307 #endif
308         return ret;
309         }

311 static void asn1_template_clear(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt)
312         {
313         /* If ADB or STACK just NULL the field */
314         if (tt->flags & (ASN1_TFLG_ADB_MASK|ASN1_TFLG_SK_MASK))
315                 *pval = NULL;
316         else
317                 asn1_item_clear(pval, ASN1_ITEM_ptr(tt->item));
318         }


321 /* NB: could probably combine most of the real XXX_new() behaviour and junk
322  * all the old functions.
323  */

325 int ASN1_primitive_new(ASN1_VALUE **pval, const ASN1_ITEM *it)
```



```
326         {
327         ASN1_TYPE *typ;
328         ASN1_STRING *str;
329         int utype;

331         if (it && it->funcs)
332                 {
333                 const ASN1_PRIMITIVE_FUNCS *pf = it->funcs;
334                 if (pf->prim_new)
335                         return pf->prim_new(pval, it);
336                 }

338         if (!it || (it->itype == ASN1_ITYPE_MSTRING))
339                 utype = -1;
340         else
341                 utype = it->utype;
342         switch(utype)
343                 {
344                 case V_ASN1_OBJECT:
345                 *pval = (ASN1_VALUE *)OBJ_nid2obj(NID_undef);
346                 return 1;

348                 case V_ASN1_BOOLEAN:
349                 *(ASN1_BOOLEAN *)pval = it->size;
350                 return 1;

352                 case V_ASN1_NULL:
353                 *pval = (ASN1_VALUE *)1;
354                 return 1;

356                 case V_ASN1_ANY:
357                 typ = OPENSSL_malloc(sizeof(ASN1_TYPE));
358                 if (!typ)
359                         return 0;
360                 typ->value.ptr = NULL;
361                 typ->type = -1;
362                 *pval = (ASN1_VALUE *)typ;
363                 break;

365                 default:
366                 str = ASN1_STRING_type_new(utype);
367                 if (it->itype == ASN1_ITYPE_MSTRING && str)
368                         str->flags |= ASN1_STRING_FLAG_MSTRING;
369                 *pval = (ASN1_VALUE *)str;
370                 break;
371                 }
372         if (*pval)
373                 return 1;
374         return 0;
375         }

377 static void asn1_primitive_clear(ASN1_VALUE **pval, const ASN1_ITEM *it)
378         {
379         int utype;
380         if (it && it->funcs)
381                 {
382                 const ASN1_PRIMITIVE_FUNCS *pf = it->funcs;
383                 if (pf->prim_clear)
384                         pf->prim_clear(pval, it);
385                 else
386                         *pval = NULL;
387                 return;
388                 }
389         if (!it || (it->itype == ASN1_ITYPE_MSTRING))
390                 utype = -1;
391         else
```

```
392                 utype = it->utype;
393         if (utype == V_ASN1_BOOLEAN)
394                 *(ASN1_BOOLEAN *)pval = it->size;
395         else *pval = NULL;
396         }
397 #endif /* ! codereview */
```

```
**********************************************************
   14492 Wed Aug 13 19:52:06 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/tasn_prn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* tasn_prn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000,2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stddef.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/objects.h>
#include <openssl/buffer.h>
#include <openssl/err.h>
#include <openssl/x509v3.h>
#include "asn1_locl.h"

/* Print routines.
 */

/* ASN1_PCTX routines */

ASN1_PCTX default_pctx =
        {
        ASN1_PCTX_FLAGS_SHOW_ABSENT,    /* flags */
        0,      /* nm_flags */
        0,      /* cert_flags */
        0,      /* oid_flags */
        0       /* str_flags */
        };


ASN1_PCTX *ASN1_PCTX_new(void)
        {
        ASN1_PCTX *ret;
        ret = OPENSSL_malloc(sizeof(ASN1_PCTX));
        if (ret == NULL)
                {
                ASN1err(ASN1_F_ASN1_PCTX_NEW, ERR_R_MALLOC_FAILURE);
                return NULL;
                }
        ret->flags = 0;
        ret->nm_flags = 0;
        ret->cert_flags = 0;
        ret->oid_flags = 0;
        ret->str_flags = 0;
        return ret;
        }

void ASN1_PCTX_free(ASN1_PCTX *p)
        {
        OPENSSL_free(p);
        }

unsigned long ASN1_PCTX_get_flags(ASN1_PCTX *p)
        {
        return p->flags;
        }

void ASN1_PCTX_set_flags(ASN1_PCTX *p, unsigned long flags)
        {
        p->flags = flags;
        }

unsigned long ASN1_PCTX_get_nm_flags(ASN1_PCTX *p)
        {
        return p->nm_flags;
        }

void ASN1_PCTX_set_nm_flags(ASN1_PCTX *p, unsigned long flags)
        {
        p->nm_flags = flags;
        }

unsigned long ASN1_PCTX_get_cert_flags(ASN1_PCTX *p)
```

```
128         {
129         return p->cert_flags;
130         }

132 void ASN1_PCTX_set_cert_flags(ASN1_PCTX *p, unsigned long flags)
133         {
134         p->cert_flags = flags;
135         }

137 unsigned long ASN1_PCTX_get_oid_flags(ASN1_PCTX *p)
138         {
139         return p->oid_flags;
140         }

142 void ASN1_PCTX_set_oid_flags(ASN1_PCTX *p, unsigned long flags)
143         {
144         p->oid_flags = flags;
145         }

147 unsigned long ASN1_PCTX_get_str_flags(ASN1_PCTX *p)
148         {
149         return p->str_flags;
150         }

152 void ASN1_PCTX_set_str_flags(ASN1_PCTX *p, unsigned long flags)
153         {
154         p->str_flags = flags;
155         }

157 /* Main print routines */

159 static int asn1_item_print_ctx(BIO *out, ASN1_VALUE **fld, int indent,
160                                const ASN1_ITEM *it,
161                                const char *fname, const char *sname,
162                                int nohdr, const ASN1_PCTX *pctx);

164 int asn1_template_print_ctx(BIO *out, ASN1_VALUE **fld, int indent,
165                                const ASN1_TEMPLATE *tt, const ASN1_PCTX *pctx);

167 static int asn1_primitive_print(BIO *out, ASN1_VALUE **fld,
168                                const ASN1_ITEM *it, int indent,
169                                const char *fname, const char *sname,
170                                const ASN1_PCTX *pctx);

172 static int asn1_print_fsname(BIO *out, int indent,
173                         const char *fname, const char *sname,
174                         const ASN1_PCTX *pctx);

176 int ASN1_item_print(BIO *out, ASN1_VALUE *ifld, int indent,
177                                const ASN1_ITEM *it, const ASN1_PCTX *pctx)
178         {
179         const char *sname;
180         if (pctx == NULL)
181                 pctx = &default_pctx;
182         if (pctx->flags & ASN1_PCTX_FLAGS_NO_STRUCT_NAME)
183                 sname = NULL;
184         else
185                 sname = it->sname;
186         return asn1_item_print_ctx(out, &ifld, indent, it,
187                                                         NULL, sname, 0, pctx);
188         }

190 static int asn1_item_print_ctx(BIO *out, ASN1_VALUE **fld, int indent,
191                                const ASN1_ITEM *it,
192                                const char *fname, const char *sname,
193                                int nohdr, const ASN1_PCTX *pctx)
```



```
194         {
195         const ASN1_TEMPLATE *tt;
196         const ASN1_EXTERN_FUNCS *ef;
197         ASN1_VALUE **tmpfld;
198         const ASN1_AUX *aux = it->funcs;
199         ASN1_aux_cb *asn1_cb;
200         ASN1_PRINT_ARG parg;
201         int i;
202         if (aux && aux->asn1_cb)
203                 {
204                 parg.out = out;
205                 parg.indent = indent;
206                 parg.pctx = pctx;
207                 asn1_cb = aux->asn1_cb;
208                 }
209         else asn1_cb = 0;

211         if(*fld == NULL)
212                 {
213                 if (pctx->flags & ASN1_PCTX_FLAGS_SHOW_ABSENT)
214                         {
215                         if (!nohdr && !asn1_print_fsname(out, indent,
216                                                         fname, sname, pctx))
217                                 return 0;
218                         if (BIO_puts(out, "<ABSENT>\n") <= 0)
219                                 return 0;
220                         }
221                 return 1;
222                 }

224         switch(it->itype)
225                 {
226                 case ASN1_ITYPE_PRIMITIVE:
227                 if(it->templates)
228                         {
229                         if (!asn1_template_print_ctx(out, fld, indent,
230                                                         it->templates, pctx))
231                                 return 0;
232                         }
233                 /* fall thru */
234                 case ASN1_ITYPE_MSTRING:
235                 if (!asn1_primitive_print(out, fld, it,
236                                 indent, fname, sname,pctx))
237                         return 0;
238                 break;

240                 case ASN1_ITYPE_EXTERN:
241                 if (!nohdr && !asn1_print_fsname(out, indent, fname, sname, pctx
242                         return 0;
243                 /* Use new style print routine if possible */
244                 ef = it->funcs;
245                 if (ef && ef->asn1_ex_print)
246                         {
247                         i = ef->asn1_ex_print(out, fld, indent, "", pctx);
248                         if (!i)
249                                 return 0;
250                         if ((i == 2) && (BIO_puts(out, "\n") <= 0))
251                                 return 0;
252                         return 1;
253                         }
254                 else if (sname &&
255                         BIO_printf(out, ":EXTERNAL TYPE %s\n", sname) <= 0)
256                         return 0;
257                 break;

259                 case ASN1_ITYPE_CHOICE:
```

```
#if 0
                if (!nohdr && !asn1_print_fsname(out, indent, fname, sname, pctx
                        return 0;
#endif
                /* CHOICE type, get selector */
                i = asn1_get_choice_selector(fld, it);
                /* This should never happen... */
                if((i < 0) || (i >= it->tcount))
                        {
                        if (BIO_printf(out,
                                "ERROR: selector [%d] invalid\n", i) <= 0)
                                return 0;
                        return 1;
                        }
                tt = it->templates + i;
                tmpfld = asn1_get_field_ptr(fld, tt);
                if (!asn1_template_print_ctx(out, tmpfld, indent, tt, pctx))
                        return 0;
                break;

                case ASN1_ITYPE_SEQUENCE:
                case ASN1_ITYPE_NDEF_SEQUENCE:
                if (!nohdr && !asn1_print_fsname(out, indent, fname, sname, pctx
                        return 0;
                if (fname || sname)
                        {
                        if (pctx->flags & ASN1_PCTX_FLAGS_SHOW_SEQUENCE)
                                {
                                if (BIO_puts(out, " {\n") <= 0)
                                        return 0;
                                }
                        else
                                {
                                if (BIO_puts(out, "\n") <= 0)
                                        return 0;
                                }
                        }

                if (asn1_cb)
                        {
                        i = asn1_cb(ASN1_OP_PRINT_PRE, fld, it, &parg);
                        if (i == 0)
                                return 0;
                        if (i == 2)
                                return 1;
                        }

                /* Print each field entry */
                for(i = 0, tt = it->templates; i < it->tcount; i++, tt++)
                        {
                        const ASN1_TEMPLATE *seqtt;
                        seqtt = asn1_do_adb(fld, tt, 1);
                        tmpfld = asn1_get_field_ptr(fld, seqtt);
                        if (!asn1_template_print_ctx(out, tmpfld,
                                                indent + 2, seqtt, pctx))
                                return 0;
                        }
                if (pctx->flags & ASN1_PCTX_FLAGS_SHOW_SEQUENCE)
                        {
                        if (BIO_printf(out, "%*s}\n", indent, "") < 0)
                                return 0;
                        }

                if (asn1_cb)
                        {
                        i = asn1_cb(ASN1_OP_PRINT_POST, fld, it, &parg);
```



```
                        if (i == 0)
                                return 0;
                        }
                break;

                default:
                BIO_printf(out, "Unprocessed type %d\n", it->itype);
                return 0;
                }

        return 1;
        }

int asn1_template_print_ctx(BIO *out, ASN1_VALUE **fld, int indent,
                                const ASN1_TEMPLATE *tt, const ASN1_PCTX *pctx)
        {
        int i, flags;
        const char *sname, *fname;
        flags = tt->flags;
        if(pctx->flags & ASN1_PCTX_FLAGS_SHOW_FIELD_STRUCT_NAME)
                sname = ASN1_ITEM_ptr(tt->item)->sname;
        else
                sname = NULL;
        if(pctx->flags & ASN1_PCTX_FLAGS_NO_FIELD_NAME)
                fname = NULL;
        else
                fname = tt->field_name;
        if(flags & ASN1_TFLG_SK_MASK)
                {
                char *tname;
                ASN1_VALUE *skitem;
                STACK_OF(ASN1_VALUE) *stack;

                /* SET OF, SEQUENCE OF */
                if (fname)
                        {
                        if(pctx->flags & ASN1_PCTX_FLAGS_SHOW_SSOF)
                                {
                                if(flags & ASN1_TFLG_SET_OF)
                                        tname = "SET";
                                else
                                        tname = "SEQUENCE";
                                if (BIO_printf(out, "%*s%s OF %s {\n",
                                        indent, "", tname, tt->field_name) <= 0)
                                        return 0;
                                }
                        else if (BIO_printf(out, "%*s%s:\n", indent, "",
                                        fname) <= 0)
                                return 0;
                        }
                stack = (STACK_OF(ASN1_VALUE) *)*fld;
                for(i = 0; i < sk_ASN1_VALUE_num(stack); i++)
                        {
                        if ((i > 0) && (BIO_puts(out, "\n") <= 0))
                                return 0;

                        skitem = sk_ASN1_VALUE_value(stack, i);
                        if (!asn1_item_print_ctx(out, &skitem, indent + 2,
                                ASN1_ITEM_ptr(tt->item), NULL, NULL, 1, pctx))
                                return 0;
                        }
                if (!i && BIO_printf(out, "%*s<EMPTY>\n", indent + 2, "") <= 0)
                                return 0;
                if(pctx->flags & ASN1_PCTX_FLAGS_SHOW_SEQUENCE)
                        {
                        if (BIO_printf(out, "%*s}\n", indent, "") <= 0)
```

```
392                                 return 0;
393                         }
394                 return 1;
395                 }
396         return asn1_item_print_ctx(out, fld, indent, ASN1_ITEM_ptr(tt->item),
397                                                         fname, sname, 0, pctx);
398         }

400 static int asn1_print_fsname(BIO *out, int indent,
401                         const char *fname, const char *sname,
402                         const ASN1_PCTX *pctx)
403         {
404         static char spaces[] = "                    ";
405         const int nspaces = sizeof(spaces) - 1;

407 #if 0
408         if (!sname && !fname)
409                 return 1;
410 #endif

412         while (indent > nspaces)
413                 {
414                 if (BIO_write(out, spaces, nspaces) != nspaces)
415                         return 0;
416                 indent -= nspaces;
417                 }
418         if (BIO_write(out, spaces, indent) != indent)
419                 return 0;
420         if (pctx->flags & ASN1_PCTX_FLAGS_NO_STRUCT_NAME)
421                 sname = NULL;
422         if (pctx->flags & ASN1_PCTX_FLAGS_NO_FIELD_NAME)
423                 fname = NULL;
424         if (!sname && !fname)
425                 return 1;
426         if (fname)
427                 {
428                 if (BIO_puts(out, fname) <= 0)
429                         return 0;
430                 }
431         if (sname)
432                 {
433                 if (fname)
434                         {
435                         if (BIO_printf(out, " (%s)", sname) <= 0)
436                                 return 0;
437                         }
438                 else
439                         {
440                         if (BIO_puts(out, sname) <= 0)
441                                 return 0;
442                         }
443                 }
444         if (BIO_write(out, ": ", 2) != 2)
445                 return 0;
446         return 1;
447         }

449 static int asn1_print_boolean_ctx(BIO *out, int boolval,
450                                                         const ASN1_PCTX *pctx)
451         {
452         const char *str;
453         switch (boolval)
454                 {
455                 case -1:
456                 str = "BOOL ABSENT";
457                 break;
```



```
459                 case 0:
460                 str = "FALSE";
461                 break;

463                 default:
464                 str = "TRUE";
465                 break;

467                 }

469         if (BIO_puts(out, str) <= 0)
470                 return 0;
471         return 1;

473         }

475 static int asn1_print_integer_ctx(BIO *out, ASN1_INTEGER *str,
476                                                         const ASN1_PCTX *pctx)
477         {
478         char *s;
479         int ret = 1;
480         s = i2s_ASN1_INTEGER(NULL, str);
481         if (BIO_puts(out, s) <= 0)
482                 ret = 0;
483         OPENSSL_free(s);
484         return ret;
485         }

487 static int asn1_print_oid_ctx(BIO *out, const ASN1_OBJECT *oid,
488                                                         const ASN1_PCTX *pctx)
489         {
490         char objbuf[80];
491         const char *ln;
492         ln = OBJ_nid2ln(OBJ_obj2nid(oid));
493         if(!ln)
494                 ln = "";
495         OBJ_obj2txt(objbuf, sizeof objbuf, oid, 1);
496         if (BIO_printf(out, "%s (%s)", ln, objbuf) <= 0)
497                 return 0;
498         return 1;
499         }

501 static int asn1_print_obstring_ctx(BIO *out, ASN1_STRING *str, int indent,
502                                                         const ASN1_PCTX *pctx)
503         {
504         if (str->type == V_ASN1_BIT_STRING)
505                 {
506                 if (BIO_printf(out, " (%ld unused bits)\n",
507                                                 str->flags & 0x7) <= 0)
508                                 return 0;
509                 }
510         else if (BIO_puts(out, "\n") <= 0)
511                 return 0;
512         if ((str->length > 0)
513                 && BIO_dump_indent(out, (char *)str->data, str->length,
514                                 indent + 2) <= 0)
515                 return 0;
516         return 1;
517         }

519 static int asn1_primitive_print(BIO *out, ASN1_VALUE **fld,
520                                 const ASN1_ITEM *it, int indent,
521                                 const char *fname, const char *sname,
522                                 const ASN1_PCTX *pctx)
523         {
```

```
524         long utype;
525         ASN1_STRING *str;
526         int ret = 1, needlf = 1;
527         const char *pname;
528         const ASN1_PRIMITIVE_FUNCS *pf;
529         pf = it->funcs;
530         if (!asn1_print_fsname(out, indent, fname, sname, pctx))
531                         return 0;
532         if (pf && pf->prim_print)
533                 return pf->prim_print(out, fld, it, indent, pctx);
534         str = (ASN1_STRING *)*fld;
535         if (it->itype == ASN1_ITYPE_MSTRING)
536                 utype = str->type & ~V_ASN1_NEG;
537         else
538                 utype = it->utype;
539         if (utype == V_ASN1_ANY)
540                 {
541                 ASN1_TYPE *atype = (ASN1_TYPE *)*fld;
542                 utype = atype->type;
543                 fld = &atype->value.asn1_value;
544                 str = (ASN1_STRING *)*fld;
545                 if (pctx->flags & ASN1_PCTX_FLAGS_NO_ANY_TYPE)
546                         pname = NULL;
547                 else
548                         pname = ASN1_tag2str(utype);
549                 }
550         else
551                 {
552                 if (pctx->flags & ASN1_PCTX_FLAGS_SHOW_TYPE)
553                         pname = ASN1_tag2str(utype);
554                 else
555                         pname = NULL;
556                 }

558         if (utype == V_ASN1_NULL)
559                 {
560                 if (BIO_puts(out, "NULL\n") <= 0)
561                         return 0;
562                 return 1;
563                 }

565         if (pname)
566                 {
567                 if (BIO_puts(out, pname) <= 0)
568                         return 0;
569                 if (BIO_puts(out, ":") <= 0)
570                         return 0;
571                 }

573         switch (utype)
574                 {
575                 case V_ASN1_BOOLEAN:
576                         {
577                         int boolval = *(int *)fld;
578                         if (boolval == -1)
579                                 boolval = it->size;
580                         ret = asn1_print_boolean_ctx(out, boolval, pctx);
581                         }
582                 break;

584                 case V_ASN1_INTEGER:
585                 case V_ASN1_ENUMERATED:
586                 ret = asn1_print_integer_ctx(out, str, pctx);
587                 break;

589                 case V_ASN1_UTCTIME:
```



```
590                 ret = ASN1_UTCTIME_print(out, str);
591                 break;

593                 case V_ASN1_GENERALIZEDTIME:
594                 ret = ASN1_GENERALIZEDTIME_print(out, str);
595                 break;

597                 case V_ASN1_OBJECT:
598                 ret = asn1_print_oid_ctx(out, (const ASN1_OBJECT *)*fld, pctx);
599                 break;

601                 case V_ASN1_OCTET_STRING:
602                 case V_ASN1_BIT_STRING:
603                 ret = asn1_print_obstring_ctx(out, str, indent, pctx);
604                 needlf = 0;
605                 break;

607                 case V_ASN1_SEQUENCE:
608                 case V_ASN1_SET:
609                 case V_ASN1_OTHER:
610                 if (BIO_puts(out, "\n") <= 0)
611                         return 0;
612                 if (ASN1_parse_dump(out, str->data, str->length,
613                                                 indent, 0) <= 0)
614                         ret = 0;
615                 needlf = 0;
616                 break;

618                 default:
619                 ret = ASN1_STRING_print_ex(out, str, pctx->str_flags);

621                 }
622         if (!ret)
623                 return 0;
624         if (needlf && BIO_puts(out, "\n") <= 0)
625                 return 0;
626         return 1;
627         }
628 #endif /* ! codereview */
```

```
**********************************************************
    5548 Wed Aug 13 19:52:06 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/tasn_typ.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* tasn_typ.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#include <stdio.h>
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
```



```
/* Declarations for string types */


IMPLEMENT_ASN1_TYPE(ASN1_INTEGER)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_INTEGER)

IMPLEMENT_ASN1_TYPE(ASN1_ENUMERATED)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_ENUMERATED)

IMPLEMENT_ASN1_TYPE(ASN1_BIT_STRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_BIT_STRING)

IMPLEMENT_ASN1_TYPE(ASN1_OCTET_STRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_OCTET_STRING)

IMPLEMENT_ASN1_TYPE(ASN1_NULL)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_NULL)

IMPLEMENT_ASN1_TYPE(ASN1_OBJECT)

IMPLEMENT_ASN1_TYPE(ASN1_UTF8STRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_UTF8STRING)

IMPLEMENT_ASN1_TYPE(ASN1_PRINTABLESTRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_PRINTABLESTRING)

IMPLEMENT_ASN1_TYPE(ASN1_T61STRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_T61STRING)

IMPLEMENT_ASN1_TYPE(ASN1_IA5STRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_IA5STRING)

IMPLEMENT_ASN1_TYPE(ASN1_GENERALSTRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_GENERALSTRING)

IMPLEMENT_ASN1_TYPE(ASN1_UTCTIME)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_UTCTIME)

IMPLEMENT_ASN1_TYPE(ASN1_GENERALIZEDTIME)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_GENERALIZEDTIME)

IMPLEMENT_ASN1_TYPE(ASN1_VISIBLESTRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_VISIBLESTRING)

IMPLEMENT_ASN1_TYPE(ASN1_UNIVERSALSTRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_UNIVERSALSTRING)

IMPLEMENT_ASN1_TYPE(ASN1_BMPSTRING)
IMPLEMENT_ASN1_FUNCTIONS(ASN1_BMPSTRING)

IMPLEMENT_ASN1_TYPE(ASN1_ANY)

/* Just swallow an ASN1_SEQUENCE in an ASN1_STRING */
IMPLEMENT_ASN1_TYPE(ASN1_SEQUENCE)

IMPLEMENT_ASN1_FUNCTIONS_fname(ASN1_TYPE, ASN1_ANY, ASN1_TYPE)

/* Multistring types */

IMPLEMENT_ASN1_MSTRING(ASN1_PRINTABLE, B_ASN1_PRINTABLE)
IMPLEMENT_ASN1_FUNCTIONS_name(ASN1_STRING, ASN1_PRINTABLE)

IMPLEMENT_ASN1_MSTRING(DISPLAYTEXT, B_ASN1_DISPLAYTEXT)
IMPLEMENT_ASN1_FUNCTIONS_name(ASN1_STRING, DISPLAYTEXT)

IMPLEMENT_ASN1_MSTRING(DIRECTORYSTRING, B_ASN1_DIRECTORYSTRING)
```

```
128 IMPLEMENT_ASN1_FUNCTIONS_name(ASN1_STRING, DIRECTORYSTRING)

130 /* Three separate BOOLEAN type: normal, DEFAULT TRUE and DEFAULT FALSE */
131 IMPLEMENT_ASN1_TYPE_ex(ASN1_BOOLEAN, ASN1_BOOLEAN, -1)
132 IMPLEMENT_ASN1_TYPE_ex(ASN1_TBOOLEAN, ASN1_BOOLEAN, 1)
133 IMPLEMENT_ASN1_TYPE_ex(ASN1_FBOOLEAN, ASN1_BOOLEAN, 0)

135 /* Special, OCTET STRING with indefinite length constructed support */

137 IMPLEMENT_ASN1_TYPE_ex(ASN1_OCTET_STRING_NDEF, ASN1_OCTET_STRING, ASN1_TFLG_NDEF

139 ASN1_ITEM_TEMPLATE(ASN1_SEQUENCE_ANY) =
140         ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, ASN1_SEQUENCE_ANY, ASN1_
141 ASN1_ITEM_TEMPLATE_END(ASN1_SEQUENCE_ANY)

143 ASN1_ITEM_TEMPLATE(ASN1_SET_ANY) =
144         ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SET_OF, 0, ASN1_SET_ANY, ASN1_ANY)
145 ASN1_ITEM_TEMPLATE_END(ASN1_SET_ANY)

147 IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(ASN1_SEQUENCE_ANY, ASN1_SEQUENCE_ANY
148 IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(ASN1_SEQUENCE_ANY, ASN1_SET_ANY, ASN
149 #endif /* ! codereview */
```

```
**********************************************************
    7791 Wed Aug 13 19:52:06 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/tasn_utl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* tasn_utl.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stddef.h>
#include <string.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/objects.h>
#include <openssl/err.h>

/* Utility functions for manipulating fields and offsets */

/* Add 'offset' to 'addr' */
#define offset2ptr(addr, offset) (void *)(((char *) addr) + offset)

/* Given an ASN1_ITEM CHOICE type return
 * the selector value
 */

int asn1_get_choice_selector(ASN1_VALUE **pval, const ASN1_ITEM *it)
        {
        int *sel = offset2ptr(*pval, it->utype);
        return *sel;
        }

/* Given an ASN1_ITEM CHOICE type set
 * the selector value, return old value.
 */

int asn1_set_choice_selector(ASN1_VALUE **pval, int value, const ASN1_ITEM *it)
        {
        int *sel, ret;
        sel = offset2ptr(*pval, it->utype);
        ret = *sel;
        *sel = value;
        return ret;
        }

/* Do reference counting. The value 'op' decides what to do.
 * if it is +1 then the count is incremented. If op is 0 count is
 * set to 1. If op is -1 count is decremented and the return value
 * is the current refrence count or 0 if no reference count exists.
 */

int asn1_do_lock(ASN1_VALUE **pval, int op, const ASN1_ITEM *it)
        {
        const ASN1_AUX *aux;
        int *lck, ret;
        if ((it->itype != ASN1_ITYPE_SEQUENCE)
           && (it->itype != ASN1_ITYPE_NDEF_SEQUENCE))
                return 0;
        aux = it->funcs;
        if (!aux || !(aux->flags & ASN1_AFLG_REFCOUNT))
                return 0;
        lck = offset2ptr(*pval, aux->ref_offset);
        if (op == 0)
                {
                *lck = 1;
                return 1;
                }
        ret = CRYPTO_add(lck, op, aux->ref_lock);
#ifdef REF_PRINT
        fprintf(stderr, "%s: Reference Count: %d\n", it->sname, *lck);
#endif
#ifdef REF_CHECK
        if (ret < 0)
                fprintf(stderr, "%s, bad reference count\n", it->sname);
#endif
        return ret;
        }
```

```
128 static ASN1_ENCODING *asn1_get_enc_ptr(ASN1_VALUE **pval, const ASN1_ITEM *it)
129         {
130         const ASN1_AUX *aux;
131         if (!pval || !*pval)
132                 return NULL;
133         aux = it->funcs;
134         if (!aux || !(aux->flags & ASN1_AFLG_ENCODING))
135                 return NULL;
136         return offset2ptr(*pval, aux->enc_offset);
137         }

139 void asn1_enc_init(ASN1_VALUE **pval, const ASN1_ITEM *it)
140         {
141         ASN1_ENCODING *enc;
142         enc = asn1_get_enc_ptr(pval, it);
143         if (enc)
144                 {
145                 enc->enc = NULL;
146                 enc->len = 0;
147                 enc->modified = 1;
148                 }
149         }

151 void asn1_enc_free(ASN1_VALUE **pval, const ASN1_ITEM *it)
152         {
153         ASN1_ENCODING *enc;
154         enc = asn1_get_enc_ptr(pval, it);
155         if (enc)
156                 {
157                 if (enc->enc)
158                         OPENSSL_free(enc->enc);
159                 enc->enc = NULL;
160                 enc->len = 0;
161                 enc->modified = 1;
162                 }
163         }

165 int asn1_enc_save(ASN1_VALUE **pval, const unsigned char *in, int inlen,
166                                                         const ASN1_ITEM *it)
167         {
168         ASN1_ENCODING *enc;
169         enc = asn1_get_enc_ptr(pval, it);
170         if (!enc)
171                 return 1;

173         if (enc->enc)
174                 OPENSSL_free(enc->enc);
175         enc->enc = OPENSSL_malloc(inlen);
176         if (!enc->enc)
177                 return 0;
178         memcpy(enc->enc, in, inlen);
179         enc->len = inlen;
180         enc->modified = 0;

182         return 1;
183         }

185 int asn1_enc_restore(int *len, unsigned char **out, ASN1_VALUE **pval,
186                                                         const ASN1_ITEM *it)
187         {
188         ASN1_ENCODING *enc;
189         enc = asn1_get_enc_ptr(pval, it);
190         if (!enc || enc->modified)
191                 return 0;
192         if (out)
193                 {
```



```
194                 memcpy(*out, enc->enc, enc->len);
195                 *out += enc->len;
196                 }
197         if (len)
198                 *len = enc->len;
199         return 1;
200         }

202 /* Given an ASN1_TEMPLATE get a pointer to a field */
203 ASN1_VALUE ** asn1_get_field_ptr(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt)
204         {
205         ASN1_VALUE **pvaltmp;
206         if (tt->flags & ASN1_TFLG_COMBINE)
207                 return pval;
208         pvaltmp = offset2ptr(*pval, tt->offset);
209         /* NOTE for BOOLEAN types the field is just a plain
210          * int so we can't return int **, so settle for
211          * (int *).
212          */
213         return pvaltmp;
214         }

216 /* Handle ANY DEFINED BY template, find the selector, look up
217  * the relevant ASN1_TEMPLATE in the table and return it.
218  */

220 const ASN1_TEMPLATE *asn1_do_adb(ASN1_VALUE **pval, const ASN1_TEMPLATE *tt,
221                                                                 int nullerr)
222         {
223         const ASN1_ADB *adb;
224         const ASN1_ADB_TABLE *atbl;
225         long selector;
226         ASN1_VALUE **sfld;
227         int i;
228         if (!(tt->flags & ASN1_TFLG_ADB_MASK))
229                 return tt;

231         /* Else ANY DEFINED BY ... get the table */
232         adb = ASN1_ADB_ptr(tt->item);

234         /* Get the selector field */
235         sfld = offset2ptr(*pval, adb->offset);

237         /* Check if NULL */
238         if (!sfld)
239                 {
240                 if (!adb->null_tt)
241                         goto err;
242                 return adb->null_tt;
243                 }

245         /* Convert type to a long:
246          * NB: don't check for NID_undef here because it
247          * might be a legitimate value in the table
248          */
249         if (tt->flags & ASN1_TFLG_ADB_OID)
250                 selector = OBJ_obj2nid((ASN1_OBJECT *)*sfld);
251         else
252                 selector = ASN1_INTEGER_get((ASN1_INTEGER *)*sfld);

254         /* Try to find matching entry in table
255          * Maybe should check application types first to
256          * allow application override? Might also be useful
257          * to have a flag which indicates table is sorted and
258          * we can do a binary search. For now stick to a
259          * linear search.
```

```
260          */
262         for (atbl = adb->tbl, i = 0; i < adb->tblcount; i++, atbl++)
263                 if (atbl->value == selector)
264                         return &atbl->tt;
266         /* FIXME: need to search application table too */
268         /* No match, return default type */
269         if (!adb->default_tt)
270                 goto err;
271         return adb->default_tt;
273         err:
274         /* FIXME: should log the value or OID of unsupported type */
275         if (nullerr)
276                 ASN1err(ASN1_F_ASN1_DO_ADB,
277                         ASN1_R_UNSUPPORTED_ANY_DEFINED_BY_TYPE);
278         return NULL;
279         }
280 #endif /* ! codereview */
```

```
**********************************************************
    4606 Wed Aug 13 19:52:06 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_algor.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x_algor.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stddef.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>
```



```
#include <openssl/asn1t.h>

ASN1_SEQUENCE(X509_ALGOR) = {
        ASN1_SIMPLE(X509_ALGOR, algorithm, ASN1_OBJECT),
        ASN1_OPT(X509_ALGOR, parameter, ASN1_ANY)
} ASN1_SEQUENCE_END(X509_ALGOR)

ASN1_ITEM_TEMPLATE(X509_ALGORS) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, algorithms, X509_ALGOR)
ASN1_ITEM_TEMPLATE_END(X509_ALGORS)

IMPLEMENT_ASN1_FUNCTIONS(X509_ALGOR)
IMPLEMENT_ASN1_ENCODE_FUNCTIONS_fname(X509_ALGORS, X509_ALGORS, X509_ALGORS)
IMPLEMENT_ASN1_DUP_FUNCTION(X509_ALGOR)

IMPLEMENT_STACK_OF(X509_ALGOR)
IMPLEMENT_ASN1_SET_OF(X509_ALGOR)

int X509_ALGOR_set0(X509_ALGOR *alg, ASN1_OBJECT *aobj, int ptype, void *pval)
        {
        if (!alg)
                return 0;
        if (ptype != V_ASN1_UNDEF)
                {
                if (alg->parameter == NULL)
                        alg->parameter = ASN1_TYPE_new();
                if (alg->parameter == NULL)
                        return 0;
                }
        if (alg)
                {
                if (alg->algorithm)
                        ASN1_OBJECT_free(alg->algorithm);
                alg->algorithm = aobj;
                }
        if (ptype == 0)
                return 1;
        if (ptype == V_ASN1_UNDEF)
                {
                if (alg->parameter)
                        {
                        ASN1_TYPE_free(alg->parameter);
                        alg->parameter = NULL;
                        }
                }
        else
                ASN1_TYPE_set(alg->parameter, ptype, pval);
        return 1;
        }

void X509_ALGOR_get0(ASN1_OBJECT **paobj, int *pptype, void **ppval,
                                                X509_ALGOR *algor)
        {
        if (paobj)
                *paobj = algor->algorithm;
        if (pptype)
                {
                if (algor->parameter == NULL)
                        {
                        *pptype = V_ASN1_UNDEF;
                        return;
                        }
                else
                        *pptype = algor->parameter->type;
                if (ppval)
                        *ppval = algor->parameter->value.ptr;
```

```
128                 }
129         }

131 /* Set up an X509_ALGOR DigestAlgorithmIdentifier from an EVP_MD */

133 void X509_ALGOR_set_md(X509_ALGOR *alg, const EVP_MD *md)
134         {
135         int param_type;

137         if (md->flags & EVP_MD_FLAG_DIGALGID_ABSENT)
138                 param_type = V_ASN1_UNDEF;
139         else
140                 param_type = V_ASN1_NULL;

142         X509_ALGOR_set0(alg, OBJ_nid2obj(EVP_MD_type(md)), param_type, NULL);

144         }
145 #endif /* ! codereview */
```

```
**********************************************************
    4891 Wed Aug 13 19:52:07 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_attrib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_attrib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/asn1t.h>
#include <openssl/x509.h>

/* X509_ATTRIBUTE: this has the following form:
 *
 * typedef struct x509_attributes_st
 *      {
 *      ASN1_OBJECT *object;
 *      int single;
 *      union   {
 *              char            *ptr;
 *              STACK_OF(ASN1_TYPE) *set;
 *              ASN1_TYPE       *single;
 *              } value;
 *      } X509_ATTRIBUTE;
 *
 * this needs some extra thought because the CHOICE type is
 * merged with the main structure and because the value can
 * be anything at all we *must* try the SET OF first because
 * the ASN1_ANY type will swallow anything including the whole
 * SET OF structure.
 */

ASN1_CHOICE(X509_ATTRIBUTE_SET) = {
        ASN1_SET_OF(X509_ATTRIBUTE, value.set, ASN1_ANY),
        ASN1_SIMPLE(X509_ATTRIBUTE, value.single, ASN1_ANY)
} ASN1_CHOICE_END_selector(X509_ATTRIBUTE, X509_ATTRIBUTE_SET, single)

ASN1_SEQUENCE(X509_ATTRIBUTE) = {
        ASN1_SIMPLE(X509_ATTRIBUTE, object, ASN1_OBJECT),
        /* CHOICE type merged with parent */
        ASN1_EX_COMBINE(0, 0, X509_ATTRIBUTE_SET)
} ASN1_SEQUENCE_END(X509_ATTRIBUTE)

IMPLEMENT_ASN1_FUNCTIONS(X509_ATTRIBUTE)
IMPLEMENT_ASN1_DUP_FUNCTION(X509_ATTRIBUTE)

X509_ATTRIBUTE *X509_ATTRIBUTE_create(int nid, int atrtype, void *value)
        {
        X509_ATTRIBUTE *ret=NULL;
        ASN1_TYPE *val=NULL;

        if ((ret=X509_ATTRIBUTE_new()) == NULL)
                return(NULL);
        ret->object=OBJ_nid2obj(nid);
        ret->single=0;
        if ((ret->value.set=sk_ASN1_TYPE_new_null()) == NULL) goto err;
        if ((val=ASN1_TYPE_new()) == NULL) goto err;
        if (!sk_ASN1_TYPE_push(ret->value.set,val)) goto err;

        ASN1_TYPE_set(val,atrtype,value);
        return(ret);
err:
        if (ret != NULL) X509_ATTRIBUTE_free(ret);
        if (val != NULL) ASN1_TYPE_free(val);
        return(NULL);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4777 Wed Aug 13 19:52:07 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_bignum.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x_bignum.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/bn.h>

/* Custom primitive type for BIGNUM handling. This reads in an ASN1_INTEGER as a
 * BIGNUM directly. Currently it ignores the sign which isn't a problem since al
 * BIGNUMs used are non negative and anything that looks negative is normally du
 * to an encoding error.
 */

#define BN_SENSITIVE    1

static int bn_new(ASN1_VALUE **pval, const ASN1_ITEM *it);
static void bn_free(ASN1_VALUE **pval, const ASN1_ITEM *it);

static int bn_i2c(ASN1_VALUE **pval, unsigned char *cont, int *putype, const ASN
static int bn_c2i(ASN1_VALUE **pval, const unsigned char *cont, int len, int uty

static ASN1_PRIMITIVE_FUNCS bignum_pf = {
        NULL, 0,
        bn_new,
        bn_free,
        0,
        bn_c2i,
        bn_i2c
};

ASN1_ITEM_start(BIGNUM)
        ASN1_ITYPE_PRIMITIVE, V_ASN1_INTEGER, NULL, 0, &bignum_pf, 0, "BIGNUM"
ASN1_ITEM_end(BIGNUM)

ASN1_ITEM_start(CBIGNUM)
        ASN1_ITYPE_PRIMITIVE, V_ASN1_INTEGER, NULL, 0, &bignum_pf, BN_SENSITIVE,
ASN1_ITEM_end(CBIGNUM)

static int bn_new(ASN1_VALUE **pval, const ASN1_ITEM *it)
{
        *pval = (ASN1_VALUE *)BN_new();
        if(*pval) return 1;
        else return 0;
}

static void bn_free(ASN1_VALUE **pval, const ASN1_ITEM *it)
{
        if(!*pval) return;
        if(it->size & BN_SENSITIVE) BN_clear_free((BIGNUM *)*pval);
        else BN_free((BIGNUM *)*pval);
        *pval = NULL;
}

static int bn_i2c(ASN1_VALUE **pval, unsigned char *cont, int *putype, const ASN
{
        BIGNUM *bn;
        int pad;
        if(!*pval) return -1;
        bn = (BIGNUM *)*pval;
        /* If MSB set in an octet we need a padding byte */
        if(BN_num_bits(bn) & 0x7) pad = 0;
        else pad = 1;
        if(cont) {
                if(pad) *cont++ = 0;
                BN_bn2bin(bn, cont);
        }
        return pad + BN_num_bytes(bn);
}

static int bn_c2i(ASN1_VALUE **pval, const unsigned char *cont, int len,
                  int utype, char *free_cont, const ASN1_ITEM *it)
```

```
128 {
129         BIGNUM *bn;
130         if(!*pval) bn_new(pval, it);
131         bn  = (BIGNUM *)*pval;
132         if(!BN_bin2bn(cont, len, bn)) {
133                 bn_free(pval, it);
134                 return 0;
135         }
136         return 1;
137 }
138 #endif /* ! codereview */
```

```
**********************************************************
   14470 Wed Aug 13 19:52:07 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_crl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_crl.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "asn1_locl.h"
```



```
#include <openssl/asn1t.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

static int X509_REVOKED_cmp(const X509_REVOKED * const *a,
                            const X509_REVOKED * const *b);
static void setup_idp(X509_CRL *crl, ISSUING_DIST_POINT *idp);

ASN1_SEQUENCE(X509_REVOKED) = {
        ASN1_SIMPLE(X509_REVOKED,serialNumber, ASN1_INTEGER),
        ASN1_SIMPLE(X509_REVOKED,revocationDate, ASN1_TIME),
        ASN1_SEQUENCE_OF_OPT(X509_REVOKED,extensions, X509_EXTENSION)
} ASN1_SEQUENCE_END(X509_REVOKED)

static int def_crl_verify(X509_CRL *crl, EVP_PKEY *r);
static int def_crl_lookup(X509_CRL *crl,
                X509_REVOKED **ret, ASN1_INTEGER *serial, X509_NAME *issuer);

static X509_CRL_METHOD int_crl_meth =
        {
        0,
        0,0,
        def_crl_lookup,
        def_crl_verify
        };

static const X509_CRL_METHOD *default_crl_method = &int_crl_meth;

/* The X509_CRL_INFO structure needs a bit of customisation.
 * Since we cache the original encoding the signature wont be affected by
 * reordering of the revoked field.
 */
static int crl_inf_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                                void *exarg)
{
        X509_CRL_INFO *a = (X509_CRL_INFO *)*pval;

        if(!a || !a->revoked) return 1;
        switch(operation) {
                /* Just set cmp function here. We don't sort because that
                 * would affect the output of X509_CRL_print().
                 */
                case ASN1_OP_D2I_POST:
                (void)sk_X509_REVOKED_set_cmp_func(a->revoked,X509_REVOKED_cmp);
                break;
        }
        return 1;
}


ASN1_SEQUENCE_enc(X509_CRL_INFO, enc, crl_inf_cb) = {
        ASN1_OPT(X509_CRL_INFO, version, ASN1_INTEGER),
        ASN1_SIMPLE(X509_CRL_INFO, sig_alg, X509_ALGOR),
        ASN1_SIMPLE(X509_CRL_INFO, issuer, X509_NAME),
        ASN1_SIMPLE(X509_CRL_INFO, lastUpdate, ASN1_TIME),
        ASN1_OPT(X509_CRL_INFO, nextUpdate, ASN1_TIME),
        ASN1_SEQUENCE_OF_OPT(X509_CRL_INFO, revoked, X509_REVOKED),
        ASN1_EXP_SEQUENCE_OF_OPT(X509_CRL_INFO, extensions, X509_EXTENSION, 0)
} ASN1_SEQUENCE_END_enc(X509_CRL_INFO, X509_CRL_INFO)

/* Set CRL entry issuer according to CRL certificate issuer extension.
 * Check for unhandled critical CRL entry extensions.
 */

static int crl_set_issuers(X509_CRL *crl)
        {
```

```
129         int i, j;
130         GENERAL_NAMES *gens, *gtmp;
131         STACK_OF(X509_REVOKED) *revoked;

133         revoked = X509_CRL_get_REVOKED(crl);

135         gens = NULL;
136         for (i = 0; i < sk_X509_REVOKED_num(revoked); i++)
137                 {
138                 X509_REVOKED *rev = sk_X509_REVOKED_value(revoked, i);
139                 STACK_OF(X509_EXTENSION) *exts;
140                 ASN1_ENUMERATED *reason;
141                 X509_EXTENSION *ext;
142                 gtmp = X509_REVOKED_get_ext_d2i(rev,
143                                                 NID_certificate_issuer,
144                                                 &j, NULL);
145                 if (!gtmp && (j != -1))
146                         {
147                         crl->flags |= EXFLAG_INVALID;
148                         return 1;
149                         }

151                 if (gtmp)
152                         {
153                         gens = gtmp;
154                         if (!crl->issuers)
155                                 {
156                                 crl->issuers = sk_GENERAL_NAMES_new_null();
157                                 if (!crl->issuers)
158                                         return 0;
159                                 }
160                         if (!sk_GENERAL_NAMES_push(crl->issuers, gtmp))
161                                 return 0;
162                         }
163                 rev->issuer = gens;

165                 reason = X509_REVOKED_get_ext_d2i(rev, NID_crl_reason,
166                                                                 &j, NULL);
167                 if (!reason && (j != -1))
168                         {
169                         crl->flags |= EXFLAG_INVALID;
170                         return 1;
171                         }

173                 if (reason)
174                         {
175                         rev->reason = ASN1_ENUMERATED_get(reason);
176                         ASN1_ENUMERATED_free(reason);
177                         }
178                 else
179                         rev->reason = CRL_REASON_NONE;

181                 /* Check for critical CRL entry extensions */

183                 exts = rev->extensions;

185                 for (j = 0; j < sk_X509_EXTENSION_num(exts); j++)
186                         {
187                         ext = sk_X509_EXTENSION_value(exts, j);
188                         if (ext->critical > 0)
189                                 {
190                                 if (OBJ_obj2nid(ext->object) ==
191                                         NID_certificate_issuer)
192                                         continue;
193                                 crl->flags |= EXFLAG_CRITICAL;
```



```
194                                 break;
195                                 }
196                         }


199                 }

201         return 1;

203         }

205 /* The X509_CRL structure needs a bit of customisation. Cache some extensions
206  * and hash of the whole CRL.
207  */
208 static int crl_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
209                                                                 void *exarg)
210         {
211         X509_CRL *crl = (X509_CRL *)*pval;
212         STACK_OF(X509_EXTENSION) *exts;
213         X509_EXTENSION *ext;
214         int idx;

216         switch(operation)
217                 {
218                 case ASN1_OP_NEW_POST:
219                 crl->idp = NULL;
220                 crl->akid = NULL;
221                 crl->flags = 0;
222                 crl->idp_flags = 0;
223                 crl->idp_reasons = CRLDP_ALL_REASONS;
224                 crl->meth = default_crl_method;
225                 crl->meth_data = NULL;
226                 crl->issuers = NULL;
227                 crl->crl_number = NULL;
228                 crl->base_crl_number = NULL;
229                 break;

231                 case ASN1_OP_D2I_POST:
232 #ifndef OPENSSL_NO_SHA
233                 X509_CRL_digest(crl, EVP_sha1(), crl->sha1_hash, NULL);
234 #endif
235                 crl->idp = X509_CRL_get_ext_d2i(crl,
236                                 NID_issuing_distribution_point, NULL, NULL);
237                 if (crl->idp)
238                         setup_idp(crl, crl->idp);

240                 crl->akid = X509_CRL_get_ext_d2i(crl,
241                                 NID_authority_key_identifier, NULL, NULL);

243                 crl->crl_number = X509_CRL_get_ext_d2i(crl,
244                                 NID_crl_number, NULL, NULL);

246                 crl->base_crl_number = X509_CRL_get_ext_d2i(crl,
247                                 NID_delta_crl, NULL, NULL);
248                 /* Delta CRLs must have CRL number */
249                 if (crl->base_crl_number && !crl->crl_number)
250                         crl->flags |= EXFLAG_INVALID;

252                 /* See if we have any unhandled critical CRL extensions and
253                  * indicate this in a flag. We only currently handle IDP so
254                  * anything else critical sets the flag.
255                  *
256                  * This code accesses the X509_CRL structure directly:
257                  * applications shouldn't do this.
258                  */
```

```
260                 exts = crl->crl->extensions;

262                 for (idx = 0; idx < sk_X509_EXTENSION_num(exts); idx++)
263                         {
264                         int nid;
265                         ext = sk_X509_EXTENSION_value(exts, idx);
266                         nid = OBJ_obj2nid(ext->object);
267                         if (nid == NID_freshest_crl)
268                                 crl->flags |= EXFLAG_FRESHEST;
269                         if (ext->critical > 0)
270                                 {
271                                 /* We handle IDP and deltas */
272                                 if ((nid == NID_issuing_distribution_point)
273                                         || (nid == NID_authority_key_identifier)
274                                         || (nid == NID_delta_crl))
275                                         break;;
276                                 crl->flags |= EXFLAG_CRITICAL;
277                                 break;
278                                 }
279                         }


282                 if (!crl_set_issuers(crl))
283                         return 0;

285                 if (crl->meth->crl_init)
286                         {
287                         if (crl->meth->crl_init(crl) == 0)
288                                 return 0;
289                         }
290                 break;

292                 case ASN1_OP_FREE_POST:
293                 if (crl->meth->crl_free)
294                         {
295                         if (!crl->meth->crl_free(crl))
296                                 return 0;
297                         }
298                 if (crl->akid)
299                         AUTHORITY_KEYID_free(crl->akid);
300                 if (crl->idp)
301                         ISSUING_DIST_POINT_free(crl->idp);
302                 ASN1_INTEGER_free(crl->crl_number);
303                 ASN1_INTEGER_free(crl->base_crl_number);
304                 sk_GENERAL_NAMES_pop_free(crl->issuers, GENERAL_NAMES_free);
305                 break;
306                 }
307         return 1;
308         }

310 /* Convert IDP into a more convenient form */

312 static void setup_idp(X509_CRL *crl, ISSUING_DIST_POINT *idp)
313         {
314         int idp_only = 0;
315         /* Set various flags according to IDP */
316         crl->idp_flags |= IDP_PRESENT;
317         if (idp->onlyuser > 0)
318                 {
319                 idp_only++;
320                 crl->idp_flags |= IDP_ONLYUSER;
321                 }
322         if (idp->onlyCA > 0)
323                 {
324                 idp_only++;
325                 crl->idp_flags |= IDP_ONLYCA;
```



```
326                 }
327         if (idp->onlyattr > 0)
328                 {
329                 idp_only++;
330                 crl->idp_flags |= IDP_ONLYATTR;
331                 }

333         if (idp_only > 1)
334                 crl->idp_flags |= IDP_INVALID;

336         if (idp->indirectCRL > 0)
337                 crl->idp_flags |= IDP_INDIRECT;

339         if (idp->onlysomereasons)
340                 {
341                 crl->idp_flags |= IDP_REASONS;
342                 if (idp->onlysomereasons->length > 0)
343                         crl->idp_reasons = idp->onlysomereasons->data[0];
344                 if (idp->onlysomereasons->length > 1)
345                         crl->idp_reasons |=
346                                 (idp->onlysomereasons->data[1] << 8);
347                 crl->idp_reasons &= CRLDP_ALL_REASONS;
348                 }

350         DIST_POINT_set_dpname(idp->distpoint, X509_CRL_get_issuer(crl));
351         }

353 ASN1_SEQUENCE_ref(X509_CRL, crl_cb, CRYPTO_LOCK_X509_CRL) = {
354         ASN1_SIMPLE(X509_CRL, crl, X509_CRL_INFO),
355         ASN1_SIMPLE(X509_CRL, sig_alg, X509_ALGOR),
356         ASN1_SIMPLE(X509_CRL, signature, ASN1_BIT_STRING)
357 } ASN1_SEQUENCE_END_ref(X509_CRL, X509_CRL)

359 IMPLEMENT_ASN1_FUNCTIONS(X509_REVOKED)
360 IMPLEMENT_ASN1_FUNCTIONS(X509_CRL_INFO)
361 IMPLEMENT_ASN1_FUNCTIONS(X509_CRL)
362 IMPLEMENT_ASN1_DUP_FUNCTION(X509_CRL)

364 static int X509_REVOKED_cmp(const X509_REVOKED * const *a,
365                              const X509_REVOKED * const *b)
366         {
367         return(ASN1_STRING_cmp(
368                 (ASN1_STRING *)(*a)->serialNumber,
369                 (ASN1_STRING *)(*b)->serialNumber));
370         }

372 int X509_CRL_add0_revoked(X509_CRL *crl, X509_REVOKED *rev)
373 {
374         X509_CRL_INFO *inf;
375         inf = crl->crl;
376         if(!inf->revoked)
377                 inf->revoked = sk_X509_REVOKED_new(X509_REVOKED_cmp);
378         if(!inf->revoked || !sk_X509_REVOKED_push(inf->revoked, rev)) {
379                 ASN1err(ASN1_F_X509_CRL_ADD0_REVOKED, ERR_R_MALLOC_FAILURE);
380                 return 0;
381         }
382         inf->enc.modified = 1;
383         return 1;
384 }

386 int X509_CRL_verify(X509_CRL *crl, EVP_PKEY *r)
387         {
388         if (crl->meth->crl_verify)
389                 return crl->meth->crl_verify(crl, r);
390         return 0;
391         }
```

```
int X509_CRL_get0_by_serial(X509_CRL *crl,
                X509_REVOKED **ret, ASN1_INTEGER *serial)
        {
        if (crl->meth->crl_lookup)
                return crl->meth->crl_lookup(crl, ret, serial, NULL);
        return 0;
        }

int X509_CRL_get0_by_cert(X509_CRL *crl, X509_REVOKED **ret, X509 *x)
        {
        if (crl->meth->crl_lookup)
                return crl->meth->crl_lookup(crl, ret,
                                             X509_get_serialNumber(x),
                                             X509_get_issuer_name(x));
        return 0;
        }

static int def_crl_verify(X509_CRL *crl, EVP_PKEY *r)
        {
        return(ASN1_item_verify(ASN1_ITEM_rptr(X509_CRL_INFO),
                crl->sig_alg, crl->signature,crl->crl,r));
        }

static int crl_revoked_issuer_match(X509_CRL *crl, X509_NAME *nm,
                                             X509_REVOKED *rev)
        {
        int i;

        if (!rev->issuer)
                {
                if (!nm)
                        return 1;
                if (!X509_NAME_cmp(nm, X509_CRL_get_issuer(crl)))
                        return 1;
                return 0;
                }

        if (!nm)
                nm = X509_CRL_get_issuer(crl);

        for (i = 0; i < sk_GENERAL_NAME_num(rev->issuer); i++)
                {
                GENERAL_NAME *gen = sk_GENERAL_NAME_value(rev->issuer, i);
                if (gen->type != GEN_DIRNAME)
                        continue;
                if (!X509_NAME_cmp(nm, gen->d.directoryName))
                        return 1;
                }
        return 0;

        }

static int def_crl_lookup(X509_CRL *crl,
                X509_REVOKED **ret, ASN1_INTEGER *serial, X509_NAME *issuer)
        {
        X509_REVOKED rtmp, *rev;
        int idx;
        rtmp.serialNumber = serial;
        /* Sort revoked into serial number order if not already sorted.
         * Do this under a lock to avoid race condition.
         */
        if (!sk_X509_REVOKED_is_sorted(crl->crl->revoked))
                {
                CRYPTO_w_lock(CRYPTO_LOCK_X509_CRL);
                sk_X509_REVOKED_sort(crl->crl->revoked);
```



```
                CRYPTO_w_unlock(CRYPTO_LOCK_X509_CRL);
                }
        idx = sk_X509_REVOKED_find(crl->crl->revoked, &rtmp);
        if(idx < 0)
                return 0;
        /* Need to look for matching name */
        for(;idx < sk_X509_REVOKED_num(crl->crl->revoked); idx++)
                {
                rev = sk_X509_REVOKED_value(crl->crl->revoked, idx);
                if (ASN1_INTEGER_cmp(rev->serialNumber, serial))
                        return 0;
                if (crl_revoked_issuer_match(crl, issuer, rev))
                        {
                        if (ret)
                                *ret = rev;
                        if (rev->reason == CRL_REASON_REMOVE_FROM_CRL)
                                return 2;
                        return 1;
                        }
                }
        return 0;
        }

void X509_CRL_set_default_method(const X509_CRL_METHOD *meth)
        {
        if (meth == NULL)
                default_crl_method = &int_crl_meth;
        else
                default_crl_method = meth;
        }

X509_CRL_METHOD *X509_CRL_METHOD_new(
        int (*crl_init)(X509_CRL *crl),
        int (*crl_free)(X509_CRL *crl),
        int (*crl_lookup)(X509_CRL *crl, X509_REVOKED **ret,
                          ASN1_INTEGER *ser, X509_NAME *issuer),
        int (*crl_verify)(X509_CRL *crl, EVP_PKEY *pk))
        {
        X509_CRL_METHOD *m;
        m = OPENSSL_malloc(sizeof(X509_CRL_METHOD));
        if (!m)
                return NULL;
        m->crl_init = crl_init;
        m->crl_free = crl_free;
        m->crl_lookup = crl_lookup;
        m->crl_verify = crl_verify;
        m->flags = X509_CRL_METHOD_DYNAMIC;
        return m;
        }

void X509_CRL_METHOD_free(X509_CRL_METHOD *m)
        {
        if (!(m->flags & X509_CRL_METHOD_DYNAMIC))
                return;
        OPENSSL_free(m);
        }

void X509_CRL_set_meth_data(X509_CRL *crl, void *dat)
        {
        crl->meth_data = dat;
        }

void *X509_CRL_get_meth_data(X509_CRL *crl)
        {
        return crl->meth_data;
        }
```

```
525 IMPLEMENT_STACK_OF(X509_REVOKED)
526 IMPLEMENT_ASN1_SET_OF(X509_REVOKED)
527 IMPLEMENT_STACK_OF(X509_CRL)
528 IMPLEMENT_ASN1_SET_OF(X509_CRL)
529 #endif /* ! codereview */
```

```
**********************************************************
    3390 Wed Aug 13 19:52:07 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_exten.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x_exten.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stddef.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>
```



```
#include <openssl/asn1t.h>

ASN1_SEQUENCE(X509_EXTENSION) = {
        ASN1_SIMPLE(X509_EXTENSION, object, ASN1_OBJECT),
        ASN1_OPT(X509_EXTENSION, critical, ASN1_BOOLEAN),
        ASN1_SIMPLE(X509_EXTENSION, value, ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END(X509_EXTENSION)

ASN1_ITEM_TEMPLATE(X509_EXTENSIONS) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, Extension, X509_EXTENSIO
ASN1_ITEM_TEMPLATE_END(X509_EXTENSIONS)

IMPLEMENT_ASN1_FUNCTIONS(X509_EXTENSION)
IMPLEMENT_ASN1_ENCODE_FUNCTIONS_fname(X509_EXTENSIONS, X509_EXTENSIONS, X509_EXT
IMPLEMENT_ASN1_DUP_FUNCTION(X509_EXTENSION)
#endif /* ! codereview */
```

```
**********************************************************
    4252 Wed Aug 13 19:52:07 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_info.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_info.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/asn1.h>
#include <openssl/x509.h>

X509_INFO *X509_INFO_new(void)
        {
        X509_INFO *ret=NULL;

        ret=(X509_INFO *)OPENSSL_malloc(sizeof(X509_INFO));
        if (ret == NULL)
                {
                ASN1err(ASN1_F_X509_INFO_NEW,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }

        ret->enc_cipher.cipher=NULL;
        ret->enc_len=0;
        ret->enc_data=NULL;

        ret->references=1;
        ret->x509=NULL;
        ret->crl=NULL;
        ret->x_pkey=NULL;
        return(ret);
        }

void X509_INFO_free(X509_INFO *x)
        {
        int i;

        if (x == NULL) return;

        i=CRYPTO_add(&x->references,-1,CRYPTO_LOCK_X509_INFO);
#ifdef REF_PRINT
        REF_PRINT("X509_INFO",x);
#endif
        if (i > 0) return;
#ifdef REF_CHECK
        if (i < 0)
                {
                fprintf(stderr,"X509_INFO_free, bad reference count\n");
                abort();
                }
#endif

        if (x->x509 != NULL) X509_free(x->x509);
        if (x->crl != NULL) X509_CRL_free(x->crl);
        if (x->x_pkey != NULL) X509_PKEY_free(x->x_pkey);
        if (x->enc_data != NULL) OPENSSL_free(x->enc_data);
        OPENSSL_free(x);
        }

IMPLEMENT_STACK_OF(X509_INFO)
#endif /* ! codereview */
```

```
**********************************************************
    5843 Wed Aug 13 19:52:07 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_long.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x_long.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/bn.h>

/* Custom primitive type for long handling. This converts between an ASN1_INTEGE
 * and a long directly.
 */


static int long_new(ASN1_VALUE **pval, const ASN1_ITEM *it);
static void long_free(ASN1_VALUE **pval, const ASN1_ITEM *it);

static int long_i2c(ASN1_VALUE **pval, unsigned char *cont, int *putype, const A
static int long_c2i(ASN1_VALUE **pval, const unsigned char *cont, int len, int u
static int long_print(BIO *out, ASN1_VALUE **pval, const ASN1_ITEM *it, int inde

static ASN1_PRIMITIVE_FUNCS long_pf = {
        NULL, 0,
        long_new,
        long_free,
        long_free,      /* Clear should set to initial value */
        long_c2i,
        long_i2c,
        long_print
};

ASN1_ITEM_start(LONG)
        ASN1_ITYPE_PRIMITIVE, V_ASN1_INTEGER, NULL, 0, &long_pf, ASN1_LONG_UNDEF
ASN1_ITEM_end(LONG)

ASN1_ITEM_start(ZLONG)
        ASN1_ITYPE_PRIMITIVE, V_ASN1_INTEGER, NULL, 0, &long_pf, 0, "ZLONG"
ASN1_ITEM_end(ZLONG)

static int long_new(ASN1_VALUE **pval, const ASN1_ITEM *it)
{
        *(long *)pval = it->size;
        return 1;
}

static void long_free(ASN1_VALUE **pval, const ASN1_ITEM *it)
{
        *(long *)pval = it->size;
}

static int long_i2c(ASN1_VALUE **pval, unsigned char *cont, int *putype, const A
{
        long ltmp;
        unsigned long utmp;
        int clen, pad, i;
        /* this exists to bypass broken gcc optimization */
        char *cp = (char *)pval;

        /* use memcpy, because we may not be long aligned */
        memcpy(&ltmp, cp, sizeof(long));

        if(ltmp == it->size) return -1;
        /* Convert the long to positive: we subtract one if negative so
         * we can cleanly handle the padding if only the MSB of the leading
         * octet is set.
         */
        if(ltmp < 0) utmp = -ltmp - 1;
        else utmp = ltmp;
        clen = BN_num_bits_word(utmp);
        /* If MSB of leading octet set we need to pad */
        if(!(clen & 0x7)) pad = 1;
        else pad = 0;
```

```
128         /* Convert number of bits to number of octets */
129         clen = (clen + 7) >> 3;

131         if(cont) {
132                 if(pad) *cont++ = (ltmp < 0) ? 0xff : 0;
133                 for(i = clen - 1; i >= 0; i--) {
134                         cont[i] = (unsigned char)(utmp & 0xff);
135                         if(ltmp < 0) cont[i] ^= 0xff;
136                         utmp >>= 8;
137                 }
138         }
139         return clen + pad;
140 }

142 static int long_c2i(ASN1_VALUE **pval, const unsigned char *cont, int len,
143                     int utype, char *free_cont, const ASN1_ITEM *it)
144 {
145         int neg, i;
146         long ltmp;
147         unsigned long utmp = 0;
148         char *cp = (char *)pval;
149         if(len > (int)sizeof(long)) {
150                 ASN1err(ASN1_F_LONG_C2I, ASN1_R_INTEGER_TOO_LARGE_FOR_LONG);
151                 return 0;
152         }
153         /* Is it negative? */
154         if(len && (cont[0] & 0x80)) neg = 1;
155         else neg = 0;
156         utmp = 0;
157         for(i = 0; i < len; i++) {
158                 utmp <<= 8;
159                 if(neg) utmp |= cont[i] ^ 0xff;
160                 else utmp |= cont[i];
161         }
162         ltmp = (long)utmp;
163         if(neg) {
164                 ltmp++;
165                 ltmp = -ltmp;
166         }
167         if(ltmp == it->size) {
168                 ASN1err(ASN1_F_LONG_C2I, ASN1_R_INTEGER_TOO_LARGE_FOR_LONG);
169                 return 0;
170         }
171         memcpy(cp, &ltmp, sizeof(long));
172         return 1;
173 }

175 static int long_print(BIO *out, ASN1_VALUE **pval, const ASN1_ITEM *it,
176                       int indent, const ASN1_PCTX *pctx)
177         {
178         return BIO_printf(out, "%ld\n", *(long *)pval);
179         }
180 #endif /* ! codereview */
```

```
**********************************************************
   14690 Wed Aug 13 19:52:08 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_name.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_name.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1t.h>
#include <openssl/x509.h>
#include "asn1_locl.h"

typedef STACK_OF(X509_NAME_ENTRY) STACK_OF_X509_NAME_ENTRY;
DECLARE_STACK_OF(STACK_OF_X509_NAME_ENTRY)

static int x509_name_ex_d2i(ASN1_VALUE **val,
                                const unsigned char **in, long len,
                                const ASN1_ITEM *it,
                                int tag, int aclass, char opt, ASN1_TLC *ctx);

static int x509_name_ex_i2d(ASN1_VALUE **val, unsigned char **out,
                                const ASN1_ITEM *it, int tag, int aclass);
static int x509_name_ex_new(ASN1_VALUE **val, const ASN1_ITEM *it);
static void x509_name_ex_free(ASN1_VALUE **val, const ASN1_ITEM *it);

static int x509_name_encode(X509_NAME *a);
static int x509_name_canon(X509_NAME *a);
static int asn1_string_canon(ASN1_STRING *out, ASN1_STRING *in);
static int i2d_name_canon(STACK_OF(STACK_OF_X509_NAME_ENTRY) *intname,
                          unsigned char **in);


static int x509_name_ex_print(BIO *out, ASN1_VALUE **pval,
                                                int indent,
                                                const char *fname,
                                                const ASN1_PCTX *pctx);

ASN1_SEQUENCE(X509_NAME_ENTRY) = {
        ASN1_SIMPLE(X509_NAME_ENTRY, object, ASN1_OBJECT),
        ASN1_SIMPLE(X509_NAME_ENTRY, value, ASN1_PRINTABLE)
} ASN1_SEQUENCE_END(X509_NAME_ENTRY)

IMPLEMENT_ASN1_FUNCTIONS(X509_NAME_ENTRY)
IMPLEMENT_ASN1_DUP_FUNCTION(X509_NAME_ENTRY)

/* For the "Name" type we need a SEQUENCE OF { SET OF X509_NAME_ENTRY }
 * so declare two template wrappers for this
 */

ASN1_ITEM_TEMPLATE(X509_NAME_ENTRIES) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SET_OF, 0, RDNS, X509_NAME_ENTRY)
ASN1_ITEM_TEMPLATE_END(X509_NAME_ENTRIES)

ASN1_ITEM_TEMPLATE(X509_NAME_INTERNAL) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, Name, X509_NAME_ENTRIES)
ASN1_ITEM_TEMPLATE_END(X509_NAME_INTERNAL)

/* Normally that's where it would end: we'd have two nested STACK structures
 * representing the ASN1. Unfortunately X509_NAME uses a completely different
 * form and caches encodings so we have to process the internal form and convert
 * to the external form.
 */

const ASN1_EXTERN_FUNCS x509_name_ff = {
        NULL,
        x509_name_ex_new,
        x509_name_ex_free,
        0,      /* Default clear behaviour is OK */
        x509_name_ex_d2i,
        x509_name_ex_i2d,
        x509_name_ex_print
};

IMPLEMENT_EXTERN_ASN1(X509_NAME, V_ASN1_SEQUENCE, x509_name_ff)
```

```
129 IMPLEMENT_ASN1_FUNCTIONS(X509_NAME)
130 IMPLEMENT_ASN1_DUP_FUNCTION(X509_NAME)

132 static int x509_name_ex_new(ASN1_VALUE **val, const ASN1_ITEM *it)
133 {
134         X509_NAME *ret = NULL;
135         ret = OPENSSL_malloc(sizeof(X509_NAME));
136         if(!ret) goto memerr;
137         if ((ret->entries=sk_X509_NAME_ENTRY_new_null()) == NULL)
138                 goto memerr;
139         if((ret->bytes = BUF_MEM_new()) == NULL) goto memerr;
140         ret->canon_enc = NULL;
141         ret->canon_enclen = 0;
142         ret->modified=1;
143         *val = (ASN1_VALUE *)ret;
144         return 1;

146  memerr:
147         ASN1err(ASN1_F_X509_NAME_EX_NEW, ERR_R_MALLOC_FAILURE);
148         if (ret)
149                 {
150                 if (ret->entries)
151                         sk_X509_NAME_ENTRY_free(ret->entries);
152                 OPENSSL_free(ret);
153                 }
154         return 0;
155 }

157 static void x509_name_ex_free(ASN1_VALUE **pval, const ASN1_ITEM *it)
158 {
159         X509_NAME *a;
160         if(!pval || !*pval)
161             return;
162         a = (X509_NAME *)*pval;

164         BUF_MEM_free(a->bytes);
165         sk_X509_NAME_ENTRY_pop_free(a->entries,X509_NAME_ENTRY_free);
166         if (a->canon_enc)
167                 OPENSSL_free(a->canon_enc);
168         OPENSSL_free(a);
169         *pval = NULL;
170 }

172 static int x509_name_ex_d2i(ASN1_VALUE **val,
173                         const unsigned char **in, long len, const ASN1_ITEM *it,
174                                 int tag, int aclass, char opt, ASN1_TLC *ctx)
175 {
176         const unsigned char *p = *in, *q;
177         union { STACK_OF(STACK_OF_X509_NAME_ENTRY) *s;
178                 ASN1_VALUE *a; } intname = {NULL};
179         union { X509_NAME *x; ASN1_VALUE *a; } nm = {NULL};
180         int i, j, ret;
181         STACK_OF(X509_NAME_ENTRY) *entries;
182         X509_NAME_ENTRY *entry;
183         q = p;

185         /* Get internal representation of Name */
186         ret = ASN1_item_ex_d2i(&intname.a,
187                                &p, len, ASN1_ITEM_rptr(X509_NAME_INTERNAL),
188                                tag, aclass, opt, ctx);

190         if(ret <= 0) return ret;

192         if(*val) x509_name_ex_free(val, NULL);
193         if(!x509_name_ex_new(&nm.a, NULL)) goto err;
```



```
194         /* We've decoded it: now cache encoding */
195         if(!BUF_MEM_grow(nm.x->bytes, p - q)) goto err;
196         memcpy(nm.x->bytes->data, q, p - q);

198         /* Convert internal representation to X509_NAME structure */
199         for(i = 0; i < sk_STACK_OF_X509_NAME_ENTRY_num(intname.s); i++) {
200                 entries = sk_STACK_OF_X509_NAME_ENTRY_value(intname.s, i);
201                 for(j = 0; j < sk_X509_NAME_ENTRY_num(entries); j++) {
202                         entry = sk_X509_NAME_ENTRY_value(entries, j);
203                         entry->set = i;
204                         if(!sk_X509_NAME_ENTRY_push(nm.x->entries, entry))
205                                 goto err;
206                 }
207                 sk_X509_NAME_ENTRY_free(entries);
208         }
209         sk_STACK_OF_X509_NAME_ENTRY_free(intname.s);
210         ret = x509_name_canon(nm.x);
211         if (!ret)
212                 goto err;
213         nm.x->modified = 0;
214         *val = nm.a;
215         *in = p;
216         return ret;
217 err:
218         if (nm.x != NULL)
219                 X509_NAME_free(nm.x);
220         ASN1err(ASN1_F_X509_NAME_EX_D2I, ERR_R_NESTED_ASN1_ERROR);
221         return 0;
222 }

224 static int x509_name_ex_i2d(ASN1_VALUE **val, unsigned char **out, const ASN1_IT
225 {
226         int ret;
227         X509_NAME *a = (X509_NAME *)*val;
228         if(a->modified) {
229                 ret = x509_name_encode(a);
230                 if(ret < 0)
231                         return ret;
232                 ret = x509_name_canon(a);
233                 if(ret < 0)
234                         return ret;
235         }
236         ret = a->bytes->length;
237         if(out != NULL) {
238                 memcpy(*out,a->bytes->data,ret);
239                 *out+=ret;
240         }
241         return ret;
242 }

244 static void local_sk_X509_NAME_ENTRY_free(STACK_OF(X509_NAME_ENTRY) *ne)
245         {
246         sk_X509_NAME_ENTRY_free(ne);
247         }

249 static void local_sk_X509_NAME_ENTRY_pop_free(STACK_OF(X509_NAME_ENTRY) *ne)
250         {
251         sk_X509_NAME_ENTRY_pop_free(ne, X509_NAME_ENTRY_free);
252         }

254 static int x509_name_encode(X509_NAME *a)
255 {
256         union { STACK_OF(STACK_OF_X509_NAME_ENTRY) *s;
257                 ASN1_VALUE *a; } intname = {NULL};
258         int len;
259         unsigned char *p;
```

```
260         STACK_OF(X509_NAME_ENTRY) *entries = NULL;
261         X509_NAME_ENTRY *entry;
262         int i, set = -1;
263         intname.s = sk_STACK_OF_X509_NAME_ENTRY_new_null();
264         if(!intname.s) goto memerr;
265         for(i = 0; i < sk_X509_NAME_ENTRY_num(a->entries); i++) {
266                 entry = sk_X509_NAME_ENTRY_value(a->entries, i);
267                 if(entry->set != set) {
268                         entries = sk_X509_NAME_ENTRY_new_null();
269                         if(!entries) goto memerr;
270                         if(!sk_STACK_OF_X509_NAME_ENTRY_push(intname.s,
271                                                              entries))
272                                 goto memerr;
273                         set = entry->set;
274                 }
275                 if(!sk_X509_NAME_ENTRY_push(entries, entry)) goto memerr;
276         }
277         len = ASN1_item_ex_i2d(&intname.a, NULL,
278                                ASN1_ITEM_rptr(X509_NAME_INTERNAL), -1, -1);
279         if (!BUF_MEM_grow(a->bytes,len)) goto memerr;
280         p=(unsigned char *)a->bytes->data;
281         ASN1_item_ex_i2d(&intname.a,
282                          &p, ASN1_ITEM_rptr(X509_NAME_INTERNAL), -1, -1);
283         sk_STACK_OF_X509_NAME_ENTRY_pop_free(intname.s,
284                                              local_sk_X509_NAME_ENTRY_free);
285         a->modified = 0;
286         return len;
287 memerr:
288         sk_STACK_OF_X509_NAME_ENTRY_pop_free(intname.s,
289                                              local_sk_X509_NAME_ENTRY_free);
290         ASN1err(ASN1_F_X509_NAME_ENCODE, ERR_R_MALLOC_FAILURE);
291         return -1;
292 }

294 static int x509_name_ex_print(BIO *out, ASN1_VALUE **pval,
295                                                 int indent,
296                                                 const char *fname,
297                                                 const ASN1_PCTX *pctx)
298         {
299         if (X509_NAME_print_ex(out, (X509_NAME *)*pval,
300                                        indent, pctx->nm_flags) <= 0)
301                 return 0;
302         return 2;
303         }

305 /* This function generates the canonical encoding of the Name structure.
306  * In it all strings are converted to UTF8, leading, trailing and
307  * multiple spaces collapsed, converted to lower case and the leading
308  * SEQUENCE header removed.
309  *
310  * In future we could also normalize the UTF8 too.
311  *
312  * By doing this comparison of Name structures can be rapidly
313  * perfomed by just using memcmp() of the canonical encoding.
314  * By omitting the leading SEQUENCE name constraints of type
315  * dirName can also be checked with a simple memcmp().
316  */

318 static int x509_name_canon(X509_NAME *a)
319         {
320         unsigned char *p;
321         STACK_OF(STACK_OF_X509_NAME_ENTRY) *intname = NULL;
322         STACK_OF(X509_NAME_ENTRY) *entries = NULL;
323         X509_NAME_ENTRY *entry, *tmpentry = NULL;
324         int i, set = -1, ret = 0;
```



```
326         if (a->canon_enc)
327                 {
328                 OPENSSL_free(a->canon_enc);
329                 a->canon_enc = NULL;
330                 }
331         /* Special case: empty X509_NAME => null encoding */
332         if (sk_X509_NAME_ENTRY_num(a->entries) == 0)
333                 {
334                 a->canon_enclen = 0;
335                 return 1;
336                 }
337         intname = sk_STACK_OF_X509_NAME_ENTRY_new_null();
338         if(!intname)
339                 goto err;
340         for(i = 0; i < sk_X509_NAME_ENTRY_num(a->entries); i++)
341                 {
342                 entry = sk_X509_NAME_ENTRY_value(a->entries, i);
343                 if(entry->set != set)
344                         {
345                         entries = sk_X509_NAME_ENTRY_new_null();
346                         if(!entries)
347                                 goto err;
348                         if(!sk_STACK_OF_X509_NAME_ENTRY_push(intname, entries))
349                                 goto err;
350                         set = entry->set;
351                         }
352                 tmpentry = X509_NAME_ENTRY_new();
353                 tmpentry->object = OBJ_dup(entry->object);
354                 if (!asn1_string_canon(tmpentry->value, entry->value))
355                         goto err;
356                 if(!sk_X509_NAME_ENTRY_push(entries, tmpentry))
357                         goto err;
358                 tmpentry = NULL;
359                 }

361         /* Finally generate encoding */

363         a->canon_enclen = i2d_name_canon(intname, NULL);

365         p = OPENSSL_malloc(a->canon_enclen);

367         if (!p)
368                 goto err;

370         a->canon_enc = p;

372         i2d_name_canon(intname, &p);

374         ret = 1;

376         err:

378         if (tmpentry)
379                 X509_NAME_ENTRY_free(tmpentry);
380         if (intname)
381                 sk_STACK_OF_X509_NAME_ENTRY_pop_free(intname,
382                                         local_sk_X509_NAME_ENTRY_pop_free);
383         return ret;
384         }

386 /* Bitmap of all the types of string that will be canonicalized. */

388 #define ASN1_MASK_CANON \
389         (B_ASN1_UTF8STRING | B_ASN1_BMPSTRING | B_ASN1_UNIVERSALSTRING \
390         | B_ASN1_PRINTABLESTRING | B_ASN1_T61STRING | B_ASN1_IA5STRING \
391         | B_ASN1_VISIBLESTRING)
```

```
394 static int asn1_string_canon(ASN1_STRING *out, ASN1_STRING *in)
395         {
396         unsigned char *to, *from;
397         int len, i;

399         /* If type not in bitmask just copy string across */
400         if (!(ASN1_tag2bit(in->type) & ASN1_MASK_CANON))
401                 {
402                 if (!ASN1_STRING_copy(out, in))
403                         return 0;
404                 return 1;
405                 }

407         out->type = V_ASN1_UTF8STRING;
408         out->length = ASN1_STRING_to_UTF8(&out->data, in);
409         if (out->length == -1)
410                 return 0;

412         to = out->data;
413         from = to;

415         len = out->length;

417         /* Convert string in place to canonical form.
418          * Ultimately we may need to handle a wider range of characters
419          * but for now ignore anything with MSB set and rely on the
420          * isspace() and tolower() functions.
421          */

423         /* Ignore leading spaces */
424         while((len > 0) && !(*from & 0x80) && isspace(*from))
425                 {
426                 from++;
427                 len--;
428                 }

430         to = from + len - 1;

432         /* Ignore trailing spaces */
433         while ((len > 0) && !(*to & 0x80) && isspace(*to))
434                 {
435                 to--;
436                 len--;
437                 }

439         to = out->data;

441         i = 0;
442         while(i < len)
443                 {
444                 /* If MSB set just copy across */
445                 if (*from & 0x80)
446                         {
447                         *to++ = *from++;
448                         i++;
449                         }
450                 /* Collapse multiple spaces */
451                 else if (isspace(*from))
452                         {
453                         /* Copy one space across */
454                         *to++ = ' ';
455                         /* Ignore subsequent spaces. Note: don't need to
456                          * check len here because we know the last
457                          * character is a non-space so we can't overflow.
```



```
458                         */
459                         do
460                                 {
461                                 from++;
462                                 i++;
463                                 }
464                         while(!(*from & 0x80) && isspace(*from));
465                         }
466                 else
467                         {
468                         *to++ = tolower(*from);
469                         from++;
470                         i++;
471                         }
472                 }

474         out->length = to - out->data;

476         return 1;

478         }

480 static int i2d_name_canon(STACK_OF(STACK_OF_X509_NAME_ENTRY) *_intname,
481                           unsigned char **in)
482         {
483         int i, len, ltmp;
484         ASN1_VALUE *v;
485         STACK_OF(ASN1_VALUE) *intname = (STACK_OF(ASN1_VALUE) *)_intname;

487         len = 0;
488         for (i = 0; i < sk_ASN1_VALUE_num(intname); i++)
489                 {
490                 v = sk_ASN1_VALUE_value(intname, i);
491                 ltmp = ASN1_item_ex_i2d(&v, in,
492                         ASN1_ITEM_rptr(X509_NAME_ENTRIES), -1, -1);
493                 if (ltmp < 0)
494                         return ltmp;
495                 len += ltmp;
496                 }
497         return len;
498         }

500 int X509_NAME_set(X509_NAME **xn, X509_NAME *name)
501         {
502         X509_NAME *in;

504         if (!xn || !name) return(0);

506         if (*xn != name)
507                 {
508                 in=X509_NAME_dup(name);
509                 if (in != NULL)
510                         {
511                         X509_NAME_free(*xn);
512                         *xn=in;
513                         }
514                 }
515         return(*xn != NULL);
516         }

518 IMPLEMENT_STACK_OF(X509_NAME_ENTRY)
519 IMPLEMENT_ASN1_SET_OF(X509_NAME_ENTRY)
520 #endif /* ! codereview */
```

```
**********************************************************
    3082 Wed Aug 13 19:52:08 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_nx509.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x_nx509.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2005.
 */
/* ====================================================================
 * Copyright (c) 2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stddef.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>
```



```
#include <openssl/asn1t.h>

/* Old netscape certificate wrapper format */

ASN1_SEQUENCE(NETSCAPE_X509) = {
        ASN1_SIMPLE(NETSCAPE_X509, header, ASN1_OCTET_STRING),
        ASN1_OPT(NETSCAPE_X509, cert, X509)
} ASN1_SEQUENCE_END(NETSCAPE_X509)

IMPLEMENT_ASN1_FUNCTIONS(NETSCAPE_X509)
#endif /* ! codereview */
```

```
**********************************************************
    5429 Wed Aug 13 19:52:08 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_pkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_pkey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/asn1_mac.h>
#include <openssl/x509.h>

/* need to implement */
int i2d_X509_PKEY(X509_PKEY *a, unsigned char **pp)
        {
        return(0);
        }

X509_PKEY *d2i_X509_PKEY(X509_PKEY **a, const unsigned char **pp, long length)
        {
        int i;
        M_ASN1_D2I_vars(a,X509_PKEY *,X509_PKEY_new);

        M_ASN1_D2I_Init();
        M_ASN1_D2I_start_sequence();
        M_ASN1_D2I_get_x(X509_ALGOR,ret->enc_algor,d2i_X509_ALGOR);
        M_ASN1_D2I_get_x(ASN1_OCTET_STRING,ret->enc_pkey,d2i_ASN1_OCTET_STRING);

        ret->cipher.cipher=EVP_get_cipherbyname(
                OBJ_nid2ln(OBJ_obj2nid(ret->enc_algor->algorithm)));
        if (ret->cipher.cipher == NULL)
                {
                c.error=ASN1_R_UNSUPPORTED_CIPHER;
                c.line=__LINE__;
                goto err;
                }
        if (ret->enc_algor->parameter->type == V_ASN1_OCTET_STRING)
                {
                i=ret->enc_algor->parameter->value.octet_string->length;
                if (i > EVP_MAX_IV_LENGTH)
                        {
                        c.error=ASN1_R_IV_TOO_LARGE;
                        c.line=__LINE__;
                        goto err;
                        }
                memcpy(ret->cipher.iv,
                        ret->enc_algor->parameter->value.octet_string->data,i);
                }
        else
                memset(ret->cipher.iv,0,EVP_MAX_IV_LENGTH);
        M_ASN1_D2I_Finish(a,X509_PKEY_free,ASN1_F_D2I_X509_PKEY);
        }

X509_PKEY *X509_PKEY_new(void)
        {
        X509_PKEY *ret=NULL;
        ASN1_CTX c;

        M_ASN1_New_Malloc(ret,X509_PKEY);
        ret->version=0;
        M_ASN1_New(ret->enc_algor,X509_ALGOR_new);
        M_ASN1_New(ret->enc_pkey,M_ASN1_OCTET_STRING_new);
        ret->dec_pkey=NULL;
        ret->key_length=0;
        ret->key_data=NULL;
        ret->key_free=0;
        ret->cipher.cipher=NULL;
        memset(ret->cipher.iv,0,EVP_MAX_IV_LENGTH);
        ret->references=1;
        return(ret);
        M_ASN1_New_Error(ASN1_F_X509_PKEY_NEW);
        }

void X509_PKEY_free(X509_PKEY *x)
```

```
128         {
129         int i;

131         if (x == NULL) return;

133         i=CRYPTO_add(&x->references,-1,CRYPTO_LOCK_X509_PKEY);
134 #ifdef REF_PRINT
135         REF_PRINT("X509_PKEY",x);
136 #endif
137         if (i > 0) return;
138 #ifdef REF_CHECK
139         if (i < 0)
140                 {
141                 fprintf(stderr,"X509_PKEY_free, bad reference count\n");
142                 abort();
143                 }
144 #endif

146         if (x->enc_algor != NULL) X509_ALGOR_free(x->enc_algor);
147         if (x->enc_pkey != NULL) M_ASN1_OCTET_STRING_free(x->enc_pkey);
148         if (x->dec_pkey != NULL)EVP_PKEY_free(x->dec_pkey);
149         if ((x->key_data != NULL) && (x->key_free)) OPENSSL_free(x->key_data);
150         OPENSSL_free(x);
151         }
152 #endif /* ! codereview */
```

```
**********************************************************
    9506 Wed Aug 13 19:52:08 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_pubkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_pubkey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>
#include "asn1_locl.h"
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif

/* Minor tweak to operation: free up EVP_PKEY */
static int pubkey_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                     void *exarg)
        {
        if (operation == ASN1_OP_FREE_POST)
                {
                X509_PUBKEY *pubkey = (X509_PUBKEY *)*pval;
                EVP_PKEY_free(pubkey->pkey);
                }
        return 1;
        }

ASN1_SEQUENCE_cb(X509_PUBKEY, pubkey_cb) = {
        ASN1_SIMPLE(X509_PUBKEY, algor, X509_ALGOR),
        ASN1_SIMPLE(X509_PUBKEY, public_key, ASN1_BIT_STRING)
} ASN1_SEQUENCE_END_cb(X509_PUBKEY, X509_PUBKEY)

IMPLEMENT_ASN1_FUNCTIONS(X509_PUBKEY)

int X509_PUBKEY_set(X509_PUBKEY **x, EVP_PKEY *pkey)
        {
        X509_PUBKEY *pk=NULL;

        if (x == NULL) return(0);

        if ((pk=X509_PUBKEY_new()) == NULL) goto error;

        if (pkey->ameth)
                {
                if (pkey->ameth->pub_encode)
                        {
                        if (!pkey->ameth->pub_encode(pk, pkey))
                                {
                                X509err(X509_F_X509_PUBKEY_SET,
                                        X509_R_PUBLIC_KEY_ENCODE_ERROR);
                                goto error;
                                }
                        }
                else
                        {
                        X509err(X509_F_X509_PUBKEY_SET,
                                X509_R_METHOD_NOT_SUPPORTED);
                        goto error;
                        }
                }
        else
                {
                X509err(X509_F_X509_PUBKEY_SET,X509_R_UNSUPPORTED_ALGORITHM);
                goto error;
                }

        if (*x != NULL)
                X509_PUBKEY_free(*x);

        *x=pk;

        return 1;
```

```
128 error:
129         if (pk != NULL) X509_PUBKEY_free(pk);
130         return 0;
131         }

133 EVP_PKEY *X509_PUBKEY_get(X509_PUBKEY *key)
134         {
135         EVP_PKEY *ret=NULL;

137         if (key == NULL) goto error;

139         if (key->pkey != NULL)
140                 {
141                 CRYPTO_add(&key->pkey->references, 1, CRYPTO_LOCK_EVP_PKEY);
142                 return key->pkey;
143                 }

145         if (key->public_key == NULL) goto error;

147         if ((ret = EVP_PKEY_new()) == NULL)
148                 {
149                 X509err(X509_F_X509_PUBKEY_GET, ERR_R_MALLOC_FAILURE);
150                 goto error;
151                 }

153         if (!EVP_PKEY_set_type(ret, OBJ_obj2nid(key->algor->algorithm)))
154                 {
155                 X509err(X509_F_X509_PUBKEY_GET,X509_R_UNSUPPORTED_ALGORITHM);
156                 goto error;
157                 }

159         if (ret->ameth->pub_decode)
160                 {
161                 if (!ret->ameth->pub_decode(ret, key))
162                         {
163                         X509err(X509_F_X509_PUBKEY_GET,
164                                                 X509_R_PUBLIC_KEY_DECODE_ERROR);
165                         goto error;
166                         }
167                 }
168         else
169                 {
170                 X509err(X509_F_X509_PUBKEY_GET, X509_R_METHOD_NOT_SUPPORTED);
171                 goto error;
172                 }

174         /* Check to see if another thread set key->pkey first */
175         CRYPTO_w_lock(CRYPTO_LOCK_EVP_PKEY);
176         if (key->pkey)
177                 {
178                 CRYPTO_w_unlock(CRYPTO_LOCK_EVP_PKEY);
179                 EVP_PKEY_free(ret);
180                 ret = key->pkey;
181                 }
182         else
183                 {
184                 key->pkey = ret;
185                 CRYPTO_w_unlock(CRYPTO_LOCK_EVP_PKEY);
186                 }
187         CRYPTO_add(&ret->references, 1, CRYPTO_LOCK_EVP_PKEY);

189         return ret;

191         error:
192         if (ret != NULL)
193                 EVP_PKEY_free(ret);
```



```
194         return(NULL);
195         }

197 /* Now two pseudo ASN1 routines that take an EVP_PKEY structure
198  * and encode or decode as X509_PUBKEY
199  */

201 EVP_PKEY *d2i_PUBKEY(EVP_PKEY **a, const unsigned char **pp,
202                      long length)
203         {
204         X509_PUBKEY *xpk;
205         EVP_PKEY *pktmp;
206         xpk = d2i_X509_PUBKEY(NULL, pp, length);
207         if(!xpk) return NULL;
208         pktmp = X509_PUBKEY_get(xpk);
209         X509_PUBKEY_free(xpk);
210         if(!pktmp) return NULL;
211         if(a)
212                 {
213                 EVP_PKEY_free(*a);
214                 *a = pktmp;
215                 }
216         return pktmp;
217         }

219 int i2d_PUBKEY(EVP_PKEY *a, unsigned char **pp)
220         {
221         X509_PUBKEY *xpk=NULL;
222         int ret;
223         if(!a) return 0;
224         if(!X509_PUBKEY_set(&xpk, a)) return 0;
225         ret = i2d_X509_PUBKEY(xpk, pp);
226         X509_PUBKEY_free(xpk);
227         return ret;
228         }

230 /* The following are equivalents but which return RSA and DSA
231  * keys
232  */
233 #ifndef OPENSSL_NO_RSA
234 RSA *d2i_RSA_PUBKEY(RSA **a, const unsigned char **pp,
235                     long length)
236         {
237         EVP_PKEY *pkey;
238         RSA *key;
239         const unsigned char *q;
240         q = *pp;
241         pkey = d2i_PUBKEY(NULL, &q, length);
242         if (!pkey) return NULL;
243         key = EVP_PKEY_get1_RSA(pkey);
244         EVP_PKEY_free(pkey);
245         if (!key) return NULL;
246         *pp = q;
247         if (a)
248                 {
249                 RSA_free(*a);
250                 *a = key;
251                 }
252         return key;
253         }

255 int i2d_RSA_PUBKEY(RSA *a, unsigned char **pp)
256         {
257         EVP_PKEY *pktmp;
258         int ret;
259         if (!a) return 0;
```

```
260         pktmp = EVP_PKEY_new();
261         if (!pktmp)
262                 {
263                 ASN1err(ASN1_F_I2D_RSA_PUBKEY, ERR_R_MALLOC_FAILURE);
264                 return 0;
265                 }
266         EVP_PKEY_set1_RSA(pktmp, a);
267         ret = i2d_PUBKEY(pktmp, pp);
268         EVP_PKEY_free(pktmp);
269         return ret;
270         }
271 #endif

273 #ifndef OPENSSL_NO_DSA
274 DSA *d2i_DSA_PUBKEY(DSA **a, const unsigned char **pp,
275              long length)
276         {
277         EVP_PKEY *pkey;
278         DSA *key;
279         const unsigned char *q;
280         q = *pp;
281         pkey = d2i_PUBKEY(NULL, &q, length);
282         if (!pkey) return NULL;
283         key = EVP_PKEY_get1_DSA(pkey);
284         EVP_PKEY_free(pkey);
285         if (!key) return NULL;
286         *pp = q;
287         if (a)
288                 {
289                 DSA_free(*a);
290                 *a = key;
291                 }
292         return key;
293         }

295 int i2d_DSA_PUBKEY(DSA *a, unsigned char **pp)
296         {
297         EVP_PKEY *pktmp;
298         int ret;
299         if(!a) return 0;
300         pktmp = EVP_PKEY_new();
301         if(!pktmp)
302                 {
303                 ASN1err(ASN1_F_I2D_DSA_PUBKEY, ERR_R_MALLOC_FAILURE);
304                 return 0;
305                 }
306         EVP_PKEY_set1_DSA(pktmp, a);
307         ret = i2d_PUBKEY(pktmp, pp);
308         EVP_PKEY_free(pktmp);
309         return ret;
310         }
311 #endif

313 #ifndef OPENSSL_NO_EC
314 EC_KEY *d2i_EC_PUBKEY(EC_KEY **a, const unsigned char **pp, long length)
315         {
316         EVP_PKEY *pkey;
317         EC_KEY *key;
318         const unsigned char *q;
319         q = *pp;
320         pkey = d2i_PUBKEY(NULL, &q, length);
321         if (!pkey) return(NULL);
322         key = EVP_PKEY_get1_EC_KEY(pkey);
323         EVP_PKEY_free(pkey);
324         if (!key)  return(NULL);
325         *pp = q;
```



```
326         if (a)
327                 {
328                 EC_KEY_free(*a);
329                 *a = key;
330                 }
331         return(key);
332         }

334 int i2d_EC_PUBKEY(EC_KEY *a, unsigned char **pp)
335         {
336         EVP_PKEY *pktmp;
337         int ret;
338         if (!a) return(0);
339         if ((pktmp = EVP_PKEY_new()) == NULL)
340                 {
341                 ASN1err(ASN1_F_I2D_EC_PUBKEY, ERR_R_MALLOC_FAILURE);
342                 return(0);
343                 }
344         EVP_PKEY_set1_EC_KEY(pktmp, a);
345         ret = i2d_PUBKEY(pktmp, pp);
346         EVP_PKEY_free(pktmp);
347         return(ret);
348         }
349 #endif

351 int X509_PUBKEY_set0_param(X509_PUBKEY *pub, ASN1_OBJECT *aobj,
352                                         int ptype, void *pval,
353                                         unsigned char *penc, int penclen)
354         {
355         if (!X509_ALGOR_set0(pub->algor, aobj, ptype, pval))
356                 return 0;
357         if (penc)
358                 {
359                 if (pub->public_key->data)
360                         OPENSSL_free(pub->public_key->data);
361                 pub->public_key->data = penc;
362                 pub->public_key->length = penclen;
363                 /* Set number of unused bits to zero */
364                 pub->public_key->flags&= ~(ASN1_STRING_FLAG_BITS_LEFT|0x07);
365                 pub->public_key->flags|=ASN1_STRING_FLAG_BITS_LEFT;
366                 }
367         return 1;
368         }

370 int X509_PUBKEY_get0_param(ASN1_OBJECT **ppkalg,
371                 const unsigned char **pk, int *ppklen,
372                 X509_ALGOR **pa,
373                 X509_PUBKEY *pub)
374         {
375         if (ppkalg)
376                 *ppkalg = pub->algor->algorithm;
377         if (pk)
378                 {
379                 *pk = pub->public_key->data;
380                 *ppklen = pub->public_key->length;
381                 }
382         if (pa)
383                 *pa = pub->algor;
384         return 1;
385         }
386 #endif /* ! codereview */
```

```
**********************************************************
    5061 Wed Aug 13 19:52:08 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_req.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_req.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>

/* X509_REQ_INFO is handled in an unusual way to get round
 * invalid encodings. Some broken certificate requests don't
 * encode the attributes field if it is empty. This is in
 * violation of PKCS#10 but we need to tolerate it. We do
 * this by making the attributes field OPTIONAL then using
 * the callback to initialise it to an empty STACK.
 *
 * This means that the field will be correctly encoded unless
 * we NULL out the field.
 *
 * As a result we no longer need the req_kludge field because
 * the information is now contained in the attributes field:
 * 1. If it is NULL then it's the invalid omission.
 * 2. If it is empty it is the correct encoding.
 * 3. If it is not empty then some attributes are present.
 *
 */

static int rinf_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                        void *exarg)
{
        X509_REQ_INFO *rinf = (X509_REQ_INFO *)*pval;

        if(operation == ASN1_OP_NEW_POST) {
                rinf->attributes = sk_X509_ATTRIBUTE_new_null();
                if(!rinf->attributes) return 0;
        }
        return 1;
}

ASN1_SEQUENCE_enc(X509_REQ_INFO, enc, rinf_cb) = {
        ASN1_SIMPLE(X509_REQ_INFO, version, ASN1_INTEGER),
        ASN1_SIMPLE(X509_REQ_INFO, subject, X509_NAME),
        ASN1_SIMPLE(X509_REQ_INFO, pubkey, X509_PUBKEY),
        /* This isn't really OPTIONAL but it gets round invalid
         * encodings
         */
        ASN1_IMP_SET_OF_OPT(X509_REQ_INFO, attributes, X509_ATTRIBUTE, 0)
} ASN1_SEQUENCE_END_enc(X509_REQ_INFO, X509_REQ_INFO)

IMPLEMENT_ASN1_FUNCTIONS(X509_REQ_INFO)

ASN1_SEQUENCE_ref(X509_REQ, 0, CRYPTO_LOCK_X509_REQ) = {
        ASN1_SIMPLE(X509_REQ, req_info, X509_REQ_INFO),
        ASN1_SIMPLE(X509_REQ, sig_alg, X509_ALGOR),
        ASN1_SIMPLE(X509_REQ, signature, ASN1_BIT_STRING)
} ASN1_SEQUENCE_END_ref(X509_REQ, X509_REQ)

IMPLEMENT_ASN1_FUNCTIONS(X509_REQ)
IMPLEMENT_ASN1_DUP_FUNCTION(X509_REQ)
#endif /* ! codereview */
```

```
**********************************************************
    3472 Wed Aug 13 19:52:08 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_sig.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_sig.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>

ASN1_SEQUENCE(X509_SIG) = {
        ASN1_SIMPLE(X509_SIG, algor, X509_ALGOR),
        ASN1_SIMPLE(X509_SIG, digest, ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END(X509_SIG)

IMPLEMENT_ASN1_FUNCTIONS(X509_SIG)
#endif /* ! codereview */
```

****
3908 Wed Aug 13 19:52:09 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_spki.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* crypto/asn1/x_spki.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* This module was send to me my Pat Richards <patr@x509.com> who
 * wrote it.  It is under my Copyright with his permission
 */
```



```
#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/asn1t.h>

ASN1_SEQUENCE(NETSCAPE_SPKAC) = {
        ASN1_SIMPLE(NETSCAPE_SPKAC, pubkey, X509_PUBKEY),
        ASN1_SIMPLE(NETSCAPE_SPKAC, challenge, ASN1_IA5STRING)
} ASN1_SEQUENCE_END(NETSCAPE_SPKAC)

IMPLEMENT_ASN1_FUNCTIONS(NETSCAPE_SPKAC)

ASN1_SEQUENCE(NETSCAPE_SPKI) = {
        ASN1_SIMPLE(NETSCAPE_SPKI, spkac, NETSCAPE_SPKAC),
        ASN1_SIMPLE(NETSCAPE_SPKI, sig_algor, X509_ALGOR),
        ASN1_SIMPLE(NETSCAPE_SPKI, signature, ASN1_BIT_STRING)
} ASN1_SEQUENCE_END(NETSCAPE_SPKI)

IMPLEMENT_ASN1_FUNCTIONS(NETSCAPE_SPKI)
#endif /* ! codereview */
```

```
**********************************************************
    3469 Wed Aug 13 19:52:09 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_val.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_val.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>

ASN1_SEQUENCE(X509_VAL) = {
        ASN1_SIMPLE(X509_VAL, notBefore, ASN1_TIME),
        ASN1_SIMPLE(X509_VAL, notAfter, ASN1_TIME)
} ASN1_SEQUENCE_END(X509_VAL)

IMPLEMENT_ASN1_FUNCTIONS(X509_VAL)
#endif /* ! codereview */
```

```
**********************************************************
    6909 Wed Aug 13 19:52:09 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_x509.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/x_x509.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/asn1t.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

ASN1_SEQUENCE_enc(X509_CINF, enc, 0) = {
        ASN1_EXP_OPT(X509_CINF, version, ASN1_INTEGER, 0),
        ASN1_SIMPLE(X509_CINF, serialNumber, ASN1_INTEGER),
        ASN1_SIMPLE(X509_CINF, signature, X509_ALGOR),
        ASN1_SIMPLE(X509_CINF, issuer, X509_NAME),
        ASN1_SIMPLE(X509_CINF, validity, X509_VAL),
        ASN1_SIMPLE(X509_CINF, subject, X509_NAME),
        ASN1_SIMPLE(X509_CINF, key, X509_PUBKEY),
        ASN1_IMP_OPT(X509_CINF, issuerUID, ASN1_BIT_STRING, 1),
        ASN1_IMP_OPT(X509_CINF, subjectUID, ASN1_BIT_STRING, 2),
        ASN1_EXP_SEQUENCE_OF_OPT(X509_CINF, extensions, X509_EXTENSION, 3)
} ASN1_SEQUENCE_END_enc(X509_CINF, X509_CINF)

IMPLEMENT_ASN1_FUNCTIONS(X509_CINF)
/* X509 top level structure needs a bit of customisation */

extern void policy_cache_free(X509_POLICY_CACHE *cache);

static int x509_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                                void *exarg)
{
        X509 *ret = (X509 *)*pval;

        switch(operation) {

                case ASN1_OP_NEW_POST:
                ret->valid=0;
                ret->name = NULL;
                ret->ex_flags = 0;
                ret->ex_pathlen = -1;
                ret->skid = NULL;
                ret->akid = NULL;
#ifndef OPENSSL_NO_RFC3779
                ret->rfc3779_addr = NULL;
                ret->rfc3779_asid = NULL;
#endif
                ret->aux = NULL;
                ret->crldp = NULL;
                CRYPTO_new_ex_data(CRYPTO_EX_INDEX_X509, ret, &ret->ex_data);
                break;

                case ASN1_OP_D2I_POST:
                if (ret->name != NULL) OPENSSL_free(ret->name);
                ret->name=X509_NAME_oneline(ret->cert_info->subject,NULL,0);
                break;

                case ASN1_OP_FREE_POST:
                CRYPTO_free_ex_data(CRYPTO_EX_INDEX_X509, ret, &ret->ex_data);
                X509_CERT_AUX_free(ret->aux);
                ASN1_OCTET_STRING_free(ret->skid);
                AUTHORITY_KEYID_free(ret->akid);
                CRL_DIST_POINTS_free(ret->crldp);
                policy_cache_free(ret->policy_cache);
                GENERAL_NAMES_free(ret->altname);
                NAME_CONSTRAINTS_free(ret->nc);
#ifndef OPENSSL_NO_RFC3779
                sk_IPAddressFamily_pop_free(ret->rfc3779_addr, IPAddressFamily_f
                ASIdentifiers_free(ret->rfc3779_asid);
#endif

                if (ret->name != NULL) OPENSSL_free(ret->name);
                break;
```

```
129         }

131         return 1;

133 }

135 ASN1_SEQUENCE_ref(X509, x509_cb, CRYPTO_LOCK_X509) = {
136         ASN1_SIMPLE(X509, cert_info, X509_CINF),
137         ASN1_SIMPLE(X509, sig_alg, X509_ALGOR),
138         ASN1_SIMPLE(X509, signature, ASN1_BIT_STRING)
139 } ASN1_SEQUENCE_END_ref(X509, X509)

141 IMPLEMENT_ASN1_FUNCTIONS(X509)
142 IMPLEMENT_ASN1_DUP_FUNCTION(X509)

144 int X509_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
145             CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
146         {
147         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_X509, argl, argp,
148                                 new_func, dup_func, free_func);
149         }

151 int X509_set_ex_data(X509 *r, int idx, void *arg)
152         {
153         return(CRYPTO_set_ex_data(&r->ex_data,idx,arg));
154         }

156 void *X509_get_ex_data(X509 *r, int idx)
157         {
158         return(CRYPTO_get_ex_data(&r->ex_data,idx));
159         }

161 /* X509_AUX ASN1 routines. X509_AUX is the name given to
162  * a certificate with extra info tagged on the end. Since these
163  * functions set how a certificate is trusted they should only
164  * be used when the certificate comes from a reliable source
165  * such as local storage.
166  *
167  */

169 X509 *d2i_X509_AUX(X509 **a, const unsigned char **pp, long length)
170 {
171         const unsigned char *q;
172         X509 *ret;
173         /* Save start position */
174         q = *pp;
175         ret = d2i_X509(a, pp, length);
176         /* If certificate unreadable then forget it */
177         if(!ret) return NULL;
178         /* update length */
179         length -= *pp - q;
180         if(!length) return ret;
181         if(!d2i_X509_CERT_AUX(&ret->aux, pp, length)) goto err;
182         return ret;
183         err:
184         X509_free(ret);
185         return NULL;
186 }

188 int i2d_X509_AUX(X509 *a, unsigned char **pp)
189 {
190         int length;
191         length = i2d_X509(a, pp);
192         if(a) length += i2d_X509_CERT_AUX(a->aux, pp);
193         return length;
```



```
194 }
195 #endif /* ! codereview */
```

```
**********************************************************
    5945 Wed Aug 13 19:52:09 2014
new/usr/src/lib/openssl/libsunw_crypto/asn1/x_x509a.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* a_x509a.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/asn1t.h>
#include <openssl/x509.h>

/* X509_CERT_AUX routines. These are used to encode additional
 * user modifiable data about a certificate. This data is
 * appended to the X509 encoding when the *_X509_AUX routines
 * are used. This means that the "traditional" X509 routines
 * will simply ignore the extra data.
 */

static X509_CERT_AUX *aux_get(X509 *x);

ASN1_SEQUENCE(X509_CERT_AUX) = {
        ASN1_SEQUENCE_OF_OPT(X509_CERT_AUX, trust, ASN1_OBJECT),
        ASN1_IMP_SEQUENCE_OF_OPT(X509_CERT_AUX, reject, ASN1_OBJECT, 0),
        ASN1_OPT(X509_CERT_AUX, alias, ASN1_UTF8STRING),
        ASN1_OPT(X509_CERT_AUX, keyid, ASN1_OCTET_STRING),
        ASN1_IMP_SEQUENCE_OF_OPT(X509_CERT_AUX, other, X509_ALGOR, 1)
} ASN1_SEQUENCE_END(X509_CERT_AUX)

IMPLEMENT_ASN1_FUNCTIONS(X509_CERT_AUX)

static X509_CERT_AUX *aux_get(X509 *x)
{
        if(!x) return NULL;
        if(!x->aux && !(x->aux = X509_CERT_AUX_new())) return NULL;
        return x->aux;
}

int X509_alias_set1(X509 *x, unsigned char *name, int len)
{
        X509_CERT_AUX *aux;
        if (!name)
                {
                if (!x || !x->aux || !x->aux->alias)
                        return 1;
                ASN1_UTF8STRING_free(x->aux->alias);
                x->aux->alias = NULL;
                return 1;
                }
        if(!(aux = aux_get(x))) return 0;
        if(!aux->alias && !(aux->alias = ASN1_UTF8STRING_new())) return 0;
        return ASN1_STRING_set(aux->alias, name, len);
}

int X509_keyid_set1(X509 *x, unsigned char *id, int len)
{
        X509_CERT_AUX *aux;
        if (!id)
                {
                if (!x || !x->aux || !x->aux->keyid)
                        return 1;
                ASN1_OCTET_STRING_free(x->aux->keyid);
                x->aux->keyid = NULL;
                return 1;
                }
        if(!(aux = aux_get(x))) return 0;
        if(!aux->keyid && !(aux->keyid = ASN1_OCTET_STRING_new())) return 0;
        return ASN1_STRING_set(aux->keyid, id, len);
}

unsigned char *X509_alias_get0(X509 *x, int *len)
{
        if(!x->aux || !x->aux->alias) return NULL;
        if(len) *len = x->aux->alias->length;
        return x->aux->alias->data;
```

```
128 }

130 unsigned char *X509_keyid_get0(X509 *x, int *len)
131 {
132         if(!x->aux || !x->aux->keyid) return NULL;
133         if(len) *len = x->aux->keyid->length;
134         return x->aux->keyid->data;
135 }

137 int X509_add1_trust_object(X509 *x, ASN1_OBJECT *obj)
138 {
139         X509_CERT_AUX *aux;
140         ASN1_OBJECT *objtmp;
141         if(!(objtmp = OBJ_dup(obj))) return 0;
142         if(!(aux = aux_get(x))) return 0;
143         if(!aux->trust
144                 && !(aux->trust = sk_ASN1_OBJECT_new_null())) return 0;
145         return sk_ASN1_OBJECT_push(aux->trust, objtmp);
146 }

148 int X509_add1_reject_object(X509 *x, ASN1_OBJECT *obj)
149 {
150         X509_CERT_AUX *aux;
151         ASN1_OBJECT *objtmp;
152         if(!(objtmp = OBJ_dup(obj))) return 0;
153         if(!(aux = aux_get(x))) return 0;
154         if(!aux->reject
155                 && !(aux->reject = sk_ASN1_OBJECT_new_null())) return 0;
156         return sk_ASN1_OBJECT_push(aux->reject, objtmp);
157 }

159 void X509_trust_clear(X509 *x)
160 {
161         if(x->aux && x->aux->trust) {
162                 sk_ASN1_OBJECT_pop_free(x->aux->trust, ASN1_OBJECT_free);
163                 x->aux->trust = NULL;
164         }
165 }

167 void X509_reject_clear(X509 *x)
168 {
169         if(x->aux && x->aux->reject) {
170                 sk_ASN1_OBJECT_pop_free(x->aux->reject, ASN1_OBJECT_free);
171                 x->aux->reject = NULL;
172         }
173 }

175 ASN1_SEQUENCE(X509_CERT_PAIR) = {
176         ASN1_EXP_OPT(X509_CERT_PAIR, forward, X509, 0),
177         ASN1_EXP_OPT(X509_CERT_PAIR, reverse, X509, 1)
178 } ASN1_SEQUENCE_END(X509_CERT_PAIR)

180 IMPLEMENT_ASN1_FUNCTIONS(X509_CERT_PAIR)
181 #endif /* ! codereview */
```

```
**********************************************************
    4451 Wed Aug 13 19:52:09 2014
new/usr/src/lib/openssl/libsunw_crypto/bf/bf_cfb64.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/bf_cfb64.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/blowfish.h>
#include "bf_locl.h"
```



```
/* The input and output encrypted as though 64bit cfb mode is being
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */

void BF_cfb64_encrypt(const unsigned char *in, unsigned char *out, long length,
             const BF_KEY *schedule, unsigned char *ivec, int *num, int encrypt)
        {
        register BF_LONG v0,v1,t;
        register int n= *num;
        register long l=length;
        BF_LONG ti[2];
        unsigned char *iv,c,cc;

        iv=(unsigned char *)ivec;
        if (encrypt)
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                n2l(iv,v0); ti[0]=v0;
                                n2l(iv,v1); ti[1]=v1;
                                BF_encrypt((BF_LONG *)ti,schedule);
                                iv=(unsigned char *)ivec;
                                t=ti[0]; l2n(t,iv);
                                t=ti[1]; l2n(t,iv);
                                iv=(unsigned char *)ivec;
                                }
                        c= *(in++)^iv[n];
                        *(out++)=c;
                        iv[n]=c;
                        n=(n+1)&0x07;
                        }
                }
        else
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                n2l(iv,v0); ti[0]=v0;
                                n2l(iv,v1); ti[1]=v1;
                                BF_encrypt((BF_LONG *)ti,schedule);
                                iv=(unsigned char *)ivec;
                                t=ti[0]; l2n(t,iv);
                                t=ti[1]; l2n(t,iv);
                                iv=(unsigned char *)ivec;
                                }
                        cc= *(in++);
                        c=iv[n];
                        iv[n]=cc;
                        *(out++)=c^cc;
                        n=(n+1)&0x07;
                        }
                }
        v0=v1=ti[0]=ti[1]=t=c=cc=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    3997 Wed Aug 13 19:52:09 2014
new/usr/src/lib/openssl/libsunw_crypto/bf/bf_ecb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/bf_ecb.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/blowfish.h>
#include "bf_locl.h"
#include <openssl/opensslv.h>
```



```
/* Blowfish as implemented from 'Blowfish: Springer-Verlag paper'
 * (From LECTURE NOTES IN COMPUTER SCIENCE 809, FAST SOFTWARE ENCRYPTION,
 * CAMBRIDGE SECURITY WORKSHOP, CAMBRIDGE, U.K., DECEMBER 9-11, 1993)
 */

const char BF_version[]="Blowfish" OPENSSL_VERSION_PTEXT;

const char *BF_options(void)
        {
#ifdef BF_PTR
        return("blowfish(ptr)");
#elif defined(BF_PTR2)
        return("blowfish(ptr2)");
#else
        return("blowfish(idx)");
#endif
        }

void BF_ecb_encrypt(const unsigned char *in, unsigned char *out,
             const BF_KEY *key, int encrypt)
        {
        BF_LONG l,d[2];

        n2l(in,l); d[0]=l;
        n2l(in,l); d[1]=l;
        if (encrypt)
                BF_encrypt(d,key);
        else
                BF_decrypt(d,key);
        l=d[0]; l2n(l,out);
        l=d[1]; l2n(l,out);
        l=d[0]=d[1]=0;
        }
#endif /* ! codereview */
```

```
**********************************************************
    7675 Wed Aug 13 19:52:09 2014
new/usr/src/lib/openssl/libsunw_crypto/bf/bf_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/bf_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/blowfish.h>
#include "bf_locl.h"
```



```
/* Blowfish as implemented from 'Blowfish: Springer-Verlag paper'
 * (From LECTURE NOTES IN COMPUTER SCIENCE 809, FAST SOFTWARE ENCRYPTION,
 * CAMBRIDGE SECURITY WORKSHOP, CAMBRIDGE, U.K., DECEMBER 9-11, 1993)
 */

#if (BF_ROUNDS != 16) && (BF_ROUNDS != 20)
#error If you set BF_ROUNDS to some value other than 16 or 20, you will have \
to modify the code.
#endif

void BF_encrypt(BF_LONG *data, const BF_KEY *key)
        {
#ifndef BF_PTR2
        register BF_LONG l,r;
        register const BF_LONG *p,*s;

        p=key->P;
        s= &(key->S[0]);
        l=data[0];
        r=data[1];

        l^=p[0];
        BF_ENC(r,l,s,p[ 1]);
        BF_ENC(l,r,s,p[ 2]);
        BF_ENC(r,l,s,p[ 3]);
        BF_ENC(l,r,s,p[ 4]);
        BF_ENC(r,l,s,p[ 5]);
        BF_ENC(l,r,s,p[ 6]);
        BF_ENC(r,l,s,p[ 7]);
        BF_ENC(l,r,s,p[ 8]);
        BF_ENC(r,l,s,p[ 9]);
        BF_ENC(l,r,s,p[10]);
        BF_ENC(r,l,s,p[11]);
        BF_ENC(l,r,s,p[12]);
        BF_ENC(r,l,s,p[13]);
        BF_ENC(l,r,s,p[14]);
        BF_ENC(r,l,s,p[15]);
        BF_ENC(l,r,s,p[16]);
#if BF_ROUNDS == 20
        BF_ENC(r,l,s,p[17]);
        BF_ENC(l,r,s,p[18]);
        BF_ENC(r,l,s,p[19]);
        BF_ENC(l,r,s,p[20]);
#endif
        r^=p[BF_ROUNDS+1];

        data[1]=l&0xffffffffL;
        data[0]=r&0xffffffffL;
#else
        register BF_LONG l,r,t,*k;

        l=data[0];
        r=data[1];
        k=(BF_LONG*)key;

        l^=k[0];
        BF_ENC(r,l,k, 1);
        BF_ENC(l,r,k, 2);
        BF_ENC(r,l,k, 3);
        BF_ENC(l,r,k, 4);
        BF_ENC(r,l,k, 5);
        BF_ENC(l,r,k, 6);
        BF_ENC(r,l,k, 7);
        BF_ENC(l,r,k, 8);
        BF_ENC(r,l,k, 9);
        BF_ENC(l,r,k,10);
```

```
128         BF_ENC(r,l,k,11);
129         BF_ENC(l,r,k,12);
130         BF_ENC(r,l,k,13);
131         BF_ENC(l,r,k,14);
132         BF_ENC(r,l,k,15);
133         BF_ENC(l,r,k,16);
134 #if BF_ROUNDS == 20
135         BF_ENC(r,l,k,17);
136         BF_ENC(l,r,k,18);
137         BF_ENC(r,l,k,19);
138         BF_ENC(l,r,k,20);
139 #endif
140         r^=k[BF_ROUNDS+1];

142         data[1]=l&0xffffffffL;
143         data[0]=r&0xffffffffL;
144 #endif
145         }

147 #ifndef BF_DEFAULT_OPTIONS

149 void BF_decrypt(BF_LONG *data, const BF_KEY *key)
150         {
151 #ifndef BF_PTR2
152         register BF_LONG l,r;
153         register const BF_LONG *p,*s;

155         p=key->P;
156         s= &(key->S[0]);
157         l=data[0];
158         r=data[1];

160         l^=p[BF_ROUNDS+1];
161 #if BF_ROUNDS == 20
162         BF_ENC(r,l,s,p[20]);
163         BF_ENC(l,r,s,p[19]);
164         BF_ENC(r,l,s,p[18]);
165         BF_ENC(l,r,s,p[17]);
166 #endif
167         BF_ENC(r,l,s,p[16]);
168         BF_ENC(l,r,s,p[15]);
169         BF_ENC(r,l,s,p[14]);
170         BF_ENC(l,r,s,p[13]);
171         BF_ENC(r,l,s,p[12]);
172         BF_ENC(l,r,s,p[11]);
173         BF_ENC(r,l,s,p[10]);
174         BF_ENC(l,r,s,p[ 9]);
175         BF_ENC(r,l,s,p[ 8]);
176         BF_ENC(l,r,s,p[ 7]);
177         BF_ENC(r,l,s,p[ 6]);
178         BF_ENC(l,r,s,p[ 5]);
179         BF_ENC(r,l,s,p[ 4]);
180         BF_ENC(l,r,s,p[ 3]);
181         BF_ENC(r,l,s,p[ 2]);
182         BF_ENC(l,r,s,p[ 1]);
183         r^=p[0];

185         data[1]=l&0xffffffffL;
186         data[0]=r&0xffffffffL;
187 #else
188         register BF_LONG l,r,t,*k;

190         l=data[0];
191         r=data[1];
192         k=(BF_LONG *)key;
```



```
194         l^=k[BF_ROUNDS+1];
195 #if BF_ROUNDS == 20
196         BF_ENC(r,l,k,20);
197         BF_ENC(l,r,k,19);
198         BF_ENC(r,l,k,18);
199         BF_ENC(l,r,k,17);
200 #endif
201         BF_ENC(r,l,k,16);
202         BF_ENC(l,r,k,15);
203         BF_ENC(r,l,k,14);
204         BF_ENC(l,r,k,13);
205         BF_ENC(r,l,k,12);
206         BF_ENC(l,r,k,11);
207         BF_ENC(r,l,k,10);
208         BF_ENC(l,r,k, 9);
209         BF_ENC(r,l,k, 8);
210         BF_ENC(l,r,k, 7);
211         BF_ENC(r,l,k, 6);
212         BF_ENC(l,r,k, 5);
213         BF_ENC(r,l,k, 4);
214         BF_ENC(l,r,k, 3);
215         BF_ENC(r,l,k, 2);
216         BF_ENC(l,r,k, 1);
217         r^=k[0];

219         data[1]=l&0xffffffffL;
220         data[0]=r&0xffffffffL;
221 #endif
222         }

224 void BF_cbc_encrypt(const unsigned char *in, unsigned char *out, long length,
225              const BF_KEY *schedule, unsigned char *ivec, int encrypt)
226         {
227         register BF_LONG tin0,tin1;
228         register BF_LONG tout0,tout1,xor0,xor1;
229         register long l=length;
230         BF_LONG tin[2];

232         if (encrypt)
233                 {
234                 n2l(ivec,tout0);
235                 n2l(ivec,tout1);
236                 ivec-=8;
237                 for (l-=8; l>=0; l-=8)
238                         {
239                         n2l(in,tin0);
240                         n2l(in,tin1);
241                         tin0^=tout0;
242                         tin1^=tout1;
243                         tin[0]=tin0;
244                         tin[1]=tin1;
245                         BF_encrypt(tin,schedule);
246                         tout0=tin[0];
247                         tout1=tin[1];
248                         l2n(tout0,out);
249                         l2n(tout1,out);
250                         }
251                 if (l != -8)
252                         {
253                         n2ln(in,tin0,tin1,l+8);
254                         tin0^=tout0;
255                         tin1^=tout1;
256                         tin[0]=tin0;
257                         tin[1]=tin1;
258                         BF_encrypt(tin,schedule);
259                         tout0=tin[0];
```

```
260                         tout1=tin[1];
261                         l2n(tout0,out);
262                         l2n(tout1,out);
263                         }
264                 l2n(tout0,ivec);
265                 l2n(tout1,ivec);
266                 }
267         else
268                 {
269                 n2l(ivec,xor0);
270                 n2l(ivec,xor1);
271                 ivec-=8;
272                 for (l-=8; l>=0; l-=8)
273                         {
274                         n2l(in,tin0);
275                         n2l(in,tin1);
276                         tin[0]=tin0;
277                         tin[1]=tin1;
278                         BF_decrypt(tin,schedule);
279                         tout0=tin[0]^xor0;
280                         tout1=tin[1]^xor1;
281                         l2n(tout0,out);
282                         l2n(tout1,out);
283                         xor0=tin0;
284                         xor1=tin1;
285                         }
286                 if (l != -8)
287                         {
288                         n2l(in,tin0);
289                         n2l(in,tin1);
290                         tin[0]=tin0;
291                         tin[1]=tin1;
292                         BF_decrypt(tin,schedule);
293                         tout0=tin[0]^xor0;
294                         tout1=tin[1]^xor1;
295                         l2nn(tout0,tout1,out,l+8);
296                         xor0=tin0;
297                         xor1=tin1;
298                         }
299                 l2n(xor0,ivec);
300                 l2n(xor1,ivec);
301                 }
302         tin0=tin1=tout0=tout1=xor0=xor1=0;
303         tin[0]=tin[1]=0;
304         }

306 #endif
307 #endif /* ! codereview */
```

```
**********************************************************
    4190 Wed Aug 13 19:52:10 2014
new/usr/src/lib/openssl/libsunw_crypto/bf/bf_ofb64.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/bf_ofb64.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/blowfish.h>
#include "bf_locl.h"
```



```
/* The input and output encrypted as though 64bit ofb mode is being
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */
void BF_ofb64_encrypt(const unsigned char *in, unsigned char *out, long length,
                      const BF_KEY *schedule, unsigned char *ivec, int *num)
        {
        register BF_LONG v0,v1,t;
        register int n= *num;
        register long l=length;
        unsigned char d[8];
        register char *dp;
        BF_LONG ti[2];
        unsigned char *iv;
        int save=0;

        iv=(unsigned char *)ivec;
        n2l(iv,v0);
        n2l(iv,v1);
        ti[0]=v0;
        ti[1]=v1;
        dp=(char *)d;
        l2n(v0,dp);
        l2n(v1,dp);
        while (l--)
                {
                if (n == 0)
                        {
                        BF_encrypt((BF_LONG *)ti,schedule);
                        dp=(char *)d;
                        t=ti[0]; l2n(t,dp);
                        t=ti[1]; l2n(t,dp);
                        save++;
                        }
                *(out++)= *(in++)^d[n];
                n=(n+1)&0x07;
                }
        if (save)
                {
                v0=ti[0];
                v1=ti[1];
                iv=(unsigned char *)ivec;
                l2n(v0,iv);
                l2n(v1,iv);
                }
        t=v0=v1=ti[0]=ti[1]=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4226 Wed Aug 13 19:52:10 2014
new/usr/src/lib/openssl/libsunw_crypto/bf/bf_skey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bf/bf_skey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <openssl/crypto.h>
```



```
#include <openssl/blowfish.h>
#include "bf_locl.h"
#include "bf_pi.h"

void BF_set_key(BF_KEY *key, int len, const unsigned char *data)
#ifdef OPENSSL_FIPS
        {
        fips_cipher_abort(BLOWFISH);
        private_BF_set_key(key, len, data);
        }
void private_BF_set_key(BF_KEY *key, int len, const unsigned char *data)
#endif
        {
        int i;
        BF_LONG *p,ri,in[2];
        const unsigned char *d,*end;


        memcpy(key,&bf_init,sizeof(BF_KEY));
        p=key->P;

        if (len > ((BF_ROUNDS+2)*4)) len=(BF_ROUNDS+2)*4;

        d=data;
        end= &(data[len]);
        for (i=0; i<(BF_ROUNDS+2); i++)
                {
                ri= *(d++);
                if (d >= end) d=data;

                ri<<=8;
                ri|= *(d++);
                if (d >= end) d=data;

                ri<<=8;
                ri|= *(d++);
                if (d >= end) d=data;

                ri<<=8;
                ri|= *(d++);
                if (d >= end) d=data;

                p[i]^=ri;
                }

        in[0]=0L;
        in[1]=0L;
        for (i=0; i<(BF_ROUNDS+2); i+=2)
                {
                BF_encrypt(in,key);
                p[i  ]=in[0];
                p[i+1]=in[1];
                }

        p=key->S;
        for (i=0; i<4*256; i+=2)
                {
                BF_encrypt(in,key);
                p[i  ]=in[0];
                p[i+1]=in[1];
                }
        }
#endif /* ! codereview */
```

```
**********************************************************
    6167 Wed Aug 13 19:52:10 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/b_dump.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/b_dump.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/*
 * Stolen from tjh's ssl/ssl_trc.c stuff.
 */
```



```
#include <stdio.h>
#include "cryptlib.h"
#include "bio_lcl.h"

#define TRUNCATE
#define DUMP_WIDTH      16
#define DUMP_WIDTH_LESS_INDENT(i) (DUMP_WIDTH-((i-(i>6?6:i)+3)/4))

int BIO_dump_cb(int (*cb)(const void *data, size_t len, void *u),
        void *u, const char *s, int len)
        {
        return BIO_dump_indent_cb(cb, u, s, len, 0);
        }

int BIO_dump_indent_cb(int (*cb)(const void *data, size_t len, void *u),
        void *u, const char *s, int len, int indent)
        {
        int ret=0;
        char buf[288+1],tmp[20],str[128+1];
        int i,j,rows,trc;
        unsigned char ch;
        int dump_width;

        trc=0;

#ifdef TRUNCATE
        for(; (len > 0) && ((s[len-1] == ' ') || (s[len-1] == '\0')); len--)
                trc++;
#endif

        if (indent < 0)
                indent = 0;
        if (indent)
                {
                if (indent > 128) indent=128;
                memset(str,' ',indent);
                }
        str[indent]='\0';

        dump_width=DUMP_WIDTH_LESS_INDENT(indent);
        rows=(len/dump_width);
        if ((rows*dump_width)<len)
                rows++;
        for(i=0;i<rows;i++)
                {
                buf[0]='\0';    /* start with empty string */
                BUF_strlcpy(buf,str,sizeof buf);
                BIO_snprintf(tmp,sizeof tmp,"%04x - ",i*dump_width);
                BUF_strlcat(buf,tmp,sizeof buf);
                for(j=0;j<dump_width;j++)
                        {
                        if (((i*dump_width)+j)>=len)
                                {
                                BUF_strlcat(buf,"   ",sizeof buf);
                                }
                        else
                                {
                                ch=((unsigned char)*(s+i*dump_width+j)) & 0xff;
                                BIO_snprintf(tmp,sizeof tmp,"%02x%c",ch,
                                        j==7?'-':' ');
                                BUF_strlcat(buf,tmp,sizeof buf);
                                }
                        }
                BUF_strlcat(buf,"  ",sizeof buf);
                for(j=0;j<dump_width;j++)
```

```
128                         {
129                         if (((i*dump_width)+j)>=len)
130                                 break;
131                         ch=((unsigned char)*(s+i*dump_width+j)) & 0xff;
132 #ifndef CHARSET_EBCDIC
133                         BIO_snprintf(tmp,sizeof tmp,"%c",
134                                 ((ch>=' ')&&(ch<='~'))?ch:'.');
135 #else
136                         BIO_snprintf(tmp,sizeof tmp,"%c",
137                                 ((ch>=os_toascii[' '])&&(ch<=os_toascii['~']))
138                                 ? os_toebcdic[ch]
139                                 : '.');
140 #endif
141                         BUF_strlcat(buf,tmp,sizeof buf);
142                         }
143                 BUF_strlcat(buf,"\n",sizeof buf);
144                 /* if this is the last call then update the ddt_dump thing so
145                  * that we will move the selection point in the debug window
146                  */
147                 ret+=cb((void *)buf,strlen(buf),u);
148                 }
149 #ifdef TRUNCATE
150         if (trc > 0)
151                 {
152                 BIO_snprintf(buf,sizeof buf,"%s%04x - <SPACES/NULS>\n",str,
153                         len+trc);
154                 ret+=cb((void *)buf,strlen(buf),u);
155                 }
156 #endif
157         return(ret);
158         }

160 #ifndef OPENSSL_NO_FP_API
161 static int write_fp(const void *data, size_t len, void *fp)
162         {
163         return UP_fwrite(data, len, 1, fp);
164         }
165 int BIO_dump_fp(FILE *fp, const char *s, int len)
166         {
167         return BIO_dump_cb(write_fp, fp, s, len);
168         }
169 int BIO_dump_indent_fp(FILE *fp, const char *s, int len, int indent)
170         {
171         return BIO_dump_indent_cb(write_fp, fp, s, len, indent);
172         }
173 #endif

175 static int write_bio(const void *data, size_t len, void *bp)
176         {
177         return BIO_write((BIO *)bp, (const char *)data, len);
178         }
179 int BIO_dump(BIO *bp, const char *s, int len)
180         {
181         return BIO_dump_cb(write_bio, bp, s, len);
182         }
183 int BIO_dump_indent(BIO *bp, const char *s, int len, int indent)
184         {
185         return BIO_dump_indent_cb(write_bio, bp, s, len, indent);
186         }
187 #endif /* ! codereview */
```

```
**********************************************************
   23662 Wed Aug 13 19:52:10 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/b_print.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/b_print.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* disable assert() unless BIO_DEBUG has been defined */
#ifndef BIO_DEBUG
# ifndef NDEBUG
```



```
#  define NDEBUG
# endif
#endif

/*
 * Stolen from tjh's ssl/ssl_trc.c stuff.
 */

#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>
#include <limits.h>
#include "cryptlib.h"
#ifndef NO_SYS_TYPES_H
#include <sys/types.h>
#endif
#include <openssl/bn.h>         /* To get BN_LLONG properly defined */
#include <openssl/bio.h>

#if defined(BN_LLONG) || defined(SIXTY_FOUR_BIT)
# ifndef HAVE_LONG_LONG
#  define HAVE_LONG_LONG 1
# endif
#endif

/***********************************************************************/

/*
 * Copyright Patrick Powell 1995
 * This code is based on code written by Patrick Powell <papowell@astart.com>
 * It may be used for any purpose as long as this notice remains intact
 * on all source code distributions.
 */

/*
 * This code contains numerious changes and enhancements which were
 * made by lots of contributors over the last years to Patrick Powell's
 * original code:
 *
 * o Patrick Powell <papowell@astart.com>      (1995)
 * o Brandon Long <blong@fiction.net>          (1996, for Mutt)
 * o Thomas Roessler <roessler@guug.de>        (1998, for Mutt)
 * o Michael Elkins <me@cs.hmc.edu>            (1998, for Mutt)
 * o Andrew Tridgell <tridge@samba.org>        (1998, for Samba)
 * o Luke Mewburn <lukem@netbsd.org>           (1999, for LukemFTP)
 * o Ralf S. Engelschall <rse@engelschall.com> (1999, for Pth)
 * o ...                                       (for OpenSSL)
 */

#ifdef HAVE_LONG_DOUBLE
#define LDOUBLE long double
#else
#define LDOUBLE double
#endif

#ifdef HAVE_LONG_LONG
# if defined(_WIN32) && !defined(__GNUC__)
# define LLONG __int64
# else
# define LLONG long long
# endif
#else
#define LLONG long
#endif
```

```
128 static void fmtstr     (char **, char **, size_t *, size_t *,
129                          const char *, int, int, int);
130 static void fmtint     (char **, char **, size_t *, size_t *,
131                          LLONG, int, int, int);
132 static void fmtfp      (char **, char **, size_t *, size_t *,
133                          LDOUBLE, int, int, int);
134 static void doapr_outch (char **, char **, size_t *, size_t *, int);
135 static void _dopr(char **sbuffer, char **buffer,
136                   size_t *maxlen, size_t *retlen, int *truncated,
137                   const char *format, va_list args);

139 /* format read states */
140 #define DP_S_DEFAULT    0
141 #define DP_S_FLAGS      1
142 #define DP_S_MIN        2
143 #define DP_S_DOT        3
144 #define DP_S_MAX        4
145 #define DP_S_MOD        5
146 #define DP_S_CONV       6
147 #define DP_S_DONE       7

149 /* format flags - Bits */
150 #define DP_F_MINUS      (1 << 0)
151 #define DP_F_PLUS       (1 << 1)
152 #define DP_F_SPACE      (1 << 2)
153 #define DP_F_NUM        (1 << 3)
154 #define DP_F_ZERO       (1 << 4)
155 #define DP_F_UP         (1 << 5)
156 #define DP_F_UNSIGNED   (1 << 6)

158 /* conversion flags */
159 #define DP_C_SHORT      1
160 #define DP_C_LONG       2
161 #define DP_C_LDOUBLE    3
162 #define DP_C_LLONG      4

164 /* some handy macros */
165 #define char_to_int(p) (p - '0')
166 #define OSSL_MAX(p,q) ((p >= q) ? p : q)

168 static void
169 _dopr(
170     char **sbuffer,
171     char **buffer,
172     size_t *maxlen,
173     size_t *retlen,
174     int *truncated,
175     const char *format,
176     va_list args)
177 {
178     char ch;
179     LLONG value;
180     LDOUBLE fvalue;
181     char *strvalue;
182     int min;
183     int max;
184     int state;
185     int flags;
186     int cflags;
187     size_t currlen;

189     state = DP_S_DEFAULT;
190     flags = currlen = cflags = min = 0;
191     max = -1;
192     ch = *format++;
```



```
194     while (state != DP_S_DONE) {
195         if (ch == '\0' || (buffer == NULL && currlen >= *maxlen))
196             state = DP_S_DONE;

198         switch (state) {
199         case DP_S_DEFAULT:
200             if (ch == '%')
201                 state = DP_S_FLAGS;
202             else
203                 doapr_outch(sbuffer,buffer, &currlen, maxlen, ch);
204             ch = *format++;
205             break;
206         case DP_S_FLAGS:
207             switch (ch) {
208             case '-':
209                 flags |= DP_F_MINUS;
210                 ch = *format++;
211                 break;
212             case '+':
213                 flags |= DP_F_PLUS;
214                 ch = *format++;
215                 break;
216             case ' ':
217                 flags |= DP_F_SPACE;
218                 ch = *format++;
219                 break;
220             case '#':
221                 flags |= DP_F_NUM;
222                 ch = *format++;
223                 break;
224             case '0':
225                 flags |= DP_F_ZERO;
226                 ch = *format++;
227                 break;
228             default:
229                 state = DP_S_MIN;
230                 break;
231             }
232             break;
233         case DP_S_MIN:
234             if (isdigit((unsigned char)ch)) {
235                 min = 10 * min + char_to_int(ch);
236                 ch = *format++;
237             } else if (ch == '*') {
238                 min = va_arg(args, int);
239                 ch = *format++;
240                 state = DP_S_DOT;
241             } else
242                 state = DP_S_DOT;
243             break;
244         case DP_S_DOT:
245             if (ch == '.') {
246                 state = DP_S_MAX;
247                 ch = *format++;
248             } else
249                 state = DP_S_MOD;
250             break;
251         case DP_S_MAX:
252             if (isdigit((unsigned char)ch)) {
253                 if (max < 0)
254                     max = 0;
255                 max = 10 * max + char_to_int(ch);
256                 ch = *format++;
257             } else if (ch == '*') {
258                 max = va_arg(args, int);
259                 ch = *format++;
```

```
260                 state = DP_S_MOD;
261             } else
262                 state = DP_S_MOD;
263             break;
264         case DP_S_MOD:
265             switch (ch) {
266             case 'h':
267                 cflags = DP_C_SHORT;
268                 ch = *format++;
269                 break;
270             case 'l':
271                 if (*format == 'l') {
272                     cflags = DP_C_LLONG;
273                     format++;
274                 } else
275                     cflags = DP_C_LONG;
276                 ch = *format++;
277                 break;
278             case 'q':
279                 cflags = DP_C_LLONG;
280                 ch = *format++;
281                 break;
282             case 'L':
283                 cflags = DP_C_LDOUBLE;
284                 ch = *format++;
285                 break;
286             default:
287                 break;
288             }
289             state = DP_S_CONV;
290             break;
291         case DP_S_CONV:
292             switch (ch) {
293             case 'd':
294             case 'i':
295                 switch (cflags) {
296                 case DP_C_SHORT:
297                     value = (short int)va_arg(args, int);
298                     break;
299                 case DP_C_LONG:
300                     value = va_arg(args, long int);
301                     break;
302                 case DP_C_LLONG:
303                     value = va_arg(args, LLONG);
304                     break;
305                 default:
306                     value = va_arg(args, int);
307                     break;
308                 }
309                 fmtint(sbuffer, buffer, &currlen, maxlen,
310                       value, 10, min, max, flags);
311                 break;
312             case 'X':
313                 flags |= DP_F_UP;
314                 /* FALLTHROUGH */
315             case 'x':
316             case 'o':
317             case 'u':
318                 flags |= DP_F_UNSIGNED;
319                 switch (cflags) {
320                 case DP_C_SHORT:
321                     value = (unsigned short int)va_arg(args, unsigned int);
322                     break;
323                 case DP_C_LONG:
324                     value = (LLONG) va_arg(args,
325                         unsigned long int);
```



```
326                     break;
327                 case DP_C_LLONG:
328                     value = va_arg(args, unsigned LLONG);
329                     break;
330                 default:
331                     value = (LLONG) va_arg(args,
332                         unsigned int);
333                     break;
334                 }
335                 fmtint(sbuffer, buffer, &currlen, maxlen, value,
336                        ch == 'o' ? 8 : (ch == 'u' ? 10 : 16),
337                        min, max, flags);
338                 break;
339             case 'f':
340                 if (cflags == DP_C_LDOUBLE)
341                     fvalue = va_arg(args, LDOUBLE);
342                 else
343                     fvalue = va_arg(args, double);
344                 fmtfp(sbuffer, buffer, &currlen, maxlen,
345                       fvalue, min, max, flags);
346                 break;
347             case 'E':
348                 flags |= DP_F_UP;
349             case 'e':
350                 if (cflags == DP_C_LDOUBLE)
351                     fvalue = va_arg(args, LDOUBLE);
352                 else
353                     fvalue = va_arg(args, double);
354                 break;
355             case 'G':
356                 flags |= DP_F_UP;
357             case 'g':
358                 if (cflags == DP_C_LDOUBLE)
359                     fvalue = va_arg(args, LDOUBLE);
360                 else
361                     fvalue = va_arg(args, double);
362                 break;
363             case 'c':
364                 doapr_outch(sbuffer, buffer, &currlen, maxlen,
365                     va_arg(args, int));
366                 break;
367             case 's':
368                 strvalue = va_arg(args, char *);
369                 if (max < 0) {
370                     if (buffer)
371                         max = INT_MAX;
372                     else
373                         max = *maxlen;
374                 }
375                 fmtstr(sbuffer, buffer, &currlen, maxlen, strvalue,
376                        flags, min, max);
377                 break;
378             case 'p':
379                 value = (long)va_arg(args, void *);
380                 fmtint(sbuffer, buffer, &currlen, maxlen,
381                     value, 16, min, max, flags|DP_F_NUM);
382                 break;
383             case 'n': /* XXX */
384                 if (cflags == DP_C_SHORT) {
385                     short int *num;
386                     num = va_arg(args, short int *);
387                     *num = currlen;
388                 } else if (cflags == DP_C_LONG) { /* XXX */
389                     long int *num;
390                     num = va_arg(args, long int *);
391                     *num = (long int) currlen;
```

```
392                 } else if (cflags == DP_C_LLONG) { /* XXX */
393                     LLONG *num;
394                     num = va_arg(args, LLONG *);
395                     *num = (LLONG) currlen;
396                 } else {
397                     int    *num;
398                     num = va_arg(args, int *);
399                     *num = currlen;
400                 }
401                 break;
402             case '%':
403                 doapr_outch(sbuffer, buffer, &currlen, maxlen, ch);
404                 break;
405             case 'w':
406                 /* not supported yet, treat as next char */
407                 ch = *format++;
408                 break;
409             default:
410                 /* unknown, skip */
411                 break;
412             }
413             ch = *format++;
414             state = DP_S_DEFAULT;
415             flags = cflags = min = 0;
416             max = -1;
417             break;
418         case DP_S_DONE:
419             break;
420         default:
421             break;
422         }
423     }
424     *truncated = (currlen > *maxlen - 1);
425     if (*truncated)
426         currlen = *maxlen - 1;
427     doapr_outch(sbuffer, buffer, &currlen, maxlen, '\0');
428     *retlen = currlen - 1;
429     return;
430 }

432 static void
433 fmtstr(
434     char **sbuffer,
435     char **buffer,
436     size_t *currlen,
437     size_t *maxlen,
438     const char *value,
439     int flags,
440     int min,
441     int max)
442 {
443     int padlen, strln;
444     int cnt = 0;

446     if (value == 0)
447         value = "<NULL>";
448     for (strln = 0; value[strln]; ++strln)
449         ;
450     padlen = min - strln;
451     if (padlen < 0)
452         padlen = 0;
453     if (flags & DP_F_MINUS)
454         padlen = -padlen;

456     while ((padlen > 0) && (cnt < max)) {
457         doapr_outch(sbuffer, buffer, currlen, maxlen, ' ');
```



```
458         --padlen;
459         ++cnt;
460     }
461     while (*value && (cnt < max)) {
462         doapr_outch(sbuffer, buffer, currlen, maxlen, *value++);
463         ++cnt;
464     }
465     while ((padlen < 0) && (cnt < max)) {
466         doapr_outch(sbuffer, buffer, currlen, maxlen, ' ');
467         ++padlen;
468         ++cnt;
469     }
470 }

472 static void
473 fmtint(
474     char **sbuffer,
475     char **buffer,
476     size_t *currlen,
477     size_t *maxlen,
478     LLONG value,
479     int base,
480     int min,
481     int max,
482     int flags)
483 {
484     int signvalue = 0;
485     const char *prefix = "";
486     unsigned LLONG uvalue;
487     char convert[DECIMAL_SIZE(value)+3];
488     int place = 0;
489     int spadlen = 0;
490     int zpadlen = 0;
491     int caps = 0;

493     if (max < 0)
494         max = 0;
495     uvalue = value;
496     if (!(flags & DP_F_UNSIGNED)) {
497         if (value < 0) {
498             signvalue = '-';
499             uvalue = -value;
500         } else if (flags & DP_F_PLUS)
501             signvalue = '+';
502         else if (flags & DP_F_SPACE)
503             signvalue = ' ';
504     }
505     if (flags & DP_F_NUM) {
506         if (base == 8) prefix = "0";
507         if (base == 16) prefix = "0x";
508     }
509     if (flags & DP_F_UP)
510         caps = 1;
511     do {
512         convert[place++] =
513             (caps ? "0123456789ABCDEF" : "0123456789abcdef")
514             [uvalue % (unsigned) base];
515         uvalue = (uvalue / (unsigned) base);
516     } while (uvalue && (place < (int)sizeof(convert)));
517     if (place == sizeof(convert))
518         place--;
519     convert[place] = 0;

521     zpadlen = max - place;
522     spadlen = min - OSSL_MAX(max, place) - (signvalue ? 1 : 0) - strlen(prefix);
523     if (zpadlen < 0)
```

```
524         zpadlen = 0;
525     if (spadlen < 0)
526         spadlen = 0;
527     if (flags & DP_F_ZERO) {
528         zpadlen = OSSL_MAX(zpadlen, spadlen);
529         spadlen = 0;
530     }
531     if (flags & DP_F_MINUS)
532         spadlen = -spadlen;

534     /* spaces */
535     while (spadlen > 0) {
536         doapr_outch(sbuffer, buffer, currlen, maxlen, ' ');
537         --spadlen;
538     }

540     /* sign */
541     if (signvalue)
542         doapr_outch(sbuffer, buffer, currlen, maxlen, signvalue);

544     /* prefix */
545     while (*prefix) {
546         doapr_outch(sbuffer, buffer, currlen, maxlen, *prefix);
547         prefix++;
548     }

550     /* zeros */
551     if (zpadlen > 0) {
552         while (zpadlen > 0) {
553             doapr_outch(sbuffer, buffer, currlen, maxlen, '0');
554             --zpadlen;
555         }
556     }
557     /* digits */
558     while (place > 0)
559         doapr_outch(sbuffer, buffer, currlen, maxlen, convert[--place]);

561     /* left justified spaces */
562     while (spadlen < 0) {
563         doapr_outch(sbuffer, buffer, currlen, maxlen, ' ');
564         ++spadlen;
565     }
566     return;
567 }

569 static LDOUBLE
570 abs_val(LDOUBLE value)
571 {
572     LDOUBLE result = value;
573     if (value < 0)
574         result = -value;
575     return result;
576 }

578 static LDOUBLE
579 pow_10(int in_exp)
580 {
581     LDOUBLE result = 1;
582     while (in_exp) {
583         result *= 10;
584         in_exp--;
585     }
586     return result;
587 }

589 static long
```



```
590 roundv(LDOUBLE value)
591 {
592     long intpart;
593     intpart = (long) value;
594     value = value - intpart;
595     if (value >= 0.5)
596         intpart++;
597     return intpart;
598 }

600 static void
601 fmtfp(
602     char **sbuffer,
603     char **buffer,
604     size_t *currlen,
605     size_t *maxlen,
606     LDOUBLE fvalue,
607     int min,
608     int max,
609     int flags)
610 {
611     int signvalue = 0;
612     LDOUBLE ufvalue;
613     char iconvert[20];
614     char fconvert[20];
615     int iplace = 0;
616     int fplace = 0;
617     int padlen = 0;
618     int zpadlen = 0;
619     int caps = 0;
620     long intpart;
621     long fracpart;
622     long max10;

624     if (max < 0)
625         max = 6;
626     ufvalue = abs_val(fvalue);
627     if (fvalue < 0)
628         signvalue = '-';
629     else if (flags & DP_F_PLUS)
630         signvalue = '+';
631     else if (flags & DP_F_SPACE)
632         signvalue = ' ';

634     intpart = (long)ufvalue;

636     /* sorry, we only support 9 digits past the decimal because of our
637        conversion method */
638     if (max > 9)
639         max = 9;

641     /* we "cheat" by converting the fractional part to integer by
642        multiplying by a factor of 10 */
643     max10 = roundv(pow_10(max));
644     fracpart = roundv(pow_10(max) * (ufvalue - intpart));

646     if (fracpart >= max10) {
647         intpart++;
648         fracpart -= max10;
649     }

651     /* convert integer part */
652     do {
653         iconvert[iplace++] =
654             (caps ? "0123456789ABCDEF"
655               : "0123456789abcdef")[intpart % 10];
```

```
656         intpart = (intpart / 10);
657     } while (intpart && (iplace < (int)sizeof(iconvert)));
658     if (iplace == sizeof iconvert)
659         iplace--;
660     iconvert[iplace] = 0;

662     /* convert fractional part */
663     do {
664         fconvert[fplace++] =
665             (caps ? "0123456789ABCDEF"
666               : "0123456789abcdef")[fracpart % 10];
667         fracpart = (fracpart / 10);
668     } while (fplace < max);
669     if (fplace == sizeof fconvert)
670         fplace--;
671     fconvert[fplace] = 0;

673     /* -1 for decimal point, another -1 if we are printing a sign */
674     padlen = min - iplace - max - 1 - ((signvalue) ? 1 : 0);
675     zpadlen = max - fplace;
676     if (zpadlen < 0)
677         zpadlen = 0;
678     if (padlen < 0)
679         padlen = 0;
680     if (flags & DP_F_MINUS)
681         padlen = -padlen;

683     if ((flags & DP_F_ZERO) && (padlen > 0)) {
684         if (signvalue) {
685             doapr_outch(sbuffer, buffer, currlen, maxlen, signvalue);
686             --padlen;
687             signvalue = 0;
688         }
689         while (padlen > 0) {
690             doapr_outch(sbuffer, buffer, currlen, maxlen, '0');
691             --padlen;
692         }
693     }
694     while (padlen > 0) {
695         doapr_outch(sbuffer, buffer, currlen, maxlen, ' ');
696         --padlen;
697     }
698     if (signvalue)
699         doapr_outch(sbuffer, buffer, currlen, maxlen, signvalue);

701     while (iplace > 0)
702         doapr_outch(sbuffer, buffer, currlen, maxlen, iconvert[--iplace]);

704     /*
705      * Decimal point. This should probably use locale to find the correct
706      * char to print out.
707      */
708     if (max > 0 || (flags & DP_F_NUM)) {
709         doapr_outch(sbuffer, buffer, currlen, maxlen, '.');

711         while (fplace > 0)
712             doapr_outch(sbuffer, buffer, currlen, maxlen, fconvert[--fplace]);
713     }
714     while (zpadlen > 0) {
715         doapr_outch(sbuffer, buffer, currlen, maxlen, '0');
716         --zpadlen;
717     }

719     while (padlen < 0) {
720         doapr_outch(sbuffer, buffer, currlen, maxlen, ' ');
721         ++padlen;
```



```
722     }
723 }

725 static void
726 doapr_outch(
727     char **sbuffer,
728     char **buffer,
729     size_t *currlen,
730     size_t *maxlen,
731     int c)
732 {
733     /* If we haven't at least one buffer, someone has doe a big booboo */
734     assert(*sbuffer != NULL || buffer != NULL);

736     if (buffer) {
737         while (*currlen >= *maxlen) {
738             if (*buffer == NULL) {
739                 if (*maxlen == 0)
740                     *maxlen = 1024;
741                 *buffer = OPENSSL_malloc(*maxlen);
742                 if (*currlen > 0) {
743                     assert(*sbuffer != NULL);
744                     memcpy(*buffer, *sbuffer, *currlen);
745                 }
746                 *sbuffer = NULL;
747             } else {
748                 *maxlen += 1024;
749                 *buffer = OPENSSL_realloc(*buffer, *maxlen);
750             }
751         }
752         /* What to do if *buffer is NULL? */
753         assert(*sbuffer != NULL || *buffer != NULL);
754     }

756     if (*currlen < *maxlen) {
757         if (*sbuffer)
758             (*sbuffer)[(*currlen)++] = (char)c;
759         else
760             (*buffer)[(*currlen)++] = (char)c;
761     }

763     return;
764 }

766 /***************************************************************************/

768 int BIO_printf (BIO *bio, const char *format, ...)
769         {
770         va_list args;
771         int ret;

773         va_start(args, format);

775         ret = BIO_vprintf(bio, format, args);

777         va_end(args);
778         return(ret);
779         }

781 int BIO_vprintf (BIO *bio, const char *format, va_list args)
782         {
783         int ret;
784         size_t retlen;
785         char hugebuf[1024*2];   /* Was previously 10k, which is unreasonable
786                                    in small-stack environments, like threads
787                                    or DOS programs. */
```

```
788         char *hugebufp = hugebuf;
789         size_t hugebufsize = sizeof(hugebuf);
790         char *dynbuf = NULL;
791         int ignored;

793         dynbuf = NULL;
794         CRYPTO_push_info("doapr()");
795         _dopr(&hugebufp, &dynbuf, &hugebufsize,
796                 &retlen, &ignored, format, args);
797         if (dynbuf)
798                 {
799                 ret=BIO_write(bio, dynbuf, (int)retlen);
800                 OPENSSL_free(dynbuf);
801                 }
802         else
803                 {
804                 ret=BIO_write(bio, hugebuf, (int)retlen);
805                 }
806         CRYPTO_pop_info();
807         return(ret);
808         }

810 /* As snprintf is not available everywhere, we provide our own implementation.
811  * This function has nothing to do with BIOs, but it's closely related
812  * to BIO_printf, and we need *some* name prefix ...
813  * (XXX  the function should be renamed, but to what?) */
814 int BIO_snprintf(char *buf, size_t n, const char *format, ...)
815         {
816         va_list args;
817         int ret;

819         va_start(args, format);

821         ret = BIO_vsnprintf(buf, n, format, args);

823         va_end(args);
824         return(ret);
825         }

827 int BIO_vsnprintf(char *buf, size_t n, const char *format, va_list args)
828         {
829         size_t retlen;
830         int truncated;

832         _dopr(&buf, NULL, &n, &retlen, &truncated, format, args);

834         if (truncated)
835                 /* In case of truncation, return -1 like traditional snprintf.
836                  * (Current drafts for ISO/IEC 9899 say snprintf should return
837                  * the number of characters that would have been written,
838                  * had the buffer been large enough.) */
839                 return -1;
840         else
841                 return (retlen <= INT_MAX) ? (int)retlen : -1;
842         }
843 #endif /* ! codereview */
```

```
**********************************************************
   24057 Wed Aug 13 19:52:10 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/b_sock.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/b_sock.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
```



```
#define USE_SOCKETS
#include "cryptlib.h"
#include <openssl/bio.h>
#if defined(OPENSSL_SYS_NETWARE) && defined(NETWARE_BSDSOCK)
#include <netdb.h>
#if defined(NETWARE_CLIB)
#include <sys/ioctl.h>
NETDB_DEFINE_CONTEXT
#endif
#endif

#ifndef OPENSSL_NO_SOCK

#include <openssl/dso.h>

#define SOCKET_PROTOCOL IPPROTO_TCP

#ifdef SO_MAXCONN
#define MAX_LISTEN  SO_MAXCONN
#elif defined(SOMAXCONN)
#define MAX_LISTEN  SOMAXCONN
#else
#define MAX_LISTEN  32
#endif

#if defined(OPENSSL_SYS_WINDOWS) || (defined(OPENSSL_SYS_NETWARE) && !defined(NE
static int wsa_init_done=0;
#endif

/*
 * WSAAPI specifier is required to make indirect calls to run-time
 * linked WinSock 2 functions used in this module, to be specific
 * [get|free]addrinfo and getnameinfo. This is because WinSock uses
 * uses non-C calling convention, __stdcall vs. __cdecl, on x86
 * Windows. On non-WinSock platforms WSAAPI needs to be void.
 */
#ifndef WSAAPI
#define WSAAPI
#endif

#if 0
static unsigned long BIO_ghbn_hits=0L;
static unsigned long BIO_ghbn_miss=0L;

#define GHBN_NUM        4
static struct ghbn_cache_st
        {
        char name[129];
        struct hostent *ent;
        unsigned long order;
        } ghbn_cache[GHBN_NUM];
#endif

static int get_ip(const char *str,unsigned char *ip);
#if 0
static void ghbn_free(struct hostent *a);
static struct hostent *ghbn_dup(struct hostent *a);
#endif
int BIO_get_host_ip(const char *str, unsigned char *ip)
        {
        int i;
        int err = 1;
        int locked = 0;
        struct hostent *he;

        i=get_ip(str,ip);
```

```
128         if (i < 0)
129                 {
130                 BIOerr(BIO_F_BIO_GET_HOST_IP,BIO_R_INVALID_IP_ADDRESS);
131                 goto err;
132                 }

134         /* At this point, we have something that is most probably correct
135            in some way, so let's init the socket. */
136         if (BIO_sock_init() != 1)
137                 return 0; /* don't generate another error code here */

139         /* If the string actually contained an IP address, we need not do
140            anything more */
141         if (i > 0) return(1);

143         /* do a gethostbyname */
144         CRYPTO_w_lock(CRYPTO_LOCK_GETHOSTBYNAME);
145         locked = 1;
146         he=BIO_gethostbyname(str);
147         if (he == NULL)
148                 {
149                 BIOerr(BIO_F_BIO_GET_HOST_IP,BIO_R_BAD_HOSTNAME_LOOKUP);
150                 goto err;
151                 }

153         /* cast to short because of win16 winsock definition */
154         if ((short)he->h_addrtype != AF_INET)
155                 {
156                 BIOerr(BIO_F_BIO_GET_HOST_IP,BIO_R_GETHOSTBYNAME_ADDR_IS_NOT_AF_
157                 goto err;
158                 }
159         for (i=0; i<4; i++)
160                 ip[i]=he->h_addr_list[0][i];
161         err = 0;

163  err:
164         if (locked)
165                 CRYPTO_w_unlock(CRYPTO_LOCK_GETHOSTBYNAME);
166         if (err)
167                 {
168                 ERR_add_error_data(2,"host=",str);
169                 return 0;
170                 }
171         else
172                 return 1;
173         }

175 int BIO_get_port(const char *str, unsigned short *port_ptr)
176         {
177         int i;
178         struct servent *s;

180         if (str == NULL)
181                 {
182                 BIOerr(BIO_F_BIO_GET_PORT,BIO_R_NO_PORT_DEFINED);
183                 return(0);
184                 }
185         i=atoi(str);
186         if (i != 0)
187                 *port_ptr=(unsigned short)i;
188         else
189                 {
190                 CRYPTO_w_lock(CRYPTO_LOCK_GETSERVBYNAME);
191                 /* Note: under VMS with SOCKETSHR, it seems like the first
192                  * parameter is 'char *', instead of 'const char *'
193                  */
```



```
194 #ifndef CONST_STRICT
195                 s=getservbyname((char *)str,"tcp");
196 #else
197                 s=getservbyname(str,"tcp");
198 #endif
199                 if(s != NULL)
200                         *port_ptr=ntohs((unsigned short)s->s_port);
201                 CRYPTO_w_unlock(CRYPTO_LOCK_GETSERVBYNAME);
202                 if(s == NULL)
203                         {
204                         if (strcmp(str,"http") == 0)
205                                 *port_ptr=80;
206                         else if (strcmp(str,"telnet") == 0)
207                                 *port_ptr=23;
208                         else if (strcmp(str,"socks") == 0)
209                                 *port_ptr=1080;
210                         else if (strcmp(str,"https") == 0)
211                                 *port_ptr=443;
212                         else if (strcmp(str,"ssl") == 0)
213                                 *port_ptr=443;
214                         else if (strcmp(str,"ftp") == 0)
215                                 *port_ptr=21;
216                         else if (strcmp(str,"gopher") == 0)
217                                 *port_ptr=70;
218 #if 0
219                         else if (strcmp(str,"wais") == 0)
220                                 *port_ptr=21;
221 #endif
222                         else
223                                 {
224                                 SYSerr(SYS_F_GETSERVBYNAME,get_last_socket_error
225                                 ERR_add_error_data(3,"service='",str,"'");
226                                 return(0);
227                                 }
228                         }
229                 }
230         return(1);
231         }

233 int BIO_sock_error(int sock)
234         {
235         int j,i;
236         int size;

238 #if defined(OPENSSL_SYS_BEOS_R5)
239         return 0;
240 #endif

242         size=sizeof(int);
243         /* Note: under Windows the third parameter is of type (char *)
244          * whereas under other systems it is (void *) if you don't have
245          * a cast it will choke the compiler: if you do have a cast then
246          * you can either go for (char *) or (void *).
247          */
248         i=getsockopt(sock,SOL_SOCKET,SO_ERROR,(void *)&j,(void *)&size);
249         if (i < 0)
250                 return(1);
251         else
252                 return(j);
253         }

255 #if 0
256 long BIO_ghbn_ctrl(int cmd, int iarg, char *parg)
257         {
258         int i;
259         char **p;
```

```
261         switch (cmd)
262                 {
263         case BIO_GHBN_CTRL_HITS:
264                 return(BIO_ghbn_hits);
265                 /* break; */
266         case BIO_GHBN_CTRL_MISSES:
267                 return(BIO_ghbn_miss);
268                 /* break; */
269         case BIO_GHBN_CTRL_CACHE_SIZE:
270                 return(GHBN_NUM);
271                 /* break; */
272         case BIO_GHBN_CTRL_GET_ENTRY:
273                 if ((iarg >= 0) && (iarg <GHBN_NUM) &&
274                         (ghbn_cache[iarg].order > 0))
275                         {
276                         p=(char **)parg;
277                         if (p == NULL) return(0);
278                         *p=ghbn_cache[iarg].name;
279                         ghbn_cache[iarg].name[128]='\0';
280                         return(1);
281                         }
282                 return(0);
283                 /* break; */
284         case BIO_GHBN_CTRL_FLUSH:
285                 for (i=0; i<GHBN_NUM; i++)
286                         ghbn_cache[i].order=0;
287                 break;
288         default:
289                 return(0);
290                 }
291         return(1);
292         }
293 #endif

295 #if 0
296 static struct hostent *ghbn_dup(struct hostent *a)
297         {
298         struct hostent *ret;
299         int i,j;

301         MemCheck_off();
302         ret=(struct hostent *)OPENSSL_malloc(sizeof(struct hostent));
303         if (ret == NULL) return(NULL);
304         memset(ret,0,sizeof(struct hostent));

306         for (i=0; a->h_aliases[i] != NULL; i++)
307                 ;
308         i++;
309         ret->h_aliases = (char **)OPENSSL_malloc(i*sizeof(char *));
310         if (ret->h_aliases == NULL)
311                 goto err;
312         memset(ret->h_aliases, 0, i*sizeof(char *));

314         for (i=0; a->h_addr_list[i] != NULL; i++)
315                 ;
316         i++;
317         ret->h_addr_list=(char **)OPENSSL_malloc(i*sizeof(char *));
318         if (ret->h_addr_list == NULL)
319                 goto err;
320         memset(ret->h_addr_list, 0, i*sizeof(char *));

322         j=strlen(a->h_name)+1;
323         if ((ret->h_name=OPENSSL_malloc(j)) == NULL) goto err;
324         memcpy((char *)ret->h_name,a->h_name,j);
325         for (i=0; a->h_aliases[i] != NULL; i++)
```



```
326                 {
327                 j=strlen(a->h_aliases[i])+1;
328                 if ((ret->h_aliases[i]=OPENSSL_malloc(j)) == NULL) goto err;
329                 memcpy(ret->h_aliases[i],a->h_aliases[i],j);
330                 }
331         ret->h_length=a->h_length;
332         ret->h_addrtype=a->h_addrtype;
333         for (i=0; a->h_addr_list[i] != NULL; i++)
334                 {
335                 if ((ret->h_addr_list[i]=OPENSSL_malloc(a->h_length)) == NULL)
336                         goto err;
337                 memcpy(ret->h_addr_list[i],a->h_addr_list[i],a->h_length);
338                 }
339         if (0)
340                 {
341 err:
342                 if (ret != NULL)
343                         ghbn_free(ret);
344                 ret=NULL;
345                 }
346         MemCheck_on();
347         return(ret);
348         }

350 static void ghbn_free(struct hostent *a)
351         {
352         int i;

354         if(a == NULL)
355             return;

357         if (a->h_aliases != NULL)
358                 {
359                 for (i=0; a->h_aliases[i] != NULL; i++)
360                         OPENSSL_free(a->h_aliases[i]);
361                 OPENSSL_free(a->h_aliases);
362                 }
363         if (a->h_addr_list != NULL)
364                 {
365                 for (i=0; a->h_addr_list[i] != NULL; i++)
366                         OPENSSL_free(a->h_addr_list[i]);
367                 OPENSSL_free(a->h_addr_list);
368                 }
369         if (a->h_name != NULL) OPENSSL_free(a->h_name);
370         OPENSSL_free(a);
371         }

373 #endif

375 struct hostent *BIO_gethostbyname(const char *name)
376         {
377 #if 1
378         /* Caching gethostbyname() results forever is wrong,
379          * so we have to let the true gethostbyname() worry about this */
380 #if (defined(NETWARE_BSDSOCK) && !defined(__NOVELL_LIBC__))
381         return gethostbyname((char*)name);
382 #else
383         return gethostbyname(name);
384 #endif
385 #else
386         struct hostent *ret;
387         int i,lowi=0,j;
388         unsigned long low= (unsigned long)-1;


391 #  if 0
```

```
392         /* It doesn't make sense to use locking here: The function interface
393          * is not thread-safe, because threads can never be sure when
394          * some other thread destroys the data they were given a pointer to.
395          */
396         CRYPTO_w_lock(CRYPTO_LOCK_GETHOSTBYNAME);
397 #  endif
398         j=strlen(name);
399         if (j < 128)
400                 {
401                 for (i=0; i<GHBN_NUM; i++)
402                         {
403                         if (low > ghbn_cache[i].order)
404                                 {
405                                 low=ghbn_cache[i].order;
406                                 lowi=i;
407                                 }
408                         if (ghbn_cache[i].order > 0)
409                                 {
410                                 if (strncmp(name,ghbn_cache[i].name,128) == 0)
411                                         break;
412                                 }
413                         }
414                 }
415         else
416                 i=GHBN_NUM;

418         if (i == GHBN_NUM) /* no hit*/
419                 {
420                 BIO_ghbn_miss++;
421                 /* Note: under VMS with SOCKETSHR, it seems like the first
422                  * parameter is 'char *', instead of 'const char *'
423                  */
424 #  ifndef CONST_STRICT
425                 ret=gethostbyname((char *)name);
426 #  else
427                 ret=gethostbyname(name);
428 #  endif

430                 if (ret == NULL)
431                         goto end;
432                 if (j > 128) /* too big to cache */
433                         {
434 #  if 0
435                         /* If we were trying to make this function thread-safe (
436                          * is bound to fail), we'd have to give up in this case
437                          * (or allocate more memory). */
438                         ret = NULL;
439 #  endif
440                         goto end;
441                         }

443                 /* else add to cache */
444                 if (ghbn_cache[lowi].ent != NULL)
445                         ghbn_free(ghbn_cache[lowi].ent); /* XXX not thread-safe
446                 ghbn_cache[lowi].name[0] = '\0';

448                 if((ret=ghbn_cache[lowi].ent=ghbn_dup(ret)) == NULL)
449                         {
450                         BIOerr(BIO_F_BIO_GETHOSTBYNAME,ERR_R_MALLOC_FAILURE);
451                         goto end;
452                         }
453                 strncpy(ghbn_cache[lowi].name,name,128);
454                 ghbn_cache[lowi].order=BIO_ghbn_miss+BIO_ghbn_hits;
455                 }
456         else
457                 {
```



```
458                 BIO_ghbn_hits++;
459                 ret= ghbn_cache[i].ent;
460                 ghbn_cache[i].order=BIO_ghbn_miss+BIO_ghbn_hits;
461                 }
462 end:
463 #  if 0
464         CRYPTO_w_unlock(CRYPTO_LOCK_GETHOSTBYNAME);
465 #  endif
466         return(ret);
467 #endif
468         }


471 int BIO_sock_init(void)
472         {
473 #ifdef OPENSSL_SYS_WINDOWS
474         static struct WSAData wsa_state;

476         if (!wsa_init_done)
477                 {
478                 int err;

480                 wsa_init_done=1;
481                 memset(&wsa_state,0,sizeof(wsa_state));
482                 /* Not making wsa_state available to the rest of the
483                  * code is formally wrong. But the structures we use
484                  * are [beleived to be] invariable among Winsock DLLs,
485                  * while API availability is [expected to be] probed
486                  * at run-time with DSO_global_lookup. */
487                 if (WSAStartup(0x0202,&wsa_state)!=0)
488                         {
489                         err=WSAGetLastError();
490                         SYSerr(SYS_F_WSASTARTUP,err);
491                         BIOerr(BIO_F_BIO_SOCK_INIT,BIO_R_WSASTARTUP);
492                         return(-1);
493                         }
494                 }
495 #endif /* OPENSSL_SYS_WINDOWS */
496 #ifdef WATT32
497         extern int _watt_do_exit;
498         _watt_do_exit = 0;    /* don't make sock_init() call exit() */
499         if (sock_init())
500                 return (-1);
501 #endif

503 #if defined(OPENSSL_SYS_NETWARE) && !defined(NETWARE_BSDSOCK)
504     WORD wVerReq;
505     WSADATA wsaData;
506     int err;

508     if (!wsa_init_done)
509     {
510         wsa_init_done=1;
511         wVerReq = MAKEWORD( 2, 0 );
512         err = WSAStartup(wVerReq,&wsaData);
513         if (err != 0)
514         {
515             SYSerr(SYS_F_WSASTARTUP,err);
516             BIOerr(BIO_F_BIO_SOCK_INIT,BIO_R_WSASTARTUP);
517             return(-1);
518                         }
519                 }
520 #endif

522         return(1);
523         }
```

```
525 void BIO_sock_cleanup(void)
526         {
527 #ifdef OPENSSL_SYS_WINDOWS
528         if (wsa_init_done)
529                 {
530                 wsa_init_done=0;
531 #if 0           /* this call is claimed to be non-present in Winsock2 */
532                 WSACancelBlockingCall();
533 #endif
534                 WSACleanup();
535                 }
536 #elif defined(OPENSSL_SYS_NETWARE) && !defined(NETWARE_BSDSOCK)
537    if (wsa_init_done)
538         {
539         wsa_init_done=0;
540         WSACleanup();
541                 }
542 #endif
543         }

545 #if !defined(OPENSSL_SYS_VMS) || __VMS_VER >= 70000000

547 int BIO_socket_ioctl(int fd, long type, void *arg)
548         {
549         int i;

551 #ifdef __DJGPP__
552         i=ioctlsocket(fd,type,(char *)arg);
553 #else
554 # if defined(OPENSSL_SYS_VMS)
555         /* 2011-02-18 SMS.
556          * VMS ioctl() can't tolerate a 64-bit "void *arg", but we
557          * observe that all the consumers pass in an "unsigned long *",
558          * so we arrange a local copy with a short pointer, and use
559          * that, instead.
560          */
561 #  if __INITIAL_POINTER_SIZE == 64
562 #   define ARG arg_32p
563 #   pragma pointer_size save
564 #   pragma pointer_size 32
565         unsigned long arg_32;
566         unsigned long *arg_32p;
567 #   pragma pointer_size restore
568         arg_32p = &arg_32;
569         arg_32 = *((unsigned long *) arg);
570 #  else /* __INITIAL_POINTER_SIZE == 64 */
571 #   define ARG arg
572 #  endif /* __INITIAL_POINTER_SIZE == 64 [else] */
573 # else /* defined(OPENSSL_SYS_VMS) */
574 #  define ARG arg
575 # endif /* defined(OPENSSL_SYS_VMS) [else] */

577         i=ioctlsocket(fd,type,ARG);
578 #endif /* __DJGPP__ */
579         if (i < 0)
580                 SYSerr(SYS_F_IOCTLSOCKET,get_last_socket_error());
581         return(i);
582         }
583 #endif /* __VMS_VER */

585 /* The reason I have implemented this instead of using sscanf is because
586  * Visual C 1.52c gives an unresolved external when linking a DLL :-( */
587 static int get_ip(const char *str, unsigned char ip[4])
588         {
589         unsigned int tmp[4];
```



```
590         int num=0,c,ok=0;

592         tmp[0]=tmp[1]=tmp[2]=tmp[3]=0;

594         for (;;)
595                 {
596                 c= *(str++);
597                 if ((c >= '0') && (c <= '9'))
598                         {
599                         ok=1;
600                         tmp[num]=tmp[num]*10+c-'0';
601                         if (tmp[num] > 255) return(0);
602                         }
603                 else if (c == '.')
604                         {
605                         if (!ok) return(-1);
606                         if (num == 3) return(0);
607                         num++;
608                         ok=0;
609                         }
610                 else if (c == '\0' && (num == 3) && ok)
611                         break;
612                 else
613                         return(0);
614                 }
615         ip[0]=tmp[0];
616         ip[1]=tmp[1];
617         ip[2]=tmp[2];
618         ip[3]=tmp[3];
619         return(1);
620         }

622 int BIO_get_accept_socket(char *host, int bind_mode)
623         {
624         int ret=0;
625         union {
626                 struct sockaddr sa;
627                 struct sockaddr_in sa_in;
628 #if OPENSSL_USE_IPV6
629                 struct sockaddr_in6 sa_in6;
630 #endif
631         } server,client;
632         int s=INVALID_SOCKET,cs,addrlen;
633         unsigned char ip[4];
634         unsigned short port;
635         char *str=NULL,*e;
636         char *h,*p;
637         unsigned long l;
638         int err_num;

640         if (BIO_sock_init() != 1) return(INVALID_SOCKET);

642         if ((str=BUF_strdup(host)) == NULL) return(INVALID_SOCKET);

644         h=p=NULL;
645         h=str;
646         for (e=str; *e; e++)
647                 {
648                 if (*e == ':')
649                         {
650                         p=e;
651                         }
652                 else if (*e == '/')
653                         {
654                         *e='\0';
655                         break;
```

```
656                         }
657                 }
658         if (p)  *p++='\0';      /* points at last ':', '::port' is special [see
659         else    p=h,h=NULL;

661 #ifdef EAI_FAMILY
662         do {
663         static union {  void *p;
664                         int (WSAAPI *f)(const char *,const char *,
665                                 const struct addrinfo *,
666                                 struct addrinfo **);
667                         } p_getaddrinfo = {NULL};
668         static union {  void *p;
669                         void (WSAAPI *f)(struct addrinfo *);
670                         } p_freeaddrinfo = {NULL};
671         struct addrinfo *res,hint;

673         if (p_getaddrinfo.p==NULL)
674                 {
675                 if ((p_getaddrinfo.p=DSO_global_lookup("getaddrinfo"))==NULL ||
676                     (p_freeaddrinfo.p=DSO_global_lookup("freeaddrinfo"))==NULL)
677                         p_getaddrinfo.p=(void*)-1;
678                 }
679         if (p_getaddrinfo.p==(void *)-1) break;

681         /* '::port' enforces IPv6 wildcard listener. Some OSes,
682          * e.g. Solaris, default to IPv6 without any hint. Also
683          * note that commonly IPv6 wildchard socket can service
684          * IPv4 connections just as well...  */
685         memset(&hint,0,sizeof(hint));
686         hint.ai_flags = AI_PASSIVE;
687         if (h)
688                 {
689                 if (strchr(h,':'))
690                         {
691                         if (h[1]=='\0') h=NULL;
692 #if OPENSSL_USE_IPV6
693                         hint.ai_family = AF_INET6;
694 #else
695                         h=NULL;
696 #endif
697                         }
698                 else if (h[0]=='*' && h[1]=='\0')
699                         {
700                         hint.ai_family = AF_INET;
701                         h=NULL;
702                         }
703                 }

705         if ((*p_getaddrinfo.f)(h,p,&hint,&res)) break;

707         addrlen = res->ai_addrlen<=sizeof(server) ?
708                         res->ai_addrlen :
709                         sizeof(server);
710         memcpy(&server, res->ai_addr, addrlen);

712         (*p_freeaddrinfo.f)(res);
713         goto again;
714         } while (0);
715 #endif

717         if (!BIO_get_port(p,&port)) goto err;

719         memset((char *)&server,0,sizeof(server));
720         server.sa_in.sin_family=AF_INET;
721         server.sa_in.sin_port=htons(port);
```



```
722         addrlen = sizeof(server.sa_in);

724         if (h == NULL || strcmp(h,"*") == 0)
725                 server.sa_in.sin_addr.s_addr=INADDR_ANY;
726         else
727                 {
728                 if (!BIO_get_host_ip(h,&(ip[0]))) goto err;
729                 l=(unsigned long)
730                         ((unsigned long)ip[0]<<24L)|
731                         ((unsigned long)ip[1]<<16L)|
732                         ((unsigned long)ip[2]<< 8L)|
733                         ((unsigned long)ip[3]);
734                 server.sa_in.sin_addr.s_addr=htonl(l);
735                 }

737 again:
738         s=socket(server.sa.sa_family,SOCK_STREAM,SOCKET_PROTOCOL);
739         if (s == INVALID_SOCKET)
740                 {
741                 SYSerr(SYS_F_SOCKET,get_last_socket_error());
742                 ERR_add_error_data(3,"port='",host,"'");
743                 BIOerr(BIO_F_BIO_GET_ACCEPT_SOCKET,BIO_R_UNABLE_TO_CREATE_SOCKET
744                 goto err;
745                 }

747 #ifdef SO_REUSEADDR
748         if (bind_mode == BIO_BIND_REUSEADDR)
749                 {
750                 int i=1;

752                 ret=setsockopt(s,SOL_SOCKET,SO_REUSEADDR,(char *)&i,sizeof(i));
753                 bind_mode=BIO_BIND_NORMAL;
754                 }
755 #endif
756         if (bind(s,&server.sa,addrlen) == -1)
757                 {
758 #ifdef SO_REUSEADDR
759                 err_num=get_last_socket_error();
760                 if ((bind_mode == BIO_BIND_REUSEADDR_IF_UNUSED) &&
761 #ifdef OPENSSL_SYS_WINDOWS
762                         /* Some versions of Windows define EADDRINUSE to
763                          * a dummy value.
764                          */
765                         (err_num == WSAEADDRINUSE))
766 #else
767                         (err_num == EADDRINUSE))
768 #endif
769                         {
770                         client = server;
771                         if (h == NULL || strcmp(h,"*") == 0)
772                                 {
773 #if OPENSSL_USE_IPV6
774                                 if (client.sa.sa_family == AF_INET6)
775                                         {
776                                         memset(&client.sa_in6.sin6_addr,0,sizeof
777                                         client.sa_in6.sin6_addr.s6_addr[15]=1;
778                                         }
779                                 else
780 #endif
781                                 if (client.sa.sa_family == AF_INET)
782                                         {
783                                         client.sa_in.sin_addr.s_addr=htonl(0x7F0
784                                         }
785                                 else    goto err;
786                                 }
787                         cs=socket(client.sa.sa_family,SOCK_STREAM,SOCKET_PROTOCO
```

```
788                         if (cs != INVALID_SOCKET)
789                                 {
790                                 int ii;
791                                 ii=connect(cs,&client.sa,addrlen);
792                                 closesocket(cs);
793                                 if (ii == INVALID_SOCKET)
794                                         {
795                                         bind_mode=BIO_BIND_REUSEADDR;
796                                         closesocket(s);
797                                         goto again;
798                                         }
799                                 /* else error */
800                                 }
801                         /* else error */
802                         }
803 #endif
804                 SYSerr(SYS_F_BIND,err_num);
805                 ERR_add_error_data(3,"port='",host,"'");
806                 BIOerr(BIO_F_BIO_GET_ACCEPT_SOCKET,BIO_R_UNABLE_TO_BIND_SOCKET);
807                 goto err;
808                 }
809         if (listen(s,MAX_LISTEN) == -1)
810                 {
811                 SYSerr(SYS_F_BIND,get_last_socket_error());
812                 ERR_add_error_data(3,"port='",host,"'");
813                 BIOerr(BIO_F_BIO_GET_ACCEPT_SOCKET,BIO_R_UNABLE_TO_LISTEN_SOCKET
814                 goto err;
815                 }
816         ret=1;
817 err:
818         if (str != NULL) OPENSSL_free(str);
819         if ((ret == 0) && (s != INVALID_SOCKET))
820                 {
821                 closesocket(s);
822                 s= INVALID_SOCKET;
823                 }
824         return(s);
825         }

827 int BIO_accept(int sock, char **addr)
828         {
829         int ret=INVALID_SOCKET;
830         unsigned long l;
831         unsigned short port;
832         char *p;

834         struct {
835         /*
836          * As for following union. Trouble is that there are platforms
837          * that have socklen_t and there are platforms that don't, on
838          * some platforms socklen_t is int and on some size_t. So what
839          * one can do? One can cook #ifdef spaghetti, which is nothing
840          * but masochistic. Or one can do union between int and size_t.
841          * One naturally does it primarily for 64-bit platforms where
842          * sizeof(int) != sizeof(size_t). But would it work? Note that
843          * if size_t member is initialized to 0, then later int member
844          * assignment naturally does the job on little-endian platforms
845          * regardless accept's expectations! What about big-endians?
846          * If accept expects int*, then it works, and if size_t*, then
847          * length value would appear as unreasonably large. But this
848          * won't prevent it from filling in the address structure. The
849          * trouble of course would be if accept returns more data than
850          * actual buffer can accomodate and overwrite stack... That's
851          * where early OPENSSL_assert comes into picture. Besides, the
852          * only 64-bit big-endian platform found so far that expects
853          * size_t* is HP-UX, where stack grows towards higher address.
```



```
854          * <appro>
855          */
856         union { size_t s; int i; } len;
857         union {
858                 struct sockaddr sa;
859                 struct sockaddr_in sa_in;
860 #if OPENSSL_USE_IPV6
861                 struct sockaddr_in6 sa_in6;
862 #endif
863                 } from;
864         } sa;

866         sa.len.s=0;
867         sa.len.i=sizeof(sa.from);
868         memset(&sa.from,0,sizeof(sa.from));
869         ret=accept(sock,&sa.from.sa,(void *)&sa.len);
870         if (sizeof(sa.len.i)!=sizeof(sa.len.s) && sa.len.i==0)
871                 {
872                 OPENSSL_assert(sa.len.s<=sizeof(sa.from));
873                 sa.len.i = (int)sa.len.s;
874                 /* use sa.len.i from this point */
875                 }
876         if (ret == INVALID_SOCKET)
877                 {
878                 if(BIO_sock_should_retry(ret)) return -2;
879                 SYSerr(SYS_F_ACCEPT,get_last_socket_error());
880                 BIOerr(BIO_F_BIO_ACCEPT,BIO_R_ACCEPT_ERROR);
881                 goto end;
882                 }

884         if (addr == NULL) goto end;

886 #ifdef EAI_FAMILY
887         do {
888         char   h[NI_MAXHOST],s[NI_MAXSERV];
889         size_t nl;
890         static union {  void *p;
891                         int (WSAAPI *f)(const struct sockaddr *,size_t/*socklen_
892                                         char *,size_t,char *,size_t,int);
893                         } p_getnameinfo = {NULL};
894                         /* 2nd argument to getnameinfo is specified to
895                          * be socklen_t. Unfortunately there is a number
896                          * of environments where socklen_t is not defined.
897                          * As it's passed by value, it's safe to pass it
898                          * as size_t... <appro> */

900         if (p_getnameinfo.p==NULL)
901                 {
902                 if ((p_getnameinfo.p=DSO_global_lookup("getnameinfo"))==NULL)
903                         p_getnameinfo.p=(void*)-1;
904                 }
905         if (p_getnameinfo.p==(void *)-1) break;

907         if ((*p_getnameinfo.f)(&sa.from.sa,sa.len.i,h,sizeof(h),s,sizeof(s),
908             NI_NUMERICHOST|NI_NUMERICSERV)) break;
909         nl = strlen(h)+strlen(s)+2;
910         p = *addr;
911         if (p)  { *p = '\0'; p = OPENSSL_realloc(p,nl); }
912         else    { p = OPENSSL_malloc(nl);               }
913         if (p==NULL)
914                 {
915                 BIOerr(BIO_F_BIO_ACCEPT,ERR_R_MALLOC_FAILURE);
916                 goto end;
917                 }
918         *addr = p;
919         BIO_snprintf(*addr,nl,"%s:%s",h,s);
```

```
920         goto end;
921         } while(0);
922 #endif
923         if (sa.from.sa.sa_family != AF_INET) goto end;
924         l=ntohl(sa.from.sa_in.sin_addr.s_addr);
925         port=ntohs(sa.from.sa_in.sin_port);
926         if (*addr == NULL)
927                 {
928                 if ((p=OPENSSL_malloc(24)) == NULL)
929                         {
930                         BIOerr(BIO_F_BIO_ACCEPT,ERR_R_MALLOC_FAILURE);
931                         goto end;
932                         }
933                 *addr=p;
934                 }
935         BIO_snprintf(*addr,24,"%d.%d.%d.%d:%d",
936                      (unsigned char)(l>>24L)&0xff,
937                      (unsigned char)(l>>16L)&0xff,
938                      (unsigned char)(l>> 8L)&0xff,
939                      (unsigned char)(l     )&0xff,
940                      port);
941 end:
942         return(ret);
943         }

945 int BIO_set_tcp_ndelay(int s, int on)
946         {
947         int ret=0;
948 #if defined(TCP_NODELAY) && (defined(IPPROTO_TCP) || defined(SOL_TCP))
949         int opt;

951 #ifdef SOL_TCP
952         opt=SOL_TCP;
953 #else
954 #ifdef IPPROTO_TCP
955         opt=IPPROTO_TCP;
956 #endif
957 #endif

959         ret=setsockopt(s,opt,TCP_NODELAY,(char *)&on,sizeof(on));
960 #endif
961         return(ret == 0);
962         }

964 int BIO_socket_nbio(int s, int mode)
965         {
966         int ret= -1;
967         int l;

969         l=mode;
970 #ifdef FIONBIO
971         ret=BIO_socket_ioctl(s,FIONBIO,&l);
972 #endif
973         return(ret == 0);
974         }
975 #endif
976 #endif /* ! codereview */
```

```
**********************************************************
   12411 Wed Aug 13 19:52:10 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bf_buff.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bf_buff.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/bio.h>

static int buffer_write(BIO *h, const char *buf,int num);
static int buffer_read(BIO *h, char *buf, int size);
static int buffer_puts(BIO *h, const char *str);
static int buffer_gets(BIO *h, char *str, int size);
static long buffer_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int buffer_new(BIO *h);
static int buffer_free(BIO *data);
static long buffer_callback_ctrl(BIO *h, int cmd, bio_info_cb *fp);
#define DEFAULT_BUFFER_SIZE     4096

static BIO_METHOD methods_buffer=
        {
        BIO_TYPE_BUFFER,
        "buffer",
        buffer_write,
        buffer_read,
        buffer_puts,
        buffer_gets,
        buffer_ctrl,
        buffer_new,
        buffer_free,
        buffer_callback_ctrl,
        };

BIO_METHOD *BIO_f_buffer(void)
        {
        return(&methods_buffer);
        }

static int buffer_new(BIO *bi)
        {
        BIO_F_BUFFER_CTX *ctx;

        ctx=(BIO_F_BUFFER_CTX *)OPENSSL_malloc(sizeof(BIO_F_BUFFER_CTX));
        if (ctx == NULL) return(0);
        ctx->ibuf=(char *)OPENSSL_malloc(DEFAULT_BUFFER_SIZE);
        if (ctx->ibuf == NULL) { OPENSSL_free(ctx); return(0); }
        ctx->obuf=(char *)OPENSSL_malloc(DEFAULT_BUFFER_SIZE);
        if (ctx->obuf == NULL) { OPENSSL_free(ctx->ibuf); OPENSSL_free(ctx); ret
        ctx->ibuf_size=DEFAULT_BUFFER_SIZE;
        ctx->obuf_size=DEFAULT_BUFFER_SIZE;
        ctx->ibuf_len=0;
        ctx->ibuf_off=0;
        ctx->obuf_len=0;
        ctx->obuf_off=0;

        bi->init=1;
        bi->ptr=(char *)ctx;
        bi->flags=0;
        return(1);
        }

static int buffer_free(BIO *a)
        {
        BIO_F_BUFFER_CTX *b;

        if (a == NULL) return(0);
        b=(BIO_F_BUFFER_CTX *)a->ptr;
        if (b->ibuf != NULL) OPENSSL_free(b->ibuf);
        if (b->obuf != NULL) OPENSSL_free(b->obuf);
        OPENSSL_free(a->ptr);
        a->ptr=NULL;
        a->init=0;
        a->flags=0;
```

```
128         return(1);
129         }

131 static int buffer_read(BIO *b, char *out, int outl)
132         {
133         int i,num=0;
134         BIO_F_BUFFER_CTX *ctx;

136         if (out == NULL) return(0);
137         ctx=(BIO_F_BUFFER_CTX *)b->ptr;

139         if ((ctx == NULL) || (b->next_bio == NULL)) return(0);
140         num=0;
141         BIO_clear_retry_flags(b);

143 start:
144         i=ctx->ibuf_len;
145         /* If there is stuff left over, grab it */
146         if (i != 0)
147                 {
148                 if (i > outl) i=outl;
149                 memcpy(out,&(ctx->ibuf[ctx->ibuf_off]),i);
150                 ctx->ibuf_off+=i;
151                 ctx->ibuf_len-=i;
152                 num+=i;
153                 if (outl == i)  return(num);
154                 outl-=i;
155                 out+=i;
156                 }

158         /* We may have done a partial read. try to do more.
159          * We have nothing in the buffer.
160          * If we get an error and have read some data, just return it
161          * and let them retry to get the error again.
162          * copy direct to parent address space */
163         if (outl > ctx->ibuf_size)
164                 {
165                 for (;;)
166                         {
167                         i=BIO_read(b->next_bio,out,outl);
168                         if (i <= 0)
169                                 {
170                                 BIO_copy_next_retry(b);
171                                 if (i < 0) return((num > 0)?num:i);
172                                 if (i == 0) return(num);
173                                 }
174                         num+=i;
175                         if (outl == i) return(num);
176                         out+=i;
177                         outl-=i;
178                         }
179                 }
180         /* else */

182         /* we are going to be doing some buffering */
183         i=BIO_read(b->next_bio,ctx->ibuf,ctx->ibuf_size);
184         if (i <= 0)
185                 {
186                 BIO_copy_next_retry(b);
187                 if (i < 0) return((num > 0)?num:i);
188                 if (i == 0) return(num);
189                 }
190         ctx->ibuf_off=0;
191         ctx->ibuf_len=i;

193         /* Lets re-read using ourselves :-) */
```



```
194         goto start;
195         }

197 static int buffer_write(BIO *b, const char *in, int inl)
198         {
199         int i,num=0;
200         BIO_F_BUFFER_CTX *ctx;

202         if ((in == NULL) || (inl <= 0)) return(0);
203         ctx=(BIO_F_BUFFER_CTX *)b->ptr;
204         if ((ctx == NULL) || (b->next_bio == NULL)) return(0);

206         BIO_clear_retry_flags(b);
207 start:
208         i=ctx->obuf_size-(ctx->obuf_len+ctx->obuf_off);
209         /* add to buffer and return */
210         if (i >= inl)
211                 {
212                 memcpy(&(ctx->obuf[ctx->obuf_off+ctx->obuf_len]),in,inl);
213                 ctx->obuf_len+=inl;
214                 return(num+inl);
215                 }
216         /* else */
217         /* stuff already in buffer, so add to it first, then flush */
218         if (ctx->obuf_len != 0)
219                 {
220                 if (i > 0) /* lets fill it up if we can */
221                         {
222                         memcpy(&(ctx->obuf[ctx->obuf_off+ctx->obuf_len]),in,i);
223                         in+=i;
224                         inl-=i;
225                         num+=i;
226                         ctx->obuf_len+=i;
227                         }
228                 /* we now have a full buffer needing flushing */
229                 for (;;)
230                         {
231                         i=BIO_write(b->next_bio,&(ctx->obuf[ctx->obuf_off]),
232                                 ctx->obuf_len);
233                         if (i <= 0)
234                                 {
235                                 BIO_copy_next_retry(b);

237                                 if (i < 0) return((num > 0)?num:i);
238                                 if (i == 0) return(num);
239                                 }
240                         ctx->obuf_off+=i;
241                         ctx->obuf_len-=i;
242                         if (ctx->obuf_len == 0) break;
243                         }
244                 }
245         /* we only get here if the buffer has been flushed and we
246          * still have stuff to write */
247         ctx->obuf_off=0;

249         /* we now have inl bytes to write */
250         while (inl >= ctx->obuf_size)
251                 {
252                 i=BIO_write(b->next_bio,in,inl);
253                 if (i <= 0)
254                         {
255                         BIO_copy_next_retry(b);
256                         if (i < 0) return((num > 0)?num:i);
257                         if (i == 0) return(num);
258                         }
259                 num+=i;
```

```
260                 in+=i;
261                 inl-=i;
262                 if (inl == 0) return(num);
263                 }

265         /* copy the rest into the buffer since we have only a small
266          * amount left */
267         goto start;
268         }

270 static long buffer_ctrl(BIO *b, int cmd, long num, void *ptr)
271         {
272         BIO *dbio;
273         BIO_F_BUFFER_CTX *ctx;
274         long ret=1;
275         char *p1,*p2;
276         int r,i,*ip;
277         int ibs,obs;

279         ctx=(BIO_F_BUFFER_CTX *)b->ptr;

281         switch (cmd)
282                 {
283         case BIO_CTRL_RESET:
284                 ctx->ibuf_off=0;
285                 ctx->ibuf_len=0;
286                 ctx->obuf_off=0;
287                 ctx->obuf_len=0;
288                 if (b->next_bio == NULL) return(0);
289                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
290                 break;
291         case BIO_CTRL_INFO:
292                 ret=(long)ctx->obuf_len;
293                 break;
294         case BIO_C_GET_BUFF_NUM_LINES:
295                 ret=0;
296                 p1=ctx->ibuf;
297                 for (i=0; i<ctx->ibuf_len; i++)
298                         {
299                         if (p1[ctx->ibuf_off + i] == '\n') ret++;
300                         }
301                 break;
302         case BIO_CTRL_WPENDING:
303                 ret=(long)ctx->obuf_len;
304                 if (ret == 0)
305                         {
306                         if (b->next_bio == NULL) return(0);
307                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
308                         }
309                 break;
310         case BIO_CTRL_PENDING:
311                 ret=(long)ctx->ibuf_len;
312                 if (ret == 0)
313                         {
314                         if (b->next_bio == NULL) return(0);
315                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
316                         }
317                 break;
318         case BIO_C_SET_BUFF_READ_DATA:
319                 if (num > ctx->ibuf_size)
320                         {
321                         p1=OPENSSL_malloc((int)num);
322                         if (p1 == NULL) goto malloc_error;
323                         if (ctx->ibuf != NULL) OPENSSL_free(ctx->ibuf);
324                         ctx->ibuf=p1;
325                         }
```



```
326                 ctx->ibuf_off=0;
327                 ctx->ibuf_len=(int)num;
328                 memcpy(ctx->ibuf,ptr,(int)num);
329                 ret=1;
330                 break;
331         case BIO_C_SET_BUFF_SIZE:
332                 if (ptr != NULL)
333                         {
334                         ip=(int *)ptr;
335                         if (*ip == 0)
336                                 {
337                                 ibs=(int)num;
338                                 obs=ctx->obuf_size;
339                                 }
340                         else /* if (*ip == 1) */
341                                 {
342                                 ibs=ctx->ibuf_size;
343                                 obs=(int)num;
344                                 }
345                         }
346                 else
347                         {
348                         ibs=(int)num;
349                         obs=(int)num;
350                         }
351                 p1=ctx->ibuf;
352                 p2=ctx->obuf;
353                 if ((ibs > DEFAULT_BUFFER_SIZE) && (ibs != ctx->ibuf_size))
354                         {
355                         p1=(char *)OPENSSL_malloc((int)num);
356                         if (p1 == NULL) goto malloc_error;
357                         }
358                 if ((obs > DEFAULT_BUFFER_SIZE) && (obs != ctx->obuf_size))
359                         {
360                         p2=(char *)OPENSSL_malloc((int)num);
361                         if (p2 == NULL)
362                                 {
363                                 if (p1 != ctx->ibuf) OPENSSL_free(p1);
364                                 goto malloc_error;
365                                 }
366                         }
367                 if (ctx->ibuf != p1)
368                         {
369                         OPENSSL_free(ctx->ibuf);
370                         ctx->ibuf=p1;
371                         ctx->ibuf_off=0;
372                         ctx->ibuf_len=0;
373                         ctx->ibuf_size=ibs;
374                         }
375                 if (ctx->obuf != p2)
376                         {
377                         OPENSSL_free(ctx->obuf);
378                         ctx->obuf=p2;
379                         ctx->obuf_off=0;
380                         ctx->obuf_len=0;
381                         ctx->obuf_size=obs;
382                         }
383                 break;
384         case BIO_C_DO_STATE_MACHINE:
385                 if (b->next_bio == NULL) return(0);
386                 BIO_clear_retry_flags(b);
387                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
388                 BIO_copy_next_retry(b);
389                 break;

391         case BIO_CTRL_FLUSH:
```

```
392                 if (b->next_bio == NULL) return(0);
393                 if (ctx->obuf_len <= 0)
394                         {
395                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
396                         break;
397                         }

399                 for (;;)
400                         {
401                         BIO_clear_retry_flags(b);
402                         if (ctx->obuf_len > 0)
403                                 {
404                                 r=BIO_write(b->next_bio,
405                                         &(ctx->obuf[ctx->obuf_off]),
406                                         ctx->obuf_len);
407 #if 0
408 fprintf(stderr,"FLUSH [%3d] %3d -> %3d\n",ctx->obuf_off,ctx->obuf_len,r);
409 #endif
410                                 BIO_copy_next_retry(b);
411                                 if (r <= 0) return((long)r);
412                                 ctx->obuf_off+=r;
413                                 ctx->obuf_len-=r;
414                                 }
415                         else
416                                 {
417                                 ctx->obuf_len=0;
418                                 ctx->obuf_off=0;
419                                 ret=1;
420                                 break;
421                                 }
422                         }
423                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
424                 break;
425         case BIO_CTRL_DUP:
426                 dbio=(BIO *)ptr;
427                 if (    !BIO_set_read_buffer_size(dbio,ctx->ibuf_size) ||
428                         !BIO_set_write_buffer_size(dbio,ctx->obuf_size))
429                         ret=0;
430                 break;
431         default:
432                 if (b->next_bio == NULL) return(0);
433                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
434                 break;
435                 }
436         return(ret);
437 malloc_error:
438         BIOerr(BIO_F_BUFFER_CTRL,ERR_R_MALLOC_FAILURE);
439         return(0);
440         }

442 static long buffer_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
443         {
444         long ret=1;

446         if (b->next_bio == NULL) return(0);
447         switch (cmd)
448                 {
449         default:
450                 ret=BIO_callback_ctrl(b->next_bio,cmd,fp);
451                 break;
452                 }
453         return(ret);
454         }

456 static int buffer_gets(BIO *b, char *buf, int size)
457         {
```



```
458         BIO_F_BUFFER_CTX *ctx;
459         int num=0,i,flag;
460         char *p;

462         ctx=(BIO_F_BUFFER_CTX *)b->ptr;
463         size--; /* reserve space for a '\0' */
464         BIO_clear_retry_flags(b);

466         for (;;)
467                 {
468                 if (ctx->ibuf_len > 0)
469                         {
470                         p= &(ctx->ibuf[ctx->ibuf_off]);
471                         flag=0;
472                         for (i=0; (i<ctx->ibuf_len) && (i<size); i++)
473                                 {
474                                 *(buf++)=p[i];
475                                 if (p[i] == '\n')
476                                         {
477                                         flag=1;
478                                         i++;
479                                         break;
480                                         }
481                                 }
482                         num+=i;
483                         size-=i;
484                         ctx->ibuf_len-=i;
485                         ctx->ibuf_off+=i;
486                         if (flag || size == 0)
487                                 {
488                                 *buf='\0';
489                                 return(num);
490                                 }
491                         }
492                 else    /* read another chunk */
493                         {
494                         i=BIO_read(b->next_bio,ctx->ibuf,ctx->ibuf_size);
495                         if (i <= 0)
496                                 {
497                                 BIO_copy_next_retry(b);
498                                 *buf='\0';
499                                 if (i < 0) return((num > 0)?num:i);
500                                 if (i == 0) return(num);
501                                 }
502                         ctx->ibuf_len=i;
503                         ctx->ibuf_off=0;
504                         }
505                 }
506         }

508 static int buffer_puts(BIO *b, const char *str)
509         {
510         return(buffer_write(b,str,strlen(str)));
511         }
512 #endif /* ! codereview */
```

```
**********************************************************
    6477 Wed Aug 13 19:52:11 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bf_nbio.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bf_nbio.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/rand.h>
#include <openssl/bio.h>

/* BIO_put and BIO_get both add to the digest,
 * BIO_gets returns the digest */

static int nbiof_write(BIO *h,const char *buf,int num);
static int nbiof_read(BIO *h,char *buf,int size);
static int nbiof_puts(BIO *h,const char *str);
static int nbiof_gets(BIO *h,char *str,int size);
static long nbiof_ctrl(BIO *h,int cmd,long arg1,void *arg2);
static int nbiof_new(BIO *h);
static int nbiof_free(BIO *data);
static long nbiof_callback_ctrl(BIO *h,int cmd,bio_info_cb *fp);
typedef struct nbio_test_st
        {
        /* only set if we sent a 'should retry' error */
        int lrn;
        int lwn;
        } NBIO_TEST;

static BIO_METHOD methods_nbiof=
        {
        BIO_TYPE_NBIO_TEST,
        "non-blocking IO test filter",
        nbiof_write,
        nbiof_read,
        nbiof_puts,
        nbiof_gets,
        nbiof_ctrl,
        nbiof_new,
        nbiof_free,
        nbiof_callback_ctrl,
        };

BIO_METHOD *BIO_f_nbio_test(void)
        {
        return(&methods_nbiof);
        }

static int nbiof_new(BIO *bi)
        {
        NBIO_TEST *nt;

        if (!(nt=(NBIO_TEST *)OPENSSL_malloc(sizeof(NBIO_TEST)))) return(0);
        nt->lrn= -1;
        nt->lwn= -1;
        bi->ptr=(char *)nt;
        bi->init=1;
        bi->flags=0;
        return(1);
        }

static int nbiof_free(BIO *a)
        {
        if (a == NULL) return(0);
        if (a->ptr != NULL)
                OPENSSL_free(a->ptr);
        a->ptr=NULL;
        a->init=0;
        a->flags=0;
        return(1);
        }

static int nbiof_read(BIO *b, char *out, int outl)
        {
```

```
128         int ret=0;
129 #if 1
130         int num;
131         unsigned char n;
132 #endif

134         if (out == NULL) return(0);
135         if (b->next_bio == NULL) return(0);

137         BIO_clear_retry_flags(b);
138 #if 1
139         RAND_pseudo_bytes(&n,1);
140         num=(n&0x07);

142         if (outl > num) outl=num;

144         if (num == 0)
145                 {
146                 ret= -1;
147                 BIO_set_retry_read(b);
148                 }
149         else
150 #endif
151                 {
152                 ret=BIO_read(b->next_bio,out,outl);
153                 if (ret < 0)
154                         BIO_copy_next_retry(b);
155                 }
156         return(ret);
157         }

159 static int nbiof_write(BIO *b, const char *in, int inl)
160         {
161         NBIO_TEST *nt;
162         int ret=0;
163         int num;
164         unsigned char n;

166         if ((in == NULL) || (inl <= 0)) return(0);
167         if (b->next_bio == NULL) return(0);
168         nt=(NBIO_TEST *)b->ptr;

170         BIO_clear_retry_flags(b);

172 #if 1
173         if (nt->lwn > 0)
174                 {
175                 num=nt->lwn;
176                 nt->lwn=0;
177                 }
178         else
179                 {
180                 RAND_pseudo_bytes(&n,1);
181                 num=(n&7);
182                 }

184         if (inl > num) inl=num;

186         if (num == 0)
187                 {
188                 ret= -1;
189                 BIO_set_retry_write(b);
190                 }
191         else
192 #endif
193                 {
```



```
194                 ret=BIO_write(b->next_bio,in,inl);
195                 if (ret < 0)
196                         {
197                         BIO_copy_next_retry(b);
198                         nt->lwn=inl;
199                         }
200                 }
201         return(ret);
202         }

204 static long nbiof_ctrl(BIO *b, int cmd, long num, void *ptr)
205         {
206         long ret;

208         if (b->next_bio == NULL) return(0);
209         switch (cmd)
210                 {
211         case BIO_C_DO_STATE_MACHINE:
212                 BIO_clear_retry_flags(b);
213                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
214                 BIO_copy_next_retry(b);
215                 break;
216         case BIO_CTRL_DUP:
217                 ret=0L;
218                 break;
219         default:
220                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
221                 break;
222                 }
223         return(ret);
224         }

226 static long nbiof_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
227         {
228         long ret=1;

230         if (b->next_bio == NULL) return(0);
231         switch (cmd)
232                 {
233         default:
234                 ret=BIO_callback_ctrl(b->next_bio,cmd,fp);
235                 break;
236                 }
237         return(ret);
238         }

240 static int nbiof_gets(BIO *bp, char *buf, int size)
241         {
242         if (bp->next_bio == NULL) return(0);
243         return(BIO_gets(bp->next_bio,buf,size));
244         }


247 static int nbiof_puts(BIO *bp, const char *str)
248         {
249         if (bp->next_bio == NULL) return(0);
250         return(BIO_puts(bp->next_bio,str));
251         }
252 #endif /* ! codereview */
```

```
**********************************************************
    5603 Wed Aug 13 19:52:11 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bf_null.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bf_null.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/bio.h>

/* BIO_put and BIO_get both add to the digest,
 * BIO_gets returns the digest */

static int nullf_write(BIO *h, const char *buf, int num);
static int nullf_read(BIO *h, char *buf, int size);
static int nullf_puts(BIO *h, const char *str);
static int nullf_gets(BIO *h, char *str, int size);
static long nullf_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int nullf_new(BIO *h);
static int nullf_free(BIO *data);
static long nullf_callback_ctrl(BIO *h, int cmd, bio_info_cb *fp);
static BIO_METHOD methods_nullf=
        {
        BIO_TYPE_NULL_FILTER,
        "NULL filter",
        nullf_write,
        nullf_read,
        nullf_puts,
        nullf_gets,
        nullf_ctrl,
        nullf_new,
        nullf_free,
        nullf_callback_ctrl,
        };

BIO_METHOD *BIO_f_null(void)
        {
        return(&methods_nullf);
        }

static int nullf_new(BIO *bi)
        {
        bi->init=1;
        bi->ptr=NULL;
        bi->flags=0;
        return(1);
        }

static int nullf_free(BIO *a)
        {
        if (a == NULL) return(0);
/*      a->ptr=NULL;
        a->init=0;
        a->flags=0;*/
        return(1);
        }

static int nullf_read(BIO *b, char *out, int outl)
        {
        int ret=0;

        if (out == NULL) return(0);
        if (b->next_bio == NULL) return(0);
        ret=BIO_read(b->next_bio,out,outl);
        BIO_clear_retry_flags(b);
        BIO_copy_next_retry(b);
        return(ret);
        }

static int nullf_write(BIO *b, const char *in, int inl)
        {
        int ret=0;

        if ((in == NULL) || (inl <= 0)) return(0);
```

```
128         if (b->next_bio == NULL) return(0);
129         ret=BIO_write(b->next_bio,in,inl);
130         BIO_clear_retry_flags(b);
131         BIO_copy_next_retry(b);
132         return(ret);
133         }

135 static long nullf_ctrl(BIO *b, int cmd, long num, void *ptr)
136         {
137         long ret;

139         if (b->next_bio == NULL) return(0);
140         switch(cmd)
141                 {
142         case BIO_C_DO_STATE_MACHINE:
143                 BIO_clear_retry_flags(b);
144                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
145                 BIO_copy_next_retry(b);
146                 break;
147         case BIO_CTRL_DUP:
148                 ret=0L;
149                 break;
150         default:
151                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
152                 }
153         return(ret);
154         }

156 static long nullf_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
157         {
158         long ret=1;

160         if (b->next_bio == NULL) return(0);
161         switch (cmd)
162                 {
163         default:
164                 ret=BIO_callback_ctrl(b->next_bio,cmd,fp);
165                 break;
166                 }
167         return(ret);
168         }

170 static int nullf_gets(BIO *bp, char *buf, int size)
171         {
172         if (bp->next_bio == NULL) return(0);
173         return(BIO_gets(bp->next_bio,buf,size));
174         }


177 static int nullf_puts(BIO *bp, const char *str)
178         {
179         if (bp->next_bio == NULL) return(0);
180         return(BIO_puts(bp->next_bio,str));
181         }
182 #endif /* ! codereview */
```

```
**********************************************************
    5461 Wed Aug 13 19:52:11 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bio_cb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bio_cb.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```



```
#include "cryptlib.h"
#include <openssl/bio.h>
#include <openssl/err.h>

long MS_CALLBACK BIO_debug_callback(BIO *bio, int cmd, const char *argp,
        int argi, long argl, long ret)
        {
        BIO *b;
        MS_STATIC char buf[256];
        char *p;
        long r=1;
        size_t p_maxlen;

        if (BIO_CB_RETURN & cmd)
                r=ret;

        BIO_snprintf(buf,sizeof buf,"BIO[%08lX]:",(unsigned long)bio);
        p= &(buf[14]);
        p_maxlen = sizeof buf - 14;
        switch (cmd)
                {
        case BIO_CB_FREE:
                BIO_snprintf(p,p_maxlen,"Free - %s\n",bio->method->name);
                break;
        case BIO_CB_READ:
                if (bio->method->type & BIO_TYPE_DESCRIPTOR)
                        BIO_snprintf(p,p_maxlen,"read(%d,%lu) - %s fd=%d\n",
                                bio->num,(unsigned long)argi,
                                bio->method->name,bio->num);
                else
                        BIO_snprintf(p,p_maxlen,"read(%d,%lu) - %s\n",
                                bio->num,(unsigned long)argi,
                                bio->method->name);
                break;
        case BIO_CB_WRITE:
                if (bio->method->type & BIO_TYPE_DESCRIPTOR)
                        BIO_snprintf(p,p_maxlen,"write(%d,%lu) - %s fd=%d\n",
                                bio->num,(unsigned long)argi,
                                bio->method->name,bio->num);
                else
                        BIO_snprintf(p,p_maxlen,"write(%d,%lu) - %s\n",
                                bio->num,(unsigned long)argi,
                                bio->method->name);
                break;
        case BIO_CB_PUTS:
                BIO_snprintf(p,p_maxlen,"puts() - %s\n",bio->method->name);
                break;
        case BIO_CB_GETS:
                BIO_snprintf(p,p_maxlen,"gets(%lu) - %s\n",(unsigned long)argi,b
                break;
        case BIO_CB_CTRL:
                BIO_snprintf(p,p_maxlen,"ctrl(%lu) - %s\n",(unsigned long)argi,b
                break;
        case BIO_CB_RETURN|BIO_CB_READ:
                BIO_snprintf(p,p_maxlen,"read return %ld\n",ret);
                break;
        case BIO_CB_RETURN|BIO_CB_WRITE:
                BIO_snprintf(p,p_maxlen,"write return %ld\n",ret);
                break;
        case BIO_CB_RETURN|BIO_CB_GETS:
                BIO_snprintf(p,p_maxlen,"gets return %ld\n",ret);
                break;
        case BIO_CB_RETURN|BIO_CB_PUTS:
                BIO_snprintf(p,p_maxlen,"puts return %ld\n",ret);
                break;
        case BIO_CB_RETURN|BIO_CB_CTRL:
```

```
128                 BIO_snprintf(p,p_maxlen,"ctrl return %ld\n",ret);
129                 break;
130         default:
131                 BIO_snprintf(p,p_maxlen,"bio callback - unknown type (%d)\n",cmd
132                 break;
133                 }

135         b=(BIO *)bio->cb_arg;
136         if (b != NULL)
137                 BIO_write(b,buf,strlen(buf));
138 #if !defined(OPENSSL_NO_STDIO) && !defined(OPENSSL_SYS_WIN16)
139         else
140                 fputs(buf,stderr);
141 #endif
142         return(r);
143         }
144 #endif /* ! codereview */
```

```
**********************************************************
    6890 Wed Aug 13 19:52:11 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bio_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bio_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/bio.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_BIO,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_BIO,0,reason)

static ERR_STRING_DATA BIO_str_functs[]=
        {
{ERR_FUNC(BIO_F_ACPT_STATE),    "ACPT_STATE"},
{ERR_FUNC(BIO_F_BIO_ACCEPT),    "BIO_accept"},
{ERR_FUNC(BIO_F_BIO_BER_GET_HEADER),    "BIO_BER_GET_HEADER"},
{ERR_FUNC(BIO_F_BIO_CALLBACK_CTRL),     "BIO_callback_ctrl"},
{ERR_FUNC(BIO_F_BIO_CTRL),      "BIO_ctrl"},
{ERR_FUNC(BIO_F_BIO_GETHOSTBYNAME),     "BIO_gethostbyname"},
{ERR_FUNC(BIO_F_BIO_GETS),      "BIO_gets"},
{ERR_FUNC(BIO_F_BIO_GET_ACCEPT_SOCKET), "BIO_get_accept_socket"},
{ERR_FUNC(BIO_F_BIO_GET_HOST_IP),       "BIO_get_host_ip"},
{ERR_FUNC(BIO_F_BIO_GET_PORT),  "BIO_get_port"},
{ERR_FUNC(BIO_F_BIO_MAKE_PAIR), "BIO_MAKE_PAIR"},
{ERR_FUNC(BIO_F_BIO_NEW),       "BIO_new"},
{ERR_FUNC(BIO_F_BIO_NEW_FILE),  "BIO_new_file"},
{ERR_FUNC(BIO_F_BIO_NEW_MEM_BUF),       "BIO_new_mem_buf"},
{ERR_FUNC(BIO_F_BIO_NREAD),     "BIO_nread"},
{ERR_FUNC(BIO_F_BIO_NREAD0),    "BIO_nread0"},
{ERR_FUNC(BIO_F_BIO_NWRITE),    "BIO_nwrite"},
{ERR_FUNC(BIO_F_BIO_NWRITE0),   "BIO_nwrite0"},
{ERR_FUNC(BIO_F_BIO_PUTS),      "BIO_puts"},
{ERR_FUNC(BIO_F_BIO_READ),      "BIO_read"},
{ERR_FUNC(BIO_F_BIO_SOCK_INIT), "BIO_sock_init"},
{ERR_FUNC(BIO_F_BIO_WRITE),     "BIO_write"},
{ERR_FUNC(BIO_F_BUFFER_CTRL),   "BUFFER_CTRL"},
{ERR_FUNC(BIO_F_CONN_CTRL),     "CONN_CTRL"},
{ERR_FUNC(BIO_F_CONN_STATE),    "CONN_STATE"},
{ERR_FUNC(BIO_F_DGRAM_SCTP_READ),       "DGRAM_SCTP_READ"},
{ERR_FUNC(BIO_F_FILE_CTRL),     "FILE_CTRL"},
{ERR_FUNC(BIO_F_FILE_READ),     "FILE_READ"},
{ERR_FUNC(BIO_F_LINEBUFFER_CTRL),       "LINEBUFFER_CTRL"},
{ERR_FUNC(BIO_F_MEM_READ),      "MEM_READ"},
{ERR_FUNC(BIO_F_MEM_WRITE),     "MEM_WRITE"},
{ERR_FUNC(BIO_F_SSL_NEW),       "SSL_new"},
{ERR_FUNC(BIO_F_WSASTARTUP),    "WSASTARTUP"},
{0,NULL}
        };

static ERR_STRING_DATA BIO_str_reasons[]=
        {
{ERR_REASON(BIO_R_ACCEPT_ERROR)          ,"accept error"},
{ERR_REASON(BIO_R_BAD_FOPEN_MODE)        ,"bad fopen mode"},
{ERR_REASON(BIO_R_BAD_HOSTNAME_LOOKUP)   ,"bad hostname lookup"},
{ERR_REASON(BIO_R_BROKEN_PIPE)           ,"broken pipe"},
{ERR_REASON(BIO_R_CONNECT_ERROR)         ,"connect error"},
{ERR_REASON(BIO_R_EOF_ON_MEMORY_BIO)     ,"EOF on memory BIO"},
{ERR_REASON(BIO_R_ERROR_SETTING_NBIO)    ,"error setting nbio"},
{ERR_REASON(BIO_R_ERROR_SETTING_NBIO_ON_ACCEPTED_SOCKET),"error setting nbio on
{ERR_REASON(BIO_R_ERROR_SETTING_NBIO_ON_ACCEPT_SOCKET),"error setting nbio on ac
{ERR_REASON(BIO_R_GETHOSTBYNAME_ADDR_IS_NOT_AF_INET),"gethostbyname addr is not
{ERR_REASON(BIO_R_INVALID_ARGUMENT)      ,"invalid argument"},
{ERR_REASON(BIO_R_INVALID_IP_ADDRESS)    ,"invalid ip address"},
{ERR_REASON(BIO_R_IN_USE)                ,"in use"},
{ERR_REASON(BIO_R_KEEPALIVE)             ,"keepalive"},
{ERR_REASON(BIO_R_NBIO_CONNECT_ERROR)    ,"nbio connect error"},
{ERR_REASON(BIO_R_NO_ACCEPT_PORT_SPECIFIED),"no accept port specified"},
{ERR_REASON(BIO_R_NO_HOSTNAME_SPECIFIED) ,"no hostname specified"},
```

```
128 {ERR_REASON(BIO_R_NO_PORT_DEFINED)       ,"no port defined"},
129 {ERR_REASON(BIO_R_NO_PORT_SPECIFIED)     ,"no port specified"},
130 {ERR_REASON(BIO_R_NO_SUCH_FILE)          ,"no such file"},
131 {ERR_REASON(BIO_R_NULL_PARAMETER)        ,"null parameter"},
132 {ERR_REASON(BIO_R_TAG_MISMATCH)          ,"tag mismatch"},
133 {ERR_REASON(BIO_R_UNABLE_TO_BIND_SOCKET) ,"unable to bind socket"},
134 {ERR_REASON(BIO_R_UNABLE_TO_CREATE_SOCKET),"unable to create socket"},
135 {ERR_REASON(BIO_R_UNABLE_TO_LISTEN_SOCKET),"unable to listen socket"},
136 {ERR_REASON(BIO_R_UNINITIALIZED)         ,"uninitialized"},
137 {ERR_REASON(BIO_R_UNSUPPORTED_METHOD)    ,"unsupported method"},
138 {ERR_REASON(BIO_R_WRITE_TO_READ_ONLY_BIO),"write to read only BIO"},
139 {ERR_REASON(BIO_R_WSASTARTUP)            ,"WSAStartup"},
140 {0,NULL}
141         };

143 #endif

145 void ERR_load_BIO_strings(void)
146         {
147 #ifndef OPENSSL_NO_ERR

149         if (ERR_func_error_string(BIO_str_functs[0].error) == NULL)
150                 {
151                 ERR_load_strings(0,BIO_str_functs);
152                 ERR_load_strings(0,BIO_str_reasons);
153                 }
154 #endif
155         }
156 #endif /* ! codereview */
```

```
**********************************************************
   13106 Wed Aug 13 19:52:11 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bio_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bio_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include <openssl/crypto.h>
```



```
#include "cryptlib.h"
#include <openssl/bio.h>
#include <openssl/stack.h>

BIO *BIO_new(BIO_METHOD *method)
        {
        BIO *ret=NULL;

        ret=(BIO *)OPENSSL_malloc(sizeof(BIO));
        if (ret == NULL)
                {
                BIOerr(BIO_F_BIO_NEW,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }
        if (!BIO_set(ret,method))
                {
                OPENSSL_free(ret);
                ret=NULL;
                }
        return(ret);
        }

int BIO_set(BIO *bio, BIO_METHOD *method)
        {
        bio->method=method;
        bio->callback=NULL;
        bio->cb_arg=NULL;
        bio->init=0;
        bio->shutdown=1;
        bio->flags=0;
        bio->retry_reason=0;
        bio->num=0;
        bio->ptr=NULL;
        bio->prev_bio=NULL;
        bio->next_bio=NULL;
        bio->references=1;
        bio->num_read=0L;
        bio->num_write=0L;
        CRYPTO_new_ex_data(CRYPTO_EX_INDEX_BIO, bio, &bio->ex_data);
        if (method->create != NULL)
                if (!method->create(bio))
                        {
                        CRYPTO_free_ex_data(CRYPTO_EX_INDEX_BIO, bio,
                                            &bio->ex_data);
                        return(0);
                        }
        return(1);
        }

int BIO_free(BIO *a)
        {
        int i;

        if (a == NULL) return(0);

        i=CRYPTO_add(&a->references,-1,CRYPTO_LOCK_BIO);
#ifdef REF_PRINT
        REF_PRINT("BIO",a);
#endif
        if (i > 0) return(1);
#ifdef REF_CHECK
        if (i < 0)
                {
                fprintf(stderr,"BIO_free, bad reference count\n");
                abort();
                }
```

```
#endif
        if ((a->callback != NULL) &&
                ((i=(int)a->callback(a,BIO_CB_FREE,NULL,0,0L,1L)) <= 0))
                        return(i);

        CRYPTO_free_ex_data(CRYPTO_EX_INDEX_BIO, a, &a->ex_data);

        if ((a->method != NULL) && (a->method->destroy != NULL))
        a->method->destroy(a);
        OPENSSL_free(a);
        return(1);
        }

void BIO_vfree(BIO *a)
    { BIO_free(a); }

void BIO_clear_flags(BIO *b, int flags)
        {
        b->flags &= ~flags;
        }

int     BIO_test_flags(const BIO *b, int flags)
        {
        return (b->flags & flags);
        }

void    BIO_set_flags(BIO *b, int flags)
        {
        b->flags |= flags;
        }

long (*BIO_get_callback(const BIO *b))(struct bio_st *,int,const char *,int, lon
        {
        return b->callback;
        }

void BIO_set_callback(BIO *b, long (*cb)(struct bio_st *,int,const char *,int, l
        {
        b->callback = cb;
        }

void BIO_set_callback_arg(BIO *b, char *arg)
        {
        b->cb_arg = arg;
        }

char * BIO_get_callback_arg(const BIO *b)
        {
        return b->cb_arg;
        }

const char * BIO_method_name(const BIO *b)
        {
        return b->method->name;
        }

int BIO_method_type(const BIO *b)
        {
        return b->method->type;
        }


int BIO_read(BIO *b, void *out, int outl)
        {
        int i;
        long (*cb)(BIO *,int,const char *,int,long,long);
```



```
        if ((b == NULL) || (b->method == NULL) || (b->method->bread == NULL))
                {
                BIOerr(BIO_F_BIO_READ,BIO_R_UNSUPPORTED_METHOD);
                return(-2);
                }

        cb=b->callback;
        if ((cb != NULL) &&
                ((i=(int)cb(b,BIO_CB_READ,out,outl,0L,1L)) <= 0))
                        return(i);

        if (!b->init)
                {
                BIOerr(BIO_F_BIO_READ,BIO_R_UNINITIALIZED);
                return(-2);
                }

        i=b->method->bread(b,out,outl);

        if (i > 0) b->num_read+=(unsigned long)i;

        if (cb != NULL)
                i=(int)cb(b,BIO_CB_READ|BIO_CB_RETURN,out,outl,
                        0L,(long)i);
        return(i);
        }

int BIO_write(BIO *b, const void *in, int inl)
        {
        int i;
        long (*cb)(BIO *,int,const char *,int,long,long);

        if (b == NULL)
                return(0);

        cb=b->callback;
        if ((b->method == NULL) || (b->method->bwrite == NULL))
                {
                BIOerr(BIO_F_BIO_WRITE,BIO_R_UNSUPPORTED_METHOD);
                return(-2);
                }

        if ((cb != NULL) &&
                ((i=(int)cb(b,BIO_CB_WRITE,in,inl,0L,1L)) <= 0))
                        return(i);

        if (!b->init)
                {
                BIOerr(BIO_F_BIO_WRITE,BIO_R_UNINITIALIZED);
                return(-2);
                }

        i=b->method->bwrite(b,in,inl);

        if (i > 0) b->num_write+=(unsigned long)i;

        if (cb != NULL)
                i=(int)cb(b,BIO_CB_WRITE|BIO_CB_RETURN,in,inl,
                        0L,(long)i);
        return(i);
        }

int BIO_puts(BIO *b, const char *in)
        {
        int i;
```

```
260         long (*cb)(BIO *,int,const char *,int,long,long);

262         if ((b == NULL) || (b->method == NULL) || (b->method->bputs == NULL))
263                 {
264                 BIOerr(BIO_F_BIO_PUTS,BIO_R_UNSUPPORTED_METHOD);
265                 return(-2);
266                 }

268         cb=b->callback;

270         if ((cb != NULL) &&
271                 ((i=(int)cb(b,BIO_CB_PUTS,in,0,0L,1L)) <= 0))
272                         return(i);

274         if (!b->init)
275                 {
276                 BIOerr(BIO_F_BIO_PUTS,BIO_R_UNINITIALIZED);
277                 return(-2);
278                 }

280         i=b->method->bputs(b,in);

282         if (i > 0) b->num_write+=(unsigned long)i;

284         if (cb != NULL)
285                 i=(int)cb(b,BIO_CB_PUTS|BIO_CB_RETURN,in,0,
286                         0L,(long)i);
287         return(i);
288         }

290 int BIO_gets(BIO *b, char *in, int inl)
291         {
292         int i;
293         long (*cb)(BIO *,int,const char *,int,long,long);

295         if ((b == NULL) || (b->method == NULL) || (b->method->bgets == NULL))
296                 {
297                 BIOerr(BIO_F_BIO_GETS,BIO_R_UNSUPPORTED_METHOD);
298                 return(-2);
299                 }

301         cb=b->callback;

303         if ((cb != NULL) &&
304                 ((i=(int)cb(b,BIO_CB_GETS,in,inl,0L,1L)) <= 0))
305                         return(i);

307         if (!b->init)
308                 {
309                 BIOerr(BIO_F_BIO_GETS,BIO_R_UNINITIALIZED);
310                 return(-2);
311                 }

313         i=b->method->bgets(b,in,inl);

315         if (cb != NULL)
316                 i=(int)cb(b,BIO_CB_GETS|BIO_CB_RETURN,in,inl,
317                         0L,(long)i);
318         return(i);
319         }

321 int BIO_indent(BIO *b,int indent,int max)
322         {
323         if(indent < 0)
324                 indent=0;
325         if(indent > max)
```



```
326                 indent=max;
327         while(indent--)
328                 if(BIO_puts(b," ") != 1)
329                         return 0;
330         return 1;
331         }

333 long BIO_int_ctrl(BIO *b, int cmd, long larg, int iarg)
334         {
335         int i;

337         i=iarg;
338         return(BIO_ctrl(b,cmd,larg,(char *)&i));
339         }

341 char *BIO_ptr_ctrl(BIO *b, int cmd, long larg)
342         {
343         char *p=NULL;

345         if (BIO_ctrl(b,cmd,larg,(char *)&p) <= 0)
346                 return(NULL);
347         else
348                 return(p);
349         }

351 long BIO_ctrl(BIO *b, int cmd, long larg, void *parg)
352         {
353         long ret;
354         long (*cb)(BIO *,int,const char *,int,long,long);

356         if (b == NULL) return(0);

358         if ((b->method == NULL) || (b->method->ctrl == NULL))
359                 {
360                 BIOerr(BIO_F_BIO_CTRL,BIO_R_UNSUPPORTED_METHOD);
361                 return(-2);
362                 }

364         cb=b->callback;

366         if ((cb != NULL) &&
367                 ((ret=cb(b,BIO_CB_CTRL,parg,cmd,larg,1L)) <= 0))
368                 return(ret);

370         ret=b->method->ctrl(b,cmd,larg,parg);

372         if (cb != NULL)
373                 ret=cb(b,BIO_CB_CTRL|BIO_CB_RETURN,parg,cmd,
374                         larg,ret);
375         return(ret);
376         }

378 long BIO_callback_ctrl(BIO *b, int cmd, void (*fp)(struct bio_st *, int, const c
379         {
380         long ret;
381         long (*cb)(BIO *,int,const char *,int,long,long);

383         if (b == NULL) return(0);

385         if ((b->method == NULL) || (b->method->callback_ctrl == NULL))
386                 {
387                 BIOerr(BIO_F_BIO_CALLBACK_CTRL,BIO_R_UNSUPPORTED_METHOD);
388                 return(-2);
389                 }

391         cb=b->callback;
```

```
393         if ((cb != NULL) &&
394                 ((ret=cb(b,BIO_CB_CTRL,(void *)&fp,cmd,0,1L)) <= 0))
395                 return(ret);

397         ret=b->method->callback_ctrl(b,cmd,fp);

399         if (cb != NULL)
400                 ret=cb(b,BIO_CB_CTRL|BIO_CB_RETURN,(void *)&fp,cmd,
401                         0,ret);
402         return(ret);
403         }

405 /* It is unfortunate to duplicate in functions what the BIO_(w)pending macros
406  * do; but those macros have inappropriate return type, and for interfacing
407  * from other programming languages, C macros aren't much of a help anyway. */
408 size_t BIO_ctrl_pending(BIO *bio)
409         {
410         return BIO_ctrl(bio, BIO_CTRL_PENDING, 0, NULL);
411         }

413 size_t BIO_ctrl_wpending(BIO *bio)
414         {
415         return BIO_ctrl(bio, BIO_CTRL_WPENDING, 0, NULL);
416         }


419 /* put the 'bio' on the end of b's list of operators */
420 BIO *BIO_push(BIO *b, BIO *bio)
421         {
422         BIO *lb;

424         if (b == NULL) return(bio);
425         lb=b;
426         while (lb->next_bio != NULL)
427                 lb=lb->next_bio;
428         lb->next_bio=bio;
429         if (bio != NULL)
430                 bio->prev_bio=lb;
431         /* called to do internal processing */
432         BIO_ctrl(b,BIO_CTRL_PUSH,0,lb);
433         return(b);
434         }

436 /* Remove the first and return the rest */
437 BIO *BIO_pop(BIO *b)
438         {
439         BIO *ret;

441         if (b == NULL) return(NULL);
442         ret=b->next_bio;

444         BIO_ctrl(b,BIO_CTRL_POP,0,b);

446         if (b->prev_bio != NULL)
447                 b->prev_bio->next_bio=b->next_bio;
448         if (b->next_bio != NULL)
449                 b->next_bio->prev_bio=b->prev_bio;

451         b->next_bio=NULL;
452         b->prev_bio=NULL;
453         return(ret);
454         }

456 BIO *BIO_get_retry_BIO(BIO *bio, int *reason)
457         {
```



```
458         BIO *b,*last;

460         b=last=bio;
461         for (;;)
462                 {
463                 if (!BIO_should_retry(b)) break;
464                 last=b;
465                 b=b->next_bio;
466                 if (b == NULL) break;
467                 }
468         if (reason != NULL) *reason=last->retry_reason;
469         return(last);
470         }

472 int BIO_get_retry_reason(BIO *bio)
473         {
474         return(bio->retry_reason);
475         }

477 BIO *BIO_find_type(BIO *bio, int type)
478         {
479         int mt,mask;

481         if(!bio) return NULL;
482         mask=type&0xff;
483         do      {
484                 if (bio->method != NULL)
485                         {
486                         mt=bio->method->type;

488                         if (!mask)
489                                 {
490                                 if (mt & type) return(bio);
491                                 }
492                         else if (mt == type)
493                                 return(bio);
494                         }
495                 bio=bio->next_bio;
496                 } while (bio != NULL);
497         return(NULL);
498         }

500 BIO *BIO_next(BIO *b)
501         {
502         if(!b) return NULL;
503         return b->next_bio;
504         }

506 void BIO_free_all(BIO *bio)
507         {
508         BIO *b;
509         int ref;

511         while (bio != NULL)
512                 {
513                 b=bio;
514                 ref=b->references;
515                 bio=bio->next_bio;
516                 BIO_free(b);
517                 /* Since ref count > 1, don't free anyone else. */
518                 if (ref > 1) break;
519                 }
520         }

522 BIO *BIO_dup_chain(BIO *in)
523         {
```

```
524         BIO *ret=NULL,*eoc=NULL,*bio,*new_bio;

526         for (bio=in; bio != NULL; bio=bio->next_bio)
527                 {
528                 if ((new_bio=BIO_new(bio->method)) == NULL) goto err;
529                 new_bio->callback=bio->callback;
530                 new_bio->cb_arg=bio->cb_arg;
531                 new_bio->init=bio->init;
532                 new_bio->shutdown=bio->shutdown;
533                 new_bio->flags=bio->flags;

535                 /* This will let SSL_s_sock() work with stdin/stdout */
536                 new_bio->num=bio->num;

538                 if (!BIO_dup_state(bio,(char *)new_bio))
539                         {
540                         BIO_free(new_bio);
541                         goto err;
542                         }

544                 /* copy app data */
545                 if (!CRYPTO_dup_ex_data(CRYPTO_EX_INDEX_BIO, &new_bio->ex_data,
546                                         &bio->ex_data))
547                         goto err;

549                 if (ret == NULL)
550                         {
551                         eoc=new_bio;
552                         ret=eoc;
553                         }
554                 else
555                         {
556                         BIO_push(eoc,new_bio);
557                         eoc=new_bio;
558                         }
559                 }
560         return(ret);
561 err:
562         if (ret != NULL)
563                 BIO_free(ret);
564         return(NULL);
565         }

567 void BIO_copy_next_retry(BIO *b)
568         {
569         BIO_set_flags(b,BIO_get_retry_flags(b->next_bio));
570         b->retry_reason=b->next_bio->retry_reason;
571         }

573 int BIO_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
574              CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
575         {
576         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_BIO, argl, argp,
577                                        new_func, dup_func, free_func);
578         }

580 int BIO_set_ex_data(BIO *bio, int idx, void *data)
581         {
582         return(CRYPTO_set_ex_data(&(bio->ex_data),idx,data));
583         }

585 void *BIO_get_ex_data(BIO *bio, int idx)
586         {
587         return(CRYPTO_get_ex_data(&(bio->ex_data),idx));
588         }
```



```
590 unsigned long BIO_number_read(BIO *bio)
591 {
592         if(bio) return bio->num_read;
593         return 0;
594 }

596 unsigned long BIO_number_written(BIO *bio)
597 {
598         if(bio) return bio->num_write;
599         return 0;
600 }

602 IMPLEMENT_STACK_OF(BIO)
603 #endif /* ! codereview */
```

```
**********************************************************
   10885 Wed Aug 13 19:52:11 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_acpt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_acpt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#define USE_SOCKETS
```



```
#include "cryptlib.h"
#include <openssl/bio.h>

#ifndef OPENSSL_NO_SOCK

#ifdef OPENSSL_SYS_WIN16
#define SOCKET_PROTOCOL 0 /* more microsoft stupidity */
#else
#define SOCKET_PROTOCOL IPPROTO_TCP
#endif

#if (defined(OPENSSL_SYS_VMS) && __VMS_VER < 70000000)
/* FIONBIO used as a switch to enable ioctl, and that isn't in VMS < 7.0 */
#undef FIONBIO
#endif

typedef struct bio_accept_st
        {
        int state;
        char *param_addr;

        int accept_sock;
        int accept_nbio;

        char *addr;
        int nbio;
        /* If 0, it means normal, if 1, do a connect on bind failure,
         * and if there is no-one listening, bind with SO_REUSEADDR.
         * If 2, always use SO_REUSEADDR. */
        int bind_mode;
        BIO *bio_chain;
        } BIO_ACCEPT;

static int acpt_write(BIO *h, const char *buf, int num);
static int acpt_read(BIO *h, char *buf, int size);
static int acpt_puts(BIO *h, const char *str);
static long acpt_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int acpt_new(BIO *h);
static int acpt_free(BIO *data);
static int acpt_state(BIO *b, BIO_ACCEPT *c);
static void acpt_close_socket(BIO *data);
static BIO_ACCEPT *BIO_ACCEPT_new(void );
static void BIO_ACCEPT_free(BIO_ACCEPT *a);

#define ACPT_S_BEFORE                   1
#define ACPT_S_GET_ACCEPT_SOCKET        2
#define ACPT_S_OK                       3

static BIO_METHOD methods_acceptp=
        {
        BIO_TYPE_ACCEPT,
        "socket accept",
        acpt_write,
        acpt_read,
        acpt_puts,
        NULL, /* connect_gets, */
        acpt_ctrl,
        acpt_new,
        acpt_free,
        NULL,
        };

BIO_METHOD *BIO_s_accept(void)
        {
        return(&methods_acceptp);
        }
```

```
129 static int acpt_new(BIO *bi)
130         {
131         BIO_ACCEPT *ba;

133         bi->init=0;
134         bi->num=INVALID_SOCKET;
135         bi->flags=0;
136         if ((ba=BIO_ACCEPT_new()) == NULL)
137                 return(0);
138         bi->ptr=(char *)ba;
139         ba->state=ACPT_S_BEFORE;
140         bi->shutdown=1;
141         return(1);
142         }

144 static BIO_ACCEPT *BIO_ACCEPT_new(void)
145         {
146         BIO_ACCEPT *ret;

148         if ((ret=(BIO_ACCEPT *)OPENSSL_malloc(sizeof(BIO_ACCEPT))) == NULL)
149                 return(NULL);

151         memset(ret,0,sizeof(BIO_ACCEPT));
152         ret->accept_sock=INVALID_SOCKET;
153         ret->bind_mode=BIO_BIND_NORMAL;
154         return(ret);
155         }

157 static void BIO_ACCEPT_free(BIO_ACCEPT *a)
158         {
159         if(a == NULL)
160             return;

162         if (a->param_addr != NULL) OPENSSL_free(a->param_addr);
163         if (a->addr != NULL) OPENSSL_free(a->addr);
164         if (a->bio_chain != NULL) BIO_free(a->bio_chain);
165         OPENSSL_free(a);
166         }

168 static void acpt_close_socket(BIO *bio)
169         {
170         BIO_ACCEPT *c;

172         c=(BIO_ACCEPT *)bio->ptr;
173         if (c->accept_sock != INVALID_SOCKET)
174                 {
175                 shutdown(c->accept_sock,2);
176                 closesocket(c->accept_sock);
177                 c->accept_sock=INVALID_SOCKET;
178                 bio->num=INVALID_SOCKET;
179                 }
180         }

182 static int acpt_free(BIO *a)
183         {
184         BIO_ACCEPT *data;

186         if (a == NULL) return(0);
187         data=(BIO_ACCEPT *)a->ptr;

189         if (a->shutdown)
190                 {
191                 acpt_close_socket(a);
192                 BIO_ACCEPT_free(data);
193                 a->ptr=NULL;
```



```
194                 a->flags=0;
195                 a->init=0;
196                 }
197         return(1);
198         }

200 static int acpt_state(BIO *b, BIO_ACCEPT *c)
201         {
202         BIO *bio=NULL,*dbio;
203         int s= -1;
204         int i;

206 again:
207         switch (c->state)
208                 {
209         case ACPT_S_BEFORE:
210                 if (c->param_addr == NULL)
211                         {
212                         BIOerr(BIO_F_ACPT_STATE,BIO_R_NO_ACCEPT_PORT_SPECIFIED);
213                         return(-1);
214                         }
215                 s=BIO_get_accept_socket(c->param_addr,c->bind_mode);
216                 if (s == INVALID_SOCKET)
217                         return(-1);

219                 if (c->accept_nbio)
220                         {
221                         if (!BIO_socket_nbio(s,1))
222                                 {
223                                 closesocket(s);
224                                 BIOerr(BIO_F_ACPT_STATE,BIO_R_ERROR_SETTING_NBIO
225                                 return(-1);
226                                 }
227                         }
228                 c->accept_sock=s;
229                 b->num=s;
230                 c->state=ACPT_S_GET_ACCEPT_SOCKET;
231                 return(1);
232                 /* break; */
233         case ACPT_S_GET_ACCEPT_SOCKET:
234                 if (b->next_bio != NULL)
235                         {
236                         c->state=ACPT_S_OK;
237                         goto again;
238                         }
239                 BIO_clear_retry_flags(b);
240                 b->retry_reason=0;
241                 i=BIO_accept(c->accept_sock,&(c->addr));

243                 /* -2 return means we should retry */
244                 if(i == -2)
245                         {
246                         BIO_set_retry_special(b);
247                         b->retry_reason=BIO_RR_ACCEPT;
248                         return -1;
249                         }

251                 if (i < 0) return(i);

253                 bio=BIO_new_socket(i,BIO_CLOSE);
254                 if (bio == NULL) goto err;

256                 BIO_set_callback(bio,BIO_get_callback(b));
257                 BIO_set_callback_arg(bio,BIO_get_callback_arg(b));

259                 if (c->nbio)
```

```
260                         {
261                         if (!BIO_socket_nbio(i,1))
262                                 {
263                                 BIOerr(BIO_F_ACPT_STATE,BIO_R_ERROR_SETTING_NBIO
264                                 goto err;
265                                 }
266                         }

268                 /* If the accept BIO has an bio_chain, we dup it and
269                  * put the new socket at the end. */
270                 if (c->bio_chain != NULL)
271                         {
272                         if ((dbio=BIO_dup_chain(c->bio_chain)) == NULL)
273                                 goto err;
274                         if (!BIO_push(dbio,bio)) goto err;
275                         bio=dbio;
276                         }
277                 if (BIO_push(b,bio) == NULL) goto err;

279                 c->state=ACPT_S_OK;
280                 return(1);
281 err:
282                 if (bio != NULL)
283                         BIO_free(bio);
284                 else if (s >= 0)
285                         closesocket(s);
286                 return(0);
287                 /* break; */
288         case ACPT_S_OK:
289                 if (b->next_bio == NULL)
290                         {
291                         c->state=ACPT_S_GET_ACCEPT_SOCKET;
292                         goto again;
293                         }
294                 return(1);
295                 /* break; */
296         default:
297                 return(0);
298                 /* break; */
299                 }

301         }

303 static int acpt_read(BIO *b, char *out, int outl)
304         {
305         int ret=0;
306         BIO_ACCEPT *data;

308         BIO_clear_retry_flags(b);
309         data=(BIO_ACCEPT *)b->ptr;

311         while (b->next_bio == NULL)
312                 {
313                 ret=acpt_state(b,data);
314                 if (ret <= 0) return(ret);
315                 }

317         ret=BIO_read(b->next_bio,out,outl);
318         BIO_copy_next_retry(b);
319         return(ret);
320         }

322 static int acpt_write(BIO *b, const char *in, int inl)
323         {
324         int ret;
325         BIO_ACCEPT *data;
```



```
327         BIO_clear_retry_flags(b);
328         data=(BIO_ACCEPT *)b->ptr;

330         while (b->next_bio == NULL)
331                 {
332                 ret=acpt_state(b,data);
333                 if (ret <= 0) return(ret);
334                 }

336         ret=BIO_write(b->next_bio,in,inl);
337         BIO_copy_next_retry(b);
338         return(ret);
339         }

341 static long acpt_ctrl(BIO *b, int cmd, long num, void *ptr)
342         {
343         int *ip;
344         long ret=1;
345         BIO_ACCEPT *data;
346         char **pp;

348         data=(BIO_ACCEPT *)b->ptr;

350         switch (cmd)
351                 {
352         case BIO_CTRL_RESET:
353                 ret=0;
354                 data->state=ACPT_S_BEFORE;
355                 acpt_close_socket(b);
356                 b->flags=0;
357                 break;
358         case BIO_C_DO_STATE_MACHINE:
359                 /* use this one to start the connection */
360                 ret=(long)acpt_state(b,data);
361                 break;
362         case BIO_C_SET_ACCEPT:
363                 if (ptr != NULL)
364                         {
365                         if (num == 0)
366                                 {
367                                 b->init=1;
368                                 if (data->param_addr != NULL)
369                                         OPENSSL_free(data->param_addr);
370                                 data->param_addr=BUF_strdup(ptr);
371                                 }
372                         else if (num == 1)
373                                 {
374                                 data->accept_nbio=(ptr != NULL);
375                                 }
376                         else if (num == 2)
377                                 {
378                                 if (data->bio_chain != NULL)
379                                         BIO_free(data->bio_chain);
380                                 data->bio_chain=(BIO *)ptr;
381                                 }
382                         }
383                 break;
384         case BIO_C_SET_NBIO:
385                 data->nbio=(int)num;
386                 break;
387         case BIO_C_SET_FD:
388                 b->init=1;
389                 b->num= *((int *)ptr);
390                 data->accept_sock=b->num;
391                 data->state=ACPT_S_GET_ACCEPT_SOCKET;
```

```
392                 b->shutdown=(int)num;
393                 b->init=1;
394                 break;
395         case BIO_C_GET_FD:
396                 if (b->init)
397                         {
398                         ip=(int *)ptr;
399                         if (ip != NULL)
400                                 *ip=data->accept_sock;
401                         ret=data->accept_sock;
402                         }
403                 else
404                         ret= -1;
405                 break;
406         case BIO_C_GET_ACCEPT:
407                 if (b->init)
408                         {
409                         if (ptr != NULL)
410                                 {
411                                 pp=(char **)ptr;
412                                 *pp=data->param_addr;
413                                 }
414                         else
415                                 ret= -1;
416                         }
417                 else
418                         ret= -1;
419                 break;
420         case BIO_CTRL_GET_CLOSE:
421                 ret=b->shutdown;
422                 break;
423         case BIO_CTRL_SET_CLOSE:
424                 b->shutdown=(int)num;
425                 break;
426         case BIO_CTRL_PENDING:
427         case BIO_CTRL_WPENDING:
428                 ret=0;
429                 break;
430         case BIO_CTRL_FLUSH:
431                 break;
432         case BIO_C_SET_BIND_MODE:
433                 data->bind_mode=(int)num;
434                 break;
435         case BIO_C_GET_BIND_MODE:
436                 ret=(long)data->bind_mode;
437                 break;
438         case BIO_CTRL_DUP:
439 /*              dbio=(BIO *)ptr;
440                 if (data->param_port) EAY EAY
441                         BIO_set_port(dbio,data->param_port);
442                 if (data->param_hostname)
443                         BIO_set_hostname(dbio,data->param_hostname);
444                 BIO_set_nbio(dbio,data->nbio); */
445                 break;

447         default:
448                 ret=0;
449                 break;
450                 }
451         return(ret);
452         }

454 static int acpt_puts(BIO *bp, const char *str)
455         {
456         int n,ret;
```



```
458         n=strlen(str);
459         ret=acpt_write(bp,str,n);
460         return(ret);
461         }

463 BIO *BIO_new_accept(char *str)
464         {
465         BIO *ret;

467         ret=BIO_new(BIO_s_accept());
468         if (ret == NULL) return(NULL);
469         if (BIO_set_accept_port(ret,str))
470                 return(ret);
471         else
472                 {
473                 BIO_free(ret);
474                 return(NULL);
475                 }
476         }

478 #endif
479 #endif /* ! codereview */
```

```
**********************************************************
   18983 Wed Aug 13 19:52:12 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_bio.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_bio.c  -*- Mode: C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 1998-2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* Special method for a BIO where the other endpoint is also a BIO
 * of this kind, handled by the same thread (i.e. the "peer" is actually
 * ourselves, wearing a different hat).
 * Such "BIO pairs" are mainly for using the SSL library with I/O interfaces
 * for which no specific BIO method is available.
 * See ssl/ssltest.c for some hints on how this can be used. */
```



```
/* BIO_DEBUG implies BIO_PAIR_DEBUG */
#ifdef BIO_DEBUG
# ifndef BIO_PAIR_DEBUG
#  define BIO_PAIR_DEBUG
# endif
#endif

/* disable assert() unless BIO_PAIR_DEBUG has been defined */
#ifndef BIO_PAIR_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif

#include <assert.h>
#include <limits.h>
#include <stdlib.h>
#include <string.h>

#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/crypto.h>

#include "e_os.h"

/* VxWorks defines SSIZE_MAX with an empty value causing compile errors */
#if defined(OPENSSL_SYS_VXWORKS)
# undef SSIZE_MAX
#endif
#ifndef SSIZE_MAX
# define SSIZE_MAX INT_MAX
#endif

static int bio_new(BIO *bio);
static int bio_free(BIO *bio);
static int bio_read(BIO *bio, char *buf, int size);
static int bio_write(BIO *bio, const char *buf, int num);
static long bio_ctrl(BIO *bio, int cmd, long num, void *ptr);
static int bio_puts(BIO *bio, const char *str);

static int bio_make_pair(BIO *bio1, BIO *bio2);
static void bio_destroy_pair(BIO *bio);

static BIO_METHOD methods_biop =
{
        BIO_TYPE_BIO,
        "BIO pair",
        bio_write,
        bio_read,
        bio_puts,
        NULL /* no bio_gets */,
        bio_ctrl,
        bio_new,
        bio_free,
        NULL /* no bio_callback_ctrl */
};

BIO_METHOD *BIO_s_bio(void)
        {
        return &methods_biop;
        }

struct bio_bio_st
{
        BIO *peer;     /* NULL if buf == NULL.
```

```
128                         * If peer != NULL, then peer->ptr is also a bio_bio_st,
129                         * and its "peer" member points back to us.
130                         * peer != NULL iff init != 0 in the BIO. */
131
132         /* This is for what we write (i.e. reading uses peer's struct): */
133         int closed;     /* valid iff peer != NULL */
134         size_t len;     /* valid iff buf != NULL; 0 if peer == NULL */
135         size_t offset;  /* valid iff buf != NULL; 0 if len == 0 */
136         size_t size;
137         char *buf;      /* "size" elements (if != NULL) */
138
139         size_t request; /* valid iff peer != NULL; 0 if len != 0,
140                          * otherwise set by peer to number of bytes
141                          * it (unsuccessfully) tried to read,
142                          * never more than buffer space (size-len) warrants. */
143 };
144
145 static int bio_new(BIO *bio)
146         {
147         struct bio_bio_st *b;
148
149         b = OPENSSL_malloc(sizeof *b);
150         if (b == NULL)
151                 return 0;
152
153         b->peer = NULL;
154         b->size = 17*1024; /* enough for one TLS record (just a default) */
155         b->buf = NULL;
156
157         bio->ptr = b;
158         return 1;
159         }
160
161
162 static int bio_free(BIO *bio)
163         {
164         struct bio_bio_st *b;
165
166         if (bio == NULL)
167                 return 0;
168         b = bio->ptr;
169
170         assert(b != NULL);
171
172         if (b->peer)
173                 bio_destroy_pair(bio);
174
175         if (b->buf != NULL)
176                 {
177                 OPENSSL_free(b->buf);
178                 }
179
180         OPENSSL_free(b);
181
182         return 1;
183         }
184
185
186
187 static int bio_read(BIO *bio, char *buf, int size_)
188         {
189         size_t size = size_;
190         size_t rest;
191         struct bio_bio_st *b, *peer_b;
192
193         BIO_clear_retry_flags(bio);
```



```
194
195         if (!bio->init)
196                 return 0;
197
198         b = bio->ptr;
199         assert(b != NULL);
200         assert(b->peer != NULL);
201         peer_b = b->peer->ptr;
202         assert(peer_b != NULL);
203         assert(peer_b->buf != NULL);
204
205         peer_b->request = 0; /* will be set in "retry_read" situation */
206
207         if (buf == NULL || size == 0)
208                 return 0;
209
210         if (peer_b->len == 0)
211                 {
212                 if (peer_b->closed)
213                         return 0; /* writer has closed, and no data is left */
214                 else
215                         {
216                         BIO_set_retry_read(bio); /* buffer is empty */
217                         if (size <= peer_b->size)
218                                 peer_b->request = size;
219                         else
220                                 /* don't ask for more than the peer can
221                                  * deliver in one write */
222                                 peer_b->request = peer_b->size;
223                         return -1;
224                         }
225                 }
226
227         /* we can read */
228         if (peer_b->len < size)
229                 size = peer_b->len;
230
231         /* now read "size" bytes */
232
233         rest = size;
234
235         assert(rest > 0);
236         do /* one or two iterations */
237                 {
238                 size_t chunk;
239
240                 assert(rest <= peer_b->len);
241                 if (peer_b->offset + rest <= peer_b->size)
242                         chunk = rest;
243                 else
244                         /* wrap around ring buffer */
245                         chunk = peer_b->size - peer_b->offset;
246                 assert(peer_b->offset + chunk <= peer_b->size);
247
248                 memcpy(buf, peer_b->buf + peer_b->offset, chunk);
249
250                 peer_b->len -= chunk;
251                 if (peer_b->len)
252                         {
253                         peer_b->offset += chunk;
254                         assert(peer_b->offset <= peer_b->size);
255                         if (peer_b->offset == peer_b->size)
256                                 peer_b->offset = 0;
257                         buf += chunk;
258                         }
259                 else
```

```
260                         {
261                         /* buffer now empty, no need to advance "buf" */
262                         assert(chunk == rest);
263                         peer_b->offset = 0;
264                         }
265                 rest -= chunk;
266                 }
267         while (rest);

269         return size;
270         }

272 /* non-copying interface: provide pointer to available data in buffer
273  *    bio_nread0:  return number of available bytes
274  *    bio_nread:   also advance index
275  * (example usage:  bio_nread0(), read from buffer, bio_nread()
276  *  or just         bio_nread(), read from buffer)
277  */
278 /* WARNING: The non-copying interface is largely untested as of yet
279  * and may contain bugs. */
280 static ossl_ssize_t bio_nread0(BIO *bio, char **buf)
281         {
282         struct bio_bio_st *b, *peer_b;
283         ossl_ssize_t num;

285         BIO_clear_retry_flags(bio);

287         if (!bio->init)
288                 return 0;

290         b = bio->ptr;
291         assert(b != NULL);
292         assert(b->peer != NULL);
293         peer_b = b->peer->ptr;
294         assert(peer_b != NULL);
295         assert(peer_b->buf != NULL);

297         peer_b->request = 0;

299         if (peer_b->len == 0)
300                 {
301                 char dummy;

303                 /* avoid code duplication -- nothing available for reading */
304                 return bio_read(bio, &dummy, 1); /* returns 0 or -1 */
305                 }

307         num = peer_b->len;
308         if (peer_b->size < peer_b->offset + num)
309                 /* no ring buffer wrap-around for non-copying interface */
310                 num = peer_b->size - peer_b->offset;
311         assert(num > 0);

313         if (buf != NULL)
314                 *buf = peer_b->buf + peer_b->offset;
315         return num;
316         }

318 static ossl_ssize_t bio_nread(BIO *bio, char **buf, size_t num_)
319         {
320         struct bio_bio_st *b, *peer_b;
321         ossl_ssize_t num, available;

323         if (num_ > SSIZE_MAX)
324                 num = SSIZE_MAX;
325         else
```



```
326                 num = (ossl_ssize_t)num_;

328         available = bio_nread0(bio, buf);
329         if (num > available)
330                 num = available;
331         if (num <= 0)
332                 return num;

334         b = bio->ptr;
335         peer_b = b->peer->ptr;

337         peer_b->len -= num;
338         if (peer_b->len)
339                 {
340                 peer_b->offset += num;
341                 assert(peer_b->offset <= peer_b->size);
342                 if (peer_b->offset == peer_b->size)
343                         peer_b->offset = 0;
344                 }
345         else
346                 peer_b->offset = 0;

348         return num;
349         }


352 static int bio_write(BIO *bio, const char *buf, int num_)
353         {
354         size_t num = num_;
355         size_t rest;
356         struct bio_bio_st *b;

358         BIO_clear_retry_flags(bio);

360         if (!bio->init || buf == NULL || num == 0)
361                 return 0;

363         b = bio->ptr;
364         assert(b != NULL);
365         assert(b->peer != NULL);
366         assert(b->buf != NULL);

368         b->request = 0;
369         if (b->closed)
370                 {
371                 /* we already closed */
372                 BIOerr(BIO_F_BIO_WRITE, BIO_R_BROKEN_PIPE);
373                 return -1;
374                 }

376         assert(b->len <= b->size);

378         if (b->len == b->size)
379                 {
380                 BIO_set_retry_write(bio); /* buffer is full */
381                 return -1;
382                 }

384         /* we can write */
385         if (num > b->size - b->len)
386                 num = b->size - b->len;

388         /* now write "num" bytes */

390         rest = num;
```

```
392         assert(rest > 0);
393         do /* one or two iterations */
394                 {
395                 size_t write_offset;
396                 size_t chunk;

398                 assert(b->len + rest <= b->size);

400                 write_offset = b->offset + b->len;
401                 if (write_offset >= b->size)
402                         write_offset -= b->size;
403                 /* b->buf[write_offset] is the first byte we can write to. */

405                 if (write_offset + rest <= b->size)
406                         chunk = rest;
407                 else
408                         /* wrap around ring buffer */
409                         chunk = b->size - write_offset;

411                 memcpy(b->buf + write_offset, buf, chunk);

413                 b->len += chunk;

415                 assert(b->len <= b->size);

417                 rest -= chunk;
418                 buf += chunk;
419                 }
420         while (rest);

422         return num;
423         }

425 /* non-copying interface: provide pointer to region to write to
426  *   bio_nwrite0:  check how much space is available
427  *   bio_nwrite:   also increase length
428  * (example usage:  bio_nwrite0(), write to buffer, bio_nwrite()
429  *  or just         bio_nwrite(), write to buffer)
430  */
431 static ossl_ssize_t bio_nwrite0(BIO *bio, char **buf)
432         {
433         struct bio_bio_st *b;
434         size_t num;
435         size_t write_offset;

437         BIO_clear_retry_flags(bio);

439         if (!bio->init)
440                 return 0;

442         b = bio->ptr;
443         assert(b != NULL);
444         assert(b->peer != NULL);
445         assert(b->buf != NULL);

447         b->request = 0;
448         if (b->closed)
449                 {
450                 BIOerr(BIO_F_BIO_NWRITE0, BIO_R_BROKEN_PIPE);
451                 return -1;
452                 }

454         assert(b->len <= b->size);

456         if (b->len == b->size)
457                 {
```



```
458                 BIO_set_retry_write(bio);
459                 return -1;
460                 }

462         num = b->size - b->len;
463         write_offset = b->offset + b->len;
464         if (write_offset >= b->size)
465                 write_offset -= b->size;
466         if (write_offset + num > b->size)
467                 /* no ring buffer wrap-around for non-copying interface
468                  * (to fulfil the promise by BIO_ctrl_get_write_guarantee,
469                  * BIO_nwrite may have to be called twice) */
470                 num = b->size - write_offset;

472         if (buf != NULL)
473                 *buf = b->buf + write_offset;
474         assert(write_offset + num <= b->size);

476         return num;
477         }

479 static ossl_ssize_t bio_nwrite(BIO *bio, char **buf, size_t num_)
480         {
481         struct bio_bio_st *b;
482         ossl_ssize_t num, space;

484         if (num_ > SSIZE_MAX)
485                 num = SSIZE_MAX;
486         else
487                 num = (ossl_ssize_t)num_;

489         space = bio_nwrite0(bio, buf);
490         if (num > space)
491                 num = space;
492         if (num <= 0)
493                 return num;
494         b = bio->ptr;
495         assert(b != NULL);
496         b->len += num;
497         assert(b->len <= b->size);

499         return num;
500         }


503 static long bio_ctrl(BIO *bio, int cmd, long num, void *ptr)
504         {
505         long ret;
506         struct bio_bio_st *b = bio->ptr;

508         assert(b != NULL);

510         switch (cmd)
511                 {
512         /* specific CTRL codes */

514         case BIO_C_SET_WRITE_BUF_SIZE:
515                 if (b->peer)
516                         {
517                         BIOerr(BIO_F_BIO_CTRL, BIO_R_IN_USE);
518                         ret = 0;
519                         }
520                 else if (num == 0)
521                         {
522                         BIOerr(BIO_F_BIO_CTRL, BIO_R_INVALID_ARGUMENT);
523                         ret = 0;
```

```
524                         }
525                 else
526                         {
527                         size_t new_size = num;
528
529                         if (b->size != new_size)
530                                 {
531                                 if (b->buf)
532                                         {
533                                         OPENSSL_free(b->buf);
534                                         b->buf = NULL;
535                                         }
536                                 b->size = new_size;
537                                 }
538                         ret = 1;
539                         }
540                 break;
541
542         case BIO_C_GET_WRITE_BUF_SIZE:
543                 ret = (long) b->size;
544                 break;
545
546         case BIO_C_MAKE_BIO_PAIR:
547                 {
548                 BIO *other_bio = ptr;
549
550                 if (bio_make_pair(bio, other_bio))
551                         ret = 1;
552                 else
553                         ret = 0;
554                 }
555                 break;
556
557         case BIO_C_DESTROY_BIO_PAIR:
558                 /* Affects both BIOs in the pair -- call just once!
559                  * Or let BIO_free(bio1); BIO_free(bio2); do the job. */
560                 bio_destroy_pair(bio);
561                 ret = 1;
562                 break;
563
564         case BIO_C_GET_WRITE_GUARANTEE:
565                 /* How many bytes can the caller feed to the next write
566                  * without having to keep any? */
567                 if (b->peer == NULL || b->closed)
568                         ret = 0;
569                 else
570                         ret = (long) b->size - b->len;
571                 break;
572
573         case BIO_C_GET_READ_REQUEST:
574                 /* If the peer unsuccessfully tried to read, how many bytes
575                  * were requested?  (As with BIO_CTRL_PENDING, that number
576                  * can usually be treated as boolean.) */
577                 ret = (long) b->request;
578                 break;
579
580         case BIO_C_RESET_READ_REQUEST:
581                 /* Reset request.  (Can be useful after read attempts
582                  * at the other side that are meant to be non-blocking,
583                  * e.g. when probing SSL_read to see if any data is
584                  * available.) */
585                 b->request = 0;
586                 ret = 1;
587                 break;
588
589         case BIO_C_SHUTDOWN_WR:
```



```
590                 /* similar to shutdown(..., SHUT_WR) */
591                 b->closed = 1;
592                 ret = 1;
593                 break;
594
595         case BIO_C_NREAD0:
596                 /* prepare for non-copying read */
597                 ret = (long) bio_nread0(bio, ptr);
598                 break;
599
600         case BIO_C_NREAD:
601                 /* non-copying read */
602                 ret = (long) bio_nread(bio, ptr, (size_t) num);
603                 break;
604
605         case BIO_C_NWRITE0:
606                 /* prepare for non-copying write */
607                 ret = (long) bio_nwrite0(bio, ptr);
608                 break;
609
610         case BIO_C_NWRITE:
611                 /* non-copying write */
612                 ret = (long) bio_nwrite(bio, ptr, (size_t) num);
613                 break;
614
615
616         /* standard CTRL codes follow */
617
618         case BIO_CTRL_RESET:
619                 if (b->buf != NULL)
620                         {
621                         b->len = 0;
622                         b->offset = 0;
623                         }
624                 ret = 0;
625                 break;
626
627         case BIO_CTRL_GET_CLOSE:
628                 ret = bio->shutdown;
629                 break;
630
631         case BIO_CTRL_SET_CLOSE:
632                 bio->shutdown = (int) num;
633                 ret = 1;
634                 break;
635
636         case BIO_CTRL_PENDING:
637                 if (b->peer != NULL)
638                         {
639                         struct bio_bio_st *peer_b = b->peer->ptr;
640
641                         ret = (long) peer_b->len;
642                         }
643                 else
644                         ret = 0;
645                 break;
646
647         case BIO_CTRL_WPENDING:
648                 if (b->buf != NULL)
649                         ret = (long) b->len;
650                 else
651                         ret = 0;
652                 break;
653
654         case BIO_CTRL_DUP:
655                 /* See BIO_dup_chain for circumstances we have to expect. */
```

```
656                 {
657                 BIO *other_bio = ptr;
658                 struct bio_bio_st *other_b;

660                 assert(other_bio != NULL);
661                 other_b = other_bio->ptr;
662                 assert(other_b != NULL);

664                 assert(other_b->buf == NULL); /* other_bio is always fresh */

666                 other_b->size = b->size;
667                 }

669                 ret = 1;
670                 break;

672         case BIO_CTRL_FLUSH:
673                 ret = 1;
674                 break;

676         case BIO_CTRL_EOF:
677                 {
678                 BIO *other_bio = ptr;

680                 if (other_bio)
681                         {
682                         struct bio_bio_st *other_b = other_bio->ptr;

684                         assert(other_b != NULL);
685                         ret = other_b->len == 0 && other_b->closed;
686                         }
687                 else
688                         ret = 1;
689                 }
690                 break;

692         default:
693                 ret = 0;
694                 }
695         return ret;
696         }

698 static int bio_puts(BIO *bio, const char *str)
699         {
700         return bio_write(bio, str, strlen(str));
701         }


704 static int bio_make_pair(BIO *bio1, BIO *bio2)
705         {
706         struct bio_bio_st *b1, *b2;

708         assert(bio1 != NULL);
709         assert(bio2 != NULL);

711         b1 = bio1->ptr;
712         b2 = bio2->ptr;

714         if (b1->peer != NULL || b2->peer != NULL)
715                 {
716                 BIOerr(BIO_F_BIO_MAKE_PAIR, BIO_R_IN_USE);
717                 return 0;
718                 }

720         if (b1->buf == NULL)
721                 {
```



```
722                 b1->buf = OPENSSL_malloc(b1->size);
723                 if (b1->buf == NULL)
724                         {
725                         BIOerr(BIO_F_BIO_MAKE_PAIR, ERR_R_MALLOC_FAILURE);
726                         return 0;
727                         }
728                 b1->len = 0;
729                 b1->offset = 0;
730                 }

732         if (b2->buf == NULL)
733                 {
734                 b2->buf = OPENSSL_malloc(b2->size);
735                 if (b2->buf == NULL)
736                         {
737                         BIOerr(BIO_F_BIO_MAKE_PAIR, ERR_R_MALLOC_FAILURE);
738                         return 0;
739                         }
740                 b2->len = 0;
741                 b2->offset = 0;
742                 }

744         b1->peer = bio2;
745         b1->closed = 0;
746         b1->request = 0;
747         b2->peer = bio1;
748         b2->closed = 0;
749         b2->request = 0;

751         bio1->init = 1;
752         bio2->init = 1;

754         return 1;
755         }

757 static void bio_destroy_pair(BIO *bio)
758         {
759         struct bio_bio_st *b = bio->ptr;

761         if (b != NULL)
762                 {
763                 BIO *peer_bio = b->peer;

765                 if (peer_bio != NULL)
766                         {
767                         struct bio_bio_st *peer_b = peer_bio->ptr;

769                         assert(peer_b != NULL);
770                         assert(peer_b->peer == bio);

772                         peer_b->peer = NULL;
773                         peer_bio->init = 0;
774                         assert(peer_b->buf != NULL);
775                         peer_b->len = 0;
776                         peer_b->offset = 0;

778                         b->peer = NULL;
779                         bio->init = 0;
780                         assert(b->buf != NULL);
781                         b->len = 0;
782                         b->offset = 0;
783                         }
784                 }
785         }
```

```
/* Exported convenience functions */
int BIO_new_bio_pair(BIO **bio1_p, size_t writebuf1,
        BIO **bio2_p, size_t writebuf2)
        {
        BIO *bio1 = NULL, *bio2 = NULL;
        long r;
        int ret = 0;

        bio1 = BIO_new(BIO_s_bio());
        if (bio1 == NULL)
                goto err;
        bio2 = BIO_new(BIO_s_bio());
        if (bio2 == NULL)
                goto err;

        if (writebuf1)
                {
                r = BIO_set_write_buf_size(bio1, writebuf1);
                if (!r)
                        goto err;
                }
        if (writebuf2)
                {
                r = BIO_set_write_buf_size(bio2, writebuf2);
                if (!r)
                        goto err;
                }

        r = BIO_make_bio_pair(bio1, bio2);
        if (!r)
                goto err;
        ret = 1;

 err:
        if (ret == 0)
                {
                if (bio1)
                        {
                        BIO_free(bio1);
                        bio1 = NULL;
                        }
                if (bio2)
                        {
                        BIO_free(bio2);
                        bio2 = NULL;
                        }
                }

        *bio1_p = bio1;
        *bio2_p = bio2;
        return ret;
        }

size_t BIO_ctrl_get_write_guarantee(BIO *bio)
        {
        return BIO_ctrl(bio, BIO_C_GET_WRITE_GUARANTEE, 0, NULL);
        }

size_t BIO_ctrl_get_read_request(BIO *bio)
        {
        return BIO_ctrl(bio, BIO_C_GET_READ_REQUEST, 0, NULL);
        }

int BIO_ctrl_reset_read_request(BIO *bio)
        {
        return (BIO_ctrl(bio, BIO_C_RESET_READ_REQUEST, 0, NULL) != 0);
```



```
        }


/* BIO_nread0/nread/nwrite0/nwrite are available only for BIO pairs for now
 * (conceivably some other BIOs could allow non-copying reads and writes too.)
 */
int BIO_nread0(BIO *bio, char **buf)
        {
        long ret;

        if (!bio->init)
                {
                BIOerr(BIO_F_BIO_NREAD0, BIO_R_UNINITIALIZED);
                return -2;
                }

        ret = BIO_ctrl(bio, BIO_C_NREAD0, 0, buf);
        if (ret > INT_MAX)
                return INT_MAX;
        else
                return (int) ret;
        }

int BIO_nread(BIO *bio, char **buf, int num)
        {
        int ret;

        if (!bio->init)
                {
                BIOerr(BIO_F_BIO_NREAD, BIO_R_UNINITIALIZED);
                return -2;
                }

        ret = (int) BIO_ctrl(bio, BIO_C_NREAD, num, buf);
        if (ret > 0)
                bio->num_read += ret;
        return ret;
        }

int BIO_nwrite0(BIO *bio, char **buf)
        {
        long ret;

        if (!bio->init)
                {
                BIOerr(BIO_F_BIO_NWRITE0, BIO_R_UNINITIALIZED);
                return -2;
                }

        ret = BIO_ctrl(bio, BIO_C_NWRITE0, 0, buf);
        if (ret > INT_MAX)
                return INT_MAX;
        else
                return (int) ret;
        }

int BIO_nwrite(BIO *bio, char **buf, int num)
        {
        int ret;

        if (!bio->init)
                {
                BIOerr(BIO_F_BIO_NWRITE, BIO_R_UNINITIALIZED);
                return -2;
                }
```

```
920         ret = BIO_ctrl(bio, BIO_C_NWRITE, num, buf);
921         if (ret > 0)
922                 bio->num_write += ret;
923         return ret;
924         }
925 #endif /* ! codereview */
```

```
**********************************************************
   14924 Wed Aug 13 19:52:12 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_conn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_conn.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#define USE_SOCKETS
```



```
#include "cryptlib.h"
#include <openssl/bio.h>

#ifndef OPENSSL_NO_SOCK

#ifdef OPENSSL_SYS_WIN16
#define SOCKET_PROTOCOL 0 /* more microsoft stupidity */
#else
#define SOCKET_PROTOCOL IPPROTO_TCP
#endif

#if (defined(OPENSSL_SYS_VMS) && __VMS_VER < 70000000)
/* FIONBIO used as a switch to enable ioctl, and that isn't in VMS < 7.0 */
#undef FIONBIO
#endif


typedef struct bio_connect_st
        {
        int state;

        char *param_hostname;
        char *param_port;
        int nbio;

        unsigned char ip[4];
        unsigned short port;

        struct sockaddr_in them;

        /* int socket; this will be kept in bio->num so that it is
         * compatible with the bss_sock bio */

        /* called when the connection is initially made
         *  callback(BIO,state,ret);  The callback should return
         * 'ret'.  state is for compatibility with the ssl info_callback */
        int (*info_callback)(const BIO *bio,int state,int ret);
        } BIO_CONNECT;

static int conn_write(BIO *h, const char *buf, int num);
static int conn_read(BIO *h, char *buf, int size);
static int conn_puts(BIO *h, const char *str);
static long conn_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int conn_new(BIO *h);
static int conn_free(BIO *data);
static long conn_callback_ctrl(BIO *h, int cmd, bio_info_cb *);

static int conn_state(BIO *b, BIO_CONNECT *c);
static void conn_close_socket(BIO *data);
BIO_CONNECT *BIO_CONNECT_new(void );
void BIO_CONNECT_free(BIO_CONNECT *a);

static BIO_METHOD methods_connectp=
        {
        BIO_TYPE_CONNECT,
        "socket connect",
        conn_write,
        conn_read,
        conn_puts,
        NULL, /* connect_gets, */
        conn_ctrl,
        conn_new,
        conn_free,
        conn_callback_ctrl,
        };
```

```
128 static int conn_state(BIO *b, BIO_CONNECT *c)
129         {
130         int ret= -1,i;
131         unsigned long l;
132         char *p,*q;
133         int (*cb)(const BIO *,int,int)=NULL;
134
135         if (c->info_callback != NULL)
136                 cb=c->info_callback;
137
138         for (;;)
139                 {
140                 switch (c->state)
141                         {
142                 case BIO_CONN_S_BEFORE:
143                         p=c->param_hostname;
144                         if (p == NULL)
145                                 {
146                                 BIOerr(BIO_F_CONN_STATE,BIO_R_NO_HOSTNAME_SPECIF
147                                 goto exit_loop;
148                                 }
149                         for ( ; *p != '\0'; p++)
150                                 {
151                                 if ((*p == ':') || (*p == '/')) break;
152                                 }
153
154                         i= *p;
155                         if ((i == ':') || (i == '/'))
156                                 {
157
158                                 *(p++)='\0';
159                                 if (i == ':')
160                                         {
161                                         for (q=p; *q; q++)
162                                                 if (*q == '/')
163                                                         {
164                                                         *q='\0';
165                                                         break;
166                                                         }
167                                         if (c->param_port != NULL)
168                                                 OPENSSL_free(c->param_port);
169                                         c->param_port=BUF_strdup(p);
170                                         }
171                                 }
172
173                         if (c->param_port == NULL)
174                                 {
175                                 BIOerr(BIO_F_CONN_STATE,BIO_R_NO_PORT_SPECIFIED)
176                                 ERR_add_error_data(2,"host=",c->param_hostname);
177                                 goto exit_loop;
178                                 }
179                         c->state=BIO_CONN_S_GET_IP;
180                         break;
181
182                 case BIO_CONN_S_GET_IP:
183                         if (BIO_get_host_ip(c->param_hostname,&(c->ip[0])) <= 0)
184                                 goto exit_loop;
185                         c->state=BIO_CONN_S_GET_PORT;
186                         break;
187
188                 case BIO_CONN_S_GET_PORT:
189                         if (c->param_port == NULL)
190                                 {
191                                 /* abort(); */
192                                 goto exit_loop;
193                                 }
```



```
194                         else if (BIO_get_port(c->param_port,&c->port) <= 0)
195                                 goto exit_loop;
196                         c->state=BIO_CONN_S_CREATE_SOCKET;
197                         break;
198
199                 case BIO_CONN_S_CREATE_SOCKET:
200                         /* now setup address */
201                         memset((char *)&c->them,0,sizeof(c->them));
202                         c->them.sin_family=AF_INET;
203                         c->them.sin_port=htons((unsigned short)c->port);
204                         l=(unsigned long)
205                                 ((unsigned long)c->ip[0]<<24L)|
206                                 ((unsigned long)c->ip[1]<<16L)|
207                                 ((unsigned long)c->ip[2]<< 8L)|
208                                 ((unsigned long)c->ip[3]);
209                         c->them.sin_addr.s_addr=htonl(l);
210                         c->state=BIO_CONN_S_CREATE_SOCKET;
211
212                         ret=socket(AF_INET,SOCK_STREAM,SOCKET_PROTOCOL);
213                         if (ret == INVALID_SOCKET)
214                                 {
215                                 SYSerr(SYS_F_SOCKET,get_last_socket_error());
216                                 ERR_add_error_data(4,"host=",c->param_hostname,
217                                         ":",c->param_port);
218                                 BIOerr(BIO_F_CONN_STATE,BIO_R_UNABLE_TO_CREATE_S
219                                 goto exit_loop;
220                                 }
221                         b->num=ret;
222                         c->state=BIO_CONN_S_NBIO;
223                         break;
224
225                 case BIO_CONN_S_NBIO:
226                         if (c->nbio)
227                                 {
228                                 if (!BIO_socket_nbio(b->num,1))
229                                         {
230                                         BIOerr(BIO_F_CONN_STATE,BIO_R_ERROR_SETT
231                                         ERR_add_error_data(4,"host=",
232                                                 c->param_hostname,
233                                                 ":",c->param_port);
234                                         goto exit_loop;
235                                         }
236                                 }
237                         c->state=BIO_CONN_S_CONNECT;
238
239 #if defined(SO_KEEPALIVE) && !defined(OPENSSL_SYS_MPE)
240                         i=1;
241                         i=setsockopt(b->num,SOL_SOCKET,SO_KEEPALIVE,(char *)&i,s
242                         if (i < 0)
243                                 {
244                                 SYSerr(SYS_F_SOCKET,get_last_socket_error());
245                                 ERR_add_error_data(4,"host=",c->param_hostname,
246                                         ":",c->param_port);
247                                 BIOerr(BIO_F_CONN_STATE,BIO_R_KEEPALIVE);
248                                 goto exit_loop;
249                                 }
250 #endif
251                         break;
252
253                 case BIO_CONN_S_CONNECT:
254                         BIO_clear_retry_flags(b);
255                         ret=connect(b->num,
256                                 (struct sockaddr *)&c->them,
257                                 sizeof(c->them));
258                         b->retry_reason=0;
259                         if (ret < 0)
```

```
260                                 {
261                                 if (BIO_sock_should_retry(ret))
262                                         {
263                                         BIO_set_retry_special(b);
264                                         c->state=BIO_CONN_S_BLOCKED_CONNECT;
265                                         b->retry_reason=BIO_RR_CONNECT;
266                                         }
267                                 else
268                                         {
269                                         SYSerr(SYS_F_CONNECT,get_last_socket_err
270                                         ERR_add_error_data(4,"host=",
271                                                 c->param_hostname,
272                                                 ":",c->param_port);
273                                         BIOerr(BIO_F_CONN_STATE,BIO_R_CONNECT_ER
274                                         }
275                                 goto exit_loop;
276                                 }
277                         else
278                                 c->state=BIO_CONN_S_OK;
279                         break;

281                 case BIO_CONN_S_BLOCKED_CONNECT:
282                         i=BIO_sock_error(b->num);
283                         if (i)
284                                 {
285                                 BIO_clear_retry_flags(b);
286                                 SYSerr(SYS_F_CONNECT,i);
287                                 ERR_add_error_data(4,"host=",
288                                         c->param_hostname,
289                                         ":",c->param_port);
290                                 BIOerr(BIO_F_CONN_STATE,BIO_R_NBIO_CONNECT_ERROR
291                                 ret=0;
292                                 goto exit_loop;
293                                 }
294                         else
295                                 c->state=BIO_CONN_S_OK;
296                         break;

298                 case BIO_CONN_S_OK:
299                         ret=1;
300                         goto exit_loop;
301                 default:
302                         /* abort(); */
303                         goto exit_loop;
304                         }

306                 if (cb != NULL)
307                         {
308                         if (!(ret=cb((BIO *)b,c->state,ret)))
309                                 goto end;
310                         }
311                 }

313         /* Loop does not exit */
314 exit_loop:
315         if (cb != NULL)
316                 ret=cb((BIO *)b,c->state,ret);
317 end:
318         return(ret);
319         }

321 BIO_CONNECT *BIO_CONNECT_new(void)
322         {
323         BIO_CONNECT *ret;

325         if ((ret=(BIO_CONNECT *)OPENSSL_malloc(sizeof(BIO_CONNECT))) == NULL)
```



```
326                 return(NULL);
327         ret->state=BIO_CONN_S_BEFORE;
328         ret->param_hostname=NULL;
329         ret->param_port=NULL;
330         ret->info_callback=NULL;
331         ret->nbio=0;
332         ret->ip[0]=0;
333         ret->ip[1]=0;
334         ret->ip[2]=0;
335         ret->ip[3]=0;
336         ret->port=0;
337         memset((char *)&ret->them,0,sizeof(ret->them));
338         return(ret);
339         }

341 void BIO_CONNECT_free(BIO_CONNECT *a)
342         {
343         if(a == NULL)
344             return;

346         if (a->param_hostname != NULL)
347                 OPENSSL_free(a->param_hostname);
348         if (a->param_port != NULL)
349                 OPENSSL_free(a->param_port);
350         OPENSSL_free(a);
351         }

353 BIO_METHOD *BIO_s_connect(void)
354         {
355         return(&methods_connectp);
356         }

358 static int conn_new(BIO *bi)
359         {
360         bi->init=0;
361         bi->num=INVALID_SOCKET;
362         bi->flags=0;
363         if ((bi->ptr=(char *)BIO_CONNECT_new()) == NULL)
364                 return(0);
365         else
366                 return(1);
367         }

369 static void conn_close_socket(BIO *bio)
370         {
371         BIO_CONNECT *c;

373         c=(BIO_CONNECT *)bio->ptr;
374         if (bio->num != INVALID_SOCKET)
375                 {
376                 /* Only do a shutdown if things were established */
377                 if (c->state == BIO_CONN_S_OK)
378                         shutdown(bio->num,2);
379                 closesocket(bio->num);
380                 bio->num=INVALID_SOCKET;
381                 }
382         }

384 static int conn_free(BIO *a)
385         {
386         BIO_CONNECT *data;

388         if (a == NULL) return(0);
389         data=(BIO_CONNECT *)a->ptr;

391         if (a->shutdown)
```

```
392                 {
393                 conn_close_socket(a);
394                 BIO_CONNECT_free(data);
395                 a->ptr=NULL;
396                 a->flags=0;
397                 a->init=0;
398                 }
399         return(1);
400         }

402 static int conn_read(BIO *b, char *out, int outl)
403         {
404         int ret=0;
405         BIO_CONNECT *data;

407         data=(BIO_CONNECT *)b->ptr;
408         if (data->state != BIO_CONN_S_OK)
409                 {
410                 ret=conn_state(b,data);
411                 if (ret <= 0)
412                                 return(ret);
413                 }

415         if (out != NULL)
416                 {
417                 clear_socket_error();
418                 ret=readsocket(b->num,out,outl);
419                 BIO_clear_retry_flags(b);
420                 if (ret <= 0)
421                         {
422                         if (BIO_sock_should_retry(ret))
423                                 BIO_set_retry_read(b);
424                         }
425                 }
426         return(ret);
427         }

429 static int conn_write(BIO *b, const char *in, int inl)
430         {
431         int ret;
432         BIO_CONNECT *data;

434         data=(BIO_CONNECT *)b->ptr;
435         if (data->state != BIO_CONN_S_OK)
436                 {
437                 ret=conn_state(b,data);
438                 if (ret <= 0) return(ret);
439                 }

441         clear_socket_error();
442         ret=writesocket(b->num,in,inl);
443         BIO_clear_retry_flags(b);
444         if (ret <= 0)
445                 {
446                 if (BIO_sock_should_retry(ret))
447                         BIO_set_retry_write(b);
448                 }
449         return(ret);
450         }

452 static long conn_ctrl(BIO *b, int cmd, long num, void *ptr)
453         {
454         BIO *dbio;
455         int *ip;
456         const char **pptr;
457         long ret=1;
```



```
458         BIO_CONNECT *data;

460         data=(BIO_CONNECT *)b->ptr;

462         switch (cmd)
463                 {
464         case BIO_CTRL_RESET:
465                 ret=0;
466                 data->state=BIO_CONN_S_BEFORE;
467                 conn_close_socket(b);
468                 b->flags=0;
469                 break;
470         case BIO_C_DO_STATE_MACHINE:
471                 /* use this one to start the connection */
472                 if (data->state != BIO_CONN_S_OK)
473                         ret=(long)conn_state(b,data);
474                 else
475                         ret=1;
476                 break;
477         case BIO_C_GET_CONNECT:
478                 if (ptr != NULL)
479                         {
480                         pptr=(const char **)ptr;
481                         if (num == 0)
482                                 {
483                                 *pptr=data->param_hostname;

485                                 }
486                         else if (num == 1)
487                                 {
488                                 *pptr=data->param_port;
489                                 }
490                         else if (num == 2)
491                                 {
492                                 *pptr= (char *)&(data->ip[0]);
493                                 }
494                         else if (num == 3)
495                                 {
496                                 *((int *)ptr)=data->port;
497                                 }
498                         if ((!b->init) || (ptr == NULL))
499                                 *pptr="not initialized";
500                         ret=1;
501                         }
502                 break;
503         case BIO_C_SET_CONNECT:
504                 if (ptr != NULL)
505                         {
506                         b->init=1;
507                         if (num == 0)
508                                 {
509                                 if (data->param_hostname != NULL)
510                                         OPENSSL_free(data->param_hostname);
511                                 data->param_hostname=BUF_strdup(ptr);
512                                 }
513                         else if (num == 1)
514                                 {
515                                 if (data->param_port != NULL)
516                                         OPENSSL_free(data->param_port);
517                                 data->param_port=BUF_strdup(ptr);
518                                 }
519                         else if (num == 2)
520                                 {
521                                 char buf[16];
522                                 unsigned char *p = ptr;
```

```
524                                 BIO_snprintf(buf,sizeof buf,"%d.%d.%d.%d",
525                                         p[0],p[1],p[2],p[3]);
526                                 if (data->param_hostname != NULL)
527                                         OPENSSL_free(data->param_hostname);
528                                 data->param_hostname=BUF_strdup(buf);
529                                 memcpy(&(data->ip[0]),ptr,4);
530                                 }
531                         else if (num == 3)
532                                 {
533                                 char buf[DECIMAL_SIZE(int)+1];

535                                 BIO_snprintf(buf,sizeof buf,"%d",*(int *)ptr);
536                                 if (data->param_port != NULL)
537                                         OPENSSL_free(data->param_port);
538                                 data->param_port=BUF_strdup(buf);
539                                 data->port= *(int *)ptr;
540                                 }
541                         }
542                 break;
543         case BIO_C_SET_NBIO:
544                 data->nbio=(int)num;
545                 break;
546         case BIO_C_GET_FD:
547                 if (b->init)
548                         {
549                         ip=(int *)ptr;
550                         if (ip != NULL)
551                                 *ip=b->num;
552                         ret=b->num;
553                         }
554                 else
555                         ret= -1;
556                 break;
557         case BIO_CTRL_GET_CLOSE:
558                 ret=b->shutdown;
559                 break;
560         case BIO_CTRL_SET_CLOSE:
561                 b->shutdown=(int)num;
562                 break;
563         case BIO_CTRL_PENDING:
564         case BIO_CTRL_WPENDING:
565                 ret=0;
566                 break;
567         case BIO_CTRL_FLUSH:
568                 break;
569         case BIO_CTRL_DUP:
570                 {
571                 dbio=(BIO *)ptr;
572                 if (data->param_port)
573                         BIO_set_conn_port(dbio,data->param_port);
574                 if (data->param_hostname)
575                         BIO_set_conn_hostname(dbio,data->param_hostname);
576                 BIO_set_nbio(dbio,data->nbio);
577                 /* FIXME: the cast of the function seems unlikely to be a good i
578                 (void)BIO_set_info_callback(dbio,(bio_info_cb *)data->info_callb
579                 }
580                 break;
581         case BIO_CTRL_SET_CALLBACK:
582                 {
583 #if 0 /* FIXME: Should this be used?  -- Richard Levitte */
584                 BIOerr(BIO_F_CONN_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
585                 ret = -1;
586 #else
587                 ret=0;
588 #endif
589                 }
```



```
590                 break;
591         case BIO_CTRL_GET_CALLBACK:
592                 {
593                 int (**fptr)(const BIO *bio,int state,int xret);

595                 fptr=(int (**)(const BIO *bio,int state,int xret))ptr;
596                 *fptr=data->info_callback;
597                 }
598                 break;
599         default:
600                 ret=0;
601                 break;
602                 }
603         return(ret);
604         }

606 static long conn_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
607         {
608         long ret=1;
609         BIO_CONNECT *data;

611         data=(BIO_CONNECT *)b->ptr;

613         switch (cmd)
614                 {
615         case BIO_CTRL_SET_CALLBACK:
616                 {
617                 data->info_callback=(int (*)(const struct bio_st *, int, int))fp
618                 }
619                 break;
620         default:
621                 ret=0;
622                 break;
623                 }
624         return(ret);
625         }

627 static int conn_puts(BIO *bp, const char *str)
628         {
629         int n,ret;

631         n=strlen(str);
632         ret=conn_write(bp,str,n);
633         return(ret);
634         }

636 BIO *BIO_new_connect(char *str)
637         {
638         BIO *ret;

640         ret=BIO_new(BIO_s_connect());
641         if (ret == NULL) return(NULL);
642         if (BIO_set_conn_hostname(ret,str))
643                 return(ret);
644         else
645                 {
646                 BIO_free(ret);
647                 return(NULL);
648                 }
649         }

651 #endif
652 #endif /* ! codereview */
```

```
**********************************************************
   47536 Wed Aug 13 19:52:12 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_dgram.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bio_dgram.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
```



```
#include <errno.h>
#define USE_SOCKETS
#include "cryptlib.h"

#include <openssl/bio.h>
#ifndef OPENSSL_NO_DGRAM

#if defined(OPENSSL_SYS_WIN32) || defined(OPENSSL_SYS_VMS)
#include <sys/timeb.h>
#endif

#ifndef OPENSSL_NO_SCTP
#include <netinet/sctp.h>
#include <fcntl.h>
#define OPENSSL_SCTP_DATA_CHUNK_TYPE            0x00
#define OPENSSL_SCTP_FORWARD_CUM_TSN_CHUNK_TYPE 0xc0
#endif

#if defined(OPENSSL_SYS_LINUX) && !defined(IP_MTU)
#define IP_MTU      14 /* linux is lame */
#endif

#if defined(__FreeBSD__) && defined(IN6_IS_ADDR_V4MAPPED)
/* Standard definition causes type-punning problems. */
#undef IN6_IS_ADDR_V4MAPPED
#define s6_addr32 __u6_addr.__u6_addr32
#define IN6_IS_ADDR_V4MAPPED(a)                 \
        (((a)->s6_addr32[0] == 0) &&            \
         ((a)->s6_addr32[1] == 0) &&            \
         ((a)->s6_addr32[2] == htonl(0x0000ffff)))
#endif

#ifdef WATT32
#define sock_write SockWrite  /* Watt-32 uses same names */
#define sock_read  SockRead
#define sock_puts  SockPuts
#endif

static int dgram_write(BIO *h, const char *buf, int num);
static int dgram_read(BIO *h, char *buf, int size);
static int dgram_puts(BIO *h, const char *str);
static long dgram_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int dgram_new(BIO *h);
static int dgram_free(BIO *data);
static int dgram_clear(BIO *bio);

#ifndef OPENSSL_NO_SCTP
static int dgram_sctp_write(BIO *h, const char *buf, int num);
static int dgram_sctp_read(BIO *h, char *buf, int size);
static int dgram_sctp_puts(BIO *h, const char *str);
static long dgram_sctp_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int dgram_sctp_new(BIO *h);
static int dgram_sctp_free(BIO *data);
#ifdef SCTP_AUTHENTICATION_EVENT
static void dgram_sctp_handle_auth_free_key_event(BIO *b, union sctp_notificatio
#endif
#endif

static int BIO_dgram_should_retry(int s);

static void get_current_time(struct timeval *t);

static BIO_METHOD methods_dgramp=
        {
        BIO_TYPE_DGRAM,
        "datagram socket",
```

```
128         dgram_write,
129         dgram_read,
130         dgram_puts,
131         NULL, /* dgram_gets, */
132         dgram_ctrl,
133         dgram_new,
134         dgram_free,
135         NULL,
136         };

138 #ifndef OPENSSL_NO_SCTP
139 static BIO_METHOD methods_dgramp_sctp=
140         {
141         BIO_TYPE_DGRAM_SCTP,
142         "datagram sctp socket",
143         dgram_sctp_write,
144         dgram_sctp_read,
145         dgram_sctp_puts,
146         NULL, /* dgram_gets, */
147         dgram_sctp_ctrl,
148         dgram_sctp_new,
149         dgram_sctp_free,
150         NULL,
151         };
152 #endif

154 typedef struct bio_dgram_data_st
155         {
156         union {
157                 struct sockaddr sa;
158                 struct sockaddr_in sa_in;
159 #if OPENSSL_USE_IPV6
160                 struct sockaddr_in6 sa_in6;
161 #endif
162         } peer;
163         unsigned int connected;
164         unsigned int _errno;
165         unsigned int mtu;
166         struct timeval next_timeout;
167         struct timeval socket_timeout;
168         } bio_dgram_data;

170 #ifndef OPENSSL_NO_SCTP
171 typedef struct bio_dgram_sctp_save_message_st
172         {
173         BIO *bio;
174         char *data;
175         int length;
176         } bio_dgram_sctp_save_message;

178 typedef struct bio_dgram_sctp_data_st
179         {
180         union {
181                 struct sockaddr sa;
182                 struct sockaddr_in sa_in;
183 #if OPENSSL_USE_IPV6
184                 struct sockaddr_in6 sa_in6;
185 #endif
186         } peer;
187         unsigned int connected;
188         unsigned int _errno;
189         unsigned int mtu;
190         struct bio_dgram_sctp_sndinfo sndinfo;
191         struct bio_dgram_sctp_rcvinfo rcvinfo;
192         struct bio_dgram_sctp_prinfo prinfo;
193         void (*handle_notifications)(BIO *bio, void *context, void *buf);
```



```
194         void* notification_context;
195         int in_handshake;
196         int ccs_rcvd;
197         int ccs_sent;
198         int save_shutdown;
199         int peer_auth_tested;
200         bio_dgram_sctp_save_message saved_message;
201         } bio_dgram_sctp_data;
202 #endif

204 BIO_METHOD *BIO_s_datagram(void)
205         {
206         return(&methods_dgramp);
207         }

209 BIO *BIO_new_dgram(int fd, int close_flag)
210         {
211         BIO *ret;

213         ret=BIO_new(BIO_s_datagram());
214         if (ret == NULL) return(NULL);
215         BIO_set_fd(ret,fd,close_flag);
216         return(ret);
217         }

219 static int dgram_new(BIO *bi)
220         {
221         bio_dgram_data *data = NULL;

223         bi->init=0;
224         bi->num=0;
225         data = OPENSSL_malloc(sizeof(bio_dgram_data));
226         if (data == NULL)
227                 return 0;
228         memset(data, 0x00, sizeof(bio_dgram_data));
229     bi->ptr = data;

231         bi->flags=0;
232         return(1);
233         }

235 static int dgram_free(BIO *a)
236         {
237         bio_dgram_data *data;

239         if (a == NULL) return(0);
240         if ( ! dgram_clear(a))
241                 return 0;

243         data = (bio_dgram_data *)a->ptr;
244         if(data != NULL) OPENSSL_free(data);

246         return(1);
247         }

249 static int dgram_clear(BIO *a)
250         {
251         if (a == NULL) return(0);
252         if (a->shutdown)
253                 {
254                 if (a->init)
255                         {
256                         SHUTDOWN2(a->num);
257                         }
258                 a->init=0;
259                 a->flags=0;
```

```
260                 }
261         return(1);
262         }

264 static void dgram_adjust_rcv_timeout(BIO *b)
265         {
266 #if defined(SO_RCVTIMEO)
267         bio_dgram_data *data = (bio_dgram_data *)b->ptr;
268         union { size_t s; int i; } sz = {0};

270         /* Is a timer active? */
271         if (data->next_timeout.tv_sec > 0 || data->next_timeout.tv_usec > 0)
272                 {
273                 struct timeval timenow, timeleft;

275                 /* Read current socket timeout */
276 #ifdef OPENSSL_SYS_WINDOWS
277                 int timeout;

279                 sz.i = sizeof(timeout);
280                 if (getsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO,
281                                        (void*)&timeout, &sz.i) < 0)
282                         { perror("getsockopt"); }
283                 else
284                         {
285                         data->socket_timeout.tv_sec = timeout / 1000;
286                         data->socket_timeout.tv_usec = (timeout % 1000) * 1000;
287                         }
288 #else
289                 sz.i = sizeof(data->socket_timeout);
290                 if ( getsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO,
291                                         &(data->socket_timeout), (void *
292                         { perror("getsockopt"); }
293                 else if (sizeof(sz.s)!=sizeof(sz.i) && sz.i==0)
294                         OPENSSL_assert(sz.s<=sizeof(data->socket_timeout));
295 #endif

297                 /* Get current time */
298                 get_current_time(&timenow);

300                 /* Calculate time left until timer expires */
301                 memcpy(&timeleft, &(data->next_timeout), sizeof(struct timeval))
302                 timeleft.tv_sec -= timenow.tv_sec;
303                 timeleft.tv_usec -= timenow.tv_usec;
304                 if (timeleft.tv_usec < 0)
305                         {
306                         timeleft.tv_sec--;
307                         timeleft.tv_usec += 1000000;
308                         }

310                 if (timeleft.tv_sec < 0)
311                         {
312                         timeleft.tv_sec = 0;
313                         timeleft.tv_usec = 1;
314                         }

316                 /* Adjust socket timeout if next handhake message timer
317                  * will expire earlier.
318                  */
319                 if ((data->socket_timeout.tv_sec == 0 && data->socket_timeout.tv
320                         (data->socket_timeout.tv_sec > timeleft.tv_sec) ||
321                         (data->socket_timeout.tv_sec == timeleft.tv_sec &&
322                          data->socket_timeout.tv_usec >= timeleft.tv_usec))
323                         {
324 #ifdef OPENSSL_SYS_WINDOWS
325                         timeout = timeleft.tv_sec * 1000 + timeleft.tv_usec / 10
```



```
326                         if (setsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO,
327                                                (void*)&timeout, sizeof(timeo
328                                 { perror("setsockopt"); }
329 #else
330                         if ( setsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO, &timele
331                                                sizeof(struct timeval))
332                                 { perror("setsockopt"); }
333 #endif
334                         }
335                 }
336 #endif
337         }

339 static void dgram_reset_rcv_timeout(BIO *b)
340         {
341 #if defined(SO_RCVTIMEO)
342         bio_dgram_data *data = (bio_dgram_data *)b->ptr;

344         /* Is a timer active? */
345         if (data->next_timeout.tv_sec > 0 || data->next_timeout.tv_usec > 0)
346                 {
347 #ifdef OPENSSL_SYS_WINDOWS
348                 int timeout = data->socket_timeout.tv_sec * 1000 +
349                                                 data->socket_timeout.tv_usec / 1000;
350                 if (setsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO,
351                                                 (void*)&timeout, sizeof(timeout)) < 0
352                         { perror("setsockopt"); }
353 #else
354                 if ( setsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO, &(data->socket_
355                                                 sizeof(struct timeval)) < 0)
356                         { perror("setsockopt"); }
357 #endif
358                 }
359 #endif
360         }

362 static int dgram_read(BIO *b, char *out, int outl)
363         {
364         int ret=0;
365         bio_dgram_data *data = (bio_dgram_data *)b->ptr;

367         struct  {
368         /*
369          * See commentary in b_sock.c. <appro>
370          */
371         union   { size_t s; int i; } len;
372         union   {
373                 struct sockaddr sa;
374                 struct sockaddr_in sa_in;
375 #if OPENSSL_USE_IPV6
376                 struct sockaddr_in6 sa_in6;
377 #endif
378                 } peer;
379         } sa;

381         sa.len.s=0;
382         sa.len.i=sizeof(sa.peer);

384         if (out != NULL)
385                 {
386                 clear_socket_error();
387                 memset(&sa.peer, 0x00, sizeof(sa.peer));
388                 dgram_adjust_rcv_timeout(b);
389                 ret=recvfrom(b->num,out,outl,0,&sa.peer.sa,(void *)&sa.len);
390                 if (sizeof(sa.len.i)!=sizeof(sa.len.s) && sa.len.i==0)
391                         {
```

```
392                         OPENSSL_assert(sa.len.s<=sizeof(sa.peer));
393                         sa.len.i = (int)sa.len.s;
394                         }

396                 if ( ! data->connected  && ret >= 0)
397                         BIO_ctrl(b, BIO_CTRL_DGRAM_SET_PEER, 0, &sa.peer);

399                 BIO_clear_retry_flags(b);
400                 if (ret < 0)
401                         {
402                         if (BIO_dgram_should_retry(ret))
403                                 {
404                                 BIO_set_retry_read(b);
405                                 data->_errno = get_last_socket_error();
406                                 }
407                         }

409                 dgram_reset_rcv_timeout(b);
410                 }
411         return(ret);
412         }

414 static int dgram_write(BIO *b, const char *in, int inl)
415         {
416         int ret;
417         bio_dgram_data *data = (bio_dgram_data *)b->ptr;
418         clear_socket_error();

420         if ( data->connected )
421                 ret=writesocket(b->num,in,inl);
422         else
423                 {
424                 int peerlen = sizeof(data->peer);

426                 if (data->peer.sa.sa_family == AF_INET)
427                         peerlen = sizeof(data->peer.sa_in);
428 #if OPENSSL_USE_IPV6
429                 else if (data->peer.sa.sa_family == AF_INET6)
430                         peerlen = sizeof(data->peer.sa_in6);
431 #endif
432 #if defined(NETWARE_CLIB) && defined(NETWARE_BSDSOCK)
433                 ret=sendto(b->num, (char *)in, inl, 0, &data->peer.sa, peerlen);
434 #else
435                 ret=sendto(b->num, in, inl, 0, &data->peer.sa, peerlen);
436 #endif
437                 }

439         BIO_clear_retry_flags(b);
440         if (ret <= 0)
441                 {
442                 if (BIO_dgram_should_retry(ret))
443                         {
444                         BIO_set_retry_write(b);
445                         data->_errno = get_last_socket_error();

447 #if 0 /* higher layers are responsible for querying MTU, if necessary */
448                         if ( data->_errno == EMSGSIZE)
449                                 /* retrieve the new MTU */
450                                 BIO_ctrl(b, BIO_CTRL_DGRAM_QUERY_MTU, 0, NULL);
451 #endif
452                         }
453                 }
454         return(ret);
455         }

457 static long dgram_ctrl(BIO *b, int cmd, long num, void *ptr)
```



```
458         {
459         long ret=1;
460         int *ip;
461         struct sockaddr *to = NULL;
462         bio_dgram_data *data = NULL;
463 #if defined(OPENSSL_SYS_LINUX) && (defined(IP_MTU_DISCOVER) || defined(IP_MTU))
464         int sockopt_val = 0;
465         socklen_t sockopt_len;  /* assume that system supporting IP_MTU is
466                                  * modern enough to define socklen_t */
467         socklen_t addr_len;
468         union   {
469                 struct sockaddr sa;
470                 struct sockaddr_in s4;
471 #if OPENSSL_USE_IPV6
472                 struct sockaddr_in6 s6;
473 #endif
474                 } addr;
475 #endif

477         data = (bio_dgram_data *)b->ptr;

479         switch (cmd)
480                 {
481         case BIO_CTRL_RESET:
482                 num=0;
483         case BIO_C_FILE_SEEK:
484                 ret=0;
485                 break;
486         case BIO_C_FILE_TELL:
487         case BIO_CTRL_INFO:
488                 ret=0;
489                 break;
490         case BIO_C_SET_FD:
491                 dgram_clear(b);
492                 b->num= *((int *)ptr);
493                 b->shutdown=(int)num;
494                 b->init=1;
495                 break;
496         case BIO_C_GET_FD:
497                 if (b->init)
498                         {
499                         ip=(int *)ptr;
500                         if (ip != NULL) *ip=b->num;
501                         ret=b->num;
502                         }
503                 else
504                         ret= -1;
505                 break;
506         case BIO_CTRL_GET_CLOSE:
507                 ret=b->shutdown;
508                 break;
509         case BIO_CTRL_SET_CLOSE:
510                 b->shutdown=(int)num;
511                 break;
512         case BIO_CTRL_PENDING:
513         case BIO_CTRL_WPENDING:
514                 ret=0;
515                 break;
516         case BIO_CTRL_DUP:
517         case BIO_CTRL_FLUSH:
518                 ret=1;
519                 break;
520         case BIO_CTRL_DGRAM_CONNECT:
521                 to = (struct sockaddr *)ptr;
522 #if 0
523                 if (connect(b->num, to, sizeof(struct sockaddr)) < 0)
```

```
524                         { perror("connect"); ret = 0; }
525                 else
526                         {
527 #endif
528                         switch (to->sa_family)
529                                 {
530                                 case AF_INET:
531                                         memcpy(&data->peer,to,sizeof(data->peer.
532                                         break;
533 #if OPENSSL_USE_IPV6
534                                 case AF_INET6:
535                                         memcpy(&data->peer,to,sizeof(data->peer.
536                                         break;
537 #endif
538                                 default:
539                                         memcpy(&data->peer,to,sizeof(data->peer.
540                                         break;
541                                 }
542 #if 0
543                         }
544 #endif
545                 break;
546                 /* (Linux)kernel sets DF bit on outgoing IP packets */
547         case BIO_CTRL_DGRAM_MTU_DISCOVER:
548 #if defined(OPENSSL_SYS_LINUX) && defined(IP_MTU_DISCOVER) && defined(IP_PMTUDIS
549                 addr_len = (socklen_t)sizeof(addr);
550                 memset((void *)&addr, 0, sizeof(addr));
551                 if (getsockname(b->num, &addr.sa, &addr_len) < 0)
552                         {
553                         ret = 0;
554                         break;
555                         }
556                 switch (addr.sa.sa_family)
557                         {
558                 case AF_INET:
559                         sockopt_val = IP_PMTUDISC_DO;
560                         if ((ret = setsockopt(b->num, IPPROTO_IP, IP_MTU_DISCOVE
561                                 &sockopt_val, sizeof(sockopt_val))) < 0)
562                                 perror("setsockopt");
563                         break;
564 #if OPENSSL_USE_IPV6 && defined(IPV6_MTU_DISCOVER) && defined(IPV6_PMTUDISC_DO)
565                 case AF_INET6:
566                         sockopt_val = IPV6_PMTUDISC_DO;
567                         if ((ret = setsockopt(b->num, IPPROTO_IPV6, IPV6_MTU_DIS
568                                 &sockopt_val, sizeof(sockopt_val))) < 0)
569                                 perror("setsockopt");
570                         break;
571 #endif
572                 default:
573                         ret = -1;
574                         break;
575                         }
576                 ret = -1;
577 #else
578                 break;
579 #endif
580         case BIO_CTRL_DGRAM_QUERY_MTU:
581 #if defined(OPENSSL_SYS_LINUX) && defined(IP_MTU)
582                 addr_len = (socklen_t)sizeof(addr);
583                 memset((void *)&addr, 0, sizeof(addr));
584                 if (getsockname(b->num, &addr.sa, &addr_len) < 0)
585                         {
586                         ret = 0;
587                         break;
588                         }
589                 sockopt_len = sizeof(sockopt_val);
```



```
590                 switch (addr.sa.sa_family)
591                         {
592                 case AF_INET:
593                         if ((ret = getsockopt(b->num, IPPROTO_IP, IP_MTU, (void
594                                 &sockopt_len)) < 0 || sockopt_val < 0)
595                                 {
596                                 ret = 0;
597                                 }
598                         else
599                                 {
600                                 /* we assume that the transport protocol is UDP
601                                  * IP options are used.
602                                  */
603                                 data->mtu = sockopt_val - 8 - 20;
604                                 ret = data->mtu;
605                                 }
606                         break;
607 #if OPENSSL_USE_IPV6 && defined(IPV6_MTU)
608                 case AF_INET6:
609                         if ((ret = getsockopt(b->num, IPPROTO_IPV6, IPV6_MTU, (v
610                                 &sockopt_len)) < 0 || sockopt_val < 0)
611                                 {
612                                 ret = 0;
613                                 }
614                         else
615                                 {
616                                 /* we assume that the transport protocol is UDP
617                                  * IPV6 options are used.
618                                  */
619                                 data->mtu = sockopt_val - 8 - 40;
620                                 ret = data->mtu;
621                                 }
622                         break;
623 #endif
624                 default:
625                         ret = 0;
626                         break;
627                         }
628 #else
629                 ret = 0;
630 #endif
631                 break;
632         case BIO_CTRL_DGRAM_GET_FALLBACK_MTU:
633                 switch (data->peer.sa.sa_family)
634                         {
635                         case AF_INET:
636                                 ret = 576 - 20 - 8;
637                                 break;
638 #if OPENSSL_USE_IPV6
639                         case AF_INET6:
640 #ifdef IN6_IS_ADDR_V4MAPPED
641                                 if (IN6_IS_ADDR_V4MAPPED(&data->peer.sa_in6.sin6
642                                         ret = 576 - 20 - 8;
643                                 else
644 #endif
645                                         ret = 1280 - 40 - 8;
646                                 break;
647 #endif
648                         default:
649                                 ret = 576 - 20 - 8;
650                                 break;
651                         }
652                 break;
653         case BIO_CTRL_DGRAM_GET_MTU:
654                 return data->mtu;
655                 break;
```

```
656         case BIO_CTRL_DGRAM_SET_MTU:
657                 data->mtu = num;
658                 ret = num;
659                 break;
660         case BIO_CTRL_DGRAM_SET_CONNECTED:
661                 to = (struct sockaddr *)ptr;

663                 if ( to != NULL)
664                         {
665                         data->connected = 1;
666                         switch (to->sa_family)
667                                 {
668                                 case AF_INET:
669                                         memcpy(&data->peer,to,sizeof(data->peer.
670                                         break;
671 #if OPENSSL_USE_IPV6
672                                 case AF_INET6:
673                                         memcpy(&data->peer,to,sizeof(data->peer.
674                                         break;
675 #endif
676                                 default:
677                                         memcpy(&data->peer,to,sizeof(data->peer.
678                                         break;
679                                 }
680                         }
681                 else
682                         {
683                         data->connected = 0;
684                         memset(&(data->peer), 0x00, sizeof(data->peer));
685                         }
686                 break;
687         case BIO_CTRL_DGRAM_GET_PEER:
688                 switch (data->peer.sa.sa_family)
689                         {
690                         case AF_INET:
691                                 ret=sizeof(data->peer.sa_in);
692                                 break;
693 #if OPENSSL_USE_IPV6
694                         case AF_INET6:
695                                 ret=sizeof(data->peer.sa_in6);
696                                 break;
697 #endif
698                         default:
699                                 ret=sizeof(data->peer.sa);
700                                 break;
701                         }
702                 if (num==0 || num>ret)
703                         num=ret;
704                 memcpy(ptr,&data->peer,(ret=num));
705                 break;
706         case BIO_CTRL_DGRAM_SET_PEER:
707                 to = (struct sockaddr *) ptr;
708                 switch (to->sa_family)
709                         {
710                         case AF_INET:
711                                 memcpy(&data->peer,to,sizeof(data->peer.sa_in));
712                                 break;
713 #if OPENSSL_USE_IPV6
714                         case AF_INET6:
715                                 memcpy(&data->peer,to,sizeof(data->peer.sa_in6))
716                                 break;
717 #endif
718                         default:
719                                 memcpy(&data->peer,to,sizeof(data->peer.sa));
720                                 break;
721                         }
```



```
722                 break;
723         case BIO_CTRL_DGRAM_SET_NEXT_TIMEOUT:
724                 memcpy(&(data->next_timeout), ptr, sizeof(struct timeval));
725                 break;
726 #if defined(SO_RCVTIMEO)
727         case BIO_CTRL_DGRAM_SET_RECV_TIMEOUT:
728 #ifdef OPENSSL_SYS_WINDOWS
729                 {
730                 struct timeval *tv = (struct timeval *)ptr;
731                 int timeout = tv->tv_sec * 1000 + tv->tv_usec/1000;
732                 if (setsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO,
733                         (void*)&timeout, sizeof(timeout)) < 0)
734                         { perror("setsockopt"); ret = -1; }
735                 }
736 #else
737                 if ( setsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO, ptr,
738                         sizeof(struct timeval)) < 0)
739                         { perror("setsockopt"); ret = -1; }
740 #endif
741                 break;
742         case BIO_CTRL_DGRAM_GET_RECV_TIMEOUT:
743                 {
744                 union { size_t s; int i; } sz = {0};
745 #ifdef OPENSSL_SYS_WINDOWS
746                 int timeout;
747                 struct timeval *tv = (struct timeval *)ptr;

749                 sz.i = sizeof(timeout);
750                 if (getsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO,
751                         (void*)&timeout, &sz.i) < 0)
752                         { perror("getsockopt"); ret = -1; }
753                 else
754                         {
755                         tv->tv_sec = timeout / 1000;
756                         tv->tv_usec = (timeout % 1000) * 1000;
757                         ret = sizeof(*tv);
758                         }
759 #else
760                 sz.i = sizeof(struct timeval);
761                 if ( getsockopt(b->num, SOL_SOCKET, SO_RCVTIMEO,
762                         ptr, (void *)&sz) < 0)
763                         { perror("getsockopt"); ret = -1; }
764                 else if (sizeof(sz.s)!=sizeof(sz.i) && sz.i==0)
765                         {
766                         OPENSSL_assert(sz.s<=sizeof(struct timeval));
767                         ret = (int)sz.s;
768                         }
769                 else
770                         ret = sz.i;
771 #endif
772                 }
773                 break;
774 #endif
775 #if defined(SO_SNDTIMEO)
776         case BIO_CTRL_DGRAM_SET_SEND_TIMEOUT:
777 #ifdef OPENSSL_SYS_WINDOWS
778                 {
779                 struct timeval *tv = (struct timeval *)ptr;
780                 int timeout = tv->tv_sec * 1000 + tv->tv_usec/1000;
781                 if (setsockopt(b->num, SOL_SOCKET, SO_SNDTIMEO,
782                         (void*)&timeout, sizeof(timeout)) < 0)
783                         { perror("setsockopt"); ret = -1; }
784                 }
785 #else
786                 if ( setsockopt(b->num, SOL_SOCKET, SO_SNDTIMEO, ptr,
787                         sizeof(struct timeval)) < 0)
```

```
788                         { perror("setsockopt"); ret = -1; }
789 #endif
790                 break;
791         case BIO_CTRL_DGRAM_GET_SEND_TIMEOUT:
792                 {
793                 union { size_t s; int i; } sz = {0};
794 #ifdef OPENSSL_SYS_WINDOWS
795                 int timeout;
796                 struct timeval *tv = (struct timeval *)ptr;

798                 sz.i = sizeof(timeout);
799                 if (getsockopt(b->num, SOL_SOCKET, SO_SNDTIMEO,
800                         (void*)&timeout, &sz.i) < 0)
801                         { perror("getsockopt"); ret = -1; }
802                 else
803                         {
804                         tv->tv_sec = timeout / 1000;
805                         tv->tv_usec = (timeout % 1000) * 1000;
806                         ret = sizeof(*tv);
807                         }
808 #else
809                 sz.i = sizeof(struct timeval);
810                 if ( getsockopt(b->num, SOL_SOCKET, SO_SNDTIMEO,
811                         ptr, (void *)&sz) < 0)
812                         { perror("getsockopt"); ret = -1; }
813                 else if (sizeof(sz.s)!=sizeof(sz.i) && sz.i==0)
814                         {
815                         OPENSSL_assert(sz.s<=sizeof(struct timeval));
816                         ret = (int)sz.s;
817                         }
818                 else
819                         ret = sz.i;
820 #endif
821                 }
822                 break;
823 #endif
824         case BIO_CTRL_DGRAM_GET_SEND_TIMER_EXP:
825                 /* fall-through */
826         case BIO_CTRL_DGRAM_GET_RECV_TIMER_EXP:
827 #ifdef OPENSSL_SYS_WINDOWS
828                 if ( data->_errno == WSAETIMEDOUT)
829 #else
830                 if ( data->_errno == EAGAIN)
831 #endif
832                         {
833                         ret = 1;
834                         data->_errno = 0;
835                         }
836                 else
837                         ret = 0;
838                 break;
839 #ifdef EMSGSIZE
840         case BIO_CTRL_DGRAM_MTU_EXCEEDED:
841                 if ( data->_errno == EMSGSIZE)
842                         {
843                         ret = 1;
844                         data->_errno = 0;
845                         }
846                 else
847                         ret = 0;
848                 break;
849 #endif
850         default:
851                 ret=0;
852                 break;
853                 }
```



```
854         return(ret);
855         }

857 static int dgram_puts(BIO *bp, const char *str)
858         {
859         int n,ret;

861         n=strlen(str);
862         ret=dgram_write(bp,str,n);
863         return(ret);
864         }

866 #ifndef OPENSSL_NO_SCTP
867 BIO_METHOD *BIO_s_datagram_sctp(void)
868         {
869         return(&methods_dgramp_sctp);
870         }

872 BIO *BIO_new_dgram_sctp(int fd, int close_flag)
873         {
874         BIO *bio;
875         int ret, optval = 20000;
876         int auth_data = 0, auth_forward = 0;
877         unsigned char *p;
878         struct sctp_authchunk auth;
879         struct sctp_authchunks *authchunks;
880         socklen_t sockopt_len;
881 #ifdef SCTP_AUTHENTICATION_EVENT
882 #ifdef SCTP_EVENT
883         struct sctp_event event;
884 #else
885         struct sctp_event_subscribe event;
886 #endif
887 #endif

889         bio=BIO_new(BIO_s_datagram_sctp());
890         if (bio == NULL) return(NULL);
891         BIO_set_fd(bio,fd,close_flag);

893         /* Activate SCTP-AUTH for DATA and FORWARD-TSN chunks */
894         auth.sauth_chunk = OPENSSL_SCTP_DATA_CHUNK_TYPE;
895         ret = setsockopt(fd, IPPROTO_SCTP, SCTP_AUTH_CHUNK, &auth, sizeof(struct
896         OPENSSL_assert(ret >= 0);
897         auth.sauth_chunk = OPENSSL_SCTP_FORWARD_CUM_TSN_CHUNK_TYPE;
898         ret = setsockopt(fd, IPPROTO_SCTP, SCTP_AUTH_CHUNK, &auth, sizeof(struct
899         OPENSSL_assert(ret >= 0);

901         /* Test if activation was successful. When using accept(),
902          * SCTP-AUTH has to be activated for the listening socket
903          * already, otherwise the connected socket won't use it. */
904         sockopt_len = (socklen_t)(sizeof(sctp_assoc_t) + 256 * sizeof(uint8_t));
905         authchunks = OPENSSL_malloc(sockopt_len);
906         memset(authchunks, 0, sizeof(sockopt_len));
907         ret = getsockopt(fd, IPPROTO_SCTP, SCTP_LOCAL_AUTH_CHUNKS, authchunks, &
908         OPENSSL_assert(ret >= 0);

910         for (p = (unsigned char*) authchunks->gauth_chunks;
911              p < (unsigned char*) authchunks + sockopt_len;
912              p += sizeof(uint8_t))
913                 {
914                 if (*p == OPENSSL_SCTP_DATA_CHUNK_TYPE) auth_data = 1;
915                 if (*p == OPENSSL_SCTP_FORWARD_CUM_TSN_CHUNK_TYPE) auth_forward
916                 }

918         OPENSSL_free(authchunks);
```

```
920         OPENSSL_assert(auth_data);
921         OPENSSL_assert(auth_forward);

923 #ifdef SCTP_AUTHENTICATION_EVENT
924 #ifdef SCTP_EVENT
925         memset(&event, 0, sizeof(struct sctp_event));
926         event.se_assoc_id = 0;
927         event.se_type = SCTP_AUTHENTICATION_EVENT;
928         event.se_on = 1;
929         ret = setsockopt(fd, IPPROTO_SCTP, SCTP_EVENT, &event, sizeof(struct sct
930         OPENSSL_assert(ret >= 0);
931 #else
932         sockopt_len = (socklen_t) sizeof(struct sctp_event_subscribe);
933         ret = getsockopt(fd, IPPROTO_SCTP, SCTP_EVENTS, &event, &sockopt_len);
934         OPENSSL_assert(ret >= 0);

936         event.sctp_authentication_event = 1;

938         ret = setsockopt(fd, IPPROTO_SCTP, SCTP_EVENTS, &event, sizeof(struct sc
939         OPENSSL_assert(ret >= 0);
940 #endif
941 #endif

943         /* Disable partial delivery by setting the min size
944          * larger than the max record size of 2^14 + 2048 + 13
945          */
946         ret = setsockopt(fd, IPPROTO_SCTP, SCTP_PARTIAL_DELIVERY_POINT, &optval,
947         OPENSSL_assert(ret >= 0);

949         return(bio);
950         }

952 int BIO_dgram_is_sctp(BIO *bio)
953         {
954         return (BIO_method_type(bio) == BIO_TYPE_DGRAM_SCTP);
955         }

957 static int dgram_sctp_new(BIO *bi)
958         {
959         bio_dgram_sctp_data *data = NULL;

961         bi->init=0;
962         bi->num=0;
963         data = OPENSSL_malloc(sizeof(bio_dgram_sctp_data));
964         if (data == NULL)
965                 return 0;
966         memset(data, 0x00, sizeof(bio_dgram_sctp_data));
967 #ifdef SCTP_PR_SCTP_NONE
968         data->prinfo.pr_policy = SCTP_PR_SCTP_NONE;
969 #endif
970     bi->ptr = data;

972         bi->flags=0;
973         return(1);
974         }

976 static int dgram_sctp_free(BIO *a)
977         {
978         bio_dgram_sctp_data *data;

980         if (a == NULL) return(0);
981         if ( ! dgram_clear(a))
982                 return 0;

984         data = (bio_dgram_sctp_data *)a->ptr;
985         if(data != NULL) OPENSSL_free(data);
```



```
987         return(1);
988         }

990 #ifdef SCTP_AUTHENTICATION_EVENT
991 void dgram_sctp_handle_auth_free_key_event(BIO *b, union sctp_notification *snp)
992         {
993         int ret;
994         struct sctp_authkey_event* authkeyevent = &snp->sn_auth_event;

996         if (authkeyevent->auth_indication == SCTP_AUTH_FREE_KEY)
997                 {
998                 struct sctp_authkeyid authkeyid;

1000                /* delete key */
1001                authkeyid.scact_keynumber = authkeyevent->auth_keynumber;
1002                ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_DELETE_KEY,
1003                     &authkeyid, sizeof(struct sctp_authkeyid));
1004                }
1005        }
1006 #endif

1008 static int dgram_sctp_read(BIO *b, char *out, int outl)
1009        {
1010        int ret = 0, n = 0, i, optval;
1011        socklen_t optlen;
1012        bio_dgram_sctp_data *data = (bio_dgram_sctp_data *)b->ptr;
1013        union sctp_notification *snp;
1014        struct msghdr msg;
1015        struct iovec iov;
1016        struct cmsghdr *cmsg;
1017        char cmsgbuf[512];

1019        if (out != NULL)
1020                {
1021                clear_socket_error();

1023                do
1024                        {
1025                        memset(&data->rcvinfo, 0x00, sizeof(struct bio_dgram_sct
1026                        iov.iov_base = out;
1027                        iov.iov_len = outl;
1028                        msg.msg_name = NULL;
1029                        msg.msg_namelen = 0;
1030                        msg.msg_iov = &iov;
1031                        msg.msg_iovlen = 1;
1032                        msg.msg_control = cmsgbuf;
1033                        msg.msg_controllen = 512;
1034                        msg.msg_flags = 0;
1035                        n = recvmsg(b->num, &msg, 0);

1037                        if (msg.msg_controllen > 0)
1038                                {
1039                                for (cmsg = CMSG_FIRSTHDR(&msg); cmsg; cmsg = CM
1040                                        {
1041                                        if (cmsg->cmsg_level != IPPROTO_SCTP)
1042                                                continue;
1043 #ifdef SCTP_RCVINFO
1044                                        if (cmsg->cmsg_type == SCTP_RCVINFO)
1045                                                {
1046                                                struct sctp_rcvinfo *rcvinfo;

1048                                                rcvinfo = (struct sctp_rcvinfo *
1049                                                data->rcvinfo.rcv_sid = rcvinfo-
1050                                                data->rcvinfo.rcv_ssn = rcvinfo-
1051                                                data->rcvinfo.rcv_flags = rcvinf
```

```
1052                                            data->rcvinfo.rcv_ppid = rcvinfo
1053                                            data->rcvinfo.rcv_tsn = rcvinfo-
1054                                            data->rcvinfo.rcv_cumtsn = rcvin
1055                                            data->rcvinfo.rcv_context = rcvi
1056                                            }
1057 #endif
1058 #ifdef SCTP_SNDRCV
1059                                    if (cmsg->cmsg_type == SCTP_SNDRCV)
1060                                            {
1061                                            struct sctp_sndrcvinfo *sndrcvin

1063                                            sndrcvinfo = (struct sctp_sndrcv
1064                                            data->rcvinfo.rcv_sid = sndrcvin
1065                                            data->rcvinfo.rcv_ssn = sndrcvin
1066                                            data->rcvinfo.rcv_flags = sndrcv
1067                                            data->rcvinfo.rcv_ppid = sndrcvi
1068                                            data->rcvinfo.rcv_tsn = sndrcvin
1069                                            data->rcvinfo.rcv_cumtsn = sndrc
1070                                            data->rcvinfo.rcv_context = sndr
1071                                            }
1072 #endif
1073                                    }
1074                            }

1076                        if (n <= 0)
1077                                {
1078                                if (n < 0)
1079                                        ret = n;
1080                                break;
1081                                }

1083                        if (msg.msg_flags & MSG_NOTIFICATION)
1084                                {
1085                                snp = (union sctp_notification*) out;
1086                                if (snp->sn_header.sn_type == SCTP_SENDER_DRY_EV
1087                                        {
1088 #ifdef SCTP_EVENT
1089                                        struct sctp_event event;
1090 #else
1091                                        struct sctp_event_subscribe event;
1092                                        socklen_t eventsize;
1093 #endif
1094                                        /* If a message has been delayed until t
1095                                         * is dry, it can be sent now.
1096                                         */
1097                                        if (data->saved_message.length > 0)
1098                                                {
1099                                                dgram_sctp_write(data->saved_mes
1100                                                                 data->saved_mes
1101                                                OPENSSL_free(data->saved_message
1102                                                data->saved_message.length = 0;
1103                                                }

1105                                        /* disable sender dry event */
1106 #ifdef SCTP_EVENT
1107                                        memset(&event, 0, sizeof(struct sctp_eve
1108                                        event.se_assoc_id = 0;
1109                                        event.se_type = SCTP_SENDER_DRY_EVENT;
1110                                        event.se_on = 0;
1111                                        i = setsockopt(b->num, IPPROTO_SCTP, SCT
1112                                        OPENSSL_assert(i >= 0);
1113 #else
1114                                        eventsize = sizeof(struct sctp_event_sub
1115                                        i = getsockopt(b->num, IPPROTO_SCTP, SCT
1116                                        OPENSSL_assert(i >= 0);
```



```
1118                                            event.sctp_sender_dry_event = 0;

1120                                            i = setsockopt(b->num, IPPROTO_SCTP, SCT
1121                                            OPENSSL_assert(i >= 0);
1122 #endif
1123                                            }

1125 #ifdef SCTP_AUTHENTICATION_EVENT
1126                                    if (snp->sn_header.sn_type == SCTP_AUTHENTICATIO
1127                                            dgram_sctp_handle_auth_free_key_event(b,
1128 #endif

1130                                    if (data->handle_notifications != NULL)
1131                                            data->handle_notifications(b, data->noti

1133                                    memset(out, 0, outl);
1134                                    }
1135                            else
1136                                    ret += n;
1137                            }
1138                    while ((msg.msg_flags & MSG_NOTIFICATION) && (msg.msg_flags & MS

1140                    if (ret > 0 && !(msg.msg_flags & MSG_EOR))
1141                            {
1142                            /* Partial message read, this should never happen! */

1144                            /* The buffer was too small, this means the peer sent
1145                             * a message that was larger than allowed. */
1146                            if (ret == outl)
1147                                    return -1;

1149                            /* Test if socket buffer can handle max record
1150                             * size (2^14 + 2048 + 13)
1151                             */
1152                            optlen = (socklen_t) sizeof(int);
1153                            ret = getsockopt(b->num, SOL_SOCKET, SO_RCVBUF, &optval,
1154                            OPENSSL_assert(ret >= 0);
1155                            OPENSSL_assert(optval >= 18445);

1157                            /* Test if SCTP doesn't partially deliver below
1158                             * max record size (2^14 + 2048 + 13)
1159                             */
1160                            optlen = (socklen_t) sizeof(int);
1161                            ret = getsockopt(b->num, IPPROTO_SCTP, SCTP_PARTIAL_DELI
1162                                             &optval, &optlen);
1163                            OPENSSL_assert(ret >= 0);
1164                            OPENSSL_assert(optval >= 18445);

1166                            /* Partially delivered notification??? Probably a bug...
1167                            OPENSSL_assert(!(msg.msg_flags & MSG_NOTIFICATION));

1169                            /* Everything seems ok till now, so it's most likely
1170                             * a message dropped by PR-SCTP.
1171                             */
1172                            memset(out, 0, outl);
1173                            BIO_set_retry_read(b);
1174                            return -1;
1175                            }

1177                    BIO_clear_retry_flags(b);
1178                    if (ret < 0)
1179                            {
1180                            if (BIO_dgram_should_retry(ret))
1181                                    {
1182                                    BIO_set_retry_read(b);
1183                                    data->_errno = get_last_socket_error();
```

```
1184                                }
1185                        }

1187                /* Test if peer uses SCTP-AUTH before continuing */
1188                if (!data->peer_auth_tested)
1189                        {
1190                        int ii, auth_data = 0, auth_forward = 0;
1191                        unsigned char *p;
1192                        struct sctp_authchunks *authchunks;

1194                        optlen = (socklen_t)(sizeof(sctp_assoc_t) + 256 * sizeof
1195                        authchunks = OPENSSL_malloc(optlen);
1196                        memset(authchunks, 0, sizeof(optlen));
1197                        ii = getsockopt(b->num, IPPROTO_SCTP, SCTP_PEER_AUTH_CHU
1198                        OPENSSL_assert(ii >= 0);

1200                        for (p = (unsigned char*) authchunks->gauth_chunks;
1201                                p < (unsigned char*) authchunks + optlen;
1202                                p += sizeof(uint8_t))
1203                                {
1204                                if (*p == OPENSSL_SCTP_DATA_CHUNK_TYPE) auth_dat
1205                                if (*p == OPENSSL_SCTP_FORWARD_CUM_TSN_CHUNK_TYP
1206                                }

1208                        OPENSSL_free(authchunks);

1210                        if (!auth_data || !auth_forward)
1211                                {
1212                                BIOerr(BIO_F_DGRAM_SCTP_READ,BIO_R_CONNECT_ERROR
1213                                return -1;
1214                                }

1216                        data->peer_auth_tested = 1;
1217                        }
1218                }
1219        return(ret);
1220        }

1222 static int dgram_sctp_write(BIO *b, const char *in, int inl)
1223        {
1224        int ret;
1225        bio_dgram_sctp_data *data = (bio_dgram_sctp_data *)b->ptr;
1226        struct bio_dgram_sctp_sndinfo *sinfo = &(data->sndinfo);
1227        struct bio_dgram_sctp_prinfo *pinfo = &(data->prinfo);
1228        struct bio_dgram_sctp_sndinfo handshake_sinfo;
1229        struct iovec iov[1];
1230        struct msghdr msg;
1231        struct cmsghdr *cmsg;
1232 #if defined(SCTP_SNDINFO) && defined(SCTP_PRINFO)
1233        char cmsgbuf[CMSG_SPACE(sizeof(struct sctp_sndinfo)) + CMSG_SPACE(sizeof
1234        struct sctp_sndinfo *sndinfo;
1235        struct sctp_prinfo *prinfo;
1236 #else
1237        char cmsgbuf[CMSG_SPACE(sizeof(struct sctp_sndrcvinfo))];
1238        struct sctp_sndrcvinfo *sndrcvinfo;
1239 #endif

1241        clear_socket_error();

1243        /* If we're send anything else than application data,
1244         * disable all user parameters and flags.
1245         */
1246        if (in[0] != 23) {
1247                memset(&handshake_sinfo, 0x00, sizeof(struct bio_dgram_sctp_sndi
1248 #ifdef SCTP_SACK_IMMEDIATELY
1249                handshake_sinfo.snd_flags = SCTP_SACK_IMMEDIATELY;
```



```
1250 #endif
1251                sinfo = &handshake_sinfo;
1252        }

1254        /* If we have to send a shutdown alert message and the
1255         * socket is not dry yet, we have to save it and send it
1256         * as soon as the socket gets dry.
1257         */
1258        if (data->save_shutdown && !BIO_dgram_sctp_wait_for_dry(b))
1259        {
1260                data->saved_message.bio = b;
1261                data->saved_message.length = inl;
1262                data->saved_message.data = OPENSSL_malloc(inl);
1263                memcpy(data->saved_message.data, in, inl);
1264                return inl;
1265        }

1267        iov[0].iov_base = (char *)in;
1268        iov[0].iov_len = inl;
1269        msg.msg_name = NULL;
1270        msg.msg_namelen = 0;
1271        msg.msg_iov = iov;
1272        msg.msg_iovlen = 1;
1273        msg.msg_control = (caddr_t)cmsgbuf;
1274        msg.msg_controllen = 0;
1275        msg.msg_flags = 0;
1276 #if defined(SCTP_SNDINFO) && defined(SCTP_PRINFO)
1277        cmsg = (struct cmsghdr *)cmsgbuf;
1278        cmsg->cmsg_level = IPPROTO_SCTP;
1279        cmsg->cmsg_type = SCTP_SNDINFO;
1280        cmsg->cmsg_len = CMSG_LEN(sizeof(struct sctp_sndinfo));
1281        sndinfo = (struct sctp_sndinfo *)CMSG_DATA(cmsg);
1282        memset(sndinfo, 0, sizeof(struct sctp_sndinfo));
1283        sndinfo->snd_sid = sinfo->snd_sid;
1284        sndinfo->snd_flags = sinfo->snd_flags;
1285        sndinfo->snd_ppid = sinfo->snd_ppid;
1286        sndinfo->snd_context = sinfo->snd_context;
1287        msg.msg_controllen += CMSG_SPACE(sizeof(struct sctp_sndinfo));

1289        cmsg = (struct cmsghdr *)&cmsgbuf[CMSG_SPACE(sizeof(struct sctp_sndinfo)
1290        cmsg->cmsg_level = IPPROTO_SCTP;
1291        cmsg->cmsg_type = SCTP_PRINFO;
1292        cmsg->cmsg_len = CMSG_LEN(sizeof(struct sctp_prinfo));
1293        prinfo = (struct sctp_prinfo *)CMSG_DATA(cmsg);
1294        memset(prinfo, 0, sizeof(struct sctp_prinfo));
1295        prinfo->pr_policy = pinfo->pr_policy;
1296        prinfo->pr_value = pinfo->pr_value;
1297        msg.msg_controllen += CMSG_SPACE(sizeof(struct sctp_prinfo));
1298 #else
1299        cmsg = (struct cmsghdr *)cmsgbuf;
1300        cmsg->cmsg_level = IPPROTO_SCTP;
1301        cmsg->cmsg_type = SCTP_SNDRCV;
1302        cmsg->cmsg_len = CMSG_LEN(sizeof(struct sctp_sndrcvinfo));
1303        sndrcvinfo = (struct sctp_sndrcvinfo *)CMSG_DATA(cmsg);
1304        memset(sndrcvinfo, 0, sizeof(struct sctp_sndrcvinfo));
1305        sndrcvinfo->sinfo_stream = sinfo->snd_sid;
1306        sndrcvinfo->sinfo_flags = sinfo->snd_flags;
1307 #ifdef __FreeBSD__
1308        sndrcvinfo->sinfo_flags |= pinfo->pr_policy;
1309 #endif
1310        sndrcvinfo->sinfo_ppid = sinfo->snd_ppid;
1311        sndrcvinfo->sinfo_context = sinfo->snd_context;
1312        sndrcvinfo->sinfo_timetolive = pinfo->pr_value;
1313        msg.msg_controllen += CMSG_SPACE(sizeof(struct sctp_sndrcvinfo));
1314 #endif
```

```
1316         ret = sendmsg(b->num, &msg, 0);

1318         BIO_clear_retry_flags(b);
1319         if (ret <= 0)
1320                 {
1321                 if (BIO_dgram_should_retry(ret))
1322                         {
1323                         BIO_set_retry_write(b);
1324                         data->_errno = get_last_socket_error();
1325                         }
1326                 }
1327         return(ret);
1328         }

1330 static long dgram_sctp_ctrl(BIO *b, int cmd, long num, void *ptr)
1331         {
1332         long ret=1;
1333         bio_dgram_sctp_data *data = NULL;
1334         socklen_t sockopt_len = 0;
1335         struct sctp_authkeyid authkeyid;
1336         struct sctp_authkey *authkey = NULL;

1338         data = (bio_dgram_sctp_data *)b->ptr;

1340         switch (cmd)
1341                 {
1342         case BIO_CTRL_DGRAM_QUERY_MTU:
1343                 /* Set to maximum (2^14)
1344                  * and ignore user input to enable transport
1345                  * protocol fragmentation.
1346                  * Returns always 2^14.
1347                  */
1348                 data->mtu = 16384;
1349                 ret = data->mtu;
1350                 break;
1351         case BIO_CTRL_DGRAM_SET_MTU:
1352                 /* Set to maximum (2^14)
1353                  * and ignore input to enable transport
1354                  * protocol fragmentation.
1355                  * Returns always 2^14.
1356                  */
1357                 data->mtu = 16384;
1358                 ret = data->mtu;
1359                 break;
1360         case BIO_CTRL_DGRAM_SET_CONNECTED:
1361         case BIO_CTRL_DGRAM_CONNECT:
1362                 /* Returns always -1. */
1363                 ret = -1;
1364                 break;
1365         case BIO_CTRL_DGRAM_SET_NEXT_TIMEOUT:
1366                 /* SCTP doesn't need the DTLS timer
1367                  * Returns always 1.
1368                  */
1369                 break;
1370         case BIO_CTRL_DGRAM_SCTP_SET_IN_HANDSHAKE:
1371                 if (num > 0)
1372                         data->in_handshake = 1;
1373                 else
1374                         data->in_handshake = 0;

1376                 ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_NODELAY, &data->in_h
1377                 break;
1378         case BIO_CTRL_DGRAM_SCTP_ADD_AUTH_KEY:
1379                 /* New shared key for SCTP AUTH.
1380                  * Returns 0 on success, -1 otherwise.
1381                  */
```



```
1383                 /* Get active key */
1384                 sockopt_len = sizeof(struct sctp_authkeyid);
1385                 ret = getsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_ACTIVE_KEY, &au
1386                 if (ret < 0) break;

1388                 /* Add new key */
1389                 sockopt_len = sizeof(struct sctp_authkey) + 64 * sizeof(uint8_t)
1390                 authkey = OPENSSL_malloc(sockopt_len);
1391                 if (authkey == NULL)
1392                         {
1393                         ret = -1;
1394                         break;
1395                         }
1396                 memset(authkey, 0x00, sockopt_len);
1397                 authkey->sca_keynumber = authkeyid.scact_keynumber + 1;
1398 #ifndef __FreeBSD__
1399                 /* This field is missing in FreeBSD 8.2 and earlier,
1400                  * and FreeBSD 8.3 and higher work without it.
1401                  */
1402                 authkey->sca_keylength = 64;
1403 #endif
1404                 memcpy(&authkey->sca_key[0], ptr, 64 * sizeof(uint8_t));

1406                 ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_KEY, authkey, s
1407                 OPENSSL_free(authkey);
1408                 authkey = NULL;
1409                 if (ret < 0) break;

1411                 /* Reset active key */
1412                 ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_ACTIVE_KEY,
1413                       &authkeyid, sizeof(struct sctp_authkeyid));
1414                 if (ret < 0) break;

1416                 break;
1417         case BIO_CTRL_DGRAM_SCTP_NEXT_AUTH_KEY:
1418                 /* Returns 0 on success, -1 otherwise. */

1420                 /* Get active key */
1421                 sockopt_len = sizeof(struct sctp_authkeyid);
1422                 ret = getsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_ACTIVE_KEY, &au
1423                 if (ret < 0) break;

1425                 /* Set active key */
1426                 authkeyid.scact_keynumber = authkeyid.scact_keynumber + 1;
1427                 ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_ACTIVE_KEY,
1428                       &authkeyid, sizeof(struct sctp_authkeyid));
1429                 if (ret < 0) break;

1431                 /* CCS has been sent, so remember that and fall through
1432                  * to check if we need to deactivate an old key
1433                  */
1434                 data->ccs_sent = 1;

1436         case BIO_CTRL_DGRAM_SCTP_AUTH_CCS_RCVD:
1437                 /* Returns 0 on success, -1 otherwise. */

1439                 /* Has this command really been called or is this just a fall-th
1440                 if (cmd == BIO_CTRL_DGRAM_SCTP_AUTH_CCS_RCVD)
1441                         data->ccs_rcvd = 1;

1443                 /* CSS has been both, received and sent, so deactivate an old ke
1444                 if (data->ccs_rcvd == 1 && data->ccs_sent == 1)
1445                         {
1446                         /* Get active key */
1447                         sockopt_len = sizeof(struct sctp_authkeyid);
```

```
1448                        ret = getsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_ACTIVE_
1449                        if (ret < 0) break;

1451                        /* Deactivate key or delete second last key if
1452                         * SCTP_AUTHENTICATION_EVENT is not available.
1453                         */
1454                        authkeyid.scact_keynumber = authkeyid.scact_keynumber -
1455 #ifdef SCTP_AUTH_DEACTIVATE_KEY
1456                        sockopt_len = sizeof(struct sctp_authkeyid);
1457                        ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH_DEACTIV
1458                              &authkeyid, sockopt_len);
1459                        if (ret < 0) break;
1460 #endif
1461 #ifndef SCTP_AUTHENTICATION_EVENT
1462                        if (authkeyid.scact_keynumber > 0)
1463                                {
1464                                authkeyid.scact_keynumber = authkeyid.scact_keyn
1465                                ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_AUTH
1466                                        &authkeyid, sizeof(struct sctp_authkey
1467                                if (ret < 0) break;
1468                                }
1469 #endif

1471                        data->ccs_rcvd = 0;
1472                        data->ccs_sent = 0;
1473                        }
1474                break;
1475        case BIO_CTRL_DGRAM_SCTP_GET_SNDINFO:
1476                /* Returns the size of the copied struct. */
1477                if (num > (long) sizeof(struct bio_dgram_sctp_sndinfo))
1478                        num = sizeof(struct bio_dgram_sctp_sndinfo);

1480                memcpy(ptr, &(data->sndinfo), num);
1481                ret = num;
1482                break;
1483        case BIO_CTRL_DGRAM_SCTP_SET_SNDINFO:
1484                /* Returns the size of the copied struct. */
1485                if (num > (long) sizeof(struct bio_dgram_sctp_sndinfo))
1486                        num = sizeof(struct bio_dgram_sctp_sndinfo);

1488                memcpy(&(data->sndinfo), ptr, num);
1489                break;
1490        case BIO_CTRL_DGRAM_SCTP_GET_RCVINFO:
1491                /* Returns the size of the copied struct. */
1492                if (num > (long) sizeof(struct bio_dgram_sctp_rcvinfo))
1493                        num = sizeof(struct bio_dgram_sctp_rcvinfo);

1495                memcpy(ptr, &data->rcvinfo, num);

1497                ret = num;
1498                break;
1499        case BIO_CTRL_DGRAM_SCTP_SET_RCVINFO:
1500                /* Returns the size of the copied struct. */
1501                if (num > (long) sizeof(struct bio_dgram_sctp_rcvinfo))
1502                        num = sizeof(struct bio_dgram_sctp_rcvinfo);

1504                memcpy(&(data->rcvinfo), ptr, num);
1505                break;
1506        case BIO_CTRL_DGRAM_SCTP_GET_PRINFO:
1507                /* Returns the size of the copied struct. */
1508                if (num > (long) sizeof(struct bio_dgram_sctp_prinfo))
1509                        num = sizeof(struct bio_dgram_sctp_prinfo);

1511                memcpy(ptr, &(data->prinfo), num);
1512                ret = num;
1513                break;
```



```
1514        case BIO_CTRL_DGRAM_SCTP_SET_PRINFO:
1515                /* Returns the size of the copied struct. */
1516                if (num > (long) sizeof(struct bio_dgram_sctp_prinfo))
1517                        num = sizeof(struct bio_dgram_sctp_prinfo);

1519                memcpy(&(data->prinfo), ptr, num);
1520                break;
1521        case BIO_CTRL_DGRAM_SCTP_SAVE_SHUTDOWN:
1522                /* Returns always 1. */
1523                if (num > 0)
1524                        data->save_shutdown = 1;
1525                else
1526                        data->save_shutdown = 0;
1527                break;

1529        default:
1530                /* Pass to default ctrl function to
1531                 * process SCTP unspecific commands
1532                 */
1533                ret=dgram_ctrl(b, cmd, num, ptr);
1534                break;
1535                }
1536        return(ret);
1537        }

1539 int BIO_dgram_sctp_notification_cb(BIO *b,
1540                                   void (*handle_notifications)(BIO *bio, void *
1541                                   void *context)
1542        {
1543        bio_dgram_sctp_data *data = (bio_dgram_sctp_data *) b->ptr;

1545        if (handle_notifications != NULL)
1546                {
1547                data->handle_notifications = handle_notifications;
1548                data->notification_context = context;
1549                }
1550        else
1551                return -1;

1553        return 0;
1554        }

1556 int BIO_dgram_sctp_wait_for_dry(BIO *b)
1557 {
1558        int is_dry = 0;
1559        int n, sockflags, ret;
1560        union sctp_notification snp;
1561        struct msghdr msg;
1562        struct iovec iov;
1563 #ifdef SCTP_EVENT
1564        struct sctp_event event;
1565 #else
1566        struct sctp_event_subscribe event;
1567        socklen_t eventsize;
1568 #endif
1569        bio_dgram_sctp_data *data = (bio_dgram_sctp_data *)b->ptr;

1571        /* set sender dry event */
1572 #ifdef SCTP_EVENT
1573        memset(&event, 0, sizeof(struct sctp_event));
1574        event.se_assoc_id = 0;
1575        event.se_type = SCTP_SENDER_DRY_EVENT;
1576        event.se_on = 1;
1577        ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_EVENT, &event, sizeof(struct
1578 #else
1579        eventsize = sizeof(struct sctp_event_subscribe);
```

```
1580         ret = getsockopt(b->num, IPPROTO_SCTP, SCTP_EVENTS, &event, &eventsize);
1581         if (ret < 0)
1582                 return -1;

1584         event.sctp_sender_dry_event = 1;

1586         ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_EVENTS, &event, sizeof(struc
1587 #endif
1588         if (ret < 0)
1589                 return -1;

1591         /* peek for notification */
1592         memset(&snp, 0x00, sizeof(union sctp_notification));
1593         iov.iov_base = (char *)&snp;
1594         iov.iov_len = sizeof(union sctp_notification);
1595         msg.msg_name = NULL;
1596         msg.msg_namelen = 0;
1597         msg.msg_iov = &iov;
1598         msg.msg_iovlen = 1;
1599         msg.msg_control = NULL;
1600         msg.msg_controllen = 0;
1601         msg.msg_flags = 0;

1603         n = recvmsg(b->num, &msg, MSG_PEEK);
1604         if (n <= 0)
1605                 {
1606                 if ((n < 0) && (get_last_socket_error() != EAGAIN) && (get_last_
1607                         return -1;
1608                 else
1609                         return 0;
1610                 }

1612         /* if we find a notification, process it and try again if necessary */
1613         while (msg.msg_flags & MSG_NOTIFICATION)
1614                 {
1615                 memset(&snp, 0x00, sizeof(union sctp_notification));
1616                 iov.iov_base = (char *)&snp;
1617                 iov.iov_len = sizeof(union sctp_notification);
1618                 msg.msg_name = NULL;
1619                 msg.msg_namelen = 0;
1620                 msg.msg_iov = &iov;
1621                 msg.msg_iovlen = 1;
1622                 msg.msg_control = NULL;
1623                 msg.msg_controllen = 0;
1624                 msg.msg_flags = 0;

1626                 n = recvmsg(b->num, &msg, 0);
1627                 if (n <= 0)
1628                         {
1629                         if ((n < 0) && (get_last_socket_error() != EAGAIN) && (g
1630                                 return -1;
1631                         else
1632                                 return is_dry;
1633                         }

1635                 if (snp.sn_header.sn_type == SCTP_SENDER_DRY_EVENT)
1636                         {
1637                         is_dry = 1;

1639                         /* disable sender dry event */
1640 #ifdef SCTP_EVENT
1641                         memset(&event, 0, sizeof(struct sctp_event));
1642                         event.se_assoc_id = 0;
1643                         event.se_type = SCTP_SENDER_DRY_EVENT;
1644                         event.se_on = 0;
1645                         ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_EVENT, &even
```



```
1646 #else
1647                         eventsize = (socklen_t) sizeof(struct sctp_event_subscri
1648                         ret = getsockopt(b->num, IPPROTO_SCTP, SCTP_EVENTS, &eve
1649                         if (ret < 0)
1650                                 return -1;

1652                         event.sctp_sender_dry_event = 0;

1654                         ret = setsockopt(b->num, IPPROTO_SCTP, SCTP_EVENTS, &eve
1655 #endif
1656                         if (ret < 0)
1657                                 return -1;
1658                         }

1660 #ifdef SCTP_AUTHENTICATION_EVENT
1661                 if (snp.sn_header.sn_type == SCTP_AUTHENTICATION_EVENT)
1662                         dgram_sctp_handle_auth_free_key_event(b, &snp);
1663 #endif

1665                 if (data->handle_notifications != NULL)
1666                         data->handle_notifications(b, data->notification_context

1668                 /* found notification, peek again */
1669                 memset(&snp, 0x00, sizeof(union sctp_notification));
1670                 iov.iov_base = (char *)&snp;
1671                 iov.iov_len = sizeof(union sctp_notification);
1672                 msg.msg_name = NULL;
1673                 msg.msg_namelen = 0;
1674                 msg.msg_iov = &iov;
1675                 msg.msg_iovlen = 1;
1676                 msg.msg_control = NULL;
1677                 msg.msg_controllen = 0;
1678                 msg.msg_flags = 0;

1680                 /* if we have seen the dry already, don't wait */
1681                 if (is_dry)
1682                         {
1683                         sockflags = fcntl(b->num, F_GETFL, 0);
1684                         fcntl(b->num, F_SETFL, O_NONBLOCK);
1685                         }

1687                 n = recvmsg(b->num, &msg, MSG_PEEK);

1689                 if (is_dry)
1690                         {
1691                         fcntl(b->num, F_SETFL, sockflags);
1692                         }

1694                 if (n <= 0)
1695                         {
1696                         if ((n < 0) && (get_last_socket_error() != EAGAIN) && (g
1697                                 return -1;
1698                         else
1699                                 return is_dry;
1700                         }
1701                 }

1703         /* read anything else */
1704         return is_dry;
1705 }

1707 int BIO_dgram_sctp_msg_waiting(BIO *b)
1708         {
1709         int n, sockflags;
1710         union sctp_notification snp;
1711         struct msghdr msg;
```

```
1712         struct iovec iov;
1713         bio_dgram_sctp_data *data = (bio_dgram_sctp_data *)b->ptr;

1715         /* Check if there are any messages waiting to be read */
1716         do
1717                 {
1718                 memset(&snp, 0x00, sizeof(union sctp_notification));
1719                 iov.iov_base = (char *)&snp;
1720                 iov.iov_len = sizeof(union sctp_notification);
1721                 msg.msg_name = NULL;
1722                 msg.msg_namelen = 0;
1723                 msg.msg_iov = &iov;
1724                 msg.msg_iovlen = 1;
1725                 msg.msg_control = NULL;
1726                 msg.msg_controllen = 0;
1727                 msg.msg_flags = 0;

1729                 sockflags = fcntl(b->num, F_GETFL, 0);
1730                 fcntl(b->num, F_SETFL, O_NONBLOCK);
1731                 n = recvmsg(b->num, &msg, MSG_PEEK);
1732                 fcntl(b->num, F_SETFL, sockflags);

1734                 /* if notification, process and try again */
1735                 if (n > 0 && (msg.msg_flags & MSG_NOTIFICATION))
1736                         {
1737 #ifdef SCTP_AUTHENTICATION_EVENT
1738                         if (snp.sn_header.sn_type == SCTP_AUTHENTICATION_EVENT)
1739                                 dgram_sctp_handle_auth_free_key_event(b, &snp);
1740 #endif

1742                         memset(&snp, 0x00, sizeof(union sctp_notification));
1743                         iov.iov_base = (char *)&snp;
1744                         iov.iov_len = sizeof(union sctp_notification);
1745                         msg.msg_name = NULL;
1746                         msg.msg_namelen = 0;
1747                         msg.msg_iov = &iov;
1748                         msg.msg_iovlen = 1;
1749                         msg.msg_control = NULL;
1750                         msg.msg_controllen = 0;
1751                         msg.msg_flags = 0;
1752                         n = recvmsg(b->num, &msg, 0);

1754                         if (data->handle_notifications != NULL)
1755                                 data->handle_notifications(b, data->notification
1756                         }

1758                 } while (n > 0 && (msg.msg_flags & MSG_NOTIFICATION));

1760         /* Return 1 if there is a message to be read, return 0 otherwise. */
1761         if (n > 0)
1762                 return 1;
1763         else
1764                 return 0;
1765         }

1767 static int dgram_sctp_puts(BIO *bp, const char *str)
1768         {
1769         int n,ret;

1771         n=strlen(str);
1772         ret=dgram_sctp_write(bp,str,n);
1773         return(ret);
1774         }
1775 #endif

1777 static int BIO_dgram_should_retry(int i)
```



```
1778         {
1779         int err;

1781         if ((i == 0) || (i == -1))
1782                 {
1783                 err=get_last_socket_error();

1785 #if defined(OPENSSL_SYS_WINDOWS)
1786         /* If the socket return value (i) is -1
1787          * and err is unexpectedly 0 at this point,
1788          * the error code was overwritten by
1789          * another system call before this error
1790          * handling is called.
1791          */
1792 #endif

1794                 return(BIO_dgram_non_fatal_error(err));
1795                 }
1796         return(0);
1797         }

1799 int BIO_dgram_non_fatal_error(int err)
1800         {
1801         switch (err)
1802                 {
1803 #if defined(OPENSSL_SYS_WINDOWS)
1804 # if defined(WSAEWOULDBLOCK)
1805         case WSAEWOULDBLOCK:
1806 # endif

1808 # if 0 /* This appears to always be an error */
1809 #  if defined(WSAENOTCONN)
1810         case WSAENOTCONN:
1811 #  endif
1812 # endif
1813 #endif

1815 #ifdef EWOULDBLOCK
1816 # ifdef WSAEWOULDBLOCK
1817 #  if WSAEWOULDBLOCK != EWOULDBLOCK
1818         case EWOULDBLOCK:
1819 #  endif
1820 # else
1821         case EWOULDBLOCK:
1822 # endif
1823 #endif

1825 #ifdef EINTR
1826         case EINTR:
1827 #endif

1829 #ifdef EAGAIN
1830 #if EWOULDBLOCK != EAGAIN
1831         case EAGAIN:
1832 # endif
1833 #endif

1835 #ifdef EPROTO
1836         case EPROTO:
1837 #endif

1839 #ifdef EINPROGRESS
1840         case EINPROGRESS:
1841 #endif

1843 #ifdef EALREADY
```

```
1844         case EALREADY:
1845 #endif

1847                 return(1);
1848                 /* break; */
1849         default:
1850                 break;
1851                 }
1852         return(0);
1853         }

1855 static void get_current_time(struct timeval *t)
1856         {
1857 #ifdef OPENSSL_SYS_WIN32
1858         struct _timeb tb;
1859         _ftime(&tb);
1860         t->tv_sec = (long)tb.time;
1861         t->tv_usec = (long)tb.millitm * 1000;
1862 #elif defined(OPENSSL_SYS_VMS)
1863         struct timeb tb;
1864         ftime(&tb);
1865         t->tv_sec = (long)tb.time;
1866         t->tv_usec = (long)tb.millitm * 1000;
1867 #else
1868         gettimeofday(t, NULL);
1869 #endif
1870         }

1872 #endif
1873 #endif /* ! codereview */
```

```
**********************************************************
    7703 Wed Aug 13 19:52:12 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_fd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_fd.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#define USE_SOCKETS
```



```
#include "cryptlib.h"

#if defined(OPENSSL_NO_POSIX_IO)
/*
 * One can argue that one should implement dummy placeholder for
 * BIO_s_fd here...
 */
#else
/*
 * As for unconditional usage of "UPLINK" interface in this module.
 * Trouble is that unlike Unix file descriptors [which are indexes
 * in kernel-side per-process table], corresponding descriptors on
 * platforms which require "UPLINK" interface seem to be indexes
 * in a user-land, non-global table. Well, in fact they are indexes
 * in stdio _iob[], and recall that _iob[] was the very reason why
 * "UPLINK" interface was introduced in first place. But one way on
 * another. Neither libcrypto or libssl use this BIO meaning that
 * file descriptors can only be provided by application. Therefore
 * "UPLINK" calls are due...
 */
#include "bio_lcl.h"

static int fd_write(BIO *h, const char *buf, int num);
static int fd_read(BIO *h, char *buf, int size);
static int fd_puts(BIO *h, const char *str);
static int fd_gets(BIO *h, char *buf, int size);
static long fd_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int fd_new(BIO *h);
static int fd_free(BIO *data);
int BIO_fd_should_retry(int s);

static BIO_METHOD methods_fdp=
        {
        BIO_TYPE_FD,"file descriptor",
        fd_write,
        fd_read,
        fd_puts,
        fd_gets,
        fd_ctrl,
        fd_new,
        fd_free,
        NULL,
        };

BIO_METHOD *BIO_s_fd(void)
        {
        return(&methods_fdp);
        }

BIO *BIO_new_fd(int fd,int close_flag)
        {
        BIO *ret;
        ret=BIO_new(BIO_s_fd());
        if (ret == NULL) return(NULL);
        BIO_set_fd(ret,fd,close_flag);
        return(ret);
        }

static int fd_new(BIO *bi)
        {
        bi->init=0;
        bi->num=-1;
        bi->ptr=NULL;
        bi->flags=BIO_FLAGS_UPLINK; /* essentially redundant */
        return(1);
        }
```

```
129 static int fd_free(BIO *a)
130         {
131         if (a == NULL) return(0);
132         if (a->shutdown)
133                 {
134                 if (a->init)
135                         {
136                         UP_close(a->num);
137                         }
138                 a->init=0;
139                 a->flags=BIO_FLAGS_UPLINK;
140                 }
141         return(1);
142         }

144 static int fd_read(BIO *b, char *out,int outl)
145         {
146         int ret=0;

148         if (out != NULL)
149                 {
150                 clear_sys_error();
151                 ret=UP_read(b->num,out,outl);
152                 BIO_clear_retry_flags(b);
153                 if (ret <= 0)
154                         {
155                         if (BIO_fd_should_retry(ret))
156                                 BIO_set_retry_read(b);
157                         }
158                 }
159         return(ret);
160         }

162 static int fd_write(BIO *b, const char *in, int inl)
163         {
164         int ret;
165         clear_sys_error();
166         ret=UP_write(b->num,in,inl);
167         BIO_clear_retry_flags(b);
168         if (ret <= 0)
169                 {
170                 if (BIO_fd_should_retry(ret))
171                         BIO_set_retry_write(b);
172                 }
173         return(ret);
174         }

176 static long fd_ctrl(BIO *b, int cmd, long num, void *ptr)
177         {
178         long ret=1;
179         int *ip;

181         switch (cmd)
182                 {
183         case BIO_CTRL_RESET:
184                 num=0;
185         case BIO_C_FILE_SEEK:
186                 ret=(long)UP_lseek(b->num,num,0);
187                 break;
188         case BIO_C_FILE_TELL:
189         case BIO_CTRL_INFO:
190                 ret=(long)UP_lseek(b->num,0,1);
191                 break;
192         case BIO_C_SET_FD:
193                 fd_free(b);
```



```
194                 b->num= *((int *)ptr);
195                 b->shutdown=(int)num;
196                 b->init=1;
197                 break;
198         case BIO_C_GET_FD:
199                 if (b->init)
200                         {
201                         ip=(int *)ptr;
202                         if (ip != NULL) *ip=b->num;
203                         ret=b->num;
204                         }
205                 else
206                         ret= -1;
207                 break;
208         case BIO_CTRL_GET_CLOSE:
209                 ret=b->shutdown;
210                 break;
211         case BIO_CTRL_SET_CLOSE:
212                 b->shutdown=(int)num;
213                 break;
214         case BIO_CTRL_PENDING:
215         case BIO_CTRL_WPENDING:
216                 ret=0;
217                 break;
218         case BIO_CTRL_DUP:
219         case BIO_CTRL_FLUSH:
220                 ret=1;
221                 break;
222         default:
223                 ret=0;
224                 break;
225                 }
226         return(ret);
227         }

229 static int fd_puts(BIO *bp, const char *str)
230         {
231         int n,ret;

233         n=strlen(str);
234         ret=fd_write(bp,str,n);
235         return(ret);
236         }

238 static int fd_gets(BIO *bp, char *buf, int size)
239         {
240         int ret=0;
241         char *ptr=buf;
242         char *end=buf+size-1;

244         while ( (ptr < end) && (fd_read(bp, ptr, 1) > 0) && (ptr[0] != '\n') )
245                 ptr++;

247         ptr[0]='\0';

249         if (buf[0] != '\0')
250                 ret=strlen(buf);
251         return(ret);
252         }

254 int BIO_fd_should_retry(int i)
255         {
256         int err;

258         if ((i == 0) || (i == -1))
259                 {
```

```
260                 err=get_last_sys_error();

262 #if defined(OPENSSL_SYS_WINDOWS) && 0 /* more microsoft stupidity? perhaps not?
263                 if ((i == -1) && (err == 0))
264                         return(1);
265 #endif

267                 return(BIO_fd_non_fatal_error(err));
268                 }
269         return(0);
270         }

272 int BIO_fd_non_fatal_error(int err)
273         {
274         switch (err)
275                 {

277 #ifdef EWOULDBLOCK
278 # ifdef WSAEWOULDBLOCK
279 #  if WSAEWOULDBLOCK != EWOULDBLOCK
280         case EWOULDBLOCK:
281 #  endif
282 # else
283         case EWOULDBLOCK:
284 # endif
285 #endif

287 #if defined(ENOTCONN)
288         case ENOTCONN:
289 #endif

291 #ifdef EINTR
292         case EINTR:
293 #endif

295 #ifdef EAGAIN
296 #if EWOULDBLOCK != EAGAIN
297         case EAGAIN:
298 # endif
299 #endif

301 #ifdef EPROTO
302         case EPROTO:
303 #endif

305 #ifdef EINPROGRESS
306         case EINPROGRESS:
307 #endif

309 #ifdef EALREADY
310         case EALREADY:
311 #endif
312                 return(1);
313                 /* break; */
314         default:
315                 break;
316                 }
317         return(0);
318         }
319 #endif
320 #endif /* ! codereview */
```

```
**********************************************************
   13230 Wed Aug 13 19:52:12 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_file.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_file.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/*
 * 03-Dec-1997  rdenny@dc3.com  Fix bug preventing use of stdin/stdout
 *              with binary data (e.g. asn1parse -inform DER < xxx) under
```



```
 *              Windows
 */

#ifndef HEADER_BSS_FILE_C
#define HEADER_BSS_FILE_C

#if defined(__linux) || defined(__sun) || defined(__hpux)
/* Following definition aliases fopen to fopen64 on above mentioned
 * platforms. This makes it possible to open and sequentially access
 * files larger than 2GB from 32-bit application. It does not allow to
 * traverse them beyond 2GB with fseek/ftell, but on the other hand *no*
 * 32-bit platform permits that, not with fseek/ftell. Not to mention
 * that breaking 2GB limit for seeking would require surgery to *our*
 * API. But sequential access suffices for practical cases when you
 * can run into large files, such as fingerprinting, so we can let API
 * alone. For reference, the list of 32-bit platforms which allow for
 * sequential access of large files without extra "magic" comprise *BSD,
 * Darwin, IRIX...
 */
#ifndef _FILE_OFFSET_BITS
#define _FILE_OFFSET_BITS 64
#endif
#endif

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
#include "bio_lcl.h"
#include <openssl/err.h>

#if defined(OPENSSL_SYS_NETWARE) && defined(NETWARE_CLIB)
#include <nwfileio.h>
#endif

#if !defined(OPENSSL_NO_STDIO)

static int MS_CALLBACK file_write(BIO *h, const char *buf, int num);
static int MS_CALLBACK file_read(BIO *h, char *buf, int size);
static int MS_CALLBACK file_puts(BIO *h, const char *str);
static int MS_CALLBACK file_gets(BIO *h, char *str, int size);
static long MS_CALLBACK file_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int MS_CALLBACK file_new(BIO *h);
static int MS_CALLBACK file_free(BIO *data);
static BIO_METHOD methods_filep=
        {
        BIO_TYPE_FILE,
        "FILE pointer",
        file_write,
        file_read,
        file_puts,
        file_gets,
        file_ctrl,
        file_new,
        file_free,
        NULL,
        };

BIO *BIO_new_file(const char *filename, const char *mode)
        {
        BIO  *ret;
        FILE *file=NULL;

#if defined(_WIN32) && defined(CP_UTF8)
        int sz, len_0 = (int)strlen(filename)+1;
        DWORD flags;
```

```
128         /*
129          * Basically there are three cases to cover: a) filename is
130          * pure ASCII string; b) actual UTF-8 encoded string and
131          * c) locale-ized string, i.e. one containing 8-bit
132          * characters that are meaningful in current system locale.
133          * If filename is pure ASCII or real UTF-8 encoded string,
134          * MultiByteToWideChar succeeds and _wfopen works. If
135          * filename is locale-ized string, chances are that
136          * MultiByteToWideChar fails reporting
137          * ERROR_NO_UNICODE_TRANSLATION, in which case we fall
138          * back to fopen...
139          */
140         if ((sz=MultiByteToWideChar(CP_UTF8,(flags=MB_ERR_INVALID_CHARS),
141                                     filename,len_0,NULL,0))>0 ||
142             (GetLastError()==ERROR_INVALID_FLAGS &&
143              (sz=MultiByteToWideChar(CP_UTF8,(flags=0),
144                                     filename,len_0,NULL,0))>0)
145            )
146                 {
147                 WCHAR  wmode[8];
148                 WCHAR *wfilename = _alloca(sz*sizeof(WCHAR));
149
150                 if (MultiByteToWideChar(CP_UTF8,flags,
151                                     filename,len_0,wfilename,sz) &&
152                     MultiByteToWideChar(CP_UTF8,0,mode,strlen(mode)+1,
153                                     wmode,sizeof(wmode)/sizeof(wmode[0])) &&
154                     (file=_wfopen(wfilename,wmode))==NULL &&
155                     (errno==ENOENT || errno==EBADF)
156                    )    /* UTF-8 decode succeeded, but no file, filename
157                          * could still have been locale-ized... */
158                     file = fopen(filename,mode);
159                 }
160         else if (GetLastError()==ERROR_NO_UNICODE_TRANSLATION)
161                 {
162                 file = fopen(filename,mode);
163                 }
164 #else
165         file=fopen(filename,mode);
166 #endif
167         if (file == NULL)
168                 {
169                 SYSerr(SYS_F_FOPEN,get_last_sys_error());
170                 ERR_add_error_data(5,"fopen('",filename,"','",mode,"')");
171                 if (errno == ENOENT)
172                         BIOerr(BIO_F_BIO_NEW_FILE,BIO_R_NO_SUCH_FILE);
173                 else
174                         BIOerr(BIO_F_BIO_NEW_FILE,ERR_R_SYS_LIB);
175                 return(NULL);
176                 }
177         if ((ret=BIO_new(BIO_s_file())) == NULL)
178                 {
179                 fclose(file);
180                 return(NULL);
181                 }
182
183         BIO_clear_flags(ret,BIO_FLAGS_UPLINK); /* we did fopen -> we disengage U
184         BIO_set_fp(ret,file,BIO_CLOSE);
185         return(ret);
186         }
187
188 BIO *BIO_new_fp(FILE *stream, int close_flag)
189         {
190         BIO *ret;
191
192         if ((ret=BIO_new(BIO_s_file())) == NULL)
193                 return(NULL);
```



```
195         BIO_set_flags(ret,BIO_FLAGS_UPLINK); /* redundant, left for documentatio
196         BIO_set_fp(ret,stream,close_flag);
197         return(ret);
198         }
199
200 BIO_METHOD *BIO_s_file(void)
201         {
202         return(&methods_filep);
203         }
204
205 static int MS_CALLBACK file_new(BIO *bi)
206         {
207         bi->init=0;
208         bi->num=0;
209         bi->ptr=NULL;
210         bi->flags=BIO_FLAGS_UPLINK; /* default to UPLINK */
211         return(1);
212         }
213
214 static int MS_CALLBACK file_free(BIO *a)
215         {
216         if (a == NULL) return(0);
217         if (a->shutdown)
218                 {
219                 if ((a->init) && (a->ptr != NULL))
220                         {
221                         if (a->flags&BIO_FLAGS_UPLINK)
222                                 UP_fclose (a->ptr);
223                         else
224                                 fclose (a->ptr);
225                         a->ptr=NULL;
226                         a->flags=BIO_FLAGS_UPLINK;
227                         }
228                 a->init=0;
229                 }
230         return(1);
231         }
232
233 static int MS_CALLBACK file_read(BIO *b, char *out, int outl)
234         {
235         int ret=0;
236
237         if (b->init && (out != NULL))
238                 {
239                 if (b->flags&BIO_FLAGS_UPLINK)
240                         ret=UP_fread(out,1,(int)outl,b->ptr);
241                 else
242                         ret=fread(out,1,(int)outl,(FILE *)b->ptr);
243                 if(ret == 0 && (b->flags&BIO_FLAGS_UPLINK)?UP_ferror((FILE *)b->
244                         {
245                         SYSerr(SYS_F_FREAD,get_last_sys_error());
246                         BIOerr(BIO_F_FILE_READ,ERR_R_SYS_LIB);
247                         ret=-1;
248                         }
249                 }
250         return(ret);
251         }
252
253 static int MS_CALLBACK file_write(BIO *b, const char *in, int inl)
254         {
255         int ret=0;
256
257         if (b->init && (in != NULL))
258                 {
259                 if (b->flags&BIO_FLAGS_UPLINK)
```

```
260                         ret=UP_fwrite(in,(int)inl,1,b->ptr);
261                 else
262                         ret=fwrite(in,(int)inl,1,(FILE *)b->ptr);
263                 if (ret)
264                         ret=inl;
265                 /* ret=fwrite(in,1,(int)inl,(FILE *)b->ptr); */
266                 /* according to Tim Hudson <tjh@cryptsoft.com>, the commented
267                  * out version above can cause 'inl' write calls under
268                  * some stupid stdio implementations (VMS) */
269                 }
270         return(ret);
271         }

273 static long MS_CALLBACK file_ctrl(BIO *b, int cmd, long num, void *ptr)
274         {
275         long ret=1;
276         FILE *fp=(FILE *)b->ptr;
277         FILE **fpp;
278         char p[4];

280         switch (cmd)
281                 {
282         case BIO_C_FILE_SEEK:
283         case BIO_CTRL_RESET:
284                 if (b->flags&BIO_FLAGS_UPLINK)
285                         ret=(long)UP_fseek(b->ptr,num,0);
286                 else
287                         ret=(long)fseek(fp,num,0);
288                 break;
289         case BIO_CTRL_EOF:
290                 if (b->flags&BIO_FLAGS_UPLINK)
291                         ret=(long)UP_feof(fp);
292                 else
293                         ret=(long)feof(fp);
294                 break;
295         case BIO_C_FILE_TELL:
296         case BIO_CTRL_INFO:
297                 if (b->flags&BIO_FLAGS_UPLINK)
298                         ret=UP_ftell(b->ptr);
299                 else
300                         ret=ftell(fp);
301                 break;
302         case BIO_C_SET_FILE_PTR:
303                 file_free(b);
304                 b->shutdown=(int)num&BIO_CLOSE;
305                 b->ptr=ptr;
306                 b->init=1;
307 #if BIO_FLAGS_UPLINK!=0
308 #if defined(__MINGW32__) && defined(__MSVCRT__) && !defined(_IOB_ENTRIES)
309 #define _IOB_ENTRIES 20
310 #endif
311 #if defined(_IOB_ENTRIES)
312                 /* Safety net to catch purely internal BIO_set_fp calls */
313                 if ((size_t)ptr >= (size_t)stdin &&
314                     (size_t)ptr <  (size_t)(stdin+_IOB_ENTRIES))
315                         BIO_clear_flags(b,BIO_FLAGS_UPLINK);
316 #endif
317 #endif
318 #ifdef UP_fsetmod
319                 if (b->flags&BIO_FLAGS_UPLINK)
320                         UP_fsetmod(b->ptr,(char)((num&BIO_FP_TEXT)?'t':'b'));
321                 else
322 #endif
323                 {
324 #if defined(OPENSSL_SYS_WINDOWS)
325                 int fd = _fileno((FILE*)ptr);
```



```
326                 if (num & BIO_FP_TEXT)
327                         _setmode(fd,_O_TEXT);
328                 else
329                         _setmode(fd,_O_BINARY);
330 #elif defined(OPENSSL_SYS_NETWARE) && defined(NETWARE_CLIB)
331                 int fd = fileno((FILE*)ptr);
332                 /* Under CLib there are differences in file modes */
333                 if (num & BIO_FP_TEXT)
334                         setmode(fd,O_TEXT);
335                 else
336                         setmode(fd,O_BINARY);
337 #elif defined(OPENSSL_SYS_MSDOS)
338                 int fd = fileno((FILE*)ptr);
339                 /* Set correct text/binary mode */
340                 if (num & BIO_FP_TEXT)
341                         _setmode(fd,_O_TEXT);
342                 /* Dangerous to set stdin/stdout to raw (unless redirected) */
343                 else
344                         {
345                         if (fd == STDIN_FILENO || fd == STDOUT_FILENO)
346                                 {
347                                 if (isatty(fd) <= 0)
348                                         _setmode(fd,_O_BINARY);
349                                 }
350                         else
351                                 _setmode(fd,_O_BINARY);
352                         }
353 #elif defined(OPENSSL_SYS_OS2) || defined(OPENSSL_SYS_WIN32_CYGWIN)
354                 int fd = fileno((FILE*)ptr);
355                 if (num & BIO_FP_TEXT)
356                         setmode(fd, O_TEXT);
357                 else
358                         setmode(fd, O_BINARY);
359 #endif
360                 }
361                 break;
362         case BIO_C_SET_FILENAME:
363                 file_free(b);
364                 b->shutdown=(int)num&BIO_CLOSE;
365                 if (num & BIO_FP_APPEND)
366                         {
367                         if (num & BIO_FP_READ)
368                                 BUF_strlcpy(p,"a+",sizeof p);
369                         else    BUF_strlcpy(p,"a",sizeof p);
370                         }
371                 else if ((num & BIO_FP_READ) && (num & BIO_FP_WRITE))
372                         BUF_strlcpy(p,"r+",sizeof p);
373                 else if (num & BIO_FP_WRITE)
374                         BUF_strlcpy(p,"w",sizeof p);
375                 else if (num & BIO_FP_READ)
376                         BUF_strlcpy(p,"r",sizeof p);
377                 else
378                         {
379                         BIOerr(BIO_F_FILE_CTRL,BIO_R_BAD_FOPEN_MODE);
380                         ret=0;
381                         break;
382                         }
383 #if defined(OPENSSL_SYS_MSDOS) || defined(OPENSSL_SYS_WINDOWS) || defined(OPENSS
384                 if (!(num & BIO_FP_TEXT))
385                         strcat(p,"b");
386                 else
387                         strcat(p,"t");
388 #endif
389 #if defined(OPENSSL_SYS_NETWARE)
390                 if (!(num & BIO_FP_TEXT))
391                         strcat(p,"b");
```

```
392                 else
393                         strcat(p,"t");
394 #endif
395                 fp=fopen(ptr,p);
396                 if (fp == NULL)
397                         {
398                         SYSerr(SYS_F_FOPEN,get_last_sys_error());
399                         ERR_add_error_data(5,"fopen('",ptr,"','",p,"')");
400                         BIOerr(BIO_F_FILE_CTRL,ERR_R_SYS_LIB);
401                         ret=0;
402                         break;
403                         }
404                 b->ptr=fp;
405                 b->init=1;
406                 BIO_clear_flags(b,BIO_FLAGS_UPLINK); /* we did fopen -> we disen
407                 break;
408         case BIO_C_GET_FILE_PTR:
409                 /* the ptr parameter is actually a FILE ** in this case. */
410                 if (ptr != NULL)
411                         {
412                         fpp=(FILE **)ptr;
413                         *fpp=(FILE *)b->ptr;
414                         }
415                 break;
416         case BIO_CTRL_GET_CLOSE:
417                 ret=(long)b->shutdown;
418                 break;
419         case BIO_CTRL_SET_CLOSE:
420                 b->shutdown=(int)num;
421                 break;
422         case BIO_CTRL_FLUSH:
423                 if (b->flags&BIO_FLAGS_UPLINK)
424                         UP_fflush(b->ptr);
425                 else
426                         fflush((FILE *)b->ptr);
427                 break;
428         case BIO_CTRL_DUP:
429                 ret=1;
430                 break;

432         case BIO_CTRL_WPENDING:
433         case BIO_CTRL_PENDING:
434         case BIO_CTRL_PUSH:
435         case BIO_CTRL_POP:
436         default:
437                 ret=0;
438                 break;
439                 }
440         return(ret);
441         }

443 static int MS_CALLBACK file_gets(BIO *bp, char *buf, int size)
444         {
445         int ret=0;

447         buf[0]='\0';
448         if (bp->flags&BIO_FLAGS_UPLINK)
449                 {
450                 if (!UP_fgets(buf,size,bp->ptr))
451                         goto err;
452                 }
453         else
454                 {
455                 if (!fgets(buf,size,(FILE *)bp->ptr))
456                         goto err;
457                 }
```



```
458         if (buf[0] != '\0')
459                 ret=strlen(buf);
460         err:
461         return(ret);
462         }

464 static int MS_CALLBACK file_puts(BIO *bp, const char *str)
465         {
466         int n,ret;

468         n=strlen(str);
469         ret=file_write(bp,str,n);
470         return(ret);
471         }

473 #endif /* OPENSSL_NO_STDIO */

475 #endif /* HEADER_BSS_FILE_C */
476 #endif /* ! codereview */
```

```
**********************************************************
   10270 Wed Aug 13 19:52:13 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_log.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_log.c */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/*
        Why BIO_s_log?

        BIO_s_log is useful for system daemons (or services under NT).
        It is one-way BIO, it sends all stuff to syslogd (on system that
        commonly use that), or event log (on NT), or OPCOM (on OpenVMS).
```



```
 */


#include <stdio.h>
#include <errno.h>

#include "cryptlib.h"

#if defined(OPENSSL_SYS_WINCE)
#elif defined(OPENSSL_SYS_WIN32)
#elif defined(OPENSSL_SYS_VMS)
#  include <opcdef.h>
#  include <descrip.h>
#  include <lib$routines.h>
#  include <starlet.h>
/* Some compiler options may mask the declaration of "_malloc32". */
#  if __INITIAL_POINTER_SIZE && defined _ANSI_C_SOURCE
#    if __INITIAL_POINTER_SIZE == 64
#      pragma pointer_size save
#      pragma pointer_size 32
    void * _malloc32  (__size_t);
#      pragma pointer_size restore
#    endif /* __INITIAL_POINTER_SIZE == 64 */
#  endif /* __INITIAL_POINTER_SIZE && defined _ANSI_C_SOURCE */
#elif defined(__ultrix)
#  include <sys/syslog.h>
#elif defined(OPENSSL_SYS_NETWARE)
#  define NO_SYSLOG
#elif (!defined(MSDOS) || defined(WATT32)) && !defined(OPENSSL_SYS_VXWORKS) && !
#  include <syslog.h>
#endif

#include <openssl/buffer.h>
#include <openssl/err.h>

#ifndef NO_SYSLOG

#if defined(OPENSSL_SYS_WIN32)
#define LOG_EMERG       0
#define LOG_ALERT       1
#define LOG_CRIT        2
#define LOG_ERR         3
#define LOG_WARNING     4
#define LOG_NOTICE      5
#define LOG_INFO        6
#define LOG_DEBUG       7

#define LOG_DAEMON      (3<<3)
#elif defined(OPENSSL_SYS_VMS)
/* On VMS, we don't really care about these, but we need them to compile */
#define LOG_EMERG       0
#define LOG_ALERT       1
#define LOG_CRIT        2
#define LOG_ERR         3
#define LOG_WARNING     4
#define LOG_NOTICE      5
#define LOG_INFO        6
#define LOG_DEBUG       7

#define LOG_DAEMON      OPC$M_NM_NTWORK
#endif

static int MS_CALLBACK slg_write(BIO *h, const char *buf, int num);
static int MS_CALLBACK slg_puts(BIO *h, const char *str);
static long MS_CALLBACK slg_ctrl(BIO *h, int cmd, long arg1, void *arg2);
```

```
128 static int MS_CALLBACK slg_new(BIO *h);
129 static int MS_CALLBACK slg_free(BIO *data);
130 static void xopenlog(BIO* bp, char* name, int level);
131 static void xsyslog(BIO* bp, int priority, const char* string);
132 static void xcloselog(BIO* bp);

134 static BIO_METHOD methods_slg=
135         {
136         BIO_TYPE_MEM,"syslog",
137         slg_write,
138         NULL,
139         slg_puts,
140         NULL,
141         slg_ctrl,
142         slg_new,
143         slg_free,
144         NULL,
145         };

147 BIO_METHOD *BIO_s_log(void)
148         {
149         return(&methods_slg);
150         }

152 static int MS_CALLBACK slg_new(BIO *bi)
153         {
154         bi->init=1;
155         bi->num=0;
156         bi->ptr=NULL;
157         xopenlog(bi, "application", LOG_DAEMON);
158         return(1);
159         }

161 static int MS_CALLBACK slg_free(BIO *a)
162         {
163         if (a == NULL) return(0);
164         xcloselog(a);
165         return(1);
166         }

168 static int MS_CALLBACK slg_write(BIO *b, const char *in, int inl)
169         {
170         int ret= inl;
171         char* buf;
172         char* pp;
173         int priority, i;
174         static const struct
175                 {
176                 int strl;
177                 char str[10];
178                 int log_level;
179                 }
180         mapping[] =
181                 {
182                 { 6, "PANIC ", LOG_EMERG },
183                 { 6, "EMERG ", LOG_EMERG },
184                 { 4, "EMR ", LOG_EMERG },
185                 { 6, "ALERT ", LOG_ALERT },
186                 { 4, "ALR ", LOG_ALERT },
187                 { 5, "CRIT ", LOG_CRIT },
188                 { 4, "CRI ", LOG_CRIT },
189                 { 6, "ERROR ", LOG_ERR },
190                 { 4, "ERR ", LOG_ERR },
191                 { 8, "WARNING ", LOG_WARNING },
192                 { 5, "WARN ", LOG_WARNING },
193                 { 4, "WAR ", LOG_WARNING },
```



```
194                 { 7, "NOTICE ", LOG_NOTICE },
195                 { 5, "NOTE ", LOG_NOTICE },
196                 { 4, "NOT ", LOG_NOTICE },
197                 { 5, "INFO ", LOG_INFO },
198                 { 4, "INF ", LOG_INFO },
199                 { 6, "DEBUG ", LOG_DEBUG },
200                 { 4, "DBG ", LOG_DEBUG },
201                 { 0, "", LOG_ERR } /* The default */
202                 };

204         if((buf= (char *)OPENSSL_malloc(inl+ 1)) == NULL){
205                 return(0);
206         }
207         strncpy(buf, in, inl);
208         buf[inl]= '\0';

210         i = 0;
211         while(strncmp(buf, mapping[i].str, mapping[i].strl) != 0) i++;
212         priority = mapping[i].log_level;
213         pp = buf + mapping[i].strl;

215         xsyslog(b, priority, pp);

217         OPENSSL_free(buf);
218         return(ret);
219         }

221 static long MS_CALLBACK slg_ctrl(BIO *b, int cmd, long num, void *ptr)
222         {
223         switch (cmd)
224                 {
225         case BIO_CTRL_SET:
226                 xcloselog(b);
227                 xopenlog(b, ptr, num);
228                 break;
229         default:
230                 break;
231                 }
232         return(0);
233         }

235 static int MS_CALLBACK slg_puts(BIO *bp, const char *str)
236         {
237         int n,ret;

239         n=strlen(str);
240         ret=slg_write(bp,str,n);
241         return(ret);
242         }

244 #if defined(OPENSSL_SYS_WIN32)

246 static void xopenlog(BIO* bp, char* name, int level)
247 {
248         if (check_winnt())
249                 bp->ptr = RegisterEventSourceA(NULL,name);
250         else
251                 bp->ptr = NULL;
252 }

254 static void xsyslog(BIO *bp, int priority, const char *string)
255 {
256         LPCSTR lpszStrings[2];
257         WORD evtype= EVENTLOG_ERROR_TYPE;
258         char pidbuf[DECIMAL_SIZE(DWORD)+4];
```

```
260         if (bp->ptr == NULL)
261                 return;

263         switch (priority)
264                 {
265         case LOG_EMERG:
266         case LOG_ALERT:
267         case LOG_CRIT:
268         case LOG_ERR:
269                 evtype = EVENTLOG_ERROR_TYPE;
270                 break;
271         case LOG_WARNING:
272                 evtype = EVENTLOG_WARNING_TYPE;
273                 break;
274         case LOG_NOTICE:
275         case LOG_INFO:
276         case LOG_DEBUG:
277                 evtype = EVENTLOG_INFORMATION_TYPE;
278                 break;
279         default:                /* Should never happen, but set it
280                                    as error anyway. */
281                 evtype = EVENTLOG_ERROR_TYPE;
282                 break;
283                 }

285         sprintf(pidbuf, "[%u] ", GetCurrentProcessId());
286         lpszStrings[0] = pidbuf;
287         lpszStrings[1] = string;

289         ReportEventA(bp->ptr, evtype, 0, 1024, NULL, 2, 0,
290                                 lpszStrings, NULL);
291 }

293 static void xcloselog(BIO* bp)
294 {
295         if(bp->ptr)
296                 DeregisterEventSource((HANDLE)(bp->ptr));
297         bp->ptr= NULL;
298 }

300 #elif defined(OPENSSL_SYS_VMS)

302 static int VMS_OPC_target = LOG_DAEMON;

304 static void xopenlog(BIO* bp, char* name, int level)
305 {
306         VMS_OPC_target = level;
307 }

309 static void xsyslog(BIO *bp, int priority, const char *string)
310 {
311         struct dsc$descriptor_s opc_dsc;

313 /* Arrange 32-bit pointer to opcdef buffer and malloc(), if needed. */
314 #if __INITIAL_POINTER_SIZE == 64
315 # pragma pointer_size save
316 # pragma pointer_size 32
317 # define OPCDEF_TYPE __char_ptr32
318 # define OPCDEF_MALLOC _malloc32
319 #else /* __INITIAL_POINTER_SIZE == 64 */
320 # define OPCDEF_TYPE char *
321 # define OPCDEF_MALLOC OPENSSL_malloc
322 #endif /* __INITIAL_POINTER_SIZE == 64 [else] */

324         struct opcdef *opcdef_p;
```



```
326 #if __INITIAL_POINTER_SIZE == 64
327 # pragma pointer_size restore
328 #endif /* __INITIAL_POINTER_SIZE == 64 */

330         char buf[10240];
331         unsigned int len;
332         struct dsc$descriptor_s buf_dsc;
333         $DESCRIPTOR(fao_cmd, "!AZ: !AZ");
334         char *priority_tag;

336         switch (priority)
337           {
338           case LOG_EMERG: priority_tag = "Emergency"; break;
339           case LOG_ALERT: priority_tag = "Alert"; break;
340           case LOG_CRIT: priority_tag = "Critical"; break;
341           case LOG_ERR: priority_tag = "Error"; break;
342           case LOG_WARNING: priority_tag = "Warning"; break;
343           case LOG_NOTICE: priority_tag = "Notice"; break;
344           case LOG_INFO: priority_tag = "Info"; break;
345           case LOG_DEBUG: priority_tag = "DEBUG"; break;
346           }

348         buf_dsc.dsc$b_dtype = DSC$K_DTYPE_T;
349         buf_dsc.dsc$b_class = DSC$K_CLASS_S;
350         buf_dsc.dsc$a_pointer = buf;
351         buf_dsc.dsc$w_length = sizeof(buf) - 1;

353         lib$sys_fao(&fao_cmd, &len, &buf_dsc, priority_tag, string);

355         /* We know there's an 8-byte header.  That's documented. */
356         opcdef_p = OPCDEF_MALLOC( 8+ len);
357         opcdef_p->opc$b_ms_type = OPC$_RQ_RQST;
358         memcpy(opcdef_p->opc$z_ms_target_classes, &VMS_OPC_target, 3);
359         opcdef_p->opc$l_ms_rqstid = 0;
360         memcpy(&opcdef_p->opc$l_ms_text, buf, len);

362         opc_dsc.dsc$b_dtype = DSC$K_DTYPE_T;
363         opc_dsc.dsc$b_class = DSC$K_CLASS_S;
364         opc_dsc.dsc$a_pointer = (OPCDEF_TYPE) opcdef_p;
365         opc_dsc.dsc$w_length = len + 8;

367         sys$sndopr(opc_dsc, 0);

369         OPENSSL_free(opcdef_p);
370 }

372 static void xcloselog(BIO* bp)
373 {
374 }

376 #else /* Unix/Watt32 */

378 static void xopenlog(BIO* bp, char* name, int level)
379 {
380 #ifdef WATT32   /* djgpp/DOS */
381         openlog(name, LOG_PID|LOG_CONS|LOG_NDELAY, level);
382 #else
383         openlog(name, LOG_PID|LOG_CONS, level);
384 #endif
385 }

387 static void xsyslog(BIO *bp, int priority, const char *string)
388 {
389         syslog(priority, "%s", string);
390 }
```

```
392 static void xcloselog(BIO* bp)
393 {
394         closelog();
395 }

397 #endif /* Unix */

399 #endif /* NO_SYSLOG */
400 #endif /* ! codereview */
```

```
**********************************************************
    7785 Wed Aug 13 19:52:13 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_mem.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_mem.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/bio.h>

static int mem_write(BIO *h, const char *buf, int num);
static int mem_read(BIO *h, char *buf, int size);
static int mem_puts(BIO *h, const char *str);
static int mem_gets(BIO *h, char *str, int size);
static long mem_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int mem_new(BIO *h);
static int mem_free(BIO *data);
static BIO_METHOD mem_method=
        {
        BIO_TYPE_MEM,
        "memory buffer",
        mem_write,
        mem_read,
        mem_puts,
        mem_gets,
        mem_ctrl,
        mem_new,
        mem_free,
        NULL,
        };

/* bio->num is used to hold the value to return on 'empty', if it is
 * 0, should_retry is not set */

BIO_METHOD *BIO_s_mem(void)
        {
        return(&mem_method);
        }

BIO *BIO_new_mem_buf(void *buf, int len)
{
        BIO *ret;
        BUF_MEM *b;
        size_t sz;

        if (!buf) {
                BIOerr(BIO_F_BIO_NEW_MEM_BUF,BIO_R_NULL_PARAMETER);
                return NULL;
        }
        sz = (len<0) ? strlen(buf) : (size_t)len;
        if(!(ret = BIO_new(BIO_s_mem())) ) return NULL;
        b = (BUF_MEM *)ret->ptr;
        b->data = buf;
        b->length = sz;
        b->max = sz;
        ret->flags |= BIO_FLAGS_MEM_RDONLY;
        /* Since this is static data retrying wont help */
        ret->num = 0;
        return ret;
}

static int mem_new(BIO *bi)
        {
        BUF_MEM *b;

        if ((b=BUF_MEM_new()) == NULL)
                return(0);
        bi->shutdown=1;
        bi->init=1;
        bi->num= -1;
        bi->ptr=(char *)b;
        return(1);
        }
```

```
128 static int mem_free(BIO *a)
129         {
130         if (a == NULL) return(0);
131         if (a->shutdown)
132                 {
133                 if ((a->init) && (a->ptr != NULL))
134                         {
135                         BUF_MEM *b;
136                         b = (BUF_MEM *)a->ptr;
137                         if(a->flags & BIO_FLAGS_MEM_RDONLY) b->data = NULL;
138                         BUF_MEM_free(b);
139                         a->ptr=NULL;
140                         }
141                 }
142         return(1);
143         }

145 static int mem_read(BIO *b, char *out, int outl)
146         {
147         int ret= -1;
148         BUF_MEM *bm;

150         bm=(BUF_MEM *)b->ptr;
151         BIO_clear_retry_flags(b);
152         ret=(outl >=0 && (size_t)outl > bm->length)?(int)bm->length:outl;
153         if ((out != NULL) && (ret > 0)) {
154                 memcpy(out,bm->data,ret);
155                 bm->length-=ret;
156                 if(b->flags & BIO_FLAGS_MEM_RDONLY) bm->data += ret;
157                 else {
158                         memmove(&(bm->data[0]),&(bm->data[ret]),bm->length);
159                 }
160         } else if (bm->length == 0)
161                 {
162                 ret = b->num;
163                 if (ret != 0)
164                         BIO_set_retry_read(b);
165                 }
166         return(ret);
167         }

169 static int mem_write(BIO *b, const char *in, int inl)
170         {
171         int ret= -1;
172         int blen;
173         BUF_MEM *bm;

175         bm=(BUF_MEM *)b->ptr;
176         if (in == NULL)
177                 {
178                 BIOerr(BIO_F_MEM_WRITE,BIO_R_NULL_PARAMETER);
179                 goto end;
180                 }

182         if(b->flags & BIO_FLAGS_MEM_RDONLY) {
183                 BIOerr(BIO_F_MEM_WRITE,BIO_R_WRITE_TO_READ_ONLY_BIO);
184                 goto end;
185         }

187         BIO_clear_retry_flags(b);
188         blen=bm->length;
189         if (BUF_MEM_grow_clean(bm,blen+inl) != (blen+inl))
190                 goto end;
191         memcpy(&(bm->data[blen]),in,inl);
192         ret=inl;
193 end:
```



```
194         return(ret);
195         }

197 static long mem_ctrl(BIO *b, int cmd, long num, void *ptr)
198         {
199         long ret=1;
200         char **pptr;

202         BUF_MEM *bm=(BUF_MEM *)b->ptr;

204         switch (cmd)
205                 {
206         case BIO_CTRL_RESET:
207                 if (bm->data != NULL)
208                         {
209                         /* For read only case reset to the start again */
210                         if(b->flags & BIO_FLAGS_MEM_RDONLY)
211                                 {
212                                 bm->data -= bm->max - bm->length;
213                                 bm->length = bm->max;
214                                 }
215                         else
216                                 {
217                                 memset(bm->data,0,bm->max);
218                                 bm->length=0;
219                                 }
220                         }
221                 break;
222         case BIO_CTRL_EOF:
223                 ret=(long)(bm->length == 0);
224                 break;
225         case BIO_C_SET_BUF_MEM_EOF_RETURN:
226                 b->num=(int)num;
227                 break;
228         case BIO_CTRL_INFO:
229                 ret=(long)bm->length;
230                 if (ptr != NULL)
231                         {
232                         pptr=(char **)ptr;
233                         *pptr=(char *)&(bm->data[0]);
234                         }
235                 break;
236         case BIO_C_SET_BUF_MEM:
237                 mem_free(b);
238                 b->shutdown=(int)num;
239                 b->ptr=ptr;
240                 break;
241         case BIO_C_GET_BUF_MEM_PTR:
242                 if (ptr != NULL)
243                         {
244                         pptr=(char **)ptr;
245                         *pptr=(char *)bm;
246                         }
247                 break;
248         case BIO_CTRL_GET_CLOSE:
249                 ret=(long)b->shutdown;
250                 break;
251         case BIO_CTRL_SET_CLOSE:
252                 b->shutdown=(int)num;
253                 break;

255         case BIO_CTRL_WPENDING:
256                 ret=0L;
257                 break;
258         case BIO_CTRL_PENDING:
259                 ret=(long)bm->length;
```

```
260                 break;
261         case BIO_CTRL_DUP:
262         case BIO_CTRL_FLUSH:
263                 ret=1;
264                 break;
265         case BIO_CTRL_PUSH:
266         case BIO_CTRL_POP:
267         default:
268                 ret=0;
269                 break;
270                 }
271         return(ret);
272         }

274 static int mem_gets(BIO *bp, char *buf, int size)
275         {
276         int i,j;
277         int ret= -1;
278         char *p;
279         BUF_MEM *bm=(BUF_MEM *)bp->ptr;

281         BIO_clear_retry_flags(bp);
282         j=bm->length;
283         if ((size-1) < j) j=size-1;
284         if (j <= 0)
285                 {
286                 *buf='\0';
287                 return 0;
288                 }
289         p=bm->data;
290         for (i=0; i<j; i++)
291                 {
292                 if (p[i] == '\n')
293                         {
294                         i++;
295                         break;
296                         }
297                 }

299         /*
300          * i is now the max num of bytes to copy, either j or up to
301          * and including the first newline
302          */

304         i=mem_read(bp,buf,i);
305         if (i > 0) buf[i]='\0';
306         ret=i;
307         return(ret);
308         }

310 static int mem_puts(BIO *bp, const char *str)
311         {
312         int n,ret;

314         n=strlen(str);
315         ret=mem_write(bp,str,n);
316         /* memory semantics is that it will always work */
317         return(ret);
318         }
319 #endif /* ! codereview */
```

```
**********************************************************
    4716 Wed Aug 13 19:52:13 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_null.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_null.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/bio.h>

static int null_write(BIO *h, const char *buf, int num);
static int null_read(BIO *h, char *buf, int size);
static int null_puts(BIO *h, const char *str);
static int null_gets(BIO *h, char *str, int size);
static long null_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int null_new(BIO *h);
static int null_free(BIO *data);
static BIO_METHOD null_method=
        {
        BIO_TYPE_NULL,
        "NULL",
        null_write,
        null_read,
        null_puts,
        null_gets,
        null_ctrl,
        null_new,
        null_free,
        NULL,
        };

BIO_METHOD *BIO_s_null(void)
        {
        return(&null_method);
        }

static int null_new(BIO *bi)
        {
        bi->init=1;
        bi->num=0;
        bi->ptr=(NULL);
        return(1);
        }

static int null_free(BIO *a)
        {
        if (a == NULL) return(0);
        return(1);
        }

static int null_read(BIO *b, char *out, int outl)
        {
        return(0);
        }

static int null_write(BIO *b, const char *in, int inl)
        {
        return(inl);
        }

static long null_ctrl(BIO *b, int cmd, long num, void *ptr)
        {
        long ret=1;

        switch (cmd)
                {
        case BIO_CTRL_RESET:
        case BIO_CTRL_EOF:
        case BIO_CTRL_SET:
        case BIO_CTRL_SET_CLOSE:
        case BIO_CTRL_FLUSH:
        case BIO_CTRL_DUP:
                ret=1;
                break;
```

```
128         case BIO_CTRL_GET_CLOSE:
129         case BIO_CTRL_INFO:
130         case BIO_CTRL_GET:
131         case BIO_CTRL_PENDING:
132         case BIO_CTRL_WPENDING:
133         default:
134                 ret=0;
135                 break;
136                 }
137         return(ret);
138         }

140 static int null_gets(BIO *bp, char *buf, int size)
141         {
142         return(0);
143         }

145 static int null_puts(BIO *bp, const char *str)
146         {
147         if (str == NULL) return(0);
148         return(strlen(str));
149         }
150 #endif /* ! codereview */
```

```
**********************************************************
    6843 Wed Aug 13 19:52:13 2014
new/usr/src/lib/openssl/libsunw_crypto/bio/bss_sock.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bio/bss_sock.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#define USE_SOCKETS
```



```
#include "cryptlib.h"

#ifndef OPENSSL_NO_SOCK

#include <openssl/bio.h>

#ifdef WATT32
#define sock_write SockWrite  /* Watt-32 uses same names */
#define sock_read  SockRead
#define sock_puts  SockPuts
#endif

static int sock_write(BIO *h, const char *buf, int num);
static int sock_read(BIO *h, char *buf, int size);
static int sock_puts(BIO *h, const char *str);
static long sock_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int sock_new(BIO *h);
static int sock_free(BIO *data);
int BIO_sock_should_retry(int s);

static BIO_METHOD methods_sockp=
        {
        BIO_TYPE_SOCKET,
        "socket",
        sock_write,
        sock_read,
        sock_puts,
        NULL, /* sock_gets, */
        sock_ctrl,
        sock_new,
        sock_free,
        NULL,
        };

BIO_METHOD *BIO_s_socket(void)
        {
        return(&methods_sockp);
        }

BIO *BIO_new_socket(int fd, int close_flag)
        {
        BIO *ret;

        ret=BIO_new(BIO_s_socket());
        if (ret == NULL) return(NULL);
        BIO_set_fd(ret,fd,close_flag);
        return(ret);
        }

static int sock_new(BIO *bi)
        {
        bi->init=0;
        bi->num=0;
        bi->ptr=NULL;
        bi->flags=0;
        return(1);
        }

static int sock_free(BIO *a)
        {
        if (a == NULL) return(0);
        if (a->shutdown)
                {
                if (a->init)
                        {
                        SHUTDOWN2(a->num);
```

```
128                         }
129                 a->init=0;
130                 a->flags=0;
131                 }
132         return(1);
133         }

135 static int sock_read(BIO *b, char *out, int outl)
136         {
137         int ret=0;

139         if (out != NULL)
140                 {
141                 clear_socket_error();
142                 ret=readsocket(b->num,out,outl);
143                 BIO_clear_retry_flags(b);
144                 if (ret <= 0)
145                         {
146                         if (BIO_sock_should_retry(ret))
147                                 BIO_set_retry_read(b);
148                         }
149                 }
150         return(ret);
151         }

153 static int sock_write(BIO *b, const char *in, int inl)
154         {
155         int ret;

157         clear_socket_error();
158         ret=writesocket(b->num,in,inl);
159         BIO_clear_retry_flags(b);
160         if (ret <= 0)
161                 {
162                 if (BIO_sock_should_retry(ret))
163                         BIO_set_retry_write(b);
164                 }
165         return(ret);
166         }

168 static long sock_ctrl(BIO *b, int cmd, long num, void *ptr)
169         {
170         long ret=1;
171         int *ip;

173         switch (cmd)
174                 {
175         case BIO_C_SET_FD:
176                 sock_free(b);
177                 b->num= *((int *)ptr);
178                 b->shutdown=(int)num;
179                 b->init=1;
180                 break;
181         case BIO_C_GET_FD:
182                 if (b->init)
183                         {
184                         ip=(int *)ptr;
185                         if (ip != NULL) *ip=b->num;
186                         ret=b->num;
187                         }
188                 else
189                         ret= -1;
190                 break;
191         case BIO_CTRL_GET_CLOSE:
192                 ret=b->shutdown;
193                 break;
```



```
194         case BIO_CTRL_SET_CLOSE:
195                 b->shutdown=(int)num;
196                 break;
197         case BIO_CTRL_DUP:
198         case BIO_CTRL_FLUSH:
199                 ret=1;
200                 break;
201         default:
202                 ret=0;
203                 break;
204                 }
205         return(ret);
206         }

208 static int sock_puts(BIO *bp, const char *str)
209         {
210         int n,ret;

212         n=strlen(str);
213         ret=sock_write(bp,str,n);
214         return(ret);
215         }

217 int BIO_sock_should_retry(int i)
218         {
219         int err;

221         if ((i == 0) || (i == -1))
222                 {
223                 err=get_last_socket_error();

225 #if defined(OPENSSL_SYS_WINDOWS) && 0 /* more microsoft stupidity? perhaps not?
226                 if ((i == -1) && (err == 0))
227                         return(1);
228 #endif

230                 return(BIO_sock_non_fatal_error(err));
231                 }
232         return(0);
233         }

235 int BIO_sock_non_fatal_error(int err)
236         {
237         switch (err)
238                 {
239 #if defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_NETWARE)
240 # if defined(WSAEWOULDBLOCK)
241         case WSAEWOULDBLOCK:
242 # endif

244 # if 0 /* This appears to always be an error */
245 #  if defined(WSAENOTCONN)
246         case WSAENOTCONN:
247 #  endif
248 # endif
249 #endif

251 #ifdef EWOULDBLOCK
252 # ifdef WSAEWOULDBLOCK
253 #  if WSAEWOULDBLOCK != EWOULDBLOCK
254         case EWOULDBLOCK:
255 #  endif
256 # else
257         case EWOULDBLOCK:
258 # endif
259 #endif
```

```
#if defined(ENOTCONN)
        case ENOTCONN:
#endif

#ifdef EINTR
        case EINTR:
#endif

#ifdef EAGAIN
# if EWOULDBLOCK != EAGAIN
        case EAGAIN:
# endif
#endif

#ifdef EPROTO
        case EPROTO:
#endif

#ifdef EINPROGRESS
        case EINPROGRESS:
#endif

#ifdef EALREADY
        case EALREADY:
#endif
                return(1);
                /* break; */
        default:
                break;
                }
        return(0);
        }

#endif  /* #ifndef OPENSSL_NO_SOCK */
#endif /* ! codereview */
```

```
**********************************************************
    6857 Wed Aug 13 19:52:13 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_add.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_add.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"
```



```
/* r can == a or b */
int BN_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b)
        {
        const BIGNUM *tmp;
        int a_neg = a->neg, ret;

        bn_check_top(a);
        bn_check_top(b);

        /*  a +  b      a+b
         *  a + -b      a-b
         * -a +  b      b-a
         * -a + -b      -(a+b)
         */
        if (a_neg ^ b->neg)
                {
                /* only one is negative */
                if (a_neg)
                        { tmp=a; a=b; b=tmp; }

                /* we are now a - b */

                if (BN_ucmp(a,b) < 0)
                        {
                        if (!BN_usub(r,b,a)) return(0);
                        r->neg=1;
                        }
                else
                        {
                        if (!BN_usub(r,a,b)) return(0);
                        r->neg=0;
                        }
                return(1);
                }

        ret = BN_uadd(r,a,b);
        r->neg = a_neg;
        bn_check_top(r);
        return ret;
        }

/* unsigned add of b to a */
int BN_uadd(BIGNUM *r, const BIGNUM *a, const BIGNUM *b)
        {
        int max,min,dif;
        BN_ULONG *ap,*bp,*rp,carry,t1,t2;
        const BIGNUM *tmp;

        bn_check_top(a);
        bn_check_top(b);

        if (a->top < b->top)
                { tmp=a; a=b; b=tmp; }
        max = a->top;
        min = b->top;
        dif = max - min;

        if (bn_wexpand(r,max+1) == NULL)
                return 0;

        r->top=max;


        ap=a->d;
        bp=b->d;
```

```
128         rp=r->d;

130         carry=bn_add_words(rp,ap,bp,min);
131         rp+=min;
132         ap+=min;
133         bp+=min;

135         if (carry)
136                 {
137                 while (dif)
138                         {
139                         dif--;
140                         t1 = *(ap++);
141                         t2 = (t1+1) & BN_MASK2;
142                         *(rp++) = t2;
143                         if (t2)
144                                 {
145                                 carry=0;
146                                 break;
147                                 }
148                         }
149                 if (carry)
150                         {
151                         /* carry != 0 => dif == 0 */
152                         *rp = 1;
153                         r->top++;
154                         }
155                 }
156         if (dif && rp != ap)
157                 while (dif--)
158                         /* copy remaining words if ap != rp */
159                         *(rp++) = *(ap++);
160         r->neg = 0;
161         bn_check_top(r);
162         return 1;
163         }

165 /* unsigned subtraction of b from a, a must be larger than b. */
166 int BN_usub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b)
167         {
168         int max,min,dif;
169         register BN_ULONG t1,t2,*ap,*bp,*rp;
170         int i,carry;
171 #if defined(IRIX_CC_BUG) && !defined(LINT)
172         int dummy;
173 #endif

175         bn_check_top(a);
176         bn_check_top(b);

178         max = a->top;
179         min = b->top;
180         dif = max - min;

182         if (dif < 0)    /* hmm... should not be happening */
183                 {
184                 BNerr(BN_F_BN_USUB,BN_R_ARG2_LT_ARG3);
185                 return(0);
186                 }

188         if (bn_wexpand(r,max) == NULL) return(0);

190         ap=a->d;
191         bp=b->d;
192         rp=r->d;
```



```
194 #if 1
195         carry=0;
196         for (i = min; i != 0; i--)
197                 {
198                 t1= *(ap++);
199                 t2= *(bp++);
200                 if (carry)
201                         {
202                         carry=(t1 <= t2);
203                         t1=(t1-t2-1)&BN_MASK2;
204                         }
205                 else
206                         {
207                         carry=(t1 < t2);
208                         t1=(t1-t2)&BN_MASK2;
209                         }
210 #if defined(IRIX_CC_BUG) && !defined(LINT)
211                 dummy=t1;
212 #endif
213                 *(rp++)=t1&BN_MASK2;
214                 }
215 #else
216         carry=bn_sub_words(rp,ap,bp,min);
217         ap+=min;
218         bp+=min;
219         rp+=min;
220 #endif
221         if (carry) /* subtracted */
222                 {
223                 if (!dif)
224                         /* error: a < b */
225                         return 0;
226                 while (dif)
227                         {
228                         dif--;
229                         t1 = *(ap++);
230                         t2 = (t1-1)&BN_MASK2;
231                         *(rp++) = t2;
232                         if (t1)
233                                 break;
234                         }
235                 }
236 #if 0
237         memcpy(rp,ap,sizeof(*rp)*(max-i));
238 #else
239         if (rp != ap)
240                 {
241                 for (;;)
242                         {
243                         if (!dif--) break;
244                         rp[0]=ap[0];
245                         if (!dif--) break;
246                         rp[1]=ap[1];
247                         if (!dif--) break;
248                         rp[2]=ap[2];
249                         if (!dif--) break;
250                         rp[3]=ap[3];
251                         rp+=4;
252                         ap+=4;
253                         }
254                 }
255 #endif

257         r->top=max;
258         r->neg=0;
259         bn_correct_top(r);
```

```
260         return(1);
261         }

263 int BN_sub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b)
264         {
265         int max;
266         int add=0,neg=0;
267         const BIGNUM *tmp;

269         bn_check_top(a);
270         bn_check_top(b);

272         /*  a -  b      a-b
273          *  a - -b      a+b
274          * -a -  b      -(a+b)
275          * -a - -b      b-a
276          */
277         if (a->neg)
278                 {
279                 if (b->neg)
280                         { tmp=a; a=b; b=tmp; }
281                 else
282                         { add=1; neg=1; }
283                 }
284         else
285                 {
286                 if (b->neg) { add=1; neg=0; }
287                 }

289         if (add)
290                 {
291                 if (!BN_uadd(r,a,b)) return(0);
292                 r->neg=neg;
293                 return(1);
294                 }

296         /* We are actually doing a - b :-) */

298         max=(a->top > b->top)?a->top:b->top;
299         if (bn_wexpand(r,max) == NULL) return(0);
300         if (BN_ucmp(a,b) < 0)
301                 {
302                 if (!BN_usub(r,b,a)) return(0);
303                 r->neg=1;
304                 }
305         else
306                 {
307                 if (!BN_usub(r,a,b)) return(0);
308                 r->neg=0;
309                 }
310         bn_check_top(r);
311         return(1);
312         }
313 #endif /* ! codereview */
```

```
**********************************************************
   23191 Wed Aug 13 19:52:13 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_asm.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_asm.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef BN_DEBUG
# undef NDEBUG /* avoid conflicting definitions */
# define NDEBUG
```



```
#endif

#include <stdio.h>
#include <assert.h>
#include "cryptlib.h"
#include "bn_lcl.h"

#if defined(BN_LLONG) || defined(BN_UMULT_HIGH)

BN_ULONG bn_mul_add_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w)
        {
        BN_ULONG c1=0;

        assert(num >= 0);
        if (num <= 0) return(c1);

#ifndef OPENSSL_SMALL_FOOTPRINT
        while (num&~3)
                {
                mul_add(rp[0],ap[0],w,c1);
                mul_add(rp[1],ap[1],w,c1);
                mul_add(rp[2],ap[2],w,c1);
                mul_add(rp[3],ap[3],w,c1);
                ap+=4; rp+=4; num-=4;
                }
#endif
        while (num)
                {
                mul_add(rp[0],ap[0],w,c1);
                ap++; rp++; num--;
                }

        return(c1);
        }

BN_ULONG bn_mul_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w)
        {
        BN_ULONG c1=0;

        assert(num >= 0);
        if (num <= 0) return(c1);

#ifndef OPENSSL_SMALL_FOOTPRINT
        while (num&~3)
                {
                mul(rp[0],ap[0],w,c1);
                mul(rp[1],ap[1],w,c1);
                mul(rp[2],ap[2],w,c1);
                mul(rp[3],ap[3],w,c1);
                ap+=4; rp+=4; num-=4;
                }
#endif
        while (num)
                {
                mul(rp[0],ap[0],w,c1);
                ap++; rp++; num--;
                }
        return(c1);
        }

void bn_sqr_words(BN_ULONG *r, const BN_ULONG *a, int n)
        {
        assert(n >= 0);
        if (n <= 0) return;

#ifndef OPENSSL_SMALL_FOOTPRINT
```

```
128         while (n&~3)
129                 {
130                 sqr(r[0],r[1],a[0]);
131                 sqr(r[2],r[3],a[1]);
132                 sqr(r[4],r[5],a[2]);
133                 sqr(r[6],r[7],a[3]);
134                 a+=4; r+=8; n-=4;
135                 }
136 #endif
137         while (n)
138                 {
139                 sqr(r[0],r[1],a[0]);
140                 a++; r+=2; n--;
141                 }
142         }

144 #else /* !(defined(BN_LLONG) || defined(BN_UMULT_HIGH)) */

146 BN_ULONG bn_mul_add_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w)
147         {
148         BN_ULONG c=0;
149         BN_ULONG bl,bh;

151         assert(num >= 0);
152         if (num <= 0) return((BN_ULONG)0);

154         bl=LBITS(w);
155         bh=HBITS(w);

157 #ifndef OPENSSL_SMALL_FOOTPRINT
158         while (num&~3)
159                 {
160                 mul_add(rp[0],ap[0],bl,bh,c);
161                 mul_add(rp[1],ap[1],bl,bh,c);
162                 mul_add(rp[2],ap[2],bl,bh,c);
163                 mul_add(rp[3],ap[3],bl,bh,c);
164                 ap+=4; rp+=4; num-=4;
165                 }
166 #endif
167         while (num)
168                 {
169                 mul_add(rp[0],ap[0],bl,bh,c);
170                 ap++; rp++; num--;
171                 }
172         return(c);
173         }

175 BN_ULONG bn_mul_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w)
176         {
177         BN_ULONG carry=0;
178         BN_ULONG bl,bh;

180         assert(num >= 0);
181         if (num <= 0) return((BN_ULONG)0);

183         bl=LBITS(w);
184         bh=HBITS(w);

186 #ifndef OPENSSL_SMALL_FOOTPRINT
187         while (num&~3)
188                 {
189                 mul(rp[0],ap[0],bl,bh,carry);
190                 mul(rp[1],ap[1],bl,bh,carry);
191                 mul(rp[2],ap[2],bl,bh,carry);
192                 mul(rp[3],ap[3],bl,bh,carry);
193                 ap+=4; rp+=4; num-=4;
```



```
194                 }
195 #endif
196         while (num)
197                 {
198                 mul(rp[0],ap[0],bl,bh,carry);
199                 ap++; rp++; num--;
200                 }
201         return(carry);
202         }

204 void bn_sqr_words(BN_ULONG *r, const BN_ULONG *a, int n)
205         {
206         assert(n >= 0);
207         if (n <= 0) return;

209 #ifndef OPENSSL_SMALL_FOOTPRINT
210         while (n&~3)
211                 {
212                 sqr64(r[0],r[1],a[0]);
213                 sqr64(r[2],r[3],a[1]);
214                 sqr64(r[4],r[5],a[2]);
215                 sqr64(r[6],r[7],a[3]);
216                 a+=4; r+=8; n-=4;
217                 }
218 #endif
219         while (n)
220                 {
221                 sqr64(r[0],r[1],a[0]);
222                 a++; r+=2; n--;
223                 }
224         }

226 #endif /* !(defined(BN_LLONG) || defined(BN_UMULT_HIGH)) */

228 #if defined(BN_LLONG) && defined(BN_DIV2W)

230 BN_ULONG bn_div_words(BN_ULONG h, BN_ULONG l, BN_ULONG d)
231         {
232         return((BN_ULONG)(((((BN_ULLONG)h)<<BN_BITS2)|l)/(BN_ULLONG)d));
233         }

235 #else

237 /* Divide h,l by d and return the result. */
238 /* I need to test this some more :-( */
239 BN_ULONG bn_div_words(BN_ULONG h, BN_ULONG l, BN_ULONG d)
240         {
241         BN_ULONG dh,dl,q,ret=0,th,tl,t;
242         int i,count=2;

244         if (d == 0) return(BN_MASK2);

246         i=BN_num_bits_word(d);
247         assert((i == BN_BITS2) || (h <= (BN_ULONG)1<<i));

249         i=BN_BITS2-i;
250         if (h >= d) h-=d;

252         if (i)
253                 {
254                 d<<=i;
255                 h=(h<<i)|(l>>(BN_BITS2-i));
256                 l<<=i;
257                 }
258         dh=(d&BN_MASK2h)>>BN_BITS4;
259         dl=(d&BN_MASK2l);
```

```
260         for (;;)
261                 {
262                 if ((h>>BN_BITS4) == dh)
263                         q=BN_MASK2l;
264                 else
265                         q=h/dh;

267                 th=q*dh;
268                 tl=dl*q;
269                 for (;;)
270                         {
271                         t=h-th;
272                         if ((t&BN_MASK2h) ||
273                                 ((tl) <= (
274                                         (t<<BN_BITS4)|
275                                         ((l&BN_MASK2h)>>BN_BITS4))))
276                                 break;
277                         q--;
278                         th-=dh;
279                         tl-=dl;
280                         }
281                 t=(tl>>BN_BITS4);
282                 tl=(tl<<BN_BITS4)&BN_MASK2h;
283                 th+=t;

285                 if (l < tl) th++;
286                 l-=tl;
287                 if (h < th)
288                         {
289                         h+=d;
290                         q--;
291                         }
292                 h-=th;

294                 if (--count == 0) break;

296                 ret=q<<BN_BITS4;
297                 h=((h<<BN_BITS4)|(l>>BN_BITS4))&BN_MASK2;
298                 l=(l&BN_MASK2l)<<BN_BITS4;
299                 }
300         ret|=q;
301         return(ret);
302         }
303 #endif /* !defined(BN_LLONG) && defined(BN_DIV2W) */

305 #ifdef BN_LLONG
306 BN_ULONG bn_add_words(BN_ULONG *r, const BN_ULONG *a, const BN_ULONG *b, int n)
307         {
308         BN_ULLONG ll=0;

310         assert(n >= 0);
311         if (n <= 0) return((BN_ULONG)0);

313 #ifndef OPENSSL_SMALL_FOOTPRINT
314         while (n&~3)
315                 {
316                 ll+=(BN_ULLONG)a[0]+b[0];
317                 r[0]=(BN_ULONG)ll&BN_MASK2;
318                 ll>>=BN_BITS2;
319                 ll+=(BN_ULLONG)a[1]+b[1];
320                 r[1]=(BN_ULONG)ll&BN_MASK2;
321                 ll>>=BN_BITS2;
322                 ll+=(BN_ULLONG)a[2]+b[2];
323                 r[2]=(BN_ULONG)ll&BN_MASK2;
324                 ll>>=BN_BITS2;
325                 ll+=(BN_ULLONG)a[3]+b[3];
```



```
326                 r[3]=(BN_ULONG)ll&BN_MASK2;
327                 ll>>=BN_BITS2;
328                 a+=4; b+=4; r+=4; n-=4;
329                 }
330 #endif
331         while (n)
332                 {
333                 ll+=(BN_ULLONG)a[0]+b[0];
334                 r[0]=(BN_ULONG)ll&BN_MASK2;
335                 ll>>=BN_BITS2;
336                 a++; b++; r++; n--;
337                 }
338         return((BN_ULONG)ll);
339         }
340 #else /* !BN_LLONG */
341 BN_ULONG bn_add_words(BN_ULONG *r, const BN_ULONG *a, const BN_ULONG *b, int n)
342         {
343         BN_ULONG c,l,t;

345         assert(n >= 0);
346         if (n <= 0) return((BN_ULONG)0);

348         c=0;
349 #ifndef OPENSSL_SMALL_FOOTPRINT
350         while (n&~3)
351                 {
352                 t=a[0];
353                 t=(t+c)&BN_MASK2;
354                 c=(t < c);
355                 l=(t+b[0])&BN_MASK2;
356                 c+=(l < t);
357                 r[0]=l;
358                 t=a[1];
359                 t=(t+c)&BN_MASK2;
360                 c=(t < c);
361                 l=(t+b[1])&BN_MASK2;
362                 c+=(l < t);
363                 r[1]=l;
364                 t=a[2];
365                 t=(t+c)&BN_MASK2;
366                 c=(t < c);
367                 l=(t+b[2])&BN_MASK2;
368                 c+=(l < t);
369                 r[2]=l;
370                 t=a[3];
371                 t=(t+c)&BN_MASK2;
372                 c=(t < c);
373                 l=(t+b[3])&BN_MASK2;
374                 c+=(l < t);
375                 r[3]=l;
376                 a+=4; b+=4; r+=4; n-=4;
377                 }
378 #endif
379         while(n)
380                 {
381                 t=a[0];
382                 t=(t+c)&BN_MASK2;
383                 c=(t < c);
384                 l=(t+b[0])&BN_MASK2;
385                 c+=(l < t);
386                 r[0]=l;
387                 a++; b++; r++; n--;
388                 }
389         return((BN_ULONG)c);
390         }
391 #endif /* !BN_LLONG */
```

```
BN_ULONG bn_sub_words(BN_ULONG *r, const BN_ULONG *a, const BN_ULONG *b, int n)
        {
        BN_ULONG t1,t2;
        int c=0;

        assert(n >= 0);
        if (n <= 0) return((BN_ULONG)0);

#ifndef OPENSSL_SMALL_FOOTPRINT
        while (n&~3)
                {
                t1=a[0]; t2=b[0];
                r[0]=(t1-t2-c)&BN_MASK2;
                if (t1 != t2) c=(t1 < t2);
                t1=a[1]; t2=b[1];
                r[1]=(t1-t2-c)&BN_MASK2;
                if (t1 != t2) c=(t1 < t2);
                t1=a[2]; t2=b[2];
                r[2]=(t1-t2-c)&BN_MASK2;
                if (t1 != t2) c=(t1 < t2);
                t1=a[3]; t2=b[3];
                r[3]=(t1-t2-c)&BN_MASK2;
                if (t1 != t2) c=(t1 < t2);
                a+=4; b+=4; r+=4; n-=4;
                }
#endif
        while (n)
                {
                t1=a[0]; t2=b[0];
                r[0]=(t1-t2-c)&BN_MASK2;
                if (t1 != t2) c=(t1 < t2);
                a++; b++; r++; n--;
                }
        return(c);
        }

#if defined(BN_MUL_COMBA) && !defined(OPENSSL_SMALL_FOOTPRINT)

#undef bn_mul_comba8
#undef bn_mul_comba4
#undef bn_sqr_comba8
#undef bn_sqr_comba4

/* mul_add_c(a,b,c0,c1,c2)  -- c+=a*b for three word number c=(c2,c1,c0) */
/* mul_add_c2(a,b,c0,c1,c2) -- c+=2*a*b for three word number c=(c2,c1,c0) */
/* sqr_add_c(a,i,c0,c1,c2)  -- c+=a[i]^2 for three word number c=(c2,c1,c0) */
/* sqr_add_c2(a,i,c0,c1,c2) -- c+=2*a[i]*a[j] for three word number c=(c2,c1,c0)

#ifdef BN_LLONG
#define mul_add_c(a,b,c0,c1,c2) \
        t=(BN_ULLONG)a*b; \
        t1=(BN_ULONG)Lw(t); \
        t2=(BN_ULONG)Hw(t); \
        c0=(c0+t1)&BN_MASK2; if ((c0) < t1) t2++; \
        c1=(c1+t2)&BN_MASK2; if ((c1) < t2) c2++;

#define mul_add_c2(a,b,c0,c1,c2) \
        t=(BN_ULLONG)a*b; \
        tt=(t+t)&BN_MASK; \
        if (tt < t) c2++; \
        t1=(BN_ULONG)Lw(tt); \
        t2=(BN_ULONG)Hw(tt); \
        c0=(c0+t1)&BN_MASK2;  \
        if ((c0 < t1) && (((++t2)&BN_MASK2) == 0)) c2++; \
        c1=(c1+t2)&BN_MASK2; if ((c1) < t2) c2++;
```



```
#define sqr_add_c(a,i,c0,c1,c2) \
        t=(BN_ULLONG)a[i]*a[i]; \
        t1=(BN_ULONG)Lw(t); \
        t2=(BN_ULONG)Hw(t); \
        c0=(c0+t1)&BN_MASK2; if ((c0) < t1) t2++; \
        c1=(c1+t2)&BN_MASK2; if ((c1) < t2) c2++;

#define sqr_add_c2(a,i,j,c0,c1,c2) \
        mul_add_c2((a)[i],(a)[j],c0,c1,c2)

#elif defined(BN_UMULT_LOHI)

#define mul_add_c(a,b,c0,c1,c2) {       \
        BN_ULONG ta=(a),tb=(b);         \
        BN_UMULT_LOHI(t1,t2,ta,tb);     \
        c0 += t1; t2 += (c0<t1)?1:0;    \
        c1 += t2; c2 += (c1<t2)?1:0;    \
        }

#define mul_add_c2(a,b,c0,c1,c2) {      \
        BN_ULONG ta=(a),tb=(b),t0;      \
        BN_UMULT_LOHI(t0,t1,ta,tb);     \
        t2 = t1+t1; c2 += (t2<t1)?1:0;  \
        t1 = t0+t0; t2 += (t1<t0)?1:0;  \
        c0 += t1; t2 += (c0<t1)?1:0;    \
        c1 += t2; c2 += (c1<t2)?1:0;    \
        }

#define sqr_add_c(a,i,c0,c1,c2) {       \
        BN_ULONG ta=(a)[i];             \
        BN_UMULT_LOHI(t1,t2,ta,ta);     \
        c0 += t1; t2 += (c0<t1)?1:0;    \
        c1 += t2; c2 += (c1<t2)?1:0;    \
        }

#define sqr_add_c2(a,i,j,c0,c1,c2)      \
        mul_add_c2((a)[i],(a)[j],c0,c1,c2)

#elif defined(BN_UMULT_HIGH)

#define mul_add_c(a,b,c0,c1,c2) {       \
        BN_ULONG ta=(a),tb=(b);         \
        t1 = ta * tb;                   \
        t2 = BN_UMULT_HIGH(ta,tb);      \
        c0 += t1; t2 += (c0<t1)?1:0;    \
        c1 += t2; c2 += (c1<t2)?1:0;    \
        }

#define mul_add_c2(a,b,c0,c1,c2) {      \
        BN_ULONG ta=(a),tb=(b),t0;      \
        t1 = BN_UMULT_HIGH(ta,tb);      \
        t0 = ta * tb;                   \
        t2 = t1+t1; c2 += (t2<t1)?1:0;  \
        t1 = t0+t0; t2 += (t1<t0)?1:0;  \
        c0 += t1; t2 += (c0<t1)?1:0;    \
        c1 += t2; c2 += (c1<t2)?1:0;    \
        }

#define sqr_add_c(a,i,c0,c1,c2) {       \
        BN_ULONG ta=(a)[i];             \
        t1 = ta * ta;                   \
        t2 = BN_UMULT_HIGH(ta,ta);      \
        c0 += t1; t2 += (c0<t1)?1:0;    \
        c1 += t2; c2 += (c1<t2)?1:0;    \
        }
```

```
#define sqr_add_c2(a,i,j,c0,c1,c2)      \
        mul_add_c2((a)[i],(a)[j],c0,c1,c2)

#else /* !BN_LLONG */
#define mul_add_c(a,b,c0,c1,c2) \
        t1=LBITS(a); t2=HBITS(a); \
        bl=LBITS(b); bh=HBITS(b); \
        mul64(t1,t2,bl,bh); \
        c0=(c0+t1)&BN_MASK2; if ((c0) < t1) t2++; \
        c1=(c1+t2)&BN_MASK2; if ((c1) < t2) c2++;

#define mul_add_c2(a,b,c0,c1,c2) \
        t1=LBITS(a); t2=HBITS(a); \
        bl=LBITS(b); bh=HBITS(b); \
        mul64(t1,t2,bl,bh); \
        if (t2 & BN_TBIT) c2++; \
        t2=(t2+t2)&BN_MASK2; \
        if (t1 & BN_TBIT) t2++; \
        t1=(t1+t1)&BN_MASK2; \
        c0=(c0+t1)&BN_MASK2;  \
        if ((c0 < t1) && (((++t2)&BN_MASK2) == 0)) c2++; \
        c1=(c1+t2)&BN_MASK2; if ((c1) < t2) c2++;

#define sqr_add_c(a,i,c0,c1,c2) \
        sqr64(t1,t2,(a)[i]); \
        c0=(c0+t1)&BN_MASK2; if ((c0) < t1) t2++; \
        c1=(c1+t2)&BN_MASK2; if ((c1) < t2) c2++;

#define sqr_add_c2(a,i,j,c0,c1,c2) \
        mul_add_c2((a)[i],(a)[j],c0,c1,c2)
#endif /* !BN_LLONG */

void bn_mul_comba8(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b)
        {
#ifdef BN_LLONG
        BN_ULLONG t;
#else
        BN_ULONG bl,bh;
#endif
        BN_ULONG t1,t2;
        BN_ULONG c1,c2,c3;

        c1=0;
        c2=0;
        c3=0;
        mul_add_c(a[0],b[0],c1,c2,c3);
        r[0]=c1;
        c1=0;
        mul_add_c(a[0],b[1],c2,c3,c1);
        mul_add_c(a[1],b[0],c2,c3,c1);
        r[1]=c2;
        c2=0;
        mul_add_c(a[2],b[0],c3,c1,c2);
        mul_add_c(a[1],b[1],c3,c1,c2);
        mul_add_c(a[0],b[2],c3,c1,c2);
        r[2]=c3;
        c3=0;
        mul_add_c(a[0],b[3],c1,c2,c3);
        mul_add_c(a[1],b[2],c1,c2,c3);
        mul_add_c(a[2],b[1],c1,c2,c3);
        mul_add_c(a[3],b[0],c1,c2,c3);
        r[3]=c1;
        c1=0;
        mul_add_c(a[4],b[0],c2,c3,c1);
        mul_add_c(a[3],b[1],c2,c3,c1);
```



```
        mul_add_c(a[2],b[2],c2,c3,c1);
        mul_add_c(a[1],b[3],c2,c3,c1);
        mul_add_c(a[0],b[4],c2,c3,c1);
        r[4]=c2;
        c2=0;
        mul_add_c(a[0],b[5],c3,c1,c2);
        mul_add_c(a[1],b[4],c3,c1,c2);
        mul_add_c(a[2],b[3],c3,c1,c2);
        mul_add_c(a[3],b[2],c3,c1,c2);
        mul_add_c(a[4],b[1],c3,c1,c2);
        mul_add_c(a[5],b[0],c3,c1,c2);
        r[5]=c3;
        c3=0;
        mul_add_c(a[6],b[0],c1,c2,c3);
        mul_add_c(a[5],b[1],c1,c2,c3);
        mul_add_c(a[4],b[2],c1,c2,c3);
        mul_add_c(a[3],b[3],c1,c2,c3);
        mul_add_c(a[2],b[4],c1,c2,c3);
        mul_add_c(a[1],b[5],c1,c2,c3);
        mul_add_c(a[0],b[6],c1,c2,c3);
        r[6]=c1;
        c1=0;
        mul_add_c(a[0],b[7],c2,c3,c1);
        mul_add_c(a[1],b[6],c2,c3,c1);
        mul_add_c(a[2],b[5],c2,c3,c1);
        mul_add_c(a[3],b[4],c2,c3,c1);
        mul_add_c(a[4],b[3],c2,c3,c1);
        mul_add_c(a[5],b[2],c2,c3,c1);
        mul_add_c(a[6],b[1],c2,c3,c1);
        mul_add_c(a[7],b[0],c2,c3,c1);
        r[7]=c2;
        c2=0;
        mul_add_c(a[7],b[1],c3,c1,c2);
        mul_add_c(a[6],b[2],c3,c1,c2);
        mul_add_c(a[5],b[3],c3,c1,c2);
        mul_add_c(a[4],b[4],c3,c1,c2);
        mul_add_c(a[3],b[5],c3,c1,c2);
        mul_add_c(a[2],b[6],c3,c1,c2);
        mul_add_c(a[1],b[7],c3,c1,c2);
        r[8]=c3;
        c3=0;
        mul_add_c(a[2],b[7],c1,c2,c3);
        mul_add_c(a[3],b[6],c1,c2,c3);
        mul_add_c(a[4],b[5],c1,c2,c3);
        mul_add_c(a[5],b[4],c1,c2,c3);
        mul_add_c(a[6],b[3],c1,c2,c3);
        mul_add_c(a[7],b[2],c1,c2,c3);
        r[9]=c1;
        c1=0;
        mul_add_c(a[7],b[3],c2,c3,c1);
        mul_add_c(a[6],b[4],c2,c3,c1);
        mul_add_c(a[5],b[5],c2,c3,c1);
        mul_add_c(a[4],b[6],c2,c3,c1);
        mul_add_c(a[3],b[7],c2,c3,c1);
        r[10]=c2;
        c2=0;
        mul_add_c(a[4],b[7],c3,c1,c2);
        mul_add_c(a[5],b[6],c3,c1,c2);
        mul_add_c(a[6],b[5],c3,c1,c2);
        mul_add_c(a[7],b[4],c3,c1,c2);
        r[11]=c3;
        c3=0;
        mul_add_c(a[7],b[5],c1,c2,c3);
        mul_add_c(a[6],b[6],c1,c2,c3);
        mul_add_c(a[5],b[7],c1,c2,c3);
        r[12]=c1;
```

```
656         c1=0;
657         mul_add_c(a[6],b[7],c2,c3,c1);
658         mul_add_c(a[7],b[6],c2,c3,c1);
659         r[13]=c2;
660         c2=0;
661         mul_add_c(a[7],b[7],c3,c1,c2);
662         r[14]=c3;
663         r[15]=c1;
664         }

666 void bn_mul_comba4(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b)
667         {
668 #ifdef BN_LLONG
669         BN_ULLONG t;
670 #else
671         BN_ULONG bl,bh;
672 #endif
673         BN_ULONG t1,t2;
674         BN_ULONG c1,c2,c3;

676         c1=0;
677         c2=0;
678         c3=0;
679         mul_add_c(a[0],b[0],c1,c2,c3);
680         r[0]=c1;
681         c1=0;
682         mul_add_c(a[0],b[1],c2,c3,c1);
683         mul_add_c(a[1],b[0],c2,c3,c1);
684         r[1]=c2;
685         c2=0;
686         mul_add_c(a[2],b[0],c3,c1,c2);
687         mul_add_c(a[1],b[1],c3,c1,c2);
688         mul_add_c(a[0],b[2],c3,c1,c2);
689         r[2]=c3;
690         c3=0;
691         mul_add_c(a[0],b[3],c1,c2,c3);
692         mul_add_c(a[1],b[2],c1,c2,c3);
693         mul_add_c(a[2],b[1],c1,c2,c3);
694         mul_add_c(a[3],b[0],c1,c2,c3);
695         r[3]=c1;
696         c1=0;
697         mul_add_c(a[3],b[1],c2,c3,c1);
698         mul_add_c(a[2],b[2],c2,c3,c1);
699         mul_add_c(a[1],b[3],c2,c3,c1);
700         r[4]=c2;
701         c2=0;
702         mul_add_c(a[2],b[3],c3,c1,c2);
703         mul_add_c(a[3],b[2],c3,c1,c2);
704         r[5]=c3;
705         c3=0;
706         mul_add_c(a[3],b[3],c1,c2,c3);
707         r[6]=c1;
708         r[7]=c2;
709         }

711 void bn_sqr_comba8(BN_ULONG *r, const BN_ULONG *a)
712         {
713 #ifdef BN_LLONG
714         BN_ULLONG t,tt;
715 #else
716         BN_ULONG bl,bh;
717 #endif
718         BN_ULONG t1,t2;
719         BN_ULONG c1,c2,c3;

721         c1=0;
```



```
722         c2=0;
723         c3=0;
724         sqr_add_c(a,0,c1,c2,c3);
725         r[0]=c1;
726         c1=0;
727         sqr_add_c2(a,1,0,c2,c3,c1);
728         r[1]=c2;
729         c2=0;
730         sqr_add_c(a,1,c3,c1,c2);
731         sqr_add_c2(a,2,0,c3,c1,c2);
732         r[2]=c3;
733         c3=0;
734         sqr_add_c2(a,3,0,c1,c2,c3);
735         sqr_add_c2(a,2,1,c1,c2,c3);
736         r[3]=c1;
737         c1=0;
738         sqr_add_c(a,2,c2,c3,c1);
739         sqr_add_c2(a,3,1,c2,c3,c1);
740         sqr_add_c2(a,4,0,c2,c3,c1);
741         r[4]=c2;
742         c2=0;
743         sqr_add_c2(a,5,0,c3,c1,c2);
744         sqr_add_c2(a,4,1,c3,c1,c2);
745         sqr_add_c2(a,3,2,c3,c1,c2);
746         r[5]=c3;
747         c3=0;
748         sqr_add_c(a,3,c1,c2,c3);
749         sqr_add_c2(a,4,2,c1,c2,c3);
750         sqr_add_c2(a,5,1,c1,c2,c3);
751         sqr_add_c2(a,6,0,c1,c2,c3);
752         r[6]=c1;
753         c1=0;
754         sqr_add_c2(a,7,0,c2,c3,c1);
755         sqr_add_c2(a,6,1,c2,c3,c1);
756         sqr_add_c2(a,5,2,c2,c3,c1);
757         sqr_add_c2(a,4,3,c2,c3,c1);
758         r[7]=c2;
759         c2=0;
760         sqr_add_c(a,4,c3,c1,c2);
761         sqr_add_c2(a,5,3,c3,c1,c2);
762         sqr_add_c2(a,6,2,c3,c1,c2);
763         sqr_add_c2(a,7,1,c3,c1,c2);
764         r[8]=c3;
765         c3=0;
766         sqr_add_c2(a,7,2,c1,c2,c3);
767         sqr_add_c2(a,6,3,c1,c2,c3);
768         sqr_add_c2(a,5,4,c1,c2,c3);
769         r[9]=c1;
770         c1=0;
771         sqr_add_c(a,5,c2,c3,c1);
772         sqr_add_c2(a,6,4,c2,c3,c1);
773         sqr_add_c2(a,7,3,c2,c3,c1);
774         r[10]=c2;
775         c2=0;
776         sqr_add_c2(a,7,4,c3,c1,c2);
777         sqr_add_c2(a,6,5,c3,c1,c2);
778         r[11]=c3;
779         c3=0;
780         sqr_add_c(a,6,c1,c2,c3);
781         sqr_add_c2(a,7,5,c1,c2,c3);
782         r[12]=c1;
783         c1=0;
784         sqr_add_c2(a,7,6,c2,c3,c1);
785         r[13]=c2;
786         c2=0;
787         sqr_add_c(a,7,c3,c1,c2);
```

```
788         r[14]=c3;
789         r[15]=c1;
790         }

792 void bn_sqr_comba4(BN_ULONG *r, const BN_ULONG *a)
793         {
794 #ifdef BN_LLONG
795         BN_ULLONG t,tt;
796 #else
797         BN_ULONG bl,bh;
798 #endif
799         BN_ULONG t1,t2;
800         BN_ULONG c1,c2,c3;

802         c1=0;
803         c2=0;
804         c3=0;
805         sqr_add_c(a,0,c1,c2,c3);
806         r[0]=c1;
807         c1=0;
808         sqr_add_c2(a,1,0,c2,c3,c1);
809         r[1]=c2;
810         c2=0;
811         sqr_add_c(a,1,c3,c1,c2);
812         sqr_add_c2(a,2,0,c3,c1,c2);
813         r[2]=c3;
814         c3=0;
815         sqr_add_c2(a,3,0,c1,c2,c3);
816         sqr_add_c2(a,2,1,c1,c2,c3);
817         r[3]=c1;
818         c1=0;
819         sqr_add_c(a,2,c2,c3,c1);
820         sqr_add_c2(a,3,1,c2,c3,c1);
821         r[4]=c2;
822         c2=0;
823         sqr_add_c2(a,3,2,c3,c1,c2);
824         r[5]=c3;
825         c3=0;
826         sqr_add_c(a,3,c1,c2,c3);
827         r[6]=c1;
828         r[7]=c2;
829         }

831 #ifdef OPENSSL_NO_ASM
832 #ifdef OPENSSL_BN_ASM_MONT
833 #include <alloca.h>
834 /*
835  * This is essentially reference implementation, which may or may not
836  * result in performance improvement. E.g. on IA-32 this routine was
837  * observed to give 40% faster rsa1024 private key operations and 10%
838  * faster rsa4096 ones, while on AMD64 it improves rsa1024 sign only
839  * by 10% and *worsens* rsa4096 sign by 15%. Once again, it's a
840  * reference implementation, one to be used as starting point for
841  * platform-specific assembler. Mentioned numbers apply to compiler
842  * generated code compiled with and without -DOPENSSL_BN_ASM_MONT and
843  * can vary not only from platform to platform, but even for compiler
844  * versions. Assembler vs. assembler improvement coefficients can
845  * [and are known to] differ and are to be documented elsewhere.
846  */
847 int bn_mul_mont(BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp, const BN_U
848         {
849         BN_ULONG c0,c1,ml,*tp,n0;
850 #ifdef mul64
851         BN_ULONG mh;
852 #endif
853         volatile BN_ULONG *vp;
```



```
854         int i=0,j;

856 #if 0   /* template for platform-specific implementation */
857         if (ap==bp)     return bn_sqr_mont(rp,ap,np,n0p,num);
858 #endif
859         vp = tp = alloca((num+2)*sizeof(BN_ULONG));

861         n0 = *n0p;

863         c0 = 0;
864         ml = bp[0];
865 #ifdef mul64
866         mh = HBITS(ml);
867         ml = LBITS(ml);
868         for (j=0;j<num;++j)
869                 mul(tp[j],ap[j],ml,mh,c0);
870 #else
871         for (j=0;j<num;++j)
872                 mul(tp[j],ap[j],ml,c0);
873 #endif

875         tp[num]   = c0;
876         tp[num+1] = 0;
877         goto enter;

879         for(i=0;i<num;i++)
880                 {
881                 c0 = 0;
882                 ml = bp[i];
883 #ifdef mul64
884                 mh = HBITS(ml);
885                 ml = LBITS(ml);
886                 for (j=0;j<num;++j)
887                         mul_add(tp[j],ap[j],ml,mh,c0);
888 #else
889                 for (j=0;j<num;++j)
890                         mul_add(tp[j],ap[j],ml,c0);
891 #endif
892                 c1 = (tp[num] + c0)&BN_MASK2;
893                 tp[num]   = c1;
894                 tp[num+1] = (c1<c0?1:0);
895         enter:
896                 c1  = tp[0];
897                 ml = (c1*n0)&BN_MASK2;
898                 c0 = 0;
899 #ifdef mul64
900                 mh = HBITS(ml);
901                 ml = LBITS(ml);
902                 mul_add(c1,np[0],ml,mh,c0);
903 #else
904                 mul_add(c1,ml,np[0],c0);
905 #endif
906                 for(j=1;j<num;j++)
907                         {
908                         c1 = tp[j];
909 #ifdef mul64
910                         mul_add(c1,np[j],ml,mh,c0);
911 #else
912                         mul_add(c1,ml,np[j],c0);
913 #endif
914                         tp[j-1] = c1&BN_MASK2;
915                         }
916                 c1      = (tp[num] + c0)&BN_MASK2;
917                 tp[num-1] = c1;
918                 tp[num]   = tp[num+1] + (c1<c0?1:0);
919                 }
```

```
921         if (tp[num]!=0 || tp[num-1]>=np[num-1])
922                 {
923                 c0 = bn_sub_words(rp,tp,np,num);
924                 if (tp[num]!=0 || c0==0)
925                         {
926                         for(i=0;i<num+2;i++)    vp[i] = 0;
927                         return 1;
928                         }
929                 }
930         for(i=0;i<num;i++)      rp[i] = tp[i],  vp[i] = 0;
931         vp[num]   = 0;
932         vp[num+1] = 0;
933         return 1;
934         }
935 #else
936 /*
937  * Return value of 0 indicates that multiplication/convolution was not
938  * performed to signal the caller to fall down to alternative/original
939  * code-path.
940  */
941 int bn_mul_mont(BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp, const BN_U
942 {       return 0;       }
943 #endif /* OPENSSL_BN_ASM_MONT */
944 #endif

946 #else /* !BN_MUL_COMBA */

948 /* hmm... is it faster just to do a multiply? */
949 #undef bn_sqr_comba4
950 void bn_sqr_comba4(BN_ULONG *r, const BN_ULONG *a)
951         {
952         BN_ULONG t[8];
953         bn_sqr_normal(r,a,4,t);
954         }

956 #undef bn_sqr_comba8
957 void bn_sqr_comba8(BN_ULONG *r, const BN_ULONG *a)
958         {
959         BN_ULONG t[16];
960         bn_sqr_normal(r,a,8,t);
961         }

963 void bn_mul_comba4(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b)
964         {
965         r[4]=bn_mul_words(    &(r[0]),a,4,b[0]);
966         r[5]=bn_mul_add_words(&(r[1]),a,4,b[1]);
967         r[6]=bn_mul_add_words(&(r[2]),a,4,b[2]);
968         r[7]=bn_mul_add_words(&(r[3]),a,4,b[3]);
969         }

971 void bn_mul_comba8(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b)
972         {
973         r[ 8]=bn_mul_words(    &(r[0]),a,8,b[0]);
974         r[ 9]=bn_mul_add_words(&(r[1]),a,8,b[1]);
975         r[10]=bn_mul_add_words(&(r[2]),a,8,b[2]);
976         r[11]=bn_mul_add_words(&(r[3]),a,8,b[3]);
977         r[12]=bn_mul_add_words(&(r[4]),a,8,b[4]);
978         r[13]=bn_mul_add_words(&(r[5]),a,8,b[5]);
979         r[14]=bn_mul_add_words(&(r[6]),a,8,b[6]);
980         r[15]=bn_mul_add_words(&(r[7]),a,8,b[7]);
981         }

983 #ifdef OPENSSL_NO_ASM
984 #ifdef OPENSSL_BN_ASM_MONT
985 #include <alloca.h>
```



```
986 int bn_mul_mont(BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp, const BN_U
987         {
988         BN_ULONG c0,c1,*tp,n0=*n0p;
989         volatile BN_ULONG *vp;
990         int i=0,j;

992         vp = tp = alloca((num+2)*sizeof(BN_ULONG));

994         for(i=0;i<=num;i++)     tp[i]=0;

996         for(i=0;i<num;i++)
997                 {
998                 c0         = bn_mul_add_words(tp,ap,num,bp[i]);
999                 c1         = (tp[num] + c0)&BN_MASK2;
1000                tp[num]    = c1;
1001                tp[num+1]  = (c1<c0?1:0);

1003                c0         = bn_mul_add_words(tp,np,num,tp[0]*n0);
1004                c1         = (tp[num] + c0)&BN_MASK2;
1005                tp[num]    = c1;
1006                tp[num+1] += (c1<c0?1:0);
1007                for(j=0;j<=num;j++)     tp[j]=tp[j+1];
1008                }

1010        if (tp[num]!=0 || tp[num-1]>=np[num-1])
1011                {
1012                c0 = bn_sub_words(rp,tp,np,num);
1013                if (tp[num]!=0 || c0==0)
1014                        {
1015                        for(i=0;i<num+2;i++)    vp[i] = 0;
1016                        return 1;
1017                        }
1018                }
1019        for(i=0;i<num;i++)      rp[i] = tp[i],  vp[i] = 0;
1020        vp[num]   = 0;
1021        vp[num+1] = 0;
1022        return 1;
1023        }
1024 #else
1025 int bn_mul_mont(BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp, const BN_U
1026 {       return 0;       }
1027 #endif /* OPENSSL_BN_ASM_MONT */
1028 #endif

1030 #endif /* !BN_MUL_COMBA */
1031 #endif /* ! codereview */
```

```
**********************************************************
   11458 Wed Aug 13 19:52:14 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_blind.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_blind.c */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
```



```
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"

#define BN_BLINDING_COUNTER     32

struct bn_blinding_st
        {
        BIGNUM *A;
        BIGNUM *Ai;
        BIGNUM *e;
        BIGNUM *mod; /* just a reference */
#ifndef OPENSSL_NO_DEPRECATED
        unsigned long thread_id; /* added in OpenSSL 0.9.6j and 0.9.7b;
                                  * used only by crypto/rsa/rsa_eay.c, rsa_lib.c
#endif
```

```
128         CRYPTO_THREADID tid;
129         int counter;
130         unsigned long flags;
131         BN_MONT_CTX *m_ctx;
132         int (*bn_mod_exp)(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
133                           const BIGNUM *m, BN_CTX *ctx,
134                           BN_MONT_CTX *m_ctx);
135         };

137 BN_BLINDING *BN_BLINDING_new(const BIGNUM *A, const BIGNUM *Ai, BIGNUM *mod)
138         {
139         BN_BLINDING *ret=NULL;

141         bn_check_top(mod);

143         if ((ret=(BN_BLINDING *)OPENSSL_malloc(sizeof(BN_BLINDING))) == NULL)
144                 {
145                 BNerr(BN_F_BN_BLINDING_NEW,ERR_R_MALLOC_FAILURE);
146                 return(NULL);
147                 }
148         memset(ret,0,sizeof(BN_BLINDING));
149         if (A != NULL)
150                 {
151                 if ((ret->A  = BN_dup(A))  == NULL) goto err;
152                 }
153         if (Ai != NULL)
154                 {
155                 if ((ret->Ai = BN_dup(Ai)) == NULL) goto err;
156                 }

158         /* save a copy of mod in the BN_BLINDING structure */
159         if ((ret->mod = BN_dup(mod)) == NULL) goto err;
160         if (BN_get_flags(mod, BN_FLG_CONSTTIME) != 0)
161                 BN_set_flags(ret->mod, BN_FLG_CONSTTIME);

163         /* Set the counter to the special value -1
164          * to indicate that this is never-used fresh blinding
165          * that does not need updating before first use. */
166         ret->counter = -1;
167         CRYPTO_THREADID_current(&ret->tid);
168         return(ret);
169 err:
170         if (ret != NULL) BN_BLINDING_free(ret);
171         return(NULL);
172         }

174 void BN_BLINDING_free(BN_BLINDING *r)
175         {
176         if(r == NULL)
177             return;

179         if (r->A  != NULL) BN_free(r->A );
180         if (r->Ai != NULL) BN_free(r->Ai);
181         if (r->e  != NULL) BN_free(r->e );
182         if (r->mod != NULL) BN_free(r->mod);
183         OPENSSL_free(r);
184         }

186 int BN_BLINDING_update(BN_BLINDING *b, BN_CTX *ctx)
187         {
188         int ret=0;

190         if ((b->A == NULL) || (b->Ai == NULL))
191                 {
192                 BNerr(BN_F_BN_BLINDING_UPDATE,BN_R_NOT_INITIALIZED);
193                 goto err;
```



```
194                 }

196         if (b->counter == -1)
197                 b->counter = 0;

199         if (++b->counter == BN_BLINDING_COUNTER && b->e != NULL &&
200                 !(b->flags & BN_BLINDING_NO_RECREATE))
201                 {
202                 /* re-create blinding parameters */
203                 if (!BN_BLINDING_create_param(b, NULL, NULL, ctx, NULL, NULL))
204                         goto err;
205                 }
206         else if (!(b->flags & BN_BLINDING_NO_UPDATE))
207                 {
208                 if (!BN_mod_mul(b->A,b->A,b->A,b->mod,ctx)) goto err;
209                 if (!BN_mod_mul(b->Ai,b->Ai,b->Ai,b->mod,ctx)) goto err;
210                 }

212         ret=1;
213 err:
214         if (b->counter == BN_BLINDING_COUNTER)
215                 b->counter = 0;
216         return(ret);
217         }

219 int BN_BLINDING_convert(BIGNUM *n, BN_BLINDING *b, BN_CTX *ctx)
220         {
221         return BN_BLINDING_convert_ex(n, NULL, b, ctx);
222         }

224 int BN_BLINDING_convert_ex(BIGNUM *n, BIGNUM *r, BN_BLINDING *b, BN_CTX *ctx)
225         {
226         int ret = 1;

228         bn_check_top(n);

230         if ((b->A == NULL) || (b->Ai == NULL))
231                 {
232                 BNerr(BN_F_BN_BLINDING_CONVERT_EX,BN_R_NOT_INITIALIZED);
233                 return(0);
234                 }

236         if (b->counter == -1)
237                 /* Fresh blinding, doesn't need updating. */
238                 b->counter = 0;
239         else if (!BN_BLINDING_update(b,ctx))
240                 return(0);

242         if (r != NULL)
243                 {
244                 if (!BN_copy(r, b->Ai)) ret=0;
245                 }

247         if (!BN_mod_mul(n,n,b->A,b->mod,ctx)) ret=0;

249         return ret;
250         }

252 int BN_BLINDING_invert(BIGNUM *n, BN_BLINDING *b, BN_CTX *ctx)
253         {
254         return BN_BLINDING_invert_ex(n, NULL, b, ctx);
255         }

257 int BN_BLINDING_invert_ex(BIGNUM *n, const BIGNUM *r, BN_BLINDING *b, BN_CTX *ct
258         {
259         int ret;
```

```
261         bn_check_top(n);

263         if (r != NULL)
264                 ret = BN_mod_mul(n, n, r, b->mod, ctx);
265         else
266                 {
267                 if (b->Ai == NULL)
268                         {
269                         BNerr(BN_F_BN_BLINDING_INVERT_EX,BN_R_NOT_INITIALIZED);
270                         return(0);
271                         }
272                 ret = BN_mod_mul(n, n, b->Ai, b->mod, ctx);
273                 }

275         bn_check_top(n);
276         return(ret);
277         }

279 #ifndef OPENSSL_NO_DEPRECATED
280 unsigned long BN_BLINDING_get_thread_id(const BN_BLINDING *b)
281         {
282         return b->thread_id;
283         }

285 void BN_BLINDING_set_thread_id(BN_BLINDING *b, unsigned long n)
286         {
287         b->thread_id = n;
288         }
289 #endif

291 CRYPTO_THREADID *BN_BLINDING_thread_id(BN_BLINDING *b)
292         {
293         return &b->tid;
294         }

296 unsigned long BN_BLINDING_get_flags(const BN_BLINDING *b)
297         {
298         return b->flags;
299         }

301 void BN_BLINDING_set_flags(BN_BLINDING *b, unsigned long flags)
302         {
303         b->flags = flags;
304         }

306 BN_BLINDING *BN_BLINDING_create_param(BN_BLINDING *b,
307         const BIGNUM *e, BIGNUM *m, BN_CTX *ctx,
308         int (*bn_mod_exp)(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
309                           const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx),
310         BN_MONT_CTX *m_ctx)
311 {
312         int    retry_counter = 32;
313         BN_BLINDING *ret = NULL;

315         if (b == NULL)
316                 ret = BN_BLINDING_new(NULL, NULL, m);
317         else
318                 ret = b;

320         if (ret == NULL)
321                 goto err;

323         if (ret->A  == NULL && (ret->A  = BN_new()) == NULL)
324                 goto err;
325         if (ret->Ai == NULL && (ret->Ai = BN_new()) == NULL)
```



```
326                 goto err;

328         if (e != NULL)
329                 {
330                 if (ret->e != NULL)
331                         BN_free(ret->e);
332                 ret->e = BN_dup(e);
333                 }
334         if (ret->e == NULL)
335                 goto err;

337         if (bn_mod_exp != NULL)
338                 ret->bn_mod_exp = bn_mod_exp;
339         if (m_ctx != NULL)
340                 ret->m_ctx = m_ctx;

342         do {
343                 if (!BN_rand_range(ret->A, ret->mod)) goto err;
344                 if (BN_mod_inverse(ret->Ai, ret->A, ret->mod, ctx) == NULL)
345                         {
346                         /* this should almost never happen for good RSA keys */
347                         unsigned long error = ERR_peek_last_error();
348                         if (ERR_GET_REASON(error) == BN_R_NO_INVERSE)
349                                 {
350                                 if (retry_counter-- == 0)
351                                 {
352                                         BNerr(BN_F_BN_BLINDING_CREATE_PARAM,
353                                                 BN_R_TOO_MANY_ITERATIONS);
354                                         goto err;
355                                 }
356                                 ERR_clear_error();
357                                 }
358                         else
359                                 goto err;
360                         }
361                 else
362                         break;
363         } while (1);

365         if (ret->bn_mod_exp != NULL && ret->m_ctx != NULL)
366                 {
367                 if (!ret->bn_mod_exp(ret->A, ret->A, ret->e, ret->mod, ctx, ret-
368                         goto err;
369                 }
370         else
371                 {
372                 if (!BN_mod_exp(ret->A, ret->A, ret->e, ret->mod, ctx))
373                         goto err;
374                 }

376         return ret;
377 err:
378         if (b == NULL && ret != NULL)
379                 {
380                 BN_BLINDING_free(ret);
381                 ret = NULL;
382                 }

384         return ret;
385 }
386 #endif /* ! codereview */
```

```
**********************************************************
   20631 Wed Aug 13 19:52:14 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_const.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/knownprimes.c */
/* Insert boilerplate */

#include <openssl/bn.h>

/* "First Oakley Default Group" from RFC2409, section 6.1.
 *
 * The prime is: 2^768 - 2 ^704 - 1 + 2^64 * { [2^638 pi] + 149686 }
 *
 * RFC2409 specifies a generator of 2.
 * RFC2412 specifies a generator of of 22.
 */

BIGNUM *get_rfc2409_prime_768(BIGNUM *bn)
        {
        static const unsigned char RFC2409_PRIME_768[]={
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
                0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
                0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
                0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
                0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
                0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
                0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
                0xA6,0x3A,0x36,0x20,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                };
        return BN_bin2bn(RFC2409_PRIME_768,sizeof(RFC2409_PRIME_768),bn);
        }

/* "Second Oakley Default Group" from RFC2409, section 6.2.
 *
 * The prime is: 2^1024 - 2^960 - 1 + 2^64 * { [2^894 pi] + 129093 }.
 *
 * RFC2409 specifies a generator of 2.
 * RFC2412 specifies a generator of 22.
 */

BIGNUM *get_rfc2409_prime_1024(BIGNUM *bn)
        {
        static const unsigned char RFC2409_PRIME_1024[]={
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
                0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
                0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
                0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
                0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
                0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
                0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
                0xA6,0x37,0xED,0x6B,0x0B,0xFF,0x5C,0xB6,0xF4,0x06,0xB7,0xED,
                0xEE,0x38,0x6B,0xFB,0x5A,0x89,0x9F,0xA5,0xAE,0x9F,0x24,0x11,
                0x7C,0x4B,0x1F,0xE6,0x49,0x28,0x66,0x51,0xEC,0xE6,0x53,0x81,
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                };
        return BN_bin2bn(RFC2409_PRIME_1024,sizeof(RFC2409_PRIME_1024),bn);
        }

/* "1536-bit MODP Group" from RFC3526, Section 2.
 *
 * The prime is: 2^1536 - 2^1472 - 1 + 2^64 * { [2^1406 pi] + 741804 }
 *
 * RFC3526 specifies a generator of 2.
 * RFC2312 specifies a generator of 22.
 */
```



```
BIGNUM *get_rfc3526_prime_1536(BIGNUM *bn)
        {
        static const unsigned char RFC3526_PRIME_1536[]={
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
                0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
                0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
                0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
                0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
                0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
                0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
                0xA6,0x37,0xED,0x6B,0x0B,0xFF,0x5C,0xB6,0xF4,0x06,0xB7,0xED,
                0xEE,0x38,0x6B,0xFB,0x5A,0x89,0x9F,0xA5,0xAE,0x9F,0x24,0x11,
                0x7C,0x4B,0x1F,0xE6,0x49,0x28,0x66,0x51,0xEC,0xE4,0x5B,0x3D,
                0xC2,0x00,0x7C,0xB8,0xA1,0x63,0xBF,0x05,0x98,0xDA,0x48,0x36,
                0x1C,0x55,0xD3,0x9A,0x69,0x16,0x3F,0xA8,0xFD,0x24,0xCF,0x5F,
                0x83,0x65,0x5D,0x23,0xDC,0xA3,0xAD,0x96,0x1C,0x62,0xF3,0x56,
                0x20,0x85,0x52,0xBB,0x9E,0xD5,0x29,0x07,0x70,0x96,0x96,0x6D,
                0x67,0x0C,0x35,0x4E,0x4A,0xBC,0x98,0x04,0xF1,0x74,0x6C,0x08,
                0xCA,0x23,0x73,0x27,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                };
        return BN_bin2bn(RFC3526_PRIME_1536,sizeof(RFC3526_PRIME_1536),bn);
        }

/* "2048-bit MODP Group" from RFC3526, Section 3.
 *
 * The prime is: 2^2048 - 2^1984 - 1 + 2^64 * { [2^1918 pi] + 124476 }
 *
 * RFC3526 specifies a generator of 2.
 */

BIGNUM *get_rfc3526_prime_2048(BIGNUM *bn)
        {
        static const unsigned char RFC3526_PRIME_2048[]={
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
                0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
                0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
                0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
                0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
                0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
                0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
                0xA6,0x37,0xED,0x6B,0x0B,0xFF,0x5C,0xB6,0xF4,0x06,0xB7,0xED,
                0xEE,0x38,0x6B,0xFB,0x5A,0x89,0x9F,0xA5,0xAE,0x9F,0x24,0x11,
                0x7C,0x4B,0x1F,0xE6,0x49,0x28,0x66,0x51,0xEC,0xE4,0x5B,0x3D,
                0xC2,0x00,0x7C,0xB8,0xA1,0x63,0xBF,0x05,0x98,0xDA,0x48,0x36,
                0x1C,0x55,0xD3,0x9A,0x69,0x16,0x3F,0xA8,0xFD,0x24,0xCF,0x5F,
                0x83,0x65,0x5D,0x23,0xDC,0xA3,0xAD,0x96,0x1C,0x62,0xF3,0x56,
                0x20,0x85,0x52,0xBB,0x9E,0xD5,0x29,0x07,0x70,0x96,0x96,0x6D,
                0x67,0x0C,0x35,0x4E,0x4A,0xBC,0x98,0x04,0xF1,0x74,0x6C,0x08,
                0xCA,0x18,0x21,0x7C,0x32,0x90,0x5E,0x46,0x2E,0x36,0xCE,0x3B,
                0xE3,0x9E,0x77,0x2C,0x18,0x0E,0x86,0x03,0x9B,0x27,0x83,0xA2,
                0xEC,0x07,0xA2,0x8F,0xB5,0xC5,0x5D,0xF0,0x6F,0x4C,0x52,0xC9,
                0xDE,0x2B,0xCB,0xF6,0x95,0x58,0x17,0x18,0x39,0x95,0x49,0x7C,
                0xEA,0x95,0x6A,0xE5,0x15,0xD2,0x26,0x18,0x98,0xFA,0x05,0x10,
                0x15,0x72,0x8E,0x5A,0x8A,0xAC,0xAA,0x68,0xFF,0xFF,0xFF,0xFF,
                0xFF,0xFF,0xFF,0xFF,
                };
        return BN_bin2bn(RFC3526_PRIME_2048,sizeof(RFC3526_PRIME_2048),bn);
        }

/* "3072-bit MODP Group" from RFC3526, Section 4.
 *
 * The prime is: 2^3072 - 2^3008 - 1 + 2^64 * { [2^2942 pi] + 1690314 }
 *
 * RFC3526 specifies a generator of 2.
 */
```

```
BIGNUM *get_rfc3526_prime_3072(BIGNUM *bn)
        {
        static const unsigned char RFC3526_PRIME_3072[]={
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
                0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
                0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
                0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
                0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
                0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
                0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
                0xA6,0x37,0xED,0x6B,0x0B,0xFF,0x5C,0xB6,0xF4,0x06,0xB7,0xED,
                0xEE,0x38,0x6B,0xFB,0x5A,0x89,0x9F,0xA5,0xAE,0x9F,0x24,0x11,
                0x7C,0x4B,0x1F,0xE6,0x49,0x28,0x66,0x51,0xEC,0xE4,0x5B,0x3D,
                0xC2,0x00,0x7C,0xB8,0xA1,0x63,0xBF,0x05,0x98,0xDA,0x48,0x36,
                0x1C,0x55,0xD3,0x9A,0x69,0x16,0x3F,0xA8,0xFD,0x24,0xCF,0x5F,
                0x83,0x65,0x5D,0x23,0xDC,0xA3,0xAD,0x96,0x1C,0x62,0xF3,0x56,
                0x20,0x85,0x52,0xBB,0x9E,0xD5,0x29,0x07,0x70,0x96,0x96,0x6D,
                0x67,0x0C,0x35,0x4E,0x4A,0xBC,0x98,0x04,0xF1,0x74,0x6C,0x08,
                0xCA,0x18,0x21,0x7C,0x32,0x90,0x5E,0x46,0x2E,0x36,0xCE,0x3B,
                0xE3,0x9E,0x77,0x2C,0x18,0x0E,0x86,0x03,0x9B,0x27,0x83,0xA2,
                0xEC,0x07,0xA2,0x8F,0xB5,0xC5,0x5D,0xF0,0x6F,0x4C,0x52,0xC9,
                0xDE,0x2B,0xCB,0xF6,0x95,0x58,0x17,0x18,0x39,0x95,0x49,0x7C,
                0xEA,0x95,0x6A,0xE5,0x15,0xD2,0x26,0x18,0x98,0xFA,0x05,0x10,
                0x15,0x72,0x8E,0x5A,0x8A,0xAA,0xC4,0x2D,0xAD,0x33,0x17,0x0D,
                0x04,0x50,0x7A,0x33,0xA8,0x55,0x21,0xAB,0xDF,0x1C,0xBA,0x64,
                0xEC,0xFB,0x85,0x04,0x58,0xDB,0xEF,0x0A,0x8A,0xEA,0x71,0x57,
                0x5D,0x06,0x0C,0x7D,0xB3,0x97,0x0F,0x85,0xA6,0xE1,0xE4,0xC7,
                0xAB,0xF5,0xAE,0x8C,0xDB,0x09,0x33,0xD7,0x1E,0x8C,0x94,0xE0,
                0x4A,0x25,0x61,0x9D,0xCE,0xE3,0xD2,0x26,0x1A,0xD2,0xEE,0x6B,
                0xF1,0x2F,0xFA,0x06,0xD9,0x8A,0x08,0x64,0xD8,0x76,0x02,0x73,
                0x3E,0xC8,0x6A,0x64,0x52,0x1F,0x2B,0x18,0x17,0x7B,0x20,0x0C,
                0xBB,0xE1,0x17,0x57,0x7A,0x61,0x5D,0x6C,0x77,0x09,0x88,0xC0,
                0xBA,0xD9,0x46,0xE2,0x08,0xE2,0x4F,0xA0,0x74,0xE5,0xAB,0x31,
                0x43,0xDB,0x5B,0xFC,0xE0,0xFD,0x10,0x8E,0x4B,0x82,0xD1,0x20,
                0xA9,0x3A,0xD2,0xCA,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                };
        return BN_bin2bn(RFC3526_PRIME_3072,sizeof(RFC3526_PRIME_3072),bn);
        }

/* "4096-bit MODP Group" from RFC3526, Section 5.
 *
 * The prime is: 2^4096 - 2^4032 - 1 + 2^64 * { [2^3966 pi] + 240904 }
 *
 * RFC3526 specifies a generator of 2.
 */

BIGNUM *get_rfc3526_prime_4096(BIGNUM *bn)
        {
        static const unsigned char RFC3526_PRIME_4096[]={
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
                0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
                0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
                0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
                0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
                0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
                0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
                0xA6,0x37,0xED,0x6B,0x0B,0xFF,0x5C,0xB6,0xF4,0x06,0xB7,0xED,
                0xEE,0x38,0x6B,0xFB,0x5A,0x89,0x9F,0xA5,0xAE,0x9F,0x24,0x11,
                0x7C,0x4B,0x1F,0xE6,0x49,0x28,0x66,0x51,0xEC,0xE4,0x5B,0x3D,
                0xC2,0x00,0x7C,0xB8,0xA1,0x63,0xBF,0x05,0x98,0xDA,0x48,0x36,
                0x1C,0x55,0xD3,0x9A,0x69,0x16,0x3F,0xA8,0xFD,0x24,0xCF,0x5F,
                0x83,0x65,0x5D,0x23,0xDC,0xA3,0xAD,0x96,0x1C,0x62,0xF3,0x56,
                0x20,0x85,0x52,0xBB,0x9E,0xD5,0x29,0x07,0x70,0x96,0x96,0x6D,
                0x67,0x0C,0x35,0x4E,0x4A,0xBC,0x98,0x04,0xF1,0x74,0x6C,0x08,
                0xCA,0x18,0x21,0x7C,0x32,0x90,0x5E,0x46,0x2E,0x36,0xCE,0x3B,
```



```
                0xE3,0x9E,0x77,0x2C,0x18,0x0E,0x86,0x03,0x9B,0x27,0x83,0xA2,
                0xEC,0x07,0xA2,0x8F,0xB5,0xC5,0x5D,0xF0,0x6F,0x4C,0x52,0xC9,
                0xDE,0x2B,0xCB,0xF6,0x95,0x58,0x17,0x18,0x39,0x95,0x49,0x7C,
                0xEA,0x95,0x6A,0xE5,0x15,0xD2,0x26,0x18,0x98,0xFA,0x05,0x10,
                0x15,0x72,0x8E,0x5A,0x8A,0xAA,0xC4,0x2D,0xAD,0x33,0x17,0x0D,
                0x04,0x50,0x7A,0x33,0xA8,0x55,0x21,0xAB,0xDF,0x1C,0xBA,0x64,
                0xEC,0xFB,0x85,0x04,0x58,0xDB,0xEF,0x0A,0x8A,0xEA,0x71,0x57,
                0x5D,0x06,0x0C,0x7D,0xB3,0x97,0x0F,0x85,0xA6,0xE1,0xE4,0xC7,
                0xAB,0xF5,0xAE,0x8C,0xDB,0x09,0x33,0xD7,0x1E,0x8C,0x94,0xE0,
                0x4A,0x25,0x61,0x9D,0xCE,0xE3,0xD2,0x26,0x1A,0xD2,0xEE,0x6B,
                0xF1,0x2F,0xFA,0x06,0xD9,0x8A,0x08,0x64,0xD8,0x76,0x02,0x73,
                0x3E,0xC8,0x6A,0x64,0x52,0x1F,0x2B,0x18,0x17,0x7B,0x20,0x0C,
                0xBB,0xE1,0x17,0x57,0x7A,0x61,0x5D,0x6C,0x77,0x09,0x88,0xC0,
                0xBA,0xD9,0x46,0xE2,0x08,0xE2,0x4F,0xA0,0x74,0xE5,0xAB,0x31,
                0x43,0xDB,0x5B,0xFC,0xE0,0xFD,0x10,0x8E,0x4B,0x82,0xD1,0x20,
                0xA9,0x21,0x08,0x01,0x1A,0x72,0x3C,0x12,0xA7,0x87,0xE6,0xD7,
                0x88,0x71,0x9A,0x10,0xBD,0xBA,0x5B,0x26,0x99,0xC3,0x27,0x18,
                0x6A,0xF4,0xE2,0x3C,0x1A,0x94,0x68,0x34,0xB6,0x15,0x0B,0xDA,
                0x25,0x83,0xE9,0xCA,0x2A,0xD4,0x4C,0xE8,0xDB,0xBB,0xC2,0xDB,
                0x04,0xDE,0x8E,0xF9,0x2E,0x8E,0xFC,0x14,0x1F,0xBE,0xCA,0xA6,
                0x28,0x7C,0x59,0x47,0x4E,0x6B,0xC0,0x5D,0x99,0xB2,0x96,0x4F,
                0xA0,0x90,0xC3,0xA2,0x23,0x3B,0xA1,0x86,0x51,0x5B,0xE7,0xED,
                0x1F,0x61,0x29,0x70,0xCE,0xE2,0xD7,0xAF,0xB8,0x1B,0xDD,0x76,
                0x21,0x70,0x48,0x1C,0xD0,0x06,0x91,0x27,0xD5,0xB0,0x5A,0xA9,
                0x93,0xB4,0xEA,0x98,0x8D,0x8F,0xDD,0xC1,0x86,0xFF,0xB7,0xDC,
                0x90,0xA6,0xC0,0x8F,0x4D,0xF4,0x35,0xC9,0x34,0x06,0x31,0x99,
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
                };
        return BN_bin2bn(RFC3526_PRIME_4096,sizeof(RFC3526_PRIME_4096),bn);
        }

/* "6144-bit MODP Group" from RFC3526, Section 6.
 *
 * The prime is: 2^6144 - 2^6080 - 1 + 2^64 * { [2^6014 pi] + 929484 }
 *
 * RFC3526 specifies a generator of 2.
 */

BIGNUM *get_rfc3526_prime_6144(BIGNUM *bn)
        {
        static const unsigned char RFC3526_PRIME_6144[]={
                0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
                0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
                0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
                0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
                0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
                0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
                0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
                0xA6,0x37,0xED,0x6B,0x0B,0xFF,0x5C,0xB6,0xF4,0x06,0xB7,0xED,
                0xEE,0x38,0x6B,0xFB,0x5A,0x89,0x9F,0xA5,0xAE,0x9F,0x24,0x11,
                0x7C,0x4B,0x1F,0xE6,0x49,0x28,0x66,0x51,0xEC,0xE4,0x5B,0x3D,
                0xC2,0x00,0x7C,0xB8,0xA1,0x63,0xBF,0x05,0x98,0xDA,0x48,0x36,
                0x1C,0x55,0xD3,0x9A,0x69,0x16,0x3F,0xA8,0xFD,0x24,0xCF,0x5F,
                0x83,0x65,0x5D,0x23,0xDC,0xA3,0xAD,0x96,0x1C,0x62,0xF3,0x56,
                0x20,0x85,0x52,0xBB,0x9E,0xD5,0x29,0x07,0x70,0x96,0x96,0x6D,
                0x67,0x0C,0x35,0x4E,0x4A,0xBC,0x98,0x04,0xF1,0x74,0x6C,0x08,
                0xCA,0x18,0x21,0x7C,0x32,0x90,0x5E,0x46,0x2E,0x36,0xCE,0x3B,
                0xE3,0x9E,0x77,0x2C,0x18,0x0E,0x86,0x03,0x9B,0x27,0x83,0xA2,
                0xEC,0x07,0xA2,0x8F,0xB5,0xC5,0x5D,0xF0,0x6F,0x4C,0x52,0xC9,
                0xDE,0x2B,0xCB,0xF6,0x95,0x58,0x17,0x18,0x39,0x95,0x49,0x7C,
                0xEA,0x95,0x6A,0xE5,0x15,0xD2,0x26,0x18,0x98,0xFA,0x05,0x10,
                0x15,0x72,0x8E,0x5A,0x8A,0xAA,0xC4,0x2D,0xAD,0x33,0x17,0x0D,
                0x04,0x50,0x7A,0x33,0xA8,0x55,0x21,0xAB,0xDF,0x1C,0xBA,0x64,
                0xEC,0xFB,0x85,0x04,0x58,0xDB,0xEF,0x0A,0x8A,0xEA,0x71,0x57,
                0x5D,0x06,0x0C,0x7D,0xB3,0x97,0x0F,0x85,0xA6,0xE1,0xE4,0xC7,
                0xAB,0xF5,0xAE,0x8C,0xDB,0x09,0x33,0xD7,0x1E,0x8C,0x94,0xE0,
```

```
260                 0x4A,0x25,0x61,0x9D,0xCE,0xE3,0xD2,0x26,0x1A,0xD2,0xEE,0x6B,
261                 0xF1,0x2F,0xFA,0x06,0xD9,0x8A,0x08,0x64,0xD8,0x76,0x02,0x73,
262                 0x3E,0xC8,0x6A,0x64,0x52,0x1F,0x2B,0x18,0x17,0x7B,0x20,0x0C,
263                 0xBB,0xE1,0x17,0x57,0x7A,0x61,0x5D,0x6C,0x77,0x09,0x88,0xC0,
264                 0xBA,0xD9,0x46,0xE2,0x08,0xE2,0x4F,0xA0,0x74,0xE5,0xAB,0x31,
265                 0x43,0xDB,0x5B,0xFC,0xE0,0xFD,0x10,0x8E,0x4B,0x82,0xD1,0x20,
266                 0xA9,0x21,0x08,0x01,0x1A,0x72,0x3C,0x12,0xA7,0x87,0xE6,0xD7,
267                 0x88,0x71,0x9A,0x10,0xBD,0xBA,0x5B,0x26,0x99,0xC3,0x27,0x18,
268                 0x6A,0xF4,0xE2,0x3C,0x1A,0x94,0x68,0x34,0xB6,0x15,0x0B,0xDA,
269                 0x25,0x83,0xE9,0xCA,0x2A,0xD4,0x4C,0xE8,0xDB,0xBB,0xC2,0xDB,
270                 0x04,0xDE,0x8E,0xF9,0x2E,0x8E,0xFC,0x14,0x1F,0xBE,0xCA,0xA6,
271                 0x28,0x7C,0x59,0x47,0x4E,0x6B,0xC0,0x5D,0x99,0xB2,0x96,0x4F,
272                 0xA0,0x90,0xC3,0xA2,0x23,0x3B,0xA1,0x86,0x51,0x5B,0xE7,0xED,
273                 0x1F,0x61,0x29,0x70,0xCE,0xE2,0xD7,0xAF,0xB8,0x1B,0xDD,0x76,
274                 0x21,0x70,0x48,0x1C,0xD0,0x06,0x91,0x27,0xD5,0xB0,0x5A,0xA9,
275                 0x93,0xB4,0xEA,0x98,0x8D,0x8F,0xDD,0xC1,0x86,0xFF,0xB7,0xDC,
276                 0x90,0xA6,0xC0,0x8F,0x4D,0xF4,0x35,0xC9,0x34,0x02,0x84,0x92,
277                 0x36,0xC3,0xFA,0xB4,0xD2,0x7C,0x70,0x26,0xC1,0xD4,0xDC,0xB2,
278                 0x60,0x26,0x46,0xDE,0xC9,0x75,0x1E,0x76,0x3D,0xBA,0x37,0xBD,
279                 0xF8,0xFF,0x94,0x06,0xAD,0x9E,0x53,0x0E,0xE5,0xDB,0x38,0x2F,
280                 0x41,0x30,0x01,0xAE,0xB0,0x6A,0x53,0xED,0x90,0x27,0xD8,0x31,
281                 0x17,0x97,0x27,0xB0,0x86,0x5A,0x89,0x18,0xDA,0x3E,0xDB,0xEB,
282                 0xCF,0x9B,0x14,0xED,0x44,0xCE,0x6C,0xBA,0xCE,0xD4,0xBB,0x1B,
283                 0xDB,0x7F,0x14,0x47,0xE6,0xCC,0x25,0x4B,0x33,0x20,0x51,0x51,
284                 0x2B,0xD7,0xAF,0x42,0x6F,0xB8,0xF4,0x01,0x37,0x8C,0xD2,0xBF,
285                 0x59,0x83,0xCA,0x01,0xC6,0x4B,0x92,0xEC,0xF0,0x32,0xEA,0x15,
286                 0xD1,0x72,0x1D,0x03,0xF4,0x82,0xD7,0xCE,0x6E,0x74,0xFE,0xF6,
287                 0xD5,0x5E,0x70,0x2F,0x46,0x98,0x0C,0x82,0xB5,0xA8,0x40,0x31,
288                 0x90,0x0B,0x1C,0x9E,0x59,0xE7,0xC9,0x7F,0xBE,0xC7,0xE8,0xF3,
289                 0x23,0xA9,0x7A,0x7E,0x36,0xCC,0x88,0xBE,0x0F,0x1D,0x45,0xB7,
290                 0xFF,0x58,0x5A,0xC5,0x4B,0xD4,0x07,0xB2,0x2B,0x41,0x54,0xAA,
291                 0xCC,0x8F,0x6D,0x7E,0xBF,0x48,0xE1,0xD8,0x14,0xCC,0x5E,0xD2,
292                 0x0F,0x80,0x37,0xE0,0xA7,0x97,0x15,0xEE,0xF2,0x9B,0xE3,0x28,
293                 0x06,0xA1,0xD5,0x8B,0xB7,0xC5,0xDA,0x76,0xF5,0x50,0xAA,0x3D,
294                 0x8A,0x1F,0xBF,0xF0,0xEB,0x19,0xCC,0xB1,0xA3,0x13,0xD5,0x5C,
295                 0xDA,0x56,0xC9,0xEC,0x2E,0xF2,0x96,0x32,0x38,0x7F,0xE8,0xD7,
296                 0x6E,0x3C,0x04,0x68,0x04,0x3E,0x8F,0x66,0x3F,0x48,0x60,0xEE,
297                 0x12,0xBF,0x2D,0x5B,0x0B,0x74,0x74,0xD6,0xE6,0x94,0xF9,0x1E,
298                 0x6D,0xCC,0x40,0x24,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
299                 };
300         return BN_bin2bn(RFC3526_PRIME_6144,sizeof(RFC3526_PRIME_6144),bn);
301         }

303 /* "8192-bit MODP Group" from RFC3526, Section 7.
304  *
305  * The prime is: 2^8192 - 2^8128 - 1 + 2^64 * { [2^8062 pi] + 4743158 }
306  *
307  * RFC3526 specifies a generator of 2.
308  */

310 BIGNUM *get_rfc3526_prime_8192(BIGNUM *bn)
311         {
312         static const unsigned char RFC3526_PRIME_8192[]={
313                 0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xC9,0x0F,0xDA,0xA2,
314                 0x21,0x68,0xC2,0x34,0xC4,0xC6,0x62,0x8B,0x80,0xDC,0x1C,0xD1,
315                 0x29,0x02,0x4E,0x08,0x8A,0x67,0xCC,0x74,0x02,0x0B,0xBE,0xA6,
316                 0x3B,0x13,0x9B,0x22,0x51,0x4A,0x08,0x79,0x8E,0x34,0x04,0xDD,
317                 0xEF,0x95,0x19,0xB3,0xCD,0x3A,0x43,0x1B,0x30,0x2B,0x0A,0x6D,
318                 0xF2,0x5F,0x14,0x37,0x4F,0xE1,0x35,0x6D,0x6D,0x51,0xC2,0x45,
319                 0xE4,0x85,0xB5,0x76,0x62,0x5E,0x7E,0xC6,0xF4,0x4C,0x42,0xE9,
320                 0xA6,0x37,0xED,0x6B,0x0B,0xFF,0x5C,0xB6,0xF4,0x06,0xB7,0xED,
321                 0xEE,0x38,0x6B,0xFB,0x5A,0x89,0x9F,0xA5,0xAE,0x9F,0x24,0x11,
322                 0x7C,0x4B,0x1F,0xE6,0x49,0x28,0x66,0x51,0xEC,0xE4,0x5B,0x3D,
323                 0xC2,0x00,0x7C,0xB8,0xA1,0x63,0xBF,0x05,0x98,0xDA,0x48,0x36,
324                 0x1C,0x55,0xD3,0x9A,0x69,0x16,0x3F,0xA8,0xFD,0x24,0xCF,0x5F,
325                 0x83,0x65,0x5D,0x23,0xDC,0xA3,0xAD,0x96,0x1C,0x62,0xF3,0x56,
```



```
326                 0x20,0x85,0x52,0xBB,0x9E,0xD5,0x29,0x07,0x70,0x96,0x96,0x6D,
327                 0x67,0x0C,0x35,0x4E,0x4A,0xBC,0x98,0x04,0xF1,0x74,0x6C,0x08,
328                 0xCA,0x18,0x21,0x7C,0x32,0x90,0x5E,0x46,0x2E,0x36,0xCE,0x3B,
329                 0xE3,0x9E,0x77,0x2C,0x18,0x0E,0x86,0x03,0x9B,0x27,0x83,0xA2,
330                 0xEC,0x07,0xA2,0x8F,0xB5,0xC5,0x5D,0xF0,0x6F,0x4C,0x52,0xC9,
331                 0xDE,0x2B,0xCB,0xF6,0x95,0x58,0x17,0x18,0x39,0x95,0x49,0x7C,
332                 0xEA,0x95,0x6A,0xE5,0x15,0xD2,0x26,0x18,0x98,0xFA,0x05,0x10,
333                 0x15,0x72,0x8E,0x5A,0x8A,0xAA,0xC4,0x2D,0xAD,0x33,0x17,0x0D,
334                 0x04,0x50,0x7A,0x33,0xA8,0x55,0x21,0xAB,0xDF,0x1C,0xBA,0x64,
335                 0xEC,0xFB,0x85,0x04,0x58,0xDB,0xEF,0x0A,0x8A,0xEA,0x71,0x57,
336                 0x5D,0x06,0x0C,0x7D,0xB3,0x97,0x0F,0x85,0xA6,0xE1,0xE4,0xC7,
337                 0xAB,0xF5,0xAE,0x8C,0xDB,0x09,0x33,0xD7,0x1E,0x8C,0x94,0xE0,
338                 0x4A,0x25,0x61,0x9D,0xCE,0xE3,0xD2,0x26,0x1A,0xD2,0xEE,0x6B,
339                 0xF1,0x2F,0xFA,0x06,0xD9,0x8A,0x08,0x64,0xD8,0x76,0x02,0x73,
340                 0x3E,0xC8,0x6A,0x64,0x52,0x1F,0x2B,0x18,0x17,0x7B,0x20,0x0C,
341                 0xBB,0xE1,0x17,0x57,0x7A,0x61,0x5D,0x6C,0x77,0x09,0x88,0xC0,
342                 0xBA,0xD9,0x46,0xE2,0x08,0xE2,0x4F,0xA0,0x74,0xE5,0xAB,0x31,
343                 0x43,0xDB,0x5B,0xFC,0xE0,0xFD,0x10,0x8E,0x4B,0x82,0xD1,0x20,
344                 0xA9,0x21,0x08,0x01,0x1A,0x72,0x3C,0x12,0xA7,0x87,0xE6,0xD7,
345                 0x88,0x71,0x9A,0x10,0xBD,0xBA,0x5B,0x26,0x99,0xC3,0x27,0x18,
346                 0x6A,0xF4,0xE2,0x3C,0x1A,0x94,0x68,0x34,0xB6,0x15,0x0B,0xDA,
347                 0x25,0x83,0xE9,0xCA,0x2A,0xD4,0x4C,0xE8,0xDB,0xBB,0xC2,0xDB,
348                 0x04,0xDE,0x8E,0xF9,0x2E,0x8E,0xFC,0x14,0x1F,0xBE,0xCA,0xA6,
349                 0x28,0x7C,0x59,0x47,0x4E,0x6B,0xC0,0x5D,0x99,0xB2,0x96,0x4F,
350                 0xA0,0x90,0xC3,0xA2,0x23,0x3B,0xA1,0x86,0x51,0x5B,0xE7,0xED,
351                 0x1F,0x61,0x29,0x70,0xCE,0xE2,0xD7,0xAF,0xB8,0x1B,0xDD,0x76,
352                 0x21,0x70,0x48,0x1C,0xD0,0x06,0x91,0x27,0xD5,0xB0,0x5A,0xA9,
353                 0x93,0xB4,0xEA,0x98,0x8D,0x8F,0xDD,0xC1,0x86,0xFF,0xB7,0xDC,
354                 0x90,0xA6,0xC0,0x8F,0x4D,0xF4,0x35,0xC9,0x34,0x02,0x84,0x92,
355                 0x36,0xC3,0xFA,0xB4,0xD2,0x7C,0x70,0x26,0xC1,0xD4,0xDC,0xB2,
356                 0x60,0x26,0x46,0xDE,0xC9,0x75,0x1E,0x76,0x3D,0xBA,0x37,0xBD,
357                 0xF8,0xFF,0x94,0x06,0xAD,0x9E,0x53,0x0E,0xE5,0xDB,0x38,0x2F,
358                 0x41,0x30,0x01,0xAE,0xB0,0x6A,0x53,0xED,0x90,0x27,0xD8,0x31,
359                 0x17,0x97,0x27,0xB0,0x86,0x5A,0x89,0x18,0xDA,0x3E,0xDB,0xEB,
360                 0xCF,0x9B,0x14,0xED,0x44,0xCE,0x6C,0xBA,0xCE,0xD4,0xBB,0x1B,
361                 0xDB,0x7F,0x14,0x47,0xE6,0xCC,0x25,0x4B,0x33,0x20,0x51,0x51,
362                 0x2B,0xD7,0xAF,0x42,0x6F,0xB8,0xF4,0x01,0x37,0x8C,0xD2,0xBF,
363                 0x59,0x83,0xCA,0x01,0xC6,0x4B,0x92,0xEC,0xF0,0x32,0xEA,0x15,
364                 0xD1,0x72,0x1D,0x03,0xF4,0x82,0xD7,0xCE,0x6E,0x74,0xFE,0xF6,
365                 0xD5,0x5E,0x70,0x2F,0x46,0x98,0x0C,0x82,0xB5,0xA8,0x40,0x31,
366                 0x90,0x0B,0x1C,0x9E,0x59,0xE7,0xC9,0x7F,0xBE,0xC7,0xE8,0xF3,
367                 0x23,0xA9,0x7A,0x7E,0x36,0xCC,0x88,0xBE,0x0F,0x1D,0x45,0xB7,
368                 0xFF,0x58,0x5A,0xC5,0x4B,0xD4,0x07,0xB2,0x2B,0x41,0x54,0xAA,
369                 0xCC,0x8F,0x6D,0x7E,0xBF,0x48,0xE1,0xD8,0x14,0xCC,0x5E,0xD2,
370                 0x0F,0x80,0x37,0xE0,0xA7,0x97,0x15,0xEE,0xF2,0x9B,0xE3,0x28,
371                 0x06,0xA1,0xD5,0x8B,0xB7,0xC5,0xDA,0x76,0xF5,0x50,0xAA,0x3D,
372                 0x8A,0x1F,0xBF,0xF0,0xEB,0x19,0xCC,0xB1,0xA3,0x13,0xD5,0x5C,
373                 0xDA,0x56,0xC9,0xEC,0x2E,0xF2,0x96,0x32,0x38,0x7F,0xE8,0xD7,
374                 0x6E,0x3C,0x04,0x68,0x04,0x3E,0x8F,0x66,0x3F,0x48,0x60,0xEE,
375                 0x12,0xBF,0x2D,0x5B,0x0B,0x74,0x74,0xD6,0xE6,0x94,0xF9,0x1E,
376                 0x6D,0xBE,0x11,0x59,0x74,0xA3,0x92,0x6F,0x12,0xFE,0xE5,0xE4,
377                 0x38,0x77,0x7C,0xB6,0xA9,0x32,0xDF,0x8C,0xD8,0xBE,0xC4,0xD0,
378                 0x73,0xB9,0x31,0xBA,0x3B,0xC8,0x32,0xB6,0x8D,0x9D,0xD3,0x00,
379                 0x74,0x1F,0xA7,0xBF,0x8A,0xFC,0x47,0xED,0x25,0x76,0xF6,0x93,
380                 0x6B,0xA4,0x24,0x66,0x3A,0xAB,0x63,0x9C,0x5A,0xE4,0xF5,0x68,
381                 0x34,0x23,0xB4,0x74,0x2B,0xF1,0xC9,0x78,0x23,0x8F,0x16,0xCB,
382                 0xE3,0x9D,0x65,0x2D,0xE3,0xFD,0xB8,0xBE,0xFC,0x84,0x8A,0xD9,
383                 0x22,0x22,0x2E,0x04,0xA4,0x03,0x7C,0x07,0x13,0xEB,0x57,0xA8,
384                 0x1A,0x23,0xF0,0xC7,0x34,0x73,0xFC,0x64,0x6C,0xEA,0x30,0x6B,
385                 0x4B,0xCB,0xC8,0x86,0x2F,0x83,0x85,0xDD,0xFA,0x9D,0x4B,0x7F,
386                 0xA2,0xC0,0x87,0xE8,0x79,0x68,0x33,0x03,0xED,0x5B,0xDD,0x3A,
387                 0x06,0x2B,0x3C,0xF5,0xB3,0xA2,0x78,0xA6,0x6D,0x2A,0x13,0xF8,
388                 0x3F,0x44,0xF8,0x2D,0xDF,0x31,0x0E,0xE0,0x74,0xAB,0x6A,0x36,
389                 0x45,0x97,0xE8,0x99,0xA0,0x25,0x5D,0xC1,0x64,0xF3,0x1C,0xC5,
390                 0x08,0x46,0x85,0x1D,0xF9,0xAB,0x48,0x19,0x5D,0xED,0x7E,0xA1,
391                 0xB1,0xD5,0x10,0xBD,0x7E,0xE7,0x4D,0x73,0xFA,0xF3,0x6B,0xC3,
```

```
392                 0x1E,0xCF,0xA2,0x68,0x35,0x90,0x46,0xF4,0xEB,0x87,0x9F,0x92,
393                 0x40,0x09,0x43,0x8B,0x48,0x1C,0x6C,0xD7,0x88,0x9A,0x00,0x2E,
394                 0xD5,0xEE,0x38,0x2B,0xC9,0x19,0x0D,0xA6,0xFC,0x02,0x6E,0x47,
395                 0x95,0x58,0xE4,0x47,0x56,0x77,0xE9,0xAA,0x9E,0x30,0x50,0xE2,
396                 0x76,0x56,0x94,0xDF,0xC8,0x1F,0x56,0xE8,0x80,0xB9,0x6E,0x71,
397                 0x60,0xC9,0x80,0xDD,0x98,0xED,0xD3,0xDF,0xFF,0xFF,0xFF,0xFF,
398                 0xFF,0xFF,0xFF,0xFF,
399                 };
400         return BN_bin2bn(RFC3526_PRIME_8192,sizeof(RFC3526_PRIME_8192),bn);
401         }
402 #endif /* ! codereview */
```

```
**********************************************************
   11807 Wed Aug 13 19:52:14 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_ctx.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_ctx.c */
/* Written by Ulf Moeller for the OpenSSL project. */
/* ====================================================================
 * Copyright (c) 1998-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#if !defined(BN_CTX_DEBUG) && !defined(BN_DEBUG)
#ifndef NDEBUG
#define NDEBUG
#endif
#endif
```



```
#include <stdio.h>
#include <assert.h>

#include "cryptlib.h"
#include "bn_lcl.h"

/* TODO list
 *
 * 1. Check a bunch of "(words+1)" type hacks in various bignum functions and
 * check they can be safely removed.
 *  - Check +1 and other ugliness in BN_from_montgomery()
 *
 * 2. Consider allowing a BN_new_ex() that, at least, lets you specify an
 * appropriate 'block' size that will be honoured by bn_expand_internal() to
 * prevent piddly little reallocations. OTOH, profiling bignum expansions in
 * BN_CTX doesn't show this to be a big issue.
 */

/* How many bignums are in each "pool item"; */
#define BN_CTX_POOL_SIZE        16
/* The stack frame info is resizing, set a first-time expansion size; */
#define BN_CTX_START_FRAMES     32

/***********/
/* BN_POOL */
/***********/

/* A bundle of bignums that can be linked with other bundles */
typedef struct bignum_pool_item
        {
        /* The bignum values */
        BIGNUM vals[BN_CTX_POOL_SIZE];
        /* Linked-list admin */
        struct bignum_pool_item *prev, *next;
        } BN_POOL_ITEM;
/* A linked-list of bignums grouped in bundles */
typedef struct bignum_pool
        {
        /* Linked-list admin */
        BN_POOL_ITEM *head, *current, *tail;
        /* Stack depth and allocation size */
        unsigned used, size;
        } BN_POOL;
static void             BN_POOL_init(BN_POOL *);
static void             BN_POOL_finish(BN_POOL *);
#ifndef OPENSSL_NO_DEPRECATED
static void             BN_POOL_reset(BN_POOL *);
#endif
static BIGNUM *         BN_POOL_get(BN_POOL *);
static void             BN_POOL_release(BN_POOL *, unsigned int);

/************/
/* BN_STACK */
/************/

/* A wrapper to manage the "stack frames" */
typedef struct bignum_ctx_stack
        {
        /* Array of indexes into the bignum stack */
        unsigned int *indexes;
        /* Number of stack frames, and the size of the allocated array */
        unsigned int depth, size;
        } BN_STACK;
static void             BN_STACK_init(BN_STACK *);
static void             BN_STACK_finish(BN_STACK *);
```

```
#ifndef OPENSSL_NO_DEPRECATED
static void             BN_STACK_reset(BN_STACK *);
#endif
static int              BN_STACK_push(BN_STACK *, unsigned int);
static unsigned int     BN_STACK_pop(BN_STACK *);

/**********/
/* BN_CTX */
/**********/

/* The opaque BN_CTX type */
struct bignum_ctx
        {
        /* The bignum bundles */
        BN_POOL pool;
        /* The "stack frames", if you will */
        BN_STACK stack;
        /* The number of bignums currently assigned */
        unsigned int used;
        /* Depth of stack overflow */
        int err_stack;
        /* Block "gets" until an "end" (compatibility behaviour) */
        int too_many;
        };

/* Enable this to find BN_CTX bugs */
#ifdef BN_CTX_DEBUG
static const char *ctxdbg_cur = NULL;
static void ctxdbg(BN_CTX *ctx)
        {
        unsigned int bnidx = 0, fpidx = 0;
        BN_POOL_ITEM *item = ctx->pool.head;
        BN_STACK *stack = &ctx->stack;
        fprintf(stderr,"(%08x): ", (unsigned int)ctx);
        while(bnidx < ctx->used)
                {
                fprintf(stderr,"%03x ", item->vals[bnidx++ % BN_CTX_POOL_SIZE].d
                if(!(bnidx % BN_CTX_POOL_SIZE))
                        item = item->next;
                }
        fprintf(stderr,"\n");
        bnidx = 0;
        fprintf(stderr,"          : ");
        while(fpidx < stack->depth)
                {
                while(bnidx++ < stack->indexes[fpidx])
                        fprintf(stderr,"    ");
                fprintf(stderr,"^^^ ");
                bnidx++;
                fpidx++;
                }
        fprintf(stderr,"\n");
        }
#define CTXDBG_ENTRY(str, ctx)  do { \
                                ctxdbg_cur = (str); \
                                fprintf(stderr,"Starting %s\n", ctxdbg_cur); \
                                ctxdbg(ctx); \
                                } while(0)
#define CTXDBG_EXIT(ctx)        do { \
                                fprintf(stderr,"Ending %s\n", ctxdbg_cur); \
                                ctxdbg(ctx); \
                                } while(0)
#define CTXDBG_RET(ctx,ret)
#else
#define CTXDBG_ENTRY(str, ctx)
#define CTXDBG_EXIT(ctx)
```



```
#define CTXDBG_RET(ctx,ret)
#endif

/* This function is an evil legacy and should not be used. This implementation
 * is WYSIWYG, though I've done my best. */
#ifndef OPENSSL_NO_DEPRECATED
void BN_CTX_init(BN_CTX *ctx)
        {
        /* Assume the caller obtained the context via BN_CTX_new() and so is
         * trying to reset it for use. Nothing else makes sense, least of all
         * binary compatibility from a time when they could declare a static
         * variable. */
        BN_POOL_reset(&ctx->pool);
        BN_STACK_reset(&ctx->stack);
        ctx->used = 0;
        ctx->err_stack = 0;
        ctx->too_many = 0;
        }
#endif

BN_CTX *BN_CTX_new(void)
        {
        BN_CTX *ret = OPENSSL_malloc(sizeof(BN_CTX));
        if(!ret)
                {
                BNerr(BN_F_BN_CTX_NEW,ERR_R_MALLOC_FAILURE);
                return NULL;
                }
        /* Initialise the structure */
        BN_POOL_init(&ret->pool);
        BN_STACK_init(&ret->stack);
        ret->used = 0;
        ret->err_stack = 0;
        ret->too_many = 0;
        return ret;
        }

void BN_CTX_free(BN_CTX *ctx)
        {
        if (ctx == NULL)
                return;
#ifdef BN_CTX_DEBUG
        {
        BN_POOL_ITEM *pool = ctx->pool.head;
        fprintf(stderr,"BN_CTX_free, stack-size=%d, pool-bignums=%d\n",
                ctx->stack.size, ctx->pool.size);
        fprintf(stderr,"dmaxs: ");
        while(pool) {
                unsigned loop = 0;
                while(loop < BN_CTX_POOL_SIZE)
                        fprintf(stderr,"%02x ", pool->vals[loop++].dmax);
                pool = pool->next;
        }
        fprintf(stderr,"\n");
        }
#endif
        BN_STACK_finish(&ctx->stack);
        BN_POOL_finish(&ctx->pool);
        OPENSSL_free(ctx);
        }

void BN_CTX_start(BN_CTX *ctx)
        {
        CTXDBG_ENTRY("BN_CTX_start", ctx);
        /* If we're already overflowing ... */
        if(ctx->err_stack || ctx->too_many)
```

```
260                 ctx->err_stack++;
261         /* (Try to) get a new frame pointer */
262         else if(!BN_STACK_push(&ctx->stack, ctx->used))
263                 {
264                 BNerr(BN_F_BN_CTX_START,BN_R_TOO_MANY_TEMPORARY_VARIABLES);
265                 ctx->err_stack++;
266                 }
267         CTXDBG_EXIT(ctx);
268         }
269
270 void BN_CTX_end(BN_CTX *ctx)
271         {
272         CTXDBG_ENTRY("BN_CTX_end", ctx);
273         if(ctx->err_stack)
274                 ctx->err_stack--;
275         else
276                 {
277                 unsigned int fp = BN_STACK_pop(&ctx->stack);
278                 /* Does this stack frame have anything to release? */
279                 if(fp < ctx->used)
280                         BN_POOL_release(&ctx->pool, ctx->used - fp);
281                 ctx->used = fp;
282                 /* Unjam "too_many" in case "get" had failed */
283                 ctx->too_many = 0;
284                 }
285         CTXDBG_EXIT(ctx);
286         }
287
288 BIGNUM *BN_CTX_get(BN_CTX *ctx)
289         {
290         BIGNUM *ret;
291         CTXDBG_ENTRY("BN_CTX_get", ctx);
292         if(ctx->err_stack || ctx->too_many) return NULL;
293         if((ret = BN_POOL_get(&ctx->pool)) == NULL)
294                 {
295                 /* Setting too_many prevents repeated "get" attempts from
296                  * cluttering the error stack. */
297                 ctx->too_many = 1;
298                 BNerr(BN_F_BN_CTX_GET,BN_R_TOO_MANY_TEMPORARY_VARIABLES);
299                 return NULL;
300                 }
301         /* OK, make sure the returned bignum is "zero" */
302         BN_zero(ret);
303         ctx->used++;
304         CTXDBG_RET(ctx, ret);
305         return ret;
306         }
307
308 /************/
309 /* BN_STACK */
310 /************/
311
312 static void BN_STACK_init(BN_STACK *st)
313         {
314         st->indexes = NULL;
315         st->depth = st->size = 0;
316         }
317
318 static void BN_STACK_finish(BN_STACK *st)
319         {
320         if(st->size) OPENSSL_free(st->indexes);
321         }
322
323 #ifndef OPENSSL_NO_DEPRECATED
324 static void BN_STACK_reset(BN_STACK *st)
325         {
```



```
326         st->depth = 0;
327         }
328 #endif
329
330 static int BN_STACK_push(BN_STACK *st, unsigned int idx)
331         {
332         if(st->depth == st->size)
333                 /* Need to expand */
334                 {
335                 unsigned int newsize = (st->size ?
336                                         (st->size * 3 / 2) : BN_CTX_START_FRAMES);
337                 unsigned int *newitems = OPENSSL_malloc(newsize *
338                                                         sizeof(unsigned int));
339                 if(!newitems) return 0;
340                 if(st->depth)
341                         memcpy(newitems, st->indexes, st->depth *
342                                                         sizeof(unsigned int));
343                 if(st->size) OPENSSL_free(st->indexes);
344                 st->indexes = newitems;
345                 st->size = newsize;
346                 }
347         st->indexes[(st->depth)++] = idx;
348         return 1;
349         }
350
351 static unsigned int BN_STACK_pop(BN_STACK *st)
352         {
353         return st->indexes[--(st->depth)];
354         }
355
356 /***********/
357 /* BN_POOL */
358 /***********/
359
360 static void BN_POOL_init(BN_POOL *p)
361         {
362         p->head = p->current = p->tail = NULL;
363         p->used = p->size = 0;
364         }
365
366 static void BN_POOL_finish(BN_POOL *p)
367         {
368         while(p->head)
369                 {
370                 unsigned int loop = 0;
371                 BIGNUM *bn = p->head->vals;
372                 while(loop++ < BN_CTX_POOL_SIZE)
373                         {
374                         if(bn->d) BN_clear_free(bn);
375                         bn++;
376                         }
377                 p->current = p->head->next;
378                 OPENSSL_free(p->head);
379                 p->head = p->current;
380                 }
381         }
382
383 #ifndef OPENSSL_NO_DEPRECATED
384 static void BN_POOL_reset(BN_POOL *p)
385         {
386         BN_POOL_ITEM *item = p->head;
387         while(item)
388                 {
389                 unsigned int loop = 0;
390                 BIGNUM *bn = item->vals;
391                 while(loop++ < BN_CTX_POOL_SIZE)
```

```
392                         {
393                         if(bn->d) BN_clear(bn);
394                         bn++;
395                         }
396                 item = item->next;
397                 }
398         p->current = p->head;
399         p->used = 0;
400         }
401 #endif

403 static BIGNUM *BN_POOL_get(BN_POOL *p)
404         {
405         if(p->used == p->size)
406                 {
407                 BIGNUM *bn;
408                 unsigned int loop = 0;
409                 BN_POOL_ITEM *item = OPENSSL_malloc(sizeof(BN_POOL_ITEM));
410                 if(!item) return NULL;
411                 /* Initialise the structure */
412                 bn = item->vals;
413                 while(loop++ < BN_CTX_POOL_SIZE)
414                         BN_init(bn++);
415                 item->prev = p->tail;
416                 item->next = NULL;
417                 /* Link it in */
418                 if(!p->head)
419                         p->head = p->current = p->tail = item;
420                 else
421                         {
422                         p->tail->next = item;
423                         p->tail = item;
424                         p->current = item;
425                         }
426                 p->size += BN_CTX_POOL_SIZE;
427                 p->used++;
428                 /* Return the first bignum from the new pool */
429                 return item->vals;
430                 }
431         if(!p->used)
432                 p->current = p->head;
433         else if((p->used % BN_CTX_POOL_SIZE) == 0)
434                 p->current = p->current->next;
435         return p->current->vals + ((p->used++) % BN_CTX_POOL_SIZE);
436         }

438 static void BN_POOL_release(BN_POOL *p, unsigned int num)
439         {
440         unsigned int offset = (p->used - 1) % BN_CTX_POOL_SIZE;
441         p->used -= num;
442         while(num--)
443                 {
444                 bn_check_top(p->current->vals + offset);
445                 if(!offset)
446                         {
447                         offset = BN_CTX_POOL_SIZE - 1;
448                         p->current = p->current->prev;
449                         }
450                 else
451                         offset--;
452                 }
453         }
454 #endif /* ! codereview */
```

```
**********************************************************
    4030 Wed Aug 13 19:52:14 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_depr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_depr.c */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* Support for deprecated functions goes here - static linkage will only slurp
 * this code if applications are using them directly. */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
```



```
#include "bn_lcl.h"
#include <openssl/rand.h>

static void *dummy=&dummy;

#ifndef OPENSSL_NO_DEPRECATED
BIGNUM *BN_generate_prime(BIGNUM *ret, int bits, int safe,
        const BIGNUM *add, const BIGNUM *rem,
        void (*callback)(int,int,void *), void *cb_arg)
        {
        BN_GENCB cb;
        BIGNUM *rnd=NULL;
        int found = 0;

        BN_GENCB_set_old(&cb, callback, cb_arg);

        if (ret == NULL)
                {
                if ((rnd=BN_new()) == NULL) goto err;
                }
        else
                rnd=ret;
        if(!BN_generate_prime_ex(rnd, bits, safe, add, rem, &cb))
                goto err;

        /* we have a prime :-) */
        found = 1;
err:
        if (!found && (ret == NULL) && (rnd != NULL)) BN_free(rnd);
        return(found ? rnd : NULL);
        }

int BN_is_prime(const BIGNUM *a, int checks, void (*callback)(int,int,void *),
        BN_CTX *ctx_passed, void *cb_arg)
        {
        BN_GENCB cb;
        BN_GENCB_set_old(&cb, callback, cb_arg);
        return BN_is_prime_ex(a, checks, ctx_passed, &cb);
        }

int BN_is_prime_fasttest(const BIGNUM *a, int checks,
                void (*callback)(int,int,void *),
                BN_CTX *ctx_passed, void *cb_arg,
                int do_trial_division)
        {
        BN_GENCB cb;
        BN_GENCB_set_old(&cb, callback, cb_arg);
        return BN_is_prime_fasttest_ex(a, checks, ctx_passed,
                                do_trial_division, &cb);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
   12601 Wed Aug 13 19:52:14 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_div.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_div.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/bn.h>
#include <cryptlib.h>
```



```
#include <bn_lcl.h>


/* The old slow way */
#if 0
int BN_div(BIGNUM *dv, BIGNUM *rem, const BIGNUM *m, const BIGNUM *d,
           BN_CTX *ctx)
        {
        int i,nm,nd;
        int ret = 0;
        BIGNUM *D;

        bn_check_top(m);
        bn_check_top(d);
        if (BN_is_zero(d))
                {
                BNerr(BN_F_BN_DIV,BN_R_DIV_BY_ZERO);
                return(0);
                }

        if (BN_ucmp(m,d) < 0)
                {
                if (rem != NULL)
                        { if (BN_copy(rem,m) == NULL) return(0); }
                if (dv != NULL) BN_zero(dv);
                return(1);
                }

        BN_CTX_start(ctx);
        D = BN_CTX_get(ctx);
        if (dv == NULL) dv = BN_CTX_get(ctx);
        if (rem == NULL) rem = BN_CTX_get(ctx);
        if (D == NULL || dv == NULL || rem == NULL)
                goto end;

        nd=BN_num_bits(d);
        nm=BN_num_bits(m);
        if (BN_copy(D,d) == NULL) goto end;
        if (BN_copy(rem,m) == NULL) goto end;

        /* The next 2 are needed so we can do a dv->d[0]|=1 later
         * since BN_lshift1 will only work once there is a value :-) */
        BN_zero(dv);
        if(bn_wexpand(dv,1) == NULL) goto end;
        dv->top=1;

        if (!BN_lshift(D,D,nm-nd)) goto end;
        for (i=nm-nd; i>=0; i--)
                {
                if (!BN_lshift1(dv,dv)) goto end;
                if (BN_ucmp(rem,D) >= 0)
                        {
                        dv->d[0]|=1;
                        if (!BN_usub(rem,rem,D)) goto end;
                        }
/* CAN IMPROVE (and have now :=) */
                if (!BN_rshift1(D,D)) goto end;
                }
        rem->neg=BN_is_zero(rem)?0:m->neg;
        dv->neg=m->neg^d->neg;
        ret = 1;
 end:
        BN_CTX_end(ctx);
        return(ret);
        }
```

```
#else

#if !defined(OPENSSL_NO_ASM) && !defined(OPENSSL_NO_INLINE_ASM) \
    && !defined(PEDANTIC) && !defined(BN_DIV3W)
# if defined(__GNUC__) && __GNUC__>=2
#  if defined(__i386) || defined (__i386__)
   /*
    * There were two reasons for implementing this template:
    * - GNU C generates a call to a function (__udivdi3 to be exact)
    *   in reply to ((((BN_ULLONG)n0)<<BN_BITS2)|n1)/d0 (I fail to
    *   understand why...);
    * - divl doesn't only calculate quotient, but also leaves
    *   remainder in %edx which we can definitely use here:-)
    *
    *                                     <appro@fy.chalmers.se>
    */
#undef bn_div_words
#  define bn_div_words(n0,n1,d0)                \
        ({  __asm volatile (                    \
                "divl   %4"                     \
                : "=a"(q), "=d"(rem)            \
                : "a"(n1), "d"(n0), "g"(d0)     \
                : "cc");                        \
            q;                                  \
        })
#  define REMAINDER_IS_ALREADY_CALCULATED
#  elif defined(__x86_64) && defined(SIXTY_FOUR_BIT_LONG)
   /*
    * Same story here, but it's 128-bit by 64-bit division. Wow!
    *                                     <appro@fy.chalmers.se>
    */
#  undef bn_div_words
#  define bn_div_words(n0,n1,d0)                \
        ({  __asm volatile (                    \
                "divq   %4"                     \
                : "=a"(q), "=d"(rem)            \
                : "a"(n1), "d"(n0), "g"(d0)     \
                : "cc");                        \
            q;                                  \
        })
#  define REMAINDER_IS_ALREADY_CALCULATED
#  endif /* __<cpu> */
# endif /* __GNUC__ */
#endif /* OPENSSL_NO_ASM */


/* BN_div computes  dv := num / divisor,  rounding towards
 * zero, and sets up rm  such that  dv*divisor + rm = num  holds.
 * Thus:
 *     dv->neg == num->neg ^ divisor->neg  (unless the result is zero)
 *     rm->neg == num->neg                 (unless the remainder is zero)
 * If 'dv' or 'rm' is NULL, the respective value is not returned.
 */
int BN_div(BIGNUM *dv, BIGNUM *rm, const BIGNUM *num, const BIGNUM *divisor,
           BN_CTX *ctx)
        {
        int norm_shift,i,loop;
        BIGNUM *tmp,wnum,*snum,*sdiv,*res;
        BN_ULONG *resp,*wnump;
        BN_ULONG d0,d1;
        int num_n,div_n;
        int no_branch=0;

        /* Invalid zero-padding would have particularly bad consequences
         * in the case of 'num', so don't just rely on bn_check_top() for this o
         * (bn_check_top() works only for BN_DEBUG builds) */
```



```
        if (num->top > 0 && num->d[num->top - 1] == 0)
                {
                BNerr(BN_F_BN_DIV,BN_R_NOT_INITIALIZED);
                return 0;
                }

        bn_check_top(num);

        if ((BN_get_flags(num, BN_FLG_CONSTTIME) != 0) || (BN_get_flags(divisor,
                {
                no_branch=1;
                }

        bn_check_top(dv);
        bn_check_top(rm);
        /* bn_check_top(num); */ /* 'num' has been checked already */
        bn_check_top(divisor);

        if (BN_is_zero(divisor))
                {
                BNerr(BN_F_BN_DIV,BN_R_DIV_BY_ZERO);
                return(0);
                }

        if (!no_branch && BN_ucmp(num,divisor) < 0)
                {
                if (rm != NULL)
                        { if (BN_copy(rm,num) == NULL) return(0); }
                if (dv != NULL) BN_zero(dv);
                return(1);
                }

        BN_CTX_start(ctx);
        tmp=BN_CTX_get(ctx);
        snum=BN_CTX_get(ctx);
        sdiv=BN_CTX_get(ctx);
        if (dv == NULL)
                res=BN_CTX_get(ctx);
        else    res=dv;
        if (sdiv == NULL || res == NULL || tmp == NULL || snum == NULL)
                goto err;

        /* First we normalise the numbers */
        norm_shift=BN_BITS2-((BN_num_bits(divisor))%BN_BITS2);
        if (!(BN_lshift(sdiv,divisor,norm_shift))) goto err;
        sdiv->neg=0;
        norm_shift+=BN_BITS2;
        if (!(BN_lshift(snum,num,norm_shift))) goto err;
        snum->neg=0;

        if (no_branch)
                {
                /* Since we don't know whether snum is larger than sdiv,
                 * we pad snum with enough zeroes without changing its
                 * value.
                 */
                if (snum->top <= sdiv->top+1)
                        {
                        if (bn_wexpand(snum, sdiv->top + 2) == NULL) goto err;
                        for (i = snum->top; i < sdiv->top + 2; i++) snum->d[i] =
                        snum->top = sdiv->top + 2;
                        }
                else
                        {
                        if (bn_wexpand(snum, snum->top + 1) == NULL) goto err;
                        snum->d[snum->top] = 0;
```

```
260                         snum->top ++;
261                         }
262                 }

264         div_n=sdiv->top;
265         num_n=snum->top;
266         loop=num_n-div_n;
267         /* Lets setup a 'window' into snum
268          * This is the part that corresponds to the current
269          * 'area' being divided */
270         wnum.neg   = 0;
271         wnum.d     = &(snum->d[loop]);
272         wnum.top   = div_n;
273         /* only needed when BN_ucmp messes up the values between top and max */
274         wnum.dmax  = snum->dmax - loop; /* so we don't step out of bounds */

276         /* Get the top 2 words of sdiv */
277         /* div_n=sdiv->top; */
278         d0=sdiv->d[div_n-1];
279         d1=(div_n == 1)?0:sdiv->d[div_n-2];

281         /* pointer to the 'top' of snum */
282         wnump= &(snum->d[num_n-1]);

284         /* Setup to 'res' */
285         res->neg= (num->neg^divisor->neg);
286         if (!bn_wexpand(res,(loop+1))) goto err;
287         res->top=loop-no_branch;
288         resp= &(res->d[loop-1]);

290         /* space for temp */
291         if (!bn_wexpand(tmp,(div_n+1))) goto err;

293         if (!no_branch)
294                 {
295                 if (BN_ucmp(&wnum,sdiv) >= 0)
296                         {
297                         /* If BN_DEBUG_RAND is defined BN_ucmp changes (via
298                          * bn_pollute) the const bignum arguments =>
299                          * clean the values between top and max again */
300                         bn_clear_top2max(&wnum);
301                         bn_sub_words(wnum.d, wnum.d, sdiv->d, div_n);
302                         *resp=1;
303                         }
304                 else
305                         res->top--;
306                 }

308         /* if res->top == 0 then clear the neg value otherwise decrease
309          * the resp pointer */
310         if (res->top == 0)
311                 res->neg = 0;
312         else
313                 resp--;

315         for (i=0; i<loop-1; i++, wnump--, resp--)
316                 {
317                 BN_ULONG q,l0;
318                 /* the first part of the loop uses the top two words of
319                  * snum and sdiv to calculate a BN_ULONG q such that
320                  * | wnum - sdiv * q | < sdiv */
321 #if defined(BN_DIV3W) && !defined(OPENSSL_NO_ASM)
322                 BN_ULONG bn_div_3_words(BN_ULONG*,BN_ULONG,BN_ULONG);
323                 q=bn_div_3_words(wnump,d1,d0);
324 #else
325                 BN_ULONG n0,n1,rem=0;
```



```
327                 n0=wnump[0];
328                 n1=wnump[-1];
329                 if (n0 == d0)
330                         q=BN_MASK2;
331                 else                    /* n0 < d0 */
332                         {
333 #ifdef BN_LLONG
334                         BN_ULLONG t2;

336 #if defined(BN_LLONG) && defined(BN_DIV2W) && !defined(bn_div_words)
337                         q=(BN_ULONG)(((((BN_ULLONG)n0)<<BN_BITS2)|n1)/d0);
338 #else
339                         q=bn_div_words(n0,n1,d0);
340 #ifdef BN_DEBUG_LEVITTE
341                         fprintf(stderr,"DEBUG: bn_div_words(0x%08X,0x%08X,0x%08\
342 X) -> 0x%08X\n",
343                                 n0, n1, d0, q);
344 #endif
345 #endif

347 #ifndef REMAINDER_IS_ALREADY_CALCULATED
348                         /*
349                          * rem doesn't have to be BN_ULLONG. The least we
350                          * know it's less that d0, isn't it?
351                          */
352                         rem=(n1-q*d0)&BN_MASK2;
353 #endif
354                         t2=(BN_ULLONG)d1*q;

356                         for (;;)
357                                 {
358                                 if (t2 <= ((((BN_ULLONG)rem)<<BN_BITS2)|wnump[-2
359                                         break;
360                                 q--;
361                                 rem += d0;
362                                 if (rem < d0) break; /* don't let rem overflow *
363                                 t2 -= d1;
364                                 }
365 #else /* !BN_LLONG */
366                         BN_ULONG t2l,t2h;

368                         q=bn_div_words(n0,n1,d0);
369 #ifdef BN_DEBUG_LEVITTE
370                         fprintf(stderr,"DEBUG: bn_div_words(0x%08X,0x%08X,0x%08\
371 X) -> 0x%08X\n",
372                                 n0, n1, d0, q);
373 #endif
374 #ifndef REMAINDER_IS_ALREADY_CALCULATED
375                         rem=(n1-q*d0)&BN_MASK2;
376 #endif

378 #if defined(BN_UMULT_LOHI)
379                         BN_UMULT_LOHI(t2l,t2h,d1,q);
380 #elif defined(BN_UMULT_HIGH)
381                         t2l = d1 * q;
382                         t2h = BN_UMULT_HIGH(d1,q);
383 #else
384                         {
385                         BN_ULONG ql, qh;
386                         t2l=LBITS(d1); t2h=HBITS(d1);
387                         ql =LBITS(q);  qh =HBITS(q);
388                         mul64(t2l,t2h,ql,qh); /* t2=(BN_ULLONG)d1*q; */
389                         }
390 #endif
```

```
392                         for (;;)
393                                 {
394                                 if ((t2h < rem) ||
395                                         ((t2h == rem) && (t2l <= wnump[-2])))
396                                         break;
397                                 q--;
398                                 rem += d0;
399                                 if (rem < d0) break; /* don't let rem overflow *
400                                 if (t2l < d1) t2h--; t2l -= d1;
401                                 }
402 #endif /* !BN_LLONG */
403                         }
404 #endif /* !BN_DIV3W */

406                 l0=bn_mul_words(tmp->d,sdiv->d,div_n,q);
407                 tmp->d[div_n]=l0;
408                 wnum.d--;
409                 /* ingore top values of the bignums just sub the two
410                  * BN_ULONG arrays with bn_sub_words */
411                 if (bn_sub_words(wnum.d, wnum.d, tmp->d, div_n+1))
412                         {
413                         /* Note: As we have considered only the leading
414                          * two BN_ULONGs in the calculation of q, sdiv * q
415                          * might be greater than wnum (but then (q-1) * sdiv
416                          * is less or equal than wnum)
417                          */
418                         q--;
419                         if (bn_add_words(wnum.d, wnum.d, sdiv->d, div_n))
420                                 /* we can't have an overflow here (assuming
421                                  * that q != 0, but if q == 0 then tmp is
422                                  * zero anyway) */
423                                 (*wnump)++;
424                         }
425                 /* store part of the result */
426                 *resp = q;
427                 }
428         bn_correct_top(snum);
429         if (rm != NULL)
430                 {
431                 /* Keep a copy of the neg flag in num because if rm==num
432                  * BN_rshift() will overwrite it.
433                  */
434                 int neg = num->neg;
435                 BN_rshift(rm,snum,norm_shift);
436                 if (!BN_is_zero(rm))
437                         rm->neg = neg;
438                 bn_check_top(rm);
439                 }
440         if (no_branch)  bn_correct_top(res);
441         BN_CTX_end(ctx);
442         return(1);
443 err:
444         bn_check_top(rm);
445         BN_CTX_end(ctx);
446         return(0);
447         }
448 #endif
449 #endif /* ! codereview */
```

```
**********************************************************
    6579 Wed Aug 13 19:52:14 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_err.c */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/bn.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_BN,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_BN,0,reason)

static ERR_STRING_DATA BN_str_functs[]=
        {
{ERR_FUNC(BN_F_BNRAND), "BNRAND"},
{ERR_FUNC(BN_F_BN_BLINDING_CONVERT_EX), "BN_BLINDING_convert_ex"},
{ERR_FUNC(BN_F_BN_BLINDING_CREATE_PARAM),       "BN_BLINDING_create_param"},
{ERR_FUNC(BN_F_BN_BLINDING_INVERT_EX),  "BN_BLINDING_invert_ex"},
{ERR_FUNC(BN_F_BN_BLINDING_NEW),        "BN_BLINDING_new"},
{ERR_FUNC(BN_F_BN_BLINDING_UPDATE),     "BN_BLINDING_update"},
{ERR_FUNC(BN_F_BN_BN2DEC),      "BN_bn2dec"},
{ERR_FUNC(BN_F_BN_BN2HEX),      "BN_bn2hex"},
{ERR_FUNC(BN_F_BN_CTX_GET),     "BN_CTX_get"},
{ERR_FUNC(BN_F_BN_CTX_NEW),     "BN_CTX_new"},
{ERR_FUNC(BN_F_BN_CTX_START),   "BN_CTX_start"},
{ERR_FUNC(BN_F_BN_DIV), "BN_div"},
{ERR_FUNC(BN_F_BN_DIV_NO_BRANCH),       "BN_div_no_branch"},
{ERR_FUNC(BN_F_BN_DIV_RECP),    "BN_div_recp"},
{ERR_FUNC(BN_F_BN_EXP), "BN_exp"},
{ERR_FUNC(BN_F_BN_EXPAND2),     "bn_expand2"},
{ERR_FUNC(BN_F_BN_EXPAND_INTERNAL),     "BN_EXPAND_INTERNAL"},
{ERR_FUNC(BN_F_BN_GF2M_MOD),    "BN_GF2m_mod"},
{ERR_FUNC(BN_F_BN_GF2M_MOD_EXP),        "BN_GF2m_mod_exp"},
{ERR_FUNC(BN_F_BN_GF2M_MOD_MUL),        "BN_GF2m_mod_mul"},
{ERR_FUNC(BN_F_BN_GF2M_MOD_SOLVE_QUAD), "BN_GF2m_mod_solve_quad"},
{ERR_FUNC(BN_F_BN_GF2M_MOD_SOLVE_QUAD_ARR),     "BN_GF2m_mod_solve_quad_arr"},
{ERR_FUNC(BN_F_BN_GF2M_MOD_SQR),        "BN_GF2m_mod_sqr"},
{ERR_FUNC(BN_F_BN_GF2M_MOD_SQRT),       "BN_GF2m_mod_sqrt"},
{ERR_FUNC(BN_F_BN_MOD_EXP2_MONT),       "BN_mod_exp2_mont"},
{ERR_FUNC(BN_F_BN_MOD_EXP_MONT),        "BN_mod_exp_mont"},
{ERR_FUNC(BN_F_BN_MOD_EXP_MONT_CONSTTIME),      "BN_mod_exp_mont_consttime"},
{ERR_FUNC(BN_F_BN_MOD_EXP_MONT_WORD),   "BN_mod_exp_mont_word"},
{ERR_FUNC(BN_F_BN_MOD_EXP_RECP),        "BN_mod_exp_recp"},
{ERR_FUNC(BN_F_BN_MOD_EXP_SIMPLE),      "BN_mod_exp_simple"},
{ERR_FUNC(BN_F_BN_MOD_INVERSE), "BN_mod_inverse"},
{ERR_FUNC(BN_F_BN_MOD_INVERSE_NO_BRANCH),       "BN_mod_inverse_no_branch"},
{ERR_FUNC(BN_F_BN_MOD_LSHIFT_QUICK),    "BN_mod_lshift_quick"},
{ERR_FUNC(BN_F_BN_MOD_MUL_RECIPROCAL),  "BN_mod_mul_reciprocal"},
{ERR_FUNC(BN_F_BN_MOD_SQRT),    "BN_mod_sqrt"},
{ERR_FUNC(BN_F_BN_MPI2BN),      "BN_mpi2bn"},
{ERR_FUNC(BN_F_BN_NEW), "BN_new"},
{ERR_FUNC(BN_F_BN_RAND),        "BN_rand"},
{ERR_FUNC(BN_F_BN_RAND_RANGE),  "BN_rand_range"},
{ERR_FUNC(BN_F_BN_USUB),        "BN_usub"},
{0,NULL}
        };

static ERR_STRING_DATA BN_str_reasons[]=
        {
{ERR_REASON(BN_R_ARG2_LT_ARG3)          ,"arg2 lt arg3"},
{ERR_REASON(BN_R_BAD_RECIPROCAL)        ,"bad reciprocal"},
{ERR_REASON(BN_R_BIGNUM_TOO_LONG)       ,"bignum too long"},
{ERR_REASON(BN_R_CALLED_WITH_EVEN_MODULUS),"called with even modulus"},
{ERR_REASON(BN_R_DIV_BY_ZERO)           ,"div by zero"},
{ERR_REASON(BN_R_ENCODING_ERROR)        ,"encoding error"},
{ERR_REASON(BN_R_EXPAND_ON_STATIC_BIGNUM_DATA),"expand on static bignum data"},
{ERR_REASON(BN_R_INPUT_NOT_REDUCED)     ,"input not reduced"},
{ERR_REASON(BN_R_INVALID_LENGTH)        ,"invalid length"},
{ERR_REASON(BN_R_INVALID_RANGE)         ,"invalid range"},
```

```
128 {ERR_REASON(BN_R_NOT_A_SQUARE)           ,"not a square"},
129 {ERR_REASON(BN_R_NOT_INITIALIZED)        ,"not initialized"},
130 {ERR_REASON(BN_R_NO_INVERSE)             ,"no inverse"},
131 {ERR_REASON(BN_R_NO_SOLUTION)            ,"no solution"},
132 {ERR_REASON(BN_R_P_IS_NOT_PRIME)         ,"p is not prime"},
133 {ERR_REASON(BN_R_TOO_MANY_ITERATIONS)    ,"too many iterations"},
134 {ERR_REASON(BN_R_TOO_MANY_TEMPORARY_VARIABLES),"too many temporary variables"},
135 {0,NULL}
136         };

138 #endif

140 void ERR_load_BN_strings(void)
141         {
142 #ifndef OPENSSL_NO_ERR

144         if (ERR_func_error_string(BN_str_functs[0].error) == NULL)
145                 {
146                 ERR_load_strings(0,BN_str_functs);
147                 ERR_load_strings(0,BN_str_reasons);
148                 }
149 #endif
150         }
151 #endif /* ! codereview */
```

```
**********************************************************
   29539 Wed Aug 13 19:52:15 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_exp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_exp.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include "cryptlib.h"
#include "bn_lcl.h"

#include <stdlib.h>
#ifdef _WIN32
# include <malloc.h>
# ifndef alloca
#  define alloca _alloca
# endif
#elif defined(__GNUC__)
# ifndef alloca
#  define alloca(s) __builtin_alloca((s))
# endif
#endif
```

```
/* maximum precomputation table size for *variable* sliding windows */
#define TABLE_SIZE      32

/* this one works - simple but works */
int BN_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx)
        {
        int i,bits,ret=0;
        BIGNUM *v,*rr;

        if (BN_get_flags(p, BN_FLG_CONSTTIME) != 0)
                {
                /* BN_FLG_CONSTTIME only supported by BN_mod_exp_mont() */
                BNerr(BN_F_BN_EXP,ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
                return -1;
                }

        BN_CTX_start(ctx);
        if ((r == a) || (r == p))
                rr = BN_CTX_get(ctx);
        else
                rr = r;
        v = BN_CTX_get(ctx);
        if (rr == NULL || v == NULL) goto err;

        if (BN_copy(v,a) == NULL) goto err;
        bits=BN_num_bits(p);

        if (BN_is_odd(p))
                { if (BN_copy(rr,a) == NULL) goto err; }
        else    { if (!BN_one(rr)) goto err; }

        for (i=1; i<bits; i++)
                {
                if (!BN_sqr(v,v,ctx)) goto err;
                if (BN_is_bit_set(p,i))
                        {
                        if (!BN_mul(rr,rr,v,ctx)) goto err;
                        }
                }
        ret=1;
err:
        if (r != rr) BN_copy(r,rr);
        BN_CTX_end(ctx);
        bn_check_top(r);
        return(ret);
        }


int BN_mod_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, const BIGNUM *m,
               BN_CTX *ctx)
        {
        int ret;

        bn_check_top(a);
        bn_check_top(p);
        bn_check_top(m);

        /* For even modulus  m = 2^k*m_odd,  it might make sense to compute
         * a^p mod m_odd  and  a^p mod 2^k  separately (with Montgomery
         * exponentiation for the odd part), using appropriate exponent
         * reductions, and combine the results using the CRT.
         *
         * For now, we use Montgomery only if the modulus is odd; otherwise,
         * exponentiation using the reciprocal-based quick remaindering
         * algorithm is used.
         *
```



```
         * (Timing obtained with expspeed.c [computations  a^p mod m
         * where  a, p, m  are of the same length: 256, 512, 1024, 2048,
         * 4096, 8192 bits], compared to the running time of the
         * standard algorithm:
         *
         *   BN_mod_exp_mont   33 .. 40 %  [AMD K6-2, Linux, debug configuration
         *                     55 .. 77 %  [UltraSparc processor, but
         *                                  debug-solaris-sparcv8-gcc conf.]
         *
         *   BN_mod_exp_recp   50 .. 70 %  [AMD K6-2, Linux, debug configuration
         *                     62 .. 118 % [UltraSparc, debug-solaris-sparcv8-gc
         *
         * On the Sparc, BN_mod_exp_recp was faster than BN_mod_exp_mont
         * at 2048 and more bits, but at 512 and 1024 bits, it was
         * slower even than the standard algorithm!
         *
         * "Real" timings [linux-elf, solaris-sparcv9-gcc configurations]
         * should be obtained when the new Montgomery reduction code
         * has been integrated into OpenSSL.)
         */

#define MONT_MUL_MOD
#define MONT_EXP_WORD
#define RECP_MUL_MOD

#ifdef MONT_MUL_MOD
        /* I have finally been able to take out this pre-condition of
         * the top bit being set.  It was caused by an error in BN_div
         * with negatives.  There was also another problem when for a^b%m
         * a >= m.  eay 07-May-97 */
/*      if ((m->d[m->top-1]&BN_TBIT) && BN_is_odd(m)) */

        if (BN_is_odd(m))
                {
#  ifdef MONT_EXP_WORD
                if (a->top == 1 && !a->neg && (BN_get_flags(p, BN_FLG_CONSTTIME)
                        {
                        BN_ULONG A = a->d[0];
                        ret=BN_mod_exp_mont_word(r,A,p,m,ctx,NULL);
                        }
                else
#  endif
                        ret=BN_mod_exp_mont(r,a,p,m,ctx,NULL);
                }
        else
#endif
#ifdef RECP_MUL_MOD
                { ret=BN_mod_exp_recp(r,a,p,m,ctx); }
#else
                { ret=BN_mod_exp_simple(r,a,p,m,ctx); }
#endif

        bn_check_top(r);
        return(ret);
        }


int BN_mod_exp_recp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
                    const BIGNUM *m, BN_CTX *ctx)
        {
        int i,j,bits,ret=0,wstart,wend,window,wvalue;
        int start=1;
        BIGNUM *aa;
        /* Table of variables obtained from 'ctx' */
        BIGNUM *val[TABLE_SIZE];
        BN_RECP_CTX recp;
```

```
261         if (BN_get_flags(p, BN_FLG_CONSTTIME) != 0)
262                 {
263                 /* BN_FLG_CONSTTIME only supported by BN_mod_exp_mont() */
264                 BNerr(BN_F_BN_MOD_EXP_RECP,ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
265                 return -1;
266                 }

268         bits=BN_num_bits(p);

270         if (bits == 0)
271                 {
272                 ret = BN_one(r);
273                 return ret;
274                 }

276         BN_CTX_start(ctx);
277         aa = BN_CTX_get(ctx);
278         val[0] = BN_CTX_get(ctx);
279         if(!aa || !val[0]) goto err;

281         BN_RECP_CTX_init(&recp);
282         if (m->neg)
283                 {
284                 /* ignore sign of 'm' */
285                 if (!BN_copy(aa, m)) goto err;
286                 aa->neg = 0;
287                 if (BN_RECP_CTX_set(&recp,aa,ctx) <= 0) goto err;
288                 }
289         else
290                 {
291                 if (BN_RECP_CTX_set(&recp,m,ctx) <= 0) goto err;
292                 }

294         if (!BN_nnmod(val[0],a,m,ctx)) goto err;                /* 1 */
295         if (BN_is_zero(val[0]))
296                 {
297                 BN_zero(r);
298                 ret = 1;
299                 goto err;
300                 }

302         window = BN_window_bits_for_exponent_size(bits);
303         if (window > 1)
304                 {
305                 if (!BN_mod_mul_reciprocal(aa,val[0],val[0],&recp,ctx))
306                         goto err;                               /* 2 */
307                 j=1<<(window-1);
308                 for (i=1; i<j; i++)
309                         {
310                         if(((val[i] = BN_CTX_get(ctx)) == NULL) ||
311                                         !BN_mod_mul_reciprocal(val[i],val[i-1],
312                                                 aa,&recp,ctx))
313                                 goto err;
314                         }
315                 }

317         start=1;        /* This is used to avoid multiplication etc
318                          * when there is only the value '1' in the
319                          * buffer. */
320         wvalue=0;       /* The 'value' of the window */
321         wstart=bits-1;  /* The top bit of the window */
322         wend=0;         /* The bottom bit of the window */

324         if (!BN_one(r)) goto err;
```



```
326         for (;;)
327                 {
328                 if (BN_is_bit_set(p,wstart) == 0)
329                         {
330                         if (!start)
331                                 if (!BN_mod_mul_reciprocal(r,r,r,&recp,ctx))
332                                 goto err;
333                         if (wstart == 0) break;
334                         wstart--;
335                         continue;
336                         }
337                 /* We now have wstart on a 'set' bit, we now need to work out
338                  * how bit a window to do.  To do this we need to scan
339                  * forward until the last set bit before the end of the
340                  * window */
341                 j=wstart;
342                 wvalue=1;
343                 wend=0;
344                 for (i=1; i<window; i++)
345                         {
346                         if (wstart-i < 0) break;
347                         if (BN_is_bit_set(p,wstart-i))
348                                 {
349                                 wvalue<<=(i-wend);
350                                 wvalue|=1;
351                                 wend=i;
352                                 }
353                         }

355                 /* wend is the size of the current window */
356                 j=wend+1;
357                 /* add the 'bytes above' */
358                 if (!start)
359                         for (i=0; i<j; i++)
360                                 {
361                                 if (!BN_mod_mul_reciprocal(r,r,r,&recp,ctx))
362                                         goto err;
363                                 }

365                 /* wvalue will be an odd number < 2^window */
366                 if (!BN_mod_mul_reciprocal(r,r,val[wvalue>>1],&recp,ctx))
367                         goto err;

369                 /* move the 'window' down further */
370                 wstart-=wend+1;
371                 wvalue=0;
372                 start=0;
373                 if (wstart < 0) break;
374                 }
375         ret=1;
376 err:
377         BN_CTX_end(ctx);
378         BN_RECP_CTX_free(&recp);
379         bn_check_top(r);
380         return(ret);
381         }


384 int BN_mod_exp_mont(BIGNUM *rr, const BIGNUM *a, const BIGNUM *p,
385                     const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *in_mont)
386         {
387         int i,j,bits,ret=0,wstart,wend,window,wvalue;
388         int start=1;
389         BIGNUM *d,*r;
390         const BIGNUM *aa;
391         /* Table of variables obtained from 'ctx' */
```

```
392         BIGNUM *val[TABLE_SIZE];
393         BN_MONT_CTX *mont=NULL;
395         if (BN_get_flags(p, BN_FLG_CONSTTIME) != 0)
396                 {
397                 return BN_mod_exp_mont_consttime(rr, a, p, m, ctx, in_mont);
398                 }
400         bn_check_top(a);
401         bn_check_top(p);
402         bn_check_top(m);
404         if (!BN_is_odd(m))
405                 {
406                 BNerr(BN_F_BN_MOD_EXP_MONT,BN_R_CALLED_WITH_EVEN_MODULUS);
407                 return(0);
408                 }
409         bits=BN_num_bits(p);
410         if (bits == 0)
411                 {
412                 ret = BN_one(rr);
413                 return ret;
414                 }
416         BN_CTX_start(ctx);
417         d = BN_CTX_get(ctx);
418         r = BN_CTX_get(ctx);
419         val[0] = BN_CTX_get(ctx);
420         if (!d || !r || !val[0]) goto err;
422         /* If this is not done, things will break in the montgomery
423          * part */
425         if (in_mont != NULL)
426                 mont=in_mont;
427         else
428                 {
429                 if ((mont=BN_MONT_CTX_new()) == NULL) goto err;
430                 if (!BN_MONT_CTX_set(mont,m,ctx)) goto err;
431                 }
433         if (a->neg || BN_ucmp(a,m) >= 0)
434                 {
435                 if (!BN_nnmod(val[0],a,m,ctx))
436                         goto err;
437                 aa= val[0];
438                 }
439         else
440                 aa=a;
441         if (BN_is_zero(aa))
442                 {
443                 BN_zero(rr);
444                 ret = 1;
445                 goto err;
446                 }
447         if (!BN_to_montgomery(val[0],aa,mont,ctx)) goto err; /* 1 */
449         window = BN_window_bits_for_exponent_size(bits);
450         if (window > 1)
451                 {
452                 if (!BN_mod_mul_montgomery(d,val[0],val[0],mont,ctx)) goto err;
453                 j=1<<(window-1);
454                 for (i=1; i<j; i++)
455                         {
456                         if(((val[i] = BN_CTX_get(ctx)) == NULL) ||
457                                         !BN_mod_mul_montgomery(val[i],val[i-1],
```



```
458                                                 d,mont,ctx))
459                                 goto err;
460                         }
461                 }
463         start=1;        /* This is used to avoid multiplication etc
464                          * when there is only the value '1' in the
465                          * buffer. */
466         wvalue=0;       /* The 'value' of the window */
467         wstart=bits-1;  /* The top bit of the window */
468         wend=0;         /* The bottom bit of the window */
470         if (!BN_to_montgomery(r,BN_value_one(),mont,ctx)) goto err;
471         for (;;)
472                 {
473                 if (BN_is_bit_set(p,wstart) == 0)
474                         {
475                         if (!start)
476                                 {
477                                 if (!BN_mod_mul_montgomery(r,r,r,mont,ctx))
478                                 goto err;
479                                 }
480                         if (wstart == 0) break;
481                         wstart--;
482                         continue;
483                         }
484                 /* We now have wstart on a 'set' bit, we now need to work out
485                  * how bit a window to do.  To do this we need to scan
486                  * forward until the last set bit before the end of the
487                  * window */
488                 j=wstart;
489                 wvalue=1;
490                 wend=0;
491                 for (i=1; i<window; i++)
492                         {
493                         if (wstart-i < 0) break;
494                         if (BN_is_bit_set(p,wstart-i))
495                                 {
496                                 wvalue<<=(i-wend);
497                                 wvalue|=1;
498                                 wend=i;
499                                 }
500                         }
502                 /* wend is the size of the current window */
503                 j=wend+1;
504                 /* add the 'bytes above' */
505                 if (!start)
506                         for (i=0; i<j; i++)
507                                 {
508                                 if (!BN_mod_mul_montgomery(r,r,r,mont,ctx))
509                                         goto err;
510                                 }
512                 /* wvalue will be an odd number < 2^window */
513                 if (!BN_mod_mul_montgomery(r,r,val[wvalue>>1],mont,ctx))
514                         goto err;
516                 /* move the 'window' down further */
517                 wstart-=wend+1;
518                 wvalue=0;
519                 start=0;
520                 if (wstart < 0) break;
521                 }
522         if (!BN_from_montgomery(rr,r,mont,ctx)) goto err;
523         ret=1;
```

```
524 err:
525         if ((in_mont == NULL) && (mont != NULL)) BN_MONT_CTX_free(mont);
526         BN_CTX_end(ctx);
527         bn_check_top(rr);
528         return(ret);
529         }


532 /* BN_mod_exp_mont_consttime() stores the precomputed powers in a specific layou
533  * so that accessing any of these table values shows the same access pattern as
534  * as cache lines are concerned.  The following functions are used to transfer a
535  * from/to that table. */

537 static int MOD_EXP_CTIME_COPY_TO_PREBUF(const BIGNUM *b, int top, unsigned char
538         {
539         size_t i, j;

541         if (top > b->top)
542                 top = b->top; /* this works because 'buf' is explicitly zeroed *
543         for (i = 0, j=idx; i < top * sizeof b->d[0]; i++, j+=width)
544                 {
545                 buf[j] = ((unsigned char*)b->d)[i];
546                 }

548         return 1;
549         }

551 static int MOD_EXP_CTIME_COPY_FROM_PREBUF(BIGNUM *b, int top, unsigned char *buf
552         {
553         size_t i, j;

555         if (bn_wexpand(b, top) == NULL)
556                 return 0;

558         for (i=0, j=idx; i < top * sizeof b->d[0]; i++, j+=width)
559                 {
560                 ((unsigned char*)b->d)[i] = buf[j];
561                 }

563         b->top = top;
564         bn_correct_top(b);
565         return 1;
566         }

568 /* Given a pointer value, compute the next address that is a cache line multiple
569 #define MOD_EXP_CTIME_ALIGN(x_) \
570         ((unsigned char*)(x_) + (MOD_EXP_CTIME_MIN_CACHE_LINE_WIDTH - (((size_t)

572 /* This variant of BN_mod_exp_mont() uses fixed windows and the special
573  * precomputation memory layout to limit data-dependency to a minimum
574  * to protect secret exponents (cf. the hyper-threading timing attacks
575  * pointed out by Colin Percival,
576  * http://www.daemonology.net/hyperthreading-considered-harmful/)
577  */
578 int BN_mod_exp_mont_consttime(BIGNUM *rr, const BIGNUM *a, const BIGNUM *p,
579                         const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *in_mont)
580         {
581         int i,bits,ret=0,window,wvalue;
582         int top;
583         BN_MONT_CTX *mont=NULL;

585         int numPowers;
586         unsigned char *powerbufFree=NULL;
587         int powerbufLen = 0;
588         unsigned char *powerbuf=NULL;
589         BIGNUM tmp, am;
```



```
591         bn_check_top(a);
592         bn_check_top(p);
593         bn_check_top(m);

595         top = m->top;

597         if (!(m->d[0] & 1))
598                 {
599                 BNerr(BN_F_BN_MOD_EXP_MONT_CONSTTIME,BN_R_CALLED_WITH_EVEN_MODUL
600                 return(0);
601                 }
602         bits=BN_num_bits(p);
603         if (bits == 0)
604                 {
605                 ret = BN_one(rr);
606                 return ret;
607                 }

609         BN_CTX_start(ctx);

611         /* Allocate a montgomery context if it was not supplied by the caller.
612          * If this is not done, things will break in the montgomery part.
613          */
614         if (in_mont != NULL)
615                 mont=in_mont;
616         else
617                 {
618                 if ((mont=BN_MONT_CTX_new()) == NULL) goto err;
619                 if (!BN_MONT_CTX_set(mont,m,ctx)) goto err;
620                 }

622         /* Get the window size to use with size of p. */
623         window = BN_window_bits_for_ctime_exponent_size(bits);
624 #if defined(OPENSSL_BN_ASM_MONT5)
625         if (window==6 && bits<=1024) window=5;  /* ~5% improvement of 2048-bit R
626 #endif

628         /* Allocate a buffer large enough to hold all of the pre-computed
629          * powers of am, am itself and tmp.
630          */
631         numPowers = 1 << window;
632         powerbufLen = sizeof(m->d[0])*(top*numPowers +
633                                 ((2*top)>numPowers?(2*top):numPowers));
634 #ifdef alloca
635         if (powerbufLen < 3072)
636                 powerbufFree = alloca(powerbufLen+MOD_EXP_CTIME_MIN_CACHE_LINE_W
637         else
638 #endif
639         if ((powerbufFree=(unsigned char*)OPENSSL_malloc(powerbufLen+MOD_EXP_CTI
640                 goto err;

642         powerbuf = MOD_EXP_CTIME_ALIGN(powerbufFree);
643         memset(powerbuf, 0, powerbufLen);

645 #ifdef alloca
646         if (powerbufLen < 3072)
647                 powerbufFree = NULL;
648 #endif

650         /* lay down tmp and am right after powers table */
651         tmp.d     = (BN_ULONG *)(powerbuf + sizeof(m->d[0])*top*numPowers);
652         am.d      = tmp.d + top;
653         tmp.top   = am.top  = 0;
654         tmp.dmax  = am.dmax = top;
655         tmp.neg   = am.neg  = 0;
```

```
656         tmp.flags = am.flags = BN_FLG_STATIC_DATA;

658         /* prepare a^0 in Montgomery domain */
659 #if 1
660         if (!BN_to_montgomery(&tmp,BN_value_one(),mont,ctx))    goto err;
661 #else
662         tmp.d[0] = (0-m->d[0])&BN_MASK2;        /* 2^(top*BN_BITS2) - m */
663         for (i=1;i<top;i++)
664                 tmp.d[i] = (~m->d[i])&BN_MASK2;
665         tmp.top = top;
666 #endif

668         /* prepare a^1 in Montgomery domain */
669         if (a->neg || BN_ucmp(a,m) >= 0)
670                 {
671                 if (!BN_mod(&am,a,m,ctx))                       goto err;
672                 if (!BN_to_montgomery(&am,&am,mont,ctx))        goto err;
673                 }
674         else    if (!BN_to_montgomery(&am,a,mont,ctx))          goto err;

676 #if defined(OPENSSL_BN_ASM_MONT5)
677     /* This optimization uses ideas from http://eprint.iacr.org/2011/239,
678      * specifically optimization of cache-timing attack countermeasures
679      * and pre-computation optimization. */

681     /* Dedicated window==4 case improves 512-bit RSA sign by ~15%, but as
682      * 512-bit RSA is hardly relevant, we omit it to spare size... */
683     if (window==5 && top>1)
684         {
685         void bn_mul_mont_gather5(BN_ULONG *rp,const BN_ULONG *ap,
686                         const void *table,const BN_ULONG *np,
687                         const BN_ULONG *n0,int num,int power);
688         void bn_scatter5(const BN_ULONG *inp,size_t num,
689                         void *table,size_t power);
690         void bn_gather5(BN_ULONG *out,size_t num,
691                         void *table,size_t power);

693         BN_ULONG *np=mont->N.d, *n0=mont->n0;

695         /* BN_to_montgomery can contaminate words above .top
696          * [in BN_DEBUG[_DEBUG] build]... */
697         for (i=am.top; i<top; i++)      am.d[i]=0;
698         for (i=tmp.top; i<top; i++)     tmp.d[i]=0;

700         bn_scatter5(tmp.d,top,powerbuf,0);
701         bn_scatter5(am.d,am.top,powerbuf,1);
702         bn_mul_mont(tmp.d,am.d,am.d,np,n0,top);
703         bn_scatter5(tmp.d,top,powerbuf,2);

705 #if 0
706         for (i=3; i<32; i++)
707                 {
708                 /* Calculate a^i = a^(i-1) * a */
709                 bn_mul_mont_gather5(tmp.d,am.d,powerbuf,np,n0,top,i-1);
710                 bn_scatter5(tmp.d,top,powerbuf,i);
711                 }
712 #else
713         /* same as above, but uses squaring for 1/2 of operations */
714         for (i=4; i<32; i*=2)
715                 {
716                 bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
717                 bn_scatter5(tmp.d,top,powerbuf,i);
718                 }
719         for (i=3; i<8; i+=2)
720                 {
721                 int j;
```



```
722                 bn_mul_mont_gather5(tmp.d,am.d,powerbuf,np,n0,top,i-1);
723                 bn_scatter5(tmp.d,top,powerbuf,i);
724                 for (j=2*i; j<32; j*=2)
725                         {
726                         bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
727                         bn_scatter5(tmp.d,top,powerbuf,j);
728                         }
729                 }
730         for (; i<16; i+=2)
731                 {
732                 bn_mul_mont_gather5(tmp.d,am.d,powerbuf,np,n0,top,i-1);
733                 bn_scatter5(tmp.d,top,powerbuf,i);
734                 bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
735                 bn_scatter5(tmp.d,top,powerbuf,2*i);
736                 }
737         for (; i<32; i+=2)
738                 {
739                 bn_mul_mont_gather5(tmp.d,am.d,powerbuf,np,n0,top,i-1);
740                 bn_scatter5(tmp.d,top,powerbuf,i);
741                 }
742 #endif
743         bits--;
744         for (wvalue=0, i=bits%5; i>=0; i--,bits--)
745                 wvalue = (wvalue<<1)+BN_is_bit_set(p,bits);
746         bn_gather5(tmp.d,top,powerbuf,wvalue);

748         /* Scan the exponent one window at a time starting from the most
749          * significant bits.
750          */
751         while (bits >= 0)
752                 {
753                 for (wvalue=0, i=0; i<5; i++,bits--)
754                         wvalue = (wvalue<<1)+BN_is_bit_set(p,bits);

756                 bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
757                 bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
758                 bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
759                 bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
760                 bn_mul_mont(tmp.d,tmp.d,tmp.d,np,n0,top);
761                 bn_mul_mont_gather5(tmp.d,tmp.d,powerbuf,np,n0,top,wvalue);
762                 }

764         tmp.top=top;
765         bn_correct_top(&tmp);
766         }
767     else
768 #endif
769         {
770         if (!MOD_EXP_CTIME_COPY_TO_PREBUF(&tmp, top, powerbuf, 0, numPowers)) go
771         if (!MOD_EXP_CTIME_COPY_TO_PREBUF(&am,  top, powerbuf, 1, numPowers)) go

773         /* If the window size is greater than 1, then calculate
774          * val[i=2..2^winsize-1]. Powers are computed as a*a^(i-1)
775          * (even powers could instead be computed as (a^(i/2))^2
776          * to use the slight performance advantage of sqr over mul).
777          */
778         if (window > 1)
779                 {
780                 if (!BN_mod_mul_montgomery(&tmp,&am,&am,mont,ctx))      goto err
781                 if (!MOD_EXP_CTIME_COPY_TO_PREBUF(&tmp, top, powerbuf, 2, numPow
782                 for (i=3; i<numPowers; i++)
783                         {
784                         /* Calculate a^i = a^(i-1) * a */
785                         if (!BN_mod_mul_montgomery(&tmp,&am,&tmp,mont,ctx))
786                                 goto err;
787                         if (!MOD_EXP_CTIME_COPY_TO_PREBUF(&tmp, top, powerbuf, i
```

```
788                         }
789                 }

791         bits--;
792         for (wvalue=0, i=bits%window; i>=0; i--,bits--)
793                 wvalue = (wvalue<<1)+BN_is_bit_set(p,bits);
794         if (!MOD_EXP_CTIME_COPY_FROM_PREBUF(&tmp,top,powerbuf,wvalue,numPowers))

796         /* Scan the exponent one window at a time starting from the most
797          * significant bits.
798          */
799         while (bits >= 0)
800                 {
801                 wvalue=0; /* The 'value' of the window */

803                 /* Scan the window, squaring the result as we go */
804                 for (i=0; i<window; i++,bits--)
805                         {
806                         if (!BN_mod_mul_montgomery(&tmp,&tmp,&tmp,mont,ctx))
807                         wvalue = (wvalue<<1)+BN_is_bit_set(p,bits);
808                         }

810                 /* Fetch the appropriate pre-computed value from the pre-buf */
811                 if (!MOD_EXP_CTIME_COPY_FROM_PREBUF(&am, top, powerbuf, wvalue,

813                 /* Multiply the result into the intermediate result */
814                 if (!BN_mod_mul_montgomery(&tmp,&tmp,&am,mont,ctx)) goto err;
815                 }
816         }

818         /* Convert the final result from montgomery to standard format */
819         if (!BN_from_montgomery(rr,&tmp,mont,ctx)) goto err;
820         ret=1;
821 err:
822         if ((in_mont == NULL) && (mont != NULL)) BN_MONT_CTX_free(mont);
823         if (powerbuf!=NULL)
824                 {
825                 OPENSSL_cleanse(powerbuf,powerbufLen);
826                 if (powerbufFree) OPENSSL_free(powerbufFree);
827                 }
828         BN_CTX_end(ctx);
829         return(ret);
830         }

832 int BN_mod_exp_mont_word(BIGNUM *rr, BN_ULONG a, const BIGNUM *p,
833                          const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *in_mont)
834         {
835         BN_MONT_CTX *mont = NULL;
836         int b, bits, ret=0;
837         int r_is_one;
838         BN_ULONG w, next_w;
839         BIGNUM *d, *r, *t;
840         BIGNUM *swap_tmp;
841 #define BN_MOD_MUL_WORD(r, w, m) \
842                 (BN_mul_word(r, (w)) && \
843                 (/* BN_ucmp(r, (m)) < 0 ? 1 :*/  \
844                         (BN_mod(t, r, m, ctx) && (swap_tmp = r, r = t, t = swap_
845                 /* BN_MOD_MUL_WORD is only used with 'w' large,
846                  * so the BN_ucmp test is probably more overhead
847                  * than always using BN_mod (which uses BN_copy if
848                  * a similar test returns true). */
849                 /* We can use BN_mod and do not need BN_nnmod because our
850                  * accumulator is never negative (the result of BN_mod does
851                  * not depend on the sign of the modulus).
852                  */
853 #define BN_TO_MONTGOMERY_WORD(r, w, mont) \
```



```
854                 (BN_set_word(r, (w)) && BN_to_montgomery(r, r, (mont), ctx))

856         if (BN_get_flags(p, BN_FLG_CONSTTIME) != 0)
857                 {
858                 /* BN_FLG_CONSTTIME only supported by BN_mod_exp_mont() */
859                 BNerr(BN_F_BN_MOD_EXP_MONT_WORD,ERR_R_SHOULD_NOT_HAVE_BEEN_CALLE
860                 return -1;
861                 }

863         bn_check_top(p);
864         bn_check_top(m);

866         if (!BN_is_odd(m))
867                 {
868                 BNerr(BN_F_BN_MOD_EXP_MONT_WORD,BN_R_CALLED_WITH_EVEN_MODULUS);
869                 return(0);
870                 }
871         if (m->top == 1)
872                 a %= m->d[0]; /* make sure that 'a' is reduced */

874         bits = BN_num_bits(p);
875         if (bits == 0)
876                 {
877                 ret = BN_one(rr);
878                 return ret;
879                 }
880         if (a == 0)
881                 {
882                 BN_zero(rr);
883                 ret = 1;
884                 return ret;
885                 }

887         BN_CTX_start(ctx);
888         d = BN_CTX_get(ctx);
889         r = BN_CTX_get(ctx);
890         t = BN_CTX_get(ctx);
891         if (d == NULL || r == NULL || t == NULL) goto err;

893         if (in_mont != NULL)
894                 mont=in_mont;
895         else
896                 {
897                 if ((mont = BN_MONT_CTX_new()) == NULL) goto err;
898                 if (!BN_MONT_CTX_set(mont, m, ctx)) goto err;
899                 }

901         r_is_one = 1; /* except for Montgomery factor */

903         /* bits-1 >= 0 */

905         /* The result is accumulated in the product r*w. */
906         w = a; /* bit 'bits-1' of 'p' is always set */
907         for (b = bits-2; b >= 0; b--)
908                 {
909                 /* First, square r*w. */
910                 next_w = w*w;
911                 if ((next_w/w) != w) /* overflow */
912                         {
913                         if (r_is_one)
914                                 {
915                                 if (!BN_TO_MONTGOMERY_WORD(r, w, mont)) goto err
916                                 r_is_one = 0;
917                                 }
918                         else
919                                 {
```

```
920                                 if (!BN_MOD_MUL_WORD(r, w, m)) goto err;
921                                 }
922                         next_w = 1;
923                         }
924                 w = next_w;
925                 if (!r_is_one)
926                         {
927                         if (!BN_mod_mul_montgomery(r, r, r, mont, ctx)) goto err
928                         }

930                 /* Second, multiply r*w by 'a' if exponent bit is set. */
931                 if (BN_is_bit_set(p, b))
932                         {
933                         next_w = w*a;
934                         if ((next_w/a) != w) /* overflow */
935                                 {
936                                 if (r_is_one)
937                                         {
938                                         if (!BN_TO_MONTGOMERY_WORD(r, w, mont))
939                                         r_is_one = 0;
940                                         }
941                                 else
942                                         {
943                                         if (!BN_MOD_MUL_WORD(r, w, m)) goto err;
944                                         }
945                                 next_w = a;
946                                 }
947                         w = next_w;
948                         }
949                 }

951         /* Finally, set r:=r*w. */
952         if (w != 1)
953                 {
954                 if (r_is_one)
955                         {
956                         if (!BN_TO_MONTGOMERY_WORD(r, w, mont)) goto err;
957                         r_is_one = 0;
958                         }
959                 else
960                         {
961                         if (!BN_MOD_MUL_WORD(r, w, m)) goto err;
962                         }
963                 }

965         if (r_is_one) /* can happen only if a == 1*/
966                 {
967                 if (!BN_one(rr)) goto err;
968                 }
969         else
970                 {
971                 if (!BN_from_montgomery(rr, r, mont, ctx)) goto err;
972                 }
973         ret = 1;
974 err:
975         if ((in_mont == NULL) && (mont != NULL)) BN_MONT_CTX_free(mont);
976         BN_CTX_end(ctx);
977         bn_check_top(rr);
978         return(ret);
979         }


982 /* The old fallback, simple version :-) */
983 int BN_mod_exp_simple(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
984                 const BIGNUM *m, BN_CTX *ctx)
985         {
```



```
986         int i,j,bits,ret=0,wstart,wend,window,wvalue;
987         int start=1;
988         BIGNUM *d;
989         /* Table of variables obtained from 'ctx' */
990         BIGNUM *val[TABLE_SIZE];

992         if (BN_get_flags(p, BN_FLG_CONSTTIME) != 0)
993                 {
994                 /* BN_FLG_CONSTTIME only supported by BN_mod_exp_mont() */
995                 BNerr(BN_F_BN_MOD_EXP_SIMPLE,ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
996                 return -1;
997                 }

999         bits=BN_num_bits(p);

1001        if (bits == 0)
1002                {
1003                ret = BN_one(r);
1004                return ret;
1005                }

1007        BN_CTX_start(ctx);
1008        d = BN_CTX_get(ctx);
1009        val[0] = BN_CTX_get(ctx);
1010        if(!d || !val[0]) goto err;

1012        if (!BN_nnmod(val[0],a,m,ctx)) goto err;                /* 1 */
1013        if (BN_is_zero(val[0]))
1014                {
1015                BN_zero(r);
1016                ret = 1;
1017                goto err;
1018                }

1020        window = BN_window_bits_for_exponent_size(bits);
1021        if (window > 1)
1022                {
1023                if (!BN_mod_mul(d,val[0],val[0],m,ctx))
1024                        goto err;                               /* 2 */
1025                j=1<<(window-1);
1026                for (i=1; i<j; i++)
1027                        {
1028                        if(((val[i] = BN_CTX_get(ctx)) == NULL) ||
1029                                        !BN_mod_mul(val[i],val[i-1],d,m,ctx))
1030                                goto err;
1031                        }
1032                }

1034        start=1;        /* This is used to avoid multiplication etc
1035                         * when there is only the value '1' in the
1036                         * buffer. */
1037        wvalue=0;       /* The 'value' of the window */
1038        wstart=bits-1;  /* The top bit of the window */
1039        wend=0;         /* The bottom bit of the window */

1041        if (!BN_one(r)) goto err;

1043        for (;;)
1044                {
1045                if (BN_is_bit_set(p,wstart) == 0)
1046                        {
1047                        if (!start)
1048                                if (!BN_mod_mul(r,r,r,m,ctx))
1049                                goto err;
1050                        if (wstart == 0) break;
1051                        wstart--;
```

```
1052                        continue;
1053                        }
1054                /* We now have wstart on a 'set' bit, we now need to work out
1055                 * how bit a window to do.  To do this we need to scan
1056                 * forward until the last set bit before the end of the
1057                 * window */
1058                j=wstart;
1059                wvalue=1;
1060                wend=0;
1061                for (i=1; i<window; i++)
1062                        {
1063                        if (wstart-i < 0) break;
1064                        if (BN_is_bit_set(p,wstart-i))
1065                                {
1066                                wvalue<<=(i-wend);
1067                                wvalue|=1;
1068                                wend=i;
1069                                }
1070                        }

1072                /* wend is the size of the current window */
1073                j=wend+1;
1074                /* add the 'bytes above' */
1075                if (!start)
1076                        for (i=0; i<j; i++)
1077                                {
1078                                if (!BN_mod_mul(r,r,r,m,ctx))
1079                                        goto err;
1080                                }

1082                /* wvalue will be an odd number < 2^window */
1083                if (!BN_mod_mul(r,r,val[wvalue>>1],m,ctx))
1084                        goto err;

1086                /* move the 'window' down further */
1087                wstart-=wend+1;
1088                wvalue=0;
1089                start=0;
1090                if (wstart < 0) break;
1091                }
1092        ret=1;
1093 err:
1094        BN_CTX_end(ctx);
1095        bn_check_top(r);
1096        return(ret);
1097        }
1098 #endif /* ! codereview */
```

```
**********************************************************
   10126 Wed Aug 13 19:52:15 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_exp2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_exp2.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"

#define TABLE_SIZE      32

int BN_mod_exp2_mont(BIGNUM *rr, const BIGNUM *a1, const BIGNUM *p1,
        const BIGNUM *a2, const BIGNUM *p2, const BIGNUM *m,
        BN_CTX *ctx, BN_MONT_CTX *in_mont)
        {
        int i,j,bits,b,bits1,bits2,ret=0,wpos1,wpos2,window1,window2,wvalue1,wva
        int r_is_one=1;
        BIGNUM *d,*r;
        const BIGNUM *a_mod_m;
        /* Tables of variables obtained from 'ctx' */
        BIGNUM *val1[TABLE_SIZE], *val2[TABLE_SIZE];
```

```
128         BN_MONT_CTX *mont=NULL;

130         bn_check_top(a1);
131         bn_check_top(p1);
132         bn_check_top(a2);
133         bn_check_top(p2);
134         bn_check_top(m);

136         if (!(m->d[0] & 1))
137                 {
138                 BNerr(BN_F_BN_MOD_EXP2_MONT,BN_R_CALLED_WITH_EVEN_MODULUS);
139                 return(0);
140                 }
141         bits1=BN_num_bits(p1);
142         bits2=BN_num_bits(p2);
143         if ((bits1 == 0) && (bits2 == 0))
144                 {
145                 ret = BN_one(rr);
146                 return ret;
147                 }

149         bits=(bits1 > bits2)?bits1:bits2;

151         BN_CTX_start(ctx);
152         d = BN_CTX_get(ctx);
153         r = BN_CTX_get(ctx);
154         val1[0] = BN_CTX_get(ctx);
155         val2[0] = BN_CTX_get(ctx);
156         if(!d || !r || !val1[0] || !val2[0]) goto err;

158         if (in_mont != NULL)
159                 mont=in_mont;
160         else
161                 {
162                 if ((mont=BN_MONT_CTX_new()) == NULL) goto err;
163                 if (!BN_MONT_CTX_set(mont,m,ctx)) goto err;
164                 }

166         window1 = BN_window_bits_for_exponent_size(bits1);
167         window2 = BN_window_bits_for_exponent_size(bits2);

169         /*
170          * Build table for a1:   val1[i] := a1^(2*i + 1) mod m  for i = 0 .. 2^(
171          */
172         if (a1->neg || BN_ucmp(a1,m) >= 0)
173                 {
174                 if (!BN_mod(val1[0],a1,m,ctx))
175                         goto err;
176                 a_mod_m = val1[0];
177                 }
178         else
179                 a_mod_m = a1;
180         if (BN_is_zero(a_mod_m))
181                 {
182                 BN_zero(rr);
183                 ret = 1;
184                 goto err;
185                 }

187         if (!BN_to_montgomery(val1[0],a_mod_m,mont,ctx)) goto err;
188         if (window1 > 1)
189                 {
190                 if (!BN_mod_mul_montgomery(d,val1[0],val1[0],mont,ctx)) goto err

192                 j=1<<(window1-1);
193                 for (i=1; i<j; i++)
```



```
194                         {
195                         if(((val1[i] = BN_CTX_get(ctx)) == NULL) ||
196                                         !BN_mod_mul_montgomery(val1[i],val1[i-1]
197                                                 d,mont,ctx))
198                                 goto err;
199                         }
200                 }


203         /*
204          * Build table for a2:   val2[i] := a2^(2*i + 1) mod m  for i = 0 .. 2^(
205          */
206         if (a2->neg || BN_ucmp(a2,m) >= 0)
207                 {
208                 if (!BN_mod(val2[0],a2,m,ctx))
209                         goto err;
210                 a_mod_m = val2[0];
211                 }
212         else
213                 a_mod_m = a2;
214         if (BN_is_zero(a_mod_m))
215                 {
216                 BN_zero(rr);
217                 ret = 1;
218                 goto err;
219                 }
220         if (!BN_to_montgomery(val2[0],a_mod_m,mont,ctx)) goto err;
221         if (window2 > 1)
222                 {
223                 if (!BN_mod_mul_montgomery(d,val2[0],val2[0],mont,ctx)) goto err

225                 j=1<<(window2-1);
226                 for (i=1; i<j; i++)
227                         {
228                         if(((val2[i] = BN_CTX_get(ctx)) == NULL) ||
229                                         !BN_mod_mul_montgomery(val2[i],val2[i-1]
230                                                 d,mont,ctx))
231                                 goto err;
232                         }
233                 }


236         /* Now compute the power product, using independent windows. */
237         r_is_one=1;
238         wvalue1=0;  /* The 'value' of the first window */
239         wvalue2=0;  /* The 'value' of the second window */
240         wpos1=0;    /* If wvalue1 > 0, the bottom bit of the first window */
241         wpos2=0;    /* If wvalue2 > 0, the bottom bit of the second window */

243         if (!BN_to_montgomery(r,BN_value_one(),mont,ctx)) goto err;
244         for (b=bits-1; b>=0; b--)
245                 {
246                 if (!r_is_one)
247                         {
248                         if (!BN_mod_mul_montgomery(r,r,r,mont,ctx))
249                                 goto err;
250                         }

252                 if (!wvalue1)
253                         if (BN_is_bit_set(p1, b))
254                                 {
255                                 /* consider bits b-window1+1 .. b for this windo
256                                 i = b-window1+1;
257                                 while (!BN_is_bit_set(p1, i)) /* works for i<0 *
258                                         i++;
259                                 wpos1 = i;
```

```
260                                 wvalue1 = 1;
261                                 for (i = b-1; i >= wpos1; i--)
262                                         {
263                                         wvalue1 <<= 1;
264                                         if (BN_is_bit_set(p1, i))
265                                                 wvalue1++;
266                                         }
267                                 }

269                 if (!wvalue2)
270                         if (BN_is_bit_set(p2, b))
271                                 {
272                                 /* consider bits b-window2+1 .. b for this windo
273                                 i = b-window2+1;
274                                 while (!BN_is_bit_set(p2, i))
275                                         i++;
276                                 wpos2 = i;
277                                 wvalue2 = 1;
278                                 for (i = b-1; i >= wpos2; i--)
279                                         {
280                                         wvalue2 <<= 1;
281                                         if (BN_is_bit_set(p2, i))
282                                                 wvalue2++;
283                                         }
284                                 }

286                 if (wvalue1 && b == wpos1)
287                         {
288                         /* wvalue1 is odd and < 2^window1 */
289                         if (!BN_mod_mul_montgomery(r,r,val1[wvalue1>>1],mont,ctx
290                                 goto err;
291                         wvalue1 = 0;
292                         r_is_one = 0;
293                         }

295                 if (wvalue2 && b == wpos2)
296                         {
297                         /* wvalue2 is odd and < 2^window2 */
298                         if (!BN_mod_mul_montgomery(r,r,val2[wvalue2>>1],mont,ctx
299                                 goto err;
300                         wvalue2 = 0;
301                         r_is_one = 0;
302                         }
303                 }
304         if (!BN_from_montgomery(rr,r,mont,ctx))
305                 goto err;
306         ret=1;
307 err:
308         if ((in_mont == NULL) && (mont != NULL)) BN_MONT_CTX_free(mont);
309         BN_CTX_end(ctx);
310         bn_check_top(rr);
311         return(ret);
312         }
313 #endif /* ! codereview */
```

```
**********************************************************
   17412 Wed Aug 13 19:52:15 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_gcd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_gcd.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include "bn_lcl.h"

static BIGNUM *euclid(BIGNUM *a, BIGNUM *b);

int BN_gcd(BIGNUM *r, const BIGNUM *in_a, const BIGNUM *in_b, BN_CTX *ctx)
        {
        BIGNUM *a,*b,*t;
        int ret=0;

        bn_check_top(in_a);
        bn_check_top(in_b);

        BN_CTX_start(ctx);
        a = BN_CTX_get(ctx);
        b = BN_CTX_get(ctx);
```

```
128         if (a == NULL || b == NULL) goto err;

130         if (BN_copy(a,in_a) == NULL) goto err;
131         if (BN_copy(b,in_b) == NULL) goto err;
132         a->neg = 0;
133         b->neg = 0;

135         if (BN_cmp(a,b) < 0) { t=a; a=b; b=t; }
136         t=euclid(a,b);
137         if (t == NULL) goto err;

139         if (BN_copy(r,t) == NULL) goto err;
140         ret=1;
141 err:
142         BN_CTX_end(ctx);
143         bn_check_top(r);
144         return(ret);
145         }

147 static BIGNUM *euclid(BIGNUM *a, BIGNUM *b)
148         {
149         BIGNUM *t;
150         int shifts=0;

152         bn_check_top(a);
153         bn_check_top(b);

155         /* 0 <= b <= a */
156         while (!BN_is_zero(b))
157                 {
158                 /* 0 < b <= a */

160                 if (BN_is_odd(a))
161                         {
162                         if (BN_is_odd(b))
163                                 {
164                                 if (!BN_sub(a,a,b)) goto err;
165                                 if (!BN_rshift1(a,a)) goto err;
166                                 if (BN_cmp(a,b) < 0)
167                                         { t=a; a=b; b=t; }
168                                 }
169                         else            /* a odd - b even */
170                                 {
171                                 if (!BN_rshift1(b,b)) goto err;
172                                 if (BN_cmp(a,b) < 0)
173                                         { t=a; a=b; b=t; }
174                                 }
175                         }
176                 else                    /* a is even */
177                         {
178                         if (BN_is_odd(b))
179                                 {
180                                 if (!BN_rshift1(a,a)) goto err;
181                                 if (BN_cmp(a,b) < 0)
182                                         { t=a; a=b; b=t; }
183                                 }
184                         else            /* a even - b even */
185                                 {
186                                 if (!BN_rshift1(a,a)) goto err;
187                                 if (!BN_rshift1(b,b)) goto err;
188                                 shifts++;
189                                 }
190                         }
191                 /* 0 <= b <= a */
192                 }
```



```
194         if (shifts)
195                 {
196                 if (!BN_lshift(a,a,shifts)) goto err;
197                 }
198         bn_check_top(a);
199         return(a);
200 err:
201         return(NULL);
202         }


205 /* solves ax == 1 (mod n) */
206 static BIGNUM *BN_mod_inverse_no_branch(BIGNUM *in,
207         const BIGNUM *a, const BIGNUM *n, BN_CTX *ctx);

209 BIGNUM *BN_mod_inverse(BIGNUM *in,
210         const BIGNUM *a, const BIGNUM *n, BN_CTX *ctx)
211         {
212         BIGNUM *A,*B,*X,*Y,*M,*D,*T,*R=NULL;
213         BIGNUM *ret=NULL;
214         int sign;

216         if ((BN_get_flags(a, BN_FLG_CONSTTIME) != 0) || (BN_get_flags(n, BN_FLG_
217                 {
218                 return BN_mod_inverse_no_branch(in, a, n, ctx);
219                 }

221         bn_check_top(a);
222         bn_check_top(n);

224         BN_CTX_start(ctx);
225         A = BN_CTX_get(ctx);
226         B = BN_CTX_get(ctx);
227         X = BN_CTX_get(ctx);
228         D = BN_CTX_get(ctx);
229         M = BN_CTX_get(ctx);
230         Y = BN_CTX_get(ctx);
231         T = BN_CTX_get(ctx);
232         if (T == NULL) goto err;

234         if (in == NULL)
235                 R=BN_new();
236         else
237                 R=in;
238         if (R == NULL) goto err;

240         BN_one(X);
241         BN_zero(Y);
242         if (BN_copy(B,a) == NULL) goto err;
243         if (BN_copy(A,n) == NULL) goto err;
244         A->neg = 0;
245         if (B->neg || (BN_ucmp(B, A) >= 0))
246                 {
247                 if (!BN_nnmod(B, B, A, ctx)) goto err;
248                 }
249         sign = -1;
250         /* From  B = a mod |n|,  A = |n|  it follows that
251          *
252          *      0 <= B < A,
253          *     -sign*X*a  ==  B   (mod |n|),
254          *      sign*Y*a  ==  A   (mod |n|).
255          */

257         if (BN_is_odd(n) && (BN_num_bits(n) <= (BN_BITS <= 32 ? 450 : 2048)))
258                 {
259                 /* Binary inversion algorithm; requires odd modulus.
```

```
260                  * This is faster than the general algorithm if the modulus
261                  * is sufficiently small (about 400 .. 500 bits on 32-bit
262                  * sytems, but much more on 64-bit systems) */
263                 int shift;
264
265                 while (!BN_is_zero(B))
266                         {
267                         /*
268                          *      0 < B < |n|,
269                          *      0 < A <= |n|,
270                          * (1) -sign*X*a  ==  B   (mod |n|),
271                          * (2)  sign*Y*a  ==  A   (mod |n|)
272                          */
273
274                         /* Now divide  B  by the maximum possible power of two i
275                          * and divide  X  by the same value mod |n|.
276                          * When we're done, (1) still holds. */
277                         shift = 0;
278                         while (!BN_is_bit_set(B, shift)) /* note that 0 < B */
279                                 {
280                                 shift++;
281
282                                 if (BN_is_odd(X))
283                                         {
284                                         if (!BN_uadd(X, X, n)) goto err;
285                                         }
286                                 /* now X is even, so we can easily divide it by
287                                 if (!BN_rshift1(X, X)) goto err;
288                                 }
289                         if (shift > 0)
290                                 {
291                                 if (!BN_rshift(B, B, shift)) goto err;
292                                 }
293
294
295                         /* Same for  A  and  Y.  Afterwards, (2) still holds. */
296                         shift = 0;
297                         while (!BN_is_bit_set(A, shift)) /* note that 0 < A */
298                                 {
299                                 shift++;
300
301                                 if (BN_is_odd(Y))
302                                         {
303                                         if (!BN_uadd(Y, Y, n)) goto err;
304                                         }
305                                 /* now Y is even */
306                                 if (!BN_rshift1(Y, Y)) goto err;
307                                 }
308                         if (shift > 0)
309                                 {
310                                 if (!BN_rshift(A, A, shift)) goto err;
311                                 }
312
313
314                         /* We still have (1) and (2).
315                          * Both  A  and  B  are odd.
316                          * The following computations ensure that
317                          *
318                          *     0 <= B < |n|,
319                          *      0 < A < |n|,
320                          * (1) -sign*X*a  ==  B   (mod |n|),
321                          * (2)  sign*Y*a  ==  A   (mod |n|),
322                          *
323                          * and that either  A  or  B  is even in the next iterat
324                          */
325                         if (BN_ucmp(B, A) >= 0)
```



```
326                                 {
327                                 /* -sign*(X + Y)*a == B - A  (mod |n|) */
328                                 if (!BN_uadd(X, X, Y)) goto err;
329                                 /* NB: we could use BN_mod_add_quick(X, X, Y, n)
330                                  * actually makes the algorithm slower */
331                                 if (!BN_usub(B, B, A)) goto err;
332                                 }
333                         else
334                                 {
335                                 /*  sign*(X + Y)*a == A - B  (mod |n|) */
336                                 if (!BN_uadd(Y, Y, X)) goto err;
337                                 /* as above, BN_mod_add_quick(Y, Y, X, n) would
338                                 if (!BN_usub(A, A, B)) goto err;
339                                 }
340                         }
341                 }
342         else
343                 {
344                 /* general inversion algorithm */
345
346                 while (!BN_is_zero(B))
347                         {
348                         BIGNUM *tmp;
349
350                         /*
351                          *      0 < B < A,
352                          * (*) -sign*X*a  ==  B   (mod |n|),
353                          *      sign*Y*a  ==  A   (mod |n|)
354                          */
355
356                         /* (D, M) := (A/B, A%B) ... */
357                         if (BN_num_bits(A) == BN_num_bits(B))
358                                 {
359                                 if (!BN_one(D)) goto err;
360                                 if (!BN_sub(M,A,B)) goto err;
361                                 }
362                         else if (BN_num_bits(A) == BN_num_bits(B) + 1)
363                                 {
364                                 /* A/B is 1, 2, or 3 */
365                                 if (!BN_lshift1(T,B)) goto err;
366                                 if (BN_ucmp(A,T) < 0)
367                                         {
368                                         /* A < 2*B, so D=1 */
369                                         if (!BN_one(D)) goto err;
370                                         if (!BN_sub(M,A,B)) goto err;
371                                         }
372                                 else
373                                         {
374                                         /* A >= 2*B, so D=2 or D=3 */
375                                         if (!BN_sub(M,A,T)) goto err;
376                                         if (!BN_add(D,T,B)) goto err; /* use D (
377                                         if (BN_ucmp(A,D) < 0)
378                                                 {
379                                                 /* A < 3*B, so D=2 */
380                                                 if (!BN_set_word(D,2)) goto err;
381                                                 /* M (= A - 2*B) already has the
382                                                 }
383                                         else
384                                                 {
385                                                 /* only D=3 remains */
386                                                 if (!BN_set_word(D,3)) goto err;
387                                                 /* currently  M = A - 2*B,  but
388                                                 if (!BN_sub(M,M,B)) goto err;
389                                                 }
390                                         }
391                                 }
```

```
392                         else
393                                 {
394                                 if (!BN_div(D,M,A,B,ctx)) goto err;
395                                 }
396
397                         /* Now
398                          *      A = D*B + M;
399                          * thus we have
400                          * (**)  sign*Y*a  ==  D*B + M   (mod |n|).
401                          */
402
403                         tmp=A; /* keep the BIGNUM object, the value does not mat
404
405                         /* (A, B) := (B, A mod B) ... */
406                         A=B;
407                         B=M;
408                         /* ... so we have  0 <= B < A  again */
409
410                         /* Since the former  M  is now  B  and the former  B  is
411                          * (**) translates into
412                          *       sign*Y*a  ==  D*A + B    (mod |n|),
413                          * i.e.
414                          *       sign*Y*a - D*A  ==  B    (mod |n|).
415                          * Similarly, (*) translates into
416                          *      -sign*X*a  ==  A          (mod |n|).
417                          *
418                          * Thus,
419                          *   sign*Y*a + D*sign*X*a  ==  B  (mod |n|),
420                          * i.e.
421                          *        sign*(Y + D*X)*a  ==  B  (mod |n|).
422                          *
423                          * So if we set  (X, Y, sign) := (Y + D*X, X, -sign),  w
424                          *      -sign*X*a  ==  B   (mod |n|),
425                          *       sign*Y*a  ==  A   (mod |n|).
426                          * Note that  X  and  Y  stay non-negative all the time.
427                          */
428
429                         /* most of the time D is very small, so we can optimize
430                         if (BN_is_one(D))
431                                 {
432                                 if (!BN_add(tmp,X,Y)) goto err;
433                                 }
434                         else
435                                 {
436                                 if (BN_is_word(D,2))
437                                         {
438                                         if (!BN_lshift1(tmp,X)) goto err;
439                                         }
440                                 else if (BN_is_word(D,4))
441                                         {
442                                         if (!BN_lshift(tmp,X,2)) goto err;
443                                         }
444                                 else if (D->top == 1)
445                                         {
446                                         if (!BN_copy(tmp,X)) goto err;
447                                         if (!BN_mul_word(tmp,D->d[0])) goto err;
448                                         }
449                                 else
450                                         {
451                                         if (!BN_mul(tmp,D,X,ctx)) goto err;
452                                         }
453                                 if (!BN_add(tmp,tmp,Y)) goto err;
454                                 }
455
456                         M=Y; /* keep the BIGNUM object, the value does not matte
457                         Y=X;
```



```
458                                 X=tmp;
459                                 sign = -sign;
460                                 }
461                         }
462
463                 /*
464                  * The while loop (Euclid's algorithm) ends when
465                  *      A == gcd(a,n);
466                  * we have
467                  *       sign*Y*a  ==  A  (mod |n|),
468                  * where  Y  is non-negative.
469                  */
470
471                 if (sign < 0)
472                         {
473                         if (!BN_sub(Y,n,Y)) goto err;
474                         }
475                 /* Now  Y*a  ==  A  (mod |n|).  */
476
477
478                 if (BN_is_one(A))
479                         {
480                         /* Y*a == 1  (mod |n|) */
481                         if (!Y->neg && BN_ucmp(Y,n) < 0)
482                                 {
483                                 if (!BN_copy(R,Y)) goto err;
484                                 }
485                         else
486                                 {
487                                 if (!BN_nnmod(R,Y,n,ctx)) goto err;
488                                 }
489                         }
490                 else
491                         {
492                         BNerr(BN_F_BN_MOD_INVERSE,BN_R_NO_INVERSE);
493                         goto err;
494                         }
495                 ret=R;
496         err:
497                 if ((ret == NULL) && (in == NULL)) BN_free(R);
498                 BN_CTX_end(ctx);
499                 bn_check_top(ret);
500                 return(ret);
501                 }
502
503
504         /* BN_mod_inverse_no_branch is a special version of BN_mod_inverse.
505          * It does not contain branches that may leak sensitive information.
506          */
507         static BIGNUM *BN_mod_inverse_no_branch(BIGNUM *in,
508                 const BIGNUM *a, const BIGNUM *n, BN_CTX *ctx)
509                 {
510                 BIGNUM *A,*B,*X,*Y,*M,*D,*T,*R=NULL;
511                 BIGNUM local_A, local_B;
512                 BIGNUM *pA, *pB;
513                 BIGNUM *ret=NULL;
514                 int sign;
515
516                 bn_check_top(a);
517                 bn_check_top(n);
518
519                 BN_CTX_start(ctx);
520                 A = BN_CTX_get(ctx);
521                 B = BN_CTX_get(ctx);
522                 X = BN_CTX_get(ctx);
523                 D = BN_CTX_get(ctx);
```

```
524         M = BN_CTX_get(ctx);
525         Y = BN_CTX_get(ctx);
526         T = BN_CTX_get(ctx);
527         if (T == NULL) goto err;

529         if (in == NULL)
530                 R=BN_new();
531         else
532                 R=in;
533         if (R == NULL) goto err;

535         BN_one(X);
536         BN_zero(Y);
537         if (BN_copy(B,a) == NULL) goto err;
538         if (BN_copy(A,n) == NULL) goto err;
539         A->neg = 0;

541         if (B->neg || (BN_ucmp(B, A) >= 0))
542                 {
543                 /* Turn BN_FLG_CONSTTIME flag on, so that when BN_div is invoked
544                  * BN_div_no_branch will be called eventually.
545                  */
546                 pB = &local_B;
547                 BN_with_flags(pB, B, BN_FLG_CONSTTIME);
548                 if (!BN_nnmod(B, pB, A, ctx)) goto err;
549                 }
550         sign = -1;
551         /* From  B = a mod |n|,  A = |n|  it follows that
552          *
553          *      0 <= B < A,
554          *     -sign*X*a  ==  B   (mod |n|),
555          *      sign*Y*a  ==  A   (mod |n|).
556          */

558         while (!BN_is_zero(B))
559                 {
560                 BIGNUM *tmp;

562                 /*
563                  *      0 < B < A,
564                  * (*) -sign*X*a  ==  B   (mod |n|),
565                  *      sign*Y*a  ==  A   (mod |n|)
566                  */

568                 /* Turn BN_FLG_CONSTTIME flag on, so that when BN_div is invoked
569                  * BN_div_no_branch will be called eventually.
570                  */
571                 pA = &local_A;
572                 BN_with_flags(pA, A, BN_FLG_CONSTTIME);

574                 /* (D, M) := (A/B, A%B) ... */
575                 if (!BN_div(D,M,pA,B,ctx)) goto err;

577                 /* Now
578                  *      A = D*B + M;
579                  * thus we have
580                  * (**)  sign*Y*a  ==  D*B + M   (mod |n|).
581                  */

583                 tmp=A; /* keep the BIGNUM object, the value does not matter */

585                 /* (A, B) := (B, A mod B) ... */
586                 A=B;
587                 B=M;
588                 /* ... so we have  0 <= B < A  again */
```



```
590                 /* Since the former  M  is now  B  and the former  B  is now  A,
591                  * (**) translates into
592                  *       sign*Y*a  ==  D*A + B    (mod |n|),
593                  * i.e.
594                  *       sign*Y*a - D*A  ==  B    (mod |n|).
595                  * Similarly, (*) translates into
596                  *      -sign*X*a  ==  A          (mod |n|).
597                  *
598                  * Thus,
599                  *   sign*Y*a + D*sign*X*a  ==  B  (mod |n|),
600                  * i.e.
601                  *        sign*(Y + D*X)*a  ==  B  (mod |n|).
602                  *
603                  * So if we set  (X, Y, sign) := (Y + D*X, X, -sign),  we arrive
604                  *      -sign*X*a  ==  B   (mod |n|),
605                  *       sign*Y*a  ==  A   (mod |n|).
606                  * Note that  X  and  Y  stay non-negative all the time.
607                  */

609                 if (!BN_mul(tmp,D,X,ctx)) goto err;
610                 if (!BN_add(tmp,tmp,Y)) goto err;

612                 M=Y; /* keep the BIGNUM object, the value does not matter */
613                 Y=X;
614                 X=tmp;
615                 sign = -sign;
616                 }

618         /*
619          * The while loop (Euclid's algorithm) ends when
620          *      A == gcd(a,n);
621          * we have
622          *       sign*Y*a  ==  A  (mod |n|),
623          * where  Y  is non-negative.
624          */

626         if (sign < 0)
627                 {
628                 if (!BN_sub(Y,n,Y)) goto err;
629                 }
630         /* Now  Y*a  ==  A  (mod |n|).  */

632         if (BN_is_one(A))
633                 {
634                 /* Y*a == 1  (mod |n|) */
635                 if (!Y->neg && BN_ucmp(Y,n) < 0)
636                         {
637                         if (!BN_copy(R,Y)) goto err;
638                         }
639                 else
640                         {
641                         if (!BN_nnmod(R,Y,n,ctx)) goto err;
642                         }
643                 }
644         else
645                 {
646                 BNerr(BN_F_BN_MOD_INVERSE_NO_BRANCH,BN_R_NO_INVERSE);
647                 goto err;
648                 }
649         ret=R;
650 err:
651         if ((ret == NULL) && (in == NULL)) BN_free(R);
652         BN_CTX_end(ctx);
653         bn_check_top(ret);
654         return(ret);
655         }
```

```
656 #endif /* ! codereview */
```

```
**********************************************************
   29612 Wed Aug 13 19:52:15 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_gf2m.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_gf2m.c */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * The Elliptic Curve Public-Key Crypto Library (ECC Code) included
 * herein is developed by SUN MICROSYSTEMS, INC., and is contributed
 * to the OpenSSL project.
 *
 * The ECC Code is licensed pursuant to the OpenSSL open source
 * license provided below.
 *
 * In addition, Sun covenants to all licensees who provide a reciprocal
 * covenant with respect to their own patents if any, not to sue under
 * current and future patent claims necessarily infringed by the making,
 * using, practicing, selling, offering for sale and/or otherwise
 * disposing of the ECC Code as delivered hereunder (or portions thereof),
 * provided that such covenant shall not apply:
 *  1) for code that a licensee deletes from the ECC Code;
 *  2) separates from the ECC Code; or
 *  3) for infringements caused by:
 *       i) the modification of the ECC Code or
 *      ii) the combination of the ECC Code with other software or
 *          devices where such combination causes the infringement.
 *
 * The software is originally written by Sheueling Chang Shantz and
 * Douglas Stebila of Sun Microsystems Laboratories.
 *
 */

/* NOTE: This file is licensed pursuant to the OpenSSL license below
 * and may be modified; but after modifications, the above covenant
 * may no longer apply!  In such cases, the corresponding paragraph
 * ["In addition, Sun covenants ... causes the infringement."] and
 * this note can be edited out; but please keep the Sun copyright
 * notice and attribution. */

/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
```



```
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <assert.h>
#include <limits.h>
#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"

#ifndef OPENSSL_NO_EC2M

/* Maximum number of iterations before BN_GF2m_mod_solve_quad_arr should fail. *
#define MAX_ITERATIONS 50

static const BN_ULONG SQR_tb[16] =
  {     0,     1,     4,     5,    16,    17,    20,    21,
       64,    65,    68,    69,    80,    81,    84,    85 };
/* Platform-specific macros to accelerate squaring. */
#if defined(SIXTY_FOUR_BIT) || defined(SIXTY_FOUR_BIT_LONG)
#define SQR1(w) \
    SQR_tb[(w) >> 60 & 0xF] << 56 | SQR_tb[(w) >> 56 & 0xF] << 48 | \
    SQR_tb[(w) >> 52 & 0xF] << 40 | SQR_tb[(w) >> 48 & 0xF] << 32 | \
    SQR_tb[(w) >> 44 & 0xF] << 24 | SQR_tb[(w) >> 40 & 0xF] << 16 | \
    SQR_tb[(w) >> 36 & 0xF] <<  8 | SQR_tb[(w) >> 32 & 0xF]
#define SQR0(w) \
    SQR_tb[(w) >> 28 & 0xF] << 56 | SQR_tb[(w) >> 24 & 0xF] << 48 | \
    SQR_tb[(w) >> 20 & 0xF] << 40 | SQR_tb[(w) >> 16 & 0xF] << 32 | \
    SQR_tb[(w) >> 12 & 0xF] << 24 | SQR_tb[(w) >>  8 & 0xF] << 16 | \
    SQR_tb[(w) >>  4 & 0xF] <<  8 | SQR_tb[(w)       & 0xF]
#endif
#ifdef THIRTY_TWO_BIT
#define SQR1(w) \
    SQR_tb[(w) >> 28 & 0xF] << 24 | SQR_tb[(w) >> 24 & 0xF] << 16 | \
    SQR_tb[(w) >> 20 & 0xF] <<  8 | SQR_tb[(w) >> 16 & 0xF]
#define SQR0(w) \
    SQR_tb[(w) >> 12 & 0xF] << 24 | SQR_tb[(w) >>  8 & 0xF] << 16 | \
    SQR_tb[(w) >>  4 & 0xF] <<  8 | SQR_tb[(w)       & 0xF]
#endif

#if !defined(OPENSSL_BN_ASM_GF2m)
```

```
/* Product of two polynomials a, b each with degree < BN_BITS2 - 1,
 * result is a polynomial r with degree < 2 * BN_BITS - 1
 * The caller MUST ensure that the variables have the right amount
 * of space allocated.
 */
#ifdef THIRTY_TWO_BIT
static void bn_GF2m_mul_1x1(BN_ULONG *r1, BN_ULONG *r0, const BN_ULONG a, const
        {
        register BN_ULONG h, l, s;
        BN_ULONG tab[8], top2b = a >> 30;
        register BN_ULONG a1, a2, a4;

        a1 = a & (0x3FFFFFFF); a2 = a1 << 1; a4 = a2 << 1;

        tab[0] =  0; tab[1] = a1;    tab[2] = a2;    tab[3] = a1^a2;
        tab[4] = a4; tab[5] = a1^a4; tab[6] = a2^a4; tab[7] = a1^a2^a4;

        s = tab[b       & 0x7]; l  = s;
        s = tab[b >>  3 & 0x7]; l ^= s <<  3; h  = s >> 29;
        s = tab[b >>  6 & 0x7]; l ^= s <<  6; h ^= s >> 26;
        s = tab[b >>  9 & 0x7]; l ^= s <<  9; h ^= s >> 23;
        s = tab[b >> 12 & 0x7]; l ^= s << 12; h ^= s >> 20;
        s = tab[b >> 15 & 0x7]; l ^= s << 15; h ^= s >> 17;
        s = tab[b >> 18 & 0x7]; l ^= s << 18; h ^= s >> 14;
        s = tab[b >> 21 & 0x7]; l ^= s << 21; h ^= s >> 11;
        s = tab[b >> 24 & 0x7]; l ^= s << 24; h ^= s >>  8;
        s = tab[b >> 27 & 0x7]; l ^= s << 27; h ^= s >>  5;
        s = tab[b >> 30      ]; l ^= s << 30; h ^= s >>  2;

        /* compensate for the top two bits of a */

        if (top2b & 01) { l ^= b << 30; h ^= b >> 2; }
        if (top2b & 02) { l ^= b << 31; h ^= b >> 1; }

        *r1 = h; *r0 = l;
        }
#endif
#if defined(SIXTY_FOUR_BIT) || defined(SIXTY_FOUR_BIT_LONG)
static void bn_GF2m_mul_1x1(BN_ULONG *r1, BN_ULONG *r0, const BN_ULONG a, const
        {
        register BN_ULONG h, l, s;
        BN_ULONG tab[16], top3b = a >> 61;
        register BN_ULONG a1, a2, a4, a8;

        a1 = a & (0x1FFFFFFFFFFFFFFFULL); a2 = a1 << 1; a4 = a2 << 1; a8 = a4 <<

        tab[ 0] = 0;     tab[ 1] = a1;       tab[ 2] = a2;       tab[ 3] = a1^a2
        tab[ 4] = a4;    tab[ 5] = a1^a4;    tab[ 6] = a2^a4;    tab[ 7] = a1^a2
        tab[ 8] = a8;    tab[ 9] = a1^a8;    tab[10] = a2^a8;    tab[11] = a1^a2
        tab[12] = a4^a8; tab[13] = a1^a4^a8; tab[14] = a2^a4^a8; tab[15] = a1^a2

        s = tab[b       & 0xF]; l  = s;
        s = tab[b >>  4 & 0xF]; l ^= s <<  4; h  = s >> 60;
        s = tab[b >>  8 & 0xF]; l ^= s <<  8; h ^= s >> 56;
        s = tab[b >> 12 & 0xF]; l ^= s << 12; h ^= s >> 52;
        s = tab[b >> 16 & 0xF]; l ^= s << 16; h ^= s >> 48;
        s = tab[b >> 20 & 0xF]; l ^= s << 20; h ^= s >> 44;
        s = tab[b >> 24 & 0xF]; l ^= s << 24; h ^= s >> 40;
        s = tab[b >> 28 & 0xF]; l ^= s << 28; h ^= s >> 36;
        s = tab[b >> 32 & 0xF]; l ^= s << 32; h ^= s >> 32;
        s = tab[b >> 36 & 0xF]; l ^= s << 36; h ^= s >> 28;
        s = tab[b >> 40 & 0xF]; l ^= s << 40; h ^= s >> 24;
        s = tab[b >> 44 & 0xF]; l ^= s << 44; h ^= s >> 20;
        s = tab[b >> 48 & 0xF]; l ^= s << 48; h ^= s >> 16;
        s = tab[b >> 52 & 0xF]; l ^= s << 52; h ^= s >> 12;
        s = tab[b >> 56 & 0xF]; l ^= s << 56; h ^= s >>  8;
```



```
        s = tab[b >> 60      ]; l ^= s << 60; h ^= s >>  4;

        /* compensate for the top three bits of a */

        if (top3b & 01) { l ^= b << 61; h ^= b >> 3; }
        if (top3b & 02) { l ^= b << 62; h ^= b >> 2; }
        if (top3b & 04) { l ^= b << 63; h ^= b >> 1; }

        *r1 = h; *r0 = l;
        }
#endif

/* Product of two polynomials a, b each with degree < 2 * BN_BITS2 - 1,
 * result is a polynomial r with degree < 4 * BN_BITS2 - 1
 * The caller MUST ensure that the variables have the right amount
 * of space allocated.
 */
static void bn_GF2m_mul_2x2(BN_ULONG *r, const BN_ULONG a1, const BN_ULONG a0, c
        {
        BN_ULONG m1, m0;
        /* r[3] = h1, r[2] = h0; r[1] = l1; r[0] = l0 */
        bn_GF2m_mul_1x1(r+3, r+2, a1, b1);
        bn_GF2m_mul_1x1(r+1, r, a0, b0);
        bn_GF2m_mul_1x1(&m1, &m0, a0 ^ a1, b0 ^ b1);
        /* Correction on m1 ^= l1 ^ h1; m0 ^= l0 ^ h0; */
        r[2] ^= m1 ^ r[1] ^ r[3];  /* h0 ^= m1 ^ l1 ^ h1; */
        r[1] = r[3] ^ r[2] ^ r[0] ^ m1 ^ m0;  /* l1 ^= l0 ^ h0 ^ m0; */
        }
#else
void bn_GF2m_mul_2x2(BN_ULONG *r, BN_ULONG a1, BN_ULONG a0, BN_ULONG b1, BN_ULON
#endif

/* Add polynomials a and b and store result in r; r could be a or b, a and b
 * could be equal; r is the bitwise XOR of a and b.
 */
int     BN_GF2m_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b)
        {
        int i;
        const BIGNUM *at, *bt;

        bn_check_top(a);
        bn_check_top(b);

        if (a->top < b->top) { at = b; bt = a; }
        else { at = a; bt = b; }

        if(bn_wexpand(r, at->top) == NULL)
                return 0;

        for (i = 0; i < bt->top; i++)
                {
                r->d[i] = at->d[i] ^ bt->d[i];
                }
        for (; i < at->top; i++)
                {
                r->d[i] = at->d[i];
                }

        r->top = at->top;
        bn_correct_top(r);

        return 1;
        }


/* Some functions allow for representation of the irreducible polynomials
```

```
 * as an int[], say p.  The irreducible f(t) is then of the form:
 *     t^p[0] + t^p[1] + ... + t^p[k]
 * where m = p[0] > p[1] > ... > p[k] = 0.
 */


/* Performs modular reduction of a and store result in r.  r could be a. */
int BN_GF2m_mod_arr(BIGNUM *r, const BIGNUM *a, const int p[])
        {
        int j, k;
        int n, dN, d0, d1;
        BN_ULONG zz, *z;

        bn_check_top(a);

        if (!p[0])
                {
                /* reduction mod 1 => return 0 */
                BN_zero(r);
                return 1;
                }

        /* Since the algorithm does reduction in the r value, if a != r, copy
         * the contents of a into r so we can do reduction in r.
         */
        if (a != r)
                {
                if (!bn_wexpand(r, a->top)) return 0;
                for (j = 0; j < a->top; j++)
                        {
                        r->d[j] = a->d[j];
                        }
                r->top = a->top;
                }
        z = r->d;

        /* start reduction */
        dN = p[0] / BN_BITS2;
        for (j = r->top - 1; j > dN;)
                {
                zz = z[j];
                if (z[j] == 0) { j--; continue; }
                z[j] = 0;

                for (k = 1; p[k] != 0; k++)
                        {
                        /* reducing component t^p[k] */
                        n = p[0] - p[k];
                        d0 = n % BN_BITS2;  d1 = BN_BITS2 - d0;
                        n /= BN_BITS2;
                        z[j-n] ^= (zz>>d0);
                        if (d0) z[j-n-1] ^= (zz<<d1);
                        }

                /* reducing component t^0 */
                n = dN;
                d0 = p[0] % BN_BITS2;
                d1 = BN_BITS2 - d0;
                z[j-n] ^= (zz >> d0);
                if (d0) z[j-n-1] ^= (zz << d1);
                }

        /* final round of reduction */
        while (j == dN)
                {
```



```
                d0 = p[0] % BN_BITS2;
                zz = z[dN] >> d0;
                if (zz == 0) break;
                d1 = BN_BITS2 - d0;

                /* clear up the top d1 bits */
                if (d0)
                        z[dN] = (z[dN] << d1) >> d1;
                else
                        z[dN] = 0;
                z[0] ^= zz; /* reduction t^0 component */

                for (k = 1; p[k] != 0; k++)
                        {
                        BN_ULONG tmp_ulong;

                        /* reducing component t^p[k]*/
                        n = p[k] / BN_BITS2;
                        d0 = p[k] % BN_BITS2;
                        d1 = BN_BITS2 - d0;
                        z[n] ^= (zz << d0);
                        tmp_ulong = zz >> d1;
                        if (d0 && tmp_ulong)
                                z[n+1] ^= tmp_ulong;
                        }


                }

        bn_correct_top(r);
        return 1;
        }

/* Performs modular reduction of a by p and store result in r.  r could be a.
 *
 * This function calls down to the BN_GF2m_mod_arr implementation; this wrapper
 * function is only provided for convenience; for best performance, use the
 * BN_GF2m_mod_arr function.
 */
int     BN_GF2m_mod(BIGNUM *r, const BIGNUM *a, const BIGNUM *p)
        {
        int ret = 0;
        int arr[6];
        bn_check_top(a);
        bn_check_top(p);
        ret = BN_GF2m_poly2arr(p, arr, sizeof(arr)/sizeof(arr[0]));
        if (!ret || ret > (int)(sizeof(arr)/sizeof(arr[0])))
                {
                BNerr(BN_F_BN_GF2M_MOD,BN_R_INVALID_LENGTH);
                return 0;
                }
        ret = BN_GF2m_mod_arr(r, a, arr);
        bn_check_top(r);
        return ret;
        }


/* Compute the product of two polynomials a and b, reduce modulo p, and store
 * the result in r.  r could be a or b; a could be b.
 */
int     BN_GF2m_mod_mul_arr(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const i
        {
        int zlen, i, j, k, ret = 0;
        BIGNUM *s;
        BN_ULONG x1, x0, y1, y0, zz[4];
```

```
392         bn_check_top(a);
393         bn_check_top(b);

395         if (a == b)
396                 {
397                 return BN_GF2m_mod_sqr_arr(r, a, p, ctx);
398                 }

400         BN_CTX_start(ctx);
401         if ((s = BN_CTX_get(ctx)) == NULL) goto err;

403         zlen = a->top + b->top + 4;
404         if (!bn_wexpand(s, zlen)) goto err;
405         s->top = zlen;

407         for (i = 0; i < zlen; i++) s->d[i] = 0;

409         for (j = 0; j < b->top; j += 2)
410                 {
411                 y0 = b->d[j];
412                 y1 = ((j+1) == b->top) ? 0 : b->d[j+1];
413                 for (i = 0; i < a->top; i += 2)
414                         {
415                         x0 = a->d[i];
416                         x1 = ((i+1) == a->top) ? 0 : a->d[i+1];
417                         bn_GF2m_mul_2x2(zz, x1, x0, y1, y0);
418                         for (k = 0; k < 4; k++) s->d[i+j+k] ^= zz[k];
419                         }
420                 }

422         bn_correct_top(s);
423         if (BN_GF2m_mod_arr(r, s, p))
424                 ret = 1;
425         bn_check_top(r);

427 err:
428         BN_CTX_end(ctx);
429         return ret;
430         }

432 /* Compute the product of two polynomials a and b, reduce modulo p, and store
433  * the result in r.  r could be a or b; a could equal b.
434  *
435  * This function calls down to the BN_GF2m_mod_mul_arr implementation; this wrap
436  * function is only provided for convenience; for best performance, use the
437  * BN_GF2m_mod_mul_arr function.
438  */
439 int     BN_GF2m_mod_mul(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNU
440         {
441         int ret = 0;
442         const int max = BN_num_bits(p) + 1;
443         int *arr=NULL;
444         bn_check_top(a);
445         bn_check_top(b);
446         bn_check_top(p);
447         if ((arr = (int *)OPENSSL_malloc(sizeof(int) * max)) == NULL) goto err;
448         ret = BN_GF2m_poly2arr(p, arr, max);
449         if (!ret || ret > max)
450                 {
451                 BNerr(BN_F_BN_GF2M_MOD_MUL,BN_R_INVALID_LENGTH);
452                 goto err;
453                 }
454         ret = BN_GF2m_mod_mul_arr(r, a, b, arr, ctx);
455         bn_check_top(r);
456 err:
457         if (arr) OPENSSL_free(arr);
```



```
458         return ret;
459         }


462 /* Square a, reduce the result mod p, and store it in a.  r could be a. */
463 int     BN_GF2m_mod_sqr_arr(BIGNUM *r, const BIGNUM *a, const int p[], BN_CTX *c
464         {
465         int i, ret = 0;
466         BIGNUM *s;

468         bn_check_top(a);
469         BN_CTX_start(ctx);
470         if ((s = BN_CTX_get(ctx)) == NULL) return 0;
471         if (!bn_wexpand(s, 2 * a->top)) goto err;

473         for (i = a->top - 1; i >= 0; i--)
474                 {
475                 s->d[2*i+1] = SQR1(a->d[i]);
476                 s->d[2*i  ] = SQR0(a->d[i]);
477                 }

479         s->top = 2 * a->top;
480         bn_correct_top(s);
481         if (!BN_GF2m_mod_arr(r, s, p)) goto err;
482         bn_check_top(r);
483         ret = 1;
484 err:
485         BN_CTX_end(ctx);
486         return ret;
487         }

489 /* Square a, reduce the result mod p, and store it in a.  r could be a.
490  *
491  * This function calls down to the BN_GF2m_mod_sqr_arr implementation; this wrap
492  * function is only provided for convenience; for best performance, use the
493  * BN_GF2m_mod_sqr_arr function.
494  */
495 int     BN_GF2m_mod_sqr(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx
496         {
497         int ret = 0;
498         const int max = BN_num_bits(p) + 1;
499         int *arr=NULL;

501         bn_check_top(a);
502         bn_check_top(p);
503         if ((arr = (int *)OPENSSL_malloc(sizeof(int) * max)) == NULL) goto err;
504         ret = BN_GF2m_poly2arr(p, arr, max);
505         if (!ret || ret > max)
506                 {
507                 BNerr(BN_F_BN_GF2M_MOD_SQR,BN_R_INVALID_LENGTH);
508                 goto err;
509                 }
510         ret = BN_GF2m_mod_sqr_arr(r, a, arr, ctx);
511         bn_check_top(r);
512 err:
513         if (arr) OPENSSL_free(arr);
514         return ret;
515         }


518 /* Invert a, reduce modulo p, and store the result in r. r could be a.
519  * Uses Modified Almost Inverse Algorithm (Algorithm 10) from
520  *     Hankerson, D., Hernandez, J.L., and Menezes, A.  "Software Implementation
521  *     of Elliptic Curve Cryptography Over Binary Fields".
522  */
523 int BN_GF2m_mod_inv(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx)
```

```
524         {
525         BIGNUM *b, *c = NULL, *u = NULL, *v = NULL, *tmp;
526         int ret = 0;

528         bn_check_top(a);
529         bn_check_top(p);

531         BN_CTX_start(ctx);

533         if ((b = BN_CTX_get(ctx))==NULL) goto err;
534         if ((c = BN_CTX_get(ctx))==NULL) goto err;
535         if ((u = BN_CTX_get(ctx))==NULL) goto err;
536         if ((v = BN_CTX_get(ctx))==NULL) goto err;

538         if (!BN_GF2m_mod(u, a, p)) goto err;
539         if (BN_is_zero(u)) goto err;

541         if (!BN_copy(v, p)) goto err;
542 #if 0
543         if (!BN_one(b)) goto err;

545         while (1)
546                 {
547                 while (!BN_is_odd(u))
548                         {
549                         if (BN_is_zero(u)) goto err;
550                         if (!BN_rshift1(u, u)) goto err;
551                         if (BN_is_odd(b))
552                                 {
553                                 if (!BN_GF2m_add(b, b, p)) goto err;
554                                 }
555                         if (!BN_rshift1(b, b)) goto err;
556                         }

558                 if (BN_abs_is_word(u, 1)) break;

560                 if (BN_num_bits(u) < BN_num_bits(v))
561                         {
562                         tmp = u; u = v; v = tmp;
563                         tmp = b; b = c; c = tmp;
564                         }

566                 if (!BN_GF2m_add(u, u, v)) goto err;
567                 if (!BN_GF2m_add(b, b, c)) goto err;
568                 }
569 #else
570         {
571         int i,  ubits = BN_num_bits(u),
572                 vbits = BN_num_bits(v), /* v is copy of p */
573                 top = p->top;
574         BN_ULONG *udp,*bdp,*vdp,*cdp;

576         bn_wexpand(u,top);      udp = u->d;
577                                 for (i=u->top;i<top;i++) udp[i] = 0;
578                                 u->top = top;
579         bn_wexpand(b,top);      bdp = b->d;
580                                 bdp[0] = 1;
581                                 for (i=1;i<top;i++) bdp[i] = 0;
582                                 b->top = top;
583         bn_wexpand(c,top);      cdp = c->d;
584                                 for (i=0;i<top;i++) cdp[i] = 0;
585                                 c->top = top;
586         vdp = v->d;     /* It pays off to "cache" *->d pointers, because
587                          * it allows optimizer to be more aggressive.
588                          * But we don't have to "cache" p->d, because *p
589                          * is declared 'const'... */
```



```
590         while (1)
591                 {
592                 while (ubits && !(udp[0]&1))
593                         {
594                         BN_ULONG u0,u1,b0,b1,mask;

596                         u0   = udp[0];
597                         b0   = bdp[0];
598                         mask = (BN_ULONG)0-(b0&1);
599                         b0  ^= p->d[0]&mask;
600                         for (i=0;i<top-1;i++)
601                                 {
602                                 u1 = udp[i+1];
603                                 udp[i] = ((u0>>1)|(u1<<(BN_BITS2-1)))&BN_MASK2;
604                                 u0 = u1;
605                                 b1 = bdp[i+1]^(p->d[i+1]&mask);
606                                 bdp[i] = ((b0>>1)|(b1<<(BN_BITS2-1)))&BN_MASK2;
607                                 b0 = b1;
608                                 }
609                         udp[i] = u0>>1;
610                         bdp[i] = b0>>1;
611                         ubits--;
612                         }

614                 if (ubits<=BN_BITS2 && udp[0]==1) break;

616                 if (ubits<vbits)
617                         {
618                         i = ubits; ubits = vbits; vbits = i;
619                         tmp = u; u = v; v = tmp;
620                         tmp = b; b = c; c = tmp;
621                         udp = vdp; vdp = v->d;
622                         bdp = cdp; cdp = c->d;
623                         }
624                 for(i=0;i<top;i++)
625                         {
626                         udp[i] ^= vdp[i];
627                         bdp[i] ^= cdp[i];
628                         }
629                 if (ubits==vbits)
630                         {
631                         BN_ULONG ul;
632                         int utop = (ubits-1)/BN_BITS2;

634                         while ((ul=udp[utop])==0 && utop) utop--;
635                         ubits = utop*BN_BITS2 + BN_num_bits_word(ul);
636                         }
637                 }
638         bn_correct_top(b);
639         }
640 #endif

642         if (!BN_copy(r, b)) goto err;
643         bn_check_top(r);
644         ret = 1;

646 err:
647 #ifdef BN_DEBUG /* BN_CTX_end would complain about the expanded form */
648         bn_correct_top(c);
649         bn_correct_top(u);
650         bn_correct_top(v);
651 #endif
652         BN_CTX_end(ctx);
653         return ret;
654         }
```

```
/* Invert xx, reduce modulo p, and store the result in r. r could be xx.
 *
 * This function calls down to the BN_GF2m_mod_inv implementation; this wrapper
 * function is only provided for convenience; for best performance, use the
 * BN_GF2m_mod_inv function.
 */
int BN_GF2m_mod_inv_arr(BIGNUM *r, const BIGNUM *xx, const int p[], BN_CTX *ctx)
        {
        BIGNUM *field;
        int ret = 0;

        bn_check_top(xx);
        BN_CTX_start(ctx);
        if ((field = BN_CTX_get(ctx)) == NULL) goto err;
        if (!BN_GF2m_arr2poly(p, field)) goto err;

        ret = BN_GF2m_mod_inv(r, xx, field, ctx);
        bn_check_top(r);

err:
        BN_CTX_end(ctx);
        return ret;
        }


#ifndef OPENSSL_SUN_GF2M_DIV
/* Divide y by x, reduce modulo p, and store the result in r. r could be x
 * or y, x could equal y.
 */
int BN_GF2m_mod_div(BIGNUM *r, const BIGNUM *y, const BIGNUM *x, const BIGNUM *p
        {
        BIGNUM *xinv = NULL;
        int ret = 0;

        bn_check_top(y);
        bn_check_top(x);
        bn_check_top(p);

        BN_CTX_start(ctx);
        xinv = BN_CTX_get(ctx);
        if (xinv == NULL) goto err;

        if (!BN_GF2m_mod_inv(xinv, x, p, ctx)) goto err;
        if (!BN_GF2m_mod_mul(r, y, xinv, p, ctx)) goto err;
        bn_check_top(r);
        ret = 1;

err:
        BN_CTX_end(ctx);
        return ret;
        }
#else
/* Divide y by x, reduce modulo p, and store the result in r. r could be x
 * or y, x could equal y.
 * Uses algorithm Modular_Division_GF(2^m) from
 *     Chang-Shantz, S.  "From Euclid's GCD to Montgomery Multiplication to
 *     the Great Divide".
 */
int BN_GF2m_mod_div(BIGNUM *r, const BIGNUM *y, const BIGNUM *x, const BIGNUM *p
        {
        BIGNUM *a, *b, *u, *v;
        int ret = 0;

        bn_check_top(y);
        bn_check_top(x);
        bn_check_top(p);
```



```
        BN_CTX_start(ctx);

        a = BN_CTX_get(ctx);
        b = BN_CTX_get(ctx);
        u = BN_CTX_get(ctx);
        v = BN_CTX_get(ctx);
        if (v == NULL) goto err;

        /* reduce x and y mod p */
        if (!BN_GF2m_mod(u, y, p)) goto err;
        if (!BN_GF2m_mod(a, x, p)) goto err;
        if (!BN_copy(b, p)) goto err;

        while (!BN_is_odd(a))
                {
                if (!BN_rshift1(a, a)) goto err;
                if (BN_is_odd(u)) if (!BN_GF2m_add(u, u, p)) goto err;
                if (!BN_rshift1(u, u)) goto err;
                }

        do
                {
                if (BN_GF2m_cmp(b, a) > 0)
                        {
                        if (!BN_GF2m_add(b, b, a)) goto err;
                        if (!BN_GF2m_add(v, v, u)) goto err;
                        do
                                {
                                if (!BN_rshift1(b, b)) goto err;
                                if (BN_is_odd(v)) if (!BN_GF2m_add(v, v, p)) got
                                if (!BN_rshift1(v, v)) goto err;
                                } while (!BN_is_odd(b));
                        }
                else if (BN_abs_is_word(a, 1))
                        break;
                else
                        {
                        if (!BN_GF2m_add(a, a, b)) goto err;
                        if (!BN_GF2m_add(u, u, v)) goto err;
                        do
                                {
                                if (!BN_rshift1(a, a)) goto err;
                                if (BN_is_odd(u)) if (!BN_GF2m_add(u, u, p)) got
                                if (!BN_rshift1(u, u)) goto err;
                                } while (!BN_is_odd(a));
                        }
                } while (1);

        if (!BN_copy(r, u)) goto err;
        bn_check_top(r);
        ret = 1;

err:
        BN_CTX_end(ctx);
        return ret;
        }
#endif

/* Divide yy by xx, reduce modulo p, and store the result in r. r could be xx
 * or yy, xx could equal yy.
 *
 * This function calls down to the BN_GF2m_mod_div implementation; this wrapper
 * function is only provided for convenience; for best performance, use the
 * BN_GF2m_mod_div function.
 */
```

```
int BN_GF2m_mod_div_arr(BIGNUM *r, const BIGNUM *yy, const BIGNUM *xx, const int
        {
        BIGNUM *field;
        int ret = 0;

        bn_check_top(yy);
        bn_check_top(xx);

        BN_CTX_start(ctx);
        if ((field = BN_CTX_get(ctx)) == NULL) goto err;
        if (!BN_GF2m_arr2poly(p, field)) goto err;

        ret = BN_GF2m_mod_div(r, yy, xx, field, ctx);
        bn_check_top(r);

err:
        BN_CTX_end(ctx);
        return ret;
        }


/* Compute the bth power of a, reduce modulo p, and store
 * the result in r.  r could be a.
 * Uses simple square-and-multiply algorithm A.5.1 from IEEE P1363.
 */
int     BN_GF2m_mod_exp_arr(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const i
        {
        int ret = 0, i, n;
        BIGNUM *u;

        bn_check_top(a);
        bn_check_top(b);

        if (BN_is_zero(b))
                return(BN_one(r));

        if (BN_abs_is_word(b, 1))
                return (BN_copy(r, a) != NULL);

        BN_CTX_start(ctx);
        if ((u = BN_CTX_get(ctx)) == NULL) goto err;

        if (!BN_GF2m_mod_arr(u, a, p)) goto err;

        n = BN_num_bits(b) - 1;
        for (i = n - 1; i >= 0; i--)
                {
                if (!BN_GF2m_mod_sqr_arr(u, u, p, ctx)) goto err;
                if (BN_is_bit_set(b, i))
                        {
                        if (!BN_GF2m_mod_mul_arr(u, u, a, p, ctx)) goto err;
                        }
                }
        if (!BN_copy(r, u)) goto err;
        bn_check_top(r);
        ret = 1;
err:
        BN_CTX_end(ctx);
        return ret;
        }

/* Compute the bth power of a, reduce modulo p, and store
 * the result in r.  r could be a.
 *
 * This function calls down to the BN_GF2m_mod_exp_arr implementation; this wrap
 * function is only provided for convenience; for best performance, use the
```



```
 * BN_GF2m_mod_exp_arr function.
 */
int BN_GF2m_mod_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *p
        {
        int ret = 0;
        const int max = BN_num_bits(p) + 1;
        int *arr=NULL;
        bn_check_top(a);
        bn_check_top(b);
        bn_check_top(p);
        if ((arr = (int *)OPENSSL_malloc(sizeof(int) * max)) == NULL) goto err;
        ret = BN_GF2m_poly2arr(p, arr, max);
        if (!ret || ret > max)
                {
                BNerr(BN_F_BN_GF2M_MOD_EXP,BN_R_INVALID_LENGTH);
                goto err;
                }
        ret = BN_GF2m_mod_exp_arr(r, a, b, arr, ctx);
        bn_check_top(r);
err:
        if (arr) OPENSSL_free(arr);
        return ret;
        }

/* Compute the square root of a, reduce modulo p, and store
 * the result in r.  r could be a.
 * Uses exponentiation as in algorithm A.4.1 from IEEE P1363.
 */
int     BN_GF2m_mod_sqrt_arr(BIGNUM *r, const BIGNUM *a, const int p[], BN_CTX *
        {
        int ret = 0;
        BIGNUM *u;

        bn_check_top(a);

        if (!p[0])
                {
                /* reduction mod 1 => return 0 */
                BN_zero(r);
                return 1;
                }

        BN_CTX_start(ctx);
        if ((u = BN_CTX_get(ctx)) == NULL) goto err;

        if (!BN_set_bit(u, p[0] - 1)) goto err;
        ret = BN_GF2m_mod_exp_arr(r, a, u, p, ctx);
        bn_check_top(r);

err:
        BN_CTX_end(ctx);
        return ret;
        }

/* Compute the square root of a, reduce modulo p, and store
 * the result in r.  r could be a.
 *
 * This function calls down to the BN_GF2m_mod_sqrt_arr implementation; this wra
 * function is only provided for convenience; for best performance, use the
 * BN_GF2m_mod_sqrt_arr function.
 */
int BN_GF2m_mod_sqrt(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx)
        {
        int ret = 0;
        const int max = BN_num_bits(p) + 1;
        int *arr=NULL;
```

```
920         bn_check_top(a);
921         bn_check_top(p);
922         if ((arr = (int *)OPENSSL_malloc(sizeof(int) * max)) == NULL) goto err;
923         ret = BN_GF2m_poly2arr(p, arr, max);
924         if (!ret || ret > max)
925                 {
926                 BNerr(BN_F_BN_GF2M_MOD_SQRT,BN_R_INVALID_LENGTH);
927                 goto err;
928                 }
929         ret = BN_GF2m_mod_sqrt_arr(r, a, arr, ctx);
930         bn_check_top(r);
931 err:
932         if (arr) OPENSSL_free(arr);
933         return ret;
934         }

936 /* Find r such that r^2 + r = a mod p.  r could be a. If no r exists returns 0.
937  * Uses algorithms A.4.7 and A.4.6 from IEEE P1363.
938  */
939 int BN_GF2m_mod_solve_quad_arr(BIGNUM *r, const BIGNUM *a_, const int p[], BN_CT
940         {
941         int ret = 0, count = 0, j;
942         BIGNUM *a, *z, *rho, *w, *w2, *tmp;

944         bn_check_top(a_);

946         if (!p[0])
947                 {
948                 /* reduction mod 1 => return 0 */
949                 BN_zero(r);
950                 return 1;
951                 }

953         BN_CTX_start(ctx);
954         a = BN_CTX_get(ctx);
955         z = BN_CTX_get(ctx);
956         w = BN_CTX_get(ctx);
957         if (w == NULL) goto err;

959         if (!BN_GF2m_mod_arr(a, a_, p)) goto err;

961         if (BN_is_zero(a))
962                 {
963                 BN_zero(r);
964                 ret = 1;
965                 goto err;
966                 }

968         if (p[0] & 0x1) /* m is odd */
969                 {
970                 /* compute half-trace of a */
971                 if (!BN_copy(z, a)) goto err;
972                 for (j = 1; j <= (p[0] - 1) / 2; j++)
973                         {
974                         if (!BN_GF2m_mod_sqr_arr(z, z, p, ctx)) goto err;
975                         if (!BN_GF2m_mod_sqr_arr(z, z, p, ctx)) goto err;
976                         if (!BN_GF2m_add(z, z, a)) goto err;
977                         }

979                 }
980         else /* m is even */
981                 {
982                 rho = BN_CTX_get(ctx);
983                 w2 = BN_CTX_get(ctx);
984                 tmp = BN_CTX_get(ctx);
985                 if (tmp == NULL) goto err;
```



```
986                 do
987                         {
988                         if (!BN_rand(rho, p[0], 0, 0)) goto err;
989                         if (!BN_GF2m_mod_arr(rho, rho, p)) goto err;
990                         BN_zero(z);
991                         if (!BN_copy(w, rho)) goto err;
992                         for (j = 1; j <= p[0] - 1; j++)
993                                 {
994                                 if (!BN_GF2m_mod_sqr_arr(z, z, p, ctx)) goto err
995                                 if (!BN_GF2m_mod_sqr_arr(w2, w, p, ctx)) goto er
996                                 if (!BN_GF2m_mod_mul_arr(tmp, w2, a, p, ctx)) go
997                                 if (!BN_GF2m_add(z, z, tmp)) goto err;
998                                 if (!BN_GF2m_add(w, w2, rho)) goto err;
999                                 }
1000                        count++;
1001                        } while (BN_is_zero(w) && (count < MAX_ITERATIONS));
1002                if (BN_is_zero(w))
1003                        {
1004                        BNerr(BN_F_BN_GF2M_MOD_SOLVE_QUAD_ARR,BN_R_TOO_MANY_ITER
1005                        goto err;
1006                        }
1007                }

1009        if (!BN_GF2m_mod_sqr_arr(w, z, p, ctx)) goto err;
1010        if (!BN_GF2m_add(w, z, w)) goto err;
1011        if (BN_GF2m_cmp(w, a))
1012                {
1013                BNerr(BN_F_BN_GF2M_MOD_SOLVE_QUAD_ARR, BN_R_NO_SOLUTION);
1014                goto err;
1015                }

1017        if (!BN_copy(r, z)) goto err;
1018        bn_check_top(r);

1020        ret = 1;

1022 err:
1023        BN_CTX_end(ctx);
1024        return ret;
1025        }

1027 /* Find r such that r^2 + r = a mod p.  r could be a. If no r exists returns 0.
1028  *
1029  * This function calls down to the BN_GF2m_mod_solve_quad_arr implementation; th
1030  * function is only provided for convenience; for best performance, use the
1031  * BN_GF2m_mod_solve_quad_arr function.
1032  */
1033 int BN_GF2m_mod_solve_quad(BIGNUM *r, const BIGNUM *a, const BIGNUM *p, BN_CTX *
1034        {
1035        int ret = 0;
1036        const int max = BN_num_bits(p) + 1;
1037        int *arr=NULL;
1038        bn_check_top(a);
1039        bn_check_top(p);
1040        if ((arr = (int *)OPENSSL_malloc(sizeof(int) *
1041                                                max)) == NULL) goto err;
1042        ret = BN_GF2m_poly2arr(p, arr, max);
1043        if (!ret || ret > max)
1044                {
1045                BNerr(BN_F_BN_GF2M_MOD_SOLVE_QUAD,BN_R_INVALID_LENGTH);
1046                goto err;
1047                }
1048        ret = BN_GF2m_mod_solve_quad_arr(r, a, arr, ctx);
1049        bn_check_top(r);
1050 err:
1051        if (arr) OPENSSL_free(arr);
```

```
1052         return ret;
1053         }

1055 /* Convert the bit-string representation of a polynomial
1056  * ( \sum_{i=0}^n a_i * x^i) into an array of integers corresponding
1057  * to the bits with non-zero coefficient.  Array is terminated with -1.
1058  * Up to max elements of the array will be filled.  Return value is total
1059  * number of array elements that would be filled if array was large enough.
1060  */
1061 int BN_GF2m_poly2arr(const BIGNUM *a, int p[], int max)
1062         {
1063         int i, j, k = 0;
1064         BN_ULONG mask;

1066         if (BN_is_zero(a))
1067                 return 0;

1069         for (i = a->top - 1; i >= 0; i--)
1070                 {
1071                 if (!a->d[i])
1072                         /* skip word if a->d[i] == 0 */
1073                         continue;
1074                 mask = BN_TBIT;
1075                 for (j = BN_BITS2 - 1; j >= 0; j--)
1076                         {
1077                         if (a->d[i] & mask)
1078                                 {
1079                                 if (k < max) p[k] = BN_BITS2 * i + j;
1080                                 k++;
1081                                 }
1082                         mask >>= 1;
1083                         }
1084                 }

1086         if (k < max) {
1087                 p[k] = -1;
1088                 k++;
1089         }

1091         return k;
1092         }

1094 /* Convert the coefficient array representation of a polynomial to a
1095  * bit-string.  The array must be terminated by -1.
1096  */
1097 int BN_GF2m_arr2poly(const int p[], BIGNUM *a)
1098         {
1099         int i;

1101         bn_check_top(a);
1102         BN_zero(a);
1103         for (i = 0; p[i] != -1; i++)
1104                 {
1105                 if (BN_set_bit(a, p[i]) == 0)
1106                         return 0;
1107                 }
1108         bn_check_top(a);

1110         return 1;
1111         }

1113 #endif
1114 #endif /* ! codereview */
```

```
**********************************************************
    5111 Wed Aug 13 19:52:15 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_kron.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_kron.c */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include "bn_lcl.h"

/* least significant word */
#define BN_lsw(n) (((n)->top == 0) ? (BN_ULONG) 0 : (n)->d[0])
```



```
/* Returns -2 for errors because both -1 and 0 are valid results. */
int BN_kronecker(const BIGNUM *a, const BIGNUM *b, BN_CTX *ctx)
        {
        int i;
        int ret = -2; /* avoid 'uninitialized' warning */
        int err = 0;
        BIGNUM *A, *B, *tmp;
        /* In 'tab', only odd-indexed entries are relevant:
         * For any odd BIGNUM n,
         *     tab[BN_lsw(n) & 7]
         * is $(-1)^{(n^2-1)/8}$ (using TeX notation).
         * Note that the sign of n does not matter.
         */
        static const int tab[8] = {0, 1, 0, -1, 0, -1, 0, 1};

        bn_check_top(a);
        bn_check_top(b);

        BN_CTX_start(ctx);
        A = BN_CTX_get(ctx);
        B = BN_CTX_get(ctx);
        if (B == NULL) goto end;

        err = !BN_copy(A, a);
        if (err) goto end;
        err = !BN_copy(B, b);
        if (err) goto end;

        /*
         * Kronecker symbol, imlemented according to Henri Cohen,
         * "A Course in Computational Algebraic Number Theory"
         * (algorithm 1.4.10).
         */

        /* Cohen's step 1: */

        if (BN_is_zero(B))
                {
                ret = BN_abs_is_word(A, 1);
                goto end;
                }

        /* Cohen's step 2: */

        if (!BN_is_odd(A) && !BN_is_odd(B))
                {
                ret = 0;
                goto end;
                }

        /* now  B  is non-zero */
        i = 0;
        while (!BN_is_bit_set(B, i))
                i++;
        err = !BN_rshift(B, B, i);
        if (err) goto end;
        if (i & 1)
                {
                /* i is odd */
                /* (thus  B  was even, thus  A  must be odd!)  */

                /* set 'ret' to $(-1)^{(A^2-1)/8}$ */
                ret = tab[BN_lsw(A) & 7];
                }
        else
                {
```

```
128                 /* i is even */
129                 ret = 1;
130                 }
131
132         if (B->neg)
133                 {
134                 B->neg = 0;
135                 if (A->neg)
136                         ret = -ret;
137                 }
138
139         /* now  B  is positive and odd, so what remains to be done is
140          * to compute the Jacobi symbol  (A/B)  and multiply it by 'ret' */
141
142         while (1)
143                 {
144                 /* Cohen's step 3: */
145
146                 /*  B  is positive and odd */
147
148                 if (BN_is_zero(A))
149                         {
150                         ret = BN_is_one(B) ? ret : 0;
151                         goto end;
152                         }
153
154                 /* now  A  is non-zero */
155                 i = 0;
156                 while (!BN_is_bit_set(A, i))
157                         i++;
158                 err = !BN_rshift(A, A, i);
159                 if (err) goto end;
160                 if (i & 1)
161                         {
162                         /* i is odd */
163                         /* multiply 'ret' by  $(-1)^{(B^2-1)/8}$ */
164                         ret = ret * tab[BN_lsw(B) & 7];
165                         }
166
167                 /* Cohen's step 4: */
168                 /* multiply 'ret' by  $(-1)^{(A-1)(B-1)/4}$ */
169                 if ((A->neg ? ~BN_lsw(A) : BN_lsw(A)) & BN_lsw(B) & 2)
170                         ret = -ret;
171
172                 /* (A, B) := (B mod |A|, |A|) */
173                 err = !BN_nnmod(B, B, A, ctx);
174                 if (err) goto end;
175                 tmp = A; A = B; B = tmp;
176                 tmp->neg = 0;
177                 }
178 end:
179         BN_CTX_end(ctx);
180         if (err)
181                 return -2;
182         else
183                 return ret;
184         }
185 #endif /* ! codereview */
```

```
**********************************************************
   19839 Wed Aug 13 19:52:15 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef BN_DEBUG
# undef NDEBUG /* avoid conflicting definitions */
# define NDEBUG
```



```
#endif

#include <assert.h>
#include <limits.h>
#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"

const char BN_version[]="Big Number" OPENSSL_VERSION_PTEXT;

/* This stuff appears to be completely unused, so is deprecated */
#ifndef OPENSSL_NO_DEPRECATED
/* For a 32 bit machine
 * 2 -   4 ==  128
 * 3 -   8 ==  256
 * 4 -  16 ==  512
 * 5 -  32 == 1024
 * 6 -  64 == 2048
 * 7 - 128 == 4096
 * 8 - 256 == 8192
 */
static int bn_limit_bits=0;
static int bn_limit_num=8;        /* (1<<bn_limit_bits) */
static int bn_limit_bits_low=0;
static int bn_limit_num_low=8;    /* (1<<bn_limit_bits_low) */
static int bn_limit_bits_high=0;
static int bn_limit_num_high=8;   /* (1<<bn_limit_bits_high) */
static int bn_limit_bits_mont=0;
static int bn_limit_num_mont=8;   /* (1<<bn_limit_bits_mont) */

void BN_set_params(int mult, int high, int low, int mont)
        {
        if (mult >= 0)
                {
                if (mult > (int)(sizeof(int)*8)-1)
                        mult=sizeof(int)*8-1;
                bn_limit_bits=mult;
                bn_limit_num=1<<mult;
                }
        if (high >= 0)
                {
                if (high > (int)(sizeof(int)*8)-1)
                        high=sizeof(int)*8-1;
                bn_limit_bits_high=high;
                bn_limit_num_high=1<<high;
                }
        if (low >= 0)
                {
                if (low > (int)(sizeof(int)*8)-1)
                        low=sizeof(int)*8-1;
                bn_limit_bits_low=low;
                bn_limit_num_low=1<<low;
                }
        if (mont >= 0)
                {
                if (mont > (int)(sizeof(int)*8)-1)
                        mont=sizeof(int)*8-1;
                bn_limit_bits_mont=mont;
                bn_limit_num_mont=1<<mont;
                }
        }

int BN_get_params(int which)
        {
        if      (which == 0) return(bn_limit_bits);
        else if (which == 1) return(bn_limit_bits_high);
```

```
128         else if (which == 2) return(bn_limit_bits_low);
129         else if (which == 3) return(bn_limit_bits_mont);
130         else return(0);
131         }
132 #endif

133 const BIGNUM *BN_value_one(void)
134         {
135         static const BN_ULONG data_one=1L;
136         static const BIGNUM const_one={(BN_ULONG *)&data_one,1,1,0,BN_FLG_STATIC

137         return(&const_one);
138         }

139 int BN_num_bits_word(BN_ULONG l)
140         {
141         static const unsigned char bits[256]={
142                 0,1,2,2,3,3,3,3,4,4,4,4,4,4,4,4,
143                 5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,
144                 6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
145                 6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6,
146                 7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,
147                 7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,
148                 7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,
149                 7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,
150                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
151                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
152                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
153                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
154                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
155                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
156                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
157                 8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,
158                 };

159 #if defined(SIXTY_FOUR_BIT_LONG)
160         if (l & 0xffffffff00000000L)
161                 {
162                 if (l & 0xffff000000000000L)
163                         {
164                         if (l & 0xff00000000000000L)
165                                 {
166                                 return(bits[(int)(l>>56)]+56);
167                                 }
168                         else    return(bits[(int)(l>>48)]+48);
169                         }
170                 else
171                         {
172                         if (l & 0x0000ff0000000000L)
173                                 {
174                                 return(bits[(int)(l>>40)]+40);
175                                 }
176                         else    return(bits[(int)(l>>32)]+32);
177                         }
178                 }
179         else
180 #else
181 #ifdef SIXTY_FOUR_BIT
182         if (l & 0xffffffff00000000LL)
183                 {
184                 if (l & 0xffff000000000000LL)
185                         {
186                         if (l & 0xff00000000000000LL)
187                                 {
188                                 return(bits[(int)(l>>56)]+56);
189                                 }
190 
```



```
191                         else    return(bits[(int)(l>>48)]+48);
192                         }
193                 else
194                         {
195                         if (l & 0x0000ff0000000000LL)
196                                 {
197                                 return(bits[(int)(l>>40)]+40);
198                                 }
199                         else    return(bits[(int)(l>>32)]+32);
200                         }
201                 }
202         else
203 #endif
204 #endif
205                 {
206 #if defined(THIRTY_TWO_BIT) || defined(SIXTY_FOUR_BIT) || defined(SIXTY_FOUR_BIT
207                 if (l & 0xffff0000L)
208                         {
209                         if (l & 0xff000000L)
210                                 return(bits[(int)(l>>24L)]+24);
211                         else    return(bits[(int)(l>>16L)]+16);
212                         }
213                 else
214 #endif
215                         {
216 #if defined(THIRTY_TWO_BIT) || defined(SIXTY_FOUR_BIT) || defined(SIXTY_FOUR_BIT
217                         if (l & 0xff00L)
218                                 return(bits[(int)(l>>8)]+8);
219                         else
220 #endif
221                                 return(bits[(int)(l   )]  );
222                         }
223                 }
224         }

225 int BN_num_bits(const BIGNUM *a)
226         {
227         int i = a->top - 1;
228         bn_check_top(a);

229         if (BN_is_zero(a)) return 0;
230         return ((i*BN_BITS2) + BN_num_bits_word(a->d[i]));
231         }

232 void BN_clear_free(BIGNUM *a)
233         {
234         int i;

235         if (a == NULL) return;
236         bn_check_top(a);
237         if (a->d != NULL)
238                 {
239                 OPENSSL_cleanse(a->d,a->dmax*sizeof(a->d[0]));
240                 if (!(BN_get_flags(a,BN_FLG_STATIC_DATA)))
241                         OPENSSL_free(a->d);
242                 }
243         i=BN_get_flags(a,BN_FLG_MALLOCED);
244         OPENSSL_cleanse(a,sizeof(BIGNUM));
245         if (i)
246                 OPENSSL_free(a);
247         }

248 void BN_free(BIGNUM *a)
249         {
250         if (a == NULL) return;
251         bn_check_top(a);
```

```
260         if ((a->d != NULL) && !(BN_get_flags(a,BN_FLG_STATIC_DATA)))
261                 OPENSSL_free(a->d);
262         if (a->flags & BN_FLG_MALLOCED)
263                 OPENSSL_free(a);
264         else
265                 {
266 #ifndef OPENSSL_NO_DEPRECATED
267                 a->flags|=BN_FLG_FREE;
268 #endif
269                 a->d = NULL;
270                 }
271         }

273 void BN_init(BIGNUM *a)
274         {
275         memset(a,0,sizeof(BIGNUM));
276         bn_check_top(a);
277         }

279 BIGNUM *BN_new(void)
280         {
281         BIGNUM *ret;

283         if ((ret=(BIGNUM *)OPENSSL_malloc(sizeof(BIGNUM))) == NULL)
284                 {
285                 BNerr(BN_F_BN_NEW,ERR_R_MALLOC_FAILURE);
286                 return(NULL);
287                 }
288         ret->flags=BN_FLG_MALLOCED;
289         ret->top=0;
290         ret->neg=0;
291         ret->dmax=0;
292         ret->d=NULL;
293         bn_check_top(ret);
294         return(ret);
295         }

297 /* This is used both by bn_expand2() and bn_dup_expand() */
298 /* The caller MUST check that words > b->dmax before calling this */
299 static BN_ULONG *bn_expand_internal(const BIGNUM *b, int words)
300         {
301         BN_ULONG *A,*a = NULL;
302         const BN_ULONG *B;
303         int i;

305         bn_check_top(b);

307         if (words > (INT_MAX/(4*BN_BITS2)))
308                 {
309                 BNerr(BN_F_BN_EXPAND_INTERNAL,BN_R_BIGNUM_TOO_LONG);
310                 return NULL;
311                 }
312         if (BN_get_flags(b,BN_FLG_STATIC_DATA))
313                 {
314                 BNerr(BN_F_BN_EXPAND_INTERNAL,BN_R_EXPAND_ON_STATIC_BIGNUM_DATA);
315                 return(NULL);
316                 }
317         a=A=(BN_ULONG *)OPENSSL_malloc(sizeof(BN_ULONG)*words);
318         if (A == NULL)
319                 {
320                 BNerr(BN_F_BN_EXPAND_INTERNAL,ERR_R_MALLOC_FAILURE);
321                 return(NULL);
322                 }
323 #ifdef PURIFY
324         /* Valgrind complains in BN_consttime_swap because we process the whole
325          * array even if it's not initialised yet. This doesn't matter in that
```



```
326          * function - what's important is constant time operation (we're not
327          * actually going to use the data)
328         */
329         memset(a, 0, sizeof(BN_ULONG)*words);
330 #endif

332 #if 1
333         B=b->d;
334         /* Check if the previous number needs to be copied */
335         if (B != NULL)
336                 {
337                 for (i=b->top>>2; i>0; i--,A+=4,B+=4)
338                         {
339                         /*
340                          * The fact that the loop is unrolled
341                          * 4-wise is a tribute to Intel. It's
342                          * the one that doesn't have enough
343                          * registers to accomodate more data.
344                          * I'd unroll it 8-wise otherwise:-)
345                          *
346                          *              <appro@fy.chalmers.se>
347                          */
348                         BN_ULONG a0,a1,a2,a3;
349                         a0=B[0]; a1=B[1]; a2=B[2]; a3=B[3];
350                         A[0]=a0; A[1]=a1; A[2]=a2; A[3]=a3;
351                         }
352                 switch (b->top&3)
353                         {
354                 case 3: A[2]=B[2];
355                 case 2: A[1]=B[1];
356                 case 1: A[0]=B[0];
357                 case 0: /* workaround for ultrix cc: without 'case 0', the optim
358                          * the switch table by doing a=top&3; a--; goto jump_tab
359                          * which fails for top== 0 */
360                         ;
361                         }
362                 }

364 #else
365         memset(A,0,sizeof(BN_ULONG)*words);
366         memcpy(A,b->d,sizeof(b->d[0])*b->top);
367 #endif

369         return(a);
370         }

372 /* This is an internal function that can be used instead of bn_expand2()
373  * when there is a need to copy BIGNUMs instead of only expanding the
374  * data part, while still expanding them.
375  * Especially useful when needing to expand BIGNUMs that are declared
376  * 'const' and should therefore not be changed.
377  * The reason to use this instead of a BN_dup() followed by a bn_expand2()
378  * is memory allocation overhead.  A BN_dup() followed by a bn_expand2()
379  * will allocate new memory for the BIGNUM data twice, and free it once,
380  * while bn_dup_expand() makes sure allocation is made only once.
381  */

383 #ifndef OPENSSL_NO_DEPRECATED
384 BIGNUM *bn_dup_expand(const BIGNUM *b, int words)
385         {
386         BIGNUM *r = NULL;

388         bn_check_top(b);

390         /* This function does not work if
391          *      words <= b->dmax && top < words
```

```
392          * because BN_dup() does not preserve 'dmax'!
393          * (But bn_dup_expand() is not used anywhere yet.)
394          */

396         if (words > b->dmax)
397                 {
398                 BN_ULONG *a = bn_expand_internal(b, words);

400                 if (a)
401                         {
402                         r = BN_new();
403                         if (r)
404                                 {
405                                 r->top = b->top;
406                                 r->dmax = words;
407                                 r->neg = b->neg;
408                                 r->d = a;
409                                 }
410                         else
411                                 {
412                                 /* r == NULL, BN_new failure */
413                                 OPENSSL_free(a);
414                                 }
415                         }
416                 /* If a == NULL, there was an error in allocation in
417                    bn_expand_internal(), and NULL should be returned */
418                 }
419         else
420                 {
421                 r = BN_dup(b);
422                 }

424         bn_check_top(r);
425         return r;
426         }
427 #endif

429 /* This is an internal function that should not be used in applications.
430  * It ensures that 'b' has enough room for a 'words' word number
431  * and initialises any unused part of b->d with leading zeros.
432  * It is mostly used by the various BIGNUM routines. If there is an error,
433  * NULL is returned. If not, 'b' is returned. */

435 BIGNUM *bn_expand2(BIGNUM *b, int words)
436         {
437         bn_check_top(b);

439         if (words > b->dmax)
440                 {
441                 BN_ULONG *a = bn_expand_internal(b, words);
442                 if(!a) return NULL;
443                 if(b->d) OPENSSL_free(b->d);
444                 b->d=a;
445                 b->dmax=words;
446                 }

448 /* None of this should be necessary because of what b->top means! */
449 #if 0
450         /* NB: bn_wexpand() calls this only if the BIGNUM really has to grow */
451         if (b->top < b->dmax)
452                 {
453                 int i;
454                 BN_ULONG *A = &(b->d[b->top]);
455                 for (i=(b->dmax - b->top)>>3; i>0; i--,A+=8)
456                         {
457                         A[0]=0; A[1]=0; A[2]=0; A[3]=0;
```



```
458                         A[4]=0; A[5]=0; A[6]=0; A[7]=0;
459                         }
460                 for (i=(b->dmax - b->top)&7; i>0; i--,A++)
461                         A[0]=0;
462                 assert(A == &(b->d[b->dmax]));
463                 }
464 #endif
465         bn_check_top(b);
466         return b;
467         }

469 BIGNUM *BN_dup(const BIGNUM *a)
470         {
471         BIGNUM *t;

473         if (a == NULL) return NULL;
474         bn_check_top(a);

476         t = BN_new();
477         if (t == NULL) return NULL;
478         if(!BN_copy(t, a))
479                 {
480                 BN_free(t);
481                 return NULL;
482                 }
483         bn_check_top(t);
484         return t;
485         }

487 BIGNUM *BN_copy(BIGNUM *a, const BIGNUM *b)
488         {
489         int i;
490         BN_ULONG *A;
491         const BN_ULONG *B;

493         bn_check_top(b);

495         if (a == b) return(a);
496         if (bn_wexpand(a,b->top) == NULL) return(NULL);

498 #if 1
499         A=a->d;
500         B=b->d;
501         for (i=b->top>>2; i>0; i--,A+=4,B+=4)
502                 {
503                 BN_ULONG a0,a1,a2,a3;
504                 a0=B[0]; a1=B[1]; a2=B[2]; a3=B[3];
505                 A[0]=a0; A[1]=a1; A[2]=a2; A[3]=a3;
506                 }
507         switch (b->top&3)
508                 {
509                 case 3: A[2]=B[2];
510                 case 2: A[1]=B[1];
511                 case 1: A[0]=B[0];
512                 case 0: ; /* ultrix cc workaround, see comments in bn_expand_int
513                 }
514 #else
515         memcpy(a->d,b->d,sizeof(b->d[0])*b->top);
516 #endif

518         a->top=b->top;
519         a->neg=b->neg;
520         bn_check_top(a);
521         return(a);
522         }
```

```
524 void BN_swap(BIGNUM *a, BIGNUM *b)
525         {
526         int flags_old_a, flags_old_b;
527         BN_ULONG *tmp_d;
528         int tmp_top, tmp_dmax, tmp_neg;
529
530         bn_check_top(a);
531         bn_check_top(b);
532
533         flags_old_a = a->flags;
534         flags_old_b = b->flags;
535
536         tmp_d = a->d;
537         tmp_top = a->top;
538         tmp_dmax = a->dmax;
539         tmp_neg = a->neg;
540
541         a->d = b->d;
542         a->top = b->top;
543         a->dmax = b->dmax;
544         a->neg = b->neg;
545
546         b->d = tmp_d;
547         b->top = tmp_top;
548         b->dmax = tmp_dmax;
549         b->neg = tmp_neg;
550
551         a->flags = (flags_old_a & BN_FLG_MALLOCED) | (flags_old_b & BN_FLG_STATI
552         b->flags = (flags_old_b & BN_FLG_MALLOCED) | (flags_old_a & BN_FLG_STATI
553         bn_check_top(a);
554         bn_check_top(b);
555         }
556
557 void BN_clear(BIGNUM *a)
558         {
559         bn_check_top(a);
560         if (a->d != NULL)
561                 memset(a->d,0,a->dmax*sizeof(a->d[0]));
562         a->top=0;
563         a->neg=0;
564         }
565
566 BN_ULONG BN_get_word(const BIGNUM *a)
567         {
568         if (a->top > 1)
569                 return BN_MASK2;
570         else if (a->top == 1)
571                 return a->d[0];
572         /* a->top == 0 */
573         return 0;
574         }
575
576 int BN_set_word(BIGNUM *a, BN_ULONG w)
577         {
578         bn_check_top(a);
579         if (bn_expand(a,(int)sizeof(BN_ULONG)*8) == NULL) return(0);
580         a->neg = 0;
581         a->d[0] = w;
582         a->top = (w ? 1 : 0);
583         bn_check_top(a);
584         return(1);
585         }
586
587 BIGNUM *BN_bin2bn(const unsigned char *s, int len, BIGNUM *ret)
588         {
589         unsigned int i,m;
```



```
590         unsigned int n;
591         BN_ULONG l;
592         BIGNUM  *bn = NULL;
593
594         if (ret == NULL)
595                 ret = bn = BN_new();
596         if (ret == NULL) return(NULL);
597         bn_check_top(ret);
598         l=0;
599         n=len;
600         if (n == 0)
601                 {
602                 ret->top=0;
603                 return(ret);
604                 }
605         i=((n-1)/BN_BYTES)+1;
606         m=((n-1)%(BN_BYTES));
607         if (bn_wexpand(ret, (int)i) == NULL)
608                 {
609                 if (bn) BN_free(bn);
610                 return NULL;
611                 }
612         ret->top=i;
613         ret->neg=0;
614         while (n--)
615                 {
616                 l=(l<<8L)| *(s++);
617                 if (m-- == 0)
618                         {
619                         ret->d[--i]=l;
620                         l=0;
621                         m=BN_BYTES-1;
622                         }
623                 }
624         /* need to call this due to clear byte at top if avoiding
625          * having the top bit set (-ve number) */
626         bn_correct_top(ret);
627         return(ret);
628         }
629
630 /* ignore negative */
631 int BN_bn2bin(const BIGNUM *a, unsigned char *to)
632         {
633         int n,i;
634         BN_ULONG l;
635
636         bn_check_top(a);
637         n=i=BN_num_bytes(a);
638         while (i--)
639                 {
640                 l=a->d[i/BN_BYTES];
641                 *(to++)=(unsigned char)(l>>(8*(i%BN_BYTES)))&0xff;
642                 }
643         return(n);
644         }
645
646 int BN_ucmp(const BIGNUM *a, const BIGNUM *b)
647         {
648         int i;
649         BN_ULONG t1,t2,*ap,*bp;
650
651         bn_check_top(a);
652         bn_check_top(b);
653
654         i=a->top-b->top;
655         if (i != 0) return(i);
```

```
656         ap=a->d;
657         bp=b->d;
658         for (i=a->top-1; i>=0; i--)
659                 {
660                 t1= ap[i];
661                 t2= bp[i];
662                 if (t1 != t2)
663                         return((t1 > t2) ? 1 : -1);
664                 }
665         return(0);
666         }

668 int BN_cmp(const BIGNUM *a, const BIGNUM *b)
669         {
670         int i;
671         int gt,lt;
672         BN_ULONG t1,t2;

674         if ((a == NULL) || (b == NULL))
675                 {
676                 if (a != NULL)
677                         return(-1);
678                 else if (b != NULL)
679                         return(1);
680                 else
681                         return(0);
682                 }

684         bn_check_top(a);
685         bn_check_top(b);

687         if (a->neg != b->neg)
688                 {
689                 if (a->neg)
690                         return(-1);
691                 else    return(1);
692                 }
693         if (a->neg == 0)
694                 { gt=1; lt= -1; }
695         else    { gt= -1; lt=1; }

697         if (a->top > b->top) return(gt);
698         if (a->top < b->top) return(lt);
699         for (i=a->top-1; i>=0; i--)
700                 {
701                 t1=a->d[i];
702                 t2=b->d[i];
703                 if (t1 > t2) return(gt);
704                 if (t1 < t2) return(lt);
705                 }
706         return(0);
707         }

709 int BN_set_bit(BIGNUM *a, int n)
710         {
711         int i,j,k;

713         if (n < 0)
714                 return 0;

716         i=n/BN_BITS2;
717         j=n%BN_BITS2;
718         if (a->top <= i)
719                 {
720                 if (bn_wexpand(a,i+1) == NULL) return(0);
721                 for(k=a->top; k<i+1; k++)
```



```
722                         a->d[k]=0;
723                 a->top=i+1;
724                 }

726         a->d[i]|=(((BN_ULONG)1)<<j);
727         bn_check_top(a);
728         return(1);
729         }

731 int BN_clear_bit(BIGNUM *a, int n)
732         {
733         int i,j;

735         bn_check_top(a);
736         if (n < 0) return 0;

738         i=n/BN_BITS2;
739         j=n%BN_BITS2;
740         if (a->top <= i) return(0);

742         a->d[i]&=(~(((BN_ULONG)1)<<j));
743         bn_correct_top(a);
744         return(1);
745         }

747 int BN_is_bit_set(const BIGNUM *a, int n)
748         {
749         int i,j;

751         bn_check_top(a);
752         if (n < 0) return 0;
753         i=n/BN_BITS2;
754         j=n%BN_BITS2;
755         if (a->top <= i) return 0;
756         return (int)(((a->d[i])>>j)&((BN_ULONG)1));
757         }

759 int BN_mask_bits(BIGNUM *a, int n)
760         {
761         int b,w;

763         bn_check_top(a);
764         if (n < 0) return 0;

766         w=n/BN_BITS2;
767         b=n%BN_BITS2;
768         if (w >= a->top) return 0;
769         if (b == 0)
770                 a->top=w;
771         else
772                 {
773                 a->top=w+1;
774                 a->d[w]&= ~(BN_MASK2<<b);
775                 }
776         bn_correct_top(a);
777         return(1);
778         }

780 void BN_set_negative(BIGNUM *a, int b)
781         {
782         if (b && !BN_is_zero(a))
783                 a->neg = 1;
784         else
785                 a->neg = 0;
786         }
```

```
int bn_cmp_words(const BN_ULONG *a, const BN_ULONG *b, int n)
        {
        int i;
        BN_ULONG aa,bb;

        aa=a[n-1];
        bb=b[n-1];
        if (aa != bb) return((aa > bb)?1:-1);
        for (i=n-2; i>=0; i--)
                {
                aa=a[i];
                bb=b[i];
                if (aa != bb) return((aa > bb)?1:-1);
                }
        return(0);
        }

/* Here follows a specialised variants of bn_cmp_words().  It has the
   property of performing the operation on arrays of different sizes.
   The sizes of those arrays is expressed through cl, which is the
   common length ( basicall, min(len(a),len(b)) ), and dl, which is the
   delta between the two lengths, calculated as len(a)-len(b).
   All lengths are the number of BN_ULONGs...  */

int bn_cmp_part_words(const BN_ULONG *a, const BN_ULONG *b,
        int cl, int dl)
        {
        int n,i;
        n = cl-1;

        if (dl < 0)
                {
                for (i=dl; i<0; i++)
                        {
                        if (b[n-i] != 0)
                                return -1; /* a < b */
                        }
                }
        if (dl > 0)
                {
                for (i=dl; i>0; i--)
                        {
                        if (a[n+i] != 0)
                                return 1; /* a > b */
                        }
                }
        return bn_cmp_words(a,b,cl);
        }

/*
 * Constant-time conditional swap of a and b.
 * a and b are swapped if condition is not 0.  The code assumes that at most one
 * nwords is the number of words to swap.  The code assumes that at least nwords
 * and that no more than nwords are used by either a or b.
 * a and b cannot be the same number
 */
void BN_consttime_swap(BN_ULONG condition, BIGNUM *a, BIGNUM *b, int nwords)
        {
        BN_ULONG t;
        int i;

        bn_wcheck_size(a, nwords);
        bn_wcheck_size(b, nwords);

        assert(a != b);
        assert((condition & (condition - 1)) == 0);
```



```
        assert(sizeof(BN_ULONG) >= sizeof(int));

        condition = ((condition - 1) >> (BN_BITS2 - 1)) - 1;

        t = (a->top^b->top) & condition;
        a->top ^= t;
        b->top ^= t;

#define BN_CONSTTIME_SWAP(ind) \
        do { \
                t = (a->d[ind] ^ b->d[ind]) & condition; \
                a->d[ind] ^= t; \
                b->d[ind] ^= t; \
        } while (0)


        switch (nwords) {
        default:
                for (i = 10; i < nwords; i++)
                        BN_CONSTTIME_SWAP(i);
                /* Fallthrough */
        case 10: BN_CONSTTIME_SWAP(9); /* Fallthrough */
        case 9: BN_CONSTTIME_SWAP(8); /* Fallthrough */
        case 8: BN_CONSTTIME_SWAP(7); /* Fallthrough */
        case 7: BN_CONSTTIME_SWAP(6); /* Fallthrough */
        case 6: BN_CONSTTIME_SWAP(5); /* Fallthrough */
        case 5: BN_CONSTTIME_SWAP(4); /* Fallthrough */
        case 4: BN_CONSTTIME_SWAP(3); /* Fallthrough */
        case 3: BN_CONSTTIME_SWAP(2); /* Fallthrough */
        case 2: BN_CONSTTIME_SWAP(1); /* Fallthrough */
        case 1: BN_CONSTTIME_SWAP(0);
        }
#undef BN_CONSTTIME_SWAP
}
#endif /* ! codereview */
```

```
**********************************************************
    9711 Wed Aug 13 19:52:16 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_mod.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/bn/bn_mod.c */
/* Includes code written by Lenka Fibikova <fibikova@exp-math.uni-essen.de>
 * for the OpenSSL project. */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
```



```
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "cryptlib.h"
#include "bn_lcl.h"


#if 0 /* now just a #define */
int BN_mod(BIGNUM *rem, const BIGNUM *m, const BIGNUM *d, BN_CTX *ctx)
        {
        return(BN_div(NULL,rem,m,d,ctx));
        /* note that  rem->neg == m->neg  (unless the remainder is zero) */
        }
#endif


int BN_nnmod(BIGNUM *r, const BIGNUM *m, const BIGNUM *d, BN_CTX *ctx)
```

```
128         {
129         /* like BN_mod, but returns non-negative remainder
130          * (i.e.,  0 <= r < |d|  always holds) */

132         if (!(BN_mod(r,m,d,ctx)))
133                 return 0;
134         if (!r->neg)
135                 return 1;
136         /* now   -|d| < r < 0,  so we have to set  r := r + |d| */
137         return (d->neg ? BN_sub : BN_add)(r, r, d);
138 }


141 int BN_mod_add(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m, BN_
142         {
143         if (!BN_add(r, a, b)) return 0;
144         return BN_nnmod(r, r, m, ctx);
145         }


148 /* BN_mod_add variant that may be used if both  a  and  b  are non-negative
149  * and less than  m */
150 int BN_mod_add_quick(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *
151         {
152         if (!BN_uadd(r, a, b)) return 0;
153         if (BN_ucmp(r, m) >= 0)
154                 return BN_usub(r, r, m);
155         return 1;
156         }


159 int BN_mod_sub(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m, BN_
160         {
161         if (!BN_sub(r, a, b)) return 0;
162         return BN_nnmod(r, r, m, ctx);
163         }


166 /* BN_mod_sub variant that may be used if both  a  and  b  are non-negative
167  * and less than  m */
168 int BN_mod_sub_quick(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *
169         {
170         if (!BN_sub(r, a, b)) return 0;
171         if (r->neg)
172                 return BN_add(r, r, m);
173         return 1;
174         }


177 /* slow but works */
178 int BN_mod_mul(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, const BIGNUM *m,
179         BN_CTX *ctx)
180         {
181         BIGNUM *t;
182         int ret=0;

184         bn_check_top(a);
185         bn_check_top(b);
186         bn_check_top(m);

188         BN_CTX_start(ctx);
189         if ((t = BN_CTX_get(ctx)) == NULL) goto err;
190         if (a == b)
191                 { if (!BN_sqr(t,a,ctx)) goto err; }
192         else
193                 { if (!BN_mul(t,a,b,ctx)) goto err; }
```



```
194         if (!BN_nnmod(r,t,m,ctx)) goto err;
195         bn_check_top(r);
196         ret=1;
197 err:
198         BN_CTX_end(ctx);
199         return(ret);
200         }


203 int BN_mod_sqr(BIGNUM *r, const BIGNUM *a, const BIGNUM *m, BN_CTX *ctx)
204         {
205         if (!BN_sqr(r, a, ctx)) return 0;
206         /* r->neg == 0,  thus we don't need BN_nnmod */
207         return BN_mod(r, r, m, ctx);
208         }


211 int BN_mod_lshift1(BIGNUM *r, const BIGNUM *a, const BIGNUM *m, BN_CTX *ctx)
212         {
213         if (!BN_lshift1(r, a)) return 0;
214         bn_check_top(r);
215         return BN_nnmod(r, r, m, ctx);
216         }


219 /* BN_mod_lshift1 variant that may be used if  a  is non-negative
220  * and less than  m */
221 int BN_mod_lshift1_quick(BIGNUM *r, const BIGNUM *a, const BIGNUM *m)
222         {
223         if (!BN_lshift1(r, a)) return 0;
224         bn_check_top(r);
225         if (BN_cmp(r, m) >= 0)
226                 return BN_sub(r, r, m);
227         return 1;
228         }


231 int BN_mod_lshift(BIGNUM *r, const BIGNUM *a, int n, const BIGNUM *m, BN_CTX *ct
232         {
233         BIGNUM *abs_m = NULL;
234         int ret;

236         if (!BN_nnmod(r, a, m, ctx)) return 0;

238         if (m->neg)
239                 {
240                 abs_m = BN_dup(m);
241                 if (abs_m == NULL) return 0;
242                 abs_m->neg = 0;
243                 }

245         ret = BN_mod_lshift_quick(r, r, n, (abs_m ? abs_m : m));
246         bn_check_top(r);

248         if (abs_m)
249                 BN_free(abs_m);
250         return ret;
251         }


254 /* BN_mod_lshift variant that may be used if  a  is non-negative
255  * and less than  m */
256 int BN_mod_lshift_quick(BIGNUM *r, const BIGNUM *a, int n, const BIGNUM *m)
257         {
258         if (r != a)
259                 {
```

```
260                 if (BN_copy(r, a) == NULL) return 0;
261                 }

263         while (n > 0)
264                 {
265                 int max_shift;

267                 /* 0 < r < m */
268                 max_shift = BN_num_bits(m) - BN_num_bits(r);
269                 /* max_shift >= 0 */

271                 if (max_shift < 0)
272                         {
273                         BNerr(BN_F_BN_MOD_LSHIFT_QUICK, BN_R_INPUT_NOT_REDUCED);
274                         return 0;
275                         }

277                 if (max_shift > n)
278                         max_shift = n;

280                 if (max_shift)
281                         {
282                         if (!BN_lshift(r, r, max_shift)) return 0;
283                         n -= max_shift;
284                         }
285                 else
286                         {
287                         if (!BN_lshift1(r, r)) return 0;
288                         --n;
289                         }

291                 /* BN_num_bits(r) <= BN_num_bits(m) */

293                 if (BN_cmp(r, m) >= 0)
294                         {
295                         if (!BN_sub(r, r, m)) return 0;
296                         }
297                 }
298         bn_check_top(r);

300         return 1;
301         }
302 #endif /* ! codereview */
```

```
**********************************************************
   15126 Wed Aug 13 19:52:16 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_mont.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_mont.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/*
 * Details about Montgomery multiplication algorithms can be found at
 * http://security.ece.orst.edu/publications.html, e.g.
 * http://security.ece.orst.edu/koc/papers/j37acmon.pdf and
 * sections 3.8 and 4.2 in http://security.ece.orst.edu/koc/papers/r01rsasw.pdf
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"

#define MONT_WORD /* use the faster word-based algorithm */

#ifdef MONT_WORD
static int BN_from_montgomery_word(BIGNUM *ret, BIGNUM *r, BN_MONT_CTX *mont);
#endif
```

```
129 int BN_mod_mul_montgomery(BIGNUM *r, const BIGNUM *a, const BIGNUM *b,
130                          BN_MONT_CTX *mont, BN_CTX *ctx)
131         {
132         BIGNUM *tmp;
133         int ret=0;
134 #if defined(OPENSSL_BN_ASM_MONT) && defined(MONT_WORD)
135         int num = mont->N.top;
136
137         if (num>1 && a->top==num && b->top==num)
138                 {
139                 if (bn_wexpand(r,num) == NULL) return(0);
140                 if (bn_mul_mont(r->d,a->d,b->d,mont->N.d,mont->n0,num))
141                         {
142                         r->neg = a->neg^b->neg;
143                         r->top = num;
144                         bn_correct_top(r);
145                         return(1);
146                         }
147                 }
148 #endif
149
150         BN_CTX_start(ctx);
151         tmp = BN_CTX_get(ctx);
152         if (tmp == NULL) goto err;
153
154         bn_check_top(tmp);
155         if (a == b)
156                 {
157                 if (!BN_sqr(tmp,a,ctx)) goto err;
158                 }
159         else
160                 {
161                 if (!BN_mul(tmp,a,b,ctx)) goto err;
162                 }
163         /* reduce from aRR to aR */
164 #ifdef MONT_WORD
165         if (!BN_from_montgomery_word(r,tmp,mont)) goto err;
166 #else
167         if (!BN_from_montgomery(r,tmp,mont,ctx)) goto err;
168 #endif
169         bn_check_top(r);
170         ret=1;
171 err:
172         BN_CTX_end(ctx);
173         return(ret);
174         }
175
176 #ifdef MONT_WORD
177 static int BN_from_montgomery_word(BIGNUM *ret, BIGNUM *r, BN_MONT_CTX *mont)
178         {
179         BIGNUM *n;
180         BN_ULONG *ap,*np,*rp,n0,v,carry;
181         int nl,max,i;
182
183         n= &(mont->N);
184         nl=n->top;
185         if (nl == 0) { ret->top=0; return(1); }
186
187         max=(2*nl); /* carry is stored separately */
188         if (bn_wexpand(r,max) == NULL) return(0);
189
190         r->neg^=n->neg;
191         np=n->d;
192         rp=r->d;
```



```
194         /* clear the top words of T */
195 #if 1
196         for (i=r->top; i<max; i++) /* memset? XXX */
197                 rp[i]=0;
198 #else
199         memset(&(rp[r->top]),0,(max-r->top)*sizeof(BN_ULONG));
200 #endif
201
202         r->top=max;
203         n0=mont->n0[0];
204
205 #ifdef BN_COUNT
206         fprintf(stderr,"word BN_from_montgomery_word %d * %d\n",nl,nl);
207 #endif
208         for (carry=0, i=0; i<nl; i++, rp++)
209                 {
210 #ifdef __TANDEM
211                 {
212                    long long t1;
213                    long long t2;
214                    long long t3;
215                    t1 = rp[0] * (n0 & 0177777);
216                    t2 = 037777600000l;
217                    t2 = n0 & t2;
218                    t3 = rp[0] & 0177777;
219                    t2 = (t3 * t2) & BN_MASK2;
220                    t1 = t1 + t2;
221                    v=bn_mul_add_words(rp,np,nl,(BN_ULONG) t1);
222                 }
223 #else
224                 v=bn_mul_add_words(rp,np,nl,(rp[0]*n0)&BN_MASK2);
225 #endif
226                 v = (v+carry+rp[nl])&BN_MASK2;
227                 carry |= (v != rp[nl]);
228                 carry &= (v <= rp[nl]);
229                 rp[nl]=v;
230                 }
231
232         if (bn_wexpand(ret,nl) == NULL) return(0);
233         ret->top=nl;
234         ret->neg=r->neg;
235
236         rp=ret->d;
237         ap=&(r->d[nl]);
238
239 #define BRANCH_FREE 1
240 #if BRANCH_FREE
241         {
242         BN_ULONG *nrp;
243         size_t m;
244
245         v=bn_sub_words(rp,ap,np,nl)-carry;
246         /* if subtraction result is real, then
247          * trick unconditional memcpy below to perform in-place
248          * "refresh" instead of actual copy. */
249         m=(0-(size_t)v);
250         nrp=(BN_ULONG *)(((PTR_SIZE_INT)rp&~m)|((PTR_SIZE_INT)ap&m));
251
252         for (i=0,nl-=4; i<nl; i+=4)
253                 {
254                 BN_ULONG t1,t2,t3,t4;
255
256                 t1=nrp[i+0];
257                 t2=nrp[i+1];
258                 t3=nrp[i+2];    ap[i+0]=0;
259                 t4=nrp[i+3];    ap[i+1]=0;
```

```
260                 rp[i+0]=t1;     ap[i+2]=0;
261                 rp[i+1]=t2;     ap[i+3]=0;
262                 rp[i+2]=t3;
263                 rp[i+3]=t4;
264                 }
265         for (nl+=4; i<nl; i++)
266                 rp[i]=nrp[i], ap[i]=0;
267         }
268 #else
269         if (bn_sub_words (rp,ap,np,nl)-carry)
270                 memcpy(rp,ap,nl*sizeof(BN_ULONG));
271 #endif
272         bn_correct_top(r);
273         bn_correct_top(ret);
274         bn_check_top(ret);

276         return(1);
277         }
278 #endif  /* MONT_WORD */

280 int BN_from_montgomery(BIGNUM *ret, const BIGNUM *a, BN_MONT_CTX *mont,
281              BN_CTX *ctx)
282         {
283         int retn=0;
284 #ifdef MONT_WORD
285         BIGNUM *t;

287         BN_CTX_start(ctx);
288         if ((t = BN_CTX_get(ctx)) && BN_copy(t,a))
289                 retn = BN_from_montgomery_word(ret,t,mont);
290         BN_CTX_end(ctx);
291 #else /* !MONT_WORD */
292         BIGNUM *t1,*t2;

294         BN_CTX_start(ctx);
295         t1 = BN_CTX_get(ctx);
296         t2 = BN_CTX_get(ctx);
297         if (t1 == NULL || t2 == NULL) goto err;

299         if (!BN_copy(t1,a)) goto err;
300         BN_mask_bits(t1,mont->ri);

302         if (!BN_mul(t2,t1,&mont->Ni,ctx)) goto err;
303         BN_mask_bits(t2,mont->ri);

305         if (!BN_mul(t1,t2,&mont->N,ctx)) goto err;
306         if (!BN_add(t2,a,t1)) goto err;
307         if (!BN_rshift(ret,t2,mont->ri)) goto err;

309         if (BN_ucmp(ret, &(mont->N)) >= 0)
310                 {
311                 if (!BN_usub(ret,ret,&(mont->N))) goto err;
312                 }
313         retn=1;
314         bn_check_top(ret);
315  err:
316         BN_CTX_end(ctx);
317 #endif /* MONT_WORD */
318         return(retn);
319         }

321 BN_MONT_CTX *BN_MONT_CTX_new(void)
322         {
323         BN_MONT_CTX *ret;

325         if ((ret=(BN_MONT_CTX *)OPENSSL_malloc(sizeof(BN_MONT_CTX))) == NULL)
```



```
326                 return(NULL);

328         BN_MONT_CTX_init(ret);
329         ret->flags=BN_FLG_MALLOCED;
330         return(ret);
331         }

333 void BN_MONT_CTX_init(BN_MONT_CTX *ctx)
334         {
335         ctx->ri=0;
336         BN_init(&(ctx->RR));
337         BN_init(&(ctx->N));
338         BN_init(&(ctx->Ni));
339         ctx->n0[0] = ctx->n0[1] = 0;
340         ctx->flags=0;
341         }

343 void BN_MONT_CTX_free(BN_MONT_CTX *mont)
344         {
345         if(mont == NULL)
346             return;

348         BN_free(&(mont->RR));
349         BN_free(&(mont->N));
350         BN_free(&(mont->Ni));
351         if (mont->flags & BN_FLG_MALLOCED)
352                 OPENSSL_free(mont);
353         }

355 int BN_MONT_CTX_set(BN_MONT_CTX *mont, const BIGNUM *mod, BN_CTX *ctx)
356         {
357         int ret = 0;
358         BIGNUM *Ri,*R;

360         BN_CTX_start(ctx);
361         if((Ri = BN_CTX_get(ctx)) == NULL) goto err;
362         R= &(mont->RR);                                 /* grab RR as a temp */
363         if (!BN_copy(&(mont->N),mod)) goto err;         /* Set N */
364         mont->N.neg = 0;

366 #ifdef MONT_WORD
367                 {
368                 BIGNUM tmod;
369                 BN_ULONG buf[2];

371                 BN_init(&tmod);
372                 tmod.d=buf;
373                 tmod.dmax=2;
374                 tmod.neg=0;

376                 mont->ri=(BN_num_bits(mod)+(BN_BITS2-1))/BN_BITS2*BN_BITS2;

378 #if defined(OPENSSL_BN_ASM_MONT) && (BN_BITS2<=32)
379                 /* Only certain BN_BITS2<=32 platforms actually make use of
380                  * n0[1], and we could use the #else case (with a shorter R
381                  * value) for the others.  However, currently only the assembler
382                  * files do know which is which. */

384                 BN_zero(R);
385                 if (!(BN_set_bit(R,2*BN_BITS2))) goto err;

387                                                                 tmod.top=0;
388                 if ((buf[0] = mod->d[0]))                       tmod.top=1;
389                 if ((buf[1] = mod->top>1 ? mod->d[1] : 0))      tmod.top=2;

391                 if ((BN_mod_inverse(Ri,R,&tmod,ctx)) == NULL)
```

```
392                     goto err;
393             if (!BN_lshift(Ri,Ri,2*BN_BITS2)) goto err; /* R*Ri */
394             if (!BN_is_zero(Ri))
395                     {
396                     if (!BN_sub_word(Ri,1)) goto err;
397                     }
398             else /* if N mod word size == 1 */
399                     {
400                     if (bn_expand(Ri,(int)sizeof(BN_ULONG)*2) == NULL)
401                             goto err;
402                     /* Ri-- (mod double word size) */
403                     Ri->neg=0;
404                     Ri->d[0]=BN_MASK2;
405                     Ri->d[1]=BN_MASK2;
406                     Ri->top=2;
407                     }
408             if (!BN_div(Ri,NULL,Ri,&tmod,ctx)) goto err;
409             /* Ni = (R*Ri-1)/N,
410              * keep only couple of least significant words: */
411             mont->n0[0] = (Ri->top > 0) ? Ri->d[0] : 0;
412             mont->n0[1] = (Ri->top > 1) ? Ri->d[1] : 0;
413 #else
414             BN_zero(R);
415             if (!(BN_set_bit(R,BN_BITS2))) goto err;        /* R */

417             buf[0]=mod->d[0]; /* tmod = N mod word size */
418             buf[1]=0;
419             tmod.top = buf[0] != 0 ? 1 : 0;
420                                                     /* Ri = R^-1 mod N*/
421             if ((BN_mod_inverse(Ri,R,&tmod,ctx)) == NULL)
422                     goto err;
423             if (!BN_lshift(Ri,Ri,BN_BITS2)) goto err; /* R*Ri */
424             if (!BN_is_zero(Ri))
425                     {
426                     if (!BN_sub_word(Ri,1)) goto err;
427                     }
428             else /* if N mod word size == 1 */
429                     {
430                     if (!BN_set_word(Ri,BN_MASK2)) goto err;  /* Ri-- (mod w
431                     }
432             if (!BN_div(Ri,NULL,Ri,&tmod,ctx)) goto err;
433             /* Ni = (R*Ri-1)/N,
434              * keep only least significant word: */
435             mont->n0[0] = (Ri->top > 0) ? Ri->d[0] : 0;
436             mont->n0[1] = 0;
437 #endif
438             }
439 #else /* !MONT_WORD */
440             { /* bignum version */
441             mont->ri=BN_num_bits(&mont->N);
442             BN_zero(R);
443             if (!BN_set_bit(R,mont->ri)) goto err;  /* R = 2^ri */
444                                                     /* Ri = R^-1 mod N*/
445             if ((BN_mod_inverse(Ri,R,&mont->N,ctx)) == NULL)
446                     goto err;
447             if (!BN_lshift(Ri,Ri,mont->ri)) goto err; /* R*Ri */
448             if (!BN_sub_word(Ri,1)) goto err;
449                                                     /* Ni = (R*Ri-1) / N */
450             if (!BN_div(&(mont->Ni),NULL,Ri,&mont->N,ctx)) goto err;
451             }
452 #endif

454         /* setup RR for conversions */
455         BN_zero(&(mont->RR));
456         if (!BN_set_bit(&(mont->RR),mont->ri*2)) goto err;
457         if (!BN_mod(&(mont->RR),&(mont->RR),&(mont->N),ctx)) goto err;
```



```
459         ret = 1;
460 err:
461         BN_CTX_end(ctx);
462         return ret;
463         }

465 BN_MONT_CTX *BN_MONT_CTX_copy(BN_MONT_CTX *to, BN_MONT_CTX *from)
466         {
467         if (to == from) return(to);

469         if (!BN_copy(&(to->RR),&(from->RR))) return NULL;
470         if (!BN_copy(&(to->N),&(from->N))) return NULL;
471         if (!BN_copy(&(to->Ni),&(from->Ni))) return NULL;
472         to->ri=from->ri;
473         to->n0[0]=from->n0[0];
474         to->n0[1]=from->n0[1];
475         return(to);
476         }

478 BN_MONT_CTX *BN_MONT_CTX_set_locked(BN_MONT_CTX **pmont, int lock,
479                                     const BIGNUM *mod, BN_CTX *ctx)
480         {
481         BN_MONT_CTX *ret;

483         CRYPTO_r_lock(lock);
484         ret = *pmont;
485         CRYPTO_r_unlock(lock);
486         if (ret)
487                 return ret;

489         /* We don't want to serialise globally while doing our lazy-init math in
490          * BN_MONT_CTX_set. That punishes threads that are doing independent
491          * things. Instead, punish the case where more than one thread tries to
492          * lazy-init the same 'pmont', by having each do the lazy-init math work
493          * independently and only use the one from the thread that wins the race
494          * (the losers throw away the work they've done). */
495         ret = BN_MONT_CTX_new();
496         if (!ret)
497                 return NULL;
498         if (!BN_MONT_CTX_set(ret, mod, ctx))
499                 {
500                 BN_MONT_CTX_free(ret);
501                 return NULL;
502                 }

504         /* The locked compare-and-set, after the local work is done. */
505         CRYPTO_w_lock(lock);
506         if (*pmont)
507                 {
508                 BN_MONT_CTX_free(ret);
509                 ret = *pmont;
510                 }
511         else
512                 *pmont = ret;
513         CRYPTO_w_unlock(lock);
514         return ret;
515         }
516 #endif /* ! codereview */
```

```
**********************************************************
    4357 Wed Aug 13 19:52:16 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_mpi.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_mpi.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"
```



```
int BN_bn2mpi(const BIGNUM *a, unsigned char *d)
        {
        int bits;
        int num=0;
        int ext=0;
        long l;

        bits=BN_num_bits(a);
        num=(bits+7)/8;
        if (bits > 0)
                {
                ext=((bits & 0x07) == 0);
                }
        if (d == NULL)
                return(num+4+ext);

        l=num+ext;
        d[0]=(unsigned char)(l>>24)&0xff;
        d[1]=(unsigned char)(l>>16)&0xff;
        d[2]=(unsigned char)(l>> 8)&0xff;
        d[3]=(unsigned char)(l    )&0xff;
        if (ext) d[4]=0;
        num=BN_bn2bin(a,&(d[4+ext]));
        if (a->neg)
                d[4]|=0x80;
        return(num+4+ext);
        }

BIGNUM *BN_mpi2bn(const unsigned char *d, int n, BIGNUM *a)
        {
        long len;
        int neg=0;

        if (n < 4)
                {
                BNerr(BN_F_BN_MPI2BN,BN_R_INVALID_LENGTH);
                return(NULL);
                }
        len=((long)d[0]<<24)|((long)d[1]<<16)|((int)d[2]<<8)|(int)d[3];
        if ((len+4) != n)
                {
                BNerr(BN_F_BN_MPI2BN,BN_R_ENCODING_ERROR);
                return(NULL);
                }

        if (a == NULL) a=BN_new();
        if (a == NULL) return(NULL);

        if (len == 0)
                {
                a->neg=0;
                a->top=0;
                return(a);
                }
        d+=4;
        if ((*d) & 0x80)
                neg=1;
        if (BN_bin2bn(d,(int)len,a) == NULL)
                return(NULL);
        a->neg=neg;
        if (neg)
                {
                BN_clear_bit(a,BN_num_bits(a)-1);
                }
        bn_check_top(a);
```

```
128         return(a);
129         }
130 #endif /* ! codereview */
```

```
**********************************************************
   25356 Wed Aug 13 19:52:16 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_mul.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_mul.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#ifndef BN_DEBUG
# undef NDEBUG /* avoid conflicting definitions */
# define NDEBUG
```



```
#endif

#include <stdio.h>
#include <assert.h>
#include "cryptlib.h"
#include "bn_lcl.h"

#if defined(OPENSSL_NO_ASM) || !defined(OPENSSL_BN_ASM_PART_WORDS)
/* Here follows specialised variants of bn_add_words() and
   bn_sub_words().  They have the property performing operations on
   arrays of different sizes.  The sizes of those arrays is expressed through
   cl, which is the common length ( basicall, min(len(a),len(b)) ), and dl,
   which is the delta between the two lengths, calculated as len(a)-len(b).
   All lengths are the number of BN_ULONGs...  For the operations that require
   a result array as parameter, it must have the length cl+abs(dl).
   These functions should probably end up in bn_asm.c as soon as there are
   assembler counterparts for the systems that use assembler files.  */

BN_ULONG bn_sub_part_words(BN_ULONG *r,
        const BN_ULONG *a, const BN_ULONG *b,
        int cl, int dl)
        {
        BN_ULONG c, t;

        assert(cl >= 0);
        c = bn_sub_words(r, a, b, cl);

        if (dl == 0)
                return c;

        r += cl;
        a += cl;
        b += cl;

        if (dl < 0)
                {
#ifdef BN_COUNT
                fprintf(stderr, "  bn_sub_part_words %d + %d (dl < 0, c = %d)\n"
#endif
                for (;;)
                        {
                        t = b[0];
                        r[0] = (0-t-c)&BN_MASK2;
                        if (t != 0) c=1;
                        if (++dl >= 0) break;

                        t = b[1];
                        r[1] = (0-t-c)&BN_MASK2;
                        if (t != 0) c=1;
                        if (++dl >= 0) break;

                        t = b[2];
                        r[2] = (0-t-c)&BN_MASK2;
                        if (t != 0) c=1;
                        if (++dl >= 0) break;

                        t = b[3];
                        r[3] = (0-t-c)&BN_MASK2;
                        if (t != 0) c=1;
                        if (++dl >= 0) break;

                        b += 4;
                        r += 4;
                        }
                }
        else
```

```
128                 {
129                 int save_dl = dl;
130 #ifdef BN_COUNT
131                 fprintf(stderr, "  bn_sub_part_words %d + %d (dl > 0, c = %d)\n"
132 #endif
133                 while(c)
134                         {
135                         t = a[0];
136                         r[0] = (t-c)&BN_MASK2;
137                         if (t != 0) c=0;
138                         if (--dl <= 0) break;
139 
140                         t = a[1];
141                         r[1] = (t-c)&BN_MASK2;
142                         if (t != 0) c=0;
143                         if (--dl <= 0) break;
144 
145                         t = a[2];
146                         r[2] = (t-c)&BN_MASK2;
147                         if (t != 0) c=0;
148                         if (--dl <= 0) break;
149 
150                         t = a[3];
151                         r[3] = (t-c)&BN_MASK2;
152                         if (t != 0) c=0;
153                         if (--dl <= 0) break;
154 
155                         save_dl = dl;
156                         a += 4;
157                         r += 4;
158                         }
159                 if (dl > 0)
160                         {
161 #ifdef BN_COUNT
162                         fprintf(stderr, "  bn_sub_part_words %d + %d (dl > 0, c
163 #endif
164                         if (save_dl > dl)
165                                 {
166                                 switch (save_dl - dl)
167                                         {
168                                 case 1:
169                                         r[1] = a[1];
170                                         if (--dl <= 0) break;
171                                 case 2:
172                                         r[2] = a[2];
173                                         if (--dl <= 0) break;
174                                 case 3:
175                                         r[3] = a[3];
176                                         if (--dl <= 0) break;
177                                         }
178                                 a += 4;
179                                 r += 4;
180                                 }
181                         }
182                 if (dl > 0)
183                         {
184 #ifdef BN_COUNT
185                         fprintf(stderr, "  bn_sub_part_words %d + %d (dl > 0, co
186 #endif
187                         for(;;)
188                                 {
189                                 r[0] = a[0];
190                                 if (--dl <= 0) break;
191                                 r[1] = a[1];
192                                 if (--dl <= 0) break;
193                                 r[2] = a[2];
```



```
194                                 if (--dl <= 0) break;
195                                 r[3] = a[3];
196                                 if (--dl <= 0) break;
197 
198                                 a += 4;
199                                 r += 4;
200                                 }
201                         }
202                 }
203         return c;
204         }
205 #endif
206 
207 BN_ULONG bn_add_part_words(BN_ULONG *r,
208         const BN_ULONG *a, const BN_ULONG *b,
209         int cl, int dl)
210         {
211         BN_ULONG c, l, t;
212 
213         assert(cl >= 0);
214         c = bn_add_words(r, a, b, cl);
215 
216         if (dl == 0)
217                 return c;
218 
219         r += cl;
220         a += cl;
221         b += cl;
222 
223         if (dl < 0)
224                 {
225                 int save_dl = dl;
226 #ifdef BN_COUNT
227                 fprintf(stderr, "  bn_add_part_words %d + %d (dl < 0, c = %d)\n"
228 #endif
229                 while (c)
230                         {
231                         l=(c+b[0])&BN_MASK2;
232                         c=(l < c);
233                         r[0]=l;
234                         if (++dl >= 0) break;
235 
236                         l=(c+b[1])&BN_MASK2;
237                         c=(l < c);
238                         r[1]=l;
239                         if (++dl >= 0) break;
240 
241                         l=(c+b[2])&BN_MASK2;
242                         c=(l < c);
243                         r[2]=l;
244                         if (++dl >= 0) break;
245 
246                         l=(c+b[3])&BN_MASK2;
247                         c=(l < c);
248                         r[3]=l;
249                         if (++dl >= 0) break;
250 
251                         save_dl = dl;
252                         b+=4;
253                         r+=4;
254                         }
255                 if (dl < 0)
256                         {
257 #ifdef BN_COUNT
258                         fprintf(stderr, "  bn_add_part_words %d + %d (dl < 0, c
259 #endif
```

```
260                         if (save_dl < dl)
261                                 {
262                                 switch (dl - save_dl)
263                                         {
264                                 case 1:
265                                         r[1] = b[1];
266                                         if (++dl >= 0) break;
267                                 case 2:
268                                         r[2] = b[2];
269                                         if (++dl >= 0) break;
270                                 case 3:
271                                         r[3] = b[3];
272                                         if (++dl >= 0) break;
273                                         }
274                                 b += 4;
275                                 r += 4;
276                                 }
277                         }
278                 if (dl < 0)
279                         {
280 #ifdef BN_COUNT
281                         fprintf(stderr, "  bn_add_part_words %d + %d (dl < 0, co
282 #endif
283                         for(;;)
284                                 {
285                                 r[0] = b[0];
286                                 if (++dl >= 0) break;
287                                 r[1] = b[1];
288                                 if (++dl >= 0) break;
289                                 r[2] = b[2];
290                                 if (++dl >= 0) break;
291                                 r[3] = b[3];
292                                 if (++dl >= 0) break;

294                                 b += 4;
295                                 r += 4;
296                                 }
297                         }
298                 }
299         else
300                 {
301                 int save_dl = dl;
302 #ifdef BN_COUNT
303                 fprintf(stderr, "  bn_add_part_words %d + %d (dl > 0)\n", cl, dl
304 #endif
305                 while (c)
306                         {
307                         t=(a[0]+c)&BN_MASK2;
308                         c=(t < c);
309                         r[0]=t;
310                         if (--dl <= 0) break;

312                         t=(a[1]+c)&BN_MASK2;
313                         c=(t < c);
314                         r[1]=t;
315                         if (--dl <= 0) break;

317                         t=(a[2]+c)&BN_MASK2;
318                         c=(t < c);
319                         r[2]=t;
320                         if (--dl <= 0) break;

322                         t=(a[3]+c)&BN_MASK2;
323                         c=(t < c);
324                         r[3]=t;
325                         if (--dl <= 0) break;
```



```
327                         save_dl = dl;
328                         a+=4;
329                         r+=4;
330                         }
331 #ifdef BN_COUNT
332                 fprintf(stderr, "  bn_add_part_words %d + %d (dl > 0, c == 0)\n"
333 #endif
334                 if (dl > 0)
335                         {
336                         if (save_dl > dl)
337                                 {
338                                 switch (save_dl - dl)
339                                         {
340                                 case 1:
341                                         r[1] = a[1];
342                                         if (--dl <= 0) break;
343                                 case 2:
344                                         r[2] = a[2];
345                                         if (--dl <= 0) break;
346                                 case 3:
347                                         r[3] = a[3];
348                                         if (--dl <= 0) break;
349                                         }
350                                 a += 4;
351                                 r += 4;
352                                 }
353                         }
354                 if (dl > 0)
355                         {
356 #ifdef BN_COUNT
357                         fprintf(stderr, "  bn_add_part_words %d + %d (dl > 0, co
358 #endif
359                         for(;;)
360                                 {
361                                 r[0] = a[0];
362                                 if (--dl <= 0) break;
363                                 r[1] = a[1];
364                                 if (--dl <= 0) break;
365                                 r[2] = a[2];
366                                 if (--dl <= 0) break;
367                                 r[3] = a[3];
368                                 if (--dl <= 0) break;

370                                 a += 4;
371                                 r += 4;
372                                 }
373                         }
374                 }
375         return c;
376         }

378 #ifdef BN_RECURSION
379 /* Karatsuba recursive multiplication algorithm
380  * (cf. Knuth, The Art of Computer Programming, Vol. 2) */

382 /* r is 2*n2 words in size,
383  * a and b are both n2 words in size.
384  * n2 must be a power of 2.
385  * We multiply and return the result.
386  * t must be 2*n2 words in size
387  * We calculate
388  * a[0]*b[0]
389  * a[0]*b[0]+a[1]*b[1]+(a[0]-a[1])*(b[1]-b[0])
390  * a[1]*b[1]
391  */
```

```
/* dnX may not be positive, but n2/2+dnX has to be */
void bn_mul_recursive(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b, int n2,
        int dna, int dnb, BN_ULONG *t)
        {
        int n=n2/2,c1,c2;
        int tna=n+dna, tnb=n+dnb;
        unsigned int neg,zero;
        BN_ULONG ln,lo,*p;

# ifdef BN_COUNT
        fprintf(stderr," bn_mul_recursive %d%+d * %d%+d\n",n2,dna,n2,dnb);
# endif
# ifdef BN_MUL_COMBA
#  if 0
        if (n2 == 4)
                {
                bn_mul_comba4(r,a,b);
                return;
                }
#  endif
        /* Only call bn_mul_comba 8 if n2 == 8 and the
         * two arrays are complete [steve]
         */
        if (n2 == 8 && dna == 0 && dnb == 0)
                {
                bn_mul_comba8(r,a,b);
                return;
                }
# endif /* BN_MUL_COMBA */
        /* Else do normal multiply */
        if (n2 < BN_MUL_RECURSIVE_SIZE_NORMAL)
                {
                bn_mul_normal(r,a,n2+dna,b,n2+dnb);
                if ((dna + dnb) < 0)
                        memset(&r[2*n2 + dna + dnb], 0,
                                sizeof(BN_ULONG) * -(dna + dnb));
                return;
                }
        /* r=(a[0]-a[1])*(b[1]-b[0]) */
        c1=bn_cmp_part_words(a,&(a[n]),tna,n-tna);
        c2=bn_cmp_part_words(&(b[n]),b,tnb,tnb-n);
        zero=neg=0;
        switch (c1*3+c2)
                {
        case -4:
                bn_sub_part_words(t,      &(a[n]),a,      tna,tna-n); /* - */
                bn_sub_part_words(&(t[n]),b,      &(b[n]),tnb,n-tnb); /* - */
                break;
        case -3:
                zero=1;
                break;
        case -2:
                bn_sub_part_words(t,      &(a[n]),a,      tna,tna-n); /* - */
                bn_sub_part_words(&(t[n]),&(b[n]),b,      tnb,tnb-n); /* + */
                neg=1;
                break;
        case -1:
        case 0:
        case 1:
                zero=1;
                break;
        case 2:
                bn_sub_part_words(t,      a,      &(a[n]),tna,n-tna); /* + */
                bn_sub_part_words(&(t[n]),b,      &(b[n]),tnb,n-tnb); /* - */
                neg=1;
                break;
```



```
        case 3:
                zero=1;
                break;
        case 4:
                bn_sub_part_words(t,      a,      &(a[n]),tna,n-tna);
                bn_sub_part_words(&(t[n]),&(b[n]),b,      tnb,tnb-n);
                break;
                }

# ifdef BN_MUL_COMBA
        if (n == 4 && dna == 0 && dnb == 0) /* XXX: bn_mul_comba4 could take
                                               extra args to do this well */
                {
                if (!zero)
                        bn_mul_comba4(&(t[n2]),t,&(t[n]));
                else
                        memset(&(t[n2]),0,8*sizeof(BN_ULONG));

                bn_mul_comba4(r,a,b);
                bn_mul_comba4(&(r[n2]),&(a[n]),&(b[n]));
                }
        else if (n == 8 && dna == 0 && dnb == 0) /* XXX: bn_mul_comba8 could
                                                    take extra args to do this
                                                    well */
                {
                if (!zero)
                        bn_mul_comba8(&(t[n2]),t,&(t[n]));
                else
                        memset(&(t[n2]),0,16*sizeof(BN_ULONG));

                bn_mul_comba8(r,a,b);
                bn_mul_comba8(&(r[n2]),&(a[n]),&(b[n]));
                }
        else
# endif /* BN_MUL_COMBA */
                {
                p= &(t[n2*2]);
                if (!zero)
                        bn_mul_recursive(&(t[n2]),t,&(t[n]),n,0,0,p);
                else
                        memset(&(t[n2]),0,n2*sizeof(BN_ULONG));
                bn_mul_recursive(r,a,b,n,0,0,p);
                bn_mul_recursive(&(r[n2]),&(a[n]),&(b[n]),n,dna,dnb,p);
                }

        /* t[32] holds (a[0]-a[1])*(b[1]-b[0]), c1 is the sign
         * r[10] holds (a[0]*b[0])
         * r[32] holds (b[1]*b[1])
         */

        c1=(int)(bn_add_words(t,r,&(r[n2]),n2));

        if (neg) /* if t[32] is negative */
                {
                c1-=(int)(bn_sub_words(&(t[n2]),t,&(t[n2]),n2));
                }
        else
                {
                /* Might have a carry */
                c1+=(int)(bn_add_words(&(t[n2]),&(t[n2]),t,n2));
                }

        /* t[32] holds (a[0]-a[1])*(b[1]-b[0])+(a[0]*b[0])+(a[1]*b[1])
         * r[10] holds (a[0]*b[0])
         * r[32] holds (b[1]*b[1])
         * c1 holds the carry bits
```

```
524          */
525         c1+=(int)(bn_add_words(&(r[n]),&(r[n]),&(t[n2]),n2));
526         if (c1)
527                 {
528                 p= &(r[n+n2]);
529                 lo= *p;
530                 ln=(lo+c1)&BN_MASK2;
531                 *p=ln;
532
533                 /* The overflow will stop before we over write
534                  * words we should not overwrite */
535                 if (ln < (BN_ULONG)c1)
536                         {
537                         do      {
538                                 p++;
539                                 lo= *p;
540                                 ln=(lo+1)&BN_MASK2;
541                                 *p=ln;
542                                 } while (ln == 0);
543                         }
544                 }
545         }
546
547 /* n+tn is the word length
548  * t needs to be n*4 is size, as does r */
549 /* tnX may not be negative but less than n */
550 void bn_mul_part_recursive(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b, int n,
551             int tna, int tnb, BN_ULONG *t)
552         {
553         int i,j,n2=n*2;
554         int c1,c2,neg;
555         BN_ULONG ln,lo,*p;
556
557 # ifdef BN_COUNT
558         fprintf(stderr," bn_mul_part_recursive (%d%+d) * (%d%+d)\n",
559                 n, tna, n, tnb);
560 # endif
561         if (n < 8)
562                 {
563                 bn_mul_normal(r,a,n+tna,b,n+tnb);
564                 return;
565                 }
566
567         /* r=(a[0]-a[1])*(b[1]-b[0]) */
568         c1=bn_cmp_part_words(a,&(a[n]),tna,n-tna);
569         c2=bn_cmp_part_words(&(b[n]),b,tnb,tnb-n);
570         neg=0;
571         switch (c1*3+c2)
572                 {
573         case -4:
574                 bn_sub_part_words(t,      &(a[n]),a,      tna,tna-n); /* - */
575                 bn_sub_part_words(&(t[n]),b,      &(b[n]),tnb,n-tnb); /* - */
576                 break;
577         case -3:
578                 /* break; */
579         case -2:
580                 bn_sub_part_words(t,      &(a[n]),a,      tna,tna-n); /* - */
581                 bn_sub_part_words(&(t[n]),&(b[n]),b,      tnb,tnb-n); /* + */
582                 neg=1;
583                 break;
584         case -1:
585         case 0:
586         case 1:
587                 /* break; */
588         case 2:
589                 bn_sub_part_words(t,      a,      &(a[n]),tna,n-tna); /* + */
```



```
590                 bn_sub_part_words(&(t[n]),b,      &(b[n]),tnb,n-tnb); /* - */
591                 neg=1;
592                 break;
593         case 3:
594                 /* break; */
595         case 4:
596                 bn_sub_part_words(t,      a,      &(a[n]),tna,n-tna);
597                 bn_sub_part_words(&(t[n]),&(b[n]),b,      tnb,tnb-n);
598                 break;
599                 }
600                 /* The zero case isn't yet implemented here. The speedup
601                    would probably be negligible. */
602 # if 0
603         if (n == 4)
604                 {
605                 bn_mul_comba4(&(t[n2]),t,&(t[n]));
606                 bn_mul_comba4(r,a,b);
607                 bn_mul_normal(&(r[n2]),&(a[n]),tn,&(b[n]),tn);
608                 memset(&(r[n2+tn*2]),0,sizeof(BN_ULONG)*(n2-tn*2));
609                 }
610         else
611 # endif
612         if (n == 8)
613                 {
614                 bn_mul_comba8(&(t[n2]),t,&(t[n]));
615                 bn_mul_comba8(r,a,b);
616                 bn_mul_normal(&(r[n2]),&(a[n]),tna,&(b[n]),tnb);
617                 memset(&(r[n2+tna+tnb]),0,sizeof(BN_ULONG)*(n2-tna-tnb));
618                 }
619         else
620                 {
621                 p= &(t[n2*2]);
622                 bn_mul_recursive(&(t[n2]),t,&(t[n]),n,0,0,p);
623                 bn_mul_recursive(r,a,b,n,0,0,p);
624                 i=n/2;
625                 /* If there is only a bottom half to the number,
626                  * just do it */
627                 if (tna > tnb)
628                         j = tna - i;
629                 else
630                         j = tnb - i;
631                 if (j == 0)
632                         {
633                         bn_mul_recursive(&(r[n2]),&(a[n]),&(b[n]),
634                                 i,tna-i,tnb-i,p);
635                         memset(&(r[n2+i*2]),0,sizeof(BN_ULONG)*(n2-i*2));
636                         }
637                 else if (j > 0) /* eg, n == 16, i == 8 and tn == 11 */
638                                 {
639                                 bn_mul_part_recursive(&(r[n2]),&(a[n]),&(b[n]),
640                                         i,tna-i,tnb-i,p);
641                                 memset(&(r[n2+tna+tnb]),0,
642                                         sizeof(BN_ULONG)*(n2-tna-tnb));
643                                 }
644                 else /* (j < 0) eg, n == 16, i == 8 and tn == 5 */
645                         {
646                         memset(&(r[n2]),0,sizeof(BN_ULONG)*n2);
647                         if (tna < BN_MUL_RECURSIVE_SIZE_NORMAL
648                                 && tnb < BN_MUL_RECURSIVE_SIZE_NORMAL)
649                                 {
650                                 bn_mul_normal(&(r[n2]),&(a[n]),tna,&(b[n]),tnb);
651                                 }
652                         else
653                                 {
654                                 for (;;)
655                                         {
```

```
656                                     i/=2;
657                                     /* these simplified conditions work
658                                      * exclusively because difference
659                                      * between tna and tnb is 1 or 0 */
660                                     if (i < tna || i < tnb)
661                                             {
662                                             bn_mul_part_recursive(&(r[n2]),
663                                                     &(a[n]),&(b[n]),
664                                                     i,tna-i,tnb-i,p);
665                                             break;
666                                             }
667                                     else if (i == tna || i == tnb)
668                                             {
669                                             bn_mul_recursive(&(r[n2]),
670                                                     &(a[n]),&(b[n]),
671                                                     i,tna-i,tnb-i,p);
672                                             break;
673                                             }
674                                     }
675                             }
676                     }
677             }

679         /* t[32] holds (a[0]-a[1])*(b[1]-b[0]), c1 is the sign
680          * r[10] holds (a[0]*b[0])
681          * r[32] holds (b[1]*b[1])
682          */

684         c1=(int)(bn_add_words(t,r,&(r[n2]),n2));

686         if (neg) /* if t[32] is negative */
687                 {
688                 c1-=(int)(bn_sub_words(&(t[n2]),t,&(t[n2]),n2));
689                 }
690         else
691                 {
692                 /* Might have a carry */
693                 c1+=(int)(bn_add_words(&(t[n2]),&(t[n2]),t,n2));
694                 }

696         /* t[32] holds (a[0]-a[1])*(b[1]-b[0])+(a[0]*b[0])+(a[1]*b[1])
697          * r[10] holds (a[0]*b[0])
698          * r[32] holds (b[1]*b[1])
699          * c1 holds the carry bits
700          */
701         c1+=(int)(bn_add_words(&(r[n]),&(r[n]),&(t[n2]),n2));
702         if (c1)
703                 {
704                 p= &(r[n+n2]);
705                 lo= *p;
706                 ln=(lo+c1)&BN_MASK2;
707                 *p=ln;

709                 /* The overflow will stop before we over write
710                  * words we should not overwrite */
711                 if (ln < (BN_ULONG)c1)
712                         {
713                         do      {
714                                 p++;
715                                 lo= *p;
716                                 ln=(lo+1)&BN_MASK2;
717                                 *p=ln;
718                                 } while (ln == 0);
719                         }
720                 }
721         }
```



```
723 /* a and b must be the same size, which is n2.
724  * r needs to be n2 words and t needs to be n2*2
725  */
726 void bn_mul_low_recursive(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b, int n2,
727                           BN_ULONG *t)
728         {
729         int n=n2/2;

731 # ifdef BN_COUNT
732         fprintf(stderr," bn_mul_low_recursive %d * %d\n",n2,n2);
733 # endif

735         bn_mul_recursive(r,a,b,n,0,0,&(t[0]));
736         if (n >= BN_MUL_LOW_RECURSIVE_SIZE_NORMAL)
737                 {
738                 bn_mul_low_recursive(&(t[0]),&(a[0]),&(b[n]),n,&(t[n2]));
739                 bn_add_words(&(r[n]),&(r[n]),&(t[0]),n);
740                 bn_mul_low_recursive(&(t[0]),&(a[n]),&(b[0]),n,&(t[n2]));
741                 bn_add_words(&(r[n]),&(r[n]),&(t[0]),n);
742                 }
743         else
744                 {
745                 bn_mul_low_normal(&(t[0]),&(a[0]),&(b[n]),n);
746                 bn_mul_low_normal(&(t[n]),&(a[n]),&(b[0]),n);
747                 bn_add_words(&(r[n]),&(r[n]),&(t[0]),n);
748                 bn_add_words(&(r[n]),&(r[n]),&(t[n]),n);
749                 }
750         }

752 /* a and b must be the same size, which is n2.
753  * r needs to be n2 words and t needs to be n2*2
754  * l is the low words of the output.
755  * t needs to be n2*3
756  */
757 void bn_mul_high(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b, BN_ULONG *l, int n2,
758                  BN_ULONG *t)
759         {
760         int i,n;
761         int c1,c2;
762         int neg,oneg,zero;
763         BN_ULONG ll,lc,*lp,*mp;

765 # ifdef BN_COUNT
766         fprintf(stderr," bn_mul_high %d * %d\n",n2,n2);
767 # endif
768         n=n2/2;

770         /* Calculate (al-ah)*(bh-bl) */
771         neg=zero=0;
772         c1=bn_cmp_words(&(a[0]),&(a[n]),n);
773         c2=bn_cmp_words(&(b[n]),&(b[0]),n);
774         switch (c1*3+c2)
775                 {
776         case -4:
777                 bn_sub_words(&(r[0]),&(a[n]),&(a[0]),n);
778                 bn_sub_words(&(r[n]),&(b[0]),&(b[n]),n);
779                 break;
780         case -3:
781                 zero=1;
782                 break;
783         case -2:
784                 bn_sub_words(&(r[0]),&(a[n]),&(a[0]),n);
785                 bn_sub_words(&(r[n]),&(b[n]),&(b[0]),n);
786                 neg=1;
787                 break;
```

```
788         case -1:
789         case 0:
790         case 1:
791                 zero=1;
792                 break;
793         case 2:
794                 bn_sub_words(&(r[0]),&(a[0]),&(a[n]),n);
795                 bn_sub_words(&(r[n]),&(b[0]),&(b[n]),n);
796                 neg=1;
797                 break;
798         case 3:
799                 zero=1;
800                 break;
801         case 4:
802                 bn_sub_words(&(r[0]),&(a[0]),&(a[n]),n);
803                 bn_sub_words(&(r[n]),&(b[n]),&(b[0]),n);
804                 break;
805                 }

807         oneg=neg;
808         /* t[10] = (a[0]-a[1])*(b[1]-b[0]) */
809         /* r[10] = (a[1]*b[1]) */
810 # ifdef BN_MUL_COMBA
811         if (n == 8)
812                 {
813                 bn_mul_comba8(&(t[0]),&(r[0]),&(r[n]));
814                 bn_mul_comba8(r,&(a[n]),&(b[n]));
815                 }
816         else
817 # endif
818                 {
819                 bn_mul_recursive(&(t[0]),&(r[0]),&(r[n]),n,0,0,&(t[n2]));
820                 bn_mul_recursive(r,&(a[n]),&(b[n]),n,0,0,&(t[n2]));
821                 }

823         /* s0 == low(al*bl)
824          * s1 == low(ah*bh)+low((al-ah)*(bh-bl))+low(al*bl)+high(al*bl)
825          * We know s0 and s1 so the only unknown is high(al*bl)
826          * high(al*bl) == s1 - low(ah*bh+s0+(al-ah)*(bh-bl))
827          * high(al*bl) == s1 - (r[0]+l[0]+t[0])
828          */
829         if (l != NULL)
830                 {
831                 lp= &(t[n2+n]);
832                 c1=(int)(bn_add_words(lp,&(r[0]),&(l[0]),n));
833                 }
834         else
835                 {
836                 c1=0;
837                 lp= &(r[0]);
838                 }

840         if (neg)
841                 neg=(int)(bn_sub_words(&(t[n2]),lp,&(t[0]),n));
842         else
843                 {
844                 bn_add_words(&(t[n2]),lp,&(t[0]),n);
845                 neg=0;
846                 }

848         if (l != NULL)
849                 {
850                 bn_sub_words(&(t[n2+n]),&(l[n]),&(t[n2]),n);
851                 }
852         else
853                 {
```



```
854                 lp= &(t[n2+n]);
855                 mp= &(t[n2]);
856                 for (i=0; i<n; i++)
857                         lp[i]=((~mp[i])+1)&BN_MASK2;
858                 }

860         /* s[0] = low(al*bl)
861          * t[3] = high(al*bl)
862          * t[10] = (a[0]-a[1])*(b[1]-b[0]) neg is the sign
863          * r[10] = (a[1]*b[1])
864          */
865         /* R[10] = al*bl
866          * R[21] = al*bl + ah*bh + (a[0]-a[1])*(b[1]-b[0])
867          * R[32] = ah*bh
868          */
869         /* R[1]=t[3]+l[0]+r[0](+-)t[0] (have carry/borrow)
870          * R[2]=r[0]+t[3]+r[1](+-)t[1] (have carry/borrow)
871          * R[3]=r[1]+(carry/borrow)
872          */
873         if (l != NULL)
874                 {
875                 lp= &(t[n2]);
876                 c1= (int)(bn_add_words(lp,&(t[n2+n]),&(l[0]),n));
877                 }
878         else
879                 {
880                 lp= &(t[n2+n]);
881                 c1=0;
882                 }
883         c1+=(int)(bn_add_words(&(t[n2]),lp,  &(r[0]),n));
884         if (oneg)
885                 c1-=(int)(bn_sub_words(&(t[n2]),&(t[n2]),&(t[0]),n));
886         else
887                 c1+=(int)(bn_add_words(&(t[n2]),&(t[n2]),&(t[0]),n));

889         c2 =(int)(bn_add_words(&(r[0]),&(r[0]),&(t[n2+n]),n));
890         c2+=(int)(bn_add_words(&(r[0]),&(r[0]),&(r[n]),n));
891         if (oneg)
892                 c2-=(int)(bn_sub_words(&(r[0]),&(r[0]),&(t[n]),n));
893         else
894                 c2+=(int)(bn_add_words(&(r[0]),&(r[0]),&(t[n]),n));

896         if (c1 != 0) /* Add starting at r[0], could be +ve or -ve */
897                 {
898                 i=0;
899                 if (c1 > 0)
900                         {
901                         lc=c1;
902                         do      {
903                                 ll=(r[i]+lc)&BN_MASK2;
904                                 r[i++]=ll;
905                                 lc=(lc > ll);
906                                 } while (lc);
907                         }
908                 else
909                         {
910                         lc= -c1;
911                         do      {
912                                 ll=r[i];
913                                 r[i++]=(ll-lc)&BN_MASK2;
914                                 lc=(lc > ll);
915                                 } while (lc);
916                         }
917                 }
918         if (c2 != 0) /* Add starting at r[1] */
919                 {
```

```
920                 i=n;
921                 if (c2 > 0)
922                         {
923                         lc=c2;
924                         do      {
925                                 ll=(r[i]+lc)&BN_MASK2;
926                                 r[i++]=ll;
927                                 lc=(lc > ll);
928                                 } while (lc);
929                         }
930                 else
931                         {
932                         lc= -c2;
933                         do      {
934                                 ll=r[i];
935                                 r[i++]=(ll-lc)&BN_MASK2;
936                                 lc=(lc > ll);
937                                 } while (lc);
938                         }
939                 }
940         }
941 #endif /* BN_RECURSION */

943 int BN_mul(BIGNUM *r, const BIGNUM *a, const BIGNUM *b, BN_CTX *ctx)
944         {
945         int ret=0;
946         int top,al,bl;
947         BIGNUM *rr;
948 #if defined(BN_MUL_COMBA) || defined(BN_RECURSION)
949         int i;
950 #endif
951 #ifdef BN_RECURSION
952         BIGNUM *t=NULL;
953         int j=0,k;
954 #endif

956 #ifdef BN_COUNT
957         fprintf(stderr,"BN_mul %d * %d\n",a->top,b->top);
958 #endif

960         bn_check_top(a);
961         bn_check_top(b);
962         bn_check_top(r);

964         al=a->top;
965         bl=b->top;

967         if ((al == 0) || (bl == 0))
968                 {
969                 BN_zero(r);
970                 return(1);
971                 }
972         top=al+bl;

974         BN_CTX_start(ctx);
975         if ((r == a) || (r == b))
976                 {
977                 if ((rr = BN_CTX_get(ctx)) == NULL) goto err;
978                 }
979         else
980                 rr = r;
981         rr->neg=a->neg^b->neg;

983 #if defined(BN_MUL_COMBA) || defined(BN_RECURSION)
984         i = al-bl;
985 #endif
```



```
986 #ifdef BN_MUL_COMBA
987         if (i == 0)
988                 {
989 # if 0
990                 if (al == 4)
991                         {
992                         if (bn_wexpand(rr,8) == NULL) goto err;
993                         rr->top=8;
994                         bn_mul_comba4(rr->d,a->d,b->d);
995                         goto end;
996                         }
997 # endif
998                 if (al == 8)
999                         {
1000                        if (bn_wexpand(rr,16) == NULL) goto err;
1001                        rr->top=16;
1002                        bn_mul_comba8(rr->d,a->d,b->d);
1003                        goto end;
1004                        }
1005                }
1006 #endif /* BN_MUL_COMBA */
1007 #ifdef BN_RECURSION
1008        if ((al >= BN_MULL_SIZE_NORMAL) && (bl >= BN_MULL_SIZE_NORMAL))
1009                {
1010                if (i >= -1 && i <= 1)
1011                        {
1012                        /* Find out the power of two lower or equal
1013                           to the longest of the two numbers */
1014                        if (i >= 0)
1015                                {
1016                                j = BN_num_bits_word((BN_ULONG)al);
1017                                }
1018                        if (i == -1)
1019                                {
1020                                j = BN_num_bits_word((BN_ULONG)bl);
1021                                }
1022                        j = 1<<(j-1);
1023                        assert(j <= al || j <= bl);
1024                        k = j+j;
1025                        t = BN_CTX_get(ctx);
1026                        if (t == NULL)
1027                                goto err;
1028                        if (al > j || bl > j)
1029                                {
1030                                if (bn_wexpand(t,k*4) == NULL) goto err;
1031                                if (bn_wexpand(rr,k*4) == NULL) goto err;
1032                                bn_mul_part_recursive(rr->d,a->d,b->d,
1033                                        j,al-j,bl-j,t->d);
1034                                }
1035                        else    /* al <= j || bl <= j */
1036                                {
1037                                if (bn_wexpand(t,k*2) == NULL) goto err;
1038                                if (bn_wexpand(rr,k*2) == NULL) goto err;
1039                                bn_mul_recursive(rr->d,a->d,b->d,
1040                                        j,al-j,bl-j,t->d);
1041                                }
1042                        rr->top=top;
1043                        goto end;
1044                        }
1045 #if 0
1046                if (i == 1 && !BN_get_flags(b,BN_FLG_STATIC_DATA))
1047                        {
1048                        BIGNUM *tmp_bn = (BIGNUM *)b;
1049                        if (bn_wexpand(tmp_bn,al) == NULL) goto err;
1050                        tmp_bn->d[bl]=0;
1051                        bl++;
```

```
1052                        i--;
1053                        }
1054                else if (i == -1 && !BN_get_flags(a,BN_FLG_STATIC_DATA))
1055                        {
1056                        BIGNUM *tmp_bn = (BIGNUM *)a;
1057                        if (bn_wexpand(tmp_bn,bl) == NULL) goto err;
1058                        tmp_bn->d[al]=0;
1059                        al++;
1060                        i++;
1061                        }
1062                if (i == 0)
1063                        {
1064                        /* symmetric and > 4 */
1065                        /* 16 or larger */
1066                        j=BN_num_bits_word((BN_ULONG)al);
1067                        j=1<<(j-1);
1068                        k=j+j;
1069                        t = BN_CTX_get(ctx);
1070                        if (al == j) /* exact multiple */
1071                                {
1072                                if (bn_wexpand(t,k*2) == NULL) goto err;
1073                                if (bn_wexpand(rr,k*2) == NULL) goto err;
1074                                bn_mul_recursive(rr->d,a->d,b->d,al,t->d);
1075                                }
1076                        else
1077                                {
1078                                if (bn_wexpand(t,k*4) == NULL) goto err;
1079                                if (bn_wexpand(rr,k*4) == NULL) goto err;
1080                                bn_mul_part_recursive(rr->d,a->d,b->d,al-j,j,t->
1081                                }
1082                        rr->top=top;
1083                        goto end;
1084                        }
1085 #endif
1086                }
1087 #endif /* BN_RECURSION */
1088        if (bn_wexpand(rr,top) == NULL) goto err;
1089        rr->top=top;
1090        bn_mul_normal(rr->d,a->d,al,b->d,bl);

1092 #if defined(BN_MUL_COMBA) || defined(BN_RECURSION)
1093 end:
1094 #endif
1095        bn_correct_top(rr);
1096        if (r != rr) BN_copy(r,rr);
1097        ret=1;
1098 err:
1099        bn_check_top(r);
1100        BN_CTX_end(ctx);
1101        return(ret);
1102        }

1104 void bn_mul_normal(BN_ULONG *r, BN_ULONG *a, int na, BN_ULONG *b, int nb)
1105        {
1106        BN_ULONG *rr;

1108 #ifdef BN_COUNT
1109        fprintf(stderr," bn_mul_normal %d * %d\n",na,nb);
1110 #endif

1112        if (na < nb)
1113                {
1114                int itmp;
1115                BN_ULONG *ltmp;

1117                itmp=na; na=nb; nb=itmp;
```



```
1118                ltmp=a;   a=b;   b=ltmp;

1120                }
1121        rr= &(r[na]);
1122        if (nb <= 0)
1123                {
1124                (void)bn_mul_words(r,a,na,0);
1125                return;
1126                }
1127        else
1128                rr[0]=bn_mul_words(r,a,na,b[0]);

1130        for (;;)
1131                {
1132                if (--nb <= 0) return;
1133                rr[1]=bn_mul_add_words(&(r[1]),a,na,b[1]);
1134                if (--nb <= 0) return;
1135                rr[2]=bn_mul_add_words(&(r[2]),a,na,b[2]);
1136                if (--nb <= 0) return;
1137                rr[3]=bn_mul_add_words(&(r[3]),a,na,b[3]);
1138                if (--nb <= 0) return;
1139                rr[4]=bn_mul_add_words(&(r[4]),a,na,b[4]);
1140                rr+=4;
1141                r+=4;
1142                b+=4;
1143                }
1144        }

1146 void bn_mul_low_normal(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b, int n)
1147        {
1148 #ifdef BN_COUNT
1149        fprintf(stderr," bn_mul_low_normal %d * %d\n",n,n);
1150 #endif
1151        bn_mul_words(r,a,n,b[0]);

1153        for (;;)
1154                {
1155                if (--n <= 0) return;
1156                bn_mul_add_words(&(r[1]),a,n,b[1]);
1157                if (--n <= 0) return;
1158                bn_mul_add_words(&(r[2]),a,n,b[2]);
1159                if (--n <= 0) return;
1160                bn_mul_add_words(&(r[3]),a,n,b[3]);
1161                if (--n <= 0) return;
1162                bn_mul_add_words(&(r[4]),a,n,b[4]);
1163                r+=4;
1164                b+=4;
1165                }
1166        }
1167 #endif /* ! codereview */
```

```
**********************************************************
   33050 Wed Aug 13 19:52:16 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_nist.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_nist.c */
/*
 * Written by Nils Larsch for the OpenSSL project
 */
/* ====================================================================
 * Copyright (c) 1998-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "bn_lcl.h"
#include "cryptlib.h"
```



```
#define BN_NIST_192_TOP (192+BN_BITS2-1)/BN_BITS2
#define BN_NIST_224_TOP (224+BN_BITS2-1)/BN_BITS2
#define BN_NIST_256_TOP (256+BN_BITS2-1)/BN_BITS2
#define BN_NIST_384_TOP (384+BN_BITS2-1)/BN_BITS2
#define BN_NIST_521_TOP (521+BN_BITS2-1)/BN_BITS2

/* pre-computed tables are "carry-less" values of modulus*(i+1) */
#if BN_BITS2 == 64
static const BN_ULONG _nist_p_192[][BN_NIST_192_TOP] = {
        {0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFEULL,0xFFFFFFFFFFFFFFFFULL},
        {0xFFFFFFFFFFFFFFFEULL,0xFFFFFFFFFFFFFFFDULL,0xFFFFFFFFFFFFFFFFULL},
        {0xFFFFFFFFFFFFFFFDULL,0xFFFFFFFFFFFFFFFCULL,0xFFFFFFFFFFFFFFFFULL}
        };
static const BN_ULONG _nist_p_192_sqr[] = {
        0x0000000000000001ULL,0x0000000000000002ULL,0x0000000000000001ULL,
        0xFFFFFFFFFFFFFFFEULL,0xFFFFFFFFFFFFFFFDULL,0xFFFFFFFFFFFFFFFFULL
        };
static const BN_ULONG _nist_p_224[][BN_NIST_224_TOP] = {
        {0x0000000000000001ULL,0xFFFFFFFF00000000ULL,
         0xFFFFFFFFFFFFFFFFULL,0x00000000FFFFFFFFULL},
        {0x0000000000000002ULL,0xFFFFFFFE00000000ULL,
         0xFFFFFFFFFFFFFFFFULL,0x00000001FFFFFFFFULL} /* this one is "carry-full
        };
static const BN_ULONG _nist_p_224_sqr[] = {
        0x0000000000000001ULL,0xFFFFFFFE00000000ULL,
        0xFFFFFFFFFFFFFFFFULL,0x0000000200000000ULL,
        0x0000000000000000ULL,0xFFFFFFFFFFFFFFFEULL,
        0xFFFFFFFFFFFFFFFFULL
        };
static const BN_ULONG _nist_p_256[][BN_NIST_256_TOP] = {
        {0xFFFFFFFFFFFFFFFFULL,0x00000000FFFFFFFFULL,
         0x0000000000000000ULL,0xFFFFFFFF00000001ULL},
        {0xFFFFFFFFFFFFFFFEULL,0x00000001FFFFFFFFULL,
         0x0000000000000000ULL,0xFFFFFFFE00000002ULL},
        {0xFFFFFFFFFFFFFFFDULL,0x00000002FFFFFFFFULL,
         0x0000000000000000ULL,0xFFFFFFFD00000003ULL},
        {0xFFFFFFFFFFFFFFFCULL,0x00000003FFFFFFFFULL,
         0x0000000000000000ULL,0xFFFFFFFC00000004ULL},
        {0xFFFFFFFFFFFFFFFBULL,0x00000004FFFFFFFFULL,
         0x0000000000000000ULL,0xFFFFFFFB00000005ULL},
        };
static const BN_ULONG _nist_p_256_sqr[] = {
        0x0000000000000001ULL,0xFFFFFFFE00000000ULL,
        0xFFFFFFFFFFFFFFFFULL,0x00000001FFFFFFFEULL,
        0x00000001FFFFFFFEULL,0x00000001FFFFFFFEULL,
        0xFFFFFFFE00000001ULL,0xFFFFFFFE00000002ULL
        };
static const BN_ULONG _nist_p_384[][BN_NIST_384_TOP] = {
        {0x00000000FFFFFFFFULL,0xFFFFFFFF00000000ULL,0xFFFFFFFFFFFFFFFEULL,
         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL},
        {0x00000001FFFFFFFEULL,0xFFFFFFFE00000000ULL,0xFFFFFFFFFFFFFFFFULL,
         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL},
        {0x00000002FFFFFFFDULL,0xFFFFFFFD00000000ULL,0xFFFFFFFFFFFFFFFCULL,
         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL},
        {0x00000003FFFFFFFCULL,0xFFFFFFFC00000000ULL,0xFFFFFFFFFFFFFFFBULL,
         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL},
        {0x00000004FFFFFFFBULL,0xFFFFFFFB00000000ULL,0xFFFFFFFFFFFFFFFAULL,
         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL},
        };
static const BN_ULONG _nist_p_384_sqr[] = {
        0xFFFFFFFE00000001ULL,0x0000000200000000ULL,0xFFFFFFFE00000000ULL,
        0x0000000200000000ULL,0x0000000000000001ULL,0x0000000000000000ULL,
        0x00000001FFFFFFFEULL,0xFFFFFFFE00000000ULL,0xFFFFFFFFFFFFFFFDULL,
        0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL
        };
```

```
128 static const BN_ULONG _nist_p_521[] =
129         {0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,
130         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,
131         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,
132         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,
133         0x00000000000001FFULL};
134 static const BN_ULONG _nist_p_521_sqr[] = {
135         0x0000000000000001ULL,0x0000000000000000ULL,0x0000000000000000ULL,
136         0x0000000000000000ULL,0x0000000000000000ULL,0x0000000000000000ULL,
137         0x0000000000000000ULL,0x0000000000000000ULL,0xFFFFFFFFFFFFFC00ULL,
138         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,
139         0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,0xFFFFFFFFFFFFFFFFULL,
140         0xFFFFFFFFFFFFFFFFULL,0x000000000003FFFFULL
141         };
142 #elif BN_BITS2 == 32
143 static const BN_ULONG _nist_p_192[][BN_NIST_192_TOP] = {
144         {0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFE,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
145         {0xFFFFFFFE,0xFFFFFFFF,0xFFFFFFFD,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
146         {0xFFFFFFFD,0xFFFFFFFF,0xFFFFFFFC,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF}
147         };
148 static const BN_ULONG _nist_p_192_sqr[] = {
149         0x00000001,0x00000000,0x00000002,0x00000000,0x00000001,0x00000000,
150         0xFFFFFFFE,0xFFFFFFFF,0xFFFFFFFD,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
151         };
152 static const BN_ULONG _nist_p_224[][BN_NIST_224_TOP] = {
153         {0x00000001,0x00000000,0x00000000,0xFFFFFFFF,
154          0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
155         {0x00000002,0x00000000,0x00000000,0xFFFFFFFE,
156          0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF}
157         };
158 static const BN_ULONG _nist_p_224_sqr[] = {
159         0x00000001,0x00000000,0x00000000,0xFFFFFFFE,
160         0xFFFFFFFF,0xFFFFFFFF,0x00000000,0x00000002,
161         0x00000000,0x00000000,0xFFFFFFFE,0xFFFFFFFF,
162         0xFFFFFFFF,0xFFFFFFFF
163         };
164 static const BN_ULONG _nist_p_256[][BN_NIST_256_TOP] = {
165         {0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0x00000000,
166          0x00000000,0x00000000,0x00000001,0xFFFFFFFF},
167         {0xFFFFFFFE,0xFFFFFFFF,0xFFFFFFFF,0x00000001,
168          0x00000000,0x00000000,0x00000002,0xFFFFFFFE},
169         {0xFFFFFFFD,0xFFFFFFFF,0xFFFFFFFF,0x00000002,
170          0x00000000,0x00000000,0x00000003,0xFFFFFFFD},
171         {0xFFFFFFFC,0xFFFFFFFF,0xFFFFFFFF,0x00000003,
172          0x00000000,0x00000000,0x00000004,0xFFFFFFFC},
173         {0xFFFFFFFB,0xFFFFFFFF,0xFFFFFFFF,0x00000004,
174          0x00000000,0x00000000,0x00000005,0xFFFFFFFB},
175         };
176 static const BN_ULONG _nist_p_256_sqr[] = {
177         0x00000001,0x00000000,0x00000000,0xFFFFFFFE,
178         0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFE,0x00000001,
179         0xFFFFFFFE,0x00000001,0xFFFFFFFE,0x00000001,
180         0x00000001,0xFFFFFFFE,0x00000002,0xFFFFFFFE
181         };
182 static const BN_ULONG _nist_p_384[][BN_NIST_384_TOP] = {
183         {0xFFFFFFFF,0x00000000,0x00000000,0xFFFFFFFF,0xFFFFFFFE,0xFFFFFFFF,
184          0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
185         {0xFFFFFFFE,0x00000001,0x00000000,0xFFFFFFFE,0xFFFFFFFD,0xFFFFFFFF,
186          0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
187         {0xFFFFFFFD,0x00000002,0x00000000,0xFFFFFFFD,0xFFFFFFFC,0xFFFFFFFF,
188          0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
189         {0xFFFFFFFC,0x00000003,0x00000000,0xFFFFFFFC,0xFFFFFFFB,0xFFFFFFFF,
190          0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
191         {0xFFFFFFFB,0x00000004,0x00000000,0xFFFFFFFB,0xFFFFFFFA,0xFFFFFFFF,
192          0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF},
193         };
```



```
194 static const BN_ULONG _nist_p_384_sqr[] = {
195         0x00000001,0xFFFFFFFE,0x00000000,0x00000002,0x00000000,0xFFFFFFFE,
196         0x00000000,0x00000002,0x00000001,0x00000000,0x00000000,0x00000000,
197         0xFFFFFFFE,0x00000001,0x00000000,0xFFFFFFFE,0xFFFFFFFD,0xFFFFFFFF,
198         0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF
199         };
200 static const BN_ULONG _nist_p_521[] = {0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,
201         0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,
202         0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,
203         0xFFFFFFFF,0x000001FF};
204 static const BN_ULONG _nist_p_521_sqr[] = {
205         0x00000001,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
206         0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,0x00000000,
207         0x00000000,0x00000000,0x00000000,0x00000000,0xFFFFFC00,0xFFFFFFFF,
208         0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,
209         0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,
210         0xFFFFFFFF,0xFFFFFFFF,0x0003FFFF
211         };
212 #else
213 #error "unsupported BN_BITS2"
214 #endif


217 static const BIGNUM _bignum_nist_p_192 =
218         {
219         (BN_ULONG *)_nist_p_192[0],
220         BN_NIST_192_TOP,
221         BN_NIST_192_TOP,
222         0,
223         BN_FLG_STATIC_DATA
224         };

226 static const BIGNUM _bignum_nist_p_224 =
227         {
228         (BN_ULONG *)_nist_p_224[0],
229         BN_NIST_224_TOP,
230         BN_NIST_224_TOP,
231         0,
232         BN_FLG_STATIC_DATA
233         };

235 static const BIGNUM _bignum_nist_p_256 =
236         {
237         (BN_ULONG *)_nist_p_256[0],
238         BN_NIST_256_TOP,
239         BN_NIST_256_TOP,
240         0,
241         BN_FLG_STATIC_DATA
242         };

244 static const BIGNUM _bignum_nist_p_384 =
245         {
246         (BN_ULONG *)_nist_p_384[0],
247         BN_NIST_384_TOP,
248         BN_NIST_384_TOP,
249         0,
250         BN_FLG_STATIC_DATA
251         };

253 static const BIGNUM _bignum_nist_p_521 =
254         {
255         (BN_ULONG *)_nist_p_521,
256         BN_NIST_521_TOP,
257         BN_NIST_521_TOP,
258         0,
259         BN_FLG_STATIC_DATA
```

```
260         };


263 const BIGNUM *BN_get0_nist_prime_192(void)
264         {
265         return &_bignum_nist_p_192;
266         }

268 const BIGNUM *BN_get0_nist_prime_224(void)
269         {
270         return &_bignum_nist_p_224;
271         }

273 const BIGNUM *BN_get0_nist_prime_256(void)
274         {
275         return &_bignum_nist_p_256;
276         }

278 const BIGNUM *BN_get0_nist_prime_384(void)
279         {
280         return &_bignum_nist_p_384;
281         }

283 const BIGNUM *BN_get0_nist_prime_521(void)
284         {
285         return &_bignum_nist_p_521;
286         }


289 static void nist_cp_bn_0(BN_ULONG *dst, const BN_ULONG *src, int top, int max)
290         {
291         int i;

293 #ifdef BN_DEBUG
294         OPENSSL_assert(top <= max);
295 #endif
296         for (i = 0; i < top; i++)
297                 dst[i] = src[i];
298         for (; i < max; i++)
299                 dst[i] = 0;
300         }

302 static void nist_cp_bn(BN_ULONG *dst, const BN_ULONG *src, int top)
303         {
304         int i;

306         for (i = 0; i < top; i++)
307                 dst[i] = src[i];
308         }

310 #if BN_BITS2 == 64
311 #define bn_cp_64(to, n, from, m)        (to)[n] = (m>=0)?((from)[m]):0;
312 #define bn_64_set_0(to, n)              (to)[n] = (BN_ULONG)0;
313 /*
314  * two following macros are implemented under assumption that they
315  * are called in a sequence with *ascending* n, i.e. as they are...
316  */
317 #define bn_cp_32_naked(to, n, from, m)  (((n)&1)?(to[(n)/2]|=((m)&1)?(from[(m)/2
318                                                 :(to[(n)/2] =((m)&1)?(from[(m)/2
319 #define bn_32_set_0(to, n)              (((n)&1)?(to[(n)/2]&=BN_MASK2l):(to[(n)/
320 #define bn_cp_32(to,n,from,m)           ((m)>=0)?bn_cp_32_naked(to,n,from,m):bn_
321 # if defined(L_ENDIAN)
322 #  if defined(__arch64__)
323 #   define NIST_INT64 long
324 #  else
325 #   define NIST_INT64 long long
```



```
326 #  endif
327 # endif
328 #else
329 #define bn_cp_64(to, n, from, m) \
330         { \
331         bn_cp_32(to, (n)*2, from, (m)*2); \
332         bn_cp_32(to, (n)*2+1, from, (m)*2+1); \
333         }
334 #define bn_64_set_0(to, n) \
335         { \
336         bn_32_set_0(to, (n)*2); \
337         bn_32_set_0(to, (n)*2+1); \
338         }
339 #define bn_cp_32(to, n, from, m)        (to)[n] = (m>=0)?((from)[m]):0;
340 #define bn_32_set_0(to, n)              (to)[n] = (BN_ULONG)0;
341 # if defined(_WIN32) && !defined(__GNUC__)
342 #  define NIST_INT64 __int64
343 # elif defined(BN_LLONG)
344 #  define NIST_INT64 long long
345 # endif
346 #endif /* BN_BITS2 != 64 */

348 #define nist_set_192(to, from, a1, a2, a3) \
349         { \
350         bn_cp_64(to, 0, from, (a3) - 3) \
351         bn_cp_64(to, 1, from, (a2) - 3) \
352         bn_cp_64(to, 2, from, (a1) - 3) \
353         }

355 int BN_nist_mod_192(BIGNUM *r, const BIGNUM *a, const BIGNUM *field,
356         BN_CTX *ctx)
357         {
358         int      top = a->top, i;
359         int      carry;
360         register BN_ULONG *r_d, *a_d = a->d;
361         union   {
362                 BN_ULONG        bn[BN_NIST_192_TOP];
363                 unsigned int    ui[BN_NIST_192_TOP*sizeof(BN_ULONG)/sizeof(unsig
364                 } buf;
365         BN_ULONG c_d[BN_NIST_192_TOP],
366                 *res;
367         PTR_SIZE_INT mask;
368         static const BIGNUM _bignum_nist_p_192_sqr = {
369                 (BN_ULONG *)_nist_p_192_sqr,
370                 sizeof(_nist_p_192_sqr)/sizeof(_nist_p_192_sqr[0]),
371                 sizeof(_nist_p_192_sqr)/sizeof(_nist_p_192_sqr[0]),
372                 0,BN_FLG_STATIC_DATA };

374         field = &_bignum_nist_p_192; /* just to make sure */

376         if (BN_is_negative(a) || BN_ucmp(a,&_bignum_nist_p_192_sqr)>=0)
377                 return BN_nnmod(r, a, field, ctx);

379         i = BN_ucmp(field, a);
380         if (i == 0)
381                 {
382                 BN_zero(r);
383                 return 1;
384                 }
385         else if (i > 0)
386                 return (r == a) ? 1 : (BN_copy(r ,a) != NULL);

388         if (r != a)
389                 {
390                 if (!bn_wexpand(r, BN_NIST_192_TOP))
391                         return 0;
```

```
392                 r_d = r->d;
393                 nist_cp_bn(r_d, a_d, BN_NIST_192_TOP);
394                 }
395         else
396                 r_d = a_d;

398         nist_cp_bn_0(buf.bn, a_d + BN_NIST_192_TOP, top - BN_NIST_192_TOP, BN_NI

400 #if defined(NIST_INT64)
401         {
402         NIST_INT64              acc;    /* accumulator */
403         unsigned int            *rp=(unsigned int *)r_d;
404         const unsigned int      *bp=(const unsigned int *)buf.ui;

406         acc  = rp[0];   acc += bp[3*2-6];
407                         acc += bp[5*2-6]; rp[0] = (unsigned int)acc; acc >>= 32;

409         acc += rp[1];   acc += bp[3*2-5];
410                         acc += bp[5*2-5]; rp[1] = (unsigned int)acc; acc >>= 32;

412         acc += rp[2];   acc += bp[3*2-6];
413                         acc += bp[4*2-6];
414                         acc += bp[5*2-6]; rp[2] = (unsigned int)acc; acc >>= 32;

416         acc += rp[3];   acc += bp[3*2-5];
417                         acc += bp[4*2-5];
418                         acc += bp[5*2-5]; rp[3] = (unsigned int)acc; acc >>= 32;

420         acc += rp[4];   acc += bp[4*2-6];
421                         acc += bp[5*2-6]; rp[4] = (unsigned int)acc; acc >>= 32;

423         acc += rp[5];   acc += bp[4*2-5];
424                         acc += bp[5*2-5]; rp[5] = (unsigned int)acc;

426         carry = (int)(acc>>32);
427         }
428 #else
429         {
430         BN_ULONG t_d[BN_NIST_192_TOP];

432         nist_set_192(t_d, buf.bn, 0, 3, 3);
433         carry = (int)bn_add_words(r_d, r_d, t_d, BN_NIST_192_TOP);
434         nist_set_192(t_d, buf.bn, 4, 4, 0);
435         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_192_TOP);
436         nist_set_192(t_d, buf.bn, 5, 5, 5)
437         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_192_TOP);
438         }
439 #endif
440         if (carry > 0)
441                 carry = (int)bn_sub_words(r_d,r_d,_nist_p_192[carry-1],BN_NIST_1
442         else
443                 carry = 1;

445         /*
446          * we need 'if (carry==0 || result>=modulus) result-=modulus;'
447          * as comparison implies subtraction, we can write
448          * 'tmp=result-modulus; if (!carry || !borrow) result=tmp;'
449          * this is what happens below, but without explicit if:-) a.
450          */
451         mask  = 0-(PTR_SIZE_INT)bn_sub_words(c_d,r_d,_nist_p_192[0],BN_NIST_192_
452         mask &= 0-(PTR_SIZE_INT)carry;
453         res   = c_d;
454         res   = (BN_ULONG *)
455          (((PTR_SIZE_INT)res&~mask) | ((PTR_SIZE_INT)r_d&mask));
456         nist_cp_bn(r_d, res, BN_NIST_192_TOP);
457         r->top = BN_NIST_192_TOP;
```



```
458         bn_correct_top(r);

460         return 1;
461         }

463 typedef BN_ULONG (*bn_addsub_f)(BN_ULONG *,const BN_ULONG *,const BN_ULONG *,int

465 #define nist_set_224(to, from, a1, a2, a3, a4, a5, a6, a7) \
466         { \
467         bn_cp_32(to, 0, from, (a7) - 7) \
468         bn_cp_32(to, 1, from, (a6) - 7) \
469         bn_cp_32(to, 2, from, (a5) - 7) \
470         bn_cp_32(to, 3, from, (a4) - 7) \
471         bn_cp_32(to, 4, from, (a3) - 7) \
472         bn_cp_32(to, 5, from, (a2) - 7) \
473         bn_cp_32(to, 6, from, (a1) - 7) \
474         }

476 int BN_nist_mod_224(BIGNUM *r, const BIGNUM *a, const BIGNUM *field,
477         BN_CTX *ctx)
478         {
479         int     top = a->top, i;
480         int     carry;
481         BN_ULONG *r_d, *a_d = a->d;
482         union   {
483                 BN_ULONG        bn[BN_NIST_224_TOP];
484                 unsigned int    ui[BN_NIST_224_TOP*sizeof(BN_ULONG)/sizeof(unsig
485                 } buf;
486         BN_ULONG c_d[BN_NIST_224_TOP],
487                 *res;
488         PTR_SIZE_INT mask;
489         union { bn_addsub_f f; PTR_SIZE_INT p; } u;
490         static const BIGNUM _bignum_nist_p_224_sqr = {
491                 (BN_ULONG *)_nist_p_224_sqr,
492                 sizeof(_nist_p_224_sqr)/sizeof(_nist_p_224_sqr[0]),
493                 sizeof(_nist_p_224_sqr)/sizeof(_nist_p_224_sqr[0]),
494                 0,BN_FLG_STATIC_DATA };


497         field = &_bignum_nist_p_224; /* just to make sure */

499         if (BN_is_negative(a) || BN_ucmp(a,&_bignum_nist_p_224_sqr)>=0)
500                 return BN_nnmod(r, a, field, ctx);

502         i = BN_ucmp(field, a);
503         if (i == 0)
504                 {
505                 BN_zero(r);
506                 return 1;
507                 }
508         else if (i > 0)
509                 return (r == a)? 1 : (BN_copy(r ,a) != NULL);

511         if (r != a)
512                 {
513                 if (!bn_wexpand(r, BN_NIST_224_TOP))
514                         return 0;
515                 r_d = r->d;
516                 nist_cp_bn(r_d, a_d, BN_NIST_224_TOP);
517                 }
518         else
519                 r_d = a_d;

521 #if BN_BITS2==64
522         /* copy upper 256 bits of 448 bit number ... */
523         nist_cp_bn_0(c_d, a_d + (BN_NIST_224_TOP-1), top - (BN_NIST_224_TOP-1),
```

```
524         /* ... and right shift by 32 to obtain upper 224 bits */
525         nist_set_224(buf.bn, c_d, 14, 13, 12, 11, 10, 9, 8);
526         /* truncate lower part to 224 bits too */
527         r_d[BN_NIST_224_TOP-1] &= BN_MASK2l;
528 #else
529         nist_cp_bn_0(buf.bn, a_d + BN_NIST_224_TOP, top - BN_NIST_224_TOP, BN_NI
530 #endif

532 #if defined(NIST_INT64) && BN_BITS2!=64
533         {
534         NIST_INT64              acc;    /* accumulator */
535         unsigned int            *rp=(unsigned int *)r_d;
536         const unsigned int      *bp=(const unsigned int *)buf.ui;

538         acc  = rp[0];   acc -= bp[7-7];
539                         acc -= bp[11-7]; rp[0] = (unsigned int)acc; acc >>= 32;

541         acc += rp[1];   acc -= bp[8-7];
542                         acc -= bp[12-7]; rp[1] = (unsigned int)acc; acc >>= 32;

544         acc += rp[2];   acc -= bp[9-7];
545                         acc -= bp[13-7]; rp[2] = (unsigned int)acc; acc >>= 32;

547         acc += rp[3];   acc += bp[7-7];
548                         acc += bp[11-7];
549                         acc -= bp[10-7]; rp[3] = (unsigned int)acc; acc>>= 32;

551         acc += rp[4];   acc += bp[8-7];
552                         acc += bp[12-7];
553                         acc -= bp[11-7]; rp[4] = (unsigned int)acc; acc >>= 32;

555         acc += rp[5];   acc += bp[9-7];
556                         acc += bp[13-7];
557                         acc -= bp[12-7]; rp[5] = (unsigned int)acc; acc >>= 32;

559         acc += rp[6];   acc += bp[10-7];
560                         acc -= bp[13-7]; rp[6] = (unsigned int)acc;

562         carry = (int)(acc>>32);
563 # if BN_BITS2==64
564         rp[7] = carry;
565 # endif
566         }
567 #else
568         {
569         BN_ULONG t_d[BN_NIST_224_TOP];

571         nist_set_224(t_d, buf.bn, 10, 9, 8, 7, 0, 0, 0);
572         carry = (int)bn_add_words(r_d, r_d, t_d, BN_NIST_224_TOP);
573         nist_set_224(t_d, buf.bn, 0, 13, 12, 11, 0, 0, 0);
574         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_224_TOP);
575         nist_set_224(t_d, buf.bn, 13, 12, 11, 10, 9, 8, 7);
576         carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_224_TOP);
577         nist_set_224(t_d, buf.bn, 0, 0, 0, 0, 13, 12, 11);
578         carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_224_TOP);

580 #if BN_BITS2==64
581         carry = (int)(r_d[BN_NIST_224_TOP-1]>>32);
582 #endif
583         }
584 #endif
585         u.f = bn_sub_words;
586         if (carry > 0)
587                 {
588                 carry = (int)bn_sub_words(r_d,r_d,_nist_p_224[carry-1],BN_NIST_2
589 #if BN_BITS2==64
```



```
590                 carry=(int)(~(r_d[BN_NIST_224_TOP-1]>>32))&1;
591 #endif
592                 }
593         else if (carry < 0)
594                 {
595                 /* it's a bit more comlicated logic in this case.
596                  * if bn_add_words yields no carry, then result
597                  * has to be adjusted by unconditionally *adding*
598                  * the modulus. but if it does, then result has
599                  * to be compared to the modulus and conditionally
600                  * adjusted by *subtracting* the latter. */
601                 carry = (int)bn_add_words(r_d,r_d,_nist_p_224[-carry-1],BN_NIST_
602                 mask = 0-(PTR_SIZE_INT)carry;
603                 u.p = ((PTR_SIZE_INT)bn_sub_words&mask) |
604                  ((PTR_SIZE_INT)bn_add_words&~mask);
605                 }
606         else
607                 carry = 1;

609         /* otherwise it's effectively same as in BN_nist_mod_192... */
610         mask  = 0-(PTR_SIZE_INT)(*u.f)(c_d,r_d,_nist_p_224[0],BN_NIST_224_TOP);
611         mask &= 0-(PTR_SIZE_INT)carry;
612         res   = c_d;
613         res   = (BN_ULONG *)(((PTR_SIZE_INT)res&~mask) |
614          ((PTR_SIZE_INT)r_d&mask));
615         nist_cp_bn(r_d, res, BN_NIST_224_TOP);
616         r->top = BN_NIST_224_TOP;
617         bn_correct_top(r);

619         return 1;
620         }

622 #define nist_set_256(to, from, a1, a2, a3, a4, a5, a6, a7, a8) \
623         { \
624         bn_cp_32(to, 0, from, (a8) - 8) \
625         bn_cp_32(to, 1, from, (a7) - 8) \
626         bn_cp_32(to, 2, from, (a6) - 8) \
627         bn_cp_32(to, 3, from, (a5) - 8) \
628         bn_cp_32(to, 4, from, (a4) - 8) \
629         bn_cp_32(to, 5, from, (a3) - 8) \
630         bn_cp_32(to, 6, from, (a2) - 8) \
631         bn_cp_32(to, 7, from, (a1) - 8) \
632         }

634 int BN_nist_mod_256(BIGNUM *r, const BIGNUM *a, const BIGNUM *field,
635         BN_CTX *ctx)
636         {
637         int     i, top = a->top;
638         int     carry = 0;
639         register BN_ULONG *a_d = a->d, *r_d;
640         union   {
641                 BN_ULONG bn[BN_NIST_256_TOP];
642                 unsigned int ui[BN_NIST_256_TOP*sizeof(BN_ULONG)/sizeof(unsigned
643                 } buf;
644         BN_ULONG c_d[BN_NIST_256_TOP],
645                 *res;
646         PTR_SIZE_INT mask;
647         union { bn_addsub_f f; PTR_SIZE_INT p; } u;
648         static const BIGNUM _bignum_nist_p_256_sqr = {
649                 (BN_ULONG *)_nist_p_256_sqr,
650                 sizeof(_nist_p_256_sqr)/sizeof(_nist_p_256_sqr[0]),
651                 sizeof(_nist_p_256_sqr)/sizeof(_nist_p_256_sqr[0]),
652                 0,BN_FLG_STATIC_DATA };

654         field = &_bignum_nist_p_256; /* just to make sure */
```

```
656         if (BN_is_negative(a) || BN_ucmp(a,&_bignum_nist_p_256_sqr)>=0)
657                 return BN_nnmod(r, a, field, ctx);

659         i = BN_ucmp(field, a);
660         if (i == 0)
661                 {
662                 BN_zero(r);
663                 return 1;
664                 }
665         else if (i > 0)
666                 return (r == a)? 1 : (BN_copy(r ,a) != NULL);

668         if (r != a)
669                 {
670                 if (!bn_wexpand(r, BN_NIST_256_TOP))
671                         return 0;
672                 r_d = r->d;
673                 nist_cp_bn(r_d, a_d, BN_NIST_256_TOP);
674                 }
675         else
676                 r_d = a_d;

678         nist_cp_bn_0(buf.bn, a_d + BN_NIST_256_TOP, top - BN_NIST_256_TOP, BN_NI

680 #if defined(NIST_INT64)
681         {
682         NIST_INT64              acc;    /* accumulator */
683         unsigned int            *rp=(unsigned int *)r_d;
684         const unsigned int      *bp=(const unsigned int *)buf.ui;

686         acc = rp[0];    acc += bp[8-8];
687                         acc += bp[9-8];
688                         acc -= bp[11-8];
689                         acc -= bp[12-8];
690                         acc -= bp[13-8];
691                         acc -= bp[14-8]; rp[0] = (unsigned int)acc; acc >>= 32;

693         acc += rp[1];   acc += bp[9-8];
694                         acc += bp[10-8];
695                         acc -= bp[12-8];
696                         acc -= bp[13-8];
697                         acc -= bp[14-8];
698                         acc -= bp[15-8]; rp[1] = (unsigned int)acc; acc >>= 32;

700         acc += rp[2];   acc += bp[10-8];
701                         acc += bp[11-8];
702                         acc -= bp[13-8];
703                         acc -= bp[14-8];
704                         acc -= bp[15-8]; rp[2] = (unsigned int)acc; acc >>= 32;

706         acc += rp[3];   acc += bp[11-8];
707                         acc += bp[11-8];
708                         acc += bp[12-8];
709                         acc += bp[12-8];
710                         acc += bp[13-8];
711                         acc -= bp[15-8];
712                         acc -= bp[8-8];
713                         acc -= bp[9-8];  rp[3] = (unsigned int)acc; acc >>= 32;

715         acc += rp[4];   acc += bp[12-8];
716                         acc += bp[12-8];
717                         acc += bp[13-8];
718                         acc += bp[13-8];
719                         acc += bp[14-8];
720                         acc -= bp[9-8];
721                         acc -= bp[10-8]; rp[4] = (unsigned int)acc; acc >>= 32;
```



```
723         acc += rp[5];   acc += bp[13-8];
724                         acc += bp[13-8];
725                         acc += bp[14-8];
726                         acc += bp[14-8];
727                         acc += bp[15-8];
728                         acc -= bp[10-8];
729                         acc -= bp[11-8]; rp[5] = (unsigned int)acc; acc >>= 32;

731         acc += rp[6];   acc += bp[14-8];
732                         acc += bp[14-8];
733                         acc += bp[15-8];
734                         acc += bp[15-8];
735                         acc += bp[14-8];
736                         acc += bp[13-8];
737                         acc -= bp[8-8];
738                         acc -= bp[9-8];  rp[6] = (unsigned int)acc; acc >>= 32;

740         acc += rp[7];   acc += bp[15-8];
741                         acc += bp[15-8];
742                         acc += bp[15-8];
743                         acc += bp[8 -8];
744                         acc -= bp[10-8];
745                         acc -= bp[11-8];
746                         acc -= bp[12-8];
747                         acc -= bp[13-8]; rp[7] = (unsigned int)acc;

749         carry = (int)(acc>>32);
750         }
751 #else
752         {
753         BN_ULONG t_d[BN_NIST_256_TOP];

755         /*S1*/
756         nist_set_256(t_d, buf.bn, 15, 14, 13, 12, 11, 0, 0, 0);
757         /*S2*/
758         nist_set_256(c_d, buf.bn, 0, 15, 14, 13, 12, 0, 0, 0);
759         carry = (int)bn_add_words(t_d, t_d, c_d, BN_NIST_256_TOP);
760         /* left shift */
761                 {
762                 register BN_ULONG *ap,t,c;
763                 ap = t_d;
764                 c=0;
765                 for (i = BN_NIST_256_TOP; i != 0; --i)
766                         {
767                         t= *ap;
768                         *(ap++)=((t<<1)|c)&BN_MASK2;
769                         c=(t & BN_TBIT)?1:0;
770                         }
771                 carry <<= 1;
772                 carry  |= c;
773                 }
774         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_256_TOP);
775         /*S3*/
776         nist_set_256(t_d, buf.bn, 15, 14, 0, 0, 0, 10, 9, 8);
777         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_256_TOP);
778         /*S4*/
779         nist_set_256(t_d, buf.bn, 8, 13, 15, 14, 13, 11, 10, 9);
780         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_256_TOP);
781         /*D1*/
782         nist_set_256(t_d, buf.bn, 10, 8, 0, 0, 0, 13, 12, 11);
783         carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_256_TOP);
784         /*D2*/
785         nist_set_256(t_d, buf.bn, 11, 9, 0, 0, 15, 14, 13, 12);
786         carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_256_TOP);
787         /*D3*/
```

```
788         nist_set_256(t_d, buf.bn, 12, 0, 10, 9, 8, 15, 14, 13);
789         carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_256_TOP);
790         /*D4*/
791         nist_set_256(t_d, buf.bn, 13, 0, 11, 10, 9, 0, 15, 14);
792         carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_256_TOP);

794         }
795 #endif
796         /* see BN_nist_mod_224 for explanation */
797         u.f = bn_sub_words;
798         if (carry > 0)
799                 carry = (int)bn_sub_words(r_d,r_d,_nist_p_256[carry-1],BN_NIST_2
800         else if (carry < 0)
801                 {
802                 carry = (int)bn_add_words(r_d,r_d,_nist_p_256[-carry-1],BN_NIST_
803                 mask = 0-(PTR_SIZE_INT)carry;
804                 u.p = ((PTR_SIZE_INT)bn_sub_words&mask) |
805                  ((PTR_SIZE_INT)bn_add_words&~mask);
806                 }
807         else
808                 carry = 1;

810         mask  = 0-(PTR_SIZE_INT)(*u.f)(c_d,r_d,_nist_p_256[0],BN_NIST_256_TOP);
811         mask &= 0-(PTR_SIZE_INT)carry;
812         res   = c_d;
813         res   = (BN_ULONG *)(((PTR_SIZE_INT)res&~mask) |
814          ((PTR_SIZE_INT)r_d&mask));
815         nist_cp_bn(r_d, res, BN_NIST_256_TOP);
816         r->top = BN_NIST_256_TOP;
817         bn_correct_top(r);

819         return 1;
820         }

822 #define nist_set_384(to,from,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12) \
823         { \
824         bn_cp_32(to, 0, from,  (a12) - 12) \
825         bn_cp_32(to, 1, from,  (a11) - 12) \
826         bn_cp_32(to, 2, from,  (a10) - 12) \
827         bn_cp_32(to, 3, from,  (a9) - 12)  \
828         bn_cp_32(to, 4, from,  (a8) - 12)  \
829         bn_cp_32(to, 5, from,  (a7) - 12)  \
830         bn_cp_32(to, 6, from,  (a6) - 12)  \
831         bn_cp_32(to, 7, from,  (a5) - 12)  \
832         bn_cp_32(to, 8, from,  (a4) - 12)  \
833         bn_cp_32(to, 9, from,  (a3) - 12)  \
834         bn_cp_32(to, 10, from, (a2) - 12)  \
835         bn_cp_32(to, 11, from, (a1) - 12)  \
836         }

838 int BN_nist_mod_384(BIGNUM *r, const BIGNUM *a, const BIGNUM *field,
839         BN_CTX *ctx)
840         {
841         int     i, top = a->top;
842         int     carry = 0;
843         register BN_ULONG *r_d, *a_d = a->d;
844         union   {
845                 BN_ULONG bn[BN_NIST_384_TOP];
846                 unsigned int ui[BN_NIST_384_TOP*sizeof(BN_ULONG)/sizeof(unsigned
847                 } buf;
848         BN_ULONG c_d[BN_NIST_384_TOP],
849                 *res;
850         PTR_SIZE_INT mask;
851         union { bn_addsub_f f; PTR_SIZE_INT p; } u;
852         static const BIGNUM _bignum_nist_p_384_sqr = {
853                 (BN_ULONG *)_nist_p_384_sqr,
```



```
854                 sizeof(_nist_p_384_sqr)/sizeof(_nist_p_384_sqr[0]),
855                 sizeof(_nist_p_384_sqr)/sizeof(_nist_p_384_sqr[0]),
856                 0,BN_FLG_STATIC_DATA };


859         field = &_bignum_nist_p_384; /* just to make sure */

861         if (BN_is_negative(a) || BN_ucmp(a,&_bignum_nist_p_384_sqr)>=0)
862                 return BN_nnmod(r, a, field, ctx);

864         i = BN_ucmp(field, a);
865         if (i == 0)
866                 {
867                 BN_zero(r);
868                 return 1;
869                 }
870         else if (i > 0)
871                 return (r == a)? 1 : (BN_copy(r ,a) != NULL);

873         if (r != a)
874                 {
875                 if (!bn_wexpand(r, BN_NIST_384_TOP))
876                         return 0;
877                 r_d = r->d;
878                 nist_cp_bn(r_d, a_d, BN_NIST_384_TOP);
879                 }
880         else
881                 r_d = a_d;

883         nist_cp_bn_0(buf.bn, a_d + BN_NIST_384_TOP, top - BN_NIST_384_TOP, BN_NI

885 #if defined(NIST_INT64)
886         {
887         NIST_INT64              acc;    /* accumulator */
888         unsigned int            *rp=(unsigned int *)r_d;
889         const unsigned int      *bp=(const unsigned int *)buf.ui;

891         acc = rp[0];    acc += bp[12-12];
892                         acc += bp[21-12];
893                         acc += bp[20-12];
894                         acc -= bp[23-12]; rp[0] = (unsigned int)acc; acc >>= 32;

896         acc += rp[1];   acc += bp[13-12];
897                         acc += bp[22-12];
898                         acc += bp[23-12];
899                         acc -= bp[12-12];
900                         acc -= bp[20-12]; rp[1] = (unsigned int)acc; acc >>= 32;

902         acc += rp[2];   acc += bp[14-12];
903                         acc += bp[23-12];
904                         acc -= bp[13-12];
905                         acc -= bp[21-12]; rp[2] = (unsigned int)acc; acc >>= 32;

907         acc += rp[3];   acc += bp[15-12];
908                         acc += bp[12-12];
909                         acc += bp[20-12];
910                         acc += bp[21-12];
911                         acc -= bp[14-12];
912                         acc -= bp[22-12];
913                         acc -= bp[23-12]; rp[3] = (unsigned int)acc; acc >>= 32;

915         acc += rp[4];   acc += bp[21-12];
916                         acc += bp[21-12];
917                         acc += bp[16-12];
918                         acc += bp[13-12];
919                         acc += bp[12-12];
```

```
920                         acc += bp[20-12];
921                         acc += bp[22-12];
922                         acc -= bp[15-12];
923                         acc -= bp[23-12];
924                         acc -= bp[23-12]; rp[4] = (unsigned int)acc; acc >>= 32;

926         acc += rp[5];   acc += bp[22-12];
927                         acc += bp[22-12];
928                         acc += bp[17-12];
929                         acc += bp[14-12];
930                         acc += bp[13-12];
931                         acc += bp[21-12];
932                         acc += bp[23-12];
933                         acc -= bp[16-12]; rp[5] = (unsigned int)acc; acc >>= 32;

935         acc += rp[6];   acc += bp[23-12];
936                         acc += bp[23-12];
937                         acc += bp[18-12];
938                         acc += bp[15-12];
939                         acc += bp[14-12];
940                         acc += bp[22-12];
941                         acc -= bp[17-12]; rp[6] = (unsigned int)acc; acc >>= 32;

943         acc += rp[7];   acc += bp[19-12];
944                         acc += bp[16-12];
945                         acc += bp[15-12];
946                         acc += bp[23-12];
947                         acc -= bp[18-12]; rp[7] = (unsigned int)acc; acc >>= 32;

949         acc += rp[8];   acc += bp[20-12];
950                         acc += bp[17-12];
951                         acc += bp[16-12];
952                         acc -= bp[19-12]; rp[8] = (unsigned int)acc; acc >>= 32;

954         acc += rp[9];   acc += bp[21-12];
955                         acc += bp[18-12];
956                         acc += bp[17-12];
957                         acc -= bp[20-12]; rp[9] = (unsigned int)acc; acc >>= 32;

959         acc += rp[10];  acc += bp[22-12];
960                         acc += bp[19-12];
961                         acc += bp[18-12];
962                         acc -= bp[21-12]; rp[10] = (unsigned int)acc; acc >>= 32

964         acc += rp[11];  acc += bp[23-12];
965                         acc += bp[20-12];
966                         acc += bp[19-12];
967                         acc -= bp[22-12]; rp[11] = (unsigned int)acc;

969         carry = (int)(acc>>32);
970         }
971 #else
972         {
973         BN_ULONG t_d[BN_NIST_384_TOP];

975         /*S1*/
976         nist_set_256(t_d, buf.bn, 0, 0, 0, 0, 0, 23-4, 22-4, 21-4);
977                 /* left shift */
978                 {
979                 register BN_ULONG *ap,t,c;
980                 ap = t_d;
981                 c=0;
982                 for (i = 3; i != 0; --i)
983                         {
984                         t= *ap;
985                         *(ap++)=((t<<1)|c)&BN_MASK2;
```



```
986                         c=(t & BN_TBIT)?1:0;
987                         }
988                 *ap=c;
989                 }
990         carry = (int)bn_add_words(r_d+(128/BN_BITS2), r_d+(128/BN_BITS2),
991                 t_d, BN_NIST_256_TOP);
992         /*S2 */
993         carry += (int)bn_add_words(r_d, r_d, buf.bn, BN_NIST_384_TOP);
994         /*S3*/
995         nist_set_384(t_d,buf.bn,20,19,18,17,16,15,14,13,12,23,22,21);
996         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_384_TOP);
997         /*S4*/
998         nist_set_384(t_d,buf.bn,19,18,17,16,15,14,13,12,20,0,23,0);
999         carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_384_TOP);
1000        /*S5*/
1001        nist_set_384(t_d, buf.bn,0,0,0,0,23,22,21,20,0,0,0,0);
1002        carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_384_TOP);
1003        /*S6*/
1004        nist_set_384(t_d,buf.bn,0,0,0,0,0,0,23,22,21,0,0,20);
1005        carry += (int)bn_add_words(r_d, r_d, t_d, BN_NIST_384_TOP);
1006        /*D1*/
1007        nist_set_384(t_d,buf.bn,22,21,20,19,18,17,16,15,14,13,12,23);
1008        carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_384_TOP);
1009        /*D2*/
1010        nist_set_384(t_d,buf.bn,0,0,0,0,0,0,0,23,22,21,20,0);
1011        carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_384_TOP);
1012        /*D3*/
1013        nist_set_384(t_d,buf.bn,0,0,0,0,0,0,0,23,23,0,0,0);
1014        carry -= (int)bn_sub_words(r_d, r_d, t_d, BN_NIST_384_TOP);

1016        }
1017 #endif
1018        /* see BN_nist_mod_224 for explanation */
1019        u.f = bn_sub_words;
1020        if (carry > 0)
1021                carry = (int)bn_sub_words(r_d,r_d,_nist_p_384[carry-1],BN_NIST_3
1022        else if (carry < 0)
1023                {
1024                carry = (int)bn_add_words(r_d,r_d,_nist_p_384[-carry-1],BN_NIST_
1025                mask = 0-(PTR_SIZE_INT)carry;
1026                u.p = ((PTR_SIZE_INT)bn_sub_words&mask) |
1027                 ((PTR_SIZE_INT)bn_add_words&~mask);
1028                }
1029        else
1030                carry = 1;

1032        mask  = 0-(PTR_SIZE_INT)(*u.f)(c_d,r_d,_nist_p_384[0],BN_NIST_384_TOP);
1033        mask &= 0-(PTR_SIZE_INT)carry;
1034        res   = c_d;
1035        res   = (BN_ULONG *)(((PTR_SIZE_INT)res&~mask) |
1036         ((PTR_SIZE_INT)r_d&mask));
1037        nist_cp_bn(r_d, res, BN_NIST_384_TOP);
1038        r->top = BN_NIST_384_TOP;
1039        bn_correct_top(r);

1041        return 1;
1042        }

1044 #define BN_NIST_521_RSHIFT      (521%BN_BITS2)
1045 #define BN_NIST_521_LSHIFT      (BN_BITS2-BN_NIST_521_RSHIFT)
1046 #define BN_NIST_521_TOP_MASK    ((BN_ULONG)BN_MASK2>>BN_NIST_521_LSHIFT)

1048 int BN_nist_mod_521(BIGNUM *r, const BIGNUM *a, const BIGNUM *field,
1049         BN_CTX *ctx)
1050         {
1051         int     top = a->top, i;
```

```
1052         BN_ULONG *r_d, *a_d = a->d,
1053                  t_d[BN_NIST_521_TOP],
1054                  val,tmp,*res;
1055         PTR_SIZE_INT mask;
1056         static const BIGNUM _bignum_nist_p_521_sqr = {
1057                 (BN_ULONG *)_nist_p_521_sqr,
1058                 sizeof(_nist_p_521_sqr)/sizeof(_nist_p_521_sqr[0]),
1059                 sizeof(_nist_p_521_sqr)/sizeof(_nist_p_521_sqr[0]),
1060                 0,BN_FLG_STATIC_DATA };

1062         field = &_bignum_nist_p_521; /* just to make sure */

1064         if (BN_is_negative(a) || BN_ucmp(a,&_bignum_nist_p_521_sqr)>=0)
1065                 return BN_nnmod(r, a, field, ctx);

1067         i = BN_ucmp(field, a);
1068         if (i == 0)
1069                 {
1070                 BN_zero(r);
1071                 return 1;
1072                 }
1073         else if (i > 0)
1074                 return (r == a)? 1 : (BN_copy(r ,a) != NULL);

1076         if (r != a)
1077                 {
1078                 if (!bn_wexpand(r,BN_NIST_521_TOP))
1079                         return 0;
1080                 r_d = r->d;
1081                 nist_cp_bn(r_d,a_d, BN_NIST_521_TOP);
1082                 }
1083         else
1084                 r_d = a_d;

1086         /* upper 521 bits, copy ... */
1087         nist_cp_bn_0(t_d,a_d + (BN_NIST_521_TOP-1), top - (BN_NIST_521_TOP-1),BN
1088         /* ... and right shift */
1089         for (val=t_d[0],i=0; i<BN_NIST_521_TOP-1; i++)
1090                 {
1091                 tmp = val>>BN_NIST_521_RSHIFT;
1092                 val = t_d[i+1];
1093                 t_d[i] = (tmp | val<<BN_NIST_521_LSHIFT) & BN_MASK2;
1094                 }
1095         t_d[i] = val>>BN_NIST_521_RSHIFT;
1096         /* lower 521 bits */
1097         r_d[i] &= BN_NIST_521_TOP_MASK;

1099         bn_add_words(r_d,r_d,t_d,BN_NIST_521_TOP);
1100         mask = 0-(PTR_SIZE_INT)bn_sub_words(t_d,r_d,_nist_p_521,BN_NIST_521_TOP)
1101         res  = t_d;
1102         res  = (BN_ULONG *)(((PTR_SIZE_INT)res&~mask) |
1103          ((PTR_SIZE_INT)r_d&mask));
1104         nist_cp_bn(r_d,res,BN_NIST_521_TOP);
1105         r->top = BN_NIST_521_TOP;
1106         bn_correct_top(r);

1108         return 1;
1109         }
1110 #endif /* ! codereview */
```

```
**********************************************************
   14195 Wed Aug 13 19:52:16 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_prime.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_prime.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
#include "bn_lcl.h"
#include <openssl/rand.h>

/* NB: these functions have been "upgraded", the deprecated versions (which are
 * compatibility wrappers using these functions) are in bn_depr.c.
 * - Geoff
 */

/* The quick sieve algorithm approach to weeding out primes is
 * Philip Zimmermann's, as implemented in PGP.  I have had a read of
 * his comments and implemented my own version.
 */
#include "bn_prime.h"
```

```
129 static int witness(BIGNUM *w, const BIGNUM *a, const BIGNUM *a1,
130         const BIGNUM *a1_odd, int k, BN_CTX *ctx, BN_MONT_CTX *mont);
131 static int probable_prime(BIGNUM *rnd, int bits);
132 static int probable_prime_dh(BIGNUM *rnd, int bits,
133         const BIGNUM *add, const BIGNUM *rem, BN_CTX *ctx);
134 static int probable_prime_dh_safe(BIGNUM *rnd, int bits,
135         const BIGNUM *add, const BIGNUM *rem, BN_CTX *ctx);

137 int BN_GENCB_call(BN_GENCB *cb, int a, int b)
138         {
139         /* No callback means continue */
140         if(!cb) return 1;
141         switch(cb->ver)
142                 {
143         case 1:
144                 /* Deprecated-style callbacks */
145                 if(!cb->cb.cb_1)
146                         return 1;
147                 cb->cb.cb_1(a, b, cb->arg);
148                 return 1;
149         case 2:
150                 /* New-style callbacks */
151                 return cb->cb.cb_2(a, b, cb);
152         default:
153                 break;
154                 }
155         /* Unrecognised callback type */
156         return 0;
157         }

159 int BN_generate_prime_ex(BIGNUM *ret, int bits, int safe,
160         const BIGNUM *add, const BIGNUM *rem, BN_GENCB *cb)
161         {
162         BIGNUM *t;
163         int found=0;
164         int i,j,c1=0;
165         BN_CTX *ctx;
166         int checks = BN_prime_checks_for_size(bits);

168         ctx=BN_CTX_new();
169         if (ctx == NULL) goto err;
170         BN_CTX_start(ctx);
171         t = BN_CTX_get(ctx);
172         if(!t) goto err;
173 loop:
174         /* make a random number and set the top and bottom bits */
175         if (add == NULL)
176                 {
177                 if (!probable_prime(ret,bits)) goto err;
178                 }
179         else
180                 {
181                 if (safe)
182                         {
183                         if (!probable_prime_dh_safe(ret,bits,add,rem,ctx))
184                                 goto err;
185                         }
186                 else
187                         {
188                         if (!probable_prime_dh(ret,bits,add,rem,ctx))
189                                 goto err;
190                         }
191                 }
192         /* if (BN_mod_word(ret,(BN_ULONG)3) == 1) goto loop; */
193         if(!BN_GENCB_call(cb, 0, c1++))
```



```
194                 /* aborted */
195                 goto err;

197         if (!safe)
198                 {
199                 i=BN_is_prime_fasttest_ex(ret,checks,ctx,0,cb);
200                 if (i == -1) goto err;
201                 if (i == 0) goto loop;
202                 }
203         else
204                 {
205                 /* for "safe prime" generation,
206                  * check that (p-1)/2 is prime.
207                  * Since a prime is odd, We just
208                  * need to divide by 2 */
209                 if (!BN_rshift1(t,ret)) goto err;

211                 for (i=0; i<checks; i++)
212                         {
213                         j=BN_is_prime_fasttest_ex(ret,1,ctx,0,cb);
214                         if (j == -1) goto err;
215                         if (j == 0) goto loop;

217                         j=BN_is_prime_fasttest_ex(t,1,ctx,0,cb);
218                         if (j == -1) goto err;
219                         if (j == 0) goto loop;

221                         if(!BN_GENCB_call(cb, 2, c1-1))
222                                 goto err;
223                         /* We have a safe prime test pass */
224                         }
225                 }
226         /* we have a prime :-) */
227         found = 1;
228 err:
229         if (ctx != NULL)
230                 {
231                 BN_CTX_end(ctx);
232                 BN_CTX_free(ctx);
233                 }
234         bn_check_top(ret);
235         return found;
236         }

238 int BN_is_prime_ex(const BIGNUM *a, int checks, BN_CTX *ctx_passed, BN_GENCB *cb
239         {
240         return BN_is_prime_fasttest_ex(a, checks, ctx_passed, 0, cb);
241         }

243 int BN_is_prime_fasttest_ex(const BIGNUM *a, int checks, BN_CTX *ctx_passed,
244                 int do_trial_division, BN_GENCB *cb)
245         {
246         int i, j, ret = -1;
247         int k;
248         BN_CTX *ctx = NULL;
249         BIGNUM *A1, *A1_odd, *check; /* taken from ctx */
250         BN_MONT_CTX *mont = NULL;
251         const BIGNUM *A = NULL;

253         if (BN_cmp(a, BN_value_one()) <= 0)
254                 return 0;

256         if (checks == BN_prime_checks)
257                 checks = BN_prime_checks_for_size(BN_num_bits(a));

259         /* first look for small factors */
```

```
260         if (!BN_is_odd(a))
261                 /* a is even => a is prime if and only if a == 2 */
262                 return BN_is_word(a, 2);
263         if (do_trial_division)
264                 {
265                 for (i = 1; i < NUMPRIMES; i++)
266                         if (BN_mod_word(a, primes[i]) == 0)
267                                 return 0;
268                 if(!BN_GENCB_call(cb, 1, -1))
269                         goto err;
270                 }

272         if (ctx_passed != NULL)
273                 ctx = ctx_passed;
274         else
275                 if ((ctx=BN_CTX_new()) == NULL)
276                         goto err;
277         BN_CTX_start(ctx);

279         /* A := abs(a) */
280         if (a->neg)
281                 {
282                 BIGNUM *t;
283                 if ((t = BN_CTX_get(ctx)) == NULL) goto err;
284                 BN_copy(t, a);
285                 t->neg = 0;
286                 A = t;
287                 }
288         else
289                 A = a;
290         A1 = BN_CTX_get(ctx);
291         A1_odd = BN_CTX_get(ctx);
292         check = BN_CTX_get(ctx);
293         if (check == NULL) goto err;

295         /* compute A1 := A - 1 */
296         if (!BN_copy(A1, A))
297                 goto err;
298         if (!BN_sub_word(A1, 1))
299                 goto err;
300         if (BN_is_zero(A1))
301                 {
302                 ret = 0;
303                 goto err;
304                 }

306         /* write  A1  as  A1_odd * 2^k */
307         k = 1;
308         while (!BN_is_bit_set(A1, k))
309                 k++;
310         if (!BN_rshift(A1_odd, A1, k))
311                 goto err;

313         /* Montgomery setup for computations mod A */
314         mont = BN_MONT_CTX_new();
315         if (mont == NULL)
316                 goto err;
317         if (!BN_MONT_CTX_set(mont, A, ctx))
318                 goto err;

320         for (i = 0; i < checks; i++)
321                 {
322                 if (!BN_pseudo_rand_range(check, A1))
323                         goto err;
324                 if (!BN_add_word(check, 1))
325                         goto err;
```



```
326                 /* now 1 <= check < A */

328                 j = witness(check, A, A1, A1_odd, k, ctx, mont);
329                 if (j == -1) goto err;
330                 if (j)
331                         {
332                         ret=0;
333                         goto err;
334                         }
335                 if(!BN_GENCB_call(cb, 1, i))
336                         goto err;
337                 }
338         ret=1;
339 err:
340         if (ctx != NULL)
341                 {
342                 BN_CTX_end(ctx);
343                 if (ctx_passed == NULL)
344                         BN_CTX_free(ctx);
345                 }
346         if (mont != NULL)
347                 BN_MONT_CTX_free(mont);

349         return(ret);
350         }

352 static int witness(BIGNUM *w, const BIGNUM *a, const BIGNUM *a1,
353         const BIGNUM *a1_odd, int k, BN_CTX *ctx, BN_MONT_CTX *mont)
354         {
355         if (!BN_mod_exp_mont(w, w, a1_odd, a, ctx, mont)) /* w := w^a1_odd mod a
356                 return -1;
357         if (BN_is_one(w))
358                 return 0; /* probably prime */
359         if (BN_cmp(w, a1) == 0)
360                 return 0; /* w == -1 (mod a),  'a' is probably prime */
361         while (--k)
362                 {
363                 if (!BN_mod_mul(w, w, w, a, ctx)) /* w := w^2 mod a */
364                         return -1;
365                 if (BN_is_one(w))
366                         return 1; /* 'a' is composite, otherwise a previous 'w'
367                                    * have been == -1 (mod 'a') */
368                 if (BN_cmp(w, a1) == 0)
369                         return 0; /* w == -1 (mod a), 'a' is probably prime */
370                 }
371         /* If we get here, 'w' is the (a-1)/2-th power of the original 'w',
372          * and it is neither -1 nor +1 -- so 'a' cannot be prime */
373         bn_check_top(w);
374         return 1;
375         }

377 static int probable_prime(BIGNUM *rnd, int bits)
378         {
379         int i;
380         prime_t mods[NUMPRIMES];
381         BN_ULONG delta,maxdelta;

383 again:
384         if (!BN_rand(rnd,bits,1,1)) return(0);
385         /* we now have a random number 'rand' to test. */
386         for (i=1; i<NUMPRIMES; i++)
387                 mods[i]=(prime_t)BN_mod_word(rnd,(BN_ULONG)primes[i]);
388         maxdelta=BN_MASK2 - primes[NUMPRIMES-1];
389         delta=0;
390         loop: for (i=1; i<NUMPRIMES; i++)
391                 {
```

```
392                 /* check that rnd is not a prime and also
393                  * that gcd(rnd-1,primes) == 1 (except for 2) */
394                 if (((mods[i]+delta)%primes[i]) <= 1)
395                         {
396                         delta+=2;
397                         if (delta > maxdelta) goto again;
398                         goto loop;
399                         }
400                 }
401         if (!BN_add_word(rnd,delta)) return(0);
402         bn_check_top(rnd);
403         return(1);
404         }

406 static int probable_prime_dh(BIGNUM *rnd, int bits,
407         const BIGNUM *add, const BIGNUM *rem, BN_CTX *ctx)
408         {
409         int i,ret=0;
410         BIGNUM *t1;

412         BN_CTX_start(ctx);
413         if ((t1 = BN_CTX_get(ctx)) == NULL) goto err;

415         if (!BN_rand(rnd,bits,0,1)) goto err;

417         /* we need ((rnd-rem) % add) == 0 */

419         if (!BN_mod(t1,rnd,add,ctx)) goto err;
420         if (!BN_sub(rnd,rnd,t1)) goto err;
421         if (rem == NULL)
422                 { if (!BN_add_word(rnd,1)) goto err; }
423         else
424                 { if (!BN_add(rnd,rnd,rem)) goto err; }

426         /* we now have a random number 'rand' to test. */

428         loop: for (i=1; i<NUMPRIMES; i++)
429                 {
430                 /* check that rnd is a prime */
431                 if (BN_mod_word(rnd,(BN_ULONG)primes[i]) <= 1)
432                         {
433                         if (!BN_add(rnd,rnd,add)) goto err;
434                         goto loop;
435                         }
436                 }
437         ret=1;
438 err:
439         BN_CTX_end(ctx);
440         bn_check_top(rnd);
441         return(ret);
442         }

444 static int probable_prime_dh_safe(BIGNUM *p, int bits, const BIGNUM *padd,
445         const BIGNUM *rem, BN_CTX *ctx)
446         {
447         int i,ret=0;
448         BIGNUM *t1,*qadd,*q;

450         bits--;
451         BN_CTX_start(ctx);
452         t1 = BN_CTX_get(ctx);
453         q = BN_CTX_get(ctx);
454         qadd = BN_CTX_get(ctx);
455         if (qadd == NULL) goto err;

457         if (!BN_rshift1(qadd,padd)) goto err;
```



```
459         if (!BN_rand(q,bits,0,1)) goto err;

461         /* we need ((rnd-rem) % add) == 0 */
462         if (!BN_mod(t1,q,qadd,ctx)) goto err;
463         if (!BN_sub(q,q,t1)) goto err;
464         if (rem == NULL)
465                 { if (!BN_add_word(q,1)) goto err; }
466         else
467                 {
468                 if (!BN_rshift1(t1,rem)) goto err;
469                 if (!BN_add(q,q,t1)) goto err;
470                 }

472         /* we now have a random number 'rand' to test. */
473         if (!BN_lshift1(p,q)) goto err;
474         if (!BN_add_word(p,1)) goto err;

476         loop: for (i=1; i<NUMPRIMES; i++)
477                 {
478                 /* check that p and q are prime */
479                 /* check that for p and q
480                  * gcd(p-1,primes) == 1 (except for 2) */
481                 if (    (BN_mod_word(p,(BN_ULONG)primes[i]) == 0) ||
482                         (BN_mod_word(q,(BN_ULONG)primes[i]) == 0))
483                         {
484                         if (!BN_add(p,p,padd)) goto err;
485                         if (!BN_add(q,q,qadd)) goto err;
486                         goto loop;
487                         }
488                 }
489         ret=1;
490 err:
491         BN_CTX_end(ctx);
492         bn_check_top(p);
493         return(ret);
494         }
495 #endif /* ! codereview */
```

```
**********************************************************
    8848 Wed Aug 13 19:52:17 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_print.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_print.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include "bn_lcl.h"

static const char Hex[]="0123456789ABCDEF";

/* Must 'OPENSSL_free' the returned data */
char *BN_bn2hex(const BIGNUM *a)
        {
        int i,j,v,z=0;
        char *buf;
        char *p;

        buf=(char *)OPENSSL_malloc(a->top*BN_BYTES*2+2);
        if (buf == NULL)
                {
                BNerr(BN_F_BN_BN2HEX,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        p=buf;
        if (a->neg) *(p++)='-';
        if (BN_is_zero(a)) *(p++)='0';
        for (i=a->top-1; i >=0; i--)
                {
                for (j=BN_BITS2-8; j >= 0; j-=8)
                        {
                        /* strip leading zeros */
                        v=((int)(a->d[i]>>(long)j))&0xff;
                        if (z || (v != 0))
                                {
                                *(p++)=Hex[v>>4];
                                *(p++)=Hex[v&0x0f];
                                z=1;
                                }
                        }
                }
        *p='\0';
err:
        return(buf);
        }

/* Must 'OPENSSL_free' the returned data */
char *BN_bn2dec(const BIGNUM *a)
        {
        int i=0,num, ok = 0;
        char *buf=NULL;
        char *p;
        BIGNUM *t=NULL;
        BN_ULONG *bn_data=NULL,*lp;

        /* get an upper bound for the length of the decimal integer
         * num <= (BN_num_bits(a) + 1) * log(2)
         *     <= 3 * BN_num_bits(a) * 0.1001 + log(2) + 1     (rounding error)
         *     <= BN_num_bits(a)/10 + BN_num_bits/1000 + 1 + 1
         */
        i=BN_num_bits(a)*3;
        num=(i/10+i/1000+1)+1;
        bn_data=(BN_ULONG *)OPENSSL_malloc((num/BN_DEC_NUM+1)*sizeof(BN_ULONG));
        buf=(char *)OPENSSL_malloc(num+3);
        if ((buf == NULL) || (bn_data == NULL))
                {
                BNerr(BN_F_BN_BN2DEC,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        if ((t=BN_dup(a)) == NULL) goto err;

#define BUF_REMAIN (num+3 - (size_t)(p - buf))
```

```
128         p=buf;
129         lp=bn_data;
130         if (BN_is_zero(t))
131                 {
132                 *(p++)='0';
133                 *(p++)='\0';
134                 }
135         else
136                 {
137                 if (BN_is_negative(t))
138                         *p++ = '-';
139
140                 i=0;
141                 while (!BN_is_zero(t))
142                         {
143                         *lp=BN_div_word(t,BN_DEC_CONV);
144                         lp++;
145                         }
146                 lp--;
147                 /* We now have a series of blocks, BN_DEC_NUM chars
148                  * in length, where the last one needs truncation.
149                  * The blocks need to be reversed in order. */
150                 BIO_snprintf(p,BUF_REMAIN,BN_DEC_FMT1,*lp);
151                 while (*p) p++;
152                 while (lp != bn_data)
153                         {
154                         lp--;
155                         BIO_snprintf(p,BUF_REMAIN,BN_DEC_FMT2,*lp);
156                         while (*p) p++;
157                         }
158                 }
159         ok = 1;
160 err:
161         if (bn_data != NULL) OPENSSL_free(bn_data);
162         if (t != NULL) BN_free(t);
163         if (!ok && buf)
164                 {
165                 OPENSSL_free(buf);
166                 buf = NULL;
167                 }
168
169         return(buf);
170         }
171
172 int BN_hex2bn(BIGNUM **bn, const char *a)
173         {
174         BIGNUM *ret=NULL;
175         BN_ULONG l=0;
176         int neg=0,h,m,i,j,k,c;
177         int num;
178
179         if ((a == NULL) || (*a == '\0')) return(0);
180
181         if (*a == '-') { neg=1; a++; }
182
183         for (i=0; isxdigit((unsigned char) a[i]); i++)
184                 ;
185
186         num=i+neg;
187         if (bn == NULL) return(num);
188
189         /* a is the start of the hex digits, and it is 'i' long */
190         if (*bn == NULL)
191                 {
192                 if ((ret=BN_new()) == NULL) return(0);
193                 }
```



```
194         else
195                 {
196                 ret= *bn;
197                 BN_zero(ret);
198                 }
199
200         /* i is the number of hex digests; */
201         if (bn_expand(ret,i*4) == NULL) goto err;
202
203         j=i; /* least significant 'hex' */
204         m=0;
205         h=0;
206         while (j > 0)
207                 {
208                 m=((BN_BYTES*2) <= j)?(BN_BYTES*2):j;
209                 l=0;
210                 for (;;)
211                         {
212                         c=a[j-m];
213                         if ((c >= '0') && (c <= '9')) k=c-'0';
214                         else if ((c >= 'a') && (c <= 'f')) k=c-'a'+10;
215                         else if ((c >= 'A') && (c <= 'F')) k=c-'A'+10;
216                         else k=0; /* paranoia */
217                         l=(l<<4)|k;
218
219                         if (--m <= 0)
220                                 {
221                                 ret->d[h++]=l;
222                                 break;
223                                 }
224                         }
225                 j-=(BN_BYTES*2);
226                 }
227         ret->top=h;
228         bn_correct_top(ret);
229         ret->neg=neg;
230
231         *bn=ret;
232         bn_check_top(ret);
233         return(num);
234 err:
235         if (*bn == NULL) BN_free(ret);
236         return(0);
237         }
238
239 int BN_dec2bn(BIGNUM **bn, const char *a)
240         {
241         BIGNUM *ret=NULL;
242         BN_ULONG l=0;
243         int neg=0,i,j;
244         int num;
245
246         if ((a == NULL) || (*a == '\0')) return(0);
247         if (*a == '-') { neg=1; a++; }
248
249         for (i=0; isdigit((unsigned char) a[i]); i++)
250                 ;
251
252         num=i+neg;
253         if (bn == NULL) return(num);
254
255         /* a is the start of the digits, and it is 'i' long.
256          * We chop it into BN_DEC_NUM digits at a time */
257         if (*bn == NULL)
258                 {
259                 if ((ret=BN_new()) == NULL) return(0);
```

```
260                 }
261         else
262                 {
263                 ret= *bn;
264                 BN_zero(ret);
265                 }

267         /* i is the number of digests, a bit of an over expand; */
268         if (bn_expand(ret,i*4) == NULL) goto err;

270         j=BN_DEC_NUM-(i%BN_DEC_NUM);
271         if (j == BN_DEC_NUM) j=0;
272         l=0;
273         while (*a)
274                 {
275                 l*=10;
276                 l+= *a-'0';
277                 a++;
278                 if (++j == BN_DEC_NUM)
279                         {
280                         BN_mul_word(ret,BN_DEC_CONV);
281                         BN_add_word(ret,l);
282                         l=0;
283                         j=0;
284                         }
285                 }
286         ret->neg=neg;

288         bn_correct_top(ret);
289         *bn=ret;
290         bn_check_top(ret);
291         return(num);
292 err:
293         if (*bn == NULL) BN_free(ret);
294         return(0);
295         }

297 int BN_asc2bn(BIGNUM **bn, const char *a)
298         {
299         const char *p = a;
300         if (*p == '-')
301                 p++;

303         if (p[0] == '0' && (p[1] == 'X' || p[1] == 'x'))
304                 {
305                 if (!BN_hex2bn(bn, p + 2))
306                         return 0;
307                 }
308         else
309                 {
310                 if (!BN_dec2bn(bn, p))
311                         return 0;
312                 }
313         if (*a == '-')
314                 (*bn)->neg = 1;
315         return 1;
316         }

318 #ifndef OPENSSL_NO_BIO
319 #ifndef OPENSSL_NO_FP_API
320 int BN_print_fp(FILE *fp, const BIGNUM *a)
321         {
322         BIO *b;
323         int ret;

325         if ((b=BIO_new(BIO_s_file())) == NULL)
```



```
326                 return(0);
327         BIO_set_fp(b,fp,BIO_NOCLOSE);
328         ret=BN_print(b,a);
329         BIO_free(b);
330         return(ret);
331         }
332 #endif

334 int BN_print(BIO *bp, const BIGNUM *a)
335         {
336         int i,j,v,z=0;
337         int ret=0;

339         if ((a->neg) && (BIO_write(bp,"-",1) != 1)) goto end;
340         if (BN_is_zero(a) && (BIO_write(bp,"0",1) != 1)) goto end;
341         for (i=a->top-1; i >=0; i--)
342                 {
343                 for (j=BN_BITS2-4; j >= 0; j-=4)
344                         {
345                         /* strip leading zeros */
346                         v=((int)(a->d[i]>>(long)j))&0x0f;
347                         if (z || (v != 0))
348                                 {
349                                 if (BIO_write(bp,&(Hex[v]),1) != 1)
350                                         goto end;
351                                 z=1;
352                                 }
353                         }
354                 }
355         ret=1;
356 end:
357         return(ret);
358         }
359 #endif

361 char *BN_options(void)
362         {
363         static int init=0;
364         static char data[16];

366         if (!init)
367                 {
368                 init++;
369 #ifdef BN_LLONG
370                 BIO_snprintf(data,sizeof data,"bn(%d,%d)",
371                              (int)sizeof(BN_ULLONG)*8,(int)sizeof(BN_ULONG)*8);
372 #else
373                 BIO_snprintf(data,sizeof data,"bn(%d,%d)",
374                              (int)sizeof(BN_ULONG)*8,(int)sizeof(BN_ULONG)*8);
375 #endif
376                 }
377         return(data);
378         }
379 #endif /* ! codereview */
```

```
**********************************************************
    9378 Wed Aug 13 19:52:17 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_rand.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/bn/bn_rand.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
#include "bn_lcl.h"
#include <openssl/rand.h>

static int bnrand(int pseudorand, BIGNUM *rnd, int bits, int top, int bottom)
        {
        unsigned char *buf=NULL;
        int ret=0,bit,bytes,mask;
        time_t tim;

        if (bits == 0)
                {
                BN_zero(rnd);
                return 1;
```

```
128                 }

130         bytes=(bits+7)/8;
131         bit=(bits-1)%8;
132         mask=0xff<<(bit+1);

134         buf=(unsigned char *)OPENSSL_malloc(bytes);
135         if (buf == NULL)
136                 {
137                 BNerr(BN_F_BNRAND,ERR_R_MALLOC_FAILURE);
138                 goto err;
139                 }

141         /* make a random number and set the top and bottom bits */
142         time(&tim);
143         RAND_add(&tim,sizeof(tim),0.0);

145         if (pseudorand)
146                 {
147                 if (RAND_pseudo_bytes(buf, bytes) == -1)
148                         goto err;
149                 }
150         else
151                 {
152                 if (RAND_bytes(buf, bytes) <= 0)
153                         goto err;
154                 }

156 #if 1
157         if (pseudorand == 2)
158                 {
159                 /* generate patterns that are more likely to trigger BN
160                    library bugs */
161                 int i;
162                 unsigned char c;

164                 for (i = 0; i < bytes; i++)
165                         {
166                         RAND_pseudo_bytes(&c, 1);
167                         if (c >= 128 && i > 0)
168                                 buf[i] = buf[i-1];
169                         else if (c < 42)
170                                 buf[i] = 0;
171                         else if (c < 84)
172                                 buf[i] = 255;
173                         }
174                 }
175 #endif

177         if (top != -1)
178                 {
179                 if (top)
180                         {
181                         if (bit == 0)
182                                 {
183                                 buf[0]=1;
184                                 buf[1]|=0x80;
185                                 }
186                         else
187                                 {
188                                 buf[0]|=(3<<(bit-1));
189                                 }
190                         }
191                 else
192                         {
193                         buf[0]|=(1<<bit);
```



```
194                         }
195                 }
196         buf[0] &= ~mask;
197         if (bottom) /* set bottom bit if requested */
198                 buf[bytes-1]|=1;
199         if (!BN_bin2bn(buf,bytes,rnd)) goto err;
200         ret=1;
201 err:
202         if (buf != NULL)
203                 {
204                 OPENSSL_cleanse(buf,bytes);
205                 OPENSSL_free(buf);
206                 }
207         bn_check_top(rnd);
208         return(ret);
209         }

211 int     BN_rand(BIGNUM *rnd, int bits, int top, int bottom)
212         {
213         return bnrand(0, rnd, bits, top, bottom);
214         }

216 int     BN_pseudo_rand(BIGNUM *rnd, int bits, int top, int bottom)
217         {
218         return bnrand(1, rnd, bits, top, bottom);
219         }

221 #if 1
222 int     BN_bntest_rand(BIGNUM *rnd, int bits, int top, int bottom)
223         {
224         return bnrand(2, rnd, bits, top, bottom);
225         }
226 #endif


229 /* random number r:  0 <= r < range */
230 static int bn_rand_range(int pseudo, BIGNUM *r, const BIGNUM *range)
231         {
232         int (*bn_rand)(BIGNUM *, int, int, int) = pseudo ? BN_pseudo_rand : BN_r
233         int n;
234         int count = 100;

236         if (range->neg || BN_is_zero(range))
237                 {
238                 BNerr(BN_F_BN_RAND_RANGE, BN_R_INVALID_RANGE);
239                 return 0;
240                 }

242         n = BN_num_bits(range); /* n > 0 */

244         /* BN_is_bit_set(range, n - 1) always holds */

246         if (n == 1)
247                 BN_zero(r);
248         else if (!BN_is_bit_set(range, n - 2) && !BN_is_bit_set(range, n - 3))
249                 {
250                 /* range = 100..._2,
251                  * so  3*range (= 11..._2)  is exactly one bit longer than  rang
252                 do
253                         {
254                         if (!bn_rand(r, n + 1, -1, 0)) return 0;
255                         /* If  r < 3*range,  use  r := r MOD range
256                          * (which is either  r, r - range,  or  r - 2*range).
257                          * Otherwise, iterate once more.
258                          * Since  3*range = 11..._2, each iteration succeeds wit
259                          * probability >= .75. */
```

```
260                         if (BN_cmp(r ,range) >= 0)
261                                 {
262                                 if (!BN_sub(r, r, range)) return 0;
263                                 if (BN_cmp(r, range) >= 0)
264                                         if (!BN_sub(r, r, range)) return 0;
265                                 }

267                         if (!--count)
268                                 {
269                                 BNerr(BN_F_BN_RAND_RANGE, BN_R_TOO_MANY_ITERATIO
270                                 return 0;
271                                 }

273                         }
274                 while (BN_cmp(r, range) >= 0);
275                 }
276         else
277                 {
278                 do
279                         {
280                         /* range = 11..._2  or  range = 101..._2 */
281                         if (!bn_rand(r, n, -1, 0)) return 0;

283                         if (!--count)
284                                 {
285                                 BNerr(BN_F_BN_RAND_RANGE, BN_R_TOO_MANY_ITERATIO
286                                 return 0;
287                                 }
288                         }
289                 while (BN_cmp(r, range) >= 0);
290                 }

292         bn_check_top(r);
293         return 1;
294         }


297 int     BN_rand_range(BIGNUM *r, const BIGNUM *range)
298         {
299         return bn_rand_range(0, r, range);
300         }

302 int     BN_pseudo_rand_range(BIGNUM *r, const BIGNUM *range)
303         {
304         return bn_rand_range(1, r, range);
305         }
306 #endif /* ! codereview */
```

```
**********************************************************
    6765 Wed Aug 13 19:52:17 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_recp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_recp.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"
```



```
void BN_RECP_CTX_init(BN_RECP_CTX *recp)
        {
        BN_init(&(recp->N));
        BN_init(&(recp->Nr));
        recp->num_bits=0;
        recp->flags=0;
        }

BN_RECP_CTX *BN_RECP_CTX_new(void)
        {
        BN_RECP_CTX *ret;

        if ((ret=(BN_RECP_CTX *)OPENSSL_malloc(sizeof(BN_RECP_CTX))) == NULL)
                return(NULL);

        BN_RECP_CTX_init(ret);
        ret->flags=BN_FLG_MALLOCED;
        return(ret);
        }

void BN_RECP_CTX_free(BN_RECP_CTX *recp)
        {
        if(recp == NULL)
            return;

        BN_free(&(recp->N));
        BN_free(&(recp->Nr));
        if (recp->flags & BN_FLG_MALLOCED)
                OPENSSL_free(recp);
        }

int BN_RECP_CTX_set(BN_RECP_CTX *recp, const BIGNUM *d, BN_CTX *ctx)
        {
        if (!BN_copy(&(recp->N),d)) return 0;
        BN_zero(&(recp->Nr));
        recp->num_bits=BN_num_bits(d);
        recp->shift=0;
        return(1);
        }

int BN_mod_mul_reciprocal(BIGNUM *r, const BIGNUM *x, const BIGNUM *y,
        BN_RECP_CTX *recp, BN_CTX *ctx)
        {
        int ret=0;
        BIGNUM *a;
        const BIGNUM *ca;

        BN_CTX_start(ctx);
        if ((a = BN_CTX_get(ctx)) == NULL) goto err;
        if (y != NULL)
                {
                if (x == y)
                        { if (!BN_sqr(a,x,ctx)) goto err; }
                else
                        { if (!BN_mul(a,x,y,ctx)) goto err; }
                ca = a;
                }
        else
                ca=x; /* Just do the mod */

        ret = BN_div_recp(NULL,r,ca,recp,ctx);
err:
        BN_CTX_end(ctx);
        bn_check_top(r);
        return(ret);
```

```
128         }

130 int BN_div_recp(BIGNUM *dv, BIGNUM *rem, const BIGNUM *m,
131         BN_RECP_CTX *recp, BN_CTX *ctx)
132         {
133         int i,j,ret=0;
134         BIGNUM *a,*b,*d,*r;

136         BN_CTX_start(ctx);
137         a=BN_CTX_get(ctx);
138         b=BN_CTX_get(ctx);
139         if (dv != NULL)
140                 d=dv;
141         else
142                 d=BN_CTX_get(ctx);
143         if (rem != NULL)
144                 r=rem;
145         else
146                 r=BN_CTX_get(ctx);
147         if (a == NULL || b == NULL || d == NULL || r == NULL) goto err;

149         if (BN_ucmp(m,&(recp->N)) < 0)
150                 {
151                 BN_zero(d);
152                 if (!BN_copy(r,m)) return 0;
153                 BN_CTX_end(ctx);
154                 return(1);
155                 }

157         /* We want the remainder
158          * Given input of ABCDEF / ab
159          * we need multiply ABCDEF by 3 digests of the reciprocal of ab
160          *
161          */

163         /* i := max(BN_num_bits(m), 2*BN_num_bits(N)) */
164         i=BN_num_bits(m);
165         j=recp->num_bits<<1;
166         if (j>i) i=j;

168         /* Nr := round(2^i / N) */
169         if (i != recp->shift)
170                 recp->shift=BN_reciprocal(&(recp->Nr),&(recp->N),
171                         i,ctx); /* BN_reciprocal returns i, or -1 for an error *
172         if (recp->shift == -1) goto err;

174         /* d := |round(round(m / 2^BN_num_bits(N)) * recp->Nr / 2^(i - BN_num_bi
175          *    = |round(round(m / 2^BN_num_bits(N)) * round(2^i / N) / 2^(i - BN_
176          *   <= |(m / 2^BN_num_bits(N)) * (2^i / N) * (2^BN_num_bits(N) / 2^i)|
177          *    = |m/N|
178          */
179         if (!BN_rshift(a,m,recp->num_bits)) goto err;
180         if (!BN_mul(b,a,&(recp->Nr),ctx)) goto err;
181         if (!BN_rshift(d,b,i-recp->num_bits)) goto err;
182         d->neg=0;

184         if (!BN_mul(b,&(recp->N),d,ctx)) goto err;
185         if (!BN_usub(r,m,b)) goto err;
186         r->neg=0;

188 #if 1
189         j=0;
190         while (BN_ucmp(r,&(recp->N)) >= 0)
191                 {
192                 if (j++ > 2)
193                         {
```



```
194                         BNerr(BN_F_BN_DIV_RECP,BN_R_BAD_RECIPROCAL);
195                         goto err;
196                         }
197                 if (!BN_usub(r,r,&(recp->N))) goto err;
198                 if (!BN_add_word(d,1)) goto err;
199                 }
200 #endif

202         r->neg=BN_is_zero(r)?0:m->neg;
203         d->neg=m->neg^recp->N.neg;
204         ret=1;
205 err:
206         BN_CTX_end(ctx);
207         bn_check_top(dv);
208         bn_check_top(rem);
209         return(ret);
210         }

212 /* len is the expected size of the result
213  * We actually calculate with an extra word of precision, so
214  * we can do faster division if the remainder is not required.
215  */
216 /* r := 2^len / m */
217 int BN_reciprocal(BIGNUM *r, const BIGNUM *m, int len, BN_CTX *ctx)
218         {
219         int ret= -1;
220         BIGNUM *t;

222         BN_CTX_start(ctx);
223         if((t = BN_CTX_get(ctx)) == NULL) goto err;

225         if (!BN_set_bit(t,len)) goto err;

227         if (!BN_div(r,NULL,t,m,ctx)) goto err;

229         ret=len;
230 err:
231         bn_check_top(r);
232         BN_CTX_end(ctx);
233         return(ret);
234         }
235 #endif /* ! codereview */
```

```
**********************************************************
    5642 Wed Aug 13 19:52:17 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_shift.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_shift.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"
```



```
int BN_lshift1(BIGNUM *r, const BIGNUM *a)
        {
        register BN_ULONG *ap,*rp,t,c;
        int i;

        bn_check_top(r);
        bn_check_top(a);

        if (r != a)
                {
                r->neg=a->neg;
                if (bn_wexpand(r,a->top+1) == NULL) return(0);
                r->top=a->top;
                }
        else
                {
                if (bn_wexpand(r,a->top+1) == NULL) return(0);
                }
        ap=a->d;
        rp=r->d;
        c=0;
        for (i=0; i<a->top; i++)
                {
                t= *(ap++);
                *(rp++)=((t<<1)|c)&BN_MASK2;
                c=(t & BN_TBIT)?1:0;
                }
        if (c)
                {
                *rp=1;
                r->top++;
                }
        bn_check_top(r);
        return(1);
        }

int BN_rshift1(BIGNUM *r, const BIGNUM *a)
        {
        BN_ULONG *ap,*rp,t,c;
        int i,j;

        bn_check_top(r);
        bn_check_top(a);

        if (BN_is_zero(a))
                {
                BN_zero(r);
                return(1);
                }
        i = a->top;
        ap= a->d;
        j = i-(ap[i-1]==1);
        if (a != r)
                {
                if (bn_wexpand(r,j) == NULL) return(0);
                r->neg=a->neg;
                }
        rp=r->d;
        t=ap[--i];
        c=(t&1)?BN_TBIT:0;
        if (t>>=1) rp[i]=t;
        while (i>0)
                {
                t=ap[--i];
                rp[i]=((t>>1)&BN_MASK2)|c;
```

```
128                 c=(t&1)?BN_TBIT:0;
129                 }
130         r->top=j;
131         bn_check_top(r);
132         return(1);
133         }

135 int BN_lshift(BIGNUM *r, const BIGNUM *a, int n)
136         {
137         int i,nw,lb,rb;
138         BN_ULONG *t,*f;
139         BN_ULONG l;

141         bn_check_top(r);
142         bn_check_top(a);

144         r->neg=a->neg;
145         nw=n/BN_BITS2;
146         if (bn_wexpand(r,a->top+nw+1) == NULL) return(0);
147         lb=n%BN_BITS2;
148         rb=BN_BITS2-lb;
149         f=a->d;
150         t=r->d;
151         t[a->top+nw]=0;
152         if (lb == 0)
153                 for (i=a->top-1; i>=0; i--)
154                         t[nw+i]=f[i];
155         else
156                 for (i=a->top-1; i>=0; i--)
157                         {
158                         l=f[i];
159                         t[nw+i+1]|=(l>>rb)&BN_MASK2;
160                         t[nw+i]=(l<<lb)&BN_MASK2;
161                         }
162         memset(t,0,nw*sizeof(t[0]));
163 /*      for (i=0; i<nw; i++)
164                 t[i]=0;*/
165         r->top=a->top+nw+1;
166         bn_correct_top(r);
167         bn_check_top(r);
168         return(1);
169         }

171 int BN_rshift(BIGNUM *r, const BIGNUM *a, int n)
172         {
173         int i,j,nw,lb,rb;
174         BN_ULONG *t,*f;
175         BN_ULONG l,tmp;

177         bn_check_top(r);
178         bn_check_top(a);

180         nw=n/BN_BITS2;
181         rb=n%BN_BITS2;
182         lb=BN_BITS2-rb;
183         if (nw >= a->top || a->top == 0)
184                 {
185                 BN_zero(r);
186                 return(1);
187                 }
188         i = (BN_num_bits(a)-n+(BN_BITS2-1))/BN_BITS2;
189         if (r != a)
190                 {
191                 r->neg=a->neg;
192                 if (bn_wexpand(r,i) == NULL) return(0);
193                 }
```



```
194         else
195                 {
196                 if (n == 0)
197                         return 1; /* or the copying loop will go berserk */
198                 }

200         f= &(a->d[nw]);
201         t=r->d;
202         j=a->top-nw;
203         r->top=i;

205         if (rb == 0)
206                 {
207                 for (i=j; i != 0; i--)
208                         *(t++)= *(f++);
209                 }
210         else
211                 {
212                 l= *(f++);
213                 for (i=j-1; i != 0; i--)
214                         {
215                         tmp =(l>>rb)&BN_MASK2;
216                         l= *(f++);
217                         *(t++) =(tmp|(l<<lb))&BN_MASK2;
218                         }
219                 if ((l = (l>>rb)&BN_MASK2)) *(t) = l;
220                 }
221         bn_check_top(r);
222         return(1);
223         }
224 #endif /* ! codereview */
```

```
**********************************************************
    7537 Wed Aug 13 19:52:17 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_sqr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_sqr.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"
```



```
/* r must not be a */
/* I've just gone over this and it is now %20 faster on x86 - eay - 27 Jun 96 */
int BN_sqr(BIGNUM *r, const BIGNUM *a, BN_CTX *ctx)
        {
        int max,al;
        int ret = 0;
        BIGNUM *tmp,*rr;

#ifdef BN_COUNT
        fprintf(stderr,"BN_sqr %d * %d\n",a->top,a->top);
#endif
        bn_check_top(a);

        al=a->top;
        if (al <= 0)
                {
                r->top=0;
                r->neg = 0;
                return 1;
                }

        BN_CTX_start(ctx);
        rr=(a != r) ? r : BN_CTX_get(ctx);
        tmp=BN_CTX_get(ctx);
        if (!rr || !tmp) goto err;

        max = 2 * al; /* Non-zero (from above) */
        if (bn_wexpand(rr,max) == NULL) goto err;

        if (al == 4)
                {
#ifndef BN_SQR_COMBA
                BN_ULONG t[8];
                bn_sqr_normal(rr->d,a->d,4,t);
#else
                bn_sqr_comba4(rr->d,a->d);
#endif
                }
        else if (al == 8)
                {
#ifndef BN_SQR_COMBA
                BN_ULONG t[16];
                bn_sqr_normal(rr->d,a->d,8,t);
#else
                bn_sqr_comba8(rr->d,a->d);
#endif
                }
        else
                {
#if defined(BN_RECURSION)
                if (al < BN_SQR_RECURSIVE_SIZE_NORMAL)
                        {
                        BN_ULONG t[BN_SQR_RECURSIVE_SIZE_NORMAL*2];
                        bn_sqr_normal(rr->d,a->d,al,t);
                        }
                else
                        {
                        int j,k;

                        j=BN_num_bits_word((BN_ULONG)al);
                        j=1<<(j-1);
                        k=j+j;
                        if (al == j)
                                {
                                if (bn_wexpand(tmp,k*2) == NULL) goto err;
```

```
128                                 bn_sqr_recursive(rr->d,a->d,al,tmp->d);
129                                 }
130                         else
131                                 {
132                                 if (bn_wexpand(tmp,max) == NULL) goto err;
133                                 bn_sqr_normal(rr->d,a->d,al,tmp->d);
134                                 }
135                         }
136 #else
137                 if (bn_wexpand(tmp,max) == NULL) goto err;
138                 bn_sqr_normal(rr->d,a->d,al,tmp->d);
139 #endif
140                 }

142         rr->neg=0;
143         /* If the most-significant half of the top word of 'a' is zero, then
144          * the square of 'a' will max-1 words. */
145         if(a->d[al - 1] == (a->d[al - 1] & BN_MASK2l))
146                 rr->top = max - 1;
147         else
148                 rr->top = max;
149         if (rr != r) BN_copy(r,rr);
150         ret = 1;
151  err:
152         bn_check_top(rr);
153         bn_check_top(tmp);
154         BN_CTX_end(ctx);
155         return(ret);
156         }

158 /* tmp must have 2*n words */
159 void bn_sqr_normal(BN_ULONG *r, const BN_ULONG *a, int n, BN_ULONG *tmp)
160         {
161         int i,j,max;
162         const BN_ULONG *ap;
163         BN_ULONG *rp;

165         max=n*2;
166         ap=a;
167         rp=r;
168         rp[0]=rp[max-1]=0;
169         rp++;
170         j=n;

172         if (--j > 0)
173                 {
174                 ap++;
175                 rp[j]=bn_mul_words(rp,ap,j,ap[-1]);
176                 rp+=2;
177                 }

179         for (i=n-2; i>0; i--)
180                 {
181                 j--;
182                 ap++;
183                 rp[j]=bn_mul_add_words(rp,ap,j,ap[-1]);
184                 rp+=2;
185                 }

187         bn_add_words(r,r,r,max);

189         /* There will not be a carry */

191         bn_sqr_words(tmp,a,n);

193         bn_add_words(r,r,tmp,max);
```



```
194         }

196 #ifdef BN_RECURSION
197 /* r is 2*n words in size,
198  * a and b are both n words in size.    (There's not actually a 'b' here ...)
199  * n must be a power of 2.
200  * We multiply and return the result.
201  * t must be 2*n words in size
202  * We calculate
203  * a[0]*b[0]
204  * a[0]*b[0]+a[1]*b[1]+(a[0]-a[1])*(b[1]-b[0])
205  * a[1]*b[1]
206  */
207 void bn_sqr_recursive(BN_ULONG *r, const BN_ULONG *a, int n2, BN_ULONG *t)
208         {
209         int n=n2/2;
210         int zero,c1;
211         BN_ULONG ln,lo,*p;

213 #ifdef BN_COUNT
214         fprintf(stderr," bn_sqr_recursive %d * %d\n",n2,n2);
215 #endif
216         if (n2 == 4)
217                 {
218 #ifndef BN_SQR_COMBA
219                 bn_sqr_normal(r,a,4,t);
220 #else
221                 bn_sqr_comba4(r,a);
222 #endif
223                 return;
224                 }
225         else if (n2 == 8)
226                 {
227 #ifndef BN_SQR_COMBA
228                 bn_sqr_normal(r,a,8,t);
229 #else
230                 bn_sqr_comba8(r,a);
231 #endif
232                 return;
233                 }
234         if (n2 < BN_SQR_RECURSIVE_SIZE_NORMAL)
235                 {
236                 bn_sqr_normal(r,a,n2,t);
237                 return;
238                 }
239         /* r=(a[0]-a[1])*(a[1]-a[0]) */
240         c1=bn_cmp_words(a,&(a[n]),n);
241         zero=0;
242         if (c1 > 0)
243                 bn_sub_words(t,a,&(a[n]),n);
244         else if (c1 < 0)
245                 bn_sub_words(t,&(a[n]),a,n);
246         else
247                 zero=1;

249         /* The result will always be negative unless it is zero */
250         p= &(t[n2*2]);

252         if (!zero)
253                 bn_sqr_recursive(&(t[n2]),t,n,p);
254         else
255                 memset(&(t[n2]),0,n2*sizeof(BN_ULONG));
256         bn_sqr_recursive(r,a,n,p);
257         bn_sqr_recursive(&(r[n2]),&(a[n]),n,p);

259         /* t[32] holds (a[0]-a[1])*(a[1]-a[0]), it is negative or zero
```

```
260          * r[10] holds (a[0]*b[0])
261          * r[32] holds (b[1]*b[1])
262          */
263
264         c1=(int)(bn_add_words(t,r,&(r[n2]),n2));
265
266         /* t[32] is negative */
267         c1-=(int)(bn_sub_words(&(t[n2]),t,&(t[n2]),n2));
268
269         /* t[32] holds (a[0]-a[1])*(a[1]-a[0])+(a[0]*a[0])+(a[1]*a[1])
270          * r[10] holds (a[0]*a[0])
271          * r[32] holds (a[1]*a[1])
272          * c1 holds the carry bits
273          */
274         c1+=(int)(bn_add_words(&(r[n]),&(r[n]),&(t[n2]),n2));
275         if (c1)
276                 {
277                 p= &(r[n+n2]);
278                 lo= *p;
279                 ln=(lo+c1)&BN_MASK2;
280                 *p=ln;
281
282                 /* The overflow will stop before we over write
283                  * words we should not overwrite */
284                 if (ln < (BN_ULONG)c1)
285                         {
286                         do      {
287                                 p++;
288                                 lo= *p;
289                                 ln=(lo+1)&BN_MASK2;
290                                 *p=ln;
291                                 } while (ln == 0);
292                         }
293                 }
294         }
295 #endif
296 #endif /* ! codereview */
```

```
**********************************************************
    9986 Wed Aug 13 19:52:17 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_sqrt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_sqrt.c */
/* Written by Lenka Fibikova <fibikova@exp-math.uni-essen.de>
 * and Bodo Moeller for the OpenSSL project. */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include "bn_lcl.h"
```



```
BIGNUM *BN_mod_sqrt(BIGNUM *in, const BIGNUM *a, const BIGNUM *p, BN_CTX *ctx)
/* Returns 'ret' such that
 *      ret^2 == a (mod p),
 * using the Tonelli/Shanks algorithm (cf. Henri Cohen, "A Course
 * in Algebraic Computational Number Theory", algorithm 1.5.1).
 * 'p' must be prime!
 */
        {
        BIGNUM *ret = in;
        int err = 1;
        int r;
        BIGNUM *A, *b, *q, *t, *x, *y;
        int e, i, j;

        if (!BN_is_odd(p) || BN_abs_is_word(p, 1))
                {
                if (BN_abs_is_word(p, 2))
                        {
                        if (ret == NULL)
                                ret = BN_new();
                        if (ret == NULL)
                                goto end;
                        if (!BN_set_word(ret, BN_is_bit_set(a, 0)))
                                {
                                if (ret != in)
                                        BN_free(ret);
                                return NULL;
                                }
                        bn_check_top(ret);
                        return ret;
                        }

                BNerr(BN_F_BN_MOD_SQRT, BN_R_P_IS_NOT_PRIME);
                return(NULL);
                }

        if (BN_is_zero(a) || BN_is_one(a))
                {
                if (ret == NULL)
                        ret = BN_new();
                if (ret == NULL)
                        goto end;
                if (!BN_set_word(ret, BN_is_one(a)))
                        {
                        if (ret != in)
                                BN_free(ret);
                        return NULL;
                        }
                bn_check_top(ret);
                return ret;
                }

        BN_CTX_start(ctx);
        A = BN_CTX_get(ctx);
        b = BN_CTX_get(ctx);
        q = BN_CTX_get(ctx);
        t = BN_CTX_get(ctx);
        x = BN_CTX_get(ctx);
        y = BN_CTX_get(ctx);
        if (y == NULL) goto end;

        if (ret == NULL)
                ret = BN_new();
        if (ret == NULL) goto end;

        /* A = a mod p */
```

```
128         if (!BN_nnmod(A, a, p, ctx)) goto end;

130         /* now write  |p| - 1  as  2^e*q  where  q  is odd */
131         e = 1;
132         while (!BN_is_bit_set(p, e))
133                 e++;
134         /* we'll set  q  later (if needed) */

136         if (e == 1)
137                 {
138                 /* The easy case:  (|p|-1)/2  is odd, so 2 has an inverse
139                  * modulo  (|p|-1)/2,  and square roots can be computed
140                  * directly by modular exponentiation.
141                  * We have
142                  *     2 * (|p|+1)/4 == 1   (mod (|p|-1)/2),
143                  * so we can use exponent  (|p|+1)/4,  i.e.  (|p|-3)/4 + 1.
144                  */
145                 if (!BN_rshift(q, p, 2)) goto end;
146                 q->neg = 0;
147                 if (!BN_add_word(q, 1)) goto end;
148                 if (!BN_mod_exp(ret, A, q, p, ctx)) goto end;
149                 err = 0;
150                 goto vrfy;
151                 }

153         if (e == 2)
154                 {
155                 /* |p| == 5  (mod 8)
156                  *
157                  * In this case  2  is always a non-square since
158                  * Legendre(2,p) = (-1)^((p^2-1)/8)  for any odd prime.
159                  * So if  a  really is a square, then  2*a  is a non-square.
160                  * Thus for
161                  *      b := (2*a)^((|p|-5)/8),
162                  *      i := (2*a)*b^2
163                  * we have
164                  *     i^2 = (2*a)^((1 + (|p|-5)/4)*2)
165                  *         = (2*a)^((p-1)/2)
166                  *         = -1;
167                  * so if we set
168                  *       x := a*b*(i-1),
169                  * then
170                  *     x^2 = a^2 * b^2 * (i^2 - 2*i + 1)
171                  *         = a^2 * b^2 * (-2*i)
172                  *         = a*(-i)*(2*a*b^2)
173                  *         = a*(-i)*i
174                  *         = a.
175                  *
176                  * (This is due to A.O.L. Atkin,
177                  * <URL: http://listserv.nodak.edu/scripts/wa.exe?A2=ind9211&L=n
178                  * November 1992.)
179                  */

181                 /* t := 2*a */
182                 if (!BN_mod_lshift1_quick(t, A, p)) goto end;

184                 /* b := (2*a)^((|p|-5)/8) */
185                 if (!BN_rshift(q, p, 3)) goto end;
186                 q->neg = 0;
187                 if (!BN_mod_exp(b, t, q, p, ctx)) goto end;

189                 /* y := b^2 */
190                 if (!BN_mod_sqr(y, b, p, ctx)) goto end;

192                 /* t := (2*a)*b^2 - 1*/
193                 if (!BN_mod_mul(t, t, y, p, ctx)) goto end;
```



```
194                 if (!BN_sub_word(t, 1)) goto end;

196                 /* x = a*b*t */
197                 if (!BN_mod_mul(x, A, b, p, ctx)) goto end;
198                 if (!BN_mod_mul(x, x, t, p, ctx)) goto end;

200                 if (!BN_copy(ret, x)) goto end;
201                 err = 0;
202                 goto vrfy;
203                 }

205         /* e > 2, so we really have to use the Tonelli/Shanks algorithm.
206          * First, find some  y  that is not a square. */
207         if (!BN_copy(q, p)) goto end; /* use 'q' as temp */
208         q->neg = 0;
209         i = 2;
210         do
211                 {
212                 /* For efficiency, try small numbers first;
213                  * if this fails, try random numbers.
214                  */
215                 if (i < 22)
216                         {
217                         if (!BN_set_word(y, i)) goto end;
218                         }
219                 else
220                         {
221                         if (!BN_pseudo_rand(y, BN_num_bits(p), 0, 0)) goto end;
222                         if (BN_ucmp(y, p) >= 0)
223                                 {
224                                 if (!(p->neg ? BN_add : BN_sub)(y, y, p)) goto e
225                                 }
226                         /* now 0 <= y < |p| */
227                         if (BN_is_zero(y))
228                                 if (!BN_set_word(y, i)) goto end;
229                         }

231                 r = BN_kronecker(y, q, ctx); /* here 'q' is |p| */
232                 if (r < -1) goto end;
233                 if (r == 0)
234                         {
235                         /* m divides p */
236                         BNerr(BN_F_BN_MOD_SQRT, BN_R_P_IS_NOT_PRIME);
237                         goto end;
238                         }
239                 }
240         while (r == 1 && ++i < 82);

242         if (r != -1)
243                 {
244                 /* Many rounds and still no non-square -- this is more likely
245                  * a bug than just bad luck.
246                  * Even if  p  is not prime, we should have found some  y
247                  * such that r == -1.
248                  */
249                 BNerr(BN_F_BN_MOD_SQRT, BN_R_TOO_MANY_ITERATIONS);
250                 goto end;
251                 }

253         /* Here's our actual 'q': */
254         if (!BN_rshift(q, q, e)) goto end;

256         /* Now that we have some non-square, we can find an element
257          * of order  2^e  by computing its q'th power. */
258         if (!BN_mod_exp(y, y, q, p, ctx)) goto end;
259         if (BN_is_one(y))
```

```
260                 {
261                 BNerr(BN_F_BN_MOD_SQRT, BN_R_P_IS_NOT_PRIME);
262                 goto end;
263                 }

265         /* Now we know that (if  p  is indeed prime) there is an integer
266          * k,  0 <= k < 2^e,  such that
267          *
268          *      a^q * y^k == 1   (mod p).
269          *
270          * As  a^q  is a square and  y  is not,  k  must be even.
271          * q+1  is even, too, so there is an element
272          *
273          *     X := a^((q+1)/2) * y^(k/2),
274          *
275          * and it satisfies
276          *
277          *     X^2 = a^q * a     * y^k
278          *         = a,
279          *
280          * so it is the square root that we are looking for.
281          */

283         /* t := (q-1)/2  (note that  q  is odd) */
284         if (!BN_rshift1(t, q)) goto end;

286         /* x := a^((q-1)/2) */
287         if (BN_is_zero(t)) /* special case: p = 2^e + 1 */
288                 {
289                 if (!BN_nnmod(t, A, p, ctx)) goto end;
290                 if (BN_is_zero(t))
291                         {
292                         /* special case: a == 0  (mod p) */
293                         BN_zero(ret);
294                         err = 0;
295                         goto end;
296                         }
297                 else
298                         if (!BN_one(x)) goto end;
299                 }
300         else
301                 {
302                 if (!BN_mod_exp(x, A, t, p, ctx)) goto end;
303                 if (BN_is_zero(x))
304                         {
305                         /* special case: a == 0  (mod p) */
306                         BN_zero(ret);
307                         err = 0;
308                         goto end;
309                         }
310                 }

312         /* b := a*x^2  (= a^q) */
313         if (!BN_mod_sqr(b, x, p, ctx)) goto end;
314         if (!BN_mod_mul(b, b, A, p, ctx)) goto end;

316         /* x := a*x    (= a^((q+1)/2)) */
317         if (!BN_mod_mul(x, x, A, p, ctx)) goto end;

319         while (1)
320                 {
321                 /* Now  b  is  a^q * y^k  for some even  k  (0 <= k < 2^E
322                  * where  E  refers to the original value of  e,  which we
323                  * don't keep in a variable),  and  x  is  a^((q+1)/2) * y^(k/2)
324                  *
325                  * We have  a*b = x^2,
```



```
326                  *    y^2^(e-1) = -1,
327                  *    b^2^(e-1) = 1.
328                  */

330                 if (BN_is_one(b))
331                         {
332                         if (!BN_copy(ret, x)) goto end;
333                         err = 0;
334                         goto vrfy;
335                         }


338                 /* find smallest  i  such that  b^(2^i) = 1 */
339                 i = 1;
340                 if (!BN_mod_sqr(t, b, p, ctx)) goto end;
341                 while (!BN_is_one(t))
342                         {
343                         i++;
344                         if (i == e)
345                                 {
346                                 BNerr(BN_F_BN_MOD_SQRT, BN_R_NOT_A_SQUARE);
347                                 goto end;
348                                 }
349                         if (!BN_mod_mul(t, t, t, p, ctx)) goto end;
350                         }


353                 /* t := y^2^(e - i - 1) */
354                 if (!BN_copy(t, y)) goto end;
355                 for (j = e - i - 1; j > 0; j--)
356                         {
357                         if (!BN_mod_sqr(t, t, p, ctx)) goto end;
358                         }
359                 if (!BN_mod_mul(y, t, t, p, ctx)) goto end;
360                 if (!BN_mod_mul(x, x, t, p, ctx)) goto end;
361                 if (!BN_mod_mul(b, b, y, p, ctx)) goto end;
362                 e = i;
363                 }

365 vrfy:
366         if (!err)
367                 {
368                 /* verify the result -- the input might have been not a square
369                  * (test added in 0.9.8) */

371                 if (!BN_mod_sqr(x, ret, p, ctx))
372                         err = 1;

374                 if (!err && 0 != BN_cmp(x, A))
375                         {
376                         BNerr(BN_F_BN_MOD_SQRT, BN_R_NOT_A_SQUARE);
377                         err = 1;
378                         }
379                 }

381 end:
382         if (err)
383                 {
384                 if (ret != NULL && ret != in)
385                         {
386                         BN_clear_free(ret);
387                         }
388                 ret = NULL;
389                 }
390         BN_CTX_end(ctx);
391         bn_check_top(ret);
```

```
392         return ret;
393         }
394 #endif /* ! codereview */
```

```
**********************************************************
    5996 Wed Aug 13 19:52:18 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_word.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/bn/bn_word.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include "bn_lcl.h"
```



```
BN_ULONG BN_mod_word(const BIGNUM *a, BN_ULONG w)
        {
#ifndef BN_LLONG
        BN_ULONG ret=0;
#else
        BN_ULLONG ret=0;
#endif
        int i;

        if (w == 0)
                return (BN_ULONG)-1;

        bn_check_top(a);
        w&=BN_MASK2;
        for (i=a->top-1; i>=0; i--)
                {
#ifndef BN_LLONG
                ret=((ret<<BN_BITS4)|((a->d[i]>>BN_BITS4)&BN_MASK2l))%w;
                ret=((ret<<BN_BITS4)|(a->d[i]&BN_MASK2l))%w;
#else
                ret=(BN_ULLONG)(((ret<<(BN_ULLONG)BN_BITS2)|a->d[i])%
                        (BN_ULLONG)w);
#endif
                }
        return((BN_ULONG)ret);
        }

BN_ULONG BN_div_word(BIGNUM *a, BN_ULONG w)
        {
        BN_ULONG ret = 0;
        int i, j;

        bn_check_top(a);
        w &= BN_MASK2;

        if (!w)
                /* actually this an error (division by zero) */
                return (BN_ULONG)-1;
        if (a->top == 0)
                return 0;

        /* normalize input (so bn_div_words doesn't complain) */
        j = BN_BITS2 - BN_num_bits_word(w);
        w <<= j;
        if (!BN_lshift(a, a, j))
                return (BN_ULONG)-1;

        for (i=a->top-1; i>=0; i--)
                {
                BN_ULONG l,d;

                l=a->d[i];
                d=bn_div_words(ret,l,w);
                ret=(l-((d*w)&BN_MASK2))&BN_MASK2;
                a->d[i]=d;
                }
        if ((a->top > 0) && (a->d[a->top-1] == 0))
                a->top--;
        ret >>= j;
        bn_check_top(a);
        return(ret);
        }

int BN_add_word(BIGNUM *a, BN_ULONG w)
        {
```

```
128         BN_ULONG l;
129         int i;

131         bn_check_top(a);
132         w &= BN_MASK2;

134         /* degenerate case: w is zero */
135         if (!w) return 1;
136         /* degenerate case: a is zero */
137         if(BN_is_zero(a)) return BN_set_word(a, w);
138         /* handle 'a' when negative */
139         if (a->neg)
140                 {
141                 a->neg=0;
142                 i=BN_sub_word(a,w);
143                 if (!BN_is_zero(a))
144                         a->neg=!(a->neg);
145                 return(i);
146                 }
147         for (i=0;w!=0 && i<a->top;i++)
148                 {
149                 a->d[i] = l = (a->d[i]+w)&BN_MASK2;
150                 w = (w>l)?1:0;
151                 }
152         if (w && i==a->top)
153                 {
154                 if (bn_wexpand(a,a->top+1) == NULL) return 0;
155                 a->top++;
156                 a->d[i]=w;
157                 }
158         bn_check_top(a);
159         return(1);
160         }

162 int BN_sub_word(BIGNUM *a, BN_ULONG w)
163         {
164         int i;

166         bn_check_top(a);
167         w &= BN_MASK2;

169         /* degenerate case: w is zero */
170         if (!w) return 1;
171         /* degenerate case: a is zero */
172         if(BN_is_zero(a))
173                 {
174                 i = BN_set_word(a,w);
175                 if (i != 0)
176                         BN_set_negative(a, 1);
177                 return i;
178                 }
179         /* handle 'a' when negative */
180         if (a->neg)
181                 {
182                 a->neg=0;
183                 i=BN_add_word(a,w);
184                 a->neg=1;
185                 return(i);
186                 }

188         if ((a->top == 1) && (a->d[0] < w))
189                 {
190                 a->d[0]=w-a->d[0];
191                 a->neg=1;
192                 return(1);
193                 }
```



```
194         i=0;
195         for (;;)
196                 {
197                 if (a->d[i] >= w)
198                         {
199                         a->d[i]-=w;
200                         break;
201                         }
202                 else
203                         {
204                         a->d[i]=(a->d[i]-w)&BN_MASK2;
205                         i++;
206                         w=1;
207                         }
208                 }
209         if ((a->d[i] == 0) && (i == (a->top-1)))
210                 a->top--;
211         bn_check_top(a);
212         return(1);
213         }

215 int BN_mul_word(BIGNUM *a, BN_ULONG w)
216         {
217         BN_ULONG ll;

219         bn_check_top(a);
220         w&=BN_MASK2;
221         if (a->top)
222                 {
223                 if (w == 0)
224                         BN_zero(a);
225                 else
226                         {
227                         ll=bn_mul_words(a->d,a->d,a->top,w);
228                         if (ll)
229                                 {
230                                 if (bn_wexpand(a,a->top+1) == NULL) return(0);
231                                 a->d[a->top++]=ll;
232                                 }
233                         }
234                 }
235         bn_check_top(a);
236         return(1);
237         }
238 #endif /* ! codereview */
```

```
**********************************************************
    6926 Wed Aug 13 19:52:18 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/bn_x931p.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* bn_x931p.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2005.
 */
/* ====================================================================
 * Copyright (c) 2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/bn.h>
```



```
/* X9.31 routines for prime derivation */

/* X9.31 prime derivation. This is used to generate the primes pi
 * (p1, p2, q1, q2) from a parameter Xpi by checking successive odd
 * integers.
 */

static int bn_x931_derive_pi(BIGNUM *pi, const BIGNUM *Xpi, BN_CTX *ctx,
                        BN_GENCB *cb)
        {
        int i = 0;
        if (!BN_copy(pi, Xpi))
                return 0;
        if (!BN_is_odd(pi) && !BN_add_word(pi, 1))
                return 0;
        for(;;)
                {
                i++;
                BN_GENCB_call(cb, 0, i);
                /* NB 27 MR is specificed in X9.31 */
                if (BN_is_prime_fasttest_ex(pi, 27, ctx, 1, cb))
                        break;
                if (!BN_add_word(pi, 2))
                        return 0;
                }
        BN_GENCB_call(cb, 2, i);
        return 1;
        }

/* This is the main X9.31 prime derivation function. From parameters
 * Xp1, Xp2 and Xp derive the prime p. If the parameters p1 or p2 are
 * not NULL they will be returned too: this is needed for testing.
 */

int BN_X931_derive_prime_ex(BIGNUM *p, BIGNUM *p1, BIGNUM *p2,
                        const BIGNUM *Xp, const BIGNUM *Xp1, const BIGNUM *Xp2,
                        const BIGNUM *e, BN_CTX *ctx, BN_GENCB *cb)
        {
        int ret = 0;

        BIGNUM *t, *p1p2, *pm1;

        /* Only even e supported */
        if (!BN_is_odd(e))
                return 0;

        BN_CTX_start(ctx);
        if (!p1)
                p1 = BN_CTX_get(ctx);

        if (!p2)
                p2 = BN_CTX_get(ctx);

        t = BN_CTX_get(ctx);

        p1p2 = BN_CTX_get(ctx);

        pm1 = BN_CTX_get(ctx);

        if (!bn_x931_derive_pi(p1, Xp1, ctx, cb))
                goto err;

        if (!bn_x931_derive_pi(p2, Xp2, ctx, cb))
                goto err;

        if (!BN_mul(p1p2, p1, p2, ctx))
```

```
128                 goto err;
129
130         /* First set p to value of Rp */
131
132         if (!BN_mod_inverse(p, p2, p1, ctx))
133                 goto err;
134
135         if (!BN_mul(p, p, p2, ctx))
136                 goto err;
137
138         if (!BN_mod_inverse(t, p1, p2, ctx))
139                 goto err;
140
141         if (!BN_mul(t, t, p1, ctx))
142                 goto err;
143
144         if (!BN_sub(p, p, t))
145                 goto err;
146
147         if (p->neg && !BN_add(p, p, p1p2))
148                 goto err;
149
150         /* p now equals Rp */
151
152         if (!BN_mod_sub(p, p, Xp, p1p2, ctx))
153                 goto err;
154
155         if (!BN_add(p, p, Xp))
156                 goto err;
157
158         /* p now equals Yp0 */
159
160         for (;;)
161                 {
162                 int i = 1;
163                 BN_GENCB_call(cb, 0, i++);
164                 if (!BN_copy(pm1, p))
165                         goto err;
166                 if (!BN_sub_word(pm1, 1))
167                         goto err;
168                 if (!BN_gcd(t, pm1, e, ctx))
169                         goto err;
170                 if (BN_is_one(t)
171                 /* X9.31 specifies 8 MR and 1 Lucas test or any prime test
172                  * offering similar or better guarantees 50 MR is considerably
173                  * better.
174                  */
175                         && BN_is_prime_fasttest_ex(p, 50, ctx, 1, cb))
176                         break;
177                 if (!BN_add(p, p, p1p2))
178                         goto err;
179                 }
180
181         BN_GENCB_call(cb, 3, 0);
182
183         ret = 1;
184
185         err:
186
187         BN_CTX_end(ctx);
188
189         return ret;
190         }
191
192 /* Generate pair of paramters Xp, Xq for X9.31 prime generation.
193  * Note: nbits paramter is sum of number of bits in both.
```



```
194  */
195
196 int BN_X931_generate_Xpq(BIGNUM *Xp, BIGNUM *Xq, int nbits, BN_CTX *ctx)
197         {
198         BIGNUM *t;
199         int i;
200         /* Number of bits for each prime is of the form
201          * 512+128s for s = 0, 1, ...
202          */
203         if ((nbits < 1024) || (nbits & 0xff))
204                 return 0;
205         nbits >>= 1;
206         /* The random value Xp must be between sqrt(2) * 2^(nbits-1) and
207          * 2^nbits - 1. By setting the top two bits we ensure that the lower
208          * bound is exceeded.
209          */
210         if (!BN_rand(Xp, nbits, 1, 0))
211                 return 0;
212
213         BN_CTX_start(ctx);
214         t = BN_CTX_get(ctx);
215
216         for (i = 0; i < 1000; i++)
217                 {
218                 if (!BN_rand(Xq, nbits, 1, 0))
219                         return 0;
220                 /* Check that |Xp - Xq| > 2^(nbits - 100) */
221                 BN_sub(t, Xp, Xq);
222                 if (BN_num_bits(t) > (nbits - 100))
223                         break;
224                 }
225
226         BN_CTX_end(ctx);
227
228         if (i < 1000)
229                 return 1;
230
231         return 0;
232
233         }
234
235 /* Generate primes using X9.31 algorithm. Of the values p, p1, p2, Xp1
236  * and Xp2 only 'p' needs to be non-NULL. If any of the others are not NULL
237  * the relevant parameter will be stored in it.
238  *
239  * Due to the fact that |Xp - Xq| > 2^(nbits - 100) must be satisfied Xp and Xq
240  * are generated using the previous function and supplied as input.
241  */
242
243 int BN_X931_generate_prime_ex(BIGNUM *p, BIGNUM *p1, BIGNUM *p2,
244                         BIGNUM *Xp1, BIGNUM *Xp2,
245                         const BIGNUM *Xp,
246                         const BIGNUM *e, BN_CTX *ctx,
247                         BN_GENCB *cb)
248         {
249         int ret = 0;
250
251         BN_CTX_start(ctx);
252         if (!Xp1)
253                 Xp1 = BN_CTX_get(ctx);
254         if (!Xp2)
255                 Xp2 = BN_CTX_get(ctx);
256
257         if (!BN_rand(Xp1, 101, 0, 0))
258                 goto error;
259         if (!BN_rand(Xp2, 101, 0, 0))
```

```
260                 goto error;
261         if (!BN_X931_derive_prime_ex(p, p1, p2, Xp, Xp1, Xp2, e, ctx, cb))
262                 goto error;

264         ret = 1;

266         error:
267         BN_CTX_end(ctx);

269         return ret;

271         }
272 #endif /* ! codereview */
```

```
**********************************************************
   13606 Wed Aug 13 19:52:18 2014
new/usr/src/lib/openssl/libsunw_crypto/bn/x86_64-gcc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#include <bn_lcl.h>
#if !(defined(__GNUC__) && __GNUC__>=2)
# include "bn_asm.c"    /* kind of dirty hack for Sun Studio */
#else
/*
 * x86_64 BIGNUM accelerator version 0.1, December 2002.
 *
 * Implemented by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
 * project.
 *
 * Rights for redistribution and usage in source and binary forms are
 * granted according to the OpenSSL license. Warranty of any kind is
 * disclaimed.
 *
 * Q. Version 0.1? It doesn't sound like Andy, he used to assign real
 *    versions, like 1.0...
 * A. Well, that's because this code is basically a quick-n-dirty
 *    proof-of-concept hack. As you can see it's implemented with
 *    inline assembler, which means that you're bound to GCC and that
 *    there might be enough room for further improvement.
 *
 * Q. Why inline assembler?
 * A. x86_64 features own ABI which I'm not familiar with. This is
 *    why I decided to let the compiler take care of subroutine
 *    prologue/epilogue as well as register allocation. For reference.
 *    Win64 implements different ABI for AMD64, different from Linux.
 *
 * Q. How much faster does it get?
 * A. 'apps/openssl speed rsa dsa' output with no-asm:
 *
 *                      sign    verify    sign/s verify/s
 *      rsa  512 bits   0.0006s   0.0001s   1683.8  18456.2
 *      rsa 1024 bits   0.0028s   0.0002s    356.0   6407.0
 *      rsa 2048 bits   0.0172s   0.0005s     58.0   1957.8
 *      rsa 4096 bits   0.1155s   0.0018s      8.7    555.6
 *                      sign    verify    sign/s verify/s
 *      dsa  512 bits   0.0005s   0.0006s   2100.8   1768.3
 *      dsa 1024 bits   0.0014s   0.0018s    692.3    559.2
 *      dsa 2048 bits   0.0049s   0.0061s    204.7    165.0
 *
 *    'apps/openssl speed rsa dsa' output with this module:
 *
 *                      sign    verify    sign/s verify/s
 *      rsa  512 bits   0.0004s   0.0000s   2767.1  33297.9
 *      rsa 1024 bits   0.0012s   0.0001s    867.4  14674.7
 *      rsa 2048 bits   0.0061s   0.0002s    164.0   5270.0
 *      rsa 4096 bits   0.0384s   0.0006s     26.1   1650.8
 *                      sign    verify    sign/s verify/s
 *      dsa  512 bits   0.0002s   0.0003s   4442.2   3786.3
 *      dsa 1024 bits   0.0005s   0.0007s   1835.1   1497.4
 *      dsa 2048 bits   0.0016s   0.0020s    620.4    504.6
 *
 *    For the reference. IA-32 assembler implementation performs
 *    very much like 64-bit code compiled with no-asm on the same
 *    machine.
 */

#ifdef _WIN64
#define BN_ULONG unsigned long long
#else
#define BN_ULONG unsigned long
```



```
#endif

#undef mul
#undef mul_add
#undef sqr

/*
 * "m"(a), "+m"(r)      is the way to favor DirectPath µ-code;
 * "g"(0)               let the compiler to decide where does it
 *                      want to keep the value of zero;
 */
#define mul_add(r,a,word,carry) do {    \
        register BN_ULONG high,low;     \
        __asm__ ("mulq %3"                      \
                : "=a"(low),"=d"(high)  \
                : "a"(word),"m"(a)      \
                : "cc");                \
        __asm__ ("addq %2,%0; adcq %3,%1"       \
                : "+r"(carry),"+d"(high)\
                : "a"(low),"g"(0)       \
                : "cc");                \
        __asm__ ("addq %2,%0; adcq %3,%1"       \
                : "+m"(r),"+d"(high)    \
                : "r"(carry),"g"(0)     \
                : "cc");                \
        carry=high;                     \
        } while (0)

#define mul(r,a,word,carry) do {        \
        register BN_ULONG high,low;     \
        __asm__ ("mulq %3"                      \
                : "=a"(low),"=d"(high)  \
                : "a"(word),"g"(a)      \
                : "cc");                \
        __asm__ ("addq %2,%0; adcq %3,%1"       \
                : "+r"(carry),"+d"(high)\
                : "a"(low),"g"(0)       \
                : "cc");                \
        (r)=carry, carry=high;          \
        } while (0)

#define sqr(r0,r1,a)                    \
        __asm__ ("mulq %2"                      \
                : "=a"(r0),"=d"(r1)     \
                : "a"(a)                \
                : "cc");

BN_ULONG bn_mul_add_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w)
        {
        BN_ULONG c1=0;

        if (num <= 0) return(c1);

        while (num&~3)
                {
                mul_add(rp[0],ap[0],w,c1);
                mul_add(rp[1],ap[1],w,c1);
                mul_add(rp[2],ap[2],w,c1);
                mul_add(rp[3],ap[3],w,c1);
                ap+=4; rp+=4; num-=4;
                }
        if (num)
                {
                mul_add(rp[0],ap[0],w,c1); if (--num==0) return c1;
                mul_add(rp[1],ap[1],w,c1); if (--num==0) return c1;
                mul_add(rp[2],ap[2],w,c1); return c1;
```

```
128                 }

130         return(c1);
131         }

133 BN_ULONG bn_mul_words(BN_ULONG *rp, const BN_ULONG *ap, int num, BN_ULONG w)
134         {
135         BN_ULONG c1=0;

137         if (num <= 0) return(c1);

139         while (num&~3)
140                 {
141                 mul(rp[0],ap[0],w,c1);
142                 mul(rp[1],ap[1],w,c1);
143                 mul(rp[2],ap[2],w,c1);
144                 mul(rp[3],ap[3],w,c1);
145                 ap+=4; rp+=4; num-=4;
146                 }
147         if (num)
148                 {
149                 mul(rp[0],ap[0],w,c1); if (--num == 0) return c1;
150                 mul(rp[1],ap[1],w,c1); if (--num == 0) return c1;
151                 mul(rp[2],ap[2],w,c1);
152                 }
153         return(c1);
154         }

156 void bn_sqr_words(BN_ULONG *r, const BN_ULONG *a, int n)
157         {
158         if (n <= 0) return;

160         while (n&~3)
161                 {
162                 sqr(r[0],r[1],a[0]);
163                 sqr(r[2],r[3],a[1]);
164                 sqr(r[4],r[5],a[2]);
165                 sqr(r[6],r[7],a[3]);
166                 a+=4; r+=8; n-=4;
167                 }
168         if (n)
169                 {
170                 sqr(r[0],r[1],a[0]); if (--n == 0) return;
171                 sqr(r[2],r[3],a[1]); if (--n == 0) return;
172                 sqr(r[4],r[5],a[2]);
173                 }
174         }

176 BN_ULONG bn_div_words(BN_ULONG h, BN_ULONG l, BN_ULONG d)
177 {       BN_ULONG ret,waste;

179         __asm__ ("divq  %4"
180                 : "=a"(ret),"=d"(waste)
181                 : "a"(l),"d"(h),"g"(d)
182                 : "cc");

184         return ret;
185 }

187 BN_ULONG bn_add_words (BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp,int
188 { BN_ULONG ret=0,i=0;

190         if (n <= 0) return 0;

192         __asm__ (
193         "       subq    %2,%2           \n"
```



```
194         ".p2align 4                     \n"
195         "1:     movq    (%4,%2,8),%0    \n"
196         "       adcq    (%5,%2,8),%0    \n"
197         "       movq    %0,(%3,%2,8)    \n"
198         "       leaq    1(%2),%2        \n"
199         "       loop    1b              \n"
200         "       sbbq    %0,%0           \n"
201                 : "=&a"(ret),"+c"(n),"=&r"(i)
202                 : "r"(rp),"r"(ap),"r"(bp)
203                 : "cc"
204         );

206   return ret&1;
207 }

209 #ifndef SIMICS
210 BN_ULONG bn_sub_words (BN_ULONG *rp, const BN_ULONG *ap, const BN_ULONG *bp,int
211 { BN_ULONG ret=0,i=0;

213         if (n <= 0) return 0;

215         __asm__ (
216         "       subq    %2,%2           \n"
217         ".p2align 4                     \n"
218         "1:     movq    (%4,%2,8),%0    \n"
219         "       sbbq    (%5,%2,8),%0    \n"
220         "       movq    %0,(%3,%2,8)    \n"
221         "       leaq    1(%2),%2        \n"
222         "       loop    1b              \n"
223         "       sbbq    %0,%0           \n"
224                 : "=&a"(ret),"+c"(n),"=&r"(i)
225                 : "r"(rp),"r"(ap),"r"(bp)
226                 : "cc"
227         );

229   return ret&1;
230 }
231 #else
232 /* Simics 1.4<7 has buggy sbbq:-( */
233 #define BN_MASK2 0xffffffffffffffffL
234 BN_ULONG bn_sub_words(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b, int n)
235         {
236         BN_ULONG t1,t2;
237         int c=0;

239         if (n <= 0) return((BN_ULONG)0);

241         for (;;)
242                 {
243                 t1=a[0]; t2=b[0];
244                 r[0]=(t1-t2-c)&BN_MASK2;
245                 if (t1 != t2) c=(t1 < t2);
246                 if (--n <= 0) break;

248                 t1=a[1]; t2=b[1];
249                 r[1]=(t1-t2-c)&BN_MASK2;
250                 if (t1 != t2) c=(t1 < t2);
251                 if (--n <= 0) break;

253                 t1=a[2]; t2=b[2];
254                 r[2]=(t1-t2-c)&BN_MASK2;
255                 if (t1 != t2) c=(t1 < t2);
256                 if (--n <= 0) break;

258                 t1=a[3]; t2=b[3];
259                 r[3]=(t1-t2-c)&BN_MASK2;
```

```
260                 if (t1 != t2) c=(t1 < t2);
261                 if (--n <= 0) break;

263                 a+=4;
264                 b+=4;
265                 r+=4;
266                 }
267         return(c);
268         }
269 #endif

271 /* mul_add_c(a,b,c0,c1,c2)  -- c+=a*b for three word number c=(c2,c1,c0) */
272 /* mul_add_c2(a,b,c0,c1,c2) -- c+=2*a*b for three word number c=(c2,c1,c0) */
273 /* sqr_add_c(a,i,c0,c1,c2)  -- c+=a[i]^2 for three word number c=(c2,c1,c0) */
274 /* sqr_add_c2(a,i,c0,c1,c2) -- c+=2*a[i]*a[j] for three word number c=(c2,c1,c0)

276 #if 0
277 /* original macros are kept for reference purposes */
278 #define mul_add_c(a,b,c0,c1,c2) {       \
279         BN_ULONG ta=(a),tb=(b);         \
280         t1 = ta * tb;                   \
281         t2 = BN_UMULT_HIGH(ta,tb);      \
282         c0 += t1; t2 += (c0<t1)?1:0;    \
283         c1 += t2; c2 += (c1<t2)?1:0;    \
284         }

286 #define mul_add_c2(a,b,c0,c1,c2) {      \
287         BN_ULONG ta=(a),tb=(b),t0;      \
288         t1 = BN_UMULT_HIGH(ta,tb);      \
289         t0 = ta * tb;                   \
290         t2 = t1+t1; c2 += (t2<t1)?1:0;  \
291         t1 = t0+t0; t2 += (t1<t0)?1:0;  \
292         c0 += t1; t2 += (c0<t1)?1:0;    \
293         c1 += t2; c2 += (c1<t2)?1:0;    \
294         }
295 #else
296 #define mul_add_c(a,b,c0,c1,c2) do {    \
297         __asm__ ("mulq %3"                      \
298                 : "=a"(t1),"=d"(t2)     \
299                 : "a"(a),"m"(b)         \
300                 : "cc");                \
301         __asm__ ("addq %2,%0; adcq %3,%1"       \
302                 : "+r"(c0),"+d"(t2)     \
303                 : "a"(t1),"g"(0)        \
304                 : "cc");                \
305         __asm__ ("addq %2,%0; adcq %3,%1"       \
306                 : "+r"(c1),"+r"(c2)     \
307                 : "d"(t2),"g"(0)        \
308                 : "cc");                \
309         } while (0)

311 #define sqr_add_c(a,i,c0,c1,c2) do {    \
312         __asm__ ("mulq %2"                      \
313                 : "=a"(t1),"=d"(t2)     \
314                 : "a"(a[i])             \
315                 : "cc");                \
316         __asm__ ("addq %2,%0; adcq %3,%1"       \
317                 : "+r"(c0),"+d"(t2)     \
318                 : "a"(t1),"g"(0)        \
319                 : "cc");                \
320         __asm__ ("addq %2,%0; adcq %3,%1"       \
321                 : "+r"(c1),"+r"(c2)     \
322                 : "d"(t2),"g"(0)        \
323                 : "cc");                \
324         } while (0)
```



```
326 #define mul_add_c2(a,b,c0,c1,c2) do {   \
327         __asm__ ("mulq %3"                      \
328                 : "=a"(t1),"=d"(t2)     \
329                 : "a"(a),"m"(b)         \
330                 : "cc");                \
331         __asm__ ("addq %0,%0; adcq %2,%1"       \
332                 : "+d"(t2),"+r"(c2)     \
333                 : "g"(0)                \
334                 : "cc");                \
335         __asm__ ("addq %0,%0; adcq %2,%1"       \
336                 : "+a"(t1),"+d"(t2)     \
337                 : "g"(0)                \
338                 : "cc");                \
339         __asm__ ("addq %2,%0; adcq %3,%1"       \
340                 : "+r"(c0),"+d"(t2)     \
341                 : "a"(t1),"g"(0)        \
342                 : "cc");                \
343         __asm__ ("addq %2,%0; adcq %3,%1"       \
344                 : "+r"(c1),"+r"(c2)     \
345                 : "d"(t2),"g"(0)        \
346                 : "cc");                \
347         } while (0)
348 #endif

350 #define sqr_add_c2(a,i,j,c0,c1,c2)      \
351         mul_add_c2((a)[i],(a)[j],c0,c1,c2)

353 void bn_mul_comba8(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b)
354         {
355         BN_ULONG t1,t2;
356         BN_ULONG c1,c2,c3;

358         c1=0;
359         c2=0;
360         c3=0;
361         mul_add_c(a[0],b[0],c1,c2,c3);
362         r[0]=c1;
363         c1=0;
364         mul_add_c(a[0],b[1],c2,c3,c1);
365         mul_add_c(a[1],b[0],c2,c3,c1);
366         r[1]=c2;
367         c2=0;
368         mul_add_c(a[2],b[0],c3,c1,c2);
369         mul_add_c(a[1],b[1],c3,c1,c2);
370         mul_add_c(a[0],b[2],c3,c1,c2);
371         r[2]=c3;
372         c3=0;
373         mul_add_c(a[0],b[3],c1,c2,c3);
374         mul_add_c(a[1],b[2],c1,c2,c3);
375         mul_add_c(a[2],b[1],c1,c2,c3);
376         mul_add_c(a[3],b[0],c1,c2,c3);
377         r[3]=c1;
378         c1=0;
379         mul_add_c(a[4],b[0],c2,c3,c1);
380         mul_add_c(a[3],b[1],c2,c3,c1);
381         mul_add_c(a[2],b[2],c2,c3,c1);
382         mul_add_c(a[1],b[3],c2,c3,c1);
383         mul_add_c(a[0],b[4],c2,c3,c1);
384         r[4]=c2;
385         c2=0;
386         mul_add_c(a[0],b[5],c3,c1,c2);
387         mul_add_c(a[1],b[4],c3,c1,c2);
388         mul_add_c(a[2],b[3],c3,c1,c2);
389         mul_add_c(a[3],b[2],c3,c1,c2);
390         mul_add_c(a[4],b[1],c3,c1,c2);
391         mul_add_c(a[5],b[0],c3,c1,c2);
```

```
392         r[5]=c3;
393         c3=0;
394         mul_add_c(a[6],b[0],c1,c2,c3);
395         mul_add_c(a[5],b[1],c1,c2,c3);
396         mul_add_c(a[4],b[2],c1,c2,c3);
397         mul_add_c(a[3],b[3],c1,c2,c3);
398         mul_add_c(a[2],b[4],c1,c2,c3);
399         mul_add_c(a[1],b[5],c1,c2,c3);
400         mul_add_c(a[0],b[6],c1,c2,c3);
401         r[6]=c1;
402         c1=0;
403         mul_add_c(a[0],b[7],c2,c3,c1);
404         mul_add_c(a[1],b[6],c2,c3,c1);
405         mul_add_c(a[2],b[5],c2,c3,c1);
406         mul_add_c(a[3],b[4],c2,c3,c1);
407         mul_add_c(a[4],b[3],c2,c3,c1);
408         mul_add_c(a[5],b[2],c2,c3,c1);
409         mul_add_c(a[6],b[1],c2,c3,c1);
410         mul_add_c(a[7],b[0],c2,c3,c1);
411         r[7]=c2;
412         c2=0;
413         mul_add_c(a[7],b[1],c3,c1,c2);
414         mul_add_c(a[6],b[2],c3,c1,c2);
415         mul_add_c(a[5],b[3],c3,c1,c2);
416         mul_add_c(a[4],b[4],c3,c1,c2);
417         mul_add_c(a[3],b[5],c3,c1,c2);
418         mul_add_c(a[2],b[6],c3,c1,c2);
419         mul_add_c(a[1],b[7],c3,c1,c2);
420         r[8]=c3;
421         c3=0;
422         mul_add_c(a[2],b[7],c1,c2,c3);
423         mul_add_c(a[3],b[6],c1,c2,c3);
424         mul_add_c(a[4],b[5],c1,c2,c3);
425         mul_add_c(a[5],b[4],c1,c2,c3);
426         mul_add_c(a[6],b[3],c1,c2,c3);
427         mul_add_c(a[7],b[2],c1,c2,c3);
428         r[9]=c1;
429         c1=0;
430         mul_add_c(a[7],b[3],c2,c3,c1);
431         mul_add_c(a[6],b[4],c2,c3,c1);
432         mul_add_c(a[5],b[5],c2,c3,c1);
433         mul_add_c(a[4],b[6],c2,c3,c1);
434         mul_add_c(a[3],b[7],c2,c3,c1);
435         r[10]=c2;
436         c2=0;
437         mul_add_c(a[4],b[7],c3,c1,c2);
438         mul_add_c(a[5],b[6],c3,c1,c2);
439         mul_add_c(a[6],b[5],c3,c1,c2);
440         mul_add_c(a[7],b[4],c3,c1,c2);
441         r[11]=c3;
442         c3=0;
443         mul_add_c(a[7],b[5],c1,c2,c3);
444         mul_add_c(a[6],b[6],c1,c2,c3);
445         mul_add_c(a[5],b[7],c1,c2,c3);
446         r[12]=c1;
447         c1=0;
448         mul_add_c(a[6],b[7],c2,c3,c1);
449         mul_add_c(a[7],b[6],c2,c3,c1);
450         r[13]=c2;
451         c2=0;
452         mul_add_c(a[7],b[7],c3,c1,c2);
453         r[14]=c3;
454         r[15]=c1;
455         }

457 void bn_mul_comba4(BN_ULONG *r, BN_ULONG *a, BN_ULONG *b)
```



```
458         {
459         BN_ULONG t1,t2;
460         BN_ULONG c1,c2,c3;

462         c1=0;
463         c2=0;
464         c3=0;
465         mul_add_c(a[0],b[0],c1,c2,c3);
466         r[0]=c1;
467         c1=0;
468         mul_add_c(a[0],b[1],c2,c3,c1);
469         mul_add_c(a[1],b[0],c2,c3,c1);
470         r[1]=c2;
471         c2=0;
472         mul_add_c(a[2],b[0],c3,c1,c2);
473         mul_add_c(a[1],b[1],c3,c1,c2);
474         mul_add_c(a[0],b[2],c3,c1,c2);
475         r[2]=c3;
476         c3=0;
477         mul_add_c(a[0],b[3],c1,c2,c3);
478         mul_add_c(a[1],b[2],c1,c2,c3);
479         mul_add_c(a[2],b[1],c1,c2,c3);
480         mul_add_c(a[3],b[0],c1,c2,c3);
481         r[3]=c1;
482         c1=0;
483         mul_add_c(a[3],b[1],c2,c3,c1);
484         mul_add_c(a[2],b[2],c2,c3,c1);
485         mul_add_c(a[1],b[3],c2,c3,c1);
486         r[4]=c2;
487         c2=0;
488         mul_add_c(a[2],b[3],c3,c1,c2);
489         mul_add_c(a[3],b[2],c3,c1,c2);
490         r[5]=c3;
491         c3=0;
492         mul_add_c(a[3],b[3],c1,c2,c3);
493         r[6]=c1;
494         r[7]=c2;
495         }

497 void bn_sqr_comba8(BN_ULONG *r, const BN_ULONG *a)
498         {
499         BN_ULONG t1,t2;
500         BN_ULONG c1,c2,c3;

502         c1=0;
503         c2=0;
504         c3=0;
505         sqr_add_c(a,0,c1,c2,c3);
506         r[0]=c1;
507         c1=0;
508         sqr_add_c2(a,1,0,c2,c3,c1);
509         r[1]=c2;
510         c2=0;
511         sqr_add_c(a,1,c3,c1,c2);
512         sqr_add_c2(a,2,0,c3,c1,c2);
513         r[2]=c3;
514         c3=0;
515         sqr_add_c2(a,3,0,c1,c2,c3);
516         sqr_add_c2(a,2,1,c1,c2,c3);
517         r[3]=c1;
518         c1=0;
519         sqr_add_c(a,2,c2,c3,c1);
520         sqr_add_c2(a,3,1,c2,c3,c1);
521         sqr_add_c2(a,4,0,c2,c3,c1);
522         r[4]=c2;
523         c2=0;
```

```
524         sqr_add_c2(a,5,0,c3,c1,c2);
525         sqr_add_c2(a,4,1,c3,c1,c2);
526         sqr_add_c2(a,3,2,c3,c1,c2);
527         r[5]=c3;
528         c3=0;
529         sqr_add_c(a,3,c1,c2,c3);
530         sqr_add_c2(a,4,2,c1,c2,c3);
531         sqr_add_c2(a,5,1,c1,c2,c3);
532         sqr_add_c2(a,6,0,c1,c2,c3);
533         r[6]=c1;
534         c1=0;
535         sqr_add_c2(a,7,0,c2,c3,c1);
536         sqr_add_c2(a,6,1,c2,c3,c1);
537         sqr_add_c2(a,5,2,c2,c3,c1);
538         sqr_add_c2(a,4,3,c2,c3,c1);
539         r[7]=c2;
540         c2=0;
541         sqr_add_c(a,4,c3,c1,c2);
542         sqr_add_c2(a,5,3,c3,c1,c2);
543         sqr_add_c2(a,6,2,c3,c1,c2);
544         sqr_add_c2(a,7,1,c3,c1,c2);
545         r[8]=c3;
546         c3=0;
547         sqr_add_c2(a,7,2,c1,c2,c3);
548         sqr_add_c2(a,6,3,c1,c2,c3);
549         sqr_add_c2(a,5,4,c1,c2,c3);
550         r[9]=c1;
551         c1=0;
552         sqr_add_c(a,5,c2,c3,c1);
553         sqr_add_c2(a,6,4,c2,c3,c1);
554         sqr_add_c2(a,7,3,c2,c3,c1);
555         r[10]=c2;
556         c2=0;
557         sqr_add_c2(a,7,4,c3,c1,c2);
558         sqr_add_c2(a,6,5,c3,c1,c2);
559         r[11]=c3;
560         c3=0;
561         sqr_add_c(a,6,c1,c2,c3);
562         sqr_add_c2(a,7,5,c1,c2,c3);
563         r[12]=c1;
564         c1=0;
565         sqr_add_c2(a,7,6,c2,c3,c1);
566         r[13]=c2;
567         c2=0;
568         sqr_add_c(a,7,c3,c1,c2);
569         r[14]=c3;
570         r[15]=c1;
571         }

573 void bn_sqr_comba4(BN_ULONG *r, const BN_ULONG *a)
574         {
575         BN_ULONG t1,t2;
576         BN_ULONG c1,c2,c3;

578         c1=0;
579         c2=0;
580         c3=0;
581         sqr_add_c(a,0,c1,c2,c3);
582         r[0]=c1;
583         c1=0;
584         sqr_add_c2(a,1,0,c2,c3,c1);
585         r[1]=c2;
586         c2=0;
587         sqr_add_c(a,1,c3,c1,c2);
588         sqr_add_c2(a,2,0,c3,c1,c2);
589         r[2]=c3;
```



```
590         c3=0;
591         sqr_add_c2(a,3,0,c1,c2,c3);
592         sqr_add_c2(a,2,1,c1,c2,c3);
593         r[3]=c1;
594         c1=0;
595         sqr_add_c(a,2,c2,c3,c1);
596         sqr_add_c2(a,3,1,c2,c3,c1);
597         r[4]=c2;
598         c2=0;
599         sqr_add_c2(a,3,2,c3,c1,c2);
600         r[5]=c3;
601         c3=0;
602         sqr_add_c(a,3,c1,c2,c3);
603         r[6]=c1;
604         r[7]=c2;
605         }
606 #endif
607 #endif /* ! codereview */
```

```
**********************************************************
    3719 Wed Aug 13 19:52:18 2014
new/usr/src/lib/openssl/libsunw_crypto/buffer/buf_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/buffer/buf_err.c */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/buffer.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_BUF,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_BUF,0,reason)

static ERR_STRING_DATA BUF_str_functs[]=
        {
{ERR_FUNC(BUF_F_BUF_MEMDUP),    "BUF_memdup"},
{ERR_FUNC(BUF_F_BUF_MEM_GROW),  "BUF_MEM_grow"},
{ERR_FUNC(BUF_F_BUF_MEM_GROW_CLEAN),    "BUF_MEM_grow_clean"},
{ERR_FUNC(BUF_F_BUF_MEM_NEW),   "BUF_MEM_new"},
{ERR_FUNC(BUF_F_BUF_STRDUP),    "BUF_strdup"},
{ERR_FUNC(BUF_F_BUF_STRNDUP),   "BUF_strndup"},
{0,NULL}
        };

static ERR_STRING_DATA BUF_str_reasons[]=
        {
{0,NULL}
        };

#endif

void ERR_load_BUF_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(BUF_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,BUF_str_functs);
                ERR_load_strings(0,BUF_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    4257 Wed Aug 13 19:52:18 2014
new/usr/src/lib/openssl/libsunw_crypto/buffer/buf_str.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/buffer/buffer.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
char *BUF_strdup(const char *str)
        {
        if (str == NULL) return(NULL);
        return BUF_strndup(str, strlen(str));
        }

char *BUF_strndup(const char *str, size_t siz)
        {
        char *ret;

        if (str == NULL) return(NULL);

        ret=OPENSSL_malloc(siz+1);
        if (ret == NULL)
                {
                BUFerr(BUF_F_BUF_STRNDUP,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }
        BUF_strlcpy(ret,str,siz+1);
        return(ret);
        }

void *BUF_memdup(const void *data, size_t siz)
        {
        void *ret;

        if (data == NULL) return(NULL);

        ret=OPENSSL_malloc(siz);
        if (ret == NULL)
                {
                BUFerr(BUF_F_BUF_MEMDUP,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }
        return memcpy(ret, data, siz);
        }

size_t BUF_strlcpy(char *dst, const char *src, size_t size)
        {
        size_t l = 0;
        for(; size > 1 && *src; size--)
                {
                *dst++ = *src++;
                l++;
                }
        if (size)
                *dst = '\0';
        return l + strlen(src);
        }

size_t BUF_strlcat(char *dst, const char *src, size_t size)
        {
        size_t l = 0;
        for(; size > 0 && *dst; size--, dst++)
                l++;
        return l + BUF_strlcpy(dst, src, size);
        }
#endif /* ! codereview */
```

```
**********************************************************
    5825 Wed Aug 13 19:52:18 2014
new/usr/src/lib/openssl/libsunw_crypto/buffer/buffer.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/buffer/buffer.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
/* LIMIT_BEFORE_EXPANSION is the maximum n such that (n+3)/3*4 < 2**31. That
 * function is applied in several functions in this file and this limit ensures
 * that the result fits in an int. */
#define LIMIT_BEFORE_EXPANSION 0x5ffffffc

BUF_MEM *BUF_MEM_new(void)
        {
        BUF_MEM *ret;

        ret=OPENSSL_malloc(sizeof(BUF_MEM));
        if (ret == NULL)
                {
                BUFerr(BUF_F_BUF_MEM_NEW,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }
        ret->length=0;
        ret->max=0;
        ret->data=NULL;
        return(ret);
        }

void BUF_MEM_free(BUF_MEM *a)
        {
        if(a == NULL)
            return;

        if (a->data != NULL)
                {
                memset(a->data,0,(unsigned int)a->max);
                OPENSSL_free(a->data);
                }
        OPENSSL_free(a);
        }

int BUF_MEM_grow(BUF_MEM *str, size_t len)
        {
        char *ret;
        size_t n;

        if (str->length >= len)
                {
                str->length=len;
                return(len);
                }
        if (str->max >= len)
                {
                memset(&str->data[str->length],0,len-str->length);
                str->length=len;
                return(len);
                }
        /* This limit is sufficient to ensure (len+3)/3*4 < 2**31 */
        if (len > LIMIT_BEFORE_EXPANSION)
                {
                BUFerr(BUF_F_BUF_MEM_GROW,ERR_R_MALLOC_FAILURE);
                return 0;
                }
        n=(len+3)/3*4;
        if (str->data == NULL)
                ret=OPENSSL_malloc(n);
        else
                ret=OPENSSL_realloc(str->data,n);
        if (ret == NULL)
                {
                BUFerr(BUF_F_BUF_MEM_GROW,ERR_R_MALLOC_FAILURE);
                len=0;
```

```
128                 }
129         else
130                 {
131                 str->data=ret;
132                 str->max=n;
133                 memset(&str->data[str->length],0,len-str->length);
134                 str->length=len;
135                 }
136         return(len);
137         }

139 int BUF_MEM_grow_clean(BUF_MEM *str, size_t len)
140         {
141         char *ret;
142         size_t n;

144         if (str->length >= len)
145                 {
146                 memset(&str->data[len],0,str->length-len);
147                 str->length=len;
148                 return(len);
149                 }
150         if (str->max >= len)
151                 {
152                 memset(&str->data[str->length],0,len-str->length);
153                 str->length=len;
154                 return(len);
155                 }
156         /* This limit is sufficient to ensure (len+3)/3*4 < 2**31 */
157         if (len > LIMIT_BEFORE_EXPANSION)
158                 {
159                 BUFerr(BUF_F_BUF_MEM_GROW_CLEAN,ERR_R_MALLOC_FAILURE);
160                 return 0;
161                 }
162         n=(len+3)/3*4;
163         if (str->data == NULL)
164                 ret=OPENSSL_malloc(n);
165         else
166                 ret=OPENSSL_realloc_clean(str->data,str->max,n);
167         if (ret == NULL)
168                 {
169                 BUFerr(BUF_F_BUF_MEM_GROW_CLEAN,ERR_R_MALLOC_FAILURE);
170                 len=0;
171                 }
172         else
173                 {
174                 str->data=ret;
175                 str->max=n;
176                 memset(&str->data[str->length],0,len-str->length);
177                 str->length=len;
178                 }
179         return(len);
180         }

182 void BUF_reverse(unsigned char *out, const unsigned char *in, size_t size)
183         {
184         size_t i;
185         if (in)
186                 {
187                 out += size - 1;
188                 for (i = 0; i < size; i++)
189                         *out-- = *in++;
190                 }
191         else
192                 {
193                 unsigned char *q;
```



```
194                 char c;
195                 q = out + size - 1;
196                 for (i = 0; i < size/2; i++)
197                         {
198                         c = *q;
199                         *q-- = *out;
200                         *out++ = c;
201                         }
202                 }
203         }
204 #endif /* ! codereview */
```

```
**********************************************************
    6794 Wed Aug 13 19:52:19 2014
new/usr/src/lib/openssl/libsunw_crypto/camellia/cmll_cfb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/camellia/camellia_cfb.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
```



```
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/opensslconf.h>
#include <openssl/camellia.h>
#include <openssl/modes.h>


/* The input and output encrypted as though 128bit cfb mode is being
 * used.  The extra state information to record how much of the
 * 128bit block we have used is contained in *num;
 */

void Camellia_cfb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char *ivec, int *num, const int enc)
        {

        CRYPTO_cfb128_encrypt(in,out,length,key,ivec,num,enc,(block128_f)Camelli
        }

/* N.B. This expects the input to be packed, MS bit first */
void Camellia_cfb1_encrypt(const unsigned char *in, unsigned char *out,
```

```
128         size_t length, const CAMELLIA_KEY *key,
129         unsigned char *ivec, int *num, const int enc)
130         {
131         CRYPTO_cfb128_1_encrypt(in,out,length,key,ivec,num,enc,(block128_f)Camel
132         }

134 void Camellia_cfb8_encrypt(const unsigned char *in, unsigned char *out,
135         size_t length, const CAMELLIA_KEY *key,
136         unsigned char *ivec, int *num, const int enc)
137         {
138         CRYPTO_cfb128_8_encrypt(in,out,length,key,ivec,num,enc,(block128_f)Camel
139         }
140 #endif /* ! codereview */
```

```
**********************************************************
    2909 Wed Aug 13 19:52:19 2014
new/usr/src/lib/openssl/libsunw_crypto/camellia/cmll_ctr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/camellia/camellia_ctr.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/camellia.h>
#include <openssl/modes.h>

void Camellia_ctr128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char ivec[CAMELLIA_BLOCK_SIZE],
        unsigned char ecount_buf[CAMELLIA_BLOCK_SIZE],
        unsigned int *num)
        {
```



```
        CRYPTO_ctr128_encrypt(in,out,length,key,ivec,ecount_buf,num,(block128_f)
        }
#endif /* ! codereview */
```

```
**********************************************************
    2987 Wed Aug 13 19:52:19 2014
new/usr/src/lib/openssl/libsunw_crypto/camellia/cmll_ecb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/camellia/camellia_ecb.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#ifndef CAMELLIA_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>

#include <openssl/camellia.h>
#include "cmll_locl.h"
```



```
void Camellia_ecb_encrypt(const unsigned char *in, unsigned char *out,
        const CAMELLIA_KEY *key, const int enc)
        {

        assert(in && out && key);
        assert((CAMELLIA_ENCRYPT == enc)||(CAMELLIA_DECRYPT == enc));

        if (CAMELLIA_ENCRYPT == enc)
                Camellia_encrypt(in, out, key);
        else
                Camellia_decrypt(in, out, key);
        }
#endif /* ! codereview */
```

```
**********************************************************
    3334 Wed Aug 13 19:52:19 2014
new/usr/src/lib/openssl/libsunw_crypto/camellia/cmll_misc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/camellia/camellia_misc.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/opensslv.h>
#include <openssl/crypto.h>
#include <openssl/camellia.h>
#include "cmll_locl.h"

const char CAMELLIA_version[]="CAMELLIA" OPENSSL_VERSION_PTEXT;

int private_Camellia_set_key(const unsigned char *userKey, const int bits,
        CAMELLIA_KEY *key)
        {
```



```
        if(!userKey || !key)
                return -1;
        if(bits != 128 && bits != 192 && bits != 256)
                return -2;
        key->grand_rounds = Camellia_Ekeygen(bits , userKey, key->u.rd_key);
        return 0;
        }

void Camellia_encrypt(const unsigned char *in, unsigned char *out,
        const CAMELLIA_KEY *key)
        {
        Camellia_EncryptBlock_Rounds(key->grand_rounds, in , key->u.rd_key , out
        }

void Camellia_decrypt(const unsigned char *in, unsigned char *out,
        const CAMELLIA_KEY *key)
        {
        Camellia_DecryptBlock_Rounds(key->grand_rounds, in , key->u.rd_key , out
        }
#endif /* ! codereview */
```

```
**********************************************************
    6168 Wed Aug 13 19:52:19 2014
new/usr/src/lib/openssl/libsunw_crypto/camellia/cmll_ofb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/camellia/camellia_ofb.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
```



```
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/camellia.h>
#include <openssl/modes.h>

/* The input and output encrypted as though 128bit ofb mode is being
 * used.  The extra state information to record how much of the
 * 128bit block we have used is contained in *num;
 */
void Camellia_ofb128_encrypt(const unsigned char *in, unsigned char *out,
        size_t length, const CAMELLIA_KEY *key,
        unsigned char *ivec, int *num) {
        CRYPTO_ofb128_encrypt(in,out,length,key,ivec,num,(block128_f)Camellia_en
}
#endif /* ! codereview */
```

```
**********************************************************
    2839 Wed Aug 13 19:52:19 2014
new/usr/src/lib/openssl/libsunw_crypto/camellia/cmll_utl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/camellia/cmll_utl.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/opensslv.h>
#include <openssl/crypto.h>
#include <openssl/camellia.h>
#include "cmll_locl.h"

int Camellia_set_key(const unsigned char *userKey, const int bits,
        CAMELLIA_KEY *key)
        {
#ifdef OPENSSL_FIPS
        fips_cipher_abort(Camellia);
```



```
#endif
        return private_Camellia_set_key(userKey, bits, key);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4372 Wed Aug 13 19:52:20 2014
new/usr/src/lib/openssl/libsunw_crypto/cast/c_cfb64.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/c_cfb64.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/cast.h>
#include "cast_lcl.h"
```



```
/* The input and output encrypted as though 64bit cfb mode is being
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */

void CAST_cfb64_encrypt(const unsigned char *in, unsigned char *out,
                        long length, const CAST_KEY *schedule, unsigned char *iv
                        int *num, int enc)
        {
        register CAST_LONG v0,v1,t;
        register int n= *num;
        register long l=length;
        CAST_LONG ti[2];
        unsigned char *iv,c,cc;

        iv=ivec;
        if (enc)
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                n2l(iv,v0); ti[0]=v0;
                                n2l(iv,v1); ti[1]=v1;
                                CAST_encrypt((CAST_LONG *)ti,schedule);
                                iv=ivec;
                                t=ti[0]; l2n(t,iv);
                                t=ti[1]; l2n(t,iv);
                                iv=ivec;
                                }
                        c= *(in++)^iv[n];
                        *(out++)=c;
                        iv[n]=c;
                        n=(n+1)&0x07;
                        }
                }
        else
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                n2l(iv,v0); ti[0]=v0;
                                n2l(iv,v1); ti[1]=v1;
                                CAST_encrypt((CAST_LONG *)ti,schedule);
                                iv=ivec;
                                t=ti[0]; l2n(t,iv);
                                t=ti[1]; l2n(t,iv);
                                iv=ivec;
                                }
                        cc= *(in++);
                        c=iv[n];
                        iv[n]=cc;
                        *(out++)=c^cc;
                        n=(n+1)&0x07;
                        }
                }
        v0=v1=ti[0]=ti[1]=t=c=cc=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    3614 Wed Aug 13 19:52:20 2014
new/usr/src/lib/openssl/libsunw_crypto/cast/c_ecb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/c_ecb.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/cast.h>
#include "cast_lcl.h"
#include <openssl/opensslv.h>
```



```
const char CAST_version[]="CAST" OPENSSL_VERSION_PTEXT;

void CAST_ecb_encrypt(const unsigned char *in, unsigned char *out,
                      const CAST_KEY *ks, int enc)
        {
        CAST_LONG l,d[2];

        n2l(in,l); d[0]=l;
        n2l(in,l); d[1]=l;
        if (enc)
                CAST_encrypt(d,ks);
        else
                CAST_decrypt(d,ks);
        l=d[0]; l2n(l,out);
        l=d[1]; l2n(l,out);
        l=d[0]=d[1]=0;
        }
#endif /* ! codereview */
```

```
**********************************************************
    5965 Wed Aug 13 19:52:20 2014
new/usr/src/lib/openssl/libsunw_crypto/cast/c_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/c_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/cast.h>
#include <cast_lcl.h>
```



```
void CAST_encrypt(CAST_LONG *data, const CAST_KEY *key)
        {
        register CAST_LONG l,r,t;
        register const CAST_LONG *k;

        k= &(key->data[0]);
        l=data[0];
        r=data[1];

        E_CAST( 0,k,l,r,+,^,-);
        E_CAST( 1,k,r,l,^,-,+);
        E_CAST( 2,k,l,r,-,+,^);
        E_CAST( 3,k,r,l,+,^,-);
        E_CAST( 4,k,l,r,^,-,+);
        E_CAST( 5,k,r,l,-,+,^);
        E_CAST( 6,k,l,r,+,^,-);
        E_CAST( 7,k,r,l,^,-,+);
        E_CAST( 8,k,l,r,-,+,^);
        E_CAST( 9,k,r,l,+,^,-);
        E_CAST(10,k,l,r,^,-,+);
        E_CAST(11,k,r,l,-,+,^);
        if(!key->short_key)
            {
            E_CAST(12,k,l,r,+,^,-);
            E_CAST(13,k,r,l,^,-,+);
            E_CAST(14,k,l,r,-,+,^);
            E_CAST(15,k,r,l,+,^,-);
            }

        data[1]=l&0xffffffffL;
        data[0]=r&0xffffffffL;
        }

void CAST_decrypt(CAST_LONG *data, const CAST_KEY *key)
        {
        register CAST_LONG l,r,t;
        register const CAST_LONG *k;

        k= &(key->data[0]);
        l=data[0];
        r=data[1];

        if(!key->short_key)
            {
            E_CAST(15,k,l,r,+,^,-);
            E_CAST(14,k,r,l,-,+,^);
            E_CAST(13,k,l,r,^,-,+);
            E_CAST(12,k,r,l,+,^,-);
            }
        E_CAST(11,k,l,r,-,+,^);
        E_CAST(10,k,r,l,^,-,+);
        E_CAST( 9,k,l,r,+,^,-);
        E_CAST( 8,k,r,l,-,+,^);
        E_CAST( 7,k,l,r,^,-,+);
        E_CAST( 6,k,r,l,+,^,-);
        E_CAST( 5,k,l,r,-,+,^);
        E_CAST( 4,k,r,l,^,-,+);
        E_CAST( 3,k,l,r,+,^,-);
        E_CAST( 2,k,r,l,-,+,^);
        E_CAST( 1,k,l,r,^,-,+);
        E_CAST( 0,k,r,l,+,^,-);

        data[1]=l&0xffffffffL;
        data[0]=r&0xffffffffL;
        }
```

```
128 void CAST_cbc_encrypt(const unsigned char *in, unsigned char *out, long length,
129              const CAST_KEY *ks, unsigned char *iv, int enc)
130         {
131         register CAST_LONG tin0,tin1;
132         register CAST_LONG tout0,tout1,xor0,xor1;
133         register long l=length;
134         CAST_LONG tin[2];
135
136         if (enc)
137                 {
138                 n2l(iv,tout0);
139                 n2l(iv,tout1);
140                 iv-=8;
141                 for (l-=8; l>=0; l-=8)
142                         {
143                         n2l(in,tin0);
144                         n2l(in,tin1);
145                         tin0^=tout0;
146                         tin1^=tout1;
147                         tin[0]=tin0;
148                         tin[1]=tin1;
149                         CAST_encrypt(tin,ks);
150                         tout0=tin[0];
151                         tout1=tin[1];
152                         l2n(tout0,out);
153                         l2n(tout1,out);
154                         }
155                 if (l != -8)
156                         {
157                         n2ln(in,tin0,tin1,l+8);
158                         tin0^=tout0;
159                         tin1^=tout1;
160                         tin[0]=tin0;
161                         tin[1]=tin1;
162                         CAST_encrypt(tin,ks);
163                         tout0=tin[0];
164                         tout1=tin[1];
165                         l2n(tout0,out);
166                         l2n(tout1,out);
167                         }
168                 l2n(tout0,iv);
169                 l2n(tout1,iv);
170                 }
171         else
172                 {
173                 n2l(iv,xor0);
174                 n2l(iv,xor1);
175                 iv-=8;
176                 for (l-=8; l>=0; l-=8)
177                         {
178                         n2l(in,tin0);
179                         n2l(in,tin1);
180                         tin[0]=tin0;
181                         tin[1]=tin1;
182                         CAST_decrypt(tin,ks);
183                         tout0=tin[0]^xor0;
184                         tout1=tin[1]^xor1;
185                         l2n(tout0,out);
186                         l2n(tout1,out);
187                         xor0=tin0;
188                         xor1=tin1;
189                         }
190                 if (l != -8)
191                         {
192                         n2l(in,tin0);
193                         n2l(in,tin1);
```



```
194                         tin[0]=tin0;
195                         tin[1]=tin1;
196                         CAST_decrypt(tin,ks);
197                         tout0=tin[0]^xor0;
198                         tout1=tin[1]^xor1;
199                         l2nn(tout0,tout1,out,l+8);
200                         xor0=tin0;
201                         xor1=tin1;
202                         }
203                 l2n(xor0,iv);
204                 l2n(xor1,iv);
205                 }
206         tin0=tin1=tout0=tout1=xor0=xor1=0;
207         tin[0]=tin[1]=0;
208         }
209 #endif /* ! codereview */
```

```
**********************************************************
    4166 Wed Aug 13 19:52:20 2014
new/usr/src/lib/openssl/libsunw_crypto/cast/c_ofb64.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/c_ofb64.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/cast.h>
#include "cast_lcl.h"
```



```
/* The input and output encrypted as though 64bit ofb mode is being
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */
void CAST_ofb64_encrypt(const unsigned char *in, unsigned char *out,
                        long length, const CAST_KEY *schedule, unsigned char *iv
                        int *num)
        {
        register CAST_LONG v0,v1,t;
        register int n= *num;
        register long l=length;
        unsigned char d[8];
        register char *dp;
        CAST_LONG ti[2];
        unsigned char *iv;
        int save=0;

        iv=ivec;
        n2l(iv,v0);
        n2l(iv,v1);
        ti[0]=v0;
        ti[1]=v1;
        dp=(char *)d;
        l2n(v0,dp);
        l2n(v1,dp);
        while (l--)
                {
                if (n == 0)
                        {
                        CAST_encrypt((CAST_LONG *)ti,schedule);
                        dp=(char *)d;
                        t=ti[0]; l2n(t,dp);
                        t=ti[1]; l2n(t,dp);
                        save++;
                        }
                *(out++)= *(in++)^d[n];
                n=(n+1)&0x07;
                }
        if (save)
                {
                v0=ti[0];
                v1=ti[1];
                iv=ivec;
                l2n(v0,iv);
                l2n(v1,iv);
                }
        t=v0=v1=ti[0]=ti[1]=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    6668 Wed Aug 13 19:52:20 2014
new/usr/src/lib/openssl/libsunw_crypto/cast/c_skey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cast/c_skey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/crypto.h>
#include <openssl/cast.h>
#include <cast_lcl.h>
```



```
#include <cast_s.h>

#define CAST_exp(l,A,a,n) \
        A[n/4]=l; \
        a[n+3]=(l    )&0xff; \
        a[n+2]=(l>> 8)&0xff; \
        a[n+1]=(l>>16)&0xff; \
        a[n+0]=(l>>24)&0xff;

#define S4 CAST_S_table4
#define S5 CAST_S_table5
#define S6 CAST_S_table6
#define S7 CAST_S_table7
void CAST_set_key(CAST_KEY *key, int len, const unsigned char *data)
#ifdef OPENSSL_FIPS
        {
        fips_cipher_abort(CAST);
        private_CAST_set_key(key, len, data);
        }
void private_CAST_set_key(CAST_KEY *key, int len, const unsigned char *data)
#endif
        {
        CAST_LONG x[16];
        CAST_LONG z[16];
        CAST_LONG k[32];
        CAST_LONG X[4],Z[4];
        CAST_LONG l,*K;
        int i;

        for (i=0; i<16; i++) x[i]=0;
        if (len > 16) len=16;
        for (i=0; i<len; i++)
                x[i]=data[i];
        if(len <= 10)
            key->short_key=1;
        else
            key->short_key=0;

        K= &k[0];
        X[0]=((x[ 0]<<24)|(x[ 1]<<16)|(x[ 2]<<8)|x[ 3])&0xffffffffL;
        X[1]=((x[ 4]<<24)|(x[ 5]<<16)|(x[ 6]<<8)|x[ 7])&0xffffffffL;
        X[2]=((x[ 8]<<24)|(x[ 9]<<16)|(x[10]<<8)|x[11])&0xffffffffL;
        X[3]=((x[12]<<24)|(x[13]<<16)|(x[14]<<8)|x[15])&0xffffffffL;

        for (;;)
                {
        l=X[0]^S4[x[13]]^S5[x[15]]^S6[x[12]]^S7[x[14]]^S6[x[ 8]];
        CAST_exp(l,Z,z, 0);
        l=X[2]^S4[z[ 0]]^S5[z[ 2]]^S6[z[ 1]]^S7[z[ 3]]^S7[x[10]];
        CAST_exp(l,Z,z, 4);
        l=X[3]^S4[z[ 7]]^S5[z[ 6]]^S6[z[ 5]]^S7[z[ 4]]^S4[x[ 9]];
        CAST_exp(l,Z,z, 8);
        l=X[1]^S4[z[10]]^S5[z[ 9]]^S6[z[11]]^S7[z[ 8]]^S5[x[11]];
        CAST_exp(l,Z,z,12);

        K[ 0]= S4[z[ 8]]^S5[z[ 9]]^S6[z[ 7]]^S7[z[ 6]]^S4[z[ 2]];
        K[ 1]= S4[z[10]]^S5[z[11]]^S6[z[ 5]]^S7[z[ 4]]^S5[z[ 6]];
        K[ 2]= S4[z[12]]^S5[z[13]]^S6[z[ 3]]^S7[z[ 2]]^S6[z[ 9]];
        K[ 3]= S4[z[14]]^S5[z[15]]^S6[z[ 1]]^S7[z[ 0]]^S7[z[12]];

        l=Z[2]^S4[z[ 5]]^S5[z[ 7]]^S6[z[ 4]]^S7[z[ 6]]^S6[z[ 0]];
        CAST_exp(l,X,x, 0);
        l=Z[0]^S4[x[ 0]]^S5[x[ 2]]^S6[x[ 1]]^S7[x[ 3]]^S7[z[ 2]];
        CAST_exp(l,X,x, 4);
        l=Z[1]^S4[x[ 7]]^S5[x[ 6]]^S6[x[ 5]]^S7[x[ 4]]^S4[z[ 1]];
        CAST_exp(l,X,x, 8);
```

```
128         l=Z[3]^S4[x[10]]^S5[x[ 9]]^S6[x[11]]^S7[x[ 8]]^S5[z[ 3]];
129         CAST_exp(l,X,x,12);

131         K[ 4]= S4[x[ 3]]^S5[x[ 2]]^S6[x[12]]^S7[x[13]]^S4[x[ 8]];
132         K[ 5]= S4[x[ 1]]^S5[x[ 0]]^S6[x[14]]^S7[x[15]]^S5[x[13]];
133         K[ 6]= S4[x[ 7]]^S5[x[ 6]]^S6[x[ 8]]^S7[x[ 9]]^S6[x[ 3]];
134         K[ 7]= S4[x[ 5]]^S5[x[ 4]]^S6[x[10]]^S7[x[11]]^S7[x[ 7]];

136         l=X[0]^S4[x[13]]^S5[x[15]]^S6[x[12]]^S7[x[14]]^S6[x[ 8]];
137         CAST_exp(l,Z,z, 0);
138         l=X[2]^S4[z[ 0]]^S5[z[ 2]]^S6[z[ 1]]^S7[z[ 3]]^S7[x[10]];
139         CAST_exp(l,Z,z, 4);
140         l=X[3]^S4[z[ 7]]^S5[z[ 6]]^S6[z[ 5]]^S7[z[ 4]]^S4[x[ 9]];
141         CAST_exp(l,Z,z, 8);
142         l=X[1]^S4[z[10]]^S5[z[ 9]]^S6[z[11]]^S7[z[ 8]]^S5[x[11]];
143         CAST_exp(l,Z,z,12);

145         K[ 8]= S4[z[ 3]]^S5[z[ 2]]^S6[z[12]]^S7[z[13]]^S4[z[ 9]];
146         K[ 9]= S4[z[ 1]]^S5[z[ 0]]^S6[z[14]]^S7[z[15]]^S5[z[12]];
147         K[10]= S4[z[ 7]]^S5[z[ 6]]^S6[z[ 8]]^S7[z[ 9]]^S6[z[ 2]];
148         K[11]= S4[z[ 5]]^S5[z[ 4]]^S6[z[10]]^S7[z[11]]^S7[z[ 6]];

150         l=Z[2]^S4[z[ 5]]^S5[z[ 7]]^S6[z[ 4]]^S7[z[ 6]]^S6[z[ 0]];
151         CAST_exp(l,X,x, 0);
152         l=Z[0]^S4[x[ 0]]^S5[x[ 2]]^S6[x[ 1]]^S7[x[ 3]]^S7[z[ 2]];
153         CAST_exp(l,X,x, 4);
154         l=Z[1]^S4[x[ 7]]^S5[x[ 6]]^S6[x[ 5]]^S7[x[ 4]]^S4[z[ 1]];
155         CAST_exp(l,X,x, 8);
156         l=Z[3]^S4[x[10]]^S5[x[ 9]]^S6[x[11]]^S7[x[ 8]]^S5[z[ 3]];
157         CAST_exp(l,X,x,12);

159         K[12]= S4[x[ 8]]^S5[x[ 9]]^S6[x[ 7]]^S7[x[ 6]]^S4[x[ 3]];
160         K[13]= S4[x[10]]^S5[x[11]]^S6[x[ 5]]^S7[x[ 4]]^S5[x[ 7]];
161         K[14]= S4[x[12]]^S5[x[13]]^S6[x[ 3]]^S7[x[ 2]]^S6[x[ 8]];
162         K[15]= S4[x[14]]^S5[x[15]]^S6[x[ 1]]^S7[x[ 0]]^S7[x[13]];
163         if (K != k)  break;
164         K+=16;
165                 }

167         for (i=0; i<16; i++)
168                 {
169                 key->data[i*2]=k[i];
170                 key->data[i*2+1]=((k[i+16])+16)&0x1f;
171                 }
172         }
173 #endif /* ! codereview */
```

```
**********************************************************
    3189 Wed Aug 13 19:52:20 2014
new/usr/src/lib/openssl/libsunw_crypto/cmac/cm_ameth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2010.
 */
/* ====================================================================
 * Copyright (c) 2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
#include <openssl/cmac.h>
#include "asn1_locl.h"

/* CMAC "ASN1" method. This is just here to indicate the
 * maximum CMAC output length and to free up a CMAC
 * key.
 */
```



```
 */

static int cmac_size(const EVP_PKEY *pkey)
        {
        return EVP_MAX_BLOCK_LENGTH;
        }

static void cmac_key_free(EVP_PKEY *pkey)
        {
        CMAC_CTX *cmctx = (CMAC_CTX *)pkey->pkey.ptr;
        if (cmctx)
                CMAC_CTX_free(cmctx);
        }

const EVP_PKEY_ASN1_METHOD cmac_asn1_meth =
        {
        EVP_PKEY_CMAC,
        EVP_PKEY_CMAC,
        0,

        "CMAC",
        "OpenSSL CMAC method",

        0,0,0,0,

        0,0,0,

        cmac_size,
        0,
        0,0,0,0,0,0,0,

        cmac_key_free,
        0,
        0,0
        };
#endif /* ! codereview */
```

```
**********************************************************
    5561 Wed Aug 13 19:52:21 2014
new/usr/src/lib/openssl/libsunw_crypto/cmac/cm_pmeth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2010.
 */
/* ====================================================================
 * Copyright (c) 2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/evp.h>
#include <openssl/cmac.h>
#include "evp_locl.h"

/* The context structure and "key" is simply a CMAC_CTX */
```



```
static int pkey_cmac_init(EVP_PKEY_CTX *ctx)
        {
        ctx->data = CMAC_CTX_new();
        if (!ctx->data)
                return 0;
        ctx->keygen_info_count = 0;
        return 1;
        }

static int pkey_cmac_copy(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src)
        {
        if (!pkey_cmac_init(dst))
                return 0;
        if (!CMAC_CTX_copy(dst->data, src->data))
                return 0;
        return 1;
        }

static void pkey_cmac_cleanup(EVP_PKEY_CTX *ctx)
        {
        CMAC_CTX_free(ctx->data);
        }

static int pkey_cmac_keygen(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey)
        {
        CMAC_CTX *cmkey = CMAC_CTX_new();
        CMAC_CTX *cmctx = ctx->data;
        if (!cmkey)
                return 0;
        if (!CMAC_CTX_copy(cmkey, cmctx))
                {
                CMAC_CTX_free(cmkey);
                return 0;
                }
        EVP_PKEY_assign(pkey, EVP_PKEY_CMAC, cmkey);

        return 1;
        }

static int int_update(EVP_MD_CTX *ctx,const void *data,size_t count)
        {
        if (!CMAC_Update(ctx->pctx->data, data, count))
                return 0;
        return 1;
        }

static int cmac_signctx_init(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx)
        {
        EVP_MD_CTX_set_flags(mctx, EVP_MD_CTX_FLAG_NO_INIT);
        mctx->update = int_update;
        return 1;
        }

static int cmac_signctx(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
                                            EVP_MD_CTX *mctx)
        {
        return CMAC_Final(ctx->data, sig, siglen);
        }

static int pkey_cmac_ctrl(EVP_PKEY_CTX *ctx, int type, int p1, void *p2)
        {
        CMAC_CTX *cmctx = ctx->data;
        switch (type)
                {
```

```
128                 case EVP_PKEY_CTRL_SET_MAC_KEY:
129                 if (!p2 || p1 < 0)
130                         return 0;
131                 if (!CMAC_Init(cmctx, p2, p1, NULL, NULL))
132                         return 0;
133                 break;

135                 case EVP_PKEY_CTRL_CIPHER:
136                 if (!CMAC_Init(cmctx, NULL, 0, p2, ctx->engine))
137                         return 0;
138                 break;

140                 case EVP_PKEY_CTRL_MD:
141                 if (ctx->pkey && !CMAC_CTX_copy(ctx->data,
142                                         (CMAC_CTX *)ctx->pkey->pkey.ptr))
143                         return 0;
144                 if (!CMAC_Init(cmctx, NULL, 0, NULL, NULL))
145                         return 0;
146                 break;

148                 default:
149                 return -2;

151                 }
152         return 1;
153         }

155 static int pkey_cmac_ctrl_str(EVP_PKEY_CTX *ctx,
156                         const char *type, const char *value)
157         {
158         if (!value)
159                 {
160                 return 0;
161                 }
162         if (!strcmp(type, "key"))
163                 {
164                 void *p = (void *)value;
165                 return pkey_cmac_ctrl(ctx, EVP_PKEY_CTRL_SET_MAC_KEY,
166                                                                 strlen(p), p);
167                 }
168         if (!strcmp(type, "cipher"))
169                 {
170                 const EVP_CIPHER *c;
171                 c = EVP_get_cipherbyname(value);
172                 if (!c)
173                         return 0;
174                 return pkey_cmac_ctrl(ctx, EVP_PKEY_CTRL_CIPHER, -1, (void *)c);
175                 }
176         if (!strcmp(type, "hexkey"))
177                 {
178                 unsigned char *key;
179                 int r;
180                 long keylen;
181                 key = string_to_hex(value, &keylen);
182                 if (!key)
183                         return 0;
184                 r = pkey_cmac_ctrl(ctx, EVP_PKEY_CTRL_SET_MAC_KEY, keylen, key);
185                 OPENSSL_free(key);
186                 return r;
187                 }
188         return -2;
189         }

191 const EVP_PKEY_METHOD cmac_pkey_meth =
192         {
193         EVP_PKEY_CMAC,
```



```
194         EVP_PKEY_FLAG_SIGCTX_CUSTOM,
195         pkey_cmac_init,
196         pkey_cmac_copy,
197         pkey_cmac_cleanup,

199         0, 0,

201         0,
202         pkey_cmac_keygen,

204         0, 0,

206         0, 0,

208         0,0,

210         cmac_signctx_init,
211         cmac_signctx,

213         0,0,

215         0,0,

217         0,0,

219         0,0,

221         pkey_cmac_ctrl,
222         pkey_cmac_ctrl_str

224         };
225 #endif /* ! codereview */
```

```
**********************************************************
    8806 Wed Aug 13 19:52:21 2014
new/usr/src/lib/openssl/libsunw_crypto/cmac/cmac.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cmac/cmac.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cryptlib.h"
#include <openssl/cmac.h>

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
```



```
#endif

struct CMAC_CTX_st
        {
        /* Cipher context to use */
        EVP_CIPHER_CTX cctx;
        /* Keys k1 and k2 */
        unsigned char k1[EVP_MAX_BLOCK_LENGTH];
        unsigned char k2[EVP_MAX_BLOCK_LENGTH];
        /* Temporary block */
        unsigned char tbl[EVP_MAX_BLOCK_LENGTH];
        /* Last (possibly partial) block */
        unsigned char last_block[EVP_MAX_BLOCK_LENGTH];
        /* Number of bytes in last block: -1 means context not initialised */
        int nlast_block;
        };


/* Make temporary keys K1 and K2 */

static void make_kn(unsigned char *k1, unsigned char *l, int bl)
        {
        int i;
        /* Shift block to left, including carry */
        for (i = 0; i < bl; i++)
                {
                k1[i] = l[i] << 1;
                if (i < bl - 1 && l[i + 1] & 0x80)
                        k1[i] |= 1;
                }
        /* If MSB set fixup with R */
        if (l[0] & 0x80)
                k1[bl - 1] ^= bl == 16 ? 0x87 : 0x1b;
        }

CMAC_CTX *CMAC_CTX_new(void)
        {
        CMAC_CTX *ctx;
        ctx = OPENSSL_malloc(sizeof(CMAC_CTX));
        if (!ctx)
                return NULL;
        EVP_CIPHER_CTX_init(&ctx->cctx);
        ctx->nlast_block = -1;
        return ctx;
        }

void CMAC_CTX_cleanup(CMAC_CTX *ctx)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !ctx->cctx.engine)
                {
                FIPS_cmac_ctx_cleanup(ctx);
                return;
                }
#endif
        EVP_CIPHER_CTX_cleanup(&ctx->cctx);
        OPENSSL_cleanse(ctx->tbl, EVP_MAX_BLOCK_LENGTH);
        OPENSSL_cleanse(ctx->k1, EVP_MAX_BLOCK_LENGTH);
        OPENSSL_cleanse(ctx->k2, EVP_MAX_BLOCK_LENGTH);
        OPENSSL_cleanse(ctx->last_block, EVP_MAX_BLOCK_LENGTH);
        ctx->nlast_block = -1;
        }

EVP_CIPHER_CTX *CMAC_CTX_get0_cipher_ctx(CMAC_CTX *ctx)
        {
        return &ctx->cctx;
```

```
128         }

130 void CMAC_CTX_free(CMAC_CTX *ctx)
131         {
132         CMAC_CTX_cleanup(ctx);
133         OPENSSL_free(ctx);
134         }

136 int CMAC_CTX_copy(CMAC_CTX *out, const CMAC_CTX *in)
137         {
138         int bl;
139         if (in->nlast_block == -1)
140                 return 0;
141         if (!EVP_CIPHER_CTX_copy(&out->cctx, &in->cctx))
142                 return 0;
143         bl = EVP_CIPHER_CTX_block_size(&in->cctx);
144         memcpy(out->k1, in->k1, bl);
145         memcpy(out->k2, in->k2, bl);
146         memcpy(out->tbl, in->tbl, bl);
147         memcpy(out->last_block, in->last_block, bl);
148         out->nlast_block = in->nlast_block;
149         return 1;
150         }

152 int CMAC_Init(CMAC_CTX *ctx, const void *key, size_t keylen,
153                       const EVP_CIPHER *cipher, ENGINE *impl)
154         {
155         static unsigned char zero_iv[EVP_MAX_BLOCK_LENGTH];
156 #ifdef OPENSSL_FIPS
157         if (FIPS_mode())
158                 {
159                 /* If we have an ENGINE need to allow non FIPS */
160                 if ((impl || ctx->cctx.engine)
161                         && !(ctx->cctx.flags & EVP_CIPH_FLAG_NON_FIPS_ALLOW))

163                         {
164                         EVPerr(EVP_F_CMAC_INIT, EVP_R_DISABLED_FOR_FIPS);
165                         return 0;
166                         }
167                 /* Other algorithm blocking will be done in FIPS_cmac_init,
168                  * via FIPS_cipherinit().
169                  */
170                 if (!impl && !ctx->cctx.engine)
171                         return FIPS_cmac_init(ctx, key, keylen, cipher, NULL);
172                 }
173 #endif
174         /* All zeros means restart */
175         if (!key && !cipher && !impl && keylen == 0)
176                 {
177                 /* Not initialised */
178                 if (ctx->nlast_block == -1)
179                         return 0;
180                 if (!EVP_EncryptInit_ex(&ctx->cctx, NULL, NULL, NULL, zero_iv))
181                         return 0;
182                 memset(ctx->tbl, 0, EVP_CIPHER_CTX_block_size(&ctx->cctx));
183                 ctx->nlast_block = 0;
184                 return 1;
185                 }
186         /* Initialiase context */
187         if (cipher && !EVP_EncryptInit_ex(&ctx->cctx, cipher, impl, NULL, NULL))
188                 return 0;
189         /* Non-NULL key means initialisation complete */
190         if (key)
191                 {
192                 int bl;
193                 if (!EVP_CIPHER_CTX_cipher(&ctx->cctx))
```



```
194                         return 0;
195                 if (!EVP_CIPHER_CTX_set_key_length(&ctx->cctx, keylen))
196                         return 0;
197                 if (!EVP_EncryptInit_ex(&ctx->cctx, NULL, NULL, key, zero_iv))
198                         return 0;
199                 bl = EVP_CIPHER_CTX_block_size(&ctx->cctx);
200                 if (!EVP_Cipher(&ctx->cctx, ctx->tbl, zero_iv, bl))
201                         return 0;
202                 make_kn(ctx->k1, ctx->tbl, bl);
203                 make_kn(ctx->k2, ctx->k1, bl);
204                 OPENSSL_cleanse(ctx->tbl, bl);
205                 /* Reset context again ready for first data block */
206                 if (!EVP_EncryptInit_ex(&ctx->cctx, NULL, NULL, NULL, zero_iv))
207                         return 0;
208                 /* Zero tbl so resume works */
209                 memset(ctx->tbl, 0, bl);
210                 ctx->nlast_block = 0;
211                 }
212         return 1;
213         }

215 int CMAC_Update(CMAC_CTX *ctx, const void *in, size_t dlen)
216         {
217         const unsigned char *data = in;
218         size_t bl;
219 #ifdef OPENSSL_FIPS
220         if (FIPS_mode() && !ctx->cctx.engine)
221                 return FIPS_cmac_update(ctx, in, dlen);
222 #endif
223         if (ctx->nlast_block == -1)
224                 return 0;
225         if (dlen == 0)
226                 return 1;
227         bl = EVP_CIPHER_CTX_block_size(&ctx->cctx);
228         /* Copy into partial block if we need to */
229         if (ctx->nlast_block > 0)
230                 {
231                 size_t nleft;
232                 nleft = bl - ctx->nlast_block;
233                 if (dlen < nleft)
234                         nleft = dlen;
235                 memcpy(ctx->last_block + ctx->nlast_block, data, nleft);
236                 dlen -= nleft;
237                 ctx->nlast_block += nleft;
238                 /* If no more to process return */
239                 if (dlen == 0)
240                         return 1;
241                 data += nleft;
242                 /* Else not final block so encrypt it */
243                 if (!EVP_Cipher(&ctx->cctx, ctx->tbl, ctx->last_block,bl))
244                         return 0;
245                 }
246         /* Encrypt all but one of the complete blocks left */
247         while(dlen > bl)
248                 {
249                 if (!EVP_Cipher(&ctx->cctx, ctx->tbl, data, bl))
250                         return 0;
251                 dlen -= bl;
252                 data += bl;
253                 }
254         /* Copy any data left to last block buffer */
255         memcpy(ctx->last_block, data, dlen);
256         ctx->nlast_block = dlen;
257         return 1;

259         }
```

```
261 int CMAC_Final(CMAC_CTX *ctx, unsigned char *out, size_t *poutlen)
262         {
263         int i, bl, lb;
264 #ifdef OPENSSL_FIPS
265         if (FIPS_mode() && !ctx->cctx.engine)
266                 return FIPS_cmac_final(ctx, out, poutlen);
267 #endif
268         if (ctx->nlast_block == -1)
269                 return 0;
270         bl = EVP_CIPHER_CTX_block_size(&ctx->cctx);
271         *poutlen = (size_t)bl;
272         if (!out)
273                 return 1;
274         lb = ctx->nlast_block;
275         /* Is last block complete? */
276         if (lb == bl)
277                 {
278                 for (i = 0; i < bl; i++)
279                         out[i] = ctx->last_block[i] ^ ctx->k1[i];
280                 }
281         else
282                 {
283                 ctx->last_block[lb] = 0x80;
284                 if (bl - lb > 1)
285                         memset(ctx->last_block + lb + 1, 0, bl - lb - 1);
286                 for (i = 0; i < bl; i++)
287                         out[i] = ctx->last_block[i] ^ ctx->k2[i];
288                 }
289         if (!EVP_Cipher(&ctx->cctx, out, out, bl))
290                 {
291                 OPENSSL_cleanse(out, bl);
292                 return 0;
293                 }
294         return 1;
295         }

297 int CMAC_resume(CMAC_CTX *ctx)
298         {
299         if (ctx->nlast_block == -1)
300                 return 0;
301         /* The buffer "tbl" containes the last fully encrypted block
302          * which is the last IV (or all zeroes if no last encrypted block).
303          * The last block has not been modified since CMAC_final().
304          * So reinitliasing using the last decrypted block will allow
305          * CMAC to continue after calling CMAC_Final().
306          */
307         return EVP_EncryptInit_ex(&ctx->cctx, NULL, NULL, NULL, ctx->tbl);
308         }
309 #endif /* ! codereview */
```

```
**********************************************************
   16753 Wed Aug 13 19:52:21 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_asn1.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/cms.h>
#include <cms_lcl.h>


ASN1_SEQUENCE(CMS_IssuerAndSerialNumber) = {
```



```
        ASN1_SIMPLE(CMS_IssuerAndSerialNumber, issuer, X509_NAME),
        ASN1_SIMPLE(CMS_IssuerAndSerialNumber, serialNumber, ASN1_INTEGER)
} ASN1_SEQUENCE_END(CMS_IssuerAndSerialNumber)

ASN1_SEQUENCE(CMS_OtherCertificateFormat) = {
        ASN1_SIMPLE(CMS_OtherCertificateFormat, otherCertFormat, ASN1_OBJECT),
        ASN1_OPT(CMS_OtherCertificateFormat, otherCert, ASN1_ANY)
} ASN1_SEQUENCE_END(CMS_OtherCertificateFormat)

ASN1_CHOICE(CMS_CertificateChoices) = {
        ASN1_SIMPLE(CMS_CertificateChoices, d.certificate, X509),
        ASN1_IMP(CMS_CertificateChoices, d.extendedCertificate, ASN1_SEQUENCE, 0
        ASN1_IMP(CMS_CertificateChoices, d.v1AttrCert, ASN1_SEQUENCE, 1),
        ASN1_IMP(CMS_CertificateChoices, d.v2AttrCert, ASN1_SEQUENCE, 2),
        ASN1_IMP(CMS_CertificateChoices, d.other, CMS_OtherCertificateFormat, 3)
} ASN1_CHOICE_END(CMS_CertificateChoices)

ASN1_CHOICE(CMS_SignerIdentifier) = {
        ASN1_SIMPLE(CMS_SignerIdentifier, d.issuerAndSerialNumber, CMS_IssuerAnd
        ASN1_IMP(CMS_SignerIdentifier, d.subjectKeyIdentifier, ASN1_OCTET_STRING
} ASN1_CHOICE_END(CMS_SignerIdentifier)

ASN1_NDEF_SEQUENCE(CMS_EncapsulatedContentInfo) = {
        ASN1_SIMPLE(CMS_EncapsulatedContentInfo, eContentType, ASN1_OBJECT),
        ASN1_NDEF_EXP_OPT(CMS_EncapsulatedContentInfo, eContent, ASN1_OCTET_STRI
} ASN1_NDEF_SEQUENCE_END(CMS_EncapsulatedContentInfo)

/* Minor tweak to operation: free up signer key, cert */
static int cms_si_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                        void *exarg)
        {
        if(operation == ASN1_OP_FREE_POST)
                {
                CMS_SignerInfo *si = (CMS_SignerInfo *)*pval;
                if (si->pkey)
                        EVP_PKEY_free(si->pkey);
                if (si->signer)
                        X509_free(si->signer);
                }
        return 1;
        }

ASN1_SEQUENCE_cb(CMS_SignerInfo, cms_si_cb) = {
        ASN1_SIMPLE(CMS_SignerInfo, version, LONG),
        ASN1_SIMPLE(CMS_SignerInfo, sid, CMS_SignerIdentifier),
        ASN1_SIMPLE(CMS_SignerInfo, digestAlgorithm, X509_ALGOR),
        ASN1_IMP_SET_OF_OPT(CMS_SignerInfo, signedAttrs, X509_ATTRIBUTE, 0),
        ASN1_SIMPLE(CMS_SignerInfo, signatureAlgorithm, X509_ALGOR),
        ASN1_SIMPLE(CMS_SignerInfo, signature, ASN1_OCTET_STRING),
        ASN1_IMP_SET_OF_OPT(CMS_SignerInfo, unsignedAttrs, X509_ATTRIBUTE, 1)
} ASN1_SEQUENCE_END_cb(CMS_SignerInfo, CMS_SignerInfo)

ASN1_SEQUENCE(CMS_OtherRevocationInfoFormat) = {
        ASN1_SIMPLE(CMS_OtherRevocationInfoFormat, otherRevInfoFormat, ASN1_OBJE
        ASN1_OPT(CMS_OtherRevocationInfoFormat, otherRevInfo, ASN1_ANY)
} ASN1_SEQUENCE_END(CMS_OtherRevocationInfoFormat)

ASN1_CHOICE(CMS_RevocationInfoChoice) = {
        ASN1_SIMPLE(CMS_RevocationInfoChoice, d.crl, X509_CRL),
        ASN1_IMP(CMS_RevocationInfoChoice, d.other, CMS_OtherRevocationInfoForma
} ASN1_CHOICE_END(CMS_RevocationInfoChoice)

ASN1_NDEF_SEQUENCE(CMS_SignedData) = {
        ASN1_SIMPLE(CMS_SignedData, version, LONG),
        ASN1_SET_OF(CMS_SignedData, digestAlgorithms, X509_ALGOR),
        ASN1_SIMPLE(CMS_SignedData, encapContentInfo, CMS_EncapsulatedContentInf
```

```
128         ASN1_IMP_SET_OF_OPT(CMS_SignedData, certificates, CMS_CertificateChoices
129         ASN1_IMP_SET_OF_OPT(CMS_SignedData, crls, CMS_RevocationInfoChoice, 1),
130         ASN1_SET_OF(CMS_SignedData, signerInfos, CMS_SignerInfo)
131 } ASN1_NDEF_SEQUENCE_END(CMS_SignedData)

133 ASN1_SEQUENCE(CMS_OriginatorInfo) = {
134         ASN1_IMP_SET_OF_OPT(CMS_OriginatorInfo, certificates, CMS_CertificateCho
135         ASN1_IMP_SET_OF_OPT(CMS_OriginatorInfo, crls, CMS_RevocationInfoChoice,
136 } ASN1_SEQUENCE_END(CMS_OriginatorInfo)

138 ASN1_NDEF_SEQUENCE(CMS_EncryptedContentInfo) = {
139         ASN1_SIMPLE(CMS_EncryptedContentInfo, contentType, ASN1_OBJECT),
140         ASN1_SIMPLE(CMS_EncryptedContentInfo, contentEncryptionAlgorithm, X509_A
141         ASN1_IMP_OPT(CMS_EncryptedContentInfo, encryptedContent, ASN1_OCTET_STRI
142 } ASN1_NDEF_SEQUENCE_END(CMS_EncryptedContentInfo)

144 ASN1_SEQUENCE(CMS_KeyTransRecipientInfo) = {
145         ASN1_SIMPLE(CMS_KeyTransRecipientInfo, version, LONG),
146         ASN1_SIMPLE(CMS_KeyTransRecipientInfo, rid, CMS_SignerIdentifier),
147         ASN1_SIMPLE(CMS_KeyTransRecipientInfo, keyEncryptionAlgorithm, X509_ALGO
148         ASN1_SIMPLE(CMS_KeyTransRecipientInfo, encryptedKey, ASN1_OCTET_STRING)
149 } ASN1_SEQUENCE_END(CMS_KeyTransRecipientInfo)

151 ASN1_SEQUENCE(CMS_OtherKeyAttribute) = {
152         ASN1_SIMPLE(CMS_OtherKeyAttribute, keyAttrId, ASN1_OBJECT),
153         ASN1_OPT(CMS_OtherKeyAttribute, keyAttr, ASN1_ANY)
154 } ASN1_SEQUENCE_END(CMS_OtherKeyAttribute)

156 ASN1_SEQUENCE(CMS_RecipientKeyIdentifier) = {
157         ASN1_SIMPLE(CMS_RecipientKeyIdentifier, subjectKeyIdentifier, ASN1_OCTET
158         ASN1_OPT(CMS_RecipientKeyIdentifier, date, ASN1_GENERALIZEDTIME),
159         ASN1_OPT(CMS_RecipientKeyIdentifier, other, CMS_OtherKeyAttribute)
160 } ASN1_SEQUENCE_END(CMS_RecipientKeyIdentifier)

162 ASN1_CHOICE(CMS_KeyAgreeRecipientIdentifier) = {
163   ASN1_SIMPLE(CMS_KeyAgreeRecipientIdentifier, d.issuerAndSerialNumber, CMS_Issu
164   ASN1_IMP(CMS_KeyAgreeRecipientIdentifier, d.rKeyId, CMS_RecipientKeyIdentifier
165 } ASN1_CHOICE_END(CMS_KeyAgreeRecipientIdentifier)

167 ASN1_SEQUENCE(CMS_RecipientEncryptedKey) = {
168         ASN1_SIMPLE(CMS_RecipientEncryptedKey, rid, CMS_KeyAgreeRecipientIdentif
169         ASN1_SIMPLE(CMS_RecipientEncryptedKey, encryptedKey, ASN1_OCTET_STRING)
170 } ASN1_SEQUENCE_END(CMS_RecipientEncryptedKey)

172 ASN1_SEQUENCE(CMS_OriginatorPublicKey) = {
173   ASN1_SIMPLE(CMS_OriginatorPublicKey, algorithm, X509_ALGOR),
174   ASN1_SIMPLE(CMS_OriginatorPublicKey, publicKey, ASN1_BIT_STRING)
175 } ASN1_SEQUENCE_END(CMS_OriginatorPublicKey)

177 ASN1_CHOICE(CMS_OriginatorIdentifierOrKey) = {
178   ASN1_SIMPLE(CMS_OriginatorIdentifierOrKey, d.issuerAndSerialNumber, CMS_Issuer
179   ASN1_IMP(CMS_OriginatorIdentifierOrKey, d.subjectKeyIdentifier, ASN1_OCTET_STR
180   ASN1_IMP(CMS_OriginatorIdentifierOrKey, d.originatorKey, CMS_OriginatorPublicK
181 } ASN1_CHOICE_END(CMS_OriginatorIdentifierOrKey)

183 ASN1_SEQUENCE(CMS_KeyAgreeRecipientInfo) = {
184         ASN1_SIMPLE(CMS_KeyAgreeRecipientInfo, version, LONG),
185         ASN1_EXP(CMS_KeyAgreeRecipientInfo, originator, CMS_OriginatorIdentifier
186         ASN1_EXP_OPT(CMS_KeyAgreeRecipientInfo, ukm, ASN1_OCTET_STRING, 1),
187         ASN1_SIMPLE(CMS_KeyAgreeRecipientInfo, keyEncryptionAlgorithm, X509_ALGO
188         ASN1_SEQUENCE_OF(CMS_KeyAgreeRecipientInfo, recipientEncryptedKeys, CMS_
189 } ASN1_SEQUENCE_END(CMS_KeyAgreeRecipientInfo)

191 ASN1_SEQUENCE(CMS_KEKIdentifier) = {
192         ASN1_SIMPLE(CMS_KEKIdentifier, keyIdentifier, ASN1_OCTET_STRING),
193         ASN1_OPT(CMS_KEKIdentifier, date, ASN1_GENERALIZEDTIME),
```



```
194         ASN1_OPT(CMS_KEKIdentifier, other, CMS_OtherKeyAttribute)
195 } ASN1_SEQUENCE_END(CMS_KEKIdentifier)

197 ASN1_SEQUENCE(CMS_KEKRecipientInfo) = {
198         ASN1_SIMPLE(CMS_KEKRecipientInfo, version, LONG),
199         ASN1_SIMPLE(CMS_KEKRecipientInfo, kekid, CMS_KEKIdentifier),
200         ASN1_SIMPLE(CMS_KEKRecipientInfo, keyEncryptionAlgorithm, X509_ALGOR),
201         ASN1_SIMPLE(CMS_KEKRecipientInfo, encryptedKey, ASN1_OCTET_STRING)
202 } ASN1_SEQUENCE_END(CMS_KEKRecipientInfo)

204 ASN1_SEQUENCE(CMS_PasswordRecipientInfo) = {
205         ASN1_SIMPLE(CMS_PasswordRecipientInfo, version, LONG),
206         ASN1_IMP_OPT(CMS_PasswordRecipientInfo, keyDerivationAlgorithm, X509_ALG
207         ASN1_SIMPLE(CMS_PasswordRecipientInfo, keyEncryptionAlgorithm, X509_ALGO
208         ASN1_SIMPLE(CMS_PasswordRecipientInfo, encryptedKey, ASN1_OCTET_STRING)
209 } ASN1_SEQUENCE_END(CMS_PasswordRecipientInfo)

211 ASN1_SEQUENCE(CMS_OtherRecipientInfo) = {
212   ASN1_SIMPLE(CMS_OtherRecipientInfo, oriType, ASN1_OBJECT),
213   ASN1_OPT(CMS_OtherRecipientInfo, oriValue, ASN1_ANY)
214 } ASN1_SEQUENCE_END(CMS_OtherRecipientInfo)

216 /* Free up RecipientInfo additional data */
217 static int cms_ri_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
218                                                         void *exarg)
219         {
220         if(operation == ASN1_OP_FREE_PRE)
221                 {
222                 CMS_RecipientInfo *ri = (CMS_RecipientInfo *)*pval;
223                 if (ri->type == CMS_RECIPINFO_TRANS)
224                         {
225                         CMS_KeyTransRecipientInfo *ktri = ri->d.ktri;
226                         if (ktri->pkey)
227                                 EVP_PKEY_free(ktri->pkey);
228                         if (ktri->recip)
229                                 X509_free(ktri->recip);
230                         }
231                 else if (ri->type == CMS_RECIPINFO_KEK)
232                         {
233                         CMS_KEKRecipientInfo *kekri = ri->d.kekri;
234                         if (kekri->key)
235                                 {
236                                 OPENSSL_cleanse(kekri->key, kekri->keylen);
237                                 OPENSSL_free(kekri->key);
238                                 }
239                         }
240                 else if (ri->type == CMS_RECIPINFO_PASS)
241                         {
242                         CMS_PasswordRecipientInfo *pwri = ri->d.pwri;
243                         if (pwri->pass)
244                                 {
245                                 OPENSSL_cleanse(pwri->pass, pwri->passlen);
246                                 OPENSSL_free(pwri->pass);
247                                 }
248                         }
249                 }
250         return 1;
251         }

253 ASN1_CHOICE_cb(CMS_RecipientInfo, cms_ri_cb) = {
254         ASN1_SIMPLE(CMS_RecipientInfo, d.ktri, CMS_KeyTransRecipientInfo),
255         ASN1_IMP(CMS_RecipientInfo, d.kari, CMS_KeyAgreeRecipientInfo, 1),
256         ASN1_IMP(CMS_RecipientInfo, d.kekri, CMS_KEKRecipientInfo, 2),
257         ASN1_IMP(CMS_RecipientInfo, d.pwri, CMS_PasswordRecipientInfo, 3),
258         ASN1_IMP(CMS_RecipientInfo, d.ori, CMS_OtherRecipientInfo, 4)
259 } ASN1_CHOICE_END_cb(CMS_RecipientInfo, CMS_RecipientInfo, type)
```

```
261 ASN1_NDEF_SEQUENCE(CMS_EnvelopedData) = {
262         ASN1_SIMPLE(CMS_EnvelopedData, version, LONG),
263         ASN1_IMP_OPT(CMS_EnvelopedData, originatorInfo, CMS_OriginatorInfo, 0),
264         ASN1_SET_OF(CMS_EnvelopedData, recipientInfos, CMS_RecipientInfo),
265         ASN1_SIMPLE(CMS_EnvelopedData, encryptedContentInfo, CMS_EncryptedConten
266         ASN1_IMP_SET_OF_OPT(CMS_EnvelopedData, unprotectedAttrs, X509_ATTRIBUTE,
267 } ASN1_NDEF_SEQUENCE_END(CMS_EnvelopedData)

269 ASN1_NDEF_SEQUENCE(CMS_DigestedData) = {
270         ASN1_SIMPLE(CMS_DigestedData, version, LONG),
271         ASN1_SIMPLE(CMS_DigestedData, digestAlgorithm, X509_ALGOR),
272         ASN1_SIMPLE(CMS_DigestedData, encapContentInfo, CMS_EncapsulatedContentI
273         ASN1_SIMPLE(CMS_DigestedData, digest, ASN1_OCTET_STRING)
274 } ASN1_NDEF_SEQUENCE_END(CMS_DigestedData)

276 ASN1_NDEF_SEQUENCE(CMS_EncryptedData) = {
277         ASN1_SIMPLE(CMS_EncryptedData, version, LONG),
278         ASN1_SIMPLE(CMS_EncryptedData, encryptedContentInfo, CMS_EncryptedConten
279         ASN1_IMP_SET_OF_OPT(CMS_EncryptedData, unprotectedAttrs, X509_ATTRIBUTE,
280 } ASN1_NDEF_SEQUENCE_END(CMS_EncryptedData)

282 ASN1_NDEF_SEQUENCE(CMS_AuthenticatedData) = {
283         ASN1_SIMPLE(CMS_AuthenticatedData, version, LONG),
284         ASN1_IMP_OPT(CMS_AuthenticatedData, originatorInfo, CMS_OriginatorInfo,
285         ASN1_SET_OF(CMS_AuthenticatedData, recipientInfos, CMS_RecipientInfo),
286         ASN1_SIMPLE(CMS_AuthenticatedData, macAlgorithm, X509_ALGOR),
287         ASN1_IMP(CMS_AuthenticatedData, digestAlgorithm, X509_ALGOR, 1),
288         ASN1_SIMPLE(CMS_AuthenticatedData, encapContentInfo, CMS_EncapsulatedCon
289         ASN1_IMP_SET_OF_OPT(CMS_AuthenticatedData, authAttrs, X509_ALGOR, 2),
290         ASN1_SIMPLE(CMS_AuthenticatedData, mac, ASN1_OCTET_STRING),
291         ASN1_IMP_SET_OF_OPT(CMS_AuthenticatedData, unauthAttrs, X509_ALGOR, 3)
292 } ASN1_NDEF_SEQUENCE_END(CMS_AuthenticatedData)

294 ASN1_NDEF_SEQUENCE(CMS_CompressedData) = {
295         ASN1_SIMPLE(CMS_CompressedData, version, LONG),
296         ASN1_SIMPLE(CMS_CompressedData, compressionAlgorithm, X509_ALGOR),
297         ASN1_SIMPLE(CMS_CompressedData, encapContentInfo, CMS_EncapsulatedConten
298 } ASN1_NDEF_SEQUENCE_END(CMS_CompressedData)

300 /* This is the ANY DEFINED BY table for the top level ContentInfo structure */

302 ASN1_ADB_TEMPLATE(cms_default) = ASN1_EXP(CMS_ContentInfo, d.other, ASN1_ANY, 0)

304 ASN1_ADB(CMS_ContentInfo) = {
305         ADB_ENTRY(NID_pkcs7_data, ASN1_NDEF_EXP(CMS_ContentInfo, d.data, ASN1_OC
306         ADB_ENTRY(NID_pkcs7_signed, ASN1_NDEF_EXP(CMS_ContentInfo, d.signedData,
307         ADB_ENTRY(NID_pkcs7_enveloped, ASN1_NDEF_EXP(CMS_ContentInfo, d.envelope
308         ADB_ENTRY(NID_pkcs7_digest, ASN1_NDEF_EXP(CMS_ContentInfo, d.digestedDat
309         ADB_ENTRY(NID_pkcs7_encrypted, ASN1_NDEF_EXP(CMS_ContentInfo, d.encrypte
310         ADB_ENTRY(NID_id_smime_ct_authData, ASN1_NDEF_EXP(CMS_ContentInfo, d.aut
311         ADB_ENTRY(NID_id_smime_ct_compressedData, ASN1_NDEF_EXP(CMS_ContentInfo,
312 } ASN1_ADB_END(CMS_ContentInfo, 0, contentType, 0, &cms_default_tt, NULL);

314 /* CMS streaming support */
315 static int cms_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
316                                                     void *exarg)
317         {
318         ASN1_STREAM_ARG *sarg = exarg;
319         CMS_ContentInfo *cms = NULL;
320         if (pval)
321                 cms = (CMS_ContentInfo *)*pval;
322         else
323                 return 1;
324         switch(operation)
325                 {
```



```
327                 case ASN1_OP_STREAM_PRE:
328                 if (CMS_stream(&sarg->boundary, cms) <= 0)
329                         return 0;
330                 case ASN1_OP_DETACHED_PRE:
331                 sarg->ndef_bio = CMS_dataInit(cms, sarg->out);
332                 if (!sarg->ndef_bio)
333                         return 0;
334                 break;

336                 case ASN1_OP_STREAM_POST:
337                 case ASN1_OP_DETACHED_POST:
338                 if (CMS_dataFinal(cms, sarg->ndef_bio) <= 0)
339                         return 0;
340                 break;

342                 }
343         return 1;
344         }

346 ASN1_NDEF_SEQUENCE_cb(CMS_ContentInfo, cms_cb) = {
347         ASN1_SIMPLE(CMS_ContentInfo, contentType, ASN1_OBJECT),
348         ASN1_ADB_OBJECT(CMS_ContentInfo)
349 } ASN1_NDEF_SEQUENCE_END_cb(CMS_ContentInfo, CMS_ContentInfo)

351 /* Specials for signed attributes */

353 /* When signing attributes we want to reorder them to match the sorted
354  * encoding.
355  */

357 ASN1_ITEM_TEMPLATE(CMS_Attributes_Sign) =
358         ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SET_ORDER, 0, CMS_ATTRIBUTES, X509_ATTRI
359 ASN1_ITEM_TEMPLATE_END(CMS_Attributes_Sign)

361 /* When verifying attributes we need to use the received order. So
362  * we use SEQUENCE OF and tag it to SET OF
363  */

365 ASN1_ITEM_TEMPLATE(CMS_Attributes_Verify) =
366         ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF | ASN1_TFLG_IMPTAG | ASN1_TF
367                               V_ASN1_SET, CMS_ATTRIBUTES, X509_ATTRIBUTE)
368 ASN1_ITEM_TEMPLATE_END(CMS_Attributes_Verify)



372 ASN1_CHOICE(CMS_ReceiptsFrom) = {
373   ASN1_IMP(CMS_ReceiptsFrom, d.allOrFirstTier, LONG, 0),
374   ASN1_IMP_SEQUENCE_OF(CMS_ReceiptsFrom, d.receiptList, GENERAL_NAMES, 1)
375 } ASN1_CHOICE_END(CMS_ReceiptsFrom)

377 ASN1_SEQUENCE(CMS_ReceiptRequest) = {
378   ASN1_SIMPLE(CMS_ReceiptRequest, signedContentIdentifier, ASN1_OCTET_STRING),
379   ASN1_SIMPLE(CMS_ReceiptRequest, receiptsFrom, CMS_ReceiptsFrom),
380   ASN1_SEQUENCE_OF(CMS_ReceiptRequest, receiptsTo, GENERAL_NAMES)
381 } ASN1_SEQUENCE_END(CMS_ReceiptRequest)

383 ASN1_SEQUENCE(CMS_Receipt) = {
384   ASN1_SIMPLE(CMS_Receipt, version, LONG),
385   ASN1_SIMPLE(CMS_Receipt, contentType, ASN1_OBJECT),
386   ASN1_SIMPLE(CMS_Receipt, signedContentIdentifier, ASN1_OCTET_STRING),
387   ASN1_SIMPLE(CMS_Receipt, originatorSignatureValue, ASN1_OCTET_STRING)
388 } ASN1_SEQUENCE_END(CMS_Receipt)
389 #endif /* ! codereview */
```

```
**********************************************************
    6086 Wed Aug 13 19:52:21 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_att.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_att.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include <cms_lcl.h>

/* CMS SignedData Attribute utilities */
```



```
int CMS_signed_get_attr_count(const CMS_SignerInfo *si)
{
        return X509at_get_attr_count(si->signedAttrs);
}

int CMS_signed_get_attr_by_NID(const CMS_SignerInfo *si, int nid,
                               int lastpos)
{
        return X509at_get_attr_by_NID(si->signedAttrs, nid, lastpos);
}

int CMS_signed_get_attr_by_OBJ(const CMS_SignerInfo *si, ASN1_OBJECT *obj,
                               int lastpos)
{
        return X509at_get_attr_by_OBJ(si->signedAttrs, obj, lastpos);
}

X509_ATTRIBUTE *CMS_signed_get_attr(const CMS_SignerInfo *si, int loc)
{
        return X509at_get_attr(si->signedAttrs, loc);
}

X509_ATTRIBUTE *CMS_signed_delete_attr(CMS_SignerInfo *si, int loc)
{
        return X509at_delete_attr(si->signedAttrs, loc);
}

int CMS_signed_add1_attr(CMS_SignerInfo *si, X509_ATTRIBUTE *attr)
{
        if(X509at_add1_attr(&si->signedAttrs, attr)) return 1;
        return 0;
}

int CMS_signed_add1_attr_by_OBJ(CMS_SignerInfo *si,
                        const ASN1_OBJECT *obj, int type,
                        const void *bytes, int len)
{
        if(X509at_add1_attr_by_OBJ(&si->signedAttrs, obj,
                                   type, bytes, len)) return 1;
        return 0;
}

int CMS_signed_add1_attr_by_NID(CMS_SignerInfo *si,
                        int nid, int type,
                        const void *bytes, int len)
{
        if(X509at_add1_attr_by_NID(&si->signedAttrs, nid,
                                   type, bytes, len)) return 1;
        return 0;
}

int CMS_signed_add1_attr_by_txt(CMS_SignerInfo *si,
                        const char *attrname, int type,
                        const void *bytes, int len)
{
        if(X509at_add1_attr_by_txt(&si->signedAttrs, attrname,
                                   type, bytes, len)) return 1;
        return 0;
}

void *CMS_signed_get0_data_by_OBJ(CMS_SignerInfo *si, ASN1_OBJECT *oid,
                                  int lastpos, int type)
{
        return X509at_get0_data_by_OBJ(si->signedAttrs, oid, lastpos, type);
}
```

```
129 int CMS_unsigned_get_attr_count(const CMS_SignerInfo *si)
130 {
131         return X509at_get_attr_count(si->unsignedAttrs);
132 }

134 int CMS_unsigned_get_attr_by_NID(const CMS_SignerInfo *si, int nid,
135                                  int lastpos)
136 {
137         return X509at_get_attr_by_NID(si->unsignedAttrs, nid, lastpos);
138 }

140 int CMS_unsigned_get_attr_by_OBJ(const CMS_SignerInfo *si, ASN1_OBJECT *obj,
141                                  int lastpos)
142 {
143         return X509at_get_attr_by_OBJ(si->unsignedAttrs, obj, lastpos);
144 }

146 X509_ATTRIBUTE *CMS_unsigned_get_attr(const CMS_SignerInfo *si, int loc)
147 {
148         return X509at_get_attr(si->unsignedAttrs, loc);
149 }

151 X509_ATTRIBUTE *CMS_unsigned_delete_attr(CMS_SignerInfo *si, int loc)
152 {
153         return X509at_delete_attr(si->unsignedAttrs, loc);
154 }

156 int CMS_unsigned_add1_attr(CMS_SignerInfo *si, X509_ATTRIBUTE *attr)
157 {
158         if(X509at_add1_attr(&si->unsignedAttrs, attr)) return 1;
159         return 0;
160 }

162 int CMS_unsigned_add1_attr_by_OBJ(CMS_SignerInfo *si,
163                                   const ASN1_OBJECT *obj, int type,
164                                   const void *bytes, int len)
165 {
166         if(X509at_add1_attr_by_OBJ(&si->unsignedAttrs, obj,
167                                    type, bytes, len)) return 1;
168         return 0;
169 }

171 int CMS_unsigned_add1_attr_by_NID(CMS_SignerInfo *si,
172                                   int nid, int type,
173                                   const void *bytes, int len)
174 {
175         if(X509at_add1_attr_by_NID(&si->unsignedAttrs, nid,
176                                    type, bytes, len)) return 1;
177         return 0;
178 }

180 int CMS_unsigned_add1_attr_by_txt(CMS_SignerInfo *si,
181                                   const char *attrname, int type,
182                                   const void *bytes, int len)
183 {
184         if(X509at_add1_attr_by_txt(&si->unsignedAttrs, attrname,
185                                    type, bytes, len)) return 1;
186         return 0;
187 }

189 void *CMS_unsigned_get0_data_by_OBJ(CMS_SignerInfo *si, ASN1_OBJECT *oid,
190                                     int lastpos, int type)
191 {
192         return X509at_get0_data_by_OBJ(si->unsignedAttrs, oid, lastpos, type);
193 }
```



```
195 /* Specific attribute cases */
196 #endif /* ! codereview */
```

```
**********************************************************
    4370 Wed Aug 13 19:52:21 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_cd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_cd.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include <openssl/bio.h>
#ifndef OPENSSL_NO_COMP
```



```
#include <openssl/comp.h>
#endif
#include "cms_lcl.h"

DECLARE_ASN1_ITEM(CMS_CompressedData)

#ifdef ZLIB

/* CMS CompressedData Utilities */

CMS_ContentInfo *cms_CompressedData_create(int comp_nid)
        {
        CMS_ContentInfo *cms;
        CMS_CompressedData *cd;
        /* Will need something cleverer if there is ever more than one
         * compression algorithm or parameters have some meaning...
         */
        if (comp_nid != NID_zlib_compression)
                {
                CMSerr(CMS_F_CMS_COMPRESSEDDATA_CREATE,
                                CMS_R_UNSUPPORTED_COMPRESSION_ALGORITHM);
                return NULL;
                }
        cms = CMS_ContentInfo_new();
        if (!cms)
                return NULL;

        cd = M_ASN1_new_of(CMS_CompressedData);

        if (!cd)
                goto err;

        cms->contentType = OBJ_nid2obj(NID_id_smime_ct_compressedData);
        cms->d.compressedData = cd;

        cd->version = 0;

        X509_ALGOR_set0(cd->compressionAlgorithm,
                        OBJ_nid2obj(NID_zlib_compression),
                        V_ASN1_UNDEF, NULL);

        cd->encapContentInfo->eContentType = OBJ_nid2obj(NID_pkcs7_data);

        return cms;

        err:

        if (cms)
                CMS_ContentInfo_free(cms);

        return NULL;
        }

BIO *cms_CompressedData_init_bio(CMS_ContentInfo *cms)
        {
        CMS_CompressedData *cd;
        ASN1_OBJECT *compoid;
        if (OBJ_obj2nid(cms->contentType) != NID_id_smime_ct_compressedData)
                {
                CMSerr(CMS_F_CMS_COMPRESSEDDATA_INIT_BIO,
                                CMS_R_CONTENT_TYPE_NOT_COMPRESSED_DATA);
                return NULL;
                }
        cd = cms->d.compressedData;
        X509_ALGOR_get0(&compoid, NULL, NULL, cd->compressionAlgorithm);
        if (OBJ_obj2nid(compoid) != NID_zlib_compression)
```

```
128                 {
129                 CMSerr(CMS_F_CMS_COMPRESSEDDATA_INIT_BIO,
130                                 CMS_R_UNSUPPORTED_COMPRESSION_ALGORITHM);
131                 return NULL;
132                 }
133         return BIO_new(BIO_f_zlib());
134         }

136 #endif
137 #endif /* ! codereview */
```

```
**********************************************************
    4333 Wed Aug 13 19:52:21 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_dd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_dd.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include "cms_lcl.h"
```



```
DECLARE_ASN1_ITEM(CMS_DigestedData)

/* CMS DigestedData Utilities */

CMS_ContentInfo *cms_DigestedData_create(const EVP_MD *md)
        {
        CMS_ContentInfo *cms;
        CMS_DigestedData *dd;
        cms = CMS_ContentInfo_new();
        if (!cms)
                return NULL;

        dd = M_ASN1_new_of(CMS_DigestedData);

        if (!dd)
                goto err;

        cms->contentType = OBJ_nid2obj(NID_pkcs7_digest);
        cms->d.digestedData = dd;

        dd->version = 0;
        dd->encapContentInfo->eContentType = OBJ_nid2obj(NID_pkcs7_data);

        cms_DigestAlgorithm_set(dd->digestAlgorithm, md);

        return cms;

        err:

        if (cms)
                CMS_ContentInfo_free(cms);

        return NULL;
        }

BIO *cms_DigestedData_init_bio(CMS_ContentInfo *cms)
        {
        CMS_DigestedData *dd;
        dd = cms->d.digestedData;
        return cms_DigestAlgorithm_init_bio(dd->digestAlgorithm);
        }

int cms_DigestedData_do_final(CMS_ContentInfo *cms, BIO *chain, int verify)
        {
        EVP_MD_CTX mctx;
        unsigned char md[EVP_MAX_MD_SIZE];
        unsigned int mdlen;
        int r = 0;
        CMS_DigestedData *dd;
        EVP_MD_CTX_init(&mctx);

        dd = cms->d.digestedData;

        if (!cms_DigestAlgorithm_find_ctx(&mctx, chain, dd->digestAlgorithm))
                goto err;

        if (EVP_DigestFinal_ex(&mctx, md, &mdlen) <= 0)
                goto err;

        if (verify)
                {
                if (mdlen != (unsigned int)dd->digest->length)
                        {
                        CMSerr(CMS_F_CMS_DIGESTEDDATA_DO_FINAL,
                                CMS_R_MESSAGEDIGEST_WRONG_LENGTH);
                        goto err;
```

```
128                         }
130                 if (memcmp(md, dd->digest->data, mdlen))
131                         CMSerr(CMS_F_CMS_DIGESTEDDATA_DO_FINAL,
132                                 CMS_R_VERIFICATION_FAILURE);
133                 else
134                         r = 1;
135                 }
136         else
137                 {
138                 if (!ASN1_STRING_set(dd->digest, md, mdlen))
139                         goto err;
140                 r = 1;
141                 }
143         err:
144         EVP_MD_CTX_cleanup(&mctx);
146         return r;
148         }
149 #endif /* ! codereview */
```

```
**********************************************************
    7645 Wed Aug 13 19:52:22 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_enc.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include <openssl/rand.h>
#include "cms_lcl.h"
```



```
/* CMS EncryptedData Utilities */

DECLARE_ASN1_ITEM(CMS_EncryptedData)

/* Return BIO based on EncryptedContentInfo and key */

BIO *cms_EncryptedContent_init_bio(CMS_EncryptedContentInfo *ec)
        {
        BIO *b;
        EVP_CIPHER_CTX *ctx;
        const EVP_CIPHER *ciph;
        X509_ALGOR *calg = ec->contentEncryptionAlgorithm;
        unsigned char iv[EVP_MAX_IV_LENGTH], *piv = NULL;
        unsigned char *tkey = NULL;
        size_t tkeylen = 0;

        int ok = 0;

        int enc, keep_key = 0;

        enc = ec->cipher ? 1 : 0;

        b = BIO_new(BIO_f_cipher());
        if (!b)
                {
                CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
                                                        ERR_R_MALLOC_FAILURE);
                return NULL;
                }

        BIO_get_cipher_ctx(b, &ctx);

        if (enc)
                {
                ciph = ec->cipher;
                /* If not keeping key set cipher to NULL so subsequent calls
                 * decrypt.
                 */
                if (ec->key)
                        ec->cipher = NULL;
                }
        else
                {
                ciph = EVP_get_cipherbyobj(calg->algorithm);

                if (!ciph)
                        {
                        CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
                                                        CMS_R_UNKNOWN_CIPHER);
                        goto err;
                        }
                }

        if (EVP_CipherInit_ex(ctx, ciph, NULL, NULL, NULL, enc) <= 0)
                {
                CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
                                CMS_R_CIPHER_INITIALISATION_ERROR);
                goto err;
                }

        if (enc)
                {
                int ivlen;
                calg->algorithm = OBJ_nid2obj(EVP_CIPHER_CTX_type(ctx));
                /* Generate a random IV if we need one */
```

```
128                 ivlen = EVP_CIPHER_CTX_iv_length(ctx);
129                 if (ivlen > 0)
130                         {
131                         if (RAND_pseudo_bytes(iv, ivlen) <= 0)
132                                 goto err;
133                         piv = iv;
134                         }
135                 }
136         else if (EVP_CIPHER_asn1_to_param(ctx, calg->parameter) <= 0)
137                 {
138                 CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
139                                 CMS_R_CIPHER_PARAMETER_INITIALISATION_ERROR);
140                 goto err;
141                 }
142         tkeylen = EVP_CIPHER_CTX_key_length(ctx);
143         /* Generate random session key */
144         if (!enc || !ec->key)
145                 {
146                 tkey = OPENSSL_malloc(tkeylen);
147                 if (!tkey)
148                         {
149                         CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
150                                                         ERR_R_MALLOC_FAILURE);
151                         goto err;
152                         }
153                 if (EVP_CIPHER_CTX_rand_key(ctx, tkey) <= 0)
154                         goto err;
155                 }

157         if (!ec->key)
158                 {
159                 ec->key = tkey;
160                 ec->keylen = tkeylen;
161                 tkey = NULL;
162                 if (enc)
163                         keep_key = 1;
164                 else
165                         ERR_clear_error();

167                 }

169         if (ec->keylen != tkeylen)
170                 {
171                 /* If necessary set key length */
172                 if (EVP_CIPHER_CTX_set_key_length(ctx, ec->keylen) <= 0)
173                         {
174                         /* Only reveal failure if debugging so we don't
175                          * leak information which may be useful in MMA.
176                          */
177                         if (enc || ec->debug)
178                                 {
179                                 CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
180                                                 CMS_R_INVALID_KEY_LENGTH);
181                                 goto err;
182                                 }
183                         else
184                                 {
185                                 /* Use random key */
186                                 OPENSSL_cleanse(ec->key, ec->keylen);
187                                 OPENSSL_free(ec->key);
188                                 ec->key = tkey;
189                                 ec->keylen = tkeylen;
190                                 tkey = NULL;
191                                 ERR_clear_error();
192                                 }
193                         }
```



```
194                 }

196         if (EVP_CipherInit_ex(ctx, NULL, NULL, ec->key, piv, enc) <= 0)
197                 {
198                 CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
199                                 CMS_R_CIPHER_INITIALISATION_ERROR);
200                 goto err;
201                 }

203         if (piv)
204                 {
205                 calg->parameter = ASN1_TYPE_new();
206                 if (!calg->parameter)
207                         {
208                         CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
209                                                         ERR_R_MALLOC_FAILURE);
210                         goto err;
211                         }
212                 if (EVP_CIPHER_param_to_asn1(ctx, calg->parameter) <= 0)
213                         {
214                         CMSerr(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO,
215                                 CMS_R_CIPHER_PARAMETER_INITIALISATION_ERROR);
216                         goto err;
217                         }
218                 }
219         ok = 1;

221         err:
222         if (ec->key && !keep_key)
223                 {
224                 OPENSSL_cleanse(ec->key, ec->keylen);
225                 OPENSSL_free(ec->key);
226                 ec->key = NULL;
227                 }
228         if (tkey)
229                 {
230                 OPENSSL_cleanse(tkey, tkeylen);
231                 OPENSSL_free(tkey);
232                 }
233         if (ok)
234                 return b;
235         BIO_free(b);
236         return NULL;
237         }

239 int cms_EncryptedContent_init(CMS_EncryptedContentInfo *ec,
240                                 const EVP_CIPHER *cipher,
241                                 const unsigned char *key, size_t keylen)
242         {
243         ec->cipher = cipher;
244         if (key)
245                 {
246                 ec->key = OPENSSL_malloc(keylen);
247                 if (!ec->key)
248                         return 0;
249                 memcpy(ec->key, key, keylen);
250                 }
251         ec->keylen = keylen;
252         if (cipher)
253                 ec->contentType = OBJ_nid2obj(NID_pkcs7_data);
254         return 1;
255         }

257 int CMS_EncryptedData_set1_key(CMS_ContentInfo *cms, const EVP_CIPHER *ciph,
258                                 const unsigned char *key, size_t keylen)
259         {
```

```
260         CMS_EncryptedContentInfo *ec;
261         if (!key || !keylen)
262                 {
263                 CMSerr(CMS_F_CMS_ENCRYPTEDDATA_SET1_KEY, CMS_R_NO_KEY);
264                 return 0;
265                 }
266         if (ciph)
267                 {
268                 cms->d.encryptedData = M_ASN1_new_of(CMS_EncryptedData);
269                 if (!cms->d.encryptedData)
270                         {
271                         CMSerr(CMS_F_CMS_ENCRYPTEDDATA_SET1_KEY,
272                                 ERR_R_MALLOC_FAILURE);
273                         return 0;
274                         }
275                 cms->contentType = OBJ_nid2obj(NID_pkcs7_encrypted);
276                 cms->d.encryptedData->version = 0;
277                 }
278         else if (OBJ_obj2nid(cms->contentType) != NID_pkcs7_encrypted)
279                 {
280                 CMSerr(CMS_F_CMS_ENCRYPTEDDATA_SET1_KEY,
281                                                 CMS_R_NOT_ENCRYPTED_DATA);
282                 return 0;
283                 }
284         ec = cms->d.encryptedData->encryptedContentInfo;
285         return cms_EncryptedContent_init(ec, ciph, key, keylen);
286         }

288 BIO *cms_EncryptedData_init_bio(CMS_ContentInfo *cms)
289         {
290         CMS_EncryptedData *enc = cms->d.encryptedData;
291         if (enc->encryptedContentInfo->cipher && enc->unprotectedAttrs)
292                 enc->version = 2;
293         return cms_EncryptedContent_init_bio(enc->encryptedContentInfo);
294         }
295 #endif /* ! codereview */
```

```
**********************************************************
   19207 Wed Aug 13 19:52:22 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_env.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_env.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include <openssl/rand.h>
#include <openssl/aes.h>
```



```
#include "cms_lcl.h"
#include "asn1_locl.h"

/* CMS EnvelopedData Utilities */

DECLARE_ASN1_ITEM(CMS_EnvelopedData)
DECLARE_ASN1_ITEM(CMS_KeyTransRecipientInfo)
DECLARE_ASN1_ITEM(CMS_KEKRecipientInfo)
DECLARE_ASN1_ITEM(CMS_OtherKeyAttribute)

DECLARE_STACK_OF(CMS_RecipientInfo)

CMS_EnvelopedData *cms_get0_enveloped(CMS_ContentInfo *cms)
        {
        if (OBJ_obj2nid(cms->contentType) != NID_pkcs7_enveloped)
                {
                CMSerr(CMS_F_CMS_GET0_ENVELOPED,
                                CMS_R_CONTENT_TYPE_NOT_ENVELOPED_DATA);
                return NULL;
                }
        return cms->d.envelopedData;
        }

static CMS_EnvelopedData *cms_enveloped_data_init(CMS_ContentInfo *cms)
        {
        if (cms->d.other == NULL)
                {
                cms->d.envelopedData = M_ASN1_new_of(CMS_EnvelopedData);
                if (!cms->d.envelopedData)
                        {
                        CMSerr(CMS_F_CMS_ENVELOPED_DATA_INIT,
                                                        ERR_R_MALLOC_FAILURE);
                        return NULL;
                        }
                cms->d.envelopedData->version = 0;
                cms->d.envelopedData->encryptedContentInfo->contentType =
                                                OBJ_nid2obj(NID_pkcs7_data);
                ASN1_OBJECT_free(cms->contentType);
                cms->contentType = OBJ_nid2obj(NID_pkcs7_enveloped);
                return cms->d.envelopedData;
                }
        return cms_get0_enveloped(cms);
        }

STACK_OF(CMS_RecipientInfo) *CMS_get0_RecipientInfos(CMS_ContentInfo *cms)
        {
        CMS_EnvelopedData *env;
        env = cms_get0_enveloped(cms);
        if (!env)
                return NULL;
        return env->recipientInfos;
        }

int CMS_RecipientInfo_type(CMS_RecipientInfo *ri)
        {
        return ri->type;
        }

CMS_ContentInfo *CMS_EnvelopedData_create(const EVP_CIPHER *cipher)
        {
        CMS_ContentInfo *cms;
        CMS_EnvelopedData *env;
        cms = CMS_ContentInfo_new();
        if (!cms)
                goto merr;
        env = cms_enveloped_data_init(cms);
```

```
128         if (!env)
129                 goto merr;
130         if (!cms_EncryptedContent_init(env->encryptedContentInfo,
131                                        cipher, NULL, 0))
132                 goto merr;
133         return cms;
134         merr:
135         if (cms)
136                 CMS_ContentInfo_free(cms);
137         CMSerr(CMS_F_CMS_ENVELOPEDDATA_CREATE, ERR_R_MALLOC_FAILURE);
138         return NULL;
139         }

141 /* Key Transport Recipient Info (KTRI) routines */

143 /* Add a recipient certificate. For now only handle key transport.
144  * If we ever handle key agreement will need updating.
145  */

147 CMS_RecipientInfo *CMS_add1_recipient_cert(CMS_ContentInfo *cms,
148                                        X509 *recip, unsigned int flags)
149         {
150         CMS_RecipientInfo *ri = NULL;
151         CMS_KeyTransRecipientInfo *ktri;
152         CMS_EnvelopedData *env;
153         EVP_PKEY *pk = NULL;
154         int i, type;
155         env = cms_get0_enveloped(cms);
156         if (!env)
157                 goto err;

159         /* Initialize recipient info */
160         ri = M_ASN1_new_of(CMS_RecipientInfo);
161         if (!ri)
162                 goto merr;

164         /* Initialize and add key transport recipient info */

166         ri->d.ktri = M_ASN1_new_of(CMS_KeyTransRecipientInfo);
167         if (!ri->d.ktri)
168                 goto merr;
169         ri->type = CMS_RECIPINFO_TRANS;

171         ktri = ri->d.ktri;

173         X509_check_purpose(recip, -1, -1);
174         pk = X509_get_pubkey(recip);
175         if (!pk)
176                 {
177                 CMSerr(CMS_F_CMS_ADD1_RECIPIENT_CERT,
178                                 CMS_R_ERROR_GETTING_PUBLIC_KEY);
179                 goto err;
180                 }
181         CRYPTO_add(&recip->references, 1, CRYPTO_LOCK_X509);
182         ktri->pkey = pk;
183         ktri->recip = recip;

185         if (flags & CMS_USE_KEYID)
186                 {
187                 ktri->version = 2;
188                 if (env->version < 2)
189                         env->version = 2;
190                 type = CMS_RECIPINFO_KEYIDENTIFIER;
191                 }
192         else
193                 {
```



```
194                 ktri->version = 0;
195                 type = CMS_RECIPINFO_ISSUER_SERIAL;
196                 }

198         /* Not a typo: RecipientIdentifier and SignerIdentifier are the
199          * same structure.
200          */

202         if (!cms_set1_SignerIdentifier(ktri->rid, recip, type))
203                 goto err;

205         if (pk->ameth && pk->ameth->pkey_ctrl)
206                 {
207                 i = pk->ameth->pkey_ctrl(pk, ASN1_PKEY_CTRL_CMS_ENVELOPE,
208                                                 0, ri);
209                 if (i == -2)
210                         {
211                         CMSerr(CMS_F_CMS_ADD1_RECIPIENT_CERT,
212                                 CMS_R_NOT_SUPPORTED_FOR_THIS_KEY_TYPE);
213                         goto err;
214                         }
215                 if (i <= 0)
216                         {
217                         CMSerr(CMS_F_CMS_ADD1_RECIPIENT_CERT,
218                                 CMS_R_CTRL_FAILURE);
219                         goto err;
220                         }
221                 }

223         if (!sk_CMS_RecipientInfo_push(env->recipientInfos, ri))
224                 goto merr;

226         return ri;

228         merr:
229         CMSerr(CMS_F_CMS_ADD1_RECIPIENT_CERT, ERR_R_MALLOC_FAILURE);
230         err:
231         if (ri)
232                 M_ASN1_free_of(ri, CMS_RecipientInfo);
233         return NULL;

235         }

237 int CMS_RecipientInfo_ktri_get0_algs(CMS_RecipientInfo *ri,
238                                         EVP_PKEY **pk, X509 **recip,
239                                         X509_ALGOR **palg)
240         {
241         CMS_KeyTransRecipientInfo *ktri;
242         if (ri->type != CMS_RECIPINFO_TRANS)
243                 {
244                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_GET0_ALGS,
245                         CMS_R_NOT_KEY_TRANSPORT);
246                 return 0;
247                 }

249         ktri = ri->d.ktri;

251         if (pk)
252                 *pk = ktri->pkey;
253         if (recip)
254                 *recip = ktri->recip;
255         if (palg)
256                 *palg = ktri->keyEncryptionAlgorithm;
257         return 1;
258         }
```

```
int CMS_RecipientInfo_ktri_get0_signer_id(CMS_RecipientInfo *ri,
                                          ASN1_OCTET_STRING **keyid,
                                          X509_NAME **issuer, ASN1_INTEGER **sno)
        {
        CMS_KeyTransRecipientInfo *ktri;
        if (ri->type != CMS_RECIPINFO_TRANS)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_GET0_SIGNER_ID,
                        CMS_R_NOT_KEY_TRANSPORT);
                return 0;
                }
        ktri = ri->d.ktri;

        return cms_SignerIdentifier_get0_signer_id(ktri->rid,
                                                   keyid, issuer, sno);
        }

int CMS_RecipientInfo_ktri_cert_cmp(CMS_RecipientInfo *ri, X509 *cert)
        {
        if (ri->type != CMS_RECIPINFO_TRANS)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_CERT_CMP,
                        CMS_R_NOT_KEY_TRANSPORT);
                return -2;
                }
        return cms_SignerIdentifier_cert_cmp(ri->d.ktri->rid, cert);
        }

int CMS_RecipientInfo_set0_pkey(CMS_RecipientInfo *ri, EVP_PKEY *pkey)
        {
        if (ri->type != CMS_RECIPINFO_TRANS)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_SET0_PKEY,
                        CMS_R_NOT_KEY_TRANSPORT);
                return 0;
                }
        ri->d.ktri->pkey = pkey;
        return 1;
        }

/* Encrypt content key in key transport recipient info */

static int cms_RecipientInfo_ktri_encrypt(CMS_ContentInfo *cms,
                                          CMS_RecipientInfo *ri)
        {
        CMS_KeyTransRecipientInfo *ktri;
        CMS_EncryptedContentInfo *ec;
        EVP_PKEY_CTX *pctx = NULL;
        unsigned char *ek = NULL;
        size_t eklen;

        int ret = 0;

        if (ri->type != CMS_RECIPINFO_TRANS)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_ENCRYPT,
                        CMS_R_NOT_KEY_TRANSPORT);
                return 0;
                }
        ktri = ri->d.ktri;
        ec = cms->d.envelopedData->encryptedContentInfo;

        pctx = EVP_PKEY_CTX_new(ktri->pkey, NULL);
        if (!pctx)
                return 0;
```



```
        if (EVP_PKEY_encrypt_init(pctx) <= 0)
                goto err;

        if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_ENCRYPT,
                                EVP_PKEY_CTRL_CMS_ENCRYPT, 0, ri) <= 0)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_ENCRYPT, CMS_R_CTRL_ERROR);
                goto err;
                }

        if (EVP_PKEY_encrypt(pctx, NULL, &eklen, ec->key, ec->keylen) <= 0)
                goto err;

        ek = OPENSSL_malloc(eklen);

        if (ek == NULL)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_ENCRYPT,
                                                        ERR_R_MALLOC_FAILURE);
                goto err;
                }

        if (EVP_PKEY_encrypt(pctx, ek, &eklen, ec->key, ec->keylen) <= 0)
                goto err;

        ASN1_STRING_set0(ktri->encryptedKey, ek, eklen);
        ek = NULL;

        ret = 1;

        err:
        if (pctx)
                EVP_PKEY_CTX_free(pctx);
        if (ek)
                OPENSSL_free(ek);
        return ret;

        }

/* Decrypt content key from KTRI */

static int cms_RecipientInfo_ktri_decrypt(CMS_ContentInfo *cms,
                                                        CMS_RecipientInfo *ri)
        {
        CMS_KeyTransRecipientInfo *ktri = ri->d.ktri;
        EVP_PKEY_CTX *pctx = NULL;
        unsigned char *ek = NULL;
        size_t eklen;
        int ret = 0;
        CMS_EncryptedContentInfo *ec;
        ec = cms->d.envelopedData->encryptedContentInfo;

        if (ktri->pkey == NULL)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_DECRYPT,
                        CMS_R_NO_PRIVATE_KEY);
                return 0;
                }

        pctx = EVP_PKEY_CTX_new(ktri->pkey, NULL);
        if (!pctx)
                return 0;

        if (EVP_PKEY_decrypt_init(pctx) <= 0)
                goto err;
```

```
392         if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_DECRYPT,
393                               EVP_PKEY_CTRL_CMS_DECRYPT, 0, ri) <= 0)
394                 {
395                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_DECRYPT, CMS_R_CTRL_ERROR);
396                 goto err;
397                 }
399         if (EVP_PKEY_decrypt(pctx, NULL, &eklen,
400                              ktri->encryptedKey->data,
401                              ktri->encryptedKey->length) <= 0)
402                 goto err;
404         ek = OPENSSL_malloc(eklen);
406         if (ek == NULL)
407                 {
408                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_DECRYPT,
409                                                         ERR_R_MALLOC_FAILURE);
410                 goto err;
411                 }
413         if (EVP_PKEY_decrypt(pctx, ek, &eklen,
414                              ktri->encryptedKey->data,
415                              ktri->encryptedKey->length) <= 0)
416                 {
417                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KTRI_DECRYPT, CMS_R_CMS_LIB);
418                 goto err;
419                 }
421         ret = 1;
423         if (ec->key)
424                 {
425                 OPENSSL_cleanse(ec->key, ec->keylen);
426                 OPENSSL_free(ec->key);
427                 }
429         ec->key = ek;
430         ec->keylen = eklen;
432         err:
433         if (pctx)
434                 EVP_PKEY_CTX_free(pctx);
435         if (!ret && ek)
436                 OPENSSL_free(ek);
438         return ret;
439         }
441 /* Key Encrypted Key (KEK) RecipientInfo routines */
443 int CMS_RecipientInfo_kekri_id_cmp(CMS_RecipientInfo *ri,
444                                    const unsigned char *id, size_t idlen)
445         {
446         ASN1_OCTET_STRING tmp_os;
447         CMS_KEKRecipientInfo *kekri;
448         if (ri->type != CMS_RECIPINFO_KEK)
449                 {
450                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_ID_CMP, CMS_R_NOT_KEK);
451                 return -2;
452                 }
453         kekri = ri->d.kekri;
454         tmp_os.type = V_ASN1_OCTET_STRING;
455         tmp_os.flags = 0;
456         tmp_os.data = (unsigned char *)id;
457         tmp_os.length = (int)idlen;
```



```
458         return ASN1_OCTET_STRING_cmp(&tmp_os, kekri->kekid->keyIdentifier);
459         }
461 /* For now hard code AES key wrap info */
463 static size_t aes_wrap_keylen(int nid)
464         {
465         switch (nid)
466                 {
467                 case NID_id_aes128_wrap:
468                 return 16;
470                 case NID_id_aes192_wrap:
471                 return  24;
473                 case NID_id_aes256_wrap:
474                 return  32;
476                 default:
477                 return 0;
478                 }
479         }
481 CMS_RecipientInfo *CMS_add0_recipient_key(CMS_ContentInfo *cms, int nid,
482                                           unsigned char *key, size_t keylen,
483                                           unsigned char *id, size_t idlen,
484                                           ASN1_GENERALIZEDTIME *date,
485                                           ASN1_OBJECT *otherTypeId,
486                                           ASN1_TYPE *otherType)
487         {
488         CMS_RecipientInfo *ri = NULL;
489         CMS_EnvelopedData *env;
490         CMS_KEKRecipientInfo *kekri;
491         env = cms_get0_enveloped(cms);
492         if (!env)
493                 goto err;
495         if (nid == NID_undef)
496                 {
497                 switch (keylen)
498                         {
499                         case 16:
500                         nid = NID_id_aes128_wrap;
501                         break;
503                         case  24:
504                         nid = NID_id_aes192_wrap;
505                         break;
507                         case  32:
508                         nid = NID_id_aes256_wrap;
509                         break;
511                         default:
512                         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_KEY,
513                                                 CMS_R_INVALID_KEY_LENGTH);
514                         goto err;
515                         }
517                 }
518         else
519                 {
521                 size_t exp_keylen = aes_wrap_keylen(nid);
523                 if (!exp_keylen)
```

```
524                         {
525                         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_KEY,
526                                         CMS_R_UNSUPPORTED_KEK_ALGORITHM);
527                         goto err;
528                         }

530                 if (keylen != exp_keylen)
531                         {
532                         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_KEY,
533                                         CMS_R_INVALID_KEY_LENGTH);
534                         goto err;
535                         }

537                 }

539         /* Initialize recipient info */
540         ri = M_ASN1_new_of(CMS_RecipientInfo);
541         if (!ri)
542                 goto merr;

544         ri->d.kekri = M_ASN1_new_of(CMS_KEKRecipientInfo);
545         if (!ri->d.kekri)
546                 goto merr;
547         ri->type = CMS_RECIPINFO_KEK;

549         kekri = ri->d.kekri;

551         if (otherTypeId)
552                 {
553                 kekri->kekid->other = M_ASN1_new_of(CMS_OtherKeyAttribute);
554                 if (kekri->kekid->other == NULL)
555                         goto merr;
556                 }

558         if (!sk_CMS_RecipientInfo_push(env->recipientInfos, ri))
559                 goto merr;


562         /* After this point no calls can fail */

564         kekri->version = 4;

566         kekri->key = key;
567         kekri->keylen = keylen;

569         ASN1_STRING_set0(kekri->kekid->keyIdentifier, id, idlen);

571         kekri->kekid->date = date;

573         if (kekri->kekid->other)
574                 {
575                 kekri->kekid->other->keyAttrId = otherTypeId;
576                 kekri->kekid->other->keyAttr = otherType;
577                 }

579         X509_ALGOR_set0(kekri->keyEncryptionAlgorithm,
580                                 OBJ_nid2obj(nid), V_ASN1_UNDEF, NULL);

582         return ri;

584         merr:
585         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_KEY, ERR_R_MALLOC_FAILURE);
586         err:
587         if (ri)
588                 M_ASN1_free_of(ri, CMS_RecipientInfo);
589         return NULL;
```



```
591         }

593 int CMS_RecipientInfo_kekri_get0_id(CMS_RecipientInfo *ri,
594                                         X509_ALGOR **palg,
595                                         ASN1_OCTET_STRING **pid,
596                                         ASN1_GENERALIZEDTIME **pdate,
597                                         ASN1_OBJECT **potherid,
598                                         ASN1_TYPE **pothertype)
599         {
600         CMS_KEKIdentifier *rkid;
601         if (ri->type != CMS_RECIPINFO_KEK)
602                 {
603                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_GET0_ID, CMS_R_NOT_KEK);
604                 return 0;
605                 }
606         rkid =  ri->d.kekri->kekid;
607         if (palg)
608                 *palg = ri->d.kekri->keyEncryptionAlgorithm;
609         if (pid)
610                 *pid = rkid->keyIdentifier;
611         if (pdate)
612                 *pdate = rkid->date;
613         if (potherid)
614                 {
615                 if (rkid->other)
616                         *potherid = rkid->other->keyAttrId;
617                 else
618                         *potherid = NULL;
619                 }
620         if (pothertype)
621                 {
622                 if (rkid->other)
623                         *pothertype = rkid->other->keyAttr;
624                 else
625                         *pothertype = NULL;
626                 }
627         return 1;
628         }

630 int CMS_RecipientInfo_set0_key(CMS_RecipientInfo *ri,
631                                 unsigned char *key, size_t keylen)
632         {
633         CMS_KEKRecipientInfo *kekri;
634         if (ri->type != CMS_RECIPINFO_KEK)
635                 {
636                 CMSerr(CMS_F_CMS_RECIPIENTINFO_SET0_KEY, CMS_R_NOT_KEK);
637                 return 0;
638                 }

640         kekri = ri->d.kekri;
641         kekri->key = key;
642         kekri->keylen = keylen;
643         return 1;
644         }


647 /* Encrypt content key in KEK recipient info */

649 static int cms_RecipientInfo_kekri_encrypt(CMS_ContentInfo *cms,
650                                         CMS_RecipientInfo *ri)
651         {
652         CMS_EncryptedContentInfo *ec;
653         CMS_KEKRecipientInfo *kekri;
654         AES_KEY actx;
655         unsigned char *wkey = NULL;
```

```
656         int wkeylen;
657         int r = 0;

659         ec = cms->d.envelopedData->encryptedContentInfo;

661         kekri = ri->d.kekri;

663         if (!kekri->key)
664                 {
665                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_ENCRYPT, CMS_R_NO_KEY);
666                 return 0;
667                 }

669         if (AES_set_encrypt_key(kekri->key, kekri->keylen << 3, &actx))
670                 {
671                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_ENCRYPT,
672                                                 CMS_R_ERROR_SETTING_KEY);
673                 goto err;
674                 }

676         wkey = OPENSSL_malloc(ec->keylen + 8);

678         if (!wkey)
679                 {
680                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_ENCRYPT,
681                                                 ERR_R_MALLOC_FAILURE);
682                 goto err;
683                 }

685         wkeylen = AES_wrap_key(&actx, NULL, wkey, ec->key, ec->keylen);

687         if (wkeylen <= 0)
688                 {
689                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_ENCRYPT, CMS_R_WRAP_ERROR);
690                 goto err;
691                 }

693         ASN1_STRING_set0(kekri->encryptedKey, wkey, wkeylen);

695         r = 1;

697         err:

699         if (!r && wkey)
700                 OPENSSL_free(wkey);
701         OPENSSL_cleanse(&actx, sizeof(actx));

703         return r;

705         }

707 /* Decrypt content key in KEK recipient info */

709 static int cms_RecipientInfo_kekri_decrypt(CMS_ContentInfo *cms,
710                                            CMS_RecipientInfo *ri)
711         {
712         CMS_EncryptedContentInfo *ec;
713         CMS_KEKRecipientInfo *kekri;
714         AES_KEY actx;
715         unsigned char *ukey = NULL;
716         int ukeylen;
717         int r = 0, wrap_nid;

719         ec = cms->d.envelopedData->encryptedContentInfo;

721         kekri = ri->d.kekri;
```



```
723         if (!kekri->key)
724                 {
725                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT, CMS_R_NO_KEY);
726                 return 0;
727                 }

729         wrap_nid = OBJ_obj2nid(kekri->keyEncryptionAlgorithm->algorithm);
730         if (aes_wrap_keylen(wrap_nid) != kekri->keylen)
731                 {
732                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT,
733                                 CMS_R_INVALID_KEY_LENGTH);
734                 return 0;
735                 }

737         /* If encrypted key length is invalid don't bother */

739         if (kekri->encryptedKey->length < 16)
740                 {
741                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT,
742                                         CMS_R_INVALID_ENCRYPTED_KEY_LENGTH);
743                 goto err;
744                 }

746         if (AES_set_decrypt_key(kekri->key, kekri->keylen << 3, &actx))
747                 {
748                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT,
749                                                 CMS_R_ERROR_SETTING_KEY);
750                 goto err;
751                 }

753         ukey = OPENSSL_malloc(kekri->encryptedKey->length - 8);

755         if (!ukey)
756                 {
757                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT,
758                                                 ERR_R_MALLOC_FAILURE);
759                 goto err;
760                 }

762         ukeylen = AES_unwrap_key(&actx, NULL, ukey,
763                                         kekri->encryptedKey->data,
764                                         kekri->encryptedKey->length);

766         if (ukeylen <= 0)
767                 {
768                 CMSerr(CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT,
769                                                         CMS_R_UNWRAP_ERROR);
770                 goto err;
771                 }

773         ec->key = ukey;
774         ec->keylen = ukeylen;

776         r = 1;

778         err:

780         if (!r && ukey)
781                 OPENSSL_free(ukey);
782         OPENSSL_cleanse(&actx, sizeof(actx));

784         return r;

786         }
```

```
int CMS_RecipientInfo_decrypt(CMS_ContentInfo *cms, CMS_RecipientInfo *ri)
        {
        switch(ri->type)
                {
                case CMS_RECIPINFO_TRANS:
                return cms_RecipientInfo_ktri_decrypt(cms, ri);

                case CMS_RECIPINFO_KEK:
                return cms_RecipientInfo_kekri_decrypt(cms, ri);

                case CMS_RECIPINFO_PASS:
                return cms_RecipientInfo_pwri_crypt(cms, ri, 0);

                default:
                CMSerr(CMS_F_CMS_RECIPIENTINFO_DECRYPT,
                        CMS_R_UNSUPPORTED_RECPIENTINFO_TYPE);
                return 0;
                }
        }

BIO *cms_EnvelopedData_init_bio(CMS_ContentInfo *cms)
        {
        CMS_EncryptedContentInfo *ec;
        STACK_OF(CMS_RecipientInfo) *rinfos;
        CMS_RecipientInfo *ri;
        int i, r, ok = 0;
        BIO *ret;

        /* Get BIO first to set up key */

        ec = cms->d.envelopedData->encryptedContentInfo;
        ret = cms_EncryptedContent_init_bio(ec);

        /* If error or no cipher end of processing */

        if (!ret || !ec->cipher)
                return ret;

        /* Now encrypt content key according to each RecipientInfo type */

        rinfos = cms->d.envelopedData->recipientInfos;

        for (i = 0; i < sk_CMS_RecipientInfo_num(rinfos); i++)
                {
                ri = sk_CMS_RecipientInfo_value(rinfos, i);

                switch (ri->type)
                        {
                        case CMS_RECIPINFO_TRANS:
                        r = cms_RecipientInfo_ktri_encrypt(cms, ri);
                        break;

                        case CMS_RECIPINFO_KEK:
                        r = cms_RecipientInfo_kekri_encrypt(cms, ri);
                        break;

                        case CMS_RECIPINFO_PASS:
                        r = cms_RecipientInfo_pwri_crypt(cms, ri, 1);
                        break;

                        default:
                        CMSerr(CMS_F_CMS_ENVELOPEDDATA_INIT_BIO,
                                CMS_R_UNSUPPORTED_RECIPIENT_TYPE);
                        goto err;
                        }
```



```
                if (r <= 0)
                        {
                        CMSerr(CMS_F_CMS_ENVELOPEDDATA_INIT_BIO,
                                CMS_R_ERROR_SETTING_RECIPIENTINFO);
                        goto err;
                        }
                }

        ok = 1;

        err:
        ec->cipher = NULL;
        if (ec->key)
                {
                OPENSSL_cleanse(ec->key, ec->keylen);
                OPENSSL_free(ec->key);
                ec->key = NULL;
                ec->keylen = 0;
                }
        if (ok)
                return ret;
        BIO_free(ret);
        return NULL;

        }
#endif /* ! codereview */
```

```
**********************************************************
   13889 Wed Aug 13 19:52:22 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_err.c */
/* ====================================================================
 * Copyright (c) 1999-2009 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/cms.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_CMS,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_CMS,0,reason)

static ERR_STRING_DATA CMS_str_functs[]=
        {
{ERR_FUNC(CMS_F_CHECK_CONTENT), "CHECK_CONTENT"},
{ERR_FUNC(CMS_F_CMS_ADD0_CERT), "CMS_add0_cert"},
{ERR_FUNC(CMS_F_CMS_ADD0_RECIPIENT_KEY),        "CMS_add0_recipient_key"},
{ERR_FUNC(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD),   "CMS_add0_recipient_password"},
{ERR_FUNC(CMS_F_CMS_ADD1_RECEIPTREQUEST),       "CMS_add1_ReceiptRequest"},
{ERR_FUNC(CMS_F_CMS_ADD1_RECIPIENT_CERT),       "CMS_add1_recipient_cert"},
{ERR_FUNC(CMS_F_CMS_ADD1_SIGNER),       "CMS_add1_signer"},
{ERR_FUNC(CMS_F_CMS_ADD1_SIGNINGTIME),  "CMS_ADD1_SIGNINGTIME"},
{ERR_FUNC(CMS_F_CMS_COMPRESS),  "CMS_compress"},
{ERR_FUNC(CMS_F_CMS_COMPRESSEDDATA_CREATE),     "cms_CompressedData_create"},
{ERR_FUNC(CMS_F_CMS_COMPRESSEDDATA_INIT_BIO),   "cms_CompressedData_init_bio"},
{ERR_FUNC(CMS_F_CMS_COPY_CONTENT),      "CMS_COPY_CONTENT"},
{ERR_FUNC(CMS_F_CMS_COPY_MESSAGEDIGEST),        "CMS_COPY_MESSAGEDIGEST"},
{ERR_FUNC(CMS_F_CMS_DATA),      "CMS_data"},
{ERR_FUNC(CMS_F_CMS_DATAFINAL), "CMS_dataFinal"},
{ERR_FUNC(CMS_F_CMS_DATAINIT),  "CMS_dataInit"},
{ERR_FUNC(CMS_F_CMS_DECRYPT),   "CMS_decrypt"},
{ERR_FUNC(CMS_F_CMS_DECRYPT_SET1_KEY),  "CMS_decrypt_set1_key"},
{ERR_FUNC(CMS_F_CMS_DECRYPT_SET1_PASSWORD),     "CMS_decrypt_set1_password"},
{ERR_FUNC(CMS_F_CMS_DECRYPT_SET1_PKEY), "CMS_decrypt_set1_pkey"},
{ERR_FUNC(CMS_F_CMS_DIGESTALGORITHM_FIND_CTX),  "cms_DigestAlgorithm_find_ctx"},
{ERR_FUNC(CMS_F_CMS_DIGESTALGORITHM_INIT_BIO),  "cms_DigestAlgorithm_init_bio"},
{ERR_FUNC(CMS_F_CMS_DIGESTEDDATA_DO_FINAL),     "cms_DigestedData_do_final"},
{ERR_FUNC(CMS_F_CMS_DIGEST_VERIFY),     "CMS_digest_verify"},
{ERR_FUNC(CMS_F_CMS_ENCODE_RECEIPT),    "cms_encode_Receipt"},
{ERR_FUNC(CMS_F_CMS_ENCRYPT),   "CMS_encrypt"},
{ERR_FUNC(CMS_F_CMS_ENCRYPTEDCONTENT_INIT_BIO), "cms_EncryptedContent_init_bio"}
{ERR_FUNC(CMS_F_CMS_ENCRYPTEDDATA_DECRYPT),     "CMS_EncryptedData_decrypt"},
{ERR_FUNC(CMS_F_CMS_ENCRYPTEDDATA_ENCRYPT),     "CMS_EncryptedData_encrypt"},
{ERR_FUNC(CMS_F_CMS_ENCRYPTEDDATA_SET1_KEY),    "CMS_EncryptedData_set1_key"},
{ERR_FUNC(CMS_F_CMS_ENVELOPEDDATA_CREATE),      "CMS_EnvelopedData_create"},
{ERR_FUNC(CMS_F_CMS_ENVELOPEDDATA_INIT_BIO),    "cms_EnvelopedData_init_bio"},
{ERR_FUNC(CMS_F_CMS_ENVELOPED_DATA_INIT),       "CMS_ENVELOPED_DATA_INIT"},
{ERR_FUNC(CMS_F_CMS_FINAL),     "CMS_final"},
{ERR_FUNC(CMS_F_CMS_GET0_CERTIFICATE_CHOICES),  "CMS_GET0_CERTIFICATE_CHOICES"},
{ERR_FUNC(CMS_F_CMS_GET0_CONTENT),      "CMS_get0_content"},
{ERR_FUNC(CMS_F_CMS_GET0_ECONTENT_TYPE),        "CMS_GET0_ECONTENT_TYPE"},
{ERR_FUNC(CMS_F_CMS_GET0_ENVELOPED),    "cms_get0_enveloped"},
{ERR_FUNC(CMS_F_CMS_GET0_REVOCATION_CHOICES),   "CMS_GET0_REVOCATION_CHOICES"},
{ERR_FUNC(CMS_F_CMS_GET0_SIGNED),       "CMS_GET0_SIGNED"},
{ERR_FUNC(CMS_F_CMS_MSGSIGDIGEST_ADD1), "cms_msgSigDigest_add1"},
{ERR_FUNC(CMS_F_CMS_RECEIPTREQUEST_CREATE0),    "CMS_ReceiptRequest_create0"},
{ERR_FUNC(CMS_F_CMS_RECEIPT_VERIFY),    "cms_Receipt_verify"},
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_DECRYPT),     "CMS_RecipientInfo_decrypt"},
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KEKRI_DECRYPT),       "CMS_RECIPIENTINFO_KEKRI
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KEKRI_ENCRYPT),       "CMS_RECIPIENTINFO_KEKRI
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KEKRI_GET0_ID),       "CMS_RecipientInfo_kekri
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KEKRI_ID_CMP),        "CMS_RecipientInfo_kekri
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KTRI_CERT_CMP),       "CMS_RecipientInfo_ktri_
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KTRI_DECRYPT),        "CMS_RECIPIENTINFO_KTRI_
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KTRI_ENCRYPT),        "CMS_RECIPIENTINFO_KTRI_
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KTRI_GET0_ALGS),      "CMS_RecipientInfo_ktri_
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_KTRI_GET0_SIGNER_ID), "CMS_RecipientInfo_ktri_
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT),  "cms_RecipientInfo_pwri_crypt"},
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_SET0_KEY),    "CMS_RecipientInfo_set0_key"},
```

```
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_SET0_PASSWORD),       "CMS_RecipientInfo_set0_
{ERR_FUNC(CMS_F_CMS_RECIPIENTINFO_SET0_PKEY),   "CMS_RecipientInfo_set0_pkey"},
{ERR_FUNC(CMS_F_CMS_SET1_SIGNERIDENTIFIER),     "cms_set1_SignerIdentifier"},
{ERR_FUNC(CMS_F_CMS_SET_DETACHED),      "CMS_set_detached"},
{ERR_FUNC(CMS_F_CMS_SIGN),      "CMS_sign"},
{ERR_FUNC(CMS_F_CMS_SIGNED_DATA_INIT),  "CMS_SIGNED_DATA_INIT"},
{ERR_FUNC(CMS_F_CMS_SIGNERINFO_CONTENT_SIGN),   "CMS_SIGNERINFO_CONTENT_SIGN"},
{ERR_FUNC(CMS_F_CMS_SIGNERINFO_SIGN),   "CMS_SignerInfo_sign"},
{ERR_FUNC(CMS_F_CMS_SIGNERINFO_VERIFY), "CMS_SignerInfo_verify"},
{ERR_FUNC(CMS_F_CMS_SIGNERINFO_VERIFY_CERT),    "CMS_SIGNERINFO_VERIFY_CERT"},
{ERR_FUNC(CMS_F_CMS_SIGNERINFO_VERIFY_CONTENT), "CMS_SignerInfo_verify_content"}
{ERR_FUNC(CMS_F_CMS_SIGN_RECEIPT),      "CMS_sign_receipt"},
{ERR_FUNC(CMS_F_CMS_STREAM),    "CMS_stream"},
{ERR_FUNC(CMS_F_CMS_UNCOMPRESS),        "CMS_uncompress"},
{ERR_FUNC(CMS_F_CMS_VERIFY),    "CMS_verify"},
{0,NULL}
        };

static ERR_STRING_DATA CMS_str_reasons[]=
        {
{ERR_REASON(CMS_R_ADD_SIGNER_ERROR)      ,"add signer error"},
{ERR_REASON(CMS_R_CERTIFICATE_ALREADY_PRESENT),"certificate already present"},
{ERR_REASON(CMS_R_CERTIFICATE_HAS_NO_KEYID),"certificate has no keyid"},
{ERR_REASON(CMS_R_CERTIFICATE_VERIFY_ERROR),"certificate verify error"},
{ERR_REASON(CMS_R_CIPHER_INITIALISATION_ERROR),"cipher initialisation error"},
{ERR_REASON(CMS_R_CIPHER_PARAMETER_INITIALISATION_ERROR),"cipher parameter initi
{ERR_REASON(CMS_R_CMS_DATAFINAL_ERROR)   ,"cms datafinal error"},
{ERR_REASON(CMS_R_CMS_LIB)               ,"cms lib"},
{ERR_REASON(CMS_R_CONTENTIDENTIFIER_MISMATCH),"contentidentifier mismatch"},
{ERR_REASON(CMS_R_CONTENT_NOT_FOUND)     ,"content not found"},
{ERR_REASON(CMS_R_CONTENT_TYPE_MISMATCH) ,"content type mismatch"},
{ERR_REASON(CMS_R_CONTENT_TYPE_NOT_COMPRESSED_DATA),"content type not compressed
{ERR_REASON(CMS_R_CONTENT_TYPE_NOT_ENVELOPED_DATA),"content type not enveloped d
{ERR_REASON(CMS_R_CONTENT_TYPE_NOT_SIGNED_DATA),"content type not signed data"},
{ERR_REASON(CMS_R_CONTENT_VERIFY_ERROR)  ,"content verify error"},
{ERR_REASON(CMS_R_CTRL_ERROR)            ,"ctrl error"},
{ERR_REASON(CMS_R_CTRL_FAILURE)          ,"ctrl failure"},
{ERR_REASON(CMS_R_DECRYPT_ERROR)         ,"decrypt error"},
{ERR_REASON(CMS_R_DIGEST_ERROR)          ,"digest error"},
{ERR_REASON(CMS_R_ERROR_GETTING_PUBLIC_KEY),"error getting public key"},
{ERR_REASON(CMS_R_ERROR_READING_MESSAGEDIGEST_ATTRIBUTE),"error reading messaged
{ERR_REASON(CMS_R_ERROR_SETTING_KEY)     ,"error setting key"},
{ERR_REASON(CMS_R_ERROR_SETTING_RECIPIENTINFO),"error setting recipientinfo"},
{ERR_REASON(CMS_R_INVALID_ENCRYPTED_KEY_LENGTH),"invalid encrypted key length"},
{ERR_REASON(CMS_R_INVALID_KEY_ENCRYPTION_PARAMETER),"invalid key encryption para
{ERR_REASON(CMS_R_INVALID_KEY_LENGTH)    ,"invalid key length"},
{ERR_REASON(CMS_R_MD_BIO_INIT_ERROR)     ,"md bio init error"},
{ERR_REASON(CMS_R_MESSAGEDIGEST_ATTRIBUTE_WRONG_LENGTH),"messagedigest attribute
{ERR_REASON(CMS_R_MESSAGEDIGEST_WRONG_LENGTH),"messagedigest wrong length"},
{ERR_REASON(CMS_R_MSGSIGDIGEST_ERROR)    ,"msgsigdigest error"},
{ERR_REASON(CMS_R_MSGSIGDIGEST_VERIFICATION_FAILURE),"msgsigdigest verification
{ERR_REASON(CMS_R_MSGSIGDIGEST_WRONG_LENGTH),"msgsigdigest wrong length"},
{ERR_REASON(CMS_R_NEED_ONE_SIGNER)       ,"need one signer"},
{ERR_REASON(CMS_R_NOT_A_SIGNED_RECEIPT)  ,"not a signed receipt"},
{ERR_REASON(CMS_R_NOT_ENCRYPTED_DATA)    ,"not encrypted data"},
{ERR_REASON(CMS_R_NOT_KEK)               ,"not kek"},
{ERR_REASON(CMS_R_NOT_KEY_TRANSPORT)     ,"not key transport"},
{ERR_REASON(CMS_R_NOT_PWRI)              ,"not pwri"},
{ERR_REASON(CMS_R_NOT_SUPPORTED_FOR_THIS_KEY_TYPE),"not supported for this key t
{ERR_REASON(CMS_R_NO_CIPHER)             ,"no cipher"},
{ERR_REASON(CMS_R_NO_CONTENT)            ,"no content"},
{ERR_REASON(CMS_R_NO_CONTENT_TYPE)       ,"no content type"},
{ERR_REASON(CMS_R_NO_DEFAULT_DIGEST)     ,"no default digest"},
{ERR_REASON(CMS_R_NO_DIGEST_SET)         ,"no digest set"},
{ERR_REASON(CMS_R_NO_KEY)                ,"no key"},
{ERR_REASON(CMS_R_NO_KEY_OR_CERT)        ,"no key or cert"},
```



```
{ERR_REASON(CMS_R_NO_MATCHING_DIGEST)    ,"no matching digest"},
{ERR_REASON(CMS_R_NO_MATCHING_RECIPIENT) ,"no matching recipient"},
{ERR_REASON(CMS_R_NO_MATCHING_SIGNATURE) ,"no matching signature"},
{ERR_REASON(CMS_R_NO_MSGSIGDIGEST)       ,"no msgsigdigest"},
{ERR_REASON(CMS_R_NO_PASSWORD)           ,"no password"},
{ERR_REASON(CMS_R_NO_PRIVATE_KEY)        ,"no private key"},
{ERR_REASON(CMS_R_NO_PUBLIC_KEY)         ,"no public key"},
{ERR_REASON(CMS_R_NO_RECEIPT_REQUEST)    ,"no receipt request"},
{ERR_REASON(CMS_R_NO_SIGNERS)            ,"no signers"},
{ERR_REASON(CMS_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE),"private key does not
{ERR_REASON(CMS_R_RECEIPT_DECODE_ERROR)  ,"receipt decode error"},
{ERR_REASON(CMS_R_RECIPIENT_ERROR)       ,"recipient error"},
{ERR_REASON(CMS_R_SIGNER_CERTIFICATE_NOT_FOUND),"signer certificate not found"},
{ERR_REASON(CMS_R_SIGNFINAL_ERROR)       ,"signfinal error"},
{ERR_REASON(CMS_R_SMIME_TEXT_ERROR)      ,"smime text error"},
{ERR_REASON(CMS_R_STORE_INIT_ERROR)      ,"store init error"},
{ERR_REASON(CMS_R_TYPE_NOT_COMPRESSED_DATA),"type not compressed data"},
{ERR_REASON(CMS_R_TYPE_NOT_DATA)         ,"type not data"},
{ERR_REASON(CMS_R_TYPE_NOT_DIGESTED_DATA),"type not digested data"},
{ERR_REASON(CMS_R_TYPE_NOT_ENCRYPTED_DATA),"type not encrypted data"},
{ERR_REASON(CMS_R_TYPE_NOT_ENVELOPED_DATA),"type not enveloped data"},
{ERR_REASON(CMS_R_UNABLE_TO_FINALIZE_CONTEXT),"unable to finalize context"},
{ERR_REASON(CMS_R_UNKNOWN_CIPHER)        ,"unknown cipher"},
{ERR_REASON(CMS_R_UNKNOWN_DIGEST_ALGORIHM),"unknown digest algorihm"},
{ERR_REASON(CMS_R_UNKNOWN_ID)            ,"unknown id"},
{ERR_REASON(CMS_R_UNSUPPORTED_COMPRESSION_ALGORITHM),"unsupported compression al
{ERR_REASON(CMS_R_UNSUPPORTED_CONTENT_TYPE),"unsupported content type"},
{ERR_REASON(CMS_R_UNSUPPORTED_KEK_ALGORITHM),"unsupported kek algorithm"},
{ERR_REASON(CMS_R_UNSUPPORTED_KEY_ENCRYPTION_ALGORITHM),"unsupported key encrypt
{ERR_REASON(CMS_R_UNSUPPORTED_RECIPIENT_TYPE),"unsupported recipient type"},
{ERR_REASON(CMS_R_UNSUPPORTED_RECPIENTINFO_TYPE),"unsupported recpientinfo type"
{ERR_REASON(CMS_R_UNSUPPORTED_TYPE)      ,"unsupported type"},
{ERR_REASON(CMS_R_UNWRAP_ERROR)          ,"unwrap error"},
{ERR_REASON(CMS_R_UNWRAP_FAILURE)        ,"unwrap failure"},
{ERR_REASON(CMS_R_VERIFICATION_FAILURE)  ,"verification failure"},
{ERR_REASON(CMS_R_WRAP_ERROR)            ,"wrap error"},
{0,NULL}
        };

#endif

void ERR_load_CMS_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(CMS_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,CMS_str_functs);
                ERR_load_strings(0,CMS_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
   10436 Wed Aug 13 19:52:22 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_ess.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_ess.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/rand.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include "cms_lcl.h"
```



```
DECLARE_ASN1_ITEM(CMS_ReceiptRequest)
DECLARE_ASN1_ITEM(CMS_Receipt)

IMPLEMENT_ASN1_FUNCTIONS(CMS_ReceiptRequest)

/* ESS services: for now just Signed Receipt related */

int CMS_get1_ReceiptRequest(CMS_SignerInfo *si, CMS_ReceiptRequest **prr)
        {
        ASN1_STRING *str;
        CMS_ReceiptRequest *rr = NULL;
        if (prr)
                *prr = NULL;
        str = CMS_signed_get0_data_by_OBJ(si,
                                OBJ_nid2obj(NID_id_smime_aa_receiptRequest),
                                        -3, V_ASN1_SEQUENCE);
        if (!str)
                return 0;

        rr = ASN1_item_unpack(str, ASN1_ITEM_rptr(CMS_ReceiptRequest));
        if (!rr)
                return -1;
        if (prr)
                *prr = rr;
        else
                CMS_ReceiptRequest_free(rr);
        return 1;
        }

CMS_ReceiptRequest *CMS_ReceiptRequest_create0(unsigned char *id, int idlen,
                                int allorfirst,
                                STACK_OF(GENERAL_NAMES) *receiptList,
                                STACK_OF(GENERAL_NAMES) *receiptsTo)
        {
        CMS_ReceiptRequest *rr = NULL;

        rr = CMS_ReceiptRequest_new();
        if (!rr)
                goto merr;
        if (id)
                ASN1_STRING_set0(rr->signedContentIdentifier, id, idlen);
        else
                {
                if (!ASN1_STRING_set(rr->signedContentIdentifier, NULL, 32))
                        goto merr;
                if (RAND_pseudo_bytes(rr->signedContentIdentifier->data, 32)
                                        <= 0)
                        goto err;
                }

        sk_GENERAL_NAMES_pop_free(rr->receiptsTo, GENERAL_NAMES_free);
        rr->receiptsTo = receiptsTo;

        if (receiptList)
                {
                rr->receiptsFrom->type = 1;
                rr->receiptsFrom->d.receiptList = receiptList;
                }
        else
                {
                rr->receiptsFrom->type = 0;
                rr->receiptsFrom->d.allOrFirstTier = allorfirst;
                }

        return rr;
```

```
129         merr:
130         CMSerr(CMS_F_CMS_RECEIPTREQUEST_CREATE0, ERR_R_MALLOC_FAILURE);

132         err:
133         if (rr)
134                 CMS_ReceiptRequest_free(rr);

136         return NULL;

138         }

140 int CMS_add1_ReceiptRequest(CMS_SignerInfo *si, CMS_ReceiptRequest *rr)
141         {
142         unsigned char *rrder = NULL;
143         int rrderlen, r = 0;

145         rrderlen = i2d_CMS_ReceiptRequest(rr, &rrder);
146         if (rrderlen < 0)
147                 goto merr;

149         if (!CMS_signed_add1_attr_by_NID(si, NID_id_smime_aa_receiptRequest,
150                                          V_ASN1_SEQUENCE, rrder, rrderlen))
151                 goto merr;

153         r = 1;

155         merr:
156         if (!r)
157                 CMSerr(CMS_F_CMS_ADD1_RECEIPTREQUEST, ERR_R_MALLOC_FAILURE);

159         if (rrder)
160                 OPENSSL_free(rrder);

162         return r;

164         }

166 void CMS_ReceiptRequest_get0_values(CMS_ReceiptRequest *rr,
167                                     ASN1_STRING **pcid,
168                                     int *pallorfirst,
169                                     STACK_OF(GENERAL_NAMES) **plist,
170                                     STACK_OF(GENERAL_NAMES) **prto)
171         {
172         if (pcid)
173                 *pcid = rr->signedContentIdentifier;
174         if (rr->receiptsFrom->type == 0)
175                 {
176                 if (pallorfirst)
177                         *pallorfirst = (int)rr->receiptsFrom->d.allOrFirstTier;
178                 if (plist)
179                         *plist = NULL;
180                 }
181         else
182                 {
183                 if (pallorfirst)
184                         *pallorfirst = -1;
185                 if (plist)
186                         *plist = rr->receiptsFrom->d.receiptList;
187                 }
188         if (prto)
189                 *prto = rr->receiptsTo;
190         }

192 /* Digest a SignerInfo structure for msgSigDigest attribute processing */
```



```
194 static int cms_msgSigDigest(CMS_SignerInfo *si,
195                             unsigned char *dig, unsigned int *diglen)
196         {
197         const EVP_MD *md;
198         md = EVP_get_digestbyobj(si->digestAlgorithm->algorithm);
199         if (md == NULL)
200                 return 0;
201         if (!ASN1_item_digest(ASN1_ITEM_rptr(CMS_Attributes_Verify), md,
202                                                 si->signedAttrs, dig, diglen))
203                 return 0;
204         return 1;
205         }

207 /* Add a msgSigDigest attribute to a SignerInfo */

209 int cms_msgSigDigest_add1(CMS_SignerInfo *dest, CMS_SignerInfo *src)
210         {
211         unsigned char dig[EVP_MAX_MD_SIZE];
212         unsigned int diglen;
213         if (!cms_msgSigDigest(src, dig, &diglen))
214                 {
215                 CMSerr(CMS_F_CMS_MSGSIGDIGEST_ADD1, CMS_R_MSGSIGDIGEST_ERROR);
216                 return 0;
217                 }
218         if (!CMS_signed_add1_attr_by_NID(dest, NID_id_smime_aa_msgSigDigest,
219                                          V_ASN1_OCTET_STRING, dig, diglen))
220                 {
221                 CMSerr(CMS_F_CMS_MSGSIGDIGEST_ADD1, ERR_R_MALLOC_FAILURE);
222                 return 0;
223                 }
224         return 1;
225         }

227 /* Verify signed receipt after it has already passed normal CMS verify */

229 int cms_Receipt_verify(CMS_ContentInfo *cms, CMS_ContentInfo *req_cms)
230         {
231         int r = 0, i;
232         CMS_ReceiptRequest *rr = NULL;
233         CMS_Receipt *rct = NULL;
234         STACK_OF(CMS_SignerInfo) *sis, *osis;
235         CMS_SignerInfo *si, *osi = NULL;
236         ASN1_OCTET_STRING *msig, **pcont;
237         ASN1_OBJECT *octype;
238         unsigned char dig[EVP_MAX_MD_SIZE];
239         unsigned int diglen;

241         /* Get SignerInfos, also checks SignedData content type */
242         osis = CMS_get0_SignerInfos(req_cms);
243         sis = CMS_get0_SignerInfos(cms);
244         if (!osis || !sis)
245                 goto err;

247         if (sk_CMS_SignerInfo_num(sis) != 1)
248                 {
249                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_NEED_ONE_SIGNER);
250                 goto err;
251                 }

253         /* Check receipt content type */
254         if (OBJ_obj2nid(CMS_get0_eContentType(cms)) != NID_id_smime_ct_receipt)
255                 {
256                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_NOT_A_SIGNED_RECEIPT);
257                 goto err;
258                 }
```

```
260         /* Extract and decode receipt content */
261         pcont = CMS_get0_content(cms);
262         if (!pcont || !*pcont)
263                 {
264                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_NO_CONTENT);
265                 goto err;
266                 }
268         rct = ASN1_item_unpack(*pcont, ASN1_ITEM_rptr(CMS_Receipt));
270         if (!rct)
271                 {
272                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_RECEIPT_DECODE_ERROR);
273                 goto err;
274                 }
276         /* Locate original request */
278         for (i = 0; i < sk_CMS_SignerInfo_num(osis); i++)
279                 {
280                 osi = sk_CMS_SignerInfo_value(osis, i);
281                 if (!ASN1_STRING_cmp(osi->signature,
282                                      rct->originatorSignatureValue))
283                         break;
284                 }
286         if (i == sk_CMS_SignerInfo_num(osis))
287                 {
288                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_NO_MATCHING_SIGNATURE);
289                 goto err;
290                 }
292         si = sk_CMS_SignerInfo_value(sis, 0);
294         /* Get msgSigDigest value and compare */
296         msig = CMS_signed_get0_data_by_OBJ(si,
297                                    OBJ_nid2obj(NID_id_smime_aa_msgSigDigest),
298                                            -3, V_ASN1_OCTET_STRING);
300         if (!msig)
301                 {
302                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_NO_MSGSIGDIGEST);
303                 goto err;
304                 }
306         if (!cms_msgSigDigest(osi, dig, &diglen))
307                 {
308                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_MSGSIGDIGEST_ERROR);
309                 goto err;
310                 }
312         if (diglen != (unsigned int)msig->length)
313                         {
314                         CMSerr(CMS_F_CMS_RECEIPT_VERIFY,
315                                 CMS_R_MSGSIGDIGEST_WRONG_LENGTH);
316                         goto err;
317                         }
319         if (memcmp(dig, msig->data, diglen))
320                         {
321                         CMSerr(CMS_F_CMS_RECEIPT_VERIFY,
322                                 CMS_R_MSGSIGDIGEST_VERIFICATION_FAILURE);
323                         goto err;
324                         }
```



```
326         /* Compare content types */
328         octype = CMS_signed_get0_data_by_OBJ(osi,
329                                    OBJ_nid2obj(NID_pkcs9_contentType),
330                                            -3, V_ASN1_OBJECT);
331         if (!octype)
332                 {
333                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_NO_CONTENT_TYPE);
334                 goto err;
335                 }
337         /* Compare details in receipt request */
339         if (OBJ_cmp(octype, rct->contentType))
340                 {
341                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_CONTENT_TYPE_MISMATCH);
342                 goto err;
343                 }
345         /* Get original receipt request details */
347         if (CMS_get1_ReceiptRequest(osi, &rr) <= 0)
348                 {
349                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY, CMS_R_NO_RECEIPT_REQUEST);
350                 goto err;
351                 }
353         if (ASN1_STRING_cmp(rr->signedContentIdentifier,
354                                      rct->signedContentIdentifier))
355                 {
356                 CMSerr(CMS_F_CMS_RECEIPT_VERIFY,
357                                      CMS_R_CONTENTIDENTIFIER_MISMATCH);
358                 goto err;
359                 }
361         r = 1;
363         err:
364         if (rr)
365                 CMS_ReceiptRequest_free(rr);
366         if (rct)
367                 M_ASN1_free_of(rct, CMS_Receipt);
369         return r;
371         }
373 /* Encode a Receipt into an OCTET STRING read for including into content of
374  * a SignedData ContentInfo.
375  */
377 ASN1_OCTET_STRING *cms_encode_Receipt(CMS_SignerInfo *si)
378         {
379         CMS_Receipt rct;
380         CMS_ReceiptRequest *rr = NULL;
381         ASN1_OBJECT *ctype;
382         ASN1_OCTET_STRING *os = NULL;
384         /* Get original receipt request */
386         /* Get original receipt request details */
388         if (CMS_get1_ReceiptRequest(si, &rr) <= 0)
389                 {
390                 CMSerr(CMS_F_CMS_ENCODE_RECEIPT, CMS_R_NO_RECEIPT_REQUEST);
391                 goto err;
```

```
392                 }

394         /* Get original content type */

396         ctype = CMS_signed_get0_data_by_OBJ(si,
397                                     OBJ_nid2obj(NID_pkcs9_contentType),
398                                             -3, V_ASN1_OBJECT);
399         if (!ctype)
400                 {
401                 CMSerr(CMS_F_CMS_ENCODE_RECEIPT, CMS_R_NO_CONTENT_TYPE);
402                 goto err;
403                 }

405         rct.version = 1;
406         rct.contentType = ctype;
407         rct.signedContentIdentifier = rr->signedContentIdentifier;
408         rct.originatorSignatureValue = si->signature;

410         os = ASN1_item_pack(&rct, ASN1_ITEM_rptr(CMS_Receipt), NULL);

412         err:
413         if (rr)
414                 CMS_ReceiptRequest_free(rr);

416         return os;

418         }
419 #endif /* ! codereview */
```

```
**********************************************************
    4662 Wed Aug 13 19:52:22 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_io.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_io.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/asn1t.h>
#include <openssl/x509.h>
#include <openssl/err.h>
#include <openssl/pem.h>
#include <openssl/cms.h>
#include <cms_lcl.h>

int CMS_stream(unsigned char ***boundary, CMS_ContentInfo *cms)
```



```
        {
        ASN1_OCTET_STRING **pos;
        pos = CMS_get0_content(cms);
        if (!pos)
                return 0;
        if (!*pos)
                *pos = ASN1_OCTET_STRING_new();
        if (*pos)
                {
                (*pos)->flags |= ASN1_STRING_FLAG_NDEF;
                (*pos)->flags &= ~ASN1_STRING_FLAG_CONT;
                *boundary = &(*pos)->data;
                return 1;
                }
        CMSerr(CMS_F_CMS_STREAM, ERR_R_MALLOC_FAILURE);
        return 0;
        }

CMS_ContentInfo *d2i_CMS_bio(BIO *bp, CMS_ContentInfo **cms)
        {
        return ASN1_item_d2i_bio(ASN1_ITEM_rptr(CMS_ContentInfo), bp, cms);
        }

int i2d_CMS_bio(BIO *bp, CMS_ContentInfo *cms)
        {
        return ASN1_item_i2d_bio(ASN1_ITEM_rptr(CMS_ContentInfo), bp, cms);
        }

IMPLEMENT_PEM_rw_const(CMS, CMS_ContentInfo, PEM_STRING_CMS, CMS_ContentInfo)

BIO *BIO_new_CMS(BIO *out, CMS_ContentInfo *cms)
        {
        return BIO_new_NDEF(out, (ASN1_VALUE *)cms,
                                ASN1_ITEM_rptr(CMS_ContentInfo));
        }

/* CMS wrappers round generalised stream and MIME routines */

int i2d_CMS_bio_stream(BIO *out, CMS_ContentInfo *cms, BIO *in, int flags)
        {
        return i2d_ASN1_bio_stream(out, (ASN1_VALUE *)cms, in, flags,
                                        ASN1_ITEM_rptr(CMS_ContentInfo));
        }

int PEM_write_bio_CMS_stream(BIO *out, CMS_ContentInfo *cms, BIO *in, int flags)
        {
        return PEM_write_bio_ASN1_stream(out, (ASN1_VALUE *) cms, in, flags,
                                        "CMS",
                                        ASN1_ITEM_rptr(CMS_ContentInfo));
        }

int SMIME_write_CMS(BIO *bio, CMS_ContentInfo *cms, BIO *data, int flags)
        {
        STACK_OF(X509_ALGOR) *mdalgs;
        int ctype_nid = OBJ_obj2nid(cms->contentType);
        int econt_nid = OBJ_obj2nid(CMS_get0_eContentType(cms));
        if (ctype_nid == NID_pkcs7_signed)
                mdalgs = cms->d.signedData->digestAlgorithms;
        else
                mdalgs = NULL;

        return SMIME_write_ASN1(bio, (ASN1_VALUE *)cms, data, flags,
                                        ctype_nid, econt_nid, mdalgs,
                                        ASN1_ITEM_rptr(CMS_ContentInfo));
        }
```

```
128 CMS_ContentInfo *SMIME_read_CMS(BIO *bio, BIO **bcont)
129         {
130         return (CMS_ContentInfo *)SMIME_read_ASN1(bio, bcont,
131                                         ASN1_ITEM_rptr(CMS_ContentInfo));
132         }
133 #endif /* ! codereview */
```

```
**********************************************************
   15325 Wed Aug 13 19:52:22 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_lib.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/asn1t.h>
#include <openssl/x509.h>
#include <openssl/err.h>
#include <openssl/pem.h>
#include <openssl/bio.h>
#include <openssl/asn1.h>
#include <openssl/cms.h>
#include <cms_lcl.h>
```



```
IMPLEMENT_ASN1_FUNCTIONS(CMS_ContentInfo)
IMPLEMENT_ASN1_PRINT_FUNCTION(CMS_ContentInfo)

DECLARE_ASN1_ITEM(CMS_CertificateChoices)
DECLARE_ASN1_ITEM(CMS_RevocationInfoChoice)
DECLARE_STACK_OF(CMS_CertificateChoices)
DECLARE_STACK_OF(CMS_RevocationInfoChoice)

const ASN1_OBJECT *CMS_get0_type(CMS_ContentInfo *cms)
        {
        return cms->contentType;
        }

CMS_ContentInfo *cms_Data_create(void)
        {
        CMS_ContentInfo *cms;
        cms = CMS_ContentInfo_new();
        if (cms)
                {
                cms->contentType = OBJ_nid2obj(NID_pkcs7_data);
                /* Never detached */
                CMS_set_detached(cms, 0);
                }
        return cms;
        }

BIO *cms_content_bio(CMS_ContentInfo *cms)
        {
        ASN1_OCTET_STRING **pos = CMS_get0_content(cms);
        if (!pos)
                return NULL;
        /* If content detached data goes nowhere: create NULL BIO */
        if (!*pos)
                return BIO_new(BIO_s_null());
        /* If content not detached and created return memory BIO
         */
        if (!*pos || ((*pos)->flags == ASN1_STRING_FLAG_CONT))
                return BIO_new(BIO_s_mem());
        /* Else content was read in: return read only BIO for it */
        return BIO_new_mem_buf((*pos)->data, (*pos)->length);
        }

BIO *CMS_dataInit(CMS_ContentInfo *cms, BIO *icont)
        {
        BIO *cmsbio, *cont;
        if (icont)
                cont = icont;
        else
                cont = cms_content_bio(cms);
        if (!cont)
                {
                CMSerr(CMS_F_CMS_DATAINIT, CMS_R_NO_CONTENT);
                return NULL;
                }
        switch (OBJ_obj2nid(cms->contentType))
                {

                case NID_pkcs7_data:
                return cont;

                case NID_pkcs7_signed:
                cmsbio = cms_SignedData_init_bio(cms);
                break;

                case NID_pkcs7_digest:
```

```
128                 cmsbio = cms_DigestedData_init_bio(cms);
129                 break;
130 #ifdef ZLIB
131                 case NID_id_smime_ct_compressedData:
132                 cmsbio = cms_CompressedData_init_bio(cms);
133                 break;
134 #endif

136                 case NID_pkcs7_encrypted:
137                 cmsbio = cms_EncryptedData_init_bio(cms);
138                 break;

140                 case NID_pkcs7_enveloped:
141                 cmsbio = cms_EnvelopedData_init_bio(cms);
142                 break;

144                 default:
145                 CMSerr(CMS_F_CMS_DATAINIT, CMS_R_UNSUPPORTED_TYPE);
146                 return NULL;
147                 }

149         if (cmsbio)
150                 return BIO_push(cmsbio, cont);

152         if (!icont)
153                 BIO_free(cont);
154         return NULL;

156         }

158 int CMS_dataFinal(CMS_ContentInfo *cms, BIO *cmsbio)
159         {
160         ASN1_OCTET_STRING **pos = CMS_get0_content(cms);
161         if (!pos)
162                 return 0;
163         /* If ebmedded content find memory BIO and set content */
164         if (*pos && ((*pos)->flags & ASN1_STRING_FLAG_CONT))
165                 {
166                 BIO *mbio;
167                 unsigned char *cont;
168                 long contlen;
169                 mbio = BIO_find_type(cmsbio, BIO_TYPE_MEM);
170                 if (!mbio)
171                         {
172                         CMSerr(CMS_F_CMS_DATAFINAL, CMS_R_CONTENT_NOT_FOUND);
173                         return 0;
174                         }
175                 contlen = BIO_get_mem_data(mbio, &cont);
176                 /* Set bio as read only so its content can't be clobbered */
177                 BIO_set_flags(mbio, BIO_FLAGS_MEM_RDONLY);
178                 BIO_set_mem_eof_return(mbio, 0);
179                 ASN1_STRING_set0(*pos, cont, contlen);
180                 (*pos)->flags &= ~ASN1_STRING_FLAG_CONT;
181                 }

183         switch (OBJ_obj2nid(cms->contentType))
184                 {

186                 case NID_pkcs7_data:
187                 case NID_pkcs7_enveloped:
188                 case NID_pkcs7_encrypted:
189                 case NID_id_smime_ct_compressedData:
190                 /* Nothing to do */
191                 return 1;

193                 case NID_pkcs7_signed:
```



```
194                 return cms_SignedData_final(cms, cmsbio);

196                 case NID_pkcs7_digest:
197                 return cms_DigestedData_do_final(cms, cmsbio, 0);

199                 default:
200                 CMSerr(CMS_F_CMS_DATAFINAL, CMS_R_UNSUPPORTED_TYPE);
201                 return 0;
202                 }
203         }

205 /* Return an OCTET STRING pointer to content. This allows it to
206  * be accessed or set later.
207  */

209 ASN1_OCTET_STRING **CMS_get0_content(CMS_ContentInfo *cms)
210         {
211         switch (OBJ_obj2nid(cms->contentType))
212                 {

214                 case NID_pkcs7_data:
215                 return &cms->d.data;

217                 case NID_pkcs7_signed:
218                 return &cms->d.signedData->encapContentInfo->eContent;

220                 case NID_pkcs7_enveloped:
221                 return &cms->d.envelopedData->encryptedContentInfo->encryptedCon

223                 case NID_pkcs7_digest:
224                 return &cms->d.digestedData->encapContentInfo->eContent;

226                 case NID_pkcs7_encrypted:
227                 return &cms->d.encryptedData->encryptedContentInfo->encryptedCon

229                 case NID_id_smime_ct_authData:
230                 return &cms->d.authenticatedData->encapContentInfo->eContent;

232                 case NID_id_smime_ct_compressedData:
233                 return &cms->d.compressedData->encapContentInfo->eContent;

235                 default:
236                 if (cms->d.other->type == V_ASN1_OCTET_STRING)
237                         return &cms->d.other->value.octet_string;
238                 CMSerr(CMS_F_CMS_GET0_CONTENT, CMS_R_UNSUPPORTED_CONTENT_TYPE);
239                 return NULL;

241                 }
242         }

244 /* Return an ASN1_OBJECT pointer to content type. This allows it to
245  * be accessed or set later.
246  */

248 static ASN1_OBJECT **cms_get0_econtent_type(CMS_ContentInfo *cms)
249         {
250         switch (OBJ_obj2nid(cms->contentType))
251                 {

253                 case NID_pkcs7_signed:
254                 return &cms->d.signedData->encapContentInfo->eContentType;

256                 case NID_pkcs7_enveloped:
257                 return &cms->d.envelopedData->encryptedContentInfo->contentType;

259                 case NID_pkcs7_digest:
```

```
260                 return &cms->d.digestedData->encapContentInfo->eContentType;

262                 case NID_pkcs7_encrypted:
263                 return &cms->d.encryptedData->encryptedContentInfo->contentType;

265                 case NID_id_smime_ct_authData:
266                 return &cms->d.authenticatedData->encapContentInfo->eContentType

268                 case NID_id_smime_ct_compressedData:
269                 return &cms->d.compressedData->encapContentInfo->eContentType;

271                 default:
272                 CMSerr(CMS_F_CMS_GET0_ECONTENT_TYPE,
273                                         CMS_R_UNSUPPORTED_CONTENT_TYPE);
274                 return NULL;

276                 }
277         }

279 const ASN1_OBJECT *CMS_get0_eContentType(CMS_ContentInfo *cms)
280         {
281         ASN1_OBJECT **petype;
282         petype = cms_get0_econtent_type(cms);
283         if (petype)
284                 return *petype;
285         return NULL;
286         }

288 int CMS_set1_eContentType(CMS_ContentInfo *cms, const ASN1_OBJECT *oid)
289         {
290         ASN1_OBJECT **petype, *etype;
291         petype = cms_get0_econtent_type(cms);
292         if (!petype)
293                 return 0;
294         if (!oid)
295                 return 1;
296         etype = OBJ_dup(oid);
297         if (!etype)
298                 return 0;
299         ASN1_OBJECT_free(*petype);
300         *petype = etype;
301         return 1;
302         }

304 int CMS_is_detached(CMS_ContentInfo *cms)
305         {
306         ASN1_OCTET_STRING **pos;
307         pos = CMS_get0_content(cms);
308         if (!pos)
309                 return -1;
310         if (*pos)
311                 return 0;
312         return 1;
313         }

315 int CMS_set_detached(CMS_ContentInfo *cms, int detached)
316         {
317         ASN1_OCTET_STRING **pos;
318         pos = CMS_get0_content(cms);
319         if (!pos)
320                 return 0;
321         if (detached)
322                 {
323                 if (*pos)
324                         {
325                         ASN1_OCTET_STRING_free(*pos);
```



```
326                         *pos = NULL;
327                         }
328                 return 1;
329                 }
330         if (!*pos)
331                 *pos = ASN1_OCTET_STRING_new();
332         if (*pos)
333                 {
334                 /* NB: special flag to show content is created and not
335                  * read in.
336                  */
337                 (*pos)->flags |= ASN1_STRING_FLAG_CONT;
338                 return 1;
339                 }
340         CMSerr(CMS_F_CMS_SET_DETACHED, ERR_R_MALLOC_FAILURE);
341         return 0;
342         }

344 /* Set up an X509_ALGOR DigestAlgorithmIdentifier from an EVP_MD */

346 void cms_DigestAlgorithm_set(X509_ALGOR *alg, const EVP_MD *md)
347         {
348         int param_type;

350         if (md->flags & EVP_MD_FLAG_DIGALGID_ABSENT)
351                 param_type = V_ASN1_UNDEF;
352         else
353                 param_type = V_ASN1_NULL;

355         X509_ALGOR_set0(alg, OBJ_nid2obj(EVP_MD_type(md)), param_type, NULL);

357         }

359 /* Create a digest BIO from an X509_ALGOR structure */

361 BIO *cms_DigestAlgorithm_init_bio(X509_ALGOR *digestAlgorithm)
362         {
363         BIO *mdbio = NULL;
364         ASN1_OBJECT *digestoid;
365         const EVP_MD *digest;
366         X509_ALGOR_get0(&digestoid, NULL, NULL, digestAlgorithm);
367         digest = EVP_get_digestbyobj(digestoid);
368         if (!digest)
369                 {
370                 CMSerr(CMS_F_CMS_DIGESTALGORITHM_INIT_BIO,
371                                 CMS_R_UNKNOWN_DIGEST_ALGORIHM);
372                 goto err;
373                 }
374         mdbio = BIO_new(BIO_f_md());
375         if (!mdbio || !BIO_set_md(mdbio, digest))
376                 {
377                 CMSerr(CMS_F_CMS_DIGESTALGORITHM_INIT_BIO,
378                                 CMS_R_MD_BIO_INIT_ERROR);
379                 goto err;
380                 }
381         return mdbio;
382         err:
383         if (mdbio)
384                 BIO_free(mdbio);
385         return NULL;
386         }

388 /* Locate a message digest content from a BIO chain based on SignerInfo */

390 int cms_DigestAlgorithm_find_ctx(EVP_MD_CTX *mctx, BIO *chain,
391                                         X509_ALGOR *mdalg)
```

```
392         {
393         int nid;
394         ASN1_OBJECT *mdoid;
395         X509_ALGOR_get0(&mdoid, NULL, NULL, mdalg);
396         nid = OBJ_obj2nid(mdoid);
397         /* Look for digest type to match signature */
398         for (;;)
399                 {
400                 EVP_MD_CTX *mtmp;
401                 chain = BIO_find_type(chain, BIO_TYPE_MD);
402                 if (chain == NULL)
403                         {
404                         CMSerr(CMS_F_CMS_DIGESTALGORITHM_FIND_CTX,
405                                                 CMS_R_NO_MATCHING_DIGEST);
406                         return 0;
407                         }
408                 BIO_get_md_ctx(chain, &mtmp);
409                 if (EVP_MD_CTX_type(mtmp) == nid
410                 /* Workaround for broken implementations that use signature
411                  * algorithm  OID instead of digest.
412                  */
413                         || EVP_MD_pkey_type(EVP_MD_CTX_md(mtmp)) == nid)
414                         return EVP_MD_CTX_copy_ex(mctx, mtmp);
415                 chain = BIO_next(chain);
416                 }
417         }

419 static STACK_OF(CMS_CertificateChoices) **cms_get0_certificate_choices(CMS_Conte
420         {
421         switch (OBJ_obj2nid(cms->contentType))
422                 {

424                 case NID_pkcs7_signed:
425                 return &cms->d.signedData->certificates;

427                 case NID_pkcs7_enveloped:
428                 return &cms->d.envelopedData->originatorInfo->certificates;

430                 default:
431                 CMSerr(CMS_F_CMS_GET0_CERTIFICATE_CHOICES,
432                                         CMS_R_UNSUPPORTED_CONTENT_TYPE);
433                 return NULL;

435                 }
436         }

438 CMS_CertificateChoices *CMS_add0_CertificateChoices(CMS_ContentInfo *cms)
439         {
440         STACK_OF(CMS_CertificateChoices) **pcerts;
441         CMS_CertificateChoices *cch;
442         pcerts = cms_get0_certificate_choices(cms);
443         if (!pcerts)
444                 return NULL;
445         if (!*pcerts)
446                 *pcerts = sk_CMS_CertificateChoices_new_null();
447         if (!*pcerts)
448                 return NULL;
449         cch = M_ASN1_new_of(CMS_CertificateChoices);
450         if (!cch)
451                 return NULL;
452         if (!sk_CMS_CertificateChoices_push(*pcerts, cch))
453                 {
454                 M_ASN1_free_of(cch, CMS_CertificateChoices);
455                 return NULL;
456                 }
457         return cch;
```



```
458         }

460 int CMS_add0_cert(CMS_ContentInfo *cms, X509 *cert)
461         {
462         CMS_CertificateChoices *cch;
463         STACK_OF(CMS_CertificateChoices) **pcerts;
464         int i;
465         pcerts = cms_get0_certificate_choices(cms);
466         if (!pcerts)
467                 return 0;
468         for (i = 0; i < sk_CMS_CertificateChoices_num(*pcerts); i++)
469                 {
470                 cch = sk_CMS_CertificateChoices_value(*pcerts, i);
471                 if (cch->type == CMS_CERTCHOICE_CERT)
472                         {
473                         if (!X509_cmp(cch->d.certificate, cert))
474                                 {
475                                 CMSerr(CMS_F_CMS_ADD0_CERT,
476                                         CMS_R_CERTIFICATE_ALREADY_PRESENT);
477                                 return 0;
478                                 }
479                         }
480                 }
481         cch = CMS_add0_CertificateChoices(cms);
482         if (!cch)
483                 return 0;
484         cch->type = CMS_CERTCHOICE_CERT;
485         cch->d.certificate = cert;
486         return 1;
487         }

489 int CMS_add1_cert(CMS_ContentInfo *cms, X509 *cert)
490         {
491         int r;
492         r = CMS_add0_cert(cms, cert);
493         if (r > 0)
494                 CRYPTO_add(&cert->references, 1, CRYPTO_LOCK_X509);
495         return r;
496         }

498 static STACK_OF(CMS_RevocationInfoChoice) **cms_get0_revocation_choices(CMS_Cont
499         {
500         switch (OBJ_obj2nid(cms->contentType))
501                 {

503                 case NID_pkcs7_signed:
504                 return &cms->d.signedData->crls;

506                 case NID_pkcs7_enveloped:
507                 return &cms->d.envelopedData->originatorInfo->crls;

509                 default:
510                 CMSerr(CMS_F_CMS_GET0_REVOCATION_CHOICES,
511                                         CMS_R_UNSUPPORTED_CONTENT_TYPE);
512                 return NULL;

514                 }
515         }

517 CMS_RevocationInfoChoice *CMS_add0_RevocationInfoChoice(CMS_ContentInfo *cms)
518         {
519         STACK_OF(CMS_RevocationInfoChoice) **pcrls;
520         CMS_RevocationInfoChoice *rch;
521         pcrls = cms_get0_revocation_choices(cms);
522         if (!pcrls)
523                 return NULL;
```

```
524         if (!*pcrls)
525                 *pcrls = sk_CMS_RevocationInfoChoice_new_null();
526         if (!*pcrls)
527                 return NULL;
528         rch = M_ASN1_new_of(CMS_RevocationInfoChoice);
529         if (!rch)
530                 return NULL;
531         if (!sk_CMS_RevocationInfoChoice_push(*pcrls, rch))
532                 {
533                 M_ASN1_free_of(rch, CMS_RevocationInfoChoice);
534                 return NULL;
535                 }
536         return rch;
537         }

539 int CMS_add0_crl(CMS_ContentInfo *cms, X509_CRL *crl)
540         {
541         CMS_RevocationInfoChoice *rch;
542         rch = CMS_add0_RevocationInfoChoice(cms);
543         if (!rch)
544                 return 0;
545         rch->type = CMS_REVCHOICE_CRL;
546         rch->d.crl = crl;
547         return 1;
548         }

550 int CMS_add1_crl(CMS_ContentInfo *cms, X509_CRL *crl)
551         {
552         int r;
553         r = CMS_add0_crl(cms, crl);
554         if (r > 0)
555                 CRYPTO_add(&crl->references, 1, CRYPTO_LOCK_X509_CRL);
556         return r;
557         }

559 STACK_OF(X509) *CMS_get1_certs(CMS_ContentInfo *cms)
560         {
561         STACK_OF(X509) *certs = NULL;
562         CMS_CertificateChoices *cch;
563         STACK_OF(CMS_CertificateChoices) **pcerts;
564         int i;
565         pcerts = cms_get0_certificate_choices(cms);
566         if (!pcerts)
567                 return NULL;
568         for (i = 0; i < sk_CMS_CertificateChoices_num(*pcerts); i++)
569                 {
570                 cch = sk_CMS_CertificateChoices_value(*pcerts, i);
571                 if (cch->type == 0)
572                         {
573                         if (!certs)
574                                 {
575                                 certs = sk_X509_new_null();
576                                 if (!certs)
577                                         return NULL;
578                                 }
579                         if (!sk_X509_push(certs, cch->d.certificate))
580                                 {
581                                 sk_X509_pop_free(certs, X509_free);
582                                 return NULL;
583                                 }
584                         CRYPTO_add(&cch->d.certificate->references,
585                                                         1, CRYPTO_LOCK_X509);
586                         }
587                 }
588         return certs;
```



```
590         }

592 STACK_OF(X509_CRL) *CMS_get1_crls(CMS_ContentInfo *cms)
593         {
594         STACK_OF(X509_CRL) *crls = NULL;
595         STACK_OF(CMS_RevocationInfoChoice) **pcrls;
596         CMS_RevocationInfoChoice *rch;
597         int i;
598         pcrls = cms_get0_revocation_choices(cms);
599         if (!pcrls)
600                 return NULL;
601         for (i = 0; i < sk_CMS_RevocationInfoChoice_num(*pcrls); i++)
602                 {
603                 rch = sk_CMS_RevocationInfoChoice_value(*pcrls, i);
604                 if (rch->type == 0)
605                         {
606                         if (!crls)
607                                 {
608                                 crls = sk_X509_CRL_new_null();
609                                 if (!crls)
610                                         return NULL;
611                                 }
612                         if (!sk_X509_CRL_push(crls, rch->d.crl))
613                                 {
614                                 sk_X509_CRL_pop_free(crls, X509_CRL_free);
615                                 return NULL;
616                                 }
617                         CRYPTO_add(&rch->d.crl->references,
618                                                         1, CRYPTO_LOCK_X509_CRL);
619                         }
620                 }
621         return crls;
622         }
623 #endif /* ! codereview */
```

```
**********************************************************
   11946 Wed Aug 13 19:52:23 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_pwri.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_pwri.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2009 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include <openssl/rand.h>
#include <openssl/aes.h>
```



```
#include "cms_lcl.h"
#include "asn1_locl.h"

int CMS_RecipientInfo_set0_password(CMS_RecipientInfo *ri,
                                    unsigned char *pass, ossl_ssize_t passlen)
        {
        CMS_PasswordRecipientInfo *pwri;
        if (ri->type != CMS_RECIPINFO_PASS)
                {
                CMSerr(CMS_F_CMS_RECIPIENTINFO_SET0_PASSWORD, CMS_R_NOT_PWRI);
                return 0;
                }

        pwri = ri->d.pwri;
        pwri->pass = pass;
        if (pass && passlen < 0)
                passlen = strlen((char *)pass);
        pwri->passlen = passlen;
        return 1;
        }

CMS_RecipientInfo *CMS_add0_recipient_password(CMS_ContentInfo *cms,
                                               int iter, int wrap_nid, int pbe_nid,
                                               unsigned char *pass,
                                               ossl_ssize_t passlen,
                                               const EVP_CIPHER *kekciph)
        {
        CMS_RecipientInfo *ri = NULL;
        CMS_EnvelopedData *env;
        CMS_PasswordRecipientInfo *pwri;
        EVP_CIPHER_CTX ctx;
        X509_ALGOR *encalg = NULL;
        unsigned char iv[EVP_MAX_IV_LENGTH];
        int ivlen;

        env = cms_get0_enveloped(cms);
        if (!env)
                return NULL;

        if (wrap_nid <= 0)
                wrap_nid = NID_id_alg_PWRI_KEK;

        if (pbe_nid <= 0)
                pbe_nid = NID_id_pbkdf2;

        /* Get from enveloped data */
        if (kekciph == NULL)
                kekciph = env->encryptedContentInfo->cipher;

        if (kekciph == NULL)
                {
                CMSerr(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD, CMS_R_NO_CIPHER);
                return NULL;
                }
        if (wrap_nid != NID_id_alg_PWRI_KEK)
                {
                CMSerr(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD,
                                CMS_R_UNSUPPORTED_KEY_ENCRYPTION_ALGORITHM);
                return NULL;
                }

        /* Setup algorithm identifier for cipher */
        encalg = X509_ALGOR_new();
        EVP_CIPHER_CTX_init(&ctx);

        if (EVP_EncryptInit_ex(&ctx, kekciph, NULL, NULL, NULL) <= 0)
```

```
128                 {
129                 CMSerr(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD, ERR_R_EVP_LIB);
130                 goto err;
131                 }

133         ivlen = EVP_CIPHER_CTX_iv_length(&ctx);

135         if (ivlen > 0)
136                 {
137                 if (RAND_pseudo_bytes(iv, ivlen) <= 0)
138                         goto err;
139                 if (EVP_EncryptInit_ex(&ctx, NULL, NULL, NULL, iv) <= 0)
140                         {
141                         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD,
142                                                         ERR_R_EVP_LIB);
143                         goto err;
144                         }
145                 encalg->parameter = ASN1_TYPE_new();
146                 if (!encalg->parameter)
147                         {
148                         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD,
149                                                         ERR_R_MALLOC_FAILURE);
150                         goto err;
151                         }
152                 if (EVP_CIPHER_param_to_asn1(&ctx, encalg->parameter) <= 0)
153                         {
154                         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD,
155                                 CMS_R_CIPHER_PARAMETER_INITIALISATION_ERROR);
156                         goto err;
157                         }
158                 }


161         encalg->algorithm = OBJ_nid2obj(EVP_CIPHER_CTX_type(&ctx));

163         EVP_CIPHER_CTX_cleanup(&ctx);

165         /* Initialize recipient info */
166         ri = M_ASN1_new_of(CMS_RecipientInfo);
167         if (!ri)
168                 goto merr;

170         ri->d.pwri = M_ASN1_new_of(CMS_PasswordRecipientInfo);
171         if (!ri->d.pwri)
172                 goto merr;
173         ri->type = CMS_RECIPINFO_PASS;

175         pwri = ri->d.pwri;
176         /* Since this is overwritten, free up empty structure already there */
177         X509_ALGOR_free(pwri->keyEncryptionAlgorithm);
178         pwri->keyEncryptionAlgorithm = X509_ALGOR_new();
179         if (!pwri->keyEncryptionAlgorithm)
180                 goto merr;
181         pwri->keyEncryptionAlgorithm->algorithm = OBJ_nid2obj(wrap_nid);
182         pwri->keyEncryptionAlgorithm->parameter = ASN1_TYPE_new();
183         if (!pwri->keyEncryptionAlgorithm->parameter)
184                 goto merr;

186         if(!ASN1_item_pack(encalg, ASN1_ITEM_rptr(X509_ALGOR),
187             &pwri->keyEncryptionAlgorithm->parameter->value.sequence))
188                 goto merr;
189         pwri->keyEncryptionAlgorithm->parameter->type = V_ASN1_SEQUENCE;

191         X509_ALGOR_free(encalg);
192         encalg = NULL;
```



```
194         /* Setup PBE algorithm */

196         pwri->keyDerivationAlgorithm = PKCS5_pbkdf2_set(iter, NULL, 0, -1, -1);

198         if (!pwri->keyDerivationAlgorithm)
199                 goto err;

201         CMS_RecipientInfo_set0_password(ri, pass, passlen);
202         pwri->version = 0;

204         if (!sk_CMS_RecipientInfo_push(env->recipientInfos, ri))
205                 goto merr;

207         return ri;

209         merr:
210         CMSerr(CMS_F_CMS_ADD0_RECIPIENT_PASSWORD, ERR_R_MALLOC_FAILURE);
211         err:
212         EVP_CIPHER_CTX_cleanup(&ctx);
213         if (ri)
214                 M_ASN1_free_of(ri, CMS_RecipientInfo);
215         if (encalg)
216                 X509_ALGOR_free(encalg);
217         return NULL;

219         }

221 /* This is an implementation of the key wrapping mechanism in RFC3211,
222  * at some point this should go into EVP.
223  */

225 static int kek_unwrap_key(unsigned char *out, size_t *outlen,
226                 const unsigned char *in, size_t inlen, EVP_CIPHER_CTX *ctx)
227         {
228         size_t blocklen = EVP_CIPHER_CTX_block_size(ctx);
229         unsigned char *tmp;
230         int outl, rv = 0;
231         if (inlen < 2 * blocklen)
232                 {
233                 /* too small */
234                 return 0;
235                 }
236         if (inlen % blocklen)
237                 {
238                 /* Invalid size */
239                 return 0;
240                 }
241         tmp = OPENSSL_malloc(inlen);
242         /* setup IV by decrypting last two blocks */
243         EVP_DecryptUpdate(ctx, tmp + inlen - 2 * blocklen, &outl,
244                                 in  + inlen - 2 * blocklen, blocklen * 2);
245         /* Do a decrypt of last decrypted block to set IV to correct value
246          * output it to start of buffer so we don't corrupt decrypted block
247          * this works because buffer is at least two block lengths long.
248          */
249         EVP_DecryptUpdate(ctx, tmp, &outl,
250                                 tmp  + inlen - blocklen, blocklen);
251         /* Can now decrypt first n - 1 blocks */
252         EVP_DecryptUpdate(ctx, tmp, &outl, in, inlen - blocklen);

254         /* Reset IV to original value */
255         EVP_DecryptInit_ex(ctx, NULL, NULL, NULL, NULL);
256         /* Decrypt again */
257         EVP_DecryptUpdate(ctx, tmp, &outl, tmp, inlen);
258         /* Check check bytes */
259         if (((tmp[1] ^ tmp[4]) & (tmp[2] ^ tmp[5]) & (tmp[3] ^ tmp[6])) != 0xff)
```

```
260                 {
261                 /* Check byte failure */
262                 goto err;
263                 }
264         if (inlen < (size_t)(tmp[0] - 4 ))
265                 {
266                 /* Invalid length value */
267                 goto err;
268                 }
269         *outlen = (size_t)tmp[0];
270         memcpy(out, tmp + 4, *outlen);
271         rv = 1;
272         err:
273         OPENSSL_cleanse(tmp, inlen);
274         OPENSSL_free(tmp);
275         return rv;

277         }

279 static int kek_wrap_key(unsigned char *out, size_t *outlen,
280                 const unsigned char *in, size_t inlen, EVP_CIPHER_CTX *ctx)
281         {
282         size_t blocklen = EVP_CIPHER_CTX_block_size(ctx);
283         size_t olen;
284         int dummy;
285         /* First decide length of output buffer: need header and round up to
286          * multiple of block length.
287          */
288         olen = (inlen + 4 + blocklen - 1)/blocklen;
289         olen *= blocklen;
290         if (olen < 2 * blocklen)
291                 {
292                 /* Key too small */
293                 return 0;
294                 }
295         if (inlen > 0xFF)
296                 {
297                 /* Key too large */
298                 return 0;
299                 }
300         if (out)
301                 {
302                 /* Set header */
303                 out[0] = (unsigned char)inlen;
304                 out[1] = in[0] ^ 0xFF;
305                 out[2] = in[1] ^ 0xFF;
306                 out[3] = in[2] ^ 0xFF;
307                 memcpy(out + 4, in, inlen);
308                 /* Add random padding to end */
309                 if (olen > inlen + 4)
310                         RAND_pseudo_bytes(out + 4 + inlen, olen - 4 - inlen);
311                 /* Encrypt twice */
312                 EVP_EncryptUpdate(ctx, out, &dummy, out, olen);
313                 EVP_EncryptUpdate(ctx, out, &dummy, out, olen);
314                 }

316         *outlen = olen;

318         return 1;
319         }

321 /* Encrypt/Decrypt content key in PWRI recipient info */

323 int cms_RecipientInfo_pwri_crypt(CMS_ContentInfo *cms, CMS_RecipientInfo *ri,
324                                                         int en_de)
325         {
```



```
326         CMS_EncryptedContentInfo *ec;
327         CMS_PasswordRecipientInfo *pwri;
328         const unsigned char *p = NULL;
329         int plen;
330         int r = 0;
331         X509_ALGOR *algtmp, *kekalg = NULL;
332         EVP_CIPHER_CTX kekctx;
333         const EVP_CIPHER *kekcipher;
334         unsigned char *key = NULL;
335         size_t keylen;

337         ec = cms->d.envelopedData->encryptedContentInfo;

339         pwri = ri->d.pwri;
340         EVP_CIPHER_CTX_init(&kekctx);

342         if (!pwri->pass)
343                 {
344                 CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT, CMS_R_NO_PASSWORD);
345                 return 0;
346                 }
347         algtmp = pwri->keyEncryptionAlgorithm;

349         if (!algtmp || OBJ_obj2nid(algtmp->algorithm) != NID_id_alg_PWRI_KEK)
350                 {
351                 CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT,
352                                 CMS_R_UNSUPPORTED_KEY_ENCRYPTION_ALGORITHM);
353                 return 0;
354                 }

356         if (algtmp->parameter->type == V_ASN1_SEQUENCE)
357                 {
358                 p = algtmp->parameter->value.sequence->data;
359                 plen = algtmp->parameter->value.sequence->length;
360                 kekalg = d2i_X509_ALGOR(NULL, &p, plen);
361                 }
362         if (kekalg == NULL)
363                 {
364                 CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT,
365                                 CMS_R_INVALID_KEY_ENCRYPTION_PARAMETER);
366                 return 0;
367                 }

369         kekcipher = EVP_get_cipherbyobj(kekalg->algorithm);

371         if(!kekcipher)
372                 {
373                 CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT,
374                                 CMS_R_UNKNOWN_CIPHER);
375                 goto err;
376                 }

378         /* Fixup cipher based on AlgorithmIdentifier to set IV etc */
379         if (!EVP_CipherInit_ex(&kekctx, kekcipher, NULL, NULL, NULL, en_de))
380                 goto err;
381         EVP_CIPHER_CTX_set_padding(&kekctx, 0);
382         if(EVP_CIPHER_asn1_to_param(&kekctx, kekalg->parameter) < 0)
383                 {
384                 CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT,
385                                 CMS_R_CIPHER_PARAMETER_INITIALISATION_ERROR);
386                 goto err;
387                 }

389         algtmp = pwri->keyDerivationAlgorithm;

391         /* Finish password based key derivation to setup key in "ctx" */
```

```
393         if (EVP_PBE_CipherInit(algtmp->algorithm,
394                                (char *)pwri->pass, pwri->passlen,
395                                algtmp->parameter, &kekctx, en_de) < 0)
396                 {
397                 CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT, ERR_R_EVP_LIB);
398                 goto err;
399                 }

401         /* Finally wrap/unwrap the key */

403         if (en_de)
404                 {

406                 if (!kek_wrap_key(NULL, &keylen, ec->key, ec->keylen, &kekctx))
407                         goto err;

409                 key = OPENSSL_malloc(keylen);

411                 if (!key)
412                         goto err;

414                 if (!kek_wrap_key(key, &keylen, ec->key, ec->keylen, &kekctx))
415                         goto err;
416                 pwri->encryptedKey->data = key;
417                 pwri->encryptedKey->length = keylen;
418                 }
419         else
420                 {
421                 key = OPENSSL_malloc(pwri->encryptedKey->length);

423                 if (!key)
424                         {
425                         CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT,
426                                                         ERR_R_MALLOC_FAILURE);
427                         goto err;
428                         }
429                 if (!kek_unwrap_key(key, &keylen,
430                                         pwri->encryptedKey->data,
431                                         pwri->encryptedKey->length, &kekctx))
432                         {
433                         CMSerr(CMS_F_CMS_RECIPIENTINFO_PWRI_CRYPT,
434                                                         CMS_R_UNWRAP_FAILURE);
435                         goto err;
436                         }

438                 ec->key = key;
439                 ec->keylen = keylen;

441                 }

443         r = 1;

445         err:

447         EVP_CIPHER_CTX_cleanup(&kekctx);

449         if (!r && key)
450                 OPENSSL_free(key);
451         X509_ALGOR_free(kekalg);

453         return r;

455         }
456 #endif /* ! codereview */
```

```
**********************************************************
   23155 Wed Aug 13 19:52:23 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_sd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_sd.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include "cms_lcl.h"
#include "asn1_locl.h"
```



```
/* CMS SignedData Utilities */

DECLARE_ASN1_ITEM(CMS_SignedData)

static CMS_SignedData *cms_get0_signed(CMS_ContentInfo *cms)
        {
        if (OBJ_obj2nid(cms->contentType) != NID_pkcs7_signed)
                {
                CMSerr(CMS_F_CMS_GET0_SIGNED, CMS_R_CONTENT_TYPE_NOT_SIGNED_DATA
                return NULL;
                }
        return cms->d.signedData;
        }

static CMS_SignedData *cms_signed_data_init(CMS_ContentInfo *cms)
        {
        if (cms->d.other == NULL)
                {
                cms->d.signedData = M_ASN1_new_of(CMS_SignedData);
                if (!cms->d.signedData)
                        {
                        CMSerr(CMS_F_CMS_SIGNED_DATA_INIT, ERR_R_MALLOC_FAILURE)
                        return NULL;
                        }
                cms->d.signedData->version = 1;
                cms->d.signedData->encapContentInfo->eContentType =
                                                        OBJ_nid2obj(NID_pkcs7_data);
                cms->d.signedData->encapContentInfo->partial = 1;
                ASN1_OBJECT_free(cms->contentType);
                cms->contentType = OBJ_nid2obj(NID_pkcs7_signed);
                return cms->d.signedData;
                }
        return cms_get0_signed(cms);
        }

/* Just initialize SignedData e.g. for certs only structure */

int CMS_SignedData_init(CMS_ContentInfo *cms)
        {
        if (cms_signed_data_init(cms))
                return 1;
        else
                return 0;
        }

/* Check structures and fixup version numbers (if necessary) */

static void cms_sd_set_version(CMS_SignedData *sd)
        {
        int i;
        CMS_CertificateChoices *cch;
        CMS_RevocationInfoChoice *rch;
        CMS_SignerInfo *si;

        for (i = 0; i < sk_CMS_CertificateChoices_num(sd->certificates); i++)
                {
                cch = sk_CMS_CertificateChoices_value(sd->certificates, i);
                if (cch->type == CMS_CERTCHOICE_OTHER)
                        {
                        if (sd->version < 5)
                                sd->version = 5;
                        }
                else if (cch->type == CMS_CERTCHOICE_V2ACERT)
                        {
                        if (sd->version < 4)
```

```
128                                 sd->version = 4;
129                         }
130                 else if (cch->type == CMS_CERTCHOICE_V1ACERT)
131                         {
132                         if (sd->version < 3)
133                                 sd->version = 3;
134                         }
135                 }
136
137         for (i = 0; i < sk_CMS_RevocationInfoChoice_num(sd->crls); i++)
138                 {
139                 rch = sk_CMS_RevocationInfoChoice_value(sd->crls, i);
140                 if (rch->type == CMS_REVCHOICE_OTHER)
141                         {
142                         if (sd->version < 5)
143                                 sd->version = 5;
144                         }
145                 }
146
147         if ((OBJ_obj2nid(sd->encapContentInfo->eContentType) != NID_pkcs7_data)
148                         && (sd->version < 3))
149                 sd->version = 3;
150
151         for (i = 0; i < sk_CMS_SignerInfo_num(sd->signerInfos); i++)
152                 {
153                 si = sk_CMS_SignerInfo_value(sd->signerInfos, i);
154                 if (si->sid->type == CMS_SIGNERINFO_KEYIDENTIFIER)
155                         {
156                         if (si->version < 3)
157                                 si->version = 3;
158                         if (sd->version < 3)
159                                 sd->version = 3;
160                         }
161                 else if (si->version < 1)
162                         si->version = 1;
163                 }
164
165         if (sd->version < 1)
166                 sd->version = 1;
167
168         }
169
170 /* Copy an existing messageDigest value */
171
172 static int cms_copy_messageDigest(CMS_ContentInfo *cms, CMS_SignerInfo *si)
173         {
174         STACK_OF(CMS_SignerInfo) *sinfos;
175         CMS_SignerInfo *sitmp;
176         int i;
177         sinfos = CMS_get0_SignerInfos(cms);
178         for (i = 0; i < sk_CMS_SignerInfo_num(sinfos); i++)
179                 {
180                 ASN1_OCTET_STRING *messageDigest;
181                 sitmp = sk_CMS_SignerInfo_value(sinfos, i);
182                 if (sitmp == si)
183                         continue;
184                 if (CMS_signed_get_attr_count(sitmp) < 0)
185                         continue;
186                 if (OBJ_cmp(si->digestAlgorithm->algorithm,
187                                 sitmp->digestAlgorithm->algorithm))
188                         continue;
189                 messageDigest = CMS_signed_get0_data_by_OBJ(sitmp,
190                                         OBJ_nid2obj(NID_pkcs9_messageDigest),
191                                         -3, V_ASN1_OCTET_STRING);
192                 if (!messageDigest)
193                         {
```



```
194                         CMSerr(CMS_F_CMS_COPY_MESSAGEDIGEST,
195                                 CMS_R_ERROR_READING_MESSAGEDIGEST_ATTRIBUTE);
196                         return 0;
197                         }
198
199                 if (CMS_signed_add1_attr_by_NID(si, NID_pkcs9_messageDigest,
200                                                 V_ASN1_OCTET_STRING,
201                                                 messageDigest, -1))
202                         return 1;
203                 else
204                         return 0;
205                 }
206                 CMSerr(CMS_F_CMS_COPY_MESSAGEDIGEST, CMS_R_NO_MATCHING_DIGEST);
207                 return 0;
208         }
209
210 int cms_set1_SignerIdentifier(CMS_SignerIdentifier *sid, X509 *cert, int type)
211         {
212         switch(type)
213                 {
214                 case CMS_SIGNERINFO_ISSUER_SERIAL:
215                 sid->d.issuerAndSerialNumber =
216                         M_ASN1_new_of(CMS_IssuerAndSerialNumber);
217                 if (!sid->d.issuerAndSerialNumber)
218                         goto merr;
219                 if (!X509_NAME_set(&sid->d.issuerAndSerialNumber->issuer,
220                                         X509_get_issuer_name(cert)))
221                         goto merr;
222                 if (!ASN1_STRING_copy(
223                         sid->d.issuerAndSerialNumber->serialNumber,
224                                 X509_get_serialNumber(cert)))
225                         goto merr;
226                 break;
227
228                 case CMS_SIGNERINFO_KEYIDENTIFIER:
229                 if (!cert->skid)
230                         {
231                         CMSerr(CMS_F_CMS_SET1_SIGNERIDENTIFIER,
232                                         CMS_R_CERTIFICATE_HAS_NO_KEYID);
233                         return 0;
234                         }
235                 sid->d.subjectKeyIdentifier = ASN1_STRING_dup(cert->skid);
236                 if (!sid->d.subjectKeyIdentifier)
237                         goto merr;
238                 break;
239
240                 default:
241                 CMSerr(CMS_F_CMS_SET1_SIGNERIDENTIFIER, CMS_R_UNKNOWN_ID);
242                 return 0;
243                 }
244
245         sid->type = type;
246
247         return 1;
248
249         merr:
250         CMSerr(CMS_F_CMS_SET1_SIGNERIDENTIFIER, ERR_R_MALLOC_FAILURE);
251         return 0;
252
253         }
254
255 int cms_SignerIdentifier_get0_signer_id(CMS_SignerIdentifier *sid,
256                                         ASN1_OCTET_STRING **keyid,
257                                         X509_NAME **issuer, ASN1_INTEGER **sno)
258         {
259         if (sid->type == CMS_SIGNERINFO_ISSUER_SERIAL)
```

```
260                 {
261                 if (issuer)
262                         *issuer = sid->d.issuerAndSerialNumber->issuer;
263                 if (sno)
264                         *sno = sid->d.issuerAndSerialNumber->serialNumber;
265                 }
266         else if (sid->type == CMS_SIGNERINFO_KEYIDENTIFIER)
267                 {
268                 if (keyid)
269                         *keyid = sid->d.subjectKeyIdentifier;
270                 }
271         else
272                 return 0;
273         return 1;
274         }

276 int cms_SignerIdentifier_cert_cmp(CMS_SignerIdentifier *sid, X509 *cert)
277         {
278         int ret;
279         if (sid->type == CMS_SIGNERINFO_ISSUER_SERIAL)
280                 {
281                 ret = X509_NAME_cmp(sid->d.issuerAndSerialNumber->issuer,
282                                     X509_get_issuer_name(cert));
283                 if (ret)
284                         return ret;
285                 return ASN1_INTEGER_cmp(sid->d.issuerAndSerialNumber->serialNumb
286                                     X509_get_serialNumber(cert));
287                 }
288         else if (sid->type == CMS_SIGNERINFO_KEYIDENTIFIER)
289                 {
290                 X509_check_purpose(cert, -1, -1);
291                 if (!cert->skid)
292                         return -1;
293                 return ASN1_OCTET_STRING_cmp(sid->d.subjectKeyIdentifier,
294                                                 cert->skid);
295                 }
296         else
297                 return -1;
298         }

300 CMS_SignerInfo *CMS_add1_signer(CMS_ContentInfo *cms,
301                         X509 *signer, EVP_PKEY *pk, const EVP_MD *md,
302                         unsigned int flags)
303         {
304         CMS_SignedData *sd;
305         CMS_SignerInfo *si = NULL;
306         X509_ALGOR *alg;
307         int i, type;
308         if(!X509_check_private_key(signer, pk))
309                 {
310                 CMSerr(CMS_F_CMS_ADD1_SIGNER,
311                         CMS_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE);
312                 return NULL;
313                 }
314         sd = cms_signed_data_init(cms);
315         if (!sd)
316                 goto err;
317         si = M_ASN1_new_of(CMS_SignerInfo);
318         if (!si)
319                 goto merr;
320         X509_check_purpose(signer, -1, -1);

322         CRYPTO_add(&pk->references, 1, CRYPTO_LOCK_EVP_PKEY);
323         CRYPTO_add(&signer->references, 1, CRYPTO_LOCK_X509);

325         si->pkey = pk;
```



```
326         si->signer = signer;

328         if (flags & CMS_USE_KEYID)
329                 {
330                 si->version = 3;
331                 if (sd->version < 3)
332                         sd->version = 3;
333                 type = CMS_SIGNERINFO_KEYIDENTIFIER;
334                 }
335         else
336                 {
337                 type = CMS_SIGNERINFO_ISSUER_SERIAL;
338                 si->version = 1;
339                 }

341         if (!cms_set1_SignerIdentifier(si->sid, signer, type))
342                 goto err;

344         if (md == NULL)
345                 {
346                 int def_nid;
347                 if (EVP_PKEY_get_default_digest_nid(pk, &def_nid) <= 0)
348                         goto err;
349                 md = EVP_get_digestbynid(def_nid);
350                 if (md == NULL)
351                         {
352                         CMSerr(CMS_F_CMS_ADD1_SIGNER, CMS_R_NO_DEFAULT_DIGEST);
353                         goto err;
354                         }
355                 }

357         if (!md)
358                 {
359                 CMSerr(CMS_F_CMS_ADD1_SIGNER, CMS_R_NO_DIGEST_SET);
360                 goto err;
361                 }

363         cms_DigestAlgorithm_set(si->digestAlgorithm, md);

365         /* See if digest is present in digestAlgorithms */
366         for (i = 0; i < sk_X509_ALGOR_num(sd->digestAlgorithms); i++)
367                 {
368                 ASN1_OBJECT *aoid;
369                 alg = sk_X509_ALGOR_value(sd->digestAlgorithms, i);
370                 X509_ALGOR_get0(&aoid, NULL, NULL, alg);
371                 if (OBJ_obj2nid(aoid) == EVP_MD_type(md))
372                         break;
373                 }

375         if (i == sk_X509_ALGOR_num(sd->digestAlgorithms))
376                 {
377                 alg = X509_ALGOR_new();
378                 if (!alg)
379                         goto merr;
380                 cms_DigestAlgorithm_set(alg, md);
381                 if (!sk_X509_ALGOR_push(sd->digestAlgorithms, alg))
382                         {
383                         X509_ALGOR_free(alg);
384                         goto merr;
385                         }
386                 }

388         if (pk->ameth && pk->ameth->pkey_ctrl)
389                 {
390                 i = pk->ameth->pkey_ctrl(pk, ASN1_PKEY_CTRL_CMS_SIGN,
391                                                 0, si);
```

```
392                 if (i == -2)
393                         {
394                         CMSerr(CMS_F_CMS_ADD1_SIGNER,
395                                 CMS_R_NOT_SUPPORTED_FOR_THIS_KEY_TYPE);
396                         goto err;
397                         }
398                 if (i <= 0)
399                         {
400                         CMSerr(CMS_F_CMS_ADD1_SIGNER, CMS_R_CTRL_FAILURE);
401                         goto err;
402                         }
403                 }

405         if (!(flags & CMS_NOATTR))
406                 {
407                 /* Initialialize signed attributes strutucture so other
408                  * attributes such as signing time etc are added later
409                  * even if we add none here.
410                  */
411                 if (!si->signedAttrs)
412                         {
413                         si->signedAttrs = sk_X509_ATTRIBUTE_new_null();
414                         if (!si->signedAttrs)
415                                 goto merr;
416                         }

418                 if (!(flags & CMS_NOSMIMECAP))
419                         {
420                         STACK_OF(X509_ALGOR) *smcap = NULL;
421                         i = CMS_add_standard_smimecap(&smcap);
422                         if (i)
423                                 i = CMS_add_smimecap(si, smcap);
424                         sk_X509_ALGOR_pop_free(smcap, X509_ALGOR_free);
425                         if (!i)
426                                 goto merr;
427                         }
428                 if (flags & CMS_REUSE_DIGEST)
429                         {
430                         if (!cms_copy_messageDigest(cms, si))
431                                 goto err;
432                         if (!(flags & CMS_PARTIAL) &&
433                                         !CMS_SignerInfo_sign(si))
434                                 goto err;
435                         }
436                 }

438         if (!(flags & CMS_NOCERTS))
439                 {
440                 /* NB ignore -1 return for duplicate cert */
441                 if (!CMS_add1_cert(cms, signer))
442                         goto merr;
443                 }

445         if (!sd->signerInfos)
446                 sd->signerInfos = sk_CMS_SignerInfo_new_null();
447         if (!sd->signerInfos ||
448                 !sk_CMS_SignerInfo_push(sd->signerInfos, si))
449                 goto merr;

451         return si;

453         merr:
454         CMSerr(CMS_F_CMS_ADD1_SIGNER, ERR_R_MALLOC_FAILURE);
455         err:
456         if (si)
457                 M_ASN1_free_of(si, CMS_SignerInfo);
```



```
458         return NULL;

460         }

462 static int cms_add1_signingTime(CMS_SignerInfo *si, ASN1_TIME *t)
463         {
464         ASN1_TIME *tt;
465         int r = 0;
466         if (t)
467                 tt = t;
468         else
469                 tt = X509_gmtime_adj(NULL, 0);

471         if (!tt)
472                 goto merr;

474         if (CMS_signed_add1_attr_by_NID(si, NID_pkcs9_signingTime,
475                                                 tt->type, tt, -1) <= 0)
476                 goto merr;

478         r = 1;

480         merr:

482         if (!t)
483                 ASN1_TIME_free(tt);

485         if (!r)
486                 CMSerr(CMS_F_CMS_ADD1_SIGNINGTIME, ERR_R_MALLOC_FAILURE);

488         return r;

490         }

492 STACK_OF(CMS_SignerInfo) *CMS_get0_SignerInfos(CMS_ContentInfo *cms)
493         {
494         CMS_SignedData *sd;
495         sd = cms_get0_signed(cms);
496         if (!sd)
497                 return NULL;
498         return sd->signerInfos;
499         }

501 STACK_OF(X509) *CMS_get0_signers(CMS_ContentInfo *cms)
502         {
503         STACK_OF(X509) *signers = NULL;
504         STACK_OF(CMS_SignerInfo) *sinfos;
505         CMS_SignerInfo *si;
506         int i;
507         sinfos = CMS_get0_SignerInfos(cms);
508         for (i = 0; i < sk_CMS_SignerInfo_num(sinfos); i++)
509                 {
510                 si = sk_CMS_SignerInfo_value(sinfos, i);
511                 if (si->signer)
512                         {
513                         if (!signers)
514                                 {
515                                 signers = sk_X509_new_null();
516                                 if (!signers)
517                                         return NULL;
518                                 }
519                         if (!sk_X509_push(signers, si->signer))
520                                 {
521                                 sk_X509_free(signers);
522                                 return NULL;
523                                 }
```

```
524                         }
525                 }
526         return signers;
527         }

529 void CMS_SignerInfo_set1_signer_cert(CMS_SignerInfo *si, X509 *signer)
530         {
531         if (signer)
532                 {
533                 CRYPTO_add(&signer->references, 1, CRYPTO_LOCK_X509);
534                 if (si->pkey)
535                         EVP_PKEY_free(si->pkey);
536                 si->pkey = X509_get_pubkey(signer);
537                 }
538         if (si->signer)
539                 X509_free(si->signer);
540         si->signer = signer;
541         }

543 int CMS_SignerInfo_get0_signer_id(CMS_SignerInfo *si,
544                                   ASN1_OCTET_STRING **keyid,
545                                   X509_NAME **issuer, ASN1_INTEGER **sno)
546         {
547         return cms_SignerIdentifier_get0_signer_id(si->sid, keyid, issuer, sno);
548         }

550 int CMS_SignerInfo_cert_cmp(CMS_SignerInfo *si, X509 *cert)
551         {
552         return cms_SignerIdentifier_cert_cmp(si->sid, cert);
553         }

555 int CMS_set1_signers_certs(CMS_ContentInfo *cms, STACK_OF(X509) *scerts,
556                            unsigned int flags)
557         {
558         CMS_SignedData *sd;
559         CMS_SignerInfo *si;
560         CMS_CertificateChoices *cch;
561         STACK_OF(CMS_CertificateChoices) *certs;
562         X509 *x;
563         int i, j;
564         int ret = 0;
565         sd = cms_get0_signed(cms);
566         if (!sd)
567                 return -1;
568         certs = sd->certificates;
569         for (i = 0; i < sk_CMS_SignerInfo_num(sd->signerInfos); i++)
570                 {
571                 si = sk_CMS_SignerInfo_value(sd->signerInfos, i);
572                 if (si->signer)
573                         continue;

575                 for (j = 0; j < sk_X509_num(scerts); j++)
576                         {
577                         x = sk_X509_value(scerts, j);
578                         if (CMS_SignerInfo_cert_cmp(si, x) == 0)
579                                 {
580                                 CMS_SignerInfo_set1_signer_cert(si, x);
581                                 ret++;
582                                 break;
583                                 }
584                         }

586                 if (si->signer || (flags & CMS_NOINTERN))
587                         continue;

589                 for (j = 0; j < sk_CMS_CertificateChoices_num(certs); j++)
```



```
590                         {
591                         cch = sk_CMS_CertificateChoices_value(certs, j);
592                         if (cch->type != 0)
593                                 continue;
594                         x = cch->d.certificate;
595                         if (CMS_SignerInfo_cert_cmp(si, x) == 0)
596                                 {
597                                 CMS_SignerInfo_set1_signer_cert(si, x);
598                                 ret++;
599                                 break;
600                                 }
601                         }
602                 }
603         return ret;
604         }

606 void CMS_SignerInfo_get0_algs(CMS_SignerInfo *si, EVP_PKEY **pk, X509 **signer,
607                                         X509_ALGOR **pdig, X509_ALGOR **psig)
608         {
609         if (pk)
610                 *pk = si->pkey;
611         if (signer)
612                 *signer = si->signer;
613         if (pdig)
614                 *pdig = si->digestAlgorithm;
615         if (psig)
616                 *psig = si->signatureAlgorithm;
617         }

619 static int cms_SignerInfo_content_sign(CMS_ContentInfo *cms,
620                                         CMS_SignerInfo *si, BIO *chain)
621         {
622         EVP_MD_CTX mctx;
623         int r = 0;
624         EVP_MD_CTX_init(&mctx);


627         if (!si->pkey)
628                 {
629                 CMSerr(CMS_F_CMS_SIGNERINFO_CONTENT_SIGN, CMS_R_NO_PRIVATE_KEY);
630                 return 0;
631                 }

633         if (!cms_DigestAlgorithm_find_ctx(&mctx, chain, si->digestAlgorithm))
634                 goto err;

636         /* If any signed attributes calculate and add messageDigest attribute */

638         if (CMS_signed_get_attr_count(si) >= 0)
639                 {
640                 ASN1_OBJECT *ctype =
641                         cms->d.signedData->encapContentInfo->eContentType;
642                 unsigned char md[EVP_MAX_MD_SIZE];
643                 unsigned int mdlen;
644                 if (!EVP_DigestFinal_ex(&mctx, md, &mdlen))
645                         goto err;
646                 if (!CMS_signed_add1_attr_by_NID(si, NID_pkcs9_messageDigest,
647                                                 V_ASN1_OCTET_STRING,
648                                                 md, mdlen))
649                         goto err;
650                 /* Copy content type across */
651                 if (CMS_signed_add1_attr_by_NID(si, NID_pkcs9_contentType,
652                                                 V_ASN1_OBJECT, ctype, -1) <= 0)
653                         goto err;
654                 if (!CMS_SignerInfo_sign(si))
655                         goto err;
```

```
656                 }
657         else
658                 {
659                 unsigned char *sig;
660                 unsigned int siglen;
661                 sig = OPENSSL_malloc(EVP_PKEY_size(si->pkey));
662                 if (!sig)
663                         {
664                         CMSerr(CMS_F_CMS_SIGNERINFO_CONTENT_SIGN,
665                                         ERR_R_MALLOC_FAILURE);
666                         goto err;
667                         }
668                 if (!EVP_SignFinal(&mctx, sig, &siglen, si->pkey))
669                         {
670                         CMSerr(CMS_F_CMS_SIGNERINFO_CONTENT_SIGN,
671                                         CMS_R_SIGNFINAL_ERROR);
672                         OPENSSL_free(sig);
673                         goto err;
674                         }
675                 ASN1_STRING_set0(si->signature, sig, siglen);
676                 }

678         r = 1;

680         err:
681         EVP_MD_CTX_cleanup(&mctx);
682         return r;

684         }

686 int cms_SignedData_final(CMS_ContentInfo *cms, BIO *chain)
687         {
688         STACK_OF(CMS_SignerInfo) *sinfos;
689         CMS_SignerInfo *si;
690         int i;
691         sinfos = CMS_get0_SignerInfos(cms);
692         for (i = 0; i < sk_CMS_SignerInfo_num(sinfos); i++)
693                 {
694                 si = sk_CMS_SignerInfo_value(sinfos, i);
695                 if (!cms_SignerInfo_content_sign(cms, si, chain))
696                         return 0;
697                 }
698         cms->d.signedData->encapContentInfo->partial = 0;
699         return 1;
700         }

702 int CMS_SignerInfo_sign(CMS_SignerInfo *si)
703         {
704         EVP_MD_CTX mctx;
705         EVP_PKEY_CTX *pctx;
706         unsigned char *abuf = NULL;
707         int alen;
708         size_t siglen;
709         const EVP_MD *md = NULL;

711         md = EVP_get_digestbyobj(si->digestAlgorithm->algorithm);
712         if (md == NULL)
713                 return 0;

715         EVP_MD_CTX_init(&mctx);

717         if (CMS_signed_get_attr_by_NID(si, NID_pkcs9_signingTime, -1) < 0)
718                 {
719                 if (!cms_add1_signingTime(si, NULL))
720                         goto err;
721                 }
```



```
723         if (EVP_DigestSignInit(&mctx, &pctx, md, NULL, si->pkey) <= 0)
724                 goto err;

726         if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_SIGN,
727                                 EVP_PKEY_CTRL_CMS_SIGN, 0, si) <= 0)
728                 {
729                 CMSerr(CMS_F_CMS_SIGNERINFO_SIGN, CMS_R_CTRL_ERROR);
730                 goto err;
731                 }

733         alen = ASN1_item_i2d((ASN1_VALUE *)si->signedAttrs,&abuf,
734                                 ASN1_ITEM_rptr(CMS_Attributes_Sign));
735         if(!abuf)
736                 goto err;
737         if (EVP_DigestSignUpdate(&mctx, abuf, alen) <= 0)
738                 goto err;
739         if (EVP_DigestSignFinal(&mctx, NULL, &siglen) <= 0)
740                 goto err;
741         OPENSSL_free(abuf);
742         abuf = OPENSSL_malloc(siglen);
743         if(!abuf)
744                 goto err;
745         if (EVP_DigestSignFinal(&mctx, abuf, &siglen) <= 0)
746                 goto err;

748         if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_SIGN,
749                                 EVP_PKEY_CTRL_CMS_SIGN, 1, si) <= 0)
750                 {
751                 CMSerr(CMS_F_CMS_SIGNERINFO_SIGN, CMS_R_CTRL_ERROR);
752                 goto err;
753                 }

755         EVP_MD_CTX_cleanup(&mctx);

757         ASN1_STRING_set0(si->signature, abuf, siglen);

759         return 1;

761         err:
762         if (abuf)
763                 OPENSSL_free(abuf);
764         EVP_MD_CTX_cleanup(&mctx);
765         return 0;

767         }

769 int CMS_SignerInfo_verify(CMS_SignerInfo *si)
770         {
771         EVP_MD_CTX mctx;
772         EVP_PKEY_CTX *pctx;
773         unsigned char *abuf = NULL;
774         int alen, r = -1;
775         const EVP_MD *md = NULL;

777         if (!si->pkey)
778                 {
779                 CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY, CMS_R_NO_PUBLIC_KEY);
780                 return -1;
781                 }

783         md = EVP_get_digestbyobj(si->digestAlgorithm->algorithm);
784         if (md == NULL)
785                 return -1;
786         EVP_MD_CTX_init(&mctx);
787         if (EVP_DigestVerifyInit(&mctx, &pctx, md, NULL, si->pkey) <= 0)
```

```
788                 goto err;

790         alen = ASN1_item_i2d((ASN1_VALUE *)si->signedAttrs,&abuf,
791                              ASN1_ITEM_rptr(CMS_Attributes_Verify));
792         if(!abuf)
793                 goto err;
794         r = EVP_DigestVerifyUpdate(&mctx, abuf, alen);
795         OPENSSL_free(abuf);
796         if (r <= 0)
797                 {
798                 r = -1;
799                 goto err;
800                 }
801         r = EVP_DigestVerifyFinal(&mctx,
802                         si->signature->data, si->signature->length);
803         if (r <= 0)
804                 CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY, CMS_R_VERIFICATION_FAILURE);
805         err:
806         EVP_MD_CTX_cleanup(&mctx);
807         return r;
808         }

810 /* Create a chain of digest BIOs from a CMS ContentInfo */

812 BIO *cms_SignedData_init_bio(CMS_ContentInfo *cms)
813         {
814         int i;
815         CMS_SignedData *sd;
816         BIO *chain = NULL;
817         sd = cms_get0_signed(cms);
818         if (!sd)
819                 return NULL;
820         if (cms->d.signedData->encapContentInfo->partial)
821                 cms_sd_set_version(sd);
822         for (i = 0; i < sk_X509_ALGOR_num(sd->digestAlgorithms); i++)
823                 {
824                 X509_ALGOR *digestAlgorithm;
825                 BIO *mdbio;
826                 digestAlgorithm = sk_X509_ALGOR_value(sd->digestAlgorithms, i);
827                 mdbio = cms_DigestAlgorithm_init_bio(digestAlgorithm);
828                 if (!mdbio)
829                         goto err;
830                 if (chain)
831                          BIO_push(chain, mdbio);
832                 else
833                         chain = mdbio;
834                 }
835         return chain;
836         err:
837         if (chain)
838                 BIO_free_all(chain);
839         return NULL;
840         }

842 int CMS_SignerInfo_verify_content(CMS_SignerInfo *si, BIO *chain)
843         {
844         ASN1_OCTET_STRING *os = NULL;
845         EVP_MD_CTX mctx;
846         int r = -1;
847         EVP_MD_CTX_init(&mctx);
848         /* If we have any signed attributes look for messageDigest value */
849         if (CMS_signed_get_attr_count(si) >= 0)
850                 {
851                 os = CMS_signed_get0_data_by_OBJ(si,
852                                         OBJ_nid2obj(NID_pkcs9_messageDigest),
853                                         -3, V_ASN1_OCTET_STRING);
```



```
854                 if (!os)
855                         {
856                         CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY_CONTENT,
857                                 CMS_R_ERROR_READING_MESSAGEDIGEST_ATTRIBUTE);
858                         goto err;
859                         }
860                 }

862         if (!cms_DigestAlgorithm_find_ctx(&mctx, chain, si->digestAlgorithm))
863                 goto err;

865         /* If messageDigest found compare it */

867         if (os)
868                 {
869                 unsigned char mval[EVP_MAX_MD_SIZE];
870                 unsigned int mlen;
871                 if (EVP_DigestFinal_ex(&mctx, mval, &mlen) <= 0)
872                         {
873                         CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY_CONTENT,
874                                 CMS_R_UNABLE_TO_FINALIZE_CONTEXT);
875                         goto err;
876                         }
877                 if (mlen != (unsigned int)os->length)
878                         {
879                         CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY_CONTENT,
880                                 CMS_R_MESSAGEDIGEST_ATTRIBUTE_WRONG_LENGTH);
881                         goto err;
882                         }

884                 if (memcmp(mval, os->data, mlen))
885                         {
886                         CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY_CONTENT,
887                                 CMS_R_VERIFICATION_FAILURE);
888                         r = 0;
889                         }
890                 else
891                         r = 1;
892                 }
893         else
894                 {
895                 r = EVP_VerifyFinal(&mctx, si->signature->data,
896                                         si->signature->length, si->pkey);
897                 if (r <= 0)
898                         {
899                         CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY_CONTENT,
900                                 CMS_R_VERIFICATION_FAILURE);
901                         r = 0;
902                         }
903                 }

905         err:
906         EVP_MD_CTX_cleanup(&mctx);
907         return r;

909         }

911 int CMS_add_smimecap(CMS_SignerInfo *si, STACK_OF(X509_ALGOR) *algs)
912         {
913         unsigned char *smder = NULL;
914         int smderlen, r;
915         smderlen = i2d_X509_ALGORS(algs, &smder);
916         if (smderlen <= 0)
917                 return 0;
918         r = CMS_signed_add1_attr_by_NID(si, NID_SMIMECapabilities,
919                                         V_ASN1_SEQUENCE, smder, smderlen);
```

```
920         OPENSSL_free(smder);
921         return r;
922         }

924 int CMS_add_simple_smimecap(STACK_OF(X509_ALGOR) **algs,
925                              int algnid, int keysize)
926         {
927         X509_ALGOR *alg;
928         ASN1_INTEGER *key = NULL;
929         if (keysize > 0)
930                 {
931                 key = ASN1_INTEGER_new();
932                 if (!key || !ASN1_INTEGER_set(key, keysize))
933                         return 0;
934                 }
935         alg = X509_ALGOR_new();
936         if (!alg)
937                 {
938                 if (key)
939                         ASN1_INTEGER_free(key);
940                 return 0;
941                 }

943         X509_ALGOR_set0(alg, OBJ_nid2obj(algnid),
944                              key ? V_ASN1_INTEGER : V_ASN1_UNDEF, key);
945         if (!*algs)
946                 *algs = sk_X509_ALGOR_new_null();
947         if (!*algs || !sk_X509_ALGOR_push(*algs, alg))
948                 {
949                 X509_ALGOR_free(alg);
950                 return 0;
951                 }
952         return 1;
953         }

955 /* Check to see if a cipher exists and if so add S/MIME capabilities */

957 static int cms_add_cipher_smcap(STACK_OF(X509_ALGOR) **sk, int nid, int arg)
958         {
959         if (EVP_get_cipherbynid(nid))
960                 return CMS_add_simple_smimecap(sk, nid, arg);
961         return 1;
962         }

964 static int cms_add_digest_smcap(STACK_OF(X509_ALGOR) **sk, int nid, int arg)
965         {
966         if (EVP_get_digestbynid(nid))
967                 return CMS_add_simple_smimecap(sk, nid, arg);
968         return 1;
969         }

971 int CMS_add_standard_smimecap(STACK_OF(X509_ALGOR) **smcap)
972         {
973         if (!cms_add_cipher_smcap(smcap, NID_aes_256_cbc, -1)
974                 || !cms_add_digest_smcap(smcap, NID_id_GostR3411_94, -1)
975                 || !cms_add_cipher_smcap(smcap, NID_id_Gost28147_89, -1)
976                 || !cms_add_cipher_smcap(smcap, NID_aes_192_cbc, -1)
977                 || !cms_add_cipher_smcap(smcap, NID_aes_128_cbc, -1)
978                 || !cms_add_cipher_smcap(smcap, NID_des_ede3_cbc, -1)
979                 || !cms_add_cipher_smcap(smcap, NID_rc2_cbc, 128)
980                 || !cms_add_cipher_smcap(smcap, NID_rc2_cbc, 64)
981                 || !cms_add_cipher_smcap(smcap, NID_des_cbc, -1)
982                 || !cms_add_cipher_smcap(smcap, NID_rc2_cbc, 40))
983                 return 0;
984         return 1;
985         }
```



```
986 #endif /* ! codereview */
```

```
**********************************************************
   19482 Wed Aug 13 19:52:23 2014
new/usr/src/lib/openssl/libsunw_crypto/cms/cms_smime.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cms/cms_smime.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>
#include <openssl/cms.h>
#include "cms_lcl.h"
```



```
static int cms_copy_content(BIO *out, BIO *in, unsigned int flags)
        {
        unsigned char buf[4096];
        int r = 0, i;
        BIO *tmpout = NULL;

        if (out == NULL)
                tmpout = BIO_new(BIO_s_null());
        else if (flags & CMS_TEXT)
                {
                tmpout = BIO_new(BIO_s_mem());
                BIO_set_mem_eof_return(tmpout, 0);
                }
        else
                tmpout = out;

        if(!tmpout)
                {
                CMSerr(CMS_F_CMS_COPY_CONTENT,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        /* Read all content through chain to process digest, decrypt etc */
        for (;;)
        {
                i=BIO_read(in,buf,sizeof(buf));
                if (i <= 0)
                        {
                        if (BIO_method_type(in) == BIO_TYPE_CIPHER)
                                {
                                if (!BIO_get_cipher_status(in))
                                        goto err;
                                }
                        if (i < 0)
                                goto err;
                        break;
                        }

                if (tmpout && (BIO_write(tmpout, buf, i) != i))
                        goto err;
        }

        if(flags & CMS_TEXT)
                {
                if(!SMIME_text(tmpout, out))
                        {
                        CMSerr(CMS_F_CMS_COPY_CONTENT,CMS_R_SMIME_TEXT_ERROR);
                        goto err;
                        }
                }

        r = 1;

        err:
        if (tmpout && (tmpout != out))
                BIO_free(tmpout);
        return r;

        }

static int check_content(CMS_ContentInfo *cms)
        {
        ASN1_OCTET_STRING **pos = CMS_get0_content(cms);
        if (!pos || !*pos)
                {
                CMSerr(CMS_F_CHECK_CONTENT, CMS_R_NO_CONTENT);
```

```
128                 return 0;
129                 }
130         return 1;
131         }

133 static void do_free_upto(BIO *f, BIO *upto)
134         {
135         if (upto)
136                 {
137                 BIO *tbio;
138                 do
139                         {
140                         tbio = BIO_pop(f);
141                         BIO_free(f);
142                         f = tbio;
143                         }
144                 while (f != upto);
145                 }
146         else
147                 BIO_free_all(f);
148         }

150 int CMS_data(CMS_ContentInfo *cms, BIO *out, unsigned int flags)
151         {
152         BIO *cont;
153         int r;
154         if (OBJ_obj2nid(CMS_get0_type(cms)) != NID_pkcs7_data)
155                 {
156                 CMSerr(CMS_F_CMS_DATA, CMS_R_TYPE_NOT_DATA);
157                 return 0;
158                 }
159         cont = CMS_dataInit(cms, NULL);
160         if (!cont)
161                 return 0;
162         r = cms_copy_content(out, cont, flags);
163         BIO_free_all(cont);
164         return r;
165         }

167 CMS_ContentInfo *CMS_data_create(BIO *in, unsigned int flags)
168         {
169         CMS_ContentInfo *cms;
170         cms = cms_Data_create();
171         if (!cms)
172                 return NULL;

174         if ((flags & CMS_STREAM) || CMS_final(cms, in, NULL, flags))
175                 return cms;

177         CMS_ContentInfo_free(cms);

179         return NULL;
180         }

182 int CMS_digest_verify(CMS_ContentInfo *cms, BIO *dcont, BIO *out,
183                                                         unsigned int flags)
184         {
185         BIO *cont;
186         int r;
187         if (OBJ_obj2nid(CMS_get0_type(cms)) != NID_pkcs7_digest)
188                 {
189                 CMSerr(CMS_F_CMS_DIGEST_VERIFY, CMS_R_TYPE_NOT_DIGESTED_DATA);
190                 return 0;
191                 }

193         if (!dcont && !check_content(cms))
```



```
194                 return 0;

196         cont = CMS_dataInit(cms, dcont);
197         if (!cont)
198                 return 0;
199         r = cms_copy_content(out, cont, flags);
200         if (r)
201                 r = cms_DigestedData_do_final(cms, cont, 1);
202         do_free_upto(cont, dcont);
203         return r;
204         }

206 CMS_ContentInfo *CMS_digest_create(BIO *in, const EVP_MD *md,
207                                         unsigned int flags)
208         {
209         CMS_ContentInfo *cms;
210         if (!md)
211                 md = EVP_sha1();
212         cms = cms_DigestedData_create(md);
213         if (!cms)
214                 return NULL;

216         if(!(flags & CMS_DETACHED))
217                 CMS_set_detached(cms, 0);

219         if ((flags & CMS_STREAM) || CMS_final(cms, in, NULL, flags))
220                 return cms;

222         CMS_ContentInfo_free(cms);
223         return NULL;
224         }

226 int CMS_EncryptedData_decrypt(CMS_ContentInfo *cms,
227                                 const unsigned char *key, size_t keylen,
228                                 BIO *dcont, BIO *out, unsigned int flags)
229         {
230         BIO *cont;
231         int r;
232         if (OBJ_obj2nid(CMS_get0_type(cms)) != NID_pkcs7_encrypted)
233                 {
234                 CMSerr(CMS_F_CMS_ENCRYPTEDDATA_DECRYPT,
235                                         CMS_R_TYPE_NOT_ENCRYPTED_DATA);
236                 return 0;
237                 }

239         if (!dcont && !check_content(cms))
240                 return 0;

242         if (CMS_EncryptedData_set1_key(cms, NULL, key, keylen) <= 0)
243                 return 0;
244         cont = CMS_dataInit(cms, dcont);
245         if (!cont)
246                 return 0;
247         r = cms_copy_content(out, cont, flags);
248         do_free_upto(cont, dcont);
249         return r;
250         }

252 CMS_ContentInfo *CMS_EncryptedData_encrypt(BIO *in, const EVP_CIPHER *cipher,
253                                         const unsigned char *key, size_t keylen,
254                                         unsigned int flags)
255         {
256         CMS_ContentInfo *cms;
257         if (!cipher)
258                 {
259                 CMSerr(CMS_F_CMS_ENCRYPTEDDATA_ENCRYPT, CMS_R_NO_CIPHER);
```

```
260                 return NULL;
261                 }
262         cms = CMS_ContentInfo_new();
263         if (!cms)
264                 return NULL;
265         if (!CMS_EncryptedData_set1_key(cms, cipher, key, keylen))
266                 return NULL;
267
268         if(!(flags & CMS_DETACHED))
269                 CMS_set_detached(cms, 0);
270
271         if ((flags & (CMS_STREAM|CMS_PARTIAL))
272                 || CMS_final(cms, in, NULL, flags))
273                 return cms;
274
275         CMS_ContentInfo_free(cms);
276         return NULL;
277         }
278
279 static int cms_signerinfo_verify_cert(CMS_SignerInfo *si,
280                                       X509_STORE *store,
281                                       STACK_OF(X509) *certs,
282                                       STACK_OF(X509_CRL) *crls,
283                                       unsigned int flags)
284         {
285         X509_STORE_CTX ctx;
286         X509 *signer;
287         int i, j, r = 0;
288         CMS_SignerInfo_get0_algs(si, NULL, &signer, NULL, NULL);
289         if (!X509_STORE_CTX_init(&ctx, store, signer, certs))
290                 {
291                 CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY_CERT,
292                                                 CMS_R_STORE_INIT_ERROR);
293                 goto err;
294                 }
295         X509_STORE_CTX_set_default(&ctx, "smime_sign");
296         if (crls)
297                 X509_STORE_CTX_set0_crls(&ctx, crls);
298
299         i = X509_verify_cert(&ctx);
300         if (i <= 0)
301                 {
302                 j = X509_STORE_CTX_get_error(&ctx);
303                 CMSerr(CMS_F_CMS_SIGNERINFO_VERIFY_CERT,
304                                                 CMS_R_CERTIFICATE_VERIFY_ERROR);
305                 ERR_add_error_data(2, "Verify error:",
306                                       X509_verify_cert_error_string(j));
307                 goto err;
308                 }
309         r = 1;
310         err:
311         X509_STORE_CTX_cleanup(&ctx);
312         return r;
313
314         }
315
316 int CMS_verify(CMS_ContentInfo *cms, STACK_OF(X509) *certs,
317                  X509_STORE *store, BIO *dcont, BIO *out, unsigned int flags)
318         {
319         CMS_SignerInfo *si;
320         STACK_OF(CMS_SignerInfo) *sinfos;
321         STACK_OF(X509) *cms_certs = NULL;
322         STACK_OF(X509_CRL) *crls = NULL;
323         X509 *signer;
324         int i, scount = 0, ret = 0;
325         BIO *cmsbio = NULL, *tmpin = NULL;
```



```
326
327         if (!dcont && !check_content(cms))
328                 return 0;
329
330         /* Attempt to find all signer certificates */
331
332         sinfos = CMS_get0_SignerInfos(cms);
333
334         if (sk_CMS_SignerInfo_num(sinfos) <= 0)
335                 {
336                 CMSerr(CMS_F_CMS_VERIFY, CMS_R_NO_SIGNERS);
337                 goto err;
338                 }
339
340         for (i = 0; i < sk_CMS_SignerInfo_num(sinfos); i++)
341                 {
342                 si = sk_CMS_SignerInfo_value(sinfos, i);
343                 CMS_SignerInfo_get0_algs(si, NULL, &signer, NULL, NULL);
344                 if (signer)
345                         scount++;
346                 }
347
348         if (scount != sk_CMS_SignerInfo_num(sinfos))
349                 scount += CMS_set1_signers_certs(cms, certs, flags);
350
351         if (scount != sk_CMS_SignerInfo_num(sinfos))
352                 {
353                 CMSerr(CMS_F_CMS_VERIFY, CMS_R_SIGNER_CERTIFICATE_NOT_FOUND);
354                 goto err;
355                 }
356
357         /* Attempt to verify all signers certs */
358
359         if (!(flags & CMS_NO_SIGNER_CERT_VERIFY))
360                 {
361                 cms_certs = CMS_get1_certs(cms);
362                 if (!(flags & CMS_NOCRL))
363                         crls = CMS_get1_crls(cms);
364                 for (i = 0; i < sk_CMS_SignerInfo_num(sinfos); i++)
365                         {
366                         si = sk_CMS_SignerInfo_value(sinfos, i);
367                         if (!cms_signerinfo_verify_cert(si, store,
368                                                         cms_certs, crls, flags))
369                                 goto err;
370                         }
371                 }
372
373         /* Attempt to verify all SignerInfo signed attribute signatures */
374
375         if (!(flags & CMS_NO_ATTR_VERIFY))
376                 {
377                 for (i = 0; i < sk_CMS_SignerInfo_num(sinfos); i++)
378                         {
379                         si = sk_CMS_SignerInfo_value(sinfos, i);
380                         if (CMS_signed_get_attr_count(si) < 0)
381                                 continue;
382                         if (CMS_SignerInfo_verify(si) <= 0)
383                                 goto err;
384                         }
385                 }
386
387         /* Performance optimization: if the content is a memory BIO then
388          * store its contents in a temporary read only memory BIO. This
389          * avoids potentially large numbers of slow copies of data which will
390          * occur when reading from a read write memory BIO when signatures
391          * are calculated.
```

```
392          */
394         if (dcont && (BIO_method_type(dcont) == BIO_TYPE_MEM))
395                 {
396                 char *ptr;
397                 long len;
398                 len = BIO_get_mem_data(dcont, &ptr);
399                 tmpin = BIO_new_mem_buf(ptr, len);
400                 if (tmpin == NULL)
401                         {
402                         CMSerr(CMS_F_CMS_VERIFY,ERR_R_MALLOC_FAILURE);
403                         return 0;
404                         }
405                 }
406         else
407                 tmpin = dcont;
410         cmsbio=CMS_dataInit(cms, tmpin);
411         if (!cmsbio)
412                 goto err;
414         if (!cms_copy_content(out, cmsbio, flags))
415                 goto err;
417         if (!(flags & CMS_NO_CONTENT_VERIFY))
418                 {
419                 for (i = 0; i < sk_CMS_SignerInfo_num(sinfos); i++)
420                         {
421                         si = sk_CMS_SignerInfo_value(sinfos, i);
422                         if (CMS_SignerInfo_verify_content(si, cmsbio) <= 0)
423                                 {
424                                 CMSerr(CMS_F_CMS_VERIFY,
425                                         CMS_R_CONTENT_VERIFY_ERROR);
426                                 goto err;
427                                 }
428                         }
429                 }
431         ret = 1;
433         err:
435         if (dcont && (tmpin == dcont))
436                 do_free_upto(cmsbio, dcont);
437         else
438                 BIO_free_all(cmsbio);
440         if (cms_certs)
441                 sk_X509_pop_free(cms_certs, X509_free);
442         if (crls)
443                 sk_X509_CRL_pop_free(crls, X509_CRL_free);
445         return ret;
446         }
448 int CMS_verify_receipt(CMS_ContentInfo *rcms, CMS_ContentInfo *ocms,
449                         STACK_OF(X509) *certs,
450                         X509_STORE *store, unsigned int flags)
451         {
452         int r;
453         flags &= ~(CMS_DETACHED|CMS_TEXT);
454         r = CMS_verify(rcms, certs, store, NULL, NULL, flags);
455         if (r <= 0)
456                 return r;
457         return cms_Receipt_verify(rcms, ocms);
```



```
458         }
460 CMS_ContentInfo *CMS_sign(X509 *signcert, EVP_PKEY *pkey, STACK_OF(X509) *certs,
461                                                 BIO *data, unsigned int flags)
462         {
463         CMS_ContentInfo *cms;
464         int i;
466         cms = CMS_ContentInfo_new();
467         if (!cms || !CMS_SignedData_init(cms))
468                 goto merr;
470         if (pkey && !CMS_add1_signer(cms, signcert, pkey, NULL, flags))
471                 {
472                 CMSerr(CMS_F_CMS_SIGN, CMS_R_ADD_SIGNER_ERROR);
473                 goto err;
474                 }
476         for (i = 0; i < sk_X509_num(certs); i++)
477                 {
478                 X509 *x = sk_X509_value(certs, i);
479                 if (!CMS_add1_cert(cms, x))
480                         goto merr;
481                 }
483         if(!(flags & CMS_DETACHED))
484                 CMS_set_detached(cms, 0);
486         if ((flags & (CMS_STREAM|CMS_PARTIAL))
487                 || CMS_final(cms, data, NULL, flags))
488                 return cms;
489         else
490                 goto err;
492         merr:
493         CMSerr(CMS_F_CMS_SIGN, ERR_R_MALLOC_FAILURE);
495         err:
496         if (cms)
497                 CMS_ContentInfo_free(cms);
498         return NULL;
499         }
501 CMS_ContentInfo *CMS_sign_receipt(CMS_SignerInfo *si,
502                                         X509 *signcert, EVP_PKEY *pkey,
503                                         STACK_OF(X509) *certs,
504                                         unsigned int flags)
505         {
506         CMS_SignerInfo *rct_si;
507         CMS_ContentInfo *cms = NULL;
508         ASN1_OCTET_STRING **pos, *os;
509         BIO *rct_cont = NULL;
510         int r = 0;
512         flags &= ~(CMS_STREAM|CMS_TEXT);
513         /* Not really detached but avoids content being allocated */
514         flags |= CMS_PARTIAL|CMS_BINARY|CMS_DETACHED;
515         if (!pkey || !signcert)
516                 {
517                 CMSerr(CMS_F_CMS_SIGN_RECEIPT, CMS_R_NO_KEY_OR_CERT);
518                 return NULL;
519                 }
521         /* Initialize signed data */
523         cms = CMS_sign(NULL, NULL, certs, NULL, flags);
```

```
524         if (!cms)
525                 goto err;

527         /* Set inner content type to signed receipt */
528         if (!CMS_set1_eContentType(cms, OBJ_nid2obj(NID_id_smime_ct_receipt)))
529                 goto err;

531         rct_si = CMS_add1_signer(cms, signcert, pkey, NULL, flags);
532         if (!rct_si)
533                 {
534                 CMSerr(CMS_F_CMS_SIGN_RECEIPT, CMS_R_ADD_SIGNER_ERROR);
535                 goto err;
536                 }

538         os = cms_encode_Receipt(si);

540         if (!os)
541                 goto err;

543         /* Set content to digest */
544         rct_cont = BIO_new_mem_buf(os->data, os->length);
545         if (!rct_cont)
546                 goto err;

548         /* Add msgSigDigest attribute */

550         if (!cms_msgSigDigest_add1(rct_si, si))
551                 goto err;

553         /* Finalize structure */
554         if (!CMS_final(cms, rct_cont, NULL, flags))
555                 goto err;

557         /* Set embedded content */
558         pos = CMS_get0_content(cms);
559         *pos = os;

561         r = 1;

563         err:
564         if (rct_cont)
565                 BIO_free(rct_cont);
566         if (r)
567                 return cms;
568         CMS_ContentInfo_free(cms);
569         return NULL;

571         }

573 CMS_ContentInfo *CMS_encrypt(STACK_OF(X509) *certs, BIO *data,
574                              const EVP_CIPHER *cipher, unsigned int flags)
575         {
576         CMS_ContentInfo *cms;
577         int i;
578         X509 *recip;
579         cms = CMS_EnvelopedData_create(cipher);
580         if (!cms)
581                 goto merr;
582         for (i = 0; i < sk_X509_num(certs); i++)
583                 {
584                 recip = sk_X509_value(certs, i);
585                 if (!CMS_add1_recipient_cert(cms, recip, flags))
586                         {
587                         CMSerr(CMS_F_CMS_ENCRYPT, CMS_R_RECIPIENT_ERROR);
588                         goto err;
589                         }
```



```
590                 }

592         if(!(flags & CMS_DETACHED))
593                 CMS_set_detached(cms, 0);

595         if ((flags & (CMS_STREAM|CMS_PARTIAL))
596                 || CMS_final(cms, data, NULL, flags))
597                 return cms;
598         else
599                 goto err;

601         merr:
602         CMSerr(CMS_F_CMS_ENCRYPT, ERR_R_MALLOC_FAILURE);
603         err:
604         if (cms)
605                 CMS_ContentInfo_free(cms);
606         return NULL;
607         }

609 int CMS_decrypt_set1_pkey(CMS_ContentInfo *cms, EVP_PKEY *pk, X509 *cert)
610         {
611         STACK_OF(CMS_RecipientInfo) *ris;
612         CMS_RecipientInfo *ri;
613         int i, r;
614         int debug = 0, ri_match = 0;
615         ris = CMS_get0_RecipientInfos(cms);
616         if (ris)
617                 debug = cms->d.envelopedData->encryptedContentInfo->debug;
618         for (i = 0; i < sk_CMS_RecipientInfo_num(ris); i++)
619                 {
620                 ri = sk_CMS_RecipientInfo_value(ris, i);
621                 if (CMS_RecipientInfo_type(ri) != CMS_RECIPINFO_TRANS)
622                         continue;
623                 ri_match = 1;
624                 /* If we have a cert try matching RecipientInfo
625                  * otherwise try them all.
626                  */
627                 if (!cert || (CMS_RecipientInfo_ktri_cert_cmp(ri, cert) == 0))
628                         {
629                         CMS_RecipientInfo_set0_pkey(ri, pk);
630                         r = CMS_RecipientInfo_decrypt(cms, ri);
631                         CMS_RecipientInfo_set0_pkey(ri, NULL);
632                         if (cert)
633                                 {
634                                 /* If not debugging clear any error and
635                                  * return success to avoid leaking of
636                                  * information useful to MMA
637                                  */
638                                 if (!debug)
639                                         {
640                                         ERR_clear_error();
641                                         return 1;
642                                         }
643                                 if (r > 0)
644                                         return 1;
645                                 CMSerr(CMS_F_CMS_DECRYPT_SET1_PKEY,
646                                                 CMS_R_DECRYPT_ERROR);
647                                 return 0;
648                                 }
649                         /* If no cert and not debugging don't leave loop
650                          * after first successful decrypt. Always attempt
651                          * to decrypt all recipients to avoid leaking timing
652                          * of a successful decrypt.
653                          */
654                         else if (r > 0 && debug)
655                                 return 1;
```

```
656                         }
657                 }
658         /* If no cert and not debugging always return success */
659         if (ri_match && !cert && !debug)
660                 {
661                 ERR_clear_error();
662                 return 1;
663                 }

665         CMSerr(CMS_F_CMS_DECRYPT_SET1_PKEY, CMS_R_NO_MATCHING_RECIPIENT);
666         return 0;

668         }

670 int CMS_decrypt_set1_key(CMS_ContentInfo *cms,
671                                 unsigned char *key, size_t keylen,
672                                 unsigned char *id, size_t idlen)
673         {
674         STACK_OF(CMS_RecipientInfo) *ris;
675         CMS_RecipientInfo *ri;
676         int i, r;
677         ris = CMS_get0_RecipientInfos(cms);
678         for (i = 0; i < sk_CMS_RecipientInfo_num(ris); i++)
679                 {
680                 ri = sk_CMS_RecipientInfo_value(ris, i);
681                 if (CMS_RecipientInfo_type(ri) != CMS_RECIPINFO_KEK)
682                                 continue;

684                 /* If we have an id try matching RecipientInfo
685                  * otherwise try them all.
686                  */
687                 if (!id || (CMS_RecipientInfo_kekri_id_cmp(ri, id, idlen) == 0))
688                         {
689                         CMS_RecipientInfo_set0_key(ri, key, keylen);
690                         r = CMS_RecipientInfo_decrypt(cms, ri);
691                         CMS_RecipientInfo_set0_key(ri, NULL, 0);
692                         if (r > 0)
693                                 return 1;
694                         if (id)
695                                 {
696                                 CMSerr(CMS_F_CMS_DECRYPT_SET1_KEY,
697                                                 CMS_R_DECRYPT_ERROR);
698                                 return 0;
699                                 }
700                         ERR_clear_error();
701                         }
702                 }

704         CMSerr(CMS_F_CMS_DECRYPT_SET1_KEY, CMS_R_NO_MATCHING_RECIPIENT);
705         return 0;

707         }

709 int CMS_decrypt_set1_password(CMS_ContentInfo *cms,
710                                 unsigned char *pass, ossl_ssize_t passlen)
711         {
712         STACK_OF(CMS_RecipientInfo) *ris;
713         CMS_RecipientInfo *ri;
714         int i, r;
715         ris = CMS_get0_RecipientInfos(cms);
716         for (i = 0; i < sk_CMS_RecipientInfo_num(ris); i++)
717                 {
718                 ri = sk_CMS_RecipientInfo_value(ris, i);
719                 if (CMS_RecipientInfo_type(ri) != CMS_RECIPINFO_PASS)
720                                 continue;
721                 CMS_RecipientInfo_set0_password(ri, pass, passlen);
```



```
722                 r = CMS_RecipientInfo_decrypt(cms, ri);
723                 CMS_RecipientInfo_set0_password(ri, NULL, 0);
724                 if (r > 0)
725                         return 1;
726                 }

728         CMSerr(CMS_F_CMS_DECRYPT_SET1_PASSWORD, CMS_R_NO_MATCHING_RECIPIENT);
729         return 0;

731         }

733 int CMS_decrypt(CMS_ContentInfo *cms, EVP_PKEY *pk, X509 *cert,
734                                 BIO *dcont, BIO *out,
735                                 unsigned int flags)
736         {
737         int r;
738         BIO *cont;
739         if (OBJ_obj2nid(CMS_get0_type(cms)) != NID_pkcs7_enveloped)
740                 {
741                 CMSerr(CMS_F_CMS_DECRYPT, CMS_R_TYPE_NOT_ENVELOPED_DATA);
742                 return 0;
743                 }
744         if (!dcont && !check_content(cms))
745                 return 0;
746         if (flags & CMS_DEBUG_DECRYPT)
747                 cms->d.envelopedData->encryptedContentInfo->debug = 1;
748         else
749                 cms->d.envelopedData->encryptedContentInfo->debug = 0;
750         if (!pk && !cert && !dcont && !out)
751                 return 1;
752         if (pk && !CMS_decrypt_set1_pkey(cms, pk, cert))
753                 return 0;
754         cont = CMS_dataInit(cms, dcont);
755         if (!cont)
756                 return 0;
757         r = cms_copy_content(out, cont, flags);
758         do_free_upto(cont, dcont);
759         return r;
760         }

762 int CMS_final(CMS_ContentInfo *cms, BIO *data, BIO *dcont, unsigned int flags)
763         {
764         BIO *cmsbio;
765         int ret = 0;
766         if (!(cmsbio = CMS_dataInit(cms, dcont)))
767                 {
768                 CMSerr(CMS_F_CMS_FINAL,ERR_R_MALLOC_FAILURE);
769                 return 0;
770                 }

772         SMIME_crlf_copy(data, cmsbio, flags);

774         (void)BIO_flush(cmsbio);


777         if (!CMS_dataFinal(cms, cmsbio))
778                 {
779                 CMSerr(CMS_F_CMS_FINAL,CMS_R_CMS_DATAFINAL_ERROR);
780                 goto err;
781                 }

783         ret = 1;

785         err:
786         do_free_upto(cmsbio, dcont);
```

```
788         return ret;

790         }

792 #ifdef ZLIB

794 int CMS_uncompress(CMS_ContentInfo *cms, BIO *dcont, BIO *out,
795                                                     unsigned int flags)
796         {
797         BIO *cont;
798         int r;
799         if (OBJ_obj2nid(CMS_get0_type(cms)) != NID_id_smime_ct_compressedData)
800                 {
801                 CMSerr(CMS_F_CMS_UNCOMPRESS,
802                                         CMS_R_TYPE_NOT_COMPRESSED_DATA);
803                 return 0;
804                 }

806         if (!dcont && !check_content(cms))
807                 return 0;

809         cont = CMS_dataInit(cms, dcont);
810         if (!cont)
811                 return 0;
812         r = cms_copy_content(out, cont, flags);
813         do_free_upto(cont, dcont);
814         return r;
815         }

817 CMS_ContentInfo *CMS_compress(BIO *in, int comp_nid, unsigned int flags)
818         {
819         CMS_ContentInfo *cms;
820         if (comp_nid <= 0)
821                 comp_nid = NID_zlib_compression;
822         cms = cms_CompressedData_create(comp_nid);
823         if (!cms)
824                 return NULL;

826         if(!(flags & CMS_DETACHED))
827                 CMS_set_detached(cms, 0);

829         if ((flags & CMS_STREAM) || CMS_final(cms, in, NULL, flags))
830                 return cms;

832         CMS_ContentInfo_free(cms);
833         return NULL;
834         }

836 #else

838 int CMS_uncompress(CMS_ContentInfo *cms, BIO *dcont, BIO *out,
839                                                     unsigned int flags)
840         {
841         CMSerr(CMS_F_CMS_UNCOMPRESS, CMS_R_UNSUPPORTED_COMPRESSION_ALGORITHM);
842         return 0;
843         }

845 CMS_ContentInfo *CMS_compress(BIO *in, int comp_nid, unsigned int flags)
846         {
847         CMSerr(CMS_F_CMS_COMPRESS, CMS_R_UNSUPPORTED_COMPRESSION_ALGORITHM);
848         return NULL;
849         }

851 #endif
852 #endif /* ! codereview */
```

```
**********************************************************
    1151 Wed Aug 13 19:52:23 2014
new/usr/src/lib/openssl/libsunw_crypto/comp/c_rle.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/objects.h>
#include <openssl/comp.h>

static int rle_compress_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen);
static int rle_expand_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen);

static COMP_METHOD rle_method={
        NID_rle_compression,
        LN_rle_compression,
        NULL,
        NULL,
        rle_compress_block,
        rle_expand_block,
        NULL,
        NULL,
        };

COMP_METHOD *COMP_rle(void)
        {
        return(&rle_method);
        }

static int rle_compress_block(COMP_CTX *ctx, unsigned char *out,
             unsigned int olen, unsigned char *in, unsigned int ilen)
        {
        /* int i; */

        if (ilen == 0 || olen < (ilen-1))
                {
                /* zzzzzzzzzzzzzzzzzzzzz */
                return(-1);
                }

        *(out++)=0;
        memcpy(out,in,ilen);
        return(ilen+1);
        }

static int rle_expand_block(COMP_CTX *ctx, unsigned char *out,
             unsigned int olen, unsigned char *in, unsigned int ilen)
        {
        int i;

        if (olen < (ilen-1))
                {
                /* zzzzzzzzzzzzzzzzzzzzz */
                return(-1);
                }

        i= *(in++);
        if (i == 0)
                {
                memcpy(out,in,ilen-1);
                }
        return(ilen-1);
        }
```



```
#endif /* ! codereview */
```

```
**********************************************************
   17998 Wed Aug 13 19:52:23 2014
new/usr/src/lib/openssl/libsunw_crypto/comp/c_zlib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/objects.h>
#include <openssl/comp.h>
#include <openssl/err.h>

COMP_METHOD *COMP_zlib(void );

static COMP_METHOD zlib_method_nozlib={
        NID_undef,
        "(undef)",
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL,
        };

#ifndef ZLIB
#undef ZLIB_SHARED
#else

#include <zlib.h>

static int zlib_stateful_init(COMP_CTX *ctx);
static void zlib_stateful_finish(COMP_CTX *ctx);
static int zlib_stateful_compress_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen);
static int zlib_stateful_expand_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen);


/* memory allocations functions for zlib intialization */
static void* zlib_zalloc(void* opaque, unsigned int no, unsigned int size)
{
        void *p;

        p=OPENSSL_malloc(no*size);
        if (p)
                memset(p, 0, no*size);
        return p;
}


static void zlib_zfree(void* opaque, void* address)
{
        OPENSSL_free(address);
}

#if 0
static int zlib_compress_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen);
static int zlib_expand_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen);

static int zz_uncompress(Bytef *dest, uLongf *destLen, const Bytef *source,
        uLong sourceLen);

static COMP_METHOD zlib_stateless_method={
```



```
        NID_zlib_compression,
        LN_zlib_compression,
        NULL,
        NULL,
        zlib_compress_block,
        zlib_expand_block,
        NULL,
        NULL,
        };
#endif

static COMP_METHOD zlib_stateful_method={
        NID_zlib_compression,
        LN_zlib_compression,
        zlib_stateful_init,
        zlib_stateful_finish,
        zlib_stateful_compress_block,
        zlib_stateful_expand_block,
        NULL,
        NULL,
        };

/*
 * When OpenSSL is built on Windows, we do not want to require that
 * the ZLIB.DLL be available in order for the OpenSSL DLLs to
 * work.  Therefore, all ZLIB routines are loaded at run time
 * and we do not link to a .LIB file when ZLIB_SHARED is set.
 */
#if defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_WIN32)
# include <windows.h>
#endif /* !(OPENSSL_SYS_WINDOWS || OPENSSL_SYS_WIN32) */

#ifdef ZLIB_SHARED
#include <openssl/dso.h>

/* Function pointers */
typedef int (*compress_ft)(Bytef *dest,uLongf *destLen,
        const Bytef *source, uLong sourceLen);
typedef int (*inflateEnd_ft)(z_streamp strm);
typedef int (*inflate_ft)(z_streamp strm, int flush);
typedef int (*inflateInit__ft)(z_streamp strm,
        const char * version, int stream_size);
typedef int (*deflateEnd_ft)(z_streamp strm);
typedef int (*deflate_ft)(z_streamp strm, int flush);
typedef int (*deflateInit__ft)(z_streamp strm, int level,
        const char * version, int stream_size);
typedef const char * (*zError__ft)(int err);
static compress_ft      p_compress=NULL;
static inflateEnd_ft    p_inflateEnd=NULL;
static inflate_ft       p_inflate=NULL;
static inflateInit__ft  p_inflateInit_=NULL;
static deflateEnd_ft    p_deflateEnd=NULL;
static deflate_ft       p_deflate=NULL;
static deflateInit__ft  p_deflateInit_=NULL;
static zError__ft       p_zError=NULL;

static int zlib_loaded = 0;     /* only attempt to init func pts once */
static DSO *zlib_dso = NULL;

#define compress                p_compress
#define inflateEnd              p_inflateEnd
#define inflate                 p_inflate
#define inflateInit_            p_inflateInit_
#define deflateEnd              p_deflateEnd
#define deflate                 p_deflate
#define deflateInit_            p_deflateInit_
```

```
#define zError                  p_zError
#endif /* ZLIB_SHARED */

struct zlib_state
        {
        z_stream istream;
        z_stream ostream;
        };

static int zlib_stateful_ex_idx = -1;

static int zlib_stateful_init(COMP_CTX *ctx)
        {
        int err;
        struct zlib_state *state =
                (struct zlib_state *)OPENSSL_malloc(sizeof(struct zlib_state));

        if (state == NULL)
                goto err;

        state->istream.zalloc = zlib_zalloc;
        state->istream.zfree = zlib_zfree;
        state->istream.opaque = Z_NULL;
        state->istream.next_in = Z_NULL;
        state->istream.next_out = Z_NULL;
        state->istream.avail_in = 0;
        state->istream.avail_out = 0;
        err = inflateInit_(&state->istream,
                ZLIB_VERSION, sizeof(z_stream));
        if (err != Z_OK)
                goto err;

        state->ostream.zalloc = zlib_zalloc;
        state->ostream.zfree = zlib_zfree;
        state->ostream.opaque = Z_NULL;
        state->ostream.next_in = Z_NULL;
        state->ostream.next_out = Z_NULL;
        state->ostream.avail_in = 0;
        state->ostream.avail_out = 0;
        err = deflateInit_(&state->ostream,Z_DEFAULT_COMPRESSION,
                ZLIB_VERSION, sizeof(z_stream));
        if (err != Z_OK)
                goto err;

        CRYPTO_new_ex_data(CRYPTO_EX_INDEX_COMP,ctx,&ctx->ex_data);
        CRYPTO_set_ex_data(&ctx->ex_data,zlib_stateful_ex_idx,state);
        return 1;
 err:
        if (state) OPENSSL_free(state);
        return 0;
        }

static void zlib_stateful_finish(COMP_CTX *ctx)
        {
        struct zlib_state *state =
                (struct zlib_state *)CRYPTO_get_ex_data(&ctx->ex_data,
                        zlib_stateful_ex_idx);
        inflateEnd(&state->istream);
        deflateEnd(&state->ostream);
        OPENSSL_free(state);
        CRYPTO_free_ex_data(CRYPTO_EX_INDEX_COMP,ctx,&ctx->ex_data);
        }

static int zlib_stateful_compress_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen)
        {
```



```
        int err = Z_OK;
        struct zlib_state *state =
                (struct zlib_state *)CRYPTO_get_ex_data(&ctx->ex_data,
                        zlib_stateful_ex_idx);

        if (state == NULL)
                return -1;

        state->ostream.next_in = in;
        state->ostream.avail_in = ilen;
        state->ostream.next_out = out;
        state->ostream.avail_out = olen;
        if (ilen > 0)
                err = deflate(&state->ostream, Z_SYNC_FLUSH);
        if (err != Z_OK)
                return -1;
#ifdef DEBUG_ZLIB
        fprintf(stderr,"compress(%4d)->%4d %s\n",
                ilen,olen - state->ostream.avail_out,
                (ilen != olen - state->ostream.avail_out)?"zlib":"clear");
#endif
        return olen - state->ostream.avail_out;
        }

static int zlib_stateful_expand_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen)
        {
        int err = Z_OK;

        struct zlib_state *state =
                (struct zlib_state *)CRYPTO_get_ex_data(&ctx->ex_data,
                        zlib_stateful_ex_idx);

        if (state == NULL)
                return 0;

        state->istream.next_in = in;
        state->istream.avail_in = ilen;
        state->istream.next_out = out;
        state->istream.avail_out = olen;
        if (ilen > 0)
                err = inflate(&state->istream, Z_SYNC_FLUSH);
        if (err != Z_OK)
                return -1;
#ifdef DEBUG_ZLIB
        fprintf(stderr,"expand(%4d)->%4d %s\n",
                ilen,olen - state->istream.avail_out,
                (ilen != olen - state->istream.avail_out)?"zlib":"clear");
#endif
        return olen - state->istream.avail_out;
        }

#if 0
static int zlib_compress_block(COMP_CTX *ctx, unsigned char *out,
        unsigned int olen, unsigned char *in, unsigned int ilen)
        {
        unsigned long l;
        int i;
        int clear=1;

        if (ilen > 128)
                {
                out[0]=1;
                l=olen-1;
                i=compress(&(out[1]),&l,in,(unsigned long)ilen);
                if (i != Z_OK)
```

```
260                         return(-1);
261                 if (ilen > l)
262                         {
263                         clear=0;
264                         l++;
265                         }
266                 }
267         if (clear)
268                 {
269                 out[0]=0;
270                 memcpy(&(out[1]),in,ilen);
271                 l=ilen+1;
272                 }
273 #ifdef DEBUG_ZLIB
274         fprintf(stderr,"compress(%4d)->%4d %s\n",
275                 ilen,(int)l,(clear)?"clear":"zlib");
276 #endif
277         return((int)l);
278         }

280 static int zlib_expand_block(COMP_CTX *ctx, unsigned char *out,
281         unsigned int olen, unsigned char *in, unsigned int ilen)
282         {
283         unsigned long l;
284         int i;

286         if (in[0])
287                 {
288                 l=olen;
289                 i=zz_uncompress(out,&l,&(in[1]),(unsigned long)ilen-1);
290                 if (i != Z_OK)
291                         return(-1);
292                 }
293         else
294                 {
295                 memcpy(out,&(in[1]),ilen-1);
296                 l=ilen-1;
297                 }
298 #ifdef DEBUG_ZLIB
299         fprintf(stderr,"expand  (%4d)->%4d %s\n",
300                 ilen,(int)l,in[0]?"zlib":"clear");
301 #endif
302         return((int)l);
303         }

305 static int zz_uncompress (Bytef *dest, uLongf *destLen, const Bytef *source,
306              uLong sourceLen)
307 {
308     z_stream stream;
309     int err;

311     stream.next_in = (Bytef*)source;
312     stream.avail_in = (uInt)sourceLen;
313     /* Check for source > 64K on 16-bit machine: */
314     if ((uLong)stream.avail_in != sourceLen) return Z_BUF_ERROR;

316     stream.next_out = dest;
317     stream.avail_out = (uInt)*destLen;
318     if ((uLong)stream.avail_out != *destLen) return Z_BUF_ERROR;

320     stream.zalloc = (alloc_func)0;
321     stream.zfree = (free_func)0;

323     err = inflateInit_(&stream,
324             ZLIB_VERSION, sizeof(z_stream));
325     if (err != Z_OK) return err;
```



```
327     err = inflate(&stream, Z_FINISH);
328     if (err != Z_STREAM_END) {
329         inflateEnd(&stream);
330         return err;
331     }
332     *destLen = stream.total_out;

334     err = inflateEnd(&stream);
335     return err;
336 }
337 #endif

339 #endif

341 COMP_METHOD *COMP_zlib(void)
342         {
343         COMP_METHOD *meth = &zlib_method_nozlib;

345 #ifdef ZLIB_SHARED
346         if (!zlib_loaded)
347                 {
348 #if defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_WIN32)
349                 zlib_dso = DSO_load(NULL, "ZLIB1", NULL, 0);
350 #else
351                 zlib_dso = DSO_load(NULL, "z", NULL, 0);
352 #endif
353                 if (zlib_dso != NULL)
354                         {
355                         p_compress
356                                 = (compress_ft) DSO_bind_func(zlib_dso,
357                                         "compress");
358                         p_inflateEnd
359                                 = (inflateEnd_ft) DSO_bind_func(zlib_dso,
360                                         "inflateEnd");
361                         p_inflate
362                                 = (inflate_ft) DSO_bind_func(zlib_dso,
363                                         "inflate");
364                         p_inflateInit_
365                                 = (inflateInit__ft) DSO_bind_func(zlib_dso,
366                                         "inflateInit_");
367                         p_deflateEnd
368                                 = (deflateEnd_ft) DSO_bind_func(zlib_dso,
369                                         "deflateEnd");
370                         p_deflate
371                                 = (deflate_ft) DSO_bind_func(zlib_dso,
372                                         "deflate");
373                         p_deflateInit_
374                                 = (deflateInit__ft) DSO_bind_func(zlib_dso,
375                                         "deflateInit_");
376                         p_zError
377                                 = (zError__ft) DSO_bind_func(zlib_dso,
378                                         "zError");

380                         if (p_compress && p_inflateEnd && p_inflate
381                                 && p_inflateInit_ && p_deflateEnd
382                                 && p_deflate && p_deflateInit_ && p_zError)
383                                 zlib_loaded++;
384                         }
385                 }

387 #endif
388 #ifdef ZLIB_SHARED
389         if (zlib_loaded)
390 #endif
391 #if defined(ZLIB) || defined(ZLIB_SHARED)
```

```
392                 {
393                 /* init zlib_stateful_ex_idx here so that in a multi-process
394                  * application it's enough to intialize openssl before forking
395                  * (idx will be inherited in all the children) */
396                 if (zlib_stateful_ex_idx == -1)
397                         {
398                         CRYPTO_w_lock(CRYPTO_LOCK_COMP);
399                         if (zlib_stateful_ex_idx == -1)
400                                 zlib_stateful_ex_idx =
401                                         CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_
402                                                 0,NULL,NULL,NULL,NULL);
403                         CRYPTO_w_unlock(CRYPTO_LOCK_COMP);
404                         if (zlib_stateful_ex_idx == -1)
405                                 goto err;
406                         }
407
408                 meth = &zlib_stateful_method;
409                 }
410 err:
411 #endif
412
413         return(meth);
414         }
415
416 void COMP_zlib_cleanup(void)
417         {
418 #ifdef ZLIB_SHARED
419         if (zlib_dso)
420                 DSO_free(zlib_dso);
421 #endif
422         }
423
424 #ifdef ZLIB
425
426 /* Zlib based compression/decompression filter BIO */
427
428 typedef struct
429         {
430         unsigned char *ibuf;    /* Input buffer */
431         int ibufsize;           /* Buffer size */
432         z_stream zin;           /* Input decompress context */
433         unsigned char *obuf;    /* Output buffer */
434         int obufsize;           /* Output buffer size */
435         unsigned char *optr;    /* Position in output buffer */
436         int ocount;             /* Amount of data in output buffer */
437         int odone;              /* deflate EOF */
438         int comp_level;         /* Compression level to use */
439         z_stream zout;          /* Output compression context */
440         } BIO_ZLIB_CTX;
441
442 #define ZLIB_DEFAULT_BUFSIZE 1024
443
444 static int bio_zlib_new(BIO *bi);
445 static int bio_zlib_free(BIO *bi);
446 static int bio_zlib_read(BIO *b, char *out, int outl);
447 static int bio_zlib_write(BIO *b, const char *in, int inl);
448 static long bio_zlib_ctrl(BIO *b, int cmd, long num, void *ptr);
449 static long bio_zlib_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp);
450
451 static BIO_METHOD bio_meth_zlib =
452         {
453         BIO_TYPE_COMP,
454         "zlib",
455         bio_zlib_write,
456         bio_zlib_read,
457         NULL,
```



```
458         NULL,
459         bio_zlib_ctrl,
460         bio_zlib_new,
461         bio_zlib_free,
462         bio_zlib_callback_ctrl
463         };
464
465 BIO_METHOD *BIO_f_zlib(void)
466         {
467         return &bio_meth_zlib;
468         }
469
470
471 static int bio_zlib_new(BIO *bi)
472         {
473         BIO_ZLIB_CTX *ctx;
474 #ifdef ZLIB_SHARED
475         (void)COMP_zlib();
476         if (!zlib_loaded)
477                 {
478                 COMPerr(COMP_F_BIO_ZLIB_NEW, COMP_R_ZLIB_NOT_SUPPORTED);
479                 return 0;
480                 }
481 #endif
482         ctx = OPENSSL_malloc(sizeof(BIO_ZLIB_CTX));
483         if(!ctx)
484                 {
485                 COMPerr(COMP_F_BIO_ZLIB_NEW, ERR_R_MALLOC_FAILURE);
486                 return 0;
487                 }
488         ctx->ibuf = NULL;
489         ctx->obuf = NULL;
490         ctx->ibufsize = ZLIB_DEFAULT_BUFSIZE;
491         ctx->obufsize = ZLIB_DEFAULT_BUFSIZE;
492         ctx->zin.zalloc = Z_NULL;
493         ctx->zin.zfree = Z_NULL;
494         ctx->zin.next_in = NULL;
495         ctx->zin.avail_in = 0;
496         ctx->zin.next_out = NULL;
497         ctx->zin.avail_out = 0;
498         ctx->zout.zalloc = Z_NULL;
499         ctx->zout.zfree = Z_NULL;
500         ctx->zout.next_in = NULL;
501         ctx->zout.avail_in = 0;
502         ctx->zout.next_out = NULL;
503         ctx->zout.avail_out = 0;
504         ctx->odone = 0;
505         ctx->comp_level = Z_DEFAULT_COMPRESSION;
506         bi->init = 1;
507         bi->ptr = (char *)ctx;
508         bi->flags = 0;
509         return 1;
510         }
511
512 static int bio_zlib_free(BIO *bi)
513         {
514         BIO_ZLIB_CTX *ctx;
515         if(!bi) return 0;
516         ctx = (BIO_ZLIB_CTX *)bi->ptr;
517         if(ctx->ibuf)
518                 {
519                 /* Destroy decompress context */
520                 inflateEnd(&ctx->zin);
521                 OPENSSL_free(ctx->ibuf);
522                 }
523         if(ctx->obuf)
```

```
524                 {
525                 /* Destroy compress context */
526                 deflateEnd(&ctx->zout);
527                 OPENSSL_free(ctx->obuf);
528                 }
529         OPENSSL_free(ctx);
530         bi->ptr = NULL;
531         bi->init = 0;
532         bi->flags = 0;
533         return 1;
534         }

536 static int bio_zlib_read(BIO *b, char *out, int outl)
537         {
538         BIO_ZLIB_CTX *ctx;
539         int ret;
540         z_stream *zin;
541         if(!out || !outl) return 0;
542         ctx = (BIO_ZLIB_CTX *)b->ptr;
543         zin = &ctx->zin;
544         BIO_clear_retry_flags(b);
545         if(!ctx->ibuf)
546                 {
547                 ctx->ibuf = OPENSSL_malloc(ctx->ibufsize);
548                 if(!ctx->ibuf)
549                         {
550                         COMPerr(COMP_F_BIO_ZLIB_READ, ERR_R_MALLOC_FAILURE);
551                         return 0;
552                         }
553                 inflateInit(zin);
554                 zin->next_in = ctx->ibuf;
555                 zin->avail_in = 0;
556                 }

558         /* Copy output data directly to supplied buffer */
559         zin->next_out = (unsigned char *)out;
560         zin->avail_out = (unsigned int)outl;
561         for(;;)
562                 {
563                 /* Decompress while data available */
564                 while(zin->avail_in)
565                         {
566                         ret = inflate(zin, 0);
567                         if((ret != Z_OK) && (ret != Z_STREAM_END))
568                                 {
569                                 COMPerr(COMP_F_BIO_ZLIB_READ,
570                                                 COMP_R_ZLIB_INFLATE_ERROR);
571                                 ERR_add_error_data(2, "zlib error:",
572                                                 zError(ret));
573                                 return 0;
574                                 }
575                         /* If EOF or we've read everything then return */
576                         if((ret == Z_STREAM_END) || !zin->avail_out)
577                                 return outl - zin->avail_out;
578                         }

580                 /* No data in input buffer try to read some in,
581                  * if an error then return the total data read.
582                  */
583                 ret = BIO_read(b->next_bio, ctx->ibuf, ctx->ibufsize);
584                 if(ret <= 0)
585                         {
586                         /* Total data read */
587                         int tot = outl - zin->avail_out;
588                         BIO_copy_next_retry(b);
589                         if(ret < 0) return (tot > 0) ? tot : ret;
```



```
590                         return tot;
591                         }
592                 zin->avail_in = ret;
593                 zin->next_in = ctx->ibuf;
594                 }
595         }

597 static int bio_zlib_write(BIO *b, const char *in, int inl)
598         {
599         BIO_ZLIB_CTX *ctx;
600         int ret;
601         z_stream *zout;
602         if(!in || !inl) return 0;
603         ctx = (BIO_ZLIB_CTX *)b->ptr;
604         if(ctx->odone) return 0;
605         zout = &ctx->zout;
606         BIO_clear_retry_flags(b);
607         if(!ctx->obuf)
608                 {
609                 ctx->obuf = OPENSSL_malloc(ctx->obufsize);
610                 /* Need error here */
611                 if(!ctx->obuf)
612                         {
613                         COMPerr(COMP_F_BIO_ZLIB_WRITE, ERR_R_MALLOC_FAILURE);
614                         return 0;
615                         }
616                 ctx->optr = ctx->obuf;
617                 ctx->ocount = 0;
618                 deflateInit(zout, ctx->comp_level);
619                 zout->next_out = ctx->obuf;
620                 zout->avail_out = ctx->obufsize;
621                 }
622         /* Obtain input data directly from supplied buffer */
623         zout->next_in = (void *)in;
624         zout->avail_in = inl;
625         for(;;)
626                 {
627                 /* If data in output buffer write it first */
628                 while(ctx->ocount) {
629                         ret = BIO_write(b->next_bio, ctx->optr, ctx->ocount);
630                         if(ret <= 0)
631                                 {
632                                 /* Total data written */
633                                 int tot = inl - zout->avail_in;
634                                 BIO_copy_next_retry(b);
635                                 if(ret < 0) return (tot > 0) ? tot : ret;
636                                 return tot;
637                                 }
638                         ctx->optr += ret;
639                         ctx->ocount -= ret;
640                 }

642                 /* Have we consumed all supplied data? */
643                 if(!zout->avail_in)
644                         return inl;

646                 /* Compress some more */

648                 /* Reset buffer */
649                 ctx->optr = ctx->obuf;
650                 zout->next_out = ctx->obuf;
651                 zout->avail_out = ctx->obufsize;
652                 /* Compress some more */
653                 ret = deflate(zout, 0);
654                 if(ret != Z_OK)
655                         {
```

```
656                         COMPerr(COMP_F_BIO_ZLIB_WRITE,
657                                                 COMP_R_ZLIB_DEFLATE_ERROR);
658                         ERR_add_error_data(2, "zlib error:", zError(ret));
659                         return 0;
660                         }
661                 ctx->ocount = ctx->obufsize - zout->avail_out;
662                 }
663         }

665 static int bio_zlib_flush(BIO *b)
666         {
667         BIO_ZLIB_CTX *ctx;
668         int ret;
669         z_stream *zout;
670         ctx = (BIO_ZLIB_CTX *)b->ptr;
671         /* If no data written or already flush show success */
672         if(!ctx->obuf || (ctx->odone && !ctx->ocount)) return 1;
673         zout = &ctx->zout;
674         BIO_clear_retry_flags(b);
675         /* No more input data */
676         zout->next_in = NULL;
677         zout->avail_in = 0;
678         for(;;)
679                 {
680                 /* If data in output buffer write it first */
681                 while(ctx->ocount)
682                         {
683                         ret = BIO_write(b->next_bio, ctx->optr, ctx->ocount);
684                         if(ret <= 0)
685                                 {
686                                 BIO_copy_next_retry(b);
687                                 return ret;
688                                 }
689                         ctx->optr += ret;
690                         ctx->ocount -= ret;
691                         }
692                 if(ctx->odone) return 1;

694                 /* Compress some more */

696                 /* Reset buffer */
697                 ctx->optr = ctx->obuf;
698                 zout->next_out = ctx->obuf;
699                 zout->avail_out = ctx->obufsize;
700                 /* Compress some more */
701                 ret = deflate(zout, Z_FINISH);
702                 if(ret == Z_STREAM_END) ctx->odone = 1;
703                 else if(ret != Z_OK)
704                         {
705                         COMPerr(COMP_F_BIO_ZLIB_FLUSH,
706                                                 COMP_R_ZLIB_DEFLATE_ERROR);
707                         ERR_add_error_data(2, "zlib error:", zError(ret));
708                         return 0;
709                         }
710                 ctx->ocount = ctx->obufsize - zout->avail_out;
711                 }
712         }

714 static long bio_zlib_ctrl(BIO *b, int cmd, long num, void *ptr)
715         {
716         BIO_ZLIB_CTX *ctx;
717         int ret, *ip;
718         int ibs, obs;
719         if(!b->next_bio) return 0;
720         ctx = (BIO_ZLIB_CTX *)b->ptr;
721         switch (cmd)
```



```
722                 {

724         case BIO_CTRL_RESET:
725                 ctx->ocount = 0;
726                 ctx->odone = 0;
727                 ret = 1;
728                 break;

730         case BIO_CTRL_FLUSH:
731                 ret = bio_zlib_flush(b);
732                 if (ret > 0)
733                         ret = BIO_flush(b->next_bio);
734                 break;

736         case BIO_C_SET_BUFF_SIZE:
737                 ibs = -1;
738                 obs = -1;
739                 if (ptr != NULL)
740                         {
741                         ip = ptr;
742                         if (*ip == 0)
743                                 ibs = (int) num;
744                         else
745                                 obs = (int) num;
746                         }
747                 else
748                         {
749                         ibs = (int)num;
750                         obs = ibs;
751                         }

753                 if (ibs != -1)
754                         {
755                         if (ctx->ibuf)
756                                 {
757                                 OPENSSL_free(ctx->ibuf);
758                                 ctx->ibuf = NULL;
759                                 }
760                         ctx->ibufsize = ibs;
761                         }

763                 if (obs != -1)
764                         {
765                         if (ctx->obuf)
766                                 {
767                                 OPENSSL_free(ctx->obuf);
768                                 ctx->obuf = NULL;
769                                 }
770                         ctx->obufsize = obs;
771                         }
772                 ret = 1;
773                 break;

775         case BIO_C_DO_STATE_MACHINE:
776                 BIO_clear_retry_flags(b);
777                 ret = BIO_ctrl(b->next_bio, cmd, num, ptr);
778                 BIO_copy_next_retry(b);
779                 break;

781         default:
782                 ret = BIO_ctrl(b->next_bio, cmd, num, ptr);
783                 break;

785                 }

787         return ret;
```

```
788         }


791 static long bio_zlib_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
792         {
793         if(!b->next_bio)
794                 return 0;
795         return
796                 BIO_callback_ctrl(b->next_bio, cmd, fp);
797         }

799 #endif
800 #endif /* ! codereview */
```

```
**********************************************************
    3837 Wed Aug 13 19:52:23 2014
new/usr/src/lib/openssl/libsunw_crypto/comp/comp_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/comp/comp_err.c */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/comp.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_COMP,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_COMP,0,reason)

static ERR_STRING_DATA COMP_str_functs[]=
        {
{ERR_FUNC(COMP_F_BIO_ZLIB_FLUSH),       "BIO_ZLIB_FLUSH"},
{ERR_FUNC(COMP_F_BIO_ZLIB_NEW), "BIO_ZLIB_NEW"},
{ERR_FUNC(COMP_F_BIO_ZLIB_READ),        "BIO_ZLIB_READ"},
{ERR_FUNC(COMP_F_BIO_ZLIB_WRITE),       "BIO_ZLIB_WRITE"},
{0,NULL}
        };

static ERR_STRING_DATA COMP_str_reasons[]=
        {
{ERR_REASON(COMP_R_ZLIB_DEFLATE_ERROR)   ,"zlib deflate error"},
{ERR_REASON(COMP_R_ZLIB_INFLATE_ERROR)   ,"zlib inflate error"},
{ERR_REASON(COMP_R_ZLIB_NOT_SUPPORTED)   ,"zlib not supported"},
{0,NULL}
        };

#endif

void ERR_load_COMP_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(COMP_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,COMP_str_functs);
                ERR_load_strings(0,COMP_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    1276 Wed Aug 13 19:52:24 2014
new/usr/src/lib/openssl/libsunw_crypto/comp/comp_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/objects.h>
#include <openssl/comp.h>

COMP_CTX *COMP_CTX_new(COMP_METHOD *meth)
        {
        COMP_CTX *ret;

        if ((ret=(COMP_CTX *)OPENSSL_malloc(sizeof(COMP_CTX))) == NULL)
                {
                /* ZZZZZZZZZZZZZZZZ */
                return(NULL);
                }
        memset(ret,0,sizeof(COMP_CTX));
        ret->meth=meth;
        if ((ret->meth->init != NULL) && !ret->meth->init(ret))
                {
                OPENSSL_free(ret);
                ret=NULL;
                }
        return(ret);
        }

void COMP_CTX_free(COMP_CTX *ctx)
        {
        if(ctx == NULL)
            return;

        if (ctx->meth->finish != NULL)
                ctx->meth->finish(ctx);

        OPENSSL_free(ctx);
        }

int COMP_compress_block(COMP_CTX *ctx, unsigned char *out, int olen,
             unsigned char *in, int ilen)
        {
        int ret;
        if (ctx->meth->compress == NULL)
                {
                /* ZZZZZZZZZZZZZZZZ */
                return(-1);
                }
        ret=ctx->meth->compress(ctx,out,olen,in,ilen);
        if (ret > 0)
                {
                ctx->compress_in+=ilen;
                ctx->compress_out+=ret;
                }
        return(ret);
        }

int COMP_expand_block(COMP_CTX *ctx, unsigned char *out, int olen,
             unsigned char *in, int ilen)
        {
        int ret;

        if (ctx->meth->expand == NULL)
                {
```



```
                /* ZZZZZZZZZZZZZZZZ */
                return(-1);
                }
        ret=ctx->meth->expand(ctx,out,olen,in,ilen);
        if (ret > 0)
                {
                ctx->expand_in+=ilen;
                ctx->expand_out+=ret;
                }
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    8512 Wed Aug 13 19:52:24 2014
new/usr/src/lib/openssl/libsunw_crypto/conf/conf_api.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* conf_api.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Part of the code in here was originally in conf.c, which is now removed */

#ifndef CONF_DEBUG
```



```
# undef NDEBUG /* avoid conflicting definitions */
# define NDEBUG
#endif

#include <assert.h>
#include <stdlib.h>
#include <string.h>
#include <openssl/conf.h>
#include <openssl/conf_api.h>
#include "e_os.h"

static void value_free_hash_doall_arg(CONF_VALUE *a,
                                      LHASH_OF(CONF_VALUE) *conf);
static void value_free_stack_doall(CONF_VALUE *a);
static IMPLEMENT_LHASH_DOALL_ARG_FN(value_free_hash, CONF_VALUE,
                                    LHASH_OF(CONF_VALUE))
static IMPLEMENT_LHASH_DOALL_FN(value_free_stack, CONF_VALUE)

/* Up until OpenSSL 0.9.5a, this was get_section */
CONF_VALUE *_CONF_get_section(const CONF *conf, const char *section)
        {
        CONF_VALUE *v,vv;

        if ((conf == NULL) || (section == NULL)) return(NULL);
        vv.name=NULL;
        vv.section=(char *)section;
        v=lh_CONF_VALUE_retrieve(conf->data,&vv);
        return(v);
        }

/* Up until OpenSSL 0.9.5a, this was CONF_get_section */
STACK_OF(CONF_VALUE) *_CONF_get_section_values(const CONF *conf,
                                               const char *section)
        {
        CONF_VALUE *v;

        v=_CONF_get_section(conf,section);
        if (v != NULL)
                return((STACK_OF(CONF_VALUE) *)v->value);
        else
                return(NULL);
        }

int _CONF_add_string(CONF *conf, CONF_VALUE *section, CONF_VALUE *value)
        {
        CONF_VALUE *v = NULL;
        STACK_OF(CONF_VALUE) *ts;

        ts = (STACK_OF(CONF_VALUE) *)section->value;

        value->section=section->section;
        if (!sk_CONF_VALUE_push(ts,value))
                {
                return 0;
                }

        v = lh_CONF_VALUE_insert(conf->data, value);
        if (v != NULL)
                {
                (void)sk_CONF_VALUE_delete_ptr(ts,v);
                OPENSSL_free(v->name);
                OPENSSL_free(v->value);
                OPENSSL_free(v);
                }
        return 1;
        }
```

```
129 char *_CONF_get_string(const CONF *conf, const char *section, const char *name)
130         {
131         CONF_VALUE *v,vv;
132         char *p;
133
134         if (name == NULL) return(NULL);
135         if (conf != NULL)
136                 {
137                 if (section != NULL)
138                         {
139                         vv.name=(char *)name;
140                         vv.section=(char *)section;
141                         v=lh_CONF_VALUE_retrieve(conf->data,&vv);
142                         if (v != NULL) return(v->value);
143                         if (strcmp(section,"ENV") == 0)
144                                 {
145                                 p=getenv(name);
146                                 if (p != NULL) return(p);
147                                 }
148                         }
149                 vv.section="default";
150                 vv.name=(char *)name;
151                 v=lh_CONF_VALUE_retrieve(conf->data,&vv);
152                 if (v != NULL)
153                         return(v->value);
154                 else
155                         return(NULL);
156                 }
157         else
158                 return(getenv(name));
159         }
160
161 #if 0 /* There's no way to provide error checking with this function, so
162          force implementors of the higher levels to get a string and read
163          the number themselves. */
164 long _CONF_get_number(CONF *conf, char *section, char *name)
165         {
166         char *str;
167         long ret=0;
168
169         str=_CONF_get_string(conf,section,name);
170         if (str == NULL) return(0);
171         for (;;)
172                 {
173                 if (conf->meth->is_number(conf, *str))
174                         ret=ret*10+conf->meth->to_int(conf, *str);
175                 else
176                         return(ret);
177                 str++;
178                 }
179         }
180 #endif
181
182 static unsigned long conf_value_hash(const CONF_VALUE *v)
183         {
184         return (lh_strhash(v->section)<<2)^lh_strhash(v->name);
185         }
186 static IMPLEMENT_LHASH_HASH_FN(conf_value, CONF_VALUE)
187
188 static int conf_value_cmp(const CONF_VALUE *a, const CONF_VALUE *b)
189         {
190         int i;
191
192         if (a->section != b->section)
193                 {
```



```
194                 i=strcmp(a->section,b->section);
195                 if (i) return(i);
196                 }
197
198         if ((a->name != NULL) && (b->name != NULL))
199                 {
200                 i=strcmp(a->name,b->name);
201                 return(i);
202                 }
203         else if (a->name == b->name)
204                 return(0);
205         else
206                 return((a->name == NULL)?-1:1);
207         }
208 static IMPLEMENT_LHASH_COMP_FN(conf_value, CONF_VALUE)
209
210 int _CONF_new_data(CONF *conf)
211         {
212         if (conf == NULL)
213                 {
214                 return 0;
215                 }
216         if (conf->data == NULL)
217                 if ((conf->data = lh_CONF_VALUE_new()) == NULL)
218                         {
219                         return 0;
220                         }
221         return 1;
222         }
223
224 void _CONF_free_data(CONF *conf)
225         {
226         if (conf == NULL || conf->data == NULL) return;
227
228         lh_CONF_VALUE_down_load(conf->data)=0; /* evil thing to make
229                                    * sure the 'OPENSSL_free()' works as
230                                    * expected */
231         lh_CONF_VALUE_doall_arg(conf->data,
232                                 LHASH_DOALL_ARG_FN(value_free_hash),
233                                 LHASH_OF(CONF_VALUE), conf->data);
234
235         /* We now have only 'section' entries in the hash table.
236          * Due to problems with */
237
238         lh_CONF_VALUE_doall(conf->data, LHASH_DOALL_FN(value_free_stack));
239         lh_CONF_VALUE_free(conf->data);
240         }
241
242 static void value_free_hash_doall_arg(CONF_VALUE *a, LHASH_OF(CONF_VALUE) *conf)
243         {
244         if (a->name != NULL)
245                 (void)lh_CONF_VALUE_delete(conf,a);
246         }
247
248 static void value_free_stack_doall(CONF_VALUE *a)
249         {
250         CONF_VALUE *vv;
251         STACK_OF(CONF_VALUE) *sk;
252         int i;
253
254         if (a->name != NULL) return;
255
256         sk=(STACK_OF(CONF_VALUE) *)a->value;
257         for (i=sk_CONF_VALUE_num(sk)-1; i>=0; i--)
258                 {
259                 vv=sk_CONF_VALUE_value(sk,i);
```

```
260                 OPENSSL_free(vv->value);
261                 OPENSSL_free(vv->name);
262                 OPENSSL_free(vv);
263                 }
264         if (sk != NULL) sk_CONF_VALUE_free(sk);
265         OPENSSL_free(a->section);
266         OPENSSL_free(a);
267         }

269 /* Up until OpenSSL 0.9.5a, this was new_section */
270 CONF_VALUE *_CONF_new_section(CONF *conf, const char *section)
271         {
272         STACK_OF(CONF_VALUE) *sk=NULL;
273         int ok=0,i;
274         CONF_VALUE *v=NULL,*vv;

276         if ((sk=sk_CONF_VALUE_new_null()) == NULL)
277                 goto err;
278         if ((v=OPENSSL_malloc(sizeof(CONF_VALUE))) == NULL)
279                 goto err;
280         i=strlen(section)+1;
281         if ((v->section=OPENSSL_malloc(i)) == NULL)
282                 goto err;

284         memcpy(v->section,section,i);
285         v->name=NULL;
286         v->value=(char *)sk;

288         vv=lh_CONF_VALUE_insert(conf->data,v);
289         OPENSSL_assert(vv == NULL);
290         ok=1;
291 err:
292         if (!ok)
293                 {
294                 if (sk != NULL) sk_CONF_VALUE_free(sk);
295                 if (v != NULL) OPENSSL_free(v);
296                 v=NULL;
297                 }
298         return(v);
299         }

301 IMPLEMENT_STACK_OF(CONF_VALUE)
302 #endif /* ! codereview */
```

```
**********************************************************
   16122 Wed Aug 13 19:52:24 2014
new/usr/src/lib/openssl/libsunw_crypto/conf/conf_def.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/conf/conf.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Part of the code in here was originally in conf.c, which is now removed */

#include <stdio.h>
```



```
#include <string.h>
#include "cryptlib.h"
#include <openssl/stack.h>
#include <openssl/lhash.h>
#include <openssl/conf.h>
#include <openssl/conf_api.h>
#include "conf_def.h"
#include <openssl/buffer.h>
#include <openssl/err.h>

static char *eat_ws(CONF *conf, char *p);
static char *eat_alpha_numeric(CONF *conf, char *p);
static void clear_comments(CONF *conf, char *p);
static int str_copy(CONF *conf,char *section,char **to, char *from);
static char *scan_quote(CONF *conf, char *p);
static char *scan_dquote(CONF *conf, char *p);
#define scan_esc(conf,p)        (((IS_EOF((conf),(p)[1]))?((p)+1):((p)+2)))

static CONF *def_create(CONF_METHOD *meth);
static int def_init_default(CONF *conf);
static int def_init_WIN32(CONF *conf);
static int def_destroy(CONF *conf);
static int def_destroy_data(CONF *conf);
static int def_load(CONF *conf, const char *name, long *eline);
static int def_load_bio(CONF *conf, BIO *bp, long *eline);
static int def_dump(const CONF *conf, BIO *bp);
static int def_is_number(const CONF *conf, char c);
static int def_to_int(const CONF *conf, char c);

const char CONF_def_version[]="CONF_def" OPENSSL_VERSION_PTEXT;

static CONF_METHOD default_method = {
        "OpenSSL default",
        def_create,
        def_init_default,
        def_destroy,
        def_destroy_data,
        def_load_bio,
        def_dump,
        def_is_number,
        def_to_int,
        def_load
        };

static CONF_METHOD WIN32_method = {
        "WIN32",
        def_create,
        def_init_WIN32,
        def_destroy,
        def_destroy_data,
        def_load_bio,
        def_dump,
        def_is_number,
        def_to_int,
        def_load
        };

CONF_METHOD *NCONF_default()
        {
        return &default_method;
        }
CONF_METHOD *NCONF_WIN32()
        {
        return &WIN32_method;
        }
```

```
128 static CONF *def_create(CONF_METHOD *meth)
129         {
130         CONF *ret;

132         ret = OPENSSL_malloc(sizeof(CONF) + sizeof(unsigned short *));
133         if (ret)
134                 if (meth->init(ret) == 0)
135                         {
136                         OPENSSL_free(ret);
137                         ret = NULL;
138                         }
139         return ret;
140         }

142 static int def_init_default(CONF *conf)
143         {
144         if (conf == NULL)
145                 return 0;

147         conf->meth = &default_method;
148         conf->meth_data = CONF_type_default;
149         conf->data = NULL;

151         return 1;
152         }

154 static int def_init_WIN32(CONF *conf)
155         {
156         if (conf == NULL)
157                 return 0;

159         conf->meth = &WIN32_method;
160         conf->meth_data = (void *)CONF_type_win32;
161         conf->data = NULL;

163         return 1;
164         }

166 static int def_destroy(CONF *conf)
167         {
168         if (def_destroy_data(conf))
169                 {
170                 OPENSSL_free(conf);
171                 return 1;
172                 }
173         return 0;
174         }

176 static int def_destroy_data(CONF *conf)
177         {
178         if (conf == NULL)
179                 return 0;
180         _CONF_free_data(conf);
181         return 1;
182         }

184 static int def_load(CONF *conf, const char *name, long *line)
185         {
186         int ret;
187         BIO *in=NULL;

189 #ifdef OPENSSL_SYS_VMS
190         in=BIO_new_file(name, "r");
191 #else
192         in=BIO_new_file(name, "rb");
193 #endif
```



```
194         if (in == NULL)
195                 {
196                 if (ERR_GET_REASON(ERR_peek_last_error()) == BIO_R_NO_SUCH_FILE)
197                         CONFerr(CONF_F_DEF_LOAD,CONF_R_NO_SUCH_FILE);
198                 else
199                         CONFerr(CONF_F_DEF_LOAD,ERR_R_SYS_LIB);
200                 return 0;
201                 }

203         ret = def_load_bio(conf, in, line);
204         BIO_free(in);

206         return ret;
207         }

209 static int def_load_bio(CONF *conf, BIO *in, long *line)
210         {
211 /* The macro BUFSIZE conflicts with a system macro in VxWorks */
212 #define CONFBUFSIZE     512
213         int bufnum=0,i,ii;
214         BUF_MEM *buff=NULL;
215         char *s,*p,*end;
216         int again;
217         long eline=0;
218         char btmp[DECIMAL_SIZE(eline)+1];
219         CONF_VALUE *v=NULL,*tv;
220         CONF_VALUE *sv=NULL;
221         char *section=NULL,*buf;
222         char *start,*psection,*pname;
223         void *h = (void *)(conf->data);

225         if ((buff=BUF_MEM_new()) == NULL)
226                 {
227                 CONFerr(CONF_F_DEF_LOAD_BIO,ERR_R_BUF_LIB);
228                 goto err;
229                 }

231         section=(char *)OPENSSL_malloc(10);
232         if (section == NULL)
233                 {
234                 CONFerr(CONF_F_DEF_LOAD_BIO,ERR_R_MALLOC_FAILURE);
235                 goto err;
236                 }
237         BUF_strlcpy(section,"default",10);

239         if (_CONF_new_data(conf) == 0)
240                 {
241                 CONFerr(CONF_F_DEF_LOAD_BIO,ERR_R_MALLOC_FAILURE);
242                 goto err;
243                 }

245         sv=_CONF_new_section(conf,section);
246         if (sv == NULL)
247                 {
248                 CONFerr(CONF_F_DEF_LOAD_BIO,
249                                         CONF_R_UNABLE_TO_CREATE_NEW_SECTION);
250                 goto err;
251                 }

253         bufnum=0;
254         again=0;
255         for (;;)
256                 {
257                 if (!BUF_MEM_grow(buff,bufnum+CONFBUFSIZE))
258                         {
259                         CONFerr(CONF_F_DEF_LOAD_BIO,ERR_R_BUF_LIB);
```

```
260                         goto err;
261                         }
262                 p= &(buff->data[bufnum]);
263                 *p='\0';
264                 BIO_gets(in, p, CONFBUFSIZE-1);
265                 p[CONFBUFSIZE-1]='\0';
266                 ii=i=strlen(p);
267                 if (i == 0 && !again) break;
268                 again=0;
269                 while (i > 0)
270                         {
271                         if ((p[i-1] != '\r') && (p[i-1] != '\n'))
272                                 break;
273                         else
274                                 i--;
275                         }
276                 /* we removed some trailing stuff so there is a new
277                  * line on the end. */
278                 if (ii && i == ii)
279                         again=1; /* long line */
280                 else
281                         {
282                         p[i]='\0';
283                         eline++; /* another input line */
284                         }

286                 /* we now have a line with trailing \r\n removed */

288                 /* i is the number of bytes */
289                 bufnum+=i;

291                 v=NULL;
292                 /* check for line continuation */
293                 if (bufnum >= 1)
294                         {
295                         /* If we have bytes and the last char '\\' and
296                          * second last char is not '\\' */
297                         p= &(buff->data[bufnum-1]);
298                         if (IS_ESC(conf,p[0]) &&
299                                 ((bufnum <= 1) || !IS_ESC(conf,p[-1])))
300                                 {
301                                 bufnum--;
302                                 again=1;
303                                 }
304                         }
305                 if (again) continue;
306                 bufnum=0;
307                 buf=buff->data;

309                 clear_comments(conf, buf);
310                 s=eat_ws(conf, buf);
311                 if (IS_EOF(conf,*s)) continue; /* blank line */
312                 if (*s == '[')
313                         {
314                         char *ss;

316                         s++;
317                         start=eat_ws(conf, s);
318                         ss=start;
319 again:
320                         end=eat_alpha_numeric(conf, ss);
321                         p=eat_ws(conf, end);
322                         if (*p != ']')
323                                 {
324                                 if (*p != '\0' && ss != p)
325                                         {
```



```
326                                         ss=p;
327                                         goto again;
328                                         }
329                                 CONFerr(CONF_F_DEF_LOAD_BIO,
330                                         CONF_R_MISSING_CLOSE_SQUARE_BRACKET);
331                                 goto err;
332                                 }
333                         *end='\0';
334                         if (!str_copy(conf,NULL,&section,start)) goto err;
335                         if ((sv=_CONF_get_section(conf,section)) == NULL)
336                                 sv=_CONF_new_section(conf,section);
337                         if (sv == NULL)
338                                 {
339                                 CONFerr(CONF_F_DEF_LOAD_BIO,
340                                         CONF_R_UNABLE_TO_CREATE_NEW_SECTION);
341                                 goto err;
342                                 }
343                         continue;
344                         }
345                 else
346                         {
347                         pname=s;
348                         psection=NULL;
349                         end=eat_alpha_numeric(conf, s);
350                         if ((end[0] == ':') && (end[1] == ':'))
351                                 {
352                                 *end='\0';
353                                 end+=2;
354                                 psection=pname;
355                                 pname=end;
356                                 end=eat_alpha_numeric(conf, end);
357                                 }
358                         p=eat_ws(conf, end);
359                         if (*p != '=')
360                                 {
361                                 CONFerr(CONF_F_DEF_LOAD_BIO,
362                                                 CONF_R_MISSING_EQUAL_SIGN);
363                                 goto err;
364                                 }
365                         *end='\0';
366                         p++;
367                         start=eat_ws(conf, p);
368                         while (!IS_EOF(conf,*p))
369                                 p++;
370                         p--;
371                         while ((p != start) && (IS_WS(conf,*p)))
372                                 p--;
373                         p++;
374                         *p='\0';

376                         if (!(v=(CONF_VALUE *)OPENSSL_malloc(sizeof(CONF_VALUE))
377                                 {
378                                 CONFerr(CONF_F_DEF_LOAD_BIO,
379                                                         ERR_R_MALLOC_FAILURE);
380                                 goto err;
381                                 }
382                         if (psection == NULL) psection=section;
383                         v->name=(char *)OPENSSL_malloc(strlen(pname)+1);
384                         v->value=NULL;
385                         if (v->name == NULL)
386                                 {
387                                 CONFerr(CONF_F_DEF_LOAD_BIO,
388                                                         ERR_R_MALLOC_FAILURE);
389                                 goto err;
390                                 }
391                         BUF_strlcpy(v->name,pname,strlen(pname)+1);
```

```
392                         if (!str_copy(conf,psection,&(v->value),start)) goto err

394                         if (strcmp(psection,section) != 0)
395                                 {
396                                 if ((tv=_CONF_get_section(conf,psection))
397                                         == NULL)
398                                         tv=_CONF_new_section(conf,psection);
399                                 if (tv == NULL)
400                                         {
401                                         CONFerr(CONF_F_DEF_LOAD_BIO,
402                                            CONF_R_UNABLE_TO_CREATE_NEW_SECTION);
403                                         goto err;
404                                         }
405                                 }
406                         else
407                                 tv=sv;
408 #if 1
409                         if (_CONF_add_string(conf, tv, v) == 0)
410                                 {
411                                 CONFerr(CONF_F_DEF_LOAD_BIO,
412                                                         ERR_R_MALLOC_FAILURE);
413                                 goto err;
414                                 }
415 #else
416                         v->section=tv->section;
417                         if (!sk_CONF_VALUE_push(ts,v))
418                                 {
419                                 CONFerr(CONF_F_DEF_LOAD_BIO,
420                                                         ERR_R_MALLOC_FAILURE);
421                                 goto err;
422                                 }
423                         vv=(CONF_VALUE *)lh_insert(conf->data,v);
424                         if (vv != NULL)
425                                 {
426                                 sk_CONF_VALUE_delete_ptr(ts,vv);
427                                 OPENSSL_free(vv->name);
428                                 OPENSSL_free(vv->value);
429                                 OPENSSL_free(vv);
430                                 }
431 #endif
432                         v=NULL;
433                         }
434                 }
435         if (buff != NULL) BUF_MEM_free(buff);
436         if (section != NULL) OPENSSL_free(section);
437         return(1);
438 err:
439         if (buff != NULL) BUF_MEM_free(buff);
440         if (section != NULL) OPENSSL_free(section);
441         if (line != NULL) *line=eline;
442         BIO_snprintf(btmp,sizeof btmp,"%ld",eline);
443         ERR_add_error_data(2,"line ",btmp);
444         if ((h != conf->data) && (conf->data != NULL))
445                 {
446                 CONF_free(conf->data);
447                 conf->data=NULL;
448                 }
449         if (v != NULL)
450                 {
451                 if (v->name != NULL) OPENSSL_free(v->name);
452                 if (v->value != NULL) OPENSSL_free(v->value);
453                 if (v != NULL) OPENSSL_free(v);
454                 }
455         return(0);
456         }
```



```
458 static void clear_comments(CONF *conf, char *p)
459         {
460         for (;;)
461                 {
462                 if (IS_FCOMMENT(conf,*p))
463                         {
464                         *p='\0';
465                         return;
466                         }
467                 if (!IS_WS(conf,*p))
468                         {
469                         break;
470                         }
471                 p++;
472                 }

474         for (;;)
475                 {
476                 if (IS_COMMENT(conf,*p))
477                         {
478                         *p='\0';
479                         return;
480                         }
481                 if (IS_DQUOTE(conf,*p))
482                         {
483                         p=scan_dquote(conf, p);
484                         continue;
485                         }
486                 if (IS_QUOTE(conf,*p))
487                         {
488                         p=scan_quote(conf, p);
489                         continue;
490                         }
491                 if (IS_ESC(conf,*p))
492                         {
493                         p=scan_esc(conf,p);
494                         continue;
495                         }
496                 if (IS_EOF(conf,*p))
497                         return;
498                 else
499                         p++;
500                 }
501         }

503 static int str_copy(CONF *conf, char *section, char **pto, char *from)
504         {
505         int q,r,rr=0,to=0,len=0;
506         char *s,*e,*rp,*p,*rrp,*np,*cp,v;
507         BUF_MEM *buf;

509         if ((buf=BUF_MEM_new()) == NULL) return(0);

511         len=strlen(from)+1;
512         if (!BUF_MEM_grow(buf,len)) goto err;

514         for (;;)
515                 {
516                 if (IS_QUOTE(conf,*from))
517                         {
518                         q= *from;
519                         from++;
520                         while (!IS_EOF(conf,*from) && (*from != q))
521                                 {
522                                 if (IS_ESC(conf,*from))
523                                         {
```

```
524                                     from++;
525                                     if (IS_EOF(conf,*from)) break;
526                                     }
527                             buf->data[to++]= *(from++);
528                             }
529                     if (*from == q) from++;
530                     }
531             else if (IS_DQUOTE(conf,*from))
532                     {
533                     q= *from;
534                     from++;
535                     while (!IS_EOF(conf,*from))
536                             {
537                             if (*from == q)
538                                     {
539                                     if (*(from+1) == q)
540                                             {
541                                             from++;
542                                             }
543                                     else
544                                             {
545                                             break;
546                                             }
547                                     }
548                             buf->data[to++]= *(from++);
549                             }
550                     if (*from == q) from++;
551                     }
552             else if (IS_ESC(conf,*from))
553                     {
554                     from++;
555                     v= *(from++);
556                     if (IS_EOF(conf,v)) break;
557                     else if (v == 'r') v='\r';
558                     else if (v == 'n') v='\n';
559                     else if (v == 'b') v='\b';
560                     else if (v == 't') v='\t';
561                     buf->data[to++]= v;
562                     }
563             else if (IS_EOF(conf,*from))
564                     break;
565             else if (*from == '$')
566                     {
567                     /* try to expand it */
568                     rrp=NULL;
569                     s= &(from[1]);
570                     if (*s == '{')
571                             q='}';
572                     else if (*s == '(')
573                             q=')';
574                     else q=0;

576                     if (q) s++;
577                     cp=section;
578                     e=np=s;
579                     while (IS_ALPHA_NUMERIC(conf,*e))
580                             e++;
581                     if ((e[0] == ':') && (e[1] == ':'))
582                             {
583                             cp=np;
584                             rrp=e;
585                             rr= *e;
586                             *rrp='\0';
587                             e+=2;
588                             np=e;
589                             while (IS_ALPHA_NUMERIC(conf,*e))
```



```
590                                     e++;
591                             }
592                     r= *e;
593                     *e='\0';
594                     rp=e;
595                     if (q)
596                             {
597                             if (r != q)
598                                     {
599                                     CONFerr(CONF_F_STR_COPY,CONF_R_NO_CLOSE_
600                                     goto err;
601                                     }
602                             e++;
603                             }
604                     /* So at this point we have
605                      * np which is the start of the name string which is
606                      *   '\0' terminated.
607                      * cp which is the start of the section string which is
608                      *   '\0' terminated.
609                      * e is the 'next point after'.
610                      * r and rr are the chars replaced by the '\0'
611                      * rp and rrp is where 'r' and 'rr' came from.
612                      */
613                     p=_CONF_get_string(conf,cp,np);
614                     if (rrp != NULL) *rrp=rr;
615                     *rp=r;
616                     if (p == NULL)
617                             {
618                             CONFerr(CONF_F_STR_COPY,CONF_R_VARIABLE_HAS_NO_V
619                             goto err;
620                             }
621                     BUF_MEM_grow_clean(buf,(strlen(p)+buf->length-(e-from)))
622                     while (*p)
623                             buf->data[to++]= *(p++);

625                     /* Since we change the pointer 'from', we also have
626                        to change the perceived length of the string it
627                        points at.  /RL */
628                     len -= e-from;
629                     from=e;

631                     /* In case there were no braces or parenthesis around
632                        the variable reference, we have to put back the
633                        character that was replaced with a '\0'.  /RL */
634                     *rp = r;
635                     }
636             else
637                     buf->data[to++]= *(from++);
638             }
639         buf->data[to]='\0';
640         if (*pto != NULL) OPENSSL_free(*pto);
641         *pto=buf->data;
642         OPENSSL_free(buf);
643         return(1);
644 err:
645         if (buf != NULL) BUF_MEM_free(buf);
646         return(0);
647         }

649 static char *eat_ws(CONF *conf, char *p)
650         {
651         while (IS_WS(conf,*p) && (!IS_EOF(conf,*p)))
652                 p++;
653         return(p);
654         }
```

```
656 static char *eat_alpha_numeric(CONF *conf, char *p)
657         {
658         for (;;)
659                 {
660                 if (IS_ESC(conf,*p))
661                         {
662                         p=scan_esc(conf,p);
663                         continue;
664                         }
665                 if (!IS_ALPHA_NUMERIC_PUNCT(conf,*p))
666                         return(p);
667                 p++;
668                 }
669         }

671 static char *scan_quote(CONF *conf, char *p)
672         {
673         int q= *p;

675         p++;
676         while (!(IS_EOF(conf,*p)) && (*p != q))
677                 {
678                 if (IS_ESC(conf,*p))
679                         {
680                         p++;
681                         if (IS_EOF(conf,*p)) return(p);
682                         }
683                 p++;
684                 }
685         if (*p == q) p++;
686         return(p);
687         }


690 static char *scan_dquote(CONF *conf, char *p)
691         {
692         int q= *p;

694         p++;
695         while (!(IS_EOF(conf,*p)))
696                 {
697                 if (*p == q)
698                         {
699                         if (*(p+1) == q)
700                                 {
701                                 p++;
702                                 }
703                         else
704                                 {
705                                 break;
706                                 }
707                         }
708                 p++;
709                 }
710         if (*p == q) p++;
711         return(p);
712         }

714 static void dump_value_doall_arg(CONF_VALUE *a, BIO *out)
715         {
716         if (a->name)
717                 BIO_printf(out, "[%s] %s=%s\n", a->section, a->name, a->value);
718         else
719                 BIO_printf(out, "[[%s]]\n", a->section);
720         }
```



```
722 static IMPLEMENT_LHASH_DOALL_ARG_FN(dump_value, CONF_VALUE, BIO)

724 static int def_dump(const CONF *conf, BIO *out)
725         {
726         lh_CONF_VALUE_doall_arg(conf->data, LHASH_DOALL_ARG_FN(dump_value),
727                                 BIO, out);
728         return 1;
729         }

731 static int def_is_number(const CONF *conf, char c)
732         {
733         return IS_NUMBER(conf,c);
734         }

736 static int def_to_int(const CONF *conf, char c)
737         {
738         return c - '0';
739         }
740 #endif /* ! codereview */
```

```
**********************************************************
    5637 Wed Aug 13 19:52:24 2014
new/usr/src/lib/openssl/libsunw_crypto/conf/conf_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/conf/conf_err.c */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/conf.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_CONF,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_CONF,0,reason)

static ERR_STRING_DATA CONF_str_functs[]=
        {
{ERR_FUNC(CONF_F_CONF_DUMP_FP), "CONF_dump_fp"},
{ERR_FUNC(CONF_F_CONF_LOAD),    "CONF_load"},
{ERR_FUNC(CONF_F_CONF_LOAD_BIO),        "CONF_load_bio"},
{ERR_FUNC(CONF_F_CONF_LOAD_FP), "CONF_load_fp"},
{ERR_FUNC(CONF_F_CONF_MODULES_LOAD),    "CONF_modules_load"},
{ERR_FUNC(CONF_F_CONF_PARSE_LIST),      "CONF_parse_list"},
{ERR_FUNC(CONF_F_DEF_LOAD),     "DEF_LOAD"},
{ERR_FUNC(CONF_F_DEF_LOAD_BIO), "DEF_LOAD_BIO"},
{ERR_FUNC(CONF_F_MODULE_INIT),  "MODULE_INIT"},
{ERR_FUNC(CONF_F_MODULE_LOAD_DSO),      "MODULE_LOAD_DSO"},
{ERR_FUNC(CONF_F_MODULE_RUN),   "MODULE_RUN"},
{ERR_FUNC(CONF_F_NCONF_DUMP_BIO),       "NCONF_dump_bio"},
{ERR_FUNC(CONF_F_NCONF_DUMP_FP),        "NCONF_dump_fp"},
{ERR_FUNC(CONF_F_NCONF_GET_NUMBER),     "NCONF_get_number"},
{ERR_FUNC(CONF_F_NCONF_GET_NUMBER_E),   "NCONF_get_number_e"},
{ERR_FUNC(CONF_F_NCONF_GET_SECTION),    "NCONF_get_section"},
{ERR_FUNC(CONF_F_NCONF_GET_STRING),     "NCONF_get_string"},
{ERR_FUNC(CONF_F_NCONF_LOAD),   "NCONF_load"},
{ERR_FUNC(CONF_F_NCONF_LOAD_BIO),       "NCONF_load_bio"},
{ERR_FUNC(CONF_F_NCONF_LOAD_FP),        "NCONF_load_fp"},
{ERR_FUNC(CONF_F_NCONF_NEW),    "NCONF_new"},
{ERR_FUNC(CONF_F_STR_COPY),     "STR_COPY"},
{0,NULL}
        };

static ERR_STRING_DATA CONF_str_reasons[]=
        {
{ERR_REASON(CONF_R_ERROR_LOADING_DSO)    ,"error loading dso"},
{ERR_REASON(CONF_R_LIST_CANNOT_BE_NULL)  ,"list cannot be null"},
{ERR_REASON(CONF_R_MISSING_CLOSE_SQUARE_BRACKET),"missing close square bracket"},
{ERR_REASON(CONF_R_MISSING_EQUAL_SIGN)   ,"missing equal sign"},
{ERR_REASON(CONF_R_MISSING_FINISH_FUNCTION),"missing finish function"},
{ERR_REASON(CONF_R_MISSING_INIT_FUNCTION),"missing init function"},
{ERR_REASON(CONF_R_MODULE_INITIALIZATION_ERROR),"module initialization error"},
{ERR_REASON(CONF_R_NO_CLOSE_BRACE)       ,"no close brace"},
{ERR_REASON(CONF_R_NO_CONF)              ,"no conf"},
{ERR_REASON(CONF_R_NO_CONF_OR_ENVIRONMENT_VARIABLE),"no conf or environment vari
{ERR_REASON(CONF_R_NO_SECTION)           ,"no section"},
{ERR_REASON(CONF_R_NO_SUCH_FILE)         ,"no such file"},
{ERR_REASON(CONF_R_NO_VALUE)             ,"no value"},
{ERR_REASON(CONF_R_UNABLE_TO_CREATE_NEW_SECTION),"unable to create new section"}
{ERR_REASON(CONF_R_UNKNOWN_MODULE_NAME)  ,"unknown module name"},
{ERR_REASON(CONF_R_VARIABLE_HAS_NO_VALUE),"variable has no value"},
{0,NULL}
        };

#endif

void ERR_load_CONF_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(CONF_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,CONF_str_functs);
```

```
128                 ERR_load_strings(0,CONF_str_reasons);
129                 }
130 #endif
131         }
132 #endif /* ! codereview */
```

```
**********************************************************
    9522 Wed Aug 13 19:52:24 2014
new/usr/src/lib/openssl/libsunw_crypto/conf/conf_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* conf_lib.c */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/crypto.h>
#include <openssl/err.h>
```



```
#include <openssl/conf.h>
#include <openssl/conf_api.h>
#include <openssl/lhash.h>

const char CONF_version[]="CONF" OPENSSL_VERSION_PTEXT;

static CONF_METHOD *default_CONF_method=NULL;

/* Init a 'CONF' structure from an old LHASH */

void CONF_set_nconf(CONF *conf, LHASH_OF(CONF_VALUE) *hash)
        {
        if (default_CONF_method == NULL)
                default_CONF_method = NCONF_default();

        default_CONF_method->init(conf);
        conf->data = hash;
        }

/* The following section contains the "CONF classic" functions,
   rewritten in terms of the new CONF interface. */

int CONF_set_default_method(CONF_METHOD *meth)
        {
        default_CONF_method = meth;
        return 1;
        }

LHASH_OF(CONF_VALUE) *CONF_load(LHASH_OF(CONF_VALUE) *conf, const char *file,
                                long *eline)
        {
        LHASH_OF(CONF_VALUE) *ltmp;
        BIO *in=NULL;

#ifdef OPENSSL_SYS_VMS
        in=BIO_new_file(file, "r");
#else
        in=BIO_new_file(file, "rb");
#endif
        if (in == NULL)
                {
                CONFerr(CONF_F_CONF_LOAD,ERR_R_SYS_LIB);
                return NULL;
                }

        ltmp = CONF_load_bio(conf, in, eline);
        BIO_free(in);

        return ltmp;
        }

#ifndef OPENSSL_NO_FP_API
LHASH_OF(CONF_VALUE) *CONF_load_fp(LHASH_OF(CONF_VALUE) *conf, FILE *fp,
                                   long *eline)
        {
        BIO *btmp;
        LHASH_OF(CONF_VALUE) *ltmp;
        if(!(btmp = BIO_new_fp(fp, BIO_NOCLOSE))) {
                CONFerr(CONF_F_CONF_LOAD_FP,ERR_R_BUF_LIB);
                return NULL;
        }
        ltmp = CONF_load_bio(conf, btmp, eline);
        BIO_free(btmp);
        return ltmp;
        }
#endif
```

```
LHASH_OF(CONF_VALUE) *CONF_load_bio(LHASH_OF(CONF_VALUE) *conf, BIO *bp,
                                    long *eline)
        {
        CONF ctmp;
        int ret;

        CONF_set_nconf(&ctmp, conf);

        ret = NCONF_load_bio(&ctmp, bp, eline);
        if (ret)
                return ctmp.data;
        return NULL;
        }

STACK_OF(CONF_VALUE) *CONF_get_section(LHASH_OF(CONF_VALUE) *conf,
                                       const char *section)
        {
        if (conf == NULL)
                {
                return NULL;
                }
        else
                {
                CONF ctmp;
                CONF_set_nconf(&ctmp, conf);
                return NCONF_get_section(&ctmp, section);
                }
        }

char *CONF_get_string(LHASH_OF(CONF_VALUE) *conf,const char *group,
                      const char *name)
        {
        if (conf == NULL)
                {
                return NCONF_get_string(NULL, group, name);
                }
        else
                {
                CONF ctmp;
                CONF_set_nconf(&ctmp, conf);
                return NCONF_get_string(&ctmp, group, name);
                }
        }

long CONF_get_number(LHASH_OF(CONF_VALUE) *conf,const char *group,
                     const char *name)
        {
        int status;
        long result = 0;

        if (conf == NULL)
                {
                status = NCONF_get_number_e(NULL, group, name, &result);
                }
        else
                {
                CONF ctmp;
                CONF_set_nconf(&ctmp, conf);
                status = NCONF_get_number_e(&ctmp, group, name, &result);
                }

        if (status == 0)
                {
                /* This function does not believe in errors... */
                ERR_clear_error();
```



```
                }
        return result;
        }

void CONF_free(LHASH_OF(CONF_VALUE) *conf)
        {
        CONF ctmp;
        CONF_set_nconf(&ctmp, conf);
        NCONF_free_data(&ctmp);
        }

#ifndef OPENSSL_NO_FP_API
int CONF_dump_fp(LHASH_OF(CONF_VALUE) *conf, FILE *out)
        {
        BIO *btmp;
        int ret;

        if(!(btmp = BIO_new_fp(out, BIO_NOCLOSE))) {
                CONFerr(CONF_F_CONF_DUMP_FP,ERR_R_BUF_LIB);
                return 0;
        }
        ret = CONF_dump_bio(conf, btmp);
        BIO_free(btmp);
        return ret;
        }
#endif

int CONF_dump_bio(LHASH_OF(CONF_VALUE) *conf, BIO *out)
        {
        CONF ctmp;
        CONF_set_nconf(&ctmp, conf);
        return NCONF_dump_bio(&ctmp, out);
        }

/* The following section contains the "New CONF" functions.  They are
   completely centralised around a new CONF structure that may contain
   basically anything, but at least a method pointer and a table of data.
   These functions are also written in terms of the bridge functions used
   by the "CONF classic" functions, for consistency.  */

CONF *NCONF_new(CONF_METHOD *meth)
        {
        CONF *ret;

        if (meth == NULL)
                meth = NCONF_default();

        ret = meth->create(meth);
        if (ret == NULL)
                {
                CONFerr(CONF_F_NCONF_NEW,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }

        return ret;
        }

void NCONF_free(CONF *conf)
        {
        if (conf == NULL)
                return;
        conf->meth->destroy(conf);
        }

void NCONF_free_data(CONF *conf)
        {
```

```
260         if (conf == NULL)
261                 return;
262         conf->meth->destroy_data(conf);
263         }

265 int NCONF_load(CONF *conf, const char *file, long *eline)
266         {
267         if (conf == NULL)
268                 {
269                 CONFerr(CONF_F_NCONF_LOAD,CONF_R_NO_CONF);
270                 return 0;
271                 }

273         return conf->meth->load(conf, file, eline);
274         }

276 #ifndef OPENSSL_NO_FP_API
277 int NCONF_load_fp(CONF *conf, FILE *fp,long *eline)
278         {
279         BIO *btmp;
280         int ret;
281         if(!(btmp = BIO_new_fp(fp, BIO_NOCLOSE)))
282                 {
283                 CONFerr(CONF_F_NCONF_LOAD_FP,ERR_R_BUF_LIB);
284                 return 0;
285                 }
286         ret = NCONF_load_bio(conf, btmp, eline);
287         BIO_free(btmp);
288         return ret;
289         }
290 #endif

292 int NCONF_load_bio(CONF *conf, BIO *bp,long *eline)
293         {
294         if (conf == NULL)
295                 {
296                 CONFerr(CONF_F_NCONF_LOAD_BIO,CONF_R_NO_CONF);
297                 return 0;
298                 }

300         return conf->meth->load_bio(conf, bp, eline);
301         }

303 STACK_OF(CONF_VALUE) *NCONF_get_section(const CONF *conf,const char *section)
304         {
305         if (conf == NULL)
306                 {
307                 CONFerr(CONF_F_NCONF_GET_SECTION,CONF_R_NO_CONF);
308                 return NULL;
309                 }

311         if (section == NULL)
312                 {
313                 CONFerr(CONF_F_NCONF_GET_SECTION,CONF_R_NO_SECTION);
314                 return NULL;
315                 }

317         return _CONF_get_section_values(conf, section);
318         }

320 char *NCONF_get_string(const CONF *conf,const char *group,const char *name)
321         {
322         char *s = _CONF_get_string(conf, group, name);

324         /* Since we may get a value from an environment variable even
325            if conf is NULL, let's check the value first */
```



```
326         if (s) return s;

328         if (conf == NULL)
329                 {
330                 CONFerr(CONF_F_NCONF_GET_STRING,
331                         CONF_R_NO_CONF_OR_ENVIRONMENT_VARIABLE);
332                 return NULL;
333                 }
334         CONFerr(CONF_F_NCONF_GET_STRING,
335                 CONF_R_NO_VALUE);
336         ERR_add_error_data(4,"group=",group," name=",name);
337         return NULL;
338         }

340 int NCONF_get_number_e(const CONF *conf,const char *group,const char *name,
341                        long *result)
342         {
343         char *str;

345         if (result == NULL)
346                 {
347                 CONFerr(CONF_F_NCONF_GET_NUMBER_E,ERR_R_PASSED_NULL_PARAMETER);
348                 return 0;
349                 }

351         str = NCONF_get_string(conf,group,name);

353         if (str == NULL)
354                 return 0;

356         for (*result = 0;conf->meth->is_number(conf, *str);)
357                 {
358                 *result = (*result)*10 + conf->meth->to_int(conf, *str);
359                 str++;
360                 }

362         return 1;
363         }

365 #ifndef OPENSSL_NO_FP_API
366 int NCONF_dump_fp(const CONF *conf, FILE *out)
367         {
368         BIO *btmp;
369         int ret;
370         if(!(btmp = BIO_new_fp(out, BIO_NOCLOSE))) {
371                 CONFerr(CONF_F_NCONF_DUMP_FP,ERR_R_BUF_LIB);
372                 return 0;
373         }
374         ret = NCONF_dump_bio(conf, btmp);
375         BIO_free(btmp);
376         return ret;
377         }
378 #endif

380 int NCONF_dump_bio(const CONF *conf, BIO *out)
381         {
382         if (conf == NULL)
383                 {
384                 CONFerr(CONF_F_NCONF_DUMP_BIO,CONF_R_NO_CONF);
385                 return 0;
386                 }

388         return conf->meth->dump(conf, out);
389         }
```

```
392 /* This function should be avoided */
393 #if 0
394 long NCONF_get_number(CONF *conf,char *group,char *name)
395         {
396         int status;
397         long ret=0;

399         status = NCONF_get_number_e(conf, group, name, &ret);
400         if (status == 0)
401                 {
402                 /* This function does not believe in errors... */
403                 ERR_get_error();
404                 }
405         return ret;
406         }
407 #endif
408 #endif /* ! codereview */
```

```
**********************************************************
    3189 Wed Aug 13 19:52:24 2014
new/usr/src/lib/openssl/libsunw_crypto/conf/conf_mall.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* conf_mall.c */
/* Written by Stephen Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/crypto.h>
#include "cryptlib.h"
```



```
#include <openssl/conf.h>
#include <openssl/dso.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

/* Load all OpenSSL builtin modules */

void OPENSSL_load_builtin_modules(void)
        {
        /* Add builtin modules here */
        ASN1_add_oid_module();
#ifndef OPENSSL_NO_ENGINE
        ENGINE_add_conf_module();
#endif
        EVP_add_alg_module();
        }
#endif /* ! codereview */
```

```
**********************************************************
   14381 Wed Aug 13 19:52:25 2014
new/usr/src/lib/openssl/libsunw_crypto/conf/conf_mod.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* conf_mod.c */
/* Written by Stephen Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <ctype.h>
#include <openssl/crypto.h>
```



```
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/dso.h>
#include <openssl/x509.h>


#define DSO_mod_init_name "OPENSSL_init"
#define DSO_mod_finish_name "OPENSSL_finish"


/* This structure contains a data about supported modules.
 * entries in this table correspond to either dynamic or
 * static modules.
 */

struct conf_module_st
        {
        /* DSO of this module or NULL if static */
        DSO *dso;
        /* Name of the module */
        char *name;
        /* Init function */
        conf_init_func *init;
        /* Finish function */
        conf_finish_func *finish;
        /* Number of successfully initialized modules */
        int links;
        void *usr_data;
        };


/* This structure contains information about modules that have been
 * successfully initialized. There may be more than one entry for a
 * given module.
 */

struct conf_imodule_st
        {
        CONF_MODULE *pmod;
        char *name;
        char *value;
        unsigned long flags;
        void *usr_data;
        };

static STACK_OF(CONF_MODULE) *supported_modules = NULL;
static STACK_OF(CONF_IMODULE) *initialized_modules = NULL;

static void module_free(CONF_MODULE *md);
static void module_finish(CONF_IMODULE *imod);
static int module_run(const CONF *cnf, char *name, char *value,
                                        unsigned long flags);
static CONF_MODULE *module_add(DSO *dso, const char *name,
                        conf_init_func *ifunc, conf_finish_func *ffunc);
static CONF_MODULE *module_find(char *name);
static int module_init(CONF_MODULE *pmod, char *name, char *value,
                                         const CONF *cnf);
static CONF_MODULE *module_load_dso(const CONF *cnf, char *name, char *value,
                                                                        unsigned

/* Main function: load modules from a CONF structure */

int CONF_modules_load(const CONF *cnf, const char *appname,
                        unsigned long flags)
        {
        STACK_OF(CONF_VALUE) *values;
```

```
128         CONF_VALUE *vl;
129         char *vsection = NULL;

131         int ret, i;

133         if (!cnf)
134                 return 1;

136         if (appname)
137                 vsection = NCONF_get_string(cnf, NULL, appname);

139         if (!appname || (!vsection && (flags & CONF_MFLAGS_DEFAULT_SECTION)))
140                 vsection = NCONF_get_string(cnf, NULL, "openssl_conf");

142         if (!vsection)
143                 {
144                 ERR_clear_error();
145                 return 1;
146                 }

148         values = NCONF_get_section(cnf, vsection);

150         if (!values)
151                 return 0;

153         for (i = 0; i < sk_CONF_VALUE_num(values); i++)
154                 {
155                 vl = sk_CONF_VALUE_value(values, i);
156                 ret = module_run(cnf, vl->name, vl->value, flags);
157                 if (ret <= 0)
158                         if(!(flags & CONF_MFLAGS_IGNORE_ERRORS))
159                                 return ret;
160                 }

162         return 1;

164         }

166 int CONF_modules_load_file(const char *filename, const char *appname,
167                         unsigned long flags)
168         {
169         char *file = NULL;
170         CONF *conf = NULL;
171         int ret = 0;
172         conf = NCONF_new(NULL);
173         if (!conf)
174                 goto err;

176         if (filename == NULL)
177                 {
178                 file = CONF_get1_default_config_file();
179                 if (!file)
180                         goto err;
181                 }
182         else
183                 file = (char *)filename;

185         if (NCONF_load(conf, file, NULL) <= 0)
186                 {
187                 if ((flags & CONF_MFLAGS_IGNORE_MISSING_FILE) &&
188                   (ERR_GET_REASON(ERR_peek_last_error()) == CONF_R_NO_SUCH_FILE)
189                         {
190                         ERR_clear_error();
191                         ret = 1;
192                         }
193                 goto err;
```



```
194                 }

196         ret = CONF_modules_load(conf, appname, flags);

198         err:
199         if (filename == NULL)
200                 OPENSSL_free(file);
201         NCONF_free(conf);

203         return ret;
204         }

206 static int module_run(const CONF *cnf, char *name, char *value,
207                       unsigned long flags)
208         {
209         CONF_MODULE *md;
210         int ret;

212         md = module_find(name);

214         /* Module not found: try to load DSO */
215         if (!md && !(flags & CONF_MFLAGS_NO_DSO))
216                 md = module_load_dso(cnf, name, value, flags);

218         if (!md)
219                 {
220                 if (!(flags & CONF_MFLAGS_SILENT))
221                         {
222                         CONFerr(CONF_F_MODULE_RUN, CONF_R_UNKNOWN_MODULE_NAME);
223                         ERR_add_error_data(2, "module=", name);
224                         }
225                 return -1;
226                 }

228         ret = module_init(md, name, value, cnf);

230         if (ret <= 0)
231                 {
232                 if (!(flags & CONF_MFLAGS_SILENT))
233                         {
234                         char rcode[DECIMAL_SIZE(ret)+1];
235                         CONFerr(CONF_F_MODULE_RUN, CONF_R_MODULE_INITIALIZATION_
236                         BIO_snprintf(rcode, sizeof rcode, "%-8d", ret);
237                         ERR_add_error_data(6, "module=", name, ", value=", value
238                         }
239                 }

241         return ret;
242         }

244 /* Load a module from a DSO */
245 static CONF_MODULE *module_load_dso(const CONF *cnf, char *name, char *value,
246                                     unsigned long flags)
247         {
248         DSO *dso = NULL;
249         conf_init_func *ifunc;
250         conf_finish_func *ffunc;
251         char *path = NULL;
252         int errcode = 0;
253         CONF_MODULE *md;
254         /* Look for alternative path in module section */
255         path = NCONF_get_string(cnf, value, "path");
256         if (!path)
257                 {
258                 ERR_clear_error();
259                 path = name;
```

```
260                 }
261         dso = DSO_load(NULL, path, NULL, 0);
262         if (!dso)
263                 {
264                 errcode = CONF_R_ERROR_LOADING_DSO;
265                 goto err;
266                 }
267         ifunc = (conf_init_func *)DSO_bind_func(dso, DSO_mod_init_name);
268         if (!ifunc)
269                 {
270                 errcode = CONF_R_MISSING_INIT_FUNCTION;
271                 goto err;
272                 }
273         ffunc = (conf_finish_func *)DSO_bind_func(dso, DSO_mod_finish_name);
274         /* All OK, add module */
275         md = module_add(dso, name, ifunc, ffunc);
276
277         if (!md)
278                 goto err;
279
280         return md;
281
282         err:
283         if (dso)
284                 DSO_free(dso);
285         CONFerr(CONF_F_MODULE_LOAD_DSO, errcode);
286         ERR_add_error_data(4, "module=", name, ", path=", path);
287         return NULL;
288         }
289
290 /* add module to list */
291 static CONF_MODULE *module_add(DSO *dso, const char *name,
292                                conf_init_func *ifunc, conf_finish_func *ffunc)
293         {
294         CONF_MODULE *tmod = NULL;
295         if (supported_modules == NULL)
296                 supported_modules = sk_CONF_MODULE_new_null();
297         if (supported_modules == NULL)
298                 return NULL;
299         tmod = OPENSSL_malloc(sizeof(CONF_MODULE));
300         if (tmod == NULL)
301                 return NULL;
302
303         tmod->dso = dso;
304         tmod->name = BUF_strdup(name);
305         tmod->init = ifunc;
306         tmod->finish = ffunc;
307         tmod->links = 0;
308
309         if (!sk_CONF_MODULE_push(supported_modules, tmod))
310                 {
311                 OPENSSL_free(tmod);
312                 return NULL;
313                 }
314
315         return tmod;
316         }
317
318 /* Find a module from the list. We allow module names of the
319  * form modname.XXXX to just search for modname to allow the
320  * same module to be initialized more than once.
321  */
322
323 static CONF_MODULE *module_find(char *name)
324         {
325         CONF_MODULE *tmod;
```



```
326         int i, nchar;
327         char *p;
328         p = strrchr(name, '.');
329
330         if (p)
331                 nchar = p - name;
332         else
333                 nchar = strlen(name);
334
335         for (i = 0; i < sk_CONF_MODULE_num(supported_modules); i++)
336                 {
337                 tmod = sk_CONF_MODULE_value(supported_modules, i);
338                 if (!strncmp(tmod->name, name, nchar))
339                         return tmod;
340                 }
341
342         return NULL;
343
344         }
345
346 /* initialize a module */
347 static int module_init(CONF_MODULE *pmod, char *name, char *value,
348                        const CONF *cnf)
349         {
350         int ret = 1;
351         int init_called = 0;
352         CONF_IMODULE *imod = NULL;
353
354         /* Otherwise add initialized module to list */
355         imod = OPENSSL_malloc(sizeof(CONF_IMODULE));
356         if (!imod)
357                 goto err;
358
359         imod->pmod = pmod;
360         imod->name = BUF_strdup(name);
361         imod->value = BUF_strdup(value);
362         imod->usr_data = NULL;
363
364         if (!imod->name || !imod->value)
365                 goto memerr;
366
367         /* Try to initialize module */
368         if(pmod->init)
369                 {
370                 ret = pmod->init(imod, cnf);
371                 init_called = 1;
372                 /* Error occurred, exit */
373                 if (ret <= 0)
374                         goto err;
375                 }
376
377         if (initialized_modules == NULL)
378                 {
379                 initialized_modules = sk_CONF_IMODULE_new_null();
380                 if (!initialized_modules)
381                         {
382                         CONFerr(CONF_F_MODULE_INIT, ERR_R_MALLOC_FAILURE);
383                         goto err;
384                         }
385                 }
386
387         if (!sk_CONF_IMODULE_push(initialized_modules, imod))
388                 {
389                 CONFerr(CONF_F_MODULE_INIT, ERR_R_MALLOC_FAILURE);
390                 goto err;
391                 }
```

```
393         pmod->links++;

395         return ret;

397         err:

399         /* We've started the module so we'd better finish it */
400         if (pmod->finish && init_called)
401                 pmod->finish(imod);

403         memerr:
404         if (imod)
405                 {
406                 if (imod->name)
407                         OPENSSL_free(imod->name);
408                 if (imod->value)
409                         OPENSSL_free(imod->value);
410                 OPENSSL_free(imod);
411                 }

413         return -1;

415         }

417 /* Unload any dynamic modules that have a link count of zero:
418  * i.e. have no active initialized modules. If 'all' is set
419  * then all modules are unloaded including static ones.
420  */

422 void CONF_modules_unload(int all)
423         {
424         int i;
425         CONF_MODULE *md;
426         CONF_modules_finish();
427         /* unload modules in reverse order */
428         for (i = sk_CONF_MODULE_num(supported_modules) - 1; i >= 0; i--)
429                 {
430                 md = sk_CONF_MODULE_value(supported_modules, i);
431                 /* If static or in use and 'all' not set ignore it */
432                 if (((md->links > 0) || !md->dso) && !all)
433                         continue;
434                 /* Since we're working in reverse this is OK */
435                 (void)sk_CONF_MODULE_delete(supported_modules, i);
436                 module_free(md);
437                 }
438         if (sk_CONF_MODULE_num(supported_modules) == 0)
439                 {
440                 sk_CONF_MODULE_free(supported_modules);
441                 supported_modules = NULL;
442                 }
443         }

445 /* unload a single module */
446 static void module_free(CONF_MODULE *md)
447         {
448         if (md->dso)
449                 DSO_free(md->dso);
450         OPENSSL_free(md->name);
451         OPENSSL_free(md);
452         }

454 /* finish and free up all modules instances */

456 void CONF_modules_finish(void)
457         {
```



```
458         CONF_IMODULE *imod;
459         while (sk_CONF_IMODULE_num(initialized_modules) > 0)
460                 {
461                 imod = sk_CONF_IMODULE_pop(initialized_modules);
462                 module_finish(imod);
463                 }
464         sk_CONF_IMODULE_free(initialized_modules);
465         initialized_modules = NULL;
466         }

468 /* finish a module instance */

470 static void module_finish(CONF_IMODULE *imod)
471         {
472         if (imod->pmod->finish)
473                 imod->pmod->finish(imod);
474         imod->pmod->links--;
475         OPENSSL_free(imod->name);
476         OPENSSL_free(imod->value);
477         OPENSSL_free(imod);
478         }

480 /* Add a static module to OpenSSL */

482 int CONF_module_add(const char *name, conf_init_func *ifunc,
483                     conf_finish_func *ffunc)
484         {
485         if (module_add(NULL, name, ifunc, ffunc))
486                 return 1;
487         else
488                 return 0;
489         }

491 void CONF_modules_free(void)
492         {
493         CONF_modules_finish();
494         CONF_modules_unload(1);
495         }

497 /* Utility functions */

499 const char *CONF_imodule_get_name(const CONF_IMODULE *md)
500         {
501         return md->name;
502         }

504 const char *CONF_imodule_get_value(const CONF_IMODULE *md)
505         {
506         return md->value;
507         }

509 void *CONF_imodule_get_usr_data(const CONF_IMODULE *md)
510         {
511         return md->usr_data;
512         }

514 void CONF_imodule_set_usr_data(CONF_IMODULE *md, void *usr_data)
515         {
516         md->usr_data = usr_data;
517         }

519 CONF_MODULE *CONF_imodule_get_module(const CONF_IMODULE *md)
520         {
521         return md->pmod;
522         }
```

```
524 unsigned long CONF_imodule_get_flags(const CONF_IMODULE *md)
525         {
526         return md->flags;
527         }

529 void CONF_imodule_set_flags(CONF_IMODULE *md, unsigned long flags)
530         {
531         md->flags = flags;
532         }

534 void *CONF_module_get_usr_data(CONF_MODULE *pmod)
535         {
536         return pmod->usr_data;
537         }

539 void CONF_module_set_usr_data(CONF_MODULE *pmod, void *usr_data)
540         {
541         pmod->usr_data = usr_data;
542         }

544 /* Return default config file name */

546 char *CONF_get1_default_config_file(void)
547         {
548         char *file;
549         int len;

551         file = getenv("OPENSSL_CONF");
552         if (file)
553                 return BUF_strdup(file);

555         len = strlen(X509_get_default_cert_area());
556 #ifndef OPENSSL_SYS_VMS
557         len++;
558 #endif
559         len += strlen(OPENSSL_CONF);

561         file = OPENSSL_malloc(len + 1);

563         if (!file)
564                 return NULL;
565         BUF_strlcpy(file,X509_get_default_cert_area(),len + 1);
566 #ifndef OPENSSL_SYS_VMS
567         BUF_strlcat(file,"/",len + 1);
568 #endif
569         BUF_strlcat(file,OPENSSL_CONF,len + 1);

571         return file;
572         }

574 /* This function takes a list separated by 'sep' and calls the
575  * callback function giving the start and length of each member
576  * optionally stripping leading and trailing whitespace. This can
577  * be used to parse comma separated lists for example.
578  */

580 int CONF_parse_list(const char *list_, int sep, int nospc,
581         int (*list_cb)(const char *elem, int len, void *usr), void *arg)
582         {
583         int ret;
584         const char *lstart, *tmpend, *p;

586         if(list_ == NULL)
587                 {
588                 CONFerr(CONF_F_CONF_PARSE_LIST, CONF_R_LIST_CANNOT_BE_NULL);
589                 return 0;
```



```
590                 }

592         lstart = list_;
593         for(;;)
594                 {
595                 if (nospc)
596                         {
597                         while(*lstart && isspace((unsigned char)*lstart))
598                                 lstart++;
599                         }
600                 p = strchr(lstart, sep);
601                 if (p == lstart || !*lstart)
602                         ret = list_cb(NULL, 0, arg);
603                 else
604                         {
605                         if (p)
606                                 tmpend = p - 1;
607                         else
608                                 tmpend = lstart + strlen(lstart) - 1;
609                         if (nospc)
610                                 {
611                                 while(isspace((unsigned char)*tmpend))
612                                         tmpend--;
613                                 }
614                         ret = list_cb(lstart, tmpend - lstart + 1, arg);
615                         }
616                 if (ret <= 0)
617                         return ret;
618                 if (p == NULL)
619                         return 1;
620                 lstart = p + 1;
621                 }
622         }
623 #endif /* ! codereview */
```

```
**********************************************************
    3900 Wed Aug 13 19:52:25 2014
new/usr/src/lib/openssl/libsunw_crypto/conf/conf_sap.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* conf_sap.c */
/* Written by Stephen Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/crypto.h>
#include "cryptlib.h"
```



```
#include <openssl/conf.h>
#include <openssl/dso.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

/* This is the automatic configuration loader: it is called automatically by
 * OpenSSL when any of a number of standard initialisation functions are called,
 * unless this is overridden by calling OPENSSL_no_config()
 */

static int openssl_configured = 0;

void OPENSSL_config(const char *config_name)
        {
        if (openssl_configured)
                return;

        OPENSSL_load_builtin_modules();
#ifndef OPENSSL_NO_ENGINE
        /* Need to load ENGINEs */
        ENGINE_load_builtin_engines();
#endif
        /* Add others here? */


        ERR_clear_error();
        if (CONF_modules_load_file(NULL, config_name,
        CONF_MFLAGS_DEFAULT_SECTION|CONF_MFLAGS_IGNORE_MISSING_FILE) <= 0)
                {
                BIO *bio_err;
                ERR_load_crypto_strings();
                if ((bio_err=BIO_new_fp(stderr, BIO_NOCLOSE)) != NULL)
                        {
                        BIO_printf(bio_err,"Auto configuration failed\n");
                        ERR_print_errors(bio_err);
                        BIO_free(bio_err);
                        }
                exit(1);
                }

        return;
        }

void OPENSSL_no_config()
        {
        openssl_configured = 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4214 Wed Aug 13 19:52:25 2014
new/usr/src/lib/openssl/libsunw_crypto/cpt_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cpt_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/crypto.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_CRYPTO,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_CRYPTO,0,reason)

static ERR_STRING_DATA CRYPTO_str_functs[]=
        {
{ERR_FUNC(CRYPTO_F_CRYPTO_GET_EX_NEW_INDEX),    "CRYPTO_get_ex_new_index"},
{ERR_FUNC(CRYPTO_F_CRYPTO_GET_NEW_DYNLOCKID),   "CRYPTO_get_new_dynlockid"},
{ERR_FUNC(CRYPTO_F_CRYPTO_GET_NEW_LOCKID),      "CRYPTO_get_new_lockid"},
{ERR_FUNC(CRYPTO_F_CRYPTO_SET_EX_DATA), "CRYPTO_set_ex_data"},
{ERR_FUNC(CRYPTO_F_DEF_ADD_INDEX),      "DEF_ADD_INDEX"},
{ERR_FUNC(CRYPTO_F_DEF_GET_CLASS),      "DEF_GET_CLASS"},
{ERR_FUNC(CRYPTO_F_FIPS_MODE_SET),      "FIPS_mode_set"},
{ERR_FUNC(CRYPTO_F_INT_DUP_EX_DATA),    "INT_DUP_EX_DATA"},
{ERR_FUNC(CRYPTO_F_INT_FREE_EX_DATA),   "INT_FREE_EX_DATA"},
{ERR_FUNC(CRYPTO_F_INT_NEW_EX_DATA),    "INT_NEW_EX_DATA"},
{0,NULL}
        };

static ERR_STRING_DATA CRYPTO_str_reasons[]=
        {
{ERR_REASON(CRYPTO_R_FIPS_MODE_NOT_SUPPORTED),"fips mode not supported"},
{ERR_REASON(CRYPTO_R_NO_DYNLOCK_CREATE_CALLBACK),"no dynlock create callback"},
{0,NULL}
        };

#endif

void ERR_load_CRYPTO_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(CRYPTO_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,CRYPTO_str_functs);
                ERR_load_strings(0,CRYPTO_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
   28436 Wed Aug 13 19:52:25 2014
new/usr/src/lib/openssl/libsunw_crypto/cryptlib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cryptlib.c */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
```



```
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */

#include "cryptlib.h"
#include <openssl/safestack.h>
#include <pthread.h>

#if defined(OPENSSL_SYS_WIN32) || defined(OPENSSL_SYS_WIN16)
static double SSLeay_MSVC5_hack=0.0; /* and for VC1.5 */
#endif

DECLARE_STACK_OF(CRYPTO_dynlock)

/* real #defines in crypto.h, keep these upto date */
```

```
128 static const char* const lock_names[CRYPTO_NUM_LOCKS] =
129         {
130         "<<ERROR>>",
131         "err",
132         "ex_data",
133         "x509",
134         "x509_info",
135         "x509_pkey",
136         "x509_crl",
137         "x509_req",
138         "dsa",
139         "rsa",
140         "evp_pkey",
141         "x509_store",
142         "ssl_ctx",
143         "ssl_cert",
144         "ssl_session",
145         "ssl_sess_cert",
146         "ssl",
147         "ssl_method",
148         "rand",
149         "rand2",
150         "debug_malloc",
151         "BIO",
152         "gethostbyname",
153         "getservbyname",
154         "readdir",
155         "RSA_blinding",
156         "dh",
157         "debug_malloc2",
158         "dso",
159         "dynlock",
160         "engine",
161         "ui",
162         "ecdsa",
163         "ec",
164         "ecdh",
165         "bn",
166         "ec_pre_comp",
167         "store",
168         "comp",
169         "fips",
170         "fips2",
171 #if CRYPTO_NUM_LOCKS != 41
172 # error "Inconsistency between crypto.h and cryptlib.c"
173 #endif
174         };

176 /* This is for applications to allocate new type names in the non-dynamic
177    array of lock names.  These are numbered with positive numbers.  */
178 static STACK_OF(OPENSSL_STRING) *app_locks=NULL;

180 /* For applications that want a more dynamic way of handling threads, the
181    following stack is used.  These are externally numbered with negative
182    numbers.  */
183 static STACK_OF(CRYPTO_dynlock) *dyn_locks=NULL;

185 static pthread_mutex_t *illumos_openssl_locks;

187 static void (MS_FAR *locking_callback)(int mode,int type,
188         const char *file,int line)=0;
189 static int (MS_FAR *add_lock_callback)(int *pointer,int amount,
190         int type,const char *file,int line)=0;
191 #ifndef OPENSSL_NO_DEPRECATED
192 static unsigned long (MS_FAR *id_callback)(void)=0;
193 #endif
```



```
194 static void (MS_FAR *threadid_callback)(CRYPTO_THREADID *)=0;
195 static struct CRYPTO_dynlock_value *(MS_FAR *dynlock_create_callback)
196         (const char *file,int line)=0;
197 static void (MS_FAR *dynlock_lock_callback)(int mode,
198         struct CRYPTO_dynlock_value *l, const char *file,int line)=0;
199 static void (MS_FAR *dynlock_destroy_callback)(struct CRYPTO_dynlock_value *l,
200         const char *file,int line)=0;

202 int CRYPTO_get_new_lockid(char *name)
203         {
204         char *str;
205         int i;

207 #if defined(OPENSSL_SYS_WIN32) || defined(OPENSSL_SYS_WIN16)
208         /* A hack to make Visual C++ 5.0 work correctly when linking as
209          * a DLL using /MT. Without this, the application cannot use
210          * any floating point printf's.
211          * It also seems to be needed for Visual C 1.5 (win16) */
212         SSLeay_MSVC5_hack=(double)name[0]*(double)name[1];
213 #endif

215         if ((app_locks == NULL) && ((app_locks=sk_OPENSSL_STRING_new_null()) ==
216                 {
217                 CRYPTOerr(CRYPTO_F_CRYPTO_GET_NEW_LOCKID,ERR_R_MALLOC_FAILURE);
218                 return(0);
219                 }
220         if ((str=BUF_strdup(name)) == NULL)
221                 {
222                 CRYPTOerr(CRYPTO_F_CRYPTO_GET_NEW_LOCKID,ERR_R_MALLOC_FAILURE);
223                 return(0);
224                 }
225         i=sk_OPENSSL_STRING_push(app_locks,str);
226         if (!i)
227                 OPENSSL_free(str);
228         else
229                 i+=CRYPTO_NUM_LOCKS; /* gap of one :-) */
230         return(i);
231         }

233 int CRYPTO_num_locks(void)
234         {
235         return CRYPTO_NUM_LOCKS;
236         }

238 int CRYPTO_get_new_dynlockid(void)
239         {
240         int i = 0;
241         CRYPTO_dynlock *pointer = NULL;

243         if (dynlock_create_callback == NULL)
244                 {
245                 CRYPTOerr(CRYPTO_F_CRYPTO_GET_NEW_DYNLOCKID,CRYPTO_R_NO_DYNLOCK_
246                 return(0);
247                 }
248         CRYPTO_w_lock(CRYPTO_LOCK_DYNLOCK);
249         if ((dyn_locks == NULL)
250                 && ((dyn_locks=sk_CRYPTO_dynlock_new_null()) == NULL))
251                 {
252                 CRYPTO_w_unlock(CRYPTO_LOCK_DYNLOCK);
253                 CRYPTOerr(CRYPTO_F_CRYPTO_GET_NEW_DYNLOCKID,ERR_R_MALLOC_FAILURE
254                 return(0);
255                 }
256         CRYPTO_w_unlock(CRYPTO_LOCK_DYNLOCK);

258         pointer = (CRYPTO_dynlock *)OPENSSL_malloc(sizeof(CRYPTO_dynlock));
259         if (pointer == NULL)
```

```
260                 {
261                 CRYPTOerr(CRYPTO_F_CRYPTO_GET_NEW_DYNLOCKID,ERR_R_MALLOC_FAILURE
262                 return(0);
263                 }
264         pointer->references = 1;
265         pointer->data = dynlock_create_callback(__FILE__,__LINE__);
266         if (pointer->data == NULL)
267                 {
268                 OPENSSL_free(pointer);
269                 CRYPTOerr(CRYPTO_F_CRYPTO_GET_NEW_DYNLOCKID,ERR_R_MALLOC_FAILURE
270                 return(0);
271                 }

273         CRYPTO_w_lock(CRYPTO_LOCK_DYNLOCK);
274         /* First, try to find an existing empty slot */
275         i=sk_CRYPTO_dynlock_find(dyn_locks,NULL);
276         /* If there was none, push, thereby creating a new one */
277         if (i == -1)
278                 /* Since sk_push() returns the number of items on the
279                    stack, not the location of the pushed item, we need
280                    to transform the returned number into a position,
281                    by decreasing it.  */
282                 i=sk_CRYPTO_dynlock_push(dyn_locks,pointer) - 1;
283         else
284                 /* If we found a place with a NULL pointer, put our pointer
285                    in it.  */
286                 (void)sk_CRYPTO_dynlock_set(dyn_locks,i,pointer);
287         CRYPTO_w_unlock(CRYPTO_LOCK_DYNLOCK);

289         if (i == -1)
290                 {
291                 dynlock_destroy_callback(pointer->data,__FILE__,__LINE__);
292                 OPENSSL_free(pointer);
293                 }
294         else
295                 i += 1; /* to avoid 0 */
296         return -i;
297         }

299 void CRYPTO_destroy_dynlockid(int i)
300         {
301         CRYPTO_dynlock *pointer = NULL;
302         if (i)
303                 i = -i-1;
304         if (dynlock_destroy_callback == NULL)
305                 return;

307         CRYPTO_w_lock(CRYPTO_LOCK_DYNLOCK);

309         if (dyn_locks == NULL || i >= sk_CRYPTO_dynlock_num(dyn_locks))
310                 {
311                 CRYPTO_w_unlock(CRYPTO_LOCK_DYNLOCK);
312                 return;
313                 }
314         pointer = sk_CRYPTO_dynlock_value(dyn_locks, i);
315         if (pointer != NULL)
316                 {
317                 --pointer->references;
318 #ifdef REF_CHECK
319                 if (pointer->references < 0)
320                         {
321                         fprintf(stderr,"CRYPTO_destroy_dynlockid, bad reference
322                         abort();
323                         }
324                 else
325 #endif
```



```
326                         if (pointer->references <= 0)
327                                 {
328                                 (void)sk_CRYPTO_dynlock_set(dyn_locks, i, NULL);
329                                 }
330                         else
331                                 pointer = NULL;
332                 }
333         CRYPTO_w_unlock(CRYPTO_LOCK_DYNLOCK);

335         if (pointer)
336                 {
337                 dynlock_destroy_callback(pointer->data,__FILE__,__LINE__);
338                 OPENSSL_free(pointer);
339                 }
340         }

342 struct CRYPTO_dynlock_value *CRYPTO_get_dynlock_value(int i)
343         {
344         CRYPTO_dynlock *pointer = NULL;
345         if (i)
346                 i = -i-1;

348         CRYPTO_w_lock(CRYPTO_LOCK_DYNLOCK);

350         if (dyn_locks != NULL && i < sk_CRYPTO_dynlock_num(dyn_locks))
351                 pointer = sk_CRYPTO_dynlock_value(dyn_locks, i);
352         if (pointer)
353                 pointer->references++;

355         CRYPTO_w_unlock(CRYPTO_LOCK_DYNLOCK);

357         if (pointer)
358                 return pointer->data;
359         return NULL;
360         }

362 struct CRYPTO_dynlock_value *(*CRYPTO_get_dynlock_create_callback(void))
363         (const char *file,int line)
364         {
365         return(dynlock_create_callback);
366         }

368 void (*CRYPTO_get_dynlock_lock_callback(void))(int mode,
369         struct CRYPTO_dynlock_value *l, const char *file,int line)
370         {
371         return(dynlock_lock_callback);
372         }

374 void (*CRYPTO_get_dynlock_destroy_callback(void))
375         (struct CRYPTO_dynlock_value *l, const char *file,int line)
376         {
377         return(dynlock_destroy_callback);
378         }

380 void CRYPTO_set_dynlock_create_callback(struct CRYPTO_dynlock_value *(*func)
381         (const char *file, int line))
382         {
383         dynlock_create_callback=func;
384         }

386 void CRYPTO_set_dynlock_lock_callback(void (*func)(int mode,
387         struct CRYPTO_dynlock_value *l, const char *file, int line))
388         {
389         dynlock_lock_callback=func;
390         }
```

```
392 void CRYPTO_set_dynlock_destroy_callback(void (*func)
393         (struct CRYPTO_dynlock_value *l, const char *file, int line))
394         {
395         dynlock_destroy_callback=func;
396         }


399 void (*CRYPTO_get_locking_callback(void))(int mode,int type,const char *file,
400                 int line)
401         {
402         return(locking_callback);
403         }

405 int (*CRYPTO_get_add_lock_callback(void))(int *num,int mount,int type,
406                                           const char *file,int line)
407         {
408         return(add_lock_callback);
409         }

411 /*
412  * This is the locking callback function which all applications will be
413  * using when CRYPTO_lock() is called.
414  */
415 static void illumos_locking_callback(int mode, int type, const char *file,
416     int line)
417         {
418         if (mode & CRYPTO_LOCK)
419                 {
420                 pthread_mutex_lock(&illumos_openssl_locks[type]);
421                 }
422         else
423                 {
424                 pthread_mutex_unlock(&illumos_openssl_locks[type]);
425                 }
426         }


429 /*
430  * This function is called when a child process is forked to setup its own
431  * global locking callback function ptr and mutexes.
432  */
433 static void illumos_fork_child(void)
434         {
435                 /*
436                  * clear locking_callback to indicate that locks should
437                  * be reinitialized.
438                  */
439                 locking_callback = NULL;
440                 illumos_locking_setup();
441         }

443 /*
444  * This function allocates and initializes the global mutex array, and
445  * sets the locking callback.
446  */
447 void illumos_locking_setup()
448         {
449         int i;
450         int num_locks;

452         /* locking callback is already setup. Nothing to do */
453         if (locking_callback != NULL)
454                 {
455                 return;
456                 }
```



```
458         /*
459          * Set atfork handler so that child can setup its own mutexes and
460          * locking callbacks when it is forked
461          */
462         (void) pthread_atfork(NULL, NULL, illumos_fork_child);

464         /* allocate locks needed by OpenSSL  */
465         num_locks = CRYPTO_num_locks();
466         illumos_openssl_locks =
467             OPENSSL_malloc(sizeof (pthread_mutex_t) * num_locks);
468         if (illumos_openssl_locks == NULL)
469                 {
470                 fprintf(stderr,
471                         "illumos_locking_setup: memory allocation failure.\n");
472                 abort();
473                 }

475         /* initialize openssl mutexes */
476         for (i = 0; i < num_locks; i++)
477                 {
478                 pthread_mutex_init(&illumos_openssl_locks[i], NULL);
479                 }
480         locking_callback = illumos_locking_callback;

482         }

484 void CRYPTO_set_locking_callback(void (*func)(int mode,int type,
485                                               const char *file,int line))
486         {
487         /* Calling this here ensures initialisation before any threads
488          * are started.
489          */
490         OPENSSL_init();

492         /*
493          * we now setup our own locking callback and mutexes, and disallow
494          * setting of another locking callback.
495          */
496         }

498 void CRYPTO_set_add_lock_callback(int (*func)(int *num,int mount,int type,
499                                               const char *file,int line))
500         {
501         add_lock_callback=func;
502         }

504 /* the memset() here and in set_pointer() seem overkill, but for the sake of
505  * CRYPTO_THREADID_cmp() this avoids any platform silliness that might cause two
506  * "equal" THREADID structs to not be memcmp()-identical. */
507 void CRYPTO_THREADID_set_numeric(CRYPTO_THREADID *id, unsigned long val)
508         {
509         memset(id, 0, sizeof(*id));
510         id->val = val;
511         }

513 static const unsigned char hash_coeffs[] = { 3, 5, 7, 11, 13, 17, 19, 23 };
514 void CRYPTO_THREADID_set_pointer(CRYPTO_THREADID *id, void *ptr)
515         {
516         unsigned char *dest = (void *)&id->val;
517         unsigned int accum = 0;
518         unsigned char dnum = sizeof(id->val);

520         memset(id, 0, sizeof(*id));
521         id->ptr = ptr;
522         if (sizeof(id->val) >= sizeof(id->ptr))
523                 {
```

```
524                 /* 'ptr' can be embedded in 'val' without loss of uniqueness */
525                 id->val = (unsigned long)id->ptr;
526                 return;
527                 }
528         /* hash ptr ==> val. Each byte of 'val' gets the mod-256 total of a
529          * linear function over the bytes in 'ptr', the co-efficients of which
530          * are a sequence of low-primes (hash_coeffs is an 8-element cycle) -
531          * the starting prime for the sequence varies for each byte of 'val'
532          * (unique polynomials unless pointers are >64-bit). For added spice,
533          * the totals accumulate rather than restarting from zero, and the index
534          * of the 'val' byte is added each time (position dependence). If I was
535          * a black-belt, I'd scan big-endian pointers in reverse to give
536          * low-order bits more play, but this isn't crypto and I'd prefer nobody
537          * mistake it as such. Plus I'm lazy. */
538         while (dnum--)
539                 {
540                 const unsigned char *src = (void *)&id->ptr;
541                 unsigned char snum = sizeof(id->ptr);
542                 while (snum--)
543                         accum += *(src++) * hash_coeffs[(snum + dnum) & 7];
544                 accum += dnum;
545                 *(dest++) = accum & 255;
546                 }
547         }

549 int CRYPTO_THREADID_set_callback(void (*func)(CRYPTO_THREADID *))
550         {
551         if (threadid_callback)
552                 return 0;
553         threadid_callback = func;
554         return 1;
555         }

557 void (*CRYPTO_THREADID_get_callback(void))(CRYPTO_THREADID *)
558         {
559         return threadid_callback;
560         }

562 void CRYPTO_THREADID_current(CRYPTO_THREADID *id)
563         {
564         if (threadid_callback)
565                 {
566                 threadid_callback(id);
567                 return;
568                 }
569 #ifndef OPENSSL_NO_DEPRECATED
570         /* If the deprecated callback was set, fall back to that */
571         if (id_callback)
572                 {
573                 CRYPTO_THREADID_set_numeric(id, id_callback());
574                 return;
575                 }
576 #endif
577         /* Else pick a backup */
578 #ifdef OPENSSL_SYS_WIN16
579         CRYPTO_THREADID_set_numeric(id, (unsigned long)GetCurrentTask());
580 #elif defined(OPENSSL_SYS_WIN32)
581         CRYPTO_THREADID_set_numeric(id, (unsigned long)GetCurrentThreadId());
582 #elif defined(OPENSSL_SYS_BEOS)
583         CRYPTO_THREADID_set_numeric(id, (unsigned long)find_thread(NULL));
584 #else
585         /* For everything else, default to using the address of 'errno' */
586         CRYPTO_THREADID_set_pointer(id, (void*)&errno);
587 #endif
588         }
```



```
590 int CRYPTO_THREADID_cmp(const CRYPTO_THREADID *a, const CRYPTO_THREADID *b)
591         {
592         return memcmp(a, b, sizeof(*a));
593         }

595 void CRYPTO_THREADID_cpy(CRYPTO_THREADID *dest, const CRYPTO_THREADID *src)
596         {
597         memcpy(dest, src, sizeof(*src));
598         }

600 unsigned long CRYPTO_THREADID_hash(const CRYPTO_THREADID *id)
601         {
602         return id->val;
603         }

605 #ifndef OPENSSL_NO_DEPRECATED
606 unsigned long (*CRYPTO_get_id_callback(void))(void)
607         {
608         return(id_callback);
609         }

611 void CRYPTO_set_id_callback(unsigned long (*func)(void))
612         {
613         id_callback=func;
614         }

616 unsigned long CRYPTO_thread_id(void)
617         {
618         unsigned long ret=0;

620         if (id_callback == NULL)
621                 {
622 #ifdef OPENSSL_SYS_WIN16
623                 ret=(unsigned long)GetCurrentTask();
624 #elif defined(OPENSSL_SYS_WIN32)
625                 ret=(unsigned long)GetCurrentThreadId();
626 #elif defined(GETPID_IS_MEANINGLESS)
627                 ret=1L;
628 #elif defined(OPENSSL_SYS_BEOS)
629                 ret=(unsigned long)find_thread(NULL);
630 #else
631                 ret=(unsigned long)getpid();
632 #endif
633                 }
634         else
635                 ret=id_callback();
636         return(ret);
637         }
638 #endif

640 void CRYPTO_lock(int mode, int type, const char *file, int line)
641         {
642 #ifdef LOCK_DEBUG
643                 {
644                 CRYPTO_THREADID id;
645                 char *rw_text,*operation_text;

647                 if (mode & CRYPTO_LOCK)
648                         operation_text="lock  ";
649                 else if (mode & CRYPTO_UNLOCK)
650                         operation_text="unlock";
651                 else
652                         operation_text="ERROR ";

654                 if (mode & CRYPTO_READ)
655                         rw_text="r";
```

```
656                 else if (mode & CRYPTO_WRITE)
657                         rw_text="w";
658                 else
659                         rw_text="ERROR";

661                 CRYPTO_THREADID_current(&id);
662                 fprintf(stderr,"lock:%08lx:(%s)%s %-18s %s:%d\n",
663                         CRYPTO_THREADID_hash(&id), rw_text, operation_text,
664                         CRYPTO_get_lock_name(type), file, line);
665                 }
666 #endif
667         if (type < 0)
668                 {
669                 if (dynlock_lock_callback != NULL)
670                         {
671                         struct CRYPTO_dynlock_value *pointer
672                                 = CRYPTO_get_dynlock_value(type);

674                         OPENSSL_assert(pointer != NULL);

676                         dynlock_lock_callback(mode, pointer, file, line);

678                         CRYPTO_destroy_dynlockid(type);
679                         }
680                 }
681         else
682                 if (locking_callback != NULL)
683                         locking_callback(mode,type,file,line);
684         }

686 int CRYPTO_add_lock(int *pointer, int amount, int type, const char *file,
687              int line)
688         {
689         int ret = 0;

691         if (add_lock_callback != NULL)
692                 {
693 #ifdef LOCK_DEBUG
694                 int before= *pointer;
695 #endif

697                 ret=add_lock_callback(pointer,amount,type,file,line);
698 #ifdef LOCK_DEBUG
699                 {
700                 CRYPTO_THREADID id;
701                 CRYPTO_THREADID_current(&id);
702                 fprintf(stderr,"ladd:%08lx:%2d+%2d->%2d %-18s %s:%d\n",
703                         CRYPTO_THREADID_hash(&id), before,amount,ret,
704                         CRYPTO_get_lock_name(type),
705                         file,line);
706                 }
707 #endif
708                 }
709         else
710                 {
711                 CRYPTO_lock(CRYPTO_LOCK|CRYPTO_WRITE,type,file,line);

713                 ret= *pointer+amount;
714 #ifdef LOCK_DEBUG
715                 {
716                 CRYPTO_THREADID id;
717                 CRYPTO_THREADID_current(&id);
718                 fprintf(stderr,"ladd:%08lx:%2d+%2d->%2d %-18s %s:%d\n",
719                         CRYPTO_THREADID_hash(&id),
720                         *pointer,amount,ret,
721                         CRYPTO_get_lock_name(type),
```



```
722                         file,line);
723                 }
724 #endif
725                 *pointer=ret;
726                 CRYPTO_lock(CRYPTO_UNLOCK|CRYPTO_WRITE,type,file,line);
727                 }
728         return(ret);
729         }

731 const char *CRYPTO_get_lock_name(int type)
732         {
733         if (type < 0)
734                 return("dynamic");
735         else if (type < CRYPTO_NUM_LOCKS)
736                 return(lock_names[type]);
737         else if (type-CRYPTO_NUM_LOCKS > sk_OPENSSL_STRING_num(app_locks))
738                 return("ERROR");
739         else
740                 return(sk_OPENSSL_STRING_value(app_locks,type-CRYPTO_NUM_LOCKS));
741         }

743 #if     defined(__i386)   || defined(__i386__)   || defined(_M_IX86) || \
744         defined(__INTEL__) || \
745         defined(__x86_64) || defined(__x86_64__) || defined(_M_AMD64) || defined

747 unsigned int  OPENSSL_ia32cap_P[2];
748 unsigned long *OPENSSL_ia32cap_loc(void)
749 {   if (sizeof(long)==4)
750         /*
751          * If 32-bit application pulls address of OPENSSL_ia32cap_P[0]
752          * clear second element to maintain the illusion that vector
753          * is 32-bit.
754          */
755         OPENSSL_ia32cap_P[1]=0;
756     return (unsigned long *)OPENSSL_ia32cap_P;
757 }

759 #if defined(OPENSSL_CPUID_OBJ) && !defined(OPENSSL_NO_ASM) && !defined(I386_ONLY
760 #define OPENSSL_CPUID_SETUP
761 #if defined(_WIN32)
762 typedef unsigned __int64 IA32CAP;
763 #else
764 typedef unsigned long long IA32CAP;
765 #endif
766 void OPENSSL_cpuid_setup(void)
767 { static int trigger=0;
768   IA32CAP OPENSSL_ia32_cpuid(void);
769   IA32CAP vec;
770   char *env;

772     if (trigger)        return;

774     trigger=1;
775     if ((env=getenv("OPENSSL_ia32cap"))) {
776         int off = (env[0]=='~')?1:0;
777 #if defined(_WIN32)
778         if (!sscanf(env+off,"%I64i",&vec)) vec = strtoul(env+off,NULL,0);
779 #else
780         if (!sscanf(env+off,"%lli",(long long *)&vec)) vec = strtoul(env+off,NUL
781 #endif
782         if (off) vec = OPENSSL_ia32_cpuid()&~vec;
783     }
784     else
785         vec = OPENSSL_ia32_cpuid();

787     /*
```

```
788      * |(1<<10) sets a reserved bit to signal that variable
789      * was initialized already... This is to avoid interference
790      * with cpuid snippets in ELF .init segment.
791      */
792     OPENSSL_ia32cap_P[0] = (unsigned int)vec|(1<<10);
793     OPENSSL_ia32cap_P[1] = (unsigned int)(vec>>32);
794 }
795 #endif

797 #else
798 unsigned long *OPENSSL_ia32cap_loc(void) { return NULL; }
799 #endif
800 int OPENSSL_NONPIC_relocated = 0;
801 #if !defined(OPENSSL_CPUID_SETUP) && !defined(OPENSSL_CPUID_OBJ)
802 void OPENSSL_cpuid_setup(void) {}
803 #endif

805 #if (defined(_WIN32) || defined(__CYGWIN__)) && defined(_WINDLL)
806 #ifdef __CYGWIN__
807 /* pick DLL_[PROCESS|THREAD]_[ATTACH|DETACH] definitions */
808 #include <windows.h>
809 /* this has side-effect of _WIN32 getting defined, which otherwise
810  * is mutually exclusive with __CYGWIN__... */
811 #endif

813 /* All we really need to do is remove the 'error' state when a thread
814  * detaches */

816 BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason,
817             LPVOID lpvReserved)
818         {
819         switch(fdwReason)
820                 {
821         case DLL_PROCESS_ATTACH:
822                 OPENSSL_cpuid_setup();
823 #if defined(_WIN32_WINNT)
824                 {
825                 IMAGE_DOS_HEADER *dos_header = (IMAGE_DOS_HEADER *)hinstDLL;
826                 IMAGE_NT_HEADERS *nt_headers;

828                 if (dos_header->e_magic==IMAGE_DOS_SIGNATURE)
829                         {
830                         nt_headers = (IMAGE_NT_HEADERS *)((char *)dos_header
831                                                           + dos_header->e_lfanew);
832                         if (nt_headers->Signature==IMAGE_NT_SIGNATURE &&
833                             hinstDLL!=(HINSTANCE)(nt_headers->OptionalHeader.Ima
834                                 OPENSSL_NONPIC_relocated=1;
835                         }
836                 }
837 #endif
838                 break;
839         case DLL_THREAD_ATTACH:
840                 break;
841         case DLL_THREAD_DETACH:
842                 break;
843         case DLL_PROCESS_DETACH:
844                 break;
845                 }
846         return(TRUE);
847         }
848 #endif

850 #if defined(_WIN32) && !defined(__CYGWIN__)
851 #include <tchar.h>
852 #include <signal.h>
853 #ifdef __WATCOMC__
```



```
854 #if defined(_UNICODE) || defined(__UNICODE__)
855 #define _vsntprintf _vsnwprintf
856 #else
857 #define _vsntprintf _vsnprintf
858 #endif
859 #endif
860 #ifdef _MSC_VER
861 #define alloca _alloca
862 #endif

864 #if defined(_WIN32_WINNT) && _WIN32_WINNT>=0x0333
865 int OPENSSL_isservice(void)
866 { HWINSTA h;
867   DWORD len;
868   WCHAR *name;
869   static union { void *p; int (*f)(void); } _OPENSSL_isservice = { NULL };

871     if (_OPENSSL_isservice.p == NULL) {
872         HANDLE h = GetModuleHandle(NULL);
873         if (h != NULL)
874             _OPENSSL_isservice.p = GetProcAddress(h,"_OPENSSL_isservice");
875         if (_OPENSSL_isservice.p == NULL)
876             _OPENSSL_isservice.p = (void *)-1;
877     }

879     if (_OPENSSL_isservice.p != (void *)-1)
880         return (*_OPENSSL_isservice.f)();

882     (void)GetDesktopWindow(); /* return value is ignored */

884     h = GetProcessWindowStation();
885     if (h==NULL) return -1;

887     if (GetUserObjectInformationW (h,UOI_NAME,NULL,0,&len) ||
888         GetLastError() != ERROR_INSUFFICIENT_BUFFER)
889         return -1;

891     if (len>512) return -1;             /* paranoia */
892     len++,len&=~1;                      /* paranoia */
893     name=(WCHAR *)alloca(len+sizeof(WCHAR));
894     if (!GetUserObjectInformationW (h,UOI_NAME,name,len,&len))
895         return -1;

897     len++,len&=~1;                      /* paranoia */
898     name[len/sizeof(WCHAR)]=L'\0';      /* paranoia */
899 #if 1
900     /* This doesn't cover "interactive" services [working with real
901      * WinSta0's] nor programs started non-interactively by Task
902      * Scheduler [those are working with SAWinSta]. */
903     if (wcsstr(name,L"Service-0x"))     return 1;
904 #else
905     /* This covers all non-interactive programs such as services. */
906     if (!wcsstr(name,L"WinSta0"))       return 1;
907 #endif
908     else                                return 0;
909 }
910 #else
911 int OPENSSL_isservice(void) { return 0; }
912 #endif

914 void OPENSSL_showfatal (const char *fmta,...)
915 { va_list ap;
916   TCHAR buf[256];
917   const TCHAR *fmt;
918 #ifdef STD_ERROR_HANDLE /* what a dirty trick! */
919   HANDLE h;
```

```
    if ((h=GetStdHandle(STD_ERROR_HANDLE)) != NULL &&
        GetFileType(h)!=FILE_TYPE_UNKNOWN)
    {   /* must be console application */
        va_start (ap,fmta);
        vfprintf (stderr,fmta,ap);
        va_end (ap);
        return;
    }
#endif

    if (sizeof(TCHAR)==sizeof(char))
        fmt=(const TCHAR *)fmta;
    else do
    { int    keepgoing;
      size_t len_0=strlen(fmta)+1,i;
      WCHAR *fmtw;

        fmtw = (WCHAR *)alloca(len_0*sizeof(WCHAR));
        if (fmtw == NULL) { fmt=(const TCHAR *)L"no stack?"; break; }

#ifndef OPENSSL_NO_MULTIBYTE
        if (!MultiByteToWideChar(CP_ACP,0,fmta,len_0,fmtw,len_0))
#endif
            for (i=0;i<len_0;i++) fmtw[i]=(WCHAR)fmta[i];

        for (i=0;i<len_0;i++)
        {   if (fmtw[i]==L'%') do
            {   keepgoing=0;
                switch (fmtw[i+1])
                {   case L'0': case L'1': case L'2': case L'3': case L'4':
                    case L'5': case L'6': case L'7': case L'8': case L'9':
                    case L'.': case L'*':
                    case L'-':  i++; keepgoing=1; break;
                    case L's':  fmtw[i+1]=L'S';   break;
                    case L'S':  fmtw[i+1]=L's';   break;
                    case L'c':  fmtw[i+1]=L'C';   break;
                    case L'C':  fmtw[i+1]=L'c';   break;
                }
            } while (keepgoing);
        }
        fmt = (const TCHAR *)fmtw;
    } while (0);

    va_start (ap,fmta);
    _vsntprintf (buf,sizeof(buf)/sizeof(TCHAR)-1,fmt,ap);
    buf [sizeof(buf)/sizeof(TCHAR)-1] = _T('\0');
    va_end (ap);

#if defined(_WIN32_WINNT) && _WIN32_WINNT>=0x0333
    /* this ------------v--- guards NT-specific calls */
    if (check_winnt() && OPENSSL_isservice() > 0)
    {   HANDLE h = RegisterEventSource(0,_T("OPENSSL"));
        const TCHAR *pmsg=buf;
        ReportEvent(h,EVENTLOG_ERROR_TYPE,0,0,0,1,0,&pmsg,0);
        DeregisterEventSource(h);
    }
    else
#endif
        MessageBox (NULL,buf,_T("OpenSSL: FATAL"),MB_OK|MB_ICONSTOP);
}
#else
void OPENSSL_showfatal (const char *fmta,...)
{ va_list ap;

    va_start (ap,fmta);
```



```
    vfprintf (stderr,fmta,ap);
    va_end (ap);
}
int OPENSSL_isservice (void) { return 0; }
#endif

void OpenSSLDie(const char *file,int line,const char *assertion)
        {
        OPENSSL_showfatal(
                "%s(%d): OpenSSL internal error, assertion failed: %s\n",
                file,line,assertion);
#if !defined(_WIN32) || defined(__CYGWIN__)
        abort();
#else
        /* Win32 abort() customarily shows a dialog, but we just did that... */
        raise(SIGABRT);
        _exit(3);
#endif
        }

void *OPENSSL_stderr(void)      { return stderr; }

int CRYPTO_memcmp(const void *in_a, const void *in_b, size_t len)
        {
        size_t i;
        const unsigned char *a = in_a;
        const unsigned char *b = in_b;
        unsigned char x = 0;

        for (i = 0; i < len; i++)
                x |= a[i] ^ b[i];

        return x;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4241 Wed Aug 13 19:52:25 2014
new/usr/src/lib/openssl/libsunw_crypto/cversion.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/cversion.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "cryptlib.h"

#ifndef NO_WINDOWS_BRAINDEATH
```



```
#include "buildinf.h"
#endif

const char *SSLeay_version(int t)
        {
        if (t == SSLEAY_VERSION)
                return OPENSSL_VERSION_TEXT;
        if (t == SSLEAY_BUILT_ON)
                {
#ifdef DATE
                static char buf[sizeof(DATE)+11];

                BIO_snprintf(buf,sizeof buf,"built on: %s",DATE);
                return(buf);
#else
                return("built on: date not available");
#endif
                }
        if (t == SSLEAY_CFLAGS)
                {
#ifdef CFLAGS
                static char buf[sizeof(CFLAGS)+11];

                BIO_snprintf(buf,sizeof buf,"compiler: %s",CFLAGS);
                return(buf);
#else
                return("compiler: information not available");
#endif
                }
        if (t == SSLEAY_PLATFORM)
                {
#ifdef PLATFORM
                static char buf[sizeof(PLATFORM)+11];

                BIO_snprintf(buf,sizeof buf,"platform: %s", PLATFORM);
                return(buf);
#else
                return("platform: information not available");
#endif
                }
        if (t == SSLEAY_DIR)
                {
#ifdef OPENSSLDIR
                return "OPENSSLDIR: \"" OPENSSLDIR "\"";
#else
                return "OPENSSLDIR: N/A";
#endif
                }
        return("not available");
        }

unsigned long SSLeay(void)
        {
        return(SSLEAY_VERSION_NUMBER);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4285 Wed Aug 13 19:52:25 2014
new/usr/src/lib/openssl/libsunw_crypto/des/cbc_cksm.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/cbc_cksm.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

DES_LONG DES_cbc_cksum(const unsigned char *in, DES_cblock *output,
```



```
                       long length, DES_key_schedule *schedule,
                       const_DES_cblock *ivec)
        {
        register DES_LONG tout0,tout1,tin0,tin1;
        register long l=length;
        DES_LONG tin[2];
        unsigned char *out = &(*output)[0];
        const unsigned char *iv = &(*ivec)[0];

        c2l(iv,tout0);
        c2l(iv,tout1);
        for (; l>0; l-=8)
                {
                if (l >= 8)
                        {
                        c2l(in,tin0);
                        c2l(in,tin1);
                        }
                else
                        c2ln(in,tin0,tin1,l);

                tin0^=tout0; tin[0]=tin0;
                tin1^=tout1; tin[1]=tin1;
                DES_encrypt1((DES_LONG *)tin,schedule,DES_ENCRYPT);
                /* fix 15/10/91 eay - thanks to keithr@sco.COM */
                tout0=tin[0];
                tout1=tin[1];
                }
        if (out != NULL)
                {
                l2c(tout0,out);
                l2c(tout1,out);
                }
        tout0=tin0=tin1=tin[0]=tin[1]=0;
        /*
          Transform the data in tout1 so that it will
          match the return value that the MIT Kerberos
          mit_des_cbc_cksum API returns.
        */
        tout1 = ((tout1 >> 24L) & 0x000000FF)
              | ((tout1 >> 8L)  & 0x0000FF00)
              | ((tout1 << 8L)  & 0x00FF0000)
              | ((tout1 << 24L) & 0xFF000000);
        return(tout1);
        }
#endif /* ! codereview */
```

```
**********************************************************
    3270 Wed Aug 13 19:52:26 2014
new/usr/src/lib/openssl/libsunw_crypto/des/cbc_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/cbc_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#define CBC_ENC_C__DONT_UPDATE_IV

#include "ncbc_enc.c" /* des_cbc_encrypt */
```



```
#endif /* ! codereview */
```

```
**********************************************************
    6955 Wed Aug 13 19:52:26 2014
new/usr/src/lib/openssl/libsunw_crypto/des/cfb64ede.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/des/cfb64ede.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"
#include "e_os.h"
```



```
/* The input and output encrypted as though 64bit cfb mode is being
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */

void DES_ede3_cfb64_encrypt(const unsigned char *in, unsigned char *out,
                            long length, DES_key_schedule *ks1,
                            DES_key_schedule *ks2, DES_key_schedule *ks3,
                            DES_cblock *ivec, int *num, int enc)
        {
        register DES_LONG v0,v1;
        register long l=length;
        register int n= *num;
        DES_LONG ti[2];
        unsigned char *iv,c,cc;

        iv=&(*ivec)[0];
        if (enc)
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                c2l(iv,v0);
                                c2l(iv,v1);

                                ti[0]=v0;
                                ti[1]=v1;
                                DES_encrypt3(ti,ks1,ks2,ks3);
                                v0=ti[0];
                                v1=ti[1];

                                iv = &(*ivec)[0];
                                l2c(v0,iv);
                                l2c(v1,iv);
                                iv = &(*ivec)[0];
                                }
                        c= *(in++)^iv[n];
                        *(out++)=c;
                        iv[n]=c;
                        n=(n+1)&0x07;
                        }
                }
        else
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                c2l(iv,v0);
                                c2l(iv,v1);

                                ti[0]=v0;
                                ti[1]=v1;
                                DES_encrypt3(ti,ks1,ks2,ks3);
                                v0=ti[0];
                                v1=ti[1];

                                iv = &(*ivec)[0];
                                l2c(v0,iv);
                                l2c(v1,iv);
                                iv = &(*ivec)[0];
                                }
                        cc= *(in++);
                        c=iv[n];
                        iv[n]=cc;
```

```
128                         *(out++)=c^cc;
129                         n=(n+1)&0x07;
130                         }
131                 }
132         v0=v1=ti[0]=ti[1]=c=cc=0;
133         *num=n;
134         }

136 #ifdef undef /* MACRO */
137 void DES_ede2_cfb64_encrypt(unsigned char *in, unsigned char *out, long length,
138              DES_key_schedule ks1, DES_key_schedule ks2, DES_cblock (*ivec),
139              int *num, int enc)
140         {
141         DES_ede3_cfb64_encrypt(in,out,length,ks1,ks2,ks1,ivec,num,enc);
142         }
143 #endif

145 /* This is compatible with the single key CFB-r for DES, even thought that's
146  * not what EVP needs.
147  */

149 void DES_ede3_cfb_encrypt(const unsigned char *in,unsigned char *out,
150                           int numbits,long length,DES_key_schedule *ks1,
151                           DES_key_schedule *ks2,DES_key_schedule *ks3,
152                           DES_cblock *ivec,int enc)
153         {
154         register DES_LONG d0,d1,v0,v1;
155         register unsigned long l=length,n=((unsigned int)numbits+7)/8;
156         register int num=numbits,i;
157         DES_LONG ti[2];
158         unsigned char *iv;
159         unsigned char ovec[16];

161         if (num > 64) return;
162         iv = &(*ivec)[0];
163         c2l(iv,v0);
164         c2l(iv,v1);
165         if (enc)
166                 {
167                 while (l >= n)
168                         {
169                         l-=n;
170                         ti[0]=v0;
171                         ti[1]=v1;
172                         DES_encrypt3(ti,ks1,ks2,ks3);
173                         c2ln(in,d0,d1,n);
174                         in+=n;
175                         d0^=ti[0];
176                         d1^=ti[1];
177                         l2cn(d0,d1,out,n);
178                         out+=n;
179                         /* 30-08-94 - eay - changed because l>>32 and
180                          * l<<32 are bad under gcc :-( */
181                         if (num == 32)
182                                 { v0=v1; v1=d0; }
183                         else if (num == 64)
184                                 { v0=d0; v1=d1; }
185                         else
186                                 {
187                                 iv=&ovec[0];
188                                 l2c(v0,iv);
189                                 l2c(v1,iv);
190                                 l2c(d0,iv);
191                                 l2c(d1,iv);
192                                 /* shift ovec left most of the bits... */
193                                 memmove(ovec,ovec+num/8,8+(num%8 ? 1 : 0));
```



```
194                                 /* now the remaining bits */
195                                 if(num%8 != 0)
196                                         for(i=0 ; i < 8 ; ++i)
197                                                 {
198                                                 ovec[i]<<=num%8;
199                                                 ovec[i]|=ovec[i+1]>>(8-num%8);
200                                                 }
201                                 iv=&ovec[0];
202                                 c2l(iv,v0);
203                                 c2l(iv,v1);
204                                 }
205                         }
206                 }
207         else
208                 {
209                 while (l >= n)
210                         {
211                         l-=n;
212                         ti[0]=v0;
213                         ti[1]=v1;
214                         DES_encrypt3(ti,ks1,ks2,ks3);
215                         c2ln(in,d0,d1,n);
216                         in+=n;
217                         /* 30-08-94 - eay - changed because l>>32 and
218                          * l<<32 are bad under gcc :-( */
219                         if (num == 32)
220                                 { v0=v1; v1=d0; }
221                         else if (num == 64)
222                                 { v0=d0; v1=d1; }
223                         else
224                                 {
225                                 iv=&ovec[0];
226                                 l2c(v0,iv);
227                                 l2c(v1,iv);
228                                 l2c(d0,iv);
229                                 l2c(d1,iv);
230                                 /* shift ovec left most of the bits... */
231                                 memmove(ovec,ovec+num/8,8+(num%8 ? 1 : 0));
232                                 /* now the remaining bits */
233                                 if(num%8 != 0)
234                                         for(i=0 ; i < 8 ; ++i)
235                                                 {
236                                                 ovec[i]<<=num%8;
237                                                 ovec[i]|=ovec[i+1]>>(8-num%8);
238                                                 }
239                                 iv=&ovec[0];
240                                 c2l(iv,v0);
241                                 c2l(iv,v1);
242                                 }
243                         d0^=ti[0];
244                         d1^=ti[1];
245                         l2cn(d0,d1,out,n);
246                         out+=n;
247                         }
248                 }
249         iv = &(*ivec)[0];
250         l2c(v0,iv);
251         l2c(v1,iv);
252         v0=v1=d0=d1=ti[0]=ti[1]=0;
253         }
254 #endif /* ! codereview */
```

```
**********************************************************
    4401 Wed Aug 13 19:52:26 2014
new/usr/src/lib/openssl/libsunw_crypto/des/cfb64enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/cfb64enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

/* The input and output encrypted as though 64bit cfb mode is being
```



```
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */

void DES_cfb64_encrypt(const unsigned char *in, unsigned char *out,
                       long length, DES_key_schedule *schedule,
                       DES_cblock *ivec, int *num, int enc)
        {
        register DES_LONG v0,v1;
        register long l=length;
        register int n= *num;
        DES_LONG ti[2];
        unsigned char *iv,c,cc;

        iv = &(*ivec)[0];
        if (enc)
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                c2l(iv,v0); ti[0]=v0;
                                c2l(iv,v1); ti[1]=v1;
                                DES_encrypt1(ti,schedule,DES_ENCRYPT);
                                iv = &(*ivec)[0];
                                v0=ti[0]; l2c(v0,iv);
                                v0=ti[1]; l2c(v0,iv);
                                iv = &(*ivec)[0];
                                }
                        c= *(in++)^iv[n];
                        *(out++)=c;
                        iv[n]=c;
                        n=(n+1)&0x07;
                        }
                }
        else
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                c2l(iv,v0); ti[0]=v0;
                                c2l(iv,v1); ti[1]=v1;
                                DES_encrypt1(ti,schedule,DES_ENCRYPT);
                                iv = &(*ivec)[0];
                                v0=ti[0]; l2c(v0,iv);
                                v0=ti[1]; l2c(v0,iv);
                                iv = &(*ivec)[0];
                                }
                        cc= *(in++);
                        c=iv[n];
                        iv[n]=cc;
                        *(out++)=c^cc;
                        n=(n+1)&0x07;
                        }
                }
        v0=v1=ti[0]=ti[1]=c=cc=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    6142 Wed Aug 13 19:52:26 2014
new/usr/src/lib/openssl/libsunw_crypto/des/cfb_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/cfb_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "e_os.h"
#include "des_locl.h"
#include <assert.h>
```



```
/* The input and output are loaded in multiples of 8 bits.
 * What this means is that if you hame numbits=12 and length=2
 * the first 12 bits will be retrieved from the first byte and half
 * the second.  The second 12 bits will come from the 3rd and half the 4th
 * byte.
 */
/* Until Aug 1 2003 this function did not correctly implement CFB-r, so it
 * will not be compatible with any encryption prior to that date. Ben. */
void DES_cfb_encrypt(const unsigned char *in, unsigned char *out, int numbits,
                     long length, DES_key_schedule *schedule, DES_cblock *ivec,
                     int enc)
        {
        register DES_LONG d0,d1,v0,v1;
        register unsigned long l=length;
        register int num=numbits/8,n=(numbits+7)/8,i,rem=numbits%8;
        DES_LONG ti[2];
        unsigned char *iv;
#ifndef L_ENDIAN
        unsigned char ovec[16];
#else
        unsigned int  sh[4];
        unsigned char *ovec=(unsigned char *)sh;

        /* I kind of count that compiler optimizes away this assertioni,*/
        assert (sizeof(sh[0])==4);      /* as this holds true for all,  */
                                        /* but 16-bit platforms...      */

#endif

        if (numbits<=0 || numbits > 64) return;
        iv = &(*ivec)[0];
        c2l(iv,v0);
        c2l(iv,v1);
        if (enc)
                {
                while (l >= (unsigned long)n)
                        {
                        l-=n;
                        ti[0]=v0;
                        ti[1]=v1;
                        DES_encrypt1((DES_LONG *)ti,schedule,DES_ENCRYPT);
                        c2ln(in,d0,d1,n);
                        in+=n;
                        d0^=ti[0];
                        d1^=ti[1];
                        l2cn(d0,d1,out,n);
                        out+=n;
                        /* 30-08-94 - eay - changed because l>>32 and
                         * l<<32 are bad under gcc :-( */
                        if (numbits == 32)
                                { v0=v1; v1=d0; }
                        else if (numbits == 64)
                                { v0=d0; v1=d1; }
                        else
                                {
#ifndef L_ENDIAN
                                iv=&ovec[0];
                                l2c(v0,iv);
                                l2c(v1,iv);
                                l2c(d0,iv);
                                l2c(d1,iv);
#else
                                sh[0]=v0, sh[1]=v1, sh[2]=d0, sh[3]=d1;
#endif
                                if (rem==0)
```

```
128                                     memmove(ovec,ovec+num,8);
129                             else
130                                     for(i=0 ; i < 8 ; ++i)
131                                             ovec[i]=ovec[i+num]<<rem |
132                                                     ovec[i+num+1]>>(8-rem);
133 #ifdef L_ENDIAN
134                             v0=sh[0], v1=sh[1];
135 #else
136                             iv=&ovec[0];
137                             c2l(iv,v0);
138                             c2l(iv,v1);
139 #endif
140                             }
141                     }
142             }
143         else
144             {
145             while (l >= (unsigned long)n)
146                     {
147                     l-=n;
148                     ti[0]=v0;
149                     ti[1]=v1;
150                     DES_encrypt1((DES_LONG *)ti,schedule,DES_ENCRYPT);
151                     c2ln(in,d0,d1,n);
152                     in+=n;
153                     /* 30-08-94 - eay - changed because l>>32 and
154                      * l<<32 are bad under gcc :-( */
155                     if (numbits == 32)
156                             { v0=v1; v1=d0; }
157                     else if (numbits == 64)
158                             { v0=d0; v1=d1; }
159                     else
160                             {
161 #ifndef L_ENDIAN
162                             iv=&ovec[0];
163                             l2c(v0,iv);
164                             l2c(v1,iv);
165                             l2c(d0,iv);
166                             l2c(d1,iv);
167 #else
168                             sh[0]=v0, sh[1]=v1, sh[2]=d0, sh[3]=d1;
169 #endif
170                             if (rem==0)
171                                     memmove(ovec,ovec+num,8);
172                             else
173                                     for(i=0 ; i < 8 ; ++i)
174                                             ovec[i]=ovec[i+num]<<rem |
175                                                     ovec[i+num+1]>>(8-rem);
176 #ifdef L_ENDIAN
177                             v0=sh[0], v1=sh[1];
178 #else
179                             iv=&ovec[0];
180                             c2l(iv,v0);
181                             c2l(iv,v1);
182 #endif
183                             }
184                     d0^=ti[0];
185                     d1^=ti[1];
186                     l2cn(d0,d1,out,n);
187                     out+=n;
188                     }
189             }
190         iv = &(*ivec)[0];
191         l2c(v0,iv);
192         l2c(v1,iv);
193         v0=v1=d0=d1=ti[0]=ti[1]=0;
```



```
194         }
195 #endif /* ! codereview */
```

```
**********************************************************
   10517 Wed Aug 13 19:52:26 2014
new/usr/src/lib/openssl/libsunw_crypto/des/des_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/des_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <des_locl.h>
#include <spr.h>
```



```
void DES_encrypt1(DES_LONG *data, DES_key_schedule *ks, int enc)
        {
        register DES_LONG l,r,t,u;
#ifdef DES_PTR
        register const unsigned char *des_SP=(const unsigned char *)DES_SPtrans;
#endif
#ifndef DES_UNROLL
        register int i;
#endif
        register DES_LONG *s;

        r=data[0];
        l=data[1];

        IP(r,l);
        /* Things have been modified so that the initial rotate is
         * done outside the loop.  This required the
         * DES_SPtrans values in sp.h to be rotated 1 bit to the right.
         * One perl script later and things have a 5% speed up on a sparc2.
         * Thanks to Richard Outerbridge <71755.204@CompuServe.COM>
         * for pointing this out. */
        /* clear the top bits on machines with 8byte longs */
        /* shift left by 2 */
        r=ROTATE(r,29)&0xffffffffL;
        l=ROTATE(l,29)&0xffffffffL;

        s=ks->ks->deslong;
        /* I don't know if it is worth the effort of loop unrolling the
         * inner loop */
        if (enc)
                {
#ifdef DES_UNROLL
                D_ENCRYPT(l,r, 0); /*  1 */
                D_ENCRYPT(r,l, 2); /*  2 */
                D_ENCRYPT(l,r, 4); /*  3 */
                D_ENCRYPT(r,l, 6); /*  4 */
                D_ENCRYPT(l,r, 8); /*  5 */
                D_ENCRYPT(r,l,10); /*  6 */
                D_ENCRYPT(l,r,12); /*  7 */
                D_ENCRYPT(r,l,14); /*  8 */
                D_ENCRYPT(l,r,16); /*  9 */
                D_ENCRYPT(r,l,18); /*  10 */
                D_ENCRYPT(l,r,20); /*  11 */
                D_ENCRYPT(r,l,22); /*  12 */
                D_ENCRYPT(l,r,24); /*  13 */
                D_ENCRYPT(r,l,26); /*  14 */
                D_ENCRYPT(l,r,28); /*  15 */
                D_ENCRYPT(r,l,30); /*  16 */
#else
                for (i=0; i<32; i+=4)
                        {
                        D_ENCRYPT(l,r,i+0); /*  1 */
                        D_ENCRYPT(r,l,i+2); /*  2 */
                        }
#endif
                }
        else
                {
#ifdef DES_UNROLL
                D_ENCRYPT(l,r,30); /* 16 */
                D_ENCRYPT(r,l,28); /* 15 */
                D_ENCRYPT(l,r,26); /* 14 */
                D_ENCRYPT(r,l,24); /* 13 */
                D_ENCRYPT(l,r,22); /* 12 */
                D_ENCRYPT(r,l,20); /* 11 */
                D_ENCRYPT(l,r,18); /* 10 */
```

```
128                 D_ENCRYPT(r,l,16); /*  9 */
129                 D_ENCRYPT(l,r,14); /*  8 */
130                 D_ENCRYPT(r,l,12); /*  7 */
131                 D_ENCRYPT(l,r,10); /*  6 */
132                 D_ENCRYPT(r,l, 8); /*  5 */
133                 D_ENCRYPT(l,r, 6); /*  4 */
134                 D_ENCRYPT(r,l, 4); /*  3 */
135                 D_ENCRYPT(l,r, 2); /*  2 */
136                 D_ENCRYPT(r,l, 0); /*  1 */
137 #else
138                 for (i=30; i>0; i-=4)
139                         {
140                         D_ENCRYPT(l,r,i-0); /* 16 */
141                         D_ENCRYPT(r,l,i-2); /* 15 */
142                         }
143 #endif
144                 }

146         /* rotate and clear the top bits on machines with 8byte longs */
147         l=ROTATE(l,3)&0xffffffffL;
148         r=ROTATE(r,3)&0xffffffffL;

150         FP(r,l);
151         data[0]=l;
152         data[1]=r;
153         l=r=t=u=0;
154         }

156 void DES_encrypt2(DES_LONG *data, DES_key_schedule *ks, int enc)
157         {
158         register DES_LONG l,r,t,u;
159 #ifdef DES_PTR
160         register const unsigned char *des_SP=(const unsigned char *)DES_SPtrans;
161 #endif
162 #ifndef DES_UNROLL
163         register int i;
164 #endif
165         register DES_LONG *s;

167         r=data[0];
168         l=data[1];

170         /* Things have been modified so that the initial rotate is
171          * done outside the loop.  This required the
172          * DES_SPtrans values in sp.h to be rotated 1 bit to the right.
173          * One perl script later and things have a 5% speed up on a sparc2.
174          * Thanks to Richard Outerbridge <71755.204@CompuServe.COM>
175          * for pointing this out. */
176         /* clear the top bits on machines with 8byte longs */
177         r=ROTATE(r,29)&0xffffffffL;
178         l=ROTATE(l,29)&0xffffffffL;

180         s=ks->ks->deslong;
181         /* I don't know if it is worth the effort of loop unrolling the
182          * inner loop */
183         if (enc)
184                 {
185 #ifdef DES_UNROLL
186                 D_ENCRYPT(l,r, 0); /*  1 */
187                 D_ENCRYPT(r,l, 2); /*  2 */
188                 D_ENCRYPT(l,r, 4); /*  3 */
189                 D_ENCRYPT(r,l, 6); /*  4 */
190                 D_ENCRYPT(l,r, 8); /*  5 */
191                 D_ENCRYPT(r,l,10); /*  6 */
192                 D_ENCRYPT(l,r,12); /*  7 */
193                 D_ENCRYPT(r,l,14); /*  8 */
```



```
194                 D_ENCRYPT(l,r,16); /*  9 */
195                 D_ENCRYPT(r,l,18); /*  10 */
196                 D_ENCRYPT(l,r,20); /*  11 */
197                 D_ENCRYPT(r,l,22); /*  12 */
198                 D_ENCRYPT(l,r,24); /*  13 */
199                 D_ENCRYPT(r,l,26); /*  14 */
200                 D_ENCRYPT(l,r,28); /*  15 */
201                 D_ENCRYPT(r,l,30); /*  16 */
202 #else
203                 for (i=0; i<32; i+=4)
204                         {
205                         D_ENCRYPT(l,r,i+0); /*  1 */
206                         D_ENCRYPT(r,l,i+2); /*  2 */
207                         }
208 #endif
209                 }
210         else
211                 {
212 #ifdef DES_UNROLL
213                 D_ENCRYPT(l,r,30); /* 16 */
214                 D_ENCRYPT(r,l,28); /* 15 */
215                 D_ENCRYPT(l,r,26); /* 14 */
216                 D_ENCRYPT(r,l,24); /* 13 */
217                 D_ENCRYPT(l,r,22); /* 12 */
218                 D_ENCRYPT(r,l,20); /* 11 */
219                 D_ENCRYPT(l,r,18); /* 10 */
220                 D_ENCRYPT(r,l,16); /*  9 */
221                 D_ENCRYPT(l,r,14); /*  8 */
222                 D_ENCRYPT(r,l,12); /*  7 */
223                 D_ENCRYPT(l,r,10); /*  6 */
224                 D_ENCRYPT(r,l, 8); /*  5 */
225                 D_ENCRYPT(l,r, 6); /*  4 */
226                 D_ENCRYPT(r,l, 4); /*  3 */
227                 D_ENCRYPT(l,r, 2); /*  2 */
228                 D_ENCRYPT(r,l, 0); /*  1 */
229 #else
230                 for (i=30; i>0; i-=4)
231                         {
232                         D_ENCRYPT(l,r,i-0); /* 16 */
233                         D_ENCRYPT(r,l,i-2); /* 15 */
234                         }
235 #endif
236                 }
237         /* rotate and clear the top bits on machines with 8byte longs */
238         data[0]=ROTATE(l,3)&0xffffffffL;
239         data[1]=ROTATE(r,3)&0xffffffffL;
240         l=r=t=u=0;
241         }

243 void DES_encrypt3(DES_LONG *data, DES_key_schedule *ks1,
244                   DES_key_schedule *ks2, DES_key_schedule *ks3)
245         {
246         register DES_LONG l,r;

248         l=data[0];
249         r=data[1];
250         IP(l,r);
251         data[0]=l;
252         data[1]=r;
253         DES_encrypt2((DES_LONG *)data,ks1,DES_ENCRYPT);
254         DES_encrypt2((DES_LONG *)data,ks2,DES_DECRYPT);
255         DES_encrypt2((DES_LONG *)data,ks3,DES_ENCRYPT);
256         l=data[0];
257         r=data[1];
258         FP(r,l);
259         data[0]=l;
```

```
260         data[1]=r;
261         }

263 void DES_decrypt3(DES_LONG *data, DES_key_schedule *ks1,
264                     DES_key_schedule *ks2, DES_key_schedule *ks3)
265         {
266         register DES_LONG l,r;

268         l=data[0];
269         r=data[1];
270         IP(l,r);
271         data[0]=l;
272         data[1]=r;
273         DES_encrypt2((DES_LONG *)data,ks3,DES_DECRYPT);
274         DES_encrypt2((DES_LONG *)data,ks2,DES_ENCRYPT);
275         DES_encrypt2((DES_LONG *)data,ks1,DES_DECRYPT);
276         l=data[0];
277         r=data[1];
278         FP(r,l);
279         data[0]=l;
280         data[1]=r;
281         }

283 #ifndef DES_DEFAULT_OPTIONS

285 #undef CBC_ENC_C__DONT_UPDATE_IV
286 #include "ncbc_enc.c" /* DES_ncbc_encrypt */

288 void DES_ede3_cbc_encrypt(const unsigned char *input, unsigned char *output,
289                           long length, DES_key_schedule *ks1,
290                           DES_key_schedule *ks2, DES_key_schedule *ks3,
291                           DES_cblock *ivec, int enc)
292         {
293         register DES_LONG tin0,tin1;
294         register DES_LONG tout0,tout1,xor0,xor1;
295         register const unsigned char *in;
296         unsigned char *out;
297         register long l=length;
298         DES_LONG tin[2];
299         unsigned char *iv;

301         in=input;
302         out=output;
303         iv = &(*ivec)[0];

305         if (enc)
306                 {
307                 c2l(iv,tout0);
308                 c2l(iv,tout1);
309                 for (l-=8; l>=0; l-=8)
310                         {
311                         c2l(in,tin0);
312                         c2l(in,tin1);
313                         tin0^=tout0;
314                         tin1^=tout1;

316                         tin[0]=tin0;
317                         tin[1]=tin1;
318                         DES_encrypt3((DES_LONG *)tin,ks1,ks2,ks3);
319                         tout0=tin[0];
320                         tout1=tin[1];

322                         l2c(tout0,out);
323                         l2c(tout1,out);
324                         }
325                 if (l != -8)
```



```
326                         {
327                         c2ln(in,tin0,tin1,l+8);
328                         tin0^=tout0;
329                         tin1^=tout1;

331                         tin[0]=tin0;
332                         tin[1]=tin1;
333                         DES_encrypt3((DES_LONG *)tin,ks1,ks2,ks3);
334                         tout0=tin[0];
335                         tout1=tin[1];

337                         l2c(tout0,out);
338                         l2c(tout1,out);
339                         }
340                 iv = &(*ivec)[0];
341                 l2c(tout0,iv);
342                 l2c(tout1,iv);
343                 }
344         else
345                 {
346                 register DES_LONG t0,t1;

348                 c2l(iv,xor0);
349                 c2l(iv,xor1);
350                 for (l-=8; l>=0; l-=8)
351                         {
352                         c2l(in,tin0);
353                         c2l(in,tin1);

355                         t0=tin0;
356                         t1=tin1;

358                         tin[0]=tin0;
359                         tin[1]=tin1;
360                         DES_decrypt3((DES_LONG *)tin,ks1,ks2,ks3);
361                         tout0=tin[0];
362                         tout1=tin[1];

364                         tout0^=xor0;
365                         tout1^=xor1;
366                         l2c(tout0,out);
367                         l2c(tout1,out);
368                         xor0=t0;
369                         xor1=t1;
370                         }
371                 if (l != -8)
372                         {
373                         c2l(in,tin0);
374                         c2l(in,tin1);

376                         t0=tin0;
377                         t1=tin1;

379                         tin[0]=tin0;
380                         tin[1]=tin1;
381                         DES_decrypt3((DES_LONG *)tin,ks1,ks2,ks3);
382                         tout0=tin[0];
383                         tout1=tin[1];

385                         tout0^=xor0;
386                         tout1^=xor1;
387                         l2cn(tout0,tout1,out,l+8);
388                         xor0=t0;
389                         xor1=t1;
390                         }
```

```
392                 iv = &(*ivec)[0];
393                 l2c(xor0,iv);
394                 l2c(xor1,iv);
395                 }
396         tin0=tin1=tout0=tout1=xor0=xor1=0;
397         tin[0]=tin[1]=0;
398         }

400 #endif /* DES_DEFAULT_OPTIONS */
401 #endif /* ! codereview */
```

```
**********************************************************
   10726 Wed Aug 13 19:52:26 2014
new/usr/src/lib/openssl/libsunw_crypto/des/des_old.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/des_old.c -*- mode:C; c-file-style: "eay" -*- */

/* WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING
 *
 * The function names in here are deprecated and are only present to
 * provide an interface compatible with libdes.  OpenSSL now provides
 * functions where "des_" has been replaced with "DES_" in the names,
 * to make it possible to make incompatible changes that are needed
 * for C type security and other stuff.
 *
 * Please consider starting to use the DES_ functions rather than the
 * des_ ones.  The des_ functions will dissapear completely before
 * OpenSSL 1.0!
 *
 * WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING
 */

/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
```



```
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#define OPENSSL_DES_LIBDES_COMPATIBILITY
#include <openssl/des.h>
#include <openssl/rand.h>

const char *_ossl_old_des_options(void)
        {
        return DES_options();
        }
void _ossl_old_des_ecb3_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
        des_key_schedule ks1,des_key_schedule ks2,
        des_key_schedule ks3, int enc)
        {
        DES_ecb3_encrypt((const_DES_cblock *)input, output,
                (DES_key_schedule *)ks1, (DES_key_schedule *)ks2,
                (DES_key_schedule *)ks3, enc);
        }
DES_LONG _ossl_old_des_cbc_cksum(_ossl_old_des_cblock *input,_ossl_old_des_cbloc
        long length,des_key_schedule schedule,_ossl_old_des_cblock *ivec)
        {
        return DES_cbc_cksum((unsigned char *)input, output, length,
                (DES_key_schedule *)schedule, ivec);
        }
void _ossl_old_des_cbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
        des_key_schedule schedule,_ossl_old_des_cblock *ivec,int enc)
        {
        DES_cbc_encrypt((unsigned char *)input, (unsigned char *)output,
                length, (DES_key_schedule *)schedule, ivec, enc);
        }
void _ossl_old_des_ncbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
        des_key_schedule schedule,_ossl_old_des_cblock *ivec,int enc)
        {
        DES_ncbc_encrypt((unsigned char *)input, (unsigned char *)output,
                length, (DES_key_schedule *)schedule, ivec, enc);
        }
void _ossl_old_des_xcbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
        des_key_schedule schedule,_ossl_old_des_cblock *ivec,
        _ossl_old_des_cblock *inw,_ossl_old_des_cblock *outw,int enc)
        {
        DES_xcbc_encrypt((unsigned char *)input, (unsigned char *)output,
                length, (DES_key_schedule *)schedule, ivec, inw, outw, enc);
        }
void _ossl_old_des_cfb_encrypt(unsigned char *in,unsigned char *out,int numbits,
        long length,des_key_schedule schedule,_ossl_old_des_cblock *ivec,int enc
        {
        DES_cfb_encrypt(in, out, numbits, length,
                (DES_key_schedule *)schedule, ivec, enc);
        }
void _ossl_old_des_ecb_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
        des_key_schedule ks,int enc)
        {
        DES_ecb_encrypt(input, output, (DES_key_schedule *)ks, enc);
        }
void _ossl_old_des_encrypt(DES_LONG *data,des_key_schedule ks, int enc)
```

```
128         {
129         DES_encrypt1(data, (DES_key_schedule *)ks, enc);
130         }
131 void _ossl_old_des_encrypt2(DES_LONG *data,des_key_schedule ks, int enc)
132         {
133         DES_encrypt2(data, (DES_key_schedule *)ks, enc);
134         }
135 void _ossl_old_des_encrypt3(DES_LONG *data, des_key_schedule ks1,
136         des_key_schedule ks2, des_key_schedule ks3)
137         {
138         DES_encrypt3(data, (DES_key_schedule *)ks1, (DES_key_schedule *)ks2,
139                 (DES_key_schedule *)ks3);
140         }
141 void _ossl_old_des_decrypt3(DES_LONG *data, des_key_schedule ks1,
142         des_key_schedule ks2, des_key_schedule ks3)
143         {
144         DES_decrypt3(data, (DES_key_schedule *)ks1, (DES_key_schedule *)ks2,
145                 (DES_key_schedule *)ks3);
146         }
147 void _ossl_old_des_ede3_cbc_encrypt(_ossl_old_des_cblock *input, _ossl_old_des_c
148         long length, des_key_schedule ks1, des_key_schedule ks2,
149         des_key_schedule ks3, _ossl_old_des_cblock *ivec, int enc)
150         {
151         DES_ede3_cbc_encrypt((unsigned char *)input, (unsigned char *)output,
152                 length, (DES_key_schedule *)ks1, (DES_key_schedule *)ks2,
153                 (DES_key_schedule *)ks3, ivec, enc);
154         }
155 void _ossl_old_des_ede3_cfb64_encrypt(unsigned char *in, unsigned char *out,
156         long length, des_key_schedule ks1, des_key_schedule ks2,
157         des_key_schedule ks3, _ossl_old_des_cblock *ivec, int *num, int enc)
158         {
159         DES_ede3_cfb64_encrypt(in, out, length,
160                 (DES_key_schedule *)ks1, (DES_key_schedule *)ks2,
161                 (DES_key_schedule *)ks3, ivec, num, enc);
162         }
163 void _ossl_old_des_ede3_ofb64_encrypt(unsigned char *in, unsigned char *out,
164         long length, des_key_schedule ks1, des_key_schedule ks2,
165         des_key_schedule ks3, _ossl_old_des_cblock *ivec, int *num)
166         {
167         DES_ede3_ofb64_encrypt(in, out, length,
168                 (DES_key_schedule *)ks1, (DES_key_schedule *)ks2,
169                 (DES_key_schedule *)ks3, ivec, num);
170         }

172 #if 0 /* broken code, preserved just in case anyone specifically looks for this
173 void _ossl_old_des_xwhite_in2out(_ossl_old_des_cblock (*des_key), _ossl_old_des_
174         _ossl_old_des_cblock (*out_white))
175         {
176         DES_xwhite_in2out(des_key, in_white, out_white);
177         }
178 #endif

180 int _ossl_old_des_enc_read(int fd,char *buf,int len,des_key_schedule sched,
181         _ossl_old_des_cblock *iv)
182         {
183         return DES_enc_read(fd, buf, len, (DES_key_schedule *)sched, iv);
184         }
185 int _ossl_old_des_enc_write(int fd,char *buf,int len,des_key_schedule sched,
186         _ossl_old_des_cblock *iv)
187         {
188         return DES_enc_write(fd, buf, len, (DES_key_schedule *)sched, iv);
189         }
190 char *_ossl_old_des_fcrypt(const char *buf,const char *salt, char *ret)
191         {
192         return DES_fcrypt(buf, salt, ret);
193         }
```



```
194 char *_ossl_old_des_crypt(const char *buf,const char *salt)
195         {
196         return DES_crypt(buf, salt);
197         }
198 char *_ossl_old_crypt(const char *buf,const char *salt)
199         {
200         return DES_crypt(buf, salt);
201         }
202 void _ossl_old_des_ofb_encrypt(unsigned char *in,unsigned char *out,
203         int numbits,long length,des_key_schedule schedule,_ossl_old_des_cblock *
204         {
205         DES_ofb_encrypt(in, out, numbits, length, (DES_key_schedule *)schedule,
206                 ivec);
207         }
208 void _ossl_old_des_pcbc_encrypt(_ossl_old_des_cblock *input,_ossl_old_des_cblock
209         des_key_schedule schedule,_ossl_old_des_cblock *ivec,int enc)
210         {
211         DES_pcbc_encrypt((unsigned char *)input, (unsigned char *)output,
212                 length, (DES_key_schedule *)schedule, ivec, enc);
213         }
214 DES_LONG _ossl_old_des_quad_cksum(_ossl_old_des_cblock *input,_ossl_old_des_cblo
215         long length,int out_count,_ossl_old_des_cblock *seed)
216         {
217         return DES_quad_cksum((unsigned char *)input, output, length,
218                 out_count, seed);
219         }
220 void _ossl_old_des_random_seed(_ossl_old_des_cblock key)
221         {
222         RAND_seed(key, sizeof(_ossl_old_des_cblock));
223         }
224 void _ossl_old_des_random_key(_ossl_old_des_cblock ret)
225         {
226         DES_random_key((DES_cblock *)ret);
227         }
228 int _ossl_old_des_read_password(_ossl_old_des_cblock *key, const char *prompt,
229                                 int verify)
230         {
231         return DES_read_password(key, prompt, verify);
232         }
233 int _ossl_old_des_read_2passwords(_ossl_old_des_cblock *key1, _ossl_old_des_cblo
234         const char *prompt, int verify)
235         {
236         return DES_read_2passwords(key1, key2, prompt, verify);
237         }
238 void _ossl_old_des_set_odd_parity(_ossl_old_des_cblock *key)
239         {
240         DES_set_odd_parity(key);
241         }
242 int _ossl_old_des_is_weak_key(_ossl_old_des_cblock *key)
243         {
244         return DES_is_weak_key(key);
245         }
246 int _ossl_old_des_set_key(_ossl_old_des_cblock *key,des_key_schedule schedule)
247         {
248         return DES_set_key(key, (DES_key_schedule *)schedule);
249         }
250 int _ossl_old_des_key_sched(_ossl_old_des_cblock *key,des_key_schedule schedule)
251         {
252         return DES_key_sched(key, (DES_key_schedule *)schedule);
253         }
254 void _ossl_old_des_string_to_key(char *str,_ossl_old_des_cblock *key)
255         {
256         DES_string_to_key(str, key);
257         }
258 void _ossl_old_des_string_to_2keys(char *str,_ossl_old_des_cblock *key1,_ossl_ol
259         {
```

```
260         DES_string_to_2keys(str, key1, key2);
261         }
262 void _ossl_old_des_cfb64_encrypt(unsigned char *in, unsigned char *out, long len
263         des_key_schedule schedule, _ossl_old_des_cblock *ivec, int *num, int enc
264         {
265         DES_cfb64_encrypt(in, out, length, (DES_key_schedule *)schedule,
266                 ivec, num, enc);
267         }
268 void _ossl_old_des_ofb64_encrypt(unsigned char *in, unsigned char *out, long len
269         des_key_schedule schedule, _ossl_old_des_cblock *ivec, int *num)
270         {
271         DES_ofb64_encrypt(in, out, length, (DES_key_schedule *)schedule,
272                 ivec, num);
273         }
274 #endif /* ! codereview */
```

```
**********************************************************
    3596 Wed Aug 13 19:52:27 2014
new/usr/src/lib/openssl/libsunw_crypto/des/des_old2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/des_old.c -*- mode:C; c-file-style: "eay" -*- */

/* WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING
 *
 * The function names in here are deprecated and are only present to
 * provide an interface compatible with OpenSSL 0.9.6c.  OpenSSL now
 * provides functions where "des_" has been replaced with "DES_" in
 * the names, to make it possible to make incompatible changes that
 * are needed for C type security and other stuff.
 *
 * Please consider starting to use the DES_ functions rather than the
 * des_ ones.  The des_ functions will dissapear completely before
 * OpenSSL 1.0!
 *
 * WARNING WARNING WARNING WARNING WARNING WARNING WARNING WARNING
 */

/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
```



```
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#undef OPENSSL_DES_LIBDES_COMPATIBILITY
#include <openssl/des.h>
#include <openssl/rand.h>

void _ossl_096_des_random_seed(DES_cblock *key)
        {
        RAND_seed(key, sizeof(DES_cblock));
        }
#endif /* ! codereview */
```

```
**********************************************************
    3688 Wed Aug 13 19:52:27 2014
new/usr/src/lib/openssl/libsunw_crypto/des/ecb3_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/ecb3_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

void DES_ecb3_encrypt(const_DES_cblock *input, DES_cblock *output,
```



```
                      DES_key_schedule *ks1, DES_key_schedule *ks2,
                      DES_key_schedule *ks3,
             int enc)
        {
        register DES_LONG l0,l1;
        DES_LONG ll[2];
        const unsigned char *in = &(*input)[0];
        unsigned char *out = &(*output)[0];

        c2l(in,l0);
        c2l(in,l1);
        ll[0]=l0;
        ll[1]=l1;
        if (enc)
                DES_encrypt3(ll,ks1,ks2,ks3);
        else
                DES_decrypt3(ll,ks1,ks2,ks3);
        l0=ll[0];
        l1=ll[1];
        l2c(l0,out);
        l2c(l1,out);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4383 Wed Aug 13 19:52:27 2014
new/usr/src/lib/openssl/libsunw_crypto/des/ecb_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/ecb_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"
#include "des_ver.h"
#include <openssl/opensslv.h>
```



```
#include <openssl/bio.h>

OPENSSL_GLOBAL const char libdes_version[]="libdes" OPENSSL_VERSION_PTEXT;
OPENSSL_GLOBAL const char DES_version[]="DES" OPENSSL_VERSION_PTEXT;

const char *DES_options(void)
        {
        static int init=1;
        static char buf[32];

        if (init)
                {
                const char *ptr,*unroll,*risc,*size;

#ifdef DES_PTR
                ptr="ptr";
#else
                ptr="idx";
#endif
#if defined(DES_RISC1) || defined(DES_RISC2)
#ifdef DES_RISC1
                risc="risc1";
#endif
#ifdef DES_RISC2
                risc="risc2";
#endif
#else
                risc="cisc";
#endif
#ifdef DES_UNROLL
                unroll="16";
#else
                unroll="2";
#endif
                if (sizeof(DES_LONG) != sizeof(long))
                        size="int";
                else
                        size="long";
                BIO_snprintf(buf,sizeof buf,"des(%s,%s,%s,%s)",ptr,risc,unroll,
                             size);
                init=0;
                }
        return(buf);
        }


void DES_ecb_encrypt(const_DES_cblock *input, DES_cblock *output,
                     DES_key_schedule *ks, int enc)
        {
        register DES_LONG l;
        DES_LONG ll[2];
        const unsigned char *in = &(*input)[0];
        unsigned char *out = &(*output)[0];

        c2l(in,l); ll[0]=l;
        c2l(in,l); ll[1]=l;
        DES_encrypt1(ll,ks,enc);
        l=ll[0]; l2c(l,out);
        l=ll[1]; l2c(l,out);
        l=ll[0]=ll[1]=0;
        }
#endif /* ! codereview */
```

```
**********************************************************
    5240 Wed Aug 13 19:52:27 2014
new/usr/src/lib/openssl/libsunw_crypto/des/ede_cbcm_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ede_cbcm_enc.c */
/* Written by Ben Laurie <ben@algroup.co.uk> for the OpenSSL
 * project 13 Feb 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/*

This is an implementation of Triple DES Cipher Block Chaining with Output
```



```
Feedback Masking, by Coppersmith, Johnson and Matyas, (IBM and Certicom).

Note that there is a known attack on this by Biham and Knudsen but it takes
a lot of work:

http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/1998/CS/CS0928.ps.gz

*/

#include <openssl/opensslconf.h> /* To see if OPENSSL_NO_DESCBCM is defined */

#ifndef OPENSSL_NO_DESCBCM
#include "des_locl.h"

void DES_ede3_cbcm_encrypt(const unsigned char *in, unsigned char *out,
             long length, DES_key_schedule *ks1, DES_key_schedule *ks2,
             DES_key_schedule *ks3, DES_cblock *ivec1, DES_cblock *ivec2,
             int enc)
    {
    register DES_LONG tin0,tin1;
    register DES_LONG tout0,tout1,xor0,xor1,m0,m1;
    register long l=length;
    DES_LONG tin[2];
    unsigned char *iv1,*iv2;

    iv1 = &(*ivec1)[0];
    iv2 = &(*ivec2)[0];

    if (enc)
        {
        c2l(iv1,m0);
        c2l(iv1,m1);
        c2l(iv2,tout0);
        c2l(iv2,tout1);
        for (l-=8; l>=-7; l-=8)
            {
            tin[0]=m0;
            tin[1]=m1;
            DES_encrypt1(tin,ks3,1);
            m0=tin[0];
            m1=tin[1];

            if(l < 0)
                {
                c2ln(in,tin0,tin1,l+8);
                }
            else
                {
                c2l(in,tin0);
                c2l(in,tin1);
                }
            tin0^=tout0;
            tin1^=tout1;

            tin[0]=tin0;
            tin[1]=tin1;
            DES_encrypt1(tin,ks1,1);
            tin[0]^=m0;
            tin[1]^=m1;
            DES_encrypt1(tin,ks2,0);
            tin[0]^=m0;
            tin[1]^=m1;
            DES_encrypt1(tin,ks1,1);
            tout0=tin[0];
            tout1=tin[1];
```

```
128             l2c(tout0,out);
129             l2c(tout1,out);
130             }
131         iv1=&(*ivec1)[0];
132         l2c(m0,iv1);
133         l2c(m1,iv1);

135         iv2=&(*ivec2)[0];
136         l2c(tout0,iv2);
137         l2c(tout1,iv2);
138         }
139     else
140         {
141         register DES_LONG t0,t1;

143         c2l(iv1,m0);
144         c2l(iv1,m1);
145         c2l(iv2,xor0);
146         c2l(iv2,xor1);
147         for (l-=8; l>=-7; l-=8)
148             {
149             tin[0]=m0;
150             tin[1]=m1;
151             DES_encrypt1(tin,ks3,1);
152             m0=tin[0];
153             m1=tin[1];

155             c2l(in,tin0);
156             c2l(in,tin1);

158             t0=tin0;
159             t1=tin1;

161             tin[0]=tin0;
162             tin[1]=tin1;
163             DES_encrypt1(tin,ks1,0);
164             tin[0]^=m0;
165             tin[1]^=m1;
166             DES_encrypt1(tin,ks2,1);
167             tin[0]^=m0;
168             tin[1]^=m1;
169             DES_encrypt1(tin,ks1,0);
170             tout0=tin[0];
171             tout1=tin[1];

173             tout0^=xor0;
174             tout1^=xor1;
175             if(l < 0)
176                 {
177                 l2cn(tout0,tout1,out,l+8);
178                 }
179             else
180                 {
181                 l2c(tout0,out);
182                 l2c(tout1,out);
183                 }
184             xor0=t0;
185             xor1=t1;
186             }

188         iv1=&(*ivec1)[0];
189         l2c(m0,iv1);
190         l2c(m1,iv1);

192         iv2=&(*ivec2)[0];
193         l2c(xor0,iv2);
```



```
194         l2c(xor1,iv2);
195         }
196     tin0=tin1=tout0=tout1=xor0=xor1=0;
197     tin[0]=tin[1]=0;
198     }
199 #endif
200 #endif /* ! codereview */
```

```
**********************************************************
    7549 Wed Aug 13 19:52:27 2014
new/usr/src/lib/openssl/libsunw_crypto/des/enc_read.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/enc_read.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include "des_locl.h"

/* This has some uglies in it but it works - even over sockets. */
/*extern int errno;*/
OPENSSL_IMPLEMENT_GLOBAL(int,DES_rw_mode,DES_PCBC_MODE)


/*
 * WARNINGS:
 *
 *  -  The data format used by DES_enc_write() and DES_enc_read()
 *     has a cryptographic weakness: When asked to write more
 *     than MAXWRITE bytes, DES_enc_write will split the data
 *     into several chunks that are all encrypted
 *     using the same IV.  So don't use these functions unless you
 *     are sure you know what you do (in which case you might
 *     not want to use them anyway).
 *
 *  -  This code cannot handle non-blocking sockets.
 *
 *  -  This function uses an internal state and thus cannot be
 *     used on multiple files.
 */


int DES_enc_read(int fd, void *buf, int len, DES_key_schedule *sched,
                 DES_cblock *iv)
        {
#if defined(OPENSSL_NO_POSIX_IO)
        return(0);
#else
        /* data to be unencrypted */
        int net_num=0;
        static unsigned char *net=NULL;
        /* extra unencrypted data
         * for when a block of 100 comes in but is des_read one byte at
         * a time. */
        static unsigned char *unnet=NULL;
        static int unnet_start=0;
        static int unnet_left=0;
        static unsigned char *tmpbuf=NULL;
        int i;
        long num=0,rnum;
        unsigned char *p;

        if (tmpbuf == NULL)
                {
                tmpbuf=OPENSSL_malloc(BSIZE);
                if (tmpbuf == NULL) return(-1);
                }
        if (net == NULL)
                {
                net=OPENSSL_malloc(BSIZE);
                if (net == NULL) return(-1);
                }
        if (unnet == NULL)
                {
                unnet=OPENSSL_malloc(BSIZE);
                if (unnet == NULL) return(-1);
                }
        /* left over data from last decrypt */
        if (unnet_left != 0)
                {
                if (unnet_left < len)
                        {
                        /* we still still need more data but will return
```

```
128                          * with the number of bytes we have - should always
129                          * check the return value */
130                         memcpy(buf,&(unnet[unnet_start]),
131                                unnet_left);
132                         /* eay 26/08/92 I had the next 2 lines
133                          * reversed :-( */
134                         i=unnet_left;
135                         unnet_start=unnet_left=0;
136                         }
137                 else
138                         {
139                         memcpy(buf,&(unnet[unnet_start]),len);
140                         unnet_start+=len;
141                         unnet_left-=len;
142                         i=len;
143                         }
144                 return(i);
145                 }

147         /* We need to get more data. */
148         if (len > MAXWRITE) len=MAXWRITE;

150         /* first - get the length */
151         while (net_num < HDRSIZE)
152                 {
153 #ifndef OPENSSL_SYS_WIN32
154                 i=read(fd,(void *)&(net[net_num]),HDRSIZE-net_num);
155 #else
156                 i=_read(fd,(void *)&(net[net_num]),HDRSIZE-net_num);
157 #endif
158 #ifdef EINTR
159                 if ((i == -1) && (errno == EINTR)) continue;
160 #endif
161                 if (i <= 0) return(0);
162                 net_num+=i;
163                 }

165         /* we now have at net_num bytes in net */
166         p=net;
167         /* num=0;  */
168         n2l(p,num);
169         /* num should be rounded up to the next group of eight
170          * we make sure that we have read a multiple of 8 bytes from the net.
171          */
172         if ((num > MAXWRITE) || (num < 0)) /* error */
173                 return(-1);
174         rnum=(num < 8)?8:((num+7)/8*8);

176         net_num=0;
177         while (net_num < rnum)
178                 {
179 #ifndef OPENSSL_SYS_WIN32
180                 i=read(fd,(void *)&(net[net_num]),rnum-net_num);
181 #else
182                 i=_read(fd,(void *)&(net[net_num]),rnum-net_num);
183 #endif
184 #ifdef EINTR
185                 if ((i == -1) && (errno == EINTR)) continue;
186 #endif
187                 if (i <= 0) return(0);
188                 net_num+=i;
189                 }

191         /* Check if there will be data left over. */
192         if (len < num)
193                 {
```



```
194                 if (DES_rw_mode & DES_PCBC_MODE)
195                         DES_pcbc_encrypt(net,unnet,num,sched,iv,DES_DECRYPT);
196                 else
197                         DES_cbc_encrypt(net,unnet,num,sched,iv,DES_DECRYPT);
198                 memcpy(buf,unnet,len);
199                 unnet_start=len;
200                 unnet_left=num-len;

202                 /* The following line is done because we return num
203                  * as the number of bytes read. */
204                 num=len;
205                 }
206         else
207                 {
208                 /* >output is a multiple of 8 byes, if len < rnum
209                  * >we must be careful.  The user must be aware that this
210                  * >routine will write more bytes than he asked for.
211                  * >The length of the buffer must be correct.
212                  * FIXED - Should be ok now 18-9-90 - eay */
213                 if (len < rnum)
214                         {

216                         if (DES_rw_mode & DES_PCBC_MODE)
217                                 DES_pcbc_encrypt(net,tmpbuf,num,sched,iv,
218                                                  DES_DECRYPT);
219                         else
220                                 DES_cbc_encrypt(net,tmpbuf,num,sched,iv,
221                                                 DES_DECRYPT);

223                         /* eay 26/08/92 fix a bug that returned more
224                          * bytes than you asked for (returned len bytes :-( */
225                         memcpy(buf,tmpbuf,num);
226                         }
227                 else
228                         {
229                         if (DES_rw_mode & DES_PCBC_MODE)
230                                 DES_pcbc_encrypt(net,buf,num,sched,iv,
231                                                  DES_DECRYPT);
232                         else
233                                 DES_cbc_encrypt(net,buf,num,sched,iv,
234                                                 DES_DECRYPT);
235                         }
236                 }
237         return num;
238 #endif /* OPENSSL_NO_POSIX_IO */
239         }
240 #endif /* ! codereview */
```

```
**********************************************************
    5750 Wed Aug 13 19:52:27 2014
new/usr/src/lib/openssl/libsunw_crypto/des/enc_writ.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/enc_writ.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <errno.h>
#include <time.h>
#include <stdio.h>
```



```
#include "cryptlib.h"
#include "des_locl.h"
#include <openssl/rand.h>

/*
 * WARNINGS:
 *
 *  -  The data format used by DES_enc_write() and DES_enc_read()
 *     has a cryptographic weakness: When asked to write more
 *     than MAXWRITE bytes, DES_enc_write will split the data
 *     into several chunks that are all encrypted
 *     using the same IV.  So don't use these functions unless you
 *     are sure you know what you do (in which case you might
 *     not want to use them anyway).
 *
 *  -  This code cannot handle non-blocking sockets.
 */

int DES_enc_write(int fd, const void *_buf, int len,
                  DES_key_schedule *sched, DES_cblock *iv)
        {
#if defined(OPENSSL_NO_POSIX_IO)
        return (-1);
#else
#ifdef _LIBC
        extern unsigned long time();
        extern int write();
#endif
        const unsigned char *buf=_buf;
        long rnum;
        int i,j,k,outnum;
        static unsigned char *outbuf=NULL;
        unsigned char shortbuf[8];
        unsigned char *p;
        const unsigned char *cp;
        static int start=1;

        if (outbuf == NULL)
                {
                outbuf=OPENSSL_malloc(BSIZE+HDRSIZE);
                if (outbuf == NULL) return(-1);
                }
        /* If we are sending less than 8 bytes, the same char will look
         * the same if we don't pad it out with random bytes */
        if (start)
                {
                start=0;
                }

        /* lets recurse if we want to send the data in small chunks */
        if (len > MAXWRITE)
                {
                j=0;
                for (i=0; i<len; i+=k)
                        {
                        k=DES_enc_write(fd,&(buf[i]),
                                ((len-i) > MAXWRITE)?MAXWRITE:(len-i),sched,iv);
                        if (k < 0)
                                return(k);
                        else
                                j+=k;
                        }
                return(j);
                }

        /* write length first */
```

```
128         p=outbuf;
129         l2n(len,p);

131         /* pad short strings */
132         if (len < 8)
133                 {
134                 cp=shortbuf;
135                 memcpy(shortbuf,buf,len);
136                 RAND_pseudo_bytes(shortbuf+len, 8-len);
137                 rnum=8;
138                 }
139         else
140                 {
141                 cp=buf;
142                 rnum=((len+7)/8*8); /* round up to nearest eight */
143                 }

145         if (DES_rw_mode & DES_PCBC_MODE)
146                 DES_pcbc_encrypt(cp,&(outbuf[HDRSIZE]),(len<8)?8:len,sched,iv,
147                                  DES_ENCRYPT);
148         else
149                 DES_cbc_encrypt(cp,&(outbuf[HDRSIZE]),(len<8)?8:len,sched,iv,
150                                 DES_ENCRYPT);

152         /* output */
153         outnum=rnum+HDRSIZE;

155         for (j=0; j<outnum; j+=i)
156                 {
157                 /* eay 26/08/92 I was not doing writing from where we
158                  * got up to. */
159 #ifndef _WIN32
160                 i=write(fd,(void *)&(outbuf[j]),outnum-j);
161 #else
162                 i=_write(fd,(void *)&(outbuf[j]),outnum-j);
163 #endif
164                 if (i == -1)
165                         {
166 #ifdef EINTR
167                         if (errno == EINTR)
168                                 i=0;
169                         else
170 #endif
171                                 /* This is really a bad error - very bad
172                                  * It will stuff-up both ends. */
173                                 return(-1);
174                         }
175                 }

177         return(len);
178 #endif /* OPENSSL_NO_POSIX_IO */
179         }
180 #endif /* ! codereview */
```

```
**********************************************************
    4233 Wed Aug 13 19:52:28 2014
new/usr/src/lib/openssl/libsunw_crypto/des/fcrypt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* NOCW */
#include <stdio.h>
#ifdef _OSD_POSIX
#ifndef CHARSET_EBCDIC
#define CHARSET_EBCDIC 1
#endif
#endif
#ifdef CHARSET_EBCDIC
#include <openssl/ebcdic.h>
#endif

/* This version of crypt has been developed from my MIT compatible
 * DES library.
 * Eric Young (eay@cryptsoft.com)
 */

/* Modification by Jens Kupferschmidt (Cu)
 * I have included directive PARA for shared memory computers.
 * I have included a directive LONGCRYPT to using this routine to cipher
 * passwords with more then 8 bytes like HP-UX 10.x it used. The MAXPLEN
 * definition is the maximum of length of password and can changed. I have
 * defined 24.
 */

#include "des_locl.h"

/* Added more values to handle illegal salt values the way normal
 * crypt() implementations do.  The patch was sent by
 * Bjorn Gronvall <bg@sics.se>
 */
static unsigned const char con_salt[128]={
0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,
0xDA,0xDB,0xDC,0xDD,0xDE,0xDF,0xE0,0xE1,
0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,
0xEA,0xEB,0xEC,0xED,0xEE,0xEF,0xF0,0xF1,
0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,
0xFA,0xFB,0xFC,0xFD,0xFE,0xFF,0x00,0x01,
0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,
0x0A,0x0B,0x05,0x06,0x07,0x08,0x09,0x0A,
0x0B,0x0C,0x0D,0x0E,0x0F,0x10,0x11,0x12,
0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,
0x1B,0x1C,0x1D,0x1E,0x1F,0x20,0x21,0x22,
0x23,0x24,0x25,0x20,0x21,0x22,0x23,0x24,
0x25,0x26,0x27,0x28,0x29,0x2A,0x2B,0x2C,
0x2D,0x2E,0x2F,0x30,0x31,0x32,0x33,0x34,
0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,0x3C,
0x3D,0x3E,0x3F,0x40,0x41,0x42,0x43,0x44,
};

static unsigned const char cov_2char[64]={
0x2E,0x2F,0x30,0x31,0x32,0x33,0x34,0x35,
0x36,0x37,0x38,0x39,0x41,0x42,0x43,0x44,
0x45,0x46,0x47,0x48,0x49,0x4A,0x4B,0x4C,
0x4D,0x4E,0x4F,0x50,0x51,0x52,0x53,0x54,
0x55,0x56,0x57,0x58,0x59,0x5A,0x61,0x62,
0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6A,
0x6B,0x6C,0x6D,0x6E,0x6F,0x70,0x71,0x72,
0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7A
};

char *DES_crypt(const char *buf, const char *salt)
```



```
        {
        static char buff[14];

#ifndef CHARSET_EBCDIC
        return(DES_fcrypt(buf,salt,buff));
#else
        char e_salt[2+1];
        char e_buf[32+1];       /* replace 32 by 8 ? */
        char *ret;

        /* Copy at most 2 chars of salt */
        if ((e_salt[0] = salt[0]) != '\0')
            e_salt[1] = salt[1];

        /* Copy at most 32 chars of password */
        strncpy (e_buf, buf, sizeof(e_buf));

        /* Make sure we have a delimiter */
        e_salt[sizeof(e_salt)-1] = e_buf[sizeof(e_buf)-1] = '\0';

        /* Convert the e_salt to ASCII, as that's what DES_fcrypt works on */
        ebcdic2ascii(e_salt, e_salt, sizeof e_salt);

        /* Convert the cleartext password to ASCII */
        ebcdic2ascii(e_buf, e_buf, sizeof e_buf);

        /* Encrypt it (from/to ASCII) */
        ret = DES_fcrypt(e_buf,e_salt,buff);

        /* Convert the result back to EBCDIC */
        ascii2ebcdic(ret, ret, strlen(ret));

        return ret;
#endif
        }


char *DES_fcrypt(const char *buf, const char *salt, char *ret)
        {
        unsigned int i,j,x,y;
        DES_LONG Eswap0,Eswap1;
        DES_LONG out[2],ll;
        DES_cblock key;
        DES_key_schedule ks;
        unsigned char bb[9];
        unsigned char *b=bb;
        unsigned char c,u;

        /* eay 25/08/92
         * If you call crypt("pwd","*") as often happens when you
         * have * as the pwd field in /etc/passwd, the function
         * returns *\0XXXXXXXXX
         * The \0 makes the string look like * so the pwd "*" would
         * crypt to "*".  This was found when replacing the crypt in
         * our shared libraries.  People found that the disabled
         * accounts effectively had no passwd :-(. */
#ifndef CHARSET_EBCDIC
        x=ret[0]=((salt[0] == '\0')?'A':salt[0]);
        Eswap0=con_salt[x]<<2;
        x=ret[1]=((salt[1] == '\0')?'A':salt[1]);
        Eswap1=con_salt[x]<<6;
#else
        x=ret[0]=((salt[0] == '\0')?os_toascii['A']:salt[0]);
        Eswap0=con_salt[x]<<2;
        x=ret[1]=((salt[1] == '\0')?os_toascii['A']:salt[1]);
        Eswap1=con_salt[x]<<6;
```

```
#endif

/* EAY
r=strlen(buf);
r=(r+7)/8;
*/
        for (i=0; i<8; i++)
                {
                c= *(buf++);
                if (!c) break;
                key[i]=(c<<1);
                }
        for (; i<8; i++)
                key[i]=0;

        DES_set_key_unchecked(&key,&ks);
        fcrypt_body(&(out[0]),&ks,Eswap0,Eswap1);

        ll=out[0]; l2c(ll,b);
        ll=out[1]; l2c(ll,b);
        y=0;
        u=0x80;
        bb[8]=0;
        for (i=2; i<13; i++)
                {
                c=0;
                for (j=0; j<6; j++)
                        {
                        c<<=1;
                        if (bb[y] & u) c|=1;
                        u>>=1;
                        if (!u)
                                {
                                y++;
                                u=0x80;
                                }
                        }
                ret[i]=cov_2char[c];
                }
        ret[13]='\0';
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4943 Wed Aug 13 19:52:28 2014
new/usr/src/lib/openssl/libsunw_crypto/des/fcrypt_b.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/fcrypt_b.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>

/* This version of crypt has been developed from my MIT compatible
```



```
 * DES library.
 * The library is available at pub/Crypto/DES at ftp.psy.uq.oz.au
 * Eric Young (eay@cryptsoft.com)
 */

#define DES_FCRYPT
#include "des_locl.h"
#undef DES_FCRYPT

#undef PERM_OP
#define PERM_OP(a,b,t,n,m) ((t)=((((a)>>(n))^(b))&(m)),\
        (b)^=(t),\
        (a)^=((t)<<(n)))

#undef HPERM_OP
#define HPERM_OP(a,t,n,m) ((t)=((((a)<<(16-(n)))^(a))&(m)),\
        (a)=(a)^(t)^(t>>(16-(n))))\

void fcrypt_body(DES_LONG *out, DES_key_schedule *ks, DES_LONG Eswap0,
                 DES_LONG Eswap1)
        {
        register DES_LONG l,r,t,u;
#ifdef DES_PTR
        register const unsigned char *des_SP=(const unsigned char *)DES_SPtrans;
#endif
        register DES_LONG *s;
        register int j;
        register DES_LONG E0,E1;

        l=0;
        r=0;

        s=(DES_LONG *)ks;
        E0=Eswap0;
        E1=Eswap1;

        for (j=0; j<25; j++)
                {
#ifndef DES_UNROLL
                register int i;

                for (i=0; i<32; i+=4)
                        {
                        D_ENCRYPT(l,r,i+0); /*  1 */
                        D_ENCRYPT(r,l,i+2); /*  2 */
                        }
#else
                D_ENCRYPT(l,r, 0); /*  1 */
                D_ENCRYPT(r,l, 2); /*  2 */
                D_ENCRYPT(l,r, 4); /*  3 */
                D_ENCRYPT(r,l, 6); /*  4 */
                D_ENCRYPT(l,r, 8); /*  5 */
                D_ENCRYPT(r,l,10); /*  6 */
                D_ENCRYPT(l,r,12); /*  7 */
                D_ENCRYPT(r,l,14); /*  8 */
                D_ENCRYPT(l,r,16); /*  9 */
                D_ENCRYPT(r,l,18); /*  10 */
                D_ENCRYPT(l,r,20); /*  11 */
                D_ENCRYPT(r,l,22); /*  12 */
                D_ENCRYPT(l,r,24); /*  13 */
                D_ENCRYPT(r,l,26); /*  14 */
                D_ENCRYPT(l,r,28); /*  15 */
                D_ENCRYPT(r,l,30); /*  16 */
#endif

                t=l;
```

```
128                 l=r;
129                 r=t;
130                 }
131         l=ROTATE(l,3)&0xffffffffL;
132         r=ROTATE(r,3)&0xffffffffL;

134         PERM_OP(l,r,t, 1,0x55555555L);
135         PERM_OP(r,l,t, 8,0x00ff00ffL);
136         PERM_OP(l,r,t, 2,0x33333333L);
137         PERM_OP(r,l,t,16,0x0000ffffL);
138         PERM_OP(l,r,t, 4,0x0f0f0f0fL);

140         out[0]=r;
141         out[1]=l;
142         }
143 #endif /* ! codereview */
```

```
**********************************************************
    5169 Wed Aug 13 19:52:28 2014
new/usr/src/lib/openssl/libsunw_crypto/des/ncbc_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/ncbc_enc.c */
/*
 * #included by:
 *    cbc_enc.c  (DES_cbc_encrypt)
 *    des_enc.c  (DES_ncbc_encrypt)
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
```



```
 */

#include "des_locl.h"

#ifdef CBC_ENC_C__DONT_UPDATE_IV
void DES_cbc_encrypt(const unsigned char *in, unsigned char *out, long length,
                     DES_key_schedule *_schedule, DES_cblock *ivec, int enc)
#else
void DES_ncbc_encrypt(const unsigned char *in, unsigned char *out, long length,
                     DES_key_schedule *_schedule, DES_cblock *ivec, int enc)
#endif
        {
        register DES_LONG tin0,tin1;
        register DES_LONG tout0,tout1,xor0,xor1;
        register long l=length;
        DES_LONG tin[2];
        unsigned char *iv;

        iv = &(*ivec)[0];

        if (enc)
                {
                c2l(iv,tout0);
                c2l(iv,tout1);
                for (l-=8; l>=0; l-=8)
                        {
                        c2l(in,tin0);
                        c2l(in,tin1);
                        tin0^=tout0; tin[0]=tin0;
                        tin1^=tout1; tin[1]=tin1;
                        DES_encrypt1((DES_LONG *)tin,_schedule,DES_ENCRYPT);
                        tout0=tin[0]; l2c(tout0,out);
                        tout1=tin[1]; l2c(tout1,out);
                        }
                if (l != -8)
                        {
                        c2ln(in,tin0,tin1,l+8);
                        tin0^=tout0; tin[0]=tin0;
                        tin1^=tout1; tin[1]=tin1;
                        DES_encrypt1((DES_LONG *)tin,_schedule,DES_ENCRYPT);
                        tout0=tin[0]; l2c(tout0,out);
                        tout1=tin[1]; l2c(tout1,out);
                        }
#ifndef CBC_ENC_C__DONT_UPDATE_IV
                iv = &(*ivec)[0];
                l2c(tout0,iv);
                l2c(tout1,iv);
#endif
                }
        else
                {
                c2l(iv,xor0);
                c2l(iv,xor1);
                for (l-=8; l>=0; l-=8)
                        {
                        c2l(in,tin0); tin[0]=tin0;
                        c2l(in,tin1); tin[1]=tin1;
                        DES_encrypt1((DES_LONG *)tin,_schedule,DES_DECRYPT);
                        tout0=tin[0]^xor0;
                        tout1=tin[1]^xor1;
                        l2c(tout0,out);
                        l2c(tout1,out);
                        xor0=tin0;
                        xor1=tin1;
                        }
                if (l != -8)
```

```
128                         {
129                         c2l(in,tin0); tin[0]=tin0;
130                         c2l(in,tin1); tin[1]=tin1;
131                         DES_encrypt1((DES_LONG *)tin,_schedule,DES_DECRYPT);
132                         tout0=tin[0]^xor0;
133                         tout1=tin[1]^xor1;
134                         l2cn(tout0,tout1,out,l+8);
135 #ifndef CBC_ENC_C__DONT_UPDATE_IV
136                         xor0=tin0;
137                         xor1=tin1;
138 #endif
139                         }
140 #ifndef CBC_ENC_C__DONT_UPDATE_IV
141                 iv = &(*ivec)[0];
142                 l2c(xor0,iv);
143                 l2c(xor1,iv);
144 #endif
145                 }
146         tin0=tin1=tout0=tout1=xor0=xor1=0;
147         tin[0]=tin[1]=0;
148         }
149 #endif /* ! codereview */
```

```
**********************************************************
    4551 Wed Aug 13 19:52:28 2014
new/usr/src/lib/openssl/libsunw_crypto/des/ofb64ede.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/ofb64ede.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

/* The input and output encrypted as though 64bit ofb mode is being
```



```
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */
void DES_ede3_ofb64_encrypt(register const unsigned char *in,
                            register unsigned char *out, long length,
                            DES_key_schedule *k1, DES_key_schedule *k2,
                            DES_key_schedule *k3, DES_cblock *ivec,
                            int *num)
        {
        register DES_LONG v0,v1;
        register int n= *num;
        register long l=length;
        DES_cblock d;
        register char *dp;
        DES_LONG ti[2];
        unsigned char *iv;
        int save=0;

        iv = &(*ivec)[0];
        c2l(iv,v0);
        c2l(iv,v1);
        ti[0]=v0;
        ti[1]=v1;
        dp=(char *)d;
        l2c(v0,dp);
        l2c(v1,dp);
        while (l--)
                {
                if (n == 0)
                        {
                        /* ti[0]=v0; */
                        /* ti[1]=v1; */
                        DES_encrypt3(ti,k1,k2,k3);
                        v0=ti[0];
                        v1=ti[1];

                        dp=(char *)d;
                        l2c(v0,dp);
                        l2c(v1,dp);
                        save++;
                        }
                *(out++)= *(in++)^d[n];
                n=(n+1)&0x07;
                }
        if (save)
                {
/*              v0=ti[0];
                v1=ti[1];*/
                iv = &(*ivec)[0];
                l2c(v0,iv);
                l2c(v1,iv);
                }
        v0=v1=ti[0]=ti[1]=0;
        *num=n;
        }

#ifdef undef /* MACRO */
void DES_ede2_ofb64_encrypt(register unsigned char *in,
             register unsigned char *out, long length, DES_key_schedule k1,
             DES_key_schedule k2, DES_cblock (*ivec), int *num)
        {
        DES_ede3_ofb64_encrypt(in, out, length, k1,k2,k1, ivec, num);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4170 Wed Aug 13 19:52:28 2014
new/usr/src/lib/openssl/libsunw_crypto/des/ofb64enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/ofb64enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

/* The input and output encrypted as though 64bit ofb mode is being
```



```
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */
void DES_ofb64_encrypt(register const unsigned char *in,
                       register unsigned char *out, long length,
                       DES_key_schedule *schedule, DES_cblock *ivec, int *num)
        {
        register DES_LONG v0,v1,t;
        register int n= *num;
        register long l=length;
        DES_cblock d;
        register unsigned char *dp;
        DES_LONG ti[2];
        unsigned char *iv;
        int save=0;

        iv = &(*ivec)[0];
        c2l(iv,v0);
        c2l(iv,v1);
        ti[0]=v0;
        ti[1]=v1;
        dp=d;
        l2c(v0,dp);
        l2c(v1,dp);
        while (l--)
                {
                if (n == 0)
                        {
                        DES_encrypt1(ti,schedule,DES_ENCRYPT);
                        dp=d;
                        t=ti[0]; l2c(t,dp);
                        t=ti[1]; l2c(t,dp);
                        save++;
                        }
                *(out++)= *(in++)^d[n];
                n=(n+1)&0x07;
                }
        if (save)
                {
                v0=ti[0];
                v1=ti[1];
                iv = &(*ivec)[0];
                l2c(v0,iv);
                l2c(v1,iv);
                }
        t=v0=v1=ti[0]=ti[1]=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4810 Wed Aug 13 19:52:28 2014
new/usr/src/lib/openssl/libsunw_crypto/des/ofb_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/ofb_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

/* The input and output are loaded in multiples of 8 bits.
```



```
 * What this means is that if you hame numbits=12 and length=2
 * the first 12 bits will be retrieved from the first byte and half
 * the second.  The second 12 bits will come from the 3rd and half the 4th
 * byte.
 */
void DES_ofb_encrypt(const unsigned char *in, unsigned char *out, int numbits,
                     long length, DES_key_schedule *schedule,
                     DES_cblock *ivec)
        {
        register DES_LONG d0,d1,vv0,vv1,v0,v1,n=(numbits+7)/8;
        register DES_LONG mask0,mask1;
        register long l=length;
        register int num=numbits;
        DES_LONG ti[2];
        unsigned char *iv;

        if (num > 64) return;
        if (num > 32)
                {
                mask0=0xffffffffL;
                if (num >= 64)
                        mask1=mask0;
                else
                        mask1=(1L<<(num-32))-1;
                }
        else
                {
                if (num == 32)
                        mask0=0xffffffffL;
                else
                        mask0=(1L<<num)-1;
                mask1=0x00000000L;
                }

        iv = &(*ivec)[0];
        c2l(iv,v0);
        c2l(iv,v1);
        ti[0]=v0;
        ti[1]=v1;
        while (l-- > 0)
                {
                ti[0]=v0;
                ti[1]=v1;
                DES_encrypt1((DES_LONG *)ti,schedule,DES_ENCRYPT);
                vv0=ti[0];
                vv1=ti[1];
                c2ln(in,d0,d1,n);
                in+=n;
                d0=(d0^vv0)&mask0;
                d1=(d1^vv1)&mask1;
                l2cn(d0,d1,out,n);
                out+=n;

                if (num == 32)
                        { v0=v1; v1=vv0; }
                else if (num == 64)
                                { v0=vv0; v1=vv1; }
                else if (num > 32) /* && num != 64 */
                        {
                        v0=((v1>>(num-32))|(vv0<<(64-num)))&0xffffffffL;
                        v1=((vv0>>(num-32))|(vv1<<(64-num)))&0xffffffffL;
                        }
                else /* num < 32 */
                        {
                        v0=((v0>>num)|(v1<<(32-num)))&0xffffffffL;
                        v1=((v1>>num)|(vv0<<(32-num)))&0xffffffffL;
```

```
128                     }
129                 }
130         iv = &(*ivec)[0];
131         l2c(v0,iv);
132         l2c(v1,iv);
133         v0=v1=d0=d1=ti[0]=ti[1]=vv0=vv1=0;
134         }
135 #endif /* ! codereview */
```

```
**********************************************************
    4407 Wed Aug 13 19:52:29 2014
new/usr/src/lib/openssl/libsunw_crypto/des/pcbc_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/pcbc_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

void DES_pcbc_encrypt(const unsigned char *input, unsigned char *output,
```



```
                      long length, DES_key_schedule *schedule,
                      DES_cblock *ivec, int enc)
        {
        register DES_LONG sin0,sin1,xor0,xor1,tout0,tout1;
        DES_LONG tin[2];
        const unsigned char *in;
        unsigned char *out,*iv;

        in=input;
        out=output;
        iv = &(*ivec)[0];

        if (enc)
                {
                c2l(iv,xor0);
                c2l(iv,xor1);
                for (; length>0; length-=8)
                        {
                        if (length >= 8)
                                {
                                c2l(in,sin0);
                                c2l(in,sin1);
                                }
                        else
                                c2ln(in,sin0,sin1,length);
                        tin[0]=sin0^xor0;
                        tin[1]=sin1^xor1;
                        DES_encrypt1((DES_LONG *)tin,schedule,DES_ENCRYPT);
                        tout0=tin[0];
                        tout1=tin[1];
                        xor0=sin0^tout0;
                        xor1=sin1^tout1;
                        l2c(tout0,out);
                        l2c(tout1,out);
                        }
                }
        else
                {
                c2l(iv,xor0); c2l(iv,xor1);
                for (; length>0; length-=8)
                        {
                        c2l(in,sin0);
                        c2l(in,sin1);
                        tin[0]=sin0;
                        tin[1]=sin1;
                        DES_encrypt1((DES_LONG *)tin,schedule,DES_DECRYPT);
                        tout0=tin[0]^xor0;
                        tout1=tin[1]^xor1;
                        if (length >= 8)
                                {
                                l2c(tout0,out);
                                l2c(tout1,out);
                                }
                        else
                                l2cn(tout0,tout1,out,length);
                        xor0=tout0^sin0;
                        xor1=tout1^sin1;
                        }
                }
        tin[0]=tin[1]=0;
        sin0=sin1=xor0=xor1=tout0=tout1=0;
        }
#endif /* ! codereview */
```

```
**********************************************************
    5108 Wed Aug 13 19:52:29 2014
new/usr/src/lib/openssl/libsunw_crypto/des/qud_cksm.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/qud_cksm.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* From "Message Authentication"  R.R. Jueneman, S.M. Matyas, C.H. Meyer
 * IEEE Communications Magazine Sept 1985 Vol. 23 No. 9 p 29-40
 * This module in only based on the code in this paper and is
```



```
 * almost definitely not the same as the MIT implementation.
 */
#include "des_locl.h"

/* bug fix for dos - 7/6/91 - Larry hughes@logos.ucs.indiana.edu */
#define Q_B0(a) (((DES_LONG)(a)))
#define Q_B1(a) (((DES_LONG)(a))<<8)
#define Q_B2(a) (((DES_LONG)(a))<<16)
#define Q_B3(a) (((DES_LONG)(a))<<24)

/* used to scramble things a bit */
/* Got the value MIT uses via brute force :-) 2/10/90 eay */
#define NOISE   ((DES_LONG)83653421L)

DES_LONG DES_quad_cksum(const unsigned char *input, DES_cblock output[],
             long length, int out_count, DES_cblock *seed)
        {
        DES_LONG z0,z1,t0,t1;
        int i;
        long l;
        const unsigned char *cp;
#ifdef _CRAY
        struct lp_st { int a:32; int b:32; } *lp;
#else
        DES_LONG *lp;
#endif

        if (out_count < 1) out_count=1;
#ifdef _CRAY
        lp = (struct lp_st *) &(output[0])[0];
#else
        lp = (DES_LONG *) &(output[0])[0];
#endif

        z0=Q_B0((*seed)[0])|Q_B1((*seed)[1])|Q_B2((*seed)[2])|Q_B3((*seed)[3]);
        z1=Q_B0((*seed)[4])|Q_B1((*seed)[5])|Q_B2((*seed)[6])|Q_B3((*seed)[7]);

        for (i=0; ((i<4)&&(i<out_count)); i++)
                {
                cp=input;
                l=length;
                while (l > 0)
                        {
                        if (l > 1)
                                {
                                t0= (DES_LONG)(*(cp++));
                                t0|=(DES_LONG)Q_B1(*(cp++));
                                l--;
                                }
                        else
                                t0= (DES_LONG)(*(cp++));
                        l--;
                        /* add */
                        t0+=z0;
                        t0&=0xffffffffL;
                        t1=z1;
                        /* square, well sort of square */
                        z0=((((t0*t0)&0xffffffffL)+((t1*t1)&0xffffffffL))
                                &0xffffffffL)%0x7fffffffL;
                        z1=((t0*((t1+NOISE)&0xffffffffL))&0xffffffffL)%0x7ffffff
                        }
                if (lp != NULL)
                        {
                        /* The MIT library assumes that the checksum is
                         * composed of 2*out_count 32 bit ints */
#ifdef _CRAY
```

```
128                         (*lp).a = z0;
129                         (*lp).b = z1;
130                         lp++;
131 #else
132                         *lp++ = z0;
133                         *lp++ = z1;
134 #endif
135                         }
136                 }
137         return(z0);
138         }
139 #endif /* ! codereview */
```

```
**********************************************************
    2917 Wed Aug 13 19:52:29 2014
new/usr/src/lib/openssl/libsunw_crypto/des/rand_key.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/rand_key.c */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/des.h>
#include <openssl/rand.h>

int DES_random_key(DES_cblock *ret)
        {
        do
```



```
                {
                if (RAND_bytes((unsigned char *)ret, sizeof(DES_cblock)) != 1)
                        return (0);
                } while (DES_is_weak_key(ret));
        DES_set_odd_parity(ret);
        return (1);
        }
#endif /* ! codereview */
```

```
**********************************************************
    6531 Wed Aug 13 19:52:29 2014
new/usr/src/lib/openssl/libsunw_crypto/des/read2pwd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/read2pwd.c */
/* ====================================================================
 * Copyright (c) 2001-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
```



```
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <string.h>
#include <openssl/des.h>
#include <openssl/ui.h>
#include <openssl/crypto.h>

int DES_read_password(DES_cblock *key, const char *prompt, int verify)
        {
        int ok;
        char buf[BUFSIZ],buff[BUFSIZ];

        if ((ok=UI_UTIL_read_pw(buf,buff,BUFSIZ,prompt,verify)) == 0)
                DES_string_to_key(buf,key);
        OPENSSL_cleanse(buf,BUFSIZ);
        OPENSSL_cleanse(buff,BUFSIZ);
        return(ok);
        }
```

```
129 int DES_read_2passwords(DES_cblock *key1, DES_cblock *key2, const char *prompt,
130             int verify)
131         {
132         int ok;
133         char buf[BUFSIZ],buff[BUFSIZ];
135         if ((ok=UI_UTIL_read_pw(buf,buff,BUFSIZ,prompt,verify)) == 0)
136                 DES_string_to_2keys(buf,key1,key2);
137         OPENSSL_cleanse(buf,BUFSIZ);
138         OPENSSL_cleanse(buff,BUFSIZ);
139         return(ok);
140         }
141 #endif /* ! codereview */
```

```
**********************************************************
    4208 Wed Aug 13 19:52:29 2014
new/usr/src/lib/openssl/libsunw_crypto/des/rpc_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/rpc_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "rpc_des.h"
#include "des_locl.h"
#include "des_ver.h"
```



```
int _des_crypt(char *buf,int len,struct desparams *desp);
int _des_crypt(char *buf, int len, struct desparams *desp)
        {
        DES_key_schedule ks;
        int enc;

        DES_set_key_unchecked(&desp->des_key,&ks);
        enc=(desp->des_dir == ENCRYPT)?DES_ENCRYPT:DES_DECRYPT;

        if (desp->des_mode == CBC)
                DES_ecb_encrypt((const_DES_cblock *)desp->UDES.UDES_buf,
                                (DES_cblock *)desp->UDES.UDES_buf,&ks,
                                enc);
        else
                {
                DES_ncbc_encrypt(desp->UDES.UDES_buf,desp->UDES.UDES_buf,
                                len,&ks,&desp->des_ivec,enc);
#ifdef undef
                /* len will always be %8 if called from common_crypt
                 * in secure_rpc.
                 * Libdes's cbc encrypt does not copy back the iv,
                 * so we have to do it here. */
                /* It does now :-) eay 20/09/95 */

                a=(char *)&(desp->UDES.UDES_buf[len-8]);
                b=(char *)&(desp->des_ivec[0]);

                *(a++)= *(b++); *(a++)= *(b++);
                *(a++)= *(b++); *(a++)= *(b++);
                *(a++)= *(b++); *(a++)= *(b++);
                *(a++)= *(b++); *(a++)= *(b++);
#endif
                }
        return(1);
        }
#endif /* ! codereview */
```

```
**********************************************************
   16381 Wed Aug 13 19:52:29 2014
new/usr/src/lib/openssl/libsunw_crypto/des/set_key.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/set_key.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* set_key.c v 1.4 eay 24/9/91
 * 1.4 Speed up by 400% :-)
 * 1.3 added register declarations.
```



```
 * 1.2 unrolled make_key_sched a bit more
 * 1.1 added norm_expand_bits
 * 1.0 First working version
 */
#include <openssl/crypto.h>
#include "des_locl.h"

OPENSSL_IMPLEMENT_GLOBAL(int,DES_check_key,0)   /* defaults to false */

static const unsigned char odd_parity[256]={
  1,  1,  2,  2,  4,  4,  7,  7,  8,  8, 11, 11, 13, 13, 14, 14,
 16, 16, 19, 19, 21, 21, 22, 22, 25, 25, 26, 26, 28, 28, 31, 31,
 32, 32, 35, 35, 37, 37, 38, 38, 41, 41, 42, 42, 44, 44, 47, 47,
 49, 49, 50, 50, 52, 52, 55, 55, 56, 56, 59, 59, 61, 61, 62, 62,
 64, 64, 67, 67, 69, 69, 70, 70, 73, 73, 74, 74, 76, 76, 79, 79,
 81, 81, 82, 82, 84, 84, 87, 87, 88, 88, 91, 91, 93, 93, 94, 94,
 97, 97, 98, 98,100,100,103,103,104,104,107,107,109,109,110,110,
112,112,115,115,117,117,118,118,121,121,122,122,124,124,127,127,
128,128,131,131,133,133,134,134,137,137,138,138,140,140,143,143,
145,145,146,146,148,148,151,151,152,152,155,155,157,157,158,158,
161,161,162,162,164,164,167,167,168,168,171,171,173,173,174,174,
176,176,179,179,181,181,182,182,185,185,186,186,188,188,191,191,
193,193,194,194,196,196,199,199,200,200,203,203,205,205,206,206,
208,208,211,211,213,213,214,214,217,217,218,218,220,220,223,223,
224,224,227,227,229,229,230,230,233,233,234,234,236,236,239,239,
241,241,242,242,244,244,247,247,248,248,251,251,253,253,254,254};

void DES_set_odd_parity(DES_cblock *key)
        {
        unsigned int i;

        for (i=0; i<DES_KEY_SZ; i++)
                (*key)[i]=odd_parity[(*key)[i]];
        }

int DES_check_key_parity(const_DES_cblock *key)
        {
        unsigned int i;

        for (i=0; i<DES_KEY_SZ; i++)
                {
                if ((*key)[i] != odd_parity[(*key)[i]])
                        return(0);
                }
        return(1);
        }

/* Weak and semi week keys as take from
 * %A D.W. Davies
 * %A W.L. Price
 * %T Security for Computer Networks
 * %I John Wiley & Sons
 * %D 1984
 * Many thanks to smb@ulysses.att.com (Steven Bellovin) for the reference
 * (and actual cblock values).
 */
#define NUM_WEAK_KEY    16
static const DES_cblock weak_keys[NUM_WEAK_KEY]={
        /* weak keys */
        {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01},
        {0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE,0xFE},
        {0x1F,0x1F,0x1F,0x1F,0x0E,0x0E,0x0E,0x0E},
        {0xE0,0xE0,0xE0,0xE0,0xF1,0xF1,0xF1,0xF1},
        /* semi-weak keys */
        {0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE},
        {0xFE,0x01,0xFE,0x01,0xFE,0x01,0xFE,0x01},
```

```
128         {0x1F,0xE0,0x1F,0xE0,0x0E,0xF1,0x0E,0xF1},
129         {0xE0,0x1F,0xE0,0x1F,0xF1,0x0E,0xF1,0x0E},
130         {0x01,0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1},
131         {0xE0,0x01,0xE0,0x01,0xF1,0x01,0xF1,0x01},
132         {0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E,0xFE},
133         {0xFE,0x1F,0xFE,0x1F,0xFE,0x0E,0xFE,0x0E},
134         {0x01,0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E},
135         {0x1F,0x01,0x1F,0x01,0x0E,0x01,0x0E,0x01},
136         {0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1,0xFE},
137         {0xFE,0xE0,0xFE,0xE0,0xFE,0xF1,0xFE,0xF1}};

139 int DES_is_weak_key(const_DES_cblock *key)
140         {
141         int i;

143         for (i=0; i<NUM_WEAK_KEY; i++)
144                 /* Added == 0 to comparison, I obviously don't run
145                  * this section very often :-(, thanks to
146                  * engineering@MorningStar.Com for the fix
147                  * eay 93/06/29
148                  * Another problem, I was comparing only the first 4
149                  * bytes, 97/03/18 */
150                 if (memcmp(weak_keys[i],key,sizeof(DES_cblock)) == 0) return(1);
151         return(0);
152         }

154 /* NOW DEFINED IN des_local.h
155  * See ecb_encrypt.c for a pseudo description of these macros.
156  * #define PERM_OP(a,b,t,n,m) ((t)=((((a)>>(n))^(b))&(m)),\
157  *      (b)^=(t),\
158  *      (a)=((a)^((t)<<(n))))
159  */

161 #define HPERM_OP(a,t,n,m) ((t)=((((a)<<(16-(n)))^(a))&(m)),\
162         (a)=(a)^(t)^(t>>(16-(n))))

164 static const DES_LONG des_skb[8][64]={
165         {
166         /* for C bits (numbered as per FIPS 46) 1 2 3 4 5 6 */
167         0x00000000L,0x00000010L,0x20000000L,0x20000010L,
168         0x00010000L,0x00010010L,0x20010000L,0x20010010L,
169         0x00000800L,0x00000810L,0x20000800L,0x20000810L,
170         0x00010800L,0x00010810L,0x20010800L,0x20010810L,
171         0x00000020L,0x00000030L,0x20000020L,0x20000030L,
172         0x00010020L,0x00010030L,0x20010020L,0x20010030L,
173         0x00000820L,0x00000830L,0x20000820L,0x20000830L,
174         0x00010820L,0x00010830L,0x20010820L,0x20010830L,
175         0x00080000L,0x00080010L,0x20080000L,0x20080010L,
176         0x00090000L,0x00090010L,0x20090000L,0x20090010L,
177         0x00080800L,0x00080810L,0x20080800L,0x20080810L,
178         0x00090800L,0x00090810L,0x20090800L,0x20090810L,
179         0x00080020L,0x00080030L,0x20080020L,0x20080030L,
180         0x00090020L,0x00090030L,0x20090020L,0x20090030L,
181         0x00080820L,0x00080830L,0x20080820L,0x20080830L,
182         0x00090820L,0x00090830L,0x20090820L,0x20090830L,
183         },{
184         /* for C bits (numbered as per FIPS 46) 7 8 10 11 12 13 */
185         0x00000000L,0x02000000L,0x00002000L,0x02002000L,
186         0x00200000L,0x02200000L,0x00202000L,0x02202000L,
187         0x00000004L,0x02000004L,0x00002004L,0x02002004L,
188         0x00200004L,0x02200004L,0x00202004L,0x02202004L,
189         0x00000400L,0x02000400L,0x00002400L,0x02002400L,
190         0x00200400L,0x02200400L,0x00202400L,0x02202400L,
191         0x00000404L,0x02000404L,0x00002404L,0x02002404L,
192         0x00200404L,0x02200404L,0x00202404L,0x02202404L,
193         0x10000000L,0x12000000L,0x10002000L,0x12002000L,
```



```
194         0x10200000L,0x12200000L,0x10202000L,0x12202000L,
195         0x10000004L,0x12000004L,0x10002004L,0x12002004L,
196         0x10200004L,0x12200004L,0x10202004L,0x12202004L,
197         0x10000400L,0x12000400L,0x10002400L,0x12002400L,
198         0x10200400L,0x12200400L,0x10202400L,0x12202400L,
199         0x10000404L,0x12000404L,0x10002404L,0x12002404L,
200         0x10200404L,0x12200404L,0x10202404L,0x12202404L,
201         },{
202         /* for C bits (numbered as per FIPS 46) 14 15 16 17 19 20 */
203         0x00000000L,0x00000001L,0x00040000L,0x00040001L,
204         0x01000000L,0x01000001L,0x01040000L,0x01040001L,
205         0x00000002L,0x00000003L,0x00040002L,0x00040003L,
206         0x01000002L,0x01000003L,0x01040002L,0x01040003L,
207         0x00000200L,0x00000201L,0x00040200L,0x00040201L,
208         0x01000200L,0x01000201L,0x01040200L,0x01040201L,
209         0x00000202L,0x00000203L,0x00040202L,0x00040203L,
210         0x01000202L,0x01000203L,0x01040202L,0x01040203L,
211         0x08000000L,0x08000001L,0x08040000L,0x08040001L,
212         0x09000000L,0x09000001L,0x09040000L,0x09040001L,
213         0x08000002L,0x08000003L,0x08040002L,0x08040003L,
214         0x09000002L,0x09000003L,0x09040002L,0x09040003L,
215         0x08000200L,0x08000201L,0x08040200L,0x08040201L,
216         0x09000200L,0x09000201L,0x09040200L,0x09040201L,
217         0x08000202L,0x08000203L,0x08040202L,0x08040203L,
218         0x09000202L,0x09000203L,0x09040202L,0x09040203L,
219         },{
220         /* for C bits (numbered as per FIPS 46) 21 23 24 26 27 28 */
221         0x00000000L,0x00100000L,0x00000100L,0x00100100L,
222         0x00000008L,0x00100008L,0x00000108L,0x00100108L,
223         0x00001000L,0x00101000L,0x00001100L,0x00101100L,
224         0x00001008L,0x00101008L,0x00001108L,0x00101108L,
225         0x04000000L,0x04100000L,0x04000100L,0x04100100L,
226         0x04000008L,0x04100008L,0x04000108L,0x04100108L,
227         0x04001000L,0x04101000L,0x04001100L,0x04101100L,
228         0x04001008L,0x04101008L,0x04001108L,0x04101108L,
229         0x00020000L,0x00120000L,0x00020100L,0x00120100L,
230         0x00020008L,0x00120008L,0x00020108L,0x00120108L,
231         0x00021000L,0x00121000L,0x00021100L,0x00121100L,
232         0x00021008L,0x00121008L,0x00021108L,0x00121108L,
233         0x04020000L,0x04120000L,0x04020100L,0x04120100L,
234         0x04020008L,0x04120008L,0x04020108L,0x04120108L,
235         0x04021000L,0x04121000L,0x04021100L,0x04121100L,
236         0x04021008L,0x04121008L,0x04021108L,0x04121108L,
237         },{
238         /* for D bits (numbered as per FIPS 46) 1 2 3 4 5 6 */
239         0x00000000L,0x10000000L,0x00010000L,0x10010000L,
240         0x00000004L,0x10000004L,0x00010004L,0x10010004L,
241         0x20000000L,0x30000000L,0x20010000L,0x30010000L,
242         0x20000004L,0x30000004L,0x20010004L,0x30010004L,
243         0x00100000L,0x10100000L,0x00110000L,0x10110000L,
244         0x00100004L,0x10100004L,0x00110004L,0x10110004L,
245         0x20100000L,0x30100000L,0x20110000L,0x30110000L,
246         0x20100004L,0x30100004L,0x20110004L,0x30110004L,
247         0x00001000L,0x10001000L,0x00011000L,0x10011000L,
248         0x00001004L,0x10001004L,0x00011004L,0x10011004L,
249         0x20001000L,0x30001000L,0x20011000L,0x30011000L,
250         0x20001004L,0x30001004L,0x20011004L,0x30011004L,
251         0x00101000L,0x10101000L,0x00111000L,0x10111000L,
252         0x00101004L,0x10101004L,0x00111004L,0x10111004L,
253         0x20101000L,0x30101000L,0x20111000L,0x30111000L,
254         0x20101004L,0x30101004L,0x20111004L,0x30111004L,
255         },{
256         /* for D bits (numbered as per FIPS 46) 8 9 11 12 13 14 */
257         0x00000000L,0x08000000L,0x00000008L,0x08000008L,
258         0x00000400L,0x08000400L,0x00000408L,0x08000408L,
259         0x00020000L,0x08020000L,0x00020008L,0x08020008L,
```

```
260         0x00020400L,0x08020400L,0x00020408L,0x08020408L,
261         0x00000001L,0x08000001L,0x00000009L,0x08000009L,
262         0x00000401L,0x08000401L,0x00000409L,0x08000409L,
263         0x00020001L,0x08020001L,0x00020009L,0x08020009L,
264         0x00020401L,0x08020401L,0x00020409L,0x08020409L,
265         0x02000000L,0x0A000000L,0x02000008L,0x0A000008L,
266         0x02000400L,0x0A000400L,0x02000408L,0x0A000408L,
267         0x02020000L,0x0A020000L,0x02020008L,0x0A020008L,
268         0x02020400L,0x0A020400L,0x02020408L,0x0A020408L,
269         0x02000001L,0x0A000001L,0x02000009L,0x0A000009L,
270         0x02000401L,0x0A000401L,0x02000409L,0x0A000409L,
271         0x02020001L,0x0A020001L,0x02020009L,0x0A020009L,
272         0x02020401L,0x0A020401L,0x02020409L,0x0A020409L,
273         },{
274         /* for D bits (numbered as per FIPS 46) 16 17 18 19 20 21 */
275         0x00000000L,0x00000100L,0x00080000L,0x00080100L,
276         0x01000000L,0x01000100L,0x01080000L,0x01080100L,
277         0x00000010L,0x00000110L,0x00080010L,0x00080110L,
278         0x01000010L,0x01000110L,0x01080010L,0x01080110L,
279         0x00200000L,0x00200100L,0x00280000L,0x00280100L,
280         0x01200000L,0x01200100L,0x01280000L,0x01280100L,
281         0x00200010L,0x00200110L,0x00280010L,0x00280110L,
282         0x01200010L,0x01200110L,0x01280010L,0x01280110L,
283         0x00000200L,0x00000300L,0x00080200L,0x00080300L,
284         0x01000200L,0x01000300L,0x01080200L,0x01080300L,
285         0x00000210L,0x00000310L,0x00080210L,0x00080310L,
286         0x01000210L,0x01000310L,0x01080210L,0x01080310L,
287         0x00200200L,0x00200300L,0x00280200L,0x00280300L,
288         0x01200200L,0x01200300L,0x01280200L,0x01280300L,
289         0x00200210L,0x00200310L,0x00280210L,0x00280310L,
290         0x01200210L,0x01200310L,0x01280210L,0x01280310L,
291         },{
292         /* for D bits (numbered as per FIPS 46) 22 23 24 25 27 28 */
293         0x00000000L,0x04000000L,0x00040000L,0x04040000L,
294         0x00000002L,0x04000002L,0x00040002L,0x04040002L,
295         0x00002000L,0x04002000L,0x00042000L,0x04042000L,
296         0x00002002L,0x04002002L,0x00042002L,0x04042002L,
297         0x00000020L,0x04000020L,0x00040020L,0x04040020L,
298         0x00000022L,0x04000022L,0x00040022L,0x04040022L,
299         0x00002020L,0x04002020L,0x00042020L,0x04042020L,
300         0x00002022L,0x04002022L,0x00042022L,0x04042022L,
301         0x00000800L,0x04000800L,0x00040800L,0x04040800L,
302         0x00000802L,0x04000802L,0x00040802L,0x04040802L,
303         0x00002800L,0x04002800L,0x00042800L,0x04042800L,
304         0x00002802L,0x04002802L,0x00042802L,0x04042802L,
305         0x00000820L,0x04000820L,0x00040820L,0x04040820L,
306         0x00000822L,0x04000822L,0x00040822L,0x04040822L,
307         0x00002820L,0x04002820L,0x00042820L,0x04042820L,
308         0x00002822L,0x04002822L,0x00042822L,0x04042822L,
309         }};

311 int DES_set_key(const_DES_cblock *key, DES_key_schedule *schedule)
312         {
313         if (DES_check_key)
314                 {
315                 return DES_set_key_checked(key, schedule);
316                 }
317         else
318                 {
319                 DES_set_key_unchecked(key, schedule);
320                 return 0;
321                 }
322         }

324 /* return 0 if key parity is odd (correct),
325  * return -1 if key parity error,
```



```
326  * return -2 if illegal weak key.
327  */
328 int DES_set_key_checked(const_DES_cblock *key, DES_key_schedule *schedule)
329         {
330         if (!DES_check_key_parity(key))
331                 return(-1);
332         if (DES_is_weak_key(key))
333                 return(-2);
334         DES_set_key_unchecked(key, schedule);
335         return 0;
336         }

338 void DES_set_key_unchecked(const_DES_cblock *key, DES_key_schedule *schedule)
339 #ifdef OPENSSL_FIPS
340         {
341         fips_cipher_abort(DES);
342         private_DES_set_key_unchecked(key, schedule);
343         }
344 void private_DES_set_key_unchecked(const_DES_cblock *key, DES_key_schedule *sche
345 #endif
346         {
347         static const int shifts2[16]={0,0,1,1,1,1,1,1,0,1,1,1,1,1,1,0};
348         register DES_LONG c,d,t,s,t2;
349         register const unsigned char *in;
350         register DES_LONG *k;
351         register int i;

353 #ifdef OPENBSD_DEV_CRYPTO
354         memcpy(schedule->key,key,sizeof schedule->key);
355         schedule->session=NULL;
356 #endif
357         k = &schedule->ks->deslong[0];
358         in = &(*key)[0];

360         c2l(in,c);
361         c2l(in,d);

363         /* do PC1 in 47 simple operations :-)
364          * Thanks to John Fletcher (john_fletcher@lccmail.ocf.llnl.gov)
365          * for the inspiration. :-) */
366         PERM_OP (d,c,t,4,0x0f0f0f0fL);
367         HPERM_OP(c,t,-2,0xcccc0000L);
368         HPERM_OP(d,t,-2,0xcccc0000L);
369         PERM_OP (d,c,t,1,0x55555555L);
370         PERM_OP (c,d,t,8,0x00ff00ffL);
371         PERM_OP (d,c,t,1,0x55555555L);
372         d=      (((d&0x000000ffL)<<16L)| (d&0x0000ff00L)     |
373                  ((d&0x00ff0000L)>>16L)|((c&0xf0000000L)>>4L));
374         c&=0x0fffffffL;

376         for (i=0; i<ITERATIONS; i++)
377                 {
378                 if (shifts2[i])
379                         { c=((c>>2L)|(c<<26L)); d=((d>>2L)|(d<<26L)); }
380                 else
381                         { c=((c>>1L)|(c<<27L)); d=((d>>1L)|(d<<27L)); }
382                 c&=0x0fffffffL;
383                 d&=0x0fffffffL;
384                 /* could be a few less shifts but I am to lazy at this
385                  * point in time to investigate */
386                 s=      des_skb[0][ (c    )&0x3f                ]|
387                         des_skb[1][((c>> 6L)&0x03)|((c>> 7L)&0x3c)]|
388                         des_skb[2][((c>>13L)&0x0f)|((c>>14L)&0x30)]|
389                         des_skb[3][((c>>20L)&0x01)|((c>>21L)&0x06) |
390                                                    ((c>>22L)&0x38)];
391                 t=      des_skb[4][ (d    )&0x3f                ]|
```

```
392                         des_skb[5][((d>> 7L)&0x03)|((d>> 8L)&0x3c)]|
393                         des_skb[6][ (d>>15L)&0x3f                ]|
394                         des_skb[7][((d>>21L)&0x0f)|((d>>22L)&0x30)];

396                 /* table contained 0213 4657 */
397                 t2=((t<<16L)|(s&0x0000ffffL))&0xffffffffL;
398                 *(k++)=ROTATE(t2,30)&0xffffffffL;

400                 t2=((s>>16L)|(t&0xffff0000L));
401                 *(k++)=ROTATE(t2,26)&0xffffffffL;
402                 }
403         }

405 int DES_key_sched(const_DES_cblock *key, DES_key_schedule *schedule)
406         {
407         return(DES_set_key(key,schedule));
408         }
409 /*
410 #undef des_fixup_key_parity
411 void des_fixup_key_parity(des_cblock *key)
412         {
413         des_set_odd_parity(key);
414         }
415 */
416 #endif /* ! codereview */
```

```
**********************************************************
    5582 Wed Aug 13 19:52:30 2014
new/usr/src/lib/openssl/libsunw_crypto/des/str2key.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/str2key.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/crypto.h>
#include "des_locl.h"
```



```
void DES_string_to_key(const char *str, DES_cblock *key)
        {
        DES_key_schedule ks;
        int i,length;
        register unsigned char j;

        memset(key,0,8);
        length=strlen(str);
#ifdef OLD_STR_TO_KEY
        for (i=0; i<length; i++)
                (*key)[i%8]^=(str[i]<<1);
#else /* MIT COMPATIBLE */
        for (i=0; i<length; i++)
                {
                j=str[i];
                if ((i%16) < 8)
                        (*key)[i%8]^=(j<<1);
                else
                        {
                        /* Reverse the bit order 05/05/92 eay */
                        j=((j<<4)&0xf0)|((j>>4)&0x0f);
                        j=((j<<2)&0xcc)|((j>>2)&0x33);
                        j=((j<<1)&0xaa)|((j>>1)&0x55);
                        (*key)[7-(i%8)]^=j;
                        }
                }
#endif
        DES_set_odd_parity(key);
#ifdef EXPERIMENTAL_STR_TO_STRONG_KEY
        if(DES_is_weak_key(key))
            (*key)[7] ^= 0xF0;
        DES_set_key(key,&ks);
#else
        DES_set_key_unchecked(key,&ks);
#endif
        DES_cbc_cksum((const unsigned char*)str,key,length,&ks,key);
        OPENSSL_cleanse(&ks,sizeof(ks));
        DES_set_odd_parity(key);
        }

void DES_string_to_2keys(const char *str, DES_cblock *key1, DES_cblock *key2)
        {
        DES_key_schedule ks;
        int i,length;
        register unsigned char j;

        memset(key1,0,8);
        memset(key2,0,8);
        length=strlen(str);
#ifdef OLD_STR_TO_KEY
        if (length <= 8)
                {
                for (i=0; i<length; i++)
                        {
                        (*key2)[i]=(*key1)[i]=(str[i]<<1);
                        }
                }
        else
                {
                for (i=0; i<length; i++)
                        {
                        if ((i/8)&1)
                                (*key2)[i%8]^=(str[i]<<1);
                        else
                                (*key1)[i%8]^=(str[i]<<1);
                        }
```

```
128                 }
129 #else /* MIT COMPATIBLE */
130         for (i=0; i<length; i++)
131                 {
132                 j=str[i];
133                 if ((i%32) < 16)
134                         {
135                         if ((i%16) < 8)
136                                 (*key1)[i%8]^=(j<<1);
137                         else
138                                 (*key2)[i%8]^=(j<<1);
139                         }
140                 else
141                         {
142                         j=((j<<4)&0xf0)|((j>>4)&0x0f);
143                         j=((j<<2)&0xcc)|((j>>2)&0x33);
144                         j=((j<<1)&0xaa)|((j>>1)&0x55);
145                         if ((i%16) < 8)
146                                 (*key1)[7-(i%8)]^=j;
147                         else
148                                 (*key2)[7-(i%8)]^=j;
149                         }
150                 }
151         if (length <= 8) memcpy(key2,key1,8);
152 #endif
153         DES_set_odd_parity(key1);
154         DES_set_odd_parity(key2);
155 #ifdef EXPERIMENTAL_STR_TO_STRONG_KEY
156         if(DES_is_weak_key(key1))
157             (*key1)[7] ^= 0xF0;
158         DES_set_key(key1,&ks);
159 #else
160         DES_set_key_unchecked(key1,&ks);
161 #endif
162         DES_cbc_cksum((const unsigned char*)str,key1,length,&ks,key1);
163 #ifdef EXPERIMENTAL_STR_TO_STRONG_KEY
164         if(DES_is_weak_key(key2))
165             (*key2)[7] ^= 0xF0;
166         DES_set_key(key2,&ks);
167 #else
168         DES_set_key_unchecked(key2,&ks);
169 #endif
170         DES_cbc_cksum((const unsigned char*)str,key2,length,&ks,key2);
171         OPENSSL_cleanse(&ks,sizeof(ks));
172         DES_set_odd_parity(key1);
173         DES_set_odd_parity(key2);
174         }
175 #endif /* ! codereview */
```

```
**********************************************************
    7158 Wed Aug 13 19:52:30 2014
new/usr/src/lib/openssl/libsunw_crypto/des/xcbc_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/des/xcbc_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "des_locl.h"

/* RSA's DESX */
```



```
#if 0 /* broken code, preserved just in case anyone specifically looks for this
static const unsigned char desx_white_in2out[256]={
0xBD,0x56,0xEA,0xF2,0xA2,0xF1,0xAC,0x2A,0xB0,0x93,0xD1,0x9C,0x1B,0x33,0xFD,0xD0,
0x30,0x04,0xB6,0xDC,0x7D,0xDF,0x32,0x4B,0xF7,0xCB,0x45,0x9B,0x31,0xBB,0x21,0x5A,
0x41,0x9F,0xE1,0xD9,0x4A,0x4D,0x9E,0xDA,0xA0,0x68,0x2C,0xC3,0x27,0x5F,0x80,0x36,
0x3E,0xEE,0xFB,0x95,0x1A,0xFE,0xCE,0xA8,0x34,0xA9,0x13,0xF0,0xA6,0x3F,0xD8,0x0C,
0x78,0x24,0xAF,0x23,0x52,0xC1,0x67,0x17,0xF5,0x66,0x90,0xE7,0xE8,0x07,0xB8,0x60,
0x48,0xE6,0x1E,0x53,0xF3,0x92,0xA4,0x72,0x8C,0x08,0x15,0x6E,0x86,0x00,0x84,0xFA,
0xF4,0x7F,0x8A,0x42,0x19,0xF6,0xDB,0xCD,0x14,0x8D,0x50,0x12,0xBA,0x3C,0x06,0x4E,
0xEC,0xB3,0x35,0x11,0xA1,0x88,0x8E,0x2B,0x94,0x99,0xB7,0x71,0x74,0xD3,0xE4,0xBF,
0x3A,0xDE,0x96,0x0E,0xBC,0x0A,0xED,0x77,0xFC,0x37,0x6B,0x03,0x79,0x89,0x62,0xC6,
0xD7,0xC0,0xD2,0x7C,0x6A,0x8B,0x22,0xA3,0x5B,0x05,0x5D,0x02,0x75,0xD5,0x61,0xE3,
0x18,0x8F,0x55,0x51,0xAD,0x1F,0x0B,0x5E,0x85,0xE5,0xC2,0x57,0x63,0xCA,0x3D,0x6C,
0xB4,0xC5,0xCC,0x70,0xB2,0x91,0x59,0x0D,0x47,0x20,0xC8,0x4F,0x58,0xE0,0x01,0xE2,
0x16,0x38,0xC4,0x6F,0x3B,0x0F,0x65,0x46,0xBE,0x7E,0x2D,0x7B,0x82,0xF9,0x40,0xB5,
0x1D,0x73,0xF8,0xEB,0x26,0xC7,0x87,0x97,0x25,0x54,0xB1,0x28,0xAA,0x98,0x9D,0xA5,
0x64,0x6D,0x7A,0xD4,0x10,0x81,0x44,0xEF,0x49,0xD6,0xAE,0x2E,0xDD,0x76,0x5C,0x2F,
0xA7,0x1C,0xC9,0x09,0x69,0x9A,0x83,0xCF,0x29,0x39,0xB9,0xE9,0x4C,0xFF,0x43,0xAB,
        };

void DES_xwhite_in2out(const_DES_cblock *des_key, const_DES_cblock *in_white,
             DES_cblock *out_white)
        {
        int out0,out1;
        int i;
        const unsigned char *key = &(*des_key)[0];
        const unsigned char *in = &(*in_white)[0];
        unsigned char *out = &(*out_white)[0];

        out[0]=out[1]=out[2]=out[3]=out[4]=out[5]=out[6]=out[7]=0;
        out0=out1=0;
        for (i=0; i<8; i++)
                {
                out[i]=key[i]^desx_white_in2out[out0^out1];
                out0=out1;
                out1=(int)out[i&0x07];
                }

        out0=out[0];
        out1=out[i]; /* BUG: out-of-bounds read */
        for (i=0; i<8; i++)
                {
                out[i]=in[i]^desx_white_in2out[out0^out1];
                out0=out1;
                out1=(int)out[i&0x07];
                }
        }
#endif

void DES_xcbc_encrypt(const unsigned char *in, unsigned char *out,
                      long length, DES_key_schedule *schedule,
                      DES_cblock *ivec, const_DES_cblock *inw,
                      const_DES_cblock *outw, int enc)
        {
        register DES_LONG tin0,tin1;
        register DES_LONG tout0,tout1,xor0,xor1;
        register DES_LONG inW0,inW1,outW0,outW1;
        register const unsigned char *in2;
        register long l=length;
        DES_LONG tin[2];
        unsigned char *iv;

        in2 = &(*inw)[0];
        c2l(in2,inW0);
        c2l(in2,inW1);
```

```
128         in2 = &(*outw)[0];
129         c2l(in2,outW0);
130         c2l(in2,outW1);

132         iv = &(*ivec)[0];

134         if (enc)
135                 {
136                 c2l(iv,tout0);
137                 c2l(iv,tout1);
138                 for (l-=8; l>=0; l-=8)
139                         {
140                         c2l(in,tin0);
141                         c2l(in,tin1);
142                         tin0^=tout0^inW0; tin[0]=tin0;
143                         tin1^=tout1^inW1; tin[1]=tin1;
144                         DES_encrypt1(tin,schedule,DES_ENCRYPT);
145                         tout0=tin[0]^outW0; l2c(tout0,out);
146                         tout1=tin[1]^outW1; l2c(tout1,out);
147                         }
148                 if (l != -8)
149                         {
150                         c2ln(in,tin0,tin1,l+8);
151                         tin0^=tout0^inW0; tin[0]=tin0;
152                         tin1^=tout1^inW1; tin[1]=tin1;
153                         DES_encrypt1(tin,schedule,DES_ENCRYPT);
154                         tout0=tin[0]^outW0; l2c(tout0,out);
155                         tout1=tin[1]^outW1; l2c(tout1,out);
156                         }
157                 iv = &(*ivec)[0];
158                 l2c(tout0,iv);
159                 l2c(tout1,iv);
160                 }
161         else
162                 {
163                 c2l(iv,xor0);
164                 c2l(iv,xor1);
165                 for (l-=8; l>0; l-=8)
166                         {
167                         c2l(in,tin0); tin[0]=tin0^outW0;
168                         c2l(in,tin1); tin[1]=tin1^outW1;
169                         DES_encrypt1(tin,schedule,DES_DECRYPT);
170                         tout0=tin[0]^xor0^inW0;
171                         tout1=tin[1]^xor1^inW1;
172                         l2c(tout0,out);
173                         l2c(tout1,out);
174                         xor0=tin0;
175                         xor1=tin1;
176                         }
177                 if (l != -8)
178                         {
179                         c2l(in,tin0); tin[0]=tin0^outW0;
180                         c2l(in,tin1); tin[1]=tin1^outW1;
181                         DES_encrypt1(tin,schedule,DES_DECRYPT);
182                         tout0=tin[0]^xor0^inW0;
183                         tout1=tin[1]^xor1^inW1;
184                         l2cn(tout0,tout1,out,l+8);
185                         xor0=tin0;
186                         xor1=tin1;
187                         }

189                 iv = &(*ivec)[0];
190                 l2c(xor0,iv);
191                 l2c(xor1,iv);
192                 }
193         tin0=tin1=tout0=tout1=xor0=xor1=0;
```



```
194         inW0=inW1=outW0=outW1=0;
195         tin[0]=tin[1]=0;
196         }
197 #endif /* ! codereview */
```

```
**********************************************************
   11152 Wed Aug 13 19:52:30 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_ameth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/asn1.h>
```



```
#include <openssl/dh.h>
#include <openssl/bn.h>
#include "asn1_locl.h"

static void int_dh_free(EVP_PKEY *pkey)
        {
        DH_free(pkey->pkey.dh);
        }

static int dh_pub_decode(EVP_PKEY *pkey, X509_PUBKEY *pubkey)
        {
        const unsigned char *p, *pm;
        int pklen, pmlen;
        int ptype;
        void *pval;
        ASN1_STRING *pstr;
        X509_ALGOR *palg;
        ASN1_INTEGER *public_key = NULL;

        DH *dh = NULL;

        if (!X509_PUBKEY_get0_param(NULL, &p, &pklen, &palg, pubkey))
                return 0;
        X509_ALGOR_get0(NULL, &ptype, &pval, palg);

        if (ptype != V_ASN1_SEQUENCE)
                {
                DHerr(DH_F_DH_PUB_DECODE, DH_R_PARAMETER_ENCODING_ERROR);
                goto err;
                }

        pstr = pval;
        pm = pstr->data;
        pmlen = pstr->length;

        if (!(dh = d2i_DHparams(NULL, &pm, pmlen)))
                {
                DHerr(DH_F_DH_PUB_DECODE, DH_R_DECODE_ERROR);
                goto err;
                }

        if (!(public_key=d2i_ASN1_INTEGER(NULL, &p, pklen)))
                {
                DHerr(DH_F_DH_PUB_DECODE, DH_R_DECODE_ERROR);
                goto err;
                }

        /* We have parameters now set public key */
        if (!(dh->pub_key = ASN1_INTEGER_to_BN(public_key, NULL)))
                {
                DHerr(DH_F_DH_PUB_DECODE, DH_R_BN_DECODE_ERROR);
                goto err;
                }

        ASN1_INTEGER_free(public_key);
        EVP_PKEY_assign_DH(pkey, dh);
        return 1;

        err:
        if (public_key)
                ASN1_INTEGER_free(public_key);
        if (dh)
                DH_free(dh);
        return 0;

        }
```

```
129 static int dh_pub_encode(X509_PUBKEY *pk, const EVP_PKEY *pkey)
130         {
131         DH *dh;
132         void *pval = NULL;
133         int ptype;
134         unsigned char *penc = NULL;
135         int penclen;
136         ASN1_STRING *str;
137         ASN1_INTEGER *pub_key = NULL;

139         dh=pkey->pkey.dh;

141         str = ASN1_STRING_new();
142         str->length = i2d_DHparams(dh, &str->data);
143         if (str->length <= 0)
144                 {
145                 DHerr(DH_F_DH_PUB_ENCODE, ERR_R_MALLOC_FAILURE);
146                 goto err;
147                 }
148         pval = str;
149         ptype = V_ASN1_SEQUENCE;

151         pub_key = BN_to_ASN1_INTEGER(dh->pub_key, NULL);
152         if (!pub_key)
153                 goto err;

155         penclen = i2d_ASN1_INTEGER(pub_key, &penc);

157         ASN1_INTEGER_free(pub_key);

159         if (penclen <= 0)
160                 {
161                 DHerr(DH_F_DH_PUB_ENCODE, ERR_R_MALLOC_FAILURE);
162                 goto err;
163                 }

165         if (X509_PUBKEY_set0_param(pk, OBJ_nid2obj(EVP_PKEY_DH),
166                                    ptype, pval, penc, penclen))
167                 return 1;

169         err:
170         if (penc)
171                 OPENSSL_free(penc);
172         if (pval)
173                 ASN1_STRING_free(pval);

175         return 0;
176         }


179 /* PKCS#8 DH is defined in PKCS#11 of all places. It is similar to DH in
180  * that the AlgorithmIdentifier contains the paramaters, the private key
181  * is explcitly included and the pubkey must be recalculated.
182  */

184 static int dh_priv_decode(EVP_PKEY *pkey, PKCS8_PRIV_KEY_INFO *p8)
185         {
186         const unsigned char *p, *pm;
187         int pklen, pmlen;
188         int ptype;
189         void *pval;
190         ASN1_STRING *pstr;
191         X509_ALGOR *palg;
192         ASN1_INTEGER *privkey = NULL;
```



```
194         DH *dh = NULL;

196         if (!PKCS8_pkey_get0(NULL, &p, &pklen, &palg, p8))
197                 return 0;

199         X509_ALGOR_get0(NULL, &ptype, &pval, palg);

201         if (ptype != V_ASN1_SEQUENCE)
202                         goto decerr;

204         if (!(privkey=d2i_ASN1_INTEGER(NULL, &p, pklen)))
205                 goto decerr;


208         pstr = pval;
209         pm = pstr->data;
210         pmlen = pstr->length;
211         if (!(dh = d2i_DHparams(NULL, &pm, pmlen)))
212                 goto decerr;
213         /* We have parameters now set private key */
214         if (!(dh->priv_key = ASN1_INTEGER_to_BN(privkey, NULL)))
215                 {
216                 DHerr(DH_F_DH_PRIV_DECODE,DH_R_BN_ERROR);
217                 goto dherr;
218                 }
219         /* Calculate public key */
220         if (!DH_generate_key(dh))
221                 goto dherr;

223         EVP_PKEY_assign_DH(pkey, dh);

225         ASN1_INTEGER_free(privkey);

227         return 1;

229         decerr:
230         DHerr(DH_F_DH_PRIV_DECODE, EVP_R_DECODE_ERROR);
231         dherr:
232         DH_free(dh);
233         return 0;
234         }

236 static int dh_priv_encode(PKCS8_PRIV_KEY_INFO *p8, const EVP_PKEY *pkey)
237 {
238         ASN1_STRING *params = NULL;
239         ASN1_INTEGER *prkey = NULL;
240         unsigned char *dp = NULL;
241         int dplen;

243         params = ASN1_STRING_new();

245         if (!params)
246                 {
247                 DHerr(DH_F_DH_PRIV_ENCODE,ERR_R_MALLOC_FAILURE);
248                 goto err;
249                 }

251         params->length = i2d_DHparams(pkey->pkey.dh, &params->data);
252         if (params->length <= 0)
253                 {
254                 DHerr(DH_F_DH_PRIV_ENCODE,ERR_R_MALLOC_FAILURE);
255                 goto err;
256                 }
257         params->type = V_ASN1_SEQUENCE;

259         /* Get private key into integer */
```

```
260         prkey = BN_to_ASN1_INTEGER(pkey->pkey.dh->priv_key, NULL);

262         if (!prkey)
263                 {
264                 DHerr(DH_F_DH_PRIV_ENCODE,DH_R_BN_ERROR);
265                 goto err;
266                 }

268         dplen = i2d_ASN1_INTEGER(prkey, &dp);

270         ASN1_INTEGER_free(prkey);

272         if (!PKCS8_pkey_set0(p8, OBJ_nid2obj(NID_dhKeyAgreement), 0,
273                              V_ASN1_SEQUENCE, params, dp, dplen))
274                 goto err;

276         return 1;

278 err:
279         if (dp != NULL)
280                 OPENSSL_free(dp);
281         if (params != NULL)
282                 ASN1_STRING_free(params);
283         if (prkey != NULL)
284                 ASN1_INTEGER_free(prkey);
285         return 0;
286 }


289 static void update_buflen(const BIGNUM *b, size_t *pbuflen)
290         {
291         size_t i;
292         if (!b)
293                 return;
294         if (*pbuflen < (i = (size_t)BN_num_bytes(b)))
295                         *pbuflen = i;
296         }

298 static int dh_param_decode(EVP_PKEY *pkey,
299                                         const unsigned char **pder, int derlen)
300         {
301         DH *dh;
302         if (!(dh = d2i_DHparams(NULL, pder, derlen)))
303                 {
304                 DHerr(DH_F_DH_PARAM_DECODE, ERR_R_DH_LIB);
305                 return 0;
306                 }
307         EVP_PKEY_assign_DH(pkey, dh);
308         return 1;
309         }

311 static int dh_param_encode(const EVP_PKEY *pkey, unsigned char **pder)
312         {
313         return i2d_DHparams(pkey->pkey.dh, pder);
314         }

316 static int do_dh_print(BIO *bp, const DH *x, int indent,
317                                                 ASN1_PCTX *ctx, int ptype)
318         {
319         unsigned char *m=NULL;
320         int reason=ERR_R_BUF_LIB,ret=0;
321         size_t buf_len=0;

323         const char *ktype = NULL;

325         BIGNUM *priv_key, *pub_key;
```



```
327         if (ptype == 2)
328                 priv_key = x->priv_key;
329         else
330                 priv_key = NULL;

332         if (ptype > 0)
333                 pub_key = x->pub_key;
334         else
335                 pub_key = NULL;

337         update_buflen(x->p, &buf_len);

339         if (buf_len == 0)
340                 {
341                 reason = ERR_R_PASSED_NULL_PARAMETER;
342                 goto err;
343                 }

345         update_buflen(x->g, &buf_len);
346         update_buflen(pub_key, &buf_len);
347         update_buflen(priv_key, &buf_len);

349         if (ptype == 2)
350                 ktype = "PKCS#3 DH Private-Key";
351         else if (ptype == 1)
352                 ktype = "PKCS#3 DH Public-Key";
353         else
354                 ktype = "PKCS#3 DH Parameters";

356         m= OPENSSL_malloc(buf_len+10);
357         if (m == NULL)
358                 {
359                 reason=ERR_R_MALLOC_FAILURE;
360                 goto err;
361                 }

363         BIO_indent(bp, indent, 128);
364         if (BIO_printf(bp,"%s: (%d bit)\n", ktype, BN_num_bits(x->p)) <= 0)
365                 goto err;
366         indent += 4;

368         if (!ASN1_bn_print(bp,"private-key:",priv_key,m,indent)) goto err;
369         if (!ASN1_bn_print(bp,"public-key:",pub_key,m,indent)) goto err;

371         if (!ASN1_bn_print(bp,"prime:",x->p,m,indent)) goto err;
372         if (!ASN1_bn_print(bp,"generator:",x->g,m,indent)) goto err;
373         if (x->length != 0)
374                 {
375                 BIO_indent(bp, indent, 128);
376                 if (BIO_printf(bp,"recommended-private-length: %d bits\n",
377                         (int)x->length) <= 0) goto err;
378                 }


381         ret=1;
382         if (0)
383                 {
384 err:
385                 DHerr(DH_F_DO_DH_PRINT,reason);
386                 }
387         if (m != NULL) OPENSSL_free(m);
388         return(ret);
389         }

391 static int int_dh_size(const EVP_PKEY *pkey)
```

```
392         {
393         return(DH_size(pkey->pkey.dh));
394         }

396 static int dh_bits(const EVP_PKEY *pkey)
397         {
398         return BN_num_bits(pkey->pkey.dh->p);
399         }

401 static int dh_cmp_parameters(const EVP_PKEY *a, const EVP_PKEY *b)
402         {
403         if (    BN_cmp(a->pkey.dh->p,b->pkey.dh->p) ||
404                 BN_cmp(a->pkey.dh->g,b->pkey.dh->g))
405                 return 0;
406         else
407                 return 1;
408         }

410 static int dh_copy_parameters(EVP_PKEY *to, const EVP_PKEY *from)
411         {
412         BIGNUM *a;

414         if ((a=BN_dup(from->pkey.dh->p)) == NULL)
415                 return 0;
416         if (to->pkey.dh->p != NULL)
417                 BN_free(to->pkey.dh->p);
418         to->pkey.dh->p=a;

420         if ((a=BN_dup(from->pkey.dh->g)) == NULL)
421                 return 0;
422         if (to->pkey.dh->g != NULL)
423                 BN_free(to->pkey.dh->g);
424         to->pkey.dh->g=a;

426         return 1;
427         }

429 static int dh_missing_parameters(const EVP_PKEY *a)
430         {
431         if (!a->pkey.dh->p || !a->pkey.dh->g)
432                 return 1;
433         return 0;
434         }

436 static int dh_pub_cmp(const EVP_PKEY *a, const EVP_PKEY *b)
437         {
438         if (dh_cmp_parameters(a, b) == 0)
439                 return 0;
440         if (BN_cmp(b->pkey.dh->pub_key,a->pkey.dh->pub_key) != 0)
441                 return 0;
442         else
443                 return 1;
444         }

446 static int dh_param_print(BIO *bp, const EVP_PKEY *pkey, int indent,
447                                                 ASN1_PCTX *ctx)
448         {
449         return do_dh_print(bp, pkey->pkey.dh, indent, ctx, 0);
450         }

452 static int dh_public_print(BIO *bp, const EVP_PKEY *pkey, int indent,
453                                                 ASN1_PCTX *ctx)
454         {
455         return do_dh_print(bp, pkey->pkey.dh, indent, ctx, 1);
456         }
```



```
458 static int dh_private_print(BIO *bp, const EVP_PKEY *pkey, int indent,
459                                                 ASN1_PCTX *ctx)
460         {
461         return do_dh_print(bp, pkey->pkey.dh, indent, ctx, 2);
462         }

464 int DHparams_print(BIO *bp, const DH *x)
465         {
466         return do_dh_print(bp, x, 4, NULL, 0);
467         }

469 const EVP_PKEY_ASN1_METHOD dh_asn1_meth =
470         {
471         EVP_PKEY_DH,
472         EVP_PKEY_DH,
473         0,

475         "DH",
476         "OpenSSL PKCS#3 DH method",

478         dh_pub_decode,
479         dh_pub_encode,
480         dh_pub_cmp,
481         dh_public_print,

483         dh_priv_decode,
484         dh_priv_encode,
485         dh_private_print,

487         int_dh_size,
488         dh_bits,

490         dh_param_decode,
491         dh_param_encode,
492         dh_missing_parameters,
493         dh_copy_parameters,
494         dh_cmp_parameters,
495         dh_param_print,
496         0,

498         int_dh_free,
499         0
500         };
501 #endif /* ! codereview */
```

```
**********************************************************
    3559 Wed Aug 13 19:52:30 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dh_asn1.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/dh.h>
#include <openssl/objects.h>
#include <openssl/asn1t.h>

/* Override the default free and new methods */
static int dh_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                 void *exarg)
{
        if(operation == ASN1_OP_NEW_PRE) {
                *pval = (ASN1_VALUE *)DH_new();
                if(*pval) return 2;
                return 0;
        } else if(operation == ASN1_OP_FREE_PRE) {
                DH_free((DH *)*pval);
                *pval = NULL;
                return 2;
        }
        return 1;
}

ASN1_SEQUENCE_cb(DHparams, dh_cb) = {
        ASN1_SIMPLE(DH, p, BIGNUM),
        ASN1_SIMPLE(DH, g, BIGNUM),
        ASN1_OPT(DH, length, ZLONG),
} ASN1_SEQUENCE_END_cb(DH, DHparams)

IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(DH, DHparams, DHparams)

DH *DHparams_dup(DH *dh)
        {
        return ASN1_item_dup(ASN1_ITEM_rptr(DHparams), dh);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4835 Wed Aug 13 19:52:30 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_check.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dh/dh_check.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/dh.h>

/* Check that p is a safe prime and
 * if g is 2, 3 or 5, check that it is a suitable generator
 * where
 * for 2, p mod 24 == 11
 * for 3, p mod 12 == 5
 * for 5, p mod 10 == 3 or 7
 * should hold.
 */

int DH_check(const DH *dh, int *ret)
        {
        int ok=0;
        BN_CTX *ctx=NULL;
        BN_ULONG l;
        BIGNUM *q=NULL;

        *ret=0;
        ctx=BN_CTX_new();
        if (ctx == NULL) goto err;
        q=BN_new();
        if (q == NULL) goto err;

        if (BN_is_word(dh->g,DH_GENERATOR_2))
                {
                l=BN_mod_word(dh->p,24);
                if (l != 11) *ret|=DH_NOT_SUITABLE_GENERATOR;
                }
#if 0
        else if (BN_is_word(dh->g,DH_GENERATOR_3))
                {
                l=BN_mod_word(dh->p,12);
                if (l != 5) *ret|=DH_NOT_SUITABLE_GENERATOR;
                }
#endif
        else if (BN_is_word(dh->g,DH_GENERATOR_5))
                {
                l=BN_mod_word(dh->p,10);
                if ((l != 3) && (l != 7))
                        *ret|=DH_NOT_SUITABLE_GENERATOR;
                }
        else
                *ret|=DH_UNABLE_TO_CHECK_GENERATOR;

        if (!BN_is_prime_ex(dh->p,BN_prime_checks,ctx,NULL))
                *ret|=DH_CHECK_P_NOT_PRIME;
        else
                {
                if (!BN_rshift1(q,dh->p)) goto err;
                if (!BN_is_prime_ex(q,BN_prime_checks,ctx,NULL))
                        *ret|=DH_CHECK_P_NOT_SAFE_PRIME;
                }
        ok=1;
err:
        if (ctx != NULL) BN_CTX_free(ctx);
        if (q != NULL) BN_free(q);
        return(ok);
        }

int DH_check_pub_key(const DH *dh, const BIGNUM *pub_key, int *ret)
        {
        int ok=0;
        BIGNUM *q=NULL;

        *ret=0;
```

```
128         q=BN_new();
129         if (q == NULL) goto err;
130         BN_set_word(q,1);
131         if (BN_cmp(pub_key,q)<=0)
132                 *ret|=DH_CHECK_PUBKEY_TOO_SMALL;
133         BN_copy(q,dh->p);
134         BN_sub_word(q,1);
135         if (BN_cmp(pub_key,q)>=0)
136                 *ret|=DH_CHECK_PUBKEY_TOO_LARGE;

138         ok = 1;
139 err:
140         if (q != NULL) BN_free(q);
141         return(ok);
142         }
143 #endif /* ! codereview */
```

```
**********************************************************
    3231 Wed Aug 13 19:52:30 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_depr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dh/dh_depr.c */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


/* This file contains deprecated functions as wrappers to the new ones */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/dh.h>

static void *dummy=&dummy;

#ifndef OPENSSL_NO_DEPRECATED
DH *DH_generate_parameters(int prime_len, int generator,
             void (*callback)(int,int,void *), void *cb_arg)
        {
        BN_GENCB cb;
        DH *ret=NULL;

        if((ret=DH_new()) == NULL)
                return NULL;

        BN_GENCB_set_old(&cb, callback, cb_arg);

        if(DH_generate_parameters_ex(ret, prime_len, generator, &cb))
                return ret;
        DH_free(ret);
        return NULL;
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    5063 Wed Aug 13 19:52:31 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dh/dh_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/dh.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_DH,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_DH,0,reason)

static ERR_STRING_DATA DH_str_functs[]=
        {
{ERR_FUNC(DH_F_COMPUTE_KEY),    "COMPUTE_KEY"},
{ERR_FUNC(DH_F_DHPARAMS_PRINT_FP),      "DHparams_print_fp"},
{ERR_FUNC(DH_F_DH_BUILTIN_GENPARAMS),   "DH_BUILTIN_GENPARAMS"},
{ERR_FUNC(DH_F_DH_COMPUTE_KEY), "DH_compute_key"},
{ERR_FUNC(DH_F_DH_GENERATE_KEY),        "DH_generate_key"},
{ERR_FUNC(DH_F_DH_GENERATE_PARAMETERS_EX),      "DH_generate_parameters_ex"},
{ERR_FUNC(DH_F_DH_NEW_METHOD),  "DH_new_method"},
{ERR_FUNC(DH_F_DH_PARAM_DECODE),        "DH_PARAM_DECODE"},
{ERR_FUNC(DH_F_DH_PRIV_DECODE), "DH_PRIV_DECODE"},
{ERR_FUNC(DH_F_DH_PRIV_ENCODE), "DH_PRIV_ENCODE"},
{ERR_FUNC(DH_F_DH_PUB_DECODE),  "DH_PUB_DECODE"},
{ERR_FUNC(DH_F_DH_PUB_ENCODE),  "DH_PUB_ENCODE"},
{ERR_FUNC(DH_F_DO_DH_PRINT),    "DO_DH_PRINT"},
{ERR_FUNC(DH_F_GENERATE_KEY),   "GENERATE_KEY"},
{ERR_FUNC(DH_F_GENERATE_PARAMETERS),    "GENERATE_PARAMETERS"},
{ERR_FUNC(DH_F_PKEY_DH_DERIVE), "PKEY_DH_DERIVE"},
{ERR_FUNC(DH_F_PKEY_DH_KEYGEN), "PKEY_DH_KEYGEN"},
{0,NULL}
        };

static ERR_STRING_DATA DH_str_reasons[]=
        {
{ERR_REASON(DH_R_BAD_GENERATOR)          ,"bad generator"},
{ERR_REASON(DH_R_BN_DECODE_ERROR)        ,"bn decode error"},
{ERR_REASON(DH_R_BN_ERROR)               ,"bn error"},
{ERR_REASON(DH_R_DECODE_ERROR)           ,"decode error"},
{ERR_REASON(DH_R_INVALID_PUBKEY)         ,"invalid public key"},
{ERR_REASON(DH_R_KEYS_NOT_SET)           ,"keys not set"},
{ERR_REASON(DH_R_KEY_SIZE_TOO_SMALL)     ,"key size too small"},
{ERR_REASON(DH_R_MODULUS_TOO_LARGE)      ,"modulus too large"},
{ERR_REASON(DH_R_NON_FIPS_METHOD)        ,"non fips method"},
{ERR_REASON(DH_R_NO_PARAMETERS_SET)      ,"no parameters set"},
{ERR_REASON(DH_R_NO_PRIVATE_VALUE)       ,"no private value"},
{ERR_REASON(DH_R_PARAMETER_ENCODING_ERROR),"parameter encoding error"},
{0,NULL}
        };

#endif

void ERR_load_DH_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(DH_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,DH_str_functs);
                ERR_load_strings(0,DH_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    6858 Wed Aug 13 19:52:31 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_gen.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dh/dh_gen.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* NB: These functions have been upgraded - the previous prototypes are in
 * dh_depr.c as wrappers to these ones.
 *  - Geoff
```



```
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
#include <openssl/dh.h>

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

static int dh_builtin_genparams(DH *ret, int prime_len, int generator, BN_GENCB

int DH_generate_parameters_ex(DH *ret, int prime_len, int generator, BN_GENCB *c
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(ret->meth->flags & DH_FLAG_FIPS_METHOD)
                        && !(ret->flags & DH_FLAG_NON_FIPS_ALLOW))
                {
                DHerr(DH_F_DH_GENERATE_PARAMETERS_EX, DH_R_NON_FIPS_METHOD);
                return 0;
                }
#endif
        if(ret->meth->generate_params)
                return ret->meth->generate_params(ret, prime_len, generator, cb)
#ifdef OPENSSL_FIPS
        if (FIPS_mode())
                return FIPS_dh_generate_parameters_ex(ret, prime_len,
                                                      generator, cb);
#endif
        return dh_builtin_genparams(ret, prime_len, generator, cb);
        }

/* We generate DH parameters as follows
 * find a prime q which is prime_len/2 bits long.
 * p=(2*q)+1 or (p-1)/2 = q
 * For this case, g is a generator if
 * g^((p-1)/q) mod p != 1 for values of q which are the factors of p-1.
 * Since the factors of p-1 are q and 2, we just need to check
 * g^2 mod p != 1 and g^q mod p != 1.
 *
 * Having said all that,
 * there is another special case method for the generators 2, 3 and 5.
 * for 2, p mod 24 == 11
 * for 3, p mod 12 == 5  <<<<< does not work for safe primes.
 * for 5, p mod 10 == 3 or 7
 *
 * Thanks to Phil Karn <karn@qualcomm.com> for the pointers about the
 * special generators and for answering some of my questions.
 *
 * I've implemented the second simple method :-).
 * Since DH should be using a safe prime (both p and q are prime),
 * this generator function can take a very very long time to run.
 */
/* Actually there is no reason to insist that 'generator' be a generator.
 * It's just as OK (and in some sense better) to use a generator of the
 * order-q subgroup.
 */
static int dh_builtin_genparams(DH *ret, int prime_len, int generator, BN_GENCB
        {
        BIGNUM *t1,*t2;
        int g,ok= -1;
        BN_CTX *ctx=NULL;

        ctx=BN_CTX_new();
        if (ctx == NULL) goto err;
```

```
128         BN_CTX_start(ctx);
129         t1 = BN_CTX_get(ctx);
130         t2 = BN_CTX_get(ctx);
131         if (t1 == NULL || t2 == NULL) goto err;

133         /* Make sure 'ret' has the necessary elements */
134         if(!ret->p && ((ret->p = BN_new()) == NULL)) goto err;
135         if(!ret->g && ((ret->g = BN_new()) == NULL)) goto err;

137         if (generator <= 1)
138                 {
139                 DHerr(DH_F_DH_BUILTIN_GENPARAMS, DH_R_BAD_GENERATOR);
140                 goto err;
141                 }
142         if (generator == DH_GENERATOR_2)
143                 {
144                 if (!BN_set_word(t1,24)) goto err;
145                 if (!BN_set_word(t2,11)) goto err;
146                 g=2;
147                 }
148 #if 0 /* does not work for safe primes */
149         else if (generator == DH_GENERATOR_3)
150                 {
151                 if (!BN_set_word(t1,12)) goto err;
152                 if (!BN_set_word(t2,5)) goto err;
153                 g=3;
154                 }
155 #endif
156         else if (generator == DH_GENERATOR_5)
157                 {
158                 if (!BN_set_word(t1,10)) goto err;
159                 if (!BN_set_word(t2,3)) goto err;
160                 /* BN_set_word(t3,7); just have to miss
161                  * out on these ones :-( */
162                 g=5;
163                 }
164         else
165                 {
166                 /* in the general case, don't worry if 'generator' is a
167                  * generator or not: since we are using safe primes,
168                  * it will generate either an order-q or an order-2q group,
169                  * which both is OK */
170                 if (!BN_set_word(t1,2)) goto err;
171                 if (!BN_set_word(t2,1)) goto err;
172                 g=generator;
173                 }

175         if(!BN_generate_prime_ex(ret->p,prime_len,1,t1,t2,cb)) goto err;
176         if(!BN_GENCB_call(cb, 3, 0)) goto err;
177         if (!BN_set_word(ret->g,g)) goto err;
178         ok=1;
179 err:
180         if (ok == -1)
181                 {
182                 DHerr(DH_F_DH_BUILTIN_GENPARAMS,ERR_R_BN_LIB);
183                 ok=0;
184                 }

186         if (ctx != NULL)
187                 {
188                 BN_CTX_end(ctx);
189                 BN_CTX_free(ctx);
190                 }
191         return ok;
192         }
193 #endif /* ! codereview */
```

```
**********************************************************
    7796 Wed Aug 13 19:52:31 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_key.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dh/dh_key.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rand.h>
#include <openssl/dh.h>

static int generate_key(DH *dh);
static int compute_key(unsigned char *key, const BIGNUM *pub_key, DH *dh);
static int dh_bn_mod_exp(const DH *dh, BIGNUM *r,
                         const BIGNUM *a, const BIGNUM *p,
                         const BIGNUM *m, BN_CTX *ctx,
                         BN_MONT_CTX *m_ctx);
static int dh_init(DH *dh);
static int dh_finish(DH *dh);

int DH_generate_key(DH *dh)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(dh->meth->flags & DH_FLAG_FIPS_METHOD)
                        && !(dh->flags & DH_FLAG_NON_FIPS_ALLOW))
                {
                DHerr(DH_F_DH_GENERATE_KEY, DH_R_NON_FIPS_METHOD);
                return 0;
                }
#endif
        return dh->meth->generate_key(dh);
        }

int DH_compute_key(unsigned char *key, const BIGNUM *pub_key, DH *dh)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(dh->meth->flags & DH_FLAG_FIPS_METHOD)
                        && !(dh->flags & DH_FLAG_NON_FIPS_ALLOW))
                {
                DHerr(DH_F_DH_COMPUTE_KEY, DH_R_NON_FIPS_METHOD);
                return 0;
                }
#endif
        return dh->meth->compute_key(key, pub_key, dh);
        }

static DH_METHOD dh_ossl = {
"OpenSSL DH Method",
generate_key,
compute_key,
dh_bn_mod_exp,
dh_init,
dh_finish,
0,
NULL,
NULL
};

const DH_METHOD *DH_OpenSSL(void)
{
        return &dh_ossl;
}

static int generate_key(DH *dh)
        {
        int ok=0;
        int generate_new_key=0;
        unsigned l;
        BN_CTX *ctx;
        BN_MONT_CTX *mont=NULL;
        BIGNUM *pub_key=NULL,*priv_key=NULL;

        ctx = BN_CTX_new();
        if (ctx == NULL) goto err;
```

```
129         if (dh->priv_key == NULL)
130                 {
131                 priv_key=BN_new();
132                 if (priv_key == NULL) goto err;
133                 generate_new_key=1;
134                 }
135         else
136                 priv_key=dh->priv_key;
137
138         if (dh->pub_key == NULL)
139                 {
140                 pub_key=BN_new();
141                 if (pub_key == NULL) goto err;
142                 }
143         else
144                 pub_key=dh->pub_key;
145
146
147         if (dh->flags & DH_FLAG_CACHE_MONT_P)
148                 {
149                 mont = BN_MONT_CTX_set_locked(&dh->method_mont_p,
150                                 CRYPTO_LOCK_DH, dh->p, ctx);
151                 if (!mont)
152                         goto err;
153                 }
154
155         if (generate_new_key)
156                 {
157                 if (dh->q)
158                         {
159                         do
160                                 {
161                                 if (!BN_rand_range(priv_key, dh->q))
162                                         goto err;
163                                 }
164                         while (BN_is_zero(priv_key) || BN_is_one(priv_key));
165                         }
166                 else
167                         {
168                         /* secret exponent length */
169                         l = dh->length ? dh->length : BN_num_bits(dh->p)-1;
170                         if (!BN_rand(priv_key, l, 0, 0)) goto err;
171                         }
172                 }
173
174         {
175                 BIGNUM local_prk;
176                 BIGNUM *prk;
177
178                 if ((dh->flags & DH_FLAG_NO_EXP_CONSTTIME) == 0)
179                         {
180                         BN_init(&local_prk);
181                         prk = &local_prk;
182                         BN_with_flags(prk, priv_key, BN_FLG_CONSTTIME);
183                         }
184                 else
185                         prk = priv_key;
186
187                 if (!dh->meth->bn_mod_exp(dh, pub_key, dh->g, prk, dh->p, ctx, m
188         }
189
190         dh->pub_key=pub_key;
191         dh->priv_key=priv_key;
192         ok=1;
193 err:
```



```
194         if (ok != 1)
195                 DHerr(DH_F_GENERATE_KEY,ERR_R_BN_LIB);
196
197         if ((pub_key != NULL)  && (dh->pub_key == NULL))  BN_free(pub_key);
198         if ((priv_key != NULL) && (dh->priv_key == NULL)) BN_free(priv_key);
199         BN_CTX_free(ctx);
200         return(ok);
201         }
202
203 static int compute_key(unsigned char *key, const BIGNUM *pub_key, DH *dh)
204         {
205         BN_CTX *ctx=NULL;
206         BN_MONT_CTX *mont=NULL;
207         BIGNUM *tmp;
208         int ret= -1;
209         int check_result;
210
211         if (BN_num_bits(dh->p) > OPENSSL_DH_MAX_MODULUS_BITS)
212                 {
213                 DHerr(DH_F_COMPUTE_KEY,DH_R_MODULUS_TOO_LARGE);
214                 goto err;
215                 }
216
217         ctx = BN_CTX_new();
218         if (ctx == NULL) goto err;
219         BN_CTX_start(ctx);
220         tmp = BN_CTX_get(ctx);
221
222         if (dh->priv_key == NULL)
223                 {
224                 DHerr(DH_F_COMPUTE_KEY,DH_R_NO_PRIVATE_VALUE);
225                 goto err;
226                 }
227
228         if (dh->flags & DH_FLAG_CACHE_MONT_P)
229                 {
230                 mont = BN_MONT_CTX_set_locked(&dh->method_mont_p,
231                                 CRYPTO_LOCK_DH, dh->p, ctx);
232                 if ((dh->flags & DH_FLAG_NO_EXP_CONSTTIME) == 0)
233                         {
234                         /* XXX */
235                         BN_set_flags(dh->priv_key, BN_FLG_CONSTTIME);
236                         }
237                 if (!mont)
238                         goto err;
239                 }
240
241         if (!DH_check_pub_key(dh, pub_key, &check_result) || check_result)
242                 {
243                 DHerr(DH_F_COMPUTE_KEY,DH_R_INVALID_PUBKEY);
244                 goto err;
245                 }
246
247         if (!dh->meth->bn_mod_exp(dh, tmp, pub_key, dh->priv_key,dh->p,ctx,mont)
248                 {
249                 DHerr(DH_F_COMPUTE_KEY,ERR_R_BN_LIB);
250                 goto err;
251                 }
252
253         ret=BN_bn2bin(tmp,key);
254 err:
255         if (ctx != NULL)
256                 {
257                 BN_CTX_end(ctx);
258                 BN_CTX_free(ctx);
259                 }
```

```
260         return(ret);
261         }

263 static int dh_bn_mod_exp(const DH *dh, BIGNUM *r,
264                         const BIGNUM *a, const BIGNUM *p,
265                         const BIGNUM *m, BN_CTX *ctx,
266                         BN_MONT_CTX *m_ctx)
267         {
268         /* If a is only one word long and constant time is false, use the faster
269          * exponenentiation function.
270          */
271         if (a->top == 1 && ((dh->flags & DH_FLAG_NO_EXP_CONSTTIME) != 0))
272                 {
273                 BN_ULONG A = a->d[0];
274                 return BN_mod_exp_mont_word(r,A,p,m,ctx,m_ctx);
275                 }
276         else
277                 return BN_mod_exp_mont(r,a,p,m,ctx,m_ctx);
278         }


281 static int dh_init(DH *dh)
282         {
283         dh->flags |= DH_FLAG_CACHE_MONT_P;
284         return(1);
285         }

287 static int dh_finish(DH *dh)
288         {
289         if(dh->method_mont_p)
290                 BN_MONT_CTX_free(dh->method_mont_p);
291         return(1);
292         }
293 #endif /* ! codereview */
```

```
**********************************************************
    7286 Wed Aug 13 19:52:31 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dh/dh_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/dh.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

const char DH_version[]="Diffie-Hellman" OPENSSL_VERSION_PTEXT;

static const DH_METHOD *default_DH_method = NULL;

void DH_set_default_method(const DH_METHOD *meth)
        {
        default_DH_method = meth;
        }

const DH_METHOD *DH_get_default_method(void)
        {
        if(!default_DH_method)
                {
#ifdef OPENSSL_FIPS
                if (FIPS_mode())
                        return FIPS_dh_openssl();
                else
                        return DH_OpenSSL();
#else
                default_DH_method = DH_OpenSSL();
#endif
                }
        return default_DH_method;
        }

int DH_set_method(DH *dh, const DH_METHOD *meth)
        {
        /* NB: The caller is specifically setting a method, so it's not up to us
         * to deal with which ENGINE it comes from. */
        const DH_METHOD *mtmp;
        mtmp = dh->meth;
        if (mtmp->finish) mtmp->finish(dh);
#ifndef OPENSSL_NO_ENGINE
        if (dh->engine)
                {
                ENGINE_finish(dh->engine);
                dh->engine = NULL;
                }
#endif
        dh->meth = meth;
        if (meth->init) meth->init(dh);
        return 1;
        }

DH *DH_new(void)
        {
        return DH_new_method(NULL);
        }

DH *DH_new_method(ENGINE *engine)
        {
        DH *ret;

        ret=(DH *)OPENSSL_malloc(sizeof(DH));
        if (ret == NULL)
                {
                DHerr(DH_F_DH_NEW_METHOD,ERR_R_MALLOC_FAILURE);
```

```
128                 return(NULL);
129                 }

131         ret->meth = DH_get_default_method();
132 #ifndef OPENSSL_NO_ENGINE
133         if (engine)
134                 {
135                 if (!ENGINE_init(engine))
136                         {
137                         DHerr(DH_F_DH_NEW_METHOD, ERR_R_ENGINE_LIB);
138                         OPENSSL_free(ret);
139                         return NULL;
140                         }
141                 ret->engine = engine;
142                 }
143         else
144                 ret->engine = ENGINE_get_default_DH();
145         if(ret->engine)
146                 {
147                 ret->meth = ENGINE_get_DH(ret->engine);
148                 if(!ret->meth)
149                         {
150                         DHerr(DH_F_DH_NEW_METHOD,ERR_R_ENGINE_LIB);
151                         ENGINE_finish(ret->engine);
152                         OPENSSL_free(ret);
153                         return NULL;
154                         }
155                 }
156 #endif

158         ret->pad=0;
159         ret->version=0;
160         ret->p=NULL;
161         ret->g=NULL;
162         ret->length=0;
163         ret->pub_key=NULL;
164         ret->priv_key=NULL;
165         ret->q=NULL;
166         ret->j=NULL;
167         ret->seed = NULL;
168         ret->seedlen = 0;
169         ret->counter = NULL;
170         ret->method_mont_p=NULL;
171         ret->references = 1;
172         ret->flags=ret->meth->flags & ~DH_FLAG_NON_FIPS_ALLOW;
173         CRYPTO_new_ex_data(CRYPTO_EX_INDEX_DH, ret, &ret->ex_data);
174         if ((ret->meth->init != NULL) && !ret->meth->init(ret))
175                 {
176 #ifndef OPENSSL_NO_ENGINE
177                 if (ret->engine)
178                         ENGINE_finish(ret->engine);
179 #endif
180                 CRYPTO_free_ex_data(CRYPTO_EX_INDEX_DH, ret, &ret->ex_data);
181                 OPENSSL_free(ret);
182                 ret=NULL;
183                 }
184         return(ret);
185         }

187 void DH_free(DH *r)
188         {
189         int i;
190         if(r == NULL) return;
191         i = CRYPTO_add(&r->references, -1, CRYPTO_LOCK_DH);
192 #ifdef REF_PRINT
193         REF_PRINT("DH",r);
```



```
194 #endif
195         if (i > 0) return;
196 #ifdef REF_CHECK
197         if (i < 0)
198                 {
199                 fprintf(stderr,"DH_free, bad reference count\n");
200                 abort();
201         }
202 #endif

204         if (r->meth->finish)
205                 r->meth->finish(r);
206 #ifndef OPENSSL_NO_ENGINE
207         if (r->engine)
208                 ENGINE_finish(r->engine);
209 #endif

211         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_DH, r, &r->ex_data);

213         if (r->p != NULL) BN_clear_free(r->p);
214         if (r->g != NULL) BN_clear_free(r->g);
215         if (r->q != NULL) BN_clear_free(r->q);
216         if (r->j != NULL) BN_clear_free(r->j);
217         if (r->seed) OPENSSL_free(r->seed);
218         if (r->counter != NULL) BN_clear_free(r->counter);
219         if (r->pub_key != NULL) BN_clear_free(r->pub_key);
220         if (r->priv_key != NULL) BN_clear_free(r->priv_key);
221         OPENSSL_free(r);
222         }

224 int DH_up_ref(DH *r)
225         {
226         int i = CRYPTO_add(&r->references, 1, CRYPTO_LOCK_DH);
227 #ifdef REF_PRINT
228         REF_PRINT("DH",r);
229 #endif
230 #ifdef REF_CHECK
231         if (i < 2)
232                 {
233                 fprintf(stderr, "DH_up, bad reference count\n");
234                 abort();
235                 }
236 #endif
237         return ((i > 1) ? 1 : 0);
238         }

240 int DH_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
241              CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
242         {
243         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_DH, argl, argp,
244                                        new_func, dup_func, free_func);
245         }

247 int DH_set_ex_data(DH *d, int idx, void *arg)
248         {
249         return(CRYPTO_set_ex_data(&d->ex_data,idx,arg));
250         }

252 void *DH_get_ex_data(DH *d, int idx)
253         {
254         return(CRYPTO_get_ex_data(&d->ex_data,idx));
255         }

257 int DH_size(const DH *dh)
258         {
259         return(BN_num_bytes(dh->p));
```

```
260         }
261 #endif /* ! codereview */
```

```
**********************************************************
    6135 Wed Aug 13 19:52:31 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_pmeth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/x509.h>
```



```
#include <openssl/evp.h>
#include <openssl/dh.h>
#include <openssl/bn.h>
#include "evp_locl.h"

/* DH pkey context structure */

typedef struct
        {
        /* Parameter gen parameters */
        int prime_len;
        int generator;
        int use_dsa;
        /* Keygen callback info */
        int gentmp[2];
        /* message digest */
        } DH_PKEY_CTX;

static int pkey_dh_init(EVP_PKEY_CTX *ctx)
        {
        DH_PKEY_CTX *dctx;
        dctx = OPENSSL_malloc(sizeof(DH_PKEY_CTX));
        if (!dctx)
                return 0;
        dctx->prime_len = 1024;
        dctx->generator = 2;
        dctx->use_dsa = 0;

        ctx->data = dctx;
        ctx->keygen_info = dctx->gentmp;
        ctx->keygen_info_count = 2;

        return 1;
        }

static int pkey_dh_copy(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src)
        {
        DH_PKEY_CTX *dctx, *sctx;
        if (!pkey_dh_init(dst))
                return 0;
        sctx = src->data;
        dctx = dst->data;
        dctx->prime_len = sctx->prime_len;
        dctx->generator = sctx->generator;
        dctx->use_dsa = sctx->use_dsa;
        return 1;
        }

static void pkey_dh_cleanup(EVP_PKEY_CTX *ctx)
        {
        DH_PKEY_CTX *dctx = ctx->data;
        if (dctx)
                OPENSSL_free(dctx);
        }

static int pkey_dh_ctrl(EVP_PKEY_CTX *ctx, int type, int p1, void *p2)
        {
        DH_PKEY_CTX *dctx = ctx->data;
        switch (type)
                {
                case EVP_PKEY_CTRL_DH_PARAMGEN_PRIME_LEN:
                if (p1 < 256)
                        return -2;
                dctx->prime_len = p1;
                return 1;
```

```
128                 case EVP_PKEY_CTRL_DH_PARAMGEN_GENERATOR:
129                 dctx->generator = p1;
130                 return 1;

132                 case EVP_PKEY_CTRL_PEER_KEY:
133                 /* Default behaviour is OK */
134                 return 1;

136                 default:
137                 return -2;

139                 }
140         }


143 static int pkey_dh_ctrl_str(EVP_PKEY_CTX *ctx,
144                         const char *type, const char *value)
145         {
146         if (!strcmp(type, "dh_paramgen_prime_len"))
147                 {
148                 int len;
149                 len = atoi(value);
150                 return EVP_PKEY_CTX_set_dh_paramgen_prime_len(ctx, len);
151                 }
152         if (!strcmp(type, "dh_paramgen_generator"))
153                 {
154                 int len;
155                 len = atoi(value);
156                 return EVP_PKEY_CTX_set_dh_paramgen_generator(ctx, len);
157                 }
158         return -2;
159         }

161 static int pkey_dh_paramgen(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey)
162         {
163         DH *dh = NULL;
164         DH_PKEY_CTX *dctx = ctx->data;
165         BN_GENCB *pcb, cb;
166         int ret;
167         if (ctx->pkey_gencb)
168                 {
169                 pcb = &cb;
170                 evp_pkey_set_cb_translate(pcb, ctx);
171                 }
172         else
173                 pcb = NULL;
174         dh = DH_new();
175         if (!dh)
176                 return 0;
177         ret = DH_generate_parameters_ex(dh,
178                                         dctx->prime_len, dctx->generator, pcb);
179         if (ret)
180                 EVP_PKEY_assign_DH(pkey, dh);
181         else
182                 DH_free(dh);
183         return ret;
184         }

186 static int pkey_dh_keygen(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey)
187         {
188         DH *dh = NULL;
189         if (ctx->pkey == NULL)
190                 {
191                 DHerr(DH_F_PKEY_DH_KEYGEN, DH_R_NO_PARAMETERS_SET);
192                 return 0;
193                 }
```



```
194         dh = DH_new();
195         if (!dh)
196                 return 0;
197         EVP_PKEY_assign_DH(pkey, dh);
198         /* Note: if error return, pkey is freed by parent routine */
199         if (!EVP_PKEY_copy_parameters(pkey, ctx->pkey))
200                 return 0;
201         return DH_generate_key(pkey->pkey.dh);
202         }

204 static int pkey_dh_derive(EVP_PKEY_CTX *ctx, unsigned char *key, size_t *keylen)
205         {
206         int ret;
207         if (!ctx->pkey || !ctx->peerkey)
208                 {
209                 DHerr(DH_F_PKEY_DH_DERIVE, DH_R_KEYS_NOT_SET);
210                 return 0;
211                 }
212         ret = DH_compute_key(key, ctx->peerkey->pkey.dh->pub_key,
213                                                         ctx->pkey->pkey.dh);
214         if (ret < 0)
215                 return ret;
216         *keylen = ret;
217         return 1;
218         }

220 const EVP_PKEY_METHOD dh_pkey_meth =
221         {
222         EVP_PKEY_DH,
223         EVP_PKEY_FLAG_AUTOARGLEN,
224         pkey_dh_init,
225         pkey_dh_copy,
226         pkey_dh_cleanup,

228         0,
229         pkey_dh_paramgen,

231         0,
232         pkey_dh_keygen,

234         0,
235         0,

237         0,
238         0,

240         0,0,

242         0,0,0,0,

244         0,0,

246         0,0,

248         0,
249         pkey_dh_derive,

251         pkey_dh_ctrl,
252         pkey_dh_ctrl_str

254         };
255 #endif /* ! codereview */
```

```
**********************************************************
    3580 Wed Aug 13 19:52:31 2014
new/usr/src/lib/openssl/libsunw_crypto/dh/dh_prn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/asn1/t_pkey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/dh.h>

#ifndef OPENSSL_NO_FP_API
int DHparams_print_fp(FILE *fp, const DH *x)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                DHerr(DH_F_DHPARAMS_PRINT_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=DHparams_print(b, x);
        BIO_free(b);
        return(ret);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
   15635 Wed Aug 13 19:52:32 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_ameth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/asn1.h>
```



```
#include <openssl/dsa.h>
#include <openssl/bn.h>
#ifndef OPENSSL_NO_CMS
#include <openssl/cms.h>
#endif
#include "asn1_locl.h"

static int dsa_pub_decode(EVP_PKEY *pkey, X509_PUBKEY *pubkey)
        {
        const unsigned char *p, *pm;
        int pklen, pmlen;
        int ptype;
        void *pval;
        ASN1_STRING *pstr;
        X509_ALGOR *palg;
        ASN1_INTEGER *public_key = NULL;

        DSA *dsa = NULL;

        if (!X509_PUBKEY_get0_param(NULL, &p, &pklen, &palg, pubkey))
                return 0;
        X509_ALGOR_get0(NULL, &ptype, &pval, palg);


        if (ptype == V_ASN1_SEQUENCE)
                {
                pstr = pval;
                pm = pstr->data;
                pmlen = pstr->length;

                if (!(dsa = d2i_DSAparams(NULL, &pm, pmlen)))
                        {
                        DSAerr(DSA_F_DSA_PUB_DECODE, DSA_R_DECODE_ERROR);
                        goto err;
                        }

                }
        else if ((ptype == V_ASN1_NULL) || (ptype == V_ASN1_UNDEF))
                {
                if (!(dsa = DSA_new()))
                        {
                        DSAerr(DSA_F_DSA_PUB_DECODE, ERR_R_MALLOC_FAILURE);
                        goto err;
                        }
                }
        else
                {
                DSAerr(DSA_F_DSA_PUB_DECODE, DSA_R_PARAMETER_ENCODING_ERROR);
                goto err;
                }

        if (!(public_key=d2i_ASN1_INTEGER(NULL, &p, pklen)))
                {
                DSAerr(DSA_F_DSA_PUB_DECODE, DSA_R_DECODE_ERROR);
                goto err;
                }

        if (!(dsa->pub_key = ASN1_INTEGER_to_BN(public_key, NULL)))
                {
                DSAerr(DSA_F_DSA_PUB_DECODE, DSA_R_BN_DECODE_ERROR);
                goto err;
                }

        ASN1_INTEGER_free(public_key);
        EVP_PKEY_assign_DSA(pkey, dsa);
        return 1;
```

```
129         err:
130         if (public_key)
131                 ASN1_INTEGER_free(public_key);
132         if (dsa)
133                 DSA_free(dsa);
134         return 0;

136         }

138 static int dsa_pub_encode(X509_PUBKEY *pk, const EVP_PKEY *pkey)
139         {
140         DSA *dsa;
141         void *pval = NULL;
142         int ptype;
143         unsigned char *penc = NULL;
144         int penclen;

146         dsa=pkey->pkey.dsa;
147         if (pkey->save_parameters && dsa->p && dsa->q && dsa->g)
148                 {
149                 ASN1_STRING *str;
150                 str = ASN1_STRING_new();
151                 str->length = i2d_DSAparams(dsa, &str->data);
152                 if (str->length <= 0)
153                         {
154                         DSAerr(DSA_F_DSA_PUB_ENCODE, ERR_R_MALLOC_FAILURE);
155                         goto err;
156                         }
157                 pval = str;
158                 ptype = V_ASN1_SEQUENCE;
159                 }
160         else
161                 ptype = V_ASN1_UNDEF;

163         dsa->write_params=0;

165         penclen = i2d_DSAPublicKey(dsa, &penc);

167         if (penclen <= 0)
168                 {
169                 DSAerr(DSA_F_DSA_PUB_ENCODE, ERR_R_MALLOC_FAILURE);
170                 goto err;
171                 }

173         if (X509_PUBKEY_set0_param(pk, OBJ_nid2obj(EVP_PKEY_DSA),
174                                    ptype, pval, penc, penclen))
175                 return 1;

177         err:
178         if (penc)
179                 OPENSSL_free(penc);
180         if (pval)
181                 ASN1_STRING_free(pval);

183         return 0;
184         }

186 /* In PKCS#8 DSA: you just get a private key integer and parameters in the
187  * AlgorithmIdentifier the pubkey must be recalculated.
188  */

190 static int dsa_priv_decode(EVP_PKEY *pkey, PKCS8_PRIV_KEY_INFO *p8)
191         {
192         const unsigned char *p, *pm;
193         int pklen, pmlen;
```



```
194         int ptype;
195         void *pval;
196         ASN1_STRING *pstr;
197         X509_ALGOR *palg;
198         ASN1_INTEGER *privkey = NULL;
199         BN_CTX *ctx = NULL;

201         STACK_OF(ASN1_TYPE) *ndsa = NULL;
202         DSA *dsa = NULL;

204         if (!PKCS8_pkey_get0(NULL, &p, &pklen, &palg, p8))
205                 return 0;
206         X509_ALGOR_get0(NULL, &ptype, &pval, palg);

208         /* Check for broken DSA PKCS#8, UGH! */
209         if (*p == (V_ASN1_SEQUENCE|V_ASN1_CONSTRUCTED))
210                 {
211                 ASN1_TYPE *t1, *t2;
212                 if(!(ndsa = d2i_ASN1_SEQUENCE_ANY(NULL, &p, pklen)))
213                         goto decerr;
214                 if (sk_ASN1_TYPE_num(ndsa) != 2)
215                         goto decerr;
216                 /* Handle Two broken types:
217                  * SEQUENCE {parameters, priv_key}
218                  * SEQUENCE {pub_key, priv_key}
219                  */

221                 t1 = sk_ASN1_TYPE_value(ndsa, 0);
222                 t2 = sk_ASN1_TYPE_value(ndsa, 1);
223                 if (t1->type == V_ASN1_SEQUENCE)
224                         {
225                         p8->broken = PKCS8_EMBEDDED_PARAM;
226                         pval = t1->value.ptr;
227                         }
228                 else if (ptype == V_ASN1_SEQUENCE)
229                         p8->broken = PKCS8_NS_DB;
230                 else
231                         goto decerr;

233                 if (t2->type != V_ASN1_INTEGER)
234                         goto decerr;

236                 privkey = t2->value.integer;
237                 }
238         else
239                 {
240                 const unsigned char *q = p;
241                 if (!(privkey=d2i_ASN1_INTEGER(NULL, &p, pklen)))
242                         goto decerr;
243                 if (privkey->type == V_ASN1_NEG_INTEGER)
244                         {
245                         p8->broken = PKCS8_NEG_PRIVKEY;
246                         ASN1_INTEGER_free(privkey);
247                         if (!(privkey=d2i_ASN1_UINTEGER(NULL, &q, pklen)))
248                                 goto decerr;
249                         }
250                 if (ptype != V_ASN1_SEQUENCE)
251                         goto decerr;
252                 }

254         pstr = pval;
255         pm = pstr->data;
256         pmlen = pstr->length;
257         if (!(dsa = d2i_DSAparams(NULL, &pm, pmlen)))
258                 goto decerr;
259         /* We have parameters now set private key */
```

```
260         if (!(dsa->priv_key = ASN1_INTEGER_to_BN(privkey, NULL)))
261                 {
262                 DSAerr(DSA_F_DSA_PRIV_DECODE,DSA_R_BN_ERROR);
263                 goto dsaerr;
264                 }
265         /* Calculate public key */
266         if (!(dsa->pub_key = BN_new()))
267                 {
268                 DSAerr(DSA_F_DSA_PRIV_DECODE, ERR_R_MALLOC_FAILURE);
269                 goto dsaerr;
270                 }
271         if (!(ctx = BN_CTX_new()))
272                 {
273                 DSAerr(DSA_F_DSA_PRIV_DECODE, ERR_R_MALLOC_FAILURE);
274                 goto dsaerr;
275                 }

277         if (!BN_mod_exp(dsa->pub_key, dsa->g, dsa->priv_key, dsa->p, ctx))
278                 {
279                 DSAerr(DSA_F_DSA_PRIV_DECODE,DSA_R_BN_ERROR);
280                 goto dsaerr;
281                 }

283         EVP_PKEY_assign_DSA(pkey, dsa);
284         BN_CTX_free (ctx);
285         if(ndsa)
286                 sk_ASN1_TYPE_pop_free(ndsa, ASN1_TYPE_free);
287         else
288                 ASN1_INTEGER_free(privkey);

290         return 1;

292         decerr:
293         DSAerr(DSA_F_DSA_PRIV_DECODE, EVP_R_DECODE_ERROR);
294         dsaerr:
295         BN_CTX_free (ctx);
296         if (privkey)
297                 ASN1_INTEGER_free(privkey);
298         sk_ASN1_TYPE_pop_free(ndsa, ASN1_TYPE_free);
299         DSA_free(dsa);
300         return 0;
301         }

303 static int dsa_priv_encode(PKCS8_PRIV_KEY_INFO *p8, const EVP_PKEY *pkey)
304 {
305         ASN1_STRING *params = NULL;
306         ASN1_INTEGER *prkey = NULL;
307         unsigned char *dp = NULL;
308         int dplen;

310         params = ASN1_STRING_new();

312         if (!params)
313                 {
314                 DSAerr(DSA_F_DSA_PRIV_ENCODE,ERR_R_MALLOC_FAILURE);
315                 goto err;
316                 }

318         params->length = i2d_DSAparams(pkey->pkey.dsa, &params->data);
319         if (params->length <= 0)
320                 {
321                 DSAerr(DSA_F_DSA_PRIV_ENCODE,ERR_R_MALLOC_FAILURE);
322                 goto err;
323                 }
324         params->type = V_ASN1_SEQUENCE;
```



```
326         /* Get private key into integer */
327         prkey = BN_to_ASN1_INTEGER(pkey->pkey.dsa->priv_key, NULL);

329         if (!prkey)
330                 {
331                 DSAerr(DSA_F_DSA_PRIV_ENCODE,DSA_R_BN_ERROR);
332                 goto err;
333                 }

335         dplen = i2d_ASN1_INTEGER(prkey, &dp);

337         ASN1_INTEGER_free(prkey);

339         if (!PKCS8_pkey_set0(p8, OBJ_nid2obj(NID_dsa), 0,
340                              V_ASN1_SEQUENCE, params, dp, dplen))
341                 goto err;

343         return 1;

345 err:
346         if (dp != NULL)
347                 OPENSSL_free(dp);
348         if (params != NULL)
349                 ASN1_STRING_free(params);
350         if (prkey != NULL)
351                 ASN1_INTEGER_free(prkey);
352         return 0;
353 }

355 static int int_dsa_size(const EVP_PKEY *pkey)
356         {
357         return(DSA_size(pkey->pkey.dsa));
358         }

360 static int dsa_bits(const EVP_PKEY *pkey)
361         {
362         return BN_num_bits(pkey->pkey.dsa->p);
363         }

365 static int dsa_missing_parameters(const EVP_PKEY *pkey)
366         {
367         DSA *dsa;
368         dsa=pkey->pkey.dsa;
369         if ((dsa->p == NULL) || (dsa->q == NULL) || (dsa->g == NULL))
370                         return 1;
371         return 0;
372         }

374 static int dsa_copy_parameters(EVP_PKEY *to, const EVP_PKEY *from)
375         {
376         BIGNUM *a;

378         if ((a=BN_dup(from->pkey.dsa->p)) == NULL)
379                 return 0;
380         if (to->pkey.dsa->p != NULL)
381                 BN_free(to->pkey.dsa->p);
382         to->pkey.dsa->p=a;

384         if ((a=BN_dup(from->pkey.dsa->q)) == NULL)
385                 return 0;
386         if (to->pkey.dsa->q != NULL)
387                 BN_free(to->pkey.dsa->q);
388         to->pkey.dsa->q=a;

390         if ((a=BN_dup(from->pkey.dsa->g)) == NULL)
391                 return 0;
```

```
392         if (to->pkey.dsa->g != NULL)
393                 BN_free(to->pkey.dsa->g);
394         to->pkey.dsa->g=a;
395         return 1;
396         }

398 static int dsa_cmp_parameters(const EVP_PKEY *a, const EVP_PKEY *b)
399         {
400         if (    BN_cmp(a->pkey.dsa->p,b->pkey.dsa->p) ||
401                 BN_cmp(a->pkey.dsa->q,b->pkey.dsa->q) ||
402                 BN_cmp(a->pkey.dsa->g,b->pkey.dsa->g))
403                 return 0;
404         else
405                 return 1;
406         }

408 static int dsa_pub_cmp(const EVP_PKEY *a, const EVP_PKEY *b)
409         {
410         if (BN_cmp(b->pkey.dsa->pub_key,a->pkey.dsa->pub_key) != 0)
411                 return 0;
412         else
413                 return 1;
414         }

416 static void int_dsa_free(EVP_PKEY *pkey)
417         {
418         DSA_free(pkey->pkey.dsa);
419         }

421 static void update_buflen(const BIGNUM *b, size_t *pbuflen)
422         {
423         size_t i;
424         if (!b)
425                 return;
426         if (*pbuflen < (i = (size_t)BN_num_bytes(b)))
427                         *pbuflen = i;
428         }

430 static int do_dsa_print(BIO *bp, const DSA *x, int off, int ptype)
431         {
432         unsigned char *m=NULL;
433         int ret=0;
434         size_t buf_len=0;
435         const char *ktype = NULL;

437         const BIGNUM *priv_key, *pub_key;

439         if (ptype == 2)
440                 priv_key = x->priv_key;
441         else
442                 priv_key = NULL;

444         if (ptype > 0)
445                 pub_key = x->pub_key;
446         else
447                 pub_key = NULL;

449         if (ptype == 2)
450                 ktype = "Private-Key";
451         else if (ptype == 1)
452                 ktype = "Public-Key";
453         else
454                 ktype = "DSA-Parameters";

456         update_buflen(x->p, &buf_len);
457         update_buflen(x->q, &buf_len);
```



```
458         update_buflen(x->g, &buf_len);
459         update_buflen(priv_key, &buf_len);
460         update_buflen(pub_key, &buf_len);

462         m=(unsigned char *)OPENSSL_malloc(buf_len+10);
463         if (m == NULL)
464                 {
465                 DSAerr(DSA_F_DO_DSA_PRINT,ERR_R_MALLOC_FAILURE);
466                 goto err;
467                 }

469         if (priv_key)
470                 {
471                 if(!BIO_indent(bp,off,128))
472                         goto err;
473                 if (BIO_printf(bp,"%s: (%d bit)\n",ktype, BN_num_bits(x->p))
474                         <= 0) goto err;
475                 }

477         if (!ASN1_bn_print(bp,"priv:",priv_key,m,off))
478                 goto err;
479         if (!ASN1_bn_print(bp,"pub: ",pub_key,m,off))
480                 goto err;
481         if (!ASN1_bn_print(bp,"P:   ",x->p,m,off)) goto err;
482         if (!ASN1_bn_print(bp,"Q:   ",x->q,m,off)) goto err;
483         if (!ASN1_bn_print(bp,"G:   ",x->g,m,off)) goto err;
484         ret=1;
485 err:
486         if (m != NULL) OPENSSL_free(m);
487         return(ret);
488         }

490 static int dsa_param_decode(EVP_PKEY *pkey,
491                                         const unsigned char **pder, int derlen)
492         {
493         DSA *dsa;
494         if (!(dsa = d2i_DSAparams(NULL, pder, derlen)))
495                 {
496                 DSAerr(DSA_F_DSA_PARAM_DECODE, ERR_R_DSA_LIB);
497                 return 0;
498                 }
499         EVP_PKEY_assign_DSA(pkey, dsa);
500         return 1;
501         }

503 static int dsa_param_encode(const EVP_PKEY *pkey, unsigned char **pder)
504         {
505         return i2d_DSAparams(pkey->pkey.dsa, pder);
506         }

508 static int dsa_param_print(BIO *bp, const EVP_PKEY *pkey, int indent,
509                                                         ASN1_PCTX *ctx)
510         {
511         return do_dsa_print(bp, pkey->pkey.dsa, indent, 0);
512         }

514 static int dsa_pub_print(BIO *bp, const EVP_PKEY *pkey, int indent,
515                                                         ASN1_PCTX *ctx)
516         {
517         return do_dsa_print(bp, pkey->pkey.dsa, indent, 1);
518         }


521 static int dsa_priv_print(BIO *bp, const EVP_PKEY *pkey, int indent,
522                                                         ASN1_PCTX *ctx)
523         {
```

```
524         return do_dsa_print(bp, pkey->pkey.dsa, indent, 2);
525         }

527 static int old_dsa_priv_decode(EVP_PKEY *pkey,
528                                 const unsigned char **pder, int derlen)
529         {
530         DSA *dsa;
531         if (!(dsa = d2i_DSAPrivateKey (NULL, pder, derlen)))
532                 {
533                 DSAerr(DSA_F_OLD_DSA_PRIV_DECODE, ERR_R_DSA_LIB);
534                 return 0;
535                 }
536         EVP_PKEY_assign_DSA(pkey, dsa);
537         return 1;
538         }

540 static int old_dsa_priv_encode(const EVP_PKEY *pkey, unsigned char **pder)
541         {
542         return i2d_DSAPrivateKey(pkey->pkey.dsa, pder);
543         }

545 static int dsa_sig_print(BIO *bp, const X509_ALGOR *sigalg,
546                                 const ASN1_STRING *sig,
547                                 int indent, ASN1_PCTX *pctx)
548         {
549         DSA_SIG *dsa_sig;
550         const unsigned char *p;
551         if (!sig)
552                 {
553                 if (BIO_puts(bp, "\n") <= 0)
554                         return 0;
555                 else
556                         return 1;
557                 }
558         p = sig->data;
559         dsa_sig = d2i_DSA_SIG(NULL, &p, sig->length);
560         if (dsa_sig)
561                 {
562                 int rv = 0;
563                 size_t buf_len = 0;
564                 unsigned char *m=NULL;
565                 update_buflen(dsa_sig->r, &buf_len);
566                 update_buflen(dsa_sig->s, &buf_len);
567                 m = OPENSSL_malloc(buf_len+10);
568                 if (m == NULL)
569                         {
570                         DSAerr(DSA_F_DSA_SIG_PRINT,ERR_R_MALLOC_FAILURE);
571                         goto err;
572                         }

574                 if (BIO_write(bp, "\n", 1) != 1)
575                         goto err;

577                 if (!ASN1_bn_print(bp,"r:   ",dsa_sig->r,m,indent))
578                         goto err;
579                 if (!ASN1_bn_print(bp,"s:   ",dsa_sig->s,m,indent))
580                         goto err;
581                 rv = 1;
582                 err:
583                 if (m)
584                         OPENSSL_free(m);
585                 DSA_SIG_free(dsa_sig);
586                 return rv;
587                 }
588         return X509_signature_dump(bp, sig, indent);
589         }
```



```
591 static int dsa_pkey_ctrl(EVP_PKEY *pkey, int op, long arg1, void *arg2)
592         {
593         switch (op)
594                 {
595                 case ASN1_PKEY_CTRL_PKCS7_SIGN:
596                 if (arg1 == 0)
597                         {
598                         int snid, hnid;
599                         X509_ALGOR *alg1, *alg2;
600                         PKCS7_SIGNER_INFO_get0_algs(arg2, NULL, &alg1, &alg2);
601                         if (alg1 == NULL || alg1->algorithm == NULL)
602                                 return -1;
603                         hnid = OBJ_obj2nid(alg1->algorithm);
604                         if (hnid == NID_undef)
605                                 return -1;
606                         if (!OBJ_find_sigid_by_algs(&snid, hnid, EVP_PKEY_id(pke
607                                 return -1;
608                         X509_ALGOR_set0(alg2, OBJ_nid2obj(snid), V_ASN1_UNDEF, 0
609                         }
610                 return 1;
611 #ifndef OPENSSL_NO_CMS
612                 case ASN1_PKEY_CTRL_CMS_SIGN:
613                 if (arg1 == 0)
614                         {
615                         int snid, hnid;
616                         X509_ALGOR *alg1, *alg2;
617                         CMS_SignerInfo_get0_algs(arg2, NULL, NULL, &alg1, &alg2)
618                         if (alg1 == NULL || alg1->algorithm == NULL)
619                                 return -1;
620                         hnid = OBJ_obj2nid(alg1->algorithm);
621                         if (hnid == NID_undef)
622                                 return -1;
623                         if (!OBJ_find_sigid_by_algs(&snid, hnid, EVP_PKEY_id(pke
624                                 return -1;
625                         X509_ALGOR_set0(alg2, OBJ_nid2obj(snid), V_ASN1_UNDEF, 0
626                         }
627                 return 1;
628 #endif

630                 case ASN1_PKEY_CTRL_DEFAULT_MD_NID:
631                 *(int *)arg2 = NID_sha1;
632                 return 2;

634                 default:
635                 return -2;

637                 }

639         }

641 /* NB these are sorted in pkey_id order, lowest first */

643 const EVP_PKEY_ASN1_METHOD dsa_asn1_meths[] =
644         {

646                 {
647                 EVP_PKEY_DSA2,
648                 EVP_PKEY_DSA,
649                 ASN1_PKEY_ALIAS
650                 },

652                 {
653                 EVP_PKEY_DSA1,
654                 EVP_PKEY_DSA,
655                 ASN1_PKEY_ALIAS
```

```
656                 },

658                 {
659                 EVP_PKEY_DSA4,
660                 EVP_PKEY_DSA,
661                 ASN1_PKEY_ALIAS
662                 },

664                 {
665                 EVP_PKEY_DSA3,
666                 EVP_PKEY_DSA,
667                 ASN1_PKEY_ALIAS
668                 },

670                 {
671                 EVP_PKEY_DSA,
672                 EVP_PKEY_DSA,
673                 0,

675                 "DSA",
676                 "OpenSSL DSA method",

678                 dsa_pub_decode,
679                 dsa_pub_encode,
680                 dsa_pub_cmp,
681                 dsa_pub_print,

683                 dsa_priv_decode,
684                 dsa_priv_encode,
685                 dsa_priv_print,

687                 int_dsa_size,
688                 dsa_bits,

690                 dsa_param_decode,
691                 dsa_param_encode,
692                 dsa_missing_parameters,
693                 dsa_copy_parameters,
694                 dsa_cmp_parameters,
695                 dsa_param_print,
696                 dsa_sig_print,

698                 int_dsa_free,
699                 dsa_pkey_ctrl,
700                 old_dsa_priv_decode,
701                 old_dsa_priv_encode
702                 }
703         };
704 #endif /* ! codereview */
```

```
**********************************************************
    6057 Wed Aug 13 19:52:32 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* dsa_asn1.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/dsa.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/rand.h>

/* Override the default new methods */
static int sig_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                                void *exarg)
{
        if(operation == ASN1_OP_NEW_PRE) {
                DSA_SIG *sig;
                sig = OPENSSL_malloc(sizeof(DSA_SIG));
                if (!sig)
                        {
                        DSAerr(DSA_F_SIG_CB, ERR_R_MALLOC_FAILURE);
                        return 0;
                        }
                sig->r = NULL;
                sig->s = NULL;
                *pval = (ASN1_VALUE *)sig;
                return 2;
        }
        return 1;
}

ASN1_SEQUENCE_cb(DSA_SIG, sig_cb) = {
        ASN1_SIMPLE(DSA_SIG, r, CBIGNUM),
        ASN1_SIMPLE(DSA_SIG, s, CBIGNUM)
} ASN1_SEQUENCE_END_cb(DSA_SIG, DSA_SIG)

IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(DSA_SIG, DSA_SIG, DSA_SIG)

/* Override the default free and new methods */
static int dsa_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                        void *exarg)
{
        if(operation == ASN1_OP_NEW_PRE) {
                *pval = (ASN1_VALUE *)DSA_new();
                if(*pval) return 2;
                return 0;
        } else if(operation == ASN1_OP_FREE_PRE) {
                DSA_free((DSA *)*pval);
                *pval = NULL;
                return 2;
        }
        return 1;
}

ASN1_SEQUENCE_cb(DSAPrivateKey, dsa_cb) = {
        ASN1_SIMPLE(DSA, version, LONG),
        ASN1_SIMPLE(DSA, p, BIGNUM),
        ASN1_SIMPLE(DSA, q, BIGNUM),
        ASN1_SIMPLE(DSA, g, BIGNUM),
        ASN1_SIMPLE(DSA, pub_key, BIGNUM),
        ASN1_SIMPLE(DSA, priv_key, BIGNUM)
} ASN1_SEQUENCE_END_cb(DSA, DSAPrivateKey)

IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(DSA, DSAPrivateKey, DSAPrivateKey)

ASN1_SEQUENCE_cb(DSAparams, dsa_cb) = {
        ASN1_SIMPLE(DSA, p, BIGNUM),
        ASN1_SIMPLE(DSA, q, BIGNUM),
        ASN1_SIMPLE(DSA, g, BIGNUM),
} ASN1_SEQUENCE_END_cb(DSA, DSAparams)

IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(DSA, DSAparams, DSAparams)
```

```
/* DSA public key is a bit trickier... its effectively a CHOICE type
 * decided by a field called write_params which can either write out
 * just the public key as an INTEGER or the parameters and public key
 * in a SEQUENCE
 */

ASN1_SEQUENCE(dsa_pub_internal) = {
        ASN1_SIMPLE(DSA, pub_key, BIGNUM),
        ASN1_SIMPLE(DSA, p, BIGNUM),
        ASN1_SIMPLE(DSA, q, BIGNUM),
        ASN1_SIMPLE(DSA, g, BIGNUM)
} ASN1_SEQUENCE_END_name(DSA, dsa_pub_internal)

ASN1_CHOICE_cb(DSAPublicKey, dsa_cb) = {
        ASN1_SIMPLE(DSA, pub_key, BIGNUM),
        ASN1_EX_COMBINE(0, 0, dsa_pub_internal)
} ASN1_CHOICE_END_cb(DSA, DSAPublicKey, write_params)

IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(DSA, DSAPublicKey, DSAPublicKey)

DSA *DSAparams_dup(DSA *dsa)
        {
        return ASN1_item_dup(ASN1_ITEM_rptr(DSAparams), dsa);
        }

int DSA_sign(int type, const unsigned char *dgst, int dlen, unsigned char *sig,
             unsigned int *siglen, DSA *dsa)
        {
        DSA_SIG *s;
        RAND_seed(dgst, dlen);
        s=DSA_do_sign(dgst,dlen,dsa);
        if (s == NULL)
                {
                *siglen=0;
                return(0);
                }
        *siglen=i2d_DSA_SIG(s,&sig);
        DSA_SIG_free(s);
        return(1);
        }

/* data has already been hashed (probably with SHA or SHA-1). */
/* returns
 *      1: correct signature
 *      0: incorrect signature
 *     -1: error
 */
int DSA_verify(int type, const unsigned char *dgst, int dgst_len,
               const unsigned char *sigbuf, int siglen, DSA *dsa)
        {
        DSA_SIG *s;
        int ret=-1;

        s = DSA_SIG_new();
        if (s == NULL) return(ret);
        if (d2i_DSA_SIG(&s,&sigbuf,siglen) == NULL) goto err;
        ret=DSA_do_verify(dgst,dgst_len,s,dsa);
err:
        DSA_SIG_free(s);
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    3870 Wed Aug 13 19:52:32 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_depr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_depr.c */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* This file contains deprecated function(s) that are now wrappers to the new
 * version(s). */

#undef GENUINE_DSA

#ifdef GENUINE_DSA
```



```
/* Parameter generation follows the original release of FIPS PUB 186,
 * Appendix 2.2 (i.e. use SHA as defined in FIPS PUB 180) */
#define HASH    EVP_sha()
#else
/* Parameter generation follows the updated Appendix 2.2 for FIPS PUB 186,
 * also Appendix 2.2 of FIPS PUB 186-1 (i.e. use SHA as defined in
 * FIPS PUB 180-1) */
#define HASH    EVP_sha1()
#endif

static void *dummy=&dummy;

#ifndef OPENSSL_NO_SHA

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
#include <openssl/evp.h>
#include <openssl/bn.h>
#include <openssl/dsa.h>
#include <openssl/rand.h>
#include <openssl/sha.h>

#ifndef OPENSSL_NO_DEPRECATED
DSA *DSA_generate_parameters(int bits,
                unsigned char *seed_in, int seed_len,
                int *counter_ret, unsigned long *h_ret,
                void (*callback)(int, int, void *),
                void *cb_arg)
        {
        BN_GENCB cb;
        DSA *ret;

        if ((ret=DSA_new()) == NULL) return NULL;

        BN_GENCB_set_old(&cb, callback, cb_arg);

        if(DSA_generate_parameters_ex(ret, bits, seed_in, seed_len,
                                      counter_ret, h_ret, &cb))
                return ret;
        DSA_free(ret);
        return NULL;
        }
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    5488 Wed Aug 13 19:52:32 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/dsa.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_DSA,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_DSA,0,reason)

static ERR_STRING_DATA DSA_str_functs[]=
        {
{ERR_FUNC(DSA_F_D2I_DSA_SIG),   "d2i_DSA_SIG"},
{ERR_FUNC(DSA_F_DO_DSA_PRINT),  "DO_DSA_PRINT"},
{ERR_FUNC(DSA_F_DSAPARAMS_PRINT),       "DSAparams_print"},
{ERR_FUNC(DSA_F_DSAPARAMS_PRINT_FP),    "DSAparams_print_fp"},
{ERR_FUNC(DSA_F_DSA_DO_SIGN),   "DSA_do_sign"},
{ERR_FUNC(DSA_F_DSA_DO_VERIFY), "DSA_do_verify"},
{ERR_FUNC(DSA_F_DSA_GENERATE_KEY),      "DSA_generate_key"},
{ERR_FUNC(DSA_F_DSA_GENERATE_PARAMETERS_EX),    "DSA_generate_parameters_ex"},
{ERR_FUNC(DSA_F_DSA_NEW_METHOD),        "DSA_new_method"},
{ERR_FUNC(DSA_F_DSA_PARAM_DECODE),      "DSA_PARAM_DECODE"},
{ERR_FUNC(DSA_F_DSA_PRINT_FP),  "DSA_print_fp"},
{ERR_FUNC(DSA_F_DSA_PRIV_DECODE),       "DSA_PRIV_DECODE"},
{ERR_FUNC(DSA_F_DSA_PRIV_ENCODE),       "DSA_PRIV_ENCODE"},
{ERR_FUNC(DSA_F_DSA_PUB_DECODE),        "DSA_PUB_DECODE"},
{ERR_FUNC(DSA_F_DSA_PUB_ENCODE),        "DSA_PUB_ENCODE"},
{ERR_FUNC(DSA_F_DSA_SIGN),      "DSA_sign"},
{ERR_FUNC(DSA_F_DSA_SIGN_SETUP),        "DSA_sign_setup"},
{ERR_FUNC(DSA_F_DSA_SIG_NEW),   "DSA_SIG_new"},
{ERR_FUNC(DSA_F_DSA_SIG_PRINT), "DSA_SIG_PRINT"},
{ERR_FUNC(DSA_F_DSA_VERIFY),    "DSA_verify"},
{ERR_FUNC(DSA_F_I2D_DSA_SIG),   "i2d_DSA_SIG"},
{ERR_FUNC(DSA_F_OLD_DSA_PRIV_DECODE),   "OLD_DSA_PRIV_DECODE"},
{ERR_FUNC(DSA_F_PKEY_DSA_CTRL), "PKEY_DSA_CTRL"},
{ERR_FUNC(DSA_F_PKEY_DSA_KEYGEN),       "PKEY_DSA_KEYGEN"},
{ERR_FUNC(DSA_F_SIG_CB),        "SIG_CB"},
{0,NULL}
        };

static ERR_STRING_DATA DSA_str_reasons[]=
        {
{ERR_REASON(DSA_R_BAD_Q_VALUE)          ,"bad q value"},
{ERR_REASON(DSA_R_BN_DECODE_ERROR)      ,"bn decode error"},
{ERR_REASON(DSA_R_BN_ERROR)             ,"bn error"},
{ERR_REASON(DSA_R_DATA_TOO_LARGE_FOR_KEY_SIZE),"data too large for key size"},
{ERR_REASON(DSA_R_DECODE_ERROR)         ,"decode error"},
{ERR_REASON(DSA_R_INVALID_DIGEST_TYPE)  ,"invalid digest type"},
{ERR_REASON(DSA_R_MISSING_PARAMETERS)   ,"missing parameters"},
{ERR_REASON(DSA_R_MODULUS_TOO_LARGE)    ,"modulus too large"},
{ERR_REASON(DSA_R_NEED_NEW_SETUP_VALUES) ,"need new setup values"},
{ERR_REASON(DSA_R_NON_FIPS_DSA_METHOD)  ,"non fips dsa method"},
{ERR_REASON(DSA_R_NO_PARAMETERS_SET)    ,"no parameters set"},
{ERR_REASON(DSA_R_PARAMETER_ENCODING_ERROR),"parameter encoding error"},
{0,NULL}
        };

#endif

void ERR_load_DSA_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(DSA_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,DSA_str_functs);
                ERR_load_strings(0,DSA_str_reasons);
```

```
128                 }
129 #endif
130         }
131 #endif /* ! codereview */
```

```
**********************************************************
    9998 Wed Aug 13 19:52:32 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_gen.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_gen.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#undef GENUINE_DSA

#ifdef GENUINE_DSA
```



```
/* Parameter generation follows the original release of FIPS PUB 186,
 * Appendix 2.2 (i.e. use SHA as defined in FIPS PUB 180) */
#define HASH    EVP_sha()
#else
/* Parameter generation follows the updated Appendix 2.2 for FIPS PUB 186,
 * also Appendix 2.2 of FIPS PUB 186-1 (i.e. use SHA as defined in
 * FIPS PUB 180-1) */
#define HASH    EVP_sha1()
#endif

#include <openssl/opensslconf.h> /* To see if OPENSSL_NO_SHA is defined */

#ifndef OPENSSL_NO_SHA

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
#include <openssl/bn.h>
#include <openssl/rand.h>
#include <openssl/sha.h>
#include "dsa_locl.h"

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

int DSA_generate_parameters_ex(DSA *ret, int bits,
                const unsigned char *seed_in, int seed_len,
                int *counter_ret, unsigned long *h_ret, BN_GENCB *cb)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(ret->meth->flags & DSA_FLAG_FIPS_METHOD)
                        && !(ret->flags & DSA_FLAG_NON_FIPS_ALLOW))
                {
                DSAerr(DSA_F_DSA_GENERATE_PARAMETERS_EX, DSA_R_NON_FIPS_DSA_METH
                return 0;
                }
#endif
        if(ret->meth->dsa_paramgen)
                return ret->meth->dsa_paramgen(ret, bits, seed_in, seed_len,
                                counter_ret, h_ret, cb);
#ifdef OPENSSL_FIPS
        else if (FIPS_mode())
                {
                return FIPS_dsa_generate_parameters_ex(ret, bits,
                                                       seed_in, seed_len,
                                                       counter_ret, h_ret, cb);
                }
#endif
        else
                {
                const EVP_MD *evpmd;
                size_t qbits = bits >= 2048 ? 256 : 160;

                if (bits >= 2048)
                        {
                        qbits = 256;
                        evpmd = EVP_sha256();
                        }
                else
                        {
                        qbits = 160;
                        evpmd = EVP_sha1();
                        }

                return dsa_builtin_paramgen(ret, bits, qbits, evpmd,
```

```
128                         seed_in, seed_len, NULL, counter_ret, h_ret, cb);
129                 }
130         }

132 int dsa_builtin_paramgen(DSA *ret, size_t bits, size_t qbits,
133         const EVP_MD *evpmd, const unsigned char *seed_in, size_t seed_len,
134         unsigned char *seed_out,
135         int *counter_ret, unsigned long *h_ret, BN_GENCB *cb)
136         {
137         int ok=0;
138         unsigned char seed[SHA256_DIGEST_LENGTH];
139         unsigned char md[SHA256_DIGEST_LENGTH];
140         unsigned char buf[SHA256_DIGEST_LENGTH],buf2[SHA256_DIGEST_LENGTH];
141         BIGNUM *r0,*W,*X,*c,*test;
142         BIGNUM *g=NULL,*q=NULL,*p=NULL;
143         BN_MONT_CTX *mont=NULL;
144         int i, k, n=0, m=0, qsize = qbits >> 3;
145         int counter=0;
146         int r=0;
147         BN_CTX *ctx=NULL;
148         unsigned int h=2;

150         if (qsize != SHA_DIGEST_LENGTH && qsize != SHA224_DIGEST_LENGTH &&
151             qsize != SHA256_DIGEST_LENGTH)
152                 /* invalid q size */
153                 return 0;

155         if (evpmd == NULL)
156                 /* use SHA1 as default */
157                 evpmd = EVP_sha1();

159         if (bits < 512)
160                 bits = 512;

162         bits = (bits+63)/64*64;

164         /* NB: seed_len == 0 is special case: copy generated seed to
165          * seed_in if it is not NULL.
166          */
167         if (seed_len && (seed_len < (size_t)qsize))
168                 seed_in = NULL;         /* seed buffer too small -- ignore */
169         if (seed_len > (size_t)qsize)
170                 seed_len = qsize;       /* App. 2.2 of FIPS PUB 186 allows large
171                                          * but our internal buffers are restrict
172         if (seed_in != NULL)
173                 memcpy(seed, seed_in, seed_len);

175         if ((ctx=BN_CTX_new()) == NULL)
176                 goto err;

178         if ((mont=BN_MONT_CTX_new()) == NULL)
179                 goto err;

181         BN_CTX_start(ctx);
182         r0 = BN_CTX_get(ctx);
183         g = BN_CTX_get(ctx);
184         W = BN_CTX_get(ctx);
185         q = BN_CTX_get(ctx);
186         X = BN_CTX_get(ctx);
187         c = BN_CTX_get(ctx);
188         p = BN_CTX_get(ctx);
189         test = BN_CTX_get(ctx);

191         if (!BN_lshift(test,BN_value_one(),bits-1))
192                 goto err;
```



```
194         for (;;)
195                 {
196                 for (;;) /* find q */
197                         {
198                         int seed_is_random;

200                         /* step 1 */
201                         if(!BN_GENCB_call(cb, 0, m++))
202                                 goto err;

204                         if (!seed_len)
205                                 {
206                                 RAND_pseudo_bytes(seed, qsize);
207                                 seed_is_random = 1;
208                                 }
209                         else
210                                 {
211                                 seed_is_random = 0;
212                                 seed_len=0; /* use random seed if 'seed_in' turn
213                                 }
214                         memcpy(buf , seed, qsize);
215                         memcpy(buf2, seed, qsize);
216                         /* precompute "SEED + 1" for step 7: */
217                         for (i = qsize-1; i >= 0; i--)
218                                 {
219                                 buf[i]++;
220                                 if (buf[i] != 0)
221                                         break;
222                                 }

224                         /* step 2 */
225                         if (!EVP_Digest(seed, qsize, md,   NULL, evpmd, NULL))
226                                 goto err;
227                         if (!EVP_Digest(buf,  qsize, buf2, NULL, evpmd, NULL))
228                                 goto err;
229                         for (i = 0; i < qsize; i++)
230                                 md[i]^=buf2[i];

232                         /* step 3 */
233                         md[0] |= 0x80;
234                         md[qsize-1] |= 0x01;
235                         if (!BN_bin2bn(md, qsize, q))
236                                 goto err;

238                         /* step 4 */
239                         r = BN_is_prime_fasttest_ex(q, DSS_prime_checks, ctx,
240                                         seed_is_random, cb);
241                         if (r > 0)
242                                 break;
243                         if (r != 0)
244                                 goto err;

246                         /* do a callback call */
247                         /* step 5 */
248                         }

250                 if(!BN_GENCB_call(cb, 2, 0)) goto err;
251                 if(!BN_GENCB_call(cb, 3, 0)) goto err;

253                 /* step 6 */
254                 counter=0;
255                 /* "offset = 2" */

257                 n=(bits-1)/160;

259                 for (;;)
```

```
260                         {
261                         if ((counter != 0) && !BN_GENCB_call(cb, 0, counter))
262                                 goto err;
263
264                         /* step 7 */
265                         BN_zero(W);
266                         /* now 'buf' contains "SEED + offset - 1" */
267                         for (k=0; k<=n; k++)
268                                 {
269                                 /* obtain "SEED + offset + k" by incrementing: *
270                                 for (i = qsize-1; i >= 0; i--)
271                                         {
272                                         buf[i]++;
273                                         if (buf[i] != 0)
274                                                 break;
275                                         }
276
277                                 if (!EVP_Digest(buf, qsize, md ,NULL, evpmd,
278                                                                         NULL))
279                                         goto err;
280
281                                 /* step 8 */
282                                 if (!BN_bin2bn(md, qsize, r0))
283                                         goto err;
284                                 if (!BN_lshift(r0,r0,(qsize << 3)*k)) goto err;
285                                 if (!BN_add(W,W,r0)) goto err;
286                                 }
287
288                         /* more of step 8 */
289                         if (!BN_mask_bits(W,bits-1)) goto err;
290                         if (!BN_copy(X,W)) goto err;
291                         if (!BN_add(X,X,test)) goto err;
292
293                         /* step 9 */
294                         if (!BN_lshift1(r0,q)) goto err;
295                         if (!BN_mod(c,X,r0,ctx)) goto err;
296                         if (!BN_sub(r0,c,BN_value_one())) goto err;
297                         if (!BN_sub(p,X,r0)) goto err;
298
299                         /* step 10 */
300                         if (BN_cmp(p,test) >= 0)
301                                 {
302                                 /* step 11 */
303                                 r = BN_is_prime_fasttest_ex(p, DSS_prime_checks,
304                                                         ctx, 1, cb);
305                                 if (r > 0)
306                                                         goto end; /* found it */
307                                 if (r != 0)
308                                         goto err;
309                                 }
310
311                         /* step 13 */
312                         counter++;
313                         /* "offset = offset + n + 1" */
314
315                         /* step 14 */
316                         if (counter >= 4096) break;
317                         }
318                 }
319 end:
320         if(!BN_GENCB_call(cb, 2, 1))
321                 goto err;
322
323         /* We now need to generate g */
324         /* Set r0=(p-1)/q */
325         if (!BN_sub(test,p,BN_value_one())) goto err;
```



```
326         if (!BN_div(r0,NULL,test,q,ctx)) goto err;
327
328         if (!BN_set_word(test,h)) goto err;
329         if (!BN_MONT_CTX_set(mont,p,ctx)) goto err;
330
331         for (;;)
332                 {
333                 /* g=test^r0%p */
334                 if (!BN_mod_exp_mont(g,test,r0,p,ctx,mont)) goto err;
335                 if (!BN_is_one(g)) break;
336                 if (!BN_add(test,test,BN_value_one())) goto err;
337                 h++;
338                 }
339
340         if(!BN_GENCB_call(cb, 3, 1))
341                 goto err;
342
343         ok=1;
344 err:
345         if (ok)
346                 {
347                 if(ret->p) BN_free(ret->p);
348                 if(ret->q) BN_free(ret->q);
349                 if(ret->g) BN_free(ret->g);
350                 ret->p=BN_dup(p);
351                 ret->q=BN_dup(q);
352                 ret->g=BN_dup(g);
353                 if (ret->p == NULL || ret->q == NULL || ret->g == NULL)
354                         {
355                         ok=0;
356                         goto err;
357                         }
358                 if (counter_ret != NULL) *counter_ret=counter;
359                 if (h_ret != NULL) *h_ret=h;
360                 if (seed_out)
361                         memcpy(seed_out, seed, qsize);
362                 }
363         if(ctx)
364                 {
365                 BN_CTX_end(ctx);
366                 BN_CTX_free(ctx);
367                 }
368         if (mont != NULL) BN_MONT_CTX_free(mont);
369         return ok;
370         }
371 #endif
372 #endif /* ! codereview */
```

```
**********************************************************
    4893 Wed Aug 13 19:52:32 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_key.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_key.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_SHA
#include <openssl/bn.h>
#include <openssl/dsa.h>
#include <openssl/rand.h>

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

static int dsa_builtin_keygen(DSA *dsa);

int DSA_generate_key(DSA *dsa)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(dsa->meth->flags & DSA_FLAG_FIPS_METHOD)
                        && !(dsa->flags & DSA_FLAG_NON_FIPS_ALLOW))
                {
                DSAerr(DSA_F_DSA_GENERATE_KEY, DSA_R_NON_FIPS_DSA_METHOD);
                return 0;
                }
#endif
        if(dsa->meth->dsa_keygen)
                return dsa->meth->dsa_keygen(dsa);
#ifdef OPENSSL_FIPS
        if (FIPS_mode())
                return FIPS_dsa_generate_key(dsa);
#endif
        return dsa_builtin_keygen(dsa);
        }

static int dsa_builtin_keygen(DSA *dsa)
        {
        int ok=0;
        BN_CTX *ctx=NULL;
        BIGNUM *pub_key=NULL,*priv_key=NULL;

        if ((ctx=BN_CTX_new()) == NULL) goto err;

        if (dsa->priv_key == NULL)
                {
                if ((priv_key=BN_new()) == NULL) goto err;
                }
        else
                priv_key=dsa->priv_key;

        do
                if (!BN_rand_range(priv_key,dsa->q)) goto err;
        while (BN_is_zero(priv_key));

        if (dsa->pub_key == NULL)
                {
                if ((pub_key=BN_new()) == NULL) goto err;
                }
        else
                pub_key=dsa->pub_key;

        {
                BIGNUM local_prk;
                BIGNUM *prk;

                if ((dsa->flags & DSA_FLAG_NO_EXP_CONSTTIME) == 0)
                        {
                        BN_init(&local_prk);
                        prk = &local_prk;
                        BN_with_flags(prk, priv_key, BN_FLG_CONSTTIME);
                        }
```

```
128                 else
129                         prk = priv_key;
130
131                 if (!BN_mod_exp(pub_key,dsa->g,prk,dsa->p,ctx)) goto err;
132         }
133
134         dsa->priv_key=priv_key;
135         dsa->pub_key=pub_key;
136         ok=1;
137
138 err:
139         if ((pub_key != NULL) && (dsa->pub_key == NULL)) BN_free(pub_key);
140         if ((priv_key != NULL) && (dsa->priv_key == NULL)) BN_free(priv_key);
141         if (ctx != NULL) BN_CTX_free(ctx);
142         return(ok);
143         }
144 #endif
145 #endif /* ! codereview */
```

```
**********************************************************
    8689 Wed Aug 13 19:52:33 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Original version from Steven Schoch <schoch@sheba.arc.nasa.gov> */

#include <stdio.h>
```



```
#include "cryptlib.h"
#include <openssl/bn.h>
#include <openssl/dsa.h>
#include <openssl/asn1.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

const char DSA_version[]="DSA" OPENSSL_VERSION_PTEXT;

static const DSA_METHOD *default_DSA_method = NULL;

void DSA_set_default_method(const DSA_METHOD *meth)
        {
        default_DSA_method = meth;
        }

const DSA_METHOD *DSA_get_default_method(void)
        {
        if(!default_DSA_method)
                {
#ifdef OPENSSL_FIPS
                if (FIPS_mode())
                        return FIPS_dsa_openssl();
                else
                        return DSA_OpenSSL();
#else
                default_DSA_method = DSA_OpenSSL();
#endif
                }
        return default_DSA_method;
        }

DSA *DSA_new(void)
        {
        return DSA_new_method(NULL);
        }

int DSA_set_method(DSA *dsa, const DSA_METHOD *meth)
        {
        /* NB: The caller is specifically setting a method, so it's not up to us
         * to deal with which ENGINE it comes from. */
        const DSA_METHOD *mtmp;
        mtmp = dsa->meth;
        if (mtmp->finish) mtmp->finish(dsa);
#ifndef OPENSSL_NO_ENGINE
        if (dsa->engine)
                {
                ENGINE_finish(dsa->engine);
                dsa->engine = NULL;
                }
#endif
        dsa->meth = meth;
        if (meth->init) meth->init(dsa);
        return 1;
        }

DSA *DSA_new_method(ENGINE *engine)
        {
```

```
128         DSA *ret;

130         ret=(DSA *)OPENSSL_malloc(sizeof(DSA));
131         if (ret == NULL)
132                 {
133                 DSAerr(DSA_F_DSA_NEW_METHOD,ERR_R_MALLOC_FAILURE);
134                 return(NULL);
135                 }
136         ret->meth = DSA_get_default_method();
137 #ifndef OPENSSL_NO_ENGINE
138         if (engine)
139                 {
140                 if (!ENGINE_init(engine))
141                         {
142                         DSAerr(DSA_F_DSA_NEW_METHOD, ERR_R_ENGINE_LIB);
143                         OPENSSL_free(ret);
144                         return NULL;
145                         }
146                 ret->engine = engine;
147                 }
148         else
149                 ret->engine = ENGINE_get_default_DSA();
150         if(ret->engine)
151                 {
152                 ret->meth = ENGINE_get_DSA(ret->engine);
153                 if(!ret->meth)
154                         {
155                         DSAerr(DSA_F_DSA_NEW_METHOD,
156                                 ERR_R_ENGINE_LIB);
157                         ENGINE_finish(ret->engine);
158                         OPENSSL_free(ret);
159                         return NULL;
160                         }
161                 }
162 #endif

164         ret->pad=0;
165         ret->version=0;
166         ret->write_params=1;
167         ret->p=NULL;
168         ret->q=NULL;
169         ret->g=NULL;

171         ret->pub_key=NULL;
172         ret->priv_key=NULL;

174         ret->kinv=NULL;
175         ret->r=NULL;
176         ret->method_mont_p=NULL;

178         ret->references=1;
179         ret->flags=ret->meth->flags & ~DSA_FLAG_NON_FIPS_ALLOW;
180         CRYPTO_new_ex_data(CRYPTO_EX_INDEX_DSA, ret, &ret->ex_data);
181         if ((ret->meth->init != NULL) && !ret->meth->init(ret))
182                 {
183 #ifndef OPENSSL_NO_ENGINE
184                 if (ret->engine)
185                         ENGINE_finish(ret->engine);
186 #endif
187                 CRYPTO_free_ex_data(CRYPTO_EX_INDEX_DSA, ret, &ret->ex_data);
188                 OPENSSL_free(ret);
189                 ret=NULL;
190                 }

192         return(ret);
193         }
```



```
195 void DSA_free(DSA *r)
196         {
197         int i;

199         if (r == NULL) return;

201         i=CRYPTO_add(&r->references,-1,CRYPTO_LOCK_DSA);
202 #ifdef REF_PRINT
203         REF_PRINT("DSA",r);
204 #endif
205         if (i > 0) return;
206 #ifdef REF_CHECK
207         if (i < 0)
208                 {
209                 fprintf(stderr,"DSA_free, bad reference count\n");
210                 abort();
211                 }
212 #endif

214         if(r->meth->finish)
215                 r->meth->finish(r);
216 #ifndef OPENSSL_NO_ENGINE
217         if(r->engine)
218                 ENGINE_finish(r->engine);
219 #endif

221         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_DSA, r, &r->ex_data);

223         if (r->p != NULL) BN_clear_free(r->p);
224         if (r->q != NULL) BN_clear_free(r->q);
225         if (r->g != NULL) BN_clear_free(r->g);
226         if (r->pub_key != NULL) BN_clear_free(r->pub_key);
227         if (r->priv_key != NULL) BN_clear_free(r->priv_key);
228         if (r->kinv != NULL) BN_clear_free(r->kinv);
229         if (r->r != NULL) BN_clear_free(r->r);
230         OPENSSL_free(r);
231         }

233 int DSA_up_ref(DSA *r)
234         {
235         int i = CRYPTO_add(&r->references, 1, CRYPTO_LOCK_DSA);
236 #ifdef REF_PRINT
237         REF_PRINT("DSA",r);
238 #endif
239 #ifdef REF_CHECK
240         if (i < 2)
241                 {
242                 fprintf(stderr, "DSA_up_ref, bad reference count\n");
243                 abort();
244                 }
245 #endif
246         return ((i > 1) ? 1 : 0);
247         }

249 int DSA_size(const DSA *r)
250         {
251         int ret,i;
252         ASN1_INTEGER bs;
253         unsigned char buf[4];   /* 4 bytes looks really small.
254                                    However, i2d_ASN1_INTEGER() will not look
255                                    beyond the first byte, as long as the second
256                                    parameter is NULL. */

258         i=BN_num_bits(r->q);
259         bs.length=(i+7)/8;
```

```
260         bs.data=buf;
261         bs.type=V_ASN1_INTEGER;
262         /* If the top bit is set the asn1 encoding is 1 larger. */
263         buf[0]=0xff;

265         i=i2d_ASN1_INTEGER(&bs,NULL);
266         i+=i; /* r and s */
267         ret=ASN1_object_size(1,i,V_ASN1_SEQUENCE);
268         return(ret);
269         }

271 int DSA_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
272              CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
273         {
274         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_DSA, argl, argp,
275                                 new_func, dup_func, free_func);
276         }

278 int DSA_set_ex_data(DSA *d, int idx, void *arg)
279         {
280         return(CRYPTO_set_ex_data(&d->ex_data,idx,arg));
281         }

283 void *DSA_get_ex_data(DSA *d, int idx)
284         {
285         return(CRYPTO_get_ex_data(&d->ex_data,idx));
286         }

288 #ifndef OPENSSL_NO_DH
289 DH *DSA_dup_DH(const DSA *r)
290         {
291         /* DSA has p, q, g, optional pub_key, optional priv_key.
292          * DH has p, optional length, g, optional pub_key, optional priv_key,
293          * optional q.
294          */

296         DH *ret = NULL;

298         if (r == NULL)
299                 goto err;
300         ret = DH_new();
301         if (ret == NULL)
302                 goto err;
303         if (r->p != NULL)
304                 if ((ret->p = BN_dup(r->p)) == NULL)
305                         goto err;
306         if (r->q != NULL)
307                 {
308                 ret->length = BN_num_bits(r->q);
309                 if ((ret->q = BN_dup(r->q)) == NULL)
310                         goto err;
311                 }
312         if (r->g != NULL)
313                 if ((ret->g = BN_dup(r->g)) == NULL)
314                         goto err;
315         if (r->pub_key != NULL)
316                 if ((ret->pub_key = BN_dup(r->pub_key)) == NULL)
317                         goto err;
318         if (r->priv_key != NULL)
319                 if ((ret->priv_key = BN_dup(r->priv_key)) == NULL)
320                         goto err;

322         return ret;

324  err:
325         if (ret != NULL)
```



```
326                 DH_free(ret);
327         return NULL;
328         }
329 #endif
330 #endif /* ! codereview */
```

```
**********************************************************
   11302 Wed Aug 13 19:52:33 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_ossl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_ossl.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Original version from Steven Schoch <schoch@sheba.arc.nasa.gov> */

#include <stdio.h>
```



```
#include "cryptlib.h"
#include <openssl/bn.h>
#include <openssl/sha.h>
#include <openssl/dsa.h>
#include <openssl/rand.h>
#include <openssl/asn1.h>

static DSA_SIG *dsa_do_sign(const unsigned char *dgst, int dlen, DSA *dsa);
static int dsa_sign_setup(DSA *dsa, BN_CTX *ctx_in, BIGNUM **kinvp, BIGNUM **rp)
static int dsa_do_verify(const unsigned char *dgst, int dgst_len, DSA_SIG *sig,
                         DSA *dsa);
static int dsa_init(DSA *dsa);
static int dsa_finish(DSA *dsa);

static DSA_METHOD openssl_dsa_meth = {
"OpenSSL DSA method",
dsa_do_sign,
dsa_sign_setup,
dsa_do_verify,
NULL, /* dsa_mod_exp, */
NULL, /* dsa_bn_mod_exp, */
dsa_init,
dsa_finish,
0,
NULL,
NULL,
NULL
};

/* These macro wrappers replace attempts to use the dsa_mod_exp() and
 * bn_mod_exp() handlers in the DSA_METHOD structure. We avoid the problem of
 * having a the macro work as an expression by bundling an "err_instr". So;
 *
 *     if (!dsa->meth->bn_mod_exp(dsa, r,dsa->g,&k,dsa->p,ctx,
 *                 dsa->method_mont_p)) goto err;
 *
 * can be replaced by;
 *
 *     DSA_BN_MOD_EXP(goto err, dsa, r, dsa->g, &k, dsa->p, ctx,
 *                 dsa->method_mont_p);
 */

#define DSA_MOD_EXP(err_instr,dsa,rr,a1,p1,a2,p2,m,ctx,in_mont) \
        do { \
        int _tmp_res53; \
        if((dsa)->meth->dsa_mod_exp) \
                _tmp_res53 = (dsa)->meth->dsa_mod_exp((dsa), (rr), (a1), (p1), \
                                (a2), (p2), (m), (ctx), (in_mont)); \
        else \
                _tmp_res53 = BN_mod_exp2_mont((rr), (a1), (p1), (a2), (p2), \
                                (m), (ctx), (in_mont)); \
        if(!_tmp_res53) err_instr; \
        } while(0)
#define DSA_BN_MOD_EXP(err_instr,dsa,r,a,p,m,ctx,m_ctx) \
        do { \
        int _tmp_res53; \
        if((dsa)->meth->bn_mod_exp) \
                _tmp_res53 = (dsa)->meth->bn_mod_exp((dsa), (r), (a), (p), \
                                (m), (ctx), (m_ctx)); \
        else \
                _tmp_res53 = BN_mod_exp_mont((r), (a), (p), (m), (ctx), (m_ctx))
        if(!_tmp_res53) err_instr; \
        } while(0)

const DSA_METHOD *DSA_OpenSSL(void)
{
```

```
128         return &openssl_dsa_meth;
129 }

131 static DSA_SIG *dsa_do_sign(const unsigned char *dgst, int dlen, DSA *dsa)
132         {
133         BIGNUM *kinv=NULL,*r=NULL,*s=NULL;
134         BIGNUM m;
135         BIGNUM xr;
136         BN_CTX *ctx=NULL;
137         int reason=ERR_R_BN_LIB;
138         DSA_SIG *ret=NULL;
139         int noredo = 0;

141         BN_init(&m);
142         BN_init(&xr);

144         if (!dsa->p || !dsa->q || !dsa->g)
145                 {
146                 reason=DSA_R_MISSING_PARAMETERS;
147                 goto err;
148                 }

150         s=BN_new();
151         if (s == NULL) goto err;
152         ctx=BN_CTX_new();
153         if (ctx == NULL) goto err;
154 redo:
155         if ((dsa->kinv == NULL) || (dsa->r == NULL))
156                 {
157                 if (!DSA_sign_setup(dsa,ctx,&kinv,&r)) goto err;
158                 }
159         else
160                 {
161                 kinv=dsa->kinv;
162                 dsa->kinv=NULL;
163                 r=dsa->r;
164                 dsa->r=NULL;
165                 noredo = 1;
166                 }


169         if (dlen > BN_num_bytes(dsa->q))
170                 /* if the digest length is greater than the size of q use the
171                  * BN_num_bits(dsa->q) leftmost bits of the digest, see
172                  * fips 186-3, 4.2 */
173                 dlen = BN_num_bytes(dsa->q);
174         if (BN_bin2bn(dgst,dlen,&m) == NULL)
175                 goto err;

177         /* Compute  s = inv(k) (m + xr) mod q */
178         if (!BN_mod_mul(&xr,dsa->priv_key,r,dsa->q,ctx)) goto err;/* s = xr */
179         if (!BN_add(s, &xr, &m)) goto err;              /* s = m + xr */
180         if (BN_cmp(s,dsa->q) > 0)
181                 if (!BN_sub(s,s,dsa->q)) goto err;
182         if (!BN_mod_mul(s,s,kinv,dsa->q,ctx)) goto err;

184         ret=DSA_SIG_new();
185         if (ret == NULL) goto err;
186         /* Redo if r or s is zero as required by FIPS 186-3: this is
187          * very unlikely.
188          */
189         if (BN_is_zero(r) || BN_is_zero(s))
190                 {
191                 if (noredo)
192                         {
193                         reason = DSA_R_NEED_NEW_SETUP_VALUES;
```



```
194                         goto err;
195                         }
196                 goto redo;
197                 }
198         ret->r = r;
199         ret->s = s;

201 err:
202         if (!ret)
203                 {
204                 DSAerr(DSA_F_DSA_DO_SIGN,reason);
205                 BN_free(r);
206                 BN_free(s);
207                 }
208         if (ctx != NULL) BN_CTX_free(ctx);
209         BN_clear_free(&m);
210         BN_clear_free(&xr);
211         if (kinv != NULL) /* dsa->kinv is NULL now if we used it */
212             BN_clear_free(kinv);
213         return(ret);
214         }

216 static int dsa_sign_setup(DSA *dsa, BN_CTX *ctx_in, BIGNUM **kinvp, BIGNUM **rp)
217         {
218         BN_CTX *ctx;
219         BIGNUM k,kq,*K,*kinv=NULL,*r=NULL;
220         int ret=0;

222         if (!dsa->p || !dsa->q || !dsa->g)
223                 {
224                 DSAerr(DSA_F_DSA_SIGN_SETUP,DSA_R_MISSING_PARAMETERS);
225                 return 0;
226                 }

228         BN_init(&k);
229         BN_init(&kq);

231         if (ctx_in == NULL)
232                 {
233                 if ((ctx=BN_CTX_new()) == NULL) goto err;
234                 }
235         else
236                 ctx=ctx_in;

238         if ((r=BN_new()) == NULL) goto err;

240         /* Get random k */
241         do
242                 if (!BN_rand_range(&k, dsa->q)) goto err;
243         while (BN_is_zero(&k));
244         if ((dsa->flags & DSA_FLAG_NO_EXP_CONSTTIME) == 0)
245                 {
246                 BN_set_flags(&k, BN_FLG_CONSTTIME);
247                 }

249         if (dsa->flags & DSA_FLAG_CACHE_MONT_P)
250                 {
251                 if (!BN_MONT_CTX_set_locked(&dsa->method_mont_p,
252                                             CRYPTO_LOCK_DSA,
253                                             dsa->p, ctx))
254                         goto err;
255                 }

257         /* Compute r = (g^k mod p) mod q */

259         if ((dsa->flags & DSA_FLAG_NO_EXP_CONSTTIME) == 0)
```

```
260                 {
261                 if (!BN_copy(&kq, &k)) goto err;

263                 /* We do not want timing information to leak the length of k,
264                  * so we compute g^k using an equivalent exponent of fixed lengt
265                  *
266                  * (This is a kludge that we need because the BN_mod_exp_mont()
267                  * does not let us specify the desired timing behaviour.) */

269                 if (!BN_add(&kq, &kq, dsa->q)) goto err;
270                 if (BN_num_bits(&kq) <= BN_num_bits(dsa->q))
271                         {
272                         if (!BN_add(&kq, &kq, dsa->q)) goto err;
273                         }

275                 K = &kq;
276                 }
277         else
278                 {
279                 K = &k;
280                 }
281         DSA_BN_MOD_EXP(goto err, dsa, r, dsa->g, K, dsa->p, ctx,
282                        dsa->method_mont_p);
283         if (!BN_mod(r,r,dsa->q,ctx)) goto err;

285         /* Compute  part of 's = inv(k) (m + xr) mod q' */
286         if ((kinv=BN_mod_inverse(NULL,&k,dsa->q,ctx)) == NULL) goto err;

288         if (*kinvp != NULL) BN_clear_free(*kinvp);
289         *kinvp=kinv;
290         kinv=NULL;
291         if (*rp != NULL) BN_clear_free(*rp);
292         *rp=r;
293         ret=1;
294 err:
295         if (!ret)
296                 {
297                 DSAerr(DSA_F_DSA_SIGN_SETUP,ERR_R_BN_LIB);
298                 if (r != NULL)
299                         BN_clear_free(r);
300                 }
301         if (ctx_in == NULL) BN_CTX_free(ctx);
302         BN_clear_free(&k);
303         BN_clear_free(&kq);
304         return(ret);
305         }

307 static int dsa_do_verify(const unsigned char *dgst, int dgst_len, DSA_SIG *sig,
308                          DSA *dsa)
309         {
310         BN_CTX *ctx;
311         BIGNUM u1,u2,t1;
312         BN_MONT_CTX *mont=NULL;
313         int ret = -1, i;
314         if (!dsa->p || !dsa->q || !dsa->g)
315                 {
316                 DSAerr(DSA_F_DSA_DO_VERIFY,DSA_R_MISSING_PARAMETERS);
317                 return -1;
318                 }

320         i = BN_num_bits(dsa->q);
321         /* fips 186-3 allows only different sizes for q */
322         if (i != 160 && i != 224 && i != 256)
323                 {
324                 DSAerr(DSA_F_DSA_DO_VERIFY,DSA_R_BAD_Q_VALUE);
325                 return -1;
```



```
326                 }

328         if (BN_num_bits(dsa->p) > OPENSSL_DSA_MAX_MODULUS_BITS)
329                 {
330                 DSAerr(DSA_F_DSA_DO_VERIFY,DSA_R_MODULUS_TOO_LARGE);
331                 return -1;
332                 }
333         BN_init(&u1);
334         BN_init(&u2);
335         BN_init(&t1);

337         if ((ctx=BN_CTX_new()) == NULL) goto err;

339         if (BN_is_zero(sig->r) || BN_is_negative(sig->r) ||
340             BN_ucmp(sig->r, dsa->q) >= 0)
341                 {
342                 ret = 0;
343                 goto err;
344                 }
345         if (BN_is_zero(sig->s) || BN_is_negative(sig->s) ||
346             BN_ucmp(sig->s, dsa->q) >= 0)
347                 {
348                 ret = 0;
349                 goto err;
350                 }

352         /* Calculate W = inv(S) mod Q
353          * save W in u2 */
354         if ((BN_mod_inverse(&u2,sig->s,dsa->q,ctx)) == NULL) goto err;

356         /* save M in u1 */
357         if (dgst_len > (i >> 3))
358                 /* if the digest length is greater than the size of q use the
359                  * BN_num_bits(dsa->q) leftmost bits of the digest, see
360                  * fips 186-3, 4.2 */
361                 dgst_len = (i >> 3);
362         if (BN_bin2bn(dgst,dgst_len,&u1) == NULL) goto err;

364         /* u1 = M * w mod q */
365         if (!BN_mod_mul(&u1,&u1,&u2,dsa->q,ctx)) goto err;

367         /* u2 = r * w mod q */
368         if (!BN_mod_mul(&u2,sig->r,&u2,dsa->q,ctx)) goto err;


371         if (dsa->flags & DSA_FLAG_CACHE_MONT_P)
372                 {
373                 mont = BN_MONT_CTX_set_locked(&dsa->method_mont_p,
374                                               CRYPTO_LOCK_DSA, dsa->p, ctx);
375                 if (!mont)
376                         goto err;
377                 }


380         DSA_MOD_EXP(goto err, dsa, &t1, dsa->g, &u1, dsa->pub_key, &u2, dsa->p,
381         /* BN_copy(&u1,&t1); */
382         /* let u1 = u1 mod q */
383         if (!BN_mod(&u1,&t1,dsa->q,ctx)) goto err;

385         /* V is now in u1.  If the signature is correct, it will be
386          * equal to R. */
387         ret=(BN_ucmp(&u1, sig->r) == 0);

389         err:
390         /* XXX: surely this is wrong - if ret is 0, it just didn't verify;
391            there is no error in BN. Test should be ret == -1 (Ben) */
```

```
392         if (ret != 1) DSAerr(DSA_F_DSA_DO_VERIFY,ERR_R_BN_LIB);
393         if (ctx != NULL) BN_CTX_free(ctx);
394         BN_free(&u1);
395         BN_free(&u2);
396         BN_free(&t1);
397         return(ret);
398         }

400 static int dsa_init(DSA *dsa)
401 {
402         dsa->flags|=DSA_FLAG_CACHE_MONT_P;
403         return(1);
404 }

406 static int dsa_finish(DSA *dsa)
407 {
408         if(dsa->method_mont_p)
409                 BN_MONT_CTX_free(dsa->method_mont_p);
410         return(1);
411 }
412 #endif /* ! codereview */
```

```
**********************************************************
    8229 Wed Aug 13 19:52:33 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_pmeth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
#include <openssl/x509.h>
```



```
#include <openssl/evp.h>
#include <openssl/bn.h>
#include "evp_locl.h"
#include "dsa_locl.h"

/* DSA pkey context structure */

typedef struct
        {
        /* Parameter gen parameters */
        int nbits;              /* size of p in bits (default: 1024) */
        int qbits;              /* size of q in bits (default: 160)  */
        const EVP_MD *pmd;      /* MD for parameter generation */
        /* Keygen callback info */
        int gentmp[2];
        /* message digest */
        const EVP_MD *md;       /* MD for the signature */
        } DSA_PKEY_CTX;

static int pkey_dsa_init(EVP_PKEY_CTX *ctx)
        {
        DSA_PKEY_CTX *dctx;
        dctx = OPENSSL_malloc(sizeof(DSA_PKEY_CTX));
        if (!dctx)
                return 0;
        dctx->nbits = 1024;
        dctx->qbits = 160;
        dctx->pmd = NULL;
        dctx->md = NULL;

        ctx->data = dctx;
        ctx->keygen_info = dctx->gentmp;
        ctx->keygen_info_count = 2;

        return 1;
        }

static int pkey_dsa_copy(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src)
        {
        DSA_PKEY_CTX *dctx, *sctx;
        if (!pkey_dsa_init(dst))
                return 0;
        sctx = src->data;
        dctx = dst->data;
        dctx->nbits = sctx->nbits;
        dctx->qbits = sctx->qbits;
        dctx->pmd = sctx->pmd;
        dctx->md  = sctx->md;
        return 1;
        }

static void pkey_dsa_cleanup(EVP_PKEY_CTX *ctx)
        {
        DSA_PKEY_CTX *dctx = ctx->data;
        if (dctx)
                OPENSSL_free(dctx);
        }

static int pkey_dsa_sign(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
                                        const unsigned char *tbs, size_t tbslen)
        {
        int ret, type;
        unsigned int sltmp;
        DSA_PKEY_CTX *dctx = ctx->data;
        DSA *dsa = ctx->pkey->pkey.dsa;
```

```
128         if (dctx->md)
129                 type = EVP_MD_type(dctx->md);
130         else
131                 type = NID_sha1;

133         ret = DSA_sign(type, tbs, tbslen, sig, &sltmp, dsa);

135         if (ret <= 0)
136                 return ret;
137         *siglen = sltmp;
138         return 1;
139         }

141 static int pkey_dsa_verify(EVP_PKEY_CTX *ctx,
142                                     const unsigned char *sig, size_t siglen,
143                                     const unsigned char *tbs, size_t tbslen)
144         {
145         int ret, type;
146         DSA_PKEY_CTX *dctx = ctx->data;
147         DSA *dsa = ctx->pkey->pkey.dsa;

149         if (dctx->md)
150                 type = EVP_MD_type(dctx->md);
151         else
152                 type = NID_sha1;

154         ret = DSA_verify(type, tbs, tbslen, sig, siglen, dsa);

156         return ret;
157         }

159 static int pkey_dsa_ctrl(EVP_PKEY_CTX *ctx, int type, int p1, void *p2)
160         {
161         DSA_PKEY_CTX *dctx = ctx->data;
162         switch (type)
163                 {
164                 case EVP_PKEY_CTRL_DSA_PARAMGEN_BITS:
165                 if (p1 < 256)
166                         return -2;
167                 dctx->nbits = p1;
168                 return 1;

170                 case EVP_PKEY_CTRL_DSA_PARAMGEN_Q_BITS:
171                 if (p1 != 160 && p1 != 224 && p1 && p1 != 256)
172                         return -2;
173                 dctx->qbits = p1;
174                 return 1;

176                 case EVP_PKEY_CTRL_DSA_PARAMGEN_MD:
177                 if (EVP_MD_type((const EVP_MD *)p2) != NID_sha1   &&
178                     EVP_MD_type((const EVP_MD *)p2) != NID_sha224 &&
179                     EVP_MD_type((const EVP_MD *)p2) != NID_sha256)
180                         {
181                         DSAerr(DSA_F_PKEY_DSA_CTRL, DSA_R_INVALID_DIGEST_TYPE);
182                         return 0;
183                         }
184                 dctx->md = p2;
185                 return 1;

187                 case EVP_PKEY_CTRL_MD:
188                 if (EVP_MD_type((const EVP_MD *)p2) != NID_sha1   &&
189                     EVP_MD_type((const EVP_MD *)p2) != NID_dsa    &&
190                     EVP_MD_type((const EVP_MD *)p2) != NID_dsaWithSHA    &&
191                     EVP_MD_type((const EVP_MD *)p2) != NID_sha224 &&
192                     EVP_MD_type((const EVP_MD *)p2) != NID_sha256 &&
193                     EVP_MD_type((const EVP_MD *)p2) != NID_sha384 &&
```



```
194                     EVP_MD_type((const EVP_MD *)p2) != NID_sha512)
195                         {
196                         DSAerr(DSA_F_PKEY_DSA_CTRL, DSA_R_INVALID_DIGEST_TYPE);
197                         return 0;
198                         }
199                 dctx->md = p2;
200                 return 1;

202                 case EVP_PKEY_CTRL_DIGESTINIT:
203                 case EVP_PKEY_CTRL_PKCS7_SIGN:
204                 case EVP_PKEY_CTRL_CMS_SIGN:
205                 return 1;

207                 case EVP_PKEY_CTRL_PEER_KEY:
208                         DSAerr(DSA_F_PKEY_DSA_CTRL,
209                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
210                         return -2;
211                 default:
212                 return -2;

214                 }
215         }

217 static int pkey_dsa_ctrl_str(EVP_PKEY_CTX *ctx,
218                         const char *type, const char *value)
219         {
220         if (!strcmp(type, "dsa_paramgen_bits"))
221                 {
222                 int nbits;
223                 nbits = atoi(value);
224                 return EVP_PKEY_CTX_set_dsa_paramgen_bits(ctx, nbits);
225                 }
226         if (!strcmp(type, "dsa_paramgen_q_bits"))
227                 {
228                 int qbits = atoi(value);
229                 return EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_DSA, EVP_PKEY_OP_PARAMGEN
230                                         EVP_PKEY_CTRL_DSA_PARAMGEN_Q_BITS, qbit
231                 }
232         if (!strcmp(type, "dsa_paramgen_md"))
233                 {
234                 return EVP_PKEY_CTX_ctrl(ctx, EVP_PKEY_DSA, EVP_PKEY_OP_PARAMGEN
235                                         EVP_PKEY_CTRL_DSA_PARAMGEN_MD, 0,
236                                         (void *)EVP_get_digestbyname(value));
237                 }
238         return -2;
239         }

241 static int pkey_dsa_paramgen(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey)
242         {
243         DSA *dsa = NULL;
244         DSA_PKEY_CTX *dctx = ctx->data;
245         BN_GENCB *pcb, cb;
246         int ret;
247         if (ctx->pkey_gencb)
248                 {
249                 pcb = &cb;
250                 evp_pkey_set_cb_translate(pcb, ctx);
251                 }
252         else
253                 pcb = NULL;
254         dsa = DSA_new();
255         if (!dsa)
256                 return 0;
257         ret = dsa_builtin_paramgen(dsa, dctx->nbits, dctx->qbits, dctx->pmd,
258                                    NULL, 0, NULL, NULL, NULL, pcb);
259         if (ret)
```

```
260                 EVP_PKEY_assign_DSA(pkey, dsa);
261         else
262                 DSA_free(dsa);
263         return ret;
264         }

266 static int pkey_dsa_keygen(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey)
267         {
268         DSA *dsa = NULL;
269         if (ctx->pkey == NULL)
270                 {
271                 DSAerr(DSA_F_PKEY_DSA_KEYGEN, DSA_R_NO_PARAMETERS_SET);
272                 return 0;
273                 }
274         dsa = DSA_new();
275         if (!dsa)
276                 return 0;
277         EVP_PKEY_assign_DSA(pkey, dsa);
278         /* Note: if error return, pkey is freed by parent routine */
279         if (!EVP_PKEY_copy_parameters(pkey, ctx->pkey))
280                 return 0;
281         return DSA_generate_key(pkey->pkey.dsa);
282         }

284 const EVP_PKEY_METHOD dsa_pkey_meth =
285         {
286         EVP_PKEY_DSA,
287         EVP_PKEY_FLAG_AUTOARGLEN,
288         pkey_dsa_init,
289         pkey_dsa_copy,
290         pkey_dsa_cleanup,

292         0,
293         pkey_dsa_paramgen,

295         0,
296         pkey_dsa_keygen,

298         0,
299         pkey_dsa_sign,

301         0,
302         pkey_dsa_verify,

304         0,0,

306         0,0,0,0,

308         0,0,

310         0,0,

312         0,0,

314         pkey_dsa_ctrl,
315         pkey_dsa_ctrl_str


318         };
319 #endif /* ! codereview */
```

```
**********************************************************
    3882 Wed Aug 13 19:52:33 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_prn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_prn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/dsa.h>

#ifndef OPENSSL_NO_FP_API
int DSA_print_fp(FILE *fp, const DSA *x, int off)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                DSAerr(DSA_F_DSA_PRINT_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=DSA_print(b,x,off);
        BIO_free(b);
        return(ret);
        }

int DSAparams_print_fp(FILE *fp, const DSA *x)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                DSAerr(DSA_F_DSAPARAMS_PRINT_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=DSAparams_print(b, x);
        BIO_free(b);
        return(ret);
        }
#endif

int DSA_print(BIO *bp, const DSA *x, int off)
        {
        EVP_PKEY *pk;
        int ret;
        pk = EVP_PKEY_new();
        if (!pk || !EVP_PKEY_set1_DSA(pk, (DSA *)x))
                return 0;
        ret = EVP_PKEY_print_private(bp, pk, off, NULL);
        EVP_PKEY_free(pk);
        return ret;
        }

int DSAparams_print(BIO *bp, const DSA *x)
        {
        EVP_PKEY *pk;
        int ret;
        pk = EVP_PKEY_new();
        if (!pk || !EVP_PKEY_set1_DSA(pk, (DSA *)x))
                return 0;
        ret = EVP_PKEY_print_params(bp, pk, 4, NULL);
        EVP_PKEY_free(pk);
        return ret;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4363 Wed Aug 13 19:52:33 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_sign.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_sign.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Original version from Steven Schoch <schoch@sheba.arc.nasa.gov> */

#include "cryptlib.h"
```



```
#include <openssl/dsa.h>
#include <openssl/rand.h>
#include <openssl/bn.h>

DSA_SIG * DSA_do_sign(const unsigned char *dgst, int dlen, DSA *dsa)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(dsa->meth->flags & DSA_FLAG_FIPS_METHOD)
                        && !(dsa->flags & DSA_FLAG_NON_FIPS_ALLOW))
                {
                DSAerr(DSA_F_DSA_DO_SIGN, DSA_R_NON_FIPS_DSA_METHOD);
                return NULL;
                }
#endif
        return dsa->meth->dsa_do_sign(dgst, dlen, dsa);
        }

int DSA_sign_setup(DSA *dsa, BN_CTX *ctx_in, BIGNUM **kinvp, BIGNUM **rp)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(dsa->meth->flags & DSA_FLAG_FIPS_METHOD)
                        && !(dsa->flags & DSA_FLAG_NON_FIPS_ALLOW))
                {
                DSAerr(DSA_F_DSA_SIGN_SETUP, DSA_R_NON_FIPS_DSA_METHOD);
                return 0;
                }
#endif
        return dsa->meth->dsa_sign_setup(dsa, ctx_in, kinvp, rp);
        }

DSA_SIG *DSA_SIG_new(void)
        {
        DSA_SIG *sig;
        sig = OPENSSL_malloc(sizeof(DSA_SIG));
        if (!sig)
                return NULL;
        sig->r = NULL;
        sig->s = NULL;
        return sig;
        }

void DSA_SIG_free(DSA_SIG *sig)
        {
        if (sig)
                {
                if (sig->r)
                        BN_free(sig->r);
                if (sig->s)
                        BN_free(sig->s);
                OPENSSL_free(sig);
                }
        }
#endif /* ! codereview */
```

```
**********************************************************
    3679 Wed Aug 13 19:52:33 2014
new/usr/src/lib/openssl/libsunw_crypto/dsa/dsa_vrf.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dsa/dsa_vrf.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Original version from Steven Schoch <schoch@sheba.arc.nasa.gov> */

#include "cryptlib.h"
```



```
#include <openssl/dsa.h>

int DSA_do_verify(const unsigned char *dgst, int dgst_len, DSA_SIG *sig,
                  DSA *dsa)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(dsa->meth->flags & DSA_FLAG_FIPS_METHOD)
                        && !(dsa->flags & DSA_FLAG_NON_FIPS_ALLOW))
                {
                DSAerr(DSA_F_DSA_DO_VERIFY, DSA_R_NON_FIPS_DSA_METHOD);
                return -1;
                }
#endif
        return dsa->meth->dsa_do_verify(dgst, dgst_len, sig, dsa);
        }
#endif /* ! codereview */
```

```
**********************************************************
    7520 Wed Aug 13 19:52:34 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_beos.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso_beos.c */
/* Written by Marcin Konicki (ahwayakchih@neoni.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <string.h>
#include "cryptlib.h"
```



```
#include <openssl/dso.h>

#if !defined(OPENSSL_SYS_BEOS)
DSO_METHOD *DSO_METHOD_beos(void)
        {
        return NULL;
        }
#else

#include <kernel/image.h>

static int beos_load(DSO *dso);
static int beos_unload(DSO *dso);
static void *beos_bind_var(DSO *dso, const char *symname);
static DSO_FUNC_TYPE beos_bind_func(DSO *dso, const char *symname);
#if 0
static int beos_unbind_var(DSO *dso, char *symname, void *symptr);
static int beos_unbind_func(DSO *dso, char *symname, DSO_FUNC_TYPE symptr);
static int beos_init(DSO *dso);
static int beos_finish(DSO *dso);
static long beos_ctrl(DSO *dso, int cmd, long larg, void *parg);
#endif
static char *beos_name_converter(DSO *dso, const char *filename);

static DSO_METHOD dso_meth_beos = {
        "OpenSSL 'beos' shared library method",
        beos_load,
        beos_unload,
        beos_bind_var,
        beos_bind_func,
/* For now, "unbind" doesn't exist */
#if 0
        NULL, /* unbind_var */
        NULL, /* unbind_func */
#endif
        NULL, /* ctrl */
        beos_name_converter,
        NULL, /* init */
        NULL  /* finish */
        };

DSO_METHOD *DSO_METHOD_beos(void)
        {
        return(&dso_meth_beos);
        }

/* For this DSO_METHOD, our meth_data STACK will contain;
 * (i) a pointer to the handle (image_id) returned from
 *     load_add_on().
 */

static int beos_load(DSO *dso)
        {
        image_id id;
        /* See applicable comments from dso_dl.c */
        char *filename = DSO_convert_filename(dso, NULL);

        if(filename == NULL)
                {
                DSOerr(DSO_F_BEOS_LOAD,DSO_R_NO_FILENAME);
                goto err;
                }
        id = load_add_on(filename);
        if(id < 1)
                {
                DSOerr(DSO_F_BEOS_LOAD,DSO_R_LOAD_FAILED);
```

```
128                 ERR_add_error_data(3, "filename(", filename, ")");
129                 goto err;
130                 }
131         if(!sk_push(dso->meth_data, (char *)id))
132                 {
133                 DSOerr(DSO_F_BEOS_LOAD,DSO_R_STACK_ERROR);
134                 goto err;
135                 }
136         /* Success */
137         dso->loaded_filename = filename;
138         return(1);
139 err:
140         /* Cleanup !*/
141         if(filename != NULL)
142                 OPENSSL_free(filename);
143         if(id > 0)
144                 unload_add_on(id);
145         return(0);
146         }

148 static int beos_unload(DSO *dso)
149         {
150         image_id id;
151         if(dso == NULL)
152                 {
153                 DSOerr(DSO_F_BEOS_UNLOAD,ERR_R_PASSED_NULL_PARAMETER);
154                 return(0);
155                 }
156         if(sk_num(dso->meth_data) < 1)
157                 return(1);
158         id = (image_id)sk_pop(dso->meth_data);
159         if(id < 1)
160                 {
161                 DSOerr(DSO_F_BEOS_UNLOAD,DSO_R_NULL_HANDLE);
162                 return(0);
163                 }
164         if(unload_add_on(id) != B_OK)
165                 {
166                 DSOerr(DSO_F_BEOS_UNLOAD,DSO_R_UNLOAD_FAILED);
167                 /* We should push the value back onto the stack in
168                  * case of a retry. */
169                 sk_push(dso->meth_data, (char *)id);
170                 return(0);
171                 }
172         return(1);
173         }

175 static void *beos_bind_var(DSO *dso, const char *symname)
176         {
177         image_id id;
178         void *sym;

180         if((dso == NULL) || (symname == NULL))
181                 {
182                 DSOerr(DSO_F_BEOS_BIND_VAR,ERR_R_PASSED_NULL_PARAMETER);
183                 return(NULL);
184                 }
185         if(sk_num(dso->meth_data) < 1)
186                 {
187                 DSOerr(DSO_F_BEOS_BIND_VAR,DSO_R_STACK_ERROR);
188                 return(NULL);
189                 }
190         id = (image_id)sk_value(dso->meth_data, sk_num(dso->meth_data) - 1);
191         if(id < 1)
192                 {
193                 DSOerr(DSO_F_BEOS_BIND_VAR,DSO_R_NULL_HANDLE);
```



```
194                 return(NULL);
195                 }
196         if(get_image_symbol(id, symname, B_SYMBOL_TYPE_DATA, &sym) != B_OK)
197                 {
198                 DSOerr(DSO_F_BEOS_BIND_VAR,DSO_R_SYM_FAILURE);
199                 ERR_add_error_data(3, "symname(", symname, ")");
200                 return(NULL);
201                 }
202         return(sym);
203         }

205 static DSO_FUNC_TYPE beos_bind_func(DSO *dso, const char *symname)
206         {
207         image_id id;
208         void *sym;

210         if((dso == NULL) || (symname == NULL))
211                 {
212                 DSOerr(DSO_F_BEOS_BIND_FUNC,ERR_R_PASSED_NULL_PARAMETER);
213                 return(NULL);
214                 }
215         if(sk_num(dso->meth_data) < 1)
216                 {
217                 DSOerr(DSO_F_BEOS_BIND_FUNC,DSO_R_STACK_ERROR);
218                 return(NULL);
219                 }
220         id = (image_id)sk_value(dso->meth_data, sk_num(dso->meth_data) - 1);
221         if(id < 1)
222                 {
223                 DSOerr(DSO_F_BEOS_BIND_FUNC,DSO_R_NULL_HANDLE);
224                 return(NULL);
225                 }
226         if(get_image_symbol(id, symname, B_SYMBOL_TYPE_TEXT, &sym) != B_OK)
227                 {
228                 DSOerr(DSO_F_BEOS_BIND_FUNC,DSO_R_SYM_FAILURE);
229                 ERR_add_error_data(3, "symname(", symname, ")");
230                 return(NULL);
231                 }
232         return((DSO_FUNC_TYPE)sym);
233         }

235 /* This one is the same as the one in dlfcn */
236 static char *beos_name_converter(DSO *dso, const char *filename)
237         {
238         char *translated;
239         int len, rsize, transform;

241         len = strlen(filename);
242         rsize = len + 1;
243         transform = (strstr(filename, "/") == NULL);
244         if(transform)
245                 {
246                 /* We will convert this to "%s.so" or "lib%s.so" */
247                 rsize += 3;     /* The length of ".so" */
248                 if ((DSO_flags(dso) & DSO_FLAG_NAME_TRANSLATION_EXT_ONLY) == 0)
249                         rsize += 3; /* The length of "lib" */
250                 }
251         translated = OPENSSL_malloc(rsize);
252         if(translated == NULL)
253                 {
254                 DSOerr(DSO_F_BEOS_NAME_CONVERTER,
255                                 DSO_R_NAME_TRANSLATION_FAILED);
256                 return(NULL);
257                 }
258         if(transform)
259                 {
```

```
260                 if ((DSO_flags(dso) & DSO_FLAG_NAME_TRANSLATION_EXT_ONLY) == 0)
261                         sprintf(translated, "lib%s.so", filename);
262                 else
263                         sprintf(translated, "%s.so", filename);
264                 }
265         else
266                 sprintf(translated, "%s", filename);
267         return(translated);
268         }

270 #endif
271 #endif /* ! codereview */
```

```
**********************************************************
   10947 Wed Aug 13 19:52:34 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_dl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso_dl.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/dso.h>
```



```
#ifndef DSO_DL
DSO_METHOD *DSO_METHOD_dl(void)
        {
        return NULL;
        }
#else

#include <dl.h>

/* Part of the hack in "dl_load" ... */
#define DSO_MAX_TRANSLATED_SIZE 256

static int dl_load(DSO *dso);
static int dl_unload(DSO *dso);
static void *dl_bind_var(DSO *dso, const char *symname);
static DSO_FUNC_TYPE dl_bind_func(DSO *dso, const char *symname);
#if 0
static int dl_unbind_var(DSO *dso, char *symname, void *symptr);
static int dl_unbind_func(DSO *dso, char *symname, DSO_FUNC_TYPE symptr);
static int dl_init(DSO *dso);
static int dl_finish(DSO *dso);
static int dl_ctrl(DSO *dso, int cmd, long larg, void *parg);
#endif
static char *dl_name_converter(DSO *dso, const char *filename);
static char *dl_merger(DSO *dso, const char *filespec1, const char *filespec2);
static int dl_pathbyaddr(void *addr,char *path,int sz);
static void *dl_globallookup(const char *name);

static DSO_METHOD dso_meth_dl = {
        "OpenSSL 'dl' shared library method",
        dl_load,
        dl_unload,
        dl_bind_var,
        dl_bind_func,
/* For now, "unbind" doesn't exist */
#if 0
        NULL, /* unbind_var */
        NULL, /* unbind_func */
#endif
        NULL, /* ctrl */
        dl_name_converter,
        dl_merger,
        NULL, /* init */
        NULL, /* finish */
        dl_pathbyaddr,
        dl_globallookup
        };

DSO_METHOD *DSO_METHOD_dl(void)
        {
        return(&dso_meth_dl);
        }

/* For this DSO_METHOD, our meth_data STACK will contain;
 * (i) the handle (shl_t) returned from shl_load().
 * NB: I checked on HPUX11 and shl_t is itself a pointer
 * type so the cast is safe.
 */

static int dl_load(DSO *dso)
        {
        shl_t ptr = NULL;
        /* We don't do any fancy retries or anything, just take the method's
         * (or DSO's if it has the callback set) best translation of the
         * platform-independant filename and try once with that. */
```

```
128         char *filename= DSO_convert_filename(dso, NULL);

130         if(filename == NULL)
131                 {
132                 DSOerr(DSO_F_DL_LOAD,DSO_R_NO_FILENAME);
133                 goto err;
134                 }
135         ptr = shl_load(filename, BIND_IMMEDIATE |
136                 (dso->flags&DSO_FLAG_NO_NAME_TRANSLATION?0:DYNAMIC_PATH), 0L);
137         if(ptr == NULL)
138                 {
139                 DSOerr(DSO_F_DL_LOAD,DSO_R_LOAD_FAILED);
140                 ERR_add_error_data(4, "filename(", filename, "): ",
141                         strerror(errno));
142                 goto err;
143                 }
144         if(!sk_push(dso->meth_data, (char *)ptr))
145                 {
146                 DSOerr(DSO_F_DL_LOAD,DSO_R_STACK_ERROR);
147                 goto err;
148                 }
149         /* Success, stick the converted filename we've loaded under into the DSO
150          * (it also serves as the indicator that we are currently loaded). */
151         dso->loaded_filename = filename;
152         return(1);
153 err:
154         /* Cleanup! */
155         if(filename != NULL)
156                 OPENSSL_free(filename);
157         if(ptr != NULL)
158                 shl_unload(ptr);
159         return(0);
160         }

162 static int dl_unload(DSO *dso)
163         {
164         shl_t ptr;
165         if(dso == NULL)
166                 {
167                 DSOerr(DSO_F_DL_UNLOAD,ERR_R_PASSED_NULL_PARAMETER);
168                 return(0);
169                 }
170         if(sk_num(dso->meth_data) < 1)
171                 return(1);
172         /* Is this statement legal? */
173         ptr = (shl_t)sk_pop(dso->meth_data);
174         if(ptr == NULL)
175                 {
176                 DSOerr(DSO_F_DL_UNLOAD,DSO_R_NULL_HANDLE);
177                 /* Should push the value back onto the stack in
178                  * case of a retry. */
179                 sk_push(dso->meth_data, (char *)ptr);
180                 return(0);
181                 }
182         shl_unload(ptr);
183         return(1);
184         }

186 static void *dl_bind_var(DSO *dso, const char *symname)
187         {
188         shl_t ptr;
189         void *sym;

191         if((dso == NULL) || (symname == NULL))
192                 {
193                 DSOerr(DSO_F_DL_BIND_VAR,ERR_R_PASSED_NULL_PARAMETER);
```



```
194                 return(NULL);
195                 }
196         if(sk_num(dso->meth_data) < 1)
197                 {
198                 DSOerr(DSO_F_DL_BIND_VAR,DSO_R_STACK_ERROR);
199                 return(NULL);
200                 }
201         ptr = (shl_t)sk_value(dso->meth_data, sk_num(dso->meth_data) - 1);
202         if(ptr == NULL)
203                 {
204                 DSOerr(DSO_F_DL_BIND_VAR,DSO_R_NULL_HANDLE);
205                 return(NULL);
206                 }
207         if (shl_findsym(&ptr, symname, TYPE_UNDEFINED, &sym) < 0)
208                 {
209                 DSOerr(DSO_F_DL_BIND_VAR,DSO_R_SYM_FAILURE);
210                 ERR_add_error_data(4, "symname(", symname, "): ",
211                         strerror(errno));
212                 return(NULL);
213                 }
214         return(sym);
215         }

217 static DSO_FUNC_TYPE dl_bind_func(DSO *dso, const char *symname)
218         {
219         shl_t ptr;
220         void *sym;

222         if((dso == NULL) || (symname == NULL))
223                 {
224                 DSOerr(DSO_F_DL_BIND_FUNC,ERR_R_PASSED_NULL_PARAMETER);
225                 return(NULL);
226                 }
227         if(sk_num(dso->meth_data) < 1)
228                 {
229                 DSOerr(DSO_F_DL_BIND_FUNC,DSO_R_STACK_ERROR);
230                 return(NULL);
231                 }
232         ptr = (shl_t)sk_value(dso->meth_data, sk_num(dso->meth_data) - 1);
233         if(ptr == NULL)
234                 {
235                 DSOerr(DSO_F_DL_BIND_FUNC,DSO_R_NULL_HANDLE);
236                 return(NULL);
237                 }
238         if (shl_findsym(&ptr, symname, TYPE_UNDEFINED, &sym) < 0)
239                 {
240                 DSOerr(DSO_F_DL_BIND_FUNC,DSO_R_SYM_FAILURE);
241                 ERR_add_error_data(4, "symname(", symname, "): ",
242                         strerror(errno));
243                 return(NULL);
244                 }
245         return((DSO_FUNC_TYPE)sym);
246         }

248 static char *dl_merger(DSO *dso, const char *filespec1, const char *filespec2)
249         {
250         char *merged;

252         if(!filespec1 && !filespec2)
253                 {
254                 DSOerr(DSO_F_DL_MERGER,
255                                 ERR_R_PASSED_NULL_PARAMETER);
256                 return(NULL);
257                 }
258         /* If the first file specification is a rooted path, it rules.
259            same goes if the second file specification is missing. */
```

```
260         if (!filespec2 || filespec1[0] == '/')
261                 {
262                 merged = OPENSSL_malloc(strlen(filespec1) + 1);
263                 if(!merged)
264                         {
265                         DSOerr(DSO_F_DL_MERGER,
266                                 ERR_R_MALLOC_FAILURE);
267                         return(NULL);
268                         }
269                 strcpy(merged, filespec1);
270                 }
271         /* If the first file specification is missing, the second one rules. */
272         else if (!filespec1)
273                 {
274                 merged = OPENSSL_malloc(strlen(filespec2) + 1);
275                 if(!merged)
276                         {
277                         DSOerr(DSO_F_DL_MERGER,
278                                 ERR_R_MALLOC_FAILURE);
279                         return(NULL);
280                         }
281                 strcpy(merged, filespec2);
282                 }
283         else
284                 /* This part isn't as trivial as it looks.  It assumes that
285                    the second file specification really is a directory, and
286                    makes no checks whatsoever.  Therefore, the result becomes
287                    the concatenation of filespec2 followed by a slash followed
288                    by filespec1. */
289                 {
290                 int spec2len, len;

292                 spec2len = (filespec2 ? strlen(filespec2) : 0);
293                 len = spec2len + (filespec1 ? strlen(filespec1) : 0);

295                 if(filespec2 && filespec2[spec2len - 1] == '/')
296                         {
297                         spec2len--;
298                         len--;
299                         }
300                 merged = OPENSSL_malloc(len + 2);
301                 if(!merged)
302                         {
303                         DSOerr(DSO_F_DL_MERGER,
304                                 ERR_R_MALLOC_FAILURE);
305                         return(NULL);
306                         }
307                 strcpy(merged, filespec2);
308                 merged[spec2len] = '/';
309                 strcpy(&merged[spec2len + 1], filespec1);
310                 }
311         return(merged);
312         }

314 /* This function is identical to the one in dso_dlfcn.c, but as it is highly
315  * unlikely that both the "dl" *and* "dlfcn" variants are being compiled at the
316  * same time, there's no great duplicating the code. Figuring out an elegant
317  * way to share one copy of the code would be more difficult and would not
318  * leave the implementations independant. */
319 #if defined(__hpux)
320 static const char extension[] = ".sl";
321 #else
322 static const char extension[] = ".so";
323 #endif
324 static char *dl_name_converter(DSO *dso, const char *filename)
325         {
```



```
326         char *translated;
327         int len, rsize, transform;

329         len = strlen(filename);
330         rsize = len + 1;
331         transform = (strstr(filename, "/") == NULL);
332                 {
333                 /* We will convert this to "%s.s?" or "lib%s.s?" */
334                 rsize += strlen(extension);/* The length of ".s?" */
335                 if ((DSO_flags(dso) & DSO_FLAG_NAME_TRANSLATION_EXT_ONLY) == 0)
336                         rsize += 3; /* The length of "lib" */
337                 }
338         translated = OPENSSL_malloc(rsize);
339         if(translated == NULL)
340                 {
341                 DSOerr(DSO_F_DL_NAME_CONVERTER,
342                                 DSO_R_NAME_TRANSLATION_FAILED);
343                 return(NULL);
344                 }
345         if(transform)
346                 {
347                 if ((DSO_flags(dso) & DSO_FLAG_NAME_TRANSLATION_EXT_ONLY) == 0)
348                         sprintf(translated, "lib%s%s", filename, extension);
349                 else
350                         sprintf(translated, "%s%s", filename, extension);
351                 }
352         else
353                 sprintf(translated, "%s", filename);
354         return(translated);
355         }

357 static int dl_pathbyaddr(void *addr,char *path,int sz)
358         {
359         struct shl_descriptor inf;
360         int i,len;

362         if (addr == NULL)
363                 {
364                 union   { int(*f)(void*,char*,int); void *p; } t =
365                         { dl_pathbyaddr };
366                 addr = t.p;
367                 }

369         for (i=-1;shl_get_r(i,&inf)==0;i++)
370                 {
371                 if (((size_t)addr >= inf.tstart && (size_t)addr < inf.tend) ||
372                     ((size_t)addr >= inf.dstart && (size_t)addr < inf.dend))
373                         {
374                         len = (int)strlen(inf.filename);
375                         if (sz <= 0) return len+1;
376                         if (len >= sz) len=sz-1;
377                         memcpy(path,inf.filename,len);
378                         path[len++] = 0;
379                         return len;
380                         }
381                 }

383         return -1;
384         }

386 static void *dl_globallookup(const char *name)
387         {
388         void *ret;
389         shl_t h = NULL;

391         return shl_findsym(&h,name,TYPE_UNDEFINED,&ret) ? NULL : ret;
```

```
392         }
393 #endif /* DSO_DL */
394 #endif /* ! codereview */
```

```
**********************************************************
   12982 Wed Aug 13 19:52:34 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_dlfcn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso_dlfcn.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* We need to do this early, because stdio.h includes the header files
   that handle _GNU_SOURCE and other similar macros.  Defining it later
   is simply too late, because those headers are protected from re-
```



```
   inclusion.  */
#ifdef __linux
# ifndef _GNU_SOURCE
#  define _GNU_SOURCE   /* make sure dladdr is declared */
# endif
#endif

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/dso.h>

#ifndef DSO_DLFCN
DSO_METHOD *DSO_METHOD_dlfcn(void)
        {
        return NULL;
        }
#else

#ifdef HAVE_DLFCN_H
# ifdef __osf__
#  define __EXTENSIONS__
# endif
# include <dlfcn.h>
# define HAVE_DLINFO 1
# if defined(_AIX) || defined(__CYGWIN__) || \
     defined(__SCO_VERSION__) || defined(_SCO_ELF) || \
     (defined(__osf__) && !defined(RTLD_NEXT))     || \
     (defined(__OpenBSD__) && !defined(RTLD_SELF)) || \
        defined(__ANDROID__)
#  undef HAVE_DLINFO
# endif
#endif

/* Part of the hack in "dlfcn_load" ... */
#define DSO_MAX_TRANSLATED_SIZE 256

static int dlfcn_load(DSO *dso);
static int dlfcn_unload(DSO *dso);
static void *dlfcn_bind_var(DSO *dso, const char *symname);
static DSO_FUNC_TYPE dlfcn_bind_func(DSO *dso, const char *symname);
#if 0
static int dlfcn_unbind(DSO *dso, char *symname, void *symptr);
static int dlfcn_init(DSO *dso);
static int dlfcn_finish(DSO *dso);
static long dlfcn_ctrl(DSO *dso, int cmd, long larg, void *parg);
#endif
static char *dlfcn_name_converter(DSO *dso, const char *filename);
static char *dlfcn_merger(DSO *dso, const char *filespec1,
        const char *filespec2);
static int dlfcn_pathbyaddr(void *addr,char *path,int sz);
static void *dlfcn_globallookup(const char *name);

static DSO_METHOD dso_meth_dlfcn = {
        "OpenSSL 'dlfcn' shared library method",
        dlfcn_load,
        dlfcn_unload,
        dlfcn_bind_var,
        dlfcn_bind_func,
/* For now, "unbind" doesn't exist */
#if 0
        NULL, /* unbind_var */
        NULL, /* unbind_func */
#endif
        NULL, /* ctrl */
        dlfcn_name_converter,
        dlfcn_merger,
```

```
128         NULL, /* init */
129         NULL, /* finish */
130         dlfcn_pathbyaddr,
131         dlfcn_globallookup
132         };

133 DSO_METHOD *DSO_METHOD_dlfcn(void)
134         {
135         return(&dso_meth_dlfcn);
136         }

137 /* Prior to using the dlopen() function, we should decide on the flag
138  * we send. There's a few different ways of doing this and it's a
139  * messy venn-diagram to match up which platforms support what. So
140  * as we don't have autoconf yet, I'm implementing a hack that could
141  * be hacked further relatively easily to deal with cases as we find
142  * them. Initially this is to cope with OpenBSD. */
143 #if defined(__OpenBSD__) || defined(__NetBSD__)
144 #       ifdef DL_LAZY
145 #               define DLOPEN_FLAG DL_LAZY
146 #       else
147 #               ifdef RTLD_NOW
148 #                       define DLOPEN_FLAG RTLD_NOW
149 #               else
150 #                       define DLOPEN_FLAG 0
151 #               endif
152 #       endif
153 #else
154 #       ifdef OPENSSL_SYS_SUNOS
155 #               define DLOPEN_FLAG 1
156 #       else
157 #               define DLOPEN_FLAG RTLD_NOW /* Hope this works everywhere else *
158 #       endif
159 #endif

160 /* For this DSO_METHOD, our meth_data STACK will contain;
161  * (i) the handle (void*) returned from dlopen().
162  */

163 static int dlfcn_load(DSO *dso)
164         {
165         void *ptr = NULL;
166         /* See applicable comments in dso_dl.c */
167         char *filename = DSO_convert_filename(dso, NULL);
168         int flags = DLOPEN_FLAG;

169         if(filename == NULL)
170                 {
171                 DSOerr(DSO_F_DLFCN_LOAD,DSO_R_NO_FILENAME);
172                 goto err;
173                 }

174 #ifdef RTLD_GLOBAL
175         if (dso->flags & DSO_FLAG_GLOBAL_SYMBOLS)
176                 flags |= RTLD_GLOBAL;
177 #endif
178         ptr = dlopen(filename, flags);
179         if(ptr == NULL)
180                 {
181                 DSOerr(DSO_F_DLFCN_LOAD,DSO_R_LOAD_FAILED);
182                 ERR_add_error_data(4, "filename(", filename, "): ", dlerror());
183                 goto err;
184                 }
185         if(!sk_void_push(dso->meth_data, (char *)ptr))
186                 {
187                 DSOerr(DSO_F_DLFCN_LOAD,DSO_R_STACK_ERROR);
```



```
188                 goto err;
189                 }
190         /* Success */
191         dso->loaded_filename = filename;
192         return(1);
193 err:
194         /* Cleanup! */
195         if(filename != NULL)
196                 OPENSSL_free(filename);
197         if(ptr != NULL)
198                 dlclose(ptr);
199         return(0);
200 }

201 static int dlfcn_unload(DSO *dso)
202         {
203         void *ptr;
204         if(dso == NULL)
205                 {
206                 DSOerr(DSO_F_DLFCN_UNLOAD,ERR_R_PASSED_NULL_PARAMETER);
207                 return(0);
208                 }
209         if(sk_void_num(dso->meth_data) < 1)
210                 return(1);
211         ptr = sk_void_pop(dso->meth_data);
212         if(ptr == NULL)
213                 {
214                 DSOerr(DSO_F_DLFCN_UNLOAD,DSO_R_NULL_HANDLE);
215                 /* Should push the value back onto the stack in
216                  * case of a retry. */
217                 sk_void_push(dso->meth_data, ptr);
218                 return(0);
219                 }
220         /* For now I'm not aware of any errors associated with dlclose() */
221         dlclose(ptr);
222         return(1);
223         }

224 static void *dlfcn_bind_var(DSO *dso, const char *symname)
225         {
226         void *ptr, *sym;

227         if((dso == NULL) || (symname == NULL))
228                 {
229                 DSOerr(DSO_F_DLFCN_BIND_VAR,ERR_R_PASSED_NULL_PARAMETER);
230                 return(NULL);
231                 }
232         if(sk_void_num(dso->meth_data) < 1)
233                 {
234                 DSOerr(DSO_F_DLFCN_BIND_VAR,DSO_R_STACK_ERROR);
235                 return(NULL);
236                 }
237         ptr = sk_void_value(dso->meth_data, sk_void_num(dso->meth_data) - 1);
238         if(ptr == NULL)
239                 {
240                 DSOerr(DSO_F_DLFCN_BIND_VAR,DSO_R_NULL_HANDLE);
241                 return(NULL);
242                 }
243         sym = dlsym(ptr, symname);
244         if(sym == NULL)
245                 {
246                 DSOerr(DSO_F_DLFCN_BIND_VAR,DSO_R_SYM_FAILURE);
247                 ERR_add_error_data(4, "symname(", symname, "): ", dlerror());
248                 return(NULL);
249                 }
250         return(sym);
```

```
260         }

262 static DSO_FUNC_TYPE dlfcn_bind_func(DSO *dso, const char *symname)
263         {
264         void *ptr;
265         union {
266                 DSO_FUNC_TYPE sym;
267                 void *dlret;
268         } u;

270         if((dso == NULL) || (symname == NULL))
271                 {
272                 DSOerr(DSO_F_DLFCN_BIND_FUNC,ERR_R_PASSED_NULL_PARAMETER);
273                 return(NULL);
274                 }
275         if(sk_void_num(dso->meth_data) < 1)
276                 {
277                 DSOerr(DSO_F_DLFCN_BIND_FUNC,DSO_R_STACK_ERROR);
278                 return(NULL);
279                 }
280         ptr = sk_void_value(dso->meth_data, sk_void_num(dso->meth_data) - 1);
281         if(ptr == NULL)
282                 {
283                 DSOerr(DSO_F_DLFCN_BIND_FUNC,DSO_R_NULL_HANDLE);
284                 return(NULL);
285                 }
286         u.dlret = dlsym(ptr, symname);
287         if(u.dlret == NULL)
288                 {
289                 DSOerr(DSO_F_DLFCN_BIND_FUNC,DSO_R_SYM_FAILURE);
290                 ERR_add_error_data(4, "symname(", symname, "): ", dlerror());
291                 return(NULL);
292                 }
293         return u.sym;
294         }

296 static char *dlfcn_merger(DSO *dso, const char *filespec1,
297         const char *filespec2)
298         {
299         char *merged;

301         if(!filespec1 && !filespec2)
302                 {
303                 DSOerr(DSO_F_DLFCN_MERGER,
304                                 ERR_R_PASSED_NULL_PARAMETER);
305                 return(NULL);
306                 }
307         /* If the first file specification is a rooted path, it rules.
308            same goes if the second file specification is missing. */
309         if (!filespec2 || (filespec1 != NULL && filespec1[0] == '/'))
310                 {
311                 merged = OPENSSL_malloc(strlen(filespec1) + 1);
312                 if(!merged)
313                         {
314                         DSOerr(DSO_F_DLFCN_MERGER, ERR_R_MALLOC_FAILURE);
315                         return(NULL);
316                         }
317                 strcpy(merged, filespec1);
318                 }
319         /* If the first file specification is missing, the second one rules. */
320         else if (!filespec1)
321                 {
322                 merged = OPENSSL_malloc(strlen(filespec2) + 1);
323                 if(!merged)
324                         {
325                         DSOerr(DSO_F_DLFCN_MERGER,
```



```
326                                 ERR_R_MALLOC_FAILURE);
327                         return(NULL);
328                         }
329                 strcpy(merged, filespec2);
330                 }
331         else
332                 /* This part isn't as trivial as it looks.  It assumes that
333                    the second file specification really is a directory, and
334                    makes no checks whatsoever.  Therefore, the result becomes
335                    the concatenation of filespec2 followed by a slash followed
336                    by filespec1. */
337                 {
338                 int spec2len, len;

340                 spec2len = strlen(filespec2);
341                 len = spec2len + (filespec1 ? strlen(filespec1) : 0);

343                 if(filespec2 && filespec2[spec2len - 1] == '/')
344                         {
345                         spec2len--;
346                         len--;
347                         }
348                 merged = OPENSSL_malloc(len + 2);
349                 if(!merged)
350                         {
351                         DSOerr(DSO_F_DLFCN_MERGER,
352                                 ERR_R_MALLOC_FAILURE);
353                         return(NULL);
354                         }
355                 strcpy(merged, filespec2);
356                 merged[spec2len] = '/';
357                 strcpy(&merged[spec2len + 1], filespec1);
358                 }
359         return(merged);
360         }

362 #ifdef OPENSSL_SYS_MACOSX
363 #define DSO_ext ".dylib"
364 #define DSO_extlen 6
365 #else
366 #define DSO_ext ".so"
367 #define DSO_extlen 3
368 #endif


371 static char *dlfcn_name_converter(DSO *dso, const char *filename)
372         {
373         char *translated;
374         int len, rsize, transform;

376         len = strlen(filename);
377         rsize = len + 1;
378         transform = (strstr(filename, "/") == NULL);
379         if(transform)
380                 {
381                 /* We will convert this to "%s.so" or "lib%s.so" etc */
382                 rsize += DSO_extlen;    /* The length of ".so" */
383                 if ((DSO_flags(dso) & DSO_FLAG_NAME_TRANSLATION_EXT_ONLY) == 0)
384                         rsize += 3; /* The length of "lib" */
385                 }
386         translated = OPENSSL_malloc(rsize);
387         if(translated == NULL)
388                 {
389                 DSOerr(DSO_F_DLFCN_NAME_CONVERTER,
390                                 DSO_R_NAME_TRANSLATION_FAILED);
391                 return(NULL);
```

```
392                 }
393         if(transform)
394                 {
395                 if ((DSO_flags(dso) & DSO_FLAG_NAME_TRANSLATION_EXT_ONLY) == 0)
396                         sprintf(translated, "lib%s" DSO_ext, filename);
397                 else
398                         sprintf(translated, "%s" DSO_ext, filename);
399                 }
400         else
401                 sprintf(translated, "%s", filename);
402         return(translated);
403         }

405 #ifdef __sgi
406 /*
407 This is a quote from IRIX manual for dladdr(3c):

409      <dlfcn.h> does not contain a prototype for dladdr or definition of
410      Dl_info.  The #include <dlfcn.h>  in the SYNOPSIS line is traditional,
411      but contains no dladdr prototype and no IRIX library contains an
412      implementation.  Write your own declaration based on the code below.

414      The following code is dependent on internal interfaces that are not
415      part of the IRIX compatibility guarantee; however, there is no future
416      intention to change this interface, so on a practical level, the code
417      below is safe to use on IRIX.
418 */
419 #include <rld_interface.h>
420 #ifndef _RLD_INTERFACE_DLFCN_H_DLADDR
421 #define _RLD_INTERFACE_DLFCN_H_DLADDR
422 typedef struct Dl_info {
423     const char * dli_fname;
424     void       * dli_fbase;
425     const char * dli_sname;
426     void       * dli_saddr;
427     int          dli_version;
428     int          dli_reserved1;
429     long         dli_reserved[4];
430 } Dl_info;
431 #else
432 typedef struct Dl_info Dl_info;
433 #endif
434 #define _RLD_DLADDR             14

436 static int dladdr(void *address, Dl_info *dl)
437 {
438         void *v;
439         v = _rld_new_interface(_RLD_DLADDR,address,dl);
440         return (int)v;
441 }
442 #endif /* __sgi */

444 static int dlfcn_pathbyaddr(void *addr,char *path,int sz)
445         {
446 #ifdef HAVE_DLINFO
447         Dl_info dli;
448         int len;

450         if (addr == NULL)
451                 {
452                 union   { int(*f)(void*,char*,int); void *p; } t =
453                         { dlfcn_pathbyaddr };
454                 addr = t.p;
455                 }

457         if (dladdr(addr,&dli))
```



```
458                 {
459                 len = (int)strlen(dli.dli_fname);
460                 if (sz <= 0) return len+1;
461                 if (len >= sz) len=sz-1;
462                 memcpy(path,dli.dli_fname,len);
463                 path[len++]=0;
464                 return len;
465                 }

467         ERR_add_error_data(2, "dlfcn_pathbyaddr(): ", dlerror());
468 #endif
469         return -1;
470         }

472 static void *dlfcn_globallookup(const char *name)
473         {
474         void *ret = NULL,*handle = dlopen(NULL,RTLD_LAZY);

476         if (handle)
477                 {
478                 ret = dlsym(handle,name);
479                 dlclose(handle);
480                 }

482         return ret;
483         }
484 #endif /* DSO_DLFCN */
485 #endif /* ! codereview */
```

```
**********************************************************
    7130 Wed Aug 13 19:52:34 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/dso/dso_err.c */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/dso.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_DSO,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_DSO,0,reason)

static ERR_STRING_DATA DSO_str_functs[]=
        {
{ERR_FUNC(DSO_F_BEOS_BIND_FUNC),        "BEOS_BIND_FUNC"},
{ERR_FUNC(DSO_F_BEOS_BIND_VAR), "BEOS_BIND_VAR"},
{ERR_FUNC(DSO_F_BEOS_LOAD),     "BEOS_LOAD"},
{ERR_FUNC(DSO_F_BEOS_NAME_CONVERTER),   "BEOS_NAME_CONVERTER"},
{ERR_FUNC(DSO_F_BEOS_UNLOAD),   "BEOS_UNLOAD"},
{ERR_FUNC(DSO_F_DLFCN_BIND_FUNC),       "DLFCN_BIND_FUNC"},
{ERR_FUNC(DSO_F_DLFCN_BIND_VAR),        "DLFCN_BIND_VAR"},
{ERR_FUNC(DSO_F_DLFCN_LOAD),    "DLFCN_LOAD"},
{ERR_FUNC(DSO_F_DLFCN_MERGER),  "DLFCN_MERGER"},
{ERR_FUNC(DSO_F_DLFCN_NAME_CONVERTER),  "DLFCN_NAME_CONVERTER"},
{ERR_FUNC(DSO_F_DLFCN_UNLOAD),  "DLFCN_UNLOAD"},
{ERR_FUNC(DSO_F_DL_BIND_FUNC),  "DL_BIND_FUNC"},
{ERR_FUNC(DSO_F_DL_BIND_VAR),   "DL_BIND_VAR"},
{ERR_FUNC(DSO_F_DL_LOAD),       "DL_LOAD"},
{ERR_FUNC(DSO_F_DL_MERGER),     "DL_MERGER"},
{ERR_FUNC(DSO_F_DL_NAME_CONVERTER),     "DL_NAME_CONVERTER"},
{ERR_FUNC(DSO_F_DL_UNLOAD),     "DL_UNLOAD"},
{ERR_FUNC(DSO_F_DSO_BIND_FUNC), "DSO_bind_func"},
{ERR_FUNC(DSO_F_DSO_BIND_VAR),  "DSO_bind_var"},
{ERR_FUNC(DSO_F_DSO_CONVERT_FILENAME),  "DSO_convert_filename"},
{ERR_FUNC(DSO_F_DSO_CTRL),      "DSO_ctrl"},
{ERR_FUNC(DSO_F_DSO_FREE),      "DSO_free"},
{ERR_FUNC(DSO_F_DSO_GET_FILENAME),      "DSO_get_filename"},
{ERR_FUNC(DSO_F_DSO_GET_LOADED_FILENAME),       "DSO_get_loaded_filename"},
{ERR_FUNC(DSO_F_DSO_GLOBAL_LOOKUP),     "DSO_global_lookup"},
{ERR_FUNC(DSO_F_DSO_LOAD),      "DSO_load"},
{ERR_FUNC(DSO_F_DSO_MERGE),     "DSO_merge"},
{ERR_FUNC(DSO_F_DSO_NEW_METHOD),        "DSO_new_method"},
{ERR_FUNC(DSO_F_DSO_PATHBYADDR),        "DSO_pathbyaddr"},
{ERR_FUNC(DSO_F_DSO_SET_FILENAME),      "DSO_set_filename"},
{ERR_FUNC(DSO_F_DSO_SET_NAME_CONVERTER),        "DSO_set_name_converter"},
{ERR_FUNC(DSO_F_DSO_UP_REF),    "DSO_up_ref"},
{ERR_FUNC(DSO_F_GLOBAL_LOOKUP_FUNC),    "GLOBAL_LOOKUP_FUNC"},
{ERR_FUNC(DSO_F_PATHBYADDR),    "PATHBYADDR"},
{ERR_FUNC(DSO_F_VMS_BIND_SYM),  "VMS_BIND_SYM"},
{ERR_FUNC(DSO_F_VMS_LOAD),      "VMS_LOAD"},
{ERR_FUNC(DSO_F_VMS_MERGER),    "VMS_MERGER"},
{ERR_FUNC(DSO_F_VMS_UNLOAD),    "VMS_UNLOAD"},
{ERR_FUNC(DSO_F_WIN32_BIND_FUNC),       "WIN32_BIND_FUNC"},
{ERR_FUNC(DSO_F_WIN32_BIND_VAR),        "WIN32_BIND_VAR"},
{ERR_FUNC(DSO_F_WIN32_GLOBALLOOKUP),    "WIN32_GLOBALLOOKUP"},
{ERR_FUNC(DSO_F_WIN32_GLOBALLOOKUP_FUNC),       "WIN32_GLOBALLOOKUP_FUNC"},
{ERR_FUNC(DSO_F_WIN32_JOINER),  "WIN32_JOINER"},
{ERR_FUNC(DSO_F_WIN32_LOAD),    "WIN32_LOAD"},
{ERR_FUNC(DSO_F_WIN32_MERGER),  "WIN32_MERGER"},
{ERR_FUNC(DSO_F_WIN32_NAME_CONVERTER),  "WIN32_NAME_CONVERTER"},
{ERR_FUNC(DSO_F_WIN32_PATHBYADDR),      "WIN32_PATHBYADDR"},
{ERR_FUNC(DSO_F_WIN32_SPLITTER),        "WIN32_SPLITTER"},
{ERR_FUNC(DSO_F_WIN32_UNLOAD),  "WIN32_UNLOAD"},
{0,NULL}
        };

static ERR_STRING_DATA DSO_str_reasons[]=
        {
{ERR_REASON(DSO_R_CTRL_FAILED)           ,"control command failed"},
```

```
128 {ERR_REASON(DSO_R_DSO_ALREADY_LOADED)    ,"dso already loaded"},
129 {ERR_REASON(DSO_R_EMPTY_FILE_STRUCTURE)  ,"empty file structure"},
130 {ERR_REASON(DSO_R_FAILURE)                ,"failure"},
131 {ERR_REASON(DSO_R_FILENAME_TOO_BIG)       ,"filename too big"},
132 {ERR_REASON(DSO_R_FINISH_FAILED)          ,"cleanup method function failed"},
133 {ERR_REASON(DSO_R_INCORRECT_FILE_SYNTAX) ,"incorrect file syntax"},
134 {ERR_REASON(DSO_R_LOAD_FAILED)            ,"could not load the shared library"},
135 {ERR_REASON(DSO_R_NAME_TRANSLATION_FAILED),"name translation failed"},
136 {ERR_REASON(DSO_R_NO_FILENAME)            ,"no filename"},
137 {ERR_REASON(DSO_R_NO_FILE_SPECIFICATION) ,"no file specification"},
138 {ERR_REASON(DSO_R_NULL_HANDLE)            ,"a null shared library handle was used
139 {ERR_REASON(DSO_R_SET_FILENAME_FAILED)    ,"set filename failed"},
140 {ERR_REASON(DSO_R_STACK_ERROR)            ,"the meth_data stack is corrupt"},
141 {ERR_REASON(DSO_R_SYM_FAILURE)            ,"could not bind to the requested symbo
142 {ERR_REASON(DSO_R_UNLOAD_FAILED)          ,"could not unload the shared library"}
143 {ERR_REASON(DSO_R_UNSUPPORTED)            ,"functionality not supported"},
144 {0,NULL}
145         };

147 #endif

149 void ERR_load_DSO_strings(void)
150         {
151 #ifndef OPENSSL_NO_ERR

153         if (ERR_func_error_string(DSO_str_functs[0].error) == NULL)
154                 {
155                 ERR_load_strings(0,DSO_str_functs);
156                 ERR_load_strings(0,DSO_str_reasons);
157                 }
158 #endif
159         }
160 #endif /* ! codereview */
```

```
**********************************************************
   11916 Wed Aug 13 19:52:34 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso_lib.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/crypto.h>
#include "cryptlib.h"
```



```
#include <openssl/dso.h>

static DSO_METHOD *default_DSO_meth = NULL;

DSO *DSO_new(void)
        {
        return(DSO_new_method(NULL));
        }

void DSO_set_default_method(DSO_METHOD *meth)
        {
        default_DSO_meth = meth;
        }

DSO_METHOD *DSO_get_default_method(void)
        {
        return(default_DSO_meth);
        }

DSO_METHOD *DSO_get_method(DSO *dso)
        {
        return(dso->meth);
        }

DSO_METHOD *DSO_set_method(DSO *dso, DSO_METHOD *meth)
        {
        DSO_METHOD *mtmp;
        mtmp = dso->meth;
        dso->meth = meth;
        return(mtmp);
        }

DSO *DSO_new_method(DSO_METHOD *meth)
        {
        DSO *ret;

        if(default_DSO_meth == NULL)
                /* We default to DSO_METH_openssl() which in turn defaults
                 * to stealing the "best available" method. Will fallback
                 * to DSO_METH_null() in the worst case. */
                default_DSO_meth = DSO_METHOD_openssl();
        ret = (DSO *)OPENSSL_malloc(sizeof(DSO));
        if(ret == NULL)
                {
                DSOerr(DSO_F_DSO_NEW_METHOD,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }
        memset(ret, 0, sizeof(DSO));
        ret->meth_data = sk_void_new_null();
        if(ret->meth_data == NULL)
                {
                /* sk_new doesn't generate any errors so we do */
                DSOerr(DSO_F_DSO_NEW_METHOD,ERR_R_MALLOC_FAILURE);
                OPENSSL_free(ret);
                return(NULL);
                }
        if(meth == NULL)
                ret->meth = default_DSO_meth;
        else
                ret->meth = meth;
        ret->references = 1;
        if((ret->meth->init != NULL) && !ret->meth->init(ret))
                {
                OPENSSL_free(ret);
                ret=NULL;
                }
```

```
128         return(ret);
129         }

131 int DSO_free(DSO *dso)
132         {
133         int i;

135         if(dso == NULL)
136                 {
137                 DSOerr(DSO_F_DSO_FREE,ERR_R_PASSED_NULL_PARAMETER);
138                 return(0);
139                 }

141         i=CRYPTO_add(&dso->references,-1,CRYPTO_LOCK_DSO);
142 #ifdef REF_PRINT
143         REF_PRINT("DSO",dso);
144 #endif
145         if(i > 0) return(1);
146 #ifdef REF_CHECK
147         if(i < 0)
148                 {
149                 fprintf(stderr,"DSO_free, bad reference count\n");
150                 abort();
151                 }
152 #endif

154         if((dso->meth->dso_unload != NULL) && !dso->meth->dso_unload(dso))
155                 {
156                 DSOerr(DSO_F_DSO_FREE,DSO_R_UNLOAD_FAILED);
157                 return(0);
158                 }

160         if((dso->meth->finish != NULL) && !dso->meth->finish(dso))
161                 {
162                 DSOerr(DSO_F_DSO_FREE,DSO_R_FINISH_FAILED);
163                 return(0);
164                 }

166         sk_void_free(dso->meth_data);
167         if(dso->filename != NULL)
168                 OPENSSL_free(dso->filename);
169         if(dso->loaded_filename != NULL)
170                 OPENSSL_free(dso->loaded_filename);

172         OPENSSL_free(dso);
173         return(1);
174         }

176 int DSO_flags(DSO *dso)
177         {
178         return((dso == NULL) ? 0 : dso->flags);
179         }


182 int DSO_up_ref(DSO *dso)
183         {
184         if (dso == NULL)
185                 {
186                 DSOerr(DSO_F_DSO_UP_REF,ERR_R_PASSED_NULL_PARAMETER);
187                 return(0);
188                 }

190         CRYPTO_add(&dso->references,1,CRYPTO_LOCK_DSO);
191         return(1);
192         }
```



```
194 DSO *DSO_load(DSO *dso, const char *filename, DSO_METHOD *meth, int flags)
195         {
196         DSO *ret;
197         int allocated = 0;

199         if(dso == NULL)
200                 {
201                 ret = DSO_new_method(meth);
202                 if(ret == NULL)
203                         {
204                         DSOerr(DSO_F_DSO_LOAD,ERR_R_MALLOC_FAILURE);
205                         goto err;
206                         }
207                 allocated = 1;
208                 /* Pass the provided flags to the new DSO object */
209                 if(DSO_ctrl(ret, DSO_CTRL_SET_FLAGS, flags, NULL) < 0)
210                         {
211                         DSOerr(DSO_F_DSO_LOAD,DSO_R_CTRL_FAILED);
212                         goto err;
213                         }
214                 }
215         else
216                 ret = dso;
217         /* Don't load if we're currently already loaded */
218         if(ret->filename != NULL)
219                 {
220                 DSOerr(DSO_F_DSO_LOAD,DSO_R_DSO_ALREADY_LOADED);
221                 goto err;
222                 }
223         /* filename can only be NULL if we were passed a dso that already has
224          * one set. */
225         if(filename != NULL)
226                 if(!DSO_set_filename(ret, filename))
227                         {
228                         DSOerr(DSO_F_DSO_LOAD,DSO_R_SET_FILENAME_FAILED);
229                         goto err;
230                         }
231         filename = ret->filename;
232         if(filename == NULL)
233                 {
234                 DSOerr(DSO_F_DSO_LOAD,DSO_R_NO_FILENAME);
235                 goto err;
236                 }
237         if(ret->meth->dso_load == NULL)
238                 {
239                 DSOerr(DSO_F_DSO_LOAD,DSO_R_UNSUPPORTED);
240                 goto err;
241                 }
242         if(!ret->meth->dso_load(ret))
243                 {
244                 DSOerr(DSO_F_DSO_LOAD,DSO_R_LOAD_FAILED);
245                 goto err;
246                 }
247         /* Load succeeded */
248         return(ret);
249 err:
250         if(allocated)
251                 DSO_free(ret);
252         return(NULL);
253         }

255 void *DSO_bind_var(DSO *dso, const char *symname)
256         {
257         void *ret = NULL;

259         if((dso == NULL) || (symname == NULL))
```

```
260                 {
261                 DSOerr(DSO_F_DSO_BIND_VAR,ERR_R_PASSED_NULL_PARAMETER);
262                 return(NULL);
263                 }
264         if(dso->meth->dso_bind_var == NULL)
265                 {
266                 DSOerr(DSO_F_DSO_BIND_VAR,DSO_R_UNSUPPORTED);
267                 return(NULL);
268                 }
269         if((ret = dso->meth->dso_bind_var(dso, symname)) == NULL)
270                 {
271                 DSOerr(DSO_F_DSO_BIND_VAR,DSO_R_SYM_FAILURE);
272                 return(NULL);
273                 }
274         /* Success */
275         return(ret);
276         }

278 DSO_FUNC_TYPE DSO_bind_func(DSO *dso, const char *symname)
279         {
280         DSO_FUNC_TYPE ret = NULL;

282         if((dso == NULL) || (symname == NULL))
283                 {
284                 DSOerr(DSO_F_DSO_BIND_FUNC,ERR_R_PASSED_NULL_PARAMETER);
285                 return(NULL);
286                 }
287         if(dso->meth->dso_bind_func == NULL)
288                 {
289                 DSOerr(DSO_F_DSO_BIND_FUNC,DSO_R_UNSUPPORTED);
290                 return(NULL);
291                 }
292         if((ret = dso->meth->dso_bind_func(dso, symname)) == NULL)
293                 {
294                 DSOerr(DSO_F_DSO_BIND_FUNC,DSO_R_SYM_FAILURE);
295                 return(NULL);
296                 }
297         /* Success */
298         return(ret);
299         }

301 /* I don't really like these *_ctrl functions very much to be perfectly
302  * honest. For one thing, I think I have to return a negative value for
303  * any error because possible DSO_ctrl() commands may return values
304  * such as "size"s that can legitimately be zero (making the standard
305  * "if(DSO_cmd(...))" form that works almost everywhere else fail at
306  * odd times. I'd prefer "output" values to be passed by reference and
307  * the return value as success/failure like usual ... but we conform
308  * when we must... :-) */
309 long DSO_ctrl(DSO *dso, int cmd, long larg, void *parg)
310         {
311         if(dso == NULL)
312                 {
313                 DSOerr(DSO_F_DSO_CTRL,ERR_R_PASSED_NULL_PARAMETER);
314                 return(-1);
315                 }
316         /* We should intercept certain generic commands and only pass control
317          * to the method-specific ctrl() function if it's something we don't
318          * handle. */
319         switch(cmd)
320                 {
321         case DSO_CTRL_GET_FLAGS:
322                 return dso->flags;
323         case DSO_CTRL_SET_FLAGS:
324                 dso->flags = (int)larg;
325                 return(0);
```



```
326         case DSO_CTRL_OR_FLAGS:
327                 dso->flags |= (int)larg;
328                 return(0);
329         default:
330                 break;
331                 }
332         if((dso->meth == NULL) || (dso->meth->dso_ctrl == NULL))
333                 {
334                 DSOerr(DSO_F_DSO_CTRL,DSO_R_UNSUPPORTED);
335                 return(-1);
336                 }
337         return(dso->meth->dso_ctrl(dso,cmd,larg,parg));
338         }

340 int DSO_set_name_converter(DSO *dso, DSO_NAME_CONVERTER_FUNC cb,
341                         DSO_NAME_CONVERTER_FUNC *oldcb)
342         {
343         if(dso == NULL)
344                 {
345                 DSOerr(DSO_F_DSO_SET_NAME_CONVERTER,
346                                 ERR_R_PASSED_NULL_PARAMETER);
347                 return(0);
348                 }
349         if(oldcb)
350                 *oldcb = dso->name_converter;
351         dso->name_converter = cb;
352         return(1);
353         }

355 const char *DSO_get_filename(DSO *dso)
356         {
357         if(dso == NULL)
358                 {
359                 DSOerr(DSO_F_DSO_GET_FILENAME,ERR_R_PASSED_NULL_PARAMETER);
360                 return(NULL);
361                 }
362         return(dso->filename);
363         }

365 int DSO_set_filename(DSO *dso, const char *filename)
366         {
367         char *copied;

369         if((dso == NULL) || (filename == NULL))
370                 {
371                 DSOerr(DSO_F_DSO_SET_FILENAME,ERR_R_PASSED_NULL_PARAMETER);
372                 return(0);
373                 }
374         if(dso->loaded_filename)
375                 {
376                 DSOerr(DSO_F_DSO_SET_FILENAME,DSO_R_DSO_ALREADY_LOADED);
377                 return(0);
378                 }
379         /* We'll duplicate filename */
380         copied = OPENSSL_malloc(strlen(filename) + 1);
381         if(copied == NULL)
382                 {
383                 DSOerr(DSO_F_DSO_SET_FILENAME,ERR_R_MALLOC_FAILURE);
384                 return(0);
385                 }
386         BUF_strlcpy(copied, filename, strlen(filename) + 1);
387         if(dso->filename)
388                 OPENSSL_free(dso->filename);
389         dso->filename = copied;
390         return(1);
391         }
```

```
393 char *DSO_merge(DSO *dso, const char *filespec1, const char *filespec2)
394         {
395         char *result = NULL;

397         if(dso == NULL || filespec1 == NULL)
398                 {
399                 DSOerr(DSO_F_DSO_MERGE,ERR_R_PASSED_NULL_PARAMETER);
400                 return(NULL);
401                 }
402         if((dso->flags & DSO_FLAG_NO_NAME_TRANSLATION) == 0)
403                 {
404                 if(dso->merger != NULL)
405                         result = dso->merger(dso, filespec1, filespec2);
406                 else if(dso->meth->dso_merger != NULL)
407                         result = dso->meth->dso_merger(dso,
408                                 filespec1, filespec2);
409                 }
410         return(result);
411         }

413 char *DSO_convert_filename(DSO *dso, const char *filename)
414         {
415         char *result = NULL;

417         if(dso == NULL)
418                 {
419                 DSOerr(DSO_F_DSO_CONVERT_FILENAME,ERR_R_PASSED_NULL_PARAMETER);
420                 return(NULL);
421                 }
422         if(filename == NULL)
423                 filename = dso->filename;
424         if(filename == NULL)
425                 {
426                 DSOerr(DSO_F_DSO_CONVERT_FILENAME,DSO_R_NO_FILENAME);
427                 return(NULL);
428                 }
429         if((dso->flags & DSO_FLAG_NO_NAME_TRANSLATION) == 0)
430                 {
431                 if(dso->name_converter != NULL)
432                         result = dso->name_converter(dso, filename);
433                 else if(dso->meth->dso_name_converter != NULL)
434                         result = dso->meth->dso_name_converter(dso, filename);
435                 }
436         if(result == NULL)
437                 {
438                 result = OPENSSL_malloc(strlen(filename) + 1);
439                 if(result == NULL)
440                         {
441                         DSOerr(DSO_F_DSO_CONVERT_FILENAME,
442                                         ERR_R_MALLOC_FAILURE);
443                         return(NULL);
444                         }
445                 BUF_strlcpy(result, filename, strlen(filename) + 1);
446                 }
447         return(result);
448         }

450 const char *DSO_get_loaded_filename(DSO *dso)
451         {
452         if(dso == NULL)
453                 {
454                 DSOerr(DSO_F_DSO_GET_LOADED_FILENAME,
455                                 ERR_R_PASSED_NULL_PARAMETER);
456                 return(NULL);
457                 }
```



```
458         return(dso->loaded_filename);
459         }

461 int DSO_pathbyaddr(void *addr,char *path,int sz)
462         {
463         DSO_METHOD *meth = default_DSO_meth;
464         if (meth == NULL) meth = DSO_METHOD_openssl();
465         if (meth->pathbyaddr == NULL)
466                 {
467                 DSOerr(DSO_F_DSO_PATHBYADDR,DSO_R_UNSUPPORTED);
468                 return -1;
469                 }
470         return (*meth->pathbyaddr)(addr,path,sz);
471         }

473 void *DSO_global_lookup(const char *name)
474         {
475         DSO_METHOD *meth = default_DSO_meth;
476         if (meth == NULL) meth = DSO_METHOD_openssl();
477         if (meth->globallookup == NULL)
478                 {
479                 DSOerr(DSO_F_DSO_GLOBAL_LOOKUP,DSO_R_UNSUPPORTED);
480                 return NULL;
481                 }
482         return (*meth->globallookup)(name);
483         }
484 #endif /* ! codereview */
```

```
**********************************************************
    3406 Wed Aug 13 19:52:34 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_null.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso_null.c */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* This "NULL" method is provided as the fallback for systems that have
 * no appropriate support for "shared-libraries". */
```



```
#include <stdio.h>
#include "cryptlib.h"
#include <openssl/dso.h>

static DSO_METHOD dso_meth_null = {
        "NULL shared library method",
        NULL, /* load */
        NULL, /* unload */
        NULL, /* bind_var */
        NULL, /* bind_func */
/* For now, "unbind" doesn't exist */
#if 0
        NULL, /* unbind_var */
        NULL, /* unbind_func */
#endif
        NULL, /* ctrl */
        NULL, /* dso_name_converter */
        NULL, /* dso_merger */
        NULL, /* init */
        NULL, /* finish */
        NULL, /* pathbyaddr */
        NULL  /* globallookup */
        };

DSO_METHOD *DSO_METHOD_null(void)
        {
        return(&dso_meth_null);
        }
#endif /* ! codereview */
```

****
3262 Wed Aug 13 19:52:35 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_openssl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* dso_openssl.c */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/dso.h>
```



```
/* We just pinch the method from an appropriate "default" method. */

DSO_METHOD *DSO_METHOD_openssl(void)
        {
#ifdef DEF_DSO_METHOD
        return(DEF_DSO_METHOD());
#elif defined(DSO_DLFCN)
        return(DSO_METHOD_dlfcn());
#elif defined(DSO_DL)
        return(DSO_METHOD_dl());
#elif defined(DSO_WIN32)
        return(DSO_METHOD_win32());
#elif defined(DSO_VMS)
        return(DSO_METHOD_vms());
#elif defined(DSO_BEOS)
        return(DSO_METHOD_beos());
#else
        return(DSO_METHOD_null());
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
   15295 Wed Aug 13 19:52:35 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_vms.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso_vms.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <string.h>
#include <errno.h>
```



```
#include "cryptlib.h"
#include <openssl/dso.h>

#ifndef OPENSSL_SYS_VMS
DSO_METHOD *DSO_METHOD_vms(void)
        {
        return NULL;
        }
#else

#pragma message disable DOLLARID
#include <rms.h>
#include <lib$routines.h>
#include <stsdef.h>
#include <descrip.h>
#include <starlet.h>
#include "vms_rms.h"

/* Some compiler options may mask the declaration of "_malloc32". */
#if __INITIAL_POINTER_SIZE && defined _ANSI_C_SOURCE
# if __INITIAL_POINTER_SIZE == 64
#  pragma pointer_size save
#  pragma pointer_size 32
    void * _malloc32  (__size_t);
#  pragma pointer_size restore
# endif /* __INITIAL_POINTER_SIZE == 64 */
#endif /* __INITIAL_POINTER_SIZE && defined _ANSI_C_SOURCE */


#pragma message disable DOLLARID

static int vms_load(DSO *dso);
static int vms_unload(DSO *dso);
static void *vms_bind_var(DSO *dso, const char *symname);
static DSO_FUNC_TYPE vms_bind_func(DSO *dso, const char *symname);
#if 0
static int vms_unbind_var(DSO *dso, char *symname, void *symptr);
static int vms_unbind_func(DSO *dso, char *symname, DSO_FUNC_TYPE symptr);
static int vms_init(DSO *dso);
static int vms_finish(DSO *dso);
static long vms_ctrl(DSO *dso, int cmd, long larg, void *parg);
#endif
static char *vms_name_converter(DSO *dso, const char *filename);
static char *vms_merger(DSO *dso, const char *filespec1,
        const char *filespec2);

static DSO_METHOD dso_meth_vms = {
        "OpenSSL 'VMS' shared library method",
        vms_load,
        NULL, /* unload */
        vms_bind_var,
        vms_bind_func,
/* For now, "unbind" doesn't exist */
#if 0
        NULL, /* unbind_var */
        NULL, /* unbind_func */
#endif
        NULL, /* ctrl */
        vms_name_converter,
        vms_merger,
        NULL, /* init */
        NULL  /* finish */
        };

/* On VMS, the only "handle" is the file name.  LIB$FIND_IMAGE_SYMBOL depends
 * on the reference to the file name being the same for all calls regarding
```

```
 * one shared image, so we'll just store it in an instance of the following
 * structure and put a pointer to that instance in the meth_data stack.
 */
typedef struct dso_internal_st
        {
        /* This should contain the name only, no directory,
         * no extension, nothing but a name. */
        struct dsc$descriptor_s filename_dsc;
        char filename[ NAMX_MAXRSS+ 1];
        /* This contains whatever is not in filename, if needed.
         * Normally not defined. */
        struct dsc$descriptor_s imagename_dsc;
        char imagename[ NAMX_MAXRSS+ 1];
        } DSO_VMS_INTERNAL;

DSO_METHOD *DSO_METHOD_vms(void)
        {
        return(&dso_meth_vms);
        }

static int vms_load(DSO *dso)
        {
        void *ptr = NULL;
        /* See applicable comments in dso_dl.c */
        char *filename = DSO_convert_filename(dso, NULL);

/* Ensure 32-bit pointer for "p", and appropriate malloc() function. */
#if __INITIAL_POINTER_SIZE == 64
# define DSO_MALLOC _malloc32
# pragma pointer_size save
# pragma pointer_size 32
#else /* __INITIAL_POINTER_SIZE == 64 */
# define DSO_MALLOC OPENSSL_malloc
#endif /* __INITIAL_POINTER_SIZE == 64 [else] */

        DSO_VMS_INTERNAL *p = NULL;

#if __INITIAL_POINTER_SIZE == 64
# pragma pointer_size restore
#endif /* __INITIAL_POINTER_SIZE == 64 */

        const char *sp1, *sp2;  /* Search result */

        if(filename == NULL)
                {
                DSOerr(DSO_F_VMS_LOAD,DSO_R_NO_FILENAME);
                goto err;
                }

        /* A file specification may look like this:
         *
         *      node::dev:[dir-spec]name.type;ver
         *
         * or (for compatibility with TOPS-20):
         *
         *      node::dev:<dir-spec>name.type;ver
         *
         * and the dir-spec uses '.' as separator.  Also, a dir-spec
         * may consist of several parts, with mixed use of [] and <>:
         *
         *      [dir1.]<dir2>
         *
         * We need to split the file specification into the name and
         * the rest (both before and after the name itself).
         */
        /* Start with trying to find the end of a dir-spec, and save the
```



```
           position of the byte after in sp1 */
        sp1 = strrchr(filename, ']');
        sp2 = strrchr(filename, '>');
        if (sp1 == NULL) sp1 = sp2;
        if (sp2 != NULL && sp2 > sp1) sp1 = sp2;
        if (sp1 == NULL) sp1 = strrchr(filename, ':');
        if (sp1 == NULL)
                sp1 = filename;
        else
                sp1++;          /* The byte after the found character */
        /* Now, let's see if there's a type, and save the position in sp2 */
        sp2 = strchr(sp1, '.');
        /* If we found it, that's where we'll cut.  Otherwise, look for a
           version number and save the position in sp2 */
        if (sp2 == NULL) sp2 = strchr(sp1, ';');
        /* If there was still nothing to find, set sp2 to point at the end of
           the string */
        if (sp2 == NULL) sp2 = sp1 + strlen(sp1);

        /* Check that we won't get buffer overflows */
        if (sp2 - sp1 > FILENAME_MAX
                || (sp1 - filename) + strlen(sp2) > FILENAME_MAX)
                {
                DSOerr(DSO_F_VMS_LOAD,DSO_R_FILENAME_TOO_BIG);
                goto err;
                }

        p = DSO_MALLOC(sizeof(DSO_VMS_INTERNAL));
        if(p == NULL)
                {
                DSOerr(DSO_F_VMS_LOAD,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        strncpy(p->filename, sp1, sp2-sp1);
        p->filename[sp2-sp1] = '\0';

        strncpy(p->imagename, filename, sp1-filename);
        p->imagename[sp1-filename] = '\0';
        strcat(p->imagename, sp2);

        p->filename_dsc.dsc$w_length = strlen(p->filename);
        p->filename_dsc.dsc$b_dtype = DSC$K_DTYPE_T;
        p->filename_dsc.dsc$b_class = DSC$K_CLASS_S;
        p->filename_dsc.dsc$a_pointer = p->filename;
        p->imagename_dsc.dsc$w_length = strlen(p->imagename);
        p->imagename_dsc.dsc$b_dtype = DSC$K_DTYPE_T;
        p->imagename_dsc.dsc$b_class = DSC$K_CLASS_S;
        p->imagename_dsc.dsc$a_pointer = p->imagename;

        if(!sk_void_push(dso->meth_data, (char *)p))
                {
                DSOerr(DSO_F_VMS_LOAD,DSO_R_STACK_ERROR);
                goto err;
                }

        /* Success (for now, we lie.  We actually do not know...) */
        dso->loaded_filename = filename;
        return(1);
err:
        /* Cleanup! */
        if(p != NULL)
                OPENSSL_free(p);
        if(filename != NULL)
                OPENSSL_free(filename);
        return(0);
```

```
260         }

262 /* Note that this doesn't actually unload the shared image, as there is no
263  * such thing in VMS.  Next time it get loaded again, a new copy will
264  * actually be loaded.
265  */
266 static int vms_unload(DSO *dso)
267         {
268         DSO_VMS_INTERNAL *p;
269         if(dso == NULL)
270                 {
271                 DSOerr(DSO_F_VMS_UNLOAD,ERR_R_PASSED_NULL_PARAMETER);
272                 return(0);
273                 }
274         if(sk_void_num(dso->meth_data) < 1)
275                 return(1);
276         p = (DSO_VMS_INTERNAL *)sk_void_pop(dso->meth_data);
277         if(p == NULL)
278                 {
279                 DSOerr(DSO_F_VMS_UNLOAD,DSO_R_NULL_HANDLE);
280                 return(0);
281                 }
282         /* Cleanup */
283         OPENSSL_free(p);
284         return(1);
285         }

287 /* We must do this in a separate function because of the way the exception
288    handler works (it makes this function return */
289 static int do_find_symbol(DSO_VMS_INTERNAL *ptr,
290         struct dsc$descriptor_s *symname_dsc, void **sym,
291         unsigned long flags)
292         {
293         /* Make sure that signals are caught and returned instead of
294            aborting the program.  The exception handler gets unestablished
295            automatically on return from this function.  */
296         lib$establish(lib$sig_to_ret);

298         if(ptr->imagename_dsc.dsc$w_length)
299                 return lib$find_image_symbol(&ptr->filename_dsc,
300                         symname_dsc, sym,
301                         &ptr->imagename_dsc, flags);
302         else
303                 return lib$find_image_symbol(&ptr->filename_dsc,
304                         symname_dsc, sym,
305                         0, flags);
306         }

308 void vms_bind_sym(DSO *dso, const char *symname, void **sym)
309         {
310         DSO_VMS_INTERNAL *ptr;
311         int status;
312 #if 0
313         int flags = (1<<4); /* LIB$M_FIS_MIXEDCASE, but this symbol isn't
314                                defined in VMS older than 7.0 or so */
315 #else
316         int flags = 0;
317 #endif
318         struct dsc$descriptor_s symname_dsc;

320 /* Arrange 32-bit pointer to (copied) string storage, if needed. */
321 #if __INITIAL_POINTER_SIZE == 64
322 # define SYMNAME symname_32p
323 # pragma pointer_size save
324 # pragma pointer_size 32
325         char *symname_32p;
```



```
326 # pragma pointer_size restore
327         char symname_32[ NAMX_MAXRSS+ 1];
328 #else /* __INITIAL_POINTER_SIZE == 64 */
329 # define SYMNAME ((char *) symname)
330 #endif /* __INITIAL_POINTER_SIZE == 64 [else] */

332         *sym = NULL;

334         if((dso == NULL) || (symname == NULL))
335                 {
336                 DSOerr(DSO_F_VMS_BIND_SYM,ERR_R_PASSED_NULL_PARAMETER);
337                 return;
338                 }

340 #if __INITIAL_POINTER_SIZE == 64
341         /* Copy the symbol name to storage with a 32-bit pointer. */
342         symname_32p = symname_32;
343         strcpy( symname_32p, symname);
344 #endif /* __INITIAL_POINTER_SIZE == 64 [else] */

346         symname_dsc.dsc$w_length = strlen(SYMNAME);
347         symname_dsc.dsc$b_dtype = DSC$K_DTYPE_T;
348         symname_dsc.dsc$b_class = DSC$K_CLASS_S;
349         symname_dsc.dsc$a_pointer = SYMNAME;

351         if(sk_void_num(dso->meth_data) < 1)
352                 {
353                 DSOerr(DSO_F_VMS_BIND_SYM,DSO_R_STACK_ERROR);
354                 return;
355                 }
356         ptr = (DSO_VMS_INTERNAL *)sk_void_value(dso->meth_data,
357                 sk_void_num(dso->meth_data) - 1);
358         if(ptr == NULL)
359                 {
360                 DSOerr(DSO_F_VMS_BIND_SYM,DSO_R_NULL_HANDLE);
361                 return;
362                 }

364         if(dso->flags & DSO_FLAG_UPCASE_SYMBOL) flags = 0;

366         status = do_find_symbol(ptr, &symname_dsc, sym, flags);

368         if(!$VMS_STATUS_SUCCESS(status))
369                 {
370                 unsigned short length;
371                 char errstring[257];
372                 struct dsc$descriptor_s errstring_dsc;

374                 errstring_dsc.dsc$w_length = sizeof(errstring);
375                 errstring_dsc.dsc$b_dtype = DSC$K_DTYPE_T;
376                 errstring_dsc.dsc$b_class = DSC$K_CLASS_S;
377                 errstring_dsc.dsc$a_pointer = errstring;

379                 *sym = NULL;

381                 status = sys$getmsg(status, &length, &errstring_dsc, 1, 0);

383                 if (!$VMS_STATUS_SUCCESS(status))
384                         lib$signal(status); /* This is really bad.  Abort!  */
385                 else
386                         {
387                         errstring[length] = '\0';

389                         DSOerr(DSO_F_VMS_BIND_SYM,DSO_R_SYM_FAILURE);
390                         if (ptr->imagename_dsc.dsc$w_length)
391                                 ERR_add_error_data(9,
```

```
392                                     "Symbol ", symname,
393                                     " in ", ptr->filename,
394                                     " (", ptr->imagename, ")",
395                                     ": ", errstring);
396                             else
397                                     ERR_add_error_data(6,
398                                     "Symbol ", symname,
399                                     " in ", ptr->filename,
400                                     ": ", errstring);
401                             }
402                     return;
403                     }
404             return;
405             }

407     static void *vms_bind_var(DSO *dso, const char *symname)
408             {
409             void *sym = 0;
410             vms_bind_sym(dso, symname, &sym);
411             return sym;
412             }

414     static DSO_FUNC_TYPE vms_bind_func(DSO *dso, const char *symname)
415             {
416             DSO_FUNC_TYPE sym = 0;
417             vms_bind_sym(dso, symname, (void **)&sym);
418             return sym;
419             }


422     static char *vms_merger(DSO *dso, const char *filespec1, const char *filespec2)
423             {
424             int status;
425             int filespec1len, filespec2len;
426             struct FAB fab;
427             struct NAMX_STRUCT nam;
428             char esa[ NAMX_MAXRSS+ 1];
429             char *merged;

431     /* Arrange 32-bit pointer to (copied) string storage, if needed. */
432     #if __INITIAL_POINTER_SIZE == 64
433     # define FILESPEC1 filespec1_32p;
434     # define FILESPEC2 filespec2_32p;
435     # pragma pointer_size save
436     # pragma pointer_size 32
437             char *filespec1_32p;
438             char *filespec2_32p;
439     # pragma pointer_size restore
440             char filespec1_32[ NAMX_MAXRSS+ 1];
441             char filespec2_32[ NAMX_MAXRSS+ 1];
442     #else /* __INITIAL_POINTER_SIZE == 64 */
443     # define FILESPEC1 ((char *) filespec1)
444     # define FILESPEC2 ((char *) filespec2)
445     #endif /* __INITIAL_POINTER_SIZE == 64 [else] */

447             if (!filespec1) filespec1 = "";
448             if (!filespec2) filespec2 = "";
449             filespec1len = strlen(filespec1);
450             filespec2len = strlen(filespec2);

452     #if __INITIAL_POINTER_SIZE == 64
453             /* Copy the file names to storage with a 32-bit pointer. */
454             filespec1_32p = filespec1_32;
455             filespec2_32p = filespec2_32;
456             strcpy( filespec1_32p, filespec1);
457             strcpy( filespec2_32p, filespec2);
```



```
458     #endif /* __INITIAL_POINTER_SIZE == 64 [else] */

460             fab = cc$rms_fab;
461             nam = CC_RMS_NAMX;

463             FAB_OR_NAML( fab, nam).FAB_OR_NAML_FNA = FILESPEC1;
464             FAB_OR_NAML( fab, nam).FAB_OR_NAML_FNS = filespec1len;
465             FAB_OR_NAML( fab, nam).FAB_OR_NAML_DNA = FILESPEC2;
466             FAB_OR_NAML( fab, nam).FAB_OR_NAML_DNS = filespec2len;
467             NAMX_DNA_FNA_SET( fab)

469             nam.NAMX_ESA = esa;
470             nam.NAMX_ESS = NAMX_MAXRSS;
471             nam.NAMX_NOP = NAM$M_SYNCHK | NAM$M_PWD;
472             SET_NAMX_NO_SHORT_UPCASE( nam);

474             fab.FAB_NAMX = &nam;

476             status = sys$parse(&fab, 0, 0);

478             if(!$VMS_STATUS_SUCCESS(status))
479                     {
480                     unsigned short length;
481                     char errstring[257];
482                     struct dsc$descriptor_s errstring_dsc;

484                     errstring_dsc.dsc$w_length = sizeof(errstring);
485                     errstring_dsc.dsc$b_dtype = DSC$K_DTYPE_T;
486                     errstring_dsc.dsc$b_class = DSC$K_CLASS_S;
487                     errstring_dsc.dsc$a_pointer = errstring;

489                     status = sys$getmsg(status, &length, &errstring_dsc, 1, 0);

491                     if (!$VMS_STATUS_SUCCESS(status))
492                             lib$signal(status); /* This is really bad.  Abort!  */
493                     else
494                             {
495                             errstring[length] = '\0';

497                             DSOerr(DSO_F_VMS_MERGER,DSO_R_FAILURE);
498                             ERR_add_error_data(7,
499                                                "filespec \"", filespec1, "\", ",
500                                                "defaults \"", filespec2, "\": ",
501                                                errstring);
502                             }
503                     return(NULL);
504                     }

506             merged = OPENSSL_malloc( nam.NAMX_ESL+ 1);
507             if(!merged)
508                     goto malloc_err;
509             strncpy( merged, nam.NAMX_ESA, nam.NAMX_ESL);
510             merged[ nam.NAMX_ESL] = '\0';
511             return(merged);
512      malloc_err:
513             DSOerr(DSO_F_VMS_MERGER,
514                     ERR_R_MALLOC_FAILURE);
515             }

517     static char *vms_name_converter(DSO *dso, const char *filename)
518             {
519             int len = strlen(filename);
520             char *not_translated = OPENSSL_malloc(len+1);
521             strcpy(not_translated,filename);
522             return(not_translated);
523             }
```

```
525 #endif /* OPENSSL_SYS_VMS */
526 #endif /* ! codereview */
```

```
**********************************************************
   20856 Wed Aug 13 19:52:35 2014
new/usr/src/lib/openssl/libsunw_crypto/dso/dso_win32.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* dso_win32.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <string.h>
#include "cryptlib.h"
```



```
#include <openssl/dso.h>

#if !defined(DSO_WIN32)
DSO_METHOD *DSO_METHOD_win32(void)
        {
        return NULL;
        }
#else

#ifdef _WIN32_WCE
# if _WIN32_WCE < 300
static FARPROC GetProcAddressA(HMODULE hModule,LPCSTR lpProcName)
        {
        WCHAR lpProcNameW[64];
        int i;

        for (i=0;lpProcName[i] && i<64;i++)
                lpProcNameW[i] = (WCHAR)lpProcName[i];
        if (i==64) return NULL;
        lpProcNameW[i] = 0;

        return GetProcAddressW(hModule,lpProcNameW);
        }
# endif
# undef GetProcAddress
# define GetProcAddress GetProcAddressA

static HINSTANCE LoadLibraryA(LPCSTR lpLibFileName)
        {
        WCHAR *fnamw;
        size_t len_0=strlen(lpLibFileName)+1,i;

#ifdef _MSC_VER
        fnamw = (WCHAR *)_alloca (len_0*sizeof(WCHAR));
#else
        fnamw = (WCHAR *)alloca (len_0*sizeof(WCHAR));
#endif
        if (fnamw == NULL)
                {
                SetLastError(ERROR_NOT_ENOUGH_MEMORY);
                return NULL;
                }

#if defined(_WIN32_WCE) && _WIN32_WCE>=101
        if (!MultiByteToWideChar(CP_ACP,0,lpLibFileName,len_0,fnamw,len_0))
#endif
                for (i=0;i<len_0;i++) fnamw[i]=(WCHAR)lpLibFileName[i];

        return LoadLibraryW(fnamw);
        }
#endif

/* Part of the hack in "win32_load" ... */
#define DSO_MAX_TRANSLATED_SIZE 256

static int win32_load(DSO *dso);
static int win32_unload(DSO *dso);
static void *win32_bind_var(DSO *dso, const char *symname);
static DSO_FUNC_TYPE win32_bind_func(DSO *dso, const char *symname);
#if 0
static int win32_unbind_var(DSO *dso, char *symname, void *symptr);
static int win32_unbind_func(DSO *dso, char *symname, DSO_FUNC_TYPE symptr);
static int win32_init(DSO *dso);
static int win32_finish(DSO *dso);
static long win32_ctrl(DSO *dso, int cmd, long larg, void *parg);
#endif
```

```
128 static char *win32_name_converter(DSO *dso, const char *filename);
129 static char *win32_merger(DSO *dso, const char *filespec1,
130         const char *filespec2);
131 static int win32_pathbyaddr(void *addr,char *path,int sz);
132 static void *win32_globallookup(const char *name);

134 static const char *openssl_strnchr(const char *string, int c, size_t len);

136 static DSO_METHOD dso_meth_win32 = {
137         "OpenSSL 'win32' shared library method",
138         win32_load,
139         win32_unload,
140         win32_bind_var,
141         win32_bind_func,
142 /* For now, "unbind" doesn't exist */
143 #if 0
144         NULL, /* unbind_var */
145         NULL, /* unbind_func */
146 #endif
147         NULL, /* ctrl */
148         win32_name_converter,
149         win32_merger,
150         NULL, /* init */
151         NULL, /* finish */
152         win32_pathbyaddr,
153         win32_globallookup
154         };

156 DSO_METHOD *DSO_METHOD_win32(void)
157         {
158         return(&dso_meth_win32);
159         }

161 /* For this DSO_METHOD, our meth_data STACK will contain;
162  * (i) a pointer to the handle (HINSTANCE) returned from
163  *     LoadLibrary(), and copied.
164  */

166 static int win32_load(DSO *dso)
167         {
168         HINSTANCE h = NULL, *p = NULL;
169         /* See applicable comments from dso_dl.c */
170         char *filename = DSO_convert_filename(dso, NULL);

172         if(filename == NULL)
173                 {
174                 DSOerr(DSO_F_WIN32_LOAD,DSO_R_NO_FILENAME);
175                 goto err;
176                 }
177         h = LoadLibraryA(filename);
178         if(h == NULL)
179                 {
180                 DSOerr(DSO_F_WIN32_LOAD,DSO_R_LOAD_FAILED);
181                 ERR_add_error_data(3, "filename(", filename, ")");
182                 goto err;
183                 }
184         p = (HINSTANCE *)OPENSSL_malloc(sizeof(HINSTANCE));
185         if(p == NULL)
186                 {
187                 DSOerr(DSO_F_WIN32_LOAD,ERR_R_MALLOC_FAILURE);
188                 goto err;
189                 }
190         *p = h;
191         if(!sk_void_push(dso->meth_data, p))
192                 {
193                 DSOerr(DSO_F_WIN32_LOAD,DSO_R_STACK_ERROR);
```



```
194                 goto err;
195                 }
196         /* Success */
197         dso->loaded_filename = filename;
198         return(1);
199 err:
200         /* Cleanup !*/
201         if(filename != NULL)
202                 OPENSSL_free(filename);
203         if(p != NULL)
204                 OPENSSL_free(p);
205         if(h != NULL)
206                 FreeLibrary(h);
207         return(0);
208         }

210 static int win32_unload(DSO *dso)
211         {
212         HINSTANCE *p;
213         if(dso == NULL)
214                 {
215                 DSOerr(DSO_F_WIN32_UNLOAD,ERR_R_PASSED_NULL_PARAMETER);
216                 return(0);
217                 }
218         if(sk_void_num(dso->meth_data) < 1)
219                 return(1);
220         p = sk_void_pop(dso->meth_data);
221         if(p == NULL)
222                 {
223                 DSOerr(DSO_F_WIN32_UNLOAD,DSO_R_NULL_HANDLE);
224                 return(0);
225                 }
226         if(!FreeLibrary(*p))
227                 {
228                 DSOerr(DSO_F_WIN32_UNLOAD,DSO_R_UNLOAD_FAILED);
229                 /* We should push the value back onto the stack in
230                  * case of a retry. */
231                 sk_void_push(dso->meth_data, p);
232                 return(0);
233                 }
234         /* Cleanup */
235         OPENSSL_free(p);
236         return(1);
237         }

239 /* Using GetProcAddress for variables? TODO: Check this out in
240  * the Win32 API docs, there's probably a variant for variables. */
241 static void *win32_bind_var(DSO *dso, const char *symname)
242         {
243         HINSTANCE *ptr;
244         void *sym;

246         if((dso == NULL) || (symname == NULL))
247                 {
248                 DSOerr(DSO_F_WIN32_BIND_VAR,ERR_R_PASSED_NULL_PARAMETER);
249                 return(NULL);
250                 }
251         if(sk_void_num(dso->meth_data) < 1)
252                 {
253                 DSOerr(DSO_F_WIN32_BIND_VAR,DSO_R_STACK_ERROR);
254                 return(NULL);
255                 }
256         ptr = sk_void_value(dso->meth_data, sk_void_num(dso->meth_data) - 1);
257         if(ptr == NULL)
258                 {
259                 DSOerr(DSO_F_WIN32_BIND_VAR,DSO_R_NULL_HANDLE);
```

```
260                 return(NULL);
261                 }
262         sym = GetProcAddress(*ptr, symname);
263         if(sym == NULL)
264                 {
265                 DSOerr(DSO_F_WIN32_BIND_VAR,DSO_R_SYM_FAILURE);
266                 ERR_add_error_data(3, "symname(", symname, ")");
267                 return(NULL);
268                 }
269         return(sym);
270         }

272 static DSO_FUNC_TYPE win32_bind_func(DSO *dso, const char *symname)
273         {
274         HINSTANCE *ptr;
275         void *sym;

277         if((dso == NULL) || (symname == NULL))
278                 {
279                 DSOerr(DSO_F_WIN32_BIND_FUNC,ERR_R_PASSED_NULL_PARAMETER);
280                 return(NULL);
281                 }
282         if(sk_void_num(dso->meth_data) < 1)
283                 {
284                 DSOerr(DSO_F_WIN32_BIND_FUNC,DSO_R_STACK_ERROR);
285                 return(NULL);
286                 }
287         ptr = sk_void_value(dso->meth_data, sk_void_num(dso->meth_data) - 1);
288         if(ptr == NULL)
289                 {
290                 DSOerr(DSO_F_WIN32_BIND_FUNC,DSO_R_NULL_HANDLE);
291                 return(NULL);
292                 }
293         sym = GetProcAddress(*ptr, symname);
294         if(sym == NULL)
295                 {
296                 DSOerr(DSO_F_WIN32_BIND_FUNC,DSO_R_SYM_FAILURE);
297                 ERR_add_error_data(3, "symname(", symname, ")");
298                 return(NULL);
299                 }
300         return((DSO_FUNC_TYPE)sym);
301         }

303 struct file_st
304         {
305         const char *node; int nodelen;
306         const char *device; int devicelen;
307         const char *predir; int predirlen;
308         const char *dir; int dirlen;
309         const char *file; int filelen;
310         };

312 static struct file_st *win32_splitter(DSO *dso, const char *filename,
313         int assume_last_is_dir)
314         {
315         struct file_st *result = NULL;
316         enum { IN_NODE, IN_DEVICE, IN_FILE } position;
317         const char *start = filename;
318         char last;

320         if (!filename)
321                 {
322                 DSOerr(DSO_F_WIN32_SPLITTER,DSO_R_NO_FILENAME);
323                 /*goto err;*/
324                 return(NULL);
325                 }
```



```
327         result = OPENSSL_malloc(sizeof(struct file_st));
328         if(result == NULL)
329                 {
330                 DSOerr(DSO_F_WIN32_SPLITTER,
331                         ERR_R_MALLOC_FAILURE);
332                 return(NULL);
333                 }

335         memset(result, 0, sizeof(struct file_st));
336         position = IN_DEVICE;

338         if((filename[0] == '\\' && filename[1] == '\\')
339                 || (filename[0] == '/' && filename[1] == '/'))
340                 {
341                 position = IN_NODE;
342                 filename += 2;
343                 start = filename;
344                 result->node = start;
345                 }

347         do
348                 {
349                 last = filename[0];
350                 switch(last)
351                         {
352                 case ':':
353                         if(position != IN_DEVICE)
354                                 {
355                                 DSOerr(DSO_F_WIN32_SPLITTER,
356                                         DSO_R_INCORRECT_FILE_SYNTAX);
357                                 /*goto err;*/
358                                 OPENSSL_free(result);
359                                 return(NULL);
360                                 }
361                         result->device = start;
362                         result->devicelen = (int)(filename - start);
363                         position = IN_FILE;
364                         start = ++filename;
365                         result->dir = start;
366                         break;
367                 case '\\':
368                 case '/':
369                         if(position == IN_NODE)
370                                 {
371                                 result->nodelen = (int)(filename - start);
372                                 position = IN_FILE;
373                                 start = ++filename;
374                                 result->dir = start;
375                                 }
376                         else if(position == IN_DEVICE)
377                                 {
378                                 position = IN_FILE;
379                                 filename++;
380                                 result->dir = start;
381                                 result->dirlen = (int)(filename - start);
382                                 start = filename;
383                                 }
384                         else
385                                 {
386                                 filename++;
387                                 result->dirlen += (int)(filename - start);
388                                 start = filename;
389                                 }
390                         break;
391                 case '\0':
```

```
392                         if(position == IN_NODE)
393                                 {
394                                 result->nodelen = (int)(filename - start);
395                                 }
396                         else
397                                 {
398                                 if(filename - start > 0)
399                                         {
400                                         if (assume_last_is_dir)
401                                                 {
402                                                 if (position == IN_DEVICE)
403                                                         {
404                                                         result->dir = start;
405                                                         result->dirlen = 0;
406                                                         }
407                                                 result->dirlen +=
408                                                         (int)(filename - start);
409                                                 }
410                                         else
411                                                 {
412                                                 result->file = start;
413                                                 result->filelen =
414                                                         (int)(filename - start);
415                                                 }
416                                         }
417                                 }
418                         break;
419                 default:
420                         filename++;
421                         break;
422                         }
423                 }
424         while(last);

426         if(!result->nodelen) result->node = NULL;
427         if(!result->devicelen) result->device = NULL;
428         if(!result->dirlen) result->dir = NULL;
429         if(!result->filelen) result->file = NULL;

431         return(result);
432         }

434 static char *win32_joiner(DSO *dso, const struct file_st *file_split)
435         {
436         int len = 0, offset = 0;
437         char *result = NULL;
438         const char *start;

440         if(!file_split)
441                 {
442                 DSOerr(DSO_F_WIN32_JOINER,
443                                 ERR_R_PASSED_NULL_PARAMETER);
444                 return(NULL);
445                 }
446         if(file_split->node)
447                 {
448                 len += 2 + file_split->nodelen; /* 2 for starting \\ */
449                 if(file_split->predir || file_split->dir || file_split->file)
450                         len++;  /* 1 for ending \ */
451                 }
452         else if(file_split->device)
453                 {
454                 len += file_split->devicelen + 1; /* 1 for ending : */
455                 }
456         len += file_split->predirlen;
457         if(file_split->predir && (file_split->dir || file_split->file))
```



```
458                 {
459                 len++;  /* 1 for ending \ */
460                 }
461         len += file_split->dirlen;
462         if(file_split->dir && file_split->file)
463                 {
464                 len++;  /* 1 for ending \ */
465                 }
466         len += file_split->filelen;

468         if(!len)
469                 {
470                 DSOerr(DSO_F_WIN32_JOINER, DSO_R_EMPTY_FILE_STRUCTURE);
471                 return(NULL);
472                 }

474         result = OPENSSL_malloc(len + 1);
475         if (!result)
476                 {
477                 DSOerr(DSO_F_WIN32_JOINER,
478                         ERR_R_MALLOC_FAILURE);
479                 return(NULL);
480                 }

482         if(file_split->node)
483                 {
484                 strcpy(&result[offset], "\\\\"); offset += 2;
485                 strncpy(&result[offset], file_split->node,
486                         file_split->nodelen); offset += file_split->nodelen;
487                 if(file_split->predir || file_split->dir || file_split->file)
488                         {
489                         result[offset] = '\\'; offset++;
490                         }
491                 }
492         else if(file_split->device)
493                 {
494                 strncpy(&result[offset], file_split->device,
495                         file_split->devicelen); offset += file_split->devicelen;
496                 result[offset] = ':'; offset++;
497                 }
498         start = file_split->predir;
499         while(file_split->predirlen > (start - file_split->predir))
500                 {
501                 const char *end = openssl_strnchr(start, '/',
502                         file_split->predirlen - (start - file_split->predir));
503                 if(!end)
504                         end = start
505                                 + file_split->predirlen
506                                 - (start - file_split->predir);
507                 strncpy(&result[offset], start,
508                         end - start); offset += (int)(end - start);
509                 result[offset] = '\\'; offset++;
510                 start = end + 1;
511                 }
512 #if 0 /* Not needed, since the directory converter above already appeneded
513          a backslash */
514         if(file_split->predir && (file_split->dir || file_split->file))
515                 {
516                 result[offset] = '\\'; offset++;
517                 }
518 #endif
519         start = file_split->dir;
520         while(file_split->dirlen > (start - file_split->dir))
521                 {
522                 const char *end = openssl_strnchr(start, '/',
523                         file_split->dirlen - (start - file_split->dir));
```

```
524                 if(!end)
525                         end = start
526                                 + file_split->dirlen
527                                 - (start - file_split->dir);
528                 strncpy(&result[offset], start,
529                         end - start); offset += (int)(end - start);
530                 result[offset] = '\\'; offset++;
531                 start = end + 1;
532                 }
533 #if 0 /* Not needed, since the directory converter above already appeneded
534          a backslash */
535         if(file_split->dir && file_split->file)
536                 {
537                 result[offset] = '\\'; offset++;
538                 }
539 #endif
540         strncpy(&result[offset], file_split->file,
541                 file_split->filelen); offset += file_split->filelen;
542         result[offset] = '\0';
543         return(result);
544         }

546 static char *win32_merger(DSO *dso, const char *filespec1, const char *filespec2
547         {
548         char *merged = NULL;
549         struct file_st *filespec1_split = NULL;
550         struct file_st *filespec2_split = NULL;

552         if(!filespec1 && !filespec2)
553                 {
554                 DSOerr(DSO_F_WIN32_MERGER,
555                                 ERR_R_PASSED_NULL_PARAMETER);
556                 return(NULL);
557                 }
558         if (!filespec2)
559                 {
560                 merged = OPENSSL_malloc(strlen(filespec1) + 1);
561                 if(!merged)
562                         {
563                         DSOerr(DSO_F_WIN32_MERGER,
564                                 ERR_R_MALLOC_FAILURE);
565                         return(NULL);
566                         }
567                 strcpy(merged, filespec1);
568                 }
569         else if (!filespec1)
570                 {
571                 merged = OPENSSL_malloc(strlen(filespec2) + 1);
572                 if(!merged)
573                         {
574                         DSOerr(DSO_F_WIN32_MERGER,
575                                 ERR_R_MALLOC_FAILURE);
576                         return(NULL);
577                         }
578                 strcpy(merged, filespec2);
579                 }
580         else
581                 {
582                 filespec1_split = win32_splitter(dso, filespec1, 0);
583                 if (!filespec1_split)
584                         {
585                         DSOerr(DSO_F_WIN32_MERGER,
586                                 ERR_R_MALLOC_FAILURE);
587                         return(NULL);
588                         }
589                 filespec2_split = win32_splitter(dso, filespec2, 1);
```



```
590                 if (!filespec2_split)
591                         {
592                         DSOerr(DSO_F_WIN32_MERGER,
593                                 ERR_R_MALLOC_FAILURE);
594                         OPENSSL_free(filespec1_split);
595                         return(NULL);
596                         }

598                 /* Fill in into filespec1_split */
599                 if (!filespec1_split->node && !filespec1_split->device)
600                         {
601                         filespec1_split->node = filespec2_split->node;
602                         filespec1_split->nodelen = filespec2_split->nodelen;
603                         filespec1_split->device = filespec2_split->device;
604                         filespec1_split->devicelen = filespec2_split->devicelen;
605                         }
606                 if (!filespec1_split->dir)
607                         {
608                         filespec1_split->dir = filespec2_split->dir;
609                         filespec1_split->dirlen = filespec2_split->dirlen;
610                         }
611                 else if (filespec1_split->dir[0] != '\\'
612                         && filespec1_split->dir[0] != '/')
613                         {
614                         filespec1_split->predir = filespec2_split->dir;
615                         filespec1_split->predirlen = filespec2_split->dirlen;
616                         }
617                 if (!filespec1_split->file)
618                         {
619                         filespec1_split->file = filespec2_split->file;
620                         filespec1_split->filelen = filespec2_split->filelen;
621                         }

623                 merged = win32_joiner(dso, filespec1_split);
624                 }
625         OPENSSL_free(filespec1_split);
626         OPENSSL_free(filespec2_split);
627         return(merged);
628         }

630 static char *win32_name_converter(DSO *dso, const char *filename)
631         {
632         char *translated;
633         int len, transform;

635         len = strlen(filename);
636         transform = ((strstr(filename, "/") == NULL) &&
637                         (strstr(filename, "\\") == NULL) &&
638                         (strstr(filename, ":") == NULL));
639         if(transform)
640                 /* We will convert this to "%s.dll" */
641                 translated = OPENSSL_malloc(len + 5);
642         else
643                 /* We will simply duplicate filename */
644                 translated = OPENSSL_malloc(len + 1);
645         if(translated == NULL)
646                 {
647                 DSOerr(DSO_F_WIN32_NAME_CONVERTER,
648                                 DSO_R_NAME_TRANSLATION_FAILED);
649                 return(NULL);
650                 }
651         if(transform)
652                 sprintf(translated, "%s.dll", filename);
653         else
654                 sprintf(translated, "%s", filename);
655         return(translated);
```

```
656         }

658 static const char *openssl_strnchr(const char *string, int c, size_t len)
659         {
660         size_t i;
661         const char *p;
662         for (i = 0, p = string; i < len && *p; i++, p++)
663                 {
664                 if (*p == c)
665                         return p;
666                 }
667         return NULL;
668         }

670 #include <tlhelp32.h>
671 #ifdef _WIN32_WCE
672 # define DLLNAME "TOOLHELP.DLL"
673 #else
674 # ifdef MODULEENTRY32
675 # undef MODULEENTRY32   /* unmask the ASCII version! */
676 # endif
677 # define DLLNAME "KERNEL32.DLL"
678 #endif

680 typedef HANDLE (WINAPI *CREATETOOLHELP32SNAPSHOT)(DWORD, DWORD);
681 typedef BOOL (WINAPI *CLOSETOOLHELP32SNAPSHOT)(HANDLE);
682 typedef BOOL (WINAPI *MODULE32)(HANDLE, MODULEENTRY32 *);

684 static int win32_pathbyaddr(void *addr,char *path,int sz)
685         {
686         HMODULE dll;
687         HANDLE hModuleSnap = INVALID_HANDLE_VALUE;
688         MODULEENTRY32 me32;
689         CREATETOOLHELP32SNAPSHOT create_snap;
690         CLOSETOOLHELP32SNAPSHOT  close_snap;
691         MODULE32 module_first, module_next;
692         int len;

694         if (addr == NULL)
695                 {
696                 union   { int(*f)(void*,char*,int); void *p; } t =
697                         { win32_pathbyaddr };
698                 addr = t.p;
699                 }

701         dll = LoadLibrary(TEXT(DLLNAME));
702         if (dll == NULL)
703                 {
704                 DSOerr(DSO_F_WIN32_PATHBYADDR,DSO_R_UNSUPPORTED);
705                 return -1;
706                 }

708         create_snap = (CREATETOOLHELP32SNAPSHOT)
709                 GetProcAddress(dll,"CreateToolhelp32Snapshot");
710         if (create_snap == NULL)
711                 {
712                 FreeLibrary(dll);
713                 DSOerr(DSO_F_WIN32_PATHBYADDR,DSO_R_UNSUPPORTED);
714                 return -1;
715                 }
716         /* We take the rest for granted... */
717 #ifdef _WIN32_WCE
718         close_snap = (CLOSETOOLHELP32SNAPSHOT)
719                 GetProcAddress(dll,"CloseToolhelp32Snapshot");
720 #else
721         close_snap = (CLOSETOOLHELP32SNAPSHOT)CloseHandle;
```



```
722 #endif
723         module_first = (MODULE32)GetProcAddress(dll,"Module32First");
724         module_next  = (MODULE32)GetProcAddress(dll,"Module32Next");

726         hModuleSnap = (*create_snap)(TH32CS_SNAPMODULE,0);
727         if( hModuleSnap == INVALID_HANDLE_VALUE )
728                 {
729                 FreeLibrary(dll);
730                 DSOerr(DSO_F_WIN32_PATHBYADDR,DSO_R_UNSUPPORTED);
731                 return -1;
732                 }

734         me32.dwSize = sizeof(me32);

736         if(!(*module_first)(hModuleSnap,&me32))
737                 {
738                 (*close_snap)(hModuleSnap);
739                 FreeLibrary(dll);
740                 DSOerr(DSO_F_WIN32_PATHBYADDR,DSO_R_FAILURE);
741                 return -1;
742                 }

744         do      {
745                 if ((BYTE *)addr >= me32.modBaseAddr &&
746                     (BYTE *)addr <  me32.modBaseAddr+me32.modBaseSize)
747                         {
748                         (*close_snap)(hModuleSnap);
749                         FreeLibrary(dll);
750 #ifdef _WIN32_WCE
751 # if _WIN32_WCE >= 101
752                         return WideCharToMultiByte(CP_ACP,0,me32.szExePath,-1,
753                                                    path,sz,NULL,NULL);
754 # else
755                         len = (int)wcslen(me32.szExePath);
756                         if (sz <= 0) return len+1;
757                         if (len >= sz) len=sz-1;
758                         for(i=0;i<len;i++)
759                                 path[i] = (char)me32.szExePath[i];
760                         path[len++] = 0;
761                         return len;
762 # endif
763 #else
764                         len = (int)strlen(me32.szExePath);
765                         if (sz <= 0) return len+1;
766                         if (len >= sz) len=sz-1;
767                         memcpy(path,me32.szExePath,len);
768                         path[len++] = 0;
769                         return len;
770 #endif
771                         }
772                 } while((*module_next)(hModuleSnap, &me32));

774         (*close_snap)(hModuleSnap);
775         FreeLibrary(dll);
776         return 0;
777         }

779 static void *win32_globallookup(const char *name)
780         {
781         HMODULE dll;
782         HANDLE hModuleSnap = INVALID_HANDLE_VALUE;
783         MODULEENTRY32 me32;
784         CREATETOOLHELP32SNAPSHOT create_snap;
785         CLOSETOOLHELP32SNAPSHOT  close_snap;
786         MODULE32 module_first, module_next;
787         FARPROC ret=NULL;
```

```
789         dll = LoadLibrary(TEXT(DLLNAME));
790         if (dll == NULL)
791                 {
792                 DSOerr(DSO_F_WIN32_GLOBALLOOKUP,DSO_R_UNSUPPORTED);
793                 return NULL;
794                 }

796         create_snap = (CREATETOOLHELP32SNAPSHOT)
797                 GetProcAddress(dll,"CreateToolhelp32Snapshot");
798         if (create_snap == NULL)
799                 {
800                 FreeLibrary(dll);
801                 DSOerr(DSO_F_WIN32_GLOBALLOOKUP,DSO_R_UNSUPPORTED);
802                 return NULL;
803                 }
804         /* We take the rest for granted... */
805 #ifdef _WIN32_WCE
806         close_snap = (CLOSETOOLHELP32SNAPSHOT)
807                 GetProcAddress(dll,"CloseToolhelp32Snapshot");
808 #else
809         close_snap = (CLOSETOOLHELP32SNAPSHOT)CloseHandle;
810 #endif
811         module_first = (MODULE32)GetProcAddress(dll,"Module32First");
812         module_next  = (MODULE32)GetProcAddress(dll,"Module32Next");

814         hModuleSnap = (*create_snap)(TH32CS_SNAPMODULE,0);
815         if( hModuleSnap == INVALID_HANDLE_VALUE )
816                 {
817                 FreeLibrary(dll);
818                 DSOerr(DSO_F_WIN32_GLOBALLOOKUP,DSO_R_UNSUPPORTED);
819                 return NULL;
820                 }

822         me32.dwSize = sizeof(me32);

824         if (!(*module_first)(hModuleSnap,&me32))
825                 {
826                 (*close_snap)(hModuleSnap);
827                 FreeLibrary(dll);
828                 return NULL;
829                 }

831         do      {
832                 if ((ret = GetProcAddress(me32.hModule,name)))
833                         {
834                         (*close_snap)(hModuleSnap);
835                         FreeLibrary(dll);
836                         return ret;
837                         }
838                 } while((*module_next)(hModuleSnap,&me32));

840         (*close_snap)(hModuleSnap);
841         FreeLibrary(dll);
842         return NULL;
843         }
844 #endif /* DSO_WIN32 */
845 #endif /* ! codereview */
```

```
**********************************************************
   11307 Wed Aug 13 19:52:35 2014
new/usr/src/lib/openssl/libsunw_crypto/ebcdic.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ebcdic.c */

#ifndef CHARSET_EBCDIC

#include <openssl/e_os2.h>
#if defined(PEDANTIC) || defined(__DECC) || defined(OPENSSL_SYS_MACOSX)
static void *dummy=&dummy;
#endif

#else /*CHARSET_EBCDIC*/

#include "ebcdic.h"
/*      Initial Port for  Apache-1.3     by <Martin.Kraemer@Mch.SNI.De>
 *      Adapted for       OpenSSL-0.9.4  by <Martin.Kraemer@Mch.SNI.De>
 */

#ifdef _OSD_POSIX
/*
    "BS2000 OSD" is a POSIX subsystem on a main frame.
    It is made by Siemens AG, Germany, for their BS2000 mainframe machines.
    Within the POSIX subsystem, the same character set was chosen as in
    "native BS2000", namely EBCDIC. (EDF04)

    The name "ASCII" in these routines is misleading: actually, conversion
    is not between EBCDIC and ASCII, but EBCDIC(EDF04) and ISO-8859.1;
    that means that (western european) national characters are preserved.

    This table is identical to the one used by rsh/rcp/ftp and other POSIX tools
*/

/* Here's the bijective ebcdic-to-ascii table: */
const unsigned char os_toascii[256] = {
/*00*/ 0x00, 0x01, 0x02, 0x03, 0x85, 0x09, 0x86, 0x7f,
       0x87, 0x8d, 0x8e, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /*................*/
/*10*/ 0x10, 0x11, 0x12, 0x13, 0x8f, 0x0a, 0x08, 0x97,
       0x18, 0x19, 0x9c, 0x9d, 0x1c, 0x1d, 0x1e, 0x1f, /*................*/
/*20*/ 0x80, 0x81, 0x82, 0x83, 0x84, 0x92, 0x17, 0x1b,
       0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x05, 0x06, 0x07, /*................*/
/*30*/ 0x90, 0x91, 0x16, 0x93, 0x94, 0x95, 0x96, 0x04,
       0x98, 0x99, 0x9a, 0x9b, 0x14, 0x15, 0x9e, 0x1a, /*................*/
/*40*/ 0x20, 0xa0, 0xe2, 0xe4, 0xe0, 0xe1, 0xe3, 0xe5,
       0xe7, 0xf1, 0x60, 0x2e, 0x3c, 0x28, 0x2b, 0x7c, /* .........'.<(+|*/
/*50*/ 0x26, 0xe9, 0xea, 0xeb, 0xe8, 0xed, 0xee, 0xef,
       0xec, 0xdf, 0x21, 0x24, 0x2a, 0x29, 0x3b, 0x9f, /*&.........!$*);.*/
/*60*/ 0x2d, 0x2f, 0xc2, 0xc4, 0xc0, 0xc1, 0xc3, 0xc5,
       0xc7, 0xd1, 0x5e, 0x2c, 0x25, 0x5f, 0x3e, 0x3f, /*-/........^,%_>?*/
/*70*/ 0xf8, 0xc9, 0xca, 0xcb, 0xc8, 0xcd, 0xce, 0xcf,
       0xcc, 0xa8, 0x3a, 0x23, 0x40, 0x27, 0x3d, 0x22, /*..........:#@'="*/
/*80*/ 0xd8, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
       0x68, 0x69, 0xab, 0xbb, 0xf0, 0xfd, 0xfe, 0xb1, /*.abcdefghi......*/
/*90*/ 0xb0, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, 0x70,
       0x71, 0x72, 0xaa, 0xba, 0xe6, 0xb8, 0xc6, 0xa4, /*.jklmnopqr......*/
/*a0*/ 0xb5, 0xaf, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78,
       0x79, 0x7a, 0xa1, 0xbf, 0xd0, 0xdd, 0xde, 0xae, /*..stuvwxyz......*/
/*b0*/ 0xa2, 0xa3, 0xa5, 0xb7, 0xa9, 0xa7, 0xb6, 0xbc,
       0xbd, 0xbe, 0xac, 0x5b, 0x5c, 0x5d, 0xb4, 0xd7, /*...........[\]..*/
/*c0*/ 0xf9, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,
       0x48, 0x49, 0xad, 0xf4, 0xf6, 0xf2, 0xf3, 0xf5, /*.ABCDEFGHI......*/
/*d0*/ 0xa6, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50,
       0x51, 0x52, 0xb9, 0xfb, 0xfc, 0xdb, 0xfa, 0xff, /*.JKLMNOPQR......*/
/*e0*/ 0xd9, 0xf7, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58,
```



```
       0x59, 0x5a, 0xb2, 0xd4, 0xd6, 0xd2, 0xd3, 0xd5, /*..STUVWXYZ......*/
/*f0*/ 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
       0x38, 0x39, 0xb3, 0x7b, 0xdc, 0x7d, 0xda, 0x7e  /*0123456789.{.}.~*/
};


/* The ascii-to-ebcdic table: */
const unsigned char os_toebcdic[256] = {
/*00*/  0x00, 0x01, 0x02, 0x03, 0x37, 0x2d, 0x2e, 0x2f,
        0x16, 0x05, 0x15, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,  /*................*/
/*10*/  0x10, 0x11, 0x12, 0x13, 0x3c, 0x3d, 0x32, 0x26,
        0x18, 0x19, 0x3f, 0x27, 0x1c, 0x1d, 0x1e, 0x1f,  /*................*/
/*20*/  0x40, 0x5a, 0x7f, 0x7b, 0x5b, 0x6c, 0x50, 0x7d,
        0x4d, 0x5d, 0x5c, 0x4e, 0x6b, 0x60, 0x4b, 0x61,  /* !"#$%&'()*+,-./ */
/*30*/  0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7,
        0xf8, 0xf9, 0x7a, 0x5e, 0x4c, 0x7e, 0x6e, 0x6f,  /*0123456789:;<=>?*/
/*40*/  0x7c, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,
        0xc8, 0xc9, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6,  /*@ABCDEFGHIJKLMNO*/
/*50*/  0xd7, 0xd8, 0xd9, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6,
        0xe7, 0xe8, 0xe9, 0xbb, 0xbc, 0xbd, 0x6a, 0x6d,  /*PQRSTUVWXYZ[\]^_*/
/*60*/  0x4a, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87,
        0x88, 0x89, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96,  /*'abcdefghijklmno*/
/*70*/  0x97, 0x98, 0x99, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6,
        0xa7, 0xa8, 0xa9, 0xfb, 0x4f, 0xfd, 0xff, 0x07,  /*pqrstuvwxyz{|}~.*/
/*80*/  0x20, 0x21, 0x22, 0x23, 0x24, 0x04, 0x06, 0x08,
        0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x09, 0x0a, 0x14,  /*................*/
/*90*/  0x30, 0x31, 0x25, 0x33, 0x34, 0x35, 0x36, 0x17,
        0x38, 0x39, 0x3a, 0x3b, 0x1a, 0x1b, 0x3e, 0x5f,  /*................*/
/*a0*/  0x41, 0xaa, 0xb0, 0xb1, 0x9f, 0xb2, 0xd0, 0xb5,
        0x79, 0xb4, 0x9a, 0x8a, 0xba, 0xca, 0xaf, 0xa1,  /*................*/
/*b0*/  0x90, 0x8f, 0xea, 0xfa, 0xbe, 0xa0, 0xb6, 0xb3,
        0x9d, 0xda, 0x9b, 0x8b, 0xb7, 0xb8, 0xb9, 0xab,  /*................*/
/*c0*/  0x64, 0x65, 0x62, 0x66, 0x63, 0x67, 0x9e, 0x68,
        0x74, 0x71, 0x72, 0x73, 0x78, 0x75, 0x76, 0x77,  /*................*/
/*d0*/  0xac, 0x69, 0xed, 0xee, 0xeb, 0xef, 0xec, 0xbf,
        0x80, 0xe0, 0xfe, 0xdd, 0xfc, 0xad, 0xae, 0x59,  /*................*/
/*e0*/  0x44, 0x45, 0x42, 0x46, 0x43, 0x47, 0x9c, 0x48,
        0x54, 0x51, 0x52, 0x53, 0x58, 0x55, 0x56, 0x57,  /*................*/
/*f0*/  0x8c, 0x49, 0xcd, 0xce, 0xcb, 0xcf, 0xcc, 0xe1,
        0x70, 0xc0, 0xde, 0xdb, 0xdc, 0x8d, 0x8e, 0xdf   /*................*/
};

#else  /*_OSD_POSIX*/

/*
This code does basic character mapping for IBM's TPF and OS/390 operating system
It is a modified version of the BS2000 table.

Bijective EBCDIC (character set IBM-1047) to US-ASCII table:
This table is bijective - there are no ambigous or duplicate characters.
*/
const unsigned char os_toascii[256] = {
    0x00, 0x01, 0x02, 0x03, 0x85, 0x09, 0x86, 0x7f, /* 00-0f:           */
    0x87, 0x8d, 0x8e, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /* ................ */
    0x10, 0x11, 0x12, 0x13, 0x8f, 0x0a, 0x08, 0x97, /* 10-1f:           */
    0x18, 0x19, 0x9c, 0x9d, 0x1c, 0x1d, 0x1e, 0x1f, /* ................ */
    0x80, 0x81, 0x82, 0x83, 0x84, 0x92, 0x17, 0x1b, /* 20-2f:           */
    0x88, 0x89, 0x8a, 0x8b, 0x8c, 0x05, 0x06, 0x07, /* ................ */
    0x90, 0x91, 0x16, 0x93, 0x94, 0x95, 0x96, 0x04, /* 30-3f:           */
    0x98, 0x99, 0x9a, 0x9b, 0x14, 0x15, 0x9e, 0x1a, /* ................ */
    0x20, 0xa0, 0xe2, 0xe4, 0xe0, 0xe1, 0xe3, 0xe5, /* 40-4f:           */
    0xe7, 0xf1, 0xa2, 0x2e, 0x3c, 0x28, 0x2b, 0x7c, /*  ...........<(+| */
    0x26, 0xe9, 0xea, 0xeb, 0xe8, 0xed, 0xee, 0xef, /* 50-5f:           */
    0xec, 0xdf, 0x21, 0x24, 0x2a, 0x29, 0x3b, 0x5e, /* &.........!$*);^ */
    0x2d, 0x2f, 0xc2, 0xc4, 0xc0, 0xc1, 0xc3, 0xc5, /* 60-6f:           */
    0xc7, 0xd1, 0xa6, 0x2c, 0x25, 0x5f, 0x3e, 0x3f, /* -/.........,%_>? */
```

```
    0xf8, 0xc9, 0xca, 0xcb, 0xc8, 0xcd, 0xce, 0xcf, /* 70-7f:           */
    0xcc, 0x60, 0x3a, 0x23, 0x40, 0x27, 0x3d, 0x22, /* .........':#@'=" */
    0xd8, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67, /* 80-8f:           */
    0x68, 0x69, 0xab, 0xbb, 0xf0, 0xfd, 0xfe, 0xb1, /* .abcdefghi...... */
    0xb0, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f, 0x70, /* 90-9f:           */
    0x71, 0x72, 0xaa, 0xba, 0xe6, 0xb8, 0xc6, 0xa4, /* .jklmnopqr...... */
    0xb5, 0x7e, 0x73, 0x74, 0x75, 0x76, 0x77, 0x78, /* a0-af:           */
    0x79, 0x7a, 0xa1, 0xbf, 0xd0, 0x5b, 0xde, 0xae, /* .~stuvwxyz...[.. */
    0xac, 0xa3, 0xa5, 0xb7, 0xa9, 0xa7, 0xb6, 0xbc, /* b0-bf:           */
    0xbd, 0xbe, 0xdd, 0xa8, 0xaf, 0x5d, 0xb4, 0xd7, /* .............].. */
    0x7b, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, /* c0-cf:           */
    0x48, 0x49, 0xad, 0xf4, 0xf6, 0xf2, 0xf3, 0xf5, /* {ABCDEFGHI...... */
    0x7d, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f, 0x50, /* d0-df:           */
    0x51, 0x52, 0xb9, 0xfb, 0xfc, 0xf9, 0xfa, 0xff, /* }JKLMNOPQR...... */
    0x5c, 0xf7, 0x53, 0x54, 0x55, 0x56, 0x57, 0x58, /* e0-ef:           */
    0x59, 0x5a, 0xb2, 0xd4, 0xd6, 0xd2, 0xd3, 0xd5, /* \.STUVWXYZ...... */
    0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, /* f0-ff:           */
    0x38, 0x39, 0xb3, 0xdb, 0xdc, 0xd9, 0xda, 0x9f  /* 0123456789...... */
};


/*
The US-ASCII to EBCDIC (character set IBM-1047) table:
This table is bijective (no ambiguous or duplicate characters)
*/
const unsigned char os_toebcdic[256] = {
    0x00, 0x01, 0x02, 0x03, 0x37, 0x2d, 0x2e, 0x2f, /* 00-0f:           */
    0x16, 0x05, 0x15, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, /* ................ */
    0x10, 0x11, 0x12, 0x13, 0x3c, 0x3d, 0x32, 0x26, /* 10-1f:           */
    0x18, 0x19, 0x3f, 0x27, 0x1c, 0x1d, 0x1e, 0x1f, /* ................ */
    0x40, 0x5a, 0x7f, 0x7b, 0x5b, 0x6c, 0x50, 0x7d, /* 20-2f:           */
    0x4d, 0x5d, 0x5c, 0x4e, 0x6b, 0x60, 0x4b, 0x61, /*  !"#$%&'()*+,-./ */
    0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, /* 30-3f:           */
    0xf8, 0xf9, 0x7a, 0x5e, 0x4c, 0x7e, 0x6e, 0x6f, /* 0123456789:;<=>? */
    0x7c, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, /* 40-4f:           */
    0xc8, 0xc9, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, /* @ABCDEFGHIJKLMNO */
    0xd7, 0xd8, 0xd9, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, /* 50-5f:           */
    0xe7, 0xe8, 0xe9, 0xad, 0xe0, 0xbd, 0x5f, 0x6d, /* PQRSTUVWXYZ[\]^_ */
    0x79, 0x81, 0x82, 0x83, 0x84, 0x85, 0x86, 0x87, /* 60-6f:           */
    0x88, 0x89, 0x91, 0x92, 0x93, 0x94, 0x95, 0x96, /* `abcdefghijklmno */
    0x97, 0x98, 0x99, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, /* 70-7f:           */
    0xa7, 0xa8, 0xa9, 0xc0, 0x4f, 0xd0, 0xa1, 0x07, /* pqrstuvwxyz{|}~. */
    0x20, 0x21, 0x22, 0x23, 0x24, 0x04, 0x06, 0x08, /* 80-8f:           */
    0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x09, 0x0a, 0x14, /* ................ */
    0x30, 0x31, 0x25, 0x33, 0x34, 0x35, 0x36, 0x17, /* 90-9f:           */
    0x38, 0x39, 0x3a, 0x3b, 0x1a, 0x1b, 0x3e, 0xff, /* ................ */
    0x41, 0xaa, 0x4a, 0xb1, 0x9f, 0xb2, 0x6a, 0xb5, /* a0-af:           */
    0xbb, 0xb4, 0x9a, 0x8a, 0xb0, 0xca, 0xaf, 0xbc, /* ................ */
    0x90, 0x8f, 0xea, 0xfa, 0xbe, 0xa0, 0xb6, 0xb3, /* b0-bf:           */
    0x9d, 0xda, 0x9b, 0x8b, 0xb7, 0xb8, 0xb9, 0xab, /* ................ */
    0x64, 0x65, 0x62, 0x66, 0x63, 0x67, 0x9e, 0x68, /* c0-cf:           */
    0x74, 0x71, 0x72, 0x73, 0x78, 0x75, 0x76, 0x77, /* ................ */
    0xac, 0x69, 0xed, 0xee, 0xeb, 0xef, 0xec, 0xbf, /* d0-df:           */
    0x80, 0xfd, 0xfe, 0xfb, 0xfc, 0xba, 0xae, 0x59, /* ................ */
    0x44, 0x45, 0x42, 0x46, 0x43, 0x47, 0x9c, 0x48, /* e0-ef:           */
    0x54, 0x51, 0x52, 0x53, 0x58, 0x55, 0x56, 0x57, /* ................ */
    0x8c, 0x49, 0xcd, 0xce, 0xcb, 0xcf, 0xcc, 0xe1, /* f0-ff:           */
    0x70, 0xdd, 0xde, 0xdb, 0xdc, 0x8d, 0x8e, 0xdf  /* ................ */
};
#endif /*_OSD_POSIX*/

/* Translate a memory block from EBCDIC (host charset) to ASCII (net charset)
 * dest and srce may be identical, or separate memory blocks, but
 * should not overlap. These functions intentionally have an interface
 * compatible to memcpy(3).
 */
```



```
void *
ebcdic2ascii(void *dest, const void *srce, size_t count)
{
    unsigned char *udest = dest;
    const unsigned char *usrce = srce;

    while (count-- != 0) {
        *udest++ = os_toascii[*usrce++];
    }

    return dest;
}

void *
ascii2ebcdic(void *dest, const void *srce, size_t count)
{
    unsigned char *udest = dest;
    const unsigned char *usrce = srce;

    while (count-- != 0) {
        *udest++ = os_toebcdic[*usrce++];
    }

    return dest;
}

#endif
#endif /* ! codereview */
```

```
**********************************************************
    4678 Wed Aug 13 19:52:35 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_all.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/engine/eng_all.c -*- mode: C; c-file-style: "eay" -*- */
/* Written by Richard Levitte <richard@levitte.org> for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include "eng_int.h"
```



```
void ENGINE_load_builtin_engines(void)
        {
        /* Some ENGINEs need this */
        OPENSSL_cpuid_setup();
#if 0
        /* There's no longer any need for an "openssl" ENGINE unless, one day,
         * it is the *only* way for standard builtin implementations to be be
         * accessed (ie. it would be possible to statically link binaries with
         * *no* builtin implementations). */
        ENGINE_load_openssl();
#endif
#if !defined(OPENSSL_NO_HW) && (defined(__OpenBSD__) || defined(__FreeBSD__) ||
        ENGINE_load_cryptodev();
#endif
#ifndef OPENSSL_NO_RSAX
        ENGINE_load_rsax();
#endif
#ifndef OPENSSL_NO_RDRAND
        ENGINE_load_rdrand();
#endif
        ENGINE_load_dynamic();
#ifndef OPENSSL_NO_STATIC_ENGINE
#ifndef OPENSSL_NO_HW
#ifndef OPENSSL_NO_HW_4758_CCA
        ENGINE_load_4758cca();
#endif
#ifndef OPENSSL_NO_HW_AEP
        ENGINE_load_aep();
#endif
#ifndef OPENSSL_NO_HW_ATALLA
        ENGINE_load_atalla();
#endif
#ifndef OPENSSL_NO_HW_CSWIFT
        ENGINE_load_cswift();
#endif
#ifndef OPENSSL_NO_HW_NCIPHER
        ENGINE_load_chil();
#endif
#ifndef OPENSSL_NO_HW_NURON
        ENGINE_load_nuron();
#endif
#ifndef OPENSSL_NO_HW_SUREWARE
        ENGINE_load_sureware();
#endif
#ifndef OPENSSL_NO_HW_UBSEC
        ENGINE_load_ubsec();
#endif
#ifndef OPENSSL_NO_HW_PADLOCK
        ENGINE_load_padlock();
#endif
#ifndef OPENSSL_NO_HW_PKCS11
        ENGINE_load_pk11();
#endif
#endif
#ifndef OPENSSL_NO_GOST
        ENGINE_load_gost();
#endif
#ifndef OPENSSL_NO_GMP
        ENGINE_load_gmp();
#endif
#if defined(OPENSSL_SYS_WIN32) && !defined(OPENSSL_NO_CAPIENG)
        ENGINE_load_capi();
#endif
#endif
        ENGINE_register_all_complete();
        }
```

```
129 #if defined(__OpenBSD__) || defined(__FreeBSD__) || defined(HAVE_CRYPTODEV)
130 void ENGINE_setup_bsd_cryptodev(void) {
131         static int bsd_cryptodev_default_loaded = 0;
132         if (!bsd_cryptodev_default_loaded) {
133                 ENGINE_load_cryptodev();
134                 ENGINE_register_all_complete();
135         }
136         bsd_cryptodev_default_loaded=1;
137 }
138 #endif
139 #endif /* ! codereview */
```

```
**********************************************************
    7088 Wed Aug 13 19:52:35 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_cnf.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* eng_cnf.c */
/* Written by Stephen Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"
#include <openssl/conf.h>
```



```
/* #define ENGINE_CONF_DEBUG */

/* ENGINE config module */

static char *skip_dot(char *name)
        {
        char *p;
        p = strchr(name, '.');
        if (p)
                return p + 1;
        return name;
        }

static STACK_OF(ENGINE) *initialized_engines = NULL;

static int int_engine_init(ENGINE *e)
        {
        if (!ENGINE_init(e))
                return 0;
        if (!initialized_engines)
                initialized_engines = sk_ENGINE_new_null();
        if (!initialized_engines || !sk_ENGINE_push(initialized_engines, e))
                {
                ENGINE_finish(e);
                return 0;
                }
        return 1;
        }


static int int_engine_configure(char *name, char *value, const CONF *cnf)
        {
        int i;
        int ret = 0;
        long do_init = -1;
        STACK_OF(CONF_VALUE) *ecmds;
        CONF_VALUE *ecmd = NULL;
        char *ctrlname, *ctrlvalue;
        ENGINE *e = NULL;
        int soft = 0;

        name = skip_dot(name);
#ifdef ENGINE_CONF_DEBUG
        fprintf(stderr, "Configuring engine %s\n", name);
#endif
        /* Value is a section containing ENGINE commands */
        ecmds = NCONF_get_section(cnf, value);

        if (!ecmds)
                {
                ENGINEerr(ENGINE_F_INT_ENGINE_CONFIGURE, ENGINE_R_ENGINE_SECTION
                return 0;
                }

        for (i = 0; i < sk_CONF_VALUE_num(ecmds); i++)
                {
                ecmd = sk_CONF_VALUE_value(ecmds, i);
                ctrlname = skip_dot(ecmd->name);
                ctrlvalue = ecmd->value;
#ifdef ENGINE_CONF_DEBUG
        fprintf(stderr, "ENGINE conf: doing ctrl(%s,%s)\n", ctrlname, ctrlvalue)
#endif

                /* First handle some special pseudo ctrls */

                /* Override engine name to use */
```

```
128                 if (!strcmp(ctrlname, "engine_id"))
129                         name = ctrlvalue;
130                 else if (!strcmp(ctrlname, "soft_load"))
131                         soft = 1;
132                 /* Load a dynamic ENGINE */
133                 else if (!strcmp(ctrlname, "dynamic_path"))
134                         {
135                         e = ENGINE_by_id("dynamic");
136                         if (!e)
137                                 goto err;
138                         if (!ENGINE_ctrl_cmd_string(e, "SO_PATH", ctrlvalue, 0))
139                                 goto err;
140                         if (!ENGINE_ctrl_cmd_string(e, "LIST_ADD", "2", 0))
141                                 goto err;
142                         if (!ENGINE_ctrl_cmd_string(e, "LOAD", NULL, 0))
143                                 goto err;
144                         }
145                 /* ... add other pseudos here ... */
146                 else
147                         {
148                         /* At this point we need an ENGINE structural reference
149                          * if we don't already have one.
150                          */
151                         if (!e)
152                                 {
153                                 e = ENGINE_by_id(name);
154                                 if (!e && soft)
155                                         {
156                                         ERR_clear_error();
157                                         return 1;
158                                         }
159                                 if (!e)
160                                         goto err;
161                                 }
162                         /* Allow "EMPTY" to mean no value: this allows a valid
163                          * "value" to be passed to ctrls of type NO_INPUT
164                          */
165                         if (!strcmp(ctrlvalue, "EMPTY"))
166                                 ctrlvalue = NULL;
167                         if (!strcmp(ctrlname, "init"))
168                                 {
169                                 if (!NCONF_get_number_e(cnf, value, "init", &do_
170                                         goto err;
171                                 if (do_init == 1)
172                                         {
173                                         if (!int_engine_init(e))
174                                                 goto err;
175                                         }
176                                 else if (do_init != 0)
177                                         {
178                                         ENGINEerr(ENGINE_F_INT_ENGINE_CONFIGURE,
179                                         goto err;
180                                         }
181                                 }
182                         else if (!strcmp(ctrlname, "default_algorithms"))
183                                 {
184                                 if (!ENGINE_set_default_string(e, ctrlvalue))
185                                         goto err;
186                                 }
187                         else if (!ENGINE_ctrl_cmd_string(e,
188                                         ctrlname, ctrlvalue, 0))
189                                 goto err;
190                         }
```



```
194                 }
195         if (e && (do_init == -1) && !int_engine_init(e))
196                 {
197                 ecmd = NULL;
198                 goto err;
199                 }
200         ret = 1;
201         err:
202         if (ret != 1)
203                 {
204                 ENGINEerr(ENGINE_F_INT_ENGINE_CONFIGURE, ENGINE_R_ENGINE_CONFIGU
205                 if (ecmd)
206                         ERR_add_error_data(6, "section=", ecmd->section,
207                                               ", name=", ecmd->name,
208                                               ", value=", ecmd->value);
209                 }
210         if (e)
211                 ENGINE_free(e);
212         return ret;
213         }


216 static int int_engine_module_init(CONF_IMODULE *md, const CONF *cnf)
217         {
218         STACK_OF(CONF_VALUE) *elist;
219         CONF_VALUE *cval;
220         int i;
221 #ifdef ENGINE_CONF_DEBUG
222         fprintf(stderr, "Called engine module: name %s, value %s\n",
223                         CONF_imodule_get_name(md), CONF_imodule_get_value(md));
224 #endif
225         /* Value is a section containing ENGINEs to configure */
226         elist = NCONF_get_section(cnf, CONF_imodule_get_value(md));

228         if (!elist)
229                 {
230                 ENGINEerr(ENGINE_F_INT_ENGINE_MODULE_INIT, ENGINE_R_ENGINES_SECT
231                 return 0;
232                 }

234         for (i = 0; i < sk_CONF_VALUE_num(elist); i++)
235                 {
236                 cval = sk_CONF_VALUE_value(elist, i);
237                 if (!int_engine_configure(cval->name, cval->value, cnf))
238                         return 0;
239                 }

241         return 1;
242         }

244 static void int_engine_module_finish(CONF_IMODULE *md)
245         {
246         ENGINE *e;
247         while ((e = sk_ENGINE_pop(initialized_engines)))
248                 ENGINE_finish(e);
249         sk_ENGINE_free(initialized_engines);
250         initialized_engines = NULL;
251         }


254 void ENGINE_add_conf_module(void)
255         {
256         CONF_module_add("engines",
257                         int_engine_module_init,
258                         int_engine_module_finish);
259         }
```

```
260 #endif /* ! codereview */
```

```
**********************************************************
   34887 Wed Aug 13 19:52:36 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_cryptodev.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Copyright (c) 2002 Bob Beck <beck@openbsd.org>
 * Copyright (c) 2002 Theo de Raadt
 * Copyright (c) 2002 Markus Friedl
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY
 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY
 * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
 * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */

#include <openssl/objects.h>
#include <openssl/engine.h>
#include <openssl/evp.h>
#include <openssl/bn.h>

#if (defined(__unix__) || defined(unix)) && !defined(USG) && \
        (defined(OpenBSD) || defined(__FreeBSD__))
#include <sys/param.h>
# if (OpenBSD >= 200112) || ((__FreeBSD_version >= 470101 && __FreeBSD_version <
#  define HAVE_CRYPTODEV
# endif
# if (OpenBSD >= 200110)
#  define HAVE_SYSLOG_R
# endif
#endif

#ifndef HAVE_CRYPTODEV

void
ENGINE_load_cryptodev(void)
{
        /* This is a NOP on platforms without /dev/crypto */
        return;
}

#else

#include <sys/types.h>
#include <crypto/cryptodev.h>
#include <crypto/dh/dh.h>
#include <crypto/dsa/dsa.h>
#include <crypto/err/err.h>
#include <crypto/rsa/rsa.h>
```



```
#include <sys/ioctl.h>
#include <errno.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdarg.h>
#include <syslog.h>
#include <errno.h>
#include <string.h>

struct dev_crypto_state {
        struct session_op d_sess;
        int d_fd;

#ifdef USE_CRYPTODEV_DIGESTS
        char dummy_mac_key[HASH_MAX_LEN];

        unsigned char digest_res[HASH_MAX_LEN];
        char *mac_data;
        int mac_len;
#endif
};

static u_int32_t cryptodev_asymfeat = 0;

static int get_asym_dev_crypto(void);
static int open_dev_crypto(void);
static int get_dev_crypto(void);
static int get_cryptodev_ciphers(const int **cnids);
#ifdef USE_CRYPTODEV_DIGESTS
static int get_cryptodev_digests(const int **cnids);
#endif
static int cryptodev_usable_ciphers(const int **nids);
static int cryptodev_usable_digests(const int **nids);
static int cryptodev_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
    const unsigned char *in, size_t inl);
static int cryptodev_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
    const unsigned char *iv, int enc);
static int cryptodev_cleanup(EVP_CIPHER_CTX *ctx);
static int cryptodev_engine_ciphers(ENGINE *e, const EVP_CIPHER **cipher,
    const int **nids, int nid);
static int cryptodev_engine_digests(ENGINE *e, const EVP_MD **digest,
    const int **nids, int nid);
static int bn2crparam(const BIGNUM *a, struct crparam *crp);
static int crparam2bn(struct crparam *crp, BIGNUM *a);
static void zapparams(struct crypt_kop *kop);
static int cryptodev_asym(struct crypt_kop *kop, int rlen, BIGNUM *r,
    int slen, BIGNUM *s);

static int cryptodev_bn_mod_exp(BIGNUM *r, const BIGNUM *a,
    const BIGNUM *p, const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx);
static int cryptodev_rsa_nocrt_mod_exp(BIGNUM *r0, const BIGNUM *I,
    RSA *rsa, BN_CTX *ctx);
static int cryptodev_rsa_mod_exp(BIGNUM *r0, const BIGNUM *I, RSA *rsa, BN_CTX *
static int cryptodev_dsa_bn_mod_exp(DSA *dsa, BIGNUM *r, BIGNUM *a,
    const BIGNUM *p, const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx);
static int cryptodev_dsa_dsa_mod_exp(DSA *dsa, BIGNUM *t1, BIGNUM *g,
    BIGNUM *u1, BIGNUM *pub_key, BIGNUM *u2, BIGNUM *p,
    BN_CTX *ctx, BN_MONT_CTX *mont);
static DSA_SIG *cryptodev_dsa_do_sign(const unsigned char *dgst,
    int dlen, DSA *dsa);
static int cryptodev_dsa_verify(const unsigned char *dgst, int dgst_len,
    DSA_SIG *sig, DSA *dsa);
static int cryptodev_mod_exp_dh(const DH *dh, BIGNUM *r, const BIGNUM *a,
    const BIGNUM *p, const BIGNUM *m, BN_CTX *ctx,
    BN_MONT_CTX *m_ctx);
```

```
128 static int cryptodev_dh_compute_key(unsigned char *key,
129     const BIGNUM *pub_key, DH *dh);
130 static int cryptodev_ctrl(ENGINE *e, int cmd, long i, void *p,
131     void (*f)(void));
132 void ENGINE_load_cryptodev(void);

134 static const ENGINE_CMD_DEFN cryptodev_defns[] = {
135         { 0, NULL, NULL, 0 }
136 };

138 static struct {
139         int     id;
140         int     nid;
141         int     ivmax;
142         int     keylen;
143 } ciphers[] = {
144         { CRYPTO_ARC4,                  NID_rc4,                0,      16, },
145         { CRYPTO_DES_CBC,               NID_des_cbc,            8,       8, },
146         { CRYPTO_3DES_CBC,              NID_des_ede3_cbc,       8,      24, },
147         { CRYPTO_AES_CBC,               NID_aes_128_cbc,        16,     16, },
148         { CRYPTO_AES_CBC,               NID_aes_192_cbc,        16,     24, },
149         { CRYPTO_AES_CBC,               NID_aes_256_cbc,        16,     32, },
150         { CRYPTO_BLF_CBC,               NID_bf_cbc,             8,      16, },
151         { CRYPTO_CAST_CBC,              NID_cast5_cbc,          8,      16, },
152         { CRYPTO_SKIPJACK_CBC,          NID_undef,              0,       0, },
153         { 0,                            NID_undef,              0,       0, },
154 };

156 #ifdef USE_CRYPTODEV_DIGESTS
157 static struct {
158         int     id;
159         int     nid;
160         int     keylen;
161 } digests[] = {
162         { CRYPTO_MD5_HMAC,              NID_hmacWithMD5,        16},
163         { CRYPTO_SHA1_HMAC,             NID_hmacWithSHA1,       20},
164         { CRYPTO_RIPEMD160_HMAC,        NID_ripemd160,          16/*?*/},
165         { CRYPTO_MD5_KPDK,              NID_undef,              0},
166         { CRYPTO_SHA1_KPDK,             NID_undef,              0},
167         { CRYPTO_MD5,                   NID_md5,                16},
168         { CRYPTO_SHA1,                  NID_sha1,               20},
169         { 0,                            NID_undef,              0},
170 };
171 #endif

173 /*
174  * Return a fd if /dev/crypto seems usable, 0 otherwise.
175  */
176 static int
177 open_dev_crypto(void)
178 {
179         static int fd = -1;

181         if (fd == -1) {
182                 if ((fd = open("/dev/crypto", O_RDWR, 0)) == -1)
183                         return (-1);
184                 /* close on exec */
185                 if (fcntl(fd, F_SETFD, 1) == -1) {
186                         close(fd);
187                         fd = -1;
188                         return (-1);
189                 }
190         }
191         return (fd);
192 }
```



```
194 static int
195 get_dev_crypto(void)
196 {
197         int fd, retfd;

199         if ((fd = open_dev_crypto()) == -1)
200                 return (-1);
201 #ifndef CRIOGET_NOT_NEEDED
202         if (ioctl(fd, CRIOGET, &retfd) == -1)
203                 return (-1);

205         /* close on exec */
206         if (fcntl(retfd, F_SETFD, 1) == -1) {
207                 close(retfd);
208                 return (-1);
209         }
210 #else
211         retfd = fd;
212 #endif
213         return (retfd);
214 }

216 static void put_dev_crypto(int fd)
217 {
218 #ifndef CRIOGET_NOT_NEEDED
219         close(fd);
220 #endif
221 }

223 /* Caching version for asym operations */
224 static int
225 get_asym_dev_crypto(void)
226 {
227         static int fd = -1;

229         if (fd == -1)
230                 fd = get_dev_crypto();
231         return fd;
232 }

234 /*
235  * Find out what ciphers /dev/crypto will let us have a session for.
236  * XXX note, that some of these openssl doesn't deal with yet!
237  * returning them here is harmless, as long as we return NULL
238  * when asked for a handler in the cryptodev_engine_ciphers routine
239  */
240 static int
241 get_cryptodev_ciphers(const int **cnids)
242 {
243         static int nids[CRYPTO_ALGORITHM_MAX];
244         struct session_op sess;
245         int fd, i, count = 0;

247         if ((fd = get_dev_crypto()) < 0) {
248                 *cnids = NULL;
249                 return (0);
250         }
251         memset(&sess, 0, sizeof(sess));
252         sess.key = (caddr_t)"123456789abcdefghijklmno";

254         for (i = 0; ciphers[i].id && count < CRYPTO_ALGORITHM_MAX; i++) {
255                 if (ciphers[i].nid == NID_undef)
256                         continue;
257                 sess.cipher = ciphers[i].id;
258                 sess.keylen = ciphers[i].keylen;
259                 sess.mac = 0;
```

```
260                 if (ioctl(fd, CIOCGSESSION, &sess) != -1 &&
261                     ioctl(fd, CIOCFSESSION, &sess.ses) != -1)
262                         nids[count++] = ciphers[i].nid;
263         }
264         put_dev_crypto(fd);
265
266         if (count > 0)
267                 *cnids = nids;
268         else
269                 *cnids = NULL;
270         return (count);
271 }
272
273 #ifdef USE_CRYPTODEV_DIGESTS
274 /*
275  * Find out what digests /dev/crypto will let us have a session for.
276  * XXX note, that some of these openssl doesn't deal with yet!
277  * returning them here is harmless, as long as we return NULL
278  * when asked for a handler in the cryptodev_engine_digests routine
279  */
280 static int
281 get_cryptodev_digests(const int **cnids)
282 {
283         static int nids[CRYPTO_ALGORITHM_MAX];
284         struct session_op sess;
285         int fd, i, count = 0;
286
287         if ((fd = get_dev_crypto()) < 0) {
288                 *cnids = NULL;
289                 return (0);
290         }
291         memset(&sess, 0, sizeof(sess));
292         sess.mackey = (caddr_t)"123456789abcdefghijklmno";
293         for (i = 0; digests[i].id && count < CRYPTO_ALGORITHM_MAX; i++) {
294                 if (digests[i].nid == NID_undef)
295                         continue;
296                 sess.mac = digests[i].id;
297                 sess.mackeylen = digests[i].keylen;
298                 sess.cipher = 0;
299                 if (ioctl(fd, CIOCGSESSION, &sess) != -1 &&
300                     ioctl(fd, CIOCFSESSION, &sess.ses) != -1)
301                         nids[count++] = digests[i].nid;
302         }
303         put_dev_crypto(fd);
304
305         if (count > 0)
306                 *cnids = nids;
307         else
308                 *cnids = NULL;
309         return (count);
310 }
311 #endif  /* 0 */
312
313 /*
314  * Find the useable ciphers|digests from dev/crypto - this is the first
315  * thing called by the engine init crud which determines what it
316  * can use for ciphers from this engine. We want to return
317  * only what we can do, anythine else is handled by software.
318  *
319  * If we can't initialize the device to do anything useful for
320  * any reason, we want to return a NULL array, and 0 length,
321  * which forces everything to be done is software. By putting
322  * the initalization of the device in here, we ensure we can
323  * use this engine as the default, and if for whatever reason
324  * /dev/crypto won't do what we want it will just be done in
325  * software
```



```
326  *
327  * This can (should) be greatly expanded to perhaps take into
328  * account speed of the device, and what we want to do.
329  * (although the disabling of particular alg's could be controlled
330  * by the device driver with sysctl's.) - this is where we
331  * want most of the decisions made about what we actually want
332  * to use from /dev/crypto.
333  */
334 static int
335 cryptodev_usable_ciphers(const int **nids)
336 {
337         return (get_cryptodev_ciphers(nids));
338 }
339
340 static int
341 cryptodev_usable_digests(const int **nids)
342 {
343 #ifdef USE_CRYPTODEV_DIGESTS
344         return (get_cryptodev_digests(nids));
345 #else
346         /*
347          * XXXX just disable all digests for now, because it sucks.
348          * we need a better way to decide this - i.e. I may not
349          * want digests on slow cards like hifn on fast machines,
350          * but might want them on slow or loaded machines, etc.
351          * will also want them when using crypto cards that don't
352          * suck moose gonads - would be nice to be able to decide something
353          * as reasonable default without having hackery that's card dependent.
354          * of course, the default should probably be just do everything,
355          * with perhaps a sysctl to turn algoritms off (or have them off
356          * by default) on cards that generally suck like the hifn.
357          */
358         *nids = NULL;
359         return (0);
360 #endif
361 }
362
363 static int
364 cryptodev_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
365     const unsigned char *in, size_t inl)
366 {
367         struct crypt_op cryp;
368         struct dev_crypto_state *state = ctx->cipher_data;
369         struct session_op *sess = &state->d_sess;
370         const void *iiv;
371         unsigned char save_iv[EVP_MAX_IV_LENGTH];
372
373         if (state->d_fd < 0)
374                 return (0);
375         if (!inl)
376                 return (1);
377         if ((inl % ctx->cipher->block_size) != 0)
378                 return (0);
379
380         memset(&cryp, 0, sizeof(cryp));
381
382         cryp.ses = sess->ses;
383         cryp.flags = 0;
384         cryp.len = inl;
385         cryp.src = (caddr_t) in;
386         cryp.dst = (caddr_t) out;
387         cryp.mac = 0;
388
389         cryp.op = ctx->encrypt ? COP_ENCRYPT : COP_DECRYPT;
390
391         if (ctx->cipher->iv_len) {
```

```
392                 cryp.iv = (caddr_t) ctx->iv;
393                 if (!ctx->encrypt) {
394                         iiv = in + inl - ctx->cipher->iv_len;
395                         memcpy(save_iv, iiv, ctx->cipher->iv_len);
396                 }
397         } else
398                 cryp.iv = NULL;
399
400         if (ioctl(state->d_fd, CIOCCRYPT, &cryp) == -1) {
401                 /* XXX need better errror handling
402                  * this can fail for a number of different reasons.
403                  */
404                 return (0);
405         }
406
407         if (ctx->cipher->iv_len) {
408                 if (ctx->encrypt)
409                         iiv = out + inl - ctx->cipher->iv_len;
410                 else
411                         iiv = save_iv;
412                 memcpy(ctx->iv, iiv, ctx->cipher->iv_len);
413         }
414         return (1);
415 }
416
417 static int
418 cryptodev_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
419     const unsigned char *iv, int enc)
420 {
421         struct dev_crypto_state *state = ctx->cipher_data;
422         struct session_op *sess = &state->d_sess;
423         int cipher = -1, i;
424
425         for (i = 0; ciphers[i].id; i++)
426                 if (ctx->cipher->nid == ciphers[i].nid &&
427                     ctx->cipher->iv_len <= ciphers[i].ivmax &&
428                     ctx->key_len == ciphers[i].keylen) {
429                         cipher = ciphers[i].id;
430                         break;
431                 }
432
433         if (!ciphers[i].id) {
434                 state->d_fd = -1;
435                 return (0);
436         }
437
438         memset(sess, 0, sizeof(struct session_op));
439
440         if ((state->d_fd = get_dev_crypto()) < 0)
441                 return (0);
442
443         sess->key = (caddr_t)key;
444         sess->keylen = ctx->key_len;
445         sess->cipher = cipher;
446
447         if (ioctl(state->d_fd, CIOCGSESSION, sess) == -1) {
448                 put_dev_crypto(state->d_fd);
449                 state->d_fd = -1;
450                 return (0);
451         }
452         return (1);
453 }
454
455 /*
456  * free anything we allocated earlier when initting a
457  * session, and close the session.
```



```
458  */
459 static int
460 cryptodev_cleanup(EVP_CIPHER_CTX *ctx)
461 {
462         int ret = 0;
463         struct dev_crypto_state *state = ctx->cipher_data;
464         struct session_op *sess = &state->d_sess;
465
466         if (state->d_fd < 0)
467                 return (0);
468
469         /* XXX if this ioctl fails, someting's wrong. the invoker
470          * may have called us with a bogus ctx, or we could
471          * have a device that for whatever reason just doesn't
472          * want to play ball - it's not clear what's right
473          * here - should this be an error? should it just
474          * increase a counter, hmm. For right now, we return
475          * 0 - I don't believe that to be "right". we could
476          * call the gorpy openssl lib error handlers that
477          * print messages to users of the library. hmm..
478          */
479
480         if (ioctl(state->d_fd, CIOCFSESSION, &sess->ses) == -1) {
481                 ret = 0;
482         } else {
483                 ret = 1;
484         }
485         put_dev_crypto(state->d_fd);
486         state->d_fd = -1;
487
488         return (ret);
489 }
490
491 /*
492  * libcrypto EVP stuff - this is how we get wired to EVP so the engine
493  * gets called when libcrypto requests a cipher NID.
494  */
495
496 /* RC4 */
497 const EVP_CIPHER cryptodev_rc4 = {
498         NID_rc4,
499         1, 16, 0,
500         EVP_CIPH_VARIABLE_LENGTH,
501         cryptodev_init_key,
502         cryptodev_cipher,
503         cryptodev_cleanup,
504         sizeof(struct dev_crypto_state),
505         NULL,
506         NULL,
507         NULL
508 };
509
510 /* DES CBC EVP */
511 const EVP_CIPHER cryptodev_des_cbc = {
512         NID_des_cbc,
513         8, 8, 8,
514         EVP_CIPH_CBC_MODE,
515         cryptodev_init_key,
516         cryptodev_cipher,
517         cryptodev_cleanup,
518         sizeof(struct dev_crypto_state),
519         EVP_CIPHER_set_asn1_iv,
520         EVP_CIPHER_get_asn1_iv,
521         NULL
522 };
```

```
524 /* 3DES CBC EVP */
525 const EVP_CIPHER cryptodev_3des_cbc = {
526         NID_des_ede3_cbc,
527         8, 24, 8,
528         EVP_CIPH_CBC_MODE,
529         cryptodev_init_key,
530         cryptodev_cipher,
531         cryptodev_cleanup,
532         sizeof(struct dev_crypto_state),
533         EVP_CIPHER_set_asn1_iv,
534         EVP_CIPHER_get_asn1_iv,
535         NULL
536 };

538 const EVP_CIPHER cryptodev_bf_cbc = {
539         NID_bf_cbc,
540         8, 16, 8,
541         EVP_CIPH_CBC_MODE,
542         cryptodev_init_key,
543         cryptodev_cipher,
544         cryptodev_cleanup,
545         sizeof(struct dev_crypto_state),
546         EVP_CIPHER_set_asn1_iv,
547         EVP_CIPHER_get_asn1_iv,
548         NULL
549 };

551 const EVP_CIPHER cryptodev_cast_cbc = {
552         NID_cast5_cbc,
553         8, 16, 8,
554         EVP_CIPH_CBC_MODE,
555         cryptodev_init_key,
556         cryptodev_cipher,
557         cryptodev_cleanup,
558         sizeof(struct dev_crypto_state),
559         EVP_CIPHER_set_asn1_iv,
560         EVP_CIPHER_get_asn1_iv,
561         NULL
562 };

564 const EVP_CIPHER cryptodev_aes_cbc = {
565         NID_aes_128_cbc,
566         16, 16, 16,
567         EVP_CIPH_CBC_MODE,
568         cryptodev_init_key,
569         cryptodev_cipher,
570         cryptodev_cleanup,
571         sizeof(struct dev_crypto_state),
572         EVP_CIPHER_set_asn1_iv,
573         EVP_CIPHER_get_asn1_iv,
574         NULL
575 };

577 const EVP_CIPHER cryptodev_aes_192_cbc = {
578         NID_aes_192_cbc,
579         16, 24, 16,
580         EVP_CIPH_CBC_MODE,
581         cryptodev_init_key,
582         cryptodev_cipher,
583         cryptodev_cleanup,
584         sizeof(struct dev_crypto_state),
585         EVP_CIPHER_set_asn1_iv,
586         EVP_CIPHER_get_asn1_iv,
587         NULL
588 };
```



```
590 const EVP_CIPHER cryptodev_aes_256_cbc = {
591         NID_aes_256_cbc,
592         16, 32, 16,
593         EVP_CIPH_CBC_MODE,
594         cryptodev_init_key,
595         cryptodev_cipher,
596         cryptodev_cleanup,
597         sizeof(struct dev_crypto_state),
598         EVP_CIPHER_set_asn1_iv,
599         EVP_CIPHER_get_asn1_iv,
600         NULL
601 };

603 /*
604  * Registered by the ENGINE when used to find out how to deal with
605  * a particular NID in the ENGINE. this says what we'll do at the
606  * top level - note, that list is restricted by what we answer with
607  */
608 static int
609 cryptodev_engine_ciphers(ENGINE *e, const EVP_CIPHER **cipher,
610     const int **nids, int nid)
611 {
612         if (!cipher)
613                 return (cryptodev_usable_ciphers(nids));

615         switch (nid) {
616         case NID_rc4:
617                 *cipher = &cryptodev_rc4;
618                 break;
619         case NID_des_ede3_cbc:
620                 *cipher = &cryptodev_3des_cbc;
621                 break;
622         case NID_des_cbc:
623                 *cipher = &cryptodev_des_cbc;
624                 break;
625         case NID_bf_cbc:
626                 *cipher = &cryptodev_bf_cbc;
627                 break;
628         case NID_cast5_cbc:
629                 *cipher = &cryptodev_cast_cbc;
630                 break;
631         case NID_aes_128_cbc:
632                 *cipher = &cryptodev_aes_cbc;
633                 break;
634         case NID_aes_192_cbc:
635                 *cipher = &cryptodev_aes_192_cbc;
636                 break;
637         case NID_aes_256_cbc:
638                 *cipher = &cryptodev_aes_256_cbc;
639                 break;
640         default:
641                 *cipher = NULL;
642                 break;
643         }
644         return (*cipher != NULL);
645 }


648 #ifdef USE_CRYPTODEV_DIGESTS

650 /* convert digest type to cryptodev */
651 static int
652 digest_nid_to_cryptodev(int nid)
653 {
654         int i;
```

```
656         for (i = 0; digests[i].id; i++)
657                 if (digests[i].nid == nid)
658                         return (digests[i].id);
659         return (0);
660 }


663 static int
664 digest_key_length(int nid)
665 {
666         int i;

668         for (i = 0; digests[i].id; i++)
669                 if (digests[i].nid == nid)
670                         return digests[i].keylen;
671         return (0);
672 }


675 static int cryptodev_digest_init(EVP_MD_CTX *ctx)
676 {
677         struct dev_crypto_state *state = ctx->md_data;
678         struct session_op *sess = &state->d_sess;
679         int digest;

681         if ((digest = digest_nid_to_cryptodev(ctx->digest->type)) == NID_undef){
682                 printf("cryptodev_digest_init: Can't get digest \n");
683                 return (0);
684         }

686         memset(state, 0, sizeof(struct dev_crypto_state));

688         if ((state->d_fd = get_dev_crypto()) < 0) {
689                 printf("cryptodev_digest_init: Can't get Dev \n");
690                 return (0);
691         }

693         sess->mackey = state->dummy_mac_key;
694         sess->mackeylen = digest_key_length(ctx->digest->type);
695         sess->mac = digest;

697         if (ioctl(state->d_fd, CIOCGSESSION, sess) < 0) {
698                 put_dev_crypto(state->d_fd);
699                 state->d_fd = -1;
700                 printf("cryptodev_digest_init: Open session failed\n");
701                 return (0);
702         }

704         return (1);
705 }

707 static int cryptodev_digest_update(EVP_MD_CTX *ctx, const void *data,
708                 size_t count)
709 {
710         struct crypt_op cryp;
711         struct dev_crypto_state *state = ctx->md_data;
712         struct session_op *sess = &state->d_sess;

714         if (!data || state->d_fd < 0) {
715                 printf("cryptodev_digest_update: illegal inputs \n");
716                 return (0);
717         }

719         if (!count) {
720                 return (0);
721         }
```



```
723         if (!(ctx->flags & EVP_MD_CTX_FLAG_ONESHOT)) {
724                 /* if application doesn't support one buffer */
725                 state->mac_data = OPENSSL_realloc(state->mac_data, state->mac_le

727                 if (!state->mac_data) {
728                         printf("cryptodev_digest_update: realloc failed\n");
729                         return (0);
730                 }

732                 memcpy(state->mac_data + state->mac_len, data, count);
733                 state->mac_len += count;

735                 return (1);
736         }

738         memset(&cryp, 0, sizeof(cryp));

740         cryp.ses = sess->ses;
741         cryp.flags = 0;
742         cryp.len = count;
743         cryp.src = (caddr_t) data;
744         cryp.dst = NULL;
745         cryp.mac = (caddr_t) state->digest_res;
746         if (ioctl(state->d_fd, CIOCCRYPT, &cryp) < 0) {
747                 printf("cryptodev_digest_update: digest failed\n");
748                 return (0);
749         }
750         return (1);
751 }


754 static int cryptodev_digest_final(EVP_MD_CTX *ctx, unsigned char *md)
755 {
756         struct crypt_op cryp;
757         struct dev_crypto_state *state = ctx->md_data;
758         struct session_op *sess = &state->d_sess;

760         int ret = 1;

762         if (!md || state->d_fd < 0) {
763                 printf("cryptodev_digest_final: illegal input\n");
764                 return(0);
765         }

767         if (! (ctx->flags & EVP_MD_CTX_FLAG_ONESHOT) ) {
768                 /* if application doesn't support one buffer */
769                 memset(&cryp, 0, sizeof(cryp));
770                 cryp.ses = sess->ses;
771                 cryp.flags = 0;
772                 cryp.len = state->mac_len;
773                 cryp.src = state->mac_data;
774                 cryp.dst = NULL;
775                 cryp.mac = (caddr_t)md;
776                 if (ioctl(state->d_fd, CIOCCRYPT, &cryp) < 0) {
777                         printf("cryptodev_digest_final: digest failed\n");
778                         return (0);
779                 }

781                 return 1;
782         }

784         memcpy(md, state->digest_res, ctx->digest->md_size);

786         return (ret);
787 }
```

```
790 static int cryptodev_digest_cleanup(EVP_MD_CTX *ctx)
791 {
792         int ret = 1;
793         struct dev_crypto_state *state = ctx->md_data;
794         struct session_op *sess = &state->d_sess;

796         if (state == NULL)
797           return 0;

799         if (state->d_fd < 0) {
800                 printf("cryptodev_digest_cleanup: illegal input\n");
801                 return (0);
802         }

804         if (state->mac_data) {
805                 OPENSSL_free(state->mac_data);
806                 state->mac_data = NULL;
807                 state->mac_len = 0;
808         }

810         if (ioctl(state->d_fd, CIOCFSESSION, &sess->ses) < 0) {
811                 printf("cryptodev_digest_cleanup: failed to close session\n");
812                 ret = 0;
813         } else {
814                 ret = 1;
815         }
816         put_dev_crypto(state->d_fd);
817         state->d_fd = -1;

819         return (ret);
820 }

822 static int cryptodev_digest_copy(EVP_MD_CTX *to,const EVP_MD_CTX *from)
823 {
824         struct dev_crypto_state *fstate = from->md_data;
825         struct dev_crypto_state *dstate = to->md_data;
826         struct session_op *sess;
827         int digest;

829         if (dstate == NULL || fstate == NULL)
830           return 1;

832         memcpy(dstate, fstate, sizeof(struct dev_crypto_state));

834         sess = &dstate->d_sess;

836         digest = digest_nid_to_cryptodev(to->digest->type);

838         sess->mackey = dstate->dummy_mac_key;
839         sess->mackeylen = digest_key_length(to->digest->type);
840         sess->mac = digest;

842         dstate->d_fd = get_dev_crypto();

844         if (ioctl(dstate->d_fd, CIOCGSESSION, sess) < 0) {
845                 put_dev_crypto(dstate->d_fd);
846                 dstate->d_fd = -1;
847                 printf("cryptodev_digest_init: Open session failed\n");
848                 return (0);
849         }

851         if (fstate->mac_len != 0) {
852                 if (fstate->mac_data != NULL)
853                         {
```



```
854                         dstate->mac_data = OPENSSL_malloc(fstate->mac_len);
855                         memcpy(dstate->mac_data, fstate->mac_data, fstate->mac_l
856                         dstate->mac_len = fstate->mac_len;
857                         }
858         }

860         return 1;
861 }


864 const EVP_MD cryptodev_sha1 = {
865         NID_sha1,
866         NID_undef,
867         SHA_DIGEST_LENGTH,
868         EVP_MD_FLAG_ONESHOT,
869         cryptodev_digest_init,
870         cryptodev_digest_update,
871         cryptodev_digest_final,
872         cryptodev_digest_copy,
873         cryptodev_digest_cleanup,
874         EVP_PKEY_NULL_method,
875         SHA_CBLOCK,
876         sizeof(struct dev_crypto_state),
877 };

879 const EVP_MD cryptodev_md5 = {
880         NID_md5,
881         NID_undef,
882         16 /* MD5_DIGEST_LENGTH */,
883         EVP_MD_FLAG_ONESHOT,
884         cryptodev_digest_init,
885         cryptodev_digest_update,
886         cryptodev_digest_final,
887         cryptodev_digest_copy,
888         cryptodev_digest_cleanup,
889         EVP_PKEY_NULL_method,
890         64 /* MD5_CBLOCK */,
891         sizeof(struct dev_crypto_state),
892 };

894 #endif /* USE_CRYPTODEV_DIGESTS */


897 static int
898 cryptodev_engine_digests(ENGINE *e, const EVP_MD **digest,
899     const int **nids, int nid)
900 {
901         if (!digest)
902                 return (cryptodev_usable_digests(nids));

904         switch (nid) {
905 #ifdef USE_CRYPTODEV_DIGESTS
906         case NID_md5:
907                 *digest = &cryptodev_md5;
908                 break;
909         case NID_sha1:
910                 *digest = &cryptodev_sha1;
911                 break;
912         default:
913 #endif /* USE_CRYPTODEV_DIGESTS */
914                 *digest = NULL;
915                 break;
916         }
917         return (*digest != NULL);
918 }
```

```
/*
 * Convert a BIGNUM to the representation that /dev/crypto needs.
 * Upon completion of use, the caller is responsible for freeing
 * crp->crp_p.
 */
static int
bn2crparam(const BIGNUM *a, struct crparam *crp)
{
        int i, j, k;
        ssize_t bytes, bits;
        u_char *b;

        crp->crp_p = NULL;
        crp->crp_nbits = 0;

        bits = BN_num_bits(a);
        bytes = (bits + 7) / 8;

        b = malloc(bytes);
        if (b == NULL)
                return (1);
        memset(b, 0, bytes);

        crp->crp_p = (caddr_t) b;
        crp->crp_nbits = bits;

        for (i = 0, j = 0; i < a->top; i++) {
                for (k = 0; k < BN_BITS2 / 8; k++) {
                        if ((j + k) >= bytes)
                                return (0);
                        b[j + k] = a->d[i] >> (k * 8);
                }
                j += BN_BITS2 / 8;
        }
        return (0);
}

/* Convert a /dev/crypto parameter to a BIGNUM */
static int
crparam2bn(struct crparam *crp, BIGNUM *a)
{
        u_int8_t *pd;
        int i, bytes;

        bytes = (crp->crp_nbits + 7) / 8;

        if (bytes == 0)
                return (-1);

        if ((pd = (u_int8_t *) malloc(bytes)) == NULL)
                return (-1);

        for (i = 0; i < bytes; i++)
                pd[i] = crp->crp_p[bytes - i - 1];

        BN_bin2bn(pd, bytes, a);
        free(pd);

        return (0);
}

static void
zapparams(struct crypt_kop *kop)
{
        int i;
```



```
        for (i = 0; i < kop->crk_iparams + kop->crk_oparams; i++) {
                if (kop->crk_param[i].crp_p)
                        free(kop->crk_param[i].crp_p);
                kop->crk_param[i].crp_p = NULL;
                kop->crk_param[i].crp_nbits = 0;
        }
}

static int
cryptodev_asym(struct crypt_kop *kop, int rlen, BIGNUM *r, int slen, BIGNUM *s)
{
        int fd, ret = -1;

        if ((fd = get_asym_dev_crypto()) < 0)
                return (ret);

        if (r) {
                kop->crk_param[kop->crk_iparams].crp_p = calloc(rlen, sizeof(cha
                kop->crk_param[kop->crk_iparams].crp_nbits = rlen * 8;
                kop->crk_oparams++;
        }
        if (s) {
                kop->crk_param[kop->crk_iparams+1].crp_p = calloc(slen, sizeof(c
                kop->crk_param[kop->crk_iparams+1].crp_nbits = slen * 8;
                kop->crk_oparams++;
        }

        if (ioctl(fd, CIOCKEY, kop) == 0) {
                if (r)
                        crparam2bn(&kop->crk_param[kop->crk_iparams], r);
                if (s)
                        crparam2bn(&kop->crk_param[kop->crk_iparams+1], s);
                ret = 0;
        }

        return (ret);
}

static int
cryptodev_bn_mod_exp(BIGNUM *r, const BIGNUM *a, const BIGNUM *p,
    const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *in_mont)
{
        struct crypt_kop kop;
        int ret = 1;

        /* Currently, we know we can do mod exp iff we can do any
         * asymmetric operations at all.
         */
        if (cryptodev_asymfeat == 0) {
                ret = BN_mod_exp(r, a, p, m, ctx);
                return (ret);
        }

        memset(&kop, 0, sizeof kop);
        kop.crk_op = CRK_MOD_EXP;

        /* inputs: a^p % m */
        if (bn2crparam(a, &kop.crk_param[0]))
                goto err;
        if (bn2crparam(p, &kop.crk_param[1]))
                goto err;
        if (bn2crparam(m, &kop.crk_param[2]))
                goto err;
        kop.crk_iparams = 3;

        if (cryptodev_asym(&kop, BN_num_bytes(m), r, 0, NULL)) {
```

```
1052                const RSA_METHOD *meth = RSA_PKCS1_SSLeay();
1053                printf("OCF asym process failed, Running in software\n");
1054                ret = meth->bn_mod_exp(r, a, p, m, ctx, in_mont);

1056        } else if (ECANCELED == kop.crk_status) {
1057                const RSA_METHOD *meth = RSA_PKCS1_SSLeay();
1058                printf("OCF hardware operation cancelled. Running in Software\n"
1059                ret = meth->bn_mod_exp(r, a, p, m, ctx, in_mont);
1060        }
1061        /* else cryptodev operation worked ok ==> ret = 1*/

1063 err:
1064        zapparams(&kop);
1065        return (ret);
1066 }

1068 static int
1069 cryptodev_rsa_nocrt_mod_exp(BIGNUM *r0, const BIGNUM *I, RSA *rsa, BN_CTX *ctx)
1070 {
1071        int r;
1072        ctx = BN_CTX_new();
1073        r = cryptodev_bn_mod_exp(r0, I, rsa->d, rsa->n, ctx, NULL);
1074        BN_CTX_free(ctx);
1075        return (r);
1076 }

1078 static int
1079 cryptodev_rsa_mod_exp(BIGNUM *r0, const BIGNUM *I, RSA *rsa, BN_CTX *ctx)
1080 {
1081        struct crypt_kop kop;
1082        int ret = 1;

1084        if (!rsa->p || !rsa->q || !rsa->dmp1 || !rsa->dmq1 || !rsa->iqmp) {
1085                /* XXX 0 means failure?? */
1086                return (0);
1087        }

1089        memset(&kop, 0, sizeof kop);
1090        kop.crk_op = CRK_MOD_EXP_CRT;
1091        /* inputs: rsa->p rsa->q I rsa->dmp1 rsa->dmq1 rsa->iqmp */
1092        if (bn2crparam(rsa->p, &kop.crk_param[0]))
1093                goto err;
1094        if (bn2crparam(rsa->q, &kop.crk_param[1]))
1095                goto err;
1096        if (bn2crparam(I, &kop.crk_param[2]))
1097                goto err;
1098        if (bn2crparam(rsa->dmp1, &kop.crk_param[3]))
1099                goto err;
1100        if (bn2crparam(rsa->dmq1, &kop.crk_param[4]))
1101                goto err;
1102        if (bn2crparam(rsa->iqmp, &kop.crk_param[5]))
1103                goto err;
1104        kop.crk_iparams = 6;

1106        if (cryptodev_asym(&kop, BN_num_bytes(rsa->n), r0, 0, NULL)) {
1107                const RSA_METHOD *meth = RSA_PKCS1_SSLeay();
1108                printf("OCF asym process failed, running in Software\n");
1109                ret = (*meth->rsa_mod_exp)(r0, I, rsa, ctx);

1111        } else if (ECANCELED == kop.crk_status) {
1112                const RSA_METHOD *meth = RSA_PKCS1_SSLeay();
1113                printf("OCF hardware operation cancelled. Running in Software\n"
1114                ret = (*meth->rsa_mod_exp)(r0, I, rsa, ctx);
1115        }
1116        /* else cryptodev operation worked ok ==> ret = 1*/
```



```
1118 err:
1119        zapparams(&kop);
1120        return (ret);
1121 }

1123 static RSA_METHOD cryptodev_rsa = {
1124        "cryptodev RSA method",
1125        NULL,                           /* rsa_pub_enc */
1126        NULL,                           /* rsa_pub_dec */
1127        NULL,                           /* rsa_priv_enc */
1128        NULL,                           /* rsa_priv_dec */
1129        NULL,
1130        NULL,
1131        NULL,                           /* init */
1132        NULL,                           /* finish */
1133        0,                              /* flags */
1134        NULL,                           /* app_data */
1135        NULL,                           /* rsa_sign */
1136        NULL                            /* rsa_verify */
1137 };

1139 static int
1140 cryptodev_dsa_bn_mod_exp(DSA *dsa, BIGNUM *r, BIGNUM *a, const BIGNUM *p,
1141     const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *m_ctx)
1142 {
1143        return (cryptodev_bn_mod_exp(r, a, p, m, ctx, m_ctx));
1144 }

1146 static int
1147 cryptodev_dsa_dsa_mod_exp(DSA *dsa, BIGNUM *t1, BIGNUM *g,
1148     BIGNUM *u1, BIGNUM *pub_key, BIGNUM *u2, BIGNUM *p,
1149     BN_CTX *ctx, BN_MONT_CTX *mont)
1150 {
1151        BIGNUM t2;
1152        int ret = 0;

1154        BN_init(&t2);

1156        /* v = ( g^u1 * y^u2 mod p ) mod q */
1157        /* let t1 = g ^ u1 mod p */
1158        ret = 0;

1160        if (!dsa->meth->bn_mod_exp(dsa,t1,dsa->g,u1,dsa->p,ctx,mont))
1161                goto err;

1163        /* let t2 = y ^ u2 mod p */
1164        if (!dsa->meth->bn_mod_exp(dsa,&t2,dsa->pub_key,u2,dsa->p,ctx,mont))
1165                goto err;
1166        /* let u1 = t1 * t2 mod p */
1167        if (!BN_mod_mul(u1,t1,&t2,dsa->p,ctx))
1168                goto err;

1170        BN_copy(t1,u1);

1172        ret = 1;
1173 err:
1174        BN_free(&t2);
1175        return(ret);
1176 }

1178 static DSA_SIG *
1179 cryptodev_dsa_do_sign(const unsigned char *dgst, int dlen, DSA *dsa)
1180 {
1181        struct crypt_kop kop;
1182        BIGNUM *r = NULL, *s = NULL;
1183        DSA_SIG *dsaret = NULL;
```

```
1185         if ((r = BN_new()) == NULL)
1186                 goto err;
1187         if ((s = BN_new()) == NULL) {
1188                 BN_free(r);
1189                 goto err;
1190         }

1192         memset(&kop, 0, sizeof kop);
1193         kop.crk_op = CRK_DSA_SIGN;

1195         /* inputs: dgst dsa->p dsa->q dsa->g dsa->priv_key */
1196         kop.crk_param[0].crp_p = (caddr_t)dgst;
1197         kop.crk_param[0].crp_nbits = dlen * 8;
1198         if (bn2crparam(dsa->p, &kop.crk_param[1]))
1199                 goto err;
1200         if (bn2crparam(dsa->q, &kop.crk_param[2]))
1201                 goto err;
1202         if (bn2crparam(dsa->g, &kop.crk_param[3]))
1203                 goto err;
1204         if (bn2crparam(dsa->priv_key, &kop.crk_param[4]))
1205                 goto err;
1206         kop.crk_iparams = 5;

1208         if (cryptodev_asym(&kop, BN_num_bytes(dsa->q), r,
1209             BN_num_bytes(dsa->q), s) == 0) {
1210                 dsaret = DSA_SIG_new();
1211                 dsaret->r = r;
1212                 dsaret->s = s;
1213         } else {
1214                 const DSA_METHOD *meth = DSA_OpenSSL();
1215                 BN_free(r);
1216                 BN_free(s);
1217                 dsaret = (meth->dsa_do_sign)(dgst, dlen, dsa);
1218         }
1219 err:
1220         kop.crk_param[0].crp_p = NULL;
1221         zapparams(&kop);
1222         return (dsaret);
1223 }

1225 static int
1226 cryptodev_dsa_verify(const unsigned char *dgst, int dlen,
1227     DSA_SIG *sig, DSA *dsa)
1228 {
1229         struct crypt_kop kop;
1230         int dsaret = 1;

1232         memset(&kop, 0, sizeof kop);
1233         kop.crk_op = CRK_DSA_VERIFY;

1235         /* inputs: dgst dsa->p dsa->q dsa->g dsa->pub_key sig->r sig->s */
1236         kop.crk_param[0].crp_p = (caddr_t)dgst;
1237         kop.crk_param[0].crp_nbits = dlen * 8;
1238         if (bn2crparam(dsa->p, &kop.crk_param[1]))
1239                 goto err;
1240         if (bn2crparam(dsa->q, &kop.crk_param[2]))
1241                 goto err;
1242         if (bn2crparam(dsa->g, &kop.crk_param[3]))
1243                 goto err;
1244         if (bn2crparam(dsa->pub_key, &kop.crk_param[4]))
1245                 goto err;
1246         if (bn2crparam(sig->r, &kop.crk_param[5]))
1247                 goto err;
1248         if (bn2crparam(sig->s, &kop.crk_param[6]))
1249                 goto err;
```



```
1250         kop.crk_iparams = 7;

1252         if (cryptodev_asym(&kop, 0, NULL, 0, NULL) == 0) {
1253 /*OCF success value is 0, if not zero, change dsaret to fail*/
1254                 if(0 != kop.crk_status) dsaret  = 0;
1255         } else {
1256                 const DSA_METHOD *meth = DSA_OpenSSL();

1258                 dsaret = (meth->dsa_do_verify)(dgst, dlen, sig, dsa);
1259         }
1260 err:
1261         kop.crk_param[0].crp_p = NULL;
1262         zapparams(&kop);
1263         return (dsaret);
1264 }

1266 static DSA_METHOD cryptodev_dsa = {
1267         "cryptodev DSA method",
1268         NULL,
1269         NULL,                           /* dsa_sign_setup */
1270         NULL,
1271         NULL,                           /* dsa_mod_exp */
1272         NULL,
1273         NULL,                           /* init */
1274         NULL,                           /* finish */
1275         0,      /* flags */
1276         NULL    /* app_data */
1277 };

1279 static int
1280 cryptodev_mod_exp_dh(const DH *dh, BIGNUM *r, const BIGNUM *a,
1281     const BIGNUM *p, const BIGNUM *m, BN_CTX *ctx,
1282     BN_MONT_CTX *m_ctx)
1283 {
1284         return (cryptodev_bn_mod_exp(r, a, p, m, ctx, m_ctx));
1285 }

1287 static int
1288 cryptodev_dh_compute_key(unsigned char *key, const BIGNUM *pub_key, DH *dh)
1289 {
1290         struct crypt_kop kop;
1291         int dhret = 1;
1292         int fd, keylen;

1294         if ((fd = get_asym_dev_crypto()) < 0) {
1295                 const DH_METHOD *meth = DH_OpenSSL();

1297                 return ((meth->compute_key)(key, pub_key, dh));
1298         }

1300         keylen = BN_num_bits(dh->p);

1302         memset(&kop, 0, sizeof kop);
1303         kop.crk_op = CRK_DH_COMPUTE_KEY;

1305         /* inputs: dh->priv_key pub_key dh->p key */
1306         if (bn2crparam(dh->priv_key, &kop.crk_param[0]))
1307                 goto err;
1308         if (bn2crparam(pub_key, &kop.crk_param[1]))
1309                 goto err;
1310         if (bn2crparam(dh->p, &kop.crk_param[2]))
1311                 goto err;
1312         kop.crk_iparams = 3;

1314         kop.crk_param[3].crp_p = (caddr_t) key;
1315         kop.crk_param[3].crp_nbits = keylen * 8;
```

```
1316         kop.crk_oparams = 1;

1318         if (ioctl(fd, CIOCKEY, &kop) == -1) {
1319                 const DH_METHOD *meth = DH_OpenSSL();

1321                 dhret = (meth->compute_key)(key, pub_key, dh);
1322         }
1323 err:
1324         kop.crk_param[3].crp_p = NULL;
1325         zapparams(&kop);
1326         return (dhret);
1327 }

1329 static DH_METHOD cryptodev_dh = {
1330         "cryptodev DH method",
1331         NULL,                           /* cryptodev_dh_generate_key */
1332         NULL,
1333         NULL,
1334         NULL,
1335         NULL,
1336         0,      /* flags */
1337         NULL    /* app_data */
1338 };

1340 /*
1341  * ctrl right now is just a wrapper that doesn't do much
1342  * but I expect we'll want some options soon.
1343  */
1344 static int
1345 cryptodev_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)(void))
1346 {
1347 #ifdef HAVE_SYSLOG_R
1348         struct syslog_data sd = SYSLOG_DATA_INIT;
1349 #endif

1351         switch (cmd) {
1352         default:
1353 #ifdef HAVE_SYSLOG_R
1354                 syslog_r(LOG_ERR, &sd,
1355                     "cryptodev_ctrl: unknown command %d", cmd);
1356 #else
1357                 syslog(LOG_ERR, "cryptodev_ctrl: unknown command %d", cmd);
1358 #endif
1359                 break;
1360         }
1361         return (1);
1362 }

1364 void
1365 ENGINE_load_cryptodev(void)
1366 {
1367         ENGINE *engine = ENGINE_new();
1368         int fd;

1370         if (engine == NULL)
1371                 return;
1372         if ((fd = get_dev_crypto()) < 0) {
1373                 ENGINE_free(engine);
1374                 return;
1375         }

1377         /*
1378          * find out what asymmetric crypto algorithms we support
1379          */
1380         if (ioctl(fd, CIOCASYMFEAT, &cryptodev_asymfeat) == -1) {
1381                 put_dev_crypto(fd);
```



```
1382                 ENGINE_free(engine);
1383                 return;
1384         }
1385         put_dev_crypto(fd);

1387         if (!ENGINE_set_id(engine, "cryptodev") ||
1388             !ENGINE_set_name(engine, "BSD cryptodev engine") ||
1389             !ENGINE_set_ciphers(engine, cryptodev_engine_ciphers) ||
1390             !ENGINE_set_digests(engine, cryptodev_engine_digests) ||
1391             !ENGINE_set_ctrl_function(engine, cryptodev_ctrl) ||
1392             !ENGINE_set_cmd_defns(engine, cryptodev_defns)) {
1393                 ENGINE_free(engine);
1394                 return;
1395         }

1397         if (ENGINE_set_RSA(engine, &cryptodev_rsa)) {
1398                 const RSA_METHOD *rsa_meth = RSA_PKCS1_SSLeay();

1400                 cryptodev_rsa.bn_mod_exp = rsa_meth->bn_mod_exp;
1401                 cryptodev_rsa.rsa_mod_exp = rsa_meth->rsa_mod_exp;
1402                 cryptodev_rsa.rsa_pub_enc = rsa_meth->rsa_pub_enc;
1403                 cryptodev_rsa.rsa_pub_dec = rsa_meth->rsa_pub_dec;
1404                 cryptodev_rsa.rsa_priv_enc = rsa_meth->rsa_priv_enc;
1405                 cryptodev_rsa.rsa_priv_dec = rsa_meth->rsa_priv_dec;
1406                 if (cryptodev_asymfeat & CRF_MOD_EXP) {
1407                         cryptodev_rsa.bn_mod_exp = cryptodev_bn_mod_exp;
1408                         if (cryptodev_asymfeat & CRF_MOD_EXP_CRT)
1409                                 cryptodev_rsa.rsa_mod_exp =
1410                                     cryptodev_rsa_mod_exp;
1411                         else
1412                                 cryptodev_rsa.rsa_mod_exp =
1413                                     cryptodev_rsa_nocrt_mod_exp;
1414                 }
1415         }

1417         if (ENGINE_set_DSA(engine, &cryptodev_dsa)) {
1418                 const DSA_METHOD *meth = DSA_OpenSSL();

1420                 memcpy(&cryptodev_dsa, meth, sizeof(DSA_METHOD));
1421                 if (cryptodev_asymfeat & CRF_DSA_SIGN)
1422                         cryptodev_dsa.dsa_do_sign = cryptodev_dsa_do_sign;
1423                 if (cryptodev_asymfeat & CRF_MOD_EXP) {
1424                         cryptodev_dsa.bn_mod_exp = cryptodev_dsa_bn_mod_exp;
1425                         cryptodev_dsa.dsa_mod_exp = cryptodev_dsa_dsa_mod_exp;
1426                 }
1427                 if (cryptodev_asymfeat & CRF_DSA_VERIFY)
1428                         cryptodev_dsa.dsa_do_verify = cryptodev_dsa_verify;
1429         }

1431         if (ENGINE_set_DH(engine, &cryptodev_dh)){
1432                 const DH_METHOD *dh_meth = DH_OpenSSL();

1434                 cryptodev_dh.generate_key = dh_meth->generate_key;
1435                 cryptodev_dh.compute_key = dh_meth->compute_key;
1436                 cryptodev_dh.bn_mod_exp = dh_meth->bn_mod_exp;
1437                 if (cryptodev_asymfeat & CRF_MOD_EXP) {
1438                         cryptodev_dh.bn_mod_exp = cryptodev_mod_exp_dh;
1439                         if (cryptodev_asymfeat & CRF_DH_COMPUTE_KEY)
1440                                 cryptodev_dh.compute_key =
1441                                     cryptodev_dh_compute_key;
1442                 }
1443         }

1445         ENGINE_add(engine);
1446         ENGINE_free(engine);
1447         ERR_clear_error();
```

```
1448 }

1450 #endif /* HAVE_CRYPTODEV */
1451 #endif /* ! codereview */
```

```
**********************************************************
   12339 Wed Aug 13 19:52:36 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_ctrl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_ctrl.c */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* When querying a ENGINE-specific control command's 'description', this string
 * is used if the ENGINE_CMD_DEFN has cmd_desc set to NULL. */
static const char *int_no_description = "";
```



```
/* These internal functions handle 'CMD'-related control commands when the
 * ENGINE in question has asked us to take care of it (ie. the ENGINE did not
 * set the ENGINE_FLAGS_MANUAL_CMD_CTRL flag. */

static int int_ctrl_cmd_is_null(const ENGINE_CMD_DEFN *defn)
        {
        if((defn->cmd_num == 0) || (defn->cmd_name == NULL))
                return 1;
        return 0;
        }

static int int_ctrl_cmd_by_name(const ENGINE_CMD_DEFN *defn, const char *s)
        {
        int idx = 0;
        while(!int_ctrl_cmd_is_null(defn) && (strcmp(defn->cmd_name, s) != 0))
                {
                idx++;
                defn++;
                }
        if(int_ctrl_cmd_is_null(defn))
                /* The given name wasn't found */
                return -1;
        return idx;
        }

static int int_ctrl_cmd_by_num(const ENGINE_CMD_DEFN *defn, unsigned int num)
        {
        int idx = 0;
        /* NB: It is stipulated that 'cmd_defn' lists are ordered by cmd_num. So
         * our searches don't need to take any longer than necessary. */
        while(!int_ctrl_cmd_is_null(defn) && (defn->cmd_num < num))
                {
                idx++;
                defn++;
                }
        if(defn->cmd_num == num)
                return idx;
        /* The given cmd_num wasn't found */
        return -1;
        }

static int int_ctrl_helper(ENGINE *e, int cmd, long i, void *p,
                           void (*f)(void))
        {
        int idx;
        char *s = (char *)p;
        /* Take care of the easy one first (eg. it requires no searches) */
        if(cmd == ENGINE_CTRL_GET_FIRST_CMD_TYPE)
                {
                if((e->cmd_defns == NULL) || int_ctrl_cmd_is_null(e->cmd_defns))
                        return 0;
                return e->cmd_defns->cmd_num;
                }
        /* One or two commands require that "p" be a valid string buffer */
        if((cmd == ENGINE_CTRL_GET_CMD_FROM_NAME) ||
                        (cmd == ENGINE_CTRL_GET_NAME_FROM_CMD) ||
                        (cmd == ENGINE_CTRL_GET_DESC_FROM_CMD))
                {
                if(s == NULL)
                        {
                        ENGINEerr(ENGINE_F_INT_CTRL_HELPER,
                                ERR_R_PASSED_NULL_PARAMETER);
                        return -1;
                        }
                }
        /* Now handle cmd_name -> cmd_num conversion */
```

```
128         if(cmd == ENGINE_CTRL_GET_CMD_FROM_NAME)
129                 {
130                 if((e->cmd_defns == NULL) || ((idx = int_ctrl_cmd_by_name(
131                                                     e->cmd_defns, s)) < 0))
132                         {
133                         ENGINEerr(ENGINE_F_INT_CTRL_HELPER,
134                                 ENGINE_R_INVALID_CMD_NAME);
135                         return -1;
136                         }
137                 return e->cmd_defns[idx].cmd_num;
138                 }
139         /* For the rest of the commands, the 'long' argument must specify a
140          * valie command number - so we need to conduct a search. */
141         if((e->cmd_defns == NULL) || ((idx = int_ctrl_cmd_by_num(e->cmd_defns,
142                                     (unsigned int)i)) < 0))
143                 {
144                 ENGINEerr(ENGINE_F_INT_CTRL_HELPER,
145                         ENGINE_R_INVALID_CMD_NUMBER);
146                 return -1;
147                 }
148         /* Now the logic splits depending on command type */
149         switch(cmd)
150                 {
151         case ENGINE_CTRL_GET_NEXT_CMD_TYPE:
152                 idx++;
153                 if(int_ctrl_cmd_is_null(e->cmd_defns + idx))
154                         /* end-of-list */
155                         return 0;
156                 else
157                         return e->cmd_defns[idx].cmd_num;
158         case ENGINE_CTRL_GET_NAME_LEN_FROM_CMD:
159                 return strlen(e->cmd_defns[idx].cmd_name);
160         case ENGINE_CTRL_GET_NAME_FROM_CMD:
161                 return BIO_snprintf(s,strlen(e->cmd_defns[idx].cmd_name) + 1,
162                                     "%s", e->cmd_defns[idx].cmd_name);
163         case ENGINE_CTRL_GET_DESC_LEN_FROM_CMD:
164                 if(e->cmd_defns[idx].cmd_desc)
165                         return strlen(e->cmd_defns[idx].cmd_desc);
166                 return strlen(int_no_description);
167         case ENGINE_CTRL_GET_DESC_FROM_CMD:
168                 if(e->cmd_defns[idx].cmd_desc)
169                         return BIO_snprintf(s,
170                                             strlen(e->cmd_defns[idx].cmd_desc) +
171                                             "%s", e->cmd_defns[idx].cmd_desc);
172                 return BIO_snprintf(s, strlen(int_no_description) + 1,"%s",
173                                     int_no_description);
174         case ENGINE_CTRL_GET_CMD_FLAGS:
175                 return e->cmd_defns[idx].cmd_flags;
176                 }
177         /* Shouldn't really be here ... */
178         ENGINEerr(ENGINE_F_INT_CTRL_HELPER,ENGINE_R_INTERNAL_LIST_ERROR);
179         return -1;
180         }

182 int ENGINE_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)(void))
183         {
184         int ctrl_exists, ref_exists;
185         if(e == NULL)
186                 {
187                 ENGINEerr(ENGINE_F_ENGINE_CTRL,ERR_R_PASSED_NULL_PARAMETER);
188                 return 0;
189                 }
190         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
191         ref_exists = ((e->struct_ref > 0) ? 1 : 0);
192         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
193         ctrl_exists = ((e->ctrl == NULL) ? 0 : 1);
```



```
194         if(!ref_exists)
195                 {
196                 ENGINEerr(ENGINE_F_ENGINE_CTRL,ENGINE_R_NO_REFERENCE);
197                 return 0;
198                 }
199         /* Intercept any "root-level" commands before trying to hand them on to
200          * ctrl() handlers. */
201         switch(cmd)
202                 {
203         case ENGINE_CTRL_HAS_CTRL_FUNCTION:
204                 return ctrl_exists;
205         case ENGINE_CTRL_GET_FIRST_CMD_TYPE:
206         case ENGINE_CTRL_GET_NEXT_CMD_TYPE:
207         case ENGINE_CTRL_GET_CMD_FROM_NAME:
208         case ENGINE_CTRL_GET_NAME_LEN_FROM_CMD:
209         case ENGINE_CTRL_GET_NAME_FROM_CMD:
210         case ENGINE_CTRL_GET_DESC_LEN_FROM_CMD:
211         case ENGINE_CTRL_GET_DESC_FROM_CMD:
212         case ENGINE_CTRL_GET_CMD_FLAGS:
213                 if(ctrl_exists && !(e->flags & ENGINE_FLAGS_MANUAL_CMD_CTRL))
214                         return int_ctrl_helper(e,cmd,i,p,f);
215                 if(!ctrl_exists)
216                         {
217                         ENGINEerr(ENGINE_F_ENGINE_CTRL,ENGINE_R_NO_CONTROL_FUNCT
218                         /* For these cmd-related functions, failure is indicated
219                          * by a -1 return value (because 0 is used as a valid
220                          * return in some places). */
221                         return -1;
222                         }
223         default:
224                 break;
225                 }
226         /* Anything else requires a ctrl() handler to exist. */
227         if(!ctrl_exists)
228                 {
229                 ENGINEerr(ENGINE_F_ENGINE_CTRL,ENGINE_R_NO_CONTROL_FUNCTION);
230                 return 0;
231                 }
232         return e->ctrl(e, cmd, i, p, f);
233         }

235 int ENGINE_cmd_is_executable(ENGINE *e, int cmd)
236         {
237         int flags;
238         if((flags = ENGINE_ctrl(e, ENGINE_CTRL_GET_CMD_FLAGS, cmd, NULL, NULL))
239                 {
240                 ENGINEerr(ENGINE_F_ENGINE_CMD_IS_EXECUTABLE,
241                         ENGINE_R_INVALID_CMD_NUMBER);
242                 return 0;
243                 }
244         if(!(flags & ENGINE_CMD_FLAG_NO_INPUT) &&
245                         !(flags & ENGINE_CMD_FLAG_NUMERIC) &&
246                         !(flags & ENGINE_CMD_FLAG_STRING))
247                 return 0;
248         return 1;
249         }

251 int ENGINE_ctrl_cmd(ENGINE *e, const char *cmd_name,
252         long i, void *p, void (*f)(void), int cmd_optional)
253         {
254         int num;

256         if((e == NULL) || (cmd_name == NULL))
257                 {
258                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD,
259                         ERR_R_PASSED_NULL_PARAMETER);
```

```
260                 return 0;
261                 }
262         if((e->ctrl == NULL) || ((num = ENGINE_ctrl(e,
263                                         ENGINE_CTRL_GET_CMD_FROM_NAME,
264                                         0, (void *)cmd_name, NULL)) <= 0))
265                 {
266                 /* If the command didn't *have* to be supported, we fake
267                  * success. This allows certain settings to be specified for
268                  * multiple ENGINEs and only require a change of ENGINE id
269                  * (without having to selectively apply settings). Eg. changing
270                  * from a hardware device back to the regular software ENGINE
271                  * without editing the config file, etc. */
272                 if(cmd_optional)
273                         {
274                         ERR_clear_error();
275                         return 1;
276                         }
277                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD,
278                         ENGINE_R_INVALID_CMD_NAME);
279                 return 0;
280                 }
281         /* Force the result of the control command to 0 or 1, for the reasons
282          * mentioned before. */
283         if (ENGINE_ctrl(e, num, i, p, f) > 0)
284                 return 1;
285         return 0;
286         }

288 int ENGINE_ctrl_cmd_string(ENGINE *e, const char *cmd_name, const char *arg,
289                            int cmd_optional)
290         {
291         int num, flags;
292         long l;
293         char *ptr;
294         if((e == NULL) || (cmd_name == NULL))
295                 {
296                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
297                         ERR_R_PASSED_NULL_PARAMETER);
298                 return 0;
299                 }
300         if((e->ctrl == NULL) || ((num = ENGINE_ctrl(e,
301                                         ENGINE_CTRL_GET_CMD_FROM_NAME,
302                                         0, (void *)cmd_name, NULL)) <= 0))
303                 {
304                 /* If the command didn't *have* to be supported, we fake
305                  * success. This allows certain settings to be specified for
306                  * multiple ENGINEs and only require a change of ENGINE id
307                  * (without having to selectively apply settings). Eg. changing
308                  * from a hardware device back to the regular software ENGINE
309                  * without editing the config file, etc. */
310                 if(cmd_optional)
311                         {
312                         ERR_clear_error();
313                         return 1;
314                         }
315                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
316                         ENGINE_R_INVALID_CMD_NAME);
317                 return 0;
318                 }
319         if(!ENGINE_cmd_is_executable(e, num))
320                 {
321                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
322                         ENGINE_R_CMD_NOT_EXECUTABLE);
323                 return 0;
324                 }
325         if((flags = ENGINE_ctrl(e, ENGINE_CTRL_GET_CMD_FLAGS, num, NULL, NULL))
```



```
326                 {
327                 /* Shouldn't happen, given that ENGINE_cmd_is_executable()
328                  * returned success. */
329                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
330                         ENGINE_R_INTERNAL_LIST_ERROR);
331                 return 0;
332                 }
333         /* If the command takes no input, there must be no input. And vice
334          * versa. */
335         if(flags & ENGINE_CMD_FLAG_NO_INPUT)
336                 {
337                 if(arg != NULL)
338                         {
339                         ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
340                                 ENGINE_R_COMMAND_TAKES_NO_INPUT);
341                         return 0;
342                         }
343                 /* We deliberately force the result of ENGINE_ctrl() to 0 or 1
344                  * rather than returning it as "return data". This is to ensure
345                  * usage of these commands is consistent across applications and
346                  * that certain applications don't understand it one way, and
347                  * others another. */
348                 if(ENGINE_ctrl(e, num, 0, (void *)arg, NULL) > 0)
349                         return 1;
350                 return 0;
351                 }
352         /* So, we require input */
353         if(arg == NULL)
354                 {
355                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
356                         ENGINE_R_COMMAND_TAKES_INPUT);
357                 return 0;
358                 }
359         /* If it takes string input, that's easy */
360         if(flags & ENGINE_CMD_FLAG_STRING)
361                 {
362                 /* Same explanation as above */
363                 if(ENGINE_ctrl(e, num, 0, (void *)arg, NULL) > 0)
364                         return 1;
365                 return 0;
366                 }
367         /* If it doesn't take numeric either, then it is unsupported for use in
368          * a config-setting situation, which is what this function is for. This
369          * should never happen though, because ENGINE_cmd_is_executable() was
370          * used. */
371         if(!(flags & ENGINE_CMD_FLAG_NUMERIC))
372                 {
373                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
374                         ENGINE_R_INTERNAL_LIST_ERROR);
375                 return 0;
376                 }
377         l = strtol(arg, &ptr, 10);
378         if((arg == ptr) || (*ptr != '\0'))
379                 {
380                 ENGINEerr(ENGINE_F_ENGINE_CTRL_CMD_STRING,
381                         ENGINE_R_ARGUMENT_IS_NOT_A_NUMBER);
382                 return 0;
383                 }
384         /* Force the result of the control command to 0 or 1, for the reasons
385          * mentioned before. */
386         if(ENGINE_ctrl(e, num, l, NULL, NULL) > 0)
387                 return 1;
388         return 0;
389         }
390 #endif /* ! codereview */
```

```
**********************************************************
   18038 Wed Aug 13 19:52:36 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_dyn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_dyn.c */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include "eng_int.h"
#include <openssl/dso.h>
```



```
/* Shared libraries implementing ENGINEs for use by the "dynamic" ENGINE loader
 * should implement the hook-up functions with the following prototypes. */

/* Our ENGINE handlers */
static int dynamic_init(ENGINE *e);
static int dynamic_finish(ENGINE *e);
static int dynamic_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)(void));
/* Predeclare our context type */
typedef struct st_dynamic_data_ctx dynamic_data_ctx;
/* The implementation for the important control command */
static int dynamic_load(ENGINE *e, dynamic_data_ctx *ctx);

#define DYNAMIC_CMD_SO_PATH             ENGINE_CMD_BASE
#define DYNAMIC_CMD_NO_VCHECK           (ENGINE_CMD_BASE + 1)
#define DYNAMIC_CMD_ID                  (ENGINE_CMD_BASE + 2)
#define DYNAMIC_CMD_LIST_ADD            (ENGINE_CMD_BASE + 3)
#define DYNAMIC_CMD_DIR_LOAD            (ENGINE_CMD_BASE + 4)
#define DYNAMIC_CMD_DIR_ADD             (ENGINE_CMD_BASE + 5)
#define DYNAMIC_CMD_LOAD                (ENGINE_CMD_BASE + 6)

/* The constants used when creating the ENGINE */
static const char *engine_dynamic_id = "dynamic";
static const char *engine_dynamic_name = "Dynamic engine loading support";
static const ENGINE_CMD_DEFN dynamic_cmd_defns[] = {
        {DYNAMIC_CMD_SO_PATH,
                "SO_PATH",
                "Specifies the path to the new ENGINE shared library",
                ENGINE_CMD_FLAG_STRING},
        {DYNAMIC_CMD_NO_VCHECK,
                "NO_VCHECK",
                "Specifies to continue even if version checking fails (boolean)"
                ENGINE_CMD_FLAG_NUMERIC},
        {DYNAMIC_CMD_ID,
                "ID",
                "Specifies an ENGINE id name for loading",
                ENGINE_CMD_FLAG_STRING},
        {DYNAMIC_CMD_LIST_ADD,
                "LIST_ADD",
                "Whether to add a loaded ENGINE to the internal list (0=no,1=yes
                ENGINE_CMD_FLAG_NUMERIC},
        {DYNAMIC_CMD_DIR_LOAD,
                "DIR_LOAD",
                "Specifies whether to load from 'DIR_ADD' directories (0=no,1=ye
                ENGINE_CMD_FLAG_NUMERIC},
        {DYNAMIC_CMD_DIR_ADD,
                "DIR_ADD",
                "Adds a directory from which ENGINEs can be loaded",
                ENGINE_CMD_FLAG_STRING},
        {DYNAMIC_CMD_LOAD,
                "LOAD",
                "Load up the ENGINE specified by other settings",
                ENGINE_CMD_FLAG_NO_INPUT},
        {0, NULL, NULL, 0}
        };
static const ENGINE_CMD_DEFN dynamic_cmd_defns_empty[] = {
        {0, NULL, NULL, 0}
        };

/* Loading code stores state inside the ENGINE structure via the "ex_data"
 * element. We load all our state into a single structure and use that as a
 * single context in the "ex_data" stack. */
struct st_dynamic_data_ctx
        {
        /* The DSO object we load that supplies the ENGINE code */
        DSO *dynamic_dso;
```

```
128         /* The function pointer to the version checking shared library function
129         dynamic_v_check_fn v_check;
130         /* The function pointer to the engine-binding shared library function */
131         dynamic_bind_engine bind_engine;
132         /* The default name/path for loading the shared library */
133         const char *DYNAMIC_LIBNAME;
134         /* Whether to continue loading on a version check failure */
135         int no_vcheck;
136         /* If non-NULL, stipulates the 'id' of the ENGINE to be loaded */
137         const char *engine_id;
138         /* If non-zero, a successfully loaded ENGINE should be added to the inte
139          * ENGINE list. If 2, the add must succeed or the entire load should fai
140         int list_add_value;
141         /* The symbol name for the version checking function */
142         const char *DYNAMIC_F1;
143         /* The symbol name for the "initialise ENGINE structure" function */
144         const char *DYNAMIC_F2;
145         /* Whether to never use 'dirs', use 'dirs' as a fallback, or only use
146          * 'dirs' for loading. Default is to use 'dirs' as a fallback. */
147         int dir_load;
148         /* A stack of directories from which ENGINEs could be loaded */
149         STACK_OF(OPENSSL_STRING) *dirs;
150         };

152 /* This is the "ex_data" index we obtain and reserve for use with our context
153  * structure. */
154 static int dynamic_ex_data_idx = -1;

156 static void int_free_str(char *s) { OPENSSL_free(s); }
157 /* Because our ex_data element may or may not get allocated depending on whether
158  * a "first-use" occurs before the ENGINE is freed, we have a memory leak
159  * problem to solve. We can't declare a "new" handler for the ex_data as we
160  * don't want a dynamic_data_ctx in *all* ENGINE structures of all types (this
161  * is a bug in the design of CRYPTO_EX_DATA). As such, we just declare a "free"
162  * handler and that will get called if an ENGINE is being destroyed and there
163  * was an ex_data element corresponding to our context type. */
164 static void dynamic_data_ctx_free_func(void *parent, void *ptr,
165                         CRYPTO_EX_DATA *ad, int idx, long argl, void *argp)
166         {
167         if(ptr)
168                 {
169                 dynamic_data_ctx *ctx = (dynamic_data_ctx *)ptr;
170                 if(ctx->dynamic_dso)
171                         DSO_free(ctx->dynamic_dso);
172                 if(ctx->DYNAMIC_LIBNAME)
173                         OPENSSL_free((void*)ctx->DYNAMIC_LIBNAME);
174                 if(ctx->engine_id)
175                         OPENSSL_free((void*)ctx->engine_id);
176                 if(ctx->dirs)
177                         sk_OPENSSL_STRING_pop_free(ctx->dirs, int_free_str);
178                 OPENSSL_free(ctx);
179                 }
180         }

182 /* Construct the per-ENGINE context. We create it blindly and then use a lock to
183  * check for a race - if so, all but one of the threads "racing" will have
184  * wasted their time. The alternative involves creating everything inside the
185  * lock which is far worse. */
186 static int dynamic_set_data_ctx(ENGINE *e, dynamic_data_ctx **ctx)
187         {
188         dynamic_data_ctx *c;
189         c = OPENSSL_malloc(sizeof(dynamic_data_ctx));
190         if(!c)
191                 {
192                 ENGINEerr(ENGINE_F_DYNAMIC_SET_DATA_CTX,ERR_R_MALLOC_FAILURE);
193                 return 0;
```



```
194                 }
195         memset(c, 0, sizeof(dynamic_data_ctx));
196         c->dynamic_dso = NULL;
197         c->v_check = NULL;
198         c->bind_engine = NULL;
199         c->DYNAMIC_LIBNAME = NULL;
200         c->no_vcheck = 0;
201         c->engine_id = NULL;
202         c->list_add_value = 0;
203         c->DYNAMIC_F1 = "v_check";
204         c->DYNAMIC_F2 = "bind_engine";
205         c->dir_load = 1;
206         c->dirs = sk_OPENSSL_STRING_new_null();
207         if(!c->dirs)
208                 {
209                 ENGINEerr(ENGINE_F_DYNAMIC_SET_DATA_CTX,ERR_R_MALLOC_FAILURE);
210                 OPENSSL_free(c);
211                 return 0;
212                 }
213         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
214         if((*ctx = (dynamic_data_ctx *)ENGINE_get_ex_data(e,
215                                 dynamic_ex_data_idx)) == NULL)
216                 {
217                 /* Good, we're the first */
218                 ENGINE_set_ex_data(e, dynamic_ex_data_idx, c);
219                 *ctx = c;
220                 c = NULL;
221                 }
222         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
223         /* If we lost the race to set the context, c is non-NULL and *ctx is the
224          * context of the thread that won. */
225         if(c)
226                 OPENSSL_free(c);
227         return 1;
228         }

230 /* This function retrieves the context structure from an ENGINE's "ex_data", or
231  * if it doesn't exist yet, sets it up. */
232 static dynamic_data_ctx *dynamic_get_data_ctx(ENGINE *e)
233         {
234         dynamic_data_ctx *ctx;
235         if(dynamic_ex_data_idx < 0)
236                 {
237                 /* Create and register the ENGINE ex_data, and associate our
238                  * "free" function with it to ensure any allocated contexts get
239                  * freed when an ENGINE goes underground. */
240                 int new_idx = ENGINE_get_ex_new_index(0, NULL, NULL, NULL,
241                                                 dynamic_data_ctx_free_func);
242                 if(new_idx == -1)
243                         {
244                         ENGINEerr(ENGINE_F_DYNAMIC_GET_DATA_CTX,ENGINE_R_NO_INDE
245                         return NULL;
246                         }
247                 CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
248                 /* Avoid a race by checking again inside this lock */
249                 if(dynamic_ex_data_idx < 0)
250                         {
251                         /* Good, someone didn't beat us to it */
252                         dynamic_ex_data_idx = new_idx;
253                         new_idx = -1;
254                         }
255                 CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
256                 /* In theory we could "give back" the index here if
257                  * (new_idx>-1), but it's not possible and wouldn't gain us much
258                  * if it were. */
259                 }
```

```
260         ctx = (dynamic_data_ctx *)ENGINE_get_ex_data(e, dynamic_ex_data_idx);
261         /* Check if the context needs to be created */
262         if((ctx == NULL) && !dynamic_set_data_ctx(e, &ctx))
263                 /* "set_data" will set errors if necessary */
264                 return NULL;
265         return ctx;
266         }

268 static ENGINE *engine_dynamic(void)
269         {
270         ENGINE *ret = ENGINE_new();
271         if(!ret)
272                 return NULL;
273         if(!ENGINE_set_id(ret, engine_dynamic_id) ||
274                         !ENGINE_set_name(ret, engine_dynamic_name) ||
275                         !ENGINE_set_init_function(ret, dynamic_init) ||
276                         !ENGINE_set_finish_function(ret, dynamic_finish) ||
277                         !ENGINE_set_ctrl_function(ret, dynamic_ctrl) ||
278                         !ENGINE_set_flags(ret, ENGINE_FLAGS_BY_ID_COPY) ||
279                         !ENGINE_set_cmd_defns(ret, dynamic_cmd_defns))
280                 {
281                 ENGINE_free(ret);
282                 return NULL;
283                 }
284         return ret;
285         }

287 void ENGINE_load_dynamic(void)
288         {
289         ENGINE *toadd = engine_dynamic();
290         if(!toadd) return;
291         ENGINE_add(toadd);
292         /* If the "add" worked, it gets a structural reference. So either way,
293          * we release our just-created reference. */
294         ENGINE_free(toadd);
295         /* If the "add" didn't work, it was probably a conflict because it was
296          * already added (eg. someone calling ENGINE_load_blah then calling
297          * ENGINE_load_builtin_engines() perhaps). */
298         ERR_clear_error();
299         }

301 static int dynamic_init(ENGINE *e)
302         {
303         /* We always return failure - the "dyanamic" engine itself can't be used
304          * for anything. */
305         return 0;
306         }

308 static int dynamic_finish(ENGINE *e)
309         {
310         /* This should never be called on account of "dynamic_init" always
311          * failing. */
312         return 0;
313         }

315 static int dynamic_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)(void))
316         {
317         dynamic_data_ctx *ctx = dynamic_get_data_ctx(e);
318         int initialised;

320         if(!ctx)
321                 {
322                 ENGINEerr(ENGINE_F_DYNAMIC_CTRL,ENGINE_R_NOT_LOADED);
323                 return 0;
324                 }
325         initialised = ((ctx->dynamic_dso == NULL) ? 0 : 1);
```



```
326         /* All our control commands require the ENGINE to be uninitialised */
327         if(initialised)
328                 {
329                 ENGINEerr(ENGINE_F_DYNAMIC_CTRL,
330                         ENGINE_R_ALREADY_LOADED);
331                 return 0;
332                 }
333         switch(cmd)
334                 {
335         case DYNAMIC_CMD_SO_PATH:
336                 /* a NULL 'p' or a string of zero-length is the same thing */
337                 if(p && (strlen((const char *)p) < 1))
338                         p = NULL;
339                 if(ctx->DYNAMIC_LIBNAME)
340                         OPENSSL_free((void*)ctx->DYNAMIC_LIBNAME);
341                 if(p)
342                         ctx->DYNAMIC_LIBNAME = BUF_strdup(p);
343                 else
344                         ctx->DYNAMIC_LIBNAME = NULL;
345                 return (ctx->DYNAMIC_LIBNAME ? 1 : 0);
346         case DYNAMIC_CMD_NO_VCHECK:
347                 ctx->no_vcheck = ((i == 0) ? 0 : 1);
348                 return 1;
349         case DYNAMIC_CMD_ID:
350                 /* a NULL 'p' or a string of zero-length is the same thing */
351                 if(p && (strlen((const char *)p) < 1))
352                         p = NULL;
353                 if(ctx->engine_id)
354                         OPENSSL_free((void*)ctx->engine_id);
355                 if(p)
356                         ctx->engine_id = BUF_strdup(p);
357                 else
358                         ctx->engine_id = NULL;
359                 return (ctx->engine_id ? 1 : 0);
360         case DYNAMIC_CMD_LIST_ADD:
361                 if((i < 0) || (i > 2))
362                         {
363                         ENGINEerr(ENGINE_F_DYNAMIC_CTRL,
364                                 ENGINE_R_INVALID_ARGUMENT);
365                         return 0;
366                         }
367                 ctx->list_add_value = (int)i;
368                 return 1;
369         case DYNAMIC_CMD_LOAD:
370                 return dynamic_load(e, ctx);
371         case DYNAMIC_CMD_DIR_LOAD:
372                 if((i < 0) || (i > 2))
373                         {
374                         ENGINEerr(ENGINE_F_DYNAMIC_CTRL,
375                                 ENGINE_R_INVALID_ARGUMENT);
376                         return 0;
377                         }
378                 ctx->dir_load = (int)i;
379                 return 1;
380         case DYNAMIC_CMD_DIR_ADD:
381                 /* a NULL 'p' or a string of zero-length is the same thing */
382                 if(!p || (strlen((const char *)p) < 1))
383                         {
384                         ENGINEerr(ENGINE_F_DYNAMIC_CTRL,
385                                 ENGINE_R_INVALID_ARGUMENT);
386                         return 0;
387                         }
388                 {
389                 char *tmp_str = BUF_strdup(p);
390                 if(!tmp_str)
391                         {
```

```
392                         ENGINEerr(ENGINE_F_DYNAMIC_CTRL,
393                                 ERR_R_MALLOC_FAILURE);
394                         return 0;
395                         }
396                 sk_OPENSSL_STRING_insert(ctx->dirs, tmp_str, -1);
397                 }
398                 return 1;
399         default:
400                 break;
401                 }
402         ENGINEerr(ENGINE_F_DYNAMIC_CTRL,ENGINE_R_CTRL_COMMAND_NOT_IMPLEMENTED);
403         return 0;
404         }

406 static int int_load(dynamic_data_ctx *ctx)
407         {
408         int num, loop;
409         /* Unless told not to, try a direct load */
410         if((ctx->dir_load != 2) && (DSO_load(ctx->dynamic_dso,
411                                 ctx->DYNAMIC_LIBNAME, NULL, 0)) != NULL)
412                 return 1;
413         /* If we're not allowed to use 'dirs' or we have none, fail */
414         if(!ctx->dir_load || (num = sk_OPENSSL_STRING_num(ctx->dirs)) < 1)
415                 return 0;
416         for(loop = 0; loop < num; loop++)
417                 {
418                 const char *s = sk_OPENSSL_STRING_value(ctx->dirs, loop);
419                 char *merge = DSO_merge(ctx->dynamic_dso, ctx->DYNAMIC_LIBNAME,
420                 if(!merge)
421                         return 0;
422                 if(DSO_load(ctx->dynamic_dso, merge, NULL, 0))
423                         {
424                         /* Found what we're looking for */
425                         OPENSSL_free(merge);
426                         return 1;
427                         }
428                 OPENSSL_free(merge);
429                 }
430         return 0;
431         }

433 static int dynamic_load(ENGINE *e, dynamic_data_ctx *ctx)
434         {
435         ENGINE cpy;
436         dynamic_fns fns;

438         if(!ctx->dynamic_dso)
439                 ctx->dynamic_dso = DSO_new();
440         if(!ctx->DYNAMIC_LIBNAME)
441                 {
442                 if(!ctx->engine_id)
443                         return 0;
444                 ctx->DYNAMIC_LIBNAME =
445                         DSO_convert_filename(ctx->dynamic_dso, ctx->engine_id);
446                 }
447         if(!int_load(ctx))
448                 {
449                 ENGINEerr(ENGINE_F_DYNAMIC_LOAD,
450                         ENGINE_R_DSO_NOT_FOUND);
451                 DSO_free(ctx->dynamic_dso);
452                 ctx->dynamic_dso = NULL;
453                 return 0;
454                 }
455         /* We have to find a bind function otherwise it'll always end badly */
456         if(!(ctx->bind_engine = (dynamic_bind_engine)DSO_bind_func(
457                                         ctx->dynamic_dso, ctx->DYNAMIC_F2)))
```



```
458                 {
459                 ctx->bind_engine = NULL;
460                 DSO_free(ctx->dynamic_dso);
461                 ctx->dynamic_dso = NULL;
462                 ENGINEerr(ENGINE_F_DYNAMIC_LOAD,
463                         ENGINE_R_DSO_FAILURE);
464                 return 0;
465                 }
466         /* Do we perform version checking? */
467         if(!ctx->no_vcheck)
468                 {
469                 unsigned long vcheck_res = 0;
470                 /* Now we try to find a version checking function and decide how
471                  * to cope with failure if/when it fails. */
472                 ctx->v_check = (dynamic_v_check_fn)DSO_bind_func(
473                                 ctx->dynamic_dso, ctx->DYNAMIC_F1);
474                 if(ctx->v_check)
475                         vcheck_res = ctx->v_check(OSSL_DYNAMIC_VERSION);
476                 /* We fail if the version checker veto'd the load *or* if it is
477                  * deferring to us (by returning its version) and we think it is
478                  * too old. */
479                 if(vcheck_res < OSSL_DYNAMIC_OLDEST)
480                         {
481                         /* Fail */
482                         ctx->bind_engine = NULL;
483                         ctx->v_check = NULL;
484                         DSO_free(ctx->dynamic_dso);
485                         ctx->dynamic_dso = NULL;
486                         ENGINEerr(ENGINE_F_DYNAMIC_LOAD,
487                                 ENGINE_R_VERSION_INCOMPATIBILITY);
488                         return 0;
489                         }
490                 }
491         /* First binary copy the ENGINE structure so that we can roll back if
492          * the hand-over fails */
493         memcpy(&cpy, e, sizeof(ENGINE));
494         /* Provide the ERR, "ex_data", memory, and locking callbacks so the
495          * loaded library uses our state rather than its own. FIXME: As noted in
496          * engine.h, much of this would be simplified if each area of code
497          * provided its own "summary" structure of all related callbacks. It
498          * would also increase opaqueness. */
499         fns.static_state = ENGINE_get_static_state();
500         fns.err_fns = ERR_get_implementation();
501         fns.ex_data_fns = CRYPTO_get_ex_data_implementation();
502         CRYPTO_get_mem_functions(&fns.mem_fns.malloc_cb,
503                                 &fns.mem_fns.realloc_cb,
504                                 &fns.mem_fns.free_cb);
505         fns.lock_fns.lock_locking_cb = CRYPTO_get_locking_callback();
506         fns.lock_fns.lock_add_lock_cb = CRYPTO_get_add_lock_callback();
507         fns.lock_fns.dynlock_create_cb = CRYPTO_get_dynlock_create_callback();
508         fns.lock_fns.dynlock_lock_cb = CRYPTO_get_dynlock_lock_callback();
509         fns.lock_fns.dynlock_destroy_cb = CRYPTO_get_dynlock_destroy_callback();
510         /* Now that we've loaded the dynamic engine, make sure no "dynamic"
511          * ENGINE elements will show through. */
512         engine_set_all_null(e);

514         /* Try to bind the ENGINE onto our own ENGINE structure */
515         if(!ctx->bind_engine(e, ctx->engine_id, &fns))
516                 {
517                 ctx->bind_engine = NULL;
518                 ctx->v_check = NULL;
519                 DSO_free(ctx->dynamic_dso);
520                 ctx->dynamic_dso = NULL;
521                 ENGINEerr(ENGINE_F_DYNAMIC_LOAD,ENGINE_R_INIT_FAILED);
522                 /* Copy the original ENGINE structure back */
523                 memcpy(e, &cpy, sizeof(ENGINE));
```

```
524                 return 0;
525                 }
526         /* Do we try to add this ENGINE to the internal list too? */
527         if(ctx->list_add_value > 0)
528                 {
529                 if(!ENGINE_add(e))
530                         {
531                         /* Do we tolerate this or fail? */
532                         if(ctx->list_add_value > 1)
533                                 {
534                                 /* Fail - NB: By this time, it's too late to
535                                  * rollback, and trying to do so allows the
536                                  * bind_engine() code to have created leaks. We
537                                  * just have to fail where we are, after the
538                                  * ENGINE has changed. */
539                                 ENGINEerr(ENGINE_F_DYNAMIC_LOAD,
540                                         ENGINE_R_CONFLICTING_ENGINE_ID);
541                                 return 0;
542                                 }
543                         /* Tolerate */
544                         ERR_clear_error();
545                         }
546                 }
547         return 1;
548         }
549 #endif /* ! codereview */
```

```
**********************************************************
    8619 Wed Aug 13 19:52:36 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_err.c */
/* ====================================================================
 * Copyright (c) 1999-2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/engine.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_ENGINE,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_ENGINE,0,reason)

static ERR_STRING_DATA ENGINE_str_functs[]=
        {
{ERR_FUNC(ENGINE_F_DYNAMIC_CTRL),       "DYNAMIC_CTRL"},
{ERR_FUNC(ENGINE_F_DYNAMIC_GET_DATA_CTX),       "DYNAMIC_GET_DATA_CTX"},
{ERR_FUNC(ENGINE_F_DYNAMIC_LOAD),       "DYNAMIC_LOAD"},
{ERR_FUNC(ENGINE_F_DYNAMIC_SET_DATA_CTX),       "DYNAMIC_SET_DATA_CTX"},
{ERR_FUNC(ENGINE_F_ENGINE_ADD), "ENGINE_add"},
{ERR_FUNC(ENGINE_F_ENGINE_BY_ID),       "ENGINE_by_id"},
{ERR_FUNC(ENGINE_F_ENGINE_CMD_IS_EXECUTABLE),   "ENGINE_cmd_is_executable"},
{ERR_FUNC(ENGINE_F_ENGINE_CTRL),        "ENGINE_ctrl"},
{ERR_FUNC(ENGINE_F_ENGINE_CTRL_CMD),    "ENGINE_ctrl_cmd"},
{ERR_FUNC(ENGINE_F_ENGINE_CTRL_CMD_STRING),     "ENGINE_ctrl_cmd_string"},
{ERR_FUNC(ENGINE_F_ENGINE_FINISH),      "ENGINE_finish"},
{ERR_FUNC(ENGINE_F_ENGINE_FREE_UTIL),   "ENGINE_FREE_UTIL"},
{ERR_FUNC(ENGINE_F_ENGINE_GET_CIPHER),  "ENGINE_get_cipher"},
{ERR_FUNC(ENGINE_F_ENGINE_GET_DEFAULT_TYPE),    "ENGINE_GET_DEFAULT_TYPE"},
{ERR_FUNC(ENGINE_F_ENGINE_GET_DIGEST),  "ENGINE_get_digest"},
{ERR_FUNC(ENGINE_F_ENGINE_GET_NEXT),    "ENGINE_get_next"},
{ERR_FUNC(ENGINE_F_ENGINE_GET_PKEY_ASN1_METH),  "ENGINE_get_pkey_asn1_meth"},
{ERR_FUNC(ENGINE_F_ENGINE_GET_PKEY_METH),       "ENGINE_get_pkey_meth"},
{ERR_FUNC(ENGINE_F_ENGINE_GET_PREV),    "ENGINE_get_prev"},
{ERR_FUNC(ENGINE_F_ENGINE_INIT),        "ENGINE_init"},
{ERR_FUNC(ENGINE_F_ENGINE_LIST_ADD),    "ENGINE_LIST_ADD"},
{ERR_FUNC(ENGINE_F_ENGINE_LIST_REMOVE), "ENGINE_LIST_REMOVE"},
{ERR_FUNC(ENGINE_F_ENGINE_LOAD_PRIVATE_KEY),    "ENGINE_load_private_key"},
{ERR_FUNC(ENGINE_F_ENGINE_LOAD_PUBLIC_KEY),     "ENGINE_load_public_key"},
{ERR_FUNC(ENGINE_F_ENGINE_LOAD_SSL_CLIENT_CERT),        "ENGINE_load_ssl_client_
{ERR_FUNC(ENGINE_F_ENGINE_NEW), "ENGINE_new"},
{ERR_FUNC(ENGINE_F_ENGINE_REMOVE),      "ENGINE_remove"},
{ERR_FUNC(ENGINE_F_ENGINE_SET_DEFAULT_STRING),  "ENGINE_set_default_string"},
{ERR_FUNC(ENGINE_F_ENGINE_SET_DEFAULT_TYPE),    "ENGINE_SET_DEFAULT_TYPE"},
{ERR_FUNC(ENGINE_F_ENGINE_SET_ID),      "ENGINE_set_id"},
{ERR_FUNC(ENGINE_F_ENGINE_SET_NAME),    "ENGINE_set_name"},
{ERR_FUNC(ENGINE_F_ENGINE_TABLE_REGISTER),      "ENGINE_TABLE_REGISTER"},
{ERR_FUNC(ENGINE_F_ENGINE_UNLOAD_KEY),  "ENGINE_UNLOAD_KEY"},
{ERR_FUNC(ENGINE_F_ENGINE_UNLOCKED_FINISH),     "ENGINE_UNLOCKED_FINISH"},
{ERR_FUNC(ENGINE_F_ENGINE_UP_REF),      "ENGINE_up_ref"},
{ERR_FUNC(ENGINE_F_INT_CTRL_HELPER),    "INT_CTRL_HELPER"},
{ERR_FUNC(ENGINE_F_INT_ENGINE_CONFIGURE),       "INT_ENGINE_CONFIGURE"},
{ERR_FUNC(ENGINE_F_INT_ENGINE_MODULE_INIT),     "INT_ENGINE_MODULE_INIT"},
{ERR_FUNC(ENGINE_F_LOG_MESSAGE),        "LOG_MESSAGE"},
{0,NULL}
        };

static ERR_STRING_DATA ENGINE_str_reasons[]=
        {
{ERR_REASON(ENGINE_R_ALREADY_LOADED)     ,"already loaded"},
{ERR_REASON(ENGINE_R_ARGUMENT_IS_NOT_A_NUMBER),"argument is not a number"},
{ERR_REASON(ENGINE_R_CMD_NOT_EXECUTABLE) ,"cmd not executable"},
{ERR_REASON(ENGINE_R_COMMAND_TAKES_INPUT),"command takes input"},
{ERR_REASON(ENGINE_R_COMMAND_TAKES_NO_INPUT),"command takes no input"},
{ERR_REASON(ENGINE_R_CONFLICTING_ENGINE_ID),"conflicting engine id"},
{ERR_REASON(ENGINE_R_CTRL_COMMAND_NOT_IMPLEMENTED),"ctrl command not implemented
{ERR_REASON(ENGINE_R_DH_NOT_IMPLEMENTED) ,"dh not implemented"},
{ERR_REASON(ENGINE_R_DSA_NOT_IMPLEMENTED),"dsa not implemented"},
{ERR_REASON(ENGINE_R_DSO_FAILURE)        ,"DSO failure"},
{ERR_REASON(ENGINE_R_DSO_NOT_FOUND)      ,"dso not found"},
```

```
{ERR_REASON(ENGINE_R_ENGINES_SECTION_ERROR),"engines section error"},
{ERR_REASON(ENGINE_R_ENGINE_CONFIGURATION_ERROR),"engine configuration error"},
{ERR_REASON(ENGINE_R_ENGINE_IS_NOT_IN_LIST),"engine is not in the list"},
{ERR_REASON(ENGINE_R_ENGINE_SECTION_ERROR),"engine section error"},
{ERR_REASON(ENGINE_R_FAILED_LOADING_PRIVATE_KEY),"failed loading private key"},
{ERR_REASON(ENGINE_R_FAILED_LOADING_PUBLIC_KEY),"failed loading public key"},
{ERR_REASON(ENGINE_R_FINISH_FAILED)      ,"finish failed"},
{ERR_REASON(ENGINE_R_GET_HANDLE_FAILED)  ,"could not obtain hardware handle"},
{ERR_REASON(ENGINE_R_ID_OR_NAME_MISSING) ,"'id' or 'name' missing"},
{ERR_REASON(ENGINE_R_INIT_FAILED)        ,"init failed"},
{ERR_REASON(ENGINE_R_INTERNAL_LIST_ERROR),"internal list error"},
{ERR_REASON(ENGINE_R_INVALID_ARGUMENT)   ,"invalid argument"},
{ERR_REASON(ENGINE_R_INVALID_CMD_NAME)   ,"invalid cmd name"},
{ERR_REASON(ENGINE_R_INVALID_CMD_NUMBER) ,"invalid cmd number"},
{ERR_REASON(ENGINE_R_INVALID_INIT_VALUE) ,"invalid init value"},
{ERR_REASON(ENGINE_R_INVALID_STRING)     ,"invalid string"},
{ERR_REASON(ENGINE_R_NOT_INITIALISED)    ,"not initialised"},
{ERR_REASON(ENGINE_R_NOT_LOADED)         ,"not loaded"},
{ERR_REASON(ENGINE_R_NO_CONTROL_FUNCTION),"no control function"},
{ERR_REASON(ENGINE_R_NO_INDEX)           ,"no index"},
{ERR_REASON(ENGINE_R_NO_LOAD_FUNCTION)   ,"no load function"},
{ERR_REASON(ENGINE_R_NO_REFERENCE)       ,"no reference"},
{ERR_REASON(ENGINE_R_NO_SUCH_ENGINE)     ,"no such engine"},
{ERR_REASON(ENGINE_R_NO_UNLOAD_FUNCTION) ,"no unload function"},
{ERR_REASON(ENGINE_R_PROVIDE_PARAMETERS) ,"provide parameters"},
{ERR_REASON(ENGINE_R_RSA_NOT_IMPLEMENTED),"rsa not implemented"},
{ERR_REASON(ENGINE_R_UNIMPLEMENTED_CIPHER),"unimplemented cipher"},
{ERR_REASON(ENGINE_R_UNIMPLEMENTED_DIGEST),"unimplemented digest"},
{ERR_REASON(ENGINE_R_UNIMPLEMENTED_PUBLIC_KEY_METHOD),"unimplemented public key
{ERR_REASON(ENGINE_R_VERSION_INCOMPATIBILITY),"version incompatibility"},
{0,NULL}
        };

#endif

void ERR_load_ENGINE_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(ENGINE_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,ENGINE_str_functs);
                ERR_load_strings(0,ENGINE_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    6089 Wed Aug 13 19:52:36 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_fat.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_fat.c */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */

#include "eng_int.h"
```



```
#include <openssl/conf.h>

int ENGINE_set_default(ENGINE *e, unsigned int flags)
        {
        if((flags & ENGINE_METHOD_CIPHERS) && !ENGINE_set_default_ciphers(e))
                return 0;
        if((flags & ENGINE_METHOD_DIGESTS) && !ENGINE_set_default_digests(e))
                return 0;
#ifndef OPENSSL_NO_RSA
        if((flags & ENGINE_METHOD_RSA) && !ENGINE_set_default_RSA(e))
                return 0;
#endif
#ifndef OPENSSL_NO_DSA
        if((flags & ENGINE_METHOD_DSA) && !ENGINE_set_default_DSA(e))
                return 0;
#endif
#ifndef OPENSSL_NO_DH
        if((flags & ENGINE_METHOD_DH) && !ENGINE_set_default_DH(e))
                return 0;
#endif
#ifndef OPENSSL_NO_ECDH
        if((flags & ENGINE_METHOD_ECDH) && !ENGINE_set_default_ECDH(e))
                return 0;
#endif
#ifndef OPENSSL_NO_ECDSA
        if((flags & ENGINE_METHOD_ECDSA) && !ENGINE_set_default_ECDSA(e))
                return 0;
#endif
        if((flags & ENGINE_METHOD_RAND) && !ENGINE_set_default_RAND(e))
                return 0;
        if((flags & ENGINE_METHOD_PKEY_METHS)
                                && !ENGINE_set_default_pkey_meths(e))
                return 0;
        if((flags & ENGINE_METHOD_PKEY_ASN1_METHS)
                                && !ENGINE_set_default_pkey_asn1_meths(e))
                return 0;
        return 1;
        }

/* Set default algorithms using a string */

static int int_def_cb(const char *alg, int len, void *arg)
        {
        unsigned int *pflags = arg;
        if (!strncmp(alg, "ALL", len))
                *pflags |= ENGINE_METHOD_ALL;
        else if (!strncmp(alg, "RSA", len))
                *pflags |= ENGINE_METHOD_RSA;
        else if (!strncmp(alg, "DSA", len))
                *pflags |= ENGINE_METHOD_DSA;
        else if (!strncmp(alg, "ECDH", len))
                *pflags |= ENGINE_METHOD_ECDH;
        else if (!strncmp(alg, "ECDSA", len))
                *pflags |= ENGINE_METHOD_ECDSA;
        else if (!strncmp(alg, "DH", len))
                *pflags |= ENGINE_METHOD_DH;
        else if (!strncmp(alg, "RAND", len))
                *pflags |= ENGINE_METHOD_RAND;
        else if (!strncmp(alg, "CIPHERS", len))
                *pflags |= ENGINE_METHOD_CIPHERS;
        else if (!strncmp(alg, "DIGESTS", len))
                *pflags |= ENGINE_METHOD_DIGESTS;
        else if (!strncmp(alg, "PKEY", len))
                *pflags |=
                        ENGINE_METHOD_PKEY_METHS|ENGINE_METHOD_PKEY_ASN1_METHS;
        else if (!strncmp(alg, "PKEY_CRYPTO", len))
```

```
128                 *pflags |= ENGINE_METHOD_PKEY_METHS;
129         else if (!strncmp(alg, "PKEY_ASN1", len))
130                 *pflags |= ENGINE_METHOD_PKEY_ASN1_METHS;
131         else
132                 return 0;
133         return 1;
134         }


137 int ENGINE_set_default_string(ENGINE *e, const char *def_list)
138         {
139         unsigned int flags = 0;
140         if (!CONF_parse_list(def_list, ',', 1, int_def_cb, &flags))
141                 {
142                 ENGINEerr(ENGINE_F_ENGINE_SET_DEFAULT_STRING,
143                                         ENGINE_R_INVALID_STRING);
144                 ERR_add_error_data(2, "str=",def_list);
145                 return 0;
146                 }
147         return ENGINE_set_default(e, flags);
148         }

150 int ENGINE_register_complete(ENGINE *e)
151         {
152         ENGINE_register_ciphers(e);
153         ENGINE_register_digests(e);
154 #ifndef OPENSSL_NO_RSA
155         ENGINE_register_RSA(e);
156 #endif
157 #ifndef OPENSSL_NO_DSA
158         ENGINE_register_DSA(e);
159 #endif
160 #ifndef OPENSSL_NO_DH
161         ENGINE_register_DH(e);
162 #endif
163 #ifndef OPENSSL_NO_ECDH
164         ENGINE_register_ECDH(e);
165 #endif
166 #ifndef OPENSSL_NO_ECDSA
167         ENGINE_register_ECDSA(e);
168 #endif
169         ENGINE_register_RAND(e);
170         ENGINE_register_pkey_meths(e);
171         return 1;
172         }

174 int ENGINE_register_all_complete(void)
175         {
176         ENGINE *e;

178         for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
179                 if (!(e->flags & ENGINE_FLAGS_NO_REGISTER_ALL))
180                         ENGINE_register_complete(e);
181         return 1;
182         }
183 #endif /* ! codereview */
```

```
**********************************************************
    5199 Wed Aug 13 19:52:36 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_init.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_init.c */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* Initialise a engine type for use (or up its functional reference count
 * if it's already in use). This version is only used internally. */
int engine_unlocked_init(ENGINE *e)
        {
```



```
        int to_return = 1;

        if((e->funct_ref == 0) && e->init)
                /* This is the first functional reference and the engine
                 * requires initialisation so we do it now. */
                to_return = e->init(e);
        if(to_return)
                {
                /* OK, we return a functional reference which is also a
                 * structural reference. */
                e->struct_ref++;
                e->funct_ref++;
                engine_ref_debug(e, 0, 1)
                engine_ref_debug(e, 1, 1)
                }
        return to_return;
        }

/* Free a functional reference to a engine type. This version is only used
 * internally. */
int engine_unlocked_finish(ENGINE *e, int unlock_for_handlers)
        {
        int to_return = 1;

        /* Reduce the functional reference count here so if it's the terminating
         * case, we can release the lock safely and call the finish() handler
         * without risk of a race. We get a race if we leave the count until
         * after and something else is calling "finish" at the same time -
         * there's a chance that both threads will together take the count from
         * 2 to 0 without either calling finish(). */
        e->funct_ref--;
        engine_ref_debug(e, 1, -1);
        if((e->funct_ref == 0) && e->finish)
                {
                if(unlock_for_handlers)
                        CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
                to_return = e->finish(e);
                if(unlock_for_handlers)
                        CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
                if(!to_return)
                        return 0;
                }
#ifdef REF_CHECK
        if(e->funct_ref < 0)
                {
                fprintf(stderr,"ENGINE_finish, bad functional reference count\n"
                abort();
                }
#endif
        /* Release the structural reference too */
        if(!engine_free_util(e, 0))
                {
                ENGINEerr(ENGINE_F_ENGINE_UNLOCKED_FINISH,ENGINE_R_FINISH_FAILED
                return 0;
                }
        return to_return;
        }

/* The API (locked) version of "init" */
int ENGINE_init(ENGINE *e)
        {
        int ret;
        if(e == NULL)
                {
                ENGINEerr(ENGINE_F_ENGINE_INIT,ERR_R_PASSED_NULL_PARAMETER);
                return 0;
```

```
128                 }
129         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
130         ret = engine_unlocked_init(e);
131         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
132         return ret;
133         }

135 /* The API (locked) version of "finish" */
136 int ENGINE_finish(ENGINE *e)
137         {
138         int to_return = 1;

140         if(e == NULL)
141                 {
142                 ENGINEerr(ENGINE_F_ENGINE_FINISH,ERR_R_PASSED_NULL_PARAMETER);
143                 return 0;
144                 }
145         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
146         to_return = engine_unlocked_finish(e, 1);
147         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
148         if(!to_return)
149                 {
150                 ENGINEerr(ENGINE_F_ENGINE_FINISH,ENGINE_R_FINISH_FAILED);
151                 return 0;
152                 }
153         return to_return;
154         }
155 #endif /* ! codereview */
```

```
**********************************************************
    8788 Wed Aug 13 19:52:37 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_lib.c */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"
#include <openssl/rand.h>
```



```
/* The "new"/"free" stuff first */

ENGINE *ENGINE_new(void)
        {
        ENGINE *ret;

        ret = (ENGINE *)OPENSSL_malloc(sizeof(ENGINE));
        if(ret == NULL)
                {
                ENGINEerr(ENGINE_F_ENGINE_NEW, ERR_R_MALLOC_FAILURE);
                return NULL;
                }
        memset(ret, 0, sizeof(ENGINE));
        ret->struct_ref = 1;
        engine_ref_debug(ret, 0, 1)
        CRYPTO_new_ex_data(CRYPTO_EX_INDEX_ENGINE, ret, &ret->ex_data);
        return ret;
        }

/* Placed here (close proximity to ENGINE_new) so that modifications to the
 * elements of the ENGINE structure are more likely to be caught and changed
 * here. */
void engine_set_all_null(ENGINE *e)
        {
        e->id = NULL;
        e->name = NULL;
        e->rsa_meth = NULL;
        e->dsa_meth = NULL;
        e->dh_meth = NULL;
        e->rand_meth = NULL;
        e->store_meth = NULL;
        e->ciphers = NULL;
        e->digests = NULL;
        e->destroy = NULL;
        e->init = NULL;
        e->finish = NULL;
        e->ctrl = NULL;
        e->load_privkey = NULL;
        e->load_pubkey = NULL;
        e->cmd_defns = NULL;
        e->flags = 0;
        }

int engine_free_util(ENGINE *e, int locked)
        {
        int i;

        if(e == NULL)
                {
                ENGINEerr(ENGINE_F_ENGINE_FREE_UTIL,
                          ERR_R_PASSED_NULL_PARAMETER);
                return 0;
                }
        if(locked)
                i = CRYPTO_add(&e->struct_ref,-1,CRYPTO_LOCK_ENGINE);
        else
                i = --e->struct_ref;
        engine_ref_debug(e, 0, -1)
        if (i > 0) return 1;
#ifdef REF_CHECK
        if (i < 0)
                {
                fprintf(stderr,"ENGINE_free, bad structural reference count\n");
                abort();
                }
#endif
```

```
128         /* Free up any dynamically allocated public key methods */
129         engine_pkey_meths_free(e);
130         engine_pkey_asn1_meths_free(e);
131         /* Give the ENGINE a chance to do any structural cleanup corresponding
132          * to allocation it did in its constructor (eg. unload error strings) */
133         if(e->destroy)
134                 e->destroy(e);
135         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_ENGINE, e, &e->ex_data);
136         OPENSSL_free(e);
137         return 1;
138         }

140 int ENGINE_free(ENGINE *e)
141         {
142         return engine_free_util(e, 1);
143         }

145 /* Cleanup stuff */

147 /* ENGINE_cleanup() is coded such that anything that does work that will need
148  * cleanup can register a "cleanup" callback here. That way we don't get linker
149  * bloat by referring to all *possible* cleanups, but any linker bloat into code
150  * "X" will cause X's cleanup function to end up here. */
151 static STACK_OF(ENGINE_CLEANUP_ITEM) *cleanup_stack = NULL;
152 static int int_cleanup_check(int create)
153         {
154         if(cleanup_stack) return 1;
155         if(!create) return 0;
156         cleanup_stack = sk_ENGINE_CLEANUP_ITEM_new_null();
157         return (cleanup_stack ? 1 : 0);
158         }
159 static ENGINE_CLEANUP_ITEM *int_cleanup_item(ENGINE_CLEANUP_CB *cb)
160         {
161         ENGINE_CLEANUP_ITEM *item = OPENSSL_malloc(sizeof(
162                                         ENGINE_CLEANUP_ITEM));
163         if(!item) return NULL;
164         item->cb = cb;
165         return item;
166         }
167 void engine_cleanup_add_first(ENGINE_CLEANUP_CB *cb)
168         {
169         ENGINE_CLEANUP_ITEM *item;
170         if(!int_cleanup_check(1)) return;
171         item = int_cleanup_item(cb);
172         if(item)
173                 sk_ENGINE_CLEANUP_ITEM_insert(cleanup_stack, item, 0);
174         }
175 void engine_cleanup_add_last(ENGINE_CLEANUP_CB *cb)
176         {
177         ENGINE_CLEANUP_ITEM *item;
178         if(!int_cleanup_check(1)) return;
179         item = int_cleanup_item(cb);
180         if(item)
181                 sk_ENGINE_CLEANUP_ITEM_push(cleanup_stack, item);
182         }
183 /* The API function that performs all cleanup */
184 static void engine_cleanup_cb_free(ENGINE_CLEANUP_ITEM *item)
185         {
186         (*(item->cb))();
187         OPENSSL_free(item);
188         }
189 void ENGINE_cleanup(void)
190         {
191         if(int_cleanup_check(0))
192                 {
193                 sk_ENGINE_CLEANUP_ITEM_pop_free(cleanup_stack,
```



```
194                         engine_cleanup_cb_free);
195                 cleanup_stack = NULL;
196                 }
197         /* FIXME: This should be handled (somehow) through RAND, eg. by it
198          * registering a cleanup callback. */
199         RAND_set_rand_method(NULL);
200         }

202 /* Now the "ex_data" support */

204 int ENGINE_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
205                 CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
206         {
207         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_ENGINE, argl, argp,
208                         new_func, dup_func, free_func);
209         }

211 int ENGINE_set_ex_data(ENGINE *e, int idx, void *arg)
212         {
213         return(CRYPTO_set_ex_data(&e->ex_data, idx, arg));
214         }

216 void *ENGINE_get_ex_data(const ENGINE *e, int idx)
217         {
218         return(CRYPTO_get_ex_data(&e->ex_data, idx));
219         }

221 /* Functions to get/set an ENGINE's elements - mainly to avoid exposing the
222  * ENGINE structure itself. */

224 int ENGINE_set_id(ENGINE *e, const char *id)
225         {
226         if(id == NULL)
227                 {
228                 ENGINEerr(ENGINE_F_ENGINE_SET_ID,
229                         ERR_R_PASSED_NULL_PARAMETER);
230                 return 0;
231                 }
232         e->id = id;
233         return 1;
234         }

236 int ENGINE_set_name(ENGINE *e, const char *name)
237         {
238         if(name == NULL)
239                 {
240                 ENGINEerr(ENGINE_F_ENGINE_SET_NAME,
241                         ERR_R_PASSED_NULL_PARAMETER);
242                 return 0;
243                 }
244         e->name = name;
245         return 1;
246         }

248 int ENGINE_set_destroy_function(ENGINE *e, ENGINE_GEN_INT_FUNC_PTR destroy_f)
249         {
250         e->destroy = destroy_f;
251         return 1;
252         }

254 int ENGINE_set_init_function(ENGINE *e, ENGINE_GEN_INT_FUNC_PTR init_f)
255         {
256         e->init = init_f;
257         return 1;
258         }
```

```
260 int ENGINE_set_finish_function(ENGINE *e, ENGINE_GEN_INT_FUNC_PTR finish_f)
261         {
262         e->finish = finish_f;
263         return 1;
264         }

266 int ENGINE_set_ctrl_function(ENGINE *e, ENGINE_CTRL_FUNC_PTR ctrl_f)
267         {
268         e->ctrl = ctrl_f;
269         return 1;
270         }

272 int ENGINE_set_flags(ENGINE *e, int flags)
273         {
274         e->flags = flags;
275         return 1;
276         }

278 int ENGINE_set_cmd_defns(ENGINE *e, const ENGINE_CMD_DEFN *defns)
279         {
280         e->cmd_defns = defns;
281         return 1;
282         }

284 const char *ENGINE_get_id(const ENGINE *e)
285         {
286         return e->id;
287         }

289 const char *ENGINE_get_name(const ENGINE *e)
290         {
291         return e->name;
292         }

294 ENGINE_GEN_INT_FUNC_PTR ENGINE_get_destroy_function(const ENGINE *e)
295         {
296         return e->destroy;
297         }

299 ENGINE_GEN_INT_FUNC_PTR ENGINE_get_init_function(const ENGINE *e)
300         {
301         return e->init;
302         }

304 ENGINE_GEN_INT_FUNC_PTR ENGINE_get_finish_function(const ENGINE *e)
305         {
306         return e->finish;
307         }

309 ENGINE_CTRL_FUNC_PTR ENGINE_get_ctrl_function(const ENGINE *e)
310         {
311         return e->ctrl;
312         }

314 int ENGINE_get_flags(const ENGINE *e)
315         {
316         return e->flags;
317         }

319 const ENGINE_CMD_DEFN *ENGINE_get_cmd_defns(const ENGINE *e)
320         {
321         return e->cmd_defns;
322         }

324 /* eng_lib.o is pretty much linked into anything that touches ENGINE already, so
325  * put the "static_state" hack here. */
```



```
327 static int internal_static_hack = 0;

329 void *ENGINE_get_static_state(void)
330         {
331         return &internal_static_hack;
332         }
333 #endif /* ! codereview */
```

```
**********************************************************
   11839 Wed Aug 13 19:52:37 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_list.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_list.c */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
```



```
 */

#include "eng_int.h"

/* The linked-list of pointers to engine types. engine_list_head
 * incorporates an implicit structural reference but engine_list_tail
 * does not - the latter is a computational niceity and only points
 * to something that is already pointed to by its predecessor in the
 * list (or engine_list_head itself). In the same way, the use of the
 * "prev" pointer in each ENGINE is to save excessive list iteration,
 * it doesn't correspond to an extra structural reference. Hence,
 * engine_list_head, and each non-null "next" pointer account for
 * the list itself assuming exactly 1 structural reference on each
 * list member. */
static ENGINE *engine_list_head = NULL;
static ENGINE *engine_list_tail = NULL;

/* This cleanup function is only needed internally. If it should be called, we
 * register it with the "ENGINE_cleanup()" stack to be called during cleanup. */

static void engine_list_cleanup(void)
        {
        ENGINE *iterator = engine_list_head;

        while(iterator != NULL)
                {
                ENGINE_remove(iterator);
                iterator = engine_list_head;
                }
        return;
        }

/* These static functions starting with a lower case "engine_" always
 * take place when CRYPTO_LOCK_ENGINE has been locked up. */
static int engine_list_add(ENGINE *e)
        {
        int conflict = 0;
        ENGINE *iterator = NULL;

        if(e == NULL)
                {
                ENGINEerr(ENGINE_F_ENGINE_LIST_ADD,
                        ERR_R_PASSED_NULL_PARAMETER);
                return 0;
                }
        iterator = engine_list_head;
        while(iterator && !conflict)
                {
                conflict = (strcmp(iterator->id, e->id) == 0);
                iterator = iterator->next;
                }
        if(conflict)
                {
                ENGINEerr(ENGINE_F_ENGINE_LIST_ADD,
                        ENGINE_R_CONFLICTING_ENGINE_ID);
                return 0;
                }
        if(engine_list_head == NULL)
                {
                /* We are adding to an empty list. */
                if(engine_list_tail)
                        {
                        ENGINEerr(ENGINE_F_ENGINE_LIST_ADD,
                                ENGINE_R_INTERNAL_LIST_ERROR);
                        return 0;
                        }
```

```
128                 engine_list_head = e;
129                 e->prev = NULL;
130                 /* The first time the list allocates, we should register the
131                  * cleanup. */
132                 engine_cleanup_add_last(engine_list_cleanup);
133                 }
134         else
135                 {
136                 /* We are adding to the tail of an existing list. */
137                 if((engine_list_tail == NULL) ||
138                                 (engine_list_tail->next != NULL))
139                         {
140                         ENGINEerr(ENGINE_F_ENGINE_LIST_ADD,
141                                 ENGINE_R_INTERNAL_LIST_ERROR);
142                         return 0;
143                         }
144                 engine_list_tail->next = e;
145                 e->prev = engine_list_tail;
146                 }
147         /* Having the engine in the list assumes a structural
148          * reference. */
149         e->struct_ref++;
150         engine_ref_debug(e, 0, 1)
151         /* However it came to be, e is the last item in the list. */
152         engine_list_tail = e;
153         e->next = NULL;
154         return 1;
155         }

157 static int engine_list_remove(ENGINE *e)
158         {
159         ENGINE *iterator;

161         if(e == NULL)
162                 {
163                 ENGINEerr(ENGINE_F_ENGINE_LIST_REMOVE,
164                         ERR_R_PASSED_NULL_PARAMETER);
165                 return 0;
166                 }
167         /* We need to check that e is in our linked list! */
168         iterator = engine_list_head;
169         while(iterator && (iterator != e))
170                 iterator = iterator->next;
171         if(iterator == NULL)
172                 {
173                 ENGINEerr(ENGINE_F_ENGINE_LIST_REMOVE,
174                         ENGINE_R_ENGINE_IS_NOT_IN_LIST);
175                 return 0;
176                 }
177         /* un-link e from the chain. */
178         if(e->next)
179                 e->next->prev = e->prev;
180         if(e->prev)
181                 e->prev->next = e->next;
182         /* Correct our head/tail if necessary. */
183         if(engine_list_head == e)
184                 engine_list_head = e->next;
185         if(engine_list_tail == e)
186                 engine_list_tail = e->prev;
187         engine_free_util(e, 0);
188         return 1;
189         }

191 /* Get the first/last "ENGINE" type available. */
192 ENGINE *ENGINE_get_first(void)
193         {
```



```
194         ENGINE *ret;

196         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
197         ret = engine_list_head;
198         if(ret)
199                 {
200                 ret->struct_ref++;
201                 engine_ref_debug(ret, 0, 1)
202                 }
203         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
204         return ret;
205         }

207 ENGINE *ENGINE_get_last(void)
208         {
209         ENGINE *ret;

211         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
212         ret = engine_list_tail;
213         if(ret)
214                 {
215                 ret->struct_ref++;
216                 engine_ref_debug(ret, 0, 1)
217                 }
218         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
219         return ret;
220         }

222 /* Iterate to the next/previous "ENGINE" type (NULL = end of the list). */
223 ENGINE *ENGINE_get_next(ENGINE *e)
224         {
225         ENGINE *ret = NULL;
226         if(e == NULL)
227                 {
228                 ENGINEerr(ENGINE_F_ENGINE_GET_NEXT,
229                         ERR_R_PASSED_NULL_PARAMETER);
230                 return 0;
231                 }
232         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
233         ret = e->next;
234         if(ret)
235                 {
236                 /* Return a valid structural refernce to the next ENGINE */
237                 ret->struct_ref++;
238                 engine_ref_debug(ret, 0, 1)
239                 }
240         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
241         /* Release the structural reference to the previous ENGINE */
242         ENGINE_free(e);
243         return ret;
244         }

246 ENGINE *ENGINE_get_prev(ENGINE *e)
247         {
248         ENGINE *ret = NULL;
249         if(e == NULL)
250                 {
251                 ENGINEerr(ENGINE_F_ENGINE_GET_PREV,
252                         ERR_R_PASSED_NULL_PARAMETER);
253                 return 0;
254                 }
255         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
256         ret = e->prev;
257         if(ret)
258                 {
259                 /* Return a valid structural reference to the next ENGINE */
```

```
260                 ret->struct_ref++;
261                 engine_ref_debug(ret, 0, 1)
262                 }
263         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
264         /* Release the structural reference to the previous ENGINE */
265         ENGINE_free(e);
266         return ret;
267         }

269 /* Add another "ENGINE" type into the list. */
270 int ENGINE_add(ENGINE *e)
271         {
272         int to_return = 1;
273         if(e == NULL)
274                 {
275                 ENGINEerr(ENGINE_F_ENGINE_ADD,
276                         ERR_R_PASSED_NULL_PARAMETER);
277                 return 0;
278                 }
279         if((e->id == NULL) || (e->name == NULL))
280                 {
281                 ENGINEerr(ENGINE_F_ENGINE_ADD,
282                         ENGINE_R_ID_OR_NAME_MISSING);
283                 }
284         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
285         if(!engine_list_add(e))
286                 {
287                 ENGINEerr(ENGINE_F_ENGINE_ADD,
288                         ENGINE_R_INTERNAL_LIST_ERROR);
289                 to_return = 0;
290                 }
291         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
292         return to_return;
293         }

295 /* Remove an existing "ENGINE" type from the array. */
296 int ENGINE_remove(ENGINE *e)
297         {
298         int to_return = 1;
299         if(e == NULL)
300                 {
301                 ENGINEerr(ENGINE_F_ENGINE_REMOVE,
302                         ERR_R_PASSED_NULL_PARAMETER);
303                 return 0;
304                 }
305         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
306         if(!engine_list_remove(e))
307                 {
308                 ENGINEerr(ENGINE_F_ENGINE_REMOVE,
309                         ENGINE_R_INTERNAL_LIST_ERROR);
310                 to_return = 0;
311                 }
312         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
313         return to_return;
314         }

316 static void engine_cpy(ENGINE *dest, const ENGINE *src)
317         {
318         dest->id = src->id;
319         dest->name = src->name;
320 #ifndef OPENSSL_NO_RSA
321         dest->rsa_meth = src->rsa_meth;
322 #endif
323 #ifndef OPENSSL_NO_DSA
324         dest->dsa_meth = src->dsa_meth;
325 #endif
```



```
326 #ifndef OPENSSL_NO_DH
327         dest->dh_meth = src->dh_meth;
328 #endif
329 #ifndef OPENSSL_NO_ECDH
330         dest->ecdh_meth = src->ecdh_meth;
331 #endif
332 #ifndef OPENSSL_NO_ECDSA
333         dest->ecdsa_meth = src->ecdsa_meth;
334 #endif
335         dest->rand_meth = src->rand_meth;
336         dest->store_meth = src->store_meth;
337         dest->ciphers = src->ciphers;
338         dest->digests = src->digests;
339         dest->pkey_meths = src->pkey_meths;
340         dest->destroy = src->destroy;
341         dest->init = src->init;
342         dest->finish = src->finish;
343         dest->ctrl = src->ctrl;
344         dest->load_privkey = src->load_privkey;
345         dest->load_pubkey = src->load_pubkey;
346         dest->cmd_defns = src->cmd_defns;
347         dest->flags = src->flags;
348         }

350 ENGINE *ENGINE_by_id(const char *id)
351         {
352         ENGINE *iterator;
353         char *load_dir = NULL;
354         if(id == NULL)
355                 {
356                 ENGINEerr(ENGINE_F_ENGINE_BY_ID,
357                         ERR_R_PASSED_NULL_PARAMETER);
358                 return NULL;
359                 }
360         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
361         iterator = engine_list_head;
362         while(iterator && (strcmp(id, iterator->id) != 0))
363                 iterator = iterator->next;
364         if(iterator)
365                 {
366                 /* We need to return a structural reference. If this is an
367                  * ENGINE type that returns copies, make a duplicate - otherwise
368                  * increment the existing ENGINE's reference count. */
369                 if(iterator->flags & ENGINE_FLAGS_BY_ID_COPY)
370                         {
371                         ENGINE *cp = ENGINE_new();
372                         if(!cp)
373                                 iterator = NULL;
374                         else
375                                 {
376                                 engine_cpy(cp, iterator);
377                                 iterator = cp;
378                                 }
379                         }
380                 else
381                         {
382                         iterator->struct_ref++;
383                         engine_ref_debug(iterator, 0, 1)
384                         }
385                 }
386         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
387 #if 0
388         if(iterator == NULL)
389                 {
390                 ENGINEerr(ENGINE_F_ENGINE_BY_ID,
391                         ENGINE_R_NO_SUCH_ENGINE);
```

```
392                 ERR_add_error_data(2, "id=", id);
393                 }
394         return iterator;
395 #else
396         /* EEK! Experimental code starts */
397         if(iterator) return iterator;
398         /* Prevent infinite recusrion if we're looking for the dynamic engine. *
399         if (strcmp(id, "dynamic"))
400                 {
401 #ifdef OPENSSL_SYS_VMS
402                 if((load_dir = getenv("OPENSSL_ENGINES")) == 0) load_dir = "SSLR
403 #else
404                 if((load_dir = getenv("OPENSSL_ENGINES")) == 0) load_dir = ENGIN
405 #endif
406                 iterator = ENGINE_by_id("dynamic");
407                 if(!iterator || !ENGINE_ctrl_cmd_string(iterator, "ID", id, 0) |
408                                 !ENGINE_ctrl_cmd_string(iterator, "DIR_LOAD", "2
409                                 !ENGINE_ctrl_cmd_string(iterator, "DIR_ADD",
410                                         load_dir, 0) ||
411                                 !ENGINE_ctrl_cmd_string(iterator, "LIST_ADD", "1
412                                 !ENGINE_ctrl_cmd_string(iterator, "LOAD", NULL,
413                                 goto notfound;
414                 return iterator;
415                 }
416 notfound:
417         ENGINE_free(iterator);
418         ENGINEerr(ENGINE_F_ENGINE_BY_ID,ENGINE_R_NO_SUCH_ENGINE);
419         ERR_add_error_data(2, "id=", id);
420         return NULL;
421         /* EEK! Experimental code ends */
422 #endif
423         }

425 int ENGINE_up_ref(ENGINE *e)
426         {
427         if (e == NULL)
428                 {
429                 ENGINEerr(ENGINE_F_ENGINE_UP_REF,ERR_R_PASSED_NULL_PARAMETER);
430                 return 0;
431                 }
432         CRYPTO_add(&e->struct_ref,1,CRYPTO_LOCK_ENGINE);
433         return 1;
434         }
435 #endif /* ! codereview */
```

```
**********************************************************
   11628 Wed Aug 13 19:52:37 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_openssl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_openssl.c */
/* Written by Geoff Thorpe (geoff@geoffthorpe.net) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECDH support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
```



```
 */


#include <stdio.h>
#include <openssl/crypto.h>
#include "cryptlib.h"
#include <openssl/engine.h>
#include <openssl/dso.h>
#include <openssl/pem.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif

/* This testing gunk is implemented (and explained) lower down. It also assumes
 * the application explicitly calls "ENGINE_load_openssl()" because this is no
 * longer automatic in ENGINE_load_builtin_engines(). */
#define TEST_ENG_OPENSSL_RC4
#define TEST_ENG_OPENSSL_PKEY
/* #define TEST_ENG_OPENSSL_RC4_OTHERS */
#define TEST_ENG_OPENSSL_RC4_P_INIT
/* #define TEST_ENG_OPENSSL_RC4_P_CIPHER */
#define TEST_ENG_OPENSSL_SHA
/* #define TEST_ENG_OPENSSL_SHA_OTHERS */
/* #define TEST_ENG_OPENSSL_SHA_P_INIT */
/* #define TEST_ENG_OPENSSL_SHA_P_UPDATE */
/* #define TEST_ENG_OPENSSL_SHA_P_FINAL */

/* Now check what of those algorithms are actually enabled */
#ifdef OPENSSL_NO_RC4
#undef TEST_ENG_OPENSSL_RC4
#undef TEST_ENG_OPENSSL_RC4_OTHERS
#undef TEST_ENG_OPENSSL_RC4_P_INIT
#undef TEST_ENG_OPENSSL_RC4_P_CIPHER
#endif
#if defined(OPENSSL_NO_SHA) || defined(OPENSSL_NO_SHA0) || defined(OPENSSL_NO_SH
#undef TEST_ENG_OPENSSL_SHA
#undef TEST_ENG_OPENSSL_SHA_OTHERS
#undef TEST_ENG_OPENSSL_SHA_P_INIT
#undef TEST_ENG_OPENSSL_SHA_P_UPDATE
#undef TEST_ENG_OPENSSL_SHA_P_FINAL
#endif

#ifdef TEST_ENG_OPENSSL_RC4
static int openssl_ciphers(ENGINE *e, const EVP_CIPHER **cipher,
                           const int **nids, int nid);
#endif
#ifdef TEST_ENG_OPENSSL_SHA
static int openssl_digests(ENGINE *e, const EVP_MD **digest,
                           const int **nids, int nid);
#endif

#ifdef TEST_ENG_OPENSSL_PKEY
static EVP_PKEY *openssl_load_privkey(ENGINE *eng, const char *key_id,
        UI_METHOD *ui_method, void *callback_data);
#endif

/* The constants used when creating the ENGINE */
static const char *engine_openssl_id = "openssl";
```

```
128 static const char *engine_openssl_name = "Software engine support";

130 /* This internal function is used by ENGINE_openssl() and possibly by the
131  * "dynamic" ENGINE support too */
132 static int bind_helper(ENGINE *e)
133         {
134         if(!ENGINE_set_id(e, engine_openssl_id)
135                         || !ENGINE_set_name(e, engine_openssl_name)
136 #ifndef TEST_ENG_OPENSSL_NO_ALGORITHMS
137 #ifndef OPENSSL_NO_RSA
138                         || !ENGINE_set_RSA(e, RSA_get_default_method())
139 #endif
140 #ifndef OPENSSL_NO_DSA
141                         || !ENGINE_set_DSA(e, DSA_get_default_method())
142 #endif
143 #ifndef OPENSSL_NO_ECDH
144                         || !ENGINE_set_ECDH(e, ECDH_OpenSSL())
145 #endif
146 #ifndef OPENSSL_NO_ECDSA
147                         || !ENGINE_set_ECDSA(e, ECDSA_OpenSSL())
148 #endif
149 #ifndef OPENSSL_NO_DH
150                         || !ENGINE_set_DH(e, DH_get_default_method())
151 #endif
152                         || !ENGINE_set_RAND(e, RAND_SSLeay())
153 #ifdef TEST_ENG_OPENSSL_RC4
154                         || !ENGINE_set_ciphers(e, openssl_ciphers)
155 #endif
156 #ifdef TEST_ENG_OPENSSL_SHA
157                         || !ENGINE_set_digests(e, openssl_digests)
158 #endif
159 #endif
160 #ifdef TEST_ENG_OPENSSL_PKEY
161                         || !ENGINE_set_load_privkey_function(e, openssl_load_pri
162 #endif
163                         )
164                 return 0;
165         /* If we add errors to this ENGINE, ensure the error handling is setup h
166         /* openssl_load_error_strings(); */
167         return 1;
168         }

170 static ENGINE *engine_openssl(void)
171         {
172         ENGINE *ret = ENGINE_new();
173         if(!ret)
174                 return NULL;
175         if(!bind_helper(ret))
176                 {
177                 ENGINE_free(ret);
178                 return NULL;
179                 }
180         return ret;
181         }

183 void ENGINE_load_openssl(void)
184         {
185         ENGINE *toadd = engine_openssl();
186         if(!toadd) return;
187         ENGINE_add(toadd);
188         /* If the "add" worked, it gets a structural reference. So either way,
189          * we release our just-created reference. */
190         ENGINE_free(toadd);
191         ERR_clear_error();
192         }
```



```
194 /* This stuff is needed if this ENGINE is being compiled into a self-contained
195  * shared-library. */
196 #ifdef ENGINE_DYNAMIC_SUPPORT
197 static int bind_fn(ENGINE *e, const char *id)
198         {
199         if(id && (strcmp(id, engine_openssl_id) != 0))
200                 return 0;
201         if(!bind_helper(e))
202                 return 0;
203         return 1;
204         }
205 IMPLEMENT_DYNAMIC_CHECK_FN()
206 IMPLEMENT_DYNAMIC_BIND_FN(bind_fn)
207 #endif /* ENGINE_DYNAMIC_SUPPORT */

209 #ifdef TEST_ENG_OPENSSL_RC4
210 /* This section of code compiles an "alternative implementation" of two modes of
211  * RC4 into this ENGINE. The result is that EVP_CIPHER operation for "rc4"
212  * should under normal circumstances go via this support rather than the default
213  * EVP support. There are other symbols to tweak the testing;
214  *    TEST_ENC_OPENSSL_RC4_OTHERS - print a one line message to stderr each time
215  *        we're asked for a cipher we don't support (should not happen).
216  *    TEST_ENG_OPENSSL_RC4_P_INIT - print a one line message to stderr each time
217  *        the "init_key" handler is called.
218  *    TEST_ENG_OPENSSL_RC4_P_CIPHER - ditto for the "cipher" handler.
219  */
220 #include <openssl/rc4.h>
221 #define TEST_RC4_KEY_SIZE               16
222 static int test_cipher_nids[] = {NID_rc4,NID_rc4_40};
223 static int test_cipher_nids_number = 2;
224 typedef struct {
225         unsigned char key[TEST_RC4_KEY_SIZE];
226         RC4_KEY ks;
227         } TEST_RC4_KEY;
228 #define test(ctx) ((TEST_RC4_KEY *)(ctx)->cipher_data)
229 static int test_rc4_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
230                         const unsigned char *iv, int enc)
231         {
232 #ifdef TEST_ENG_OPENSSL_RC4_P_INIT
233         fprintf(stderr, "(TEST_ENG_OPENSSL_RC4) test_init_key() called\n");
234 #endif
235         memcpy(&test(ctx)->key[0],key,EVP_CIPHER_CTX_key_length(ctx));
236         RC4_set_key(&test(ctx)->ks,EVP_CIPHER_CTX_key_length(ctx),
237                 test(ctx)->key);
238         return 1;
239         }
240 static int test_rc4_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
241                         const unsigned char *in, size_t inl)
242         {
243 #ifdef TEST_ENG_OPENSSL_RC4_P_CIPHER
244         fprintf(stderr, "(TEST_ENG_OPENSSL_RC4) test_cipher() called\n");
245 #endif
246         RC4(&test(ctx)->ks,inl,in,out);
247         return 1;
248         }
249 static const EVP_CIPHER test_r4_cipher=
250         {
251         NID_rc4,
252         1,TEST_RC4_KEY_SIZE,0,
253         EVP_CIPH_VARIABLE_LENGTH,
254         test_rc4_init_key,
255         test_rc4_cipher,
256         NULL,
257         sizeof(TEST_RC4_KEY),
258         NULL,
259         NULL,
```

```
260         NULL,
261         NULL
262         };
263 static const EVP_CIPHER test_r4_40_cipher=
264         {
265         NID_rc4_40,
266         1,5 /* 40 bit */,0,
267         EVP_CIPH_VARIABLE_LENGTH,
268         test_rc4_init_key,
269         test_rc4_cipher,
270         NULL,
271         sizeof(TEST_RC4_KEY),
272         NULL,
273         NULL,
274         NULL,
275         NULL
276         };
277 static int openssl_ciphers(ENGINE *e, const EVP_CIPHER **cipher,
278                         const int **nids, int nid)
279         {
280         if(!cipher)
281                 {
282                 /* We are returning a list of supported nids */
283                 *nids = test_cipher_nids;
284                 return test_cipher_nids_number;
285                 }
286         /* We are being asked for a specific cipher */
287         if(nid == NID_rc4)
288                 *cipher = &test_r4_cipher;
289         else if(nid == NID_rc4_40)
290                 *cipher = &test_r4_40_cipher;
291         else
292                 {
293 #ifdef TEST_ENG_OPENSSL_RC4_OTHERS
294                 fprintf(stderr, "(TEST_ENG_OPENSSL_RC4) returning NULL for "
295                                 "nid %d\n", nid);
296 #endif
297                 *cipher = NULL;
298                 return 0;
299                 }
300         return 1;
301         }
302 #endif

304 #ifdef TEST_ENG_OPENSSL_SHA
305 /* Much the same sort of comment as for TEST_ENG_OPENSSL_RC4 */
306 #include <openssl/sha.h>
307 static int test_digest_nids[] = {NID_sha1};
308 static int test_digest_nids_number = 1;
309 static int test_sha1_init(EVP_MD_CTX *ctx)
310         {
311 #ifdef TEST_ENG_OPENSSL_SHA_P_INIT
312         fprintf(stderr, "(TEST_ENG_OPENSSL_SHA) test_sha1_init() called\n");
313 #endif
314         return SHA1_Init(ctx->md_data);
315         }
316 static int test_sha1_update(EVP_MD_CTX *ctx,const void *data,size_t count)
317         {
318 #ifdef TEST_ENG_OPENSSL_SHA_P_UPDATE
319         fprintf(stderr, "(TEST_ENG_OPENSSL_SHA) test_sha1_update() called\n");
320 #endif
321         return SHA1_Update(ctx->md_data,data,count);
322         }
323 static int test_sha1_final(EVP_MD_CTX *ctx,unsigned char *md)
324         {
325 #ifdef TEST_ENG_OPENSSL_SHA_P_FINAL
```



```
326         fprintf(stderr, "(TEST_ENG_OPENSSL_SHA) test_sha1_final() called\n");
327 #endif
328         return SHA1_Final(md,ctx->md_data);
329         }
330 static const EVP_MD test_sha_md=
331         {
332         NID_sha1,
333         NID_sha1WithRSAEncryption,
334         SHA_DIGEST_LENGTH,
335         0,
336         test_sha1_init,
337         test_sha1_update,
338         test_sha1_final,
339         NULL,
340         NULL,
341         EVP_PKEY_RSA_method,
342         SHA_CBLOCK,
343         sizeof(EVP_MD *)+sizeof(SHA_CTX),
344         };
345 static int openssl_digests(ENGINE *e, const EVP_MD **digest,
346                         const int **nids, int nid)
347         {
348         if(!digest)
349                 {
350                 /* We are returning a list of supported nids */
351                 *nids = test_digest_nids;
352                 return test_digest_nids_number;
353                 }
354         /* We are being asked for a specific digest */
355         if(nid == NID_sha1)
356                 *digest = &test_sha_md;
357         else
358                 {
359 #ifdef TEST_ENG_OPENSSL_SHA_OTHERS
360                 fprintf(stderr, "(TEST_ENG_OPENSSL_SHA) returning NULL for "
361                                 "nid %d\n", nid);
362 #endif
363                 *digest = NULL;
364                 return 0;
365                 }
366         return 1;
367         }
368 #endif

370 #ifdef TEST_ENG_OPENSSL_PKEY
371 static EVP_PKEY *openssl_load_privkey(ENGINE *eng, const char *key_id,
372         UI_METHOD *ui_method, void *callback_data)
373         {
374         BIO *in;
375         EVP_PKEY *key;
376         fprintf(stderr, "(TEST_ENG_OPENSSL_PKEY)Loading Private key %s\n", key_i
377         in = BIO_new_file(key_id, "r");
378         if (!in)
379                 return NULL;
380         key = PEM_read_bio_PrivateKey(in, NULL, 0, NULL);
381         BIO_free(in);
382         return key;
383         }
384 #endif
385 #endif /* ! codereview */
```

```
**********************************************************
    5845 Wed Aug 13 19:52:37 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_pkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_pkey.c */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* Basic get/set stuff */

int ENGINE_set_load_privkey_function(ENGINE *e, ENGINE_LOAD_KEY_PTR loadpriv_f)
        {
```



```
        e->load_privkey = loadpriv_f;
        return 1;
        }

int ENGINE_set_load_pubkey_function(ENGINE *e, ENGINE_LOAD_KEY_PTR loadpub_f)
        {
        e->load_pubkey = loadpub_f;
        return 1;
        }

int ENGINE_set_load_ssl_client_cert_function(ENGINE *e,
                                ENGINE_SSL_CLIENT_CERT_PTR loadssl_f)
        {
        e->load_ssl_client_cert = loadssl_f;
        return 1;
        }

ENGINE_LOAD_KEY_PTR ENGINE_get_load_privkey_function(const ENGINE *e)
        {
        return e->load_privkey;
        }

ENGINE_LOAD_KEY_PTR ENGINE_get_load_pubkey_function(const ENGINE *e)
        {
        return e->load_pubkey;
        }

ENGINE_SSL_CLIENT_CERT_PTR ENGINE_get_ssl_client_cert_function(const ENGINE *e)
        {
        return e->load_ssl_client_cert;
        }

/* API functions to load public/private keys */

EVP_PKEY *ENGINE_load_private_key(ENGINE *e, const char *key_id,
        UI_METHOD *ui_method, void *callback_data)
        {
        EVP_PKEY *pkey;

        if(e == NULL)
                {
                ENGINEerr(ENGINE_F_ENGINE_LOAD_PRIVATE_KEY,
                        ERR_R_PASSED_NULL_PARAMETER);
                return 0;
                }
        CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
        if(e->funct_ref == 0)
                {
                CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
                ENGINEerr(ENGINE_F_ENGINE_LOAD_PRIVATE_KEY,
                        ENGINE_R_NOT_INITIALISED);
                return 0;
                }
        CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
        if (!e->load_privkey)
                {
                ENGINEerr(ENGINE_F_ENGINE_LOAD_PRIVATE_KEY,
                        ENGINE_R_NO_LOAD_FUNCTION);
                return 0;
                }
        pkey = e->load_privkey(e, key_id, ui_method, callback_data);
        if (!pkey)
                {
                ENGINEerr(ENGINE_F_ENGINE_LOAD_PRIVATE_KEY,
                        ENGINE_R_FAILED_LOADING_PRIVATE_KEY);
                return 0;
```

```
128                 }
129         return pkey;
130         }

132 EVP_PKEY *ENGINE_load_public_key(ENGINE *e, const char *key_id,
133         UI_METHOD *ui_method, void *callback_data)
134         {
135         EVP_PKEY *pkey;

137         if(e == NULL)
138                 {
139                 ENGINEerr(ENGINE_F_ENGINE_LOAD_PUBLIC_KEY,
140                         ERR_R_PASSED_NULL_PARAMETER);
141                 return 0;
142                 }
143         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
144         if(e->funct_ref == 0)
145                 {
146                 CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
147                 ENGINEerr(ENGINE_F_ENGINE_LOAD_PUBLIC_KEY,
148                         ENGINE_R_NOT_INITIALISED);
149                 return 0;
150                 }
151         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
152         if (!e->load_pubkey)
153                 {
154                 ENGINEerr(ENGINE_F_ENGINE_LOAD_PUBLIC_KEY,
155                         ENGINE_R_NO_LOAD_FUNCTION);
156                 return 0;
157                 }
158         pkey = e->load_pubkey(e, key_id, ui_method, callback_data);
159         if (!pkey)
160                 {
161                 ENGINEerr(ENGINE_F_ENGINE_LOAD_PUBLIC_KEY,
162                         ENGINE_R_FAILED_LOADING_PUBLIC_KEY);
163                 return 0;
164                 }
165         return pkey;
166         }

168 int ENGINE_load_ssl_client_cert(ENGINE *e, SSL *s,
169         STACK_OF(X509_NAME) *ca_dn, X509 **pcert, EVP_PKEY **ppkey,
170         STACK_OF(X509) **pother, UI_METHOD *ui_method, void *callback_data)
171         {

173         if(e == NULL)
174                 {
175                 ENGINEerr(ENGINE_F_ENGINE_LOAD_SSL_CLIENT_CERT,
176                         ERR_R_PASSED_NULL_PARAMETER);
177                 return 0;
178                 }
179         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
180         if(e->funct_ref == 0)
181                 {
182                 CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
183                 ENGINEerr(ENGINE_F_ENGINE_LOAD_SSL_CLIENT_CERT,
184                         ENGINE_R_NOT_INITIALISED);
185                 return 0;
186                 }
187         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
188         if (!e->load_ssl_client_cert)
189                 {
190                 ENGINEerr(ENGINE_F_ENGINE_LOAD_SSL_CLIENT_CERT,
191                         ENGINE_R_NO_LOAD_FUNCTION);
192                 return 0;
193                 }
```



```
194         return e->load_ssl_client_cert(e, s, ca_dn, pcert, ppkey, pother,
195                                        ui_method, callback_data);
196         }
197 #endif /* ! codereview */
```

```
**********************************************************
    4338 Wed Aug 13 19:52:37 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_rdrand.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/opensslconf.h>

#include <stdio.h>
#include <string.h>
#include <openssl/engine.h>
#include <openssl/rand.h>
#include <openssl/err.h>

#if (defined(__i386)   || defined(__i386__)   || defined(_M_IX86) || \
     defined(__x86_64) || defined(__x86_64__) || \
     defined(_M_AMD64) || defined (_M_X64)) && defined(OPENSSL_CPUID_OBJ)
```



```
size_t OPENSSL_ia32_rdrand(void);

static int get_random_bytes (unsigned char *buf, int num)
        {
        size_t rnd;

        while (num>=(int)sizeof(size_t)) {
                if ((rnd = OPENSSL_ia32_rdrand()) == 0) return 0;

                *((size_t *)buf) = rnd;
                buf += sizeof(size_t);
                num -= sizeof(size_t);
        }
        if (num) {
                if ((rnd = OPENSSL_ia32_rdrand()) == 0) return 0;

                memcpy (buf,&rnd,num);
        }

        return 1;
        }

static int random_status (void)
{       return 1;       }

static RAND_METHOD rdrand_meth =
        {
        NULL,   /* seed */
        get_random_bytes,
        NULL,   /* cleanup */
        NULL,   /* add */
        get_random_bytes,
        random_status,
        };

static int rdrand_init(ENGINE *e)
{       return 1;       }

static const char *engine_e_rdrand_id = "rdrand";
static const char *engine_e_rdrand_name = "Intel RDRAND engine";

static int bind_helper(ENGINE *e)
        {
        if (!ENGINE_set_id(e, engine_e_rdrand_id) ||
            !ENGINE_set_name(e, engine_e_rdrand_name) ||
            !ENGINE_set_flags(e, ENGINE_FLAGS_NO_REGISTER_ALL) ||
            !ENGINE_set_init_function(e, rdrand_init) ||
            !ENGINE_set_RAND(e, &rdrand_meth) )
                return 0;

        return 1;
        }

static ENGINE *ENGINE_rdrand(void)
        {
        ENGINE *ret = ENGINE_new();
        if(!ret)
                return NULL;
        if(!bind_helper(ret))
                {
                ENGINE_free(ret);
                return NULL;
                }
        return ret;
        }
```

```
128 void ENGINE_load_rdrand (void)
129         {
130         extern unsigned int OPENSSL_ia32cap_P[];

132         if (OPENSSL_ia32cap_P[1] & (1<<(62-32)))
133                 {
134                 ENGINE *toadd = ENGINE_rdrand();
135                 if(!toadd) return;
136                 ENGINE_add(toadd);
137                 ENGINE_free(toadd);
138                 ERR_clear_error();
139                 }
140         }
141 #else
142 void ENGINE_load_rdrand (void) {}
143 #endif
144 #endif /* ! codereview */
```

```
**********************************************************
   17904 Wed Aug 13 19:52:37 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_rsax.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/eng_rsax.c */
/* Copyright (c) 2010-2010 Intel Corp.
 *   Author: Vinodh.Gopal@intel.com
 *           Jim Guilford
 *           Erdinc.Ozturk@intel.com
 *           Maxim.Perminov@intel.com
 *           Ying.Huang@intel.com
 *
 * More information about algorithm used can be found at:
 *   http://www.cse.buffalo.edu/srds2009/escs2009_submission_Gopal.pdf
 */
/* ====================================================================
 * Copyright (c) 1999-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
```



```
 * Hudson (tjh@cryptsoft.com).
 */

#include <openssl/opensslconf.h>

#include <stdio.h>
#include <string.h>
#include <openssl/crypto.h>
#include <openssl/buffer.h>
#include <openssl/engine.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#include <openssl/bn.h>
#include <openssl/err.h>

/* RSAX is available **ONLY* on x86_64 CPUs */
#undef COMPILE_RSAX

#if (defined(__x86_64) || defined(__x86_64__) || \
     defined(_M_AMD64) || defined (_M_X64)) && !defined(OPENSSL_NO_ASM)
#define COMPILE_RSAX
static ENGINE *ENGINE_rsax (void);
#endif

void ENGINE_load_rsax (void)
        {
/* On non-x86 CPUs it just returns. */
#ifdef COMPILE_RSAX
        ENGINE *toadd = ENGINE_rsax();
        if(!toadd) return;
        ENGINE_add(toadd);
        ENGINE_free(toadd);
        ERR_clear_error();
#endif
        }

#ifdef COMPILE_RSAX
#define E_RSAX_LIB_NAME "rsax engine"

static int e_rsax_destroy(ENGINE *e);
static int e_rsax_init(ENGINE *e);
static int e_rsax_finish(ENGINE *e);
static int e_rsax_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)(void));

#ifndef OPENSSL_NO_RSA
/* RSA stuff */
static int e_rsax_rsa_mod_exp(BIGNUM *r, const BIGNUM *I, RSA *rsa, BN_CTX *ctx)
static int e_rsax_rsa_finish(RSA *r);
#endif

static const ENGINE_CMD_DEFN e_rsax_cmd_defns[] = {
        {0, NULL, NULL, 0}
        };

#ifndef OPENSSL_NO_RSA
/* Our internal RSA_METHOD that we provide pointers to */
static RSA_METHOD e_rsax_rsa =
        {
        "Intel RSA-X method",
        NULL,
        NULL,
        NULL,
        NULL,
        e_rsax_rsa_mod_exp,
        NULL,
```

```
128         NULL,
129         e_rsax_rsa_finish,
130         RSA_FLAG_CACHE_PUBLIC|RSA_FLAG_CACHE_PRIVATE,
131         NULL,
132         NULL,
133         NULL
134         };
135 #endif

137 /* Constants used when creating the ENGINE */
138 static const char *engine_e_rsax_id = "rsax";
139 static const char *engine_e_rsax_name = "RSAX engine support";

141 /* This internal function is used by ENGINE_rsax() */
142 static int bind_helper(ENGINE *e)
143         {
144 #ifndef OPENSSL_NO_RSA
145         const RSA_METHOD *meth1;
146 #endif
147         if(!ENGINE_set_id(e, engine_e_rsax_id) ||
148                         !ENGINE_set_name(e, engine_e_rsax_name) ||
149 #ifndef OPENSSL_NO_RSA
150                         !ENGINE_set_RSA(e, &e_rsax_rsa) ||
151 #endif
152                         !ENGINE_set_destroy_function(e, e_rsax_destroy) ||
153                         !ENGINE_set_init_function(e, e_rsax_init) ||
154                         !ENGINE_set_finish_function(e, e_rsax_finish) ||
155                         !ENGINE_set_ctrl_function(e, e_rsax_ctrl) ||
156                         !ENGINE_set_cmd_defns(e, e_rsax_cmd_defns))
157                 return 0;

159 #ifndef OPENSSL_NO_RSA
160         meth1 = RSA_PKCS1_SSLeay();
161         e_rsax_rsa.rsa_pub_enc = meth1->rsa_pub_enc;
162         e_rsax_rsa.rsa_pub_dec = meth1->rsa_pub_dec;
163         e_rsax_rsa.rsa_priv_enc = meth1->rsa_priv_enc;
164         e_rsax_rsa.rsa_priv_dec = meth1->rsa_priv_dec;
165         e_rsax_rsa.bn_mod_exp = meth1->bn_mod_exp;
166 #endif
167         return 1;
168         }

170 static ENGINE *ENGINE_rsax(void)
171         {
172         ENGINE *ret = ENGINE_new();
173         if(!ret)
174                 return NULL;
175         if(!bind_helper(ret))
176                 {
177                 ENGINE_free(ret);
178                 return NULL;
179                 }
180         return ret;
181         }

183 #ifndef OPENSSL_NO_RSA
184 /* Used to attach our own key-data to an RSA structure */
185 static int rsax_ex_data_idx = -1;
186 #endif

188 static int e_rsax_destroy(ENGINE *e)
189         {
190         return 1;
191         }

193 /* (de)initialisation functions. */
```



```
194 static int e_rsax_init(ENGINE *e)
195         {
196 #ifndef OPENSSL_NO_RSA
197         if (rsax_ex_data_idx == -1)
198                 rsax_ex_data_idx = RSA_get_ex_new_index(0,
199                         NULL,
200                         NULL, NULL, NULL);
201 #endif
202         if (rsax_ex_data_idx  == -1)
203                 return 0;
204         return 1;
205         }

207 static int e_rsax_finish(ENGINE *e)
208         {
209         return 1;
210         }

212 static int e_rsax_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)(void))
213         {
214         int to_return = 1;

216         switch(cmd)
217                 {
218         /* The command isn't understood by this engine */
219         default:
220                 to_return = 0;
221                 break;
222                 }

224         return to_return;
225         }


228 #ifndef OPENSSL_NO_RSA

230 #ifdef _WIN32
231 typedef unsigned __int64 UINT64;
232 #else
233 typedef unsigned long long UINT64;
234 #endif
235 typedef unsigned short UINT16;

237 /* Table t is interleaved in the following manner:
238  * The order in memory is t[0][0], t[0][1], ..., t[0][7], t[1][0], ...
239  * A particular 512-bit value is stored in t[][index] rather than the more
240  * normal t[index][]; i.e. the qwords of a particular entry in t are not
241  * adjacent in memory
242  */

244 /* Init BIGNUM b from the interleaved UINT64 array */
245 static int interleaved_array_to_bn_512(BIGNUM* b, UINT64 *array);

247 /* Extract array elements from BIGNUM b
248  * To set the whole array from b, call with n=8
249  */
250 static int bn_extract_to_array_512(const BIGNUM* b, unsigned int n, UINT64 *arra

252 struct mod_ctx_512 {
253     UINT64 t[8][8];
254     UINT64 m[8];
255     UINT64 m1[8]; /* 2^278 % m */
256     UINT64 m2[8]; /* 2^640 % m */
257     UINT64 k1[2]; /* (- 1/m) % 2^128 */
258 };
```

```
260 static int mod_exp_pre_compute_data_512(UINT64 *m, struct mod_ctx_512 *data);

262 void mod_exp_512(UINT64 *result, /* 512 bits, 8 qwords */
263                  UINT64 *g,      /* 512 bits, 8 qwords */
264                  UINT64 *exp,    /* 512 bits, 8 qwords */
265                  struct mod_ctx_512 *data);

267 typedef struct st_e_rsax_mod_ctx
268 {
269   UINT64 type;
270   union {
271     struct mod_ctx_512 b512;
272   } ctx;

274 } E_RSAX_MOD_CTX;

276 static E_RSAX_MOD_CTX *e_rsax_get_ctx(RSA *rsa, int idx, BIGNUM* m)
277 {
278         E_RSAX_MOD_CTX *hptr;

280         if (idx < 0 || idx > 2)
281            return NULL;

283         hptr = RSA_get_ex_data(rsa, rsax_ex_data_idx);
284         if (!hptr) {
285             hptr = OPENSSL_malloc(3*sizeof(E_RSAX_MOD_CTX));
286             if (!hptr) return NULL;
287             hptr[2].type = hptr[1].type= hptr[0].type = 0;
288             RSA_set_ex_data(rsa, rsax_ex_data_idx, hptr);
289         }

291         if (hptr[idx].type == (UINT64)BN_num_bits(m))
292             return hptr+idx;

294         if (BN_num_bits(m) == 512) {
295             UINT64 _m[8];
296             bn_extract_to_array_512(m, 8, _m);
297             memset( &hptr[idx].ctx.b512, 0, sizeof(struct mod_ctx_512));
298             mod_exp_pre_compute_data_512(_m, &hptr[idx].ctx.b512);
299         }

301         hptr[idx].type = BN_num_bits(m);
302         return hptr+idx;
303 }

305 static int e_rsax_rsa_finish(RSA *rsa)
306         {
307         E_RSAX_MOD_CTX *hptr = RSA_get_ex_data(rsa, rsax_ex_data_idx);
308         if(hptr)
309                 {
310                 OPENSSL_free(hptr);
311                 RSA_set_ex_data(rsa, rsax_ex_data_idx, NULL);
312                 }
313         if (rsa->_method_mod_n)
314                 BN_MONT_CTX_free(rsa->_method_mod_n);
315         if (rsa->_method_mod_p)
316                 BN_MONT_CTX_free(rsa->_method_mod_p);
317         if (rsa->_method_mod_q)
318                 BN_MONT_CTX_free(rsa->_method_mod_q);
319         return 1;
320         }


323 static int e_rsax_bn_mod_exp(BIGNUM *r, const BIGNUM *g, const BIGNUM *e,
324                     const BIGNUM *m, BN_CTX *ctx, BN_MONT_CTX *in_mont, E_RSAX_M
325 {
```



```
326         if (rsax_mod_ctx && BN_get_flags(e, BN_FLG_CONSTTIME) != 0) {
327            if (BN_num_bits(m) == 512) {
328                 UINT64 _r[8];
329                 UINT64 _g[8];
330                 UINT64 _e[8];

332                 /* Init the arrays from the BIGNUMs */
333                 bn_extract_to_array_512(g, 8, _g);
334                 bn_extract_to_array_512(e, 8, _e);

336                 mod_exp_512(_r, _g, _e, &rsax_mod_ctx->ctx.b512);
337                 /* Return the result in the BIGNUM */
338                 interleaved_array_to_bn_512(r, _r);
339                 return 1;
340            }
341         }

343         return BN_mod_exp_mont(r, g, e, m, ctx, in_mont);
344 }

346 /* Declares for the Intel CIAP 512-bit / CRT / 1024 bit RSA modular
347  * exponentiation routine precalculations and a structure to hold the
348  * necessary values.  These files are meant to live in crypto/rsa/ in
349  * the target openssl.
350  */

352 /*
353  * Local method: extracts a piece from a BIGNUM, to fit it into
354  * an array. Call with n=8 to extract an entire 512-bit BIGNUM
355  */
356 static int bn_extract_to_array_512(const BIGNUM* b, unsigned int n, UINT64 *arra
357 {
358         int i;
359         UINT64 tmp;
360         unsigned char bn_buff[64];
361         memset(bn_buff, 0, 64);
362         if (BN_num_bytes(b) > 64) {
363                 printf ("Can't support this byte size\n");
364                 return 0; }
365         if (BN_num_bytes(b)!=0) {
366                 if (!BN_bn2bin(b, bn_buff+(64-BN_num_bytes(b)))) {
367                         printf ("Error's in bn2bin\n");
368                         /* We have to error, here */
369                         return 0; } }
370         while (n-- > 0) {
371                 array[n] = 0;
372                 for (i=7; i>=0; i--) {
373                         tmp = bn_buff[63-(n*8+i)];
374                         array[n] |= tmp << (8*i); } }
375         return 1;
376 }

378 /* Init a 512-bit BIGNUM from the UINT64*_ (8 * 64) interleaved array */
379 static int interleaved_array_to_bn_512(BIGNUM* b, UINT64 *array)
380 {
381         unsigned char tmp[64];
382         int n=8;
383         int i;
384         while (n-- > 0) {
385                 for (i = 7; i>=0; i--) {
386                         tmp[63-(n*8+i)] = (unsigned char)(array[n]>>(8*i)); } }
387         BN_bin2bn(tmp, 64, b);
388         return 0;
389 }
```

```
/* The main 512bit precompute call */
static int mod_exp_pre_compute_data_512(UINT64 *m, struct mod_ctx_512 *data)
 {
    BIGNUM two_768, two_640, two_128, two_512, tmp, _m, tmp2;

    /* We need a BN_CTX for the modulo functions */
    BN_CTX* ctx;
    /* Some tmps */
    UINT64 _t[8];
    int i, j, ret = 0;

    /* Init _m with m */
    BN_init(&_m);
    interleaved_array_to_bn_512(&_m, m);
    memset(_t, 0, 64);

    /* Inits */
    BN_init(&two_768);
    BN_init(&two_640);
    BN_init(&two_128);
    BN_init(&two_512);
    BN_init(&tmp);
    BN_init(&tmp2);

    /* Create our context */
    if ((ctx=BN_CTX_new()) == NULL) { goto err; }
        BN_CTX_start(ctx);

    /*
     * For production, if you care, these only need to be set once,
     * and may be made constants.
     */
    BN_lshift(&two_768, BN_value_one(), 768);
    BN_lshift(&two_640, BN_value_one(), 640);
    BN_lshift(&two_128, BN_value_one(), 128);
    BN_lshift(&two_512, BN_value_one(), 512);

    if (0 == (m[7] & 0x8000000000000000)) {
        exit(1);
    }
    if (0 == (m[0] & 0x1)) { /* Odd modulus required for Mont */
        exit(1);
    }

    /* Precompute m1 */
    BN_mod(&tmp, &two_768, &_m, ctx);
    if (!bn_extract_to_array_512(&tmp, 8, &data->m1[0])) {
            goto err; }

    /* Precompute m2 */
    BN_mod(&tmp, &two_640, &_m, ctx);
    if (!bn_extract_to_array_512(&tmp, 8, &data->m2[0])) {
            goto err;
    }

    /*
     * Precompute k1, a 128b number = ((-1)* m-1 ) mod 2128; k1 should
     * be non-negative.
     */
    BN_mod_inverse(&tmp, &_m, &two_128, ctx);
    if (!BN_is_zero(&tmp)) { BN_sub(&tmp, &two_128, &tmp); }
    if (!bn_extract_to_array_512(&tmp, 2, &data->k1[0])) {
            goto err; }

    /* Precompute t */
    for (i=0; i<8; i++) {
```



```
        BN_zero(&tmp);
        if (i & 1) { BN_add(&tmp, &two_512, &tmp); }
        if (i & 2) { BN_add(&tmp, &two_512, &tmp); }
        if (i & 4) { BN_add(&tmp, &two_640, &tmp); }

        BN_nnmod(&tmp2, &tmp, &_m, ctx);
        if (!bn_extract_to_array_512(&tmp2, 8, _t)) {
                goto err; }
        for (j=0; j<8; j++) data->t[j][i] = _t[j]; }

    /* Precompute m */
    for (i=0; i<8; i++) {
        data->m[i] = m[i]; }

    ret = 1;

err:
    /* Cleanup */
        if (ctx != NULL) {
                BN_CTX_end(ctx); BN_CTX_free(ctx); }
    BN_free(&two_768);
    BN_free(&two_640);
    BN_free(&two_128);
    BN_free(&two_512);
    BN_free(&tmp);
    BN_free(&tmp2);
    BN_free(&_m);

    return ret;
}


static int e_rsax_rsa_mod_exp(BIGNUM *r0, const BIGNUM *I, RSA *rsa, BN_CTX *ctx
        {
        BIGNUM *r1,*m1,*vrfy;
        BIGNUM local_dmp1,local_dmq1,local_c,local_r1;
        BIGNUM *dmp1,*dmq1,*c,*pr1;
        int ret=0;

        BN_CTX_start(ctx);
        r1 = BN_CTX_get(ctx);
        m1 = BN_CTX_get(ctx);
        vrfy = BN_CTX_get(ctx);

        {
                BIGNUM local_p, local_q;
                BIGNUM *p = NULL, *q = NULL;
                int error = 0;

                /* Make sure BN_mod_inverse in Montgomery
                 * intialization uses the BN_FLG_CONSTTIME flag
                 * (unless RSA_FLAG_NO_CONSTTIME is set)
                 */
                if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
                        {
                        BN_init(&local_p);
                        p = &local_p;
                        BN_with_flags(p, rsa->p, BN_FLG_CONSTTIME);

                        BN_init(&local_q);
                        q = &local_q;
                        BN_with_flags(q, rsa->q, BN_FLG_CONSTTIME);
                        }
                else
                        {
                        p = rsa->p;
```

```
524                         q = rsa->q;
525                         }

527                 if (rsa->flags & RSA_FLAG_CACHE_PRIVATE)
528                         {
529                         if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_p, CRYPTO_
530                                 error = 1;
531                         if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_q, CRYPTO_
532                                 error = 1;
533                         }

535                 /* clean up */
536                 if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
537                         {
538                         BN_free(&local_p);
539                         BN_free(&local_q);
540                         }
541                 if ( error )
542                         goto err;
543         }

545         if (rsa->flags & RSA_FLAG_CACHE_PUBLIC)
546                 if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_n, CRYPTO_LOCK_RSA
547                         goto err;

549         /* compute I mod q */
550         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
551                 {
552                 c = &local_c;
553                 BN_with_flags(c, I, BN_FLG_CONSTTIME);
554                 if (!BN_mod(r1,c,rsa->q,ctx)) goto err;
555                 }
556         else
557                 {
558                 if (!BN_mod(r1,I,rsa->q,ctx)) goto err;
559                 }

561         /* compute r1^dmq1 mod q */
562         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
563                 {
564                 dmq1 = &local_dmq1;
565                 BN_with_flags(dmq1, rsa->dmq1, BN_FLG_CONSTTIME);
566                 }
567         else
568                 dmq1 = rsa->dmq1;

570         if (!e_rsax_bn_mod_exp(m1,r1,dmq1,rsa->q,ctx,
571                 rsa->_method_mod_q, e_rsax_get_ctx(rsa, 0, rsa->q) )) goto err;

573         /* compute I mod p */
574         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
575                 {
576                 c = &local_c;
577                 BN_with_flags(c, I, BN_FLG_CONSTTIME);
578                 if (!BN_mod(r1,c,rsa->p,ctx)) goto err;
579                 }
580         else
581                 {
582                 if (!BN_mod(r1,I,rsa->p,ctx)) goto err;
583                 }

585         /* compute r1^dmp1 mod p */
586         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
587                 {
588                 dmp1 = &local_dmp1;
589                 BN_with_flags(dmp1, rsa->dmp1, BN_FLG_CONSTTIME);
```



```
590                 }
591         else
592                 dmp1 = rsa->dmp1;

594         if (!e_rsax_bn_mod_exp(r0,r1,dmp1,rsa->p,ctx,
595                 rsa->_method_mod_p, e_rsax_get_ctx(rsa, 1, rsa->p) )) goto err;

597         if (!BN_sub(r0,r0,m1)) goto err;
598         /* This will help stop the size of r0 increasing, which does
599          * affect the multiply if it optimised for a power of 2 size */
600         if (BN_is_negative(r0))
601                 if (!BN_add(r0,r0,rsa->p)) goto err;

603         if (!BN_mul(r1,r0,rsa->iqmp,ctx)) goto err;

605         /* Turn BN_FLG_CONSTTIME flag on before division operation */
606         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
607                 {
608                 pr1 = &local_r1;
609                 BN_with_flags(pr1, r1, BN_FLG_CONSTTIME);
610                 }
611         else
612                 pr1 = r1;
613         if (!BN_mod(r0,pr1,rsa->p,ctx)) goto err;

615         /* If p < q it is occasionally possible for the correction of
616          * adding 'p' if r0 is negative above to leave the result still
617          * negative. This can break the private key operations: the following
618          * second correction should *always* correct this rare occurrence.
619          * This will *never* happen with OpenSSL generated keys because
620          * they ensure p > q [steve]
621          */
622         if (BN_is_negative(r0))
623                 if (!BN_add(r0,r0,rsa->p)) goto err;
624         if (!BN_mul(r1,r0,rsa->q,ctx)) goto err;
625         if (!BN_add(r0,r1,m1)) goto err;

627         if (rsa->e && rsa->n)
628                 {
629                 if (!e_rsax_bn_mod_exp(vrfy,r0,rsa->e,rsa->n,ctx,rsa->_method_mo
630                     goto err;

632                 /* If 'I' was greater than (or equal to) rsa->n, the operation
633                  * will be equivalent to using 'I mod n'. However, the result of
634                  * the verify will *always* be less than 'n' so we don't check
635                  * for absolute equality, just congruency. */
636                 if (!BN_sub(vrfy, vrfy, I)) goto err;
637                 if (!BN_mod(vrfy, vrfy, rsa->n, ctx)) goto err;
638                 if (BN_is_negative(vrfy))
639                         if (!BN_add(vrfy, vrfy, rsa->n)) goto err;
640                 if (!BN_is_zero(vrfy))
641                         {
642                         /* 'I' and 'vrfy' aren't congruent mod n. Don't leak
643                          * miscalculated CRT output, just do a raw (slower)
644                          * mod_exp and return that instead. */

646                         BIGNUM local_d;
647                         BIGNUM *d = NULL;

649                         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
650                                 {
651                                 d = &local_d;
652                                 BN_with_flags(d, rsa->d, BN_FLG_CONSTTIME);
653                                 }
654                         else
655                                 d = rsa->d;
```

```
656                         if (!e_rsax_bn_mod_exp(r0,I,d,rsa->n,ctx,
657                                                rsa->_method_mod_n, e_rsax_ge
658                         }
659                 }
660         ret=1;

662 err:
663         BN_CTX_end(ctx);

665         return ret;
666         }
667 #endif /* !OPENSSL_NO_RSA */
668 #endif /* !COMPILE_RSAX */
669 #endif /* ! codereview */
```

****
10123 Wed Aug 13 19:52:38 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/eng_table.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/evp.h>
#include <openssl/lhash.h>
#include "eng_int.h"

/* The type of the items in the table */
typedef struct st_engine_pile
        {
        /* The 'nid' of this algorithm/mode */
        int nid;
        /* ENGINEs that implement this algorithm/mode. */
        STACK_OF(ENGINE) *sk;
        /* The default ENGINE to perform this algorithm/mode. */
        ENGINE *funct;
        /* Zero if 'sk' is newer than the cached 'funct', non-zero otherwise */
        int uptodate;
        } ENGINE_PILE;

DECLARE_LHASH_OF(ENGINE_PILE);

/* The type exposed in eng_int.h */
struct st_engine_table
        {
        LHASH_OF(ENGINE_PILE) piles;
        }; /* ENGINE_TABLE */


typedef struct st_engine_pile_doall
        {
        engine_table_doall_cb *cb;
        void *arg;
        } ENGINE_PILE_DOALL;


/* Global flags (ENGINE_TABLE_FLAG_***). */
static unsigned int table_flags = 0;

/* API function manipulating 'table_flags' */
unsigned int ENGINE_get_table_flags(void)
        {
        return table_flags;
        }

void ENGINE_set_table_flags(unsigned int flags)
        {
        table_flags = flags;
        }

/* Internal functions for the "piles" hash table */
static unsigned long engine_pile_hash(const ENGINE_PILE *c)
        {
        return c->nid;
        }

static int engine_pile_cmp(const ENGINE_PILE *a, const ENGINE_PILE *b)
        {
        return a->nid - b->nid;
        }
static IMPLEMENT_LHASH_HASH_FN(engine_pile, ENGINE_PILE)
static IMPLEMENT_LHASH_COMP_FN(engine_pile, ENGINE_PILE)

static int int_table_check(ENGINE_TABLE **t, int create)
        {
        LHASH_OF(ENGINE_PILE) *lh;

        if(*t) return 1;
        if(!create) return 0;
        if((lh = lh_ENGINE_PILE_new()) == NULL)
                return 0;
        *t = (ENGINE_TABLE *)lh;
        return 1;
        }
```

```
/* Privately exposed (via eng_int.h) functions for adding and/or removing
 * ENGINEs from the implementation table */
int engine_table_register(ENGINE_TABLE **table, ENGINE_CLEANUP_CB *cleanup,
                ENGINE *e, const int *nids, int num_nids, int setdefault)
        {
        int ret = 0, added = 0;
        ENGINE_PILE tmplate, *fnd;
        CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
        if(!(*table))
                added = 1;
        if(!int_table_check(table, 1))
                goto end;
        if(added)
                /* The cleanup callback needs to be added */
                engine_cleanup_add_first(cleanup);
        while(num_nids--)
                {
                tmplate.nid = *nids;
                fnd = lh_ENGINE_PILE_retrieve(&(*table)->piles, &tmplate);
                if(!fnd)
                        {
                        fnd = OPENSSL_malloc(sizeof(ENGINE_PILE));
                        if(!fnd) goto end;
                        fnd->uptodate = 1;
                        fnd->nid = *nids;
                        fnd->sk = sk_ENGINE_new_null();
                        if(!fnd->sk)
                                {
                                OPENSSL_free(fnd);
                                goto end;
                                }
                        fnd->funct = NULL;
                        (void)lh_ENGINE_PILE_insert(&(*table)->piles, fnd);
                        }
                /* A registration shouldn't add duplciate entries */
                (void)sk_ENGINE_delete_ptr(fnd->sk, e);
                /* if 'setdefault', this ENGINE goes to the head of the list */
                if(!sk_ENGINE_push(fnd->sk, e))
                        goto end;
                /* "touch" this ENGINE_PILE */
                fnd->uptodate = 0;
                if(setdefault)
                        {
                        if(!engine_unlocked_init(e))
                                {
                                ENGINEerr(ENGINE_F_ENGINE_TABLE_REGISTER,
                                                ENGINE_R_INIT_FAILED);
                                goto end;
                                }
                        if(fnd->funct)
                                engine_unlocked_finish(fnd->funct, 0);
                        fnd->funct = e;
                        fnd->uptodate = 1;
                        }
                nids++;
                }
        ret = 1;
end:
        CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
        return ret;
        }
static void int_unregister_cb_doall_arg(ENGINE_PILE *pile, ENGINE *e)
        {
        int n;
        /* Iterate the 'c->sk' stack removing any occurance of 'e' */
        while((n = sk_ENGINE_find(pile->sk, e)) >= 0)
```



```
                {
                (void)sk_ENGINE_delete(pile->sk, n);
                pile->uptodate = 0;
                }
        if(pile->funct == e)
                {
                engine_unlocked_finish(e, 0);
                pile->funct = NULL;
                }
        }
static IMPLEMENT_LHASH_DOALL_ARG_FN(int_unregister_cb, ENGINE_PILE, ENGINE)

void engine_table_unregister(ENGINE_TABLE **table, ENGINE *e)
        {
        CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
        if(int_table_check(table, 0))
                lh_ENGINE_PILE_doall_arg(&(*table)->piles,
                                         LHASH_DOALL_ARG_FN(int_unregister_cb),
                                         ENGINE, e);
        CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
        }

static void int_cleanup_cb_doall(ENGINE_PILE *p)
        {
        sk_ENGINE_free(p->sk);
        if(p->funct)
                engine_unlocked_finish(p->funct, 0);
        OPENSSL_free(p);
        }
static IMPLEMENT_LHASH_DOALL_FN(int_cleanup_cb, ENGINE_PILE)

void engine_table_cleanup(ENGINE_TABLE **table)
        {
        CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
        if(*table)
                {
                lh_ENGINE_PILE_doall(&(*table)->piles,
                                     LHASH_DOALL_FN(int_cleanup_cb));
                lh_ENGINE_PILE_free(&(*table)->piles);
                *table = NULL;
                }
        CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
        }

/* return a functional reference for a given 'nid' */
#ifndef ENGINE_TABLE_DEBUG
ENGINE *engine_table_select(ENGINE_TABLE **table, int nid)
#else
ENGINE *engine_table_select_tmp(ENGINE_TABLE **table, int nid, const char *f, in
#endif
        {
        ENGINE *ret = NULL;
        ENGINE_PILE tmplate, *fnd=NULL;
        int initres, loop = 0;

        if(!(*table))
                {
#ifdef ENGINE_TABLE_DEBUG
                fprintf(stderr, "engine_table_dbg: %s:%d, nid=%d, nothing "
                        "registered!\n", f, l, nid);
#endif
                return NULL;
                }
        ERR_set_mark();
        CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
        /* Check again inside the lock otherwise we could race against cleanup
```

```
260          * operations. But don't worry about a fprintf(stderr). */
261         if(!int_table_check(table, 0)) goto end;
262         tmplate.nid = nid;
263         fnd = lh_ENGINE_PILE_retrieve(&(*table)->piles, &tmplate);
264         if(!fnd) goto end;
265         if(fnd->funct && engine_unlocked_init(fnd->funct))
266                 {
267 #ifdef ENGINE_TABLE_DEBUG
268                 fprintf(stderr, "engine_table_dbg: %s:%d, nid=%d, using "
269                         "ENGINE '%s' cached\n", f, l, nid, fnd->funct->id);
270 #endif
271                 ret = fnd->funct;
272                 goto end;
273                 }
274         if(fnd->uptodate)
275                 {
276                 ret = fnd->funct;
277                 goto end;
278                 }
279 trynext:
280         ret = sk_ENGINE_value(fnd->sk, loop++);
281         if(!ret)
282                 {
283 #ifdef ENGINE_TABLE_DEBUG
284                 fprintf(stderr, "engine_table_dbg: %s:%d, nid=%d, no "
285                                 "registered implementations would initialise\n",
286                                 f, l, nid);
287 #endif
288                 goto end;
289                 }
290         /* Try to initialise the ENGINE? */
291         if((ret->funct_ref > 0) || !(table_flags & ENGINE_TABLE_FLAG_NOINIT))
292                 initres = engine_unlocked_init(ret);
293         else
294                 initres = 0;
295         if(initres)
296                 {
297                 /* Update 'funct' */
298                 if((fnd->funct != ret) && engine_unlocked_init(ret))
299                         {
300                         /* If there was a previous default we release it. */
301                         if(fnd->funct)
302                                 engine_unlocked_finish(fnd->funct, 0);
303                         fnd->funct = ret;
304 #ifdef ENGINE_TABLE_DEBUG
305                         fprintf(stderr, "engine_table_dbg: %s:%d, nid=%d, "
306                                 "setting default to '%s'\n", f, l, nid, ret->id)
307 #endif
308                         }
309 #ifdef ENGINE_TABLE_DEBUG
310                 fprintf(stderr, "engine_table_dbg: %s:%d, nid=%d, using "
311                                 "newly initialised '%s'\n", f, l, nid, ret->id);
312 #endif
313                 goto end;
314                 }
315         goto trynext;
316 end:
317         /* If it failed, it is unlikely to succeed again until some future
318          * registrations have taken place. In all cases, we cache. */
319         if(fnd) fnd->uptodate = 1;
320 #ifdef ENGINE_TABLE_DEBUG
321         if(ret)
322                 fprintf(stderr, "engine_table_dbg: %s:%d, nid=%d, caching "
323                                 "ENGINE '%s'\n", f, l, nid, ret->id);
324         else
325                 fprintf(stderr, "engine_table_dbg: %s:%d, nid=%d, caching "
```



```
326                                 "'no matching ENGINE'\n", f, l, nid);
327 #endif
328         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
329         /* Whatever happened, any failed init()s are not failures in this
330          * context, so clear our error state. */
331         ERR_pop_to_mark();
332         return ret;
333         }

335 /* Table enumeration */

337 static void int_cb_doall_arg(ENGINE_PILE *pile, ENGINE_PILE_DOALL *dall)
338         {
339         dall->cb(pile->nid, pile->sk, pile->funct, dall->arg);
340         }
341 static IMPLEMENT_LHASH_DOALL_ARG_FN(int_cb, ENGINE_PILE,ENGINE_PILE_DOALL)

343 void engine_table_doall(ENGINE_TABLE *table, engine_table_doall_cb *cb,
344                                                                 void *arg)
345         {
346         ENGINE_PILE_DOALL dall;
347         dall.cb = cb;
348         dall.arg = arg;
349         lh_ENGINE_PILE_doall_arg(&table->piles, LHASH_DOALL_ARG_FN(int_cb),
350                                  ENGINE_PILE_DOALL, &dall);
351         }
352 #endif /* ! codereview */
```

```
**********************************************************
   92357 Wed Aug 13 19:52:38 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/hw_pk11.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* crypto/engine/hw_pk11.c */
/*
 * This product includes software developed by the OpenSSL Project for
 * use in the OpenSSL Toolkit (http://www.openssl.org/).
 *
 * This project also referenced hw_pkcs11-0.9.7b.patch written by
 * Afchine Madjlessi.
 */
/*
 * ====================================================================
 * Copyright (c) 2000-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
```



```
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>

#include <openssl/opensslconf.h>
#include <openssl/e_os2.h>
#include <openssl/crypto.h>
#include <openssl/engine.h>
#include <openssl/dso.h>
#include <openssl/err.h>
#include <openssl/bn.h>
#include <openssl/md5.h>
#include <openssl/pem.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/aes.h>
#include <cryptlib.h>
#include <dlfcn.h>
#include <pthread.h>

#ifndef OPENSSL_NO_HW
#ifndef OPENSSL_NO_HW_PK11

/* label for debug messages printed on stderr */
#define PK11_DBG        "PKCS#11 ENGINE DEBUG"
/* prints a lot of debug messages on stderr about slot selection process */
#undef  DEBUG_SLOT_SELECTION
/*
 * Solaris specific code. See comment at check_hw_mechanisms() for more
 * information.
 */
#if defined (__SVR4) && defined (__sun)
#define SOLARIS_HW_SLOT_SELECTION
#endif

/*
 * AES counter mode is not supported in the OpenSSL EVP API yet and neither
 * there are official OIDs for mechanisms based on this mode. With our changes,
 * an application can define its own EVP calls for AES counter mode and then
 * it can make use of hardware acceleration through this engine. However, it's
 * better if we keep AES CTR support code under ifdef's.
 */
#define SOLARIS_AES_CTR

#include "cryptoki.h"
#include "pkcs11.h"
#include "hw_pk11_err.c"
```

```
#ifdef  SOLARIS_HW_SLOT_SELECTION
/*
 * Tables for symmetric ciphers and digest mechs found in the pkcs11_kernel
 * library. See comment at check_hw_mechanisms() for more information.
 */
int *hw_cnids;
int *hw_dnids;
#endif  /* SOLARIS_HW_SLOT_SELECTION */

/* PKCS#11 session caches and their locks for all operation types */
static PK11_CACHE session_cache[OP_MAX];

/*
 * As stated in v2.20, 11.7 Object Management Function, in section for
 * C_FindObjectsInit(), at most one search operation may be active at a given
 * time in a given session. Therefore, C_Find{,Init,Final}Objects() should be
 * grouped together to form one atomic search operation. This is already
 * ensured by the property of unique PKCS#11 session handle used for each
 * PK11_SESSION object.
 *
 * This is however not the biggest concern - maintaining consistency of the
 * underlying object store is more important. The same section of the spec also
 * says that one thread can be in the middle of a search operation while another
 * thread destroys the object matching the search template which would result in
 * invalid handle returned from the search operation.
 *
 * Hence, the following locks are used for both protection of the object stores.
 * They are also used for active list protection.
 */
pthread_mutex_t *find_lock[OP_MAX] = { NULL };

/*
 * lists of asymmetric key handles which are active (referenced by at least one
 * PK11_SESSION structure, either held by a thread or present in free_session
 * list) for given algorithm type
 */
PK11_active *active_list[OP_MAX] = { NULL };

/*
 * Create all secret key objects in a global session so that they are available
 * to use for other sessions. These other sessions may be opened or closed
 * without losing the secret key objects.
 */
static CK_SESSION_HANDLE        global_session = CK_INVALID_HANDLE;

/* ENGINE level stuff */
static int pk11_init(ENGINE *e);
static int pk11_library_init(ENGINE *e);
static int pk11_finish(ENGINE *e);
static int pk11_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)());
static int pk11_destroy(ENGINE *e);

/* RAND stuff */
static void pk11_rand_seed(const void *buf, int num);
static void pk11_rand_add(const void *buf, int num, double add_entropy);
static void pk11_rand_cleanup(void);
static int pk11_rand_bytes(unsigned char *buf, int num);
static int pk11_rand_status(void);

/* These functions are also used in other files */
PK11_SESSION *pk11_get_session(PK11_OPTYPE optype);
void pk11_return_session(PK11_SESSION *sp, PK11_OPTYPE optype);

/* active list manipulation functions used in this file */
extern int pk11_active_delete(CK_OBJECT_HANDLE h, PK11_OPTYPE type);
```



```
extern void pk11_free_active_list(PK11_OPTYPE type);

#ifndef OPENSSL_NO_RSA
int pk11_destroy_rsa_key_objects(PK11_SESSION *session);
int pk11_destroy_rsa_object_pub(PK11_SESSION *sp, CK_BBOOL uselock);
int pk11_destroy_rsa_object_priv(PK11_SESSION *sp, CK_BBOOL uselock);
#endif
#ifndef OPENSSL_NO_DSA
int pk11_destroy_dsa_key_objects(PK11_SESSION *session);
int pk11_destroy_dsa_object_pub(PK11_SESSION *sp, CK_BBOOL uselock);
int pk11_destroy_dsa_object_priv(PK11_SESSION *sp, CK_BBOOL uselock);
#endif
#ifndef OPENSSL_NO_DH
int pk11_destroy_dh_key_objects(PK11_SESSION *session);
int pk11_destroy_dh_object(PK11_SESSION *session, CK_BBOOL uselock);
#endif

/* Local helper functions */
static int pk11_free_all_sessions(void);
static int pk11_free_session_list(PK11_OPTYPE optype);
static int pk11_setup_session(PK11_SESSION *sp, PK11_OPTYPE optype);
static int pk11_destroy_cipher_key_objects(PK11_SESSION *session);
static int pk11_destroy_object(CK_SESSION_HANDLE session,
        CK_OBJECT_HANDLE oh);
static const char *get_PK11_LIBNAME(void);
static void free_PK11_LIBNAME(void);
static long set_PK11_LIBNAME(const char *name);

/* Symmetric cipher and digest support functions */
static int cipher_nid_to_pk11(int nid);
#ifdef  SOLARIS_AES_CTR
static int pk11_add_aes_ctr_NIDs(void);
#endif  /* SOLARIS_AES_CTR */
static int pk11_usable_ciphers(const int **nids);
static int pk11_usable_digests(const int **nids);
static int pk11_cipher_init(EVP_CIPHER_CTX *ctx, const unsigned char *key,
        const unsigned char *iv, int enc);
static int pk11_cipher_final(PK11_SESSION *sp);
static int pk11_cipher_do_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
        const unsigned char *in, size_t inl);
static int pk11_cipher_cleanup(EVP_CIPHER_CTX *ctx);
static int pk11_engine_ciphers(ENGINE *e, const EVP_CIPHER **cipher,
        const int **nids, int nid);
static int pk11_engine_digests(ENGINE *e, const EVP_MD **digest,
        const int **nids, int nid);
static CK_OBJECT_HANDLE pk11_get_cipher_key(EVP_CIPHER_CTX *ctx,
        const unsigned char *key, CK_KEY_TYPE key_type, PK11_SESSION *sp);
static int check_new_cipher_key(PK11_SESSION *sp, const unsigned char *key,
        int key_len);
static int md_nid_to_pk11(int nid);
static int pk11_digest_init(EVP_MD_CTX *ctx);
static int pk11_digest_update(EVP_MD_CTX *ctx, const void *data,
        size_t count);
static int pk11_digest_final(EVP_MD_CTX *ctx, unsigned char *md);
static int pk11_digest_copy(EVP_MD_CTX *to, const EVP_MD_CTX *from);
static int pk11_digest_cleanup(EVP_MD_CTX *ctx);

static int pk11_choose_slots(int *any_slot_found);
static void pk11_find_symmetric_ciphers(CK_FUNCTION_LIST_PTR pflist,
    CK_SLOT_ID current_slot, int *current_slot_n_cipher,
    int *local_cipher_nids);
static void pk11_find_digests(CK_FUNCTION_LIST_PTR pflist,
    CK_SLOT_ID current_slot, int *current_slot_n_digest,
    int *local_digest_nids);
static void pk11_get_symmetric_cipher(CK_FUNCTION_LIST_PTR, int slot_id,
    CK_MECHANISM_TYPE mech, int *current_slot_n_cipher, int *local_cipher_nids,
```

```
260     int id);
261 static void pk11_get_digest(CK_FUNCTION_LIST_PTR pflist, int slot_id,
262     CK_MECHANISM_TYPE mech, int *current_slot_n_digest, int *local_digest_nids,
263     int id);

265 static int pk11_init_all_locks(void);
266 static void pk11_free_all_locks(void);

268 #ifdef  SOLARIS_HW_SLOT_SELECTION
269 static int check_hw_mechanisms(void);
270 static int nid_in_table(int nid, int *nid_table);
271 #endif  /* SOLARIS_HW_SLOT_SELECTION */

273 /* Index for the supported ciphers */
274 enum pk11_cipher_id {
275         PK11_DES_CBC,
276         PK11_DES3_CBC,
277         PK11_DES_ECB,
278         PK11_DES3_ECB,
279         PK11_RC4,
280         PK11_AES_128_CBC,
281         PK11_AES_192_CBC,
282         PK11_AES_256_CBC,
283         PK11_AES_128_ECB,
284         PK11_AES_192_ECB,
285         PK11_AES_256_ECB,
286         PK11_BLOWFISH_CBC,
287 #ifdef  SOLARIS_AES_CTR
288         PK11_AES_128_CTR,
289         PK11_AES_192_CTR,
290         PK11_AES_256_CTR,
291 #endif  /* SOLARIS_AES_CTR */
292         PK11_CIPHER_MAX
293 };

295 /* Index for the supported digests */
296 enum pk11_digest_id {
297         PK11_MD5,
298         PK11_SHA1,
299         PK11_SHA224,
300         PK11_SHA256,
301         PK11_SHA384,
302         PK11_SHA512,
303         PK11_DIGEST_MAX
304 };

306 #define TRY_OBJ_DESTROY(sess_hdl, obj_hdl, retval, uselock, alg_type)   \
307         {                                                               \
308         if (uselock)                                                    \
309                 LOCK_OBJSTORE(alg_type);                                \
310         if (pk11_active_delete(obj_hdl, alg_type) == 1)                 \
311                 {                                                       \
312                 retval = pk11_destroy_object(sess_hdl, obj_hdl);        \
313                 }                                                       \
314         if (uselock)                                                    \
315                 UNLOCK_OBJSTORE(alg_type);                              \
316         }

318 static int cipher_nids[PK11_CIPHER_MAX];
319 static int digest_nids[PK11_DIGEST_MAX];
320 static int cipher_count         = 0;
321 static int digest_count         = 0;
322 static CK_BBOOL pk11_have_rsa   = CK_FALSE;
323 static CK_BBOOL pk11_have_dsa   = CK_FALSE;
324 static CK_BBOOL pk11_have_dh    = CK_FALSE;
325 static CK_BBOOL pk11_have_random = CK_FALSE;
```



```
327 typedef struct PK11_CIPHER_st
328         {
329         enum pk11_cipher_id     id;
330         int                     nid;
331         int                     iv_len;
332         int                     min_key_len;
333         int                     max_key_len;
334         CK_KEY_TYPE             key_type;
335         CK_MECHANISM_TYPE       mech_type;
336         } PK11_CIPHER;

338 static PK11_CIPHER ciphers[] =
339         {
340         { PK11_DES_CBC,         NID_des_cbc,            8,       8,   8,
341                 CKK_DES,        CKM_DES_CBC, },
342         { PK11_DES3_CBC,        NID_des_ede3_cbc,       8,      24,  24,
343                 CKK_DES3,       CKM_DES3_CBC, },
344         { PK11_DES_ECB,         NID_des_ecb,            0,       8,   8,
345                 CKK_DES,        CKM_DES_ECB, },
346         { PK11_DES3_ECB,        NID_des_ede3_ecb,       0,      24,  24,
347                 CKK_DES3,       CKM_DES3_ECB, },
348         { PK11_RC4,             NID_rc4,                0,      16, 256,
349                 CKK_RC4,        CKM_RC4, },
350         { PK11_AES_128_CBC,     NID_aes_128_cbc,        16,     16,  16,
351                 CKK_AES,        CKM_AES_CBC, },
352         { PK11_AES_192_CBC,     NID_aes_192_cbc,        16,     24,  24,
353                 CKK_AES,        CKM_AES_CBC, },
354         { PK11_AES_256_CBC,     NID_aes_256_cbc,        16,     32,  32,
355                 CKK_AES,        CKM_AES_CBC, },
356         { PK11_AES_128_ECB,     NID_aes_128_ecb,        0,      16,  16,
357                 CKK_AES,        CKM_AES_ECB, },
358         { PK11_AES_192_ECB,     NID_aes_192_ecb,        0,      24,  24,
359                 CKK_AES,        CKM_AES_ECB, },
360         { PK11_AES_256_ECB,     NID_aes_256_ecb,        0,      32,  32,
361                 CKK_AES,        CKM_AES_ECB, },
362         { PK11_BLOWFISH_CBC,    NID_bf_cbc,             8,      16,  16,
363                 CKK_BLOWFISH,   CKM_BLOWFISH_CBC, },
364 #ifdef  SOLARIS_AES_CTR
365         /* we don't know the correct NIDs until the engine is initialized */
366         { PK11_AES_128_CTR,     NID_undef,              16,     16,  16,
367                 CKK_AES,        CKM_AES_CTR, },
368         { PK11_AES_192_CTR,     NID_undef,              16,     24,  24,
369                 CKK_AES,        CKM_AES_CTR, },
370         { PK11_AES_256_CTR,     NID_undef,              16,     32,  32,
371                 CKK_AES,        CKM_AES_CTR, },
372 #endif  /* SOLARIS_AES_CTR */
373         };

375 typedef struct PK11_DIGEST_st
376         {
377         enum pk11_digest_id     id;
378         int                     nid;
379         CK_MECHANISM_TYPE       mech_type;
380         } PK11_DIGEST;

382 static PK11_DIGEST digests[] =
383         {
384         {PK11_MD5,      NID_md5,        CKM_MD5, },
385         {PK11_SHA1,     NID_sha1,       CKM_SHA_1, },
386         {PK11_SHA224,   NID_sha224,     CKM_SHA224, },
387         {PK11_SHA256,   NID_sha256,     CKM_SHA256, },
388         {PK11_SHA384,   NID_sha384,     CKM_SHA384, },
389         {PK11_SHA512,   NID_sha512,     CKM_SHA512, },
390         {0,             NID_undef,      0xFFFF, },
391         };
```

```
393 /*
394  * Structure to be used for the cipher_data/md_data in
395  * EVP_CIPHER_CTX/EVP_MD_CTX structures in order to use the same pk11
396  * session in multiple cipher_update calls
397  */
398 typedef struct PK11_CIPHER_STATE_st
399         {
400         PK11_SESSION    *sp;
401         } PK11_CIPHER_STATE;


404 /*
405  * libcrypto EVP stuff - this is how we get wired to EVP so the engine gets
406  * called when libcrypto requests a cipher NID.
407  *
408  * Note how the PK11_CIPHER_STATE is used here.
409  */

411 /* DES CBC EVP */
412 static const EVP_CIPHER pk11_des_cbc =
413         {
414         NID_des_cbc,
415         8, 8, 8,
416         EVP_CIPH_CBC_MODE,
417         pk11_cipher_init,
418         pk11_cipher_do_cipher,
419         pk11_cipher_cleanup,
420         sizeof (PK11_CIPHER_STATE),
421         EVP_CIPHER_set_asn1_iv,
422         EVP_CIPHER_get_asn1_iv,
423         NULL
424         };

426 /* 3DES CBC EVP */
427 static const EVP_CIPHER pk11_3des_cbc =
428         {
429         NID_des_ede3_cbc,
430         8, 24, 8,
431         EVP_CIPH_CBC_MODE,
432         pk11_cipher_init,
433         pk11_cipher_do_cipher,
434         pk11_cipher_cleanup,
435         sizeof (PK11_CIPHER_STATE),
436         EVP_CIPHER_set_asn1_iv,
437         EVP_CIPHER_get_asn1_iv,
438         NULL
439         };

441 /*
442  * ECB modes don’t use an Initial Vector so that’s why set_asn1_parameters and
443  * get_asn1_parameters fields are set to NULL.
444  */
445 static const EVP_CIPHER pk11_des_ecb =
446         {
447         NID_des_ecb,
448         8, 8, 8,
449         EVP_CIPH_ECB_MODE,
450         pk11_cipher_init,
451         pk11_cipher_do_cipher,
452         pk11_cipher_cleanup,
453         sizeof (PK11_CIPHER_STATE),
454         NULL,
455         NULL,
456         NULL
457         };
```



```
459 static const EVP_CIPHER pk11_3des_ecb =
460         {
461         NID_des_ede3_ecb,
462         8, 24, 8,
463         EVP_CIPH_ECB_MODE,
464         pk11_cipher_init,
465         pk11_cipher_do_cipher,
466         pk11_cipher_cleanup,
467         sizeof (PK11_CIPHER_STATE),
468         NULL,
469         NULL,
470         NULL
471         };


474 static const EVP_CIPHER pk11_aes_128_cbc =
475         {
476         NID_aes_128_cbc,
477         16, 16, 16,
478         EVP_CIPH_CBC_MODE,
479         pk11_cipher_init,
480         pk11_cipher_do_cipher,
481         pk11_cipher_cleanup,
482         sizeof (PK11_CIPHER_STATE),
483         EVP_CIPHER_set_asn1_iv,
484         EVP_CIPHER_get_asn1_iv,
485         NULL
486         };

488 static const EVP_CIPHER pk11_aes_192_cbc =
489         {
490         NID_aes_192_cbc,
491         16, 24, 16,
492         EVP_CIPH_CBC_MODE,
493         pk11_cipher_init,
494         pk11_cipher_do_cipher,
495         pk11_cipher_cleanup,
496         sizeof (PK11_CIPHER_STATE),
497         EVP_CIPHER_set_asn1_iv,
498         EVP_CIPHER_get_asn1_iv,
499         NULL
500         };

502 static const EVP_CIPHER pk11_aes_256_cbc =
503         {
504         NID_aes_256_cbc,
505         16, 32, 16,
506         EVP_CIPH_CBC_MODE,
507         pk11_cipher_init,
508         pk11_cipher_do_cipher,
509         pk11_cipher_cleanup,
510         sizeof (PK11_CIPHER_STATE),
511         EVP_CIPHER_set_asn1_iv,
512         EVP_CIPHER_get_asn1_iv,
513         NULL
514         };

516 /*
517  * ECB modes don’t use IV so that’s why set_asn1_parameters and
518  * get_asn1_parameters are set to NULL.
519  */
520 static const EVP_CIPHER pk11_aes_128_ecb =
521         {
522         NID_aes_128_ecb,
523         16, 16, 0,
```

```
524         EVP_CIPH_ECB_MODE,
525         pk11_cipher_init,
526         pk11_cipher_do_cipher,
527         pk11_cipher_cleanup,
528         sizeof (PK11_CIPHER_STATE),
529         NULL,
530         NULL,
531         NULL
532         };

534 static const EVP_CIPHER pk11_aes_192_ecb =
535         {
536         NID_aes_192_ecb,
537         16, 24, 0,
538         EVP_CIPH_ECB_MODE,
539         pk11_cipher_init,
540         pk11_cipher_do_cipher,
541         pk11_cipher_cleanup,
542         sizeof (PK11_CIPHER_STATE),
543         NULL,
544         NULL,
545         NULL
546         };

548 static const EVP_CIPHER pk11_aes_256_ecb =
549         {
550         NID_aes_256_ecb,
551         16, 32, 0,
552         EVP_CIPH_ECB_MODE,
553         pk11_cipher_init,
554         pk11_cipher_do_cipher,
555         pk11_cipher_cleanup,
556         sizeof (PK11_CIPHER_STATE),
557         NULL,
558         NULL,
559         NULL
560         };

562 #ifdef  SOLARIS_AES_CTR
563 /*
564  * NID_undef's will be changed to the AES counter mode NIDs as soon they are
565  * created in pk11_library_init(). Note that the need to change these structures
566  * is the reason why we don't define them with the const keyword.
567  */
568 static EVP_CIPHER pk11_aes_128_ctr =
569         {
570         NID_undef,
571         16, 16, 16,
572         EVP_CIPH_CBC_MODE,
573         pk11_cipher_init,
574         pk11_cipher_do_cipher,
575         pk11_cipher_cleanup,
576         sizeof (PK11_CIPHER_STATE),
577         EVP_CIPHER_set_asn1_iv,
578         EVP_CIPHER_get_asn1_iv,
579         NULL
580         };

582 static EVP_CIPHER pk11_aes_192_ctr =
583         {
584         NID_undef,
585         16, 24, 16,
586         EVP_CIPH_CBC_MODE,
587         pk11_cipher_init,
588         pk11_cipher_do_cipher,
589         pk11_cipher_cleanup,
```



```
590         sizeof (PK11_CIPHER_STATE),
591         EVP_CIPHER_set_asn1_iv,
592         EVP_CIPHER_get_asn1_iv,
593         NULL
594         };

596 static EVP_CIPHER pk11_aes_256_ctr =
597         {
598         NID_undef,
599         16, 32, 16,
600         EVP_CIPH_CBC_MODE,
601         pk11_cipher_init,
602         pk11_cipher_do_cipher,
603         pk11_cipher_cleanup,
604         sizeof (PK11_CIPHER_STATE),
605         EVP_CIPHER_set_asn1_iv,
606         EVP_CIPHER_get_asn1_iv,
607         NULL
608         };
609 #endif  /* SOLARIS_AES_CTR */

611 static const EVP_CIPHER pk11_bf_cbc =
612         {
613         NID_bf_cbc,
614         8, 16, 8,
615         EVP_CIPH_VARIABLE_LENGTH,
616         pk11_cipher_init,
617         pk11_cipher_do_cipher,
618         pk11_cipher_cleanup,
619         sizeof (PK11_CIPHER_STATE),
620         EVP_CIPHER_set_asn1_iv,
621         EVP_CIPHER_get_asn1_iv,
622         NULL
623         };

625 static const EVP_CIPHER pk11_rc4 =
626         {
627         NID_rc4,
628         1, 16, 0,
629         EVP_CIPH_VARIABLE_LENGTH,
630         pk11_cipher_init,
631         pk11_cipher_do_cipher,
632         pk11_cipher_cleanup,
633         sizeof (PK11_CIPHER_STATE),
634         NULL,
635         NULL,
636         NULL
637         };

639 static const EVP_MD pk11_md5 =
640         {
641         NID_md5,
642         NID_md5WithRSAEncryption,
643         MD5_DIGEST_LENGTH,
644         0,
645         pk11_digest_init,
646         pk11_digest_update,
647         pk11_digest_final,
648         pk11_digest_copy,
649         pk11_digest_cleanup,
650         EVP_PKEY_RSA_method,
651         MD5_CBLOCK,
652         sizeof (PK11_CIPHER_STATE),
653         };

655 static const EVP_MD pk11_sha1 =
```

```
656         {
657         NID_sha1,
658         NID_sha1WithRSAEncryption,
659         SHA_DIGEST_LENGTH,
660         0,
661         pk11_digest_init,
662         pk11_digest_update,
663         pk11_digest_final,
664         pk11_digest_copy,
665         pk11_digest_cleanup,
666         EVP_PKEY_RSA_method,
667         SHA_CBLOCK,
668         sizeof (PK11_CIPHER_STATE),
669         };

671 static const EVP_MD pk11_sha224 =
672         {
673         NID_sha224,
674         NID_sha224WithRSAEncryption,
675         SHA224_DIGEST_LENGTH,
676         0,
677         pk11_digest_init,
678         pk11_digest_update,
679         pk11_digest_final,
680         pk11_digest_copy,
681         pk11_digest_cleanup,
682         EVP_PKEY_RSA_method,
683         /* SHA-224 uses the same cblock size as SHA-256 */
684         SHA256_CBLOCK,
685         sizeof (PK11_CIPHER_STATE),
686         };

688 static const EVP_MD pk11_sha256 =
689         {
690         NID_sha256,
691         NID_sha256WithRSAEncryption,
692         SHA256_DIGEST_LENGTH,
693         0,
694         pk11_digest_init,
695         pk11_digest_update,
696         pk11_digest_final,
697         pk11_digest_copy,
698         pk11_digest_cleanup,
699         EVP_PKEY_RSA_method,
700         SHA256_CBLOCK,
701         sizeof (PK11_CIPHER_STATE),
702         };

704 static const EVP_MD pk11_sha384 =
705         {
706         NID_sha384,
707         NID_sha384WithRSAEncryption,
708         SHA384_DIGEST_LENGTH,
709         0,
710         pk11_digest_init,
711         pk11_digest_update,
712         pk11_digest_final,
713         pk11_digest_copy,
714         pk11_digest_cleanup,
715         EVP_PKEY_RSA_method,
716         /* SHA-384 uses the same cblock size as SHA-512 */
717         SHA512_CBLOCK,
718         sizeof (PK11_CIPHER_STATE),
719         };

721 static const EVP_MD pk11_sha512 =
```



```
722         {
723         NID_sha512,
724         NID_sha512WithRSAEncryption,
725         SHA512_DIGEST_LENGTH,
726         0,
727         pk11_digest_init,
728         pk11_digest_update,
729         pk11_digest_final,
730         pk11_digest_copy,
731         pk11_digest_cleanup,
732         EVP_PKEY_RSA_method,
733         SHA512_CBLOCK,
734         sizeof (PK11_CIPHER_STATE),
735         };

737 /*
738  * Initialization function. Sets up various PKCS#11 library components.
739  * The definitions for control commands specific to this engine
740  */
741 #define PK11_CMD_SO_PATH                ENGINE_CMD_BASE
742 static const ENGINE_CMD_DEFN pk11_cmd_defns[] =
743         {
744                 {
745                 PK11_CMD_SO_PATH,
746                 "SO_PATH",
747                 "Specifies the path to the 'pkcs#11' shared library",
748                 ENGINE_CMD_FLAG_STRING
749                 },
750                 {0, NULL, NULL, 0}
751         };


754 static RAND_METHOD pk11_random =
755         {
756         pk11_rand_seed,
757         pk11_rand_bytes,
758         pk11_rand_cleanup,
759         pk11_rand_add,
760         pk11_rand_bytes,
761         pk11_rand_status
762         };


765 /* Constants used when creating the ENGINE */
766 static const char *engine_pk11_id = "pkcs11";
767 static const char *engine_pk11_name = "PKCS #11 engine support";

769 CK_FUNCTION_LIST_PTR pFuncList = NULL;
770 static const char PK11_GET_FUNCTION_LIST[] = "C_GetFunctionList";

772 /*
773  * These is the static string constant for the DSO file name and the function
774  * symbol names to bind to.
775  */
776 static const char def_PK11_LIBNAME[] = PK11_LIB_LOCATION;

778 static CK_BBOOL true = TRUE;
779 static CK_BBOOL false = FALSE;
780 static CK_SLOT_ID pubkey_SLOTID = 0;
781 static CK_SLOT_ID rand_SLOTID = 0;
782 static CK_SLOT_ID SLOTID = 0;
783 static CK_BBOOL pk11_library_initialized = FALSE;
784 static CK_BBOOL pk11_atfork_initialized = FALSE;
785 static int pk11_pid = 0;

787 static DSO *pk11_dso = NULL;
```

```
/* allocate and initialize all locks used by the engine itself */
static int pk11_init_all_locks(void)
        {
        int type;

#ifndef OPENSSL_NO_RSA
        find_lock[OP_RSA] = OPENSSL_malloc(sizeof (pthread_mutex_t));
        if (find_lock[OP_RSA] == NULL)
                goto malloc_err;
        (void) pthread_mutex_init(find_lock[OP_RSA], NULL);
#endif /* OPENSSL_NO_RSA */

#ifndef OPENSSL_NO_DSA
        find_lock[OP_DSA] = OPENSSL_malloc(sizeof (pthread_mutex_t));
        if (find_lock[OP_DSA] == NULL)
                goto malloc_err;
        (void) pthread_mutex_init(find_lock[OP_DSA], NULL);
#endif /* OPENSSL_NO_DSA */

#ifndef OPENSSL_NO_DH
        find_lock[OP_DH] = OPENSSL_malloc(sizeof (pthread_mutex_t));
        if (find_lock[OP_DH] == NULL)
                goto malloc_err;
        (void) pthread_mutex_init(find_lock[OP_DH], NULL);
#endif /* OPENSSL_NO_DH */

        for (type = 0; type < OP_MAX; type++)
                {
                session_cache[type].lock =
                    OPENSSL_malloc(sizeof (pthread_mutex_t));
                if (session_cache[type].lock == NULL)
                        goto malloc_err;
                (void) pthread_mutex_init(session_cache[type].lock, NULL);
                }

        return (1);

malloc_err:
        pk11_free_all_locks();
        PK11err(PK11_F_INIT_ALL_LOCKS, PK11_R_MALLOC_FAILURE);
        return (0);
        }

static void pk11_free_all_locks(void)
        {
        int type;

#ifndef OPENSSL_NO_RSA
        if (find_lock[OP_RSA] != NULL)
                {
                (void) pthread_mutex_destroy(find_lock[OP_RSA]);
                OPENSSL_free(find_lock[OP_RSA]);
                find_lock[OP_RSA] = NULL;
                }
#endif /* OPENSSL_NO_RSA */
#ifndef OPENSSL_NO_DSA
        if (find_lock[OP_DSA] != NULL)
                {
                (void) pthread_mutex_destroy(find_lock[OP_DSA]);
                OPENSSL_free(find_lock[OP_DSA]);
                find_lock[OP_DSA] = NULL;
                }
#endif /* OPENSSL_NO_DSA */
#ifndef OPENSSL_NO_DH
        if (find_lock[OP_DH] != NULL)
```



```
                {
                (void) pthread_mutex_destroy(find_lock[OP_DH]);
                OPENSSL_free(find_lock[OP_DH]);
                find_lock[OP_DH] = NULL;
                }
#endif /* OPENSSL_NO_DH */

        for (type = 0; type < OP_MAX; type++)
                {
                if (session_cache[type].lock != NULL)
                        {
                        (void) pthread_mutex_destroy(session_cache[type].lock);
                        OPENSSL_free(session_cache[type].lock);
                        session_cache[type].lock = NULL;
                        }
                }
        }

/*
 * This internal function is used by ENGINE_pk11() and "dynamic" ENGINE support.
 */
static int bind_pk11(ENGINE *e)
        {
#ifndef OPENSSL_NO_RSA
        const RSA_METHOD *rsa = NULL;
        RSA_METHOD *pk11_rsa = PK11_RSA();
#endif  /* OPENSSL_NO_RSA */
        if (!pk11_library_initialized)
                if (!pk11_library_init(e))
                        return (0);

        if (!ENGINE_set_id(e, engine_pk11_id) ||
            !ENGINE_set_name(e, engine_pk11_name) ||
            !ENGINE_set_ciphers(e, pk11_engine_ciphers) ||
            !ENGINE_set_digests(e, pk11_engine_digests))
                return (0);
#ifndef OPENSSL_NO_RSA
        if (pk11_have_rsa == CK_TRUE)
                {
                if (!ENGINE_set_RSA(e, PK11_RSA()) ||
                    !ENGINE_set_load_privkey_function(e, pk11_load_privkey) ||
                    !ENGINE_set_load_pubkey_function(e, pk11_load_pubkey))
                        return (0);
#ifdef  DEBUG_SLOT_SELECTION
                fprintf(stderr, "%s: registered RSA\n", PK11_DBG);
#endif  /* DEBUG_SLOT_SELECTION */
                }
#endif  /* OPENSSL_NO_RSA */
#ifndef OPENSSL_NO_DSA
        if (pk11_have_dsa == CK_TRUE)
                {
                if (!ENGINE_set_DSA(e, PK11_DSA()))
                        return (0);
#ifdef  DEBUG_SLOT_SELECTION
                fprintf(stderr, "%s: registered DSA\n", PK11_DBG);
#endif  /* DEBUG_SLOT_SELECTION */
                }
#endif  /* OPENSSL_NO_DSA */
#ifndef OPENSSL_NO_DH
        if (pk11_have_dh == CK_TRUE)
                {
                if (!ENGINE_set_DH(e, PK11_DH()))
                        return (0);
#ifdef  DEBUG_SLOT_SELECTION
                fprintf(stderr, "%s: registered DH\n", PK11_DBG);
#endif  /* DEBUG_SLOT_SELECTION */
```

```
920                 }
921 #endif  /* OPENSSL_NO_DH */
922         if (pk11_have_random)
923                 {
924                 if (!ENGINE_set_RAND(e, &pk11_random))
925                         return (0);
926 #ifdef  DEBUG_SLOT_SELECTION
927                 fprintf(stderr, "%s: registered random\n", PK11_DBG);
928 #endif  /* DEBUG_SLOT_SELECTION */
929                 }
930         if (!ENGINE_set_init_function(e, pk11_init) ||
931             !ENGINE_set_destroy_function(e, pk11_destroy) ||
932             !ENGINE_set_finish_function(e, pk11_finish) ||
933             !ENGINE_set_ctrl_function(e, pk11_ctrl) ||
934             !ENGINE_set_cmd_defns(e, pk11_cmd_defns))
935                 return (0);

937 /*
938  * Apache calls OpenSSL function RSA_blinding_on() once during startup
939  * which in turn calls bn_mod_exp. Since we do not implement bn_mod_exp
940  * here, we wire it back to the OpenSSL software implementation.
941  * Since it is used only once, performance is not a concern.
942  */
943 #ifndef OPENSSL_NO_RSA
944         rsa = RSA_PKCS1_SSLeay();
945         pk11_rsa->rsa_mod_exp = rsa->rsa_mod_exp;
946         pk11_rsa->bn_mod_exp = rsa->bn_mod_exp;
947 #endif  /* OPENSSL_NO_RSA */

949         /* Ensure the pk11 error handling is set up */
950         ERR_load_pk11_strings();

952         return (1);
953         }

955 /* Dynamic engine support is disabled at a higher level for Solaris */
956 #ifdef  ENGINE_DYNAMIC_SUPPORT
957 static int bind_helper(ENGINE *e, const char *id)
958         {
959         if (id && (strcmp(id, engine_pk11_id) != 0))
960                 return (0);

962         if (!bind_pk11(e))
963                 return (0);

965         return (1);
966         }

968 IMPLEMENT_DYNAMIC_CHECK_FN()
969 IMPLEMENT_DYNAMIC_BIND_FN(bind_helper)

971 #else
972 static ENGINE *engine_pk11(void)
973         {
974         ENGINE *ret = ENGINE_new();

976         if (!ret)
977                 return (NULL);

979         if (!bind_pk11(ret))
980                 {
981                 ENGINE_free(ret);
982                 return (NULL);
983                 }

985         return (ret);
```



```
986         }

988 void
989 ENGINE_load_pk11(void)
990         {
991         ENGINE *e_pk11 = NULL;

993         /*
994          * Do not use dynamic PKCS#11 library on Solaris due to
995          * security reasons. We will link it in statically.
996          */
997         /* Attempt to load PKCS#11 library */
998         if (!pk11_dso)
999                 pk11_dso = DSO_load(NULL, get_PK11_LIBNAME(), NULL, 0);

1001        if (pk11_dso == NULL)
1002                {
1003                PK11err(PK11_F_LOAD, PK11_R_DSO_FAILURE);
1004                return;
1005                }

1007        e_pk11 = engine_pk11();
1008        if (!e_pk11)
1009                {
1010                DSO_free(pk11_dso);
1011                pk11_dso = NULL;
1012                return;
1013                }

1015        /*
1016         * At this point, the pk11 shared library is either dynamically
1017         * loaded or statically linked in. So, initialize the pk11
1018         * library before calling ENGINE_set_default since the latter
1019         * needs cipher and digest algorithm information
1020         */
1021        if (!pk11_library_init(e_pk11))
1022                {
1023                DSO_free(pk11_dso);
1024                pk11_dso = NULL;
1025                ENGINE_free(e_pk11);
1026                return;
1027                }

1029        ENGINE_add(e_pk11);

1031        ENGINE_free(e_pk11);
1032        ERR_clear_error();
1033        }
1034 #endif  /* ENGINE_DYNAMIC_SUPPORT */

1036 /*
1037  * These are the static string constants for the DSO file name and
1038  * the function symbol names to bind to.
1039  */
1040 static const char *PK11_LIBNAME = NULL;

1042 static const char *get_PK11_LIBNAME(void)
1043         {
1044         if (PK11_LIBNAME)
1045                 return (PK11_LIBNAME);

1047         return (def_PK11_LIBNAME);
1048         }

1050 static void free_PK11_LIBNAME(void)
1051         {
```

```
1052         if (PK11_LIBNAME)
1053                 OPENSSL_free((void*)PK11_LIBNAME);
1054
1055         PK11_LIBNAME = NULL;
1056         }
1057
1058 static long set_PK11_LIBNAME(const char *name)
1059         {
1060         free_PK11_LIBNAME();
1061
1062         return ((PK11_LIBNAME = BUF_strdup(name)) != NULL ? 1 : 0);
1063         }
1064
1065 /* acquire all engine specific mutexes before fork */
1066 static void pk11_fork_prepare(void)
1067         {
1068         int i;
1069
1070         if (!pk11_library_initialized)
1071                 return;
1072
1073         LOCK_OBJSTORE(OP_RSA);
1074         LOCK_OBJSTORE(OP_DSA);
1075         LOCK_OBJSTORE(OP_DH);
1076         for (i = 0; i < OP_MAX; i++)
1077                 {
1078                 (void) pthread_mutex_lock(session_cache[i].lock);
1079                 }
1080         }
1081
1082 /* release all engine specific mutexes */
1083 static void pk11_fork_parent(void)
1084         {
1085         int i;
1086
1087         if (!pk11_library_initialized)
1088                 return;
1089
1090         for (i = OP_MAX - 1; i >= 0; i--)
1091                 {
1092                 (void) pthread_mutex_unlock(session_cache[i].lock);
1093                 }
1094         UNLOCK_OBJSTORE(OP_DH);
1095         UNLOCK_OBJSTORE(OP_DSA);
1096         UNLOCK_OBJSTORE(OP_RSA);
1097         }
1098
1099 /*
1100  * same situation as in parent - we need to unlock all locks to make them
1101  * accessible to all threads.
1102  */
1103 static void pk11_fork_child(void)
1104         {
1105         int i;
1106
1107         if (!pk11_library_initialized)
1108                 return;
1109
1110         for (i = OP_MAX - 1; i >= 0; i--)
1111                 {
1112                 (void) pthread_mutex_unlock(session_cache[i].lock);
1113                 }
1114         UNLOCK_OBJSTORE(OP_DH);
1115         UNLOCK_OBJSTORE(OP_DSA);
1116         UNLOCK_OBJSTORE(OP_RSA);
1117         }
```



```
1119 /* Initialization function for the pk11 engine */
1120 static int pk11_init(ENGINE *e)
1121 {
1122         return (pk11_library_init(e));
1123 }
1124
1125 /*
1126  * Initialization function. Sets up various PKCS#11 library components.
1127  * It selects a slot based on predefined critiera. In the process, it also
1128  * count how many ciphers and digests to support. Since the cipher and
1129  * digest information is needed when setting default engine, this function
1130  * needs to be called before calling ENGINE_set_default.
1131  */
1132 /* ARGSUSED */
1133 static int pk11_library_init(ENGINE *e)
1134         {
1135         CK_C_GetFunctionList p;
1136         CK_RV rv = CKR_OK;
1137         CK_INFO info;
1138         CK_ULONG ul_state_len;
1139         int any_slot_found;
1140         int i;
1141
1142         /*
1143          * pk11_library_initialized is set to 0 in pk11_finish() which is called
1144          * from ENGINE_finish(). However, if there is still at least one
1145          * existing functional reference to the engine (see engine(3) for more
1146          * information), pk11_finish() is skipped. For example, this can happen
1147          * if an application forgets to clear one cipher context. In case of a
1148          * fork() when the application is finishing the engine so that it can be
1149          * reinitialized in the child, forgotten functional reference causes
1150          * pk11_library_initialized to stay 1. In that case we need the PID
1151          * check so that we properly initialize the engine again.
1152          */
1153         if (pk11_library_initialized)
1154                 {
1155                 if (pk11_pid == getpid())
1156                         {
1157                         return (1);
1158                         }
1159                 else
1160                         {
1161                         global_session = CK_INVALID_HANDLE;
1162                         /*
1163                          * free the locks first to prevent memory leak in case
1164                          * the application calls fork() without finishing the
1165                          * engine first.
1166                          */
1167                         pk11_free_all_locks();
1168                         }
1169                 }
1170
1171         if (pk11_dso == NULL)
1172                 {
1173                 PK11err(PK11_F_LIBRARY_INIT, PK11_R_DSO_FAILURE);
1174                 goto err;
1175                 }
1176
1177 #ifdef  SOLARIS_AES_CTR
1178         /*
1179          * We must do this before we start working with slots since we need all
1180          * NIDs there.
1181          */
1182         if (pk11_add_aes_ctr_NIDs() == 0)
1183                 goto err;
```

```
#endif  /* SOLARIS_AES_CTR */

#ifdef  SOLARIS_HW_SLOT_SELECTION
        if (check_hw_mechanisms() == 0)
                goto err;
#endif  /* SOLARIS_HW_SLOT_SELECTION */

        /* get the C_GetFunctionList function from the loaded library */
        p = (CK_C_GetFunctionList)DSO_bind_func(pk11_dso,
                PK11_GET_FUNCTION_LIST);
        if (!p)
                {
                PK11err(PK11_F_LIBRARY_INIT, PK11_R_DSO_FAILURE);
                goto err;
                }

        /* get the full function list from the loaded library */
        rv = p(&pFuncList);
        if (rv != CKR_OK)
                {
                PK11err_add_data(PK11_F_LIBRARY_INIT, PK11_R_DSO_FAILURE, rv);
                goto err;
                }

        rv = pFuncList->C_Initialize(NULL_PTR);
        if ((rv != CKR_OK) && (rv != CKR_CRYPTOKI_ALREADY_INITIALIZED))
                {
                PK11err_add_data(PK11_F_LIBRARY_INIT, PK11_R_INITIALIZE, rv);
                goto err;
                }

        rv = pFuncList->C_GetInfo(&info);
        if (rv != CKR_OK)
                {
                PK11err_add_data(PK11_F_LIBRARY_INIT, PK11_R_GETINFO, rv);
                goto err;
                }

        if (pk11_choose_slots(&any_slot_found) == 0)
                goto err;

        /*
         * The library we use, set in def_PK11_LIBNAME, may not offer any
         * slot(s). In that case, we must not proceed but we must not return an
         * error. The reason is that applications that try to set up the PKCS#11
         * engine don't exit on error during the engine initialization just
         * because no slot was present.
         */
        if (any_slot_found == 0)
                return (1);

        if (global_session == CK_INVALID_HANDLE)
                {
                /* Open the global_session for the new process */
                rv = pFuncList->C_OpenSession(SLOTID, CKF_SERIAL_SESSION,
                        NULL_PTR, NULL_PTR, &global_session);
                if (rv != CKR_OK)
                        {
                        PK11err_add_data(PK11_F_LIBRARY_INIT,
                            PK11_R_OPENSESSION, rv);
                        goto err;
                        }
                }

        /*
         * Disable digest if C_GetOperationState is not supported since
```



```
         * this function is required by OpenSSL digest copy function
         */
        if (pFuncList->C_GetOperationState(global_session, NULL, &ul_state_len)
                        == CKR_FUNCTION_NOT_SUPPORTED) {
#ifdef  DEBUG_SLOT_SELECTION
                fprintf(stderr, "%s: C_GetOperationState() not supported, "
                    "setting digest_count to 0\n", PK11_DBG);
#endif  /* DEBUG_SLOT_SELECTION */
                digest_count = 0;
        }

        pk11_library_initialized = TRUE;
        pk11_pid = getpid();
        /*
         * if initialization of the locks fails pk11_init_all_locks()
         * will do the cleanup.
         */
        if (!pk11_init_all_locks())
                goto err;
        for (i = 0; i < OP_MAX; i++)
                session_cache[i].head = NULL;
        /*
         * initialize active lists. We only use active lists
         * for asymmetric ciphers.
         */
        for (i = 0; i < OP_MAX; i++)
                active_list[i] = NULL;

        if (!pk11_atfork_initialized)
                {
                if (pthread_atfork(pk11_fork_prepare, pk11_fork_parent,
                    pk11_fork_child) != 0)
                        {
                        PK11err(PK11_F_LIBRARY_INIT, PK11_R_ATFORK_FAILED);
                        goto err;
                        }
                pk11_atfork_initialized = TRUE;
                }

        return (1);

err:
        return (0);
        }

/* Destructor (complements the "ENGINE_pk11()" constructor) */
/* ARGSUSED */
static int pk11_destroy(ENGINE *e)
        {
        free_PK11_LIBNAME();
        ERR_unload_pk11_strings();
        return (1);
        }

/*
 * Termination function to clean up the session, the token, and the pk11
 * library.
 */
/* ARGSUSED */
static int pk11_finish(ENGINE *e)
        {
        int i;

        if (pk11_dso == NULL)
                {
                PK11err(PK11_F_FINISH, PK11_R_NOT_LOADED);
```

```
1316                goto err;
1317                }

1319        OPENSSL_assert(pFuncList != NULL);

1321        if (pk11_free_all_sessions() == 0)
1322                goto err;

1324        /* free all active lists */
1325        for (i = 0; i < OP_MAX; i++)
1326                pk11_free_active_list(i);

1328        pFuncList->C_CloseSession(global_session);
1329        global_session = CK_INVALID_HANDLE;

1331        /*
1332         * Since we are part of a library (libcrypto.so), calling this function
1333         * may have side-effects.
1334         */
1335 #if 0
1336        pFuncList->C_Finalize(NULL);
1337 #endif

1339        if (!DSO_free(pk11_dso))
1340                {
1341                PK11err(PK11_F_FINISH, PK11_R_DSO_FAILURE);
1342                goto err;
1343                }
1344        pk11_dso = NULL;
1345        pFuncList = NULL;
1346        pk11_library_initialized = FALSE;
1347        pk11_pid = 0;
1348        /*
1349         * There is no way how to unregister atfork handlers (other than
1350         * unloading the library) so we just free the locks. For this reason
1351         * the atfork handlers check if the engine is initialized and bail out
1352         * immediately if not. This is necessary in case a process finishes
1353         * the engine before calling fork().
1354         */
1355        pk11_free_all_locks();

1357        return (1);

1359 err:
1360        return (0);
1361        }

1363 /* Standard engine interface function to set the dynamic library path */
1364 /* ARGSUSED */
1365 static int pk11_ctrl(ENGINE *e, int cmd, long i, void *p, void (*f)())
1366        {
1367        int initialized = ((pk11_dso == NULL) ? 0 : 1);

1369        switch (cmd)
1370                {
1371        case PK11_CMD_SO_PATH:
1372                if (p == NULL)
1373                        {
1374                        PK11err(PK11_F_CTRL, ERR_R_PASSED_NULL_PARAMETER);
1375                        return (0);
1376                        }

1378                if (initialized)
1379                        {
1380                        PK11err(PK11_F_CTRL, PK11_R_ALREADY_LOADED);
1381                        return (0);
```



```
1382                        }

1384                return (set_PK11_LIBNAME((const char *)p));
1385        default:
1386                break;
1387                }

1389        PK11err(PK11_F_CTRL, PK11_R_CTRL_COMMAND_NOT_IMPLEMENTED);

1391        return (0);
1392        }


1395 /* Required function by the engine random interface. It does nothing here */
1396 static void pk11_rand_cleanup(void)
1397        {
1398        return;
1399        }

1401 /* ARGSUSED */
1402 static void pk11_rand_add(const void *buf, int num, double add)
1403        {
1404        PK11_SESSION *sp;

1406        if ((sp = pk11_get_session(OP_RAND)) == NULL)
1407                return;

1409        /*
1410         * Ignore any errors (e.g. CKR_RANDOM_SEED_NOT_SUPPORTED) since
1411         * the calling functions do not care anyway
1412         */
1413        pFuncList->C_SeedRandom(sp->session, (unsigned char *) buf, num);
1414        pk11_return_session(sp, OP_RAND);

1416        return;
1417        }

1419 static void pk11_rand_seed(const void *buf, int num)
1420        {
1421        pk11_rand_add(buf, num, 0);
1422        }

1424 static int pk11_rand_bytes(unsigned char *buf, int num)
1425        {
1426        CK_RV rv;
1427        PK11_SESSION *sp;

1429        if ((sp = pk11_get_session(OP_RAND)) == NULL)
1430                return (0);

1432        rv = pFuncList->C_GenerateRandom(sp->session, buf, num);
1433        if (rv != CKR_OK)
1434                {
1435                PK11err_add_data(PK11_F_RAND_BYTES, PK11_R_GENERATERANDOM, rv);
1436                pk11_return_session(sp, OP_RAND);
1437                return (0);
1438                }

1440        pk11_return_session(sp, OP_RAND);
1441        return (1);
1442        }

1444 /* Required function by the engine random interface. It does nothing here */
1445 static int pk11_rand_status(void)
1446        {
1447        return (1);
```

```
1448         }

1450 /* Free all BIGNUM structures from PK11_SESSION. */
1451 static void pk11_free_nums(PK11_SESSION *sp, PK11_OPTYPE optype)
1452         {
1453         switch (optype)
1454                 {
1455 #ifndef OPENSSL_NO_RSA
1456                 case OP_RSA:
1457                         if (sp->opdata_rsa_n_num != NULL)
1458                                 {
1459                                 BN_free(sp->opdata_rsa_n_num);
1460                                 sp->opdata_rsa_n_num = NULL;
1461                                 }
1462                         if (sp->opdata_rsa_e_num != NULL)
1463                                 {
1464                                 BN_free(sp->opdata_rsa_e_num);
1465                                 sp->opdata_rsa_e_num = NULL;
1466                                 }
1467                         if (sp->opdata_rsa_d_num != NULL)
1468                                 {
1469                                 BN_free(sp->opdata_rsa_d_num);
1470                                 sp->opdata_rsa_d_num = NULL;
1471                                 }
1472                         break;
1473 #endif
1474 #ifndef OPENSSL_NO_DSA
1475                 case OP_DSA:
1476                         if (sp->opdata_dsa_pub_num != NULL)
1477                                 {
1478                                 BN_free(sp->opdata_dsa_pub_num);
1479                                 sp->opdata_dsa_pub_num = NULL;
1480                                 }
1481                         if (sp->opdata_dsa_priv_num != NULL)
1482                                 {
1483                                 BN_free(sp->opdata_dsa_priv_num);
1484                                 sp->opdata_dsa_priv_num = NULL;
1485                                 }
1486                         break;
1487 #endif
1488 #ifndef OPENSSL_NO_DH
1489                 case OP_DH:
1490                         if (sp->opdata_dh_priv_num != NULL)
1491                                 {
1492                                 BN_free(sp->opdata_dh_priv_num);
1493                                 sp->opdata_dh_priv_num = NULL;
1494                                 }
1495                         break;
1496 #endif
1497                 default:
1498                         break;
1499                 }
1500         }

1502 /*
1503  * Get new PK11_SESSION structure ready for use. Every process must have
1504  * its own freelist of PK11_SESSION structures so handle fork() here
1505  * by destroying the old and creating new freelist.
1506  * The returned PK11_SESSION structure is disconnected from the freelist.
1507  */
1508 PK11_SESSION *
1509 pk11_get_session(PK11_OPTYPE optype)
1510         {
1511         PK11_SESSION *sp = NULL, *sp1, *freelist;
1512         pthread_mutex_t *freelist_lock;
1513         CK_RV rv;
```



```
1515         switch (optype)
1516                 {
1517                 case OP_RSA:
1518                 case OP_DSA:
1519                 case OP_DH:
1520                 case OP_RAND:
1521                 case OP_DIGEST:
1522                 case OP_CIPHER:
1523                         freelist_lock = session_cache[optype].lock;
1524                         break;
1525                 default:
1526                         PK11err(PK11_F_GET_SESSION,
1527                                 PK11_R_INVALID_OPERATION_TYPE);
1528                         return (NULL);
1529                 }
1530         (void) pthread_mutex_lock(freelist_lock);
1531         freelist = session_cache[optype].head;
1532         sp = freelist;

1534         /*
1535          * If the free list is empty, allocate new unitialized (filled
1536          * with zeroes) PK11_SESSION structure otherwise return first
1537          * structure from the freelist.
1538          */
1539         if (sp == NULL)
1540                 {
1541                 if ((sp = OPENSSL_malloc(sizeof (PK11_SESSION))) == NULL)
1542                         {
1543                         PK11err(PK11_F_GET_SESSION,
1544                                 PK11_R_MALLOC_FAILURE);
1545                         goto err;
1546                         }
1547                 (void) memset(sp, 0, sizeof (PK11_SESSION));
1548                 }
1549         else
1550                 {
1551                 freelist = sp->next;
1552                 }

1554         if (sp->pid != 0 && sp->pid != getpid())
1555                 {
1556                 /*
1557                  * We are a new process and thus need to free any inherited
1558                  * PK11_SESSION objects.
1559                  */
1560                 while ((sp1 = freelist) != NULL)
1561                         {
1562                         freelist = sp1->next;
1563                         /*
1564                          * NOTE: we do not want to call pk11_free_all_sessions()
1565                          * here because it would close underlying PKCS#11
1566                          * sessions and destroy all objects.
1567                          */
1568                         pk11_free_nums(sp1, optype);
1569                         OPENSSL_free(sp1);
1570                         }

1572                 /* we have to free the active list as well. */
1573                 pk11_free_active_list(optype);

1575                 /* Initialize the process */
1576                 rv = pFuncList->C_Initialize(NULL_PTR);
1577                 if ((rv != CKR_OK) && (rv != CKR_CRYPTOKI_ALREADY_INITIALIZED))
1578                         {
1579                         PK11err_add_data(PK11_F_GET_SESSION, PK11_R_INITIALIZE,
```

```
1580                            rv);
1581                    OPENSSL_free(sp);
1582                    sp = NULL;
1583                    goto err;
1584                    }

1586            /*
1587             * Choose slot here since the slot table is different on this
1588             * process. If we are here then we must have found at least one
1589             * usable slot before so we don't need to check any_slot_found.
1590             * See pk11_library_init()'s usage of this function for more
1591             * information.
1592             */
1593 #ifdef  SOLARIS_HW_SLOT_SELECTION
1594            if (check_hw_mechanisms() == 0)
1595                    goto err;
1596 #endif  /* SOLARIS_HW_SLOT_SELECTION */
1597            if (pk11_choose_slots(NULL) == 0)
1598                    goto err;

1600            /* Open the global_session for the new process */
1601            rv = pFuncList->C_OpenSession(SLOTID, CKF_SERIAL_SESSION,
1602                    NULL_PTR, NULL_PTR, &global_session);
1603            if (rv != CKR_OK)
1604                    {
1605                    PK11err_add_data(PK11_F_GET_SESSION, PK11_R_OPENSESSION,
1606                            rv);
1607                    OPENSSL_free(sp);
1608                    sp = NULL;
1609                    goto err;
1610                    }

1612            /* It is an inherited session and needs re-initialization. */
1613            if (pk11_setup_session(sp, optype) == 0)
1614                    {
1615                    OPENSSL_free(sp);
1616                    sp = NULL;
1617                    }
1618            }
1619        if (sp->pid == 0)
1620                {
1621                /* It is a new session and needs initialization. */
1622                if (pk11_setup_session(sp, optype) == 0)
1623                        {
1624                        OPENSSL_free(sp);
1625                        sp = NULL;
1626                        }
1627                }

1629        /* set new head for the list of PK11_SESSION objects */
1630        session_cache[optype].head = freelist;

1632 err:
1633        if (sp != NULL)
1634                sp->next = NULL;

1636        (void) pthread_mutex_unlock(freelist_lock);

1638        return (sp);
1639        }


1642 void
1643 pk11_return_session(PK11_SESSION *sp, PK11_OPTYPE optype)
1644        {
1645        pthread_mutex_t *freelist_lock;
```



```
1646        PK11_SESSION *freelist;

1648        if (sp == NULL || sp->pid != getpid())
1649                return;

1651        switch (optype)
1652                {
1653                case OP_RSA:
1654                case OP_DSA:
1655                case OP_DH:
1656                case OP_RAND:
1657                case OP_DIGEST:
1658                case OP_CIPHER:
1659                        freelist_lock = session_cache[optype].lock;
1660                        break;
1661                default:
1662                        PK11err(PK11_F_RETURN_SESSION,
1663                                PK11_R_INVALID_OPERATION_TYPE);
1664                        return;
1665                }

1667        (void) pthread_mutex_lock(freelist_lock);
1668        freelist = session_cache[optype].head;
1669        sp->next = freelist;
1670        session_cache[optype].head = sp;
1671        (void) pthread_mutex_unlock(freelist_lock);
1672        }


1675 /* Destroy all objects. This function is called when the engine is finished */
1676 static int pk11_free_all_sessions()
1677        {
1678        int ret = 1;
1679        int type;

1681 #ifndef OPENSSL_NO_RSA
1682        (void) pk11_destroy_rsa_key_objects(NULL);
1683 #endif  /* OPENSSL_NO_RSA */
1684 #ifndef OPENSSL_NO_DSA
1685        (void) pk11_destroy_dsa_key_objects(NULL);
1686 #endif  /* OPENSSL_NO_DSA */
1687 #ifndef OPENSSL_NO_DH
1688        (void) pk11_destroy_dh_key_objects(NULL);
1689 #endif  /* OPENSSL_NO_DH */
1690        (void) pk11_destroy_cipher_key_objects(NULL);

1692        /*
1693         * We try to release as much as we can but any error means that we will
1694         * return 0 on exit.
1695         */
1696        for (type = 0; type < OP_MAX; type++)
1697                {
1698                if (pk11_free_session_list(type) == 0)
1699                        ret = 0;
1700                }

1702        return (ret);
1703        }

1705 /*
1706  * Destroy session structures from the linked list specified. Free as many
1707  * sessions as possible but any failure in C_CloseSession() means that we
1708  * return an error on return.
1709  */
1710 static int pk11_free_session_list(PK11_OPTYPE optype)
1711        {
```

```
1712         CK_RV rv;
1713         PK11_SESSION *sp = NULL;
1714         PK11_SESSION *freelist = NULL;
1715         pid_t mypid = getpid();
1716         pthread_mutex_t *freelist_lock;
1717         int ret = 1;

1719         switch (optype)
1720                 {
1721                 case OP_RSA:
1722                 case OP_DSA:
1723                 case OP_DH:
1724                 case OP_RAND:
1725                 case OP_DIGEST:
1726                 case OP_CIPHER:
1727                         freelist_lock = session_cache[optype].lock;
1728                         break;
1729                 default:
1730                         PK11err(PK11_F_FREE_ALL_SESSIONS,
1731                                 PK11_R_INVALID_OPERATION_TYPE);
1732                         return (0);
1733                 }

1735         (void) pthread_mutex_lock(freelist_lock);
1736         freelist = session_cache[optype].head;
1737         while ((sp = freelist) != NULL)
1738                 {
1739                 if (sp->session != CK_INVALID_HANDLE && sp->pid == mypid)
1740                         {
1741                         rv = pFuncList->C_CloseSession(sp->session);
1742                         if (rv != CKR_OK)
1743                                 {
1744                                 PK11err_add_data(PK11_F_FREE_ALL_SESSIONS,
1745                                         PK11_R_CLOSESESSION, rv);
1746                                 ret = 0;
1747                                 }
1748                         }
1749                 freelist = sp->next;
1750                 pk11_free_nums(sp, optype);
1751                 OPENSSL_free(sp);
1752                 }

1754         (void) pthread_mutex_unlock(freelist_lock);
1755         return (ret);
1756         }


1759 static int pk11_setup_session(PK11_SESSION *sp, PK11_OPTYPE optype)
1760         {
1761         CK_RV rv;
1762         CK_SLOT_ID myslot;

1764         switch (optype)
1765                 {
1766                 case OP_RSA:
1767                 case OP_DSA:
1768                 case OP_DH:
1769                         myslot = pubkey_SLOTID;
1770                         break;
1771                 case OP_RAND:
1772                         myslot = rand_SLOTID;
1773                         break;
1774                 case OP_DIGEST:
1775                 case OP_CIPHER:
1776                         myslot = SLOTID;
1777                         break;
```



```
1778                 default:
1779                         PK11err(PK11_F_SETUP_SESSION,
1780                             PK11_R_INVALID_OPERATION_TYPE);
1781                         return (0);
1782                 }

1784         sp->session = CK_INVALID_HANDLE;
1785 #ifdef  DEBUG_SLOT_SELECTION
1786         fprintf(stderr, "%s: myslot=%d optype=%d\n", PK11_DBG, myslot, optype);
1787 #endif  /* DEBUG_SLOT_SELECTION */
1788         rv = pFuncList->C_OpenSession(myslot, CKF_SERIAL_SESSION,
1789                 NULL_PTR, NULL_PTR, &sp->session);
1790         if (rv == CKR_CRYPTOKI_NOT_INITIALIZED)
1791                 {
1792                 /*
1793                  * We are probably a child process so force the
1794                  * reinitialize of the session
1795                  */
1796                 pk11_library_initialized = FALSE;
1797                 if (!pk11_library_init(NULL))
1798                         return (0);
1799                 rv = pFuncList->C_OpenSession(myslot, CKF_SERIAL_SESSION,
1800                         NULL_PTR, NULL_PTR, &sp->session);
1801                 }
1802         if (rv != CKR_OK)
1803                 {
1804                 PK11err_add_data(PK11_F_SETUP_SESSION, PK11_R_OPENSESSION, rv);
1805                 return (0);
1806                 }

1808         sp->pid = getpid();

1810         switch (optype)
1811                 {
1812 #ifndef OPENSSL_NO_RSA
1813                 case OP_RSA:
1814                         sp->opdata_rsa_pub_key = CK_INVALID_HANDLE;
1815                         sp->opdata_rsa_priv_key = CK_INVALID_HANDLE;
1816                         sp->opdata_rsa_pub = NULL;
1817                         sp->opdata_rsa_n_num = NULL;
1818                         sp->opdata_rsa_e_num = NULL;
1819                         sp->opdata_rsa_priv = NULL;
1820                         sp->opdata_rsa_d_num = NULL;
1821                         break;
1822 #endif  /* OPENSSL_NO_RSA */
1823 #ifndef OPENSSL_NO_DSA
1824                 case OP_DSA:
1825                         sp->opdata_dsa_pub_key = CK_INVALID_HANDLE;
1826                         sp->opdata_dsa_priv_key = CK_INVALID_HANDLE;
1827                         sp->opdata_dsa_pub = NULL;
1828                         sp->opdata_dsa_pub_num = NULL;
1829                         sp->opdata_dsa_priv = NULL;
1830                         sp->opdata_dsa_priv_num = NULL;
1831                         break;
1832 #endif  /* OPENSSL_NO_DSA */
1833 #ifndef OPENSSL_NO_DH
1834                 case OP_DH:
1835                         sp->opdata_dh_key = CK_INVALID_HANDLE;
1836                         sp->opdata_dh = NULL;
1837                         sp->opdata_dh_priv_num = NULL;
1838                         break;
1839 #endif  /* OPENSSL_NO_DH */
1840                 case OP_CIPHER:
1841                         sp->opdata_cipher_key = CK_INVALID_HANDLE;
1842                         sp->opdata_encrypt = -1;
1843                         break;
```

```
1844                }

1846        return (1);
1847        }

1849 #ifndef OPENSSL_NO_RSA
1850 /* Destroy RSA public key from single session. */
1851 int
1852 pk11_destroy_rsa_object_pub(PK11_SESSION *sp, CK_BBOOL uselock)
1853         {
1854         int ret = 0;

1856         if (sp->opdata_rsa_pub_key != CK_INVALID_HANDLE)
1857                 {
1858                 TRY_OBJ_DESTROY(sp->session, sp->opdata_rsa_pub_key,
1859                     ret, uselock, OP_RSA);
1860                 sp->opdata_rsa_pub_key = CK_INVALID_HANDLE;
1861                 sp->opdata_rsa_pub = NULL;
1862                 if (sp->opdata_rsa_n_num != NULL)
1863                         {
1864                         BN_free(sp->opdata_rsa_n_num);
1865                         sp->opdata_rsa_n_num = NULL;
1866                         }
1867                 if (sp->opdata_rsa_e_num != NULL)
1868                         {
1869                         BN_free(sp->opdata_rsa_e_num);
1870                         sp->opdata_rsa_e_num = NULL;
1871                         }
1872                 }

1874         return (ret);
1875         }

1877 /* Destroy RSA private key from single session. */
1878 int
1879 pk11_destroy_rsa_object_priv(PK11_SESSION *sp, CK_BBOOL uselock)
1880         {
1881         int ret = 0;

1883         if (sp->opdata_rsa_priv_key != CK_INVALID_HANDLE)
1884                 {
1885                 TRY_OBJ_DESTROY(sp->session, sp->opdata_rsa_priv_key,
1886                     ret, uselock, OP_RSA);
1887                 sp->opdata_rsa_priv_key = CK_INVALID_HANDLE;
1888                 sp->opdata_rsa_priv = NULL;
1889                 if (sp->opdata_rsa_d_num != NULL)
1890                         {
1891                         BN_free(sp->opdata_rsa_d_num);
1892                         sp->opdata_rsa_d_num = NULL;
1893                         }
1894                 }

1896         return (ret);
1897         }

1899 /*
1900  * Destroy RSA key object wrapper. If session is NULL, try to destroy all
1901  * objects in the free list.
1902  */
1903 int
1904 pk11_destroy_rsa_key_objects(PK11_SESSION *session)
1905         {
1906         int ret = 1;
1907         PK11_SESSION *sp = NULL;
1908         PK11_SESSION *local_free_session;
1909         CK_BBOOL uselock = TRUE;
```



```
1911         if (session != NULL)
1912                 local_free_session = session;
1913         else
1914                 {
1915                 (void) pthread_mutex_lock(session_cache[OP_RSA].lock);
1916                 local_free_session = session_cache[OP_RSA].head;
1917                 uselock = FALSE;
1918                 }

1920         /*
1921          * go through the list of sessions and delete key objects
1922          */
1923         while ((sp = local_free_session) != NULL)
1924                 {
1925                 local_free_session = sp->next;

1927                 /*
1928                  * Do not terminate list traversal if one of the
1929                  * destroy operations fails.
1930                  */
1931                 if (pk11_destroy_rsa_object_pub(sp, uselock) == 0)
1932                         {
1933                         ret = 0;
1934                         continue;
1935                         }
1936                 if (pk11_destroy_rsa_object_priv(sp, uselock) == 0)
1937                         {
1938                         ret = 0;
1939                         continue;
1940                         }
1941                 }

1943         if (session == NULL)
1944                 (void) pthread_mutex_unlock(session_cache[OP_RSA].lock);

1946         return (ret);
1947         }
1948 #endif  /* OPENSSL_NO_RSA */

1950 #ifndef OPENSSL_NO_DSA
1951 /* Destroy DSA public key from single session. */
1952 int
1953 pk11_destroy_dsa_object_pub(PK11_SESSION *sp, CK_BBOOL uselock)
1954         {
1955         int ret = 0;

1957         if (sp->opdata_dsa_pub_key != CK_INVALID_HANDLE)
1958                 {
1959                 TRY_OBJ_DESTROY(sp->session, sp->opdata_dsa_pub_key,
1960                     ret, uselock, OP_DSA);
1961                 sp->opdata_dsa_pub_key = CK_INVALID_HANDLE;
1962                 sp->opdata_dsa_pub = NULL;
1963                 if (sp->opdata_dsa_pub_num != NULL)
1964                         {
1965                         BN_free(sp->opdata_dsa_pub_num);
1966                         sp->opdata_dsa_pub_num = NULL;
1967                         }
1968                 }

1970         return (ret);
1971         }

1973 /* Destroy DSA private key from single session. */
1974 int
1975 pk11_destroy_dsa_object_priv(PK11_SESSION *sp, CK_BBOOL uselock)
```

```
1976         {
1977         int ret = 0;

1979         if (sp->opdata_dsa_priv_key != CK_INVALID_HANDLE)
1980                 {
1981                 TRY_OBJ_DESTROY(sp->session, sp->opdata_dsa_priv_key,
1982                     ret, uselock, OP_DSA);
1983                 sp->opdata_dsa_priv_key = CK_INVALID_HANDLE;
1984                 sp->opdata_dsa_priv = NULL;
1985                 if (sp->opdata_dsa_priv_num != NULL)
1986                         {
1987                         BN_free(sp->opdata_dsa_priv_num);
1988                         sp->opdata_dsa_priv_num = NULL;
1989                         }
1990                 }

1992         return (ret);
1993         }

1995 /*
1996  * Destroy DSA key object wrapper. If session is NULL, try to destroy all
1997  * objects in the free list.
1998  */
1999 int
2000 pk11_destroy_dsa_key_objects(PK11_SESSION *session)
2001         {
2002         int ret = 1;
2003         PK11_SESSION *sp = NULL;
2004         PK11_SESSION *local_free_session;
2005         CK_BBOOL uselock = TRUE;

2007         if (session != NULL)
2008                 local_free_session = session;
2009         else
2010                 {
2011                 (void) pthread_mutex_lock(session_cache[OP_DSA].lock);
2012                 local_free_session = session_cache[OP_DSA].head;
2013                 uselock = FALSE;
2014                 }

2016         /*
2017          * go through the list of sessions and delete key objects
2018          */
2019         while ((sp = local_free_session) != NULL)
2020                 {
2021                 local_free_session = sp->next;

2023                 /*
2024                  * Do not terminate list traversal if one of the
2025                  * destroy operations fails.
2026                  */
2027                 if (pk11_destroy_dsa_object_pub(sp, uselock) == 0)
2028                         {
2029                         ret = 0;
2030                         continue;
2031                         }
2032                 if (pk11_destroy_dsa_object_priv(sp, uselock) == 0)
2033                         {
2034                         ret = 0;
2035                         continue;
2036                         }
2037                 }

2039         if (session == NULL)
2040                 (void) pthread_mutex_unlock(session_cache[OP_DSA].lock);
```



```
2042         return (ret);
2043         }
2044 #endif  /* OPENSSL_NO_DSA */

2046 #ifndef OPENSSL_NO_DH
2047 /* Destroy DH key from single session. */
2048 int
2049 pk11_destroy_dh_object(PK11_SESSION *sp, CK_BBOOL uselock)
2050         {
2051         int ret = 0;

2053         if (sp->opdata_dh_key != CK_INVALID_HANDLE)
2054                 {
2055                 TRY_OBJ_DESTROY(sp->session, sp->opdata_dh_key,
2056                     ret, uselock, OP_DH);
2057                 sp->opdata_dh_key = CK_INVALID_HANDLE;
2058                 sp->opdata_dh = NULL;
2059                 if (sp->opdata_dh_priv_num != NULL)
2060                         {
2061                         BN_free(sp->opdata_dh_priv_num);
2062                         sp->opdata_dh_priv_num = NULL;
2063                         }
2064                 }

2066         return (ret);
2067         }

2069 /*
2070  * Destroy DH key object wrapper.
2071  *
2072  * arg0: pointer to PKCS#11 engine session structure
2073  *       if session is NULL, try to destroy all objects in the free list
2074  */
2075 int
2076 pk11_destroy_dh_key_objects(PK11_SESSION *session)
2077         {
2078         int ret = 1;
2079         PK11_SESSION *sp = NULL;
2080         PK11_SESSION *local_free_session;
2081         CK_BBOOL uselock = TRUE;

2083         if (session != NULL)
2084                 local_free_session = session;
2085         else
2086                 {
2087                 (void) pthread_mutex_lock(session_cache[OP_DH].lock);
2088                 local_free_session = session_cache[OP_DH].head;
2089                 uselock = FALSE;
2090                 }

2092         while ((sp = local_free_session) != NULL)
2093                 {
2094                 local_free_session = sp->next;

2096                 /*
2097                  * Do not terminate list traversal if one of the
2098                  * destroy operations fails.
2099                  */
2100                 if (pk11_destroy_dh_object(sp, uselock) == 0)
2101                         {
2102                         ret = 0;
2103                         continue;
2104                         }
2105                 }
2106         if (session == NULL)
2107                 (void) pthread_mutex_unlock(session_cache[OP_DH].lock);
```

```
2109         return (ret);
2110         }
2111 #endif  /* OPENSSL_NO_DH */

2113 static int pk11_destroy_object(CK_SESSION_HANDLE session, CK_OBJECT_HANDLE oh)
2114         {
2115         CK_RV rv;
2116         rv = pFuncList->C_DestroyObject(session, oh);
2117         if (rv != CKR_OK)
2118                 {
2119                 PK11err_add_data(PK11_F_DESTROY_OBJECT, PK11_R_DESTROYOBJECT,
2120                     rv);
2121                 return (0);
2122                 }

2124         return (1);
2125         }


2128 /* Symmetric ciphers and digests support functions */

2130 static int
2131 cipher_nid_to_pk11(int nid)
2132         {
2133         int i;

2135         for (i = 0; i < PK11_CIPHER_MAX; i++)
2136                 if (ciphers[i].nid == nid)
2137                         return (ciphers[i].id);
2138         return (-1);
2139         }

2141 static int
2142 pk11_usable_ciphers(const int **nids)
2143         {
2144         if (cipher_count > 0)
2145                 *nids = cipher_nids;
2146         else
2147                 *nids = NULL;
2148         return (cipher_count);
2149         }

2151 static int
2152 pk11_usable_digests(const int **nids)
2153         {
2154         if (digest_count > 0)
2155                 *nids = digest_nids;
2156         else
2157                 *nids = NULL;
2158         return (digest_count);
2159         }

2161 /*
2162  * Init context for encryption or decryption using a symmetric key.
2163  */
2164 static int pk11_init_symmetric(EVP_CIPHER_CTX *ctx, PK11_CIPHER *pcipher,
2165         PK11_SESSION *sp, CK_MECHANISM_PTR pmech)
2166         {
2167         CK_RV rv;
2168 #ifdef  SOLARIS_AES_CTR
2169         CK_AES_CTR_PARAMS ctr_params;
2170 #endif  /* SOLARIS_AES_CTR */

2172         /*
2173          * We expect pmech->mechanism to be already set and
```



```
2174          * pParameter/ulParameterLen initialized to NULL/0 before
2175          * pk11_init_symetric() is called.
2176          */
2177         OPENSSL_assert(pmech->mechanism != NULL);
2178         OPENSSL_assert(pmech->pParameter == NULL);
2179         OPENSSL_assert(pmech->ulParameterLen == 0);

2181 #ifdef  SOLARIS_AES_CTR
2182         if (ctx->cipher->nid == NID_aes_128_ctr ||
2183             ctx->cipher->nid == NID_aes_192_ctr ||
2184             ctx->cipher->nid == NID_aes_256_ctr)
2185                 {
2186                 pmech->pParameter = (void *)(&ctr_params);
2187                 pmech->ulParameterLen = sizeof (ctr_params);
2188                 /*
2189                  * For now, we are limited to the fixed length of the counter,
2190                  * it covers the whole counter block. That's what RFC 4344
2191                  * needs. For more information on internal structure of the
2192                  * counter block, see RFC 3686. If needed in the future, we can
2193                  * add code so that the counter length can be set via
2194                  * ENGINE_ctrl() function.
2195                  */
2196                 ctr_params.ulCounterBits = AES_BLOCK_SIZE * 8;
2197                 OPENSSL_assert(pcipher->iv_len == AES_BLOCK_SIZE);
2198                 (void) memcpy(ctr_params.cb, ctx->iv, AES_BLOCK_SIZE);
2199                 }
2200         else
2201 #endif  /* SOLARIS_AES_CTR */
2202                 {
2203                 if (pcipher->iv_len > 0)
2204                         {
2205                         pmech->pParameter = (void *)ctx->iv;
2206                         pmech->ulParameterLen = pcipher->iv_len;
2207                         }
2208                 }

2210         /* if we get here, the encryption needs to be reinitialized */
2211         if (ctx->encrypt)
2212                 rv = pFuncList->C_EncryptInit(sp->session, pmech,
2213                         sp->opdata_cipher_key);
2214         else
2215                 rv = pFuncList->C_DecryptInit(sp->session, pmech,
2216                         sp->opdata_cipher_key);

2218         if (rv != CKR_OK)
2219                 {
2220                 PK11err_add_data(PK11_F_CIPHER_INIT, ctx->encrypt ?
2221                     PK11_R_ENCRYPTINIT : PK11_R_DECRYPTINIT, rv);
2222                 pk11_return_session(sp, OP_CIPHER);
2223                 return (0);
2224                 }

2226         return (1);
2227         }

2229 /* ARGSUSED */
2230 static int
2231 pk11_cipher_init(EVP_CIPHER_CTX *ctx, const unsigned char *key,
2232     const unsigned char *iv, int enc)
2233         {
2234         CK_MECHANISM mech;
2235         int index;
2236         PK11_CIPHER_STATE *state = (PK11_CIPHER_STATE *) ctx->cipher_data;
2237         PK11_SESSION *sp;
2238         PK11_CIPHER *p_ciph_table_row;
```

```
2240         state->sp = NULL;

2242         index = cipher_nid_to_pk11(ctx->cipher->nid);
2243         if (index < 0 || index >= PK11_CIPHER_MAX)
2244                 return (0);

2246         p_ciph_table_row = &ciphers[index];
2247         /*
2248          * iv_len in the ctx->cipher structure is the maximum IV length for the
2249          * current cipher and it must be less or equal to the IV length in our
2250          * ciphers table. The key length must be in the allowed interval. From
2251          * all cipher modes that the PKCS#11 engine supports only RC4 allows a
2252          * key length to be in some range, all other NIDs have a precise key
2253          * length. Every application can define its own EVP functions so this
2254          * code serves as a sanity check.
2255          *
2256          * Note that the reason why the IV length in ctx->cipher might be
2257          * greater than the actual length is that OpenSSL uses BLOCK_CIPHER_defs
2258          * macro to define functions that return EVP structures for all DES
2259          * modes. So, even ECB modes get 8 byte IV.
2260          */
2261         if (ctx->cipher->iv_len < p_ciph_table_row->iv_len ||
2262             ctx->key_len < p_ciph_table_row->min_key_len ||
2263             ctx->key_len > p_ciph_table_row->max_key_len) {
2264                 PK11err(PK11_F_CIPHER_INIT, PK11_R_KEY_OR_IV_LEN_PROBLEM);
2265                 return (0);
2266         }

2268         if ((sp = pk11_get_session(OP_CIPHER)) == NULL)
2269                 return (0);

2271         /* if applicable, the mechanism parameter is used for IV */
2272         mech.mechanism = p_ciph_table_row->mech_type;
2273         mech.pParameter = NULL;
2274         mech.ulParameterLen = 0;

2276         /* The key object is destroyed here if it is not the current key. */
2277         (void) check_new_cipher_key(sp, key, ctx->key_len);

2279         /*
2280          * If the key is the same and the encryption is also the same, then
2281          * just reuse it. However, we must not forget to reinitialize the
2282          * context that was finalized in pk11_cipher_cleanup().
2283          */
2284         if (sp->opdata_cipher_key != CK_INVALID_HANDLE &&
2285             sp->opdata_encrypt == ctx->encrypt)
2286                 {
2287                 state->sp = sp;
2288                 if (pk11_init_symmetric(ctx, p_ciph_table_row, sp, &mech) == 0)
2289                         return (0);

2291                 return (1);
2292                 }

2294         /*
2295          * Check if the key has been invalidated. If so, a new key object
2296          * needs to be created.
2297          */
2298         if (sp->opdata_cipher_key == CK_INVALID_HANDLE)
2299                 {
2300                 sp->opdata_cipher_key = pk11_get_cipher_key(
2301                         ctx, key, p_ciph_table_row->key_type, sp);
2302                 }

2304         if (sp->opdata_encrypt != ctx->encrypt && sp->opdata_encrypt != -1)
2305                 {
```



```
2306                 /*
2307                  * The previous encryption/decryption is different. Need to
2308                  * terminate the previous * active encryption/decryption here.
2309                  */
2310                 if (!pk11_cipher_final(sp))
2311                         {
2312                         pk11_return_session(sp, OP_CIPHER);
2313                         return (0);
2314                         }
2315                 }

2317         if (sp->opdata_cipher_key == CK_INVALID_HANDLE)
2318                 {
2319                 pk11_return_session(sp, OP_CIPHER);
2320                 return (0);
2321                 }

2323         /* now initialize the context with a new key */
2324         if (pk11_init_symmetric(ctx, p_ciph_table_row, sp, &mech) == 0)
2325                 return (0);

2327         sp->opdata_encrypt = ctx->encrypt;
2328         state->sp = sp;

2330         return (1);
2331         }

2333 /*
2334  * When reusing the same key in an encryption/decryption session for a
2335  * decryption/encryption session, we need to close the active session
2336  * and recreate a new one. Note that the key is in the global session so
2337  * that it needs not be recreated.
2338  *
2339  * It is more appropriate to use C_En/DecryptFinish here. At the time of this
2340  * development, these two functions in the PKCS#11 libraries used return
2341  * unexpected errors when passing in 0 length output. It may be a good
2342  * idea to try them again if performance is a problem here and fix
2343  * C_En/DecryptFinial if there are bugs there causing the problem.
2344  */
2345 static int
2346 pk11_cipher_final(PK11_SESSION *sp)
2347         {
2348         CK_RV rv;

2350         rv = pFuncList->C_CloseSession(sp->session);
2351         if (rv != CKR_OK)
2352                 {
2353                 PK11err_add_data(PK11_F_CIPHER_FINAL, PK11_R_CLOSESESSION, rv);
2354                 return (0);
2355                 }

2357         rv = pFuncList->C_OpenSession(SLOTID, CKF_SERIAL_SESSION,
2358                 NULL_PTR, NULL_PTR, &sp->session);
2359         if (rv != CKR_OK)
2360                 {
2361                 PK11err_add_data(PK11_F_CIPHER_FINAL, PK11_R_OPENSESSION, rv);
2362                 return (0);
2363                 }

2365         return (1);
2366         }

2368 /*
2369  * An engine interface function. The calling function allocates sufficient
2370  * memory for the output buffer "out" to hold the results.
2371  */
```

```
2372 static int
2373 pk11_cipher_do_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
2374         const unsigned char *in, size_t inl)
2375         {
2376         PK11_CIPHER_STATE *state = (PK11_CIPHER_STATE *) ctx->cipher_data;
2377         PK11_SESSION *sp;
2378         CK_RV rv;
2379         unsigned long outl = inl;

2381         if (state == NULL || state->sp == NULL)
2382                 return (0);

2384         sp = (PK11_SESSION *) state->sp;

2386         if (!inl)
2387                 return (1);

2389         /* RC4 is the only stream cipher we support */
2390         if (ctx->cipher->nid != NID_rc4 && (inl % ctx->cipher->block_size) != 0)
2391                 return (0);

2393         if (ctx->encrypt)
2394                 {
2395                 rv = pFuncList->C_EncryptUpdate(sp->session,
2396                         (unsigned char *)in, inl, out, &outl);

2398                 if (rv != CKR_OK)
2399                         {
2400                         PK11err_add_data(PK11_F_CIPHER_DO_CIPHER,
2401                             PK11_R_ENCRYPTUPDATE, rv);
2402                         return (0);
2403                         }
2404                 }
2405         else
2406                 {
2407                 rv = pFuncList->C_DecryptUpdate(sp->session,
2408                         (unsigned char *)in, inl, out, &outl);

2410                 if (rv != CKR_OK)
2411                         {
2412                         PK11err_add_data(PK11_F_CIPHER_DO_CIPHER,
2413                             PK11_R_DECRYPTUPDATE, rv);
2414                         return (0);
2415                         }
2416                 }

2418         /*
2419          * For DES_CBC, DES3_CBC, AES_CBC, and RC4, the output size is always
2420          * the same size of input.
2421          * The application has guaranteed to call the block ciphers with
2422          * correctly aligned buffers.
2423          */
2424         if (inl != outl)
2425                 return (0);

2427         return (1);
2428         }

2430 /*
2431  * Return the session to the pool. Calling C_EncryptFinal() and C_DecryptFinal()
2432  * here is the right thing because in EVP_DecryptFinal_ex(), engine's
2433  * do_cipher() is not even called, and in EVP_EncryptFinal_ex() it is called but
2434  * the engine can't find out that it's the finalizing call. We wouldn't
2435  * necessarily have to finalize the context here since reinitializing it with
2436  * C_(Encrypt|Decrypt)Init() should be fine but for the sake of correctness,
2437  * let's do it. Some implementations might leak memory if the previously used
```



```
2438  * context is initialized without finalizing it first.
2439  */
2440 static int
2441 pk11_cipher_cleanup(EVP_CIPHER_CTX *ctx)
2442         {
2443         CK_RV rv;
2444         CK_ULONG len = EVP_MAX_BLOCK_LENGTH;
2445         CK_BYTE buf[EVP_MAX_BLOCK_LENGTH];
2446         PK11_CIPHER_STATE *state = ctx->cipher_data;

2448         if (state != NULL && state->sp != NULL)
2449                 {
2450                 /*
2451                  * We are not interested in the data here, we just need to get
2452                  * rid of the context.
2453                  */
2454                 if (ctx->encrypt)
2455                         rv = pFuncList->C_EncryptFinal(
2456                             state->sp->session, buf, &len);
2457                 else
2458                         rv = pFuncList->C_DecryptFinal(
2459                             state->sp->session, buf, &len);

2461                 if (rv != CKR_OK)
2462                         {
2463                         PK11err_add_data(PK11_F_CIPHER_CLEANUP, ctx->encrypt ?
2464                             PK11_R_ENCRYPTFINAL : PK11_R_DECRYPTFINAL, rv);
2465                         pk11_return_session(state->sp, OP_CIPHER);
2466                         return (0);
2467                         }

2469                 pk11_return_session(state->sp, OP_CIPHER);
2470                 state->sp = NULL;
2471                 }

2473         return (1);
2474         }

2476 /*
2477  * Registered by the ENGINE when used to find out how to deal with
2478  * a particular NID in the ENGINE. This says what we'll do at the
2479  * top level - note, that list is restricted by what we answer with
2480  */
2481 /* ARGSUSED */
2482 static int
2483 pk11_engine_ciphers(ENGINE *e, const EVP_CIPHER **cipher,
2484         const int **nids, int nid)
2485         {
2486         if (!cipher)
2487                 return (pk11_usable_ciphers(nids));

2489         switch (nid)
2490                 {
2491                 case NID_des_ede3_cbc:
2492                         *cipher = &pk11_3des_cbc;
2493                         break;
2494                 case NID_des_cbc:
2495                         *cipher = &pk11_des_cbc;
2496                         break;
2497                 case NID_des_ede3_ecb:
2498                         *cipher = &pk11_3des_ecb;
2499                         break;
2500                 case NID_des_ecb:
2501                         *cipher = &pk11_des_ecb;
2502                         break;
2503                 case NID_aes_128_cbc:
```

```
2504                        *cipher = &pk11_aes_128_cbc;
2505                        break;
2506                case NID_aes_192_cbc:
2507                        *cipher = &pk11_aes_192_cbc;
2508                        break;
2509                case NID_aes_256_cbc:
2510                        *cipher = &pk11_aes_256_cbc;
2511                        break;
2512                case NID_aes_128_ecb:
2513                        *cipher = &pk11_aes_128_ecb;
2514                        break;
2515                case NID_aes_192_ecb:
2516                        *cipher = &pk11_aes_192_ecb;
2517                        break;
2518                case NID_aes_256_ecb:
2519                        *cipher = &pk11_aes_256_ecb;
2520                        break;
2521                case NID_bf_cbc:
2522                        *cipher = &pk11_bf_cbc;
2523                        break;
2524                case NID_rc4:
2525                        *cipher = &pk11_rc4;
2526                        break;
2527                default:
2528 #ifdef  SOLARIS_AES_CTR
2529                        /*
2530                         * These can't be in separated cases because the NIDs
2531                         * here are not constants.
2532                         */
2533                        if (nid == NID_aes_128_ctr)
2534                                *cipher = &pk11_aes_128_ctr;
2535                        else if (nid == NID_aes_192_ctr)
2536                                *cipher = &pk11_aes_192_ctr;
2537                        else if (nid == NID_aes_256_ctr)
2538                                *cipher = &pk11_aes_256_ctr;
2539                        else
2540 #endif  /* SOLARIS_AES_CTR */
2541                        *cipher = NULL;
2542                        break;
2543                }
2544        return (*cipher != NULL);
2545        }

2547 /* ARGSUSED */
2548 static int
2549 pk11_engine_digests(ENGINE *e, const EVP_MD **digest,
2550        const int **nids, int nid)
2551        {
2552        if (!digest)
2553                return (pk11_usable_digests(nids));

2555        switch (nid)
2556                {
2557                case NID_md5:
2558                        *digest = &pk11_md5;
2559                        break;
2560                case NID_sha1:
2561                        *digest = &pk11_sha1;
2562                        break;
2563                case NID_sha224:
2564                        *digest = &pk11_sha224;
2565                        break;
2566                case NID_sha256:
2567                        *digest = &pk11_sha256;
2568                        break;
2569                case NID_sha384:
```



```
2570                        *digest = &pk11_sha384;
2571                        break;
2572                case NID_sha512:
2573                        *digest = &pk11_sha512;
2574                        break;
2575                default:
2576                        *digest = NULL;
2577                        break;
2578                }
2579        return (*digest != NULL);
2580        }


2583 /* Create a secret key object in a PKCS#11 session */
2584 static CK_OBJECT_HANDLE pk11_get_cipher_key(EVP_CIPHER_CTX *ctx,
2585        const unsigned char *key, CK_KEY_TYPE key_type, PK11_SESSION *sp)
2586        {
2587        CK_RV rv;
2588        CK_OBJECT_HANDLE h_key = CK_INVALID_HANDLE;
2589        CK_OBJECT_CLASS obj_key = CKO_SECRET_KEY;
2590        CK_ULONG ul_key_attr_count = 6;

2592        CK_ATTRIBUTE  a_key_template[] =
2593                {
2594                {CKA_CLASS, (void*) NULL, sizeof (CK_OBJECT_CLASS)},
2595                {CKA_KEY_TYPE, (void*) NULL, sizeof (CK_KEY_TYPE)},
2596                {CKA_TOKEN, &false, sizeof (false)},
2597                {CKA_ENCRYPT, &true, sizeof (true)},
2598                {CKA_DECRYPT, &true, sizeof (true)},
2599                {CKA_VALUE, (void*) NULL, 0},
2600                };

2602        /*
2603         * Create secret key object in global_session. All other sessions
2604         * can use the key handles. Here is why:
2605         * OpenSSL will call EncryptInit and EncryptUpdate using a secret key.
2606         * It may then call DecryptInit and DecryptUpdate using the same key.
2607         * To use the same key object, we need to call EncryptFinal with
2608         * a 0 length message. Currently, this does not work for 3DES
2609         * mechanism. To get around this problem, we close the session and
2610         * then create a new session to use the same key object. When a session
2611         * is closed, all the object handles will be invalid. Thus, create key
2612         * objects in a global session, an individual session may be closed to
2613         * terminate the active operation.
2614         */
2615        CK_SESSION_HANDLE session = global_session;
2616        a_key_template[0].pValue = &obj_key;
2617        a_key_template[1].pValue = &key_type;
2618        a_key_template[5].pValue = (void *) key;
2619        a_key_template[5].ulValueLen = (unsigned long) ctx->key_len;

2621        rv = pFuncList->C_CreateObject(session,
2622                a_key_template, ul_key_attr_count, &h_key);
2623        if (rv != CKR_OK)
2624                {
2625                PK11err_add_data(PK11_F_GET_CIPHER_KEY, PK11_R_CREATEOBJECT,
2626                    rv);
2627                goto err;
2628                }

2630        /*
2631         * Save the key information used in this session.
2632         * The max can be saved is PK11_KEY_LEN_MAX.
2633         */
2634        sp->opdata_key_len = ctx->key_len > PK11_KEY_LEN_MAX ?
2635                PK11_KEY_LEN_MAX : ctx->key_len;
```

```
2636         (void) memcpy(sp->opdata_key, key, sp->opdata_key_len);
2637 err:

2639         return (h_key);
2640         }

2642 static int
2643 md_nid_to_pk11(int nid)
2644         {
2645         int i;

2647         for (i = 0; i < PK11_DIGEST_MAX; i++)
2648                 if (digests[i].nid == nid)
2649                         return (digests[i].id);
2650         return (-1);
2651         }

2653 static int
2654 pk11_digest_init(EVP_MD_CTX *ctx)
2655         {
2656         CK_RV rv;
2657         CK_MECHANISM mech;
2658         int index;
2659         PK11_SESSION *sp;
2660         PK11_DIGEST *pdp;
2661         PK11_CIPHER_STATE *state = (PK11_CIPHER_STATE *) ctx->md_data;

2663         state->sp = NULL;

2665         index = md_nid_to_pk11(ctx->digest->type);
2666         if (index < 0 || index >= PK11_DIGEST_MAX)
2667                 return (0);

2669         pdp = &digests[index];
2670         if ((sp = pk11_get_session(OP_DIGEST)) == NULL)
2671                 return (0);

2673         /* at present, no parameter is needed for supported digests */
2674         mech.mechanism = pdp->mech_type;
2675         mech.pParameter = NULL;
2676         mech.ulParameterLen = 0;

2678         rv = pFuncList->C_DigestInit(sp->session, &mech);

2680         if (rv != CKR_OK)
2681                 {
2682                 PK11err_add_data(PK11_F_DIGEST_INIT, PK11_R_DIGESTINIT, rv);
2683                 pk11_return_session(sp, OP_DIGEST);
2684                 return (0);
2685                 }

2687         state->sp = sp;

2689         return (1);
2690         }

2692 static int
2693 pk11_digest_update(EVP_MD_CTX *ctx, const void *data, size_t count)
2694         {
2695         CK_RV rv;
2696         PK11_CIPHER_STATE *state = (PK11_CIPHER_STATE *) ctx->md_data;

2698         /* 0 length message will cause a failure in C_DigestFinal */
2699         if (count == 0)
2700                 return (1);
```



```
2702         if (state == NULL || state->sp == NULL)
2703                 return (0);

2705         rv = pFuncList->C_DigestUpdate(state->sp->session, (CK_BYTE *) data,
2706                 count);

2708         if (rv != CKR_OK)
2709                 {
2710                 PK11err_add_data(PK11_F_DIGEST_UPDATE, PK11_R_DIGESTUPDATE, rv);
2711                 pk11_return_session(state->sp, OP_DIGEST);
2712                 state->sp = NULL;
2713                 return (0);
2714                 }

2716         return (1);
2717         }

2719 static int
2720 pk11_digest_final(EVP_MD_CTX *ctx, unsigned char *md)
2721         {
2722         CK_RV rv;
2723         unsigned long len;
2724         PK11_CIPHER_STATE *state = (PK11_CIPHER_STATE *) ctx->md_data;
2725         len = ctx->digest->md_size;

2727         if (state == NULL || state->sp == NULL)
2728                 return (0);

2730         rv = pFuncList->C_DigestFinal(state->sp->session, md, &len);

2732         if (rv != CKR_OK)
2733                 {
2734                 PK11err_add_data(PK11_F_DIGEST_FINAL, PK11_R_DIGESTFINAL, rv);
2735                 pk11_return_session(state->sp, OP_DIGEST);
2736                 state->sp = NULL;
2737                 return (0);
2738                 }

2740         if (ctx->digest->md_size != len)
2741                 return (0);

2743         /*
2744          * Final is called and digest is returned, so return the session
2745          * to the pool
2746          */
2747         pk11_return_session(state->sp, OP_DIGEST);
2748         state->sp = NULL;

2750         return (1);
2751         }

2753 static int
2754 pk11_digest_copy(EVP_MD_CTX *to, const EVP_MD_CTX *from)
2755         {
2756         CK_RV rv;
2757         int ret = 0;
2758         PK11_CIPHER_STATE *state, *state_to;
2759         CK_BYTE_PTR pstate = NULL;
2760         CK_ULONG ul_state_len;

2762         /* The copy-from state */
2763         state = (PK11_CIPHER_STATE *) from->md_data;
2764         if (state == NULL || state->sp == NULL)
2765                 goto err;

2767         /* Initialize the copy-to state */
```

```
2768         if (!pk11_digest_init(to))
2769                 goto err;
2770         state_to = (PK11_CIPHER_STATE *) to->md_data;

2772         /* Get the size of the operation state of the copy-from session */
2773         rv = pFuncList->C_GetOperationState(state->sp->session, NULL,
2774                 &ul_state_len);

2776         if (rv != CKR_OK)
2777                 {
2778                 PK11err_add_data(PK11_F_DIGEST_COPY, PK11_R_GET_OPERATION_STATE,
2779                     rv);
2780                 goto err;
2781                 }
2782         if (ul_state_len == 0)
2783                 {
2784                 goto err;
2785                 }

2787         pstate = OPENSSL_malloc(ul_state_len);
2788         if (pstate == NULL)
2789                 {
2790                 PK11err(PK11_F_DIGEST_COPY, PK11_R_MALLOC_FAILURE);
2791                 goto err;
2792                 }

2794         /* Get the operation state of the copy-from session */
2795         rv = pFuncList->C_GetOperationState(state->sp->session, pstate,
2796                 &ul_state_len);

2798         if (rv != CKR_OK)
2799                 {
2800                 PK11err_add_data(PK11_F_DIGEST_COPY, PK11_R_GET_OPERATION_STATE,
2801                     rv);
2802                 goto err;
2803                 }

2805         /* Set the operation state of the copy-to session */
2806         rv = pFuncList->C_SetOperationState(state_to->sp->session, pstate,
2807                 ul_state_len, 0, 0);

2809         if (rv != CKR_OK)
2810                 {
2811                 PK11err_add_data(PK11_F_DIGEST_COPY,
2812                     PK11_R_SET_OPERATION_STATE, rv);
2813                 goto err;
2814                 }

2816         ret = 1;
2817 err:
2818         if (pstate != NULL)
2819                 OPENSSL_free(pstate);

2821         return (ret);
2822         }

2824 /* Return any pending session state to the pool */
2825 static int
2826 pk11_digest_cleanup(EVP_MD_CTX *ctx)
2827         {
2828         PK11_CIPHER_STATE *state = ctx->md_data;
2829         unsigned char buf[EVP_MAX_MD_SIZE];

2831         if (state != NULL && state->sp != NULL)
2832                 {
2833                 /*
```



```
2834                  * If state->sp is not NULL then pk11_digest_final() has not
2835                  * been called yet. We must call it now to free any memory
2836                  * that might have been allocated in the token when
2837                  * pk11_digest_init() was called. pk11_digest_final()
2838                  * will return the session to the cache.
2839                  */
2840                 if (!pk11_digest_final(ctx, buf))
2841                         return (0);
2842                 }

2844         return (1);
2845         }

2847 /*
2848  * Check if the new key is the same as the key object in the session. If the key
2849  * is the same, no need to create a new key object. Otherwise, the old key
2850  * object needs to be destroyed and a new one will be created. Return 1 for
2851  * cache hit, 0 for cache miss. Note that we must check the key length first
2852  * otherwise we could end up reusing a different, longer key with the same
2853  * prefix.
2854  */
2855 static int check_new_cipher_key(PK11_SESSION *sp, const unsigned char *key,
2856         int key_len)
2857         {
2858         if (sp->opdata_key_len != key_len ||
2859             memcmp(sp->opdata_key, key, key_len) != 0)
2860                 {
2861                 (void) pk11_destroy_cipher_key_objects(sp);
2862                 return (0);
2863                 }
2864         return (1);
2865         }

2867 /* Destroy one or more secret key objects. */
2868 static int pk11_destroy_cipher_key_objects(PK11_SESSION *session)
2869         {
2870         int ret = 0;
2871         PK11_SESSION *sp = NULL;
2872         PK11_SESSION *local_free_session;

2874         if (session != NULL)
2875                 local_free_session = session;
2876         else
2877                 {
2878                 (void) pthread_mutex_lock(session_cache[OP_CIPHER].lock);
2879                 local_free_session = session_cache[OP_CIPHER].head;
2880                 }

2882         while ((sp = local_free_session) != NULL)
2883                 {
2884                 local_free_session = sp->next;

2886                 if (sp->opdata_cipher_key != CK_INVALID_HANDLE)
2887                         {
2888                         /*
2889                          * The secret key object is created in the
2890                          * global_session. See pk11_get_cipher_key
2891                          */
2892                         if (pk11_destroy_object(global_session,
2893                                 sp->opdata_cipher_key) == 0)
2894                                 goto err;
2895                         sp->opdata_cipher_key = CK_INVALID_HANDLE;
2896                         }
2897                 }
2898         ret = 1;
2899 err:
```

```
2901         if (session == NULL)
2902                 (void) pthread_mutex_unlock(session_cache[OP_CIPHER].lock);

2904         return (ret);
2905         }


2908 /*
2909  * Public key mechanisms optionally supported
2910  *
2911  * CKM_RSA_X_509
2912  * CKM_RSA_PKCS
2913  * CKM_DSA
2914  *
2915  * The first slot that supports at least one of those mechanisms is chosen as a
2916  * public key slot.
2917  *
2918  * Symmetric ciphers optionally supported
2919  *
2920  * CKM_DES3_CBC
2921  * CKM_DES_CBC
2922  * CKM_AES_CBC
2923  * CKM_DES3_ECB
2924  * CKM_DES_ECB
2925  * CKM_AES_ECB
2926  * CKM_AES_CTR
2927  * CKM_RC4
2928  * CKM_BLOWFISH_CBC
2929  *
2930  * Digests optionally supported
2931  *
2932  * CKM_MD5
2933  * CKM_SHA_1
2934  * CKM_SHA224
2935  * CKM_SHA256
2936  * CKM_SHA384
2937  * CKM_SHA512
2938  *
2939  * The output of this function is a set of global variables indicating which
2940  * mechanisms from RSA, DSA, DH and RAND are present, and also two arrays of
2941  * mechanisms, one for symmetric ciphers and one for digests. Also, 3 global
2942  * variables carry information about which slot was chosen for (a) public key
2943  * mechanisms, (b) random operations, and (c) symmetric ciphers and digests.
2944  */
2945 static int
2946 pk11_choose_slots(int *any_slot_found)
2947         {
2948         CK_SLOT_ID_PTR pSlotList = NULL_PTR;
2949         CK_ULONG ulSlotCount = 0;
2950         CK_MECHANISM_INFO mech_info;
2951         CK_TOKEN_INFO token_info;
2952         int i;
2953         CK_RV rv;
2954         CK_SLOT_ID best_slot_sofar = 0;
2955         CK_BBOOL found_candidate_slot = CK_FALSE;
2956         int slot_n_cipher = 0;
2957         int slot_n_digest = 0;
2958         CK_SLOT_ID current_slot = 0;
2959         int current_slot_n_cipher = 0;
2960         int current_slot_n_digest = 0;

2962         int local_cipher_nids[PK11_CIPHER_MAX];
2963         int local_digest_nids[PK11_DIGEST_MAX];

2965         /* let's initialize the output parameter */
```



```
2966         if (any_slot_found != NULL)
2967                 *any_slot_found = 0;

2969         /* Get slot list for memory allocation */
2970         rv = pFuncList->C_GetSlotList(0, NULL_PTR, &ulSlotCount);

2972         if (rv != CKR_OK)
2973                 {
2974                 PK11err_add_data(PK11_F_CHOOSE_SLOT, PK11_R_GETSLOTLIST, rv);
2975                 return (0);
2976                 }

2978         /* it's not an error if we didn't find any providers */
2979         if (ulSlotCount == 0)
2980                 {
2981 #ifdef  DEBUG_SLOT_SELECTION
2982                 fprintf(stderr, "%s: no crypto providers found\n", PK11_DBG);
2983 #endif  /* DEBUG_SLOT_SELECTION */
2984                 return (1);
2985                 }

2987         pSlotList = OPENSSL_malloc(ulSlotCount * sizeof (CK_SLOT_ID));

2989         if (pSlotList == NULL)
2990                 {
2991                 PK11err(PK11_F_CHOOSE_SLOT, PK11_R_MALLOC_FAILURE);
2992                 return (0);
2993                 }

2995         /* Get the slot list for processing */
2996         rv = pFuncList->C_GetSlotList(0, pSlotList, &ulSlotCount);
2997         if (rv != CKR_OK)
2998                 {
2999                 PK11err_add_data(PK11_F_CHOOSE_SLOT, PK11_R_GETSLOTLIST, rv);
3000                 OPENSSL_free(pSlotList);
3001                 return (0);
3002                 }

3004 #ifdef  DEBUG_SLOT_SELECTION
3005         fprintf(stderr, "%s: provider: %s\n", PK11_DBG, def_PK11_LIBNAME);
3006         fprintf(stderr, "%s: number of slots: %d\n", PK11_DBG, ulSlotCount);

3008         fprintf(stderr, "%s: == checking rand slots ==\n", PK11_DBG);
3009 #endif  /* DEBUG_SLOT_SELECTION */
3010         for (i = 0; i < ulSlotCount; i++)
3011                 {
3012                 current_slot = pSlotList[i];

3014 #ifdef  DEBUG_SLOT_SELECTION
3015         fprintf(stderr, "%s: checking slot: %d\n", PK11_DBG, i);
3016 #endif  /* DEBUG_SLOT_SELECTION */
3017                 /* Check if slot has random support. */
3018                 rv = pFuncList->C_GetTokenInfo(current_slot, &token_info);
3019                 if (rv != CKR_OK)
3020                         continue;

3022 #ifdef  DEBUG_SLOT_SELECTION
3023         fprintf(stderr, "%s: token label: %.32s\n", PK11_DBG, token_info.label);
3024 #endif  /* DEBUG_SLOT_SELECTION */

3026                 if (token_info.flags & CKF_RNG)
3027                         {
3028 #ifdef  DEBUG_SLOT_SELECTION
3029         fprintf(stderr, "%s: this token has CKF_RNG flag\n", PK11_DBG);
3030 #endif  /* DEBUG_SLOT_SELECTION */
3031                         pk11_have_random = CK_TRUE;
```

```
3032                        rand_SLOTID = current_slot;
3033                        break;
3034                        }
3035                }

3037 #ifdef  DEBUG_SLOT_SELECTION
3038         fprintf(stderr, "%s: == checking pubkey slots ==\n", PK11_DBG);
3039 #endif  /* DEBUG_SLOT_SELECTION */

3041         pubkey_SLOTID = pSlotList[0];
3042         for (i = 0; i < ulSlotCount; i++)
3043                 {
3044                 CK_BBOOL slot_has_rsa = CK_FALSE;
3045                 CK_BBOOL slot_has_dsa = CK_FALSE;
3046                 CK_BBOOL slot_has_dh = CK_FALSE;
3047                 current_slot = pSlotList[i];

3049 #ifdef  DEBUG_SLOT_SELECTION
3050         fprintf(stderr, "%s: checking slot: %d\n", PK11_DBG, i);
3051 #endif  /* DEBUG_SLOT_SELECTION */
3052                 rv = pFuncList->C_GetTokenInfo(current_slot, &token_info);
3053                 if (rv != CKR_OK)
3054                         continue;

3056 #ifdef  DEBUG_SLOT_SELECTION
3057         fprintf(stderr, "%s: token label: %.32s\n", PK11_DBG, token_info.label);
3058 #endif  /* DEBUG_SLOT_SELECTION */

3060 #ifndef OPENSSL_NO_RSA
3061                 /*
3062                  * Check if this slot is capable of signing and
3063                  * verifying with CKM_RSA_PKCS.
3064                  */
3065                 rv = pFuncList->C_GetMechanismInfo(current_slot, CKM_RSA_PKCS,
3066                         &mech_info);

3068                 if (rv == CKR_OK && ((mech_info.flags & CKF_SIGN) &&
3069                                 (mech_info.flags & CKF_VERIFY)))
3070                         {
3071                         /*
3072                          * Check if this slot is capable of encryption,
3073                          * decryption, sign, and verify with CKM_RSA_X_509.
3074                          */
3075                         rv = pFuncList->C_GetMechanismInfo(current_slot,
3076                             CKM_RSA_X_509, &mech_info);

3078                         if (rv == CKR_OK && ((mech_info.flags & CKF_SIGN) &&
3079                             (mech_info.flags & CKF_VERIFY) &&
3080                             (mech_info.flags & CKF_ENCRYPT) &&
3081                             (mech_info.flags & CKF_VERIFY_RECOVER) &&
3082                             (mech_info.flags & CKF_DECRYPT)))
3083                                 {
3084                                 slot_has_rsa = CK_TRUE;
3085                                 }
3086                         }
3087 #endif  /* OPENSSL_NO_RSA */

3089 #ifndef OPENSSL_NO_DSA
3090                 /*
3091                  * Check if this slot is capable of signing and
3092                  * verifying with CKM_DSA.
3093                  */
3094                 rv = pFuncList->C_GetMechanismInfo(current_slot, CKM_DSA,
3095                         &mech_info);
3096                 if (rv == CKR_OK && ((mech_info.flags & CKF_SIGN) &&
3097                     (mech_info.flags & CKF_VERIFY)))
```



```
3098                         {
3099                         slot_has_dsa = CK_TRUE;
3100                         }

3102 #endif  /* OPENSSL_NO_DSA */

3104 #ifndef OPENSSL_NO_DH
3105                 /*
3106                  * Check if this slot is capable of DH key generataion and
3107                  * derivation.
3108                  */
3109                 rv = pFuncList->C_GetMechanismInfo(current_slot,
3110                     CKM_DH_PKCS_KEY_PAIR_GEN, &mech_info);

3112                 if (rv == CKR_OK && (mech_info.flags & CKF_GENERATE_KEY_PAIR))
3113                         {
3114                         rv = pFuncList->C_GetMechanismInfo(current_slot,
3115                                 CKM_DH_PKCS_DERIVE, &mech_info);
3116                         if (rv == CKR_OK && (mech_info.flags & CKF_DERIVE))
3117                                 {
3118                                 slot_has_dh = CK_TRUE;
3119                                 }
3120                         }
3121 #endif  /* OPENSSL_NO_DH */

3123                 if (!found_candidate_slot &&
3124                     (slot_has_rsa || slot_has_dsa || slot_has_dh))
3125                         {
3126 #ifdef  DEBUG_SLOT_SELECTION
3127                         fprintf(stderr,
3128                             "%s: potential slot: %d\n", PK11_DBG, current_slot);
3129 #endif  /* DEBUG_SLOT_SELECTION */
3130                         best_slot_sofar = current_slot;
3131                         pk11_have_rsa = slot_has_rsa;
3132                         pk11_have_dsa = slot_has_dsa;
3133                         pk11_have_dh = slot_has_dh;
3134                         found_candidate_slot = CK_TRUE;
3135 #ifdef  DEBUG_SLOT_SELECTION
3136                         fprintf(stderr,
3137                             "%s: setting found_candidate_slot to CK_TRUE\n",
3138                             PK11_DBG);
3139                         fprintf(stderr,
3140                             "%s: best so far slot: %d\n", PK11_DBG,
3141                             best_slot_sofar);
3142                         }
3143                 else
3144                         {
3145                         fprintf(stderr,
3146                             "%s: no rsa/dsa/dh\n", PK11_DBG);
3147                         }
3148 #else
3149                         } /* if */
3150 #endif  /* DEBUG_SLOT_SELECTION */
3151                 } /* for */

3153         if (found_candidate_slot)
3154                 {
3155                 pubkey_SLOTID = best_slot_sofar;
3156                 }

3158         found_candidate_slot = CK_FALSE;
3159         best_slot_sofar = 0;

3161 #ifdef  DEBUG_SLOT_SELECTION
3162         fprintf(stderr, "%s: == checking cipher/digest ==\n", PK11_DBG);
3163 #endif  /* DEBUG_SLOT_SELECTION */
```

```
3165         SLOTID = pSlotList[0];
3166         for (i = 0; i < ulSlotCount; i++)
3167                 {
3168 #ifdef  DEBUG_SLOT_SELECTION
3169         fprintf(stderr, "%s: checking slot: %d\n", PK11_DBG, i);
3170 #endif  /* DEBUG_SLOT_SELECTION */

3172                 current_slot = pSlotList[i];
3173                 current_slot_n_cipher = 0;
3174                 current_slot_n_digest = 0;
3175                 (void) memset(local_cipher_nids, 0, sizeof (local_cipher_nids));
3176                 (void) memset(local_digest_nids, 0, sizeof (local_digest_nids));

3178                 pk11_find_symmetric_ciphers(pFuncList, current_slot,
3179                     &current_slot_n_cipher, local_cipher_nids);

3181                 pk11_find_digests(pFuncList, current_slot,
3182                     &current_slot_n_digest, local_digest_nids);

3184 #ifdef  DEBUG_SLOT_SELECTION
3185                 fprintf(stderr, "%s: current_slot_n_cipher %d\n", PK11_DBG,
3186                         current_slot_n_cipher);
3187                 fprintf(stderr, "%s: current_slot_n_digest %d\n", PK11_DBG,
3188                         current_slot_n_digest);
3189                 fprintf(stderr, "%s: best so far cipher/digest slot: %d\n",
3190                         PK11_DBG, best_slot_sofar);
3191 #endif  /* DEBUG_SLOT_SELECTION */

3193                 /*
3194                  * If the current slot supports more ciphers/digests than
3195                  * the previous best one we change the current best to this one,
3196                  * otherwise leave it where it is.
3197                  */
3198                 if ((current_slot_n_cipher + current_slot_n_digest) >
3199                     (slot_n_cipher + slot_n_digest))
3200                         {
3201 #ifdef  DEBUG_SLOT_SELECTION
3202                         fprintf(stderr,
3203                                 "%s: changing best so far slot to %d\n",
3204                                 PK11_DBG, current_slot);
3205 #endif  /* DEBUG_SLOT_SELECTION */
3206                         best_slot_sofar = SLOTID = current_slot;
3207                         cipher_count = slot_n_cipher = current_slot_n_cipher;
3208                         digest_count = slot_n_digest = current_slot_n_digest;
3209                         (void) memcpy(cipher_nids, local_cipher_nids,
3210                             sizeof (local_cipher_nids));
3211                         (void) memcpy(digest_nids, local_digest_nids,
3212                             sizeof (local_digest_nids));
3213                         }
3214                 }

3216 #ifdef  DEBUG_SLOT_SELECTION
3217         fprintf(stderr,
3218             "%s: chosen pubkey slot: %d\n", PK11_DBG, pubkey_SLOTID);
3219         fprintf(stderr,
3220             "%s: chosen rand slot: %d\n", PK11_DBG, rand_SLOTID);
3221         fprintf(stderr,
3222             "%s: chosen cipher/digest slot: %d\n", PK11_DBG, SLOTID);
3223         fprintf(stderr,
3224             "%s: pk11_have_rsa %d\n", PK11_DBG, pk11_have_rsa);
3225         fprintf(stderr,
3226             "%s: pk11_have_dsa %d\n", PK11_DBG, pk11_have_dsa);
3227         fprintf(stderr,
3228             "%s: pk11_have_dh %d\n", PK11_DBG, pk11_have_dh);
3229         fprintf(stderr,
```



```
3230             "%s: pk11_have_random %d\n", PK11_DBG, pk11_have_random);
3231         fprintf(stderr,
3232             "%s: cipher_count %d\n", PK11_DBG, cipher_count);
3233         fprintf(stderr,
3234             "%s: digest_count %d\n", PK11_DBG, digest_count);
3235 #endif  /* DEBUG_SLOT_SELECTION */

3237         if (pSlotList != NULL)
3238                 OPENSSL_free(pSlotList);

3240 #ifdef  SOLARIS_HW_SLOT_SELECTION
3241         OPENSSL_free(hw_cnids);
3242         OPENSSL_free(hw_dnids);
3243 #endif  /* SOLARIS_HW_SLOT_SELECTION */

3245         if (any_slot_found != NULL)
3246                 *any_slot_found = 1;
3247         return (1);
3248         }

3250 static void pk11_get_symmetric_cipher(CK_FUNCTION_LIST_PTR pflist,
3251     int slot_id, CK_MECHANISM_TYPE mech, int *current_slot_n_cipher,
3252     int *local_cipher_nids, int id)
3253         {
3254         CK_MECHANISM_INFO mech_info;
3255         CK_RV rv;

3257 #ifdef  DEBUG_SLOT_SELECTION
3258         fprintf(stderr, "%s: checking mech: %x", PK11_DBG, mech);
3259 #endif  /* DEBUG_SLOT_SELECTION */
3260         rv = pflist->C_GetMechanismInfo(slot_id, mech, &mech_info);

3262         if (rv != CKR_OK)
3263                 {
3264 #ifdef  DEBUG_SLOT_SELECTION
3265                 fprintf(stderr, " not found\n");
3266 #endif  /* DEBUG_SLOT_SELECTION */
3267                 return;
3268                 }

3270         if ((mech_info.flags & CKF_ENCRYPT) &&
3271             (mech_info.flags & CKF_DECRYPT))
3272                 {
3273 #ifdef  SOLARIS_HW_SLOT_SELECTION
3274                 if (nid_in_table(ciphers[id].nid, hw_cnids))
3275 #endif  /* SOLARIS_HW_SLOT_SELECTION */
3276                         {
3277 #ifdef  DEBUG_SLOT_SELECTION
3278                         fprintf(stderr, " usable\n");
3279 #endif  /* DEBUG_SLOT_SELECTION */
3280                         local_cipher_nids[(*current_slot_n_cipher)++] =
3281                             ciphers[id].nid;
3282                         }
3283 #ifdef  SOLARIS_HW_SLOT_SELECTION
3284 #ifdef  DEBUG_SLOT_SELECTION
3285                 else
3286                         {
3287                         fprintf(stderr, " rejected, software implementation only\n");
3288                         }
3289 #endif  /* DEBUG_SLOT_SELECTION */
3290 #endif  /* SOLARIS_HW_SLOT_SELECTION */
3291                 }
3292 #ifdef  DEBUG_SLOT_SELECTION
3293         else
3294                 {
3295                 fprintf(stderr, " unusable\n");
```

```
3296                 }
3297 #endif  /* DEBUG_SLOT_SELECTION */

3299         return;
3300         }

3302 static void pk11_get_digest(CK_FUNCTION_LIST_PTR pflist, int slot_id,
3303     CK_MECHANISM_TYPE mech, int *current_slot_n_digest, int *local_digest_nids,
3304     int id)
3305         {
3306         CK_MECHANISM_INFO mech_info;
3307         CK_RV rv;

3309 #ifdef  DEBUG_SLOT_SELECTION
3310         fprintf(stderr, "%s: checking mech: %x", PK11_DBG, mech);
3311 #endif  /* DEBUG_SLOT_SELECTION */
3312         rv = pflist->C_GetMechanismInfo(slot_id, mech, &mech_info);

3314         if (rv != CKR_OK)
3315                 {
3316 #ifdef  DEBUG_SLOT_SELECTION
3317                 fprintf(stderr, " not found\n");
3318 #endif  /* DEBUG_SLOT_SELECTION */
3319                 return;
3320                 }

3322         if (mech_info.flags & CKF_DIGEST)
3323                 {
3324 #ifdef  SOLARIS_HW_SLOT_SELECTION
3325                 if (nid_in_table(digests[id].nid, hw_dnids))
3326 #endif  /* SOLARIS_HW_SLOT_SELECTION */
3327                         {
3328 #ifdef  DEBUG_SLOT_SELECTION
3329                 fprintf(stderr, " usable\n");
3330 #endif  /* DEBUG_SLOT_SELECTION */
3331                         local_digest_nids[(*current_slot_n_digest)++] =
3332                             digests[id].nid;
3333                         }
3334 #ifdef  SOLARIS_HW_SLOT_SELECTION
3335 #ifdef  DEBUG_SLOT_SELECTION
3336                 else
3337                         {
3338                 fprintf(stderr, " rejected, software implementation only\n");
3339                         }
3340 #endif  /* DEBUG_SLOT_SELECTION */
3341 #endif  /* SOLARIS_HW_SLOT_SELECTION */
3342                 }
3343 #ifdef  DEBUG_SLOT_SELECTION
3344         else
3345                 {
3346                 fprintf(stderr, " unusable\n");
3347                 }
3348 #endif  /* DEBUG_SLOT_SELECTION */

3350         return;
3351         }

3353 #ifdef  SOLARIS_AES_CTR
3354 /*
3355  * Create new NIDs for AES counter mode. OpenSSL doesn't support them now so we
3356  * have to help ourselves here.
3357  */
3358 static int pk11_add_aes_ctr_NIDs(void)
3359         {
3360         /* are we already set? */
3361         if (NID_aes_256_ctr != NID_undef)
```



```
3362                 return (1);

3364         /*
3365          * There are no official names for AES counter modes yet so we just
3366          * follow the format of those that exist.
3367          */
3368         ciphers[PK11_AES_128_CTR].nid = pk11_aes_128_ctr.nid = NID_aes_128_ctr;
3369         ciphers[PK11_AES_192_CTR].nid = pk11_aes_192_ctr.nid = NID_aes_192_ctr;
3370         ciphers[PK11_AES_256_CTR].nid = pk11_aes_256_ctr.nid = NID_aes_256_ctr;
3371         return (1);

3373         }
3374 #endif  /* SOLARIS_AES_CTR */

3376 /* Find what symmetric ciphers this slot supports. */
3377 static void pk11_find_symmetric_ciphers(CK_FUNCTION_LIST_PTR pflist,
3378     CK_SLOT_ID current_slot, int *current_slot_n_cipher, int *local_cipher_nids)
3379         {
3380         int i;

3382         for (i = 0; i < PK11_CIPHER_MAX; ++i)
3383                 {
3384                 pk11_get_symmetric_cipher(pflist, current_slot,
3385                     ciphers[i].mech_type, current_slot_n_cipher,
3386                     local_cipher_nids, ciphers[i].id);
3387                 }
3388         }

3390 /* Find what digest algorithms this slot supports. */
3391 static void pk11_find_digests(CK_FUNCTION_LIST_PTR pflist,
3392     CK_SLOT_ID current_slot, int *current_slot_n_digest, int *local_digest_nids)
3393         {
3394         int i;

3396         for (i = 0; i < PK11_DIGEST_MAX; ++i)
3397                 {
3398                 pk11_get_digest(pflist, current_slot, digests[i].mech_type,
3399                     current_slot_n_digest, local_digest_nids, digests[i].id);
3400                 }
3401         }

3403 #ifdef  SOLARIS_HW_SLOT_SELECTION
3404 /*
3405  * It would be great if we could use pkcs11_kernel directly since this library
3406  * offers hardware slots only. That's the easiest way to achieve the situation
3407  * where we use the hardware accelerators when present and OpenSSL native code
3408  * otherwise. That presumes the fact that OpenSSL native code is faster than the
3409  * code in the soft token. It's a logical assumption - Crypto Framework has some
3410  * inherent overhead so going there for the software implementation of a
3411  * mechanism should be logically slower in contrast to the OpenSSL native code,
3412  * presuming that both implementations are of similar speed. For example, the
3413  * soft token for AES is roughly three times slower than OpenSSL for 64 byte
3414  * blocks and still 20% slower for 8KB blocks. So, if we want to ship products
3415  * that use the PKCS#11 engine by default, we must somehow avoid that regression
3416  * on machines without hardware acceleration. That's why switching to the
3417  * pkcs11_kernel library seems like a very good idea.
3418  *
3419  * The problem is that OpenSSL built with SunStudio is roughly 2x slower for
3420  * asymmetric operations (RSA/DSA/DH) than the soft token built with the same
3421  * compiler. That means that if we switched to pkcs11_kernel from the libpkcs11
3422  * library, we would have had a performance regression on machines without
3423  * hardware acceleration for asymmetric operations for all applications that use
3424  * the PKCS#11 engine. There is one such application - Apache web server since
3425  * it's shipped configured to use the PKCS#11 engine by default. Having said
3426  * that, we can't switch to the pkcs11_kernel library now and have to come with
3427  * a solution that, on non-accelerated machines, uses the OpenSSL native code
```

```
3428  * for all symmetric ciphers and digests while it uses the soft token for
3429  * asymmetric operations.
3430  *
3431  * This is the idea: dlopen() pkcs11_kernel directly and find out what
3432  * mechanisms are there. We don't care about duplications (more slots can
3433  * support the same mechanism), we just want to know what mechanisms can be
3434  * possibly supported in hardware on that particular machine. As said before,
3435  * pkcs11_kernel will show you hardware providers only.
3436  *
3437  * Then, we rely on the fact that since we use libpkcs11 library we will find
3438  * the metaslot. When we go through the metaslot's mechanisms for symmetric
3439  * ciphers and digests, we check that any found mechanism is in the table
3440  * created using the pkcs11_kernel library. So, as a result we have two arrays
3441  * of mechanisms that were advertised as supported in hardware which was the
3442  * goal of that whole excercise. Thus, we can use libpkcs11 but avoid soft token
3443  * code for symmetric ciphers and digests. See pk11_choose_slots() for more
3444  * information.
3445  *
3446  * This is Solaris specific code, if SOLARIS_HW_SLOT_SELECTION is not defined
3447  * the code won't be used.
3448  */
3449 #if defined(__sparcv9) || defined(__x86_64) || defined(__amd64)
3450 static const char pkcs11_kernel[] = "/usr/lib/security/64/pkcs11_kernel.so.1";
3451 #else
3452 static const char pkcs11_kernel[] = "/usr/lib/security/pkcs11_kernel.so.1";
3453 #endif

3455 /*
3456  * Check hardware capabilities of the machines. The output are two lists,
3457  * hw_cnids and hw_dnids, that contain hardware mechanisms found in all hardware
3458  * providers together. They are not sorted and may contain duplicate mechanisms.
3459  */
3460 static int check_hw_mechanisms(void)
3461         {
3462         int i;
3463         CK_RV rv;
3464         void *handle;
3465         CK_C_GetFunctionList p;
3466         CK_TOKEN_INFO token_info;
3467         CK_ULONG ulSlotCount = 0;
3468         int n_cipher = 0, n_digest = 0;
3469         CK_FUNCTION_LIST_PTR pflist = NULL;
3470         CK_SLOT_ID_PTR pSlotList = NULL_PTR;
3471         int *tmp_hw_cnids=NULL, *tmp_hw_dnids=NULL;
3472         int hw_ctable_size, hw_dtable_size;

3474 #ifdef  DEBUG_SLOT_SELECTION
3475         fprintf(stderr, "%s: SOLARIS_HW_SLOT_SELECTION code running\n",
3476             PK11_DBG);
3477 #endif
3478         if ((handle = dlopen(pkcs11_kernel, RTLD_LAZY)) == NULL)
3479                 {
3480                 PK11err(PK11_F_CHECK_HW_MECHANISMS, PK11_R_DSO_FAILURE);
3481                 goto err;
3482                 }

3484         if ((p = (CK_C_GetFunctionList)dlsym(handle,
3485             PK11_GET_FUNCTION_LIST)) == NULL)
3486                 {
3487                 PK11err(PK11_F_CHECK_HW_MECHANISMS, PK11_R_DSO_FAILURE);
3488                 goto err;
3489                 }

3491         /* get the full function list from the loaded library */
3492         if (p(&pflist) != CKR_OK)
3493                 {
```



```
3494                 PK11err(PK11_F_CHECK_HW_MECHANISMS, PK11_R_DSO_FAILURE);
3495                 goto err;
3496                 }

3498         rv = pflist->C_Initialize(NULL_PTR);
3499         if ((rv != CKR_OK) && (rv != CKR_CRYPTOKI_ALREADY_INITIALIZED))
3500                 {
3501                 PK11err_add_data(PK11_F_CHECK_HW_MECHANISMS,
3502                     PK11_R_INITIALIZE, rv);
3503                 goto err;
3504                 }

3506         if (pflist->C_GetSlotList(0, NULL_PTR, &ulSlotCount) != CKR_OK)
3507                 {
3508                 PK11err(PK11_F_CHECK_HW_MECHANISMS, PK11_R_GETSLOTLIST);
3509                 goto err;
3510                 }

3512         /* no slots, set the hw mechanism tables as empty */
3513         if (ulSlotCount == 0)
3514                 {
3515 #ifdef  DEBUG_SLOT_SELECTION
3516         fprintf(stderr, "%s: no hardware mechanisms found\n", PK11_DBG);
3517 #endif
3518                 hw_cnids = OPENSSL_malloc(sizeof (int));
3519                 hw_dnids = OPENSSL_malloc(sizeof (int));
3520                 if (hw_cnids == NULL || hw_dnids == NULL)
3521                         {
3522                         PK11err(PK11_F_CHECK_HW_MECHANISMS,
3523                             PK11_R_MALLOC_FAILURE);
3524                         return (0);
3525                         }
3526                 /* this means empty tables */
3527                 hw_cnids[0] = NID_undef;
3528                 hw_dnids[0] = NID_undef;
3529                 return (1);
3530                 }

3532         pSlotList = OPENSSL_malloc(ulSlotCount * sizeof (CK_SLOT_ID));
3533         if (pSlotList == NULL)
3534                 {
3535                 PK11err(PK11_F_CHECK_HW_MECHANISMS, PK11_R_MALLOC_FAILURE);
3536                 goto err;
3537                 }

3539         /* Get the slot list for processing */
3540         if (pflist->C_GetSlotList(0, pSlotList, &ulSlotCount) != CKR_OK)
3541                 {
3542                 PK11err(PK11_F_CHECK_HW_MECHANISMS, PK11_R_GETSLOTLIST);
3543                 goto err;
3544                 }

3546         /*
3547          * We don't care about duplicit mechanisms in multiple slots and also
3548          * reserve one slot for the terminal NID_undef which we use to stop the
3549          * search.
3550          */
3551         hw_ctable_size = ulSlotCount * PK11_CIPHER_MAX + 1;
3552         hw_dtable_size = ulSlotCount * PK11_DIGEST_MAX + 1;
3553         tmp_hw_cnids = OPENSSL_malloc(hw_ctable_size * sizeof (int));
3554         tmp_hw_dnids = OPENSSL_malloc(hw_dtable_size * sizeof (int));
3555         if (tmp_hw_cnids == NULL || tmp_hw_dnids == NULL)
3556                 {
3557                 PK11err(PK11_F_CHECK_HW_MECHANISMS, PK11_R_MALLOC_FAILURE);
3558                 goto err;
3559                 }
```

```
3561         /*
3562          * Do not use memset since we should not rely on the fact that NID_undef
3563          * is zero now.
3564          */
3565         for (i = 0; i < hw_ctable_size; ++i)
3566                 tmp_hw_cnids[i] = NID_undef;
3567         for (i = 0; i < hw_dtable_size; ++i)
3568                 tmp_hw_dnids[i] = NID_undef;

3570 #ifdef  DEBUG_SLOT_SELECTION
3571         fprintf(stderr, "%s: provider: %s\n", PK11_DBG, pkcs11_kernel);
3572         fprintf(stderr, "%s: found %d hardware slots\n", PK11_DBG, ulSlotCount);
3573         fprintf(stderr, "%s: now looking for mechs supported in hw\n",
3574             PK11_DBG);
3575 #endif  /* DEBUG_SLOT_SELECTION */

3577         for (i = 0; i < ulSlotCount; i++)
3578                 {
3579                 if (pflist->C_GetTokenInfo(pSlotList[i], &token_info) != CKR_OK)
3580                         continue;

3582 #ifdef  DEBUG_SLOT_SELECTION
3583         fprintf(stderr, "%s: token label: %.32s\n", PK11_DBG, token_info.label);
3584 #endif  /* DEBUG_SLOT_SELECTION */

3586                 /*
3587                  * We are filling the hw mech tables here. Global tables are
3588                  * still NULL so all mechanisms are put into tmp tables.
3589                  */
3590                 pk11_find_symmetric_ciphers(pflist, pSlotList[i],
3591                     &n_cipher, tmp_hw_cnids);
3592                 pk11_find_digests(pflist, pSlotList[i],
3593                     &n_digest, tmp_hw_dnids);
3594                 }

3596         /*
3597          * Since we are part of a library (libcrypto.so), calling this function
3598          * may have side-effects. Also, C_Finalize() is triggered by
3599          * dlclose(3C).
3600          */
3601 #if 0
3602         pflist->C_Finalize(NULL);
3603 #endif
3604         OPENSSL_free(pSlotList);
3605         (void) dlclose(handle);
3606         hw_cnids = tmp_hw_cnids;
3607         hw_dnids = tmp_hw_dnids;

3609 #ifdef  DEBUG_SLOT_SELECTION
3610         fprintf(stderr, "%s: hw mechs check complete\n", PK11_DBG);
3611 #endif  /* DEBUG_SLOT_SELECTION */
3612         return (1);

3614 err:
3615         if (pSlotList != NULL)
3616                 OPENSSL_free(pSlotList);
3617         if (tmp_hw_cnids != NULL)
3618                 OPENSSL_free(tmp_hw_cnids);
3619         if (tmp_hw_dnids != NULL)
3620                 OPENSSL_free(tmp_hw_dnids);

3622         return (0);
3623         }

3625 /*
```



```
3626  * Check presence of a NID in the table of NIDs. The table may be NULL (i.e.,
3627  * non-existent).
3628  */
3629 static int nid_in_table(int nid, int *nid_table)
3630         {
3631         int i = 0;

3633         /*
3634          * a special case. NULL means that we are initializing a new
3635          * table.
3636          */
3637         if (nid_table == NULL)
3638                 return (1);

3640         /*
3641          * the table is never full, there is always at least one
3642          * NID_undef.
3643          */
3644         while (nid_table[i] != NID_undef)
3645                 {
3646                 if (nid_table[i++] == nid)
3647                         {
3648 #ifdef  DEBUG_SLOT_SELECTION
3649         fprintf(stderr, " (NID %d in hw table, idx %d)", nid, i);
3650 #endif  /* DEBUG_SLOT_SELECTION */
3651                         return (1);
3652                         }
3653                 }

3655         return (0);
3656         }
3657 #endif  /* SOLARIS_HW_SLOT_SELECTION */

3659 #endif  /* OPENSSL_NO_HW_PK11 */
3660 #endif  /* OPENSSL_NO_HW */
3661 #endif /* ! codereview */
```

```
**********************************************************
   11334 Wed Aug 13 19:52:38 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/hw_pk11_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* crypto/engine/hw_pk11_err.c */
/*
 * This product includes software developed by the OpenSSL Project for
 * use in the OpenSSL Toolkit (http://www.openssl.org/).
 *
 * This project also referenced hw_pkcs11-0.9.7b.patch written by
 * Afchine Madjlessi.
 */
/*
 * ====================================================================
 * Copyright (c) 2000-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
```



```
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/opensslconf.h>
#include <openssl/err.h>
#include "hw_pk11_err.h"

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR
static ERR_STRING_DATA pk11_str_functs[]=
{
{ ERR_PACK(0, PK11_F_INIT, 0),                  "PK11_INIT"},
{ ERR_PACK(0, PK11_F_FINISH, 0),                "PK11_FINISH"},
{ ERR_PACK(0, PK11_F_DESTROY, 0),               "PK11_DESTROY"},
{ ERR_PACK(0, PK11_F_CTRL, 0),                  "PK11_CTRL"},
{ ERR_PACK(0, PK11_F_RSA_INIT, 0),              "PK11_RSA_INIT"},
{ ERR_PACK(0, PK11_F_RSA_FINISH, 0),            "PK11_RSA_FINISH"},
{ ERR_PACK(0, PK11_F_GET_PUB_RSA_KEY, 0),       "PK11_GET_PUB_RSA_KEY"},
{ ERR_PACK(0, PK11_F_GET_PRIV_RSA_KEY, 0),      "PK11_GET_PRIV_RSA_KEY"},
{ ERR_PACK(0, PK11_F_RSA_GEN_KEY, 0),           "PK11_RSA_GEN_KEY"},
{ ERR_PACK(0, PK11_F_RSA_PUB_ENC, 0),           "PK11_RSA_PUB_ENC"},
{ ERR_PACK(0, PK11_F_RSA_PRIV_ENC, 0),          "PK11_RSA_PRIV_ENC"},
{ ERR_PACK(0, PK11_F_RSA_PUB_DEC, 0),           "PK11_RSA_PUB_DEC"},
{ ERR_PACK(0, PK11_F_RSA_PRIV_DEC, 0),          "PK11_RSA_PRIV_DEC"},
{ ERR_PACK(0, PK11_F_RSA_SIGN, 0),              "PK11_RSA_SIGN"},
{ ERR_PACK(0, PK11_F_RSA_VERIFY, 0),            "PK11_RSA_VERIFY"},
{ ERR_PACK(0, PK11_F_RAND_ADD, 0),              "PK11_RAND_ADD"},
{ ERR_PACK(0, PK11_F_RAND_BYTES, 0),            "PK11_RAND_BYTES"},
{ ERR_PACK(0, PK11_F_GET_SESSION, 0),           "PK11_GET_SESSION"},
{ ERR_PACK(0, PK11_F_FREE_SESSION, 0),          "PK11_FREE_SESSION"},
{ ERR_PACK(0, PK11_F_LOAD_PUBKEY, 0),           "PK11_LOAD_PUBKEY"},
{ ERR_PACK(0, PK11_F_LOAD_PRIVKEY, 0),          "PK11_LOAD_PRIV_KEY"},
{ ERR_PACK(0, PK11_F_RSA_PUB_ENC_LOW, 0),       "PK11_RSA_PUB_ENC_LOW"},
{ ERR_PACK(0, PK11_F_RSA_PRIV_ENC_LOW, 0),      "PK11_RSA_PRIV_ENC_LOW"},
{ ERR_PACK(0, PK11_F_RSA_PUB_DEC_LOW, 0),       "PK11_RSA_PUB_DEC_LOW"},
{ ERR_PACK(0, PK11_F_RSA_PRIV_DEC_LOW, 0),      "PK11_RSA_PRIV_DEC_LOW"},
{ ERR_PACK(0, PK11_F_DSA_SIGN, 0),              "PK11_DSA_SIGN"},
{ ERR_PACK(0, PK11_F_DSA_VERIFY, 0),            "PK11_DSA_VERIFY"},
{ ERR_PACK(0, PK11_F_DSA_INIT, 0),              "PK11_DSA_INIT"},
{ ERR_PACK(0, PK11_F_DSA_FINISH, 0),            "PK11_DSA_FINISH"},
{ ERR_PACK(0, PK11_F_GET_PUB_DSA_KEY, 0),       "PK11_GET_PUB_DSA_KEY"},
{ ERR_PACK(0, PK11_F_GET_PRIV_DSA_KEY, 0),      "PK11_GET_PRIV_DSA_KEY"},
{ ERR_PACK(0, PK11_F_DH_INIT, 0),               "PK11_DH_INIT"},
{ ERR_PACK(0, PK11_F_DH_FINISH, 0),             "PK11_DH_FINISH"},
{ ERR_PACK(0, PK11_F_MOD_EXP_DH, 0),            "PK11_MOD_EXP_DH"},
{ ERR_PACK(0, PK11_F_GET_DH_KEY, 0),            "PK11_GET_DH_KEY"},
{ ERR_PACK(0, PK11_F_FREE_ALL_SESSIONS, 0),     "PK11_FREE_ALL_SESSIONS"},
{ ERR_PACK(0, PK11_F_SETUP_SESSION, 0),         "PK11_SETUP_SESSION"},
{ ERR_PACK(0, PK11_F_DESTROY_OBJECT, 0),        "PK11_DESTROY_OBJECT"},
{ ERR_PACK(0, PK11_F_CIPHER_INIT, 0),           "PK11_CIPHER_INIT"},
{ ERR_PACK(0, PK11_F_CIPHER_DO_CIPHER, 0),      "PK11_CIPHER_DO_CIPHER"},
{ ERR_PACK(0, PK11_F_GET_CIPHER_KEY, 0),        "PK11_GET_CIPHER_KEY"},
{ ERR_PACK(0, PK11_F_DIGEST_INIT, 0),           "PK11_DIGEST_INIT"},
{ ERR_PACK(0, PK11_F_DIGEST_UPDATE, 0),         "PK11_DIGEST_UPDATE"},
{ ERR_PACK(0, PK11_F_DIGEST_FINAL, 0),          "PK11_DIGEST_FINAL"},
{ ERR_PACK(0, PK11_F_CHOOSE_SLOT, 0),           "PK11_CHOOSE_SLOT"},
{ ERR_PACK(0, PK11_F_CIPHER_FINAL, 0),          "PK11_CIPHER_FINAL"},
{ ERR_PACK(0, PK11_F_LIBRARY_INIT, 0),          "PK11_LIBRARY_INIT"},
{ ERR_PACK(0, PK11_F_LOAD, 0),                  "ENGINE_LOAD_PK11"},
{ ERR_PACK(0, PK11_F_DH_GEN_KEY, 0),            "PK11_DH_GEN_KEY"},
{ ERR_PACK(0, PK11_F_DH_COMP_KEY, 0),           "PK11_DH_COMP_KEY"},
```

```
{ ERR_PACK(0, PK11_F_DIGEST_COPY, 0),          "PK11_DIGEST_COPY"},
{ ERR_PACK(0, PK11_F_CIPHER_CLEANUP, 0),       "PK11_CIPHER_CLEANUP"},
{ ERR_PACK(0, PK11_F_ACTIVE_ADD, 0),           "PK11_ACTIVE_ADD"},
{ ERR_PACK(0, PK11_F_ACTIVE_DELETE, 0),        "PK11_ACTIVE_DELETE"},
{ ERR_PACK(0, PK11_F_CHECK_HW_MECHANISMS, 0),  "PK11_CHECK_HW_MECHANISMS"},
{ ERR_PACK(0, PK11_F_INIT_SYMMETRIC, 0),       "PK11_INIT_SYMMETRIC"},
{ ERR_PACK(0, PK11_F_ADD_AES_CTR_NIDS, 0),     "PK11_ADD_AES_CTR_NIDS"},
{ 0, NULL}
};

static ERR_STRING_DATA pk11_str_reasons[]=
{
{ PK11_R_ALREADY_LOADED,               "PKCS#11 DSO already loaded"},
{ PK11_R_DSO_FAILURE,                  "unable to load PKCS#11 DSO"},
{ PK11_R_NOT_LOADED,                   "PKCS#11 DSO not loaded"},
{ PK11_R_PASSED_NULL_PARAMETER,        "null parameter passed"},
{ PK11_R_COMMAND_NOT_IMPLEMENTED,      "command not implemented"},
{ PK11_R_INITIALIZE,                   "C_Initialize failed"},
{ PK11_R_FINALIZE,                     "C_Finalize failed"},
{ PK11_R_GETINFO,                      "C_GetInfo faile"},
{ PK11_R_GETSLOTLIST,                  "C_GetSlotList failed"},
{ PK11_R_NO_MODULUS_OR_NO_EXPONENT,    "no modulus or no exponent"},
{ PK11_R_ATTRIBUT_SENSITIVE_OR_INVALID, "attr sensitive or invalid"},
{ PK11_R_GETATTRIBUTVALUE,             "C_GetAttributeValue failed"},
{ PK11_R_NO_MODULUS,                   "no modulus"},
{ PK11_R_NO_EXPONENT,                  "no exponent"},
{ PK11_R_FINDOBJECTSINIT,              "C_FindObjectsInit failed"},
{ PK11_R_FINDOBJECTS,                  "C_FindObjects failed"},
{ PK11_R_FINDOBJECTSFINAL,             "C_FindObjectsFinal failed"},
{ PK11_R_CREATEOBJECT,                 "C_CreateObject failed"},
{ PK11_R_DESTROYOBJECT,                "C_DestroyObject failed"},
{ PK11_R_OPENSESSION,                  "C_OpenSession failed"},
{ PK11_R_CLOSESESSION,                 "C_CloseSession failed"},
{ PK11_R_ENCRYPTINIT,                  "C_EncryptInit failed"},
{ PK11_R_ENCRYPT,                      "C_Encrypt failed"},
{ PK11_R_SIGNINIT,                     "C_SignInit failed"},
{ PK11_R_SIGN,                         "C_Sign failed"},
{ PK11_R_DECRYPTINIT,                  "C_DecryptInit failed"},
{ PK11_R_DECRYPT,                      "C_Decrypt failed"},
{ PK11_R_VERIFYINIT,                   "C_VerifyRecover failed"},
{ PK11_R_VERIFY,                       "C_Verify failed"},
{ PK11_R_VERIFYRECOVERINIT,            "C_VerifyRecoverInit failed"},
{ PK11_R_VERIFYRECOVER,                "C_VerifyRecover failed"},
{ PK11_R_GEN_KEY,                      "C_GenerateKeyPair failed"},
{ PK11_R_SEEDRANDOM,                   "C_SeedRandom failed"},
{ PK11_R_GENERATERANDOM,               "C_GenerateRandom failed"},
{ PK11_R_INVALID_MESSAGE_LENGTH,       "invalid message length"},
{ PK11_R_UNKNOWN_ALGORITHM_TYPE,       "unknown algorithm type"},
{ PK11_R_UNKNOWN_ASN1_OBJECT_ID,       "unknown asn1 onject id"},
{ PK11_R_UNKNOWN_PADDING_TYPE,         "unknown padding type"},
{ PK11_R_PADDING_CHECK_FAILED,         "padding check failed"},
{ PK11_R_DIGEST_TOO_BIG,               "digest too big"},
{ PK11_R_MALLOC_FAILURE,               "malloc failure"},
{ PK11_R_CTRL_COMMAND_NOT_IMPLEMENTED, "ctl command not implemented"},
{ PK11_R_DATA_GREATER_THAN_MOD_LEN,    "data is bigger than mod"},
{ PK11_R_DATA_TOO_LARGE_FOR_MODULUS,   "data is too larger for mod"},
{ PK11_R_MISSING_KEY_COMPONENT,        "a dsa component is missing"},
{ PK11_R_INVALID_SIGNATURE_LENGTH,     "invalid signature length"},
{ PK11_R_INVALID_DSA_SIGNATURE_R,      "missing r in dsa verify"},
{ PK11_R_INVALID_DSA_SIGNATURE_S,      "missing s in dsa verify"},
{ PK11_R_INCONSISTENT_KEY,             "inconsistent key type"},
{ PK11_R_ENCRYPTUPDATE,                "C_EncryptUpdate failed"},
{ PK11_R_DECRYPTUPDATE,                "C_DecryptUpdate failed"},
{ PK11_R_DIGESTINIT,                   "C_DigestInit failed"},
{ PK11_R_DIGESTUPDATE,                 "C_DigestUpdate failed"},
{ PK11_R_DIGESTFINAL,                  "C_DigestFinal failed"},
```



```
{ PK11_R_ENCRYPTFINAL,                 "C_EncryptFinal failed"},
{ PK11_R_DECRYPTFINAL,                 "C_DecryptFinal failed"},
{ PK11_R_NO_PRNG_SUPPORT,              "Slot does not support PRNG"},
{ PK11_R_GETTOKENINFO,                 "C_GetTokenInfo failed"},
{ PK11_R_DERIVEKEY,                    "C_DeriveKey failed"},
{ PK11_R_GET_OPERATION_STATE,          "C_GetOperationState failed"},
{ PK11_R_SET_OPERATION_STATE,          "C_SetOperationState failed"},
{ PK11_R_INVALID_HANDLE,               "invalid PKCS#11 object handle"},
{ PK11_R_KEY_OR_IV_LEN_PROBLEM,        "IV or key length incorrect"},
{ PK11_R_INVALID_OPERATION_TYPE,       "invalid operation type"},
{ PK11_R_ADD_NID_FAILED,               "failed to add NID" },
{ 0,    NULL}
};
#endif  /* OPENSSL_NO_ERR */

static int pk11_lib_error_code = 0;
static int pk11_error_init = 1;

static void
ERR_load_pk11_strings(void)
        {
        if (pk11_lib_error_code == 0)
                pk11_lib_error_code = ERR_get_next_error_library();

        if (pk11_error_init)
                {
                pk11_error_init = 0;
#ifndef OPENSSL_NO_ERR
                ERR_load_strings(pk11_lib_error_code, pk11_str_functs);
                ERR_load_strings(pk11_lib_error_code, pk11_str_reasons);
#endif
                }
}

static void
ERR_unload_pk11_strings(void)
        {
        if (pk11_error_init == 0)
                {
#ifndef OPENSSL_NO_ERR
                ERR_unload_strings(pk11_lib_error_code, pk11_str_functs);
                ERR_unload_strings(pk11_lib_error_code, pk11_str_reasons);
#endif
                pk11_error_init = 1;
                }
}

void
ERR_pk11_error(int function, int reason, char *file, int line)
{
        if (pk11_lib_error_code == 0)
                pk11_lib_error_code = ERR_get_next_error_library();
        ERR_PUT_error(pk11_lib_error_code, function, reason, file, line);
}

void
PK11err_add_data(int function, int reason, CK_RV rv)
{
        char tmp_buf[20];

        PK11err(function, reason);
        (void) snprintf(tmp_buf, sizeof (tmp_buf), "%lx", rv);
        ERR_add_error_data(2, "PK11 CK_RV=0X", tmp_buf);
}
#endif /* ! codereview */
```

```
**********************************************************
   70067 Wed Aug 13 19:52:38 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/hw_pk11_pub.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/* crypto/engine/hw_pk11_pub.c */
/*
 * This product includes software developed by the OpenSSL Project for
 * use in the OpenSSL Toolkit (http://www.openssl.org/).
 *
 * This project also referenced hw_pkcs11-0.9.7b.patch written by
 * Afchine Madjlessi.
 */
/*
 * ====================================================================
 * Copyright (c) 2000-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
```



```
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>

#include <openssl/e_os2.h>
#include <openssl/crypto.h>
#include <openssl/engine.h>
#include <openssl/dso.h>
#include <openssl/err.h>
#include <openssl/bn.h>
#include <openssl/pem.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif /* OPENSSL_NO_RSA */
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif /* OPENSSL_NO_DSA */
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif /* OPENSSL_NO_DH */
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <cryptlib.h>
#include <pthread.h>

#ifndef OPENSSL_NO_HW
#ifndef OPENSSL_NO_HW_PK11

#include "cryptoki.h"
#include "pkcs11.h"
#include "hw_pk11_err.h"

#ifndef OPENSSL_NO_RSA
/* RSA stuff */
static int pk11_RSA_public_encrypt(int flen, const unsigned char *from,
        unsigned char *to, RSA *rsa, int padding);
static int pk11_RSA_private_encrypt(int flen, const unsigned char *from,
        unsigned char *to, RSA *rsa, int padding);
static int pk11_RSA_public_decrypt(int flen, const unsigned char *from,
        unsigned char *to, RSA *rsa, int padding);
static int pk11_RSA_private_decrypt(int flen, const unsigned char *from,
        unsigned char *to, RSA *rsa, int padding);
static int pk11_RSA_init(RSA *rsa);
static int pk11_RSA_finish(RSA *rsa);
static int pk11_RSA_sign(int type, const unsigned char *m, unsigned int m_len,
        unsigned char *sigret, unsigned int *siglen, const RSA *rsa);
static int pk11_RSA_verify(int dtype, const unsigned char *m,
        unsigned int m_len, const unsigned char *sigbuf, unsigned int siglen,
        const RSA *rsa);
EVP_PKEY *pk11_load_privkey(ENGINE*, const char *pubkey_file,
        UI_METHOD *ui_method, void *callback_data);
EVP_PKEY *pk11_load_pubkey(ENGINE*, const char *pubkey_file,
        UI_METHOD *ui_method, void *callback_data);

static int pk11_RSA_public_encrypt_low(int flen, const unsigned char *from,
        unsigned char *to, RSA *rsa);
```

```
128 static int pk11_RSA_private_encrypt_low(int flen, const unsigned char *from,
129         unsigned char *to, RSA *rsa);
130 static int pk11_RSA_public_decrypt_low(int flen, const unsigned char *from,
131         unsigned char *to, RSA *rsa);
132 static int pk11_RSA_private_decrypt_low(int flen, const unsigned char *from,
133         unsigned char *to, RSA *rsa);

135 static CK_OBJECT_HANDLE pk11_get_public_rsa_key(RSA* rsa, RSA** key_ptr,
136         BIGNUM **rsa_n_num, BIGNUM **rsa_e_num, CK_SESSION_HANDLE session);
137 static CK_OBJECT_HANDLE pk11_get_private_rsa_key(RSA* rsa, RSA** key_ptr,
138         BIGNUM **rsa_d_num, CK_SESSION_HANDLE session);

140 static int check_new_rsa_key_pub(PK11_SESSION *sp, const RSA *rsa);
141 static int check_new_rsa_key_priv(PK11_SESSION *sp, const RSA *rsa);
142 #endif

144 /* DSA stuff */
145 #ifndef OPENSSL_NO_DSA
146 static int pk11_DSA_init(DSA *dsa);
147 static int pk11_DSA_finish(DSA *dsa);
148 static DSA_SIG *pk11_dsa_do_sign(const unsigned char *dgst, int dlen,
149         DSA *dsa);
150 static int pk11_dsa_do_verify(const unsigned char *dgst, int dgst_len,
151         DSA_SIG *sig, DSA *dsa);

153 static CK_OBJECT_HANDLE pk11_get_public_dsa_key(DSA* dsa, DSA **key_ptr,
154         BIGNUM **dsa_pub_num, CK_SESSION_HANDLE session);
155 static CK_OBJECT_HANDLE pk11_get_private_dsa_key(DSA* dsa, DSA **key_ptr,
156         BIGNUM **dsa_priv_num, CK_SESSION_HANDLE session);

158 static int check_new_dsa_key_pub(PK11_SESSION *sp, DSA *dsa);
159 static int check_new_dsa_key_priv(PK11_SESSION *sp, DSA *dsa);
160 #endif

162 /* DH stuff */
163 #ifndef OPENSSL_NO_DH
164 static int pk11_DH_init(DH *dh);
165 static int pk11_DH_finish(DH *dh);
166 static int pk11_DH_generate_key(DH *dh);
167 static int pk11_DH_compute_key(unsigned char *key,
168         const BIGNUM *pub_key, DH *dh);

170 static CK_OBJECT_HANDLE pk11_get_dh_key(DH* dh, DH **key_ptr,
171         BIGNUM **priv_key, CK_SESSION_HANDLE session);

173 static int check_new_dh_key(PK11_SESSION *sp, DH *dh);
174 #endif

176 static int init_template_value(BIGNUM *bn, CK_VOID_PTR *pValue,
177         CK_ULONG *ulValueLen);

179 /* Read mode string to be used for fopen() */
180 #if SOLARIS_OPENSSL
181 static char *read_mode_flags = "rF";
182 #else
183 static char *read_mode_flags = "r";
184 #endif

186 /*
187  * increment/create reference for an asymmetric key handle via active list
188  * manipulation. If active list operation fails, unlock (if locked), set error
189  * variable and jump to the specified label.
190  */
191 #define KEY_HANDLE_REFHOLD(key_handle, alg_type, unlock, var, label)    \
192         {                                                               \
193         if (sunw_pk11_active_add(key_handle, alg_type) < 0)
```



```
194                 {                                                       \
195                 var = TRUE;                                             \
196                 if (unlock)                                             \
197                         UNLOCK_OBJSTORE(alg_type);                      \
198                 goto label;                                             \
199                 }                                                       \
200         }

202 /*
203  * Find active list entry according to object handle and return pointer to the
204  * entry otherwise return NULL.
205  *
206  * This function presumes it is called with lock protecting the active list
207  * held.
208  */
209 static PK11_active *pk11_active_find(CK_OBJECT_HANDLE h, PK11_OPTYPE type)
210         {
211         PK11_active *entry;

213         for (entry = active_list[type]; entry != NULL; entry = entry->next)
214                 if (entry->h == h)
215                         return (entry);

217         return (NULL);
218         }

220 /*
221  * Search for an entry in the active list using PKCS#11 object handle as a
222  * search key and return refcnt of the found/created entry or -1 in case of
223  * failure.
224  *
225  * This function presumes it is called with lock protecting the active list
226  * held.
227  */
228 int
229 sunw_pk11_active_add(CK_OBJECT_HANDLE h, PK11_OPTYPE type)
230         {
231         PK11_active *entry = NULL;

233         if (h == CK_INVALID_HANDLE)
234                 {
235                 PK11err(PK11_F_ACTIVE_ADD, PK11_R_INVALID_HANDLE);
236                 return (-1);
237                 }

239         /* search for entry in the active list */
240         if ((entry = pk11_active_find(h, type)) != NULL)
241                 entry->refcnt++;
242         else
243                 {
244                 /* not found, create new entry and add it to the list */
245                 entry = OPENSSL_malloc(sizeof (PK11_active));
246                 if (entry == NULL)
247                         {
248                         PK11err(PK11_F_ACTIVE_ADD, PK11_R_MALLOC_FAILURE);
249                         return (-1);
250                         }
251                 entry->h = h;
252                 entry->refcnt = 1;
253                 entry->prev = NULL;
254                 entry->next = NULL;
255                 /* connect the newly created entry to the list */
256                 if (active_list[type] == NULL)
257                         active_list[type] = entry;
258                 else /* make the entry first in the list */
259                         {
```

```
260                         entry->next = active_list[type];
261                         active_list[type]->prev = entry;
262                         active_list[type] = entry;
263                         }
264                 }

266         return (entry->refcnt);
267         }

269 /*
270  * Remove active list entry from the list and free it.
271  *
272  * This function presumes it is called with lock protecting the active list
273  * held.
274  */
275 void
276 sunw_pk11_active_remove(PK11_active *entry, PK11_OPTYPE type)
277         {
278         PK11_active *prev_entry;

280         /* remove the entry from the list and free it */
281         if ((prev_entry = entry->prev) != NULL)
282                 {
283                 prev_entry->next = entry->next;
284                 if (entry->next != NULL)
285                         entry->next->prev = prev_entry;
286                 }
287         else
288                 {
289                 active_list[type] = entry->next;
290                 /* we were the first but not the only one */
291                 if (entry->next != NULL)
292                         entry->next->prev = NULL;
293                 }

295         /* sanitization */
296         entry->h = CK_INVALID_HANDLE;
297         entry->prev = NULL;
298         entry->next = NULL;
299         OPENSSL_free(entry);
300         }

302 /* Free all entries from the active list. */
303 void
304 sunw_pk11_free_active_list(PK11_OPTYPE type)
305         {
306         PK11_active *entry;

308         /* only for asymmetric types since only they have C_Find* locks. */
309         switch (type)
310                 {
311                 case OP_RSA:
312                 case OP_DSA:
313                 case OP_DH:
314                         break;
315                 default:
316                         return;
317                 }

319         /* see find_lock array definition for more info on object locking */
320         LOCK_OBJSTORE(type);
321         while ((entry = active_list[type]) != NULL)
322                 sunw_pk11_active_remove(entry, type);
323         UNLOCK_OBJSTORE(type);
324         }
```



```
326 /*
327  * Search for active list entry associated with given PKCS#11 object handle,
328  * decrement its refcnt and if it drops to 0, disconnect the entry and free it.
329  *
330  * Return 1 if the PKCS#11 object associated with the entry has no references,
331  * return 0 if there is at least one reference, -1 on error.
332  *
333  * This function presumes it is called with lock protecting the active list
334  * held.
335  */
336 int
337 sunw_pk11_active_delete(CK_OBJECT_HANDLE h, PK11_OPTYPE type)
338         {
339         PK11_active *entry = NULL;

341         if ((entry = pk11_active_find(h, type)) == NULL)
342                 {
343                 PK11err(PK11_F_ACTIVE_DELETE, PK11_R_INVALID_HANDLE);
344                 return (-1);
345                 }

347         OPENSSL_assert(entry->refcnt > 0);
348         entry->refcnt--;
349         if (entry->refcnt == 0)
350                 {
351                 sunw_pk11_active_remove(entry, type);
352                 return (1);
353                 }

355         return (0);
356         }

358 #ifndef OPENSSL_NO_RSA
359 /* Our internal RSA_METHOD that we provide pointers to */
360 static RSA_METHOD pk11_rsa =
361         {
362         "PKCS#11 RSA method",
363         pk11_RSA_public_encrypt,                /* rsa_pub_encrypt */
364         pk11_RSA_public_decrypt,                /* rsa_pub_decrypt */
365         pk11_RSA_private_encrypt,               /* rsa_priv_encrypt */
366         pk11_RSA_private_decrypt,               /* rsa_priv_decrypt */
367         NULL,                                   /* rsa_mod_exp */
368         NULL,                                   /* bn_mod_exp */
369         pk11_RSA_init,                          /* init */
370         pk11_RSA_finish,                        /* finish */
371         RSA_FLAG_SIGN_VER,                      /* flags */
372         NULL,                                   /* app_data */
373         pk11_RSA_sign,                          /* rsa_sign */
374         pk11_RSA_verify                         /* rsa_verify */
375         };

377 RSA_METHOD *
378 PK11_RSA(void)
379         {
380         return (&pk11_rsa);
381         }
382 #endif

384 #ifndef OPENSSL_NO_DSA
385 /* Our internal DSA_METHOD that we provide pointers to */
386 static DSA_METHOD pk11_dsa =
387         {
388         "PKCS#11 DSA method",
389         pk11_dsa_do_sign,       /* dsa_do_sign */
390         NULL,                   /* dsa_sign_setup */
391         pk11_dsa_do_verify,     /* dsa_do_verify */
```

```
392         NULL,                   /* dsa_mod_exp */
393         NULL,                   /* bn_mod_exp */
394         pk11_DSA_init,          /* init */
395         pk11_DSA_finish,        /* finish */
396         0,                      /* flags */
397         NULL                    /* app_data */
398         };

400 DSA_METHOD *
401 PK11_DSA(void)
402         {
403         return (&pk11_dsa);
404         }
405 #endif

407 #ifndef OPENSSL_NO_DH
408 /*
409  * PKCS #11 V2.20, section 11.2 specifies that the number of bytes needed for
410  * output buffer may somewhat exceed the precise number of bytes needed, but
411  * should not exceed it by a large amount. That may be caused, for example, by
412  * rounding it up to multiple of X in the underlying bignum library. 8 should be
413  * enough.
414  */
415 #define DH_BUF_RESERVE  8

417 /* Our internal DH_METHOD that we provide pointers to */
418 static DH_METHOD pk11_dh =
419         {
420         "PKCS#11 DH method",
421         pk11_DH_generate_key,   /* generate_key */
422         pk11_DH_compute_key,    /* compute_key */
423         NULL,                   /* bn_mod_exp */
424         pk11_DH_init,           /* init */
425         pk11_DH_finish,         /* finish */
426         0,                      /* flags */
427         NULL,                   /* app_data */
428         NULL                    /* generate_params */
429         };

431 DH_METHOD *
432 PK11_DH(void)
433         {
434         return (&pk11_dh);
435         }
436 #endif

438 /* Size of an SSL signature: MD5+SHA1 */
439 #define SSL_SIG_LENGTH          36

441 /* Lengths of DSA data and signature */
442 #define DSA_DATA_LEN            20
443 #define DSA_SIGNATURE_LEN       40

445 static CK_BBOOL true = TRUE;
446 static CK_BBOOL false = FALSE;

448 #ifndef OPENSSL_NO_RSA
449 /*
450  * Similiar to OpenSSL to take advantage of the paddings. The goal is to
451  * support all paddings in this engine although PK11 library does not
452  * support all the paddings used in OpenSSL.
453  * The input errors should have been checked in the padding functions.
454  */
455 static int pk11_RSA_public_encrypt(int flen, const unsigned char *from,
456                 unsigned char *to, RSA *rsa, int padding)
457         {
```



```
458         int i, num = 0, r = -1;
459         unsigned char *buf = NULL;

461         num = BN_num_bytes(rsa->n);
462         if ((buf = (unsigned char *)OPENSSL_malloc(num)) == NULL)
463                 {
464                 RSAerr(PK11_F_RSA_PUB_ENC, PK11_R_MALLOC_FAILURE);
465                 goto err;
466                 }

468         switch (padding)
469                 {
470         case RSA_PKCS1_PADDING:
471                 i = RSA_padding_add_PKCS1_type_2(buf, num, from, flen);
472                 break;
473 #ifndef OPENSSL_NO_SHA
474         case RSA_PKCS1_OAEP_PADDING:
475                 i = RSA_padding_add_PKCS1_OAEP(buf, num, from, flen, NULL, 0);
476                 break;
477 #endif
478         case RSA_SSLV23_PADDING:
479                 i = RSA_padding_add_SSLv23(buf, num, from, flen);
480                 break;
481         case RSA_NO_PADDING:
482                 i = RSA_padding_add_none(buf, num, from, flen);
483                 break;
484         default:
485                 RSAerr(PK11_F_RSA_PUB_ENC, PK11_R_UNKNOWN_PADDING_TYPE);
486                 goto err;
487                 }
488         if (i <= 0) goto err;

490         /* PK11 functions are called here */
491         r = pk11_RSA_public_encrypt_low(num, buf, to, rsa);
492 err:
493         if (buf != NULL)
494                 {
495                 OPENSSL_cleanse(buf, num);
496                 OPENSSL_free(buf);
497                 }
498         return (r);
499         }


502 /*
503  * Similar to Openssl to take advantage of the paddings. The input errors
504  * should be catched in the padding functions
505  */
506 static int pk11_RSA_private_encrypt(int flen, const unsigned char *from,
507         unsigned char *to, RSA *rsa, int padding)
508         {
509         int i, num = 0, r = -1;
510         unsigned char *buf = NULL;

512         num = BN_num_bytes(rsa->n);
513         if ((buf = (unsigned char *)OPENSSL_malloc(num)) == NULL)
514                 {
515                 RSAerr(PK11_F_RSA_PRIV_ENC, PK11_R_MALLOC_FAILURE);
516                 goto err;
517                 }

519         switch (padding)
520                 {
521         case RSA_PKCS1_PADDING:
522                 i = RSA_padding_add_PKCS1_type_1(buf, num, from, flen);
523                 break;
```

```
524         case RSA_NO_PADDING:
525                 i = RSA_padding_add_none(buf, num, from, flen);
526                 break;
527         case RSA_SSLV23_PADDING:
528         default:
529                 RSAerr(PK11_F_RSA_PRIV_ENC, PK11_R_UNKNOWN_PADDING_TYPE);
530                 goto err;
531                 }
532         if (i <= 0) goto err;

534         /* PK11 functions are called here */
535         r = pk11_RSA_private_encrypt_low(num, buf, to, rsa);
536 err:
537         if (buf != NULL)
538                 {
539                 OPENSSL_cleanse(buf, num);
540                 OPENSSL_free(buf);
541                 }
542         return (r);
543         }

545 /* Similar to OpenSSL code. Input errors are also checked here */
546 static int pk11_RSA_private_decrypt(int flen, const unsigned char *from,
547         unsigned char *to, RSA *rsa, int padding)
548         {
549         BIGNUM f;
550         int j, num = 0, r = -1;
551         unsigned char *p;
552         unsigned char *buf = NULL;

554         BN_init(&f);

556         num = BN_num_bytes(rsa->n);

558         if ((buf = (unsigned char *)OPENSSL_malloc(num)) == NULL)
559                 {
560                 RSAerr(PK11_F_RSA_PRIV_DEC, PK11_R_MALLOC_FAILURE);
561                 goto err;
562                 }

564         /*
565          * This check was for equality but PGP does evil things
566          * and chops off the top '0' bytes
567          */
568         if (flen > num)
569                 {
570                 RSAerr(PK11_F_RSA_PRIV_DEC,
571                         PK11_R_DATA_GREATER_THAN_MOD_LEN);
572                 goto err;
573                 }

575         /* make data into a big number */
576         if (BN_bin2bn(from, (int)flen, &f) == NULL)
577                 goto err;

579         if (BN_ucmp(&f, rsa->n) >= 0)
580                 {
581                 RSAerr(PK11_F_RSA_PRIV_DEC,
582                         PK11_R_DATA_TOO_LARGE_FOR_MODULUS);
583                 goto err;
584                 }

586         /* PK11 functions are called here */
587         r = pk11_RSA_private_decrypt_low(flen, from, buf, rsa);

589         /*
```



```
590          * PK11 CKM_RSA_X_509 mechanism pads 0's at the beginning.
591          * Needs to skip these 0's paddings here.
592          */
593         for (j = 0; j < r; j++)
594                 if (buf[j] != 0)
595                         break;

597         p = buf + j;
598         j = r - j;  /* j is only used with no-padding mode */

600         switch (padding)
601                 {
602         case RSA_PKCS1_PADDING:
603                 r = RSA_padding_check_PKCS1_type_2(to, num, p, j, num);
604                 break;
605 #ifndef OPENSSL_NO_SHA
606         case RSA_PKCS1_OAEP_PADDING:
607                 r = RSA_padding_check_PKCS1_OAEP(to, num, p, j, num, NULL, 0);
608                 break;
609 #endif
610         case RSA_SSLV23_PADDING:
611                 r = RSA_padding_check_SSLv23(to, num, p, j, num);
612                 break;
613         case RSA_NO_PADDING:
614                 r = RSA_padding_check_none(to, num, p, j, num);
615                 break;
616         default:
617                 RSAerr(PK11_F_RSA_PRIV_DEC, PK11_R_UNKNOWN_PADDING_TYPE);
618                 goto err;
619                 }
620         if (r < 0)
621                 RSAerr(PK11_F_RSA_PRIV_DEC, PK11_R_PADDING_CHECK_FAILED);

623 err:
624         BN_clear_free(&f);
625         if (buf != NULL)
626                 {
627                 OPENSSL_cleanse(buf, num);
628                 OPENSSL_free(buf);
629                 }
630         return (r);
631         }

633 /* Similar to OpenSSL code. Input errors are also checked here */
634 static int pk11_RSA_public_decrypt(int flen, const unsigned char *from,
635         unsigned char *to, RSA *rsa, int padding)
636         {
637         BIGNUM f;
638         int i, num = 0, r = -1;
639         unsigned char *p;
640         unsigned char *buf = NULL;

642         BN_init(&f);
643         num = BN_num_bytes(rsa->n);
644         buf = (unsigned char *)OPENSSL_malloc(num);
645         if (buf == NULL)
646                 {
647                 RSAerr(PK11_F_RSA_PUB_DEC, PK11_R_MALLOC_FAILURE);
648                 goto err;
649                 }

651         /*
652          * This check was for equality but PGP does evil things
653          * and chops off the top '0' bytes
654          */
655         if (flen > num)
```

```
656                 {
657                 RSAerr(PK11_F_RSA_PUB_DEC, PK11_R_DATA_GREATER_THAN_MOD_LEN);
658                 goto err;
659                 }

661         if (BN_bin2bn(from, flen, &f) == NULL)
662                 goto err;

664         if (BN_ucmp(&f, rsa->n) >= 0)
665                 {
666                 RSAerr(PK11_F_RSA_PUB_DEC,
667                         PK11_R_DATA_TOO_LARGE_FOR_MODULUS);
668                 goto err;
669                 }

671         /* PK11 functions are called here */
672         r = pk11_RSA_public_decrypt_low(flen, from, buf, rsa);

674         /*
675          * PK11 CKM_RSA_X_509 mechanism pads 0's at the beginning.
676          * Needs to skip these 0's here
677          */
678         for (i = 0; i < r; i++)
679                 if (buf[i] != 0)
680                         break;

682         p = buf + i;
683         i = r - i;  /* i is only used with no-padding mode */

685         switch (padding)
686                 {
687         case RSA_PKCS1_PADDING:
688                 r = RSA_padding_check_PKCS1_type_1(to, num, p, i, num);
689                 break;
690         case RSA_NO_PADDING:
691                 r = RSA_padding_check_none(to, num, p, i, num);
692                 break;
693         default:
694                 RSAerr(PK11_F_RSA_PUB_DEC, PK11_R_UNKNOWN_PADDING_TYPE);
695                 goto err;
696                 }
697         if (r < 0)
698                 RSAerr(PK11_F_RSA_PUB_DEC, PK11_R_PADDING_CHECK_FAILED);

700 err:
701         BN_clear_free(&f);
702         if (buf != NULL)
703                 {
704                 OPENSSL_cleanse(buf, num);
705                 OPENSSL_free(buf);
706                 }
707         return (r);
708         }

710 /*
711  * This function implements RSA public encryption using C_EncryptInit and
712  * C_Encrypt pk11 interfaces. Note that the CKM_RSA_X_509 is used here.
713  * The calling function allocated sufficient memory in "to" to store results.
714  */
715 static int pk11_RSA_public_encrypt_low(int flen,
716         const unsigned char *from, unsigned char *to, RSA *rsa)
717         {
718         CK_ULONG bytes_encrypted = flen;
719         int retval = -1;
720         CK_RV rv;
721         CK_MECHANISM mech_rsa = {CKM_RSA_X_509, NULL, 0};
```



```
722         CK_MECHANISM *p_mech = &mech_rsa;
723         CK_OBJECT_HANDLE h_pub_key = CK_INVALID_HANDLE;
724         PK11_SESSION *sp;

726         if ((sp = pk11_get_session(OP_RSA)) == NULL)
727                 return (-1);

729         (void) check_new_rsa_key_pub(sp, rsa);

731         h_pub_key = sp->opdata_rsa_pub_key;
732         if (h_pub_key == CK_INVALID_HANDLE)
733                 h_pub_key = sp->opdata_rsa_pub_key =
734                         pk11_get_public_rsa_key(rsa, &sp->opdata_rsa_pub,
735                             &sp->opdata_rsa_n_num, &sp->opdata_rsa_e_num,
736                             sp->session);

738         if (h_pub_key != CK_INVALID_HANDLE)
739                 {
740                 rv = pFuncList->C_EncryptInit(sp->session, p_mech,
741                         h_pub_key);

743                 if (rv != CKR_OK)
744                         {
745                         PK11err_add_data(PK11_F_RSA_PUB_ENC_LOW,
746                             PK11_R_ENCRYPTINIT, rv);
747                         pk11_return_session(sp, OP_RSA);
748                         return (-1);
749                         }

751                 rv = pFuncList->C_Encrypt(sp->session,
752                         (unsigned char *)from, flen, to, &bytes_encrypted);

754                 if (rv != CKR_OK)
755                         {
756                         PK11err_add_data(PK11_F_RSA_PUB_ENC_LOW,
757                             PK11_R_ENCRYPT, rv);
758                         pk11_return_session(sp, OP_RSA);
759                         return (-1);
760                         }
761                 retval = bytes_encrypted;
762                 }

764         pk11_return_session(sp, OP_RSA);
765         return (retval);
766         }


769 /*
770  * This function implements RSA private encryption using C_SignInit and
771  * C_Sign pk11 APIs. Note that CKM_RSA_X_509 is used here.
772  * The calling function allocated sufficient memory in "to" to store results.
773  */
774 static int pk11_RSA_private_encrypt_low(int flen,
775         const unsigned char *from, unsigned char *to, RSA *rsa)
776         {
777         CK_ULONG ul_sig_len = flen;
778         int retval = -1;
779         CK_RV rv;
780         CK_MECHANISM mech_rsa = {CKM_RSA_X_509, NULL, 0};
781         CK_MECHANISM *p_mech = &mech_rsa;
782         CK_OBJECT_HANDLE h_priv_key = CK_INVALID_HANDLE;
783         PK11_SESSION *sp;

785         if ((sp = pk11_get_session(OP_RSA)) == NULL)
786                 return (-1);
```

```
788         (void) check_new_rsa_key_priv(sp, rsa);

790         h_priv_key = sp->opdata_rsa_priv_key;
791         if (h_priv_key == CK_INVALID_HANDLE)
792                 h_priv_key = sp->opdata_rsa_priv_key =
793                         pk11_get_private_rsa_key(rsa, &sp->opdata_rsa_priv,
794                             &sp->opdata_rsa_d_num, sp->session);

796         if (h_priv_key != CK_INVALID_HANDLE)
797                 {
798                 rv = pFuncList->C_SignInit(sp->session, p_mech,
799                         h_priv_key);

801                 if (rv != CKR_OK)
802                         {
803                         PK11err_add_data(PK11_F_RSA_PRIV_ENC_LOW,
804                             PK11_R_SIGNINIT, rv);
805                         pk11_return_session(sp, OP_RSA);
806                         return (-1);
807                         }

809                 rv = pFuncList->C_Sign(sp->session,
810                         (unsigned char *)from, flen, to, &ul_sig_len);

812                 if (rv != CKR_OK)
813                         {
814                         PK11err_add_data(PK11_F_RSA_PRIV_ENC_LOW, PK11_R_SIGN,
815                             rv);
816                         pk11_return_session(sp, OP_RSA);
817                         return (-1);
818                         }

820                 retval = ul_sig_len;
821                 }

823         pk11_return_session(sp, OP_RSA);
824         return (retval);
825         }


828 /*
829  * This function implements RSA private decryption using C_DecryptInit and
830  * C_Decrypt pk11 APIs. Note that CKM_RSA_X_509 mechanism is used here.
831  * The calling function allocated sufficient memory in "to" to store results.
832  */
833 static int pk11_RSA_private_decrypt_low(int flen,
834         const unsigned char *from, unsigned char *to, RSA *rsa)
835         {
836         CK_ULONG bytes_decrypted = flen;
837         int retval = -1;
838         CK_RV rv;
839         CK_MECHANISM mech_rsa = {CKM_RSA_X_509, NULL, 0};
840         CK_MECHANISM *p_mech = &mech_rsa;
841         CK_OBJECT_HANDLE h_priv_key;
842         PK11_SESSION *sp;

844         if ((sp = pk11_get_session(OP_RSA)) == NULL)
845                 return (-1);

847         (void) check_new_rsa_key_priv(sp, rsa);

849         h_priv_key = sp->opdata_rsa_priv_key;
850         if (h_priv_key == CK_INVALID_HANDLE)
851                 h_priv_key = sp->opdata_rsa_priv_key =
852                         pk11_get_private_rsa_key(rsa, &sp->opdata_rsa_priv,
853                             &sp->opdata_rsa_d_num, sp->session);
```



```
855         if (h_priv_key != CK_INVALID_HANDLE)
856                 {
857                 rv = pFuncList->C_DecryptInit(sp->session, p_mech,
858                         h_priv_key);

860                 if (rv != CKR_OK)
861                         {
862                         PK11err_add_data(PK11_F_RSA_PRIV_DEC_LOW,
863                                 PK11_R_DECRYPTINIT, rv);
864                         pk11_return_session(sp, OP_RSA);
865                         return (-1);
866                         }

868                 rv = pFuncList->C_Decrypt(sp->session,
869                         (unsigned char *)from, flen, to, &bytes_decrypted);

871                 if (rv != CKR_OK)
872                         {
873                         PK11err_add_data(PK11_F_RSA_PRIV_DEC_LOW,
874                             PK11_R_DECRYPT, rv);
875                         pk11_return_session(sp, OP_RSA);
876                         return (-1);
877                         }
878                 retval = bytes_decrypted;
879                 }

881         pk11_return_session(sp, OP_RSA);
882         return (retval);
883         }


886 /*
887  * This function implements RSA public decryption using C_VerifyRecoverInit
888  * and C_VerifyRecover pk11 APIs. Note that CKM_RSA_X_509 is used here.
889  * The calling function allocated sufficient memory in "to" to store results.
890  */
891 static int pk11_RSA_public_decrypt_low(int flen,
892         const unsigned char *from, unsigned char *to, RSA *rsa)
893         {
894         CK_ULONG bytes_decrypted = flen;
895         int retval = -1;
896         CK_RV rv;
897         CK_MECHANISM mech_rsa = {CKM_RSA_X_509, NULL, 0};
898         CK_MECHANISM *p_mech = &mech_rsa;
899         CK_OBJECT_HANDLE h_pub_key = CK_INVALID_HANDLE;
900         PK11_SESSION *sp;

902         if ((sp = pk11_get_session(OP_RSA)) == NULL)
903                 return (-1);

905         (void) check_new_rsa_key_pub(sp, rsa);

907         h_pub_key = sp->opdata_rsa_pub_key;
908         if (h_pub_key == CK_INVALID_HANDLE)
909                 h_pub_key = sp->opdata_rsa_pub_key =
910                         pk11_get_public_rsa_key(rsa, &sp->opdata_rsa_pub,
911                             &sp->opdata_rsa_n_num, &sp->opdata_rsa_e_num,
912                             sp->session);

914         if (h_pub_key != CK_INVALID_HANDLE)
915                 {
916                 rv = pFuncList->C_VerifyRecoverInit(sp->session,
917                         p_mech, h_pub_key);

919                 if (rv != CKR_OK)
```

```
920                         {
921                         PK11err_add_data(PK11_F_RSA_PUB_DEC_LOW,
922                                 PK11_R_VERIFYRECOVERINIT, rv);
923                         pk11_return_session(sp, OP_RSA);
924                         return (-1);
925                         }

927                 rv = pFuncList->C_VerifyRecover(sp->session,
928                         (unsigned char *)from, flen, to, &bytes_decrypted);

930                 if (rv != CKR_OK)
931                         {
932                         PK11err_add_data(PK11_F_RSA_PUB_DEC_LOW,
933                             PK11_R_VERIFYRECOVER, rv);
934                         pk11_return_session(sp, OP_RSA);
935                         return (-1);
936                         }
937                 retval = bytes_decrypted;
938                 }

940         pk11_return_session(sp, OP_RSA);
941         return (retval);
942         }

944 static int pk11_RSA_init(RSA *rsa)
945         {
946         /*
947          * This flag in the RSA_METHOD enables the new rsa_sign,
948          * rsa_verify functions. See rsa.h for details.
949          */
950         rsa->flags |= RSA_FLAG_SIGN_VER;

952         return (1);
953         }

955 static int pk11_RSA_finish(RSA *rsa)
956         {
957         /*
958          * Since we are overloading OpenSSL's native RSA_eay_finish() we need
959          * to do the same as in the original function, i.e. to free bignum
960          * structures.
961          */
962         if (rsa->_method_mod_n != NULL)
963                 BN_MONT_CTX_free(rsa->_method_mod_n);
964         if (rsa->_method_mod_p != NULL)
965                 BN_MONT_CTX_free(rsa->_method_mod_p);
966         if (rsa->_method_mod_q != NULL)
967                 BN_MONT_CTX_free(rsa->_method_mod_q);

969         return (1);
970         }

972 /*
973  * Standard engine interface function. Majority codes here are from
974  * rsa/rsa_sign.c. We replaced the decrypt function call by C_Sign of PKCS#11.
975  * See more details in rsa/rsa_sign.c
976  */
977 static int pk11_RSA_sign(int type, const unsigned char *m, unsigned int m_len,
978         unsigned char *sigret, unsigned int *siglen, const RSA *rsa)
979         {
980         X509_SIG sig;
981         ASN1_TYPE parameter;
982         int i, j=0;
983         unsigned char *p, *s = NULL;
984         X509_ALGOR algor;
985         ASN1_OCTET_STRING digest;
```



```
986         CK_RV rv;
987         CK_MECHANISM mech_rsa = {CKM_RSA_PKCS, NULL, 0};
988         CK_MECHANISM *p_mech = &mech_rsa;
989         CK_OBJECT_HANDLE h_priv_key;
990         PK11_SESSION *sp = NULL;
991         int ret = 0;
992         unsigned long ulsiglen;

994         /* Encode the digest */
995         /* Special case: SSL signature, just check the length */
996         if (type == NID_md5_sha1)
997                 {
998                 if (m_len != SSL_SIG_LENGTH)
999                         {
1000                        PK11err(PK11_F_RSA_SIGN,
1001                                PK11_R_INVALID_MESSAGE_LENGTH);
1002                        goto err;
1003                        }
1004                i = SSL_SIG_LENGTH;
1005                s = (unsigned char *)m;
1006                }
1007        else
1008                {
1009                sig.algor = &algor;
1010                sig.algor->algorithm = OBJ_nid2obj(type);
1011                if (sig.algor->algorithm == NULL)
1012                        {
1013                        PK11err(PK11_F_RSA_SIGN,
1014                                PK11_R_UNKNOWN_ALGORITHM_TYPE);
1015                        goto err;
1016                        }
1017                if (sig.algor->algorithm->length == 0)
1018                        {
1019                        PK11err(PK11_F_RSA_SIGN,
1020                                PK11_R_UNKNOWN_ASN1_OBJECT_ID);
1021                        goto err;
1022                        }
1023                parameter.type = V_ASN1_NULL;
1024                parameter.value.ptr = NULL;
1025                sig.algor->parameter = &parameter;

1027                sig.digest = &digest;
1028                sig.digest->data = (unsigned char *)m;
1029                sig.digest->length = m_len;

1031                i = i2d_X509_SIG(&sig, NULL);
1032                }

1034        j = RSA_size(rsa);
1035        if ((i - RSA_PKCS1_PADDING) > j)
1036                {
1037                PK11err(PK11_F_RSA_SIGN, PK11_R_DIGEST_TOO_BIG);
1038                goto err;
1039                }

1041        if (type != NID_md5_sha1)
1042                {
1043                s = (unsigned char *)OPENSSL_malloc((unsigned int)(j + 1));
1044                if (s == NULL)
1045                        {
1046                        PK11err(PK11_F_RSA_SIGN, PK11_R_MALLOC_FAILURE);
1047                        goto err;
1048                        }
1049                p = s;
1050                (void) i2d_X509_SIG(&sig, &p);
1051                }
```

```
1053         if ((sp = pk11_get_session(OP_RSA)) == NULL)
1054                 goto err;

1056         (void) check_new_rsa_key_priv(sp, rsa);

1058         h_priv_key = sp->opdata_rsa_priv_key;
1059         if (h_priv_key == CK_INVALID_HANDLE)
1060                 h_priv_key = sp->opdata_rsa_priv_key =
1061                         pk11_get_private_rsa_key((RSA *)rsa,
1062                             &sp->opdata_rsa_priv,
1063                             &sp->opdata_rsa_d_num, sp->session);

1065         if (h_priv_key != CK_INVALID_HANDLE)
1066                 {
1067                 rv = pFuncList->C_SignInit(sp->session, p_mech, h_priv_key);

1069                 if (rv != CKR_OK)
1070                         {
1071                         PK11err_add_data(PK11_F_RSA_SIGN, PK11_R_SIGNINIT, rv);
1072                         goto err;
1073                         }

1075                 ulsiglen = j;
1076                 rv = pFuncList->C_Sign(sp->session, s, i, sigret,
1077                         (CK_ULONG_PTR) &ulsiglen);
1078                 *siglen = ulsiglen;

1080                 if (rv != CKR_OK)
1081                         {
1082                         PK11err_add_data(PK11_F_RSA_SIGN, PK11_R_SIGN, rv);
1083                         goto err;
1084                         }
1085                 ret = 1;
1086                 }

1088 err:
1089         if (type != NID_md5_sha1)
1090                 {
1091                 (void) memset(s, 0, (unsigned int)(j + 1));
1092                 OPENSSL_free(s);
1093                 }

1095         pk11_return_session(sp, OP_RSA);
1096         return (ret);
1097         }

1099 static int pk11_RSA_verify(int type, const unsigned char *m,
1100         unsigned int m_len, const unsigned char *sigbuf, unsigned int siglen,
1101         const RSA *rsa)
1102         {
1103         X509_SIG sig;
1104         ASN1_TYPE parameter;
1105         int i, j;
1106         unsigned char *p, *s = NULL;
1107         X509_ALGOR algor;
1108         ASN1_OCTET_STRING digest;
1109         CK_RV rv;
1110         CK_MECHANISM mech_rsa = {CKM_RSA_PKCS, NULL, 0};
1111         CK_MECHANISM *p_mech = &mech_rsa;
1112         CK_OBJECT_HANDLE h_pub_key;
1113         PK11_SESSION *sp = NULL;
1114         int ret = 0;

1116         /* Encode the digest    */
1117         /* Special case: SSL signature, just check the length */
```



```
1118         if (type == NID_md5_sha1)
1119                 {
1120                 if (m_len != SSL_SIG_LENGTH)
1121                         {
1122                         PK11err(PK11_F_RSA_VERIFY,
1123                                 PK11_R_INVALID_MESSAGE_LENGTH);
1124                         goto err;
1125                         }
1126                 i = SSL_SIG_LENGTH;
1127                 s = (unsigned char *)m;
1128                 }
1129         else
1130                 {
1131                 sig.algor = &algor;
1132                 sig.algor->algorithm = OBJ_nid2obj(type);
1133                 if (sig.algor->algorithm == NULL)
1134                         {
1135                         PK11err(PK11_F_RSA_VERIFY,
1136                                 PK11_R_UNKNOWN_ALGORITHM_TYPE);
1137                         goto err;
1138                         }
1139                 if (sig.algor->algorithm->length == 0)
1140                         {
1141                         PK11err(PK11_F_RSA_VERIFY,
1142                                 PK11_R_UNKNOWN_ASN1_OBJECT_ID);
1143                         goto err;
1144                         }
1145                 parameter.type = V_ASN1_NULL;
1146                 parameter.value.ptr = NULL;
1147                 sig.algor->parameter = &parameter;
1148                 sig.digest = &digest;
1149                 sig.digest->data = (unsigned char *)m;
1150                 sig.digest->length = m_len;
1151                 i = i2d_X509_SIG(&sig, NULL);
1152                 }

1154         j = RSA_size(rsa);
1155         if ((i - RSA_PKCS1_PADDING) > j)
1156                 {
1157                 PK11err(PK11_F_RSA_VERIFY, PK11_R_DIGEST_TOO_BIG);
1158                 goto err;
1159                 }

1161         if (type != NID_md5_sha1)
1162                 {
1163                 s = (unsigned char *)OPENSSL_malloc((unsigned int)(j + 1));
1164                 if (s == NULL)
1165                         {
1166                         PK11err(PK11_F_RSA_VERIFY, PK11_R_MALLOC_FAILURE);
1167                         goto err;
1168                         }
1169                 p = s;
1170                 (void) i2d_X509_SIG(&sig, &p);
1171                 }

1173         if ((sp = pk11_get_session(OP_RSA)) == NULL)
1174                 goto err;

1176         (void) check_new_rsa_key_pub(sp, rsa);

1178         h_pub_key = sp->opdata_rsa_pub_key;
1179         if (h_pub_key == CK_INVALID_HANDLE)
1180                 h_pub_key = sp->opdata_rsa_pub_key =
1181                         pk11_get_public_rsa_key((RSA *)rsa, &sp->opdata_rsa_pub,
1182                             &sp->opdata_rsa_n_num, &sp->opdata_rsa_e_num,
1183                             sp->session);
```

```
1185         if (h_pub_key != CK_INVALID_HANDLE)
1186                 {
1187                 rv = pFuncList->C_VerifyInit(sp->session, p_mech,
1188                         h_pub_key);

1190                 if (rv != CKR_OK)
1191                         {
1192                         PK11err_add_data(PK11_F_RSA_VERIFY, PK11_R_VERIFYINIT,
1193                             rv);
1194                         goto err;
1195                         }
1196                 rv = pFuncList->C_Verify(sp->session, s, i, (CK_BYTE_PTR) sigbuf
1197                         (CK_ULONG)siglen);

1199                 if (rv != CKR_OK)
1200                         {
1201                         PK11err_add_data(PK11_F_RSA_VERIFY, PK11_R_VERIFY, rv);
1202                         goto err;
1203                         }
1204                 ret = 1;
1205                 }

1207 err:
1208         if (type != NID_md5_sha1)
1209                 {
1210                 (void) memset(s, 0, (unsigned int)siglen);
1211                 OPENSSL_free(s);
1212                 }

1214         pk11_return_session(sp, OP_RSA);
1215         return (ret);
1216         }

1218 /* load RSA private key from a file */
1219 /* ARGSUSED */
1220 EVP_PKEY *pk11_load_privkey(ENGINE* e, const char *privkey_file,
1221         UI_METHOD *ui_method, void *callback_data)
1222         {
1223         EVP_PKEY *pkey = NULL;
1224         FILE *pubkey;
1225         CK_OBJECT_HANDLE  h_priv_key = CK_INVALID_HANDLE;
1226         RSA *rsa;
1227         PK11_SESSION *sp;

1229         if ((sp = pk11_get_session(OP_RSA)) == NULL)
1230                 return (NULL);

1232         if ((pubkey = fopen(privkey_file, read_mode_flags)) != NULL)
1233                 {
1234                 pkey = PEM_read_PrivateKey(pubkey, NULL, NULL, NULL);
1235                 (void) fclose(pubkey);
1236                 if (pkey != NULL)
1237                         {
1238                         rsa = EVP_PKEY_get1_RSA(pkey);
1239                         if (rsa != NULL)
1240                                 {
1241                                 (void) check_new_rsa_key_priv(sp, rsa);

1243                                 h_priv_key = sp->opdata_rsa_priv_key =
1244                                     pk11_get_private_rsa_key(rsa,
1245                                     &sp->opdata_rsa_priv, &sp->opdata_rsa_d_num,
1246                                     sp->session);
1247                                 if (h_priv_key == CK_INVALID_HANDLE)
1248                                         {
1249                                         EVP_PKEY_free(pkey);
```



```
1250                                         pkey = NULL;
1251                                         }
1252                                 }
1253                         else
1254                                 {
1255                                 EVP_PKEY_free(pkey);
1256                                 pkey = NULL;
1257                                 }
1258                         }
1259                 }

1261         pk11_return_session(sp, OP_RSA);
1262         return (pkey);
1263         }

1265 /* load RSA public key from a file */
1266 /* ARGSUSED */
1267 EVP_PKEY *pk11_load_pubkey(ENGINE* e, const char *pubkey_file,
1268         UI_METHOD *ui_method, void *callback_data)
1269         {
1270         EVP_PKEY *pkey = NULL;
1271         FILE *pubkey;
1272         CK_OBJECT_HANDLE  h_pub_key = CK_INVALID_HANDLE;
1273         RSA *rsa;
1274         PK11_SESSION *sp;

1276         if ((sp = pk11_get_session(OP_RSA)) == NULL)
1277                 return (NULL);

1279         if ((pubkey = fopen(pubkey_file, read_mode_flags)) != NULL)
1280                 {
1281                 pkey = PEM_read_PUBKEY(pubkey, NULL, NULL, NULL);
1282                 (void) fclose(pubkey);
1283                 if (pkey != NULL)
1284                         {
1285                         rsa = EVP_PKEY_get1_RSA(pkey);
1286                         if (rsa != NULL)
1287                                 {
1288                                 (void) check_new_rsa_key_pub(sp, rsa);

1290                                 h_pub_key = sp->opdata_rsa_pub_key =
1291                                     pk11_get_public_rsa_key(rsa,
1292                                     &sp->opdata_rsa_pub, &sp->opdata_rsa_n_num,
1293                                     &sp->opdata_rsa_e_num, sp->session);
1294                                 if (h_pub_key == CK_INVALID_HANDLE)
1295                                         {
1296                                         EVP_PKEY_free(pkey);
1297                                         pkey = NULL;
1298                                         }
1299                                 }
1300                         else
1301                                 {
1302                                 EVP_PKEY_free(pkey);
1303                                 pkey = NULL;
1304                                 }
1305                         }
1306                 }

1308         pk11_return_session(sp, OP_RSA);
1309         return (pkey);
1310         }

1312 /*
1313  * Create a public key object in a session from a given rsa structure.
1314  * The *rsa_n_num and *rsa_e_num pointers are non-NULL for RSA public keys.
1315  */
```

```
1316 static CK_OBJECT_HANDLE pk11_get_public_rsa_key(RSA* rsa,
1317     RSA** key_ptr, BIGNUM **rsa_n_num, BIGNUM **rsa_e_num,
1318     CK_SESSION_HANDLE session)
1319         {
1320         CK_RV rv;
1321         CK_OBJECT_HANDLE h_key = CK_INVALID_HANDLE;
1322         CK_ULONG found;
1323         CK_OBJECT_CLASS o_key = CKO_PUBLIC_KEY;
1324         CK_KEY_TYPE k_type = CKK_RSA;
1325         CK_ULONG ul_key_attr_count = 7;
1326         CK_BBOOL rollback = FALSE;

1328         CK_ATTRIBUTE  a_key_template[] =
1329                 {
1330                 {CKA_CLASS, (void *) NULL, sizeof (CK_OBJECT_CLASS)},
1331                 {CKA_KEY_TYPE, (void *) NULL, sizeof (CK_KEY_TYPE)},
1332                 {CKA_TOKEN, &false, sizeof (true)},
1333                 {CKA_ENCRYPT, &true, sizeof (true)},
1334                 {CKA_VERIFY_RECOVER, &true, sizeof (true)},
1335                 {CKA_MODULUS, (void *)NULL, 0},
1336                 {CKA_PUBLIC_EXPONENT, (void *)NULL, 0}
1337                 };

1339         int i;

1341         a_key_template[0].pValue = &o_key;
1342         a_key_template[1].pValue = &k_type;

1344         a_key_template[5].ulValueLen = BN_num_bytes(rsa->n);
1345         a_key_template[5].pValue = (CK_VOID_PTR)OPENSSL_malloc(
1346                 (size_t)a_key_template[5].ulValueLen);
1347         if (a_key_template[5].pValue == NULL)
1348                 {
1349                 PK11err(PK11_F_GET_PUB_RSA_KEY, PK11_R_MALLOC_FAILURE);
1350                 goto malloc_err;
1351                 }

1353         BN_bn2bin(rsa->n, a_key_template[5].pValue);

1355         a_key_template[6].ulValueLen = BN_num_bytes(rsa->e);
1356         a_key_template[6].pValue = (CK_VOID_PTR)OPENSSL_malloc(
1357                 (size_t)a_key_template[6].ulValueLen);
1358         if (a_key_template[6].pValue == NULL)
1359                 {
1360                 PK11err(PK11_F_GET_PUB_RSA_KEY, PK11_R_MALLOC_FAILURE);
1361                 goto malloc_err;
1362                 }

1364         BN_bn2bin(rsa->e, a_key_template[6].pValue);

1366         /* see find_lock array definition for more info on object locking */
1367         LOCK_OBJSTORE(OP_RSA);
1368         rv = pFuncList->C_FindObjectsInit(session, a_key_template,
1369                 ul_key_attr_count);

1371         if (rv != CKR_OK)
1372                 {
1373                 PK11err_add_data(PK11_F_GET_PUB_RSA_KEY, PK11_R_FINDOBJECTSINIT,
1374                     rv);
1375                 goto err;
1376                 }

1378         rv = pFuncList->C_FindObjects(session, &h_key, 1, &found);

1380         if (rv != CKR_OK)
1381                 {
```



```
1382                 PK11err_add_data(PK11_F_GET_PUB_RSA_KEY,
1383                     PK11_R_FINDOBJECTS, rv);
1384                 goto err;
1385                 }

1387         rv = pFuncList->C_FindObjectsFinal(session);

1389         if (rv != CKR_OK)
1390                 {
1391                 PK11err_add_data(PK11_F_GET_PUB_RSA_KEY,
1392                     PK11_R_FINDOBJECTSFINAL, rv);
1393                 goto err;
1394                 }

1396         if (found == 0)
1397                 {
1398                 rv = pFuncList->C_CreateObject(session,
1399                         a_key_template, ul_key_attr_count, &h_key);
1400                 if (rv != CKR_OK)
1401                         {
1402                         PK11err_add_data(PK11_F_GET_PUB_RSA_KEY,
1403                             PK11_R_CREATEOBJECT, rv);
1404                         goto err;
1405                         }
1406                 }

1408         if (rsa_n_num != NULL)
1409                 if ((*rsa_n_num = BN_dup(rsa->n)) == NULL)
1410                         {
1411                         PK11err(PK11_F_GET_PUB_RSA_KEY, PK11_R_MALLOC_FAILURE);
1412                         rollback = TRUE;
1413                         goto err;
1414                         }
1415         if (rsa_e_num != NULL)
1416                 if ((*rsa_e_num = BN_dup(rsa->e)) == NULL)
1417                         {
1418                         PK11err(PK11_F_GET_PUB_RSA_KEY, PK11_R_MALLOC_FAILURE);
1419                         BN_free(*rsa_n_num);
1420                         *rsa_n_num = NULL;
1421                         rollback = TRUE;
1422                         goto err;
1423                         }

1425         /* LINTED: E_CONSTANT_CONDITION */
1426         KEY_HANDLE_REFHOLD(h_key, OP_RSA, FALSE, rollback, err);
1427         if (key_ptr != NULL)
1428                 *key_ptr = rsa;

1430 err:
1431         if (rollback)
1432                 {
1433                 /*
1434                  * We do not care about the return value from C_DestroyObject()
1435                  * since we are doing rollback.
1436                  */
1437                 if (found == 0)
1438                         (void) pFuncList->C_DestroyObject(session, h_key);
1439                 h_key = CK_INVALID_HANDLE;
1440                 }

1442         UNLOCK_OBJSTORE(OP_RSA);

1444 malloc_err:
1445         for (i = 5; i <= 6; i++)
1446                 {
1447                 if (a_key_template[i].pValue != NULL)
```

```
1448                        {
1449                        OPENSSL_free(a_key_template[i].pValue);
1450                        a_key_template[i].pValue = NULL;
1451                        }
1452                }

1454        return (h_key);
1455        }

1457 /*
1458  * Create a private key object in the session from a given rsa structure.
1459  * The *rsa_d_num pointer is non-NULL for RSA private keys.
1460  */
1461 static CK_OBJECT_HANDLE pk11_get_private_rsa_key(RSA* rsa,
1462     RSA** key_ptr, BIGNUM **rsa_d_num, CK_SESSION_HANDLE session)
1463        {
1464        CK_RV rv;
1465        CK_OBJECT_HANDLE h_key = CK_INVALID_HANDLE;
1466        int i;
1467        CK_ULONG found;
1468        CK_OBJECT_CLASS o_key = CKO_PRIVATE_KEY;
1469        CK_KEY_TYPE k_type = CKK_RSA;
1470        CK_ULONG ul_key_attr_count = 14;
1471        CK_BBOOL rollback = FALSE;

1473        /* Both CKA_TOKEN and CKA_SENSITIVE have to be FALSE for session keys */
1474        CK_ATTRIBUTE  a_key_template[] =
1475                {
1476                {CKA_CLASS, (void *) NULL, sizeof (CK_OBJECT_CLASS)},
1477                {CKA_KEY_TYPE, (void *) NULL, sizeof (CK_KEY_TYPE)},
1478                {CKA_TOKEN, &false, sizeof (true)},
1479                {CKA_SENSITIVE, &false, sizeof (true)},
1480                {CKA_DECRYPT, &true, sizeof (true)},
1481                {CKA_SIGN, &true, sizeof (true)},
1482                {CKA_MODULUS, (void *)NULL, 0},
1483                {CKA_PUBLIC_EXPONENT, (void *)NULL, 0},
1484                {CKA_PRIVATE_EXPONENT, (void *)NULL, 0},
1485                {CKA_PRIME_1, (void *)NULL, 0},
1486                {CKA_PRIME_2, (void *)NULL, 0},
1487                {CKA_EXPONENT_1, (void *)NULL, 0},
1488                {CKA_EXPONENT_2, (void *)NULL, 0},
1489                {CKA_COEFFICIENT, (void *)NULL, 0}
1490                };

1492        a_key_template[0].pValue = &o_key;
1493        a_key_template[1].pValue = &k_type;

1495        /* Put the private key components into the template */
1496        if (init_template_value(rsa->n, &a_key_template[6].pValue,
1497                &a_key_template[6].ulValueLen) == 0 ||
1498            init_template_value(rsa->e, &a_key_template[7].pValue,
1499                &a_key_template[7].ulValueLen) == 0 ||
1500            init_template_value(rsa->d, &a_key_template[8].pValue,
1501                &a_key_template[8].ulValueLen) == 0 ||
1502            init_template_value(rsa->p, &a_key_template[9].pValue,
1503                &a_key_template[9].ulValueLen) == 0 ||
1504            init_template_value(rsa->q, &a_key_template[10].pValue,
1505                &a_key_template[10].ulValueLen) == 0 ||
1506            init_template_value(rsa->dmp1, &a_key_template[11].pValue,
1507                &a_key_template[11].ulValueLen) == 0 ||
1508            init_template_value(rsa->dmq1, &a_key_template[12].pValue,
1509                &a_key_template[12].ulValueLen) == 0 ||
1510            init_template_value(rsa->iqmp, &a_key_template[13].pValue,
1511                &a_key_template[13].ulValueLen) == 0)
1512                {
1513                PK11err(PK11_F_GET_PRIV_RSA_KEY, PK11_R_MALLOC_FAILURE);
```



```
1514                goto malloc_err;
1515                }

1517        /* see find_lock array definition for more info on object locking */
1518        LOCK_OBJSTORE(OP_RSA);
1519        rv = pFuncList->C_FindObjectsInit(session, a_key_template,
1520                ul_key_attr_count);

1522        if (rv != CKR_OK)
1523                {
1524                PK11err_add_data(PK11_F_GET_PRIV_RSA_KEY,
1525                    PK11_R_FINDOBJECTSINIT, rv);
1526                goto err;
1527                }

1529        rv = pFuncList->C_FindObjects(session, &h_key, 1, &found);

1531        if (rv != CKR_OK)
1532                {
1533                PK11err_add_data(PK11_F_GET_PRIV_RSA_KEY,
1534                    PK11_R_FINDOBJECTS, rv);
1535                goto err;
1536                }

1538        rv = pFuncList->C_FindObjectsFinal(session);

1540        if (rv != CKR_OK)
1541                {
1542                PK11err_add_data(PK11_F_GET_PRIV_RSA_KEY,
1543                    PK11_R_FINDOBJECTSFINAL, rv);
1544                goto err;
1545                }

1547        if (found == 0)
1548                {
1549                rv = pFuncList->C_CreateObject(session,
1550                        a_key_template, ul_key_attr_count, &h_key);
1551                if (rv != CKR_OK)
1552                        {
1553                        PK11err_add_data(PK11_F_GET_PRIV_RSA_KEY,
1554                                PK11_R_CREATEOBJECT, rv);
1555                        goto err;
1556                        }
1557                }

1559        if (rsa_d_num != NULL)
1560                if ((*rsa_d_num = BN_dup(rsa->d)) == NULL)
1561                        {
1562                        PK11err(PK11_F_GET_PRIV_RSA_KEY, PK11_R_MALLOC_FAILURE);
1563                        rollback = TRUE;
1564                        goto err;
1565                        }

1567        /* LINTED: E_CONSTANT_CONDITION */
1568        KEY_HANDLE_REFHOLD(h_key, OP_RSA, FALSE, rollback, err);
1569        if (key_ptr != NULL)
1570                *key_ptr = rsa;

1572 err:
1573        if (rollback)
1574                {
1575                /*
1576                 * We do not care about the return value from C_DestroyObject()
1577                 * since we are doing rollback.
1578                 */
1579                if (found == 0)
```

```
1580                        (void) pFuncList->C_DestroyObject(session, h_key);
1581                h_key = CK_INVALID_HANDLE;
1582                }

1584        UNLOCK_OBJSTORE(OP_RSA);

1586 malloc_err:
1587        /*
1588         * 6 to 13 entries in the key template are key components.
1589         * They need to be freed apon exit or error.
1590         */
1591        for (i = 6; i <= 13; i++)
1592                {
1593                if (a_key_template[i].pValue != NULL)
1594                        {
1595                        (void) memset(a_key_template[i].pValue, 0,
1596                                a_key_template[i].ulValueLen);
1597                        OPENSSL_free(a_key_template[i].pValue);
1598                        a_key_template[i].pValue = NULL;
1599                        }
1600                }

1602        return (h_key);
1603        }

1605 /*
1606  * Check for cache miss and clean the object pointer and handle
1607  * in such case. Return 1 for cache hit, 0 for cache miss.
1608  */
1609 static int check_new_rsa_key_pub(PK11_SESSION *sp, const RSA *rsa)
1610        {
1611        /*
1612         * Provide protection against RSA structure reuse by making the
1613         * check for cache hit stronger. Only public components of RSA
1614         * key matter here so it is sufficient to compare them with values
1615         * cached in PK11_SESSION structure.
1616         */
1617        if ((sp->opdata_rsa_pub != rsa) ||
1618            (BN_cmp(sp->opdata_rsa_n_num, rsa->n) != 0) ||
1619            (BN_cmp(sp->opdata_rsa_e_num, rsa->e) != 0))
1620                {
1621                /*
1622                 * We do not check the return value because even in case of
1623                 * failure the sp structure will have both key pointer
1624                 * and object handle cleaned and pk11_destroy_object()
1625                 * reports the failure to the OpenSSL error message buffer.
1626                 */
1627                (void) pk11_destroy_rsa_object_pub(sp, TRUE);
1628                return (0);
1629                }
1630        return (1);
1631        }

1633 /*
1634  * Check for cache miss and clean the object pointer and handle
1635  * in such case. Return 1 for cache hit, 0 for cache miss.
1636  */
1637 static int check_new_rsa_key_priv(PK11_SESSION *sp, const RSA *rsa)
1638        {
1639        /*
1640         * Provide protection against RSA structure reuse by making the
1641         * check for cache hit stronger. Comparing private exponent of RSA
1642         * key with value cached in PK11_SESSION structure should
1643         * be sufficient.
1644         */
1645        if ((sp->opdata_rsa_priv != rsa) ||
```



```
1646            (BN_cmp(sp->opdata_rsa_d_num, rsa->d) != 0))
1647                {
1648                /*
1649                 * We do not check the return value because even in case of
1650                 * failure the sp structure will have both key pointer
1651                 * and object handle cleaned and pk11_destroy_object()
1652                 * reports the failure to the OpenSSL error message buffer.
1653                 */
1654                (void) pk11_destroy_rsa_object_priv(sp, TRUE);
1655                return (0);
1656                }
1657        return (1);
1658        }
1659 #endif

1661 #ifndef OPENSSL_NO_DSA
1662 /* The DSA function implementation */
1663 /* ARGSUSED */
1664 static int pk11_DSA_init(DSA *dsa)
1665        {
1666        return (1);
1667        }

1669 /* ARGSUSED */
1670 static int pk11_DSA_finish(DSA *dsa)
1671        {
1672        return (1);
1673        }


1676 static DSA_SIG *
1677 pk11_dsa_do_sign(const unsigned char *dgst, int dlen, DSA *dsa)
1678        {
1679        BIGNUM *r = NULL, *s = NULL;
1680        int i;
1681        DSA_SIG *dsa_sig = NULL;

1683        CK_RV rv;
1684        CK_MECHANISM Mechanism_dsa = {CKM_DSA, NULL, 0};
1685        CK_MECHANISM *p_mech = &Mechanism_dsa;
1686        CK_OBJECT_HANDLE h_priv_key;

1688        /*
1689         * The signature is the concatenation of r and s,
1690         * each is 20 bytes long
1691         */
1692        unsigned char sigret[DSA_SIGNATURE_LEN];
1693        unsigned long siglen = DSA_SIGNATURE_LEN;
1694        unsigned int siglen2 = DSA_SIGNATURE_LEN / 2;

1696        PK11_SESSION *sp = NULL;

1698        if ((dsa->p == NULL) || (dsa->q == NULL) || (dsa->g == NULL))
1699                {
1700                PK11err(PK11_F_DSA_SIGN, PK11_R_MISSING_KEY_COMPONENT);
1701                goto ret;
1702                }

1704        i = BN_num_bytes(dsa->q); /* should be 20 */
1705        if (dlen > i)
1706                {
1707                PK11err(PK11_F_DSA_SIGN, PK11_R_INVALID_SIGNATURE_LENGTH);
1708                goto ret;
1709                }

1711        if ((sp = pk11_get_session(OP_DSA)) == NULL)
```

```
1712                goto ret;

1714        (void) check_new_dsa_key_priv(sp, dsa);

1716        h_priv_key = sp->opdata_dsa_priv_key;
1717        if (h_priv_key == CK_INVALID_HANDLE)
1718                h_priv_key = sp->opdata_dsa_priv_key =
1719                        pk11_get_private_dsa_key((DSA *)dsa,
1720                            &sp->opdata_dsa_priv,
1721                            &sp->opdata_dsa_priv_num, sp->session);

1723        if (h_priv_key != CK_INVALID_HANDLE)
1724                {
1725                rv = pFuncList->C_SignInit(sp->session, p_mech, h_priv_key);

1727                if (rv != CKR_OK)
1728                        {
1729                        PK11err_add_data(PK11_F_DSA_SIGN, PK11_R_SIGNINIT, rv);
1730                        goto ret;
1731                        }

1733                        (void) memset(sigret, 0, siglen);
1734                        rv = pFuncList->C_Sign(sp->session,
1735                            (unsigned char *) dgst, dlen, sigret,
1736                            (CK_ULONG_PTR) &siglen);

1738                if (rv != CKR_OK)
1739                        {
1740                        PK11err_add_data(PK11_F_DSA_SIGN, PK11_R_SIGN, rv);
1741                        goto ret;
1742                        }
1743                }


1746        if ((s = BN_new()) == NULL)
1747                {
1748                PK11err(PK11_F_DSA_SIGN, PK11_R_MALLOC_FAILURE);
1749                goto ret;
1750                }

1752        if ((r = BN_new()) == NULL)
1753                {
1754                PK11err(PK11_F_DSA_SIGN, PK11_R_MALLOC_FAILURE);
1755                goto ret;
1756                }

1758        if ((dsa_sig = DSA_SIG_new()) == NULL)
1759                {
1760                PK11err(PK11_F_DSA_SIGN, PK11_R_MALLOC_FAILURE);
1761                goto ret;
1762                }

1764        if (BN_bin2bn(sigret, siglen2, r) == NULL ||
1765            BN_bin2bn(&sigret[siglen2], siglen2, s) == NULL)
1766                {
1767                PK11err(PK11_F_DSA_SIGN, PK11_R_MALLOC_FAILURE);
1768                goto ret;
1769                }

1771        dsa_sig->r = r;
1772        dsa_sig->s = s;

1774 ret:
1775        if (dsa_sig == NULL)
1776                {
1777                if (r != NULL)
```



```
1778                        BN_free(r);
1779                if (s != NULL)
1780                        BN_free(s);
1781                }

1783        pk11_return_session(sp, OP_DSA);
1784        return (dsa_sig);
1785        }

1787 static int
1788 pk11_dsa_do_verify(const unsigned char *dgst, int dlen, DSA_SIG *sig,
1789        DSA *dsa)
1790        {
1791        int i;
1792        CK_RV rv;
1793        int retval = 0;
1794        CK_MECHANISM Mechanism_dsa = {CKM_DSA, NULL, 0};
1795        CK_MECHANISM *p_mech = &Mechanism_dsa;
1796        CK_OBJECT_HANDLE h_pub_key;

1798        unsigned char sigbuf[DSA_SIGNATURE_LEN];
1799        unsigned long siglen = DSA_SIGNATURE_LEN;
1800        unsigned long siglen2 = DSA_SIGNATURE_LEN/2;

1802        PK11_SESSION *sp = NULL;

1804        if (BN_is_zero(sig->r) || sig->r->neg || BN_ucmp(sig->r, dsa->q) >= 0)
1805                {
1806                PK11err(PK11_F_DSA_VERIFY,
1807                        PK11_R_INVALID_DSA_SIGNATURE_R);
1808                goto ret;
1809                }

1811        if (BN_is_zero(sig->s) || sig->s->neg || BN_ucmp(sig->s, dsa->q) >= 0)
1812                {
1813                PK11err(PK11_F_DSA_VERIFY,
1814                        PK11_R_INVALID_DSA_SIGNATURE_S);
1815                goto ret;
1816                }

1818        i = BN_num_bytes(dsa->q); /* should be 20 */

1820        if (dlen > i)
1821                {
1822                PK11err(PK11_F_DSA_VERIFY,
1823                        PK11_R_INVALID_SIGNATURE_LENGTH);
1824                goto ret;
1825                }

1827        if ((sp = pk11_get_session(OP_DSA)) == NULL)
1828                goto ret;

1830        (void) check_new_dsa_key_pub(sp, dsa);

1832        h_pub_key = sp->opdata_dsa_pub_key;
1833        if (h_pub_key == CK_INVALID_HANDLE)
1834                h_pub_key = sp->opdata_dsa_pub_key =
1835                        pk11_get_public_dsa_key((DSA *)dsa, &sp->opdata_dsa_pub,
1836                            &sp->opdata_dsa_pub_num, sp->session);

1838        if (h_pub_key != CK_INVALID_HANDLE)
1839                {
1840                rv = pFuncList->C_VerifyInit(sp->session, p_mech,
1841                        h_pub_key);

1843                if (rv != CKR_OK)
```

```
1844                        {
1845                        PK11err_add_data(PK11_F_DSA_VERIFY, PK11_R_VERIFYINIT,
1846                            rv);
1847                        goto ret;
1848                        }

1850                /*
1851                 * The representation of each of the two big numbers could
1852                 * be shorter than DSA_SIGNATURE_LEN/2 bytes so we need
1853                 * to act accordingly and shift if necessary.
1854                 */
1855                (void) memset(sigbuf, 0, siglen);
1856                BN_bn2bin(sig->r, sigbuf + siglen2 - BN_num_bytes(sig->r));
1857                BN_bn2bin(sig->s, &sigbuf[siglen2] + siglen2 -
1858                    BN_num_bytes(sig->s));

1860                rv = pFuncList->C_Verify(sp->session,
1861                        (unsigned char *) dgst, dlen, sigbuf, (CK_ULONG)siglen);

1863                if (rv != CKR_OK)
1864                        {
1865                        PK11err_add_data(PK11_F_DSA_VERIFY, PK11_R_VERIFY, rv);
1866                        goto ret;
1867                        }
1868                }

1870        retval = 1;
1871 ret:

1873        pk11_return_session(sp, OP_DSA);
1874        return (retval);
1875        }


1878 /*
1879  * Create a public key object in a session from a given dsa structure.
1880  * The *dsa_pub_num pointer is non-NULL for DSA public keys.
1881  */
1882 static CK_OBJECT_HANDLE pk11_get_public_dsa_key(DSA* dsa,
1883     DSA **key_ptr, BIGNUM **dsa_pub_num, CK_SESSION_HANDLE session)
1884        {
1885        CK_RV rv;
1886        CK_OBJECT_CLASS o_key = CKO_PUBLIC_KEY;
1887        CK_OBJECT_HANDLE h_key = CK_INVALID_HANDLE;
1888        CK_ULONG found;
1889        CK_KEY_TYPE k_type = CKK_DSA;
1890        CK_ULONG ul_key_attr_count = 8;
1891        CK_BBOOL rollback = FALSE;
1892        int i;

1894        CK_ATTRIBUTE  a_key_template[] =
1895                {
1896                {CKA_CLASS, (void *) NULL, sizeof (CK_OBJECT_CLASS)},
1897                {CKA_KEY_TYPE, (void *) NULL, sizeof (CK_KEY_TYPE)},
1898                {CKA_TOKEN, &false, sizeof (true)},
1899                {CKA_VERIFY, &true, sizeof (true)},
1900                {CKA_PRIME, (void *)NULL, 0},           /* p */
1901                {CKA_SUBPRIME, (void *)NULL, 0},        /* q */
1902                {CKA_BASE, (void *)NULL, 0},            /* g */
1903                {CKA_VALUE, (void *)NULL, 0}            /* pub_key - y */
1904                };

1906        a_key_template[0].pValue = &o_key;
1907        a_key_template[1].pValue = &k_type;

1909        if (init_template_value(dsa->p, &a_key_template[4].pValue,
```



```
1910                &a_key_template[4].ulValueLen) == 0 ||
1911            init_template_value(dsa->q, &a_key_template[5].pValue,
1912                &a_key_template[5].ulValueLen) == 0 ||
1913            init_template_value(dsa->g, &a_key_template[6].pValue,
1914                &a_key_template[6].ulValueLen) == 0 ||
1915            init_template_value(dsa->pub_key, &a_key_template[7].pValue,
1916                &a_key_template[7].ulValueLen) == 0)
1917                {
1918                PK11err(PK11_F_GET_PUB_DSA_KEY, PK11_R_MALLOC_FAILURE);
1919                goto malloc_err;
1920                }

1922        /* see find_lock array definition for more info on object locking */
1923        LOCK_OBJSTORE(OP_DSA);
1924        rv = pFuncList->C_FindObjectsInit(session, a_key_template,
1925                ul_key_attr_count);

1927        if (rv != CKR_OK)
1928                {
1929                PK11err_add_data(PK11_F_GET_PUB_DSA_KEY, PK11_R_FINDOBJECTSINIT,
1930                    rv);
1931                goto err;
1932                }

1934        rv = pFuncList->C_FindObjects(session, &h_key, 1, &found);

1936        if (rv != CKR_OK)
1937                {
1938                PK11err_add_data(PK11_F_GET_PUB_DSA_KEY,
1939                    PK11_R_FINDOBJECTS, rv);
1940                goto err;
1941                }

1943        rv = pFuncList->C_FindObjectsFinal(session);

1945        if (rv != CKR_OK)
1946                {
1947                PK11err_add_data(PK11_F_GET_PUB_DSA_KEY,
1948                    PK11_R_FINDOBJECTSFINAL, rv);
1949                goto err;
1950                }

1952        if (found == 0)
1953                {
1954                rv = pFuncList->C_CreateObject(session,
1955                        a_key_template, ul_key_attr_count, &h_key);
1956                if (rv != CKR_OK)
1957                        {
1958                        PK11err_add_data(PK11_F_GET_PUB_DSA_KEY,
1959                            PK11_R_CREATEOBJECT, rv);
1960                        goto err;
1961                        }
1962                }

1964        if (dsa_pub_num != NULL)
1965                if ((*dsa_pub_num = BN_dup(dsa->pub_key)) == NULL)
1966                        {
1967                        PK11err(PK11_F_GET_PUB_DSA_KEY, PK11_R_MALLOC_FAILURE);
1968                        rollback = TRUE;
1969                        goto err;
1970                        }

1972        /* LINTED: E_CONSTANT_CONDITION */
1973        KEY_HANDLE_REFHOLD(h_key, OP_DSA, FALSE, rollback, err);
1974        if (key_ptr != NULL)
1975                *key_ptr = dsa;
```

```
1977 err:
1978         if (rollback)
1979                 {
1980                 /*
1981                  * We do not care about the return value from C_DestroyObject()
1982                  * since we are doing rollback.
1983                  */
1984                 if (found == 0)
1985                         (void) pFuncList->C_DestroyObject(session, h_key);
1986                 h_key = CK_INVALID_HANDLE;
1987                 }

1989         UNLOCK_OBJSTORE(OP_DSA);

1991 malloc_err:
1992         for (i = 4; i <= 7; i++)
1993                 {
1994                 if (a_key_template[i].pValue != NULL)
1995                         {
1996                         OPENSSL_free(a_key_template[i].pValue);
1997                         a_key_template[i].pValue = NULL;
1998                         }
1999                 }

2001         return (h_key);
2002         }

2004 /*
2005  * Create a private key object in the session from a given dsa structure
2006  * The *dsa_priv_num pointer is non-NULL for DSA private keys.
2007  */
2008 static CK_OBJECT_HANDLE pk11_get_private_dsa_key(DSA* dsa,
2009     DSA **key_ptr, BIGNUM **dsa_priv_num, CK_SESSION_HANDLE session)
2010         {
2011         CK_RV rv;
2012         CK_OBJECT_HANDLE h_key = CK_INVALID_HANDLE;
2013         CK_OBJECT_CLASS o_key = CKO_PRIVATE_KEY;
2014         int i;
2015         CK_ULONG found;
2016         CK_KEY_TYPE k_type = CKK_DSA;
2017         CK_ULONG ul_key_attr_count = 9;
2018         CK_BBOOL rollback = FALSE;

2020         /* Both CKA_TOKEN and CKA_SENSITIVE have to be FALSE for session keys */
2021         CK_ATTRIBUTE  a_key_template[] =
2022                 {
2023                 {CKA_CLASS, (void *) NULL, sizeof (CK_OBJECT_CLASS)},
2024                 {CKA_KEY_TYPE, (void *) NULL, sizeof (CK_KEY_TYPE)},
2025                 {CKA_TOKEN, &false, sizeof (true)},
2026                 {CKA_SENSITIVE, &false, sizeof (true)},
2027                 {CKA_SIGN, &true, sizeof (true)},
2028                 {CKA_PRIME, (void *)NULL, 0},           /* p */
2029                 {CKA_SUBPRIME, (void *)NULL, 0},        /* q */
2030                 {CKA_BASE, (void *)NULL, 0},            /* g */
2031                 {CKA_VALUE, (void *)NULL, 0}            /* priv_key - x */
2032                 };

2034         a_key_template[0].pValue = &o_key;
2035         a_key_template[1].pValue = &k_type;

2037         /* Put the private key components into the template */
2038         if (init_template_value(dsa->p, &a_key_template[5].pValue,
2039                 &a_key_template[5].ulValueLen) == 0 ||
2040             init_template_value(dsa->q, &a_key_template[6].pValue,
2041                 &a_key_template[6].ulValueLen) == 0 ||
```



```
2042             init_template_value(dsa->g, &a_key_template[7].pValue,
2043                 &a_key_template[7].ulValueLen) == 0 ||
2044             init_template_value(dsa->priv_key, &a_key_template[8].pValue,
2045                 &a_key_template[8].ulValueLen) == 0)
2046                 {
2047                 PK11err(PK11_F_GET_PRIV_DSA_KEY, PK11_R_MALLOC_FAILURE);
2048                 goto malloc_err;
2049                 }

2051         /* see find_lock array definition for more info on object locking */
2052         LOCK_OBJSTORE(OP_DSA);
2053         rv = pFuncList->C_FindObjectsInit(session, a_key_template,
2054                 ul_key_attr_count);

2056         if (rv != CKR_OK)
2057                 {
2058                 PK11err_add_data(PK11_F_GET_PRIV_DSA_KEY,
2059                     PK11_R_FINDOBJECTSINIT, rv);
2060                 goto err;
2061                 }

2063         rv = pFuncList->C_FindObjects(session, &h_key, 1, &found);

2065         if (rv != CKR_OK)
2066                 {
2067                 PK11err_add_data(PK11_F_GET_PRIV_DSA_KEY,
2068                     PK11_R_FINDOBJECTS, rv);
2069                 goto err;
2070                 }

2072         rv = pFuncList->C_FindObjectsFinal(session);

2074         if (rv != CKR_OK)
2075                 {
2076                 PK11err_add_data(PK11_F_GET_PRIV_DSA_KEY,
2077                     PK11_R_FINDOBJECTSFINAL, rv);
2078                 goto err;
2079                 }

2081         if (found == 0)
2082                 {
2083                 rv = pFuncList->C_CreateObject(session,
2084                         a_key_template, ul_key_attr_count, &h_key);
2085                 if (rv != CKR_OK)
2086                         {
2087                         PK11err_add_data(PK11_F_GET_PRIV_DSA_KEY,
2088                             PK11_R_CREATEOBJECT, rv);
2089                         goto err;
2090                         }
2091                 }

2093         if (dsa_priv_num != NULL)
2094                 if ((*dsa_priv_num = BN_dup(dsa->priv_key)) == NULL)
2095                         {
2096                         PK11err(PK11_F_GET_PRIV_DSA_KEY, PK11_R_MALLOC_FAILURE);
2097                         rollback = TRUE;
2098                         goto err;
2099                         }

2101         /* LINTED: E_CONSTANT_CONDITION */
2102         KEY_HANDLE_REFHOLD(h_key, OP_DSA, FALSE, rollback, err);
2103         if (key_ptr != NULL)
2104                 *key_ptr = dsa;

2106 err:
2107         if (rollback)
```

```
2108                {
2109                /*
2110                 * We do not care about the return value from C_DestroyObject()
2111                 * since we are doing rollback.
2112                 */
2113                if (found == 0)
2114                        (void) pFuncList->C_DestroyObject(session, h_key);
2115                h_key = CK_INVALID_HANDLE;
2116                }

2118        UNLOCK_OBJSTORE(OP_DSA);

2120 malloc_err:
2121        /*
2122         * 5 to 8 entries in the key template are key components.
2123         * They need to be freed apon exit or error.
2124         */
2125        for (i = 5; i <= 8; i++)
2126                {
2127                if (a_key_template[i].pValue != NULL)
2128                        {
2129                        (void) memset(a_key_template[i].pValue, 0,
2130                                a_key_template[i].ulValueLen);
2131                        OPENSSL_free(a_key_template[i].pValue);
2132                        a_key_template[i].pValue = NULL;
2133                        }
2134                }

2136        return (h_key);
2137        }

2139 /*
2140  * Check for cache miss and clean the object pointer and handle
2141  * in such case. Return 1 for cache hit, 0 for cache miss.
2142  */
2143 static int check_new_dsa_key_pub(PK11_SESSION *sp, DSA *dsa)
2144        {
2145        /*
2146         * Provide protection against DSA structure reuse by making the
2147         * check for cache hit stronger. Only public key component of DSA
2148         * key matters here so it is sufficient to compare it with value
2149         * cached in PK11_SESSION structure.
2150         */
2151        if ((sp->opdata_dsa_pub != dsa) ||
2152            (BN_cmp(sp->opdata_dsa_pub_num, dsa->pub_key) != 0))
2153                {
2154                /*
2155                 * We do not check the return value because even in case of
2156                 * failure the sp structure will have both key pointer
2157                 * and object handle cleaned and pk11_destroy_object()
2158                 * reports the failure to the OpenSSL error message buffer.
2159                 */
2160                (void) pk11_destroy_dsa_object_pub(sp, TRUE);
2161                return (0);
2162                }
2163        return (1);
2164        }

2166 /*
2167  * Check for cache miss and clean the object pointer and handle
2168  * in such case. Return 1 for cache hit, 0 for cache miss.
2169  */
2170 static int check_new_dsa_key_priv(PK11_SESSION *sp, DSA *dsa)
2171        {
2172        /*
2173         * Provide protection against DSA structure reuse by making the
```



```
2174         * check for cache hit stronger. Only private key component of DSA
2175         * key matters here so it is sufficient to compare it with value
2176         * cached in PK11_SESSION structure.
2177         */
2178        if ((sp->opdata_dsa_priv != dsa) ||
2179            (BN_cmp(sp->opdata_dsa_priv_num, dsa->priv_key) != 0))
2180                {
2181                /*
2182                 * We do not check the return value because even in case of
2183                 * failure the sp structure will have both key pointer
2184                 * and object handle cleaned and pk11_destroy_object()
2185                 * reports the failure to the OpenSSL error message buffer.
2186                 */
2187                (void) pk11_destroy_dsa_object_priv(sp, TRUE);
2188                return (0);
2189                }
2190        return (1);
2191        }
2192 #endif


2195 #ifndef OPENSSL_NO_DH
2196 /* The DH function implementation */
2197 /* ARGSUSED */
2198 static int pk11_DH_init(DH *dh)
2199        {
2200        return (1);
2201        }

2203 /* ARGSUSED */
2204 static int pk11_DH_finish(DH *dh)
2205        {
2206        return (1);
2207        }

2209 /*
2210  * Generate DH key-pair.
2211  *
2212  * Warning: Unlike OpenSSL's DH_generate_key(3) we ignore dh->priv_key
2213  * and override it even if it is set. OpenSSL does not touch dh->priv_key
2214  * if set and just computes dh->pub_key. It looks like PKCS#11 standard
2215  * is not capable of providing this functionality. This could be a problem
2216  * for applications relying on OpenSSL's semantics.
2217  */
2218 static int pk11_DH_generate_key(DH *dh)
2219        {
2220        CK_ULONG i;
2221        CK_RV rv, rv1;
2222        int reuse_mem_len = 0, ret = 0;
2223        PK11_SESSION *sp = NULL;
2224        CK_BYTE_PTR reuse_mem;

2226        CK_MECHANISM mechanism = {CKM_DH_PKCS_KEY_PAIR_GEN, NULL_PTR, 0};
2227        CK_OBJECT_HANDLE h_pub_key = CK_INVALID_HANDLE;
2228        CK_OBJECT_HANDLE h_priv_key = CK_INVALID_HANDLE;

2230        CK_ULONG ul_pub_key_attr_count = 3;
2231        CK_ATTRIBUTE pub_key_template[] =
2232                {
2233                {CKA_PRIVATE, &false, sizeof (false)},
2234                {CKA_PRIME, (void *)NULL, 0},
2235                {CKA_BASE, (void *)NULL, 0}
2236                };

2238        CK_ULONG ul_priv_key_attr_count = 3;
2239        CK_ATTRIBUTE priv_key_template[] =
```

```
2240                {
2241                {CKA_PRIVATE, &false, sizeof (false)},
2242                {CKA_SENSITIVE, &false, sizeof (false)},
2243                {CKA_DERIVE, &true, sizeof (true)}
2244                };

2246        CK_ULONG pub_key_attr_result_count = 1;
2247        CK_ATTRIBUTE pub_key_result[] =
2248                {
2249                {CKA_VALUE, (void *)NULL, 0}
2250                };

2252        CK_ULONG priv_key_attr_result_count = 1;
2253        CK_ATTRIBUTE priv_key_result[] =
2254                {
2255                {CKA_VALUE, (void *)NULL, 0}
2256                };

2258        pub_key_template[1].ulValueLen = BN_num_bytes(dh->p);
2259        if (pub_key_template[1].ulValueLen > 0)
2260                {
2261                /*
2262                 * We must not increase ulValueLen by DH_BUF_RESERVE since that
2263                 * could cause the same rounding problem. See definition of
2264                 * DH_BUF_RESERVE above.
2265                 */
2266                pub_key_template[1].pValue =
2267                        OPENSSL_malloc(pub_key_template[1].ulValueLen +
2268                        DH_BUF_RESERVE);
2269                if (pub_key_template[1].pValue == NULL)
2270                        {
2271                        PK11err(PK11_F_DH_GEN_KEY, PK11_R_MALLOC_FAILURE);
2272                        goto err;
2273                        }

2275                i = BN_bn2bin(dh->p, pub_key_template[1].pValue);
2276                }
2277        else
2278                goto err;

2280        pub_key_template[2].ulValueLen = BN_num_bytes(dh->g);
2281        if (pub_key_template[2].ulValueLen > 0)
2282                {
2283                pub_key_template[2].pValue =
2284                        OPENSSL_malloc(pub_key_template[2].ulValueLen +
2285                        DH_BUF_RESERVE);
2286                if (pub_key_template[2].pValue == NULL)
2287                        {
2288                        PK11err(PK11_F_DH_GEN_KEY, PK11_R_MALLOC_FAILURE);
2289                        goto err;
2290                        }

2292                i = BN_bn2bin(dh->g, pub_key_template[2].pValue);
2293                }
2294        else
2295                goto err;

2297        /*
2298         * Note: we are only using PK11_SESSION structure for getting
2299         *       a session handle. The objects created in this function are
2300         *       destroyed before return and thus not cached.
2301         */
2302        if ((sp = pk11_get_session(OP_DH)) == NULL)
2303                goto err;

2305        rv = pFuncList->C_GenerateKeyPair(sp->session,
```



```
2306                &mechanism,
2307                pub_key_template,
2308                ul_pub_key_attr_count,
2309                priv_key_template,
2310                ul_priv_key_attr_count,
2311                &h_pub_key,
2312                &h_priv_key);
2313        if (rv != CKR_OK)
2314                {
2315                PK11err_add_data(PK11_F_DH_GEN_KEY, PK11_R_GEN_KEY, rv);
2316                goto err;
2317                }

2319        /*
2320         * Reuse the larger memory allocated. We know the larger memory
2321         * should be sufficient for reuse.
2322         */
2323        if (pub_key_template[1].ulValueLen > pub_key_template[2].ulValueLen)
2324                {
2325                reuse_mem = pub_key_template[1].pValue;
2326                reuse_mem_len = pub_key_template[1].ulValueLen + DH_BUF_RESERVE;
2327                }
2328        else
2329                {
2330                reuse_mem = pub_key_template[2].pValue;
2331                reuse_mem_len = pub_key_template[2].ulValueLen + DH_BUF_RESERVE;
2332                }

2334        rv = pFuncList->C_GetAttributeValue(sp->session, h_pub_key,
2335                pub_key_result, pub_key_attr_result_count);
2336        rv1 = pFuncList->C_GetAttributeValue(sp->session, h_priv_key,
2337                priv_key_result, priv_key_attr_result_count);

2339        if (rv != CKR_OK || rv1 != CKR_OK)
2340                {
2341                rv = (rv != CKR_OK) ? rv : rv1;
2342                PK11err_add_data(PK11_F_DH_GEN_KEY,
2343                    PK11_R_GETATTRIBUTVALUE, rv);
2344                goto err;
2345                }

2347        if (((CK_LONG) pub_key_result[0].ulValueLen) <= 0 ||
2348                ((CK_LONG) priv_key_result[0].ulValueLen) <= 0)
2349                {
2350                PK11err(PK11_F_DH_GEN_KEY, PK11_R_GETATTRIBUTVALUE);
2351                goto err;
2352                }

2354        /* Reuse the memory allocated */
2355        pub_key_result[0].pValue = reuse_mem;
2356        pub_key_result[0].ulValueLen = reuse_mem_len;

2358        rv = pFuncList->C_GetAttributeValue(sp->session, h_pub_key,
2359                pub_key_result, pub_key_attr_result_count);

2361        if (rv != CKR_OK)
2362                {
2363                PK11err_add_data(PK11_F_DH_GEN_KEY,
2364                    PK11_R_GETATTRIBUTVALUE, rv);
2365                goto err;
2366                }

2368        if (pub_key_result[0].type == CKA_VALUE)
2369                {
2370                if (dh->pub_key == NULL)
2371                        if ((dh->pub_key = BN_new()) == NULL)
```

```
2372                                {
2373                                PK11err(PK11_F_DH_GEN_KEY,
2374                                        PK11_R_MALLOC_FAILURE);
2375                                goto err;
2376                                }
2377                dh->pub_key = BN_bin2bn(pub_key_result[0].pValue,
2378                        pub_key_result[0].ulValueLen, dh->pub_key);
2379                if (dh->pub_key == NULL)
2380                        {
2381                        PK11err(PK11_F_DH_GEN_KEY, PK11_R_MALLOC_FAILURE);
2382                        goto err;
2383                        }
2384                }

2386        /* Reuse the memory allocated */
2387        priv_key_result[0].pValue = reuse_mem;
2388        priv_key_result[0].ulValueLen = reuse_mem_len;

2390        rv = pFuncList->C_GetAttributeValue(sp->session, h_priv_key,
2391                priv_key_result, priv_key_attr_result_count);

2393        if (rv != CKR_OK)
2394                {
2395                PK11err_add_data(PK11_F_DH_GEN_KEY,
2396                    PK11_R_GETATTRIBUTVALUE, rv);
2397                goto err;
2398                }

2400        if (priv_key_result[0].type == CKA_VALUE)
2401                {
2402                if (dh->priv_key == NULL)
2403                        if ((dh->priv_key = BN_new()) == NULL)
2404                                {
2405                                PK11err(PK11_F_DH_GEN_KEY,
2406                                        PK11_R_MALLOC_FAILURE);
2407                                goto err;
2408                                }
2409                dh->priv_key = BN_bin2bn(priv_key_result[0].pValue,
2410                        priv_key_result[0].ulValueLen, dh->priv_key);
2411                if (dh->priv_key == NULL)
2412                        {
2413                        PK11err(PK11_F_DH_GEN_KEY, PK11_R_MALLOC_FAILURE);
2414                        goto err;
2415                        }
2416                }

2418        ret = 1;

2420 err:

2422        if (h_pub_key != CK_INVALID_HANDLE)
2423                {
2424                rv = pFuncList->C_DestroyObject(sp->session, h_pub_key);
2425                if (rv != CKR_OK)
2426                        {
2427                        PK11err_add_data(PK11_F_DH_GEN_KEY,
2428                                PK11_R_DESTROYOBJECT, rv);
2429                        }
2430                }

2432        if (h_priv_key != CK_INVALID_HANDLE)
2433                {
2434                rv = pFuncList->C_DestroyObject(sp->session, h_priv_key);
2435                if (rv != CKR_OK)
2436                        {
2437                        PK11err_add_data(PK11_F_DH_GEN_KEY,
```



```
2438                                PK11_R_DESTROYOBJECT, rv);
2439                        }
2440                }

2442        for (i = 1; i <= 2; i++)
2443                {
2444                if (pub_key_template[i].pValue != NULL)
2445                        {
2446                        OPENSSL_free(pub_key_template[i].pValue);
2447                        pub_key_template[i].pValue = NULL;
2448                        }
2449                }

2451        pk11_return_session(sp, OP_DH);
2452        return (ret);
2453        }

2455 static int pk11_DH_compute_key(unsigned char *key, const BIGNUM *pub_key,
2456        DH *dh)
2457        {
2458        int i;
2459        CK_MECHANISM mechanism = {CKM_DH_PKCS_DERIVE, NULL_PTR, 0};
2460        CK_OBJECT_CLASS key_class = CKO_SECRET_KEY;
2461        CK_KEY_TYPE key_type = CKK_GENERIC_SECRET;
2462        CK_OBJECT_HANDLE h_derived_key = CK_INVALID_HANDLE;
2463        CK_OBJECT_HANDLE h_key = CK_INVALID_HANDLE;

2465        CK_ULONG ul_priv_key_attr_count = 2;
2466        CK_ATTRIBUTE priv_key_template[] =
2467                {
2468                {CKA_CLASS, (void*) NULL, sizeof (key_class)},
2469                {CKA_KEY_TYPE, (void*) NULL, sizeof (key_type)},
2470                };

2472        CK_ULONG priv_key_attr_result_count = 1;
2473        CK_ATTRIBUTE priv_key_result[] =
2474                {
2475                {CKA_VALUE, (void *)NULL, 0}
2476                };

2478        CK_RV rv;
2479        int ret = -1;
2480        PK11_SESSION *sp = NULL;

2482        if (dh->priv_key == NULL)
2483                goto err;

2485        priv_key_template[0].pValue = &key_class;
2486        priv_key_template[1].pValue = &key_type;

2488        if ((sp = pk11_get_session(OP_DH)) == NULL)
2489                goto err;

2491        mechanism.ulParameterLen = BN_num_bytes(pub_key);
2492        mechanism.pParameter = OPENSSL_malloc(mechanism.ulParameterLen);
2493        if (mechanism.pParameter == NULL)
2494                {
2495                PK11err(PK11_F_DH_COMP_KEY, PK11_R_MALLOC_FAILURE);
2496                goto err;
2497                }
2498        BN_bn2bin(pub_key, mechanism.pParameter);

2500        (void) check_new_dh_key(sp, dh);

2502        h_key = sp->opdata_dh_key;
2503        if (h_key == CK_INVALID_HANDLE)
```

```
2504                 h_key = sp->opdata_dh_key =
2505                         pk11_get_dh_key((DH*) dh, &sp->opdata_dh,
2506                             &sp->opdata_dh_priv_num, sp->session);

2508         if (h_key == CK_INVALID_HANDLE)
2509                 {
2510                 PK11err(PK11_F_DH_COMP_KEY, PK11_R_CREATEOBJECT);
2511                 goto err;
2512                 }

2514         rv = pFuncList->C_DeriveKey(sp->session,
2515             &mechanism,
2516             h_key,
2517             priv_key_template,
2518             ul_priv_key_attr_count,
2519             &h_derived_key);
2520         if (rv != CKR_OK)
2521                 {
2522                 PK11err_add_data(PK11_F_DH_COMP_KEY, PK11_R_DERIVEKEY, rv);
2523                 goto err;
2524                 }

2526         rv = pFuncList->C_GetAttributeValue(sp->session, h_derived_key,
2527             priv_key_result, priv_key_attr_result_count);

2529         if (rv != CKR_OK)
2530                 {
2531                 PK11err_add_data(PK11_F_DH_COMP_KEY, PK11_R_GETATTRIBUTVALUE,
2532                     rv);
2533                 goto err;
2534                 }

2536         if (((CK_LONG) priv_key_result[0].ulValueLen) <= 0)
2537                 {
2538                 PK11err(PK11_F_DH_COMP_KEY, PK11_R_GETATTRIBUTVALUE);
2539                 goto err;
2540                 }
2541         priv_key_result[0].pValue =
2542                 OPENSSL_malloc(priv_key_result[0].ulValueLen);
2543         if (!priv_key_result[0].pValue)
2544                 {
2545                 PK11err(PK11_F_DH_COMP_KEY, PK11_R_MALLOC_FAILURE);
2546                 goto err;
2547                 }

2549         rv = pFuncList->C_GetAttributeValue(sp->session, h_derived_key,
2550                 priv_key_result, priv_key_attr_result_count);

2552         if (rv != CKR_OK)
2553                 {
2554                 PK11err_add_data(PK11_F_DH_COMP_KEY, PK11_R_GETATTRIBUTVALUE,
2555                     rv);
2556                 goto err;
2557                 }

2559         /*
2560          * OpenSSL allocates the output buffer 'key' which is the same
2561          * length of the public key. It is long enough for the derived key
2562          */
2563         if (priv_key_result[0].type == CKA_VALUE)
2564                 {
2565                 /*
2566                  * CKM_DH_PKCS_DERIVE mechanism is not supposed to strip
2567                  * leading zeros from a computed shared secret. However,
2568                  * OpenSSL always did it so we must do the same here. The
2569                  * vagueness of the spec regarding leading zero bytes was
```



```
2570                  * finally cleared with TLS 1.1 (RFC 4346) saying that leading
2571                  * zeros are stripped before the computed data is used as the
2572                  * pre-master secret.
2573                  */
2574                 for (i = 0; i < priv_key_result[0].ulValueLen; ++i)
2575                         {
2576                         if (((char *)priv_key_result[0].pValue)[i] != 0)
2577                                 break;
2578                         }

2580                 (void) memcpy(key, ((char *)priv_key_result[0].pValue) + i,
2581                         priv_key_result[0].ulValueLen - i);
2582                 ret = priv_key_result[0].ulValueLen - i;
2583                 }

2585 err:

2587         if (h_derived_key != CK_INVALID_HANDLE)
2588                 {
2589                 rv = pFuncList->C_DestroyObject(sp->session, h_derived_key);
2590                 if (rv != CKR_OK)
2591                         {
2592                         PK11err_add_data(PK11_F_DH_COMP_KEY,
2593                             PK11_R_DESTROYOBJECT, rv);
2594                         }
2595                 }
2596         if (priv_key_result[0].pValue)
2597                 {
2598                 OPENSSL_free(priv_key_result[0].pValue);
2599                 priv_key_result[0].pValue = NULL;
2600                 }

2602         if (mechanism.pParameter)
2603                 {
2604                 OPENSSL_free(mechanism.pParameter);
2605                 mechanism.pParameter = NULL;
2606                 }

2608         pk11_return_session(sp, OP_DH);
2609         return (ret);
2610         }


2613 static CK_OBJECT_HANDLE pk11_get_dh_key(DH* dh,
2614         DH **key_ptr, BIGNUM **dh_priv_num, CK_SESSION_HANDLE session)
2615         {
2616         CK_RV rv;
2617         CK_OBJECT_HANDLE h_key = CK_INVALID_HANDLE;
2618         CK_OBJECT_CLASS class = CKO_PRIVATE_KEY;
2619         CK_KEY_TYPE key_type = CKK_DH;
2620         CK_ULONG found;
2621         CK_BBOOL rollback = FALSE;
2622         int i;

2624         CK_ULONG ul_key_attr_count = 7;
2625         CK_ATTRIBUTE key_template[] =
2626                 {
2627                 {CKA_CLASS, (void*) NULL, sizeof (class)},
2628                 {CKA_KEY_TYPE, (void*) NULL, sizeof (key_type)},
2629                 {CKA_DERIVE, &true, sizeof (true)},
2630                 {CKA_PRIVATE, &false, sizeof (false)},
2631                 {CKA_PRIME, (void *) NULL, 0},
2632                 {CKA_BASE, (void *) NULL, 0},
2633                 {CKA_VALUE, (void *) NULL, 0},
2634                 };
```

```
2636         key_template[0].pValue = &class;
2637         key_template[1].pValue = &key_type;

2639         key_template[4].ulValueLen = BN_num_bytes(dh->p);
2640         key_template[4].pValue = (CK_VOID_PTR)OPENSSL_malloc(
2641                 (size_t)key_template[4].ulValueLen);
2642         if (key_template[4].pValue == NULL)
2643                 {
2644                 PK11err(PK11_F_GET_DH_KEY, PK11_R_MALLOC_FAILURE);
2645                 goto malloc_err;
2646                 }

2648         BN_bn2bin(dh->p, key_template[4].pValue);

2650         key_template[5].ulValueLen = BN_num_bytes(dh->g);
2651         key_template[5].pValue = (CK_VOID_PTR)OPENSSL_malloc(
2652                 (size_t)key_template[5].ulValueLen);
2653         if (key_template[5].pValue == NULL)
2654                 {
2655                 PK11err(PK11_F_GET_DH_KEY, PK11_R_MALLOC_FAILURE);
2656                 goto malloc_err;
2657                 }

2659         BN_bn2bin(dh->g, key_template[5].pValue);

2661         key_template[6].ulValueLen = BN_num_bytes(dh->priv_key);
2662         key_template[6].pValue = (CK_VOID_PTR)OPENSSL_malloc(
2663                 (size_t)key_template[6].ulValueLen);
2664         if (key_template[6].pValue == NULL)
2665                 {
2666                 PK11err(PK11_F_GET_DH_KEY, PK11_R_MALLOC_FAILURE);
2667                 goto malloc_err;
2668                 }

2670         BN_bn2bin(dh->priv_key, key_template[6].pValue);

2672         /* see find_lock array definition for more info on object locking */
2673         LOCK_OBJSTORE(OP_DH);
2674         rv = pFuncList->C_FindObjectsInit(session, key_template,
2675                 ul_key_attr_count);

2677         if (rv != CKR_OK)
2678                 {
2679                 PK11err_add_data(PK11_F_GET_DH_KEY, PK11_R_FINDOBJECTSINIT, rv);
2680                 goto err;
2681                 }

2683         rv = pFuncList->C_FindObjects(session, &h_key, 1, &found);

2685         if (rv != CKR_OK)
2686                 {
2687                 PK11err_add_data(PK11_F_GET_DH_KEY, PK11_R_FINDOBJECTS, rv);
2688                 goto err;
2689                 }

2691         rv = pFuncList->C_FindObjectsFinal(session);

2693         if (rv != CKR_OK)
2694                 {
2695                 PK11err_add_data(PK11_F_GET_DH_KEY, PK11_R_FINDOBJECTSFINAL,
2696                     rv);
2697                 goto err;
2698                 }

2700         if (found == 0)
2701                 {
```



```
2702                 rv = pFuncList->C_CreateObject(session,
2703                         key_template, ul_key_attr_count, &h_key);
2704                 if (rv != CKR_OK)
2705                         {
2706                         PK11err_add_data(PK11_F_GET_DH_KEY, PK11_R_CREATEOBJECT,
2707                             rv);
2708                         goto err;
2709                         }
2710                 }

2712         if (dh_priv_num != NULL)
2713                 if ((*dh_priv_num = BN_dup(dh->priv_key)) == NULL)
2714                         {
2715                         PK11err(PK11_F_GET_DH_KEY, PK11_R_MALLOC_FAILURE);
2716                         rollback = TRUE;
2717                         goto err;
2718                         }

2720         /* LINTED: E_CONSTANT_CONDITION */
2721         KEY_HANDLE_REFHOLD(h_key, OP_DH, FALSE, rollback, err);
2722         if (key_ptr != NULL)
2723                 *key_ptr = dh;

2725 err:
2726         if (rollback)
2727                 {
2728                 /*
2729                  * We do not care about the return value from C_DestroyObject()
2730                  * since we are doing rollback.
2731                  */
2732                 if (found == 0)
2733                         (void) pFuncList->C_DestroyObject(session, h_key);
2734                 h_key = CK_INVALID_HANDLE;
2735                 }

2737         UNLOCK_OBJSTORE(OP_DH);

2739 malloc_err:
2740         for (i = 4; i <= 6; i++)
2741                 {
2742                 if (key_template[i].pValue != NULL)
2743                         {
2744                         OPENSSL_free(key_template[i].pValue);
2745                         key_template[i].pValue = NULL;
2746                         }
2747                 }

2749         return (h_key);
2750         }

2752 /*
2753  * Check for cache miss and clean the object pointer and handle
2754  * in such case. Return 1 for cache hit, 0 for cache miss.
2755  *
2756  * Note: we rely on pk11_destroy_dh_key_objects() to set sp->opdata_dh
2757  *       to CK_INVALID_HANDLE even when it fails to destroy the object.
2758  */
2759 static int check_new_dh_key(PK11_SESSION *sp, DH *dh)
2760         {
2761         /*
2762          * Provide protection against DH structure reuse by making the
2763          * check for cache hit stronger. Private key component of DH key
2764          * is unique so it is sufficient to compare it with value cached
2765          * in PK11_SESSION structure.
2766          */
2767         if ((sp->opdata_dh != dh) ||
```

```
2768             (BN_cmp(sp->opdata_dh_priv_num, dh->priv_key) != 0))
2769                 {
2770                 /*
2771                  * We do not check the return value because even in case of
2772                  * failure the sp structure will have both key pointer
2773                  * and object handle cleaned and pk11_destroy_object()
2774                  * reports the failure to the OpenSSL error message buffer.
2775                  */
2776                 (void) pk11_destroy_dh_object(sp, TRUE);
2777                 return (0);
2778                 }
2779         return (1);
2780         }
2781 #endif

2783 /*
2784  * Local function to simplify key template population
2785  * Return 0 -- error, 1 -- no error
2786  */
2787 static int init_template_value(BIGNUM *bn, CK_VOID_PTR *p_value,
2788         CK_ULONG *ul_value_len)
2789         {
2790         CK_ULONG len = BN_num_bytes(bn);
2791         if (len == 0)
2792                 return (1);

2794         *ul_value_len = len;
2795         *p_value = (CK_VOID_PTR)OPENSSL_malloc((size_t)*ul_value_len);
2796         if (*p_value == NULL)
2797                 return (0);

2799         BN_bn2bin(bn, *p_value);

2801         return (1);
2802         }

2804 #endif  /* OPENSSL_NO_HW_PK11 */
2805 #endif  /* OPENSSL_NO_HW */
2806 #endif /* ! codereview */
```

```
**********************************************************
    7443 Wed Aug 13 19:52:38 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_asnmth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"
#include "asn1_locl.h"
#include <openssl/evp.h>

/* If this symbol is defined then ENGINE_get_pkey_asn1_meth_engine(), the
 * function that is used by EVP to hook in pkey_asn1_meth code and cache
 * defaults (etc), will display brief debugging summaries to stderr with the
```



```
 * 'nid'. */
/* #define ENGINE_PKEY_ASN1_METH_DEBUG */

static ENGINE_TABLE *pkey_asn1_meth_table = NULL;

void ENGINE_unregister_pkey_asn1_meths(ENGINE *e)
        {
        engine_table_unregister(&pkey_asn1_meth_table, e);
        }

static void engine_unregister_all_pkey_asn1_meths(void)
        {
        engine_table_cleanup(&pkey_asn1_meth_table);
        }

int ENGINE_register_pkey_asn1_meths(ENGINE *e)
        {
        if(e->pkey_asn1_meths)
                {
                const int *nids;
                int num_nids = e->pkey_asn1_meths(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&pkey_asn1_meth_table,
                                engine_unregister_all_pkey_asn1_meths, e, nids,
                                        num_nids, 0);
                }
        return 1;
        }

void ENGINE_register_all_pkey_asn1_meths(void)
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_pkey_asn1_meths(e);
        }

int ENGINE_set_default_pkey_asn1_meths(ENGINE *e)
        {
        if(e->pkey_asn1_meths)
                {
                const int *nids;
                int num_nids = e->pkey_asn1_meths(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&pkey_asn1_meth_table,
                                engine_unregister_all_pkey_asn1_meths, e, nids,
                                        num_nids, 1);
                }
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references) for a given pkey_asn1_meth 'nid' */
ENGINE *ENGINE_get_pkey_asn1_meth_engine(int nid)
        {
        return engine_table_select(&pkey_asn1_meth_table, nid);
        }

/* Obtains a pkey_asn1_meth implementation from an ENGINE functional reference *
const EVP_PKEY_ASN1_METHOD *ENGINE_get_pkey_asn1_meth(ENGINE *e, int nid)
        {
        EVP_PKEY_ASN1_METHOD *ret;
        ENGINE_PKEY_ASN1_METHS_PTR fn = ENGINE_get_pkey_asn1_meths(e);
        if(!fn || !fn(e, &ret, NULL, nid))
                {
```

```
128                 ENGINEerr(ENGINE_F_ENGINE_GET_PKEY_ASN1_METH,
129                                 ENGINE_R_UNIMPLEMENTED_PUBLIC_KEY_METHOD);
130                 return NULL;
131                 }
132         return ret;
133         }

135 /* Gets the pkey_asn1_meth callback from an ENGINE structure */
136 ENGINE_PKEY_ASN1_METHS_PTR ENGINE_get_pkey_asn1_meths(const ENGINE *e)
137         {
138         return e->pkey_asn1_meths;
139         }

141 /* Sets the pkey_asn1_meth callback in an ENGINE structure */
142 int ENGINE_set_pkey_asn1_meths(ENGINE *e, ENGINE_PKEY_ASN1_METHS_PTR f)
143         {
144         e->pkey_asn1_meths = f;
145         return 1;
146         }

148 /* Internal function to free up EVP_PKEY_ASN1_METHOD structures before an
149  * ENGINE is destroyed
150  */

152 void engine_pkey_asn1_meths_free(ENGINE *e)
153         {
154         int i;
155         EVP_PKEY_ASN1_METHOD *pkm;
156         if (e->pkey_asn1_meths)
157                 {
158                 const int *pknids;
159                 int npknids;
160                 npknids = e->pkey_asn1_meths(e, NULL, &pknids, 0);
161                 for (i = 0; i < npknids; i++)
162                         {
163                         if (e->pkey_asn1_meths(e, &pkm, NULL, pknids[i]))
164                                 {
165                                 EVP_PKEY_asn1_free(pkm);
166                                 }
167                         }
168                 }
169         }

171 /* Find a method based on a string. This does a linear search through
172  * all implemented algorithms. This is OK in practice because only
173  * a small number of algorithms are likely to be implemented in an engine
174  * and it is not used for speed critical operations.
175  */

177 const EVP_PKEY_ASN1_METHOD *ENGINE_get_pkey_asn1_meth_str(ENGINE *e,
178                                         const char *str, int len)
179         {
180         int i, nidcount;
181         const int *nids;
182         EVP_PKEY_ASN1_METHOD *ameth;
183         if (!e->pkey_asn1_meths)
184                 return NULL;
185         if (len == -1)
186                 len = strlen(str);
187         nidcount = e->pkey_asn1_meths(e, NULL, &nids, 0);
188         for (i = 0; i < nidcount; i++)
189                 {
190                 e->pkey_asn1_meths(e, &ameth, NULL, nids[i]);
191                 if (((int)strlen(ameth->pem_str) == len) &&
192                                         !strncasecmp(ameth->pem_str, str, len))
193                         return ameth;
```



```
194                 }
195         return NULL;
196         }

198 typedef struct
199         {
200         ENGINE *e;
201         const EVP_PKEY_ASN1_METHOD *ameth;
202         const char *str;
203         int len;
204         } ENGINE_FIND_STR;

206 static void look_str_cb(int nid, STACK_OF(ENGINE) *sk, ENGINE *def, void *arg)
207         {
208         ENGINE_FIND_STR *lk = arg;
209         int i;
210         if (lk->ameth)
211                 return;
212         for (i = 0; i < sk_ENGINE_num(sk); i++)
213                 {
214                 ENGINE *e = sk_ENGINE_value(sk, i);
215                 EVP_PKEY_ASN1_METHOD *ameth;
216                 e->pkey_asn1_meths(e, &ameth, NULL, nid);
217                 if (((int)strlen(ameth->pem_str) == lk->len) &&
218                                 !strncasecmp(ameth->pem_str, lk->str, lk->len))
219                         {
220                         lk->e = e;
221                         lk->ameth = ameth;
222                         return;
223                         }
224                 }
225         }

227 const EVP_PKEY_ASN1_METHOD *ENGINE_pkey_asn1_find_str(ENGINE **pe,
228                                         const char *str, int len)
229         {
230         ENGINE_FIND_STR fstr;
231         fstr.e = NULL;
232         fstr.ameth = NULL;
233         fstr.str = str;
234         fstr.len = len;
235         CRYPTO_w_lock(CRYPTO_LOCK_ENGINE);
236         engine_table_doall(pkey_asn1_meth_table, look_str_cb, &fstr);
237         /* If found obtain a structural reference to engine */
238         if (fstr.e)
239                 {
240                 fstr.e->struct_ref++;
241                 engine_ref_debug(fstr.e, 0, 1)
242                 }
243         *pe = fstr.e;
244         CRYPTO_w_unlock(CRYPTO_LOCK_ENGINE);
245         return fstr.ameth;
246         }
247 #endif /* ! codereview */
```

```
**********************************************************
    4761 Wed Aug 13 19:52:39 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_cipher.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_cipher_engine(), the function that
 * is used by EVP to hook in cipher code and cache defaults (etc), will display
 * brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_CIPHER_DEBUG */
```



```
static ENGINE_TABLE *cipher_table = NULL;

void ENGINE_unregister_ciphers(ENGINE *e)
        {
        engine_table_unregister(&cipher_table, e);
        }

static void engine_unregister_all_ciphers(void)
        {
        engine_table_cleanup(&cipher_table);
        }

int ENGINE_register_ciphers(ENGINE *e)
        {
        if(e->ciphers)
                {
                const int *nids;
                int num_nids = e->ciphers(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&cipher_table,
                                        engine_unregister_all_ciphers, e, nids,
                                        num_nids, 0);
                }
        return 1;
        }

void ENGINE_register_all_ciphers()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_ciphers(e);
        }

int ENGINE_set_default_ciphers(ENGINE *e)
        {
        if(e->ciphers)
                {
                const int *nids;
                int num_nids = e->ciphers(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&cipher_table,
                                        engine_unregister_all_ciphers, e, nids,
                                        num_nids, 1);
                }
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references) for a given cipher 'nid' */
ENGINE *ENGINE_get_cipher_engine(int nid)
        {
        return engine_table_select(&cipher_table, nid);
        }

/* Obtains a cipher implementation from an ENGINE functional reference */
const EVP_CIPHER *ENGINE_get_cipher(ENGINE *e, int nid)
        {
        const EVP_CIPHER *ret;
        ENGINE_CIPHERS_PTR fn = ENGINE_get_ciphers(e);
        if(!fn || !fn(e, &ret, NULL, nid))
                {
                ENGINEerr(ENGINE_F_ENGINE_GET_CIPHER,
                                ENGINE_R_UNIMPLEMENTED_CIPHER);
                return NULL;
```

```
128                 }
129         return ret;
130         }

132 /* Gets the cipher callback from an ENGINE structure */
133 ENGINE_CIPHERS_PTR ENGINE_get_ciphers(const ENGINE *e)
134         {
135         return e->ciphers;
136         }

138 /* Sets the cipher callback in an ENGINE structure */
139 int ENGINE_set_ciphers(ENGINE *e, ENGINE_CIPHERS_PTR f)
140         {
141         e->ciphers = f;
142         return 1;
143         }
144 #endif /* ! codereview */
```

```
**********************************************************
    4158 Wed Aug 13 19:52:39 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_dh.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_default_DH(), the function that is
 * used by DH to hook in implementation code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_DH_DEBUG */
```



```
static ENGINE_TABLE *dh_table = NULL;
static const int dummy_nid = 1;

void ENGINE_unregister_DH(ENGINE *e)
        {
        engine_table_unregister(&dh_table, e);
        }

static void engine_unregister_all_DH(void)
        {
        engine_table_cleanup(&dh_table);
        }

int ENGINE_register_DH(ENGINE *e)
        {
        if(e->dh_meth)
                return engine_table_register(&dh_table,
                                engine_unregister_all_DH, e, &dummy_nid, 1, 0);
        return 1;
        }

void ENGINE_register_all_DH()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_DH(e);
        }

int ENGINE_set_default_DH(ENGINE *e)
        {
        if(e->dh_meth)
                return engine_table_register(&dh_table,
                                engine_unregister_all_DH, e, &dummy_nid, 1, 1);
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references). */
ENGINE *ENGINE_get_default_DH(void)
        {
        return engine_table_select(&dh_table, dummy_nid);
        }

/* Obtains an DH implementation from an ENGINE functional reference */
const DH_METHOD *ENGINE_get_DH(const ENGINE *e)
        {
        return e->dh_meth;
        }

/* Sets an DH implementation in an ENGINE structure */
int ENGINE_set_DH(ENGINE *e, const DH_METHOD *dh_meth)
        {
        e->dh_meth = dh_meth;
        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4753 Wed Aug 13 19:52:39 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_digest.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_digest_engine(), the function that
 * is used by EVP to hook in digest code and cache defaults (etc), will display
 * brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_DIGEST_DEBUG */
```



```
static ENGINE_TABLE *digest_table = NULL;

void ENGINE_unregister_digests(ENGINE *e)
        {
        engine_table_unregister(&digest_table, e);
        }

static void engine_unregister_all_digests(void)
        {
        engine_table_cleanup(&digest_table);
        }

int ENGINE_register_digests(ENGINE *e)
        {
        if(e->digests)
                {
                const int *nids;
                int num_nids = e->digests(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&digest_table,
                                        engine_unregister_all_digests, e, nids,
                                        num_nids, 0);
                }
        return 1;
        }

void ENGINE_register_all_digests()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_digests(e);
        }

int ENGINE_set_default_digests(ENGINE *e)
        {
        if(e->digests)
                {
                const int *nids;
                int num_nids = e->digests(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&digest_table,
                                        engine_unregister_all_digests, e, nids,
                                        num_nids, 1);
                }
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references) for a given digest 'nid' */
ENGINE *ENGINE_get_digest_engine(int nid)
        {
        return engine_table_select(&digest_table, nid);
        }

/* Obtains a digest implementation from an ENGINE functional reference */
const EVP_MD *ENGINE_get_digest(ENGINE *e, int nid)
        {
        const EVP_MD *ret;
        ENGINE_DIGESTS_PTR fn = ENGINE_get_digests(e);
        if(!fn || !fn(e, &ret, NULL, nid))
                {
                ENGINEerr(ENGINE_F_ENGINE_GET_DIGEST,
                                ENGINE_R_UNIMPLEMENTED_DIGEST);
                return NULL;
```

```
128                 }
129         return ret;
130         }

132 /* Gets the digest callback from an ENGINE structure */
133 ENGINE_DIGESTS_PTR ENGINE_get_digests(const ENGINE *e)
134         {
135         return e->digests;
136         }

138 /* Sets the digest callback in an ENGINE structure */
139 int ENGINE_set_digests(ENGINE *e, ENGINE_DIGESTS_PTR f)
140         {
141         e->digests = f;
142         return 1;
143         }
144 #endif /* ! codereview */
```

```
**********************************************************
    4188 Wed Aug 13 19:52:39 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_dsa.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_default_DSA(), the function that is
 * used by DSA to hook in implementation code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_DSA_DEBUG */
```



```
static ENGINE_TABLE *dsa_table = NULL;
static const int dummy_nid = 1;

void ENGINE_unregister_DSA(ENGINE *e)
        {
        engine_table_unregister(&dsa_table, e);
        }

static void engine_unregister_all_DSA(void)
        {
        engine_table_cleanup(&dsa_table);
        }

int ENGINE_register_DSA(ENGINE *e)
        {
        if(e->dsa_meth)
                return engine_table_register(&dsa_table,
                                engine_unregister_all_DSA, e, &dummy_nid, 1, 0);
        return 1;
        }

void ENGINE_register_all_DSA()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_DSA(e);
        }

int ENGINE_set_default_DSA(ENGINE *e)
        {
        if(e->dsa_meth)
                return engine_table_register(&dsa_table,
                                engine_unregister_all_DSA, e, &dummy_nid, 1, 1);
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references). */
ENGINE *ENGINE_get_default_DSA(void)
        {
        return engine_table_select(&dsa_table, dummy_nid);
        }

/* Obtains an DSA implementation from an ENGINE functional reference */
const DSA_METHOD *ENGINE_get_DSA(const ENGINE *e)
        {
        return e->dsa_meth;
        }

/* Sets an DSA implementation in an ENGINE structure */
int ENGINE_set_DSA(ENGINE *e, const DSA_METHOD *dsa_meth)
        {
        e->dsa_meth = dsa_meth;
        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4778 Wed Aug 13 19:52:39 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_ecdh.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/engine/tb_ecdh.c */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * The Elliptic Curve Public-Key Crypto Library (ECC Code) included
 * herein is developed by SUN MICROSYSTEMS, INC., and is contributed
 * to the OpenSSL project.
 *
 * The ECC Code is licensed pursuant to the OpenSSL open source
 * license provided below.
 *
 * The ECDH engine software is originally written by Nils Gura and
 * Douglas Stebila of Sun Microsystems Laboratories.
 *
 */
/* ====================================================================
 * Copyright (c) 2000-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
```



```
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_default_ECDH(), the function that i
 * used by ECDH to hook in implementation code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_ECDH_DEBUG */

static ENGINE_TABLE *ecdh_table = NULL;
static const int dummy_nid = 1;

void ENGINE_unregister_ECDH(ENGINE *e)
        {
        engine_table_unregister(&ecdh_table, e);
        }

static void engine_unregister_all_ECDH(void)
        {
        engine_table_cleanup(&ecdh_table);
        }

int ENGINE_register_ECDH(ENGINE *e)
        {
        if(e->ecdh_meth)
                return engine_table_register(&ecdh_table,
                                engine_unregister_all_ECDH, e, &dummy_nid, 1, 0)
        return 1;
        }

void ENGINE_register_all_ECDH()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_ECDH(e);
        }

int ENGINE_set_default_ECDH(ENGINE *e)
        {
        if(e->ecdh_meth)
                return engine_table_register(&ecdh_table,
                                engine_unregister_all_ECDH, e, &dummy_nid, 1, 1)
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references). */
ENGINE *ENGINE_get_default_ECDH(void)
        {
        return engine_table_select(&ecdh_table, dummy_nid);
        }

/* Obtains an ECDH implementation from an ENGINE functional reference */
const ECDH_METHOD *ENGINE_get_ECDH(const ENGINE *e)
        {
        return e->ecdh_meth;
        }
```

```
128 /* Sets an ECDH implementation in an ENGINE structure */
129 int ENGINE_set_ECDH(ENGINE *e, const ECDH_METHOD *ecdh_meth)
130         {
131         e->ecdh_meth = ecdh_meth;
132         return 1;
133         }
134 #endif /* ! codereview */
```

```
**********************************************************
    4253 Wed Aug 13 19:52:39 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_ecdsa.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2000-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_default_ECDSA(), the function that
 * used by ECDSA to hook in implementation code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_ECDSA_DEBUG */
```



```
static ENGINE_TABLE *ecdsa_table = NULL;
static const int dummy_nid = 1;

void ENGINE_unregister_ECDSA(ENGINE *e)
        {
        engine_table_unregister(&ecdsa_table, e);
        }

static void engine_unregister_all_ECDSA(void)
        {
        engine_table_cleanup(&ecdsa_table);
        }

int ENGINE_register_ECDSA(ENGINE *e)
        {
        if(e->ecdsa_meth)
                return engine_table_register(&ecdsa_table,
                                engine_unregister_all_ECDSA, e, &dummy_nid, 1, 0
        return 1;
        }

void ENGINE_register_all_ECDSA()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_ECDSA(e);
        }

int ENGINE_set_default_ECDSA(ENGINE *e)
        {
        if(e->ecdsa_meth)
                return engine_table_register(&ecdsa_table,
                                engine_unregister_all_ECDSA, e, &dummy_nid, 1, 1
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references). */
ENGINE *ENGINE_get_default_ECDSA(void)
        {
        return engine_table_select(&ecdsa_table, dummy_nid);
        }

/* Obtains an ECDSA implementation from an ENGINE functional reference */
const ECDSA_METHOD *ENGINE_get_ECDSA(const ENGINE *e)
        {
        return e->ecdsa_meth;
        }

/* Sets an ECDSA implementation in an ENGINE structure */
int ENGINE_set_ECDSA(ENGINE *e, const ECDSA_METHOD *ecdsa_meth)
        {
        e->ecdsa_meth = ecdsa_meth;
        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    5324 Wed Aug 13 19:52:40 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_pkmeth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"
#include <openssl/evp.h>

/* If this symbol is defined then ENGINE_get_pkey_meth_engine(), the function
 * that is used by EVP to hook in pkey_meth code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_PKEY_METH_DEBUG */
```



```
static ENGINE_TABLE *pkey_meth_table = NULL;

void ENGINE_unregister_pkey_meths(ENGINE *e)
        {
        engine_table_unregister(&pkey_meth_table, e);
        }

static void engine_unregister_all_pkey_meths(void)
        {
        engine_table_cleanup(&pkey_meth_table);
        }

int ENGINE_register_pkey_meths(ENGINE *e)
        {
        if(e->pkey_meths)
                {
                const int *nids;
                int num_nids = e->pkey_meths(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&pkey_meth_table,
                                engine_unregister_all_pkey_meths, e, nids,
                                        num_nids, 0);
                }
        return 1;
        }

void ENGINE_register_all_pkey_meths()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_pkey_meths(e);
        }

int ENGINE_set_default_pkey_meths(ENGINE *e)
        {
        if(e->pkey_meths)
                {
                const int *nids;
                int num_nids = e->pkey_meths(e, NULL, &nids, 0);
                if(num_nids > 0)
                        return engine_table_register(&pkey_meth_table,
                                engine_unregister_all_pkey_meths, e, nids,
                                        num_nids, 1);
                }
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references) for a given pkey_meth 'nid' */
ENGINE *ENGINE_get_pkey_meth_engine(int nid)
        {
        return engine_table_select(&pkey_meth_table, nid);
        }

/* Obtains a pkey_meth implementation from an ENGINE functional reference */
const EVP_PKEY_METHOD *ENGINE_get_pkey_meth(ENGINE *e, int nid)
        {
        EVP_PKEY_METHOD *ret;
        ENGINE_PKEY_METHS_PTR fn = ENGINE_get_pkey_meths(e);
        if(!fn || !fn(e, &ret, NULL, nid))
                {
                ENGINEerr(ENGINE_F_ENGINE_GET_PKEY_METH,
                                ENGINE_R_UNIMPLEMENTED_PUBLIC_KEY_METHOD);
```

```
128                 return NULL;
129                 }
130         return ret;
131         }

133 /* Gets the pkey_meth callback from an ENGINE structure */
134 ENGINE_PKEY_METHS_PTR ENGINE_get_pkey_meths(const ENGINE *e)
135         {
136         return e->pkey_meths;
137         }

139 /* Sets the pkey_meth callback in an ENGINE structure */
140 int ENGINE_set_pkey_meths(ENGINE *e, ENGINE_PKEY_METHS_PTR f)
141         {
142         e->pkey_meths = f;
143         return 1;
144         }

146 /* Internal function to free up EVP_PKEY_METHOD structures before an
147  * ENGINE is destroyed
148  */

150 void engine_pkey_meths_free(ENGINE *e)
151         {
152         int i;
153         EVP_PKEY_METHOD *pkm;
154         if (e->pkey_meths)
155                 {
156                 const int *pknids;
157                 int npknids;
158                 npknids = e->pkey_meths(e, NULL, &pknids, 0);
159                 for (i = 0; i < npknids; i++)
160                         {
161                         if (e->pkey_meths(e, &pkm, NULL, pknids[i]))
162                                 {
163                                 EVP_PKEY_meth_free(pkm);
164                                 }
165                         }
166                 }
167         }
168 #endif /* ! codereview */
```

```
**********************************************************
    4218 Wed Aug 13 19:52:40 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_rand.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_default_RAND(), the function that i
 * used by RAND to hook in implementation code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_RAND_DEBUG */
```



```
static ENGINE_TABLE *rand_table = NULL;
static const int dummy_nid = 1;

void ENGINE_unregister_RAND(ENGINE *e)
        {
        engine_table_unregister(&rand_table, e);
        }

static void engine_unregister_all_RAND(void)
        {
        engine_table_cleanup(&rand_table);
        }

int ENGINE_register_RAND(ENGINE *e)
        {
        if(e->rand_meth)
                return engine_table_register(&rand_table,
                                engine_unregister_all_RAND, e, &dummy_nid, 1, 0)
        return 1;
        }

void ENGINE_register_all_RAND()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_RAND(e);
        }

int ENGINE_set_default_RAND(ENGINE *e)
        {
        if(e->rand_meth)
                return engine_table_register(&rand_table,
                                engine_unregister_all_RAND, e, &dummy_nid, 1, 1)
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references). */
ENGINE *ENGINE_get_default_RAND(void)
        {
        return engine_table_select(&rand_table, dummy_nid);
        }

/* Obtains an RAND implementation from an ENGINE functional reference */
const RAND_METHOD *ENGINE_get_RAND(const ENGINE *e)
        {
        return e->rand_meth;
        }

/* Sets an RAND implementation in an ENGINE structure */
int ENGINE_set_RAND(ENGINE *e, const RAND_METHOD *rand_meth)
        {
        e->rand_meth = rand_meth;
        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4188 Wed Aug 13 19:52:40 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_rsa.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_default_RSA(), the function that is
 * used by RSA to hook in implementation code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_RSA_DEBUG */
```



```
static ENGINE_TABLE *rsa_table = NULL;
static const int dummy_nid = 1;

void ENGINE_unregister_RSA(ENGINE *e)
        {
        engine_table_unregister(&rsa_table, e);
        }

static void engine_unregister_all_RSA(void)
        {
        engine_table_cleanup(&rsa_table);
        }

int ENGINE_register_RSA(ENGINE *e)
        {
        if(e->rsa_meth)
                return engine_table_register(&rsa_table,
                                engine_unregister_all_RSA, e, &dummy_nid, 1, 0);
        return 1;
        }

void ENGINE_register_all_RSA()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_RSA(e);
        }

int ENGINE_set_default_RSA(ENGINE *e)
        {
        if(e->rsa_meth)
                return engine_table_register(&rsa_table,
                                engine_unregister_all_RSA, e, &dummy_nid, 1, 1);
        return 1;
        }

/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references). */
ENGINE *ENGINE_get_default_RSA(void)
        {
        return engine_table_select(&rsa_table, dummy_nid);
        }

/* Obtains an RSA implementation from an ENGINE functional reference */
const RSA_METHOD *ENGINE_get_RSA(const ENGINE *e)
        {
        return e->rsa_meth;
        }

/* Sets an RSA implementation in an ENGINE structure */
int ENGINE_set_RSA(ENGINE *e, const RSA_METHOD *rsa_meth)
        {
        e->rsa_meth = rsa_meth;
        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4345 Wed Aug 13 19:52:40 2014
new/usr/src/lib/openssl/libsunw_crypto/engine/tb_store.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "eng_int.h"

/* If this symbol is defined then ENGINE_get_default_STORE(), the function that
 * used by STORE to hook in implementation code and cache defaults (etc), will
 * display brief debugging summaries to stderr with the 'nid'. */
/* #define ENGINE_STORE_DEBUG */
```



```
static ENGINE_TABLE *store_table = NULL;
static const int dummy_nid = 1;

void ENGINE_unregister_STORE(ENGINE *e)
        {
        engine_table_unregister(&store_table, e);
        }

static void engine_unregister_all_STORE(void)
        {
        engine_table_cleanup(&store_table);
        }

int ENGINE_register_STORE(ENGINE *e)
        {
        if(e->store_meth)
                return engine_table_register(&store_table,
                                engine_unregister_all_STORE, e, &dummy_nid, 1, 0
        return 1;
        }

void ENGINE_register_all_STORE()
        {
        ENGINE *e;

        for(e=ENGINE_get_first() ; e ; e=ENGINE_get_next(e))
                ENGINE_register_STORE(e);
        }

/* The following two functions are removed because they're useless. */
#if 0
int ENGINE_set_default_STORE(ENGINE *e)
        {
        if(e->store_meth)
                return engine_table_register(&store_table,
                                engine_unregister_all_STORE, e, &dummy_nid, 1, 1
        return 1;
        }
#endif

#if 0
/* Exposed API function to get a functional reference from the implementation
 * table (ie. try to get a functional reference from the tabled structural
 * references). */
ENGINE *ENGINE_get_default_STORE(void)
        {
        return engine_table_select(&store_table, dummy_nid);
        }
#endif

/* Obtains an STORE implementation from an ENGINE functional reference */
const STORE_METHOD *ENGINE_get_STORE(const ENGINE *e)
        {
        return e->store_meth;
        }

/* Sets an STORE implementation in an ENGINE structure */
int ENGINE_set_STORE(ENGINE *e, const STORE_METHOD *store_meth)
        {
        e->store_meth = store_meth;
        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
   30122 Wed Aug 13 19:52:40 2014
new/usr/src/lib/openssl/libsunw_crypto/err/err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/err/err.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "cryptlib.h"
#include <openssl/lhash.h>
#include <openssl/crypto.h>
#include <openssl/buffer.h>
#include <openssl/bio.h>
#include <openssl/err.h>

DECLARE_LHASH_OF(ERR_STRING_DATA);
DECLARE_LHASH_OF(ERR_STATE);

static void err_load_strings(int lib, ERR_STRING_DATA *str);

static void ERR_STATE_free(ERR_STATE *s);
```

```
#ifndef OPENSSL_NO_ERR
static ERR_STRING_DATA ERR_str_libraries[]=
        {
{ERR_PACK(ERR_LIB_NONE,0,0)             ,"unknown library"},
{ERR_PACK(ERR_LIB_SYS,0,0)              ,"system library"},
{ERR_PACK(ERR_LIB_BN,0,0)               ,"bignum routines"},
{ERR_PACK(ERR_LIB_RSA,0,0)              ,"rsa routines"},
{ERR_PACK(ERR_LIB_DH,0,0)               ,"Diffie-Hellman routines"},
{ERR_PACK(ERR_LIB_EVP,0,0)              ,"digital envelope routines"},
{ERR_PACK(ERR_LIB_BUF,0,0)              ,"memory buffer routines"},
{ERR_PACK(ERR_LIB_OBJ,0,0)              ,"object identifier routines"},
{ERR_PACK(ERR_LIB_PEM,0,0)              ,"PEM routines"},
{ERR_PACK(ERR_LIB_DSA,0,0)              ,"dsa routines"},
{ERR_PACK(ERR_LIB_X509,0,0)             ,"x509 certificate routines"},
{ERR_PACK(ERR_LIB_ASN1,0,0)             ,"asn1 encoding routines"},
{ERR_PACK(ERR_LIB_CONF,0,0)             ,"configuration file routines"},
{ERR_PACK(ERR_LIB_CRYPTO,0,0)           ,"common libcrypto routines"},
{ERR_PACK(ERR_LIB_EC,0,0)               ,"elliptic curve routines"},
{ERR_PACK(ERR_LIB_SSL,0,0)              ,"SSL routines"},
{ERR_PACK(ERR_LIB_BIO,0,0)              ,"BIO routines"},
{ERR_PACK(ERR_LIB_PKCS7,0,0)            ,"PKCS7 routines"},
{ERR_PACK(ERR_LIB_X509V3,0,0)           ,"X509 V3 routines"},
{ERR_PACK(ERR_LIB_PKCS12,0,0)           ,"PKCS12 routines"},
{ERR_PACK(ERR_LIB_RAND,0,0)             ,"random number generator"},
{ERR_PACK(ERR_LIB_DSO,0,0)              ,"DSO support routines"},
{ERR_PACK(ERR_LIB_TS,0,0)               ,"time stamp routines"},
{ERR_PACK(ERR_LIB_ENGINE,0,0)           ,"engine routines"},
{ERR_PACK(ERR_LIB_OCSP,0,0)             ,"OCSP routines"},
{ERR_PACK(ERR_LIB_FIPS,0,0)             ,"FIPS routines"},
{ERR_PACK(ERR_LIB_CMS,0,0)              ,"CMS routines"},
{ERR_PACK(ERR_LIB_HMAC,0,0)             ,"HMAC routines"},
{0,NULL},
        };

static ERR_STRING_DATA ERR_str_functs[]=
        {
        {ERR_PACK(0,SYS_F_FOPEN,0),             "fopen"},
        {ERR_PACK(0,SYS_F_CONNECT,0),           "connect"},
        {ERR_PACK(0,SYS_F_GETSERVBYNAME,0),     "getservbyname"},
        {ERR_PACK(0,SYS_F_SOCKET,0),            "socket"},
        {ERR_PACK(0,SYS_F_IOCTLSOCKET,0),       "ioctlsocket"},
        {ERR_PACK(0,SYS_F_BIND,0),              "bind"},
        {ERR_PACK(0,SYS_F_LISTEN,0),            "listen"},
        {ERR_PACK(0,SYS_F_ACCEPT,0),            "accept"},
#ifdef OPENSSL_SYS_WINDOWS
        {ERR_PACK(0,SYS_F_WSASTARTUP,0),        "WSAstartup"},
#endif
        {ERR_PACK(0,SYS_F_OPENDIR,0),           "opendir"},
        {ERR_PACK(0,SYS_F_FREAD,0),             "fread"},
        {0,NULL},
        };

static ERR_STRING_DATA ERR_str_reasons[]=
        {
{ERR_R_SYS_LIB                          ,"system lib"},
{ERR_R_BN_LIB                           ,"BN lib"},
{ERR_R_RSA_LIB                          ,"RSA lib"},
{ERR_R_DH_LIB                           ,"DH lib"},
{ERR_R_EVP_LIB                          ,"EVP lib"},
{ERR_R_BUF_LIB                          ,"BUF lib"},
{ERR_R_OBJ_LIB                          ,"OBJ lib"},
{ERR_R_PEM_LIB                          ,"PEM lib"},
{ERR_R_DSA_LIB                          ,"DSA lib"},
{ERR_R_X509_LIB                         ,"X509 lib"},
{ERR_R_ASN1_LIB                         ,"ASN1 lib"},
{ERR_R_CONF_LIB                         ,"CONF lib"},
```



```
{ERR_R_CRYPTO_LIB                       ,"CRYPTO lib"},
{ERR_R_EC_LIB                           ,"EC lib"},
{ERR_R_SSL_LIB                          ,"SSL lib"},
{ERR_R_BIO_LIB                          ,"BIO lib"},
{ERR_R_PKCS7_LIB                        ,"PKCS7 lib"},
{ERR_R_X509V3_LIB                       ,"X509V3 lib"},
{ERR_R_PKCS12_LIB                       ,"PKCS12 lib"},
{ERR_R_RAND_LIB                         ,"RAND lib"},
{ERR_R_DSO_LIB                          ,"DSO lib"},
{ERR_R_ENGINE_LIB                       ,"ENGINE lib"},
{ERR_R_OCSP_LIB                         ,"OCSP lib"},
{ERR_R_TS_LIB                           ,"TS lib"},

{ERR_R_NESTED_ASN1_ERROR                ,"nested asn1 error"},
{ERR_R_BAD_ASN1_OBJECT_HEADER           ,"bad asn1 object header"},
{ERR_R_BAD_GET_ASN1_OBJECT_CALL         ,"bad get asn1 object call"},
{ERR_R_EXPECTING_AN_ASN1_SEQUENCE       ,"expecting an asn1 sequence"},
{ERR_R_ASN1_LENGTH_MISMATCH             ,"asn1 length mismatch"},
{ERR_R_MISSING_ASN1_EOS                 ,"missing asn1 eos"},

{ERR_R_FATAL                            ,"fatal"},
{ERR_R_MALLOC_FAILURE                   ,"malloc failure"},
{ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED      ,"called a function you should not call"
{ERR_R_PASSED_NULL_PARAMETER            ,"passed a null parameter"},
{ERR_R_INTERNAL_ERROR                   ,"internal error"},
{ERR_R_DISABLED                         ,"called a function that was disabled at

{0,NULL},
        };
#endif


/* Define the predeclared (but externally opaque) "ERR_FNS" type */
struct st_ERR_FNS
        {
        /* Works on the "error_hash" string table */
        LHASH_OF(ERR_STRING_DATA) *(*cb_err_get)(int create);
        void (*cb_err_del)(void);
        ERR_STRING_DATA *(*cb_err_get_item)(const ERR_STRING_DATA *);
        ERR_STRING_DATA *(*cb_err_set_item)(ERR_STRING_DATA *);
        ERR_STRING_DATA *(*cb_err_del_item)(ERR_STRING_DATA *);
        /* Works on the "thread_hash" error-state table */
        LHASH_OF(ERR_STATE) *(*cb_thread_get)(int create);
        void (*cb_thread_release)(LHASH_OF(ERR_STATE) **hash);
        ERR_STATE *(*cb_thread_get_item)(const ERR_STATE *);
        ERR_STATE *(*cb_thread_set_item)(ERR_STATE *);
        void (*cb_thread_del_item)(const ERR_STATE *);
        /* Returns the next available error "library" numbers */
        int (*cb_get_next_lib)(void);
        };

/* Predeclarations of the "err_defaults" functions */
static LHASH_OF(ERR_STRING_DATA) *int_err_get(int create);
static void int_err_del(void);
static ERR_STRING_DATA *int_err_get_item(const ERR_STRING_DATA *);
static ERR_STRING_DATA *int_err_set_item(ERR_STRING_DATA *);
static ERR_STRING_DATA *int_err_del_item(ERR_STRING_DATA *);
static LHASH_OF(ERR_STATE) *int_thread_get(int create);
static void int_thread_release(LHASH_OF(ERR_STATE) **hash);
static ERR_STATE *int_thread_get_item(const ERR_STATE *);
static ERR_STATE *int_thread_set_item(ERR_STATE *);
static void int_thread_del_item(const ERR_STATE *);
static int int_err_get_next_lib(void);
/* The static ERR_FNS table using these defaults functions */
static const ERR_FNS err_defaults =
        {
```

```
260         int_err_get,
261         int_err_del,
262         int_err_get_item,
263         int_err_set_item,
264         int_err_del_item,
265         int_thread_get,
266         int_thread_release,
267         int_thread_get_item,
268         int_thread_set_item,
269         int_thread_del_item,
270         int_err_get_next_lib
271         };

273 /* The replacable table of ERR_FNS functions we use at run-time */
274 static const ERR_FNS *err_fns = NULL;

276 /* Eg. rather than using "err_get()", use "ERRFN(err_get)()". */
277 #define ERRFN(a) err_fns->cb_##a

279 /* The internal state used by "err_defaults" - as such, the setting, reading,
280  * creating, and deleting of this data should only be permitted via the
281  * "err_defaults" functions. This way, a linked module can completely defer all
282  * ERR state operation (together with requisite locking) to the implementations
283  * and state in the loading application. */
284 static LHASH_OF(ERR_STRING_DATA) *int_error_hash = NULL;
285 static LHASH_OF(ERR_STATE) *int_thread_hash = NULL;
286 static int int_thread_hash_references = 0;
287 static int int_err_library_number= ERR_LIB_USER;

289 /* Internal function that checks whether "err_fns" is set and if not, sets it to
290  * the defaults. */
291 static void err_fns_check(void)
292         {
293         if (err_fns) return;

295         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
296         if (!err_fns)
297                 err_fns = &err_defaults;
298         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);
299         }

301 /* API functions to get or set the underlying ERR functions. */

303 const ERR_FNS *ERR_get_implementation(void)
304         {
305         err_fns_check();
306         return err_fns;
307         }

309 int ERR_set_implementation(const ERR_FNS *fns)
310         {
311         int ret = 0;

313         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
314         /* It's too late if 'err_fns' is non-NULL. BTW: not much point setting
315          * an error is there?! */
316         if (!err_fns)
317                 {
318                 err_fns = fns;
319                 ret = 1;
320                 }
321         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);
322         return ret;
323         }

325 /* These are the callbacks provided to "lh_new()" when creating the LHASH tables
```



```
326  * internal to the "err_defaults" implementation. */

328 static unsigned long get_error_values(int inc,int top,const char **file,int *lin
329                                       const char **data,int *flags);

331 /* The internal functions used in the "err_defaults" implementation */

333 static unsigned long err_string_data_hash(const ERR_STRING_DATA *a)
334         {
335         unsigned long ret,l;

337         l=a->error;
338         ret=l^ERR_GET_LIB(l)^ERR_GET_FUNC(l);
339         return(ret^ret%19*13);
340         }
341 static IMPLEMENT_LHASH_HASH_FN(err_string_data, ERR_STRING_DATA)

343 static int err_string_data_cmp(const ERR_STRING_DATA *a,
344                                const ERR_STRING_DATA *b)
345         {
346         return (int)(a->error - b->error);
347         }
348 static IMPLEMENT_LHASH_COMP_FN(err_string_data, ERR_STRING_DATA)

350 static LHASH_OF(ERR_STRING_DATA) *int_err_get(int create)
351         {
352         LHASH_OF(ERR_STRING_DATA) *ret = NULL;

354         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
355         if (!int_error_hash && create)
356                 {
357                 CRYPTO_push_info("int_err_get (err.c)");
358                 int_error_hash = lh_ERR_STRING_DATA_new();
359                 CRYPTO_pop_info();
360                 }
361         if (int_error_hash)
362                 ret = int_error_hash;
363         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);

365         return ret;
366         }

368 static void int_err_del(void)
369         {
370         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
371         if (int_error_hash)
372                 {
373                 lh_ERR_STRING_DATA_free(int_error_hash);
374                 int_error_hash = NULL;
375                 }
376         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);
377         }

379 static ERR_STRING_DATA *int_err_get_item(const ERR_STRING_DATA *d)
380         {
381         ERR_STRING_DATA *p;
382         LHASH_OF(ERR_STRING_DATA) *hash;

384         err_fns_check();
385         hash = ERRFN(err_get)(0);
386         if (!hash)
387                 return NULL;

389         CRYPTO_r_lock(CRYPTO_LOCK_ERR);
390         p = lh_ERR_STRING_DATA_retrieve(hash, d);
391         CRYPTO_r_unlock(CRYPTO_LOCK_ERR);
```

```
393         return p;
394         }

396 static ERR_STRING_DATA *int_err_set_item(ERR_STRING_DATA *d)
397         {
398         ERR_STRING_DATA *p;
399         LHASH_OF(ERR_STRING_DATA) *hash;

401         err_fns_check();
402         hash = ERRFN(err_get)(1);
403         if (!hash)
404                 return NULL;

406         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
407         p = lh_ERR_STRING_DATA_insert(hash, d);
408         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);

410         return p;
411         }

413 static ERR_STRING_DATA *int_err_del_item(ERR_STRING_DATA *d)
414         {
415         ERR_STRING_DATA *p;
416         LHASH_OF(ERR_STRING_DATA) *hash;

418         err_fns_check();
419         hash = ERRFN(err_get)(0);
420         if (!hash)
421                 return NULL;

423         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
424         p = lh_ERR_STRING_DATA_delete(hash, d);
425         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);

427         return p;
428         }

430 static unsigned long err_state_hash(const ERR_STATE *a)
431         {
432         return CRYPTO_THREADID_hash(&a->tid) * 13;
433         }
434 static IMPLEMENT_LHASH_HASH_FN(err_state, ERR_STATE)

436 static int err_state_cmp(const ERR_STATE *a, const ERR_STATE *b)
437         {
438         return CRYPTO_THREADID_cmp(&a->tid, &b->tid);
439         }
440 static IMPLEMENT_LHASH_COMP_FN(err_state, ERR_STATE)

442 static LHASH_OF(ERR_STATE) *int_thread_get(int create)
443         {
444         LHASH_OF(ERR_STATE) *ret = NULL;

446         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
447         if (!int_thread_hash && create)
448                 {
449                 CRYPTO_push_info("int_thread_get (err.c)");
450                 int_thread_hash = lh_ERR_STATE_new();
451                 CRYPTO_pop_info();
452                 }
453         if (int_thread_hash)
454                 {
455                 int_thread_hash_references++;
456                 ret = int_thread_hash;
457                 }
```



```
458         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);
459         return ret;
460         }

462 static void int_thread_release(LHASH_OF(ERR_STATE) **hash)
463         {
464         int i;

466         if (hash == NULL || *hash == NULL)
467                 return;

469         i = CRYPTO_add(&int_thread_hash_references, -1, CRYPTO_LOCK_ERR);

471 #ifdef REF_PRINT
472         fprintf(stderr,"%4d:%s\n",int_thread_hash_references,"ERR");
473 #endif
474         if (i > 0) return;
475 #ifdef REF_CHECK
476         if (i < 0)
477                 {
478                 fprintf(stderr,"int_thread_release, bad reference count\n");
479                 abort(); /* ok */
480                 }
481 #endif
482         *hash = NULL;
483         }

485 static ERR_STATE *int_thread_get_item(const ERR_STATE *d)
486         {
487         ERR_STATE *p;
488         LHASH_OF(ERR_STATE) *hash;

490         err_fns_check();
491         hash = ERRFN(thread_get)(0);
492         if (!hash)
493                 return NULL;

495         CRYPTO_r_lock(CRYPTO_LOCK_ERR);
496         p = lh_ERR_STATE_retrieve(hash, d);
497         CRYPTO_r_unlock(CRYPTO_LOCK_ERR);

499         ERRFN(thread_release)(&hash);
500         return p;
501         }

503 static ERR_STATE *int_thread_set_item(ERR_STATE *d)
504         {
505         ERR_STATE *p;
506         LHASH_OF(ERR_STATE) *hash;

508         err_fns_check();
509         hash = ERRFN(thread_get)(1);
510         if (!hash)
511                 return NULL;

513         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
514         p = lh_ERR_STATE_insert(hash, d);
515         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);

517         ERRFN(thread_release)(&hash);
518         return p;
519         }

521 static void int_thread_del_item(const ERR_STATE *d)
522         {
523         ERR_STATE *p;
```

```
524         LHASH_OF(ERR_STATE) *hash;

526         err_fns_check();
527         hash = ERRFN(thread_get)(0);
528         if (!hash)
529                 return;

531         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
532         p = lh_ERR_STATE_delete(hash, d);
533         /* make sure we don't leak memory */
534         if (int_thread_hash_references == 1
535             && int_thread_hash && lh_ERR_STATE_num_items(int_thread_hash) == 0)
536                 {
537                 lh_ERR_STATE_free(int_thread_hash);
538                 int_thread_hash = NULL;
539                 }
540         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);

542         ERRFN(thread_release)(&hash);
543         if (p)
544                 ERR_STATE_free(p);
545         }

547 static int int_err_get_next_lib(void)
548         {
549         int ret;

551         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
552         ret = int_err_library_number++;
553         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);

555         return ret;
556         }


559 #ifndef OPENSSL_NO_ERR
560 #define NUM_SYS_STR_REASONS 127
561 #define LEN_SYS_STR_REASON 32

563 static ERR_STRING_DATA SYS_str_reasons[NUM_SYS_STR_REASONS + 1];
564 /* SYS_str_reasons is filled with copies of strerror() results at
565  * initialization.
566  * 'errno' values up to 127 should cover all usual errors,
567  * others will be displayed numerically by ERR_error_string.
568  * It is crucial that we have something for each reason code
569  * that occurs in ERR_str_reasons, or bogus reason strings
570  * will be returned for SYSerr(), which always gets an errno
571  * value and never one of those 'standard' reason codes. */

573 static void build_SYS_str_reasons(void)
574         {
575         /* OPENSSL_malloc cannot be used here, use static storage instead */
576         static char strerror_tab[NUM_SYS_STR_REASONS][LEN_SYS_STR_REASON];
577         int i;
578         static int init = 1;

580         CRYPTO_r_lock(CRYPTO_LOCK_ERR);
581         if (!init)
582                 {
583                 CRYPTO_r_unlock(CRYPTO_LOCK_ERR);
584                 return;
585                 }

587         CRYPTO_r_unlock(CRYPTO_LOCK_ERR);
588         CRYPTO_w_lock(CRYPTO_LOCK_ERR);
589         if (!init)
```



```
590                 {
591                 CRYPTO_w_unlock(CRYPTO_LOCK_ERR);
592                 return;
593                 }

595         for (i = 1; i <= NUM_SYS_STR_REASONS; i++)
596                 {
597                 ERR_STRING_DATA *str = &SYS_str_reasons[i - 1];

599                 str->error = (unsigned long)i;
600                 if (str->string == NULL)
601                         {
602                         char (*dest)[LEN_SYS_STR_REASON] = &(strerror_tab[i - 1]
603                         char *src = strerror(i);
604                         if (src != NULL)
605                                 {
606                                 strncpy(*dest, src, sizeof *dest);
607                                 (*dest)[sizeof *dest - 1] = '\0';
608                                 str->string = *dest;
609                                 }
610                         }
611                 if (str->string == NULL)
612                         str->string = "unknown";
613                 }

615         /* Now we still have SYS_str_reasons[NUM_SYS_STR_REASONS] = {0, NULL},
616          * as required by ERR_load_strings. */

618         init = 0;

620         CRYPTO_w_unlock(CRYPTO_LOCK_ERR);
621         }
622 #endif

624 #define err_clear_data(p,i) \
625         do { \
626         if (((p)->err_data[i] != NULL) && \
627                 (p)->err_data_flags[i] & ERR_TXT_MALLOCED) \
628                 { \
629                 OPENSSL_free((p)->err_data[i]); \
630                 (p)->err_data[i]=NULL; \
631                 } \
632         (p)->err_data_flags[i]=0; \
633         } while(0)

635 #define err_clear(p,i) \
636         do { \
637         (p)->err_flags[i]=0; \
638         (p)->err_buffer[i]=0; \
639         err_clear_data(p,i); \
640         (p)->err_file[i]=NULL; \
641         (p)->err_line[i]= -1; \
642         } while(0)

644 static void ERR_STATE_free(ERR_STATE *s)
645         {
646         int i;

648         if (s == NULL)
649             return;

651         for (i=0; i<ERR_NUM_ERRORS; i++)
652                 {
653                 err_clear_data(s,i);
654                 }
655         OPENSSL_free(s);
```

```
656         }

658 void ERR_load_ERR_strings(void)
659         {
660         err_fns_check();
661 #ifndef OPENSSL_NO_ERR
662         err_load_strings(0,ERR_str_libraries);
663         err_load_strings(0,ERR_str_reasons);
664         err_load_strings(ERR_LIB_SYS,ERR_str_functs);
665         build_SYS_str_reasons();
666         err_load_strings(ERR_LIB_SYS,SYS_str_reasons);
667 #endif
668         }

670 static void err_load_strings(int lib, ERR_STRING_DATA *str)
671         {
672         while (str->error)
673                 {
674                 if (lib)
675                         str->error|=ERR_PACK(lib,0,0);
676                 ERRFN(err_set_item)(str);
677                 str++;
678                 }
679         }

681 void ERR_load_strings(int lib, ERR_STRING_DATA *str)
682         {
683         ERR_load_ERR_strings();
684         err_load_strings(lib, str);
685         }

687 void ERR_unload_strings(int lib, ERR_STRING_DATA *str)
688         {
689         while (str->error)
690                 {
691                 if (lib)
692                         str->error|=ERR_PACK(lib,0,0);
693                 ERRFN(err_del_item)(str);
694                 str++;
695                 }
696         }

698 void ERR_free_strings(void)
699         {
700         err_fns_check();
701         ERRFN(err_del)();
702         }

704 /********************************************************/

706 void ERR_put_error(int lib, int func, int reason, const char *file,
707              int line)
708         {
709         ERR_STATE *es;

711 #ifdef _OSD_POSIX
712         /* In the BS2000-OSD POSIX subsystem, the compiler generates
713          * path names in the form "*POSIX(/etc/passwd)".
714          * This dirty hack strips them to something sensible.
715          * @@@ We shouldn't modify a const string, though.
716          */
717         if (strncmp(file,"*POSIX(", sizeof("*POSIX(")-1) == 0) {
718                 char *end;

720                 /* Skip the "*POSIX(" prefix */
721                 file += sizeof("*POSIX(")-1;
```



```
722                 end = &file[strlen(file)-1];
723                 if (*end == ')')
724                         *end = '\0';
725                 /* Optional: use the basename of the path only. */
726                 if ((end = strrchr(file, '/')) != NULL)
727                         file = &end[1];
728         }
729 #endif
730         es=ERR_get_state();

732         es->top=(es->top+1)%ERR_NUM_ERRORS;
733         if (es->top == es->bottom)
734                 es->bottom=(es->bottom+1)%ERR_NUM_ERRORS;
735         es->err_flags[es->top]=0;
736         es->err_buffer[es->top]=ERR_PACK(lib,func,reason);
737         es->err_file[es->top]=file;
738         es->err_line[es->top]=line;
739         err_clear_data(es,es->top);
740         }

742 void ERR_clear_error(void)
743         {
744         int i;
745         ERR_STATE *es;

747         es=ERR_get_state();

749         for (i=0; i<ERR_NUM_ERRORS; i++)
750                 {
751                 err_clear(es,i);
752                 }
753         es->top=es->bottom=0;
754         }


757 unsigned long ERR_get_error(void)
758         { return(get_error_values(1,0,NULL,NULL,NULL,NULL)); }

760 unsigned long ERR_get_error_line(const char **file,
761              int *line)
762         { return(get_error_values(1,0,file,line,NULL,NULL)); }

764 unsigned long ERR_get_error_line_data(const char **file, int *line,
765              const char **data, int *flags)
766         { return(get_error_values(1,0,file,line,data,flags)); }


769 unsigned long ERR_peek_error(void)
770         { return(get_error_values(0,0,NULL,NULL,NULL,NULL)); }

772 unsigned long ERR_peek_error_line(const char **file, int *line)
773         { return(get_error_values(0,0,file,line,NULL,NULL)); }

775 unsigned long ERR_peek_error_line_data(const char **file, int *line,
776              const char **data, int *flags)
777         { return(get_error_values(0,0,file,line,data,flags)); }


780 unsigned long ERR_peek_last_error(void)
781         { return(get_error_values(0,1,NULL,NULL,NULL,NULL)); }

783 unsigned long ERR_peek_last_error_line(const char **file, int *line)
784         { return(get_error_values(0,1,file,line,NULL,NULL)); }

786 unsigned long ERR_peek_last_error_line_data(const char **file, int *line,
787              const char **data, int *flags)
```

```
788         { return(get_error_values(0,1,file,line,data,flags)); }


791 static unsigned long get_error_values(int inc, int top, const char **file, int *
792              const char **data, int *flags)
793         {
794         int i=0;
795         ERR_STATE *es;
796         unsigned long ret;

798         es=ERR_get_state();

800         if (inc && top)
801                 {
802                 if (file) *file = "";
803                 if (line) *line = 0;
804                 if (data) *data = "";
805                 if (flags) *flags = 0;

807                 return ERR_R_INTERNAL_ERROR;
808                 }

810         if (es->bottom == es->top) return 0;
811         if (top)
812                 i=es->top;                          /* last error */
813         else
814                 i=(es->bottom+1)%ERR_NUM_ERRORS; /* first error */

816         ret=es->err_buffer[i];
817         if (inc)
818                 {
819                 es->bottom=i;
820                 es->err_buffer[i]=0;
821                 }

823         if ((file != NULL) && (line != NULL))
824                 {
825                 if (es->err_file[i] == NULL)
826                         {
827                         *file="NA";
828                         if (line != NULL) *line=0;
829                         }
830                 else
831                         {
832                         *file=es->err_file[i];
833                         if (line != NULL) *line=es->err_line[i];
834                         }
835                 }

837         if (data == NULL)
838                 {
839                 if (inc)
840                         {
841                         err_clear_data(es, i);
842                         }
843                 }
844         else
845                 {
846                 if (es->err_data[i] == NULL)
847                         {
848                         *data="";
849                         if (flags != NULL) *flags=0;
850                         }
851                 else
852                         {
853                         *data=es->err_data[i];
```



```
854                         if (flags != NULL) *flags=es->err_data_flags[i];
855                         }
856                 }
857         return ret;
858         }

860 void ERR_error_string_n(unsigned long e, char *buf, size_t len)
861         {
862         char lsbuf[64], fsbuf[64], rsbuf[64];
863         const char *ls,*fs,*rs;
864         unsigned long l,f,r;

866         l=ERR_GET_LIB(e);
867         f=ERR_GET_FUNC(e);
868         r=ERR_GET_REASON(e);

870         ls=ERR_lib_error_string(e);
871         fs=ERR_func_error_string(e);
872         rs=ERR_reason_error_string(e);

874         if (ls == NULL)
875                 BIO_snprintf(lsbuf, sizeof(lsbuf), "lib(%lu)", l);
876         if (fs == NULL)
877                 BIO_snprintf(fsbuf, sizeof(fsbuf), "func(%lu)", f);
878         if (rs == NULL)
879                 BIO_snprintf(rsbuf, sizeof(rsbuf), "reason(%lu)", r);

881         BIO_snprintf(buf, len,"error:%08lX:%s:%s:%s", e, ls?ls:lsbuf,
882                 fs?fs:fsbuf, rs?rs:rsbuf);
883         if (strlen(buf) == len-1)
884                 {
885                 /* output may be truncated; make sure we always have 5
886                  * colon-separated fields, i.e. 4 colons ... */
887 #define NUM_COLONS 4
888                 if (len > NUM_COLONS) /* ... if possible */
889                         {
890                         int i;
891                         char *s = buf;

893                         for (i = 0; i < NUM_COLONS; i++)
894                                 {
895                                 char *colon = strchr(s, ':');
896                                 if (colon == NULL || colon > &buf[len-1] - NUM_C
897                                         {
898                                         /* set colon no. i at last possible posi
899                                          * (buf[len-1] is the terminating 0)*/
900                                         colon = &buf[len-1] - NUM_COLONS + i;
901                                         *colon = ':';
902                                         }
903                                 s = colon + 1;
904                                 }
905                         }
906                 }
907         }

909 /* BAD for multi-threading: uses a local buffer if ret == NULL */
910 /* ERR_error_string_n should be used instead for ret != NULL
911  * as ERR_error_string cannot know how large the buffer is */
912 char *ERR_error_string(unsigned long e, char *ret)
913         {
914         static char buf[256];

916         if (ret == NULL) ret=buf;
917         ERR_error_string_n(e, ret, 256);

919         return ret;
```

```
920         }

922 LHASH_OF(ERR_STRING_DATA) *ERR_get_string_table(void)
923         {
924         err_fns_check();
925         return ERRFN(err_get)(0);
926         }

928 LHASH_OF(ERR_STATE) *ERR_get_err_state_table(void)
929         {
930         err_fns_check();
931         return ERRFN(thread_get)(0);
932         }

934 void ERR_release_err_state_table(LHASH_OF(ERR_STATE) **hash)
935         {
936         err_fns_check();
937         ERRFN(thread_release)(hash);
938         }

940 const char *ERR_lib_error_string(unsigned long e)
941         {
942         ERR_STRING_DATA d,*p;
943         unsigned long l;

945         err_fns_check();
946         l=ERR_GET_LIB(e);
947         d.error=ERR_PACK(l,0,0);
948         p=ERRFN(err_get_item)(&d);
949         return((p == NULL)?NULL:p->string);
950         }

952 const char *ERR_func_error_string(unsigned long e)
953         {
954         ERR_STRING_DATA d,*p;
955         unsigned long l,f;

957         err_fns_check();
958         l=ERR_GET_LIB(e);
959         f=ERR_GET_FUNC(e);
960         d.error=ERR_PACK(l,f,0);
961         p=ERRFN(err_get_item)(&d);
962         return((p == NULL)?NULL:p->string);
963         }

965 const char *ERR_reason_error_string(unsigned long e)
966         {
967         ERR_STRING_DATA d,*p=NULL;
968         unsigned long l,r;

970         err_fns_check();
971         l=ERR_GET_LIB(e);
972         r=ERR_GET_REASON(e);
973         d.error=ERR_PACK(l,0,r);
974         p=ERRFN(err_get_item)(&d);
975         if (!p)
976                 {
977                 d.error=ERR_PACK(0,0,r);
978                 p=ERRFN(err_get_item)(&d);
979                 }
980         return((p == NULL)?NULL:p->string);
981         }

983 void ERR_remove_thread_state(const CRYPTO_THREADID *id)
984         {
985         ERR_STATE tmp;
```



```
987         if (id)
988                 CRYPTO_THREADID_cpy(&tmp.tid, id);
989         else
990                 CRYPTO_THREADID_current(&tmp.tid);
991         err_fns_check();
992         /* thread_del_item automatically destroys the LHASH if the number of
993          * items reaches zero. */
994         ERRFN(thread_del_item)(&tmp);
995         }

997 #ifndef OPENSSL_NO_DEPRECATED
998 void ERR_remove_state(unsigned long pid)
999         {
1000        ERR_remove_thread_state(NULL);
1001        }
1002 #endif

1004 ERR_STATE *ERR_get_state(void)
1005        {
1006        static ERR_STATE fallback;
1007        ERR_STATE *ret,tmp,*tmpp=NULL;
1008        int i;
1009        CRYPTO_THREADID tid;

1011        err_fns_check();
1012        CRYPTO_THREADID_current(&tid);
1013        CRYPTO_THREADID_cpy(&tmp.tid, &tid);
1014        ret=ERRFN(thread_get_item)(&tmp);

1016        /* ret == the error state, if NULL, make a new one */
1017        if (ret == NULL)
1018                {
1019                ret=(ERR_STATE *)OPENSSL_malloc(sizeof(ERR_STATE));
1020                if (ret == NULL) return(&fallback);
1021                CRYPTO_THREADID_cpy(&ret->tid, &tid);
1022                ret->top=0;
1023                ret->bottom=0;
1024                for (i=0; i<ERR_NUM_ERRORS; i++)
1025                        {
1026                        ret->err_data[i]=NULL;
1027                        ret->err_data_flags[i]=0;
1028                        }
1029                tmpp = ERRFN(thread_set_item)(ret);
1030                /* To check if insertion failed, do a get. */
1031                if (ERRFN(thread_get_item)(ret) != ret)
1032                        {
1033                        ERR_STATE_free(ret); /* could not insert it */
1034                        return(&fallback);
1035                        }
1036                /* If a race occured in this function and we came second, tmpp
1037                 * is the first one that we just replaced. */
1038                if (tmpp)
1039                        ERR_STATE_free(tmpp);
1040                }
1041        return ret;
1042        }

1044 int ERR_get_next_error_library(void)
1045        {
1046        err_fns_check();
1047        return ERRFN(get_next_lib)();
1048        }

1050 void ERR_set_error_data(char *data, int flags)
1051        {
```

```
1052         ERR_STATE *es;
1053         int i;

1055         es=ERR_get_state();

1057         i=es->top;
1058         if (i == 0)
1059                 i=ERR_NUM_ERRORS-1;

1061         err_clear_data(es,i);
1062         es->err_data[i]=data;
1063         es->err_data_flags[i]=flags;
1064         }

1066 void ERR_add_error_data(int num, ...)
1067         {
1068         va_list args;
1069         va_start(args, num);
1070         ERR_add_error_vdata(num, args);
1071         va_end(args);
1072         }

1074 void ERR_add_error_vdata(int num, va_list args)
1075         {
1076         int i,n,s;
1077         char *str,*p,*a;

1079         s=80;
1080         str=OPENSSL_malloc(s+1);
1081         if (str == NULL) return;
1082         str[0]='\0';

1084         n=0;
1085         for (i=0; i<num; i++)
1086                 {
1087                 a=va_arg(args, char*);
1088                 /* ignore NULLs, thanks to Bob Beck <beck@obtuse.com> */
1089                 if (a != NULL)
1090                         {
1091                         n+=strlen(a);
1092                         if (n > s)
1093                                 {
1094                                 s=n+20;
1095                                 p=OPENSSL_realloc(str,s+1);
1096                                 if (p == NULL)
1097                                         {
1098                                         OPENSSL_free(str);
1099                                         return;
1100                                         }
1101                                 else
1102                                         str=p;
1103                                 }
1104                         BUF_strlcat(str,a,(size_t)s+1);
1105                         }
1106                 }
1107         ERR_set_error_data(str,ERR_TXT_MALLOCED|ERR_TXT_STRING);
1108         }

1110 int ERR_set_mark(void)
1111         {
1112         ERR_STATE *es;

1114         es=ERR_get_state();

1116         if (es->bottom == es->top) return 0;
1117         es->err_flags[es->top]|=ERR_FLAG_MARK;
```



```
1118         return 1;
1119         }

1121 int ERR_pop_to_mark(void)
1122         {
1123         ERR_STATE *es;

1125         es=ERR_get_state();

1127         while(es->bottom != es->top
1128                 && (es->err_flags[es->top] & ERR_FLAG_MARK) == 0)
1129                 {
1130                 err_clear(es,es->top);
1131                 es->top-=1;
1132                 if (es->top == -1) es->top=ERR_NUM_ERRORS-1;
1133                 }

1135         if (es->bottom == es->top) return 0;
1136         es->err_flags[es->top]&=~ERR_FLAG_MARK;
1137         return 1;
1138         }
1139 #endif /* ! codereview */
```

```
**********************************************************
    5553 Wed Aug 13 19:52:40 2014
new/usr/src/lib/openssl/libsunw_crypto/err/err_all.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/err/err_all.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/asn1.h>
#include <openssl/bn.h>
```



```
#ifndef OPENSSL_NO_EC
#include <openssl/ec.h>
#endif
#include <openssl/buffer.h>
#include <openssl/bio.h>
#ifndef OPENSSL_NO_COMP
#include <openssl/comp.h>
#endif
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_ECDSA
#include <openssl/ecdsa.h>
#endif
#ifndef OPENSSL_NO_ECDH
#include <openssl/ecdh.h>
#endif
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/pem2.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/conf.h>
#include <openssl/pkcs12.h>
#include <openssl/rand.h>
#include <openssl/dso.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include <openssl/ui.h>
#include <openssl/ocsp.h>
#include <openssl/err.h>
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif
#include <openssl/ts.h>
#ifndef OPENSSL_NO_CMS
#include <openssl/cms.h>
#endif
#ifndef OPENSSL_NO_JPAKE
#include <openssl/jpake.h>
#endif

void ERR_load_crypto_strings(void)
        {
#ifndef OPENSSL_NO_ERR
        ERR_load_ERR_strings(); /* include error strings for SYSerr */
        ERR_load_BN_strings();
#ifndef OPENSSL_NO_RSA
        ERR_load_RSA_strings();
#endif
#ifndef OPENSSL_NO_DH
        ERR_load_DH_strings();
#endif
        ERR_load_EVP_strings();
        ERR_load_BUF_strings();
        ERR_load_OBJ_strings();
        ERR_load_PEM_strings();
#ifndef OPENSSL_NO_DSA
        ERR_load_DSA_strings();
```

```
#endif
        ERR_load_X509_strings();
        ERR_load_ASN1_strings();
        ERR_load_CONF_strings();
        ERR_load_CRYPTO_strings();
#ifndef OPENSSL_NO_COMP
        ERR_load_COMP_strings();
#endif
#ifndef OPENSSL_NO_EC
        ERR_load_EC_strings();
#endif
#ifndef OPENSSL_NO_ECDSA
        ERR_load_ECDSA_strings();
#endif
#ifndef OPENSSL_NO_ECDH
        ERR_load_ECDH_strings();
#endif
        /* skip ERR_load_SSL_strings() because it is not in this library */
        ERR_load_BIO_strings();
        ERR_load_PKCS7_strings();
        ERR_load_X509V3_strings();
        ERR_load_PKCS12_strings();
        ERR_load_RAND_strings();
        ERR_load_DSO_strings();
        ERR_load_TS_strings();
#ifndef OPENSSL_NO_ENGINE
        ERR_load_ENGINE_strings();
#endif
        ERR_load_OCSP_strings();
        ERR_load_UI_strings();
#ifdef OPENSSL_FIPS
        ERR_load_FIPS_strings();
#endif
#ifndef OPENSSL_NO_CMS
        ERR_load_CMS_strings();
#endif
#ifndef OPENSSL_NO_JPAKE
        ERR_load_JPAKE_strings();
#endif
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    4428 Wed Aug 13 19:52:41 2014
new/usr/src/lib/openssl/libsunw_crypto/err/err_prn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/err/err_prn.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/lhash.h>
```



```
#include <openssl/crypto.h>
#include <openssl/buffer.h>
#include <openssl/err.h>

void ERR_print_errors_cb(int (*cb)(const char *str, size_t len, void *u),
                         void *u)
        {
        unsigned long l;
        char buf[256];
        char buf2[4096];
        const char *file,*data;
        int line,flags;
        unsigned long es;
        CRYPTO_THREADID cur;

        CRYPTO_THREADID_current(&cur);
        es=CRYPTO_THREADID_hash(&cur);
        while ((l=ERR_get_error_line_data(&file,&line,&data,&flags)) != 0)
                {
                ERR_error_string_n(l, buf, sizeof buf);
                BIO_snprintf(buf2, sizeof(buf2), "%lu:%s:%s:%d:%s\n", es, buf,
                        file, line, (flags & ERR_TXT_STRING) ? data : "");
                if (cb(buf2, strlen(buf2), u) <= 0)
                        break; /* abort outputting the error report */
                }
        }

#ifndef OPENSSL_NO_FP_API
static int print_fp(const char *str, size_t len, void *fp)
        {
        BIO bio;

        BIO_set(&bio,BIO_s_file());
        BIO_set_fp(&bio,fp,BIO_NOCLOSE);

        return BIO_printf(&bio, "%s", str);
        }
void ERR_print_errors_fp(FILE *fp)
        {
        ERR_print_errors_cb(print_fp, fp);
        }
#endif

static int print_bio(const char *str, size_t len, void *bp)
        {
        return BIO_write((BIO *)bp, str, len);
        }
void ERR_print_errors(BIO *bp)
        {
        ERR_print_errors_cb(print_bio, bp);
        }
#endif /* ! codereview */
```

```
**********************************************************
   14420 Wed Aug 13 19:52:41 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/bio_b64.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/bio_b64.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include <openssl/evp.h>

static int b64_write(BIO *h, const char *buf, int num);
static int b64_read(BIO *h, char *buf, int size);
static int b64_puts(BIO *h, const char *str);
/*static int b64_gets(BIO *h, char *str, int size); */
static long b64_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int b64_new(BIO *h);
static int b64_free(BIO *data);
static long b64_callback_ctrl(BIO *h,int cmd,bio_info_cb *fp);
#define B64_BLOCK_SIZE  1024
#define B64_BLOCK_SIZE2 768
#define B64_NONE        0
#define B64_ENCODE      1
#define B64_DECODE      2

typedef struct b64_struct
        {
        /*BIO *bio; moved to the BIO structure */
        int buf_len;
        int buf_off;
        int tmp_len;            /* used to find the start when decoding */
        int tmp_nl;             /* If true, scan until '\n' */
        int encode;
        int start;              /* have we started decoding yet? */
        int cont;               /* <= 0 when finished */
        EVP_ENCODE_CTX base64;
        char buf[EVP_ENCODE_LENGTH(B64_BLOCK_SIZE)+10];
        char tmp[B64_BLOCK_SIZE];
        } BIO_B64_CTX;

static BIO_METHOD methods_b64=
        {
        BIO_TYPE_BASE64,"base64 encoding",
        b64_write,
        b64_read,
        b64_puts,
        NULL, /* b64_gets, */
        b64_ctrl,
        b64_new,
        b64_free,
        b64_callback_ctrl,
        };

BIO_METHOD *BIO_f_base64(void)
        {
        return(&methods_b64);
        }

static int b64_new(BIO *bi)
        {
        BIO_B64_CTX *ctx;

        ctx=(BIO_B64_CTX *)OPENSSL_malloc(sizeof(BIO_B64_CTX));
        if (ctx == NULL) return(0);

        ctx->buf_len=0;
        ctx->tmp_len=0;
        ctx->tmp_nl=0;
        ctx->buf_off=0;
        ctx->cont=1;
        ctx->start=1;
        ctx->encode=0;

        bi->init=1;
```

```
128         bi->ptr=(char *)ctx;
129         bi->flags=0;
130         bi->num = 0;
131         return(1);
132         }

134 static int b64_free(BIO *a)
135         {
136         if (a == NULL) return(0);
137         OPENSSL_free(a->ptr);
138         a->ptr=NULL;
139         a->init=0;
140         a->flags=0;
141         return(1);
142         }

144 static int b64_read(BIO *b, char *out, int outl)
145         {
146         int ret=0,i,ii,j,k,x,n,num,ret_code=0;
147         BIO_B64_CTX *ctx;
148         unsigned char *p,*q;

150         if (out == NULL) return(0);
151         ctx=(BIO_B64_CTX *)b->ptr;

153         if ((ctx == NULL) || (b->next_bio == NULL)) return(0);

155         BIO_clear_retry_flags(b);

157         if (ctx->encode != B64_DECODE)
158                 {
159                 ctx->encode=B64_DECODE;
160                 ctx->buf_len=0;
161                 ctx->buf_off=0;
162                 ctx->tmp_len=0;
163                 EVP_DecodeInit(&(ctx->base64));
164                 }

166         /* First check if there are bytes decoded/encoded */
167         if (ctx->buf_len > 0)
168                 {
169                 OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
170                 i=ctx->buf_len-ctx->buf_off;
171                 if (i > outl) i=outl;
172                 OPENSSL_assert(ctx->buf_off+i < (int)sizeof(ctx->buf));
173                 memcpy(out,&(ctx->buf[ctx->buf_off]),i);
174                 ret=i;
175                 out+=i;
176                 outl-=i;
177                 ctx->buf_off+=i;
178                 if (ctx->buf_len == ctx->buf_off)
179                         {
180                         ctx->buf_len=0;
181                         ctx->buf_off=0;
182                         }
183                 }

185         /* At this point, we have room of outl bytes and an empty
186          * buffer, so we should read in some more. */

188         ret_code=0;
189         while (outl > 0)
190                 {
191                 if (ctx->cont <= 0)
192                         break;
```



```
194                 i=BIO_read(b->next_bio,&(ctx->tmp[ctx->tmp_len]),
195                         B64_BLOCK_SIZE-ctx->tmp_len);

197                 if (i <= 0)
198                         {
199                         ret_code=i;

201                         /* Should we continue next time we are called? */
202                         if (!BIO_should_retry(b->next_bio))
203                                 {
204                                 ctx->cont=i;
205                                 /* If buffer empty break */
206                                 if(ctx->tmp_len == 0)
207                                         break;
208                                 /* Fall through and process what we have */
209                                 else
210                                         i = 0;
211                                 }
212                         /* else we retry and add more data to buffer */
213                         else
214                                 break;
215                         }
216                 i+=ctx->tmp_len;
217                 ctx->tmp_len = i;

219                 /* We need to scan, a line at a time until we
220                  * have a valid line if we are starting. */
221                 if (ctx->start && (BIO_get_flags(b) & BIO_FLAGS_BASE64_NO_NL))
222                         {
223                         /* ctx->start=1; */
224                         ctx->tmp_len=0;
225                         }
226                 else if (ctx->start)
227                         {
228                         q=p=(unsigned char *)ctx->tmp;
229                         num = 0;
230                         for (j=0; j<i; j++)
231                                 {
232                                 if (*(q++) != '\n') continue;

234                                 /* due to a previous very long line,
235                                  * we need to keep on scanning for a '\n'
236                                  * before we even start looking for
237                                  * base64 encoded stuff. */
238                                 if (ctx->tmp_nl)
239                                         {
240                                         p=q;
241                                         ctx->tmp_nl=0;
242                                         continue;
243                                         }

245                                 k=EVP_DecodeUpdate(&(ctx->base64),
246                                         (unsigned char *)ctx->buf,
247                                         &num,p,q-p);
248                                 if ((k <= 0) && (num == 0) && (ctx->start))
249                                         EVP_DecodeInit(&ctx->base64);
250                                 else
251                                         {
252                                         if (p != (unsigned char *)
253                                                 &(ctx->tmp[0]))
254                                                 {
255                                                 i-=(p- (unsigned char *)
256                                                         &(ctx->tmp[0]));
257                                                 for (x=0; x < i; x++)
258                                                         ctx->tmp[x]=p[x];
259                                                 }
```

```
260                                     EVP_DecodeInit(&ctx->base64);
261                                     ctx->start=0;
262                                     break;
263                                     }
264                             p=q;
265                             }

267                     /* we fell off the end without starting */
268                     if ((j == i) && (num == 0))
269                             {
270                             /* Is this is one long chunk?, if so, keep on
271                              * reading until a new line. */
272                             if (p == (unsigned char *)&(ctx->tmp[0]))
273                                     {
274                                     /* Check buffer full */
275                                     if (i == B64_BLOCK_SIZE)
276                                             {
277                                             ctx->tmp_nl=1;
278                                             ctx->tmp_len=0;
279                                             }
280                                     }
281                             else if (p != q) /* finished on a '\n' */
282                                     {
283                                     n=q-p;
284                                     for (ii=0; ii<n; ii++)
285                                             ctx->tmp[ii]=p[ii];
286                                     ctx->tmp_len=n;
287                                     }
288                             /* else finished on a '\n' */
289                             continue;
290                             }
291                     else
292                     {
293                             ctx->tmp_len=0;
294                     }
295             }
296             else if ((i < B64_BLOCK_SIZE) && (ctx->cont > 0))
297             {
298                     /* If buffer isn't full and we can retry then
299                      * restart to read in more data.
300                      */
301                     continue;
302             }

304             if (BIO_get_flags(b) & BIO_FLAGS_BASE64_NO_NL)
305                     {
306                     int z,jj;

308 #if 0
309                     jj=(i>>2)<<2;
310 #else
311                     jj = i & ~3; /* process per 4 */
312 #endif
313                     z=EVP_DecodeBlock((unsigned char *)ctx->buf,
314                             (unsigned char *)ctx->tmp,jj);
315                     if (jj > 2)
316                             {
317                             if (ctx->tmp[jj-1] == '=')
318                                     {
319                                     z--;
320                                     if (ctx->tmp[jj-2] == '=')
321                                             z--;
322                                     }
323                             }
324                     /* z is now number of output bytes and jj is the
325                      * number consumed */
```



```
326                     if (jj != i)
327                             {
328                             memmove(ctx->tmp, &ctx->tmp[jj], i-jj);
329                             ctx->tmp_len=i-jj;
330                             }
331                     ctx->buf_len=0;
332                     if (z > 0)
333                             {
334                             ctx->buf_len=z;
335                             }
336                     i=z;
337                     }
338             else
339                     {
340                     i=EVP_DecodeUpdate(&(ctx->base64),
341                             (unsigned char *)ctx->buf,&ctx->buf_len,
342                             (unsigned char *)ctx->tmp,i);
343                     ctx->tmp_len = 0;
344                     }
345             ctx->buf_off=0;
346             if (i < 0)
347                     {
348                     ret_code=0;
349                     ctx->buf_len=0;
350                     break;
351                     }

353             if (ctx->buf_len <= outl)
354                     i=ctx->buf_len;
355             else
356                     i=outl;

358             memcpy(out,ctx->buf,i);
359             ret+=i;
360             ctx->buf_off=i;
361             if (ctx->buf_off == ctx->buf_len)
362                     {
363                     ctx->buf_len=0;
364                     ctx->buf_off=0;
365                     }
366             outl-=i;
367             out+=i;
368             }
369         /* BIO_clear_retry_flags(b); */
370         BIO_copy_next_retry(b);
371         return((ret == 0)?ret_code:ret);
372         }

374 static int b64_write(BIO *b, const char *in, int inl)
375         {
376         int ret=0;
377         int n;
378         int i;
379         BIO_B64_CTX *ctx;

381         ctx=(BIO_B64_CTX *)b->ptr;
382         BIO_clear_retry_flags(b);

384         if (ctx->encode != B64_ENCODE)
385                 {
386                 ctx->encode=B64_ENCODE;
387                 ctx->buf_len=0;
388                 ctx->buf_off=0;
389                 ctx->tmp_len=0;
390                 EVP_EncodeInit(&(ctx->base64));
391                 }
```

```
393         OPENSSL_assert(ctx->buf_off < (int)sizeof(ctx->buf));
394         OPENSSL_assert(ctx->buf_len <= (int)sizeof(ctx->buf));
395         OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
396         n=ctx->buf_len-ctx->buf_off;
397         while (n > 0)
398                 {
399                 i=BIO_write(b->next_bio,&(ctx->buf[ctx->buf_off]),n);
400                 if (i <= 0)
401                         {
402                         BIO_copy_next_retry(b);
403                         return(i);
404                         }
405                 OPENSSL_assert(i <= n);
406                 ctx->buf_off+=i;
407                 OPENSSL_assert(ctx->buf_off <= (int)sizeof(ctx->buf));
408                 OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
409                 n-=i;
410                 }
411         /* at this point all pending data has been written */
412         ctx->buf_off=0;
413         ctx->buf_len=0;

415         if ((in == NULL) || (inl <= 0)) return(0);

417         while (inl > 0)
418                 {
419                 n=(inl > B64_BLOCK_SIZE)?B64_BLOCK_SIZE:inl;

421                 if (BIO_get_flags(b) & BIO_FLAGS_BASE64_NO_NL)
422                         {
423                         if (ctx->tmp_len > 0)
424                                 {
425                                 OPENSSL_assert(ctx->tmp_len <= 3);
426                                 n=3-ctx->tmp_len;
427                                 /* There's a theoretical possibility for this */
428                                 if (n > inl)
429                                         n=inl;
430                                 memcpy(&(ctx->tmp[ctx->tmp_len]),in,n);
431                                 ctx->tmp_len+=n;
432                                 ret += n;
433                                 if (ctx->tmp_len < 3)
434                                         break;
435                                 ctx->buf_len=EVP_EncodeBlock((unsigned char *)ct
436                                 OPENSSL_assert(ctx->buf_len <= (int)sizeof(ctx->
437                                 OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
438                                 /* Since we're now done using the temporary
439                                    buffer, the length should be 0'd */
440                                 ctx->tmp_len=0;
441                                 }
442                         else
443                                 {
444                                 if (n < 3)
445                                         {
446                                         memcpy(ctx->tmp,in,n);
447                                         ctx->tmp_len=n;
448                                         ret += n;
449                                         break;
450                                         }
451                                 n-=n%3;
452                                 ctx->buf_len=EVP_EncodeBlock((unsigned char *)ct
453                                 OPENSSL_assert(ctx->buf_len <= (int)sizeof(ctx->
454                                 OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
455                                 ret += n;
456                                 }
457                         }
```



```
458                 else
459                         {
460                         EVP_EncodeUpdate(&(ctx->base64),
461                                 (unsigned char *)ctx->buf,&ctx->buf_len,
462                                 (unsigned char *)in,n);
463                         OPENSSL_assert(ctx->buf_len <= (int)sizeof(ctx->buf));
464                         OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
465                         ret += n;
466                         }
467                 inl-=n;
468                 in+=n;

470                 ctx->buf_off=0;
471                 n=ctx->buf_len;
472                 while (n > 0)
473                         {
474                         i=BIO_write(b->next_bio,&(ctx->buf[ctx->buf_off]),n);
475                         if (i <= 0)
476                                 {
477                                 BIO_copy_next_retry(b);
478                                 return((ret == 0)?i:ret);
479                                 }
480                         OPENSSL_assert(i <= n);
481                         n-=i;
482                         ctx->buf_off+=i;
483                         OPENSSL_assert(ctx->buf_off <= (int)sizeof(ctx->buf));
484                         OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
485                         }
486                 ctx->buf_len=0;
487                 ctx->buf_off=0;
488                 }
489         return(ret);
490         }

492 static long b64_ctrl(BIO *b, int cmd, long num, void *ptr)
493         {
494         BIO_B64_CTX *ctx;
495         long ret=1;
496         int i;

498         ctx=(BIO_B64_CTX *)b->ptr;

500         switch (cmd)
501                 {
502         case BIO_CTRL_RESET:
503                 ctx->cont=1;
504                 ctx->start=1;
505                 ctx->encode=B64_NONE;
506                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
507                 break;
508         case BIO_CTRL_EOF:      /* More to read */
509                 if (ctx->cont <= 0)
510                         ret=1;
511                 else
512                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
513                 break;
514         case BIO_CTRL_WPENDING: /* More to write in buffer */
515                 OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
516                 ret=ctx->buf_len-ctx->buf_off;
517                 if ((ret == 0) && (ctx->encode != B64_NONE)
518                         && (ctx->base64.num != 0))
519                         ret=1;
520                 else if (ret <= 0)
521                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
522                 break;
523         case BIO_CTRL_PENDING: /* More to read in buffer */
```

```
524                 OPENSSL_assert(ctx->buf_len >= ctx->buf_off);
525                 ret=ctx->buf_len-ctx->buf_off;
526                 if (ret <= 0)
527                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
528                 break;
529         case BIO_CTRL_FLUSH:
530                 /* do a final write */
531 again:
532                 while (ctx->buf_len != ctx->buf_off)
533                         {
534                         i=b64_write(b,NULL,0);
535                         if (i < 0)
536                                 return i;
537                         }
538                 if (BIO_get_flags(b) & BIO_FLAGS_BASE64_NO_NL)
539                         {
540                         if (ctx->tmp_len != 0)
541                                 {
542                                 ctx->buf_len=EVP_EncodeBlock(
543                                         (unsigned char *)ctx->buf,
544                                         (unsigned char *)ctx->tmp,
545                                         ctx->tmp_len);
546                                 ctx->buf_off=0;
547                                 ctx->tmp_len=0;
548                                 goto again;
549                                 }
550                         }
551                 else if (ctx->encode != B64_NONE && ctx->base64.num != 0)
552                         {
553                         ctx->buf_off=0;
554                         EVP_EncodeFinal(&(ctx->base64),
555                                 (unsigned char *)ctx->buf,
556                                 &(ctx->buf_len));
557                         /* push out the bytes */
558                         goto again;
559                         }
560                 /* Finally flush the underlying BIO */
561                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
562                 break;

564         case BIO_C_DO_STATE_MACHINE:
565                 BIO_clear_retry_flags(b);
566                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
567                 BIO_copy_next_retry(b);
568                 break;

570         case BIO_CTRL_DUP:
571                 break;
572         case BIO_CTRL_INFO:
573         case BIO_CTRL_GET:
574         case BIO_CTRL_SET:
575         default:
576                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
577                 break;
578                 }
579         return(ret);
580         }

582 static long b64_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
583         {
584         long ret=1;

586         if (b->next_bio == NULL) return(0);
587         switch (cmd)
588                 {
589         default:
```



```
590                 ret=BIO_callback_ctrl(b->next_bio,cmd,fp);
591                 break;
592                 }
593         return(ret);
594         }

596 static int b64_puts(BIO *b, const char *str)
597         {
598         return b64_write(b,str,strlen(str));
599         }
600 #endif /* ! codereview */
```

```
**********************************************************
   10737 Wed Aug 13 19:52:41 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/bio_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/bio_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include <openssl/evp.h>

static int enc_write(BIO *h, const char *buf, int num);
static int enc_read(BIO *h, char *buf, int size);
/*static int enc_puts(BIO *h, const char *str); */
/*static int enc_gets(BIO *h, char *str, int size); */
static long enc_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int enc_new(BIO *h);
static int enc_free(BIO *data);
static long enc_callback_ctrl(BIO *h, int cmd, bio_info_cb *fps);
#define ENC_BLOCK_SIZE  (1024*4)
#define BUF_OFFSET      (EVP_MAX_BLOCK_LENGTH*2)

typedef struct enc_struct
        {
        int buf_len;
        int buf_off;
        int cont;               /* <= 0 when finished */
        int finished;
        int ok;                 /* bad decrypt */
        EVP_CIPHER_CTX cipher;
        /* buf is larger than ENC_BLOCK_SIZE because EVP_DecryptUpdate
         * can return up to a block more data than is presented to it
         */
        char buf[ENC_BLOCK_SIZE+BUF_OFFSET+2];
        } BIO_ENC_CTX;

static BIO_METHOD methods_enc=
        {
        BIO_TYPE_CIPHER,"cipher",
        enc_write,
        enc_read,
        NULL, /* enc_puts, */
        NULL, /* enc_gets, */
        enc_ctrl,
        enc_new,
        enc_free,
        enc_callback_ctrl,
        };

BIO_METHOD *BIO_f_cipher(void)
        {
        return(&methods_enc);
        }

static int enc_new(BIO *bi)
        {
        BIO_ENC_CTX *ctx;

        ctx=(BIO_ENC_CTX *)OPENSSL_malloc(sizeof(BIO_ENC_CTX));
        if (ctx == NULL) return(0);
        EVP_CIPHER_CTX_init(&ctx->cipher);

        ctx->buf_len=0;
        ctx->buf_off=0;
        ctx->cont=1;
        ctx->finished=0;
        ctx->ok=1;

        bi->init=0;
        bi->ptr=(char *)ctx;
        bi->flags=0;
        return(1);
        }
```

```
128 static int enc_free(BIO *a)
129         {
130         BIO_ENC_CTX *b;

132         if (a == NULL) return(0);
133         b=(BIO_ENC_CTX *)a->ptr;
134         EVP_CIPHER_CTX_cleanup(&(b->cipher));
135         OPENSSL_cleanse(a->ptr,sizeof(BIO_ENC_CTX));
136         OPENSSL_free(a->ptr);
137         a->ptr=NULL;
138         a->init=0;
139         a->flags=0;
140         return(1);
141         }

143 static int enc_read(BIO *b, char *out, int outl)
144         {
145         int ret=0,i;
146         BIO_ENC_CTX *ctx;

148         if (out == NULL) return(0);
149         ctx=(BIO_ENC_CTX *)b->ptr;

151         if ((ctx == NULL) || (b->next_bio == NULL)) return(0);

153         /* First check if there are bytes decoded/encoded */
154         if (ctx->buf_len > 0)
155                 {
156                 i=ctx->buf_len-ctx->buf_off;
157                 if (i > outl) i=outl;
158                 memcpy(out,&(ctx->buf[ctx->buf_off]),i);
159                 ret=i;
160                 out+=i;
161                 outl-=i;
162                 ctx->buf_off+=i;
163                 if (ctx->buf_len == ctx->buf_off)
164                         {
165                         ctx->buf_len=0;
166                         ctx->buf_off=0;
167                         }
168                 }

170         /* At this point, we have room of outl bytes and an empty
171          * buffer, so we should read in some more. */

173         while (outl > 0)
174                 {
175                 if (ctx->cont <= 0) break;

177                 /* read in at IV offset, read the EVP_Cipher
178                  * documentation about why */
179                 i=BIO_read(b->next_bio,&(ctx->buf[BUF_OFFSET]),ENC_BLOCK_SIZE);

181                 if (i <= 0)
182                         {
183                         /* Should be continue next time we are called? */
184                         if (!BIO_should_retry(b->next_bio))
185                                 {
186                                 ctx->cont=i;
187                                 i=EVP_CipherFinal_ex(&(ctx->cipher),
188                                         (unsigned char *)ctx->buf,
189                                         &(ctx->buf_len));
190                                 ctx->ok=i;
191                                 ctx->buf_off=0;
192                                 }
193                         else
```



```
194                                 {
195                                 ret=(ret == 0)?i:ret;
196                                 break;
197                                 }
198                         }
199                 else
200                         {
201                         EVP_CipherUpdate(&(ctx->cipher),
202                                 (unsigned char *)ctx->buf,&ctx->buf_len,
203                                 (unsigned char *)&(ctx->buf[BUF_OFFSET]),i);
204                         ctx->cont=1;
205                         /* Note: it is possible for EVP_CipherUpdate to
206                          * decrypt zero bytes because this is or looks like
207                          * the final block: if this happens we should retry
208                          * and either read more data or decrypt the final
209                          * block
210                          */
211                         if(ctx->buf_len == 0) continue;
212                         }

214                 if (ctx->buf_len <= outl)
215                         i=ctx->buf_len;
216                 else
217                         i=outl;
218                 if (i <= 0) break;
219                 memcpy(out,ctx->buf,i);
220                 ret+=i;
221                 ctx->buf_off=i;
222                 outl-=i;
223                 out+=i;
224                 }

226         BIO_clear_retry_flags(b);
227         BIO_copy_next_retry(b);
228         return((ret == 0)?ctx->cont:ret);
229         }

231 static int enc_write(BIO *b, const char *in, int inl)
232         {
233         int ret=0,n,i;
234         BIO_ENC_CTX *ctx;

236         ctx=(BIO_ENC_CTX *)b->ptr;
237         ret=inl;

239         BIO_clear_retry_flags(b);
240         n=ctx->buf_len-ctx->buf_off;
241         while (n > 0)
242                 {
243                 i=BIO_write(b->next_bio,&(ctx->buf[ctx->buf_off]),n);
244                 if (i <= 0)
245                         {
246                         BIO_copy_next_retry(b);
247                         return(i);
248                         }
249                 ctx->buf_off+=i;
250                 n-=i;
251                 }
252         /* at this point all pending data has been written */

254         if ((in == NULL) || (inl <= 0)) return(0);

256         ctx->buf_off=0;
257         while (inl > 0)
258                 {
259                 n=(inl > ENC_BLOCK_SIZE)?ENC_BLOCK_SIZE:inl;
```

```
260                 EVP_CipherUpdate(&(ctx->cipher),
261                         (unsigned char *)ctx->buf,&ctx->buf_len,
262                         (unsigned char *)in,n);
263                 inl-=n;
264                 in+=n;

266                 ctx->buf_off=0;
267                 n=ctx->buf_len;
268                 while (n > 0)
269                         {
270                         i=BIO_write(b->next_bio,&(ctx->buf[ctx->buf_off]),n);
271                         if (i <= 0)
272                                 {
273                                 BIO_copy_next_retry(b);
274                                 return (ret == inl) ? i : ret - inl;
275                                 }
276                         n-=i;
277                         ctx->buf_off+=i;
278                         }
279                 ctx->buf_len=0;
280                 ctx->buf_off=0;
281                 }
282         BIO_copy_next_retry(b);
283         return(ret);
284         }

286 static long enc_ctrl(BIO *b, int cmd, long num, void *ptr)
287         {
288         BIO *dbio;
289         BIO_ENC_CTX *ctx,*dctx;
290         long ret=1;
291         int i;
292         EVP_CIPHER_CTX **c_ctx;

294         ctx=(BIO_ENC_CTX *)b->ptr;

296         switch (cmd)
297                 {
298         case BIO_CTRL_RESET:
299                 ctx->ok=1;
300                 ctx->finished=0;
301                 EVP_CipherInit_ex(&(ctx->cipher),NULL,NULL,NULL,NULL,
302                         ctx->cipher.encrypt);
303                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
304                 break;
305         case BIO_CTRL_EOF:      /* More to read */
306                 if (ctx->cont <= 0)
307                         ret=1;
308                 else
309                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
310                 break;
311         case BIO_CTRL_WPENDING:
312                 ret=ctx->buf_len-ctx->buf_off;
313                 if (ret <= 0)
314                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
315                 break;
316         case BIO_CTRL_PENDING: /* More to read in buffer */
317                 ret=ctx->buf_len-ctx->buf_off;
318                 if (ret <= 0)
319                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
320                 break;
321         case BIO_CTRL_FLUSH:
322                 /* do a final write */
323 again:
324                 while (ctx->buf_len != ctx->buf_off)
325                         {
```



```
326                         i=enc_write(b,NULL,0);
327                         if (i < 0)
328                                 return i;
329                         }

331                 if (!ctx->finished)
332                         {
333                         ctx->finished=1;
334                         ctx->buf_off=0;
335                         ret=EVP_CipherFinal_ex(&(ctx->cipher),
336                                 (unsigned char *)ctx->buf,
337                                 &(ctx->buf_len));
338                         ctx->ok=(int)ret;
339                         if (ret <= 0) break;

341                         /* push out the bytes */
342                         goto again;
343                         }

345                 /* Finally flush the underlying BIO */
346                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
347                 break;
348         case BIO_C_GET_CIPHER_STATUS:
349                 ret=(long)ctx->ok;
350                 break;
351         case BIO_C_DO_STATE_MACHINE:
352                 BIO_clear_retry_flags(b);
353                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
354                 BIO_copy_next_retry(b);
355                 break;
356         case BIO_C_GET_CIPHER_CTX:
357                 c_ctx=(EVP_CIPHER_CTX **)ptr;
358                 (*c_ctx)= &(ctx->cipher);
359                 b->init=1;
360                 break;
361         case BIO_CTRL_DUP:
362                 dbio=(BIO *)ptr;
363                 dctx=(BIO_ENC_CTX *)dbio->ptr;
364                 EVP_CIPHER_CTX_init(&dctx->cipher);
365                 ret = EVP_CIPHER_CTX_copy(&dctx->cipher,&ctx->cipher);
366                 if (ret)
367                         dbio->init=1;
368                 break;
369         default:
370                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
371                 break;
372                 }
373         return(ret);
374         }

376 static long enc_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
377         {
378         long ret=1;

380         if (b->next_bio == NULL) return(0);
381         switch (cmd)
382                 {
383         default:
384                 ret=BIO_callback_ctrl(b->next_bio,cmd,fp);
385                 break;
386                 }
387         return(ret);
388         }

390 /*
391 void BIO_set_cipher_ctx(b,c)
```

```
392 BIO *b;
393 EVP_CIPHER_ctx *c;
394         {
395         if (b == NULL) return;

397         if ((b->callback != NULL) &&
398                 (b->callback(b,BIO_CB_CTRL,(char *)c,BIO_CTRL_SET,e,0L) <= 0))
399                 return;

401         b->init=1;
402         ctx=(BIO_ENC_CTX *)b->ptr;
403         memcpy(ctx->cipher,c,sizeof(EVP_CIPHER_CTX));

405         if (b->callback != NULL)
406                 b->callback(b,BIO_CB_CTRL,(char *)c,BIO_CTRL_SET,e,1L);
407         }
408 */

410 void BIO_set_cipher(BIO *b, const EVP_CIPHER *c, const unsigned char *k,
411              const unsigned char *i, int e)
412         {
413         BIO_ENC_CTX *ctx;

415         if (b == NULL) return;

417         if ((b->callback != NULL) &&
418                 (b->callback(b,BIO_CB_CTRL,(const char *)c,BIO_CTRL_SET,e,0L) <=
419                 return;

421         b->init=1;
422         ctx=(BIO_ENC_CTX *)b->ptr;
423         EVP_CipherInit_ex(&(ctx->cipher),c,NULL, k,i,e);

425         if (b->callback != NULL)
426                 b->callback(b,BIO_CB_CTRL,(const char *)c,BIO_CTRL_SET,e,1L);
427         }
428 #endif /* ! codereview */
```

```
**********************************************************
    6853 Wed Aug 13 19:52:41 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/bio_md.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/bio_md.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include <openssl/evp.h>

/* BIO_put and BIO_get both add to the digest,
 * BIO_gets returns the digest */

static int md_write(BIO *h, char const *buf, int num);
static int md_read(BIO *h, char *buf, int size);
/*static int md_puts(BIO *h, const char *str); */
static int md_gets(BIO *h, char *str, int size);
static long md_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int md_new(BIO *h);
static int md_free(BIO *data);
static long md_callback_ctrl(BIO *h,int cmd,bio_info_cb *fp);

static BIO_METHOD methods_md=
        {
        BIO_TYPE_MD,"message digest",
        md_write,
        md_read,
        NULL, /* md_puts, */
        md_gets,
        md_ctrl,
        md_new,
        md_free,
        md_callback_ctrl,
        };

BIO_METHOD *BIO_f_md(void)
        {
        return(&methods_md);
        }

static int md_new(BIO *bi)
        {
        EVP_MD_CTX *ctx;

        ctx=EVP_MD_CTX_create();
        if (ctx == NULL) return(0);

        bi->init=0;
        bi->ptr=(char *)ctx;
        bi->flags=0;
        return(1);
        }

static int md_free(BIO *a)
        {
        if (a == NULL) return(0);
        EVP_MD_CTX_destroy(a->ptr);
        a->ptr=NULL;
        a->init=0;
        a->flags=0;
        return(1);
        }

static int md_read(BIO *b, char *out, int outl)
        {
        int ret=0;
        EVP_MD_CTX *ctx;

        if (out == NULL) return(0);
        ctx=b->ptr;

        if ((ctx == NULL) || (b->next_bio == NULL)) return(0);
```

```
128         ret=BIO_read(b->next_bio,out,outl);
129         if (b->init)
130                 {
131                 if (ret > 0)
132                         {
133                         if (EVP_DigestUpdate(ctx,(unsigned char *)out,
134                                 (unsigned int)ret)<=0) return (-1);
135                         }
136                 }
137         BIO_clear_retry_flags(b);
138         BIO_copy_next_retry(b);
139         return(ret);
140         }

142 static int md_write(BIO *b, const char *in, int inl)
143         {
144         int ret=0;
145         EVP_MD_CTX *ctx;

147         if ((in == NULL) || (inl <= 0)) return(0);
148         ctx=b->ptr;

150         if ((ctx != NULL) && (b->next_bio != NULL))
151                 ret=BIO_write(b->next_bio,in,inl);
152         if (b->init)
153                 {
154                 if (ret > 0)
155                         {
156                         if (!EVP_DigestUpdate(ctx,(const unsigned char *)in,
157                                 (unsigned int)ret))
158                                 {
159                                 BIO_clear_retry_flags(b);
160                                 return 0;
161                                 }
162                         }
163                 }
164         if(b->next_bio != NULL)
165                 {
166                 BIO_clear_retry_flags(b);
167                 BIO_copy_next_retry(b);
168                 }
169         return(ret);
170         }

172 static long md_ctrl(BIO *b, int cmd, long num, void *ptr)
173         {
174         EVP_MD_CTX *ctx,*dctx,**pctx;
175         const EVP_MD **ppmd;
176         EVP_MD *md;
177         long ret=1;
178         BIO *dbio;

180         ctx=b->ptr;

182         switch (cmd)
183                 {
184         case BIO_CTRL_RESET:
185                 if (b->init)
186                         ret = EVP_DigestInit_ex(ctx,ctx->digest, NULL);
187                 else
188                         ret=0;
189                 if (ret > 0)
190                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
191                 break;
192         case BIO_C_GET_MD:
193                 if (b->init)
```



```
194                         {
195                         ppmd=ptr;
196                         *ppmd=ctx->digest;
197                         }
198                 else
199                         ret=0;
200                 break;
201         case BIO_C_GET_MD_CTX:
202                 pctx=ptr;
203                 *pctx=ctx;
204                 b->init = 1;
205                 break;
206         case BIO_C_SET_MD_CTX:
207                 if (b->init)
208                         b->ptr=ptr;
209                 else
210                         ret=0;
211                 break;
212         case BIO_C_DO_STATE_MACHINE:
213                 BIO_clear_retry_flags(b);
214                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
215                 BIO_copy_next_retry(b);
216                 break;

218         case BIO_C_SET_MD:
219                 md=ptr;
220                 ret = EVP_DigestInit_ex(ctx,md, NULL);
221                 if (ret > 0)
222                         b->init=1;
223                 break;
224         case BIO_CTRL_DUP:
225                 dbio=ptr;
226                 dctx=dbio->ptr;
227                 if (!EVP_MD_CTX_copy_ex(dctx,ctx))
228                         return 0;
229                 b->init=1;
230                 break;
231         default:
232                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
233                 break;
234                 }
235         return(ret);
236         }

238 static long md_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
239         {
240         long ret=1;

242         if (b->next_bio == NULL) return(0);
243         switch (cmd)
244                 {
245         default:
246                 ret=BIO_callback_ctrl(b->next_bio,cmd,fp);
247                 break;
248                 }
249         return(ret);
250         }

252 static int md_gets(BIO *bp, char *buf, int size)
253         {
254         EVP_MD_CTX *ctx;
255         unsigned int ret;


258         ctx=bp->ptr;
259         if (size < ctx->digest->md_size)
```

```
260                 return(0);
261         if (EVP_DigestFinal_ex(ctx,(unsigned char *)buf,&ret)<=0)
262                 return -1;

264         return((int)ret);
265         }

267 /*
268 static int md_puts(bp,str)
269 BIO *bp;
270 char *str;
271         {
272         return(-1);
273         }
274 */
275 #endif /* ! codereview */
```

```
**********************************************************
   15813 Wed Aug 13 19:52:41 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/bio_ok.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/bio_ok.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/*
        From: Arne Ansper <arne@cyber.ee>
```



```
        Why BIO_f_reliable?

        I wrote function which took BIO* as argument, read data from it
        and processed it. Then I wanted to store the input file in
        encrypted form. OK I pushed BIO_f_cipher to the BIO stack
        and everything was OK. BUT if user types wrong password
        BIO_f_cipher outputs only garbage and my function crashes. Yes
        I can and I should fix my function, but BIO_f_cipher is
        easy way to add encryption support to many existing applications
        and it's hard to debug and fix them all.

        So I wanted another BIO which would catch the incorrect passwords and
        file damages which cause garbage on BIO_f_cipher's output.

        The easy way is to push the BIO_f_md and save the checksum at
        the end of the file. However there are several problems with this
        approach:

        1) you must somehow separate checksum from actual data.
        2) you need lot's of memory when reading the file, because you
        must read to the end of the file and verify the checksum before
        letting the application to read the data.

        BIO_f_reliable tries to solve both problems, so that you can
        read and write arbitrary long streams using only fixed amount
        of memory.

        BIO_f_reliable splits data stream into blocks. Each block is prefixed
        with it's length and suffixed with it's digest. So you need only
        several Kbytes of memory to buffer single block before verifying
        it's digest.

        BIO_f_reliable goes further and adds several important capabilities:

        1) the digest of the block is computed over the whole stream
        -- so nobody can rearrange the blocks or remove or replace them.

        2) to detect invalid passwords right at the start BIO_f_reliable
        adds special prefix to the stream. In order to avoid known plain-text
        attacks this prefix is generated as follows:

                *) digest is initialized with random seed instead of
                standardized one.
                *) same seed is written to output
                *) well-known text is then hashed and the output
                of the digest is also written to output.

        reader can now read the seed from stream, hash the same string
        and then compare the digest output.

        Bad things: BIO_f_reliable knows what's going on in EVP_Digest. I
        initially wrote and tested this code on x86 machine and wrote the
        digests out in machine-dependent order :( There are people using
        this code and I cannot change this easily without making existing
        data files unreadable.

*/

#include <stdio.h>
#include <errno.h>
#include <assert.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
#include <openssl/bio.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
```

```
129 static int ok_write(BIO *h, const char *buf, int num);
130 static int ok_read(BIO *h, char *buf, int size);
131 static long ok_ctrl(BIO *h, int cmd, long arg1, void *arg2);
132 static int ok_new(BIO *h);
133 static int ok_free(BIO *data);
134 static long ok_callback_ctrl(BIO *h, int cmd, bio_info_cb *fp);

136 static int sig_out(BIO* b);
137 static int sig_in(BIO* b);
138 static int block_out(BIO* b);
139 static int block_in(BIO* b);
140 #define OK_BLOCK_SIZE   (1024*4)
141 #define OK_BLOCK_BLOCK  4
142 #define IOBS            (OK_BLOCK_SIZE+ OK_BLOCK_BLOCK+ 3*EVP_MAX_MD_SIZE)
143 #define WELLKNOWN "The quick brown fox jumped over the lazy dog's back."

145 typedef struct ok_struct
146         {
147         size_t buf_len;
148         size_t buf_off;
149         size_t buf_len_save;
150         size_t buf_off_save;
151         int cont;               /* <= 0 when finished */
152         int finished;
153         EVP_MD_CTX md;
154         int blockout;           /* output block is ready */
155         int sigio;              /* must process signature */
156         unsigned char buf[IOBS];
157         } BIO_OK_CTX;

159 static BIO_METHOD methods_ok=
160         {
161         BIO_TYPE_CIPHER,"reliable",
162         ok_write,
163         ok_read,
164         NULL, /* ok_puts, */
165         NULL, /* ok_gets, */
166         ok_ctrl,
167         ok_new,
168         ok_free,
169         ok_callback_ctrl,
170         };

172 BIO_METHOD *BIO_f_reliable(void)
173         {
174         return(&methods_ok);
175         }

177 static int ok_new(BIO *bi)
178         {
179         BIO_OK_CTX *ctx;

181         ctx=(BIO_OK_CTX *)OPENSSL_malloc(sizeof(BIO_OK_CTX));
182         if (ctx == NULL) return(0);

184         ctx->buf_len=0;
185         ctx->buf_off=0;
186         ctx->buf_len_save=0;
187         ctx->buf_off_save=0;
188         ctx->cont=1;
189         ctx->finished=0;
190         ctx->blockout= 0;
191         ctx->sigio=1;

193         EVP_MD_CTX_init(&ctx->md);
```



```
195         bi->init=0;
196         bi->ptr=(char *)ctx;
197         bi->flags=0;
198         return(1);
199         }

201 static int ok_free(BIO *a)
202         {
203         if (a == NULL) return(0);
204         EVP_MD_CTX_cleanup(&((BIO_OK_CTX *)a->ptr)->md);
205         OPENSSL_cleanse(a->ptr,sizeof(BIO_OK_CTX));
206         OPENSSL_free(a->ptr);
207         a->ptr=NULL;
208         a->init=0;
209         a->flags=0;
210         return(1);
211         }

213 static int ok_read(BIO *b, char *out, int outl)
214         {
215         int ret=0,i,n;
216         BIO_OK_CTX *ctx;

218         if (out == NULL) return(0);
219         ctx=(BIO_OK_CTX *)b->ptr;

221         if ((ctx == NULL) || (b->next_bio == NULL) || (b->init == 0)) return(0);

223         while(outl > 0)
224                 {

226                 /* copy clean bytes to output buffer */
227                 if (ctx->blockout)
228                         {
229                         i=ctx->buf_len-ctx->buf_off;
230                         if (i > outl) i=outl;
231                         memcpy(out,&(ctx->buf[ctx->buf_off]),i);
232                         ret+=i;
233                         out+=i;
234                         outl-=i;
235                         ctx->buf_off+=i;

237                         /* all clean bytes are out */
238                         if (ctx->buf_len == ctx->buf_off)
239                                 {
240                                 ctx->buf_off=0;

242                                 /* copy start of the next block into proper plac
243                                 if(ctx->buf_len_save- ctx->buf_off_save > 0)
244                                         {
245                                         ctx->buf_len= ctx->buf_len_save- ctx->bu
246                                         memmove(ctx->buf, &(ctx->buf[ctx->buf_of
247                                                                 ctx->buf_len);
248                                         }
249                                 else
250                                         {
251                                         ctx->buf_len=0;
252                                         }
253                                 ctx->blockout= 0;
254                                 }
255                         }

257                 /* output buffer full -- cancel */
258                 if (outl == 0) break;
```

```
260                 /* no clean bytes in buffer -- fill it */
261                 n=IOBS- ctx->buf_len;
262                 i=BIO_read(b->next_bio,&(ctx->buf[ctx->buf_len]),n);

264                 if (i <= 0) break;      /* nothing new */

266                 ctx->buf_len+= i;

268                 /* no signature yet -- check if we got one */
269                 if (ctx->sigio == 1)
270                         {
271                         if (!sig_in(b))
272                                 {
273                                 BIO_clear_retry_flags(b);
274                                 return 0;
275                                 }
276                         }

278                 /* signature ok -- check if we got block */
279                 if (ctx->sigio == 0)
280                         {
281                         if (!block_in(b))
282                                 {
283                                 BIO_clear_retry_flags(b);
284                                 return 0;
285                                 }
286                         }

288                 /* invalid block -- cancel */
289                 if (ctx->cont <= 0) break;

291                 }

293         BIO_clear_retry_flags(b);
294         BIO_copy_next_retry(b);
295         return(ret);
296         }

298 static int ok_write(BIO *b, const char *in, int inl)
299         {
300         int ret=0,n,i;
301         BIO_OK_CTX *ctx;

303         if (inl <= 0) return inl;

305         ctx=(BIO_OK_CTX *)b->ptr;
306         ret=inl;

308         if ((ctx == NULL) || (b->next_bio == NULL) || (b->init == 0)) return(0);

310         if(ctx->sigio && !sig_out(b))
311                 return 0;

313         do{
314                 BIO_clear_retry_flags(b);
315                 n=ctx->buf_len-ctx->buf_off;
316                 while (ctx->blockout && n > 0)
317                         {
318                         i=BIO_write(b->next_bio,&(ctx->buf[ctx->buf_off]),n);
319                         if (i <= 0)
320                                 {
321                                 BIO_copy_next_retry(b);
322                                 if(!BIO_should_retry(b))
323                                         ctx->cont= 0;
324                                 return(i);
325                                 }
```



```
326                         ctx->buf_off+=i;
327                         n-=i;
328                         }

330                 /* at this point all pending data has been written */
331                 ctx->blockout= 0;
332                 if (ctx->buf_len == ctx->buf_off)
333                         {
334                         ctx->buf_len=OK_BLOCK_BLOCK;
335                         ctx->buf_off=0;
336                         }

338                 if ((in == NULL) || (inl <= 0)) return(0);

340                 n= (inl+ ctx->buf_len > OK_BLOCK_SIZE+ OK_BLOCK_BLOCK) ?
341                         (int)(OK_BLOCK_SIZE+OK_BLOCK_BLOCK-ctx->buf_len) : inl;

343                 memcpy((unsigned char *)(&(ctx->buf[ctx->buf_len])),(unsigned ch
344                 ctx->buf_len+= n;
345                 inl-=n;
346                 in+=n;

348                 if(ctx->buf_len >= OK_BLOCK_SIZE+ OK_BLOCK_BLOCK)
349                         {
350                         if (!block_out(b))
351                                 {
352                                 BIO_clear_retry_flags(b);
353                                 return 0;
354                                 }
355                         }
356         }while(inl > 0);

358         BIO_clear_retry_flags(b);
359         BIO_copy_next_retry(b);
360         return(ret);
361         }

363 static long ok_ctrl(BIO *b, int cmd, long num, void *ptr)
364         {
365         BIO_OK_CTX *ctx;
366         EVP_MD *md;
367         const EVP_MD **ppmd;
368         long ret=1;
369         int i;

371         ctx=b->ptr;

373         switch (cmd)
374                 {
375         case BIO_CTRL_RESET:
376                 ctx->buf_len=0;
377                 ctx->buf_off=0;
378                 ctx->buf_len_save=0;
379                 ctx->buf_off_save=0;
380                 ctx->cont=1;
381                 ctx->finished=0;
382                 ctx->blockout= 0;
383                 ctx->sigio=1;
384                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
385                 break;
386         case BIO_CTRL_EOF:      /* More to read */
387                 if (ctx->cont <= 0)
388                         ret=1;
389                 else
390                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
391                 break;
```

```
392         case BIO_CTRL_PENDING: /* More to read in buffer */
393         case BIO_CTRL_WPENDING: /* More to read in buffer */
394                 ret=ctx->blockout ? ctx->buf_len-ctx->buf_off : 0;
395                 if (ret <= 0)
396                         ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
397                 break;
398         case BIO_CTRL_FLUSH:
399                 /* do a final write */
400                 if(ctx->blockout == 0)
401                         if (!block_out(b))
402                                 return 0;

404                 while (ctx->blockout)
405                         {
406                         i=ok_write(b,NULL,0);
407                         if (i < 0)
408                                 {
409                                 ret=i;
410                                 break;
411                                 }
412                         }

414                 ctx->finished=1;
415                 ctx->buf_off=ctx->buf_len=0;
416                 ctx->cont=(int)ret;

418                 /* Finally flush the underlying BIO */
419                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
420                 break;
421         case BIO_C_DO_STATE_MACHINE:
422                 BIO_clear_retry_flags(b);
423                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
424                 BIO_copy_next_retry(b);
425                 break;
426         case BIO_CTRL_INFO:
427                 ret=(long)ctx->cont;
428                 break;
429         case BIO_C_SET_MD:
430                 md=ptr;
431                 if (!EVP_DigestInit_ex(&ctx->md, md, NULL))
432                         return 0;
433                 b->init=1;
434                 break;
435         case BIO_C_GET_MD:
436                 if (b->init)
437                         {
438                         ppmd=ptr;
439                         *ppmd=ctx->md.digest;
440                         }
441                 else
442                         ret=0;
443                 break;
444         default:
445                 ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
446                 break;
447                 }
448         return(ret);
449         }

451 static long ok_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
452         {
453         long ret=1;

455         if (b->next_bio == NULL) return(0);
456         switch (cmd)
457                 {
```



```
458         default:
459                 ret=BIO_callback_ctrl(b->next_bio,cmd,fp);
460                 break;
461                 }
462         return(ret);
463         }

465 static void longswap(void *_ptr, size_t len)
466 {       const union { long one; char little; } is_endian = {1};

468         if (is_endian.little) {
469                 size_t i;
470                 unsigned char *p=_ptr,c;

472                 for(i= 0;i < len;i+= 4) {
473                         c=p[0],p[0]=p[3],p[3]=c;
474                         c=p[1],p[1]=p[2],p[2]=c;
475                 }
476         }
477 }

479 static int sig_out(BIO* b)
480         {
481         BIO_OK_CTX *ctx;
482         EVP_MD_CTX *md;

484         ctx=b->ptr;
485         md=&ctx->md;

487         if(ctx->buf_len+ 2* md->digest->md_size > OK_BLOCK_SIZE) return 1;

489         if (!EVP_DigestInit_ex(md, md->digest, NULL))
490                 goto berr;
491         /* FIXME: there's absolutely no guarantee this makes any sense at all,
492          * particularly now EVP_MD_CTX has been restructured.
493          */
494         RAND_pseudo_bytes(md->md_data, md->digest->md_size);
495         memcpy(&(ctx->buf[ctx->buf_len]), md->md_data, md->digest->md_size);
496         longswap(&(ctx->buf[ctx->buf_len]), md->digest->md_size);
497         ctx->buf_len+= md->digest->md_size;

499         if (!EVP_DigestUpdate(md, WELLKNOWN, strlen(WELLKNOWN)))
500                 goto berr;
501         if (!EVP_DigestFinal_ex(md, &(ctx->buf[ctx->buf_len]), NULL))
502                 goto berr;
503         ctx->buf_len+= md->digest->md_size;
504         ctx->blockout= 1;
505         ctx->sigio= 0;
506         return 1;
507         berr:
508         BIO_clear_retry_flags(b);
509         return 0;
510         }

512 static int sig_in(BIO* b)
513         {
514         BIO_OK_CTX *ctx;
515         EVP_MD_CTX *md;
516         unsigned char tmp[EVP_MAX_MD_SIZE];
517         int ret= 0;

519         ctx=b->ptr;
520         md=&ctx->md;

522         if((int)(ctx->buf_len-ctx->buf_off) < 2*md->digest->md_size) return 1;
```

```
524         if (!EVP_DigestInit_ex(md, md->digest, NULL))
525                 goto berr;
526         memcpy(md->md_data, &(ctx->buf[ctx->buf_off]), md->digest->md_size);
527         longswap(md->md_data, md->digest->md_size);
528         ctx->buf_off+= md->digest->md_size;

530         if (!EVP_DigestUpdate(md, WELLKNOWN, strlen(WELLKNOWN)))
531                 goto berr;
532         if (!EVP_DigestFinal_ex(md, tmp, NULL))
533                 goto berr;
534         ret= memcmp(&(ctx->buf[ctx->buf_off]), tmp, md->digest->md_size) == 0;
535         ctx->buf_off+= md->digest->md_size;
536         if(ret == 1)
537                 {
538                 ctx->sigio= 0;
539                 if(ctx->buf_len != ctx->buf_off)
540                         {
541                         memmove(ctx->buf, &(ctx->buf[ctx->buf_off]), ctx->buf_le
542                         }
543                 ctx->buf_len-= ctx->buf_off;
544                 ctx->buf_off= 0;
545                 }
546         else
547                 {
548                 ctx->cont= 0;
549                 }
550         return 1;
551         berr:
552         BIO_clear_retry_flags(b);
553         return 0;
554         }

556 static int block_out(BIO* b)
557         {
558         BIO_OK_CTX *ctx;
559         EVP_MD_CTX *md;
560         unsigned long tl;

562         ctx=b->ptr;
563         md=&ctx->md;

565         tl= ctx->buf_len- OK_BLOCK_BLOCK;
566         ctx->buf[0]=(unsigned char)(tl>>24);
567         ctx->buf[1]=(unsigned char)(tl>>16);
568         ctx->buf[2]=(unsigned char)(tl>>8);
569         ctx->buf[3]=(unsigned char)(tl);
570         if (!EVP_DigestUpdate(md,
571                 (unsigned char*) &(ctx->buf[OK_BLOCK_BLOCK]), tl))
572                 goto berr;
573         if (!EVP_DigestFinal_ex(md, &(ctx->buf[ctx->buf_len]), NULL))
574                 goto berr;
575         ctx->buf_len+= md->digest->md_size;
576         ctx->blockout= 1;
577         return 1;
578         berr:
579         BIO_clear_retry_flags(b);
580         return 0;
581         }

583 static int block_in(BIO* b)
584         {
585         BIO_OK_CTX *ctx;
586         EVP_MD_CTX *md;
587         unsigned long tl= 0;
588         unsigned char tmp[EVP_MAX_MD_SIZE];
```



```
590         ctx=b->ptr;
591         md=&ctx->md;

593         assert(sizeof(tl)>=OK_BLOCK_BLOCK);     /* always true */
594         tl =ctx->buf[0]; tl<<=8;
595         tl|=ctx->buf[1]; tl<<=8;
596         tl|=ctx->buf[2]; tl<<=8;
597         tl|=ctx->buf[3];

599         if (ctx->buf_len < tl+ OK_BLOCK_BLOCK+ md->digest->md_size) return 1;

601         if (!EVP_DigestUpdate(md,
602                         (unsigned char*) &(ctx->buf[OK_BLOCK_BLOCK]), tl))
603                 goto berr;
604         if (!EVP_DigestFinal_ex(md, tmp, NULL))
605                 goto berr;
606         if(memcmp(&(ctx->buf[tl+ OK_BLOCK_BLOCK]), tmp, md->digest->md_size) ==
607                 {
608                 /* there might be parts from next block lurking around ! */
609                 ctx->buf_off_save= tl+ OK_BLOCK_BLOCK+ md->digest->md_size;
610                 ctx->buf_len_save= ctx->buf_len;
611                 ctx->buf_off= OK_BLOCK_BLOCK;
612                 ctx->buf_len= tl+ OK_BLOCK_BLOCK;
613                 ctx->blockout= 1;
614                 }
615         else
616                 {
617                 ctx->cont= 0;
618                 }
619         return 1;
620         berr:
621         BIO_clear_retry_flags(b);
622         return 0;
623         }
624 #endif /* ! codereview */
```

```
**********************************************************
    3861 Wed Aug 13 19:52:41 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/c_all.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/c_all.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

#if 0
#undef OpenSSL_add_all_algorithms

void OpenSSL_add_all_algorithms(void)
        {
        OPENSSL_add_all_algorithms_noconf();
        }
#endif

void OPENSSL_add_all_algorithms_noconf(void)
        {
        /*
         * For the moment OPENSSL_cpuid_setup does something
         * only on IA-32, but we reserve the option for all
         * platforms...
         */
        OPENSSL_cpuid_setup();
        OpenSSL_add_all_ciphers();
        OpenSSL_add_all_digests();
#ifndef OPENSSL_NO_ENGINE
# if defined(__OpenBSD__) || defined(__FreeBSD__) || defined(HAVE_CRYPTODEV)
        ENGINE_setup_bsd_cryptodev();
# endif
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    8659 Wed Aug 13 19:52:42 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/c_allc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/c_allc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/pkcs12.h>
#include <openssl/objects.h>

void OpenSSL_add_all_ciphers(void)
        {

#ifndef OPENSSL_NO_DES
        EVP_add_cipher(EVP_des_cfb());
        EVP_add_cipher(EVP_des_cfb1());
        EVP_add_cipher(EVP_des_cfb8());
        EVP_add_cipher(EVP_des_ede_cfb());
        EVP_add_cipher(EVP_des_ede3_cfb());
        EVP_add_cipher(EVP_des_ede3_cfb1());
        EVP_add_cipher(EVP_des_ede3_cfb8());

        EVP_add_cipher(EVP_des_ofb());
        EVP_add_cipher(EVP_des_ede_ofb());
        EVP_add_cipher(EVP_des_ede3_ofb());

        EVP_add_cipher(EVP_desx_cbc());
        EVP_add_cipher_alias(SN_desx_cbc,"DESX");
        EVP_add_cipher_alias(SN_desx_cbc,"desx");

        EVP_add_cipher(EVP_des_cbc());
        EVP_add_cipher_alias(SN_des_cbc,"DES");
        EVP_add_cipher_alias(SN_des_cbc,"des");
        EVP_add_cipher(EVP_des_ede_cbc());
        EVP_add_cipher(EVP_des_ede3_cbc());
        EVP_add_cipher_alias(SN_des_ede3_cbc,"DES3");
        EVP_add_cipher_alias(SN_des_ede3_cbc,"des3");

        EVP_add_cipher(EVP_des_ecb());
        EVP_add_cipher(EVP_des_ede());
        EVP_add_cipher(EVP_des_ede3());
#endif

#ifndef OPENSSL_NO_RC4
        EVP_add_cipher(EVP_rc4());
        EVP_add_cipher(EVP_rc4_40());
#ifndef OPENSSL_NO_MD5
        EVP_add_cipher(EVP_rc4_hmac_md5());
#endif
#endif

#ifndef OPENSSL_NO_IDEA
        EVP_add_cipher(EVP_idea_ecb());
        EVP_add_cipher(EVP_idea_cfb());
        EVP_add_cipher(EVP_idea_ofb());
        EVP_add_cipher(EVP_idea_cbc());
        EVP_add_cipher_alias(SN_idea_cbc,"IDEA");
        EVP_add_cipher_alias(SN_idea_cbc,"idea");
#endif

#ifndef OPENSSL_NO_SEED
        EVP_add_cipher(EVP_seed_ecb());
        EVP_add_cipher(EVP_seed_cfb());
        EVP_add_cipher(EVP_seed_ofb());
        EVP_add_cipher(EVP_seed_cbc());
        EVP_add_cipher_alias(SN_seed_cbc,"SEED");
        EVP_add_cipher_alias(SN_seed_cbc,"seed");
#endif

#ifndef OPENSSL_NO_RC2
        EVP_add_cipher(EVP_rc2_ecb());
        EVP_add_cipher(EVP_rc2_cfb());
        EVP_add_cipher(EVP_rc2_ofb());
```

```
128         EVP_add_cipher(EVP_rc2_cbc());
129         EVP_add_cipher(EVP_rc2_40_cbc());
130         EVP_add_cipher(EVP_rc2_64_cbc());
131         EVP_add_cipher_alias(SN_rc2_cbc,"RC2");
132         EVP_add_cipher_alias(SN_rc2_cbc,"rc2");
133 #endif

135 #ifndef OPENSSL_NO_BF
136         EVP_add_cipher(EVP_bf_ecb());
137         EVP_add_cipher(EVP_bf_cfb());
138         EVP_add_cipher(EVP_bf_ofb());
139         EVP_add_cipher(EVP_bf_cbc());
140         EVP_add_cipher_alias(SN_bf_cbc,"BF");
141         EVP_add_cipher_alias(SN_bf_cbc,"bf");
142         EVP_add_cipher_alias(SN_bf_cbc,"blowfish");
143 #endif

145 #ifndef OPENSSL_NO_CAST
146         EVP_add_cipher(EVP_cast5_ecb());
147         EVP_add_cipher(EVP_cast5_cfb());
148         EVP_add_cipher(EVP_cast5_ofb());
149         EVP_add_cipher(EVP_cast5_cbc());
150         EVP_add_cipher_alias(SN_cast5_cbc,"CAST");
151         EVP_add_cipher_alias(SN_cast5_cbc,"cast");
152         EVP_add_cipher_alias(SN_cast5_cbc,"CAST-cbc");
153         EVP_add_cipher_alias(SN_cast5_cbc,"cast-cbc");
154 #endif

156 #ifndef OPENSSL_NO_RC5
157         EVP_add_cipher(EVP_rc5_32_12_16_ecb());
158         EVP_add_cipher(EVP_rc5_32_12_16_cfb());
159         EVP_add_cipher(EVP_rc5_32_12_16_ofb());
160         EVP_add_cipher(EVP_rc5_32_12_16_cbc());
161         EVP_add_cipher_alias(SN_rc5_cbc,"rc5");
162         EVP_add_cipher_alias(SN_rc5_cbc,"RC5");
163 #endif

165 #ifndef OPENSSL_NO_AES
166         EVP_add_cipher(EVP_aes_128_ecb());
167         EVP_add_cipher(EVP_aes_128_cbc());
168         EVP_add_cipher(EVP_aes_128_cfb());
169         EVP_add_cipher(EVP_aes_128_cfb1());
170         EVP_add_cipher(EVP_aes_128_cfb8());
171         EVP_add_cipher(EVP_aes_128_ofb());
172         EVP_add_cipher(EVP_aes_128_ctr());
173         EVP_add_cipher(EVP_aes_128_gcm());
174         EVP_add_cipher(EVP_aes_128_xts());
175         EVP_add_cipher_alias(SN_aes_128_cbc,"AES128");
176         EVP_add_cipher_alias(SN_aes_128_cbc,"aes128");
177         EVP_add_cipher(EVP_aes_192_ecb());
178         EVP_add_cipher(EVP_aes_192_cbc());
179         EVP_add_cipher(EVP_aes_192_cfb());
180         EVP_add_cipher(EVP_aes_192_cfb1());
181         EVP_add_cipher(EVP_aes_192_cfb8());
182         EVP_add_cipher(EVP_aes_192_ofb());
183         EVP_add_cipher(EVP_aes_192_ctr());
184         EVP_add_cipher(EVP_aes_192_gcm());
185         EVP_add_cipher_alias(SN_aes_192_cbc,"AES192");
186         EVP_add_cipher_alias(SN_aes_192_cbc,"aes192");
187         EVP_add_cipher(EVP_aes_256_ecb());
188         EVP_add_cipher(EVP_aes_256_cbc());
189         EVP_add_cipher(EVP_aes_256_cfb());
190         EVP_add_cipher(EVP_aes_256_cfb1());
191         EVP_add_cipher(EVP_aes_256_cfb8());
192         EVP_add_cipher(EVP_aes_256_ofb());
193         EVP_add_cipher(EVP_aes_256_ctr());
```



```
194         EVP_add_cipher(EVP_aes_256_gcm());
195         EVP_add_cipher(EVP_aes_256_xts());
196         EVP_add_cipher_alias(SN_aes_256_cbc,"AES256");
197         EVP_add_cipher_alias(SN_aes_256_cbc,"aes256");
198 #if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA1)
199         EVP_add_cipher(EVP_aes_128_cbc_hmac_sha1());
200         EVP_add_cipher(EVP_aes_256_cbc_hmac_sha1());
201 #endif
202 #endif

204 #ifndef OPENSSL_NO_CAMELLIA
205         EVP_add_cipher(EVP_camellia_128_ecb());
206         EVP_add_cipher(EVP_camellia_128_cbc());
207         EVP_add_cipher(EVP_camellia_128_cfb());
208         EVP_add_cipher(EVP_camellia_128_cfb1());
209         EVP_add_cipher(EVP_camellia_128_cfb8());
210         EVP_add_cipher(EVP_camellia_128_ofb());
211         EVP_add_cipher_alias(SN_camellia_128_cbc,"CAMELLIA128");
212         EVP_add_cipher_alias(SN_camellia_128_cbc,"camellia128");
213         EVP_add_cipher(EVP_camellia_192_ecb());
214         EVP_add_cipher(EVP_camellia_192_cbc());
215         EVP_add_cipher(EVP_camellia_192_cfb());
216         EVP_add_cipher(EVP_camellia_192_cfb1());
217         EVP_add_cipher(EVP_camellia_192_cfb8());
218         EVP_add_cipher(EVP_camellia_192_ofb());
219         EVP_add_cipher_alias(SN_camellia_192_cbc,"CAMELLIA192");
220         EVP_add_cipher_alias(SN_camellia_192_cbc,"camellia192");
221         EVP_add_cipher(EVP_camellia_256_ecb());
222         EVP_add_cipher(EVP_camellia_256_cbc());
223         EVP_add_cipher(EVP_camellia_256_cfb());
224         EVP_add_cipher(EVP_camellia_256_cfb1());
225         EVP_add_cipher(EVP_camellia_256_cfb8());
226         EVP_add_cipher(EVP_camellia_256_ofb());
227         EVP_add_cipher_alias(SN_camellia_256_cbc,"CAMELLIA256");
228         EVP_add_cipher_alias(SN_camellia_256_cbc,"camellia256");
229 #endif
230         }
231 #endif /* ! codereview */
```

```
**********************************************************
    4690 Wed Aug 13 19:52:42 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/c_alld.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/c_alld.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/pkcs12.h>
#include <openssl/objects.h>

void OpenSSL_add_all_digests(void)
        {
#ifndef OPENSSL_NO_MD4
        EVP_add_digest(EVP_md4());
#endif
#ifndef OPENSSL_NO_MD5
        EVP_add_digest(EVP_md5());
        EVP_add_digest_alias(SN_md5,"ssl2-md5");
        EVP_add_digest_alias(SN_md5,"ssl3-md5");
#endif
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA0)
        EVP_add_digest(EVP_sha());
#ifndef OPENSSL_NO_DSA
        EVP_add_digest(EVP_dss());
#endif
#endif
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA1)
        EVP_add_digest(EVP_sha1());
        EVP_add_digest_alias(SN_sha1,"ssl3-sha1");
        EVP_add_digest_alias(SN_sha1WithRSAEncryption,SN_sha1WithRSA);
#ifndef OPENSSL_NO_DSA
        EVP_add_digest(EVP_dss1());
        EVP_add_digest_alias(SN_dsaWithSHA1,SN_dsaWithSHA1_2);
        EVP_add_digest_alias(SN_dsaWithSHA1,"DSS1");
        EVP_add_digest_alias(SN_dsaWithSHA1,"dss1");
#endif
#ifndef OPENSSL_NO_ECDSA
        EVP_add_digest(EVP_ecdsa());
#endif
#endif
#if !defined(OPENSSL_NO_MDC2) && !defined(OPENSSL_NO_DES)
        EVP_add_digest(EVP_mdc2());
#endif
#ifndef OPENSSL_NO_RIPEMD
        EVP_add_digest(EVP_ripemd160());
        EVP_add_digest_alias(SN_ripemd160,"ripemd");
        EVP_add_digest_alias(SN_ripemd160,"rmd160");
#endif
#ifndef OPENSSL_NO_SHA256
        EVP_add_digest(EVP_sha224());
        EVP_add_digest(EVP_sha256());
#endif
#ifndef OPENSSL_NO_SHA512
        EVP_add_digest(EVP_sha384());
        EVP_add_digest(EVP_sha512());
#endif
#ifndef OPENSSL_NO_WHIRLPOOL
        EVP_add_digest(EVP_whirlpool());
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
   12742 Wed Aug 13 19:52:42 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/digest.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/digest.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
#include <openssl/evp.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

void EVP_MD_CTX_init(EVP_MD_CTX *ctx)
        {
        memset(ctx,'\0',sizeof *ctx);
        }
```

```
129 EVP_MD_CTX *EVP_MD_CTX_create(void)
130         {
131         EVP_MD_CTX *ctx=OPENSSL_malloc(sizeof *ctx);
132
133         if (ctx)
134                 EVP_MD_CTX_init(ctx);
135
136         return ctx;
137         }
138
139 int EVP_DigestInit(EVP_MD_CTX *ctx, const EVP_MD *type)
140         {
141         EVP_MD_CTX_init(ctx);
142         return EVP_DigestInit_ex(ctx, type, NULL);
143         }
144
145 int EVP_DigestInit_ex(EVP_MD_CTX *ctx, const EVP_MD *type, ENGINE *impl)
146         {
147         EVP_MD_CTX_clear_flags(ctx,EVP_MD_CTX_FLAG_CLEANED);
148 #ifndef OPENSSL_NO_ENGINE
149         /* Whether it's nice or not, "Inits" can be used on "Final"'d contexts
150          * so this context may already have an ENGINE! Try to avoid releasing
151          * the previous handle, re-querying for an ENGINE, and having a
152          * reinitialisation, when it may all be unecessary. */
153         if (ctx->engine && ctx->digest && (!type ||
154                         (type && (type->type == ctx->digest->type))))
155                 goto skip_to_init;
156         if (type)
157                 {
158                 /* Ensure an ENGINE left lying around from last time is cleared
159                  * (the previous check attempted to avoid this if the same
160                  * ENGINE and EVP_MD could be used). */
161                 if(ctx->engine)
162                         ENGINE_finish(ctx->engine);
163                 if(impl)
164                         {
165                         if (!ENGINE_init(impl))
166                                 {
167                                 EVPerr(EVP_F_EVP_DIGESTINIT_EX,EVP_R_INITIALIZAT
168                                 return 0;
169                                 }
170                         }
171                 else
172                         /* Ask if an ENGINE is reserved for this job */
173                         impl = ENGINE_get_digest_engine(type->type);
174                 if(impl)
175                         {
176                         /* There's an ENGINE for this job ... (apparently) */
177                         const EVP_MD *d = ENGINE_get_digest(impl, type->type);
178                         if(!d)
179                                 {
180                                 /* Same comment from evp_enc.c */
181                                 EVPerr(EVP_F_EVP_DIGESTINIT_EX,EVP_R_INITIALIZAT
182                                 ENGINE_finish(impl);
183                                 return 0;
184                                 }
185                         /* We'll use the ENGINE's private digest definition */
186                         type = d;
187                         /* Store the ENGINE functional reference so we know
188                          * 'type' came from an ENGINE and we need to release
189                          * it when done. */
190                         ctx->engine = impl;
191                         }
192                 else
193                         ctx->engine = NULL;
```



```
194                 }
195         else
196         if(!ctx->digest)
197                 {
198                 EVPerr(EVP_F_EVP_DIGESTINIT_EX,EVP_R_NO_DIGEST_SET);
199                 return 0;
200                 }
201 #endif
202         if (ctx->digest != type)
203                 {
204                 if (ctx->digest && ctx->digest->ctx_size)
205                         OPENSSL_free(ctx->md_data);
206                 ctx->digest=type;
207                 if (!(ctx->flags & EVP_MD_CTX_FLAG_NO_INIT) && type->ctx_size)
208                         {
209                         ctx->update = type->update;
210                         ctx->md_data=OPENSSL_malloc(type->ctx_size);
211                         if (ctx->md_data == NULL)
212                                 {
213                                 EVPerr(EVP_F_EVP_DIGESTINIT_EX,
214                                                                 ERR_R_MALLOC_FAILURE);
215                                 return 0;
216                                 }
217                         }
218                 }
219 #ifndef OPENSSL_NO_ENGINE
220 skip_to_init:
221 #endif
222         if (ctx->pctx)
223                 {
224                 int r;
225                 r = EVP_PKEY_CTX_ctrl(ctx->pctx, -1, EVP_PKEY_OP_TYPE_SIG,
226                                         EVP_PKEY_CTRL_DIGESTINIT, 0, ctx);
227                 if (r <= 0 && (r != -2))
228                         return 0;
229                 }
230         if (ctx->flags & EVP_MD_CTX_FLAG_NO_INIT)
231                 return 1;
232 #ifdef OPENSSL_FIPS
233         if (FIPS_mode())
234                 {
235                 if (FIPS_digestinit(ctx, type))
236                         return 1;
237                 OPENSSL_free(ctx->md_data);
238                 ctx->md_data = NULL;
239                 return 0;
240                 }
241 #endif
242         return ctx->digest->init(ctx);
243         }
244
245 int EVP_DigestUpdate(EVP_MD_CTX *ctx, const void *data, size_t count)
246         {
247 #ifdef OPENSSL_FIPS
248         return FIPS_digestupdate(ctx, data, count);
249 #else
250         return ctx->update(ctx,data,count);
251 #endif
252         }
253
254 /* The caller can assume that this removes any secret data from the context */
255 int EVP_DigestFinal(EVP_MD_CTX *ctx, unsigned char *md, unsigned int *size)
256         {
257         int ret;
258         ret = EVP_DigestFinal_ex(ctx, md, size);
259         EVP_MD_CTX_cleanup(ctx);
```

```
260         return ret;
261         }

263 /* The caller can assume that this removes any secret data from the context */
264 int EVP_DigestFinal_ex(EVP_MD_CTX *ctx, unsigned char *md, unsigned int *size)
265         {
266 #ifdef OPENSSL_FIPS
267         return FIPS_digestfinal(ctx, md, size);
268 #else
269         int ret;

271         OPENSSL_assert(ctx->digest->md_size <= EVP_MAX_MD_SIZE);
272         ret=ctx->digest->final(ctx,md);
273         if (size != NULL)
274                 *size=ctx->digest->md_size;
275         if (ctx->digest->cleanup)
276                 {
277                 ctx->digest->cleanup(ctx);
278                 EVP_MD_CTX_set_flags(ctx,EVP_MD_CTX_FLAG_CLEANED);
279                 }
280         memset(ctx->md_data,0,ctx->digest->ctx_size);
281         return ret;
282 #endif
283         }

285 int EVP_MD_CTX_copy(EVP_MD_CTX *out, const EVP_MD_CTX *in)
286         {
287         EVP_MD_CTX_init(out);
288         return EVP_MD_CTX_copy_ex(out, in);
289         }

291 int EVP_MD_CTX_copy_ex(EVP_MD_CTX *out, const EVP_MD_CTX *in)
292         {
293         unsigned char *tmp_buf;
294         if ((in == NULL) || (in->digest == NULL))
295                 {
296                 EVPerr(EVP_F_EVP_MD_CTX_COPY_EX,EVP_R_INPUT_NOT_INITIALIZED);
297                 return 0;
298                 }
299 #ifndef OPENSSL_NO_ENGINE
300         /* Make sure it's safe to copy a digest context using an ENGINE */
301         if (in->engine && !ENGINE_init(in->engine))
302                 {
303                 EVPerr(EVP_F_EVP_MD_CTX_COPY_EX,ERR_R_ENGINE_LIB);
304                 return 0;
305                 }
306 #endif

308         if (out->digest == in->digest)
309                 {
310                 tmp_buf = out->md_data;
311                 EVP_MD_CTX_set_flags(out,EVP_MD_CTX_FLAG_REUSE);
312                 }
313         else tmp_buf = NULL;
314         EVP_MD_CTX_cleanup(out);
315         memcpy(out,in,sizeof *out);

317         if (in->md_data && out->digest->ctx_size)
318                 {
319                 if (tmp_buf)
320                         out->md_data = tmp_buf;
321                 else
322                         {
323                         out->md_data=OPENSSL_malloc(out->digest->ctx_size);
324                         if (!out->md_data)
325                                 {
```



```
326                                 EVPerr(EVP_F_EVP_MD_CTX_COPY_EX,ERR_R_MALLOC_FAI
327                                 return 0;
328                                 }
329                         }
330                 memcpy(out->md_data,in->md_data,out->digest->ctx_size);
331                 }

333         out->update = in->update;

335         if (in->pctx)
336                 {
337                 out->pctx = EVP_PKEY_CTX_dup(in->pctx);
338                 if (!out->pctx)
339                         {
340                         EVP_MD_CTX_cleanup(out);
341                         return 0;
342                         }
343                 }

345         if (out->digest->copy)
346                 return out->digest->copy(out,in);

348         return 1;
349         }

351 int EVP_Digest(const void *data, size_t count,
352                 unsigned char *md, unsigned int *size, const EVP_MD *type, ENGIN
353         {
354         EVP_MD_CTX ctx;
355         int ret;

357         EVP_MD_CTX_init(&ctx);
358         EVP_MD_CTX_set_flags(&ctx,EVP_MD_CTX_FLAG_ONESHOT);
359         ret=EVP_DigestInit_ex(&ctx, type, impl)
360           && EVP_DigestUpdate(&ctx, data, count)
361           && EVP_DigestFinal_ex(&ctx, md, size);
362         EVP_MD_CTX_cleanup(&ctx);

364         return ret;
365         }

367 void EVP_MD_CTX_destroy(EVP_MD_CTX *ctx)
368         {
369         if (ctx)
370                 {
371                 EVP_MD_CTX_cleanup(ctx);
372                 OPENSSL_free(ctx);
373                 }
374         }

376 /* This call frees resources associated with the context */
377 int EVP_MD_CTX_cleanup(EVP_MD_CTX *ctx)
378         {
379 #ifndef OPENSSL_FIPS
380         /* Don't assume ctx->md_data was cleaned in EVP_Digest_Final,
381          * because sometimes only copies of the context are ever finalised.
382          */
383         if (ctx->digest && ctx->digest->cleanup
384             && !EVP_MD_CTX_test_flags(ctx,EVP_MD_CTX_FLAG_CLEANED))
385                 ctx->digest->cleanup(ctx);
386         if (ctx->digest && ctx->digest->ctx_size && ctx->md_data
387             && !EVP_MD_CTX_test_flags(ctx, EVP_MD_CTX_FLAG_REUSE))
388                 {
389                 OPENSSL_cleanse(ctx->md_data,ctx->digest->ctx_size);
390                 OPENSSL_free(ctx->md_data);
391                 }
```

```
392 #endif
393         if (ctx->pctx)
394                 EVP_PKEY_CTX_free(ctx->pctx);
395 #ifndef OPENSSL_NO_ENGINE
396         if(ctx->engine)
397                 /* The EVP_MD we used belongs to an ENGINE, release the
398                  * functional reference we held for this reason. */
399                 ENGINE_finish(ctx->engine);
400 #endif
401 #ifdef OPENSSL_FIPS
402         FIPS_md_ctx_cleanup(ctx);
403 #endif
404         memset(ctx,'\0',sizeof *ctx);

406         return 1;
407         }
408 #endif /* ! codereview */
```

```
**********************************************************
   36304 Wed Aug 13 19:52:42 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_aes.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2001-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/opensslconf.h>
#ifndef OPENSSL_NO_AES
#include <openssl/evp.h>
#include <openssl/err.h>
#include <string.h>
#include <assert.h>
#include <openssl/aes.h>
#include "evp_locl.h"
#ifndef OPENSSL_FIPS
#include "modes_lcl.h"
#include <openssl/rand.h>
```



```
typedef struct
        {
        AES_KEY ks;
        block128_f block;
        union {
                cbc128_f cbc;
                ctr128_f ctr;
        } stream;
        } EVP_AES_KEY;

typedef struct
        {
        AES_KEY ks;             /* AES key schedule to use */
        int key_set;            /* Set if key initialised */
        int iv_set;             /* Set if an iv is set */
        GCM128_CONTEXT gcm;
        unsigned char *iv;      /* Temporary IV store */
        int ivlen;              /* IV length */
        int taglen;
        int iv_gen;             /* It is OK to generate IVs */
        int tls_aad_len;        /* TLS AAD length */
        ctr128_f ctr;
        } EVP_AES_GCM_CTX;

typedef struct
        {
        AES_KEY ks1, ks2;       /* AES key schedules to use */
        XTS128_CONTEXT xts;
        void     (*stream)(const unsigned char *in,
                        unsigned char *out, size_t length,
                        const AES_KEY *key1, const AES_KEY *key2,
                        const unsigned char iv[16]);
        } EVP_AES_XTS_CTX;

typedef struct
        {
        AES_KEY ks;             /* AES key schedule to use */
        int key_set;            /* Set if key initialised */
        int iv_set;             /* Set if an iv is set */
        int tag_set;            /* Set if tag is valid */
        int len_set;            /* Set if message length set */
        int L, M;               /* L and M parameters from RFC3610 */
        CCM128_CONTEXT ccm;
        ccm128_f str;
        } EVP_AES_CCM_CTX;

#define MAXBITCHUNK     ((size_t)1<<(sizeof(size_t)*8-4))

#ifdef VPAES_ASM
int vpaes_set_encrypt_key(const unsigned char *userKey, int bits,
                        AES_KEY *key);
int vpaes_set_decrypt_key(const unsigned char *userKey, int bits,
                        AES_KEY *key);

void vpaes_encrypt(const unsigned char *in, unsigned char *out,
                        const AES_KEY *key);
void vpaes_decrypt(const unsigned char *in, unsigned char *out,
                        const AES_KEY *key);

void vpaes_cbc_encrypt(const unsigned char *in,
                        unsigned char *out,
                        size_t length,
                        const AES_KEY *key,
                        unsigned char *ivec, int enc);
#endif
```

```
#ifdef BSAES_ASM
void bsaes_cbc_encrypt(const unsigned char *in, unsigned char *out,
                        size_t length, const AES_KEY *key,
                        unsigned char ivec[16], int enc);
void bsaes_ctr32_encrypt_blocks(const unsigned char *in, unsigned char *out,
                        size_t len, const AES_KEY *key,
                        const unsigned char ivec[16]);
void bsaes_xts_encrypt(const unsigned char *inp, unsigned char *out,
                        size_t len, const AES_KEY *key1,
                        const AES_KEY *key2, const unsigned char iv[16]);
void bsaes_xts_decrypt(const unsigned char *inp, unsigned char *out,
                        size_t len, const AES_KEY *key1,
                        const AES_KEY *key2, const unsigned char iv[16]);
#endif
#ifdef AES_CTR_ASM
void AES_ctr32_encrypt(const unsigned char *in, unsigned char *out,
                        size_t blocks, const AES_KEY *key,
                        const unsigned char ivec[AES_BLOCK_SIZE]);
#endif
#ifdef AES_XTS_ASM
void AES_xts_encrypt(const char *inp,char *out,size_t len,
                        const AES_KEY *key1, const AES_KEY *key2,
                        const unsigned char iv[16]);
void AES_xts_decrypt(const char *inp,char *out,size_t len,
                        const AES_KEY *key1, const AES_KEY *key2,
                        const unsigned char iv[16]);
#endif

#if     defined(AES_ASM) && !defined(I386_ONLY) &&      (  \
        ((defined(__i386)       || defined(__i386__)    || \
          defined(_M_IX86)) && defined(OPENSSL_IA32_SSE2))|| \
        defined(__x86_64)       || defined(__x86_64__)  || \
        defined(_M_AMD64)       || defined(_M_X64)      || \
        defined(__INTEL__)                              )

extern unsigned int OPENSSL_ia32cap_P[2];

#ifdef VPAES_ASM
#define VPAES_CAPABLE   (OPENSSL_ia32cap_P[1]&(1<<(41-32)))
#endif
#ifdef BSAES_ASM
#define BSAES_CAPABLE   VPAES_CAPABLE
#endif
/*
 * AES-NI section
 */
#define AESNI_CAPABLE   (OPENSSL_ia32cap_P[1]&(1<<(57-32)))

int aesni_set_encrypt_key(const unsigned char *userKey, int bits,
                        AES_KEY *key);
int aesni_set_decrypt_key(const unsigned char *userKey, int bits,
                        AES_KEY *key);

void aesni_encrypt(const unsigned char *in, unsigned char *out,
                        const AES_KEY *key);
void aesni_decrypt(const unsigned char *in, unsigned char *out,
                        const AES_KEY *key);

void aesni_ecb_encrypt(const unsigned char *in,
                        unsigned char *out,
                        size_t length,
                        const AES_KEY *key,
                        int enc);
void aesni_cbc_encrypt(const unsigned char *in,
                        unsigned char *out,
                        size_t length,
```



```
                        const AES_KEY *key,
                        unsigned char *ivec, int enc);

void aesni_ctr32_encrypt_blocks(const unsigned char *in,
                        unsigned char *out,
                        size_t blocks,
                        const void *key,
                        const unsigned char *ivec);

void aesni_xts_encrypt(const unsigned char *in,
                        unsigned char *out,
                        size_t length,
                        const AES_KEY *key1, const AES_KEY *key2,
                        const unsigned char iv[16]);

void aesni_xts_decrypt(const unsigned char *in,
                        unsigned char *out,
                        size_t length,
                        const AES_KEY *key1, const AES_KEY *key2,
                        const unsigned char iv[16]);

void aesni_ccm64_encrypt_blocks (const unsigned char *in,
                        unsigned char *out,
                        size_t blocks,
                        const void *key,
                        const unsigned char ivec[16],
                        unsigned char cmac[16]);

void aesni_ccm64_decrypt_blocks (const unsigned char *in,
                        unsigned char *out,
                        size_t blocks,
                        const void *key,
                        const unsigned char ivec[16],
                        unsigned char cmac[16]);

static int aesni_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                    const unsigned char *iv, int enc)
        {
        int ret, mode;
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        mode = ctx->cipher->flags & EVP_CIPH_MODE;
        if ((mode == EVP_CIPH_ECB_MODE || mode == EVP_CIPH_CBC_MODE)
            && !enc)
                {
                ret = aesni_set_decrypt_key(key, ctx->key_len*8, ctx->cipher_dat
                dat->block      = (block128_f)aesni_decrypt;
                dat->stream.cbc = mode==EVP_CIPH_CBC_MODE ?
                                        (cbc128_f)aesni_cbc_encrypt :
                                        NULL;
                }
        else    {
                ret = aesni_set_encrypt_key(key, ctx->key_len*8, ctx->cipher_dat
                dat->block      = (block128_f)aesni_encrypt;
                if (mode==EVP_CIPH_CBC_MODE)
                        dat->stream.cbc = (cbc128_f)aesni_cbc_encrypt;
                else if (mode==EVP_CIPH_CTR_MODE)
                        dat->stream.ctr = (ctr128_f)aesni_ctr32_encrypt_blocks;
                else
                        dat->stream.cbc = NULL;
                }

        if(ret < 0)
                {
                EVPerr(EVP_F_AESNI_INIT_KEY,EVP_R_AES_KEY_SETUP_FAILED);
                return 0;
```

```
260                 }

262         return 1;
263         }

265 static int aesni_cbc_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
266         const unsigned char *in, size_t len)
267 {
268         aesni_cbc_encrypt(in,out,len,ctx->cipher_data,ctx->iv,ctx->encrypt);

270         return 1;
271 }

273 static int aesni_ecb_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
274         const unsigned char *in, size_t len)
275 {
276         size_t  bl = ctx->cipher->block_size;

278         if (len<bl)     return 1;

280         aesni_ecb_encrypt(in,out,len,ctx->cipher_data,ctx->encrypt);

282         return 1;
283 }

285 #define aesni_ofb_cipher aes_ofb_cipher
286 static int aesni_ofb_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
287         const unsigned char *in,size_t len);

289 #define aesni_cfb_cipher aes_cfb_cipher
290 static int aesni_cfb_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
291         const unsigned char *in,size_t len);

293 #define aesni_cfb8_cipher aes_cfb8_cipher
294 static int aesni_cfb8_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
295         const unsigned char *in,size_t len);

297 #define aesni_cfb1_cipher aes_cfb1_cipher
298 static int aesni_cfb1_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
299         const unsigned char *in,size_t len);

301 #define aesni_ctr_cipher aes_ctr_cipher
302 static int aesni_ctr_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
303                 const unsigned char *in, size_t len);

305 static int aesni_gcm_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
306                         const unsigned char *iv, int enc)
307         {
308         EVP_AES_GCM_CTX *gctx = ctx->cipher_data;
309         if (!iv && !key)
310                 return 1;
311         if (key)
312                 {
313                 aesni_set_encrypt_key(key, ctx->key_len * 8, &gctx->ks);
314                 CRYPTO_gcm128_init(&gctx->gcm, &gctx->ks,
315                                    (block128_f)aesni_encrypt);
316                 gctx->ctr = (ctr128_f)aesni_ctr32_encrypt_blocks;
317                 /* If we have an iv can set it directly, otherwise use
318                  * saved IV.
319                  */
320                 if (iv == NULL && gctx->iv_set)
321                         iv = gctx->iv;
322                 if (iv)
323                         {
324                         CRYPTO_gcm128_setiv(&gctx->gcm, iv, gctx->ivlen);
325                         gctx->iv_set = 1;
```



```
326                         }
327                 gctx->key_set = 1;
328                 }
329         else
330                 {
331                 /* If key set use IV, otherwise copy */
332                 if (gctx->key_set)
333                         CRYPTO_gcm128_setiv(&gctx->gcm, iv, gctx->ivlen);
334                 else
335                         memcpy(gctx->iv, iv, gctx->ivlen);
336                 gctx->iv_set = 1;
337                 gctx->iv_gen = 0;
338                 }
339         return 1;
340         }

342 #define aesni_gcm_cipher aes_gcm_cipher
343 static int aesni_gcm_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
344                 const unsigned char *in, size_t len);

346 static int aesni_xts_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
347                         const unsigned char *iv, int enc)
348         {
349         EVP_AES_XTS_CTX *xctx = ctx->cipher_data;
350         if (!iv && !key)
351                 return 1;

353         if (key)
354                 {
355                 /* key_len is two AES keys */
356                 if (enc)
357                         {
358                         aesni_set_encrypt_key(key, ctx->key_len * 4, &xctx->ks1)
359                         xctx->xts.block1 = (block128_f)aesni_encrypt;
360                         xctx->stream = aesni_xts_encrypt;
361                         }
362                 else
363                         {
364                         aesni_set_decrypt_key(key, ctx->key_len * 4, &xctx->ks1)
365                         xctx->xts.block1 = (block128_f)aesni_decrypt;
366                         xctx->stream = aesni_xts_decrypt;
367                         }

369                 aesni_set_encrypt_key(key + ctx->key_len/2,
370                                                 ctx->key_len * 4, &xctx->ks2);
371                 xctx->xts.block2 = (block128_f)aesni_encrypt;

373                 xctx->xts.key1 = &xctx->ks1;
374                 }

376         if (iv)
377                 {
378                 xctx->xts.key2 = &xctx->ks2;
379                 memcpy(ctx->iv, iv, 16);
380                 }

382         return 1;
383         }

385 #define aesni_xts_cipher aes_xts_cipher
386 static int aesni_xts_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
387                 const unsigned char *in, size_t len);

389 static int aesni_ccm_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
390                         const unsigned char *iv, int enc)
391         {
```

```
392         EVP_AES_CCM_CTX *cctx = ctx->cipher_data;
393         if (!iv && !key)
394                 return 1;
395         if (key)
396                 {
397                 aesni_set_encrypt_key(key, ctx->key_len * 8, &cctx->ks);
398                 CRYPTO_ccm128_init(&cctx->ccm, cctx->M, cctx->L,
399                                    &cctx->ks, (block128_f)aesni_encrypt);
400                 cctx->str = enc?(ccm128_f)aesni_ccm64_encrypt_blocks :
401                                 (ccm128_f)aesni_ccm64_decrypt_blocks;
402                 cctx->key_set = 1;
403                 }
404         if (iv)
405                 {
406                 memcpy(ctx->iv, iv, 15 - cctx->L);
407                 cctx->iv_set = 1;
408                 }
409         return 1;
410         }

412 #define aesni_ccm_cipher aes_ccm_cipher
413 static int aesni_ccm_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
414                 const unsigned char *in, size_t len);

416 #define BLOCK_CIPHER_generic(nid,keylen,blocksize,ivlen,nmode,mode,MODE,flags) \
417 static const EVP_CIPHER aesni_##keylen##_##mode = { \
418         nid##_##keylen##_##nmode,blocksize,keylen/8,ivlen, \
419         flags|EVP_CIPH_##MODE##_MODE,   \
420         aesni_init_key,                 \
421         aesni_##mode##_cipher,          \
422         NULL,                           \
423         sizeof(EVP_AES_KEY),            \
424         NULL,NULL,NULL,NULL }; \
425 static const EVP_CIPHER aes_##keylen##_##mode = { \
426         nid##_##keylen##_##nmode,blocksize,     \
427         keylen/8,ivlen, \
428         flags|EVP_CIPH_##MODE##_MODE,   \
429         aes_init_key,                   \
430         aes_##mode##_cipher,            \
431         NULL,                           \
432         sizeof(EVP_AES_KEY),            \
433         NULL,NULL,NULL,NULL }; \
434 const EVP_CIPHER *EVP_aes_##keylen##_##mode(void) \
435 { return AESNI_CAPABLE?&aesni_##keylen##_##mode:&aes_##keylen##_##mode; }

437 #define BLOCK_CIPHER_custom(nid,keylen,blocksize,ivlen,mode,MODE,flags) \
438 static const EVP_CIPHER aesni_##keylen##_##mode = { \
439         nid##_##keylen##_##mode,blocksize, \
440         (EVP_CIPH_##MODE##_MODE==EVP_CIPH_XTS_MODE?2:1)*keylen/8, ivlen, \
441         flags|EVP_CIPH_##MODE##_MODE,   \
442         aesni_##mode##_init_key,        \
443         aesni_##mode##_cipher,          \
444         aes_##mode##_cleanup,           \
445         sizeof(EVP_AES_##MODE##_CTX),   \
446         NULL,NULL,aes_##mode##_ctrl,NULL }; \
447 static const EVP_CIPHER aes_##keylen##_##mode = { \
448         nid##_##keylen##_##mode,blocksize, \
449         (EVP_CIPH_##MODE##_MODE==EVP_CIPH_XTS_MODE?2:1)*keylen/8, ivlen, \
450         flags|EVP_CIPH_##MODE##_MODE,   \
451         aes_##mode##_init_key,          \
452         aes_##mode##_cipher,            \
453         aes_##mode##_cleanup,           \
454         sizeof(EVP_AES_##MODE##_CTX),   \
455         NULL,NULL,aes_##mode##_ctrl,NULL }; \
456 const EVP_CIPHER *EVP_aes_##keylen##_##mode(void) \
457 { return AESNI_CAPABLE?&aesni_##keylen##_##mode:&aes_##keylen##_##mode; }
```



```
459 #else

461 #define BLOCK_CIPHER_generic(nid,keylen,blocksize,ivlen,nmode,mode,MODE,flags) \
462 static const EVP_CIPHER aes_##keylen##_##mode = { \
463         nid##_##keylen##_##nmode,blocksize,keylen/8,ivlen, \
464         flags|EVP_CIPH_##MODE##_MODE,   \
465         aes_init_key,                   \
466         aes_##mode##_cipher,            \
467         NULL,                           \
468         sizeof(EVP_AES_KEY),            \
469         NULL,NULL,NULL,NULL }; \
470 const EVP_CIPHER *EVP_aes_##keylen##_##mode(void) \
471 { return &aes_##keylen##_##mode; }

473 #define BLOCK_CIPHER_custom(nid,keylen,blocksize,ivlen,mode,MODE,flags) \
474 static const EVP_CIPHER aes_##keylen##_##mode = { \
475         nid##_##keylen##_##mode,blocksize, \
476         (EVP_CIPH_##MODE##_MODE==EVP_CIPH_XTS_MODE?2:1)*keylen/8, ivlen, \
477         flags|EVP_CIPH_##MODE##_MODE,   \
478         aes_##mode##_init_key,          \
479         aes_##mode##_cipher,            \
480         aes_##mode##_cleanup,           \
481         sizeof(EVP_AES_##MODE##_CTX),   \
482         NULL,NULL,aes_##mode##_ctrl,NULL }; \
483 const EVP_CIPHER *EVP_aes_##keylen##_##mode(void) \
484 { return &aes_##keylen##_##mode; }
485 #endif

487 #define BLOCK_CIPHER_generic_pack(nid,keylen,flags)             \
488         BLOCK_CIPHER_generic(nid,keylen,16,16,cbc,cbc,CBC,flags|EVP_CIPH_FLAG_DE
489         BLOCK_CIPHER_generic(nid,keylen,16,0,ecb,ecb,ECB,flags|EVP_CIPH_FLAG_DEF
490         BLOCK_CIPHER_generic(nid,keylen,1,16,ofb128,ofb,OFB,flags|EVP_CIPH_FLAG_
491         BLOCK_CIPHER_generic(nid,keylen,1,16,cfb128,cfb,CFB,flags|EVP_CIPH_FLAG_
492         BLOCK_CIPHER_generic(nid,keylen,1,16,cfb1,cfb1,CFB,flags)       \
493         BLOCK_CIPHER_generic(nid,keylen,1,16,cfb8,cfb8,CFB,flags)       \
494         BLOCK_CIPHER_generic(nid,keylen,1,16,ctr,ctr,CTR,flags)

496 static int aes_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
497                         const unsigned char *iv, int enc)
498         {
499         int ret, mode;
500         EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

502         mode = ctx->cipher->flags & EVP_CIPH_MODE;
503         if ((mode == EVP_CIPH_ECB_MODE || mode == EVP_CIPH_CBC_MODE)
504             && !enc)
505 #ifdef BSAES_CAPABLE
506             if (BSAES_CAPABLE && mode==EVP_CIPH_CBC_MODE)
507                 {
508                 ret = AES_set_decrypt_key(key,ctx->key_len*8,&dat->ks);
509                 dat->block      = (block128_f)AES_decrypt;
510                 dat->stream.cbc = (cbc128_f)bsaes_cbc_encrypt;
511                 }
512             else
513 #endif
514 #ifdef VPAES_CAPABLE
515             if (VPAES_CAPABLE)
516                 {
517                 ret = vpaes_set_decrypt_key(key,ctx->key_len*8,&dat->ks);
518                 dat->block      = (block128_f)vpaes_decrypt;
519                 dat->stream.cbc = mode==EVP_CIPH_CBC_MODE ?
520                                         (cbc128_f)vpaes_cbc_encrypt :
521                                         NULL;
522                 }
523             else
```

```
#endif
                {
                ret = AES_set_decrypt_key(key,ctx->key_len*8,&dat->ks);
                dat->block      = (block128_f)AES_decrypt;
                dat->stream.cbc = mode==EVP_CIPH_CBC_MODE ?
                                        (cbc128_f)AES_cbc_encrypt :
                                        NULL;
                }
        else
#ifdef BSAES_CAPABLE
            if (BSAES_CAPABLE && mode==EVP_CIPH_CTR_MODE)
                {
                ret = AES_set_encrypt_key(key,ctx->key_len*8,&dat->ks);
                dat->block      = (block128_f)AES_encrypt;
                dat->stream.ctr = (ctr128_f)bsaes_ctr32_encrypt_blocks;
                }
            else
#endif
#ifdef VPAES_CAPABLE
            if (VPAES_CAPABLE)
                {
                ret = vpaes_set_encrypt_key(key,ctx->key_len*8,&dat->ks);
                dat->block      = (block128_f)vpaes_encrypt;
                dat->stream.cbc = mode==EVP_CIPH_CBC_MODE ?
                                        (cbc128_f)vpaes_cbc_encrypt :
                                        NULL;
                }
            else
#endif
                {
                ret = AES_set_encrypt_key(key,ctx->key_len*8,&dat->ks);
                dat->block      = (block128_f)AES_encrypt;
                dat->stream.cbc = mode==EVP_CIPH_CBC_MODE ?
                                        (cbc128_f)AES_cbc_encrypt :
                                        NULL;
#ifdef AES_CTR_ASM
                if (mode==EVP_CIPH_CTR_MODE)
                        dat->stream.ctr = (ctr128_f)AES_ctr32_encrypt;
#endif
                }

        if(ret < 0)
                {
                EVPerr(EVP_F_AES_INIT_KEY,EVP_R_AES_KEY_SETUP_FAILED);
                return 0;
                }

        return 1;
        }

static int aes_cbc_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
        const unsigned char *in, size_t len)
{
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        if (dat->stream.cbc)
                (*dat->stream.cbc)(in,out,len,&dat->ks,ctx->iv,ctx->encrypt);
        else if (ctx->encrypt)
                CRYPTO_cbc128_encrypt(in,out,len,&dat->ks,ctx->iv,dat->block);
        else
                CRYPTO_cbc128_encrypt(in,out,len,&dat->ks,ctx->iv,dat->block);

        return 1;
}

static int aes_ecb_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
```



```
        const unsigned char *in, size_t len)
{
        size_t  bl = ctx->cipher->block_size;
        size_t  i;
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        if (len<bl)     return 1;

        for (i=0,len-=bl;i<=len;i+=bl)
                (*dat->block)(in+i,out+i,&dat->ks);

        return 1;
}

static int aes_ofb_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
        const unsigned char *in,size_t len)
{
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        CRYPTO_ofb128_encrypt(in,out,len,&dat->ks,
                        ctx->iv,&ctx->num,dat->block);
        return 1;
}

static int aes_cfb_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
        const unsigned char *in,size_t len)
{
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        CRYPTO_cfb128_encrypt(in,out,len,&dat->ks,
                        ctx->iv,&ctx->num,ctx->encrypt,dat->block);
        return 1;
}

static int aes_cfb8_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
        const unsigned char *in,size_t len)
{
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        CRYPTO_cfb128_8_encrypt(in,out,len,&dat->ks,
                        ctx->iv,&ctx->num,ctx->encrypt,dat->block);
        return 1;
}

static int aes_cfb1_cipher(EVP_CIPHER_CTX *ctx,unsigned char *out,
        const unsigned char *in,size_t len)
{
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        if (ctx->flags&EVP_CIPH_FLAG_LENGTH_BITS) {
                CRYPTO_cfb128_1_encrypt(in,out,len,&dat->ks,
                        ctx->iv,&ctx->num,ctx->encrypt,dat->block);
                return 1;
        }

        while (len>=MAXBITCHUNK) {
                CRYPTO_cfb128_1_encrypt(in,out,MAXBITCHUNK*8,&dat->ks,
                        ctx->iv,&ctx->num,ctx->encrypt,dat->block);
                len-=MAXBITCHUNK;
        }
        if (len)
                CRYPTO_cfb128_1_encrypt(in,out,len*8,&dat->ks,
                        ctx->iv,&ctx->num,ctx->encrypt,dat->block);

        return 1;
}
```

```
static int aes_ctr_cipher (EVP_CIPHER_CTX *ctx, unsigned char *out,
                const unsigned char *in, size_t len)
{
        unsigned int num = ctx->num;
        EVP_AES_KEY *dat = (EVP_AES_KEY *)ctx->cipher_data;

        if (dat->stream.ctr)
                CRYPTO_ctr128_encrypt_ctr32(in,out,len,&dat->ks,
                        ctx->iv,ctx->buf,&num,dat->stream.ctr);
        else
                CRYPTO_ctr128_encrypt(in,out,len,&dat->ks,
                        ctx->iv,ctx->buf,&num,dat->block);
        ctx->num = (size_t)num;
        return 1;
}

BLOCK_CIPHER_generic_pack(NID_aes,128,EVP_CIPH_FLAG_FIPS)
BLOCK_CIPHER_generic_pack(NID_aes,192,EVP_CIPH_FLAG_FIPS)
BLOCK_CIPHER_generic_pack(NID_aes,256,EVP_CIPH_FLAG_FIPS)

static int aes_gcm_cleanup(EVP_CIPHER_CTX *c)
        {
        EVP_AES_GCM_CTX *gctx = c->cipher_data;
        OPENSSL_cleanse(&gctx->gcm, sizeof(gctx->gcm));
        if (gctx->iv != c->iv)
                OPENSSL_free(gctx->iv);
        return 1;
        }

/* increment counter (64-bit int) by 1 */
static void ctr64_inc(unsigned char *counter) {
        int n=8;
        unsigned char  c;

        do {
                --n;
                c = counter[n];
                ++c;
                counter[n] = c;
                if (c) return;
        } while (n);
}

static int aes_gcm_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr)
        {
        EVP_AES_GCM_CTX *gctx = c->cipher_data;
        switch (type)
                {
        case EVP_CTRL_INIT:
                gctx->key_set = 0;
                gctx->iv_set = 0;
                gctx->ivlen = c->cipher->iv_len;
                gctx->iv = c->iv;
                gctx->taglen = -1;
                gctx->iv_gen = 0;
                gctx->tls_aad_len = -1;
                return 1;

        case EVP_CTRL_GCM_SET_IVLEN:
                if (arg <= 0)
                        return 0;
#ifdef OPENSSL_FIPS
                if (FIPS_module_mode() && !(c->flags & EVP_CIPH_FLAG_NON_FIPS_AL
                                                && arg < 12)
                        return 0;
```



```
#endif
                /* Allocate memory for IV if needed */
                if ((arg > EVP_MAX_IV_LENGTH) && (arg > gctx->ivlen))
                        {
                        if (gctx->iv != c->iv)
                                OPENSSL_free(gctx->iv);
                        gctx->iv = OPENSSL_malloc(arg);
                        if (!gctx->iv)
                                return 0;
                        }
                gctx->ivlen = arg;
                return 1;

        case EVP_CTRL_GCM_SET_TAG:
                if (arg <= 0 || arg > 16 || c->encrypt)
                        return 0;
                memcpy(c->buf, ptr, arg);
                gctx->taglen = arg;
                return 1;

        case EVP_CTRL_GCM_GET_TAG:
                if (arg <= 0 || arg > 16 || !c->encrypt || gctx->taglen < 0)
                        return 0;
                memcpy(ptr, c->buf, arg);
                return 1;

        case EVP_CTRL_GCM_SET_IV_FIXED:
                /* Special case: -1 length restores whole IV */
                if (arg == -1)
                        {
                        memcpy(gctx->iv, ptr, gctx->ivlen);
                        gctx->iv_gen = 1;
                        return 1;
                        }
                /* Fixed field must be at least 4 bytes and invocation field
                 * at least 8.
                 */
                if ((arg < 4) || (gctx->ivlen - arg) < 8)
                        return 0;
                if (arg)
                        memcpy(gctx->iv, ptr, arg);
                if (c->encrypt &&
                        RAND_bytes(gctx->iv + arg, gctx->ivlen - arg) <= 0)
                        return 0;
                gctx->iv_gen = 1;
                return 1;

        case EVP_CTRL_GCM_IV_GEN:
                if (gctx->iv_gen == 0 || gctx->key_set == 0)
                        return 0;
                CRYPTO_gcm128_setiv(&gctx->gcm, gctx->iv, gctx->ivlen);
                if (arg <= 0 || arg > gctx->ivlen)
                        arg = gctx->ivlen;
                memcpy(ptr, gctx->iv + gctx->ivlen - arg, arg);
                /* Invocation field will be at least 8 bytes in size and
                 * so no need to check wrap around or increment more than
                 * last 8 bytes.
                 */
                ctr64_inc(gctx->iv + gctx->ivlen - 8);
                gctx->iv_set = 1;
                return 1;

        case EVP_CTRL_GCM_SET_IV_INV:
                if (gctx->iv_gen == 0 || gctx->key_set == 0 || c->encrypt)
                        return 0;
                memcpy(gctx->iv + gctx->ivlen - arg, ptr, arg);
```

```
788                 CRYPTO_gcm128_setiv(&gctx->gcm, gctx->iv, gctx->ivlen);
789                 gctx->iv_set = 1;
790                 return 1;

792         case EVP_CTRL_AEAD_TLS1_AAD:
793                 /* Save the AAD for later use */
794                 if (arg != 13)
795                         return 0;
796                 memcpy(c->buf, ptr, arg);
797                 gctx->tls_aad_len = arg;
798                         {
799                         unsigned int len=c->buf[arg-2]<<8|c->buf[arg-1];
800                         /* Correct length for explicit IV */
801                         len -= EVP_GCM_TLS_EXPLICIT_IV_LEN;
802                         /* If decrypting correct for tag too */
803                         if (!c->encrypt)
804                                 len -= EVP_GCM_TLS_TAG_LEN;
805                         c->buf[arg-2] = len>>8;
806                         c->buf[arg-1] = len & 0xff;
807                         }
808                 /* Extra padding: tag appended to record */
809                 return EVP_GCM_TLS_TAG_LEN;

811         case EVP_CTRL_COPY:
812                 {
813                         EVP_CIPHER_CTX *out = ptr;
814                         EVP_AES_GCM_CTX *gctx_out = out->cipher_data;
815                         if (gctx->gcm.key)
816                                 {
817                                 if (gctx->gcm.key != &gctx->ks)
818                                         return 0;
819                                 gctx_out->gcm.key = &gctx_out->ks;
820                                 }
821                         if (gctx->iv == c->iv)
822                                 gctx_out->iv = out->iv;
823                         else
824                         {
825                                 gctx_out->iv = OPENSSL_malloc(gctx->ivlen);
826                                 if (!gctx_out->iv)
827                                         return 0;
828                                 memcpy(gctx_out->iv, gctx->iv, gctx->ivlen);
829                         }
830                         return 1;
831                 }

833         default:
834                 return -1;

836                 }
837         }

839 static int aes_gcm_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
840                         const unsigned char *iv, int enc)
841         {
842         EVP_AES_GCM_CTX *gctx = ctx->cipher_data;
843         if (!iv && !key)
844                 return 1;
845         if (key)
846                 { do {
847 #ifdef BSAES_CAPABLE
848                 if (BSAES_CAPABLE)
849                         {
850                         AES_set_encrypt_key(key,ctx->key_len*8,&gctx->ks);
851                         CRYPTO_gcm128_init(&gctx->gcm,&gctx->ks,
852                                         (block128_f)AES_encrypt);
853                         gctx->ctr = (ctr128_f)bsaes_ctr32_encrypt_blocks;
```



```
854                         break;
855                         }
856                 else
857 #endif
858 #ifdef VPAES_CAPABLE
859                 if (VPAES_CAPABLE)
860                         {
861                         vpaes_set_encrypt_key(key,ctx->key_len*8,&gctx->ks);
862                         CRYPTO_gcm128_init(&gctx->gcm,&gctx->ks,
863                                         (block128_f)vpaes_encrypt);
864                         gctx->ctr = NULL;
865                         break;
866                         }
867                 else
868 #endif
869                 (void)0;        /* terminate potentially open 'else' */

871                 AES_set_encrypt_key(key, ctx->key_len * 8, &gctx->ks);
872                 CRYPTO_gcm128_init(&gctx->gcm, &gctx->ks, (block128_f)AES_encryp
873 #ifdef AES_CTR_ASM
874                 gctx->ctr = (ctr128_f)AES_ctr32_encrypt;
875 #else
876                 gctx->ctr = NULL;
877 #endif
878                 } while (0);

880                 /* If we have an iv can set it directly, otherwise use
881                  * saved IV.
882                  */
883                 if (iv == NULL && gctx->iv_set)
884                         iv = gctx->iv;
885                 if (iv)
886                         {
887                         CRYPTO_gcm128_setiv(&gctx->gcm, iv, gctx->ivlen);
888                         gctx->iv_set = 1;
889                         }
890                 gctx->key_set = 1;
891                 }
892         else
893                 {
894                 /* If key set use IV, otherwise copy */
895                 if (gctx->key_set)
896                         CRYPTO_gcm128_setiv(&gctx->gcm, iv, gctx->ivlen);
897                 else
898                         memcpy(gctx->iv, iv, gctx->ivlen);
899                 gctx->iv_set = 1;
900                 gctx->iv_gen = 0;
901                 }
902         return 1;
903         }

905 /* Handle TLS GCM packet format. This consists of the last portion of the IV
906  * followed by the payload and finally the tag. On encrypt generate IV,
907  * encrypt payload and write the tag. On verify retrieve IV, decrypt payload
908  * and verify tag.
909  */

911 static int aes_gcm_tls_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
912                 const unsigned char *in, size_t len)
913         {
914         EVP_AES_GCM_CTX *gctx = ctx->cipher_data;
915         int rv = -1;
916         /* Encrypt/decrypt must be performed in place */
917         if (out != in || len < (EVP_GCM_TLS_EXPLICIT_IV_LEN+EVP_GCM_TLS_TAG_LEN)
918                 return -1;
919         /* Set IV from start of buffer or generate IV and write to start
```

```
920          * of buffer.
921          */
922         if (EVP_CIPHER_CTX_ctrl(ctx, ctx->encrypt ?
923                                 EVP_CTRL_GCM_IV_GEN : EVP_CTRL_GCM_SET_IV_INV,
924                                 EVP_GCM_TLS_EXPLICIT_IV_LEN, out) <= 0)
925                 goto err;
926         /* Use saved AAD */
927         if (CRYPTO_gcm128_aad(&gctx->gcm, ctx->buf, gctx->tls_aad_len))
928                 goto err;
929         /* Fix buffer and length to point to payload */
930         in += EVP_GCM_TLS_EXPLICIT_IV_LEN;
931         out += EVP_GCM_TLS_EXPLICIT_IV_LEN;
932         len -= EVP_GCM_TLS_EXPLICIT_IV_LEN + EVP_GCM_TLS_TAG_LEN;
933         if (ctx->encrypt)
934                 {
935                 /* Encrypt payload */
936                 if (gctx->ctr)
937                         {
938                         if (CRYPTO_gcm128_encrypt_ctr32(&gctx->gcm,
939                                                         in, out, len,
940                                                         gctx->ctr))
941                                 goto err;
942                         }
943                 else    {
944                         if (CRYPTO_gcm128_encrypt(&gctx->gcm, in, out, len))
945                                 goto err;
946                         }
947                 out += len;
948                 /* Finally write tag */
949                 CRYPTO_gcm128_tag(&gctx->gcm, out, EVP_GCM_TLS_TAG_LEN);
950                 rv = len + EVP_GCM_TLS_EXPLICIT_IV_LEN + EVP_GCM_TLS_TAG_LEN;
951                 }
952         else
953                 {
954                 /* Decrypt */
955                 if (gctx->ctr)
956                         {
957                         if (CRYPTO_gcm128_decrypt_ctr32(&gctx->gcm,
958                                                         in, out, len,
959                                                         gctx->ctr))
960                                 goto err;
961                         }
962                 else    {
963                         if (CRYPTO_gcm128_decrypt(&gctx->gcm, in, out, len))
964                                 goto err;
965                         }
966                 /* Retrieve tag */
967                 CRYPTO_gcm128_tag(&gctx->gcm, ctx->buf,
968                                   EVP_GCM_TLS_TAG_LEN);
969                 /* If tag mismatch wipe buffer */
970                 if (memcmp(ctx->buf, in + len, EVP_GCM_TLS_TAG_LEN))
971                         {
972                         OPENSSL_cleanse(out, len);
973                         goto err;
974                         }
975                 rv = len;
976                 }

978         err:
979         gctx->iv_set = 0;
980         gctx->tls_aad_len = -1;
981         return rv;
982         }

984 static int aes_gcm_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
985                 const unsigned char *in, size_t len)
```



```
986         {
987         EVP_AES_GCM_CTX *gctx = ctx->cipher_data;
988         /* If not set up, return error */
989         if (!gctx->key_set)
990                 return -1;

992         if (gctx->tls_aad_len >= 0)
993                 return aes_gcm_tls_cipher(ctx, out, in, len);

995         if (!gctx->iv_set)
996                 return -1;
997         if (in)
998                 {
999                 if (out == NULL)
1000                        {
1001                        if (CRYPTO_gcm128_aad(&gctx->gcm, in, len))
1002                                return -1;
1003                        }
1004                else if (ctx->encrypt)
1005                        {
1006                        if (gctx->ctr)
1007                                {
1008                                if (CRYPTO_gcm128_encrypt_ctr32(&gctx->gcm,
1009                                                                in, out, len,
1010                                                                gctx->ctr))
1011                                        return -1;
1012                                }
1013                        else    {
1014                                if (CRYPTO_gcm128_encrypt(&gctx->gcm, in, out, l
1015                                        return -1;
1016                                }
1017                        }
1018                else
1019                        {
1020                        if (gctx->ctr)
1021                                {
1022                                if (CRYPTO_gcm128_decrypt_ctr32(&gctx->gcm,
1023                                                                in, out, len,
1024                                                                gctx->ctr))
1025                                        return -1;
1026                                }
1027                        else    {
1028                                if (CRYPTO_gcm128_decrypt(&gctx->gcm, in, out, l
1029                                        return -1;
1030                                }
1031                        }
1032                return len;
1033                }
1034        else
1035                {
1036                if (!ctx->encrypt)
1037                        {
1038                        if (gctx->taglen < 0)
1039                                return -1;
1040                        if (CRYPTO_gcm128_finish(&gctx->gcm,
1041                                                 ctx->buf, gctx->taglen) != 0)
1042                                return -1;
1043                        gctx->iv_set = 0;
1044                        return 0;
1045                        }
1046                CRYPTO_gcm128_tag(&gctx->gcm, ctx->buf, 16);
1047                gctx->taglen = 16;
1048                /* Don't reuse the IV */
1049                gctx->iv_set = 0;
1050                return 0;
1051                }
```

```
1053         }

1055 #define CUSTOM_FLAGS    (EVP_CIPH_FLAG_DEFAULT_ASN1 \
1056                 | EVP_CIPH_CUSTOM_IV | EVP_CIPH_FLAG_CUSTOM_CIPHER \
1057                 | EVP_CIPH_ALWAYS_CALL_INIT | EVP_CIPH_CTRL_INIT \
1058                 | EVP_CIPH_CUSTOM_COPY)

1060 BLOCK_CIPHER_custom(NID_aes,128,1,12,gcm,GCM,
1061                 EVP_CIPH_FLAG_FIPS|EVP_CIPH_FLAG_AEAD_CIPHER|CUSTOM_FLAGS)
1062 BLOCK_CIPHER_custom(NID_aes,192,1,12,gcm,GCM,
1063                 EVP_CIPH_FLAG_FIPS|EVP_CIPH_FLAG_AEAD_CIPHER|CUSTOM_FLAGS)
1064 BLOCK_CIPHER_custom(NID_aes,256,1,12,gcm,GCM,
1065                 EVP_CIPH_FLAG_FIPS|EVP_CIPH_FLAG_AEAD_CIPHER|CUSTOM_FLAGS)

1067 static int aes_xts_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr)
1068         {
1069         EVP_AES_XTS_CTX *xctx = c->cipher_data;
1070         if (type == EVP_CTRL_COPY)
1071                 {
1072                 EVP_CIPHER_CTX *out = ptr;
1073                 EVP_AES_XTS_CTX *xctx_out = out->cipher_data;
1074                 if (xctx->xts.key1)
1075                         {
1076                         if (xctx->xts.key1 != &xctx->ks1)
1077                                 return 0;
1078                         xctx_out->xts.key1 = &xctx_out->ks1;
1079                         }
1080                 if (xctx->xts.key2)
1081                         {
1082                         if (xctx->xts.key2 != &xctx->ks2)
1083                                 return 0;
1084                         xctx_out->xts.key2 = &xctx_out->ks2;
1085                         }
1086                 return 1;
1087                 }
1088         else if (type != EVP_CTRL_INIT)
1089                 return -1;
1090         /* key1 and key2 are used as an indicator both key and IV are set */
1091         xctx->xts.key1 = NULL;
1092         xctx->xts.key2 = NULL;
1093         return 1;
1094         }

1096 static int aes_xts_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
1097                         const unsigned char *iv, int enc)
1098         {
1099         EVP_AES_XTS_CTX *xctx = ctx->cipher_data;
1100         if (!iv && !key)
1101                 return 1;

1103         if (key) do
1104                 {
1105 #ifdef AES_XTS_ASM
1106                 xctx->stream = enc ? AES_xts_encrypt : AES_xts_decrypt;
1107 #else
1108                 xctx->stream = NULL;
1109 #endif
1110                 /* key_len is two AES keys */
1111 #ifdef BSAES_CAPABLE
1112                 if (BSAES_CAPABLE)
1113                         xctx->stream = enc ? bsaes_xts_encrypt : bsaes_xts_decry
1114                 else
1115 #endif
1116 #ifdef VPAES_CAPABLE
1117                 if (VPAES_CAPABLE)
```



```
1118                     {
1119                     if (enc)
1120                         {
1121                         vpaes_set_encrypt_key(key, ctx->key_len * 4, &xctx->ks1)
1122                         xctx->xts.block1 = (block128_f)vpaes_encrypt;
1123                         }
1124                     else
1125                         {
1126                         vpaes_set_decrypt_key(key, ctx->key_len * 4, &xctx->ks1)
1127                         xctx->xts.block1 = (block128_f)vpaes_decrypt;
1128                         }

1130                     vpaes_set_encrypt_key(key + ctx->key_len/2,
1131                                           ctx->key_len * 4, &xctx->ks2);
1132                     xctx->xts.block2 = (block128_f)vpaes_encrypt;

1134                     xctx->xts.key1 = &xctx->ks1;
1135                     break;
1136                     }
1137                 else
1138 #endif
1139                 (void)0;        /* terminate potentially open 'else' */

1141                 if (enc)
1142                         {
1143                         AES_set_encrypt_key(key, ctx->key_len * 4, &xctx->ks1);
1144                         xctx->xts.block1 = (block128_f)AES_encrypt;
1145                         }
1146                 else
1147                         {
1148                         AES_set_decrypt_key(key, ctx->key_len * 4, &xctx->ks1);
1149                         xctx->xts.block1 = (block128_f)AES_decrypt;
1150                         }

1152                 AES_set_encrypt_key(key + ctx->key_len/2,
1153                                     ctx->key_len * 4, &xctx->ks2);
1154                 xctx->xts.block2 = (block128_f)AES_encrypt;

1156                 xctx->xts.key1 = &xctx->ks1;
1157                 } while (0);

1159         if (iv)
1160                 {
1161                 xctx->xts.key2 = &xctx->ks2;
1162                 memcpy(ctx->iv, iv, 16);
1163                 }

1165         return 1;
1166         }

1168 static int aes_xts_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
1169                 const unsigned char *in, size_t len)
1170         {
1171         EVP_AES_XTS_CTX *xctx = ctx->cipher_data;
1172         if (!xctx->xts.key1 || !xctx->xts.key2)
1173                 return 0;
1174         if (!out || !in || len<AES_BLOCK_SIZE)
1175                 return 0;
1176 #ifdef OPENSSL_FIPS
1177         /* Requirement of SP800-38E */
1178         if (FIPS_module_mode() && !(ctx->flags & EVP_CIPH_FLAG_NON_FIPS_ALLOW) &
1179                 (len > (1UL<<20)*16))
1180                 {
1181                 EVPerr(EVP_F_AES_XTS_CIPHER, EVP_R_TOO_LARGE);
1182                 return 0;
1183                 }
```

```
#endif
        if (xctx->stream)
                (*xctx->stream)(in, out, len,
                                xctx->xts.key1, xctx->xts.key2, ctx->iv);
        else if (CRYPTO_xts128_encrypt(&xctx->xts, ctx->iv, in, out, len,
                                                                ctx->encrypt))
                return 0;
        return 1;
        }

#define aes_xts_cleanup NULL

#define XTS_FLAGS       (EVP_CIPH_FLAG_DEFAULT_ASN1 | EVP_CIPH_CUSTOM_IV \
                         | EVP_CIPH_ALWAYS_CALL_INIT | EVP_CIPH_CTRL_INIT \
                         | EVP_CIPH_CUSTOM_COPY)

BLOCK_CIPHER_custom(NID_aes,128,1,16,xts,XTS,EVP_CIPH_FLAG_FIPS|XTS_FLAGS)
BLOCK_CIPHER_custom(NID_aes,256,1,16,xts,XTS,EVP_CIPH_FLAG_FIPS|XTS_FLAGS)

static int aes_ccm_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr)
        {
        EVP_AES_CCM_CTX *cctx = c->cipher_data;
        switch (type)
                {
        case EVP_CTRL_INIT:
                cctx->key_set = 0;
                cctx->iv_set = 0;
                cctx->L = 8;
                cctx->M = 12;
                cctx->tag_set = 0;
                cctx->len_set = 0;
                return 1;

        case EVP_CTRL_CCM_SET_IVLEN:
                arg = 15 - arg;
        case EVP_CTRL_CCM_SET_L:
                if (arg < 2 || arg > 8)
                        return 0;
                cctx->L = arg;
                return 1;

        case EVP_CTRL_CCM_SET_TAG:
                if ((arg & 1) || arg < 4 || arg > 16)
                        return 0;
                if ((c->encrypt && ptr) || (!c->encrypt && !ptr))
                        return 0;
                if (ptr)
                        {
                        cctx->tag_set = 1;
                        memcpy(c->buf, ptr, arg);
                        }
                cctx->M = arg;
                return 1;

        case EVP_CTRL_CCM_GET_TAG:
                if (!c->encrypt || !cctx->tag_set)
                        return 0;
                if(!CRYPTO_ccm128_tag(&cctx->ccm, ptr, (size_t)arg))
                        return 0;
                cctx->tag_set = 0;
                cctx->iv_set = 0;
                cctx->len_set = 0;
                return 1;

        case EVP_CTRL_COPY:
                {
```



```
                        EVP_CIPHER_CTX *out = ptr;
                        EVP_AES_CCM_CTX *cctx_out = out->cipher_data;
                        if (cctx->ccm.key)
                                {
                                if (cctx->ccm.key != &cctx->ks)
                                        return 0;
                                cctx_out->ccm.key = &cctx_out->ks;
                                }
                        return 1;
                }

        default:
                return -1;

                }
        }

static int aes_ccm_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                        const unsigned char *iv, int enc)
        {
        EVP_AES_CCM_CTX *cctx = ctx->cipher_data;
        if (!iv && !key)
                return 1;
        if (key) do
                {
#ifdef VPAES_CAPABLE
                if (VPAES_CAPABLE)
                        {
                        vpaes_set_encrypt_key(key, ctx->key_len*8, &cctx->ks);
                        CRYPTO_ccm128_init(&cctx->ccm, cctx->M, cctx->L,
                                           &cctx->ks, (block128_f)vpaes_encrypt);
                        cctx->str = NULL;
                        cctx->key_set = 1;
                        break;
                        }
#endif
                AES_set_encrypt_key(key, ctx->key_len * 8, &cctx->ks);
                CRYPTO_ccm128_init(&cctx->ccm, cctx->M, cctx->L,
                                   &cctx->ks, (block128_f)AES_encrypt);
                cctx->str = NULL;
                cctx->key_set = 1;
                } while (0);
        if (iv)
                {
                memcpy(ctx->iv, iv, 15 - cctx->L);
                cctx->iv_set = 1;
                }
        return 1;
        }

static int aes_ccm_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                const unsigned char *in, size_t len)
        {
        EVP_AES_CCM_CTX *cctx = ctx->cipher_data;
        CCM128_CONTEXT *ccm = &cctx->ccm;
        /* If not set up, return error */
        if (!cctx->iv_set && !cctx->key_set)
                return -1;
        if (!ctx->encrypt && !cctx->tag_set)
                return -1;
        if (!out)
                {
                if (!in)
                        {
                        if (CRYPTO_ccm128_setiv(ccm, ctx->iv, 15 - cctx->L,len))
                                return -1;
```

```
1316                        cctx->len_set = 1;
1317                        return len;
1318                        }
1319                /* If have AAD need message length */
1320                if (!cctx->len_set && len)
1321                        return -1;
1322                CRYPTO_ccm128_aad(ccm, in, len);
1323                return len;
1324                }
1325        /* EVP_*Final() doesn't return any data */
1326        if (!in)
1327                return 0;
1328        /* If not set length yet do it */
1329        if (!cctx->len_set)
1330                {
1331                if (CRYPTO_ccm128_setiv(ccm, ctx->iv, 15 - cctx->L, len))
1332                        return -1;
1333                cctx->len_set = 1;
1334                }
1335        if (ctx->encrypt)
1336                {
1337                if (cctx->str ? CRYPTO_ccm128_encrypt_ccm64(ccm, in, out, len,
1338                                                cctx->str) :
1339                                CRYPTO_ccm128_encrypt(ccm, in, out, len))
1340                        return -1;
1341                cctx->tag_set = 1;
1342                return len;
1343                }
1344        else
1345                {
1346                int rv = -1;
1347                if (cctx->str ? !CRYPTO_ccm128_decrypt_ccm64(ccm, in, out, len,
1348                                                cctx->str) :
1349                                !CRYPTO_ccm128_decrypt(ccm, in, out, len))
1350                        {
1351                        unsigned char tag[16];
1352                        if (CRYPTO_ccm128_tag(ccm, tag, cctx->M))
1353                                {
1354                                if (!memcmp(tag, ctx->buf, cctx->M))
1355                                        rv = len;
1356                                }
1357                        }
1358                if (rv == -1)
1359                        OPENSSL_cleanse(out, len);
1360                cctx->iv_set = 0;
1361                cctx->tag_set = 0;
1362                cctx->len_set = 0;
1363                return rv;
1364                }

1366        }

1368 #define aes_ccm_cleanup NULL

1370 BLOCK_CIPHER_custom(NID_aes,128,1,12,ccm,CCM,EVP_CIPH_FLAG_FIPS|CUSTOM_FLAGS)
1371 BLOCK_CIPHER_custom(NID_aes,192,1,12,ccm,CCM,EVP_CIPH_FLAG_FIPS|CUSTOM_FLAGS)
1372 BLOCK_CIPHER_custom(NID_aes,256,1,12,ccm,CCM,EVP_CIPH_FLAG_FIPS|CUSTOM_FLAGS)

1374 #endif
1375 #endif
1376 #endif /* ! codereview */
```

```
**********************************************************
   15521 Wed Aug 13 19:52:42 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_aes_cbc_hmac_sha1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2011-2013 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/opensslconf.h>

#include <stdio.h>
#include <string.h>

#if !defined(OPENSSL_NO_AES) && !defined(OPENSSL_NO_SHA1)

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/aes.h>
#include <openssl/sha.h>
#include "evp_locl.h"
```



```
#ifndef EVP_CIPH_FLAG_AEAD_CIPHER
#define EVP_CIPH_FLAG_AEAD_CIPHER       0x200000
#define EVP_CTRL_AEAD_TLS1_AAD          0x16
#define EVP_CTRL_AEAD_SET_MAC_KEY       0x17
#endif

#if !defined(EVP_CIPH_FLAG_DEFAULT_ASN1)
#define EVP_CIPH_FLAG_DEFAULT_ASN1 0
#endif

#define TLS1_1_VERSION 0x0302

typedef struct
    {
    AES_KEY             ks;
    SHA_CTX             head,tail,md;
    size_t              payload_length; /* AAD length in decrypt case */
    union {
        unsigned int    tls_ver;
        unsigned char   tls_aad[16];    /* 13 used */
    } aux;
    } EVP_AES_HMAC_SHA1;

#define NO_PAYLOAD_LENGTH       ((size_t)-1)

#if     defined(AES_ASM) &&     ( \
        defined(__x86_64)       || defined(__x86_64__)  || \
        defined(_M_AMD64)       || defined(_M_X64)      || \
        defined(__INTEL__)      )

#if defined(__GNUC__) && __GNUC__>=2 && !defined(PEDANTIC)
# define BSWAP(x) ({ unsigned int r=(x); __asm__ ("bswapl %0":"=r"(r):"0"(r)); r
#endif

extern unsigned int OPENSSL_ia32cap_P[2];
#define AESNI_CAPABLE   (1<<(57-32))

int aesni_set_encrypt_key(const unsigned char *userKey, int bits,
                                AES_KEY *key);
int aesni_set_decrypt_key(const unsigned char *userKey, int bits,
                                AES_KEY *key);

void aesni_cbc_encrypt(const unsigned char *in,
                           unsigned char *out,
                           size_t length,
                           const AES_KEY *key,
                           unsigned char *ivec, int enc);

void aesni_cbc_sha1_enc (const void *inp, void *out, size_t blocks,
                const AES_KEY *key, unsigned char iv[16],
                SHA_CTX *ctx,const void *in0);

#define data(ctx) ((EVP_AES_HMAC_SHA1 *)(ctx)->cipher_data)

static int aesni_cbc_hmac_sha1_init_key(EVP_CIPHER_CTX *ctx,
                        const unsigned char *inkey,
                        const unsigned char *iv, int enc)
        {
        EVP_AES_HMAC_SHA1 *key = data(ctx);
        int ret;

        if (enc)
                ret=aesni_set_encrypt_key(inkey,ctx->key_len*8,&key->ks);
        else
                ret=aesni_set_decrypt_key(inkey,ctx->key_len*8,&key->ks);
```

```
129         SHA1_Init(&key->head);  /* handy when benchmarking */
130         key->tail = key->head;
131         key->md   = key->head;

133         key->payload_length = NO_PAYLOAD_LENGTH;

135         return ret<0?0:1;
136         }

138 #define STITCHED_CALL

140 #if !defined(STITCHED_CALL)
141 #define aes_off 0
142 #endif

144 void sha1_block_data_order (void *c,const void *p,size_t len);

146 static void sha1_update(SHA_CTX *c,const void *data,size_t len)
147 {       const unsigned char *ptr = data;
148         size_t res;

150         if ((res = c->num)) {
151                 res = SHA_CBLOCK-res;
152                 if (len<res) res=len;
153                 SHA1_Update (c,ptr,res);
154                 ptr += res;
155                 len -= res;
156         }

158         res = len % SHA_CBLOCK;
159         len -= res;

161         if (len) {
162                 sha1_block_data_order(c,ptr,len/SHA_CBLOCK);

164                 ptr += len;
165                 c->Nh += len>>29;
166                 c->Nl += len<<=3;
167                 if (c->Nl<(unsigned int)len) c->Nh++;
168         }

170         if (res)
171                 SHA1_Update(c,ptr,res);
172 }

174 #ifdef SHA1_Update
175 #undef SHA1_Update
176 #endif
177 #define SHA1_Update sha1_update

179 static int aesni_cbc_hmac_sha1_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
180                         const unsigned char *in, size_t len)
181         {
182         EVP_AES_HMAC_SHA1 *key = data(ctx);
183         unsigned int l;
184         size_t  plen = key->payload_length,
185                 iv = 0,         /* explicit IV in TLS 1.1 and later */
186                 sha_off = 0;
187 #if defined(STITCHED_CALL)
188         size_t  aes_off = 0,
189                 blocks;

191         sha_off = SHA_CBLOCK-key->md.num;
192 #endif
```



```
194         key->payload_length = NO_PAYLOAD_LENGTH;

196         if (len%AES_BLOCK_SIZE) return 0;

198         if (ctx->encrypt) {
199                 if (plen==NO_PAYLOAD_LENGTH)
200                         plen = len;
201                 else if (len!=((plen+SHA_DIGEST_LENGTH+AES_BLOCK_SIZE)&-AES_BLOC
202                         return 0;
203                 else if (key->aux.tls_ver >= TLS1_1_VERSION)
204                         iv = AES_BLOCK_SIZE;

206 #if defined(STITCHED_CALL)
207                 if (plen>(sha_off+iv) && (blocks=(plen-(sha_off+iv))/SHA_CBLOCK)
208                         SHA1_Update(&key->md,in+iv,sha_off);

210                         aesni_cbc_sha1_enc(in,out,blocks,&key->ks,
211                                 ctx->iv,&key->md,in+iv+sha_off);
212                         blocks *= SHA_CBLOCK;
213                         aes_off += blocks;
214                         sha_off += blocks;
215                         key->md.Nh += blocks>>29;
216                         key->md.Nl += blocks<<=3;
217                         if (key->md.Nl<(unsigned int)blocks) key->md.Nh++;
218                 } else {
219                         sha_off = 0;
220                 }
221 #endif
222                 sha_off += iv;
223                 SHA1_Update(&key->md,in+sha_off,plen-sha_off);

225                 if (plen!=len)  {       /* "TLS" mode of operation */
226                         if (in!=out)
227                                 memcpy(out+aes_off,in+aes_off,plen-aes_off);

229                         /* calculate HMAC and append it to payload */
230                         SHA1_Final(out+plen,&key->md);
231                         key->md = key->tail;
232                         SHA1_Update(&key->md,out+plen,SHA_DIGEST_LENGTH);
233                         SHA1_Final(out+plen,&key->md);

235                         /* pad the payload|hmac */
236                         plen += SHA_DIGEST_LENGTH;
237                         for (l=len-plen-1;plen<len;plen++) out[plen]=l;
238                         /* encrypt HMAC|padding at once */
239                         aesni_cbc_encrypt(out+aes_off,out+aes_off,len-aes_off,
240                                         &key->ks,ctx->iv,1);
241                 } else {
242                         aesni_cbc_encrypt(in+aes_off,out+aes_off,len-aes_off,
243                                         &key->ks,ctx->iv,1);
244                 }
245         } else {
246                 union { unsigned int  u[SHA_DIGEST_LENGTH/sizeof(unsigned int)];
247                         unsigned char c[32+SHA_DIGEST_LENGTH]; } mac, *pmac;

249                 /* arrange cache line alignment */
250                 pmac = (void *)(((size_t)mac.c+31)&((size_t)0-32));

252                 /* decrypt HMAC|padding at once */
253                 aesni_cbc_encrypt(in,out,len,
254                                 &key->ks,ctx->iv,0);

256                 if (plen) {     /* "TLS" mode of operation */
257                         size_t inp_len, mask, j, i;
258                         unsigned int res, maxpad, pad, bitlen;
259                         int ret = 1;
```

```
260                         union { unsigned int  u[SHA_LBLOCK];
261                                 unsigned char c[SHA_CBLOCK]; }
262                                 *data = (void *)key->md.data;

264                         if ((key->aux.tls_aad[plen-4]<<8|key->aux.tls_aad[plen-3
265                             >= TLS1_1_VERSION)
266                                 iv = AES_BLOCK_SIZE;

268                         if (len<(iv+SHA_DIGEST_LENGTH+1))
269                                 return 0;

271                         /* omit explicit iv */
272                         out += iv;
273                         len -= iv;

275                         /* figure out payload length */
276                         pad = out[len-1];
277                         maxpad = len-(SHA_DIGEST_LENGTH+1);
278                         maxpad |= (255-maxpad)>>(sizeof(maxpad)*8-8);
279                         maxpad &= 255;

281                         inp_len = len - (SHA_DIGEST_LENGTH+pad+1);
282                         mask = (0-((inp_len-len)>>(sizeof(inp_len)*8-1)));
283                         inp_len &= mask;
284                         ret &= (int)mask;

286                         key->aux.tls_aad[plen-2] = inp_len>>8;
287                         key->aux.tls_aad[plen-1] = inp_len;

289                         /* calculate HMAC */
290                         key->md = key->head;
291                         SHA1_Update(&key->md,key->aux.tls_aad,plen);

293 #if 1
294                         len -= SHA_DIGEST_LENGTH;               /* amend mac */
295                         if (len>=(256+SHA_CBLOCK)) {
296                                 j = (len-(256+SHA_CBLOCK))&(0-SHA_CBLOCK);
297                                 j += SHA_CBLOCK-key->md.num;
298                                 SHA1_Update(&key->md,out,j);
299                                 out += j;
300                                 len -= j;
301                                 inp_len -= j;
302                         }

304                         /* but pretend as if we hashed padded payload */
305                         bitlen = key->md.Nl+(inp_len<<3);       /* at most 18 bi
306 #ifdef BSWAP
307                         bitlen = BSWAP(bitlen);
308 #else
309                         mac.c[0] = 0;
310                         mac.c[1] = (unsigned char)(bitlen>>16);
311                         mac.c[2] = (unsigned char)(bitlen>>8);
312                         mac.c[3] = (unsigned char)bitlen;
313                         bitlen = mac.u[0];
314 #endif

316                         pmac->u[0]=0;
317                         pmac->u[1]=0;
318                         pmac->u[2]=0;
319                         pmac->u[3]=0;
320                         pmac->u[4]=0;

322                         for (res=key->md.num, j=0;j<len;j++) {
323                                 size_t c = out[j];
324                                 mask = (j-inp_len)>>(sizeof(j)*8-8);
325                                 c &= mask;
```



```
326                                 c |= 0x80&~mask&~((inp_len-j)>>(sizeof(j)*8-8));
327                                 data->c[res++]=(unsigned char)c;

329                                 if (res!=SHA_CBLOCK) continue;

331                                 /* j is not incremented yet */
332                                 mask = 0-((inp_len+7-j)>>(sizeof(j)*8-1));
333                                 data->u[SHA_LBLOCK-1] |= bitlen&mask;
334                                 sha1_block_data_order(&key->md,data,1);
335                                 mask &= 0-((j-inp_len-72)>>(sizeof(j)*8-1));
336                                 pmac->u[0] |= key->md.h0 & mask;
337                                 pmac->u[1] |= key->md.h1 & mask;
338                                 pmac->u[2] |= key->md.h2 & mask;
339                                 pmac->u[3] |= key->md.h3 & mask;
340                                 pmac->u[4] |= key->md.h4 & mask;
341                                 res=0;
342                         }

344                         for(i=res;i<SHA_CBLOCK;i++,j++) data->c[i]=0;

346                         if (res>SHA_CBLOCK-8) {
347                                 mask = 0-((inp_len+8-j)>>(sizeof(j)*8-1));
348                                 data->u[SHA_LBLOCK-1] |= bitlen&mask;
349                                 sha1_block_data_order(&key->md,data,1);
350                                 mask &= 0-((j-inp_len-73)>>(sizeof(j)*8-1));
351                                 pmac->u[0] |= key->md.h0 & mask;
352                                 pmac->u[1] |= key->md.h1 & mask;
353                                 pmac->u[2] |= key->md.h2 & mask;
354                                 pmac->u[3] |= key->md.h3 & mask;
355                                 pmac->u[4] |= key->md.h4 & mask;

357                                 memset(data,0,SHA_CBLOCK);
358                                 j+=64;
359                         }
360                         data->u[SHA_LBLOCK-1] = bitlen;
361                         sha1_block_data_order(&key->md,data,1);
362                         mask = 0-((j-inp_len-73)>>(sizeof(j)*8-1));
363                         pmac->u[0] |= key->md.h0 & mask;
364                         pmac->u[1] |= key->md.h1 & mask;
365                         pmac->u[2] |= key->md.h2 & mask;
366                         pmac->u[3] |= key->md.h3 & mask;
367                         pmac->u[4] |= key->md.h4 & mask;

369 #ifdef BSWAP
370                         pmac->u[0] = BSWAP(pmac->u[0]);
371                         pmac->u[1] = BSWAP(pmac->u[1]);
372                         pmac->u[2] = BSWAP(pmac->u[2]);
373                         pmac->u[3] = BSWAP(pmac->u[3]);
374                         pmac->u[4] = BSWAP(pmac->u[4]);
375 #else
376                         for (i=0;i<5;i++) {
377                                 res = pmac->u[i];
378                                 pmac->c[4*i+0]=(unsigned char)(res>>24);
379                                 pmac->c[4*i+1]=(unsigned char)(res>>16);
380                                 pmac->c[4*i+2]=(unsigned char)(res>>8);
381                                 pmac->c[4*i+3]=(unsigned char)res;
382                         }
383 #endif
384                         len += SHA_DIGEST_LENGTH;
385 #else
386                         SHA1_Update(&key->md,out,inp_len);
387                         res = key->md.num;
388                         SHA1_Final(pmac->c,&key->md);

390                         {
391                         unsigned int inp_blocks, pad_blocks;
```

```
393                         /* but pretend as if we hashed padded payload */
394                         inp_blocks = 1+((SHA_CBLOCK-9-res)>>(sizeof(res)*8-1));
395                         res += (unsigned int)(len-inp_len);
396                         pad_blocks = res / SHA_CBLOCK;
397                         res %= SHA_CBLOCK;
398                         pad_blocks += 1+((SHA_CBLOCK-9-res)>>(sizeof(res)*8-1));
399                         for (;inp_blocks<pad_blocks;inp_blocks++)
400                                 sha1_block_data_order(&key->md,data,1);
401                         }
402 #endif
403                         key->md = key->tail;
404                         SHA1_Update(&key->md,pmac->c,SHA_DIGEST_LENGTH);
405                         SHA1_Final(pmac->c,&key->md);

407                         /* verify HMAC */
408                         out += inp_len;
409                         len -= inp_len;
410 #if 1
411                         {
412                         unsigned char *p = out+len-1-maxpad-SHA_DIGEST_LENGTH;
413                         size_t off = out-p;
414                         unsigned int c, cmask;

416                         maxpad += SHA_DIGEST_LENGTH;
417                         for (res=0,i=0,j=0;j<maxpad;j++) {
418                                 c = p[j];
419                                 cmask = ((int)(j-off-SHA_DIGEST_LENGTH))>>(sizeo
420                                 res |= (c^pad)&~cmask;  /* ... and padding */
421                                 cmask &= ((int)(off-1-j))>>(sizeof(int)*8-1);
422                                 res |= (c^pmac->c[i])&cmask;
423                                 i += 1&cmask;
424                         }
425                         maxpad -= SHA_DIGEST_LENGTH;

427                         res = 0-((0-res)>>(sizeof(res)*8-1));
428                         ret &= (int)~res;
429                         }
430 #else
431                         for (res=0,i=0;i<SHA_DIGEST_LENGTH;i++)
432                                 res |= out[i]^pmac->c[i];
433                         res = 0-((0-res)>>(sizeof(res)*8-1));
434                         ret &= (int)~res;

436                         /* verify padding */
437                         pad = (pad&~res) | (maxpad&res);
438                         out = out+len-1-pad;
439                         for (res=0,i=0;i<pad;i++)
440                                 res |= out[i]^pad;

442                         res = (0-res)>>(sizeof(res)*8-1);
443                         ret &= (int)~res;
444 #endif
445                         return ret;
446                 } else {
447                         SHA1_Update(&key->md,out,len);
448                 }
449         }

451         return 1;
452         }

454 static int aesni_cbc_hmac_sha1_ctrl(EVP_CIPHER_CTX *ctx, int type, int arg, void
455         {
456         EVP_AES_HMAC_SHA1 *key = data(ctx);
```



```
458         switch (type)
459                 {
460         case EVP_CTRL_AEAD_SET_MAC_KEY:
461                 {
462                 unsigned int  i;
463                 unsigned char hmac_key[64];

465                 memset (hmac_key,0,sizeof(hmac_key));

467                 if (arg > (int)sizeof(hmac_key)) {
468                         SHA1_Init(&key->head);
469                         SHA1_Update(&key->head,ptr,arg);
470                         SHA1_Final(hmac_key,&key->head);
471                 } else {
472                         memcpy(hmac_key,ptr,arg);
473                 }

475                 for (i=0;i<sizeof(hmac_key);i++)
476                         hmac_key[i] ^= 0x36;            /* ipad */
477                 SHA1_Init(&key->head);
478                 SHA1_Update(&key->head,hmac_key,sizeof(hmac_key));

480                 for (i=0;i<sizeof(hmac_key);i++)
481                         hmac_key[i] ^= 0x36^0x5c;       /* opad */
482                 SHA1_Init(&key->tail);
483                 SHA1_Update(&key->tail,hmac_key,sizeof(hmac_key));

485                 OPENSSL_cleanse(hmac_key,sizeof(hmac_key));

487                 return 1;
488                 }
489         case EVP_CTRL_AEAD_TLS1_AAD:
490                 {
491                 unsigned char *p=ptr;
492                 unsigned int   len=p[arg-2]<<8|p[arg-1];

494                 if (ctx->encrypt)
495                         {
496                         key->payload_length = len;
497                         if ((key->aux.tls_ver=p[arg-4]<<8|p[arg-3]) >= TLS1_1_VE
498                                 len -= AES_BLOCK_SIZE;
499                                 p[arg-2] = len>>8;
500                                 p[arg-1] = len;
501                         }
502                         key->md = key->head;
503                         SHA1_Update(&key->md,p,arg);

505                         return (int)(((len+SHA_DIGEST_LENGTH+AES_BLOCK_SIZE)&-AE
506                                 - len);
507                         }
508                 else
509                         {
510                         if (arg>13) arg = 13;
511                         memcpy(key->aux.tls_aad,ptr,arg);
512                         key->payload_length = arg;

514                         return SHA_DIGEST_LENGTH;
515                         }
516                 }
517         default:
518                 return -1;
519                 }
520         }

522 static EVP_CIPHER aesni_128_cbc_hmac_sha1_cipher =
523         {
```

```
#ifdef NID_aes_128_cbc_hmac_sha1
        NID_aes_128_cbc_hmac_sha1,
#else
        NID_undef,
#endif
        16,16,16,
        EVP_CIPH_CBC_MODE|EVP_CIPH_FLAG_DEFAULT_ASN1|EVP_CIPH_FLAG_AEAD_CIPHER,
        aesni_cbc_hmac_sha1_init_key,
        aesni_cbc_hmac_sha1_cipher,
        NULL,
        sizeof(EVP_AES_HMAC_SHA1),
        EVP_CIPH_FLAG_DEFAULT_ASN1?NULL:EVP_CIPHER_set_asn1_iv,
        EVP_CIPH_FLAG_DEFAULT_ASN1?NULL:EVP_CIPHER_get_asn1_iv,
        aesni_cbc_hmac_sha1_ctrl,
        NULL
        };

static EVP_CIPHER aesni_256_cbc_hmac_sha1_cipher =
        {
#ifdef NID_aes_256_cbc_hmac_sha1
        NID_aes_256_cbc_hmac_sha1,
#else
        NID_undef,
#endif
        16,32,16,
        EVP_CIPH_CBC_MODE|EVP_CIPH_FLAG_DEFAULT_ASN1|EVP_CIPH_FLAG_AEAD_CIPHER,
        aesni_cbc_hmac_sha1_init_key,
        aesni_cbc_hmac_sha1_cipher,
        NULL,
        sizeof(EVP_AES_HMAC_SHA1),
        EVP_CIPH_FLAG_DEFAULT_ASN1?NULL:EVP_CIPHER_set_asn1_iv,
        EVP_CIPH_FLAG_DEFAULT_ASN1?NULL:EVP_CIPHER_get_asn1_iv,
        aesni_cbc_hmac_sha1_ctrl,
        NULL
        };

const EVP_CIPHER *EVP_aes_128_cbc_hmac_sha1(void)
        {
        return(OPENSSL_ia32cap_P[1]&AESNI_CAPABLE?
                &aesni_128_cbc_hmac_sha1_cipher:NULL);
        }

const EVP_CIPHER *EVP_aes_256_cbc_hmac_sha1(void)
        {
        return(OPENSSL_ia32cap_P[1]&AESNI_CAPABLE?
                &aesni_256_cbc_hmac_sha1_cipher:NULL);
        }
#else
const EVP_CIPHER *EVP_aes_128_cbc_hmac_sha1(void)
        {
        return NULL;
        }
const EVP_CIPHER *EVP_aes_256_cbc_hmac_sha1(void)
        {
        return NULL;
        }
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3942 Wed Aug 13 19:52:42 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_bf.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_bf.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#ifndef OPENSSL_NO_BF
```



```
#include <openssl/evp.h>
#include "evp_locl.h"
#include <openssl/objects.h>
#include <openssl/blowfish.h>

static int bf_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                       const unsigned char *iv, int enc);

typedef struct
        {
        BF_KEY ks;
        } EVP_BF_KEY;

#define data(ctx)       EVP_C_DATA(EVP_BF_KEY,ctx)

IMPLEMENT_BLOCK_CIPHER(bf, ks, BF, EVP_BF_KEY, NID_bf, 8, 16, 8, 64,
                        EVP_CIPH_VARIABLE_LENGTH, bf_init_key, NULL,
                        EVP_CIPHER_set_asn1_iv, EVP_CIPHER_get_asn1_iv, NULL)

static int bf_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                       const unsigned char *iv, int enc)
        {
        BF_set_key(&data(ctx)->ks,EVP_CIPHER_CTX_key_length(ctx),key);
        return 1;
        }

#endif
#endif /* ! codereview */
```

```
**********************************************************
    4489 Wed Aug 13 19:52:43 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_camellia.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_camellia.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/opensslconf.h>
#ifndef OPENSSL_NO_CAMELLIA
#include <openssl/evp.h>
#include <openssl/err.h>
#include <string.h>
#include <assert.h>
```



```
#include <openssl/camellia.h>
#include "evp_locl.h"

static int camellia_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
        const unsigned char *iv, int enc);

/* Camellia subkey Structure */
typedef struct
        {
        CAMELLIA_KEY ks;
        } EVP_CAMELLIA_KEY;

/* Attribute operation for Camellia */
#define data(ctx)       EVP_C_DATA(EVP_CAMELLIA_KEY,ctx)

IMPLEMENT_BLOCK_CIPHER(camellia_128, ks, Camellia, EVP_CAMELLIA_KEY,
        NID_camellia_128, 16, 16, 16, 128,
        0, camellia_init_key, NULL,
        EVP_CIPHER_set_asn1_iv,
        EVP_CIPHER_get_asn1_iv,
        NULL)
IMPLEMENT_BLOCK_CIPHER(camellia_192, ks, Camellia, EVP_CAMELLIA_KEY,
        NID_camellia_192, 16, 24, 16, 128,
        0, camellia_init_key, NULL,
        EVP_CIPHER_set_asn1_iv,
        EVP_CIPHER_get_asn1_iv,
        NULL)
IMPLEMENT_BLOCK_CIPHER(camellia_256, ks, Camellia, EVP_CAMELLIA_KEY,
        NID_camellia_256, 16, 32, 16, 128,
        0, camellia_init_key, NULL,
        EVP_CIPHER_set_asn1_iv,
        EVP_CIPHER_get_asn1_iv,
        NULL)

#define IMPLEMENT_CAMELLIA_CFBR(ksize,cbits)    IMPLEMENT_CFBR(camellia,Camellia

IMPLEMENT_CAMELLIA_CFBR(128,1)
IMPLEMENT_CAMELLIA_CFBR(192,1)
IMPLEMENT_CAMELLIA_CFBR(256,1)

IMPLEMENT_CAMELLIA_CFBR(128,8)
IMPLEMENT_CAMELLIA_CFBR(192,8)
IMPLEMENT_CAMELLIA_CFBR(256,8)



/* The subkey for Camellia is generated. */
static int camellia_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
        const unsigned char *iv, int enc)
        {
        int ret;

        ret=Camellia_set_key(key, ctx->key_len * 8, ctx->cipher_data);

        if(ret < 0)
                {
                EVPerr(EVP_F_CAMELLIA_INIT_KEY,EVP_R_CAMELLIA_KEY_SETUP_FAILED);
                return 0;
                }

        return 1;
        }

#else

# ifdef PEDANTIC
```

```
128 static void *dummy=&dummy;
129 # endif

131 #endif
132 #endif /* ! codereview */
```

```
**********************************************************
    3972 Wed Aug 13 19:52:43 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_cast.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_cast.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_CAST
#include <openssl/evp.h>
#include <openssl/objects.h>
#include "evp_locl.h"
#include <openssl/cast.h>

static int cast_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                         const unsigned char *iv,int enc);

typedef struct
        {
        CAST_KEY ks;
        } EVP_CAST_KEY;

#define data(ctx)       EVP_C_DATA(EVP_CAST_KEY,ctx)

IMPLEMENT_BLOCK_CIPHER(cast5, ks, CAST, EVP_CAST_KEY,
                         NID_cast5, 8, CAST_KEY_LENGTH, 8, 64,
                         EVP_CIPH_VARIABLE_LENGTH, cast_init_key, NULL,
                         EVP_CIPHER_set_asn1_iv, EVP_CIPHER_get_asn1_iv, NULL)

static int cast_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                         const unsigned char *iv, int enc)
        {
        CAST_set_key(&data(ctx)->ks,EVP_CIPHER_CTX_key_length(ctx),key);
        return 1;
        }

#endif
#endif /* ! codereview */
```

```
**********************************************************
    7496 Wed Aug 13 19:52:43 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_des.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_des.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#ifndef OPENSSL_NO_DES
```



```
#include <openssl/evp.h>
#include <openssl/objects.h>
#include "evp_locl.h"
#include <openssl/des.h>
#include <openssl/rand.h>

static int des_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                        const unsigned char *iv, int enc);
static int des_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr);

/* Because of various casts and different names can't use IMPLEMENT_BLOCK_CIPHER

static int des_ecb_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                          const unsigned char *in, size_t inl)
{
        BLOCK_CIPHER_ecb_loop()
                DES_ecb_encrypt((DES_cblock *)(in + i), (DES_cblock *)(out + i),
        return 1;
}

static int des_ofb_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                          const unsigned char *in, size_t inl)
{
        while(inl>=EVP_MAXCHUNK)
                {
                DES_ofb64_encrypt(in, out, (long)EVP_MAXCHUNK, ctx->cipher_data,
                                  (DES_cblock *)ctx->iv, &ctx->num);
                inl-=EVP_MAXCHUNK;
                in +=EVP_MAXCHUNK;
                out+=EVP_MAXCHUNK;
                }
        if (inl)
                DES_ofb64_encrypt(in, out, (long)inl, ctx->cipher_data,
                                  (DES_cblock *)ctx->iv, &ctx->num);
        return 1;
}

static int des_cbc_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                          const unsigned char *in, size_t inl)
{
        while(inl>=EVP_MAXCHUNK)
                {
                DES_ncbc_encrypt(in, out, (long)EVP_MAXCHUNK, ctx->cipher_data,
                                 (DES_cblock *)ctx->iv, ctx->encrypt);
                inl-=EVP_MAXCHUNK;
                in +=EVP_MAXCHUNK;
                out+=EVP_MAXCHUNK;
                }
        if (inl)
                DES_ncbc_encrypt(in, out, (long)inl, ctx->cipher_data,
                                 (DES_cblock *)ctx->iv, ctx->encrypt);
        return 1;
}

static int des_cfb64_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                            const unsigned char *in, size_t inl)
{
        while(inl>=EVP_MAXCHUNK)
                {
                DES_cfb64_encrypt(in,out, (long)EVP_MAXCHUNK, ctx->cipher_data,
                                  (DES_cblock *)ctx->iv, &ctx->num, ctx->encrypt);
                inl-=EVP_MAXCHUNK;
                in +=EVP_MAXCHUNK;
                out+=EVP_MAXCHUNK;
                }
        if (inl)
```

```
128                 DES_cfb64_encrypt(in, out, (long)inl, ctx->cipher_data,
129                                   (DES_cblock *)ctx->iv, &ctx->num, ctx->encrypt);
130         return 1;
131 }

133 /* Although we have a CFB-r implementation for DES, it doesn't pack the right
134    way, so wrap it here */
135 static int des_cfb1_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
136                            const unsigned char *in, size_t inl)
137     {
138     size_t n,chunk=EVP_MAXCHUNK/8;
139     unsigned char c[1],d[1];

141     if (inl<chunk) chunk=inl;

143     while (inl && inl>=chunk)
144         {
145         for(n=0 ; n < chunk*8; ++n)
146             {
147             c[0]=(in[n/8]&(1 << (7-n%8))) ? 0x80 : 0;
148             DES_cfb_encrypt(c,d,1,1,ctx->cipher_data,(DES_cblock *)ctx->iv,
149                             ctx->encrypt);
150             out[n/8]=(out[n/8]&~(0x80 >> (unsigned int)(n%8))) |
151                     ((d[0]&0x80) >> (unsigned int)(n%8));
152             }
153         inl-=chunk;
154         in +=chunk;
155         out+=chunk;
156         if (inl<chunk) chunk=inl;
157         }

159     return 1;
160     }

162 static int des_cfb8_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
163                            const unsigned char *in, size_t inl)
164     {
165     while (inl>=EVP_MAXCHUNK)
166         {
167         DES_cfb_encrypt(in,out,8,(long)EVP_MAXCHUNK,ctx->cipher_data,
168                         (DES_cblock *)ctx->iv,ctx->encrypt);
169         inl-=EVP_MAXCHUNK;
170         in +=EVP_MAXCHUNK;
171         out+=EVP_MAXCHUNK;
172         }
173     if (inl)
174         DES_cfb_encrypt(in,out,8,(long)inl,ctx->cipher_data,
175                         (DES_cblock *)ctx->iv,ctx->encrypt);
176     return 1;
177     }

179 BLOCK_CIPHER_defs(des, DES_key_schedule, NID_des, 8, 8, 8, 64,
180                        EVP_CIPH_RAND_KEY, des_init_key, NULL,
181                        EVP_CIPHER_set_asn1_iv,
182                        EVP_CIPHER_get_asn1_iv,
183                        des_ctrl)

185 BLOCK_CIPHER_def_cfb(des,DES_key_schedule,NID_des,8,8,1,
186                      EVP_CIPH_RAND_KEY, des_init_key,NULL,
187                      EVP_CIPHER_set_asn1_iv,
188                      EVP_CIPHER_get_asn1_iv,des_ctrl)

190 BLOCK_CIPHER_def_cfb(des,DES_key_schedule,NID_des,8,8,8,
191                      EVP_CIPH_RAND_KEY,des_init_key,NULL,
192                      EVP_CIPHER_set_asn1_iv,
193                      EVP_CIPHER_get_asn1_iv,des_ctrl)
```



```
195 static int des_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
196                          const unsigned char *iv, int enc)
197         {
198         DES_cblock *deskey = (DES_cblock *)key;
199 #ifdef EVP_CHECK_DES_KEY
200         if(DES_set_key_checked(deskey,ctx->cipher_data) != 0)
201                 return 0;
202 #else
203         DES_set_key_unchecked(deskey,ctx->cipher_data);
204 #endif
205         return 1;
206         }

208 static int des_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr)
209         {

211         switch(type)
212                 {
213         case EVP_CTRL_RAND_KEY:
214                 if (RAND_bytes(ptr, 8) <= 0)
215                         return 0;
216                 DES_set_odd_parity((DES_cblock *)ptr);
217                 return 1;

219         default:
220                 return -1;
221                 }
222         }

224 #endif
225 #endif /* ! codereview */
```

```
**********************************************************
   10301 Wed Aug 13 19:52:43 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_des3.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_des3.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#ifndef OPENSSL_NO_DES
```



```
#include <openssl/evp.h>
#include <openssl/objects.h>
#include "evp_locl.h"
#include <openssl/des.h>
#include <openssl/rand.h>

#ifndef OPENSSL_FIPS

static int des_ede_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                            const unsigned char *iv,int enc);

static int des_ede3_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                             const unsigned char *iv,int enc);

static int des3_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr);

typedef struct
    {
    DES_key_schedule ks1;/* key schedule */
    DES_key_schedule ks2;/* key schedule (for ede) */
    DES_key_schedule ks3;/* key schedule (for ede3) */
    } DES_EDE_KEY;

#define data(ctx) ((DES_EDE_KEY *)(ctx)->cipher_data)

/* Because of various casts and different args can't use IMPLEMENT_BLOCK_CIPHER

static int des_ede_ecb_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                              const unsigned char *in, size_t inl)
{
        BLOCK_CIPHER_ecb_loop()
                DES_ecb3_encrypt((const_DES_cblock *)(in + i),
                                 (DES_cblock *)(out + i),
                                 &data(ctx)->ks1, &data(ctx)->ks2,
                                 &data(ctx)->ks3,
                                 ctx->encrypt);
        return 1;
}

static int des_ede_ofb_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                              const unsigned char *in, size_t inl)
{
        while (inl>=EVP_MAXCHUNK)
                {
                DES_ede3_ofb64_encrypt(in, out, (long)EVP_MAXCHUNK,
                                 &data(ctx)->ks1, &data(ctx)->ks2, &data(ctx)->ks3
                                 (DES_cblock *)ctx->iv, &ctx->num);
                inl-=EVP_MAXCHUNK;
                in +=EVP_MAXCHUNK;
                out+=EVP_MAXCHUNK;
                }
        if (inl)
                DES_ede3_ofb64_encrypt(in, out, (long)inl,
                                 &data(ctx)->ks1, &data(ctx)->ks2, &data(ctx)->ks
                                (DES_cblock *)ctx->iv, &ctx->num);

        return 1;
}

static int des_ede_cbc_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                              const unsigned char *in, size_t inl)
{
#ifdef KSSL_DEBUG
        {
        int i;
        char *cp;
```

```
128         printf("des_ede_cbc_cipher(ctx=%lx, buflen=%d)\n", ctx, ctx->buf_len);
129         printf("\t iv= ");
130         for(i=0;i<8;i++)
131                 printf("%02X",ctx->iv[i]);
132         printf("\n");
133         }
134 #endif    /* KSSL_DEBUG */
135         while (inl>=EVP_MAXCHUNK)
136                 {
137                 DES_ede3_cbc_encrypt(in, out, (long)EVP_MAXCHUNK,
138                                      &data(ctx)->ks1, &data(ctx)->ks2, &data(ctx)->ks3,
139                                      (DES_cblock *)ctx->iv, ctx->encrypt);
140                 inl-=EVP_MAXCHUNK;
141                 in +=EVP_MAXCHUNK;
142                 out+=EVP_MAXCHUNK;
143                 }
144         if (inl)
145                 DES_ede3_cbc_encrypt(in, out, (long)inl,
146                                      &data(ctx)->ks1, &data(ctx)->ks2, &data(ctx)->ks3,
147                                      (DES_cblock *)ctx->iv, ctx->encrypt);
148         return 1;
149 }

151 static int des_ede_cfb64_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
152                                 const unsigned char *in, size_t inl)
153 {
154         while (inl>=EVP_MAXCHUNK)
155                 {
156                 DES_ede3_cfb64_encrypt(in, out, (long)EVP_MAXCHUNK,
157                                        &data(ctx)->ks1, &data(ctx)->ks2, &data(ctx)->ks3
158                                        (DES_cblock *)ctx->iv, &ctx->num, ctx->encrypt);
159                 inl-=EVP_MAXCHUNK;
160                 in +=EVP_MAXCHUNK;
161                 out+=EVP_MAXCHUNK;
162                 }
163         if (inl)
164                 DES_ede3_cfb64_encrypt(in, out, (long)inl,
165                                        &data(ctx)->ks1, &data(ctx)->ks2, &data(ctx)->ks3
166                                        (DES_cblock *)ctx->iv, &ctx->num, ctx->encrypt);
167         return 1;
168 }

170 /* Although we have a CFB-r implementation for 3-DES, it doesn't pack the right
171    way, so wrap it here */
172 static int des_ede3_cfb1_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
173                                 const unsigned char *in, size_t inl)
174     {
175     size_t n;
176     unsigned char c[1],d[1];

178     for(n=0 ; n < inl ; ++n)
179         {
180         c[0]=(in[n/8]&(1 << (7-n%8))) ? 0x80 : 0;
181         DES_ede3_cfb_encrypt(c,d,1,1,
182                              &data(ctx)->ks1,&data(ctx)->ks2,&data(ctx)->ks3,
183                              (DES_cblock *)ctx->iv,ctx->encrypt);
184         out[n/8]=(out[n/8]&~(0x80 >> (unsigned int)(n%8))) |
185                  ((d[0]&0x80) >> (unsigned int)(n%8));
186         }

188     return 1;
189     }

191 static int des_ede3_cfb8_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
192                                 const unsigned char *in, size_t inl)
193     {
```



```
194     while (inl>=EVP_MAXCHUNK)
195         {
196         DES_ede3_cfb_encrypt(in,out,8,(long)EVP_MAXCHUNK,
197                              &data(ctx)->ks1,&data(ctx)->ks2,&data(ctx)->ks3,
198                              (DES_cblock *)ctx->iv,ctx->encrypt);
199         inl-=EVP_MAXCHUNK;
200         in +=EVP_MAXCHUNK;
201         out+=EVP_MAXCHUNK;
202         }
203     if (inl)
204         DES_ede3_cfb_encrypt(in,out,8,(long)inl,
205                              &data(ctx)->ks1,&data(ctx)->ks2,&data(ctx)->ks3,
206                              (DES_cblock *)ctx->iv,ctx->encrypt);
207     return 1;
208     }

210 BLOCK_CIPHER_defs(des_ede, DES_EDE_KEY, NID_des_ede, 8, 16, 8, 64,
211                         EVP_CIPH_RAND_KEY, des_ede_init_key, NULL,
212                         EVP_CIPHER_set_asn1_iv,
213                         EVP_CIPHER_get_asn1_iv,
214                         des3_ctrl)

216 #define des_ede3_cfb64_cipher des_ede_cfb64_cipher
217 #define des_ede3_ofb_cipher des_ede_ofb_cipher
218 #define des_ede3_cbc_cipher des_ede_cbc_cipher
219 #define des_ede3_ecb_cipher des_ede_ecb_cipher

221 BLOCK_CIPHER_defs(des_ede3, DES_EDE_KEY, NID_des_ede3, 8, 24, 8, 64,
222                         EVP_CIPH_RAND_KEY, des_ede3_init_key, NULL,
223                         EVP_CIPHER_set_asn1_iv,
224                         EVP_CIPHER_get_asn1_iv,
225                         des3_ctrl)

227 BLOCK_CIPHER_def_cfb(des_ede3,DES_EDE_KEY,NID_des_ede3,24,8,1,
228                      EVP_CIPH_RAND_KEY, des_ede3_init_key,NULL,
229                      EVP_CIPHER_set_asn1_iv,
230                      EVP_CIPHER_get_asn1_iv,
231                      des3_ctrl)

233 BLOCK_CIPHER_def_cfb(des_ede3,DES_EDE_KEY,NID_des_ede3,24,8,8,
234                      EVP_CIPH_RAND_KEY, des_ede3_init_key,NULL,
235                      EVP_CIPHER_set_asn1_iv,
236                      EVP_CIPHER_get_asn1_iv,
237                      des3_ctrl)

239 static int des_ede_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
240                              const unsigned char *iv, int enc)
241         {
242         DES_cblock *deskey = (DES_cblock *)key;
243 #ifdef EVP_CHECK_DES_KEY
244         if (DES_set_key_checked(&deskey[0],&data(ctx)->ks1)
245                 !! DES_set_key_checked(&deskey[1],&data(ctx)->ks2))
246                 return 0;
247 #else
248         DES_set_key_unchecked(&deskey[0],&data(ctx)->ks1);
249         DES_set_key_unchecked(&deskey[1],&data(ctx)->ks2);
250 #endif
251         memcpy(&data(ctx)->ks3,&data(ctx)->ks1,
252               sizeof(data(ctx)->ks1));
253         return 1;
254         }

256 static int des_ede3_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
257                              const unsigned char *iv, int enc)
258         {
259         DES_cblock *deskey = (DES_cblock *)key;
```

```
#ifdef KSSL_DEBUG
        {
        int i;
        printf("des_ede3_init_key(ctx=%lx)\n", ctx);
        printf("\tKEY= ");
        for(i=0;i<24;i++) printf("%02X",key[i]); printf("\n");
        printf("\t IV= ");
        for(i=0;i<8;i++) printf("%02X",iv[i]); printf("\n");
        }
#endif  /* KSSL_DEBUG */

#ifdef EVP_CHECK_DES_KEY
        if (DES_set_key_checked(&deskey[0],&data(ctx)->ks1)
                || DES_set_key_checked(&deskey[1],&data(ctx)->ks2)
                || DES_set_key_checked(&deskey[2],&data(ctx)->ks3))
                return 0;
#else
        DES_set_key_unchecked(&deskey[0],&data(ctx)->ks1);
        DES_set_key_unchecked(&deskey[1],&data(ctx)->ks2);
        DES_set_key_unchecked(&deskey[2],&data(ctx)->ks3);
#endif
        return 1;
        }

static int des3_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr)
        {

        DES_cblock *deskey = ptr;

        switch(type)
                {
        case EVP_CTRL_RAND_KEY:
                if (RAND_bytes(ptr, c->key_len) <= 0)
                        return 0;
                DES_set_odd_parity(deskey);
                if (c->key_len >= 16)
                        DES_set_odd_parity(deskey + 1);
                if (c->key_len >= 24)
                        DES_set_odd_parity(deskey + 2);
                return 1;

        default:
                return -1;
                }
        }

const EVP_CIPHER *EVP_des_ede(void)
{
        return &des_ede_ecb;
}

const EVP_CIPHER *EVP_des_ede3(void)
{
        return &des_ede3_ecb;
}
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4757 Wed Aug 13 19:52:43 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_idea.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_idea.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_IDEA
#include <openssl/evp.h>
#include <openssl/objects.h>
#include "evp_locl.h"
#include <openssl/idea.h>

static int idea_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                         const unsigned char *iv,int enc);

/* NB idea_ecb_encrypt doesn't take an 'encrypt' argument so we treat it as a sp
 * case
 */

static int idea_ecb_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                           const unsigned char *in, size_t inl)
{
        BLOCK_CIPHER_ecb_loop()
                idea_ecb_encrypt(in + i, out + i, ctx->cipher_data);
        return 1;
}

/* Can't use IMPLEMENT_BLOCK_CIPHER because idea_ecb_encrypt is different */

typedef struct
        {
        IDEA_KEY_SCHEDULE ks;
        } EVP_IDEA_KEY;

BLOCK_CIPHER_func_cbc(idea, idea, EVP_IDEA_KEY, ks)
BLOCK_CIPHER_func_ofb(idea, idea, 64, EVP_IDEA_KEY, ks)
BLOCK_CIPHER_func_cfb(idea, idea, 64, EVP_IDEA_KEY, ks)

BLOCK_CIPHER_defs(idea, IDEA_KEY_SCHEDULE, NID_idea, 8, 16, 8, 64,
                        0, idea_init_key, NULL,
                        EVP_CIPHER_set_asn1_iv, EVP_CIPHER_get_asn1_iv, NULL)

static int idea_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                         const unsigned char *iv, int enc)
        {
        if(!enc) {
                if (EVP_CIPHER_CTX_mode(ctx) == EVP_CIPH_OFB_MODE) enc = 1;
                else if (EVP_CIPHER_CTX_mode(ctx) == EVP_CIPH_CFB_MODE) enc = 1;
        }
        if (enc) idea_set_encrypt_key(key,ctx->cipher_data);
        else
                {
                IDEA_KEY_SCHEDULE tmp;

                idea_set_encrypt_key(key,&tmp);
                idea_set_decrypt_key(&tmp,ctx->cipher_data);
                OPENSSL_cleanse((unsigned char *)&tmp,
                                sizeof(IDEA_KEY_SCHEDULE));
                }
        return 1;
        }

#endif
#endif /* ! codereview */
```

```
**********************************************************
    4079 Wed Aug 13 19:52:43 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_null.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_null.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>

#ifndef OPENSSL_FIPS

static int null_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
        const unsigned char *iv,int enc);
static int null_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
        const unsigned char *in, size_t inl);
static const EVP_CIPHER n_cipher=
        {
        NID_undef,
        1,0,0,
        0,
        null_init_key,
        null_cipher,
        NULL,
        0,
        NULL,
        NULL,
        NULL,
        NULL
        };

const EVP_CIPHER *EVP_enc_null(void)
        {
        return(&n_cipher);
        }

static int null_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
             const unsigned char *iv, int enc)
        {
        /*      memset(&(ctx->c),0,sizeof(ctx->c));*/
        return 1;
        }

static int null_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
             const unsigned char *in, size_t inl)
        {
        if (in != out)
                memcpy((char *)out,(const char *)in,inl);
        return 1;
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4800 Wed Aug 13 19:52:44 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_old.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_old.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifdef OPENSSL_NO_DEPRECATED
static void *dummy = &dummy;
#else
```



```
#include <openssl/evp.h>

/* Define some deprecated functions, so older programs
   don't crash and burn too quickly.  On Windows and VMS,
   these will never be used, since functions and variables
   in shared libraries are selected by entry point location,
   not by name.  */

#ifndef OPENSSL_NO_BF
#undef EVP_bf_cfb
const EVP_CIPHER *EVP_bf_cfb(void);
const EVP_CIPHER *EVP_bf_cfb(void) { return EVP_bf_cfb64(); }
#endif

#ifndef OPENSSL_NO_DES
#undef EVP_des_cfb
const EVP_CIPHER *EVP_des_cfb(void);
const EVP_CIPHER *EVP_des_cfb(void) { return EVP_des_cfb64(); }
#undef EVP_des_ede3_cfb
const EVP_CIPHER *EVP_des_ede3_cfb(void);
const EVP_CIPHER *EVP_des_ede3_cfb(void) { return EVP_des_ede3_cfb64(); }
#undef EVP_des_ede_cfb
const EVP_CIPHER *EVP_des_ede_cfb(void);
const EVP_CIPHER *EVP_des_ede_cfb(void) { return EVP_des_ede_cfb64(); }
#endif

#ifndef OPENSSL_NO_IDEA
#undef EVP_idea_cfb
const EVP_CIPHER *EVP_idea_cfb(void);
const EVP_CIPHER *EVP_idea_cfb(void) { return EVP_idea_cfb64(); }
#endif

#ifndef OPENSSL_NO_RC2
#undef EVP_rc2_cfb
const EVP_CIPHER *EVP_rc2_cfb(void);
const EVP_CIPHER *EVP_rc2_cfb(void) { return EVP_rc2_cfb64(); }
#endif

#ifndef OPENSSL_NO_CAST
#undef EVP_cast5_cfb
const EVP_CIPHER *EVP_cast5_cfb(void);
const EVP_CIPHER *EVP_cast5_cfb(void) { return EVP_cast5_cfb64(); }
#endif

#ifndef OPENSSL_NO_RC5
#undef EVP_rc5_32_12_16_cfb
const EVP_CIPHER *EVP_rc5_32_12_16_cfb(void);
const EVP_CIPHER *EVP_rc5_32_12_16_cfb(void) { return EVP_rc5_32_12_16_cfb64();
#endif

#ifndef OPENSSL_NO_AES
#undef EVP_aes_128_cfb
const EVP_CIPHER *EVP_aes_128_cfb(void);
const EVP_CIPHER *EVP_aes_128_cfb(void) { return EVP_aes_128_cfb128(); }
#undef EVP_aes_192_cfb
const EVP_CIPHER *EVP_aes_192_cfb(void);
const EVP_CIPHER *EVP_aes_192_cfb(void) { return EVP_aes_192_cfb128(); }
#undef EVP_aes_256_cfb
const EVP_CIPHER *EVP_aes_256_cfb(void);
const EVP_CIPHER *EVP_aes_256_cfb(void) { return EVP_aes_256_cfb128(); }
#endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
    7107 Wed Aug 13 19:52:44 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_rc2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_rc2.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_RC2

#include <openssl/evp.h>
#include <openssl/objects.h>
#include "evp_locl.h"
#include <openssl/rc2.h>

static int rc2_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                        const unsigned char *iv,int enc);
static int rc2_meth_to_magic(EVP_CIPHER_CTX *ctx);
static int rc2_magic_to_meth(int i);
static int rc2_set_asn1_type_and_iv(EVP_CIPHER_CTX *c, ASN1_TYPE *type);
static int rc2_get_asn1_type_and_iv(EVP_CIPHER_CTX *c, ASN1_TYPE *type);
static int rc2_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr);

typedef struct
        {
        int key_bits;   /* effective key bits */
        RC2_KEY ks;     /* key schedule */
        } EVP_RC2_KEY;

#define data(ctx)       ((EVP_RC2_KEY *)(ctx)->cipher_data)

IMPLEMENT_BLOCK_CIPHER(rc2, ks, RC2, EVP_RC2_KEY, NID_rc2,
                       8,
                       RC2_KEY_LENGTH, 8, 64,
                       EVP_CIPH_VARIABLE_LENGTH | EVP_CIPH_CTRL_INIT,
                       rc2_init_key, NULL,
                       rc2_set_asn1_type_and_iv, rc2_get_asn1_type_and_iv,
                       rc2_ctrl)

#define RC2_40_MAGIC    0xa0
#define RC2_64_MAGIC    0x78
#define RC2_128_MAGIC   0x3a

static const EVP_CIPHER r2_64_cbc_cipher=
        {
        NID_rc2_64_cbc,
        8,8 /* 64 bit */,8,
        EVP_CIPH_CBC_MODE | EVP_CIPH_VARIABLE_LENGTH | EVP_CIPH_CTRL_INIT,
        rc2_init_key,
        rc2_cbc_cipher,
        NULL,
        sizeof(EVP_RC2_KEY),
        rc2_set_asn1_type_and_iv,
        rc2_get_asn1_type_and_iv,
        rc2_ctrl,
        NULL
        };

static const EVP_CIPHER r2_40_cbc_cipher=
        {
        NID_rc2_40_cbc,
        8,5 /* 40 bit */,8,
        EVP_CIPH_CBC_MODE | EVP_CIPH_VARIABLE_LENGTH | EVP_CIPH_CTRL_INIT,
        rc2_init_key,
        rc2_cbc_cipher,
        NULL,
        sizeof(EVP_RC2_KEY),
        rc2_set_asn1_type_and_iv,
        rc2_get_asn1_type_and_iv,
        rc2_ctrl,
        NULL
        };

const EVP_CIPHER *EVP_rc2_64_cbc(void)
```

```
128         {
129         return(&r2_64_cbc_cipher);
130         }

132 const EVP_CIPHER *EVP_rc2_40_cbc(void)
133         {
134         return(&r2_40_cbc_cipher);
135         }

137 static int rc2_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
138                         const unsigned char *iv, int enc)
139         {
140         RC2_set_key(&data(ctx)->ks,EVP_CIPHER_CTX_key_length(ctx),
141                     key,data(ctx)->key_bits);
142         return 1;
143         }

145 static int rc2_meth_to_magic(EVP_CIPHER_CTX *e)
146         {
147         int i;

149         EVP_CIPHER_CTX_ctrl(e, EVP_CTRL_GET_RC2_KEY_BITS, 0, &i);
150         if      (i == 128) return(RC2_128_MAGIC);
151         else if (i == 64)  return(RC2_64_MAGIC);
152         else if (i == 40)  return(RC2_40_MAGIC);
153         else return(0);
154         }

156 static int rc2_magic_to_meth(int i)
157         {
158         if      (i == RC2_128_MAGIC) return 128;
159         else if (i == RC2_64_MAGIC)  return 64;
160         else if (i == RC2_40_MAGIC)  return 40;
161         else
162                 {
163                 EVPerr(EVP_F_RC2_MAGIC_TO_METH,EVP_R_UNSUPPORTED_KEY_SIZE);
164                 return(0);
165                 }
166         }

168 static int rc2_get_asn1_type_and_iv(EVP_CIPHER_CTX *c, ASN1_TYPE *type)
169         {
170         long num=0;
171         int i=0;
172         int key_bits;
173         unsigned int l;
174         unsigned char iv[EVP_MAX_IV_LENGTH];

176         if (type != NULL)
177                 {
178                 l=EVP_CIPHER_CTX_iv_length(c);
179                 OPENSSL_assert(l <= sizeof(iv));
180                 i=ASN1_TYPE_get_int_octetstring(type,&num,iv,l);
181                 if (i != (int)l)
182                         return(-1);
183                 key_bits =rc2_magic_to_meth((int)num);
184                 if (!key_bits)
185                         return(-1);
186                 if(i > 0 && !EVP_CipherInit_ex(c, NULL, NULL, NULL, iv, -1))
187                         return -1;
188                 EVP_CIPHER_CTX_ctrl(c, EVP_CTRL_SET_RC2_KEY_BITS, key_bits, NULL
189                 EVP_CIPHER_CTX_set_key_length(c, key_bits / 8);
190                 }
191         return(i);
192         }
```



```
194 static int rc2_set_asn1_type_and_iv(EVP_CIPHER_CTX *c, ASN1_TYPE *type)
195         {
196         long num;
197         int i=0,j;

199         if (type != NULL)
200                 {
201                 num=rc2_meth_to_magic(c);
202                 j=EVP_CIPHER_CTX_iv_length(c);
203                 i=ASN1_TYPE_set_int_octetstring(type,num,c->oiv,j);
204                 }
205         return(i);
206         }

208 static int rc2_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr)
209         {
210         switch(type)
211                 {
212         case EVP_CTRL_INIT:
213                 data(c)->key_bits = EVP_CIPHER_CTX_key_length(c) * 8;
214                 return 1;

216         case EVP_CTRL_GET_RC2_KEY_BITS:
217                 *(int *)ptr = data(c)->key_bits;
218                 return 1;

220         case EVP_CTRL_SET_RC2_KEY_BITS:
221                 if(arg > 0)
222                         {
223                         data(c)->key_bits = arg;
224                         return 1;
225                         }
226                 return 0;
227 #ifdef PBE_PRF_TEST
228         case EVP_CTRL_PBE_PRF_NID:
229                 *(int *)ptr = NID_hmacWithMD5;
230                 return 1;
231 #endif

233         default:
234                 return -1;
235                 }
236         }

238 #endif
239 #endif /* ! codereview */
```

```
**********************************************************
    4650 Wed Aug 13 19:52:44 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_rc4.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_rc4.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_RC4

#include <openssl/evp.h>
#include "evp_locl.h"
#include <openssl/objects.h>
#include <openssl/rc4.h>

/* FIXME: surely this is available elsewhere? */
#define EVP_RC4_KEY_SIZE               16

typedef struct
    {
    RC4_KEY ks; /* working key */
    } EVP_RC4_KEY;

#define data(ctx) ((EVP_RC4_KEY *)(ctx)->cipher_data)

static int rc4_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                        const unsigned char *iv,int enc);
static int rc4_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                      const unsigned char *in, size_t inl);
static const EVP_CIPHER r4_cipher=
        {
        NID_rc4,
        1,EVP_RC4_KEY_SIZE,0,
        EVP_CIPH_VARIABLE_LENGTH,
        rc4_init_key,
        rc4_cipher,
        NULL,
        sizeof(EVP_RC4_KEY),
        NULL,
        NULL,
        NULL,
        NULL
        };

static const EVP_CIPHER r4_40_cipher=
        {
        NID_rc4_40,
        1,5 /* 40 bit */,0,
        EVP_CIPH_VARIABLE_LENGTH,
        rc4_init_key,
        rc4_cipher,
        NULL,
        sizeof(EVP_RC4_KEY),
        NULL,
        NULL,
        NULL,
        NULL
        };

const EVP_CIPHER *EVP_rc4(void)
        {
        return(&r4_cipher);
        }

const EVP_CIPHER *EVP_rc4_40(void)
        {
        return(&r4_40_cipher);
        }

static int rc4_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                        const unsigned char *iv, int enc)
        {
        RC4_set_key(&data(ctx)->ks,EVP_CIPHER_CTX_key_length(ctx),
                    key);
```

```
128         return 1;
129         }

131 static int rc4_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
132                        const unsigned char *in, size_t inl)
133         {
134         RC4(&data(ctx)->ks,inl,in,out);
135         return 1;
136         }
137 #endif
138 #endif /* ! codereview */
```

```
**********************************************************
    8367 Wed Aug 13 19:52:44 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_rc4_hmac_md5.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/opensslconf.h>

#include <stdio.h>
#include <string.h>

#if !defined(OPENSSL_NO_RC4) && !defined(OPENSSL_NO_MD5)

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/rc4.h>
#include <openssl/md5.h>
```



```
#ifndef EVP_CIPH_FLAG_AEAD_CIPHER
#define EVP_CIPH_FLAG_AEAD_CIPHER       0x200000
#define EVP_CTRL_AEAD_TLS1_AAD          0x16
#define EVP_CTRL_AEAD_SET_MAC_KEY       0x17
#endif

/* FIXME: surely this is available elsewhere? */
#define EVP_RC4_KEY_SIZE                16

typedef struct
    {
    RC4_KEY             ks;
    MD5_CTX             head,tail,md;
    size_t              payload_length;
    } EVP_RC4_HMAC_MD5;

#define NO_PAYLOAD_LENGTH       ((size_t)-1)

void rc4_md5_enc (RC4_KEY *key, const void *in0, void *out,
                MD5_CTX *ctx,const void *inp,size_t blocks);

#define data(ctx) ((EVP_RC4_HMAC_MD5 *)(ctx)->cipher_data)

static int rc4_hmac_md5_init_key(EVP_CIPHER_CTX *ctx,
                        const unsigned char *inkey,
                        const unsigned char *iv, int enc)
        {
        EVP_RC4_HMAC_MD5 *key = data(ctx);

        RC4_set_key(&key->ks,EVP_CIPHER_CTX_key_length(ctx),
                    inkey);

        MD5_Init(&key->head);   /* handy when benchmarking */
        key->tail = key->head;
        key->md   = key->head;

        key->payload_length = NO_PAYLOAD_LENGTH;

        return 1;
        }

#if     !defined(OPENSSL_NO_ASM) &&     ( \
        defined(__x86_64)       || defined(__x86_64__)  || \
        defined(_M_AMD64)       || defined(_M_X64)      || \
        defined(__INTEL__)                              ) && \
        !(defined(__APPLE__) && defined(__MACH__))
#define STITCHED_CALL
#endif

#if !defined(STITCHED_CALL)
#define rc4_off 0
#define md5_off 0
#endif

static int rc4_hmac_md5_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                        const unsigned char *in, size_t len)
        {
        EVP_RC4_HMAC_MD5 *key = data(ctx);
#if defined(STITCHED_CALL)
        size_t  rc4_off = 32-1-(key->ks.x&(32-1)),      /* 32 is $MOD from rc4_m
                md5_off = MD5_CBLOCK-key->md.num,
                blocks;
        unsigned int l;
        extern unsigned int OPENSSL_ia32cap_P[];
#endif
        size_t  plen = key->payload_length;
```

```
129         if (plen!=NO_PAYLOAD_LENGTH && len!=(plen+MD5_DIGEST_LENGTH)) return 0;

131         if (ctx->encrypt) {
132                 if (plen==NO_PAYLOAD_LENGTH) plen = len;
133 #if defined(STITCHED_CALL)
134                 /* cipher has to "fall behind" */
135                 if (rc4_off>md5_off) md5_off+=MD5_CBLOCK;

137                 if (plen>md5_off && (blocks=(plen-md5_off)/MD5_CBLOCK) &&
138                     (OPENSSL_ia32cap_P[0]&(1<<20))==0) {
139                         MD5_Update(&key->md,in,md5_off);
140                         RC4(&key->ks,rc4_off,in,out);

142                         rc4_md5_enc(&key->ks,in+rc4_off,out+rc4_off,
143                                 &key->md,in+md5_off,blocks);
144                         blocks *= MD5_CBLOCK;
145                         rc4_off += blocks;
146                         md5_off += blocks;
147                         key->md.Nh += blocks>>29;
148                         key->md.Nl += blocks<<=3;
149                         if (key->md.Nl<(unsigned int)blocks) key->md.Nh++;
150                 } else {
151                         rc4_off = 0;
152                         md5_off = 0;
153                 }
154 #endif
155                 MD5_Update(&key->md,in+md5_off,plen-md5_off);

157                 if (plen!=len) {        /* "TLS" mode of operation */
158                         if (in!=out)
159                                 memcpy(out+rc4_off,in+rc4_off,plen-rc4_off);

161                         /* calculate HMAC and append it to payload */
162                         MD5_Final(out+plen,&key->md);
163                         key->md = key->tail;
164                         MD5_Update(&key->md,out+plen,MD5_DIGEST_LENGTH);
165                         MD5_Final(out+plen,&key->md);
166                         /* encrypt HMAC at once */
167                         RC4(&key->ks,len-rc4_off,out+rc4_off,out+rc4_off);
168                 } else {
169                         RC4(&key->ks,len-rc4_off,in+rc4_off,out+rc4_off);
170                 }
171         } else {
172                 unsigned char mac[MD5_DIGEST_LENGTH];
173 #if defined(STITCHED_CALL)
174                 /* digest has to "fall behind" */
175                 if (md5_off>rc4_off)    rc4_off += 2*MD5_CBLOCK;
176                 else                    rc4_off += MD5_CBLOCK;

178                 if (len>rc4_off && (blocks=(len-rc4_off)/MD5_CBLOCK) &&
179                     (OPENSSL_ia32cap_P[0]&(1<<20))==0) {
180                         RC4(&key->ks,rc4_off,in,out);
181                         MD5_Update(&key->md,out,md5_off);

183                         rc4_md5_enc(&key->ks,in+rc4_off,out+rc4_off,
184                                 &key->md,out+md5_off,blocks);
185                         blocks *= MD5_CBLOCK;
186                         rc4_off += blocks;
187                         md5_off += blocks;
188                         l = (key->md.Nl+(blocks<<3))&0xffffffffU;
189                         if (l<key->md.Nl) key->md.Nh++;
190                         key->md.Nl  = l;
191                         key->md.Nh += blocks>>29;
192                 } else {
193                         md5_off=0;
```



```
194                         rc4_off=0;
195                 }
196 #endif
197                 /* decrypt HMAC at once */
198                 RC4(&key->ks,len-rc4_off,in+rc4_off,out+rc4_off);
199                 if (plen!=NO_PAYLOAD_LENGTH) {  /* "TLS" mode of operation */
200                         MD5_Update(&key->md,out+md5_off,plen-md5_off);

202                         /* calculate HMAC and verify it */
203                         MD5_Final(mac,&key->md);
204                         key->md = key->tail;
205                         MD5_Update(&key->md,mac,MD5_DIGEST_LENGTH);
206                         MD5_Final(mac,&key->md);

208                         if (memcmp(out+plen,mac,MD5_DIGEST_LENGTH))
209                                 return 0;
210                 } else {
211                         MD5_Update(&key->md,out+md5_off,len-md5_off);
212                 }
213         }

215         key->payload_length = NO_PAYLOAD_LENGTH;

217         return 1;
218         }

220 static int rc4_hmac_md5_ctrl(EVP_CIPHER_CTX *ctx, int type, int arg, void *ptr)
221         {
222         EVP_RC4_HMAC_MD5 *key = data(ctx);

224         switch (type)
225                 {
226         case EVP_CTRL_AEAD_SET_MAC_KEY:
227                 {
228                 unsigned int  i;
229                 unsigned char hmac_key[64];

231                 memset (hmac_key,0,sizeof(hmac_key));

233                 if (arg > (int)sizeof(hmac_key)) {
234                         MD5_Init(&key->head);
235                         MD5_Update(&key->head,ptr,arg);
236                         MD5_Final(hmac_key,&key->head);
237                 } else {
238                         memcpy(hmac_key,ptr,arg);
239                 }

241                 for (i=0;i<sizeof(hmac_key);i++)
242                         hmac_key[i] ^= 0x36;            /* ipad */
243                 MD5_Init(&key->head);
244                 MD5_Update(&key->head,hmac_key,sizeof(hmac_key));

246                 for (i=0;i<sizeof(hmac_key);i++)
247                         hmac_key[i] ^= 0x36^0x5c;       /* opad */
248                 MD5_Init(&key->tail);
249                 MD5_Update(&key->tail,hmac_key,sizeof(hmac_key));

251                 return 1;
252                 }
253         case EVP_CTRL_AEAD_TLS1_AAD:
254                 {
255                 unsigned char *p=ptr;
256                 unsigned int   len=p[arg-2]<<8|p[arg-1];

258                 if (!ctx->encrypt)
259                         {
```

```
260                         len -= MD5_DIGEST_LENGTH;
261                         p[arg-2] = len>>8;
262                         p[arg-1] = len;
263                         }
264                 key->payload_length=len;
265                 key->md = key->head;
266                 MD5_Update(&key->md,p,arg);

268                 return MD5_DIGEST_LENGTH;
269                 }
270         default:
271                 return -1;
272                 }
273         }

275 static EVP_CIPHER r4_hmac_md5_cipher=
276         {
277 #ifdef NID_rc4_hmac_md5
278         NID_rc4_hmac_md5,
279 #else
280         NID_undef,
281 #endif
282         1,EVP_RC4_KEY_SIZE,0,
283         EVP_CIPH_STREAM_CIPHER|EVP_CIPH_VARIABLE_LENGTH|EVP_CIPH_FLAG_AEAD_CIPHE
284         rc4_hmac_md5_init_key,
285         rc4_hmac_md5_cipher,
286         NULL,
287         sizeof(EVP_RC4_HMAC_MD5),
288         NULL,
289         NULL,
290         rc4_hmac_md5_ctrl,
291         NULL
292         };

294 const EVP_CIPHER *EVP_rc4_hmac_md5(void)
295         {
296         return(&r4_hmac_md5_cipher);
297         }
298 #endif
299 #endif /* ! codereview */
```

```
**********************************************************
    4693 Wed Aug 13 19:52:44 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_rc5.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_rc5.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_RC5

#include <openssl/evp.h>
#include <openssl/objects.h>
#include "evp_locl.h"
#include <openssl/rc5.h>

static int r_32_12_16_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                               const unsigned char *iv,int enc);
static int rc5_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr);

typedef struct
        {
        int rounds;     /* number of rounds */
        RC5_32_KEY ks;  /* key schedule */
        } EVP_RC5_KEY;

#define data(ctx)       EVP_C_DATA(EVP_RC5_KEY,ctx)

IMPLEMENT_BLOCK_CIPHER(rc5_32_12_16, ks, RC5_32, EVP_RC5_KEY, NID_rc5,
                       8, RC5_32_KEY_LENGTH, 8, 64,
                       EVP_CIPH_VARIABLE_LENGTH | EVP_CIPH_CTRL_INIT,
                       r_32_12_16_init_key, NULL,
                       NULL, NULL, rc5_ctrl)

static int rc5_ctrl(EVP_CIPHER_CTX *c, int type, int arg, void *ptr)
        {
        switch(type)
                {
        case EVP_CTRL_INIT:
                data(c)->rounds = RC5_12_ROUNDS;
                return 1;

        case EVP_CTRL_GET_RC5_ROUNDS:
                *(int *)ptr = data(c)->rounds;
                return 1;

        case EVP_CTRL_SET_RC5_ROUNDS:
                switch(arg)
                        {
                case RC5_8_ROUNDS:
                case RC5_12_ROUNDS:
                case RC5_16_ROUNDS:
                        data(c)->rounds = arg;
                        return 1;

                default:
                        EVPerr(EVP_F_RC5_CTRL, EVP_R_UNSUPORTED_NUMBER_OF_ROUNDS
                        return 0;
                        }

        default:
                return -1;
                }
        }

static int r_32_12_16_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                               const unsigned char *iv, int enc)
        {
        RC5_32_set_key(&data(ctx)->ks,EVP_CIPHER_CTX_key_length(ctx),
                       key,data(ctx)->rounds);
        return 1;
        }

#endif
#endif /* ! codereview */
```

```
**********************************************************
    3405 Wed Aug 13 19:52:44 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_seed.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_seed.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/opensslconf.h>
#ifndef OPENSSL_NO_SEED
#include <openssl/evp.h>
#include <openssl/err.h>
#include <string.h>
#include <assert.h>
```



```
#include <openssl/seed.h>
#include "evp_locl.h"

static int seed_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key, const un

typedef struct
        {
        SEED_KEY_SCHEDULE ks;
        } EVP_SEED_KEY;

IMPLEMENT_BLOCK_CIPHER(seed, ks, SEED, EVP_SEED_KEY, NID_seed,
                       16, 16, 16, 128,
                       0, seed_init_key, 0, 0, 0, 0)

static int seed_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                         const unsigned char *iv, int enc)
        {
        SEED_set_key(key, ctx->cipher_data);
        return 1;
        }

#endif
#endif /* ! codereview */
```

```
**********************************************************
    4913 Wed Aug 13 19:52:45 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/e_xcbc_d.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/e_xcbc_d.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_DES

#include <openssl/evp.h>
#include <openssl/objects.h>
#include "evp_locl.h"
#include <openssl/des.h>

static int desx_cbc_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                             const unsigned char *iv,int enc);
static int desx_cbc_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                           const unsigned char *in, size_t inl);


typedef struct
    {
    DES_key_schedule ks;/* key schedule */
    DES_cblock inw;
    DES_cblock outw;
    } DESX_CBC_KEY;

#define data(ctx) ((DESX_CBC_KEY *)(ctx)->cipher_data)

static const EVP_CIPHER d_xcbc_cipher=
        {
        NID_desx_cbc,
        8,24,8,
        EVP_CIPH_CBC_MODE,
        desx_cbc_init_key,
        desx_cbc_cipher,
        NULL,
        sizeof(DESX_CBC_KEY),
        EVP_CIPHER_set_asn1_iv,
        EVP_CIPHER_get_asn1_iv,
        NULL,
        NULL
        };

const EVP_CIPHER *EVP_desx_cbc(void)
        {
        return(&d_xcbc_cipher);
        }

static int desx_cbc_init_key(EVP_CIPHER_CTX *ctx, const unsigned char *key,
                             const unsigned char *iv, int enc)
        {
        DES_cblock *deskey = (DES_cblock *)key;

        DES_set_key_unchecked(deskey,&data(ctx)->ks);
        memcpy(&data(ctx)->inw[0],&key[8],8);
        memcpy(&data(ctx)->outw[0],&key[16],8);

        return 1;
        }

static int desx_cbc_cipher(EVP_CIPHER_CTX *ctx, unsigned char *out,
                           const unsigned char *in, size_t inl)
        {
        while (inl>=EVP_MAXCHUNK)
                {
                DES_xcbc_encrypt(in,out,(long)EVP_MAXCHUNK,&data(ctx)->ks,
                        (DES_cblock *)&(ctx->iv[0]),
                        &data(ctx)->inw,
                        &data(ctx)->outw,
                        ctx->encrypt);
                inl-=EVP_MAXCHUNK;
                in +=EVP_MAXCHUNK;
```

```
128                 out+=EVP_MAXCHUNK;
129                 }
130         if (inl)
131                 DES_xcbc_encrypt(in,out,(long)inl,&data(ctx)->ks,
132                         (DES_cblock *)&(ctx->iv[0]),
133                         &data(ctx)->inw,
134                         &data(ctx)->outw,
135                         ctx->encrypt);
136         return 1;
137         }
138 #endif
139 #endif /* ! codereview */
```

```
**********************************************************
   11610 Wed Aug 13 19:52:45 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/encode.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/encode.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#ifndef CHARSET_EBCDIC
#define conv_bin2ascii(a)       (data_bin2ascii[(a)&0x3f])
#define conv_ascii2bin(a)       (data_ascii2bin[(a)&0x7f])
#else
/* We assume that PEM encoded files are EBCDIC files
 * (i.e., printable text files). Convert them here while decoding.
 * When encoding, output is EBCDIC (text) format again.
 * (No need for conversion in the conv_bin2ascii macro, as the
 * underlying textstring data_bin2ascii[] is already EBCDIC)
 */
#define conv_bin2ascii(a)       (data_bin2ascii[(a)&0x3f])
#define conv_ascii2bin(a)       (data_ascii2bin[os_toascii[a]&0x7f])
#endif

/* 64 char lines
 * pad input with 0
 * left over chars are set to =
 * 1 byte  => xx==
 * 2 bytes => xxx=
 * 3 bytes => xxxx
 */
#define BIN_PER_LINE    (64/4*3)
#define CHUNKS_PER_LINE (64/4)
#define CHAR_PER_LINE   (64+1)

static const unsigned char data_bin2ascii[65]="ABCDEFGHIJKLMNOPQRSTUVWXYZ\
abcdefghijklmnopqrstuvwxyz0123456789+/";

/* 0xF0 is a EOLN
 * 0xF1 is ignore but next needs to be 0xF0 (for \r\n processing).
 * 0xF2 is EOF
 * 0xE0 is ignore at start of line.
 * 0xFF is error
 */

#define B64_EOLN                0xF0
#define B64_CR                  0xF1
#define B64_EOF                 0xF2
#define B64_WS                  0xE0
#define B64_ERROR               0xFF
#define B64_NOT_BASE64(a)       (((a)|0x13) == 0xF3)

static const unsigned char data_ascii2bin[128]={
        0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
        0xFF,0xE0,0xF0,0xFF,0xFF,0xF1,0xFF,0xFF,
        0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
        0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
        0xE0,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
        0xFF,0xFF,0xFF,0x3E,0xFF,0xF2,0xFF,0x3F,
        0x34,0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,
        0x3C,0x3D,0xFF,0xFF,0xFF,0x00,0xFF,0xFF,
        0xFF,0x00,0x01,0x02,0x03,0x04,0x05,0x06,
        0x07,0x08,0x09,0x0A,0x0B,0x0C,0x0D,0x0E,
        0x0F,0x10,0x11,0x12,0x13,0x14,0x15,0x16,
        0x17,0x18,0x19,0xFF,0xFF,0xFF,0xFF,0xFF,
        0xFF,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,0x20,
        0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,
        0x29,0x2A,0x2B,0x2C,0x2D,0x2E,0x2F,0x30,
        0x31,0x32,0x33,0xFF,0xFF,0xFF,0xFF,0xFF,
        };

void EVP_EncodeInit(EVP_ENCODE_CTX *ctx)
        {
        ctx->length=48;
        ctx->num=0;
```

```
128         ctx->line_num=0;
129         }

131 void EVP_EncodeUpdate(EVP_ENCODE_CTX *ctx, unsigned char *out, int *outl,
132              const unsigned char *in, int inl)
133         {
134         int i,j;
135         unsigned int total=0;

137         *outl=0;
138         if (inl == 0) return;
139         OPENSSL_assert(ctx->length <= (int)sizeof(ctx->enc_data));
140         if ((ctx->num+inl) < ctx->length)
141                 {
142                 memcpy(&(ctx->enc_data[ctx->num]),in,inl);
143                 ctx->num+=inl;
144                 return;
145                 }
146         if (ctx->num != 0)
147                 {
148                 i=ctx->length-ctx->num;
149                 memcpy(&(ctx->enc_data[ctx->num]),in,i);
150                 in+=i;
151                 inl-=i;
152                 j=EVP_EncodeBlock(out,ctx->enc_data,ctx->length);
153                 ctx->num=0;
154                 out+=j;
155                 *(out++)='\n';
156                 *out='\0';
157                 total=j+1;
158                 }
159         while (inl >= ctx->length)
160                 {
161                 j=EVP_EncodeBlock(out,in,ctx->length);
162                 in+=ctx->length;
163                 inl-=ctx->length;
164                 out+=j;
165                 *(out++)='\n';
166                 *out='\0';
167                 total+=j+1;
168                 }
169         if (inl != 0)
170                 memcpy(&(ctx->enc_data[0]),in,inl);
171         ctx->num=inl;
172         *outl=total;
173         }

175 void EVP_EncodeFinal(EVP_ENCODE_CTX *ctx, unsigned char *out, int *outl)
176         {
177         unsigned int ret=0;

179         if (ctx->num != 0)
180                 {
181                 ret=EVP_EncodeBlock(out,ctx->enc_data,ctx->num);
182                 out[ret++]='\n';
183                 out[ret]='\0';
184                 ctx->num=0;
185                 }
186         *outl=ret;
187         }

189 int EVP_EncodeBlock(unsigned char *t, const unsigned char *f, int dlen)
190         {
191         int i,ret=0;
192         unsigned long l;
```



```
194         for (i=dlen; i > 0; i-=3)
195                 {
196                 if (i >= 3)
197                         {
198                         l=      (((unsigned long)f[0])<<16L)|
199                                 (((unsigned long)f[1])<< 8L)|f[2];
200                         *(t++)=conv_bin2ascii(l>>18L);
201                         *(t++)=conv_bin2ascii(l>>12L);
202                         *(t++)=conv_bin2ascii(l>> 6L);
203                         *(t++)=conv_bin2ascii(l     );
204                         }
205                 else
206                         {
207                         l=((unsigned long)f[0])<<16L;
208                         if (i == 2) l|=((unsigned long)f[1]<<8L);

210                         *(t++)=conv_bin2ascii(l>>18L);
211                         *(t++)=conv_bin2ascii(l>>12L);
212                         *(t++)=(i == 1)?'=':conv_bin2ascii(l>> 6L);
213                         *(t++)='=';
214                         }
215                 ret+=4;
216                 f+=3;
217                 }

219         *t='\0';
220         return(ret);
221         }

223 void EVP_DecodeInit(EVP_ENCODE_CTX *ctx)
224         {
225         ctx->length=30;
226         ctx->num=0;
227         ctx->line_num=0;
228         ctx->expect_nl=0;
229         }

231 /* -1 for error
232  *  0 for last line
233  *  1 for full line
234  */
235 int EVP_DecodeUpdate(EVP_ENCODE_CTX *ctx, unsigned char *out, int *outl,
236              const unsigned char *in, int inl)
237         {
238         int seof= -1,eof=0,rv= -1,ret=0,i,v,tmp,n,ln,exp_nl;
239         unsigned char *d;

241         n=ctx->num;
242         d=ctx->enc_data;
243         ln=ctx->line_num;
244         exp_nl=ctx->expect_nl;

246         /* last line of input. */
247         if ((inl == 0) || ((n == 0) && (conv_ascii2bin(in[0]) == B64_EOF)))
248                 { rv=0; goto end; }

250         /* We parse the input data */
251         for (i=0; i<inl; i++)
252                 {
253                 /* If the current line is > 80 characters, scream alot */
254                 if (ln >= 80) { rv= -1; goto end; }

256                 /* Get char and put it into the buffer */
257                 tmp= *(in++);
258                 v=conv_ascii2bin(tmp);
259                 /* only save the good data :-) */
```

```
260                 if (!B64_NOT_BASE64(v))
261                         {
262                         OPENSSL_assert(n < (int)sizeof(ctx->enc_data));
263                         d[n++]=tmp;
264                         ln++;
265                         }
266                 else if (v == B64_ERROR)
267                         {
268                         rv= -1;
269                         goto end;
270                         }
271
272                 /* have we seen a '=' which is 'definitly' the last
273                  * input line.  seof will point to the character that
274                  * holds it. and eof will hold how many characters to
275                  * chop off. */
276                 if (tmp == '=')
277                         {
278                         if (seof == -1) seof=n;
279                         eof++;
280                         }
281
282                 if (v == B64_CR)
283                         {
284                         ln = 0;
285                         if (exp_nl)
286                                 continue;
287                         }
288
289                 /* eoln */
290                 if (v == B64_EOLN)
291                         {
292                         ln=0;
293                         if (exp_nl)
294                                 {
295                                 exp_nl=0;
296                                 continue;
297                                 }
298                         }
299                 exp_nl=0;
300
301                 /* If we are at the end of input and it looks like a
302                  * line, process it. */
303                 if (((i+1) == inl) && (((n&3) == 0) || eof))
304                         {
305                         v=B64_EOF;
306                         /* In case things were given us in really small
307                            records (so two '=' were given in separate
308                            updates), eof may contain the incorrect number
309                            of ending bytes to skip, so let's redo the count */
310                         eof = 0;
311                         if (d[n-1] == '=') eof++;
312                         if (d[n-2] == '=') eof++;
313                         /* There will never be more than two '=' */
314                         }
315
316                 if ((v == B64_EOF && (n&3) == 0) || (n >= 64))
317                         {
318                         /* This is needed to work correctly on 64 byte input
319                          * lines.  We process the line and then need to
320                          * accept the '\n' */
321                         if ((v != B64_EOF) && (n >= 64)) exp_nl=1;
322                         if (n > 0)
323                                 {
324                                 v=EVP_DecodeBlock(out,d,n);
325                                 n=0;
```



```
326                                 if (v < 0) { rv=0; goto end; }
327                                 if (eof > v) { rv=-1; goto end; }
328                                 ret+=(v-eof);
329                                 }
330                         else
331                                 {
332                                 eof=1;
333                                 v=0;
334                                 }
335
336                         /* This is the case where we have had a short
337                          * but valid input line */
338                         if ((v < ctx->length) && eof)
339                                 {
340                                 rv=0;
341                                 goto end;
342                                 }
343                         else
344                                 ctx->length=v;
345
346                         if (seof >= 0) { rv=0; goto end; }
347                         out+=v;
348                         }
349                 }
350         rv=1;
351 end:
352         *outl=ret;
353         ctx->num=n;
354         ctx->line_num=ln;
355         ctx->expect_nl=exp_nl;
356         return(rv);
357         }
358
359 int EVP_DecodeBlock(unsigned char *t, const unsigned char *f, int n)
360         {
361         int i,ret=0,a,b,c,d;
362         unsigned long l;
363
364         /* trim white space from the start of the line. */
365         while ((conv_ascii2bin(*f) == B64_WS) && (n > 0))
366                 {
367                 f++;
368                 n--;
369                 }
370
371         /* strip off stuff at the end of the line
372          * ascii2bin values B64_WS, B64_EOLN, B64_EOLN and B64_EOF */
373         while ((n > 3) && (B64_NOT_BASE64(conv_ascii2bin(f[n-1]))))
374                 n--;
375
376         if (n%4 != 0) return(-1);
377
378         for (i=0; i<n; i+=4)
379                 {
380                 a=conv_ascii2bin(*(f++));
381                 b=conv_ascii2bin(*(f++));
382                 c=conv_ascii2bin(*(f++));
383                 d=conv_ascii2bin(*(f++));
384                 if (    (a & 0x80) || (b & 0x80) ||
385                         (c & 0x80) || (d & 0x80))
386                         return(-1);
387                 l=(     (((unsigned long)a)<<18L)|
388                         (((unsigned long)b)<<12L)|
389                         (((unsigned long)c)<< 6L)|
390                         (((unsigned long)d)     ));
391                 *(t++)=(unsigned char)(l>>16L)&0xff;
```

```
392                 *(t++)=(unsigned char)(l>> 8L)&0xff;
393                 *(t++)=(unsigned char)(l     )&0xff;
394                 ret+=3;
395                 }
396         return(ret);
397         }

399 int EVP_DecodeFinal(EVP_ENCODE_CTX *ctx, unsigned char *out, int *outl)
400         {
401         int i;

403         *outl=0;
404         if (ctx->num != 0)
405                 {
406                 i=EVP_DecodeBlock(out,ctx->enc_data,ctx->num);
407                 if (i < 0) return(-1);
408                 ctx->num=0;
409                 *outl=i;
410                 return(1);
411                 }
412         else
413                 return(1);
414         }

416 #ifdef undef
417 int EVP_DecodeValid(unsigned char *buf, int len)
418         {
419         int i,num=0,bad=0;

421         if (len == 0) return(-1);
422         while (conv_ascii2bin(*buf) == B64_WS)
423                 {
424                 buf++;
425                 len--;
426                 if (len == 0) return(-1);
427                 }

429         for (i=len; i >= 4; i-=4)
430                 {
431                 if (    (conv_ascii2bin(buf[0]) >= 0x40) ||
432                         (conv_ascii2bin(buf[1]) >= 0x40) ||
433                         (conv_ascii2bin(buf[2]) >= 0x40) ||
434                         (conv_ascii2bin(buf[3]) >= 0x40))
435                         return(-1);
436                 buf+=4;
437                 num+=1+(buf[2] != '=')+(buf[3] != '=');
438                 }
439         if ((i == 1) && (conv_ascii2bin(buf[0]) == B64_EOLN))
440                 return(num);
441         if ((i == 2) && (conv_ascii2bin(buf[0]) == B64_EOLN) &&
442                 (conv_ascii2bin(buf[0]) == B64_EOLN))
443                 return(num);
444         return(1);
445         }
446 #endif
447 #endif /* ! codereview */
```

```
**********************************************************
    3040 Wed Aug 13 19:52:45 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_acnf.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* evp_acnf.c */
/* Written by Stephen Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/evp.h>
#include <openssl/conf.h>
```



```
/* Load all algorithms and configure OpenSSL.
 * This function is called automatically when
 * OPENSSL_LOAD_CONF is set.
 */

void OPENSSL_add_all_algorithms_conf(void)
        {
        OPENSSL_add_all_algorithms_noconf();
        OPENSSL_config(NULL);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4156 Wed Aug 13 19:52:45 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_cnf.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* evp_cnf.c */
/* Written by Stephen Henson (steve@openssl.org) for the OpenSSL
 * project 2007.
 */
/* ====================================================================
 * Copyright (c) 2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <ctype.h>
#include <openssl/crypto.h>
```



```
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/dso.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif


/* Algorithm configuration module. */

static int alg_module_init(CONF_IMODULE *md, const CONF *cnf)
        {
        int i;
        const char *oid_section;
        STACK_OF(CONF_VALUE) *sktmp;
        CONF_VALUE *oval;
        oid_section = CONF_imodule_get_value(md);
        if(!(sktmp = NCONF_get_section(cnf, oid_section)))
                {
                EVPerr(EVP_F_ALG_MODULE_INIT, EVP_R_ERROR_LOADING_SECTION);
                return 0;
                }
        for(i = 0; i < sk_CONF_VALUE_num(sktmp); i++)
                {
                oval = sk_CONF_VALUE_value(sktmp, i);
                if (!strcmp(oval->name, "fips_mode"))
                        {
                        int m;
                        if (!X509V3_get_value_bool(oval, &m))
                                {
                                EVPerr(EVP_F_ALG_MODULE_INIT, EVP_R_INVALID_FIPS
                                return 0;
                                }
                        if (m > 0)
                                {
#ifdef OPENSSL_FIPS
                                if (!FIPS_mode() && !FIPS_mode_set(1))
                                        {
                                        EVPerr(EVP_F_ALG_MODULE_INIT, EVP_R_ERRO
                                        return 0;
                                        }
#else
                                EVPerr(EVP_F_ALG_MODULE_INIT, EVP_R_FIPS_MODE_NO
                                return 0;
#endif
                                }
                        }
                else
                        {
                        EVPerr(EVP_F_ALG_MODULE_INIT, EVP_R_UNKNOWN_OPTION);
                        ERR_add_error_data(4, "name=", oval->name,
                                                ", value=", oval->value);
                        }

                }
        return 1;
        }

void EVP_add_alg_module(void)
        {
        CONF_module_add("alg_section", alg_module_init, 0);
        }
#endif /* ! codereview */
```

```
**********************************************************
   16730 Wed Aug 13 19:52:45 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/evp_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/err.h>
#include <openssl/rand.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif
#include "evp_locl.h"

#ifdef OPENSSL_FIPS
#define M_do_cipher(ctx, out, in, inl) FIPS_cipher(ctx, out, in, inl)
#else
#define M_do_cipher(ctx, out, in, inl) ctx->cipher->do_cipher(ctx, out, in, inl)
#endif


const char EVP_version[]="EVP" OPENSSL_VERSION_PTEXT;

void EVP_CIPHER_CTX_init(EVP_CIPHER_CTX *ctx)
        {
        memset(ctx,0,sizeof(EVP_CIPHER_CTX));
        /* ctx->cipher=NULL; */
        }

EVP_CIPHER_CTX *EVP_CIPHER_CTX_new(void)
        {
        EVP_CIPHER_CTX *ctx=OPENSSL_malloc(sizeof *ctx);
        if (ctx)
                EVP_CIPHER_CTX_init(ctx);
        return ctx;
        }

int EVP_CipherInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher,
             const unsigned char *key, const unsigned char *iv, int enc)
        {
        if (cipher)
                EVP_CIPHER_CTX_init(ctx);
        return EVP_CipherInit_ex(ctx,cipher,NULL,key,iv,enc);
        }

int EVP_CipherInit_ex(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *imp
             const unsigned char *key, const unsigned char *iv, int enc)
        {
        if (enc == -1)
                enc = ctx->encrypt;
        else
                {
                if (enc)
                        enc = 1;
                ctx->encrypt = enc;
                }
#ifndef OPENSSL_NO_ENGINE
        /* Whether it's nice or not, "Inits" can be used on "Final"'d contexts
         * so this context may already have an ENGINE! Try to avoid releasing
         * the previous handle, re-querying for an ENGINE, and having a
         * reinitialisation, when it may all be unecessary. */
        if (ctx->engine && ctx->cipher && (!cipher ||
                        (cipher && (cipher->nid == ctx->cipher->nid))))
                goto skip_to_init;
#endif
        if (cipher)
                {
                /* Ensure a context left lying around from last time is cleared
                 * (the previous check attempted to avoid this if the same
                 * ENGINE and EVP_CIPHER could be used). */
```

```
128                 if (ctx->cipher)
129                         {
130                         unsigned long flags = ctx->flags;
131                         EVP_CIPHER_CTX_cleanup(ctx);
132                         /* Restore encrypt and flags */
133                         ctx->encrypt = enc;
134                         ctx->flags = flags;
135                         }
136 #ifndef OPENSSL_NO_ENGINE
137                 if(impl)
138                         {
139                         if (!ENGINE_init(impl))
140                                 {
141                                 EVPerr(EVP_F_EVP_CIPHERINIT_EX, EVP_R_INITIALIZA
142                                 return 0;
143                                 }
144                         }
145                 else
146                         /* Ask if an ENGINE is reserved for this job */
147                         impl = ENGINE_get_cipher_engine(cipher->nid);
148                 if(impl)
149                         {
150                         /* There's an ENGINE for this job ... (apparently) */
151                         const EVP_CIPHER *c = ENGINE_get_cipher(impl, cipher->ni
152                         if(!c)
153                                 {
154                                 /* One positive side-effect of US's export
155                                  * control history, is that we should at least
156                                  * be able to avoid using US mispellings of
157                                  * "initialisation"? */
158                                 EVPerr(EVP_F_EVP_CIPHERINIT_EX, EVP_R_INITIALIZA
159                                 return 0;
160                                 }
161                         /* We'll use the ENGINE's private cipher definition */
162                         cipher = c;
163                         /* Store the ENGINE functional reference so we know
164                          * 'cipher' came from an ENGINE and we need to release
165                          * it when done. */
166                         ctx->engine = impl;
167                         }
168                 else
169                         ctx->engine = NULL;
170 #endif
171
172 #ifdef OPENSSL_FIPS
173                 if (FIPS_mode())
174                         return FIPS_cipherinit(ctx, cipher, key, iv, enc);
175 #endif
176                 ctx->cipher=cipher;
177                 if (ctx->cipher->ctx_size)
178                         {
179                         ctx->cipher_data=OPENSSL_malloc(ctx->cipher->ctx_size);
180                         if (!ctx->cipher_data)
181                                 {
182                                 EVPerr(EVP_F_EVP_CIPHERINIT_EX, ERR_R_MALLOC_FAI
183                                 return 0;
184                                 }
185                         }
186                 else
187                         {
188                         ctx->cipher_data = NULL;
189                         }
190                 ctx->key_len = cipher->key_len;
191                 ctx->flags = 0;
192                 if(ctx->cipher->flags & EVP_CIPH_CTRL_INIT)
193                         {
```



```
194                         if(!EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_INIT, 0, NULL))
195                                 {
196                                 EVPerr(EVP_F_EVP_CIPHERINIT_EX, EVP_R_INITIALIZA
197                                 return 0;
198                                 }
199                         }
200                 }
201         else if(!ctx->cipher)
202                 {
203                 EVPerr(EVP_F_EVP_CIPHERINIT_EX, EVP_R_NO_CIPHER_SET);
204                 return 0;
205                 }
206 #ifndef OPENSSL_NO_ENGINE
207 skip_to_init:
208 #endif
209 #ifdef OPENSSL_FIPS
210         if (FIPS_mode())
211                 return FIPS_cipherinit(ctx, cipher, key, iv, enc);
212 #endif
213         /* we assume block size is a power of 2 in *cryptUpdate */
214         OPENSSL_assert(ctx->cipher->block_size == 1
215             || ctx->cipher->block_size == 8
216             || ctx->cipher->block_size == 16);
217
218         if(!(EVP_CIPHER_CTX_flags(ctx) & EVP_CIPH_CUSTOM_IV)) {
219                 switch(EVP_CIPHER_CTX_mode(ctx)) {
220
221                         case EVP_CIPH_STREAM_CIPHER:
222                         case EVP_CIPH_ECB_MODE:
223                         break;
224
225                         case EVP_CIPH_CFB_MODE:
226                         case EVP_CIPH_OFB_MODE:
227
228                         ctx->num = 0;
229                         /* fall-through */
230
231                         case EVP_CIPH_CBC_MODE:
232
233                         OPENSSL_assert(EVP_CIPHER_CTX_iv_length(ctx) <=
234                                        (int)sizeof(ctx->iv));
235                         if(iv) memcpy(ctx->oiv, iv, EVP_CIPHER_CTX_iv_length(ctx
236                         memcpy(ctx->iv, ctx->oiv, EVP_CIPHER_CTX_iv_length(ctx))
237                         break;
238
239                         case EVP_CIPH_CTR_MODE:
240                         ctx->num = 0;
241                         /* Don't reuse IV for CTR mode */
242                         if(iv)
243                                 memcpy(ctx->iv, iv, EVP_CIPHER_CTX_iv_length(ctx
244                         break;
245
246                         default:
247                         return 0;
248                         break;
249                 }
250         }
251
252         if(key || (ctx->cipher->flags & EVP_CIPH_ALWAYS_CALL_INIT)) {
253                 if(!ctx->cipher->init(ctx,key,iv,enc)) return 0;
254         }
255         ctx->buf_len=0;
256         ctx->final_used=0;
257         ctx->block_mask=ctx->cipher->block_size-1;
258         return 1;
259         }
```

```
int EVP_CipherUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl,
             const unsigned char *in, int inl)
        {
        if (ctx->encrypt)
                return EVP_EncryptUpdate(ctx,out,outl,in,inl);
        else    return EVP_DecryptUpdate(ctx,out,outl,in,inl);
        }

int EVP_CipherFinal_ex(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
        {
        if (ctx->encrypt)
                return EVP_EncryptFinal_ex(ctx,out,outl);
        else    return EVP_DecryptFinal_ex(ctx,out,outl);
        }

int EVP_CipherFinal(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
        {
        if (ctx->encrypt)
                return EVP_EncryptFinal(ctx,out,outl);
        else    return EVP_DecryptFinal(ctx,out,outl);
        }

int EVP_EncryptInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher,
             const unsigned char *key, const unsigned char *iv)
        {
        return EVP_CipherInit(ctx, cipher, key, iv, 1);
        }

int EVP_EncryptInit_ex(EVP_CIPHER_CTX *ctx,const EVP_CIPHER *cipher, ENGINE *imp
                const unsigned char *key, const unsigned char *iv)
        {
        return EVP_CipherInit_ex(ctx, cipher, impl, key, iv, 1);
        }

int EVP_DecryptInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher,
             const unsigned char *key, const unsigned char *iv)
        {
        return EVP_CipherInit(ctx, cipher, key, iv, 0);
        }

int EVP_DecryptInit_ex(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *cipher, ENGINE *im
             const unsigned char *key, const unsigned char *iv)
        {
        return EVP_CipherInit_ex(ctx, cipher, impl, key, iv, 0);
        }

int EVP_EncryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl,
             const unsigned char *in, int inl)
        {
        int i,j,bl;

        if (ctx->cipher->flags & EVP_CIPH_FLAG_CUSTOM_CIPHER)
                {
                i = M_do_cipher(ctx, out, in, inl);
                if (i < 0)
                        return 0;
                else
                        *outl = i;
                return 1;
                }

        if (inl <= 0)
                {
                *outl = 0;
                return inl == 0;
```



```
                }

        if(ctx->buf_len == 0 && (inl&(ctx->block_mask)) == 0)
                {
                if(M_do_cipher(ctx,out,in,inl))
                        {
                        *outl=inl;
                        return 1;
                        }
                else
                        {
                        *outl=0;
                        return 0;
                        }
                }
        i=ctx->buf_len;
        bl=ctx->cipher->block_size;
        OPENSSL_assert(bl <= (int)sizeof(ctx->buf));
        if (i != 0)
                {
                if (i+inl < bl)
                        {
                        memcpy(&(ctx->buf[i]),in,inl);
                        ctx->buf_len+=inl;
                        *outl=0;
                        return 1;
                        }
                else
                        {
                        j=bl-i;
                        memcpy(&(ctx->buf[i]),in,j);
                        if(!M_do_cipher(ctx,out,ctx->buf,bl)) return 0;
                        inl-=j;
                        in+=j;
                        out+=bl;
                        *outl=bl;
                        }
                }
        else
                *outl = 0;
        i=inl&(bl-1);
        inl-=i;
        if (inl > 0)
                {
                if(!M_do_cipher(ctx,out,in,inl)) return 0;
                *outl+=inl;
                }

        if (i != 0)
                memcpy(ctx->buf,&(in[inl]),i);
        ctx->buf_len=i;
        return 1;
        }

int EVP_EncryptFinal(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
        {
        int ret;
        ret = EVP_EncryptFinal_ex(ctx, out, outl);
        return ret;
        }

int EVP_EncryptFinal_ex(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
        {
        int n,ret;
        unsigned int i, b, bl;
```

```
392         if (ctx->cipher->flags & EVP_CIPH_FLAG_CUSTOM_CIPHER)
393                 {
394                 ret = M_do_cipher(ctx, out, NULL, 0);
395                 if (ret < 0)
396                         return 0;
397                 else
398                         *outl = ret;
399                 return 1;
400                 }

402         b=ctx->cipher->block_size;
403         OPENSSL_assert(b <= sizeof ctx->buf);
404         if (b == 1)
405                 {
406                 *outl=0;
407                 return 1;
408                 }
409         bl=ctx->buf_len;
410         if (ctx->flags & EVP_CIPH_NO_PADDING)
411                 {
412                 if(bl)
413                         {
414                         EVPerr(EVP_F_EVP_ENCRYPTFINAL_EX,EVP_R_DATA_NOT_MULTIPLE
415                         return 0;
416                         }
417                 *outl = 0;
418                 return 1;
419                 }

421         n=b-bl;
422         for (i=bl; i<b; i++)
423                 ctx->buf[i]=n;
424         ret=M_do_cipher(ctx,out,ctx->buf,b);


427         if(ret)
428                 *outl=b;

430         return ret;
431         }

433 int EVP_DecryptUpdate(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl,
434                       const unsigned char *in, int inl)
435         {
436         int fix_len;
437         unsigned int b;

439         if (ctx->cipher->flags & EVP_CIPH_FLAG_CUSTOM_CIPHER)
440                 {
441                 fix_len = M_do_cipher(ctx, out, in, inl);
442                 if (fix_len < 0)
443                         {
444                         *outl = 0;
445                         return 0;
446                         }
447                 else
448                         *outl = fix_len;
449                 return 1;
450                 }

452         if (inl <= 0)
453                 {
454                 *outl = 0;
455                 return inl == 0;
456                 }
```



```
458         if (ctx->flags & EVP_CIPH_NO_PADDING)
459                 return EVP_EncryptUpdate(ctx, out, outl, in, inl);

461         b=ctx->cipher->block_size;
462         OPENSSL_assert(b <= sizeof ctx->final);

464         if(ctx->final_used)
465                 {
466                 memcpy(out,ctx->final,b);
467                 out+=b;
468                 fix_len = 1;
469                 }
470         else
471                 fix_len = 0;


474         if(!EVP_EncryptUpdate(ctx,out,outl,in,inl))
475                 return 0;

477         /* if we have 'decrypted' a multiple of block size, make sure
478          * we have a copy of this last block */
479         if (b > 1 && !ctx->buf_len)
480                 {
481                 *outl-=b;
482                 ctx->final_used=1;
483                 memcpy(ctx->final,&out[*outl],b);
484                 }
485         else
486                 ctx->final_used = 0;

488         if (fix_len)
489                 *outl += b;

491         return 1;
492         }

494 int EVP_DecryptFinal(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
495         {
496         int ret;
497         ret = EVP_DecryptFinal_ex(ctx, out, outl);
498         return ret;
499         }

501 int EVP_DecryptFinal_ex(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
502         {
503         int i,n;
504         unsigned int b;
505         *outl=0;

507         if (ctx->cipher->flags & EVP_CIPH_FLAG_CUSTOM_CIPHER)
508                 {
509                 i = M_do_cipher(ctx, out, NULL, 0);
510                 if (i < 0)
511                         return 0;
512                 else
513                         *outl = i;
514                 return 1;
515                 }

517         b=ctx->cipher->block_size;
518         if (ctx->flags & EVP_CIPH_NO_PADDING)
519                 {
520                 if(ctx->buf_len)
521                         {
522                         EVPerr(EVP_F_EVP_DECRYPTFINAL_EX,EVP_R_DATA_NOT_MULTIPLE
523                         return 0;
```

```
524                     }
525                 *outl = 0;
526                 return 1;
527                 }
528         if (b > 1)
529                 {
530                 if (ctx->buf_len || !ctx->final_used)
531                         {
532                         EVPerr(EVP_F_EVP_DECRYPTFINAL_EX,EVP_R_WRONG_FINAL_BLOCK
533                         return(0);
534                         }
535                 OPENSSL_assert(b <= sizeof ctx->final);
536                 n=ctx->final[b-1];
537                 if (n == 0 || n > (int)b)
538                         {
539                         EVPerr(EVP_F_EVP_DECRYPTFINAL_EX,EVP_R_BAD_DECRYPT);
540                         return(0);
541                         }
542                 for (i=0; i<n; i++)
543                         {
544                         if (ctx->final[--b] != n)
545                                 {
546                                 EVPerr(EVP_F_EVP_DECRYPTFINAL_EX,EVP_R_BAD_DECRY
547                                 return(0);
548                                 }
549                         }
550                 n=ctx->cipher->block_size-n;
551                 for (i=0; i<n; i++)
552                         out[i]=ctx->final[i];
553                 *outl=n;
554                 }
555         else
556                 *outl=0;
557         return(1);
558         }

560 void EVP_CIPHER_CTX_free(EVP_CIPHER_CTX *ctx)
561         {
562         if (ctx)
563                 {
564                 EVP_CIPHER_CTX_cleanup(ctx);
565                 OPENSSL_free(ctx);
566                 }
567         }

569 int EVP_CIPHER_CTX_cleanup(EVP_CIPHER_CTX *c)
570         {
571 #ifndef OPENSSL_FIPS
572         if (c->cipher != NULL)
573                 {
574                 if(c->cipher->cleanup && !c->cipher->cleanup(c))
575                         return 0;
576                 /* Cleanse cipher context data */
577                 if (c->cipher_data)
578                         OPENSSL_cleanse(c->cipher_data, c->cipher->ctx_size);
579                 }
580         if (c->cipher_data)
581                 OPENSSL_free(c->cipher_data);
582 #endif
583 #ifndef OPENSSL_NO_ENGINE
584         if (c->engine)
585                 /* The EVP_CIPHER we used belongs to an ENGINE, release the
586                  * functional reference we held for this reason. */
587                 ENGINE_finish(c->engine);
588 #endif
589 #ifdef OPENSSL_FIPS
```



```
590         FIPS_cipher_ctx_cleanup(c);
591 #endif
592         memset(c,0,sizeof(EVP_CIPHER_CTX));
593         return 1;
594         }

596 int EVP_CIPHER_CTX_set_key_length(EVP_CIPHER_CTX *c, int keylen)
597         {
598         if(c->cipher->flags & EVP_CIPH_CUSTOM_KEY_LENGTH)
599                 return EVP_CIPHER_CTX_ctrl(c, EVP_CTRL_SET_KEY_LENGTH, keylen, N
600         if(c->key_len == keylen) return 1;
601         if((keylen > 0) && (c->cipher->flags & EVP_CIPH_VARIABLE_LENGTH))
602                 {
603                 c->key_len = keylen;
604                 return 1;
605                 }
606         EVPerr(EVP_F_EVP_CIPHER_CTX_SET_KEY_LENGTH,EVP_R_INVALID_KEY_LENGTH);
607         return 0;
608         }

610 int EVP_CIPHER_CTX_set_padding(EVP_CIPHER_CTX *ctx, int pad)
611         {
612         if (pad) ctx->flags &= ~EVP_CIPH_NO_PADDING;
613         else ctx->flags |= EVP_CIPH_NO_PADDING;
614         return 1;
615         }

617 int EVP_CIPHER_CTX_ctrl(EVP_CIPHER_CTX *ctx, int type, int arg, void *ptr)
618 {
619         int ret;
620         if(!ctx->cipher) {
621                 EVPerr(EVP_F_EVP_CIPHER_CTX_CTRL, EVP_R_NO_CIPHER_SET);
622                 return 0;
623         }

625         if(!ctx->cipher->ctrl) {
626                 EVPerr(EVP_F_EVP_CIPHER_CTX_CTRL, EVP_R_CTRL_NOT_IMPLEMENTED);
627                 return 0;
628         }

630         ret = ctx->cipher->ctrl(ctx, type, arg, ptr);
631         if(ret == -1) {
632                 EVPerr(EVP_F_EVP_CIPHER_CTX_CTRL, EVP_R_CTRL_OPERATION_NOT_IMPLE
633                 return 0;
634         }
635         return ret;
636 }

638 int EVP_CIPHER_CTX_rand_key(EVP_CIPHER_CTX *ctx, unsigned char *key)
639         {
640         if (ctx->cipher->flags & EVP_CIPH_RAND_KEY)
641                 return EVP_CIPHER_CTX_ctrl(ctx, EVP_CTRL_RAND_KEY, 0, key);
642         if (RAND_bytes(key, ctx->key_len) <= 0)
643                 return 0;
644         return 1;
645         }

647 int EVP_CIPHER_CTX_copy(EVP_CIPHER_CTX *out, const EVP_CIPHER_CTX *in)
648         {
649         if ((in == NULL) || (in->cipher == NULL))
650                 {
651                 EVPerr(EVP_F_EVP_CIPHER_CTX_COPY,EVP_R_INPUT_NOT_INITIALIZED);
652                 return 0;
653                 }
654 #ifndef OPENSSL_NO_ENGINE
655         /* Make sure it's safe to copy a cipher context using an ENGINE */
```

```
656         if (in->engine && !ENGINE_init(in->engine))
657                 {
658                 EVPerr(EVP_F_EVP_CIPHER_CTX_COPY,ERR_R_ENGINE_LIB);
659                 return 0;
660                 }
661 #endif

663         EVP_CIPHER_CTX_cleanup(out);
664         memcpy(out,in,sizeof *out);

666         if (in->cipher_data && in->cipher->ctx_size)
667                 {
668                 out->cipher_data=OPENSSL_malloc(in->cipher->ctx_size);
669                 if (!out->cipher_data)
670                         {
671                         EVPerr(EVP_F_EVP_CIPHER_CTX_COPY,ERR_R_MALLOC_FAILURE);
672                         return 0;
673                         }
674                 memcpy(out->cipher_data,in->cipher_data,in->cipher->ctx_size);
675                 }

677         if (in->cipher->flags & EVP_CIPH_CUSTOM_COPY)
678                 return in->cipher->ctrl((EVP_CIPHER_CTX *)in, EVP_CTRL_COPY, 0,
679         return 1;
680         }
681 #endif /* ! codereview */
```

```
**********************************************************
   12655 Wed Aug 13 19:52:45 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/evp_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/evp.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_EVP,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_EVP,0,reason)

static ERR_STRING_DATA EVP_str_functs[]=
        {
{ERR_FUNC(EVP_F_AESNI_INIT_KEY),        "AESNI_INIT_KEY"},
{ERR_FUNC(EVP_F_AESNI_XTS_CIPHER),      "AESNI_XTS_CIPHER"},
{ERR_FUNC(EVP_F_AES_INIT_KEY),  "AES_INIT_KEY"},
{ERR_FUNC(EVP_F_AES_XTS),       "AES_XTS"},
{ERR_FUNC(EVP_F_AES_XTS_CIPHER),        "AES_XTS_CIPHER"},
{ERR_FUNC(EVP_F_ALG_MODULE_INIT),       "ALG_MODULE_INIT"},
{ERR_FUNC(EVP_F_CAMELLIA_INIT_KEY),     "CAMELLIA_INIT_KEY"},
{ERR_FUNC(EVP_F_CMAC_INIT),     "CMAC_INIT"},
{ERR_FUNC(EVP_F_D2I_PKEY),      "D2I_PKEY"},
{ERR_FUNC(EVP_F_DO_SIGVER_INIT),        "DO_SIGVER_INIT"},
{ERR_FUNC(EVP_F_DSAPKEY2PKCS8), "DSAPKEY2PKCS8"},
{ERR_FUNC(EVP_F_DSA_PKEY2PKCS8),        "DSA_PKEY2PKCS8"},
{ERR_FUNC(EVP_F_ECDSA_PKEY2PKCS8),      "ECDSA_PKEY2PKCS8"},
{ERR_FUNC(EVP_F_ECKEY_PKEY2PKCS8),      "ECKEY_PKEY2PKCS8"},
{ERR_FUNC(EVP_F_EVP_CIPHERINIT_EX),     "EVP_CipherInit_ex"},
{ERR_FUNC(EVP_F_EVP_CIPHER_CTX_COPY),   "EVP_CIPHER_CTX_copy"},
{ERR_FUNC(EVP_F_EVP_CIPHER_CTX_CTRL),   "EVP_CIPHER_CTX_ctrl"},
{ERR_FUNC(EVP_F_EVP_CIPHER_CTX_SET_KEY_LENGTH), "EVP_CIPHER_CTX_set_key_length"}
{ERR_FUNC(EVP_F_EVP_DECRYPTFINAL_EX),   "EVP_DecryptFinal_ex"},
{ERR_FUNC(EVP_F_EVP_DIGESTINIT_EX),     "EVP_DigestInit_ex"},
{ERR_FUNC(EVP_F_EVP_ENCRYPTFINAL_EX),   "EVP_EncryptFinal_ex"},
{ERR_FUNC(EVP_F_EVP_MD_CTX_COPY_EX),    "EVP_MD_CTX_copy_ex"},
{ERR_FUNC(EVP_F_EVP_MD_SIZE),   "EVP_MD_size"},
{ERR_FUNC(EVP_F_EVP_OPENINIT),  "EVP_OpenInit"},
{ERR_FUNC(EVP_F_EVP_PBE_ALG_ADD),       "EVP_PBE_alg_add"},
{ERR_FUNC(EVP_F_EVP_PBE_ALG_ADD_TYPE),  "EVP_PBE_alg_add_type"},
{ERR_FUNC(EVP_F_EVP_PBE_CIPHERINIT),    "EVP_PBE_CipherInit"},
{ERR_FUNC(EVP_F_EVP_PKCS82PKEY),        "EVP_PKCS82PKEY"},
{ERR_FUNC(EVP_F_EVP_PKCS82PKEY_BROKEN), "EVP_PKCS82PKEY_BROKEN"},
{ERR_FUNC(EVP_F_EVP_PKEY2PKCS8_BROKEN), "EVP_PKEY2PKCS8_broken"},
{ERR_FUNC(EVP_F_EVP_PKEY_COPY_PARAMETERS),      "EVP_PKEY_copy_parameters"},
{ERR_FUNC(EVP_F_EVP_PKEY_CTX_CTRL),     "EVP_PKEY_CTX_ctrl"},
{ERR_FUNC(EVP_F_EVP_PKEY_CTX_CTRL_STR), "EVP_PKEY_CTX_ctrl_str"},
{ERR_FUNC(EVP_F_EVP_PKEY_CTX_DUP),      "EVP_PKEY_CTX_dup"},
{ERR_FUNC(EVP_F_EVP_PKEY_DECRYPT),      "EVP_PKEY_decrypt"},
{ERR_FUNC(EVP_F_EVP_PKEY_DECRYPT_INIT), "EVP_PKEY_decrypt_init"},
{ERR_FUNC(EVP_F_EVP_PKEY_DECRYPT_OLD),  "EVP_PKEY_decrypt_old"},
{ERR_FUNC(EVP_F_EVP_PKEY_DERIVE),       "EVP_PKEY_derive"},
{ERR_FUNC(EVP_F_EVP_PKEY_DERIVE_INIT),  "EVP_PKEY_derive_init"},
{ERR_FUNC(EVP_F_EVP_PKEY_DERIVE_SET_PEER),      "EVP_PKEY_derive_set_peer"},
{ERR_FUNC(EVP_F_EVP_PKEY_ENCRYPT),      "EVP_PKEY_encrypt"},
{ERR_FUNC(EVP_F_EVP_PKEY_ENCRYPT_INIT), "EVP_PKEY_encrypt_init"},
{ERR_FUNC(EVP_F_EVP_PKEY_ENCRYPT_OLD),  "EVP_PKEY_encrypt_old"},
{ERR_FUNC(EVP_F_EVP_PKEY_GET1_DH),      "EVP_PKEY_get1_DH"},
{ERR_FUNC(EVP_F_EVP_PKEY_GET1_DSA),     "EVP_PKEY_get1_DSA"},
{ERR_FUNC(EVP_F_EVP_PKEY_GET1_ECDSA),   "EVP_PKEY_GET1_ECDSA"},
{ERR_FUNC(EVP_F_EVP_PKEY_GET1_EC_KEY),  "EVP_PKEY_get1_EC_KEY"},
{ERR_FUNC(EVP_F_EVP_PKEY_GET1_RSA),     "EVP_PKEY_get1_RSA"},
{ERR_FUNC(EVP_F_EVP_PKEY_KEYGEN),       "EVP_PKEY_keygen"},
{ERR_FUNC(EVP_F_EVP_PKEY_KEYGEN_INIT),  "EVP_PKEY_keygen_init"},
{ERR_FUNC(EVP_F_EVP_PKEY_NEW),  "EVP_PKEY_new"},
{ERR_FUNC(EVP_F_EVP_PKEY_PARAMGEN),     "EVP_PKEY_paramgen"},
{ERR_FUNC(EVP_F_EVP_PKEY_PARAMGEN_INIT),        "EVP_PKEY_paramgen_init"},
{ERR_FUNC(EVP_F_EVP_PKEY_SIGN), "EVP_PKEY_sign"},
{ERR_FUNC(EVP_F_EVP_PKEY_SIGN_INIT),    "EVP_PKEY_sign_init"},
```

```
{ERR_FUNC(EVP_F_EVP_PKEY_VERIFY),       "EVP_PKEY_verify"},
{ERR_FUNC(EVP_F_EVP_PKEY_VERIFY_INIT),  "EVP_PKEY_verify_init"},
{ERR_FUNC(EVP_F_EVP_PKEY_VERIFY_RECOVER),       "EVP_PKEY_verify_recover"},
{ERR_FUNC(EVP_F_EVP_PKEY_VERIFY_RECOVER_INIT),  "EVP_PKEY_verify_recover_init"},
{ERR_FUNC(EVP_F_EVP_RIJNDAEL),  "EVP_RIJNDAEL"},
{ERR_FUNC(EVP_F_EVP_SIGNFINAL), "EVP_SignFinal"},
{ERR_FUNC(EVP_F_EVP_VERIFYFINAL),       "EVP_VerifyFinal"},
{ERR_FUNC(EVP_F_FIPS_CIPHERINIT),       "FIPS_CIPHERINIT"},
{ERR_FUNC(EVP_F_FIPS_CIPHER_CTX_COPY),  "FIPS_CIPHER_CTX_COPY"},
{ERR_FUNC(EVP_F_FIPS_CIPHER_CTX_CTRL),  "FIPS_CIPHER_CTX_CTRL"},
{ERR_FUNC(EVP_F_FIPS_CIPHER_CTX_SET_KEY_LENGTH),        "FIPS_CIPHER_CTX_SET_KEY
{ERR_FUNC(EVP_F_FIPS_DIGESTINIT),       "FIPS_DIGESTINIT"},
{ERR_FUNC(EVP_F_FIPS_MD_CTX_COPY),      "FIPS_MD_CTX_COPY"},
{ERR_FUNC(EVP_F_HMAC_INIT_EX),  "HMAC_Init_ex"},
{ERR_FUNC(EVP_F_INT_CTX_NEW),   "INT_CTX_NEW"},
{ERR_FUNC(EVP_F_PKCS5_PBE_KEYIVGEN),    "PKCS5_PBE_keyivgen"},
{ERR_FUNC(EVP_F_PKCS5_V2_PBE_KEYIVGEN), "PKCS5_v2_PBE_keyivgen"},
{ERR_FUNC(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN),      "PKCS5_V2_PBKDF2_KEYIVGEN"},
{ERR_FUNC(EVP_F_PKCS8_SET_BROKEN),      "PKCS8_set_broken"},
{ERR_FUNC(EVP_F_PKEY_SET_TYPE), "PKEY_SET_TYPE"},
{ERR_FUNC(EVP_F_RC2_MAGIC_TO_METH),     "RC2_MAGIC_TO_METH"},
{ERR_FUNC(EVP_F_RC5_CTRL),      "RC5_CTRL"},
{0,NULL}
        };

static ERR_STRING_DATA EVP_str_reasons[]=
        {
{ERR_REASON(EVP_R_AES_IV_SETUP_FAILED)   ,"aes iv setup failed"},
{ERR_REASON(EVP_R_AES_KEY_SETUP_FAILED)  ,"aes key setup failed"},
{ERR_REASON(EVP_R_ASN1_LIB)              ,"asn1 lib"},
{ERR_REASON(EVP_R_BAD_BLOCK_LENGTH)      ,"bad block length"},
{ERR_REASON(EVP_R_BAD_DECRYPT)           ,"bad decrypt"},
{ERR_REASON(EVP_R_BAD_KEY_LENGTH)        ,"bad key length"},
{ERR_REASON(EVP_R_BN_DECODE_ERROR)       ,"bn decode error"},
{ERR_REASON(EVP_R_BN_PUBKEY_ERROR)       ,"bn pubkey error"},
{ERR_REASON(EVP_R_BUFFER_TOO_SMALL)      ,"buffer too small"},
{ERR_REASON(EVP_R_CAMELLIA_KEY_SETUP_FAILED),"camellia key setup failed"},
{ERR_REASON(EVP_R_CIPHER_PARAMETER_ERROR),"cipher parameter error"},
{ERR_REASON(EVP_R_COMMAND_NOT_SUPPORTED) ,"command not supported"},
{ERR_REASON(EVP_R_CTRL_NOT_IMPLEMENTED)  ,"ctrl not implemented"},
{ERR_REASON(EVP_R_CTRL_OPERATION_NOT_IMPLEMENTED),"ctrl operation not implemente
{ERR_REASON(EVP_R_DATA_NOT_MULTIPLE_OF_BLOCK_LENGTH),"data not multiple of block
{ERR_REASON(EVP_R_DECODE_ERROR)          ,"decode error"},
{ERR_REASON(EVP_R_DIFFERENT_KEY_TYPES)   ,"different key types"},
{ERR_REASON(EVP_R_DIFFERENT_PARAMETERS)  ,"different parameters"},
{ERR_REASON(EVP_R_DISABLED_FOR_FIPS)     ,"disabled for fips"},
{ERR_REASON(EVP_R_ENCODE_ERROR)          ,"encode error"},
{ERR_REASON(EVP_R_ERROR_LOADING_SECTION) ,"error loading section"},
{ERR_REASON(EVP_R_ERROR_SETTING_FIPS_MODE),"error setting fips mode"},
{ERR_REASON(EVP_R_EVP_PBE_CIPHERINIT_ERROR),"evp pbe cipherinit error"},
{ERR_REASON(EVP_R_EXPECTING_AN_RSA_KEY)  ,"expecting an rsa key"},
{ERR_REASON(EVP_R_EXPECTING_A_DH_KEY)    ,"expecting a dh key"},
{ERR_REASON(EVP_R_EXPECTING_A_DSA_KEY)   ,"expecting a dsa key"},
{ERR_REASON(EVP_R_EXPECTING_A_ECDSA_KEY) ,"expecting a ecdsa key"},
{ERR_REASON(EVP_R_EXPECTING_A_EC_KEY)    ,"expecting a ec key"},
{ERR_REASON(EVP_R_FIPS_MODE_NOT_SUPPORTED),"fips mode not supported"},
{ERR_REASON(EVP_R_INITIALIZATION_ERROR)  ,"initialization error"},
{ERR_REASON(EVP_R_INPUT_NOT_INITIALIZED) ,"input not initialized"},
{ERR_REASON(EVP_R_INVALID_DIGEST)        ,"invalid digest"},
{ERR_REASON(EVP_R_INVALID_FIPS_MODE)     ,"invalid fips mode"},
{ERR_REASON(EVP_R_INVALID_KEY_LENGTH)    ,"invalid key length"},
{ERR_REASON(EVP_R_INVALID_OPERATION)     ,"invalid operation"},
{ERR_REASON(EVP_R_IV_TOO_LARGE)          ,"iv too large"},
{ERR_REASON(EVP_R_KEYGEN_FAILURE)        ,"keygen failure"},
{ERR_REASON(EVP_R_MESSAGE_DIGEST_IS_NULL),"message digest is null"},
{ERR_REASON(EVP_R_METHOD_NOT_SUPPORTED)  ,"method not supported"},
```



```
{ERR_REASON(EVP_R_MISSING_PARAMETERS)    ,"missing parameters"},
{ERR_REASON(EVP_R_NO_CIPHER_SET)         ,"no cipher set"},
{ERR_REASON(EVP_R_NO_DEFAULT_DIGEST)     ,"no default digest"},
{ERR_REASON(EVP_R_NO_DIGEST_SET)         ,"no digest set"},
{ERR_REASON(EVP_R_NO_DSA_PARAMETERS)     ,"no dsa parameters"},
{ERR_REASON(EVP_R_NO_KEY_SET)            ,"no key set"},
{ERR_REASON(EVP_R_NO_OPERATION_SET)      ,"no operation set"},
{ERR_REASON(EVP_R_NO_SIGN_FUNCTION_CONFIGURED),"no sign function configured"},
{ERR_REASON(EVP_R_NO_VERIFY_FUNCTION_CONFIGURED),"no verify function configured"
{ERR_REASON(EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE),"operation not suppo
{ERR_REASON(EVP_R_OPERATON_NOT_INITIALIZED),"operaton not initialized"},
{ERR_REASON(EVP_R_PKCS8_UNKNOWN_BROKEN_TYPE),"pkcs8 unknown broken type"},
{ERR_REASON(EVP_R_PRIVATE_KEY_DECODE_ERROR),"private key decode error"},
{ERR_REASON(EVP_R_PRIVATE_KEY_ENCODE_ERROR),"private key encode error"},
{ERR_REASON(EVP_R_PUBLIC_KEY_NOT_RSA)    ,"public key not rsa"},
{ERR_REASON(EVP_R_TOO_LARGE)             ,"too large"},
{ERR_REASON(EVP_R_UNKNOWN_CIPHER)        ,"unknown cipher"},
{ERR_REASON(EVP_R_UNKNOWN_DIGEST)        ,"unknown digest"},
{ERR_REASON(EVP_R_UNKNOWN_OPTION)        ,"unknown option"},
{ERR_REASON(EVP_R_UNKNOWN_PBE_ALGORITHM) ,"unknown pbe algorithm"},
{ERR_REASON(EVP_R_UNSUPORTED_NUMBER_OF_ROUNDS),"unsuported number of rounds"},
{ERR_REASON(EVP_R_UNSUPPORTED_ALGORITHM) ,"unsupported algorithm"},
{ERR_REASON(EVP_R_UNSUPPORTED_CIPHER)    ,"unsupported cipher"},
{ERR_REASON(EVP_R_UNSUPPORTED_KEYLENGTH) ,"unsupported keylength"},
{ERR_REASON(EVP_R_UNSUPPORTED_KEY_DERIVATION_FUNCTION),"unsupported key derivati
{ERR_REASON(EVP_R_UNSUPPORTED_KEY_SIZE)  ,"unsupported key size"},
{ERR_REASON(EVP_R_UNSUPPORTED_PRF)       ,"unsupported prf"},
{ERR_REASON(EVP_R_UNSUPPORTED_PRIVATE_KEY_ALGORITHM),"unsupported private key al
{ERR_REASON(EVP_R_UNSUPPORTED_SALT_TYPE) ,"unsupported salt type"},
{ERR_REASON(EVP_R_WRONG_FINAL_BLOCK_LENGTH),"wrong final block length"},
{ERR_REASON(EVP_R_WRONG_PUBLIC_KEY_TYPE) ,"wrong public key type"},
{0,NULL}
        };

#endif

void ERR_load_EVP_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(EVP_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,EVP_str_functs);
                ERR_load_strings(0,EVP_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    6352 Wed Aug 13 19:52:46 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_fips.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/evp_fips.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */


#include <openssl/evp.h>

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>

const EVP_CIPHER *EVP_aes_128_cbc(void)  { return FIPS_evp_aes_128_cbc(); }
const EVP_CIPHER *EVP_aes_128_ccm(void)  { return FIPS_evp_aes_128_ccm(); }
```



```
const EVP_CIPHER *EVP_aes_128_cfb1(void)  { return FIPS_evp_aes_128_cfb1(); }
const EVP_CIPHER *EVP_aes_128_cfb128(void)  { return FIPS_evp_aes_128_cfb128();
const EVP_CIPHER *EVP_aes_128_cfb8(void)  { return FIPS_evp_aes_128_cfb8(); }
const EVP_CIPHER *EVP_aes_128_ctr(void)  { return FIPS_evp_aes_128_ctr(); }
const EVP_CIPHER *EVP_aes_128_ecb(void)  { return FIPS_evp_aes_128_ecb(); }
const EVP_CIPHER *EVP_aes_128_gcm(void)  { return FIPS_evp_aes_128_gcm(); }
const EVP_CIPHER *EVP_aes_128_ofb(void)  { return FIPS_evp_aes_128_ofb(); }
const EVP_CIPHER *EVP_aes_128_xts(void)  { return FIPS_evp_aes_128_xts(); }
const EVP_CIPHER *EVP_aes_192_cbc(void)  { return FIPS_evp_aes_192_cbc(); }
const EVP_CIPHER *EVP_aes_192_ccm(void)  { return FIPS_evp_aes_192_ccm(); }
const EVP_CIPHER *EVP_aes_192_cfb1(void)  { return FIPS_evp_aes_192_cfb1(); }
const EVP_CIPHER *EVP_aes_192_cfb128(void)  { return FIPS_evp_aes_192_cfb128();
const EVP_CIPHER *EVP_aes_192_cfb8(void)  { return FIPS_evp_aes_192_cfb8(); }
const EVP_CIPHER *EVP_aes_192_ctr(void)  { return FIPS_evp_aes_192_ctr(); }
const EVP_CIPHER *EVP_aes_192_ecb(void)  { return FIPS_evp_aes_192_ecb(); }
const EVP_CIPHER *EVP_aes_192_gcm(void)  { return FIPS_evp_aes_192_gcm(); }
const EVP_CIPHER *EVP_aes_192_ofb(void)  { return FIPS_evp_aes_192_ofb(); }
const EVP_CIPHER *EVP_aes_256_cbc(void)  { return FIPS_evp_aes_256_cbc(); }
const EVP_CIPHER *EVP_aes_256_ccm(void)  { return FIPS_evp_aes_256_ccm(); }
const EVP_CIPHER *EVP_aes_256_cfb1(void)  { return FIPS_evp_aes_256_cfb1(); }
const EVP_CIPHER *EVP_aes_256_cfb128(void)  { return FIPS_evp_aes_256_cfb128();
const EVP_CIPHER *EVP_aes_256_cfb8(void)  { return FIPS_evp_aes_256_cfb8(); }
const EVP_CIPHER *EVP_aes_256_ctr(void)  { return FIPS_evp_aes_256_ctr(); }
const EVP_CIPHER *EVP_aes_256_ecb(void)  { return FIPS_evp_aes_256_ecb(); }
const EVP_CIPHER *EVP_aes_256_gcm(void)  { return FIPS_evp_aes_256_gcm(); }
const EVP_CIPHER *EVP_aes_256_ofb(void)  { return FIPS_evp_aes_256_ofb(); }
const EVP_CIPHER *EVP_aes_256_xts(void)  { return FIPS_evp_aes_256_xts(); }
const EVP_CIPHER *EVP_des_ede(void)  { return FIPS_evp_des_ede(); }
const EVP_CIPHER *EVP_des_ede3(void)  { return FIPS_evp_des_ede3(); }
const EVP_CIPHER *EVP_des_ede3_cbc(void)  { return FIPS_evp_des_ede3_cbc(); }
const EVP_CIPHER *EVP_des_ede3_cfb1(void)  { return FIPS_evp_des_ede3_cfb1(); }
const EVP_CIPHER *EVP_des_ede3_cfb64(void)  { return FIPS_evp_des_ede3_cfb64();
const EVP_CIPHER *EVP_des_ede3_cfb8(void)  { return FIPS_evp_des_ede3_cfb8(); }
const EVP_CIPHER *EVP_des_ede3_ecb(void)  { return FIPS_evp_des_ede3_ecb(); }
const EVP_CIPHER *EVP_des_ede3_ofb(void)  { return FIPS_evp_des_ede3_ofb(); }
const EVP_CIPHER *EVP_des_ede_cbc(void)  { return FIPS_evp_des_ede_cbc(); }
const EVP_CIPHER *EVP_des_ede_cfb64(void)  { return FIPS_evp_des_ede_cfb64(); }
const EVP_CIPHER *EVP_des_ede_ecb(void)  { return FIPS_evp_des_ede_ecb(); }
const EVP_CIPHER *EVP_des_ede_ofb(void)  { return FIPS_evp_des_ede_ofb(); }
const EVP_CIPHER *EVP_enc_null(void)  { return FIPS_evp_enc_null(); }

const EVP_MD *EVP_sha1(void)  { return FIPS_evp_sha1(); }
const EVP_MD *EVP_sha224(void)  { return FIPS_evp_sha224(); }
const EVP_MD *EVP_sha256(void)  { return FIPS_evp_sha256(); }
const EVP_MD *EVP_sha384(void)  { return FIPS_evp_sha384(); }
const EVP_MD *EVP_sha512(void)  { return FIPS_evp_sha512(); }

const EVP_MD *EVP_dss(void)  { return FIPS_evp_dss(); }
const EVP_MD *EVP_dss1(void)  { return FIPS_evp_dss1(); }
const EVP_MD *EVP_ecdsa(void)  { return FIPS_evp_ecdsa(); }

#endif
#endif /* ! codereview */
```

```
**********************************************************
    6034 Wed Aug 13 19:52:46 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_key.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/evp_key.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/ui.h>

/* should be init to zeros. */
static char prompt_string[80];

void EVP_set_pw_prompt(const char *prompt)
        {
        if (prompt == NULL)
                prompt_string[0]='\0';
        else
                {
                strncpy(prompt_string,prompt,79);
                prompt_string[79]='\0';
                }
        }

char *EVP_get_pw_prompt(void)
        {
        if (prompt_string[0] == '\0')
                return(NULL);
        else
                return(prompt_string);
        }

/* For historical reasons, the standard function for reading passwords is
 * in the DES library -- if someone ever wants to disable DES,
 * this function will fail */
int EVP_read_pw_string(char *buf, int len, const char *prompt, int verify)
        {
        return EVP_read_pw_string_min(buf, 0, len, prompt, verify);
        }

int EVP_read_pw_string_min(char *buf, int min, int len, const char *prompt, int
        {
        int ret;
        char buff[BUFSIZ];
        UI *ui;

        if ((prompt == NULL) && (prompt_string[0] != '\0'))
                prompt=prompt_string;
        ui = UI_new();
        UI_add_input_string(ui,prompt,0,buf,min,(len>=BUFSIZ)?BUFSIZ-1:len);
        if (verify)
                UI_add_verify_string(ui,prompt,0,
                        buff,min,(len>=BUFSIZ)?BUFSIZ-1:len,buf);
        ret = UI_process(ui);
        UI_free(ui);
        OPENSSL_cleanse(buff,BUFSIZ);
        return ret;
        }

int EVP_BytesToKey(const EVP_CIPHER *type, const EVP_MD *md,
             const unsigned char *salt, const unsigned char *data, int datal,
             int count, unsigned char *key, unsigned char *iv)
        {
        EVP_MD_CTX c;
        unsigned char md_buf[EVP_MAX_MD_SIZE];
        int niv,nkey,addmd=0;
        unsigned int mds=0,i;
        int rv = 0;
        nkey=type->key_len;
        niv=type->iv_len;
        OPENSSL_assert(nkey <= EVP_MAX_KEY_LENGTH);
        OPENSSL_assert(niv <= EVP_MAX_IV_LENGTH);
```

```
129         if (data == NULL) return(nkey);

131         EVP_MD_CTX_init(&c);
132         for (;;)
133                 {
134                 if (!EVP_DigestInit_ex(&c,md, NULL))
135                         return 0;
136                 if (addmd++)
137                         if (!EVP_DigestUpdate(&c,&(md_buf[0]),mds))
138                                 goto err;
139                 if (!EVP_DigestUpdate(&c,data,datal))
140                         goto err;
141                 if (salt != NULL)
142                         if (!EVP_DigestUpdate(&c,salt,PKCS5_SALT_LEN))
143                                 goto err;
144                 if (!EVP_DigestFinal_ex(&c,&(md_buf[0]),&mds))
145                         goto err;

147                 for (i=1; i<(unsigned int)count; i++)
148                         {
149                         if (!EVP_DigestInit_ex(&c,md, NULL))
150                                 goto err;
151                         if (!EVP_DigestUpdate(&c,&(md_buf[0]),mds))
152                                 goto err;
153                         if (!EVP_DigestFinal_ex(&c,&(md_buf[0]),&mds))
154                                 goto err;
155                         }
156                 i=0;
157                 if (nkey)
158                         {
159                         for (;;)
160                                 {
161                                 if (nkey == 0) break;
162                                 if (i == mds) break;
163                                 if (key != NULL)
164                                         *(key++)=md_buf[i];
165                                 nkey--;
166                                 i++;
167                                 }
168                         }
169                 if (niv && (i != mds))
170                         {
171                         for (;;)
172                                 {
173                                 if (niv == 0) break;
174                                 if (i == mds) break;
175                                 if (iv != NULL)
176                                         *(iv++)=md_buf[i];
177                                 niv--;
178                                 i++;
179                                 }
180                         }
181                 if ((nkey == 0) && (niv == 0)) break;
182                 }
183         rv = type->key_len;
184         err:
185         EVP_MD_CTX_cleanup(&c);
186         OPENSSL_cleanse(&(md_buf[0]),EVP_MAX_MD_SIZE);
187         return rv;
188         }
189 #endif /* ! codereview */
```

```
**********************************************************
    7876 Wed Aug 13 19:52:46 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/evp_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>

int EVP_CIPHER_param_to_asn1(EVP_CIPHER_CTX *c, ASN1_TYPE *type)
        {
        int ret;

        if (c->cipher->set_asn1_parameters != NULL)
                ret=c->cipher->set_asn1_parameters(c,type);
        else if (c->cipher->flags & EVP_CIPH_FLAG_DEFAULT_ASN1)
                ret=EVP_CIPHER_set_asn1_iv(c, type);
        else
                ret=-1;
        return(ret);
        }

int EVP_CIPHER_asn1_to_param(EVP_CIPHER_CTX *c, ASN1_TYPE *type)
        {
        int ret;

        if (c->cipher->get_asn1_parameters != NULL)
                ret=c->cipher->get_asn1_parameters(c,type);
        else if (c->cipher->flags & EVP_CIPH_FLAG_DEFAULT_ASN1)
                ret=EVP_CIPHER_get_asn1_iv(c, type);
        else
                ret=-1;
        return(ret);
        }

int EVP_CIPHER_get_asn1_iv(EVP_CIPHER_CTX *c, ASN1_TYPE *type)
        {
        int i=0;
        unsigned int l;

        if (type != NULL)
                {
                l=EVP_CIPHER_CTX_iv_length(c);
                OPENSSL_assert(l <= sizeof(c->iv));
                i=ASN1_TYPE_get_octetstring(type,c->oiv,l);
                if (i != (int)l)
                        return(-1);
                else if (i > 0)
                        memcpy(c->iv,c->oiv,l);
                }
        return(i);
        }

int EVP_CIPHER_set_asn1_iv(EVP_CIPHER_CTX *c, ASN1_TYPE *type)
        {
        int i=0;
        unsigned int j;

        if (type != NULL)
                {
                j=EVP_CIPHER_CTX_iv_length(c);
                OPENSSL_assert(j <= sizeof(c->iv));
                i=ASN1_TYPE_set_octetstring(type,c->oiv,j);
                }
        return(i);
        }

/* Convert the various cipher NIDs and dummies to a proper OID NID */
int EVP_CIPHER_type(const EVP_CIPHER *ctx)
{
        int nid;
        ASN1_OBJECT *otmp;
        nid = EVP_CIPHER_nid(ctx);
```

```
129         switch(nid) {

131                 case NID_rc2_cbc:
132                 case NID_rc2_64_cbc:
133                 case NID_rc2_40_cbc:

135                 return NID_rc2_cbc;

137                 case NID_rc4:
138                 case NID_rc4_40:

140                 return NID_rc4;

142                 case NID_aes_128_cfb128:
143                 case NID_aes_128_cfb8:
144                 case NID_aes_128_cfb1:

146                 return NID_aes_128_cfb128;

148                 case NID_aes_192_cfb128:
149                 case NID_aes_192_cfb8:
150                 case NID_aes_192_cfb1:

152                 return NID_aes_192_cfb128;

154                 case NID_aes_256_cfb128:
155                 case NID_aes_256_cfb8:
156                 case NID_aes_256_cfb1:

158                 return NID_aes_256_cfb128;

160                 case NID_des_cfb64:
161                 case NID_des_cfb8:
162                 case NID_des_cfb1:

164                 return NID_des_cfb64;

166                 case NID_des_ede3_cfb64:
167                 case NID_des_ede3_cfb8:
168                 case NID_des_ede3_cfb1:

170                 return NID_des_cfb64;

172                 default:
173                 /* Check it has an OID and it is valid */
174                 otmp = OBJ_nid2obj(nid);
175                 if(!otmp || !otmp->data) nid = NID_undef;
176                 ASN1_OBJECT_free(otmp);
177                 return nid;
178         }
179 }

181 int EVP_CIPHER_block_size(const EVP_CIPHER *e)
182         {
183         return e->block_size;
184         }

186 int EVP_CIPHER_CTX_block_size(const EVP_CIPHER_CTX *ctx)
187         {
188         return ctx->cipher->block_size;
189         }

191 int EVP_Cipher(EVP_CIPHER_CTX *ctx, unsigned char *out, const unsigned char *in,
192         {
193         return ctx->cipher->do_cipher(ctx,out,in,inl);
```



```
194         }

196 const EVP_CIPHER *EVP_CIPHER_CTX_cipher(const EVP_CIPHER_CTX *ctx)
197         {
198         return ctx->cipher;
199         }

201 unsigned long EVP_CIPHER_flags(const EVP_CIPHER *cipher)
202         {
203         return cipher->flags;
204         }

206 unsigned long EVP_CIPHER_CTX_flags(const EVP_CIPHER_CTX *ctx)
207         {
208         return ctx->cipher->flags;
209         }

211 void *EVP_CIPHER_CTX_get_app_data(const EVP_CIPHER_CTX *ctx)
212         {
213         return ctx->app_data;
214         }

216 void EVP_CIPHER_CTX_set_app_data(EVP_CIPHER_CTX *ctx, void *data)
217         {
218         ctx->app_data = data;
219         }

221 int EVP_CIPHER_iv_length(const EVP_CIPHER *cipher)
222         {
223         return cipher->iv_len;
224         }

226 int EVP_CIPHER_CTX_iv_length(const EVP_CIPHER_CTX *ctx)
227         {
228         return ctx->cipher->iv_len;
229         }

231 int EVP_CIPHER_key_length(const EVP_CIPHER *cipher)
232         {
233         return cipher->key_len;
234         }

236 int EVP_CIPHER_CTX_key_length(const EVP_CIPHER_CTX *ctx)
237         {
238         return ctx->key_len;
239         }

241 int EVP_CIPHER_nid(const EVP_CIPHER *cipher)
242         {
243         return cipher->nid;
244         }

246 int EVP_CIPHER_CTX_nid(const EVP_CIPHER_CTX *ctx)
247         {
248         return ctx->cipher->nid;
249         }

251 int EVP_MD_block_size(const EVP_MD *md)
252         {
253         return md->block_size;
254         }

256 int EVP_MD_type(const EVP_MD *md)
257         {
258         return md->type;
259         }
```

```
261 int EVP_MD_pkey_type(const EVP_MD *md)
262         {
263         return md->pkey_type;
264         }

266 int EVP_MD_size(const EVP_MD *md)
267         {
268         if (!md)
269                 {
270                 EVPerr(EVP_F_EVP_MD_SIZE, EVP_R_MESSAGE_DIGEST_IS_NULL);
271                 return -1;
272                 }
273         return md->md_size;
274         }

276 unsigned long EVP_MD_flags(const EVP_MD *md)
277         {
278         return md->flags;
279         }

281 const EVP_MD *EVP_MD_CTX_md(const EVP_MD_CTX *ctx)
282         {
283         if (!ctx)
284                 return NULL;
285         return ctx->digest;
286         }

288 void EVP_MD_CTX_set_flags(EVP_MD_CTX *ctx, int flags)
289         {
290         ctx->flags |= flags;
291         }

293 void EVP_MD_CTX_clear_flags(EVP_MD_CTX *ctx, int flags)
294         {
295         ctx->flags &= ~flags;
296         }

298 int EVP_MD_CTX_test_flags(const EVP_MD_CTX *ctx, int flags)
299         {
300         return (ctx->flags & flags);
301         }

303 void EVP_CIPHER_CTX_set_flags(EVP_CIPHER_CTX *ctx, int flags)
304         {
305         ctx->flags |= flags;
306         }

308 void EVP_CIPHER_CTX_clear_flags(EVP_CIPHER_CTX *ctx, int flags)
309         {
310         ctx->flags &= ~flags;
311         }

313 int EVP_CIPHER_CTX_test_flags(const EVP_CIPHER_CTX *ctx, int flags)
314         {
315         return (ctx->flags & flags);
316         }
317 #endif /* ! codereview */
```

```
**********************************************************
    9133 Wed Aug 13 19:52:46 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_pbe.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* evp_pbe.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/pkcs12.h>
#include <openssl/x509.h>
#include "evp_locl.h"

/* Password based encryption (PBE) functions */

DECLARE_STACK_OF(EVP_PBE_CTL)
static STACK_OF(EVP_PBE_CTL) *pbe_algs;

/* Setup a cipher context from a PBE algorithm */

typedef struct
        {
        int pbe_type;
        int pbe_nid;
        int cipher_nid;
        int md_nid;
        EVP_PBE_KEYGEN *keygen;
        } EVP_PBE_CTL;

static const EVP_PBE_CTL builtin_pbe[] =
        {
        {EVP_PBE_TYPE_OUTER, NID_pbeWithMD2AndDES_CBC,
                        NID_des_cbc, NID_md2, PKCS5_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbeWithMD5AndDES_CBC,
                        NID_des_cbc, NID_md5, PKCS5_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbeWithSHA1AndRC2_CBC,
                        NID_rc2_64_cbc, NID_sha1, PKCS5_PBE_keyivgen},

#ifndef OPENSSL_NO_HMAC
        {EVP_PBE_TYPE_OUTER, NID_id_pbkdf2, -1, -1, PKCS5_v2_PBKDF2_keyivgen},
#endif

        {EVP_PBE_TYPE_OUTER, NID_pbe_WithSHA1And128BitRC4,
                        NID_rc4, NID_sha1, PKCS12_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbe_WithSHA1And40BitRC4,
                        NID_rc4_40, NID_sha1, PKCS12_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbe_WithSHA1And3_Key_TripleDES_CBC,
                        NID_des_ede3_cbc, NID_sha1, PKCS12_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbe_WithSHA1And2_Key_TripleDES_CBC,
                        NID_des_ede_cbc, NID_sha1, PKCS12_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbe_WithSHA1And128BitRC2_CBC,
                        NID_rc2_cbc, NID_sha1, PKCS12_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbe_WithSHA1And40BitRC2_CBC,
                        NID_rc2_40_cbc, NID_sha1, PKCS12_PBE_keyivgen},

#ifndef OPENSSL_NO_HMAC
        {EVP_PBE_TYPE_OUTER, NID_pbes2, -1, -1, PKCS5_v2_PBE_keyivgen},
#endif
        {EVP_PBE_TYPE_OUTER, NID_pbeWithMD2AndRC2_CBC,
                        NID_rc2_64_cbc, NID_md2, PKCS5_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbeWithMD5AndRC2_CBC,
                        NID_rc2_64_cbc, NID_md5, PKCS5_PBE_keyivgen},
        {EVP_PBE_TYPE_OUTER, NID_pbeWithSHA1AndDES_CBC,
                        NID_des_cbc, NID_sha1, PKCS5_PBE_keyivgen},


        {EVP_PBE_TYPE_PRF, NID_hmacWithSHA1, -1, NID_sha1, 0},
        {EVP_PBE_TYPE_PRF, NID_hmacWithMD5, -1, NID_md5, 0},
        {EVP_PBE_TYPE_PRF, NID_hmacWithSHA224, -1, NID_sha224, 0},
        {EVP_PBE_TYPE_PRF, NID_hmacWithSHA256, -1, NID_sha256, 0},
        {EVP_PBE_TYPE_PRF, NID_hmacWithSHA384, -1, NID_sha384, 0},
        {EVP_PBE_TYPE_PRF, NID_hmacWithSHA512, -1, NID_sha512, 0},
        {EVP_PBE_TYPE_PRF, NID_id_HMACGostR3411_94, -1, NID_id_GostR3411_94, 0},
        };
```

```
#ifdef TEST
int main(int argc, char **argv)
        {
        int i, nid_md, nid_cipher;
        EVP_PBE_CTL *tpbe, *tpbe2;
        /*OpenSSL_add_all_algorithms();*/

        for (i = 0; i < sizeof(builtin_pbe)/sizeof(EVP_PBE_CTL); i++)
                {
                tpbe = builtin_pbe + i;
                fprintf(stderr, "%d %d %s ", tpbe->pbe_type, tpbe->pbe_nid,
                                                OBJ_nid2sn(tpbe->pbe_nid));
                if (EVP_PBE_find(tpbe->pbe_type, tpbe->pbe_nid,
                                        &nid_cipher ,&nid_md,0))
                        fprintf(stderr, "Found %s %s\n",
                                        OBJ_nid2sn(nid_cipher),
                                        OBJ_nid2sn(nid_md));
                else
                        fprintf(stderr, "Find ERROR!!\n");
                }

        return 0;
        }
#endif



int EVP_PBE_CipherInit(ASN1_OBJECT *pbe_obj, const char *pass, int passlen,
                       ASN1_TYPE *param, EVP_CIPHER_CTX *ctx, int en_de)
        {
        const EVP_CIPHER *cipher;
        const EVP_MD *md;
        int cipher_nid, md_nid;
        EVP_PBE_KEYGEN *keygen;

        if (!EVP_PBE_find(EVP_PBE_TYPE_OUTER, OBJ_obj2nid(pbe_obj),
                                        &cipher_nid, &md_nid, &keygen))
                {
                char obj_tmp[80];
                EVPerr(EVP_F_EVP_PBE_CIPHERINIT,EVP_R_UNKNOWN_PBE_ALGORITHM);
                if (!pbe_obj) BUF_strlcpy (obj_tmp, "NULL", sizeof obj_tmp);
                else i2t_ASN1_OBJECT(obj_tmp, sizeof obj_tmp, pbe_obj);
                ERR_add_error_data(2, "TYPE=", obj_tmp);
                return 0;
                }

        if(!pass)
                passlen = 0;
        else if (passlen == -1)
                passlen = strlen(pass);

        if (cipher_nid == -1)
                cipher = NULL;
        else
                {
                cipher = EVP_get_cipherbynid(cipher_nid);
                if (!cipher)
                        {
                        EVPerr(EVP_F_EVP_PBE_CIPHERINIT,EVP_R_UNKNOWN_CIPHER);
                        return 0;
                        }
                }

        if (md_nid == -1)
                md = NULL;
        else
```



```
                {
                md = EVP_get_digestbynid(md_nid);
                if (!md)
                        {
                        EVPerr(EVP_F_EVP_PBE_CIPHERINIT,EVP_R_UNKNOWN_DIGEST);
                        return 0;
                        }
                }

        if (!keygen(ctx, pass, passlen, param, cipher, md, en_de))
                {
                EVPerr(EVP_F_EVP_PBE_CIPHERINIT,EVP_R_KEYGEN_FAILURE);
                return 0;
                }
        return 1;
}

DECLARE_OBJ_BSEARCH_CMP_FN(EVP_PBE_CTL, EVP_PBE_CTL, pbe2);

static int pbe2_cmp(const EVP_PBE_CTL *pbe1, const EVP_PBE_CTL *pbe2)
        {
        int ret = pbe1->pbe_type - pbe2->pbe_type;
        if (ret)
                return ret;
        else
                return pbe1->pbe_nid - pbe2->pbe_nid;
        }

IMPLEMENT_OBJ_BSEARCH_CMP_FN(EVP_PBE_CTL, EVP_PBE_CTL, pbe2);

static int pbe_cmp(const EVP_PBE_CTL * const *a, const EVP_PBE_CTL * const *b)
        {
        int ret = (*a)->pbe_type - (*b)->pbe_type;
        if (ret)
                return ret;
        else
                return (*a)->pbe_nid - (*b)->pbe_nid;
        }

/* Add a PBE algorithm */

int EVP_PBE_alg_add_type(int pbe_type, int pbe_nid, int cipher_nid, int md_nid,
                         EVP_PBE_KEYGEN *keygen)
        {
        EVP_PBE_CTL *pbe_tmp;
        if (!pbe_algs)
                pbe_algs = sk_EVP_PBE_CTL_new(pbe_cmp);
        if (!(pbe_tmp = (EVP_PBE_CTL*) OPENSSL_malloc (sizeof(EVP_PBE_CTL))))
                {
                EVPerr(EVP_F_EVP_PBE_ALG_ADD_TYPE,ERR_R_MALLOC_FAILURE);
                return 0;
                }
        pbe_tmp->pbe_type = pbe_type;
        pbe_tmp->pbe_nid = pbe_nid;
        pbe_tmp->cipher_nid = cipher_nid;
        pbe_tmp->md_nid = md_nid;
        pbe_tmp->keygen = keygen;


        sk_EVP_PBE_CTL_push (pbe_algs, pbe_tmp);
        return 1;
        }

int EVP_PBE_alg_add(int nid, const EVP_CIPHER *cipher, const EVP_MD *md,
                    EVP_PBE_KEYGEN *keygen)
        {
```

```
260         int cipher_nid, md_nid;
261         if (cipher)
262                 cipher_nid = EVP_CIPHER_nid(cipher);
263         else
264                 cipher_nid = -1;
265         if (md)
266                 md_nid = EVP_MD_type(md);
267         else
268                 md_nid = -1;

270         return EVP_PBE_alg_add_type(EVP_PBE_TYPE_OUTER, nid,
271                                     cipher_nid, md_nid, keygen);
272         }

274 int EVP_PBE_find(int type, int pbe_nid,
275                  int *pcnid, int *pmnid, EVP_PBE_KEYGEN **pkeygen)
276         {
277         EVP_PBE_CTL *pbetmp = NULL, pbelu;
278         int i;
279         if (pbe_nid == NID_undef)
280                 return 0;

282         pbelu.pbe_type = type;
283         pbelu.pbe_nid = pbe_nid;

285         if (pbe_algs)
286                 {
287                 i = sk_EVP_PBE_CTL_find(pbe_algs, &pbelu);
288                 if (i != -1)
289                         pbetmp = sk_EVP_PBE_CTL_value (pbe_algs, i);
290                 }
291         if (pbetmp == NULL)
292                 {
293                 pbetmp = OBJ_bsearch_pbe2(&pbelu, builtin_pbe,
294                                           sizeof(builtin_pbe)/sizeof(EVP_PBE_CTL));
295                 }
296         if (pbetmp == NULL)
297                 return 0;
298         if (pcnid)
299                 *pcnid = pbetmp->cipher_nid;
300         if (pmnid)
301                 *pmnid = pbetmp->md_nid;
302         if (pkeygen)
303                 *pkeygen = pbetmp->keygen;
304         return 1;
305         }

307 static void free_evp_pbe_ctl(EVP_PBE_CTL *pbe)
308          {
309          OPENSSL_freeFunc(pbe);
310          }

312 void EVP_PBE_cleanup(void)
313         {
314         sk_EVP_PBE_CTL_pop_free(pbe_algs, free_evp_pbe_ctl);
315         pbe_algs = NULL;
316         }
317 #endif /* ! codereview */
```

```
**********************************************************
    6600 Wed Aug 13 19:52:46 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/evp_pkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* evp_pkey.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include "cryptlib.h"
```



```
#include <openssl/x509.h>
#include <openssl/rand.h>
#include "asn1_locl.h"

/* Extract a private key from a PKCS8 structure */

EVP_PKEY *EVP_PKCS82PKEY(PKCS8_PRIV_KEY_INFO *p8)
{
        EVP_PKEY *pkey = NULL;
        ASN1_OBJECT *algoid;
        char obj_tmp[80];

        if (!PKCS8_pkey_get0(&algoid, NULL, NULL, NULL, p8))
                return NULL;

        if (!(pkey = EVP_PKEY_new())) {
                EVPerr(EVP_F_EVP_PKCS82PKEY,ERR_R_MALLOC_FAILURE);
                return NULL;
        }

        if (!EVP_PKEY_set_type(pkey, OBJ_obj2nid(algoid)))
                {
                EVPerr(EVP_F_EVP_PKCS82PKEY, EVP_R_UNSUPPORTED_PRIVATE_KEY_ALGOR
                i2t_ASN1_OBJECT(obj_tmp, 80, algoid);
                ERR_add_error_data(2, "TYPE=", obj_tmp);
                goto error;
                }

        if (pkey->ameth->priv_decode)
                {
                if (!pkey->ameth->priv_decode(pkey, p8))
                        {
                        EVPerr(EVP_F_EVP_PKCS82PKEY,
                                        EVP_R_PRIVATE_KEY_DECODE_ERROR);
                        goto error;
                        }
                }
        else
                {
                EVPerr(EVP_F_EVP_PKCS82PKEY, EVP_R_METHOD_NOT_SUPPORTED);
                goto error;
                }

        return pkey;

        error:
        EVP_PKEY_free (pkey);
        return NULL;
}

PKCS8_PRIV_KEY_INFO *EVP_PKEY2PKCS8(EVP_PKEY *pkey)
{
        return EVP_PKEY2PKCS8_broken(pkey, PKCS8_OK);
}

/* Turn a private key into a PKCS8 structure */

PKCS8_PRIV_KEY_INFO *EVP_PKEY2PKCS8_broken(EVP_PKEY *pkey, int broken)
{
        PKCS8_PRIV_KEY_INFO *p8;

        if (!(p8 = PKCS8_PRIV_KEY_INFO_new())) {
                EVPerr(EVP_F_EVP_PKEY2PKCS8_BROKEN,ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        p8->broken = broken;
```

```
129         if (pkey->ameth)
130                 {
131                 if (pkey->ameth->priv_encode)
132                         {
133                         if (!pkey->ameth->priv_encode(p8, pkey))
134                                 {
135                                 EVPerr(EVP_F_EVP_PKEY2PKCS8_BROKEN,
136                                         EVP_R_PRIVATE_KEY_ENCODE_ERROR);
137                                 goto error;
138                                 }
139                         }
140                 else
141                         {
142                         EVPerr(EVP_F_EVP_PKEY2PKCS8_BROKEN,
143                                         EVP_R_METHOD_NOT_SUPPORTED);
144                         goto error;
145                         }
146                 }
147         else
148                 {
149                 EVPerr(EVP_F_EVP_PKEY2PKCS8_BROKEN,
150                                 EVP_R_UNSUPPORTED_PRIVATE_KEY_ALGORITHM);
151                 goto error;
152                 }
153         RAND_add(p8->pkey->value.octet_string->data,
154                  p8->pkey->value.octet_string->length, 0.0);
155         return p8;
156         error:
157         PKCS8_PRIV_KEY_INFO_free(p8);
158         return NULL;
159 }

161 PKCS8_PRIV_KEY_INFO *PKCS8_set_broken(PKCS8_PRIV_KEY_INFO *p8, int broken)
162 {
163         switch (broken) {

165                 case PKCS8_OK:
166                 p8->broken = PKCS8_OK;
167                 return p8;
168                 break;

170                 case PKCS8_NO_OCTET:
171                 p8->broken = PKCS8_NO_OCTET;
172                 p8->pkey->type = V_ASN1_SEQUENCE;
173                 return p8;
174                 break;

176                 default:
177                 EVPerr(EVP_F_PKCS8_SET_BROKEN,EVP_R_PKCS8_UNKNOWN_BROKEN_TYPE);
178                 return NULL;
179         }
180 }

182 /* EVP_PKEY attribute functions */

184 int EVP_PKEY_get_attr_count(const EVP_PKEY *key)
185 {
186         return X509at_get_attr_count(key->attributes);
187 }

189 int EVP_PKEY_get_attr_by_NID(const EVP_PKEY *key, int nid,
190                              int lastpos)
191 {
192         return X509at_get_attr_by_NID(key->attributes, nid, lastpos);
193 }
```



```
195 int EVP_PKEY_get_attr_by_OBJ(const EVP_PKEY *key, ASN1_OBJECT *obj,
196                              int lastpos)
197 {
198         return X509at_get_attr_by_OBJ(key->attributes, obj, lastpos);
199 }

201 X509_ATTRIBUTE *EVP_PKEY_get_attr(const EVP_PKEY *key, int loc)
202 {
203         return X509at_get_attr(key->attributes, loc);
204 }

206 X509_ATTRIBUTE *EVP_PKEY_delete_attr(EVP_PKEY *key, int loc)
207 {
208         return X509at_delete_attr(key->attributes, loc);
209 }

211 int EVP_PKEY_add1_attr(EVP_PKEY *key, X509_ATTRIBUTE *attr)
212 {
213         if(X509at_add1_attr(&key->attributes, attr)) return 1;
214         return 0;
215 }

217 int EVP_PKEY_add1_attr_by_OBJ(EVP_PKEY *key,
218                               const ASN1_OBJECT *obj, int type,
219                               const unsigned char *bytes, int len)
220 {
221         if(X509at_add1_attr_by_OBJ(&key->attributes, obj,
222                                    type, bytes, len)) return 1;
223         return 0;
224 }

226 int EVP_PKEY_add1_attr_by_NID(EVP_PKEY *key,
227                               int nid, int type,
228                               const unsigned char *bytes, int len)
229 {
230         if(X509at_add1_attr_by_NID(&key->attributes, nid,
231                                    type, bytes, len)) return 1;
232         return 0;
233 }

235 int EVP_PKEY_add1_attr_by_txt(EVP_PKEY *key,
236                               const char *attrname, int type,
237                               const unsigned char *bytes, int len)
238 {
239         if(X509at_add1_attr_by_txt(&key->attributes, attrname,
240                                    type, bytes, len)) return 1;
241         return 0;
242 }
243 #endif /* ! codereview */
```

```
**********************************************************
    3981 Wed Aug 13 19:52:46 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_dss.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_dss.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/sha.h>
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif

#ifndef OPENSSL_NO_SHA
#ifndef OPENSSL_FIPS

static int init(EVP_MD_CTX *ctx)
        { return SHA1_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return SHA1_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return SHA1_Final(md,ctx->md_data); }

static const EVP_MD dsa_md=
        {
        NID_dsaWithSHA,
        NID_dsaWithSHA,
        SHA_DIGEST_LENGTH,
        EVP_MD_FLAG_PKEY_DIGEST,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_DSA_method,
        SHA_CBLOCK,
        sizeof(EVP_MD *)+sizeof(SHA_CTX),
        };

const EVP_MD *EVP_dss(void)
        {
        return(&dsa_md);
        }
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3981 Wed Aug 13 19:52:47 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_dss1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_dss1.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_SHA

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/sha.h>
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif

#ifndef OPENSSL_FIPS

static int init(EVP_MD_CTX *ctx)
        { return SHA1_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return SHA1_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return SHA1_Final(md,ctx->md_data); }

static const EVP_MD dss1_md=
        {
        NID_dsa,
        NID_dsaWithSHA1,
        SHA_DIGEST_LENGTH,
        EVP_MD_FLAG_PKEY_DIGEST,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_DSA_method,
        SHA_CBLOCK,
        sizeof(EVP_MD *)+sizeof(SHA_CTX),
        };

const EVP_MD *EVP_dss1(void)
        {
        return(&dss1_md);
        }
#endif
#endif
#endif /* ! codereview */
```

```
**********************************************************
    6580 Wed Aug 13 19:52:47 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_ecdsa.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_ecdsa.c */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
```



```
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>

#ifndef OPENSSL_NO_SHA
#ifndef OPENSSL_FIPS

static int init(EVP_MD_CTX *ctx)
        { return SHA1_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return SHA1_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
```

```
128         { return SHA1_Final(md,ctx->md_data); }

130 static const EVP_MD ecdsa_md=
131         {
132         NID_ecdsa_with_SHA1,
133         NID_ecdsa_with_SHA1,
134         SHA_DIGEST_LENGTH,
135         EVP_MD_FLAG_PKEY_DIGEST,
136         init,
137         update,
138         final,
139         NULL,
140         NULL,
141         EVP_PKEY_ECDSA_method,
142         SHA_CBLOCK,
143         sizeof(EVP_MD *)+sizeof(SHA_CTX),
144         };

146 const EVP_MD *EVP_ecdsa(void)
147         {
148         return(&ecdsa_md);
149         }
150 #endif
151 #endif
152 #endif /* ! codereview */
```

```
**********************************************************
    3957 Wed Aug 13 19:52:47 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_md2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_md2.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_MD2

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/md2.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif

static int init(EVP_MD_CTX *ctx)
        { return MD2_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return MD2_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return MD2_Final(md,ctx->md_data); }

static const EVP_MD md2_md=
        {
        NID_md2,
        NID_md2WithRSAEncryption,
        MD2_DIGEST_LENGTH,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_RSA_method,
        MD2_BLOCK,
        sizeof(EVP_MD *)+sizeof(MD2_CTX),
        };

const EVP_MD *EVP_md2(void)
        {
        return(&md2_md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3981 Wed Aug 13 19:52:47 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_md4.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_md4.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_MD4

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/md4.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif

#include "evp_locl.h"

static int init(EVP_MD_CTX *ctx)
        { return MD4_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return MD4_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return MD4_Final(md,ctx->md_data); }

static const EVP_MD md4_md=
        {
        NID_md4,
        NID_md4WithRSAEncryption,
        MD4_DIGEST_LENGTH,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_RSA_method,
        MD4_CBLOCK,
        sizeof(EVP_MD *)+sizeof(MD4_CTX),
        };

const EVP_MD *EVP_md4(void)
        {
        return(&md4_md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3980 Wed Aug 13 19:52:47 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_md5.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_md5.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_MD5

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/md5.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#include "evp_locl.h"

static int init(EVP_MD_CTX *ctx)
        { return MD5_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return MD5_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return MD5_Final(md,ctx->md_data); }

static const EVP_MD md5_md=
        {
        NID_md5,
        NID_md5WithRSAEncryption,
        MD5_DIGEST_LENGTH,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_RSA_method,
        MD5_CBLOCK,
        sizeof(EVP_MD *)+sizeof(MD5_CTX),
        };

const EVP_MD *EVP_md5(void)
        {
        return(&md5_md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4002 Wed Aug 13 19:52:47 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_mdc2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_mdc2.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_MDC2

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/mdc2.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif

#include "evp_locl.h"

static int init(EVP_MD_CTX *ctx)
        { return MDC2_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return MDC2_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return MDC2_Final(md,ctx->md_data); }

static const EVP_MD mdc2_md=
        {
        NID_mdc2,
        NID_mdc2WithRSA,
        MDC2_DIGEST_LENGTH,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_RSA_ASN1_OCTET_STRING_method,
        MDC2_BLOCK,
        sizeof(EVP_MD *)+sizeof(MDC2_CTX),
        };

const EVP_MD *EVP_mdc2(void)
        {
        return(&mdc2_md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3720 Wed Aug 13 19:52:48 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_null.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_null.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>

static int init(EVP_MD_CTX *ctx)
        { return 1; }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return 1; }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return 1; }

static const EVP_MD null_md=
        {
        NID_undef,
        NID_undef,
        0,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_NULL_method,
        0,
        sizeof(EVP_MD *),
        };

const EVP_MD *EVP_md_null(void)
        {
        return(&null_md);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4045 Wed Aug 13 19:52:48 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_ripemd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_ripemd.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_RIPEMD

#include <openssl/ripemd.h>
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#include "evp_locl.h"

static int init(EVP_MD_CTX *ctx)
        { return RIPEMD160_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return RIPEMD160_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return RIPEMD160_Final(md,ctx->md_data); }

static const EVP_MD ripemd160_md=
        {
        NID_ripemd160,
        NID_ripemd160WithRSA,
        RIPEMD160_DIGEST_LENGTH,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_RSA_method,
        RIPEMD160_CBLOCK,
        sizeof(EVP_MD *)+sizeof(RIPEMD160_CTX),
        };

const EVP_MD *EVP_ripemd160(void)
        {
        return(&ripemd160_md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3990 Wed Aug 13 19:52:48 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_sha.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_sha.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA0)

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#include "evp_locl.h"

static int init(EVP_MD_CTX *ctx)
        { return SHA_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return SHA_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return SHA_Final(md,ctx->md_data); }

static const EVP_MD sha_md=
        {
        NID_sha,
        NID_shaWithRSAEncryption,
        SHA_DIGEST_LENGTH,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_RSA_method,
        SHA_CBLOCK,
        sizeof(EVP_MD *)+sizeof(SHA_CTX),
        };

const EVP_MD *EVP_sha(void)
        {
        return(&sha_md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    6521 Wed Aug 13 19:52:48 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_sha1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_sha1.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_FIPS

#ifndef OPENSSL_NO_SHA

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/sha.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif


static int init(EVP_MD_CTX *ctx)
        { return SHA1_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return SHA1_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return SHA1_Final(md,ctx->md_data); }

static const EVP_MD sha1_md=
        {
        NID_sha1,
        NID_sha1WithRSAEncryption,
        SHA_DIGEST_LENGTH,
        EVP_MD_FLAG_PKEY_METHOD_SIGNATURE|EVP_MD_FLAG_DIGALGID_ABSENT,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_RSA_method,
        SHA_CBLOCK,
        sizeof(EVP_MD *)+sizeof(SHA_CTX),
        };

const EVP_MD *EVP_sha1(void)
        {
        return(&sha1_md);
        }
#endif

#ifndef OPENSSL_NO_SHA256
static int init224(EVP_MD_CTX *ctx)
        { return SHA224_Init(ctx->md_data); }
static int init256(EVP_MD_CTX *ctx)
        { return SHA256_Init(ctx->md_data); }
/*
 * Even though there're separate SHA224_[Update|Final], we call
 * SHA256 functions even in SHA224 context. This is what happens
 * there anyway, so we can spare few CPU cycles:-)
 */
static int update256(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return SHA256_Update(ctx->md_data,data,count); }
static int final256(EVP_MD_CTX *ctx,unsigned char *md)
        { return SHA256_Final(md,ctx->md_data); }

static const EVP_MD sha224_md=
        {
        NID_sha224,
        NID_sha224WithRSAEncryption,
        SHA224_DIGEST_LENGTH,
        EVP_MD_FLAG_PKEY_METHOD_SIGNATURE|EVP_MD_FLAG_DIGALGID_ABSENT,
        init224,
        update256,
```

```
128         final256,
129         NULL,
130         NULL,
131         EVP_PKEY_RSA_method,
132         SHA256_CBLOCK,
133         sizeof(EVP_MD *)+sizeof(SHA256_CTX),
134         };

136 const EVP_MD *EVP_sha224(void)
137         { return(&sha224_md); }

139 static const EVP_MD sha256_md=
140         {
141         NID_sha256,
142         NID_sha256WithRSAEncryption,
143         SHA256_DIGEST_LENGTH,
144         EVP_MD_FLAG_PKEY_METHOD_SIGNATURE|EVP_MD_FLAG_DIGALGID_ABSENT,
145         init256,
146         update256,
147         final256,
148         NULL,
149         NULL,
150         EVP_PKEY_RSA_method,
151         SHA256_CBLOCK,
152         sizeof(EVP_MD *)+sizeof(SHA256_CTX),
153         };

155 const EVP_MD *EVP_sha256(void)
156         { return(&sha256_md); }
157 #endif  /* ifndef OPENSSL_NO_SHA256 */

159 #ifndef OPENSSL_NO_SHA512
160 static int init384(EVP_MD_CTX *ctx)
161         { return SHA384_Init(ctx->md_data); }
162 static int init512(EVP_MD_CTX *ctx)
163         { return SHA512_Init(ctx->md_data); }
164 /* See comment in SHA224/256 section */
165 static int update512(EVP_MD_CTX *ctx,const void *data,size_t count)
166         { return SHA512_Update(ctx->md_data,data,count); }
167 static int final512(EVP_MD_CTX *ctx,unsigned char *md)
168         { return SHA512_Final(md,ctx->md_data); }

170 static const EVP_MD sha384_md=
171         {
172         NID_sha384,
173         NID_sha384WithRSAEncryption,
174         SHA384_DIGEST_LENGTH,
175         EVP_MD_FLAG_PKEY_METHOD_SIGNATURE|EVP_MD_FLAG_DIGALGID_ABSENT,
176         init384,
177         update512,
178         final512,
179         NULL,
180         NULL,
181         EVP_PKEY_RSA_method,
182         SHA512_CBLOCK,
183         sizeof(EVP_MD *)+sizeof(SHA512_CTX),
184         };

186 const EVP_MD *EVP_sha384(void)
187         { return(&sha384_md); }

189 static const EVP_MD sha512_md=
190         {
191         NID_sha512,
192         NID_sha512WithRSAEncryption,
193         SHA512_DIGEST_LENGTH,
```



```
194         EVP_MD_FLAG_PKEY_METHOD_SIGNATURE|EVP_MD_FLAG_DIGALGID_ABSENT,
195         init512,
196         update512,
197         final512,
198         NULL,
199         NULL,
200         EVP_PKEY_RSA_method,
201         SHA512_CBLOCK,
202         sizeof(EVP_MD *)+sizeof(SHA512_CTX),
203         };

205 const EVP_MD *EVP_sha512(void)
206         { return(&sha512_md); }
207 #endif  /* ifndef OPENSSL_NO_SHA512 */

209 #endif
210 #endif /* ! codereview */
```

```
**********************************************************
    5876 Wed Aug 13 19:52:48 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_sigver.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* m_sigver.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006,2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>
#include "evp_locl.h"

static int do_sigver_init(EVP_MD_CTX *ctx, EVP_PKEY_CTX **pctx,
                          const EVP_MD *type, ENGINE *e, EVP_PKEY *pkey,
                          int ver)
        {
        if (ctx->pctx == NULL)
                ctx->pctx = EVP_PKEY_CTX_new(pkey, e);
        if (ctx->pctx == NULL)
                return 0;

        if (type == NULL)
                {
                int def_nid;
                if (EVP_PKEY_get_default_digest_nid(pkey, &def_nid) > 0)
                        type = EVP_get_digestbynid(def_nid);
                }

        if (type == NULL)
                {
                EVPerr(EVP_F_DO_SIGVER_INIT, EVP_R_NO_DEFAULT_DIGEST);
                return 0;
                }

        if (ver)
                {
                if (ctx->pctx->pmeth->verifyctx_init)
                        {
                        if (ctx->pctx->pmeth->verifyctx_init(ctx->pctx, ctx) <=0
                                return 0;
                        ctx->pctx->operation = EVP_PKEY_OP_VERIFYCTX;
                        }
                else if (EVP_PKEY_verify_init(ctx->pctx) <= 0)
                        return 0;
                }
        else
                {
                if (ctx->pctx->pmeth->signctx_init)
                        {
                        if (ctx->pctx->pmeth->signctx_init(ctx->pctx, ctx) <= 0)
                                return 0;
                        ctx->pctx->operation = EVP_PKEY_OP_SIGNCTX;
                        }
                else if (EVP_PKEY_sign_init(ctx->pctx) <= 0)
                        return 0;
                }
        if (EVP_PKEY_CTX_set_signature_md(ctx->pctx, type) <= 0)
                return 0;
        if (pctx)
                *pctx = ctx->pctx;
        if (!EVP_DigestInit_ex(ctx, type, e))
                return 0;
        return 1;
        }

int EVP_DigestSignInit(EVP_MD_CTX *ctx, EVP_PKEY_CTX **pctx,
                        const EVP_MD *type, ENGINE *e, EVP_PKEY *pkey)
        {
        return do_sigver_init(ctx, pctx, type, e, pkey, 0);
        }

int EVP_DigestVerifyInit(EVP_MD_CTX *ctx, EVP_PKEY_CTX **pctx,
                        const EVP_MD *type, ENGINE *e, EVP_PKEY *pkey)
        {
```

```
128         return do_sigver_init(ctx, pctx, type, e, pkey, 1);
129         }

131 int EVP_DigestSignFinal(EVP_MD_CTX *ctx, unsigned char *sigret, size_t *siglen)
132         {
133         int sctx, r = 0;
134         if (ctx->pctx->pmeth->signctx)
135                 sctx = 1;
136         else
137                 sctx = 0;
138         if (sigret)
139                 {
140                 EVP_MD_CTX tmp_ctx;
141                 unsigned char md[EVP_MAX_MD_SIZE];
142                 unsigned int mdlen;
143                 EVP_MD_CTX_init(&tmp_ctx);
144                 if (!EVP_MD_CTX_copy_ex(&tmp_ctx,ctx))
145                         return 0;
146                 if (sctx)
147                         r = tmp_ctx.pctx->pmeth->signctx(tmp_ctx.pctx,
148                                         sigret, siglen, &tmp_ctx);
149                 else
150                         r = EVP_DigestFinal_ex(&tmp_ctx,md,&mdlen);
151                 EVP_MD_CTX_cleanup(&tmp_ctx);
152                 if (sctx || !r)
153                         return r;
154                 if (EVP_PKEY_sign(ctx->pctx, sigret, siglen, md, mdlen) <= 0)
155                         return 0;
156                 }
157         else
158                 {
159                 if (sctx)
160                         {
161                         if (ctx->pctx->pmeth->signctx(ctx->pctx, sigret, siglen,
162                                 return 0;
163                         }
164                 else
165                         {
166                         int s = EVP_MD_size(ctx->digest);
167                         if (s < 0 || EVP_PKEY_sign(ctx->pctx, sigret, siglen, NU
168                                 return 0;
169                         }
170                 }
171         return 1;
172         }

174 int EVP_DigestVerifyFinal(EVP_MD_CTX *ctx, unsigned char *sig, size_t siglen)
175         {
176         EVP_MD_CTX tmp_ctx;
177         unsigned char md[EVP_MAX_MD_SIZE];
178         int r;
179         unsigned int mdlen;
180         int vctx;

182         if (ctx->pctx->pmeth->verifyctx)
183                 vctx = 1;
184         else
185                 vctx = 0;
186         EVP_MD_CTX_init(&tmp_ctx);
187         if (!EVP_MD_CTX_copy_ex(&tmp_ctx,ctx))
188                 return -1;
189         if (vctx)
190                 {
191                 r = tmp_ctx.pctx->pmeth->verifyctx(tmp_ctx.pctx,
192                                 sig, siglen, &tmp_ctx);
193                 }
```



```
194         else
195                 r = EVP_DigestFinal_ex(&tmp_ctx,md,&mdlen);
196         EVP_MD_CTX_cleanup(&tmp_ctx);
197         if (vctx || !r)
198                 return r;
199         return EVP_PKEY_verify(ctx->pctx, sig, siglen, md, mdlen);
200         }
201 #endif /* ! codereview */
```

```
**********************************************************
     812 Wed Aug 13 19:52:48 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/m_wp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/m_wp.c */

#include <stdio.h>
#include "cryptlib.h"

#ifndef OPENSSL_NO_WHIRLPOOL

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/whrlpool.h>
#include "evp_locl.h"

static int init(EVP_MD_CTX *ctx)
        { return WHIRLPOOL_Init(ctx->md_data); }

static int update(EVP_MD_CTX *ctx,const void *data,size_t count)
        { return WHIRLPOOL_Update(ctx->md_data,data,count); }

static int final(EVP_MD_CTX *ctx,unsigned char *md)
        { return WHIRLPOOL_Final(md,ctx->md_data); }

static const EVP_MD whirlpool_md=
        {
        NID_whirlpool,
        0,
        WHIRLPOOL_DIGEST_LENGTH,
        0,
        init,
        update,
        final,
        NULL,
        NULL,
        EVP_PKEY_NULL_method,
        WHIRLPOOL_BBLOCK/8,
        sizeof(EVP_MD *)+sizeof(WHIRLPOOL_CTX),
        };

const EVP_MD *EVP_whirlpool(void)
        {
        return(&whirlpool_md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    6672 Wed Aug 13 19:52:49 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/names.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/names.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>

int EVP_add_cipher(const EVP_CIPHER *c)
        {
        int r;

        if (c == NULL) return 0;

        OPENSSL_init();

        r=OBJ_NAME_add(OBJ_nid2sn(c->nid),OBJ_NAME_TYPE_CIPHER_METH,(const char
        if (r == 0) return(0);
        check_defer(c->nid);
        r=OBJ_NAME_add(OBJ_nid2ln(c->nid),OBJ_NAME_TYPE_CIPHER_METH,(const char
        return(r);
        }


int EVP_add_digest(const EVP_MD *md)
        {
        int r;
        const char *name;
        OPENSSL_init();

        name=OBJ_nid2sn(md->type);
        r=OBJ_NAME_add(name,OBJ_NAME_TYPE_MD_METH,(const char *)md);
        if (r == 0) return(0);
        check_defer(md->type);
        r=OBJ_NAME_add(OBJ_nid2ln(md->type),OBJ_NAME_TYPE_MD_METH,(const char *)
        if (r == 0) return(0);

        if (md->pkey_type && md->type != md->pkey_type)
                {
                r=OBJ_NAME_add(OBJ_nid2sn(md->pkey_type),
                        OBJ_NAME_TYPE_MD_METH|OBJ_NAME_ALIAS,name);
                if (r == 0) return(0);
                check_defer(md->pkey_type);
                r=OBJ_NAME_add(OBJ_nid2ln(md->pkey_type),
                        OBJ_NAME_TYPE_MD_METH|OBJ_NAME_ALIAS,name);
                }
        return(r);
        }

const EVP_CIPHER *EVP_get_cipherbyname(const char *name)
        {
        const EVP_CIPHER *cp;

        cp=(const EVP_CIPHER *)OBJ_NAME_get(name,OBJ_NAME_TYPE_CIPHER_METH);
        return(cp);
        }

const EVP_MD *EVP_get_digestbyname(const char *name)
        {
        const EVP_MD *cp;

        cp=(const EVP_MD *)OBJ_NAME_get(name,OBJ_NAME_TYPE_MD_METH);
        return(cp);
        }

void EVP_cleanup(void)
        {
        OBJ_NAME_cleanup(OBJ_NAME_TYPE_CIPHER_METH);
        OBJ_NAME_cleanup(OBJ_NAME_TYPE_MD_METH);
        /* The above calls will only clean out the contents of the name
           hash table, but not the hash table itself.  The following line
```

```
128            does that part.  -- Richard Levitte */
129         OBJ_NAME_cleanup(-1);

131         EVP_PBE_cleanup();
132         if (obj_cleanup_defer == 2)
133                 {
134                 obj_cleanup_defer = 0;
135                 OBJ_cleanup();
136                 }
137         OBJ_sigid_free();
138         }

140 struct doall_cipher
141         {
142         void *arg;
143         void (*fn)(const EVP_CIPHER *ciph,
144                         const char *from, const char *to, void *arg);
145         };

147 static void do_all_cipher_fn(const OBJ_NAME *nm, void *arg)
148         {
149         struct doall_cipher *dc = arg;
150         if (nm->alias)
151                 dc->fn(NULL, nm->name, nm->data, dc->arg);
152         else
153                 dc->fn((const EVP_CIPHER *)nm->data, nm->name, NULL, dc->arg);
154         }

156 void EVP_CIPHER_do_all(void (*fn)(const EVP_CIPHER *ciph,
157                 const char *from, const char *to, void *x), void *arg)
158         {
159         struct doall_cipher dc;
160         dc.fn = fn;
161         dc.arg = arg;
162         OBJ_NAME_do_all(OBJ_NAME_TYPE_CIPHER_METH, do_all_cipher_fn, &dc);
163         }

165 void EVP_CIPHER_do_all_sorted(void (*fn)(const EVP_CIPHER *ciph,
166                 const char *from, const char *to, void *x), void *arg)
167         {
168         struct doall_cipher dc;
169         dc.fn = fn;
170         dc.arg = arg;
171         OBJ_NAME_do_all_sorted(OBJ_NAME_TYPE_CIPHER_METH, do_all_cipher_fn,&dc);
172         }

174 struct doall_md
175         {
176         void *arg;
177         void (*fn)(const EVP_MD *ciph,
178                         const char *from, const char *to, void *arg);
179         };

181 static void do_all_md_fn(const OBJ_NAME *nm, void *arg)
182         {
183         struct doall_md *dc = arg;
184         if (nm->alias)
185                 dc->fn(NULL, nm->name, nm->data, dc->arg);
186         else
187                 dc->fn((const EVP_MD *)nm->data, nm->name, NULL, dc->arg);
188         }

190 void EVP_MD_do_all(void (*fn)(const EVP_MD *md,
191                 const char *from, const char *to, void *x), void *arg)
192         {
193         struct doall_md dc;
```



```
194         dc.fn = fn;
195         dc.arg = arg;
196         OBJ_NAME_do_all(OBJ_NAME_TYPE_MD_METH, do_all_md_fn, &dc);
197         }

199 void EVP_MD_do_all_sorted(void (*fn)(const EVP_MD *md,
200                 const char *from, const char *to, void *x), void *arg)
201         {
202         struct doall_md dc;
203         dc.fn = fn;
204         dc.arg = arg;
205         OBJ_NAME_do_all_sorted(OBJ_NAME_TYPE_MD_METH, do_all_md_fn, &dc);
206         }
207 #endif /* ! codereview */
```

```
**********************************************************
    5004 Wed Aug 13 19:52:49 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p5_crpt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p5_crpt.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include "cryptlib.h"
```



```
#include <openssl/x509.h>
#include <openssl/evp.h>

/* Doesn't do anything now: Builtin PBE algorithms in static table.
 */

void PKCS5_PBE_add(void)
{
}

int PKCS5_PBE_keyivgen(EVP_CIPHER_CTX *cctx, const char *pass, int passlen,
                       ASN1_TYPE *param, const EVP_CIPHER *cipher, const EVP_M
                       int en_de)
{
        EVP_MD_CTX ctx;
        unsigned char md_tmp[EVP_MAX_MD_SIZE];
        unsigned char key[EVP_MAX_KEY_LENGTH], iv[EVP_MAX_IV_LENGTH];
        int i;
        PBEPARAM *pbe;
        int saltlen, iter;
        unsigned char *salt;
        const unsigned char *pbuf;
        int mdsize;
        int rv = 0;
        EVP_MD_CTX_init(&ctx);

        /* Extract useful info from parameter */
        if (param == NULL || param->type != V_ASN1_SEQUENCE ||
            param->value.sequence == NULL) {
                EVPerr(EVP_F_PKCS5_PBE_KEYIVGEN,EVP_R_DECODE_ERROR);
                return 0;
        }

        pbuf = param->value.sequence->data;
        if (!(pbe = d2i_PBEPARAM(NULL, &pbuf, param->value.sequence->length))) {
                EVPerr(EVP_F_PKCS5_PBE_KEYIVGEN,EVP_R_DECODE_ERROR);
                return 0;
        }

        if (!pbe->iter) iter = 1;
        else iter = ASN1_INTEGER_get (pbe->iter);
        salt = pbe->salt->data;
        saltlen = pbe->salt->length;

        if(!pass) passlen = 0;
        else if(passlen == -1) passlen = strlen(pass);

        if (!EVP_DigestInit_ex(&ctx, md, NULL))
                goto err;
        if (!EVP_DigestUpdate(&ctx, pass, passlen))
                goto err;
        if (!EVP_DigestUpdate(&ctx, salt, saltlen))
                goto err;
        PBEPARAM_free(pbe);
        if (!EVP_DigestFinal_ex(&ctx, md_tmp, NULL))
                goto err;
        mdsize = EVP_MD_size(md);
        if (mdsize < 0)
            return 0;
        for (i = 1; i < iter; i++) {
                if (!EVP_DigestInit_ex(&ctx, md, NULL))
                        goto err;
                if (!EVP_DigestUpdate(&ctx, md_tmp, mdsize))
                        goto err;
                if (!EVP_DigestFinal_ex (&ctx, md_tmp, NULL))
                        goto err;
```

```
128         }
129         OPENSSL_assert(EVP_CIPHER_key_length(cipher) <= (int)sizeof(md_tmp));
130         memcpy(key, md_tmp, EVP_CIPHER_key_length(cipher));
131         OPENSSL_assert(EVP_CIPHER_iv_length(cipher) <= 16);
132         memcpy(iv, md_tmp + (16 - EVP_CIPHER_iv_length(cipher)),
133                                                 EVP_CIPHER_iv_length(cipher));
134         if (!EVP_CipherInit_ex(cctx, cipher, NULL, key, iv, en_de))
135                 goto err;
136         OPENSSL_cleanse(md_tmp, EVP_MAX_MD_SIZE);
137         OPENSSL_cleanse(key, EVP_MAX_KEY_LENGTH);
138         OPENSSL_cleanse(iv, EVP_MAX_IV_LENGTH);
139         rv = 1;
140         err:
141         EVP_MD_CTX_cleanup(&ctx);
142         return rv;
143 }
144 #endif /* ! codereview */
```

```
**********************************************************
   10001 Wed Aug 13 19:52:49 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p5_crpt2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p5_crpt2.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include "cryptlib.h"
#if !defined(OPENSSL_NO_HMAC) && !defined(OPENSSL_NO_SHA)
```



```
#include <openssl/x509.h>
#include <openssl/evp.h>
#include <openssl/hmac.h>
#include "evp_locl.h"

/* set this to print out info about the keygen algorithm */
/* #define DEBUG_PKCS5V2 */

#ifdef DEBUG_PKCS5V2
        static void h__dump (const unsigned char *p, int len);
#endif

/* This is an implementation of PKCS#5 v2.0 password based encryption key
 * derivation function PBKDF2.
 * SHA1 version verified against test vectors posted by Peter Gutmann
 * <pgut001@cs.auckland.ac.nz> to the PKCS-TNG <pkcs-tng@rsa.com> mailing list.
 */

int PKCS5_PBKDF2_HMAC(const char *pass, int passlen,
                          const unsigned char *salt, int saltlen, int iter,
                          const EVP_MD *digest,
                          int keylen, unsigned char *out)
        {
        unsigned char digtmp[EVP_MAX_MD_SIZE], *p, itmp[4];
        int cplen, j, k, tkeylen, mdlen;
        unsigned long i = 1;
        HMAC_CTX hctx_tpl, hctx;

        mdlen = EVP_MD_size(digest);
        if (mdlen < 0)
                return 0;

        HMAC_CTX_init(&hctx_tpl);
        p = out;
        tkeylen = keylen;
        if(!pass)
                passlen = 0;
        else if(passlen == -1)
                passlen = strlen(pass);
        if (!HMAC_Init_ex(&hctx_tpl, pass, passlen, digest, NULL))
                {
                HMAC_CTX_cleanup(&hctx_tpl);
                return 0;
                }
        while(tkeylen)
                {
                if(tkeylen > mdlen)
                        cplen = mdlen;
                else
                        cplen = tkeylen;
                /* We are unlikely to ever use more than 256 blocks (5120 bits!)
                 * but just in case...
                 */
                itmp[0] = (unsigned char)((i >> 24) & 0xff);
                itmp[1] = (unsigned char)((i >> 16) & 0xff);
                itmp[2] = (unsigned char)((i >> 8) & 0xff);
                itmp[3] = (unsigned char)(i & 0xff);
                if (!HMAC_CTX_copy(&hctx, &hctx_tpl))
                        {
                        HMAC_CTX_cleanup(&hctx_tpl);
                        return 0;
                        }
                if (!HMAC_Update(&hctx, salt, saltlen)
                    || !HMAC_Update(&hctx, itmp, 4)
                    || !HMAC_Final(&hctx, digtmp, NULL))
                        {
```

```
128                         HMAC_CTX_cleanup(&hctx_tpl);
129                         HMAC_CTX_cleanup(&hctx);
130                         return 0;
131                         }
132                 HMAC_CTX_cleanup(&hctx);
133                 memcpy(p, digtmp, cplen);
134                 for(j = 1; j < iter; j++)
135                         {
136                         if (!HMAC_CTX_copy(&hctx, &hctx_tpl))
137                                 {
138                                 HMAC_CTX_cleanup(&hctx_tpl);
139                                 return 0;
140                                 }
141                         if (!HMAC_Update(&hctx, digtmp, mdlen)
142                             || !HMAC_Final(&hctx, digtmp, NULL))
143                                 {
144                                 HMAC_CTX_cleanup(&hctx_tpl);
145                                 HMAC_CTX_cleanup(&hctx);
146                                 return 0;
147                                 }
148                         HMAC_CTX_cleanup(&hctx);
149                         for(k = 0; k < cplen; k++)
150                                 p[k] ^= digtmp[k];
151                         }
152                 tkeylen-= cplen;
153                 i++;
154                 p+= cplen;
155                 }
156         HMAC_CTX_cleanup(&hctx_tpl);
157 #ifdef DEBUG_PKCS5V2
158         fprintf(stderr, "Password:\n");
159         h__dump (pass, passlen);
160         fprintf(stderr, "Salt:\n");
161         h__dump (salt, saltlen);
162         fprintf(stderr, "Iteration count %d\n", iter);
163         fprintf(stderr, "Key:\n");
164         h__dump (out, keylen);
165 #endif
166         return 1;
167         }

169 int PKCS5_PBKDF2_HMAC_SHA1(const char *pass, int passlen,
170                            const unsigned char *salt, int saltlen, int iter,
171                            int keylen, unsigned char *out)
172         {
173         return PKCS5_PBKDF2_HMAC(pass, passlen, salt, saltlen, iter, EVP_sha1(),
174                                  keylen, out);
175         }

177 #ifdef DO_TEST
178 main()
179 {
180         unsigned char out[4];
181         unsigned char salt[] = {0x12, 0x34, 0x56, 0x78};
182         PKCS5_PBKDF2_HMAC_SHA1("password", -1, salt, 4, 5, 4, out);
183         fprintf(stderr, "Out %02X %02X %02X %02X\n",
184                                  out[0], out[1], out[2], out[3]);
185 }

187 #endif

189 /* Now the key derivation function itself. This is a bit evil because
190  * it has to check the ASN1 parameters are valid: and there are quite a
191  * few of them...
192  */
```



```
194 int PKCS5_v2_PBE_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
195                           ASN1_TYPE *param, const EVP_CIPHER *c, const EVP_MD *md
196                           int en_de)
197 {
198         const unsigned char *pbuf;
199         int plen;
200         PBE2PARAM *pbe2 = NULL;
201         const EVP_CIPHER *cipher;

203         int rv = 0;

205         if (param == NULL || param->type != V_ASN1_SEQUENCE ||
206             param->value.sequence == NULL) {
207                 EVPerr(EVP_F_PKCS5_V2_PBE_KEYIVGEN,EVP_R_DECODE_ERROR);
208                 goto err;
209         }

211         pbuf = param->value.sequence->data;
212         plen = param->value.sequence->length;
213         if(!(pbe2 = d2i_PBE2PARAM(NULL, &pbuf, plen))) {
214                 EVPerr(EVP_F_PKCS5_V2_PBE_KEYIVGEN,EVP_R_DECODE_ERROR);
215                 goto err;
216         }

218         /* See if we recognise the key derivation function */

220         if(OBJ_obj2nid(pbe2->keyfunc->algorithm) != NID_id_pbkdf2) {
221                 EVPerr(EVP_F_PKCS5_V2_PBE_KEYIVGEN,
222                                 EVP_R_UNSUPPORTED_KEY_DERIVATION_FUNCTION);
223                 goto err;
224         }

226         /* lets see if we recognise the encryption algorithm.
227          */

229         cipher = EVP_get_cipherbyobj(pbe2->encryption->algorithm);

231         if(!cipher) {
232                 EVPerr(EVP_F_PKCS5_V2_PBE_KEYIVGEN,
233                                                 EVP_R_UNSUPPORTED_CIPHER);
234                 goto err;
235         }

237         /* Fixup cipher based on AlgorithmIdentifier */
238         if (!EVP_CipherInit_ex(ctx, cipher, NULL, NULL, NULL, en_de))
239                 goto err;
240         if(EVP_CIPHER_asn1_to_param(ctx, pbe2->encryption->parameter) < 0) {
241                 EVPerr(EVP_F_PKCS5_V2_PBE_KEYIVGEN,
242                                         EVP_R_CIPHER_PARAMETER_ERROR);
243                 goto err;
244         }
245         rv = PKCS5_v2_PBKDF2_keyivgen(ctx, pass, passlen,
246                                         pbe2->keyfunc->parameter, c, md, en_de);
247         err:
248         PBE2PARAM_free(pbe2);
249         return rv;
250 }

252 int PKCS5_v2_PBKDF2_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
253                           ASN1_TYPE *param,
254                           const EVP_CIPHER *c, const EVP_MD *md, int en_de)
255 {
256         unsigned char *salt, key[EVP_MAX_KEY_LENGTH];
257         const unsigned char *pbuf;
258         int saltlen, iter, plen;
259         int rv = 0;
```

```
260         unsigned int keylen = 0;
261         int prf_nid, hmac_md_nid;
262         PBKDF2PARAM *kdf = NULL;
263         const EVP_MD *prfmd;
264
265         if (EVP_CIPHER_CTX_cipher(ctx) == NULL)
266                 {
267                 EVPerr(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN,EVP_R_NO_CIPHER_SET);
268                 goto err;
269                 }
270         keylen = EVP_CIPHER_CTX_key_length(ctx);
271         OPENSSL_assert(keylen <= sizeof key);
272
273         /* Decode parameter */
274
275         if(!param || (param->type != V_ASN1_SEQUENCE))
276                 {
277                 EVPerr(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN,EVP_R_DECODE_ERROR);
278                 goto err;
279                 }
280
281         pbuf = param->value.sequence->data;
282         plen = param->value.sequence->length;
283
284         if(!(kdf = d2i_PBKDF2PARAM(NULL, &pbuf, plen)) ) {
285                 EVPerr(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN,EVP_R_DECODE_ERROR);
286                 goto err;
287         }
288
289         keylen = EVP_CIPHER_CTX_key_length(ctx);
290
291         /* Now check the parameters of the kdf */
292
293         if(kdf->keylength && (ASN1_INTEGER_get(kdf->keylength) != (int)keylen)){
294                 EVPerr(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN,
295                                                 EVP_R_UNSUPPORTED_KEYLENGTH);
296                 goto err;
297         }
298
299         if (kdf->prf)
300                 prf_nid = OBJ_obj2nid(kdf->prf->algorithm);
301         else
302                 prf_nid = NID_hmacWithSHA1;
303
304         if (!EVP_PBE_find(EVP_PBE_TYPE_PRF, prf_nid, NULL, &hmac_md_nid, 0))
305                 {
306                 EVPerr(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN, EVP_R_UNSUPPORTED_PRF);
307                 goto err;
308                 }
309
310         prfmd = EVP_get_digestbynid(hmac_md_nid);
311         if (prfmd == NULL)
312                 {
313                 EVPerr(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN, EVP_R_UNSUPPORTED_PRF);
314                 goto err;
315                 }
316
317         if(kdf->salt->type != V_ASN1_OCTET_STRING) {
318                 EVPerr(EVP_F_PKCS5_V2_PBKDF2_KEYIVGEN,
319                                                 EVP_R_UNSUPPORTED_SALT_TYPE);
320                 goto err;
321         }
322
323         /* it seems that its all OK */
324         salt = kdf->salt->value.octet_string->data;
325         saltlen = kdf->salt->value.octet_string->length;
```



```
326         iter = ASN1_INTEGER_get(kdf->iter);
327         if(!PKCS5_PBKDF2_HMAC(pass, passlen, salt, saltlen, iter, prfmd,
328                                                 keylen, key))
329                 goto err;
330         rv = EVP_CipherInit_ex(ctx, NULL, NULL, key, NULL, en_de);
331         err:
332         OPENSSL_cleanse(key, keylen);
333         PBKDF2PARAM_free(kdf);
334         return rv;
335 }
336
337 #ifdef DEBUG_PKCS5V2
338 static void h__dump (const unsigned char *p, int len)
339 {
340         for (; len --; p++) fprintf(stderr, "%02X ", *p);
341         fprintf(stderr, "\n");
342 }
343 #endif
344 #endif
345 #endif /* ! codereview */
```

```
**********************************************************
    3795 Wed Aug 13 19:52:49 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p_dec.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/p_dec.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/rand.h>
```



```
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>

int EVP_PKEY_decrypt_old(unsigned char *key, const unsigned char *ek, int ekl,
             EVP_PKEY *priv)
        {
        int ret= -1;

#ifndef OPENSSL_NO_RSA
        if (priv->type != EVP_PKEY_RSA)
                {
#endif
                EVPerr(EVP_F_EVP_PKEY_DECRYPT_OLD,EVP_R_PUBLIC_KEY_NOT_RSA);
#ifndef OPENSSL_NO_RSA
                goto err;
                }

        ret=RSA_private_decrypt(ekl,ek,key,priv->pkey.rsa,RSA_PKCS1_PADDING);
err:
#endif
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    3785 Wed Aug 13 19:52:49 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/p_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/rand.h>
```



```
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>

int EVP_PKEY_encrypt_old(unsigned char *ek, const unsigned char *key, int key_le
             EVP_PKEY *pubk)
        {
        int ret=0;

#ifndef OPENSSL_NO_RSA
        if (pubk->type != EVP_PKEY_RSA)
                {
#endif
                EVPerr(EVP_F_EVP_PKEY_ENCRYPT_OLD,EVP_R_PUBLIC_KEY_NOT_RSA);
#ifndef OPENSSL_NO_RSA
                goto err;
                }
        ret=RSA_public_encrypt(key_len,key,ek,pubk->pkey.rsa,RSA_PKCS1_PADDING);
err:
#endif
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
   11296 Wed Aug 13 19:52:50 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/p_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/err.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/asn1_mac.h>
#include <openssl/x509.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif

#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

#include "asn1_locl.h"

static void EVP_PKEY_free_it(EVP_PKEY *x);

int EVP_PKEY_bits(EVP_PKEY *pkey)
        {
        if (pkey && pkey->ameth && pkey->ameth->pkey_bits)
                return pkey->ameth->pkey_bits(pkey);
        return 0;
        }

int EVP_PKEY_size(EVP_PKEY *pkey)
        {
        if (pkey && pkey->ameth && pkey->ameth->pkey_size)
                return pkey->ameth->pkey_size(pkey);
        return 0;
        }

int EVP_PKEY_save_parameters(EVP_PKEY *pkey, int mode)
        {
#ifndef OPENSSL_NO_DSA
        if (pkey->type == EVP_PKEY_DSA)
                {
                int ret=pkey->save_parameters;

                if (mode >= 0)
                        pkey->save_parameters=mode;
                return(ret);
                }
#endif
#ifndef OPENSSL_NO_EC
        if (pkey->type == EVP_PKEY_EC)
                {
                int ret = pkey->save_parameters;

                if (mode >= 0)
                        pkey->save_parameters = mode;
                return(ret);
                }
#endif
        return(0);
        }

int EVP_PKEY_copy_parameters(EVP_PKEY *to, const EVP_PKEY *from)
        {
        if (to->type != from->type)
                {
```

```
128                 EVPerr(EVP_F_EVP_PKEY_COPY_PARAMETERS,EVP_R_DIFFERENT_KEY_TYPES)
129                 goto err;
130                 }

132         if (EVP_PKEY_missing_parameters(from))
133                 {
134                 EVPerr(EVP_F_EVP_PKEY_COPY_PARAMETERS,EVP_R_MISSING_PARAMETERS);
135                 goto err;
136                 }
137         if (from->ameth && from->ameth->param_copy)
138                 return from->ameth->param_copy(to, from);
139 err:
140         return 0;
141         }

143 int EVP_PKEY_missing_parameters(const EVP_PKEY *pkey)
144         {
145         if (pkey->ameth && pkey->ameth->param_missing)
146                 return pkey->ameth->param_missing(pkey);
147         return 0;
148         }

150 int EVP_PKEY_cmp_parameters(const EVP_PKEY *a, const EVP_PKEY *b)
151         {
152         if (a->type != b->type)
153                 return -1;
154         if (a->ameth && a->ameth->param_cmp)
155                 return a->ameth->param_cmp(a, b);
156         return -2;
157         }

159 int EVP_PKEY_cmp(const EVP_PKEY *a, const EVP_PKEY *b)
160         {
161         if (a->type != b->type)
162                 return -1;

164         if (a->ameth)
165                 {
166                 int ret;
167                 /* Compare parameters if the algorithm has them */
168                 if (a->ameth->param_cmp)
169                         {
170                         ret = a->ameth->param_cmp(a, b);
171                         if (ret <= 0)
172                                 return ret;
173                         }

175                 if (a->ameth->pub_cmp)
176                         return a->ameth->pub_cmp(a, b);
177                 }

179         return -2;
180         }

182 EVP_PKEY *EVP_PKEY_new(void)
183         {
184         EVP_PKEY *ret;

186         ret=(EVP_PKEY *)OPENSSL_malloc(sizeof(EVP_PKEY));
187         if (ret == NULL)
188                 {
189                 EVPerr(EVP_F_EVP_PKEY_NEW,ERR_R_MALLOC_FAILURE);
190                 return(NULL);
191                 }
192         ret->type=EVP_PKEY_NONE;
193         ret->save_type=EVP_PKEY_NONE;
```



```
194         ret->references=1;
195         ret->ameth=NULL;
196         ret->engine=NULL;
197         ret->pkey.ptr=NULL;
198         ret->attributes=NULL;
199         ret->save_parameters=1;
200         return(ret);
201         }

203 /* Setup a public key ASN1 method and ENGINE from a NID or a string.
204  * If pkey is NULL just return 1 or 0 if the algorithm exists.
205  */

207 static int pkey_set_type(EVP_PKEY *pkey, int type, const char *str, int len)
208         {
209         const EVP_PKEY_ASN1_METHOD *ameth;
210         ENGINE *e = NULL;
211         if (pkey)
212                 {
213                 if (pkey->pkey.ptr)
214                         EVP_PKEY_free_it(pkey);
215                 /* If key type matches and a method exists then this
216                  * lookup has succeeded once so just indicate success.
217                  */
218                 if ((type == pkey->save_type) && pkey->ameth)
219                         return 1;
220 #ifndef OPENSSL_NO_ENGINE
221                 /* If we have an ENGINE release it */
222                 if (pkey->engine)
223                         {
224                         ENGINE_finish(pkey->engine);
225                         pkey->engine = NULL;
226                         }
227 #endif
228                 }
229         if (str)
230                 ameth = EVP_PKEY_asn1_find_str(&e, str, len);
231         else
232                 ameth = EVP_PKEY_asn1_find(&e, type);
233 #ifndef OPENSSL_NO_ENGINE
234         if (!pkey && e)
235                 ENGINE_finish(e);
236 #endif
237         if (!ameth)
238                 {
239                 EVPerr(EVP_F_PKEY_SET_TYPE, EVP_R_UNSUPPORTED_ALGORITHM);
240                 return 0;
241                 }
242         if (pkey)
243                 {
244                 pkey->ameth = ameth;
245                 pkey->engine = e;

247                 pkey->type = pkey->ameth->pkey_id;
248                 pkey->save_type=type;
249                 }
250         return 1;
251         }

253 int EVP_PKEY_set_type(EVP_PKEY *pkey, int type)
254         {
255         return pkey_set_type(pkey, type, NULL, -1);
256         }

258 int EVP_PKEY_set_type_str(EVP_PKEY *pkey, const char *str, int len)
259         {
```

```
260         return pkey_set_type(pkey, EVP_PKEY_NONE, str, len);
261         }

263 int EVP_PKEY_assign(EVP_PKEY *pkey, int type, void *key)
264         {
265         if (!EVP_PKEY_set_type(pkey, type))
266                 return 0;
267         pkey->pkey.ptr=key;
268         return (key != NULL);
269         }

271 void *EVP_PKEY_get0(EVP_PKEY *pkey)
272         {
273         return pkey->pkey.ptr;
274         }

276 #ifndef OPENSSL_NO_RSA
277 int EVP_PKEY_set1_RSA(EVP_PKEY *pkey, RSA *key)
278 {
279         int ret = EVP_PKEY_assign_RSA(pkey, key);
280         if(ret)
281                 RSA_up_ref(key);
282         return ret;
283 }

285 RSA *EVP_PKEY_get1_RSA(EVP_PKEY *pkey)
286         {
287         if(pkey->type != EVP_PKEY_RSA) {
288                 EVPerr(EVP_F_EVP_PKEY_GET1_RSA, EVP_R_EXPECTING_AN_RSA_KEY);
289                 return NULL;
290         }
291         RSA_up_ref(pkey->pkey.rsa);
292         return pkey->pkey.rsa;
293 }
294 #endif

296 #ifndef OPENSSL_NO_DSA
297 int EVP_PKEY_set1_DSA(EVP_PKEY *pkey, DSA *key)
298 {
299         int ret = EVP_PKEY_assign_DSA(pkey, key);
300         if(ret)
301                 DSA_up_ref(key);
302         return ret;
303 }

305 DSA *EVP_PKEY_get1_DSA(EVP_PKEY *pkey)
306         {
307         if(pkey->type != EVP_PKEY_DSA) {
308                 EVPerr(EVP_F_EVP_PKEY_GET1_DSA, EVP_R_EXPECTING_A_DSA_KEY);
309                 return NULL;
310         }
311         DSA_up_ref(pkey->pkey.dsa);
312         return pkey->pkey.dsa;
313 }
314 #endif

316 #ifndef OPENSSL_NO_EC

318 int EVP_PKEY_set1_EC_KEY(EVP_PKEY *pkey, EC_KEY *key)
319 {
320         int ret = EVP_PKEY_assign_EC_KEY(pkey,key);
321         if (ret)
322                 EC_KEY_up_ref(key);
323         return ret;
324 }
```



```
326 EC_KEY *EVP_PKEY_get1_EC_KEY(EVP_PKEY *pkey)
327 {
328         if (pkey->type != EVP_PKEY_EC)
329         {
330                 EVPerr(EVP_F_EVP_PKEY_GET1_EC_KEY, EVP_R_EXPECTING_A_EC_KEY);
331                 return NULL;
332         }
333         EC_KEY_up_ref(pkey->pkey.ec);
334         return pkey->pkey.ec;
335 }
336 #endif


339 #ifndef OPENSSL_NO_DH

341 int EVP_PKEY_set1_DH(EVP_PKEY *pkey, DH *key)
342 {
343         int ret = EVP_PKEY_assign_DH(pkey, key);
344         if(ret)
345                 DH_up_ref(key);
346         return ret;
347 }

349 DH *EVP_PKEY_get1_DH(EVP_PKEY *pkey)
350         {
351         if(pkey->type != EVP_PKEY_DH) {
352                 EVPerr(EVP_F_EVP_PKEY_GET1_DH, EVP_R_EXPECTING_A_DH_KEY);
353                 return NULL;
354         }
355         DH_up_ref(pkey->pkey.dh);
356         return pkey->pkey.dh;
357 }
358 #endif

360 int EVP_PKEY_type(int type)
361         {
362         int ret;
363         const EVP_PKEY_ASN1_METHOD *ameth;
364         ENGINE *e;
365         ameth = EVP_PKEY_asn1_find(&e, type);
366         if (ameth)
367                 ret = ameth->pkey_id;
368         else
369                 ret = NID_undef;
370 #ifndef OPENSSL_NO_ENGINE
371         if (e)
372                 ENGINE_finish(e);
373 #endif
374         return ret;
375         }

377 int EVP_PKEY_id(const EVP_PKEY *pkey)
378         {
379         return pkey->type;
380         }

382 int EVP_PKEY_base_id(const EVP_PKEY *pkey)
383         {
384         return EVP_PKEY_type(pkey->type);
385         }

387 void EVP_PKEY_free(EVP_PKEY *x)
388         {
389         int i;

391         if (x == NULL) return;
```

```
393         i=CRYPTO_add(&x->references,-1,CRYPTO_LOCK_EVP_PKEY);
394 #ifdef REF_PRINT
395         REF_PRINT("EVP_PKEY",x);
396 #endif
397         if (i > 0) return;
398 #ifdef REF_CHECK
399         if (i < 0)
400                 {
401                 fprintf(stderr,"EVP_PKEY_free, bad reference count\n");
402                 abort();
403                 }
404 #endif
405         EVP_PKEY_free_it(x);
406         if (x->attributes)
407                 sk_X509_ATTRIBUTE_pop_free(x->attributes, X509_ATTRIBUTE_free);
408         OPENSSL_free(x);
409         }

411 static void EVP_PKEY_free_it(EVP_PKEY *x)
412         {
413         if (x->ameth && x->ameth->pkey_free)
414                 {
415                 x->ameth->pkey_free(x);
416                 x->pkey.ptr = NULL;
417                 }
418 #ifndef OPENSSL_NO_ENGINE
419         if (x->engine)
420                 {
421                 ENGINE_finish(x->engine);
422                 x->engine = NULL;
423                 }
424 #endif
425         }

427 static int unsup_alg(BIO *out, const EVP_PKEY *pkey, int indent,
428                              const char *kstr)
429         {
430         BIO_indent(out, indent, 128);
431         BIO_printf(out, "%s algorithm \"%s\" unsupported\n",
432                                                 kstr, OBJ_nid2ln(pkey->type));
433         return 1;
434         }

436 int EVP_PKEY_print_public(BIO *out, const EVP_PKEY *pkey,
437                              int indent, ASN1_PCTX *pctx)
438         {
439         if (pkey->ameth && pkey->ameth->pub_print)
440                 return pkey->ameth->pub_print(out, pkey, indent, pctx);

442         return unsup_alg(out, pkey, indent, "Public Key");
443         }

445 int EVP_PKEY_print_private(BIO *out, const EVP_PKEY *pkey,
446                              int indent, ASN1_PCTX *pctx)
447         {
448         if (pkey->ameth && pkey->ameth->priv_print)
449                 return pkey->ameth->priv_print(out, pkey, indent, pctx);

451         return unsup_alg(out, pkey, indent, "Private Key");
452         }

454 int EVP_PKEY_print_params(BIO *out, const EVP_PKEY *pkey,
455                              int indent, ASN1_PCTX *pctx)
456         {
457         if (pkey->ameth && pkey->ameth->param_print)
```



```
458                 return pkey->ameth->param_print(out, pkey, indent, pctx);
459         return unsup_alg(out, pkey, indent, "Parameters");
460         }

462 int EVP_PKEY_get_default_digest_nid(EVP_PKEY *pkey, int *pnid)
463         {
464         if (!pkey->ameth || !pkey->ameth->pkey_ctrl)
465                 return -2;
466         return pkey->ameth->pkey_ctrl(pkey, ASN1_PKEY_CTRL_DEFAULT_MD_NID,
467                                                 0, pnid);
468         }
469 #endif /* ! codereview */
```

```
**********************************************************
    4555 Wed Aug 13 19:52:50 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p_open.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/p_open.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
```



```
#ifndef OPENSSL_NO_RSA

#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/rsa.h>

int EVP_OpenInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type,
        const unsigned char *ek, int ekl, const unsigned char *iv,
        EVP_PKEY *priv)
        {
        unsigned char *key=NULL;
        int i,size=0,ret=0;

        if(type) {
                EVP_CIPHER_CTX_init(ctx);
                if(!EVP_DecryptInit_ex(ctx,type,NULL, NULL,NULL)) return 0;
        }

        if(!priv) return 1;

        if (priv->type != EVP_PKEY_RSA)
                {
                EVPerr(EVP_F_EVP_OPENINIT,EVP_R_PUBLIC_KEY_NOT_RSA);
                goto err;
                }

        size=RSA_size(priv->pkey.rsa);
        key=(unsigned char *)OPENSSL_malloc(size+2);
        if (key == NULL)
                {
                /* ERROR */
                EVPerr(EVP_F_EVP_OPENINIT,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        i=EVP_PKEY_decrypt_old(key,ek,ekl,priv);
        if ((i <= 0) || !EVP_CIPHER_CTX_set_key_length(ctx, i))
                {
                /* ERROR */
                goto err;
                }
        if(!EVP_DecryptInit_ex(ctx,NULL,NULL,key,iv)) goto err;

        ret=1;
err:
        if (key != NULL) OPENSSL_cleanse(key,size);
        OPENSSL_free(key);
        return(ret);
        }

int EVP_OpenFinal(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
        {
        int i;

        i=EVP_DecryptFinal_ex(ctx,out,outl);
        if (i)
                i = EVP_DecryptInit_ex(ctx,NULL,NULL,NULL,NULL);
        return(i);
        }
#else /* !OPENSSL_NO_RSA */

# ifdef PEDANTIC
static void *dummy=&dummy;
# endif
```

```
128 #endif
129 #endif /* ! codereview */
```

```
**********************************************************
    4485 Wed Aug 13 19:52:50 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p_seal.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/p_seal.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/rand.h>
```



```
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>

int EVP_SealInit(EVP_CIPHER_CTX *ctx, const EVP_CIPHER *type, unsigned char **ek
             int *ekl, unsigned char *iv, EVP_PKEY **pubk, int npubk)
        {
        unsigned char key[EVP_MAX_KEY_LENGTH];
        int i;

        if(type) {
                EVP_CIPHER_CTX_init(ctx);
                if(!EVP_EncryptInit_ex(ctx,type,NULL,NULL,NULL)) return 0;
        }
        if ((npubk <= 0) || !pubk)
                return 1;
        if (EVP_CIPHER_CTX_rand_key(ctx, key) <= 0)
                return 0;
        if (EVP_CIPHER_CTX_iv_length(ctx))
                RAND_pseudo_bytes(iv,EVP_CIPHER_CTX_iv_length(ctx));

        if(!EVP_EncryptInit_ex(ctx,NULL,NULL,key,iv)) return 0;

        for (i=0; i<npubk; i++)
                {
                ekl[i]=EVP_PKEY_encrypt_old(ek[i],key,EVP_CIPHER_CTX_key_length(
                        pubk[i]);
                if (ekl[i] <= 0) return(-1);
                }
        return(npubk);
        }

/* MACRO
void EVP_SealUpdate(ctx,out,outl,in,inl)
EVP_CIPHER_CTX *ctx;
unsigned char *out;
int *outl;
unsigned char *in;
int inl;
        {
        EVP_EncryptUpdate(ctx,out,outl,in,inl);
        }
*/

int EVP_SealFinal(EVP_CIPHER_CTX *ctx, unsigned char *out, int *outl)
        {
        int i;
        i = EVP_EncryptFinal_ex(ctx,out,outl);
        if (i)
                i = EVP_EncryptInit_ex(ctx,NULL,NULL,NULL,NULL);
        return i;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4870 Wed Aug 13 19:52:50 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p_sign.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/p_sign.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>

#ifdef undef
void EVP_SignInit(EVP_MD_CTX *ctx, EVP_MD *type)
        {
        EVP_DigestInit_ex(ctx,type);
        }

void EVP_SignUpdate(EVP_MD_CTX *ctx, unsigned char *data,
             unsigned int count)
        {
        EVP_DigestUpdate(ctx,data,count);
        }
#endif

int EVP_SignFinal(EVP_MD_CTX *ctx, unsigned char *sigret, unsigned int *siglen,
             EVP_PKEY *pkey)
        {
        unsigned char m[EVP_MAX_MD_SIZE];
        unsigned int m_len;
        int i = 0,ok = 0,v;
        EVP_MD_CTX tmp_ctx;
        EVP_PKEY_CTX *pkctx = NULL;

        *siglen=0;
        EVP_MD_CTX_init(&tmp_ctx);
        if (!EVP_MD_CTX_copy_ex(&tmp_ctx,ctx))
                goto err;
        if (!EVP_DigestFinal_ex(&tmp_ctx,&(m[0]),&m_len))
                goto err;
        EVP_MD_CTX_cleanup(&tmp_ctx);

        if (ctx->digest->flags & EVP_MD_FLAG_PKEY_METHOD_SIGNATURE)
                {
                size_t sltmp = (size_t)EVP_PKEY_size(pkey);
                i = 0;
                pkctx = EVP_PKEY_CTX_new(pkey, NULL);
                if (!pkctx)
                        goto err;
                if (EVP_PKEY_sign_init(pkctx) <= 0)
                        goto err;
                if (EVP_PKEY_CTX_set_signature_md(pkctx, ctx->digest) <= 0)
                        goto err;
                if (EVP_PKEY_sign(pkctx, sigret, &sltmp, m, m_len) <= 0)
                        goto err;
                *siglen = sltmp;
                i = 1;
                err:
                EVP_PKEY_CTX_free(pkctx);
                return i;
                }

        for (i=0; i<4; i++)
                {
                v=ctx->digest->required_pkey_type[i];
                if (v == 0) break;
                if (pkey->type == v)
                        {
                        ok=1;
                        break;
                        }
                }
        if (!ok)
                {
                EVPerr(EVP_F_EVP_SIGNFINAL,EVP_R_WRONG_PUBLIC_KEY_TYPE);
```

```
128                 return(0);
129                 }

131         if (ctx->digest->sign == NULL)
132                 {
133                 EVPerr(EVP_F_EVP_SIGNFINAL,EVP_R_NO_SIGN_FUNCTION_CONFIGURED);
134                 return(0);
135                 }
136         return(ctx->digest->sign(ctx->digest->type,m,m_len,sigret,siglen,
137                 pkey->pkey.ptr));
138         }
139 #endif /* ! codereview */
```

```
**********************************************************
    4580 Wed Aug 13 19:52:50 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/p_verify.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/evp/p_verify.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>

int EVP_VerifyFinal(EVP_MD_CTX *ctx, const unsigned char *sigbuf,
             unsigned int siglen, EVP_PKEY *pkey)
        {
        unsigned char m[EVP_MAX_MD_SIZE];
        unsigned int m_len;
        int i = 0,ok = 0,v;
        EVP_MD_CTX tmp_ctx;
        EVP_PKEY_CTX *pkctx = NULL;

        EVP_MD_CTX_init(&tmp_ctx);
        if (!EVP_MD_CTX_copy_ex(&tmp_ctx,ctx))
                goto err;
        if (!EVP_DigestFinal_ex(&tmp_ctx,&(m[0]),&m_len))
                goto err;
        EVP_MD_CTX_cleanup(&tmp_ctx);

        if (ctx->digest->flags & EVP_MD_FLAG_PKEY_METHOD_SIGNATURE)
                {
                i = -1;
                pkctx = EVP_PKEY_CTX_new(pkey, NULL);
                if (!pkctx)
                        goto err;
                if (EVP_PKEY_verify_init(pkctx) <= 0)
                        goto err;
                if (EVP_PKEY_CTX_set_signature_md(pkctx, ctx->digest) <= 0)
                        goto err;
                i = EVP_PKEY_verify(pkctx, sigbuf, siglen, m, m_len);
                err:
                EVP_PKEY_CTX_free(pkctx);
                return i;
                }

        for (i=0; i<4; i++)
                {
                v=ctx->digest->required_pkey_type[i];
                if (v == 0) break;
                if (pkey->type == v)
                        {
                        ok=1;
                        break;
                        }
                }
        if (!ok)
                {
                EVPerr(EVP_F_EVP_VERIFYFINAL,EVP_R_WRONG_PUBLIC_KEY_TYPE);
                return(-1);
                }
        if (ctx->digest->verify == NULL)
                {
                EVPerr(EVP_F_EVP_VERIFYFINAL,EVP_R_NO_VERIFY_FUNCTION_CONFIGURED
                return(0);
                }

        return(ctx->digest->verify(ctx->digest->type,m,m_len,
                sigbuf,siglen,pkey->pkey.ptr));
        }
#endif /* ! codereview */
```

```
**********************************************************
   10544 Wed Aug 13 19:52:50 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/pmeth_fn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pmeth_fn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include "cryptlib.h"
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include "evp_locl.h"

#define M_check_autoarg(ctx, arg, arglen, err) \
        if (ctx->pmeth->flags & EVP_PKEY_FLAG_AUTOARGLEN) \
                { \
                size_t pksize = (size_t)EVP_PKEY_size(ctx->pkey); \
                if (!arg) \
                        { \
                        *arglen = pksize; \
                        return 1; \
                        } \
                else if (*arglen < pksize) \
                        { \
                        EVPerr(err, EVP_R_BUFFER_TOO_SMALL); /*ckerr_ignore*/\
                        return 0; \
                        } \
                }

int EVP_PKEY_sign_init(EVP_PKEY_CTX *ctx)
        {
        int ret;
        if (!ctx || !ctx->pmeth || !ctx->pmeth->sign)
                {
                EVPerr(EVP_F_EVP_PKEY_SIGN_INIT,
                        EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
                return -2;
                }
        ctx->operation = EVP_PKEY_OP_SIGN;
        if (!ctx->pmeth->sign_init)
                return 1;
        ret = ctx->pmeth->sign_init(ctx);
        if (ret <= 0)
                ctx->operation = EVP_PKEY_OP_UNDEFINED;
        return ret;
        }

int EVP_PKEY_sign(EVP_PKEY_CTX *ctx,
                        unsigned char *sig, size_t *siglen,
                        const unsigned char *tbs, size_t tbslen)
        {
        if (!ctx || !ctx->pmeth || !ctx->pmeth->sign)
                {
                EVPerr(EVP_F_EVP_PKEY_SIGN,
                        EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
                return -2;
                }
        if (ctx->operation != EVP_PKEY_OP_SIGN)
                {
                EVPerr(EVP_F_EVP_PKEY_SIGN, EVP_R_OPERATON_NOT_INITIALIZED);
                return -1;
                }
        M_check_autoarg(ctx, sig, siglen, EVP_F_EVP_PKEY_SIGN)
        return ctx->pmeth->sign(ctx, sig, siglen, tbs, tbslen);
        }

int EVP_PKEY_verify_init(EVP_PKEY_CTX *ctx)
        {
        int ret;
        if (!ctx || !ctx->pmeth || !ctx->pmeth->verify)
                {
                EVPerr(EVP_F_EVP_PKEY_VERIFY_INIT,
                        EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
                return -2;
                }
```

```
128         ctx->operation = EVP_PKEY_OP_VERIFY;
129         if (!ctx->pmeth->verify_init)
130                 return 1;
131         ret = ctx->pmeth->verify_init(ctx);
132         if (ret <= 0)
133                 ctx->operation = EVP_PKEY_OP_UNDEFINED;
134         return ret;
135         }

137 int EVP_PKEY_verify(EVP_PKEY_CTX *ctx,
138                         const unsigned char *sig, size_t siglen,
139                         const unsigned char *tbs, size_t tbslen)
140         {
141         if (!ctx || !ctx->pmeth || !ctx->pmeth->verify)
142                 {
143                 EVPerr(EVP_F_EVP_PKEY_VERIFY,
144                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
145                 return -2;
146                 }
147         if (ctx->operation != EVP_PKEY_OP_VERIFY)
148                 {
149                 EVPerr(EVP_F_EVP_PKEY_VERIFY, EVP_R_OPERATON_NOT_INITIALIZED);
150                 return -1;
151                 }
152         return ctx->pmeth->verify(ctx, sig, siglen, tbs, tbslen);
153         }

155 int EVP_PKEY_verify_recover_init(EVP_PKEY_CTX *ctx)
156         {
157         int ret;
158         if (!ctx || !ctx->pmeth || !ctx->pmeth->verify_recover)
159                 {
160                 EVPerr(EVP_F_EVP_PKEY_VERIFY_RECOVER_INIT,
161                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
162                 return -2;
163                 }
164         ctx->operation = EVP_PKEY_OP_VERIFYRECOVER;
165         if (!ctx->pmeth->verify_recover_init)
166                 return 1;
167         ret = ctx->pmeth->verify_recover_init(ctx);
168         if (ret <= 0)
169                 ctx->operation = EVP_PKEY_OP_UNDEFINED;
170         return ret;
171         }

173 int EVP_PKEY_verify_recover(EVP_PKEY_CTX *ctx,
174                         unsigned char *rout, size_t *routlen,
175                         const unsigned char *sig, size_t siglen)
176         {
177         if (!ctx || !ctx->pmeth || !ctx->pmeth->verify_recover)
178                 {
179                 EVPerr(EVP_F_EVP_PKEY_VERIFY_RECOVER,
180                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
181                 return -2;
182                 }
183         if (ctx->operation != EVP_PKEY_OP_VERIFYRECOVER)
184                 {
185                 EVPerr(EVP_F_EVP_PKEY_VERIFY_RECOVER, EVP_R_OPERATON_NOT_INITIAL
186                 return -1;
187                 }
188         M_check_autoarg(ctx, rout, routlen, EVP_F_EVP_PKEY_VERIFY_RECOVER)
189         return ctx->pmeth->verify_recover(ctx, rout, routlen, sig, siglen);
190         }

192 int EVP_PKEY_encrypt_init(EVP_PKEY_CTX *ctx)
193         {
```



```
194         int ret;
195         if (!ctx || !ctx->pmeth || !ctx->pmeth->encrypt)
196                 {
197                 EVPerr(EVP_F_EVP_PKEY_ENCRYPT_INIT,
198                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
199                 return -2;
200                 }
201         ctx->operation = EVP_PKEY_OP_ENCRYPT;
202         if (!ctx->pmeth->encrypt_init)
203                 return 1;
204         ret = ctx->pmeth->encrypt_init(ctx);
205         if (ret <= 0)
206                 ctx->operation = EVP_PKEY_OP_UNDEFINED;
207         return ret;
208         }

210 int EVP_PKEY_encrypt(EVP_PKEY_CTX *ctx,
211                         unsigned char *out, size_t *outlen,
212                         const unsigned char *in, size_t inlen)
213         {
214         if (!ctx || !ctx->pmeth || !ctx->pmeth->encrypt)
215                 {
216                 EVPerr(EVP_F_EVP_PKEY_ENCRYPT,
217                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
218                 return -2;
219                 }
220         if (ctx->operation != EVP_PKEY_OP_ENCRYPT)
221                 {
222                 EVPerr(EVP_F_EVP_PKEY_ENCRYPT, EVP_R_OPERATON_NOT_INITIALIZED);
223                 return -1;
224                 }
225         M_check_autoarg(ctx, out, outlen, EVP_F_EVP_PKEY_ENCRYPT)
226         return ctx->pmeth->encrypt(ctx, out, outlen, in, inlen);
227         }

229 int EVP_PKEY_decrypt_init(EVP_PKEY_CTX *ctx)
230         {
231         int ret;
232         if (!ctx || !ctx->pmeth || !ctx->pmeth->decrypt)
233                 {
234                 EVPerr(EVP_F_EVP_PKEY_DECRYPT_INIT,
235                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
236                 return -2;
237                 }
238         ctx->operation = EVP_PKEY_OP_DECRYPT;
239         if (!ctx->pmeth->decrypt_init)
240                 return 1;
241         ret = ctx->pmeth->decrypt_init(ctx);
242         if (ret <= 0)
243                 ctx->operation = EVP_PKEY_OP_UNDEFINED;
244         return ret;
245         }

247 int EVP_PKEY_decrypt(EVP_PKEY_CTX *ctx,
248                         unsigned char *out, size_t *outlen,
249                         const unsigned char *in, size_t inlen)
250         {
251         if (!ctx || !ctx->pmeth || !ctx->pmeth->decrypt)
252                 {
253                 EVPerr(EVP_F_EVP_PKEY_DECRYPT,
254                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
255                 return -2;
256                 }
257         if (ctx->operation != EVP_PKEY_OP_DECRYPT)
258                 {
259                 EVPerr(EVP_F_EVP_PKEY_DECRYPT, EVP_R_OPERATON_NOT_INITIALIZED);
```

```
260                 return -1;
261                 }
262         M_check_autoarg(ctx, out, outlen, EVP_F_EVP_PKEY_DECRYPT)
263         return ctx->pmeth->decrypt(ctx, out, outlen, in, inlen);
264         }


267 int EVP_PKEY_derive_init(EVP_PKEY_CTX *ctx)
268         {
269         int ret;
270         if (!ctx || !ctx->pmeth || !ctx->pmeth->derive)
271                 {
272                 EVPerr(EVP_F_EVP_PKEY_DERIVE_INIT,
273                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
274                 return -2;
275                 }
276         ctx->operation = EVP_PKEY_OP_DERIVE;
277         if (!ctx->pmeth->derive_init)
278                 return 1;
279         ret = ctx->pmeth->derive_init(ctx);
280         if (ret <= 0)
281                 ctx->operation = EVP_PKEY_OP_UNDEFINED;
282         return ret;
283         }

285 int EVP_PKEY_derive_set_peer(EVP_PKEY_CTX *ctx, EVP_PKEY *peer)
286         {
287         int ret;
288         if (!ctx || !ctx->pmeth || !(ctx->pmeth->derive||ctx->pmeth->encrypt||ct
289                 {
290                 EVPerr(EVP_F_EVP_PKEY_DERIVE_SET_PEER,
291                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
292                 return -2;
293                 }
294         if (ctx->operation != EVP_PKEY_OP_DERIVE && ctx->operation != EVP_PKEY_O
295                 {
296                 EVPerr(EVP_F_EVP_PKEY_DERIVE_SET_PEER,
297                                         EVP_R_OPERATON_NOT_INITIALIZED);
298                 return -1;
299                 }

301         ret = ctx->pmeth->ctrl(ctx, EVP_PKEY_CTRL_PEER_KEY, 0, peer);

303         if (ret <= 0)
304                 return ret;

306         if (ret == 2)
307                 return 1;

309         if (!ctx->pkey)
310                 {
311                 EVPerr(EVP_F_EVP_PKEY_DERIVE_SET_PEER, EVP_R_NO_KEY_SET);
312                 return -1;
313                 }

315         if (ctx->pkey->type != peer->type)
316                 {
317                 EVPerr(EVP_F_EVP_PKEY_DERIVE_SET_PEER,
318                                                 EVP_R_DIFFERENT_KEY_TYPES);
319                 return -1;
320                 }

322         /* ran@cryptocom.ru: For clarity.  The error is if parameters in peer ar
323          * present (!missing) but don't match.  EVP_PKEY_cmp_parameters may retu
324          * 1 (match), 0 (don't match) and -2 (comparison is not defined).  -1
325          * (different key types) is impossible here because it is checked earlie
```



```
326          * -2 is OK for us here, as well as 1, so we can check for 0 only. */
327         if (!EVP_PKEY_missing_parameters(peer) &&
328                 !EVP_PKEY_cmp_parameters(ctx->pkey, peer))
329                 {
330                 EVPerr(EVP_F_EVP_PKEY_DERIVE_SET_PEER,
331                                                 EVP_R_DIFFERENT_PARAMETERS);
332                 return -1;
333                 }

335         if (ctx->peerkey)
336                 EVP_PKEY_free(ctx->peerkey);
337         ctx->peerkey = peer;

339         ret = ctx->pmeth->ctrl(ctx, EVP_PKEY_CTRL_PEER_KEY, 1, peer);

341         if (ret <= 0)
342                 {
343                 ctx->peerkey = NULL;
344                 return ret;
345                 }

347         CRYPTO_add(&peer->references,1,CRYPTO_LOCK_EVP_PKEY);
348         return 1;
349         }


352 int EVP_PKEY_derive(EVP_PKEY_CTX *ctx, unsigned char *key, size_t *pkeylen)
353         {
354         if (!ctx || !ctx->pmeth || !ctx->pmeth->derive)
355                 {
356                 EVPerr(EVP_F_EVP_PKEY_DERIVE,
357                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
358                 return -2;
359                 }
360         if (ctx->operation != EVP_PKEY_OP_DERIVE)
361                 {
362                 EVPerr(EVP_F_EVP_PKEY_DERIVE, EVP_R_OPERATON_NOT_INITIALIZED);
363                 return -1;
364                 }
365         M_check_autoarg(ctx, key, pkeylen, EVP_F_EVP_PKEY_DERIVE)
366         return ctx->pmeth->derive(ctx, key, pkeylen);
367         }
368 #endif /* ! codereview */
```

```
**********************************************************
    6128 Wed Aug 13 19:52:51 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/pmeth_gn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pmeth_gn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include "cryptlib.h"
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/bn.h>
#include "evp_locl.h"

int EVP_PKEY_paramgen_init(EVP_PKEY_CTX *ctx)
        {
        int ret;
        if (!ctx || !ctx->pmeth || !ctx->pmeth->paramgen)
                {
                EVPerr(EVP_F_EVP_PKEY_PARAMGEN_INIT,
                        EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
                return -2;
                }
        ctx->operation = EVP_PKEY_OP_PARAMGEN;
        if (!ctx->pmeth->paramgen_init)
                return 1;
        ret = ctx->pmeth->paramgen_init(ctx);
        if (ret <= 0)
                ctx->operation = EVP_PKEY_OP_UNDEFINED;
        return ret;
        }

int EVP_PKEY_paramgen(EVP_PKEY_CTX *ctx, EVP_PKEY **ppkey)
        {
        int ret;
        if (!ctx || !ctx->pmeth || !ctx->pmeth->paramgen)
                {
                EVPerr(EVP_F_EVP_PKEY_PARAMGEN,
                        EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
                return -2;
                }

        if (ctx->operation != EVP_PKEY_OP_PARAMGEN)
                {
                EVPerr(EVP_F_EVP_PKEY_PARAMGEN, EVP_R_OPERATON_NOT_INITIALIZED);
                return -1;
                }

        if (!ppkey)
                return -1;

        if (!*ppkey)
                *ppkey = EVP_PKEY_new();

        ret = ctx->pmeth->paramgen(ctx, *ppkey);
        if (ret <= 0)
                {
                EVP_PKEY_free(*ppkey);
                *ppkey = NULL;
                }
        return ret;
        }

int EVP_PKEY_keygen_init(EVP_PKEY_CTX *ctx)
        {
        int ret;
        if (!ctx || !ctx->pmeth || !ctx->pmeth->keygen)
                {
                EVPerr(EVP_F_EVP_PKEY_KEYGEN_INIT,
                        EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
                return -2;
                }
        ctx->operation = EVP_PKEY_OP_KEYGEN;
        if (!ctx->pmeth->keygen_init)
                return 1;
```

```
128         ret = ctx->pmeth->keygen_init(ctx);
129         if (ret <= 0)
130                 ctx->operation = EVP_PKEY_OP_UNDEFINED;
131         return ret;
132         }

134 int EVP_PKEY_keygen(EVP_PKEY_CTX *ctx, EVP_PKEY **ppkey)
135         {
136         int ret;

138         if (!ctx || !ctx->pmeth || !ctx->pmeth->keygen)
139                 {
140                 EVPerr(EVP_F_EVP_PKEY_KEYGEN,
141                         EVP_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
142                 return -2;
143                 }
144         if (ctx->operation != EVP_PKEY_OP_KEYGEN)
145                 {
146                 EVPerr(EVP_F_EVP_PKEY_KEYGEN, EVP_R_OPERATON_NOT_INITIALIZED);
147                 return -1;
148                 }

150         if (!ppkey)
151                 return -1;

153         if (!*ppkey)
154                 *ppkey = EVP_PKEY_new();

156         ret = ctx->pmeth->keygen(ctx, *ppkey);
157         if (ret <= 0)
158                 {
159                 EVP_PKEY_free(*ppkey);
160                 *ppkey = NULL;
161                 }
162         return ret;
163         }

165 void EVP_PKEY_CTX_set_cb(EVP_PKEY_CTX *ctx, EVP_PKEY_gen_cb *cb)
166         {
167         ctx->pkey_gencb = cb;
168         }

170 EVP_PKEY_gen_cb *EVP_PKEY_CTX_get_cb(EVP_PKEY_CTX *ctx)
171         {
172         return ctx->pkey_gencb;
173         }

175 /* "translation callback" to call EVP_PKEY_CTX callbacks using BN_GENCB
176  * style callbacks.
177  */

179 static int trans_cb(int a, int b, BN_GENCB *gcb)
180         {
181         EVP_PKEY_CTX *ctx = gcb->arg;
182         ctx->keygen_info[0] = a;
183         ctx->keygen_info[1] = b;
184         return ctx->pkey_gencb(ctx);
185         }

187 void evp_pkey_set_cb_translate(BN_GENCB *cb, EVP_PKEY_CTX *ctx)
188         {
189         BN_GENCB_set(cb, trans_cb, ctx)
190         }

192 int EVP_PKEY_CTX_get_keygen_info(EVP_PKEY_CTX *ctx, int idx)
193         {
```



```
194         if (idx == -1)
195                 return ctx->keygen_info_count;
196         if (idx < 0 || idx > ctx->keygen_info_count)
197                 return 0;
198         return ctx->keygen_info[idx];
199         }

201 EVP_PKEY *EVP_PKEY_new_mac_key(int type, ENGINE *e,
202                                 const unsigned char *key, int keylen)
203         {
204         EVP_PKEY_CTX *mac_ctx = NULL;
205         EVP_PKEY *mac_key = NULL;
206         mac_ctx = EVP_PKEY_CTX_new_id(type, e);
207         if (!mac_ctx)
208                 return NULL;
209         if (EVP_PKEY_keygen_init(mac_ctx) <= 0)
210                 goto merr;
211         if (EVP_PKEY_CTX_ctrl(mac_ctx, -1, EVP_PKEY_OP_KEYGEN,
212                                 EVP_PKEY_CTRL_SET_MAC_KEY,
213                                 keylen, (void *)key) <= 0)
214                 goto merr;
215         if (EVP_PKEY_keygen(mac_ctx, &mac_key) <= 0)
216                 goto merr;
217         merr:
218         if (mac_ctx)
219                 EVP_PKEY_CTX_free(mac_ctx);
220         return mac_key;
221         }
222 #endif /* ! codereview */
```

```
**********************************************************
   14998 Wed Aug 13 19:52:51 2014
new/usr/src/lib/openssl/libsunw_crypto/evp/pmeth_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pmeth_lib.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include "cryptlib.h"
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include "asn1_locl.h"
#include "evp_locl.h"

typedef int sk_cmp_fn_type(const char * const *a, const char * const *b);

DECLARE_STACK_OF(EVP_PKEY_METHOD)
STACK_OF(EVP_PKEY_METHOD) *app_pkey_methods = NULL;

extern const EVP_PKEY_METHOD rsa_pkey_meth, dh_pkey_meth, dsa_pkey_meth;
extern const EVP_PKEY_METHOD ec_pkey_meth, hmac_pkey_meth, cmac_pkey_meth;

static const EVP_PKEY_METHOD *standard_methods[] =
        {
#ifndef OPENSSL_NO_RSA
        &rsa_pkey_meth,
#endif
#ifndef OPENSSL_NO_DH
        &dh_pkey_meth,
#endif
#ifndef OPENSSL_NO_DSA
        &dsa_pkey_meth,
#endif
#ifndef OPENSSL_NO_EC
        &ec_pkey_meth,
#endif
        &hmac_pkey_meth,
        &cmac_pkey_meth
        };

DECLARE_OBJ_BSEARCH_CMP_FN(const EVP_PKEY_METHOD *, const EVP_PKEY_METHOD *,
                           pmeth);

static int pmeth_cmp(const EVP_PKEY_METHOD * const *a,
                     const EVP_PKEY_METHOD * const *b)
        {
        return ((*a)->pkey_id - (*b)->pkey_id);
        }

IMPLEMENT_OBJ_BSEARCH_CMP_FN(const EVP_PKEY_METHOD *, const EVP_PKEY_METHOD *,
                             pmeth);

const EVP_PKEY_METHOD *EVP_PKEY_meth_find(int type)
        {
        EVP_PKEY_METHOD tmp;
        const EVP_PKEY_METHOD *t = &tmp, **ret;
        tmp.pkey_id = type;
        if (app_pkey_methods)
                {
                int idx;
                idx = sk_EVP_PKEY_METHOD_find(app_pkey_methods, &tmp);
                if (idx >= 0)
                        return sk_EVP_PKEY_METHOD_value(app_pkey_methods, idx);
                }
        ret = OBJ_bsearch_pmeth(&t, standard_methods,
                                sizeof(standard_methods)/sizeof(EVP_PKEY_METHOD *));
        if (!ret || !*ret)
                return NULL;
        return *ret;
        }

static EVP_PKEY_CTX *int_ctx_new(EVP_PKEY *pkey, ENGINE *e, int id)
```

```
128         {
129         EVP_PKEY_CTX *ret;
130         const EVP_PKEY_METHOD *pmeth;
131         if (id == -1)
132                 {
133                 if (!pkey || !pkey->ameth)
134                         return NULL;
135                 id = pkey->ameth->pkey_id;
136                 }
137 #ifndef OPENSSL_NO_ENGINE
138         if (pkey && pkey->engine)
139                 e = pkey->engine;
140         /* Try to find an ENGINE which implements this method */
141         if (e)
142                 {
143                 if (!ENGINE_init(e))
144                         {
145                         EVPerr(EVP_F_INT_CTX_NEW,ERR_R_ENGINE_LIB);
146                         return NULL;
147                         }
148                 }
149         else
150                 e = ENGINE_get_pkey_meth_engine(id);

152         /* If an ENGINE handled this method look it up. Othewise
153          * use internal tables.
154          */

156         if (e)
157                 pmeth = ENGINE_get_pkey_meth(e, id);
158         else
159 #endif
160                 pmeth = EVP_PKEY_meth_find(id);

162         if (pmeth == NULL)
163                 {
164                 EVPerr(EVP_F_INT_CTX_NEW,EVP_R_UNSUPPORTED_ALGORITHM);
165                 return NULL;
166                 }

168         ret = OPENSSL_malloc(sizeof(EVP_PKEY_CTX));
169         if (!ret)
170                 {
171 #ifndef OPENSSL_NO_ENGINE
172                 if (e)
173                         ENGINE_finish(e);
174 #endif
175                 EVPerr(EVP_F_INT_CTX_NEW,ERR_R_MALLOC_FAILURE);
176                 return NULL;
177                 }
178         ret->engine = e;
179         ret->pmeth = pmeth;
180         ret->operation = EVP_PKEY_OP_UNDEFINED;
181         ret->pkey = pkey;
182         ret->peerkey = NULL;
183         ret->pkey_gencb = 0;
184         if (pkey)
185                 CRYPTO_add(&pkey->references,1,CRYPTO_LOCK_EVP_PKEY);
186         ret->data = NULL;

188         if (pmeth->init)
189                 {
190                 if (pmeth->init(ret) <= 0)
191                         {
192                         EVP_PKEY_CTX_free(ret);
193                         return NULL;
```



```
194                         }
195                 }

197         return ret;
198         }

200 EVP_PKEY_METHOD* EVP_PKEY_meth_new(int id, int flags)
201         {
202         EVP_PKEY_METHOD *pmeth;
203         pmeth = OPENSSL_malloc(sizeof(EVP_PKEY_METHOD));
204         if (!pmeth)
205                 return NULL;

207         memset(pmeth, 0, sizeof(EVP_PKEY_METHOD));

209         pmeth->pkey_id = id;
210         pmeth->flags = flags | EVP_PKEY_FLAG_DYNAMIC;

212         pmeth->init = 0;
213         pmeth->copy = 0;
214         pmeth->cleanup = 0;
215         pmeth->paramgen_init = 0;
216         pmeth->paramgen = 0;
217         pmeth->keygen_init = 0;
218         pmeth->keygen = 0;
219         pmeth->sign_init = 0;
220         pmeth->sign = 0;
221         pmeth->verify_init = 0;
222         pmeth->verify = 0;
223         pmeth->verify_recover_init = 0;
224         pmeth->verify_recover = 0;
225         pmeth->signctx_init = 0;
226         pmeth->signctx = 0;
227         pmeth->verifyctx_init = 0;
228         pmeth->verifyctx = 0;
229         pmeth->encrypt_init = 0;
230         pmeth->encrypt = 0;
231         pmeth->decrypt_init = 0;
232         pmeth->decrypt = 0;
233         pmeth->derive_init = 0;
234         pmeth->derive = 0;
235         pmeth->ctrl = 0;
236         pmeth->ctrl_str = 0;

238         return pmeth;
239         }

241 void EVP_PKEY_meth_get0_info(int *ppkey_id, int *pflags,
242                                 const EVP_PKEY_METHOD *meth)
243         {
244         if (ppkey_id)
245                 *ppkey_id = meth->pkey_id;
246         if (pflags)
247                 *pflags = meth->flags;
248         }

250 void EVP_PKEY_meth_copy(EVP_PKEY_METHOD *dst, const EVP_PKEY_METHOD *src)
251         {

253         dst->init = src->init;
254         dst->copy = src->copy;
255         dst->cleanup = src->cleanup;

257         dst->paramgen_init = src->paramgen_init;
258         dst->paramgen = src->paramgen;
```

```
260         dst->keygen_init = src->keygen_init;
261         dst->keygen = src->keygen;

263         dst->sign_init = src->sign_init;
264         dst->sign = src->sign;

266         dst->verify_init = src->verify_init;
267         dst->verify = src->verify;

269         dst->verify_recover_init = src->verify_recover_init;
270         dst->verify_recover = src->verify_recover;

272         dst->signctx_init = src->signctx_init;
273         dst->signctx = src->signctx;

275         dst->verifyctx_init = src->verifyctx_init;
276         dst->verifyctx = src->verifyctx;

278         dst->encrypt_init = src->encrypt_init;
279         dst->encrypt = src->encrypt;

281         dst->decrypt_init = src->decrypt_init;
282         dst->decrypt = src->decrypt;

284         dst->derive_init = src->derive_init;
285         dst->derive = src->derive;

287         dst->ctrl = src->ctrl;
288         dst->ctrl_str = src->ctrl_str;
289         }

291 void EVP_PKEY_meth_free(EVP_PKEY_METHOD *pmeth)
292         {
293         if (pmeth && (pmeth->flags & EVP_PKEY_FLAG_DYNAMIC))
294                 OPENSSL_free(pmeth);
295         }

297 EVP_PKEY_CTX *EVP_PKEY_CTX_new(EVP_PKEY *pkey, ENGINE *e)
298         {
299         return int_ctx_new(pkey, e, -1);
300         }

302 EVP_PKEY_CTX *EVP_PKEY_CTX_new_id(int id, ENGINE *e)
303         {
304         return int_ctx_new(NULL, e, id);
305         }

307 EVP_PKEY_CTX *EVP_PKEY_CTX_dup(EVP_PKEY_CTX *pctx)
308         {
309         EVP_PKEY_CTX *rctx;
310         if (!pctx->pmeth || !pctx->pmeth->copy)
311                 return NULL;
312 #ifndef OPENSSL_NO_ENGINE
313         /* Make sure it's safe to copy a pkey context using an ENGINE */
314         if (pctx->engine && !ENGINE_init(pctx->engine))
315                 {
316                 EVPerr(EVP_F_EVP_PKEY_CTX_DUP,ERR_R_ENGINE_LIB);
317                 return 0;
318                 }
319 #endif
320         rctx = OPENSSL_malloc(sizeof(EVP_PKEY_CTX));
321         if (!rctx)
322                 return NULL;

324         rctx->pmeth = pctx->pmeth;
325 #ifndef OPENSSL_NO_ENGINE
```



```
326         rctx->engine = pctx->engine;
327 #endif

329         if (pctx->pkey)
330                 CRYPTO_add(&pctx->pkey->references,1,CRYPTO_LOCK_EVP_PKEY);

332         rctx->pkey = pctx->pkey;

334         if (pctx->peerkey)
335                 CRYPTO_add(&pctx->peerkey->references,1,CRYPTO_LOCK_EVP_PKEY);

337         rctx->peerkey = pctx->peerkey;

339         rctx->data = NULL;
340         rctx->app_data = NULL;
341         rctx->operation = pctx->operation;

343         if (pctx->pmeth->copy(rctx, pctx) > 0)
344                 return rctx;

346         EVP_PKEY_CTX_free(rctx);
347         return NULL;

349         }

351 int EVP_PKEY_meth_add0(const EVP_PKEY_METHOD *pmeth)
352         {
353         if (app_pkey_methods == NULL)
354                 {
355                 app_pkey_methods = sk_EVP_PKEY_METHOD_new(pmeth_cmp);
356                 if (!app_pkey_methods)
357                         return 0;
358                 }
359         if (!sk_EVP_PKEY_METHOD_push(app_pkey_methods, pmeth))
360                 return 0;
361         sk_EVP_PKEY_METHOD_sort(app_pkey_methods);
362         return 1;
363         }

365 void EVP_PKEY_CTX_free(EVP_PKEY_CTX *ctx)
366         {
367         if (ctx == NULL)
368                 return;
369         if (ctx->pmeth && ctx->pmeth->cleanup)
370                 ctx->pmeth->cleanup(ctx);
371         if (ctx->pkey)
372                 EVP_PKEY_free(ctx->pkey);
373         if (ctx->peerkey)
374                 EVP_PKEY_free(ctx->peerkey);
375 #ifndef OPENSSL_NO_ENGINE
376         if(ctx->engine)
377                 /* The EVP_PKEY_CTX we used belongs to an ENGINE, release the
378                  * functional reference we held for this reason. */
379                 ENGINE_finish(ctx->engine);
380 #endif
381         OPENSSL_free(ctx);
382         }

384 int EVP_PKEY_CTX_ctrl(EVP_PKEY_CTX *ctx, int keytype, int optype,
385                                 int cmd, int p1, void *p2)
386         {
387         int ret;
388         if (!ctx || !ctx->pmeth || !ctx->pmeth->ctrl)
389                 {
390                 EVPerr(EVP_F_EVP_PKEY_CTX_CTRL, EVP_R_COMMAND_NOT_SUPPORTED);
391                 return -2;
```

```
392                 }
393         if ((keytype != -1) && (ctx->pmeth->pkey_id != keytype))
394                 return -1;

396         if (ctx->operation == EVP_PKEY_OP_UNDEFINED)
397                 {
398                 EVPerr(EVP_F_EVP_PKEY_CTX_CTRL, EVP_R_NO_OPERATION_SET);
399                 return -1;
400                 }

402         if ((optype != -1) && !(ctx->operation & optype))
403                 {
404                 EVPerr(EVP_F_EVP_PKEY_CTX_CTRL, EVP_R_INVALID_OPERATION);
405                 return -1;
406                 }

408         ret = ctx->pmeth->ctrl(ctx, cmd, p1, p2);

410         if (ret == -2)
411                 EVPerr(EVP_F_EVP_PKEY_CTX_CTRL, EVP_R_COMMAND_NOT_SUPPORTED);

413         return ret;

415         }

417 int EVP_PKEY_CTX_ctrl_str(EVP_PKEY_CTX *ctx,
418                                     const char *name, const char *value)
419         {
420         if (!ctx || !ctx->pmeth || !ctx->pmeth->ctrl_str)
421                 {
422                 EVPerr(EVP_F_EVP_PKEY_CTX_CTRL_STR,
423                                                 EVP_R_COMMAND_NOT_SUPPORTED);
424                 return -2;
425                 }
426         if (!strcmp(name, "digest"))
427                 {
428                 const EVP_MD *md;
429                 if (!value || !(md = EVP_get_digestbyname(value)))
430                         {
431                         EVPerr(EVP_F_EVP_PKEY_CTX_CTRL_STR,
432                                                 EVP_R_INVALID_DIGEST);
433                         return 0;
434                         }
435                 return EVP_PKEY_CTX_set_signature_md(ctx, md);
436                 }
437         return ctx->pmeth->ctrl_str(ctx, name, value);
438         }

440 int EVP_PKEY_CTX_get_operation(EVP_PKEY_CTX *ctx)
441         {
442         return ctx->operation;
443         }

445 void EVP_PKEY_CTX_set0_keygen_info(EVP_PKEY_CTX *ctx, int *dat, int datlen)
446         {
447         ctx->keygen_info = dat;
448         ctx->keygen_info_count = datlen;
449         }

451 void EVP_PKEY_CTX_set_data(EVP_PKEY_CTX *ctx, void *data)
452         {
453         ctx->data = data;
454         }

456 void *EVP_PKEY_CTX_get_data(EVP_PKEY_CTX *ctx)
457         {
```



```
458         return ctx->data;
459         }

461 EVP_PKEY *EVP_PKEY_CTX_get0_pkey(EVP_PKEY_CTX *ctx)
462         {
463         return ctx->pkey;
464         }

466 EVP_PKEY *EVP_PKEY_CTX_get0_peerkey(EVP_PKEY_CTX *ctx)
467         {
468         return ctx->peerkey;
469         }

471 void EVP_PKEY_CTX_set_app_data(EVP_PKEY_CTX *ctx, void *data)
472         {
473         ctx->app_data = data;
474         }

476 void *EVP_PKEY_CTX_get_app_data(EVP_PKEY_CTX *ctx)
477         {
478         return ctx->app_data;
479         }

481 void EVP_PKEY_meth_set_init(EVP_PKEY_METHOD *pmeth,
482         int (*init)(EVP_PKEY_CTX *ctx))
483         {
484         pmeth->init = init;
485         }

487 void EVP_PKEY_meth_set_copy(EVP_PKEY_METHOD *pmeth,
488         int (*copy)(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src))
489         {
490         pmeth->copy = copy;
491         }

493 void EVP_PKEY_meth_set_cleanup(EVP_PKEY_METHOD *pmeth,
494         void (*cleanup)(EVP_PKEY_CTX *ctx))
495         {
496         pmeth->cleanup = cleanup;
497         }

499 void EVP_PKEY_meth_set_paramgen(EVP_PKEY_METHOD *pmeth,
500         int (*paramgen_init)(EVP_PKEY_CTX *ctx),
501         int (*paramgen)(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey))
502         {
503         pmeth->paramgen_init = paramgen_init;
504         pmeth->paramgen = paramgen;
505         }

507 void EVP_PKEY_meth_set_keygen(EVP_PKEY_METHOD *pmeth,
508         int (*keygen_init)(EVP_PKEY_CTX *ctx),
509         int (*keygen)(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey))
510         {
511         pmeth->keygen_init = keygen_init;
512         pmeth->keygen = keygen;
513         }

515 void EVP_PKEY_meth_set_sign(EVP_PKEY_METHOD *pmeth,
516         int (*sign_init)(EVP_PKEY_CTX *ctx),
517         int (*sign)(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
518                                         const unsigned char *tbs, size_t tbslen)
519         {
520         pmeth->sign_init = sign_init;
521         pmeth->sign = sign;
522         }
```

```
524 void EVP_PKEY_meth_set_verify(EVP_PKEY_METHOD *pmeth,
525         int (*verify_init)(EVP_PKEY_CTX *ctx),
526         int (*verify)(EVP_PKEY_CTX *ctx, const unsigned char *sig, size_t siglen
527                                         const unsigned char *tbs, size_t tbslen)
528         {
529         pmeth->verify_init = verify_init;
530         pmeth->verify = verify;
531         }

533 void EVP_PKEY_meth_set_verify_recover(EVP_PKEY_METHOD *pmeth,
534         int (*verify_recover_init)(EVP_PKEY_CTX *ctx),
535         int (*verify_recover)(EVP_PKEY_CTX *ctx,
536                                         unsigned char *sig, size_t *siglen,
537                                         const unsigned char *tbs, size_t tbslen)
538         {
539         pmeth->verify_recover_init = verify_recover_init;
540         pmeth->verify_recover = verify_recover;
541         }

543 void EVP_PKEY_meth_set_signctx(EVP_PKEY_METHOD *pmeth,
544         int (*signctx_init)(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx),
545         int (*signctx)(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
546                                         EVP_MD_CTX *mctx))
547         {
548         pmeth->signctx_init = signctx_init;
549         pmeth->signctx = signctx;
550         }

552 void EVP_PKEY_meth_set_verifyctx(EVP_PKEY_METHOD *pmeth,
553         int (*verifyctx_init)(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx),
554         int (*verifyctx)(EVP_PKEY_CTX *ctx, const unsigned char *sig,int siglen,
555                                         EVP_MD_CTX *mctx))
556         {
557         pmeth->verifyctx_init = verifyctx_init;
558         pmeth->verifyctx = verifyctx;
559         }

561 void EVP_PKEY_meth_set_encrypt(EVP_PKEY_METHOD *pmeth,
562         int (*encrypt_init)(EVP_PKEY_CTX *ctx),
563         int (*encryptfn)(EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,
564                                         const unsigned char *in, size_t inlen))
565         {
566         pmeth->encrypt_init = encrypt_init;
567         pmeth->encrypt = encryptfn;
568         }

570 void EVP_PKEY_meth_set_decrypt(EVP_PKEY_METHOD *pmeth,
571         int (*decrypt_init)(EVP_PKEY_CTX *ctx),
572         int (*decrypt)(EVP_PKEY_CTX *ctx, unsigned char *out, size_t *outlen,
573                                         const unsigned char *in, size_t inlen))
574         {
575         pmeth->decrypt_init = decrypt_init;
576         pmeth->decrypt = decrypt;
577         }

579 void EVP_PKEY_meth_set_derive(EVP_PKEY_METHOD *pmeth,
580         int (*derive_init)(EVP_PKEY_CTX *ctx),
581         int (*derive)(EVP_PKEY_CTX *ctx, unsigned char *key, size_t *keylen))
582         {
583         pmeth->derive_init = derive_init;
584         pmeth->derive = derive;
585         }

587 void EVP_PKEY_meth_set_ctrl(EVP_PKEY_METHOD *pmeth,
588         int (*ctrl)(EVP_PKEY_CTX *ctx, int type, int p1, void *p2),
589         int (*ctrl_str)(EVP_PKEY_CTX *ctx, const char *type, const char *value))
```



```
590         {
591         pmeth->ctrl = ctrl;
592         pmeth->ctrl_str = ctrl_str;
593         }
594 #endif /* ! codereview */
```

```
**********************************************************
   21398 Wed Aug 13 19:52:51 2014
new/usr/src/lib/openssl/libsunw_crypto/ex_data.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ex_data.c */

/*
 * Overhaul notes;
 *
 * This code is now *mostly* thread-safe. It is now easier to understand in what
 * ways it is safe and in what ways it is not, which is an improvement. Firstly,
 * all per-class stacks and index-counters for ex_data are stored in the same
 * global LHASH table (keyed by class). This hash table uses locking for all
 * access with the exception of CRYPTO_cleanup_all_ex_data(), which must only be
 * called when no other threads can possibly race against it (even if it was
 * locked, the race would mean it's possible the hash table might have been
 * recreated after the cleanup). As classes can only be added to the hash table,
 * and within each class, the stack of methods can only be incremented, the
 * locking mechanics are simpler than they would otherwise be. For example, the
 * new/dup/free ex_data functions will lock the hash table, copy the method
 * pointers it needs from the relevant class, then unlock the hash table before
 * actually applying those method pointers to the task of the new/dup/free
 * operations. As they can't be removed from the method-stack, only
 * supplemented, there's no race conditions associated with using them outside
 * the lock. The get/set_ex_data functions are not locked because they do not
 * involve this global state at all - they operate directly with a previously
 * obtained per-class method index and a particular "ex_data" variable. These
 * variables are usually instantiated per-context (eg. each RSA structure has
 * one) so locking on read/write access to that variable can be locked locally
 * if required (eg. using the "RSA" lock to synchronise access to a
 * per-RSA-structure ex_data variable if required).
 * [Geoff]
 */

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
```



```
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
```

```
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <cryptlib.h>
#include <openssl/lhash.h>

/* What an "implementation of ex_data functionality" looks like */
struct st_CRYPTO_EX_DATA_IMPL
        {
        /*********************/
        /* GLOBAL OPERATIONS */
        /* Return a new class index */
        int (*cb_new_class)(void);
        /* Cleanup all state used by the implementation */
        void (*cb_cleanup)(void);
        /************************/
        /* PER-CLASS OPERATIONS */
        /* Get a new method index within a class */
        int (*cb_get_new_index)(int class_index, long argl, void *argp,
                        CRYPTO_EX_new *new_func, CRYPTO_EX_dup *dup_func,
                        CRYPTO_EX_free *free_func);
        /* Initialise a new CRYPTO_EX_DATA of a given class */
        int (*cb_new_ex_data)(int class_index, void *obj,
                        CRYPTO_EX_DATA *ad);
        /* Duplicate a CRYPTO_EX_DATA of a given class onto a copy */
        int (*cb_dup_ex_data)(int class_index, CRYPTO_EX_DATA *to,
                        CRYPTO_EX_DATA *from);
        /* Cleanup a CRYPTO_EX_DATA of a given class */
        void (*cb_free_ex_data)(int class_index, void *obj,
                        CRYPTO_EX_DATA *ad);
        };

/* The implementation we use at run-time */
static const CRYPTO_EX_DATA_IMPL *impl = NULL;

/* To call "impl" functions, use this macro rather than referring to 'impl' dire
 * EX_IMPL(get_new_index)(...); */
#define EX_IMPL(a) impl->cb_##a

/* Predeclare the "default" ex_data implementation */
static int int_new_class(void);
static void int_cleanup(void);
static int int_get_new_index(int class_index, long argl, void *argp,
                CRYPTO_EX_new *new_func, CRYPTO_EX_dup *dup_func,
                CRYPTO_EX_free *free_func);
static int int_new_ex_data(int class_index, void *obj,
                CRYPTO_EX_DATA *ad);
static int int_dup_ex_data(int class_index, CRYPTO_EX_DATA *to,
                CRYPTO_EX_DATA *from);
static void int_free_ex_data(int class_index, void *obj,
                CRYPTO_EX_DATA *ad);
static CRYPTO_EX_DATA_IMPL impl_default =
        {
        int_new_class,
        int_cleanup,
        int_get_new_index,
```



```
        int_new_ex_data,
        int_dup_ex_data,
        int_free_ex_data
        };

/* Internal function that checks whether "impl" is set and if not, sets it to
 * the default. */
static void impl_check(void)
        {
        CRYPTO_w_lock(CRYPTO_LOCK_EX_DATA);
        if(!impl)
                impl = &impl_default;
        CRYPTO_w_unlock(CRYPTO_LOCK_EX_DATA);
        }
/* A macro wrapper for impl_check that first uses a non-locked test before
 * invoking the function (which checks again inside a lock). */
#define IMPL_CHECK if(!impl) impl_check();

/* API functions to get/set the "ex_data" implementation */
const CRYPTO_EX_DATA_IMPL *CRYPTO_get_ex_data_implementation(void)
        {
        IMPL_CHECK
        return impl;
        }
int CRYPTO_set_ex_data_implementation(const CRYPTO_EX_DATA_IMPL *i)
        {
        int toret = 0;
        CRYPTO_w_lock(CRYPTO_LOCK_EX_DATA);
        if(!impl)
                {
                impl = i;
                toret = 1;
                }
        CRYPTO_w_unlock(CRYPTO_LOCK_EX_DATA);
        return toret;
        }

/****************************************************************************/
/* Interal (default) implementation of "ex_data" support. API functions are
 * further down. */

/* The type that represents what each "class" used to implement locally. A STACK
 * of CRYPTO_EX_DATA_FUNCS plus a index-counter. The 'class_index' is the global
 * value representing the class that is used to distinguish these items. */
typedef struct st_ex_class_item {
        int class_index;
        STACK_OF(CRYPTO_EX_DATA_FUNCS) *meth;
        int meth_num;
} EX_CLASS_ITEM;

/* When assigning new class indexes, this is our counter */
static int ex_class = CRYPTO_EX_INDEX_USER;

/* The global hash table of EX_CLASS_ITEM items */
DECLARE_LHASH_OF(EX_CLASS_ITEM);
static LHASH_OF(EX_CLASS_ITEM) *ex_data = NULL;

/* The callbacks required in the "ex_data" hash table */
static unsigned long ex_class_item_hash(const EX_CLASS_ITEM *a)
        {
        return a->class_index;
        }
static IMPLEMENT_LHASH_HASH_FN(ex_class_item, EX_CLASS_ITEM)

static int ex_class_item_cmp(const EX_CLASS_ITEM *a, const EX_CLASS_ITEM *b)
        {
```

```
260         return a->class_index - b->class_index;
261         }
262 static IMPLEMENT_LHASH_COMP_FN(ex_class_item, EX_CLASS_ITEM)

264 /* Internal functions used by the "impl_default" implementation to access the
265  * state */

267 static int ex_data_check(void)
268         {
269         int toret = 1;
270         CRYPTO_w_lock(CRYPTO_LOCK_EX_DATA);
271         if(!ex_data
272            && (ex_data = lh_EX_CLASS_ITEM_new()) == NULL)
273                 toret = 0;
274         CRYPTO_w_unlock(CRYPTO_LOCK_EX_DATA);
275         return toret;
276         }
277 /* This macros helps reduce the locking from repeated checks because the
278  * ex_data_check() function checks ex_data again inside a lock. */
279 #define EX_DATA_CHECK(iffail) if(!ex_data && !ex_data_check()) {iffail}

281 /* This "inner" callback is used by the callback function that follows it */
282 static void def_cleanup_util_cb(CRYPTO_EX_DATA_FUNCS *funcs)
283         {
284         OPENSSL_free(funcs);
285         }

287 /* This callback is used in lh_doall to destroy all EX_CLASS_ITEM values from
288  * "ex_data" prior to the ex_data hash table being itself destroyed. Doesn't do
289  * any locking. */
290 static void def_cleanup_cb(void *a_void)
291         {
292         EX_CLASS_ITEM *item = (EX_CLASS_ITEM *)a_void;
293         sk_CRYPTO_EX_DATA_FUNCS_pop_free(item->meth, def_cleanup_util_cb);
294         OPENSSL_free(item);
295         }

297 /* Return the EX_CLASS_ITEM from the "ex_data" hash table that corresponds to a
298  * given class. Handles locking. */
299 static EX_CLASS_ITEM *def_get_class(int class_index)
300         {
301         EX_CLASS_ITEM d, *p, *gen;
302         EX_DATA_CHECK(return NULL;)
303         d.class_index = class_index;
304         CRYPTO_w_lock(CRYPTO_LOCK_EX_DATA);
305         p = lh_EX_CLASS_ITEM_retrieve(ex_data, &d);
306         if(!p)
307                 {
308                 gen = OPENSSL_malloc(sizeof(EX_CLASS_ITEM));
309                 if(gen)
310                         {
311                         gen->class_index = class_index;
312                         gen->meth_num = 0;
313                         gen->meth = sk_CRYPTO_EX_DATA_FUNCS_new_null();
314                         if(!gen->meth)
315                                 OPENSSL_free(gen);
316                         else
317                                 {
318                                 /* Because we're inside the ex_data lock, the
319                                  * return value from the insert will be NULL */
320                                 (void)lh_EX_CLASS_ITEM_insert(ex_data, gen);
321                                 p = gen;
322                                 }
323                         }
324                 }
325         CRYPTO_w_unlock(CRYPTO_LOCK_EX_DATA);
```



```
326         if(!p)
327                 CRYPTOerr(CRYPTO_F_DEF_GET_CLASS,ERR_R_MALLOC_FAILURE);
328         return p;
329         }

331 /* Add a new method to the given EX_CLASS_ITEM and return the corresponding
332  * index (or -1 for error). Handles locking. */
333 static int def_add_index(EX_CLASS_ITEM *item, long argl, void *argp,
334                 CRYPTO_EX_new *new_func, CRYPTO_EX_dup *dup_func,
335                 CRYPTO_EX_free *free_func)
336         {
337         int toret = -1;
338         CRYPTO_EX_DATA_FUNCS *a = (CRYPTO_EX_DATA_FUNCS *)OPENSSL_malloc(
339                                         sizeof(CRYPTO_EX_DATA_FUNCS));
340         if(!a)
341                 {
342                 CRYPTOerr(CRYPTO_F_DEF_ADD_INDEX,ERR_R_MALLOC_FAILURE);
343                 return -1;
344                 }
345         a->argl=argl;
346         a->argp=argp;
347         a->new_func=new_func;
348         a->dup_func=dup_func;
349         a->free_func=free_func;
350         CRYPTO_w_lock(CRYPTO_LOCK_EX_DATA);
351         while (sk_CRYPTO_EX_DATA_FUNCS_num(item->meth) <= item->meth_num)
352                 {
353                 if (!sk_CRYPTO_EX_DATA_FUNCS_push(item->meth, NULL))
354                         {
355                         CRYPTOerr(CRYPTO_F_DEF_ADD_INDEX,ERR_R_MALLOC_FAILURE);
356                         OPENSSL_free(a);
357                         goto err;
358                         }
359                 }
360         toret = item->meth_num++;
361         (void)sk_CRYPTO_EX_DATA_FUNCS_set(item->meth, toret, a);
362 err:
363         CRYPTO_w_unlock(CRYPTO_LOCK_EX_DATA);
364         return toret;
365         }

367 /************************************************************/
368 /* The functions in the default CRYPTO_EX_DATA_IMPL structure */

370 static int int_new_class(void)
371         {
372         int toret;
373         CRYPTO_w_lock(CRYPTO_LOCK_EX_DATA);
374         toret = ex_class++;
375         CRYPTO_w_unlock(CRYPTO_LOCK_EX_DATA);
376         return toret;
377         }

379 static void int_cleanup(void)
380         {
381         EX_DATA_CHECK(return;)
382         lh_EX_CLASS_ITEM_doall(ex_data, def_cleanup_cb);
383         lh_EX_CLASS_ITEM_free(ex_data);
384         ex_data = NULL;
385         impl = NULL;
386         }

388 static int int_get_new_index(int class_index, long argl, void *argp,
389                 CRYPTO_EX_new *new_func, CRYPTO_EX_dup *dup_func,
390                 CRYPTO_EX_free *free_func)
391         {
```

```
392         EX_CLASS_ITEM *item = def_get_class(class_index);
393         if(!item)
394                 return -1;
395         return def_add_index(item, argl, argp, new_func, dup_func, free_func);
396         }

398 /* Thread-safe by copying a class's array of "CRYPTO_EX_DATA_FUNCS" entries in
399  * the lock, then using them outside the lock. NB: Thread-safety only applies to
400  * the global "ex_data" state (ie. class definitions), not thread-safe on 'ad'
401  * itself. */
402 static int int_new_ex_data(int class_index, void *obj,
403                 CRYPTO_EX_DATA *ad)
404         {
405         int mx,i;
406         void *ptr;
407         CRYPTO_EX_DATA_FUNCS **storage = NULL;
408         EX_CLASS_ITEM *item = def_get_class(class_index);
409         if(!item)
410                 /* error is already set */
411                 return 0;
412         ad->sk = NULL;
413         CRYPTO_r_lock(CRYPTO_LOCK_EX_DATA);
414         mx = sk_CRYPTO_EX_DATA_FUNCS_num(item->meth);
415         if(mx > 0)
416                 {
417                 storage = OPENSSL_malloc(mx * sizeof(CRYPTO_EX_DATA_FUNCS*));
418                 if(!storage)
419                         goto skip;
420                 for(i = 0; i < mx; i++)
421                         storage[i] = sk_CRYPTO_EX_DATA_FUNCS_value(item->meth,i)
422                 }
423 skip:
424         CRYPTO_r_unlock(CRYPTO_LOCK_EX_DATA);
425         if((mx > 0) && !storage)
426                 {
427                 CRYPTOerr(CRYPTO_F_INT_NEW_EX_DATA,ERR_R_MALLOC_FAILURE);
428                 return 0;
429                 }
430         for(i = 0; i < mx; i++)
431                 {
432                 if(storage[i] && storage[i]->new_func)
433                         {
434                         ptr = CRYPTO_get_ex_data(ad, i);
435                         storage[i]->new_func(obj,ptr,ad,i,
436                                 storage[i]->argl,storage[i]->argp);
437                         }
438                 }
439         if(storage)
440                 OPENSSL_free(storage);
441         return 1;
442         }

444 /* Same thread-safety notes as for "int_new_ex_data" */
445 static int int_dup_ex_data(int class_index, CRYPTO_EX_DATA *to,
446                 CRYPTO_EX_DATA *from)
447         {
448         int mx, j, i;
449         char *ptr;
450         CRYPTO_EX_DATA_FUNCS **storage = NULL;
451         EX_CLASS_ITEM *item;
452         if(!from->sk)
453                 /* 'to' should be "blank" which *is* just like 'from' */
454                 return 1;
455         if((item = def_get_class(class_index)) == NULL)
456                 return 0;
457         CRYPTO_r_lock(CRYPTO_LOCK_EX_DATA);
```



```
458         mx = sk_CRYPTO_EX_DATA_FUNCS_num(item->meth);
459         j = sk_void_num(from->sk);
460         if(j < mx)
461                 mx = j;
462         if(mx > 0)
463                 {
464                 storage = OPENSSL_malloc(mx * sizeof(CRYPTO_EX_DATA_FUNCS*));
465                 if(!storage)
466                         goto skip;
467                 for(i = 0; i < mx; i++)
468                         storage[i] = sk_CRYPTO_EX_DATA_FUNCS_value(item->meth,i)
469                 }
470 skip:
471         CRYPTO_r_unlock(CRYPTO_LOCK_EX_DATA);
472         if((mx > 0) && !storage)
473                 {
474                 CRYPTOerr(CRYPTO_F_INT_DUP_EX_DATA,ERR_R_MALLOC_FAILURE);
475                 return 0;
476                 }
477         for(i = 0; i < mx; i++)
478                 {
479                 ptr = CRYPTO_get_ex_data(from, i);
480                 if(storage[i] && storage[i]->dup_func)
481                         storage[i]->dup_func(to,from,&ptr,i,
482                                 storage[i]->argl,storage[i]->argp);
483                 CRYPTO_set_ex_data(to,i,ptr);
484                 }
485         if(storage)
486                 OPENSSL_free(storage);
487         return 1;
488         }

490 /* Same thread-safety notes as for "int_new_ex_data" */
491 static void int_free_ex_data(int class_index, void *obj,
492                 CRYPTO_EX_DATA *ad)
493         {
494         int mx,i;
495         EX_CLASS_ITEM *item;
496         void *ptr;
497         CRYPTO_EX_DATA_FUNCS **storage = NULL;
498         if((item = def_get_class(class_index)) == NULL)
499                 return;
500         CRYPTO_r_lock(CRYPTO_LOCK_EX_DATA);
501         mx = sk_CRYPTO_EX_DATA_FUNCS_num(item->meth);
502         if(mx > 0)
503                 {
504                 storage = OPENSSL_malloc(mx * sizeof(CRYPTO_EX_DATA_FUNCS*));
505                 if(!storage)
506                         goto skip;
507                 for(i = 0; i < mx; i++)
508                         storage[i] = sk_CRYPTO_EX_DATA_FUNCS_value(item->meth,i)
509                 }
510 skip:
511         CRYPTO_r_unlock(CRYPTO_LOCK_EX_DATA);
512         if((mx > 0) && !storage)
513                 {
514                 CRYPTOerr(CRYPTO_F_INT_FREE_EX_DATA,ERR_R_MALLOC_FAILURE);
515                 return;
516                 }
517         for(i = 0; i < mx; i++)
518                 {
519                 if(storage[i] && storage[i]->free_func)
520                         {
521                         ptr = CRYPTO_get_ex_data(ad,i);
522                         storage[i]->free_func(obj,ptr,ad,i,
523                                 storage[i]->argl,storage[i]->argp);
```

```
524                         }
525                 }
526         if(storage)
527                 OPENSSL_free(storage);
528         if(ad->sk)
529                 {
530                 sk_void_free(ad->sk);
531                 ad->sk=NULL;
532                 }
533         }

535 /************************************************************/
536 /* API functions that defer all "state" operations to the "ex_data"
537  * implementation we have set. */

539 /* Obtain an index for a new class (not the same as getting a new index within
540  * an existing class - this is actually getting a new *class*) */
541 int CRYPTO_ex_data_new_class(void)
542         {
543         IMPL_CHECK
544         return EX_IMPL(new_class)();
545         }

547 /* Release all "ex_data" state to prevent memory leaks. This can't be made
548  * thread-safe without overhauling a lot of stuff, and shouldn't really be
549  * called under potential race-conditions anyway (it's for program shutdown
550  * after all). */
551 void CRYPTO_cleanup_all_ex_data(void)
552         {
553         IMPL_CHECK
554         EX_IMPL(cleanup)();
555         }

557 /* Inside an existing class, get/register a new index. */
558 int CRYPTO_get_ex_new_index(int class_index, long argl, void *argp,
559                 CRYPTO_EX_new *new_func, CRYPTO_EX_dup *dup_func,
560                 CRYPTO_EX_free *free_func)
561         {
562         int ret = -1;

564         IMPL_CHECK
565         ret = EX_IMPL(get_new_index)(class_index,
566                         argl, argp, new_func, dup_func, free_func);
567         return ret;
568         }

570 /* Initialise a new CRYPTO_EX_DATA for use in a particular class - including
571  * calling new() callbacks for each index in the class used by this variable */
572 int CRYPTO_new_ex_data(int class_index, void *obj, CRYPTO_EX_DATA *ad)
573         {
574         IMPL_CHECK
575         return EX_IMPL(new_ex_data)(class_index, obj, ad);
576         }

578 /* Duplicate a CRYPTO_EX_DATA variable - including calling dup() callbacks for
579  * each index in the class used by this variable */
580 int CRYPTO_dup_ex_data(int class_index, CRYPTO_EX_DATA *to,
581             CRYPTO_EX_DATA *from)
582         {
583         IMPL_CHECK
584         return EX_IMPL(dup_ex_data)(class_index, to, from);
585         }

587 /* Cleanup a CRYPTO_EX_DATA variable - including calling free() callbacks for
588  * each index in the class used by this variable */
589 void CRYPTO_free_ex_data(int class_index, void *obj, CRYPTO_EX_DATA *ad)
```



```
590         {
591         IMPL_CHECK
592         EX_IMPL(free_ex_data)(class_index, obj, ad);
593         }

595 /* For a given CRYPTO_EX_DATA variable, set the value corresponding to a
596  * particular index in the class used by this variable */
597 int CRYPTO_set_ex_data(CRYPTO_EX_DATA *ad, int idx, void *val)
598         {
599         int i;

601         if (ad->sk == NULL)
602                 {
603                 if ((ad->sk=sk_void_new_null()) == NULL)
604                         {
605                         CRYPTOerr(CRYPTO_F_CRYPTO_SET_EX_DATA,ERR_R_MALLOC_FAILU
606                         return(0);
607                         }
608                 }
609         i=sk_void_num(ad->sk);

611         while (i <= idx)
612                 {
613                 if (!sk_void_push(ad->sk,NULL))
614                         {
615                         CRYPTOerr(CRYPTO_F_CRYPTO_SET_EX_DATA,ERR_R_MALLOC_FAILU
616                         return(0);
617                         }
618                 i++;
619                 }
620         sk_void_set(ad->sk,idx,val);
621         return(1);
622         }

624 /* For a given CRYPTO_EX_DATA_ variable, get the value corresponding to a
625  * particular index in the class used by this variable */
626 void *CRYPTO_get_ex_data(const CRYPTO_EX_DATA *ad, int idx)
627         {
628         if (ad->sk == NULL)
629                 return(0);
630         else if (idx >= sk_void_num(ad->sk))
631                 return(0);
632         else
633                 return(sk_void_value(ad->sk,idx));
634         }

636 IMPLEMENT_STACK_OF(CRYPTO_EX_DATA_FUNCS)
637 #endif /* ! codereview */
```

```
**********************************************************
     117 Wed Aug 13 19:52:51 2014
new/usr/src/lib/openssl/libsunw_crypto/fips_ers.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#include <openssl/opensslconf.h>

#ifdef OPENSSL_FIPS
# include "fips_err.h"
#else
static void *dummy=&dummy;
#endif
#endif /* ! codereview */
```

```
**********************************************************
    4630 Wed Aug 13 19:52:51 2014
new/usr/src/lib/openssl/libsunw_crypto/hmac/hm_ameth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2007.
 */
/* ====================================================================
 * Copyright (c) 2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
#include "asn1_locl.h"
```



```
#define HMAC_TEST_PRIVATE_KEY_FORMAT

/* HMAC "ASN1" method. This is just here to indicate the
 * maximum HMAC output length and to free up an HMAC
 * key.
 */

static int hmac_size(const EVP_PKEY *pkey)
        {
        return EVP_MAX_MD_SIZE;
        }

static void hmac_key_free(EVP_PKEY *pkey)
        {
        ASN1_OCTET_STRING *os = (ASN1_OCTET_STRING *)pkey->pkey.ptr;
        if (os)
                {
                if (os->data)
                        OPENSSL_cleanse(os->data, os->length);
                ASN1_OCTET_STRING_free(os);
                }
        }


static int hmac_pkey_ctrl(EVP_PKEY *pkey, int op, long arg1, void *arg2)
        {
        switch (op)
                {
                case ASN1_PKEY_CTRL_DEFAULT_MD_NID:
                *(int *)arg2 = NID_sha1;
                return 1;

                default:
                return -2;
                }
        }

#ifdef HMAC_TEST_PRIVATE_KEY_FORMAT
/* A bogus private key format for test purposes. This is simply the
 * HMAC key with "HMAC PRIVATE KEY" in the headers. When enabled the
 * genpkey utility can be used to "generate" HMAC keys.
 */

static int old_hmac_decode(EVP_PKEY *pkey,
                                        const unsigned char **pder, int derlen)
        {
        ASN1_OCTET_STRING *os;
        os = ASN1_OCTET_STRING_new();
        if (!os || !ASN1_OCTET_STRING_set(os, *pder, derlen))
                return 0;
        EVP_PKEY_assign(pkey, EVP_PKEY_HMAC, os);
        return 1;
        }

static int old_hmac_encode(const EVP_PKEY *pkey, unsigned char **pder)
        {
        int inc;
        ASN1_OCTET_STRING *os = (ASN1_OCTET_STRING *)pkey->pkey.ptr;
        if (pder)
                {
                if (!*pder)
                        {
                        *pder = OPENSSL_malloc(os->length);
                        inc = 0;
                        }
```

```
128                 else inc = 1;
130                 memcpy(*pder, os->data, os->length);
132                 if (inc)
133                         *pder += os->length;
134                 }
136         return os->length;
137         }
139 #endif
141 const EVP_PKEY_ASN1_METHOD hmac_asn1_meth =
142         {
143         EVP_PKEY_HMAC,
144         EVP_PKEY_HMAC,
145         0,
147         "HMAC",
148         "OpenSSL HMAC method",
150         0,0,0,0,
152         0,0,0,
154         hmac_size,
155         0,
156         0,0,0,0,0,0,0,
158         hmac_key_free,
159         hmac_pkey_ctrl,
160 #ifdef HMAC_TEST_PRIVATE_KEY_FORMAT
161         old_hmac_decode,
162         old_hmac_encode
163 #else
164         0,0
165 #endif
166         };
167 #endif /* ! codereview */
```

```
**********************************************************
    6660 Wed Aug 13 19:52:51 2014
new/usr/src/lib/openssl/libsunw_crypto/hmac/hm_pmeth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2007.
 */
/* ====================================================================
 * Copyright (c) 2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/x509v3.h>
```



```
#include <openssl/evp.h>
#include <openssl/hmac.h>
#include "evp_locl.h"

/* HMAC pkey context structure */

typedef struct
        {
        const EVP_MD *md;       /* MD for HMAC use */
        ASN1_OCTET_STRING ktmp; /* Temp storage for key */
        HMAC_CTX ctx;
        } HMAC_PKEY_CTX;

static int pkey_hmac_init(EVP_PKEY_CTX *ctx)
        {
        HMAC_PKEY_CTX *hctx;
        hctx = OPENSSL_malloc(sizeof(HMAC_PKEY_CTX));
        if (!hctx)
                return 0;
        hctx->md = NULL;
        hctx->ktmp.data = NULL;
        hctx->ktmp.length = 0;
        hctx->ktmp.flags = 0;
        hctx->ktmp.type = V_ASN1_OCTET_STRING;
        HMAC_CTX_init(&hctx->ctx);

        ctx->data = hctx;
        ctx->keygen_info_count = 0;

        return 1;
        }

static int pkey_hmac_copy(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src)
        {
        HMAC_PKEY_CTX *sctx, *dctx;
        if (!pkey_hmac_init(dst))
                return 0;
        sctx = src->data;
        dctx = dst->data;
        dctx->md = sctx->md;
        HMAC_CTX_init(&dctx->ctx);
        if (!HMAC_CTX_copy(&dctx->ctx, &sctx->ctx))
                return 0;
        if (sctx->ktmp.data)
                {
                if (!ASN1_OCTET_STRING_set(&dctx->ktmp,
                                        sctx->ktmp.data, sctx->ktmp.length))
                        return 0;
                }
        return 1;
        }

static void pkey_hmac_cleanup(EVP_PKEY_CTX *ctx)
        {
        HMAC_PKEY_CTX *hctx = ctx->data;
        HMAC_CTX_cleanup(&hctx->ctx);
        if (hctx->ktmp.data)
                {
                if (hctx->ktmp.length)
                        OPENSSL_cleanse(hctx->ktmp.data, hctx->ktmp.length);
                OPENSSL_free(hctx->ktmp.data);
                hctx->ktmp.data = NULL;
                }
        OPENSSL_free(hctx);
        }
```

```
128 static int pkey_hmac_keygen(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey)
129         {
130         ASN1_OCTET_STRING *hkey = NULL;
131         HMAC_PKEY_CTX *hctx = ctx->data;
132         if (!hctx->ktmp.data)
133                 return 0;
134         hkey = ASN1_OCTET_STRING_dup(&hctx->ktmp);
135         if (!hkey)
136                 return 0;
137         EVP_PKEY_assign(pkey, EVP_PKEY_HMAC, hkey);
138
139         return 1;
140         }
141
142 static int int_update(EVP_MD_CTX *ctx,const void *data,size_t count)
143         {
144         HMAC_PKEY_CTX *hctx = ctx->pctx->data;
145         if (!HMAC_Update(&hctx->ctx, data, count))
146                 return 0;
147         return 1;
148         }
149
150 static int hmac_signctx_init(EVP_PKEY_CTX *ctx, EVP_MD_CTX *mctx)
151         {
152         HMAC_PKEY_CTX *hctx = ctx->data;
153         HMAC_CTX_set_flags(&hctx->ctx, mctx->flags & ~EVP_MD_CTX_FLAG_NO_INIT);
154         EVP_MD_CTX_set_flags(mctx, EVP_MD_CTX_FLAG_NO_INIT);
155         mctx->update = int_update;
156         return 1;
157         }
158
159 static int hmac_signctx(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
160                                         EVP_MD_CTX *mctx)
161         {
162         unsigned int hlen;
163         HMAC_PKEY_CTX *hctx = ctx->data;
164         int l = EVP_MD_CTX_size(mctx);
165
166         if (l < 0)
167                 return 0;
168         *siglen = l;
169         if (!sig)
170                 return 1;
171
172         if (!HMAC_Final(&hctx->ctx, sig, &hlen))
173                 return 0;
174         *siglen = (size_t)hlen;
175         return 1;
176         }
177
178 static int pkey_hmac_ctrl(EVP_PKEY_CTX *ctx, int type, int p1, void *p2)
179         {
180         HMAC_PKEY_CTX *hctx = ctx->data;
181         ASN1_OCTET_STRING *key;
182         switch (type)
183                 {
184
185                 case EVP_PKEY_CTRL_SET_MAC_KEY:
186                 if ((!p2 && p1 > 0) || (p1 < -1))
187                         return 0;
188                 if (!ASN1_OCTET_STRING_set(&hctx->ktmp, p2, p1))
189                         return 0;
190                 break;
191
192                 case EVP_PKEY_CTRL_MD:
193                 hctx->md = p2;
```



```
194                 break;
195
196                 case EVP_PKEY_CTRL_DIGESTINIT:
197                 key = (ASN1_OCTET_STRING *)ctx->pkey->pkey.ptr;
198                 if (!HMAC_Init_ex(&hctx->ctx, key->data, key->length, hctx->md,
199                                         ctx->engine))
200                         return 0;
201                 break;
202
203                 default:
204                 return -2;
205
206                 }
207         return 1;
208         }
209
210 static int pkey_hmac_ctrl_str(EVP_PKEY_CTX *ctx,
211                         const char *type, const char *value)
212         {
213         if (!value)
214                 {
215                 return 0;
216                 }
217         if (!strcmp(type, "key"))
218                 {
219                 void *p = (void *)value;
220                 return pkey_hmac_ctrl(ctx, EVP_PKEY_CTRL_SET_MAC_KEY,
221                                 -1, p);
222                 }
223         if (!strcmp(type, "hexkey"))
224                 {
225                 unsigned char *key;
226                 int r;
227                 long keylen;
228                 key = string_to_hex(value, &keylen);
229                 if (!key)
230                         return 0;
231                 r = pkey_hmac_ctrl(ctx, EVP_PKEY_CTRL_SET_MAC_KEY, keylen, key);
232                 OPENSSL_free(key);
233                 return r;
234                 }
235         return -2;
236         }
237
238 const EVP_PKEY_METHOD hmac_pkey_meth =
239         {
240         EVP_PKEY_HMAC,
241         0,
242         pkey_hmac_init,
243         pkey_hmac_copy,
244         pkey_hmac_cleanup,
245
246         0, 0,
247
248         0,
249         pkey_hmac_keygen,
250
251         0, 0,
252
253         0, 0,
254
255         0,0,
256
257         hmac_signctx_init,
258         hmac_signctx,
```

```
260         0,0,

262         0,0,

264         0,0,

266         0,0,

268         pkey_hmac_ctrl,
269         pkey_hmac_ctrl_str

271         };
272 #endif /* ! codereview */
```

```
**********************************************************
    7542 Wed Aug 13 19:52:52 2014
new/usr/src/lib/openssl/libsunw_crypto/hmac/hmac.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/hmac/hmac.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cryptlib.h"
```



```
#include <openssl/hmac.h>

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

int HMAC_Init_ex(HMAC_CTX *ctx, const void *key, int len,
                 const EVP_MD *md, ENGINE *impl)
        {
        int i,j,reset=0;
        unsigned char pad[HMAC_MAX_MD_CBLOCK];

#ifdef OPENSSL_FIPS
        if (FIPS_mode())
                {
                /* If we have an ENGINE need to allow non FIPS */
                if ((impl || ctx->i_ctx.engine)
                        &&  !(ctx->i_ctx.flags & EVP_CIPH_FLAG_NON_FIPS_ALLOW))
                        {
                        EVPerr(EVP_F_HMAC_INIT_EX, EVP_R_DISABLED_FOR_FIPS);
                        return 0;
                        }
                /* Other algorithm blocking will be done in FIPS_cmac_init,
                 * via FIPS_hmac_init_ex().
                 */
                if (!impl && !ctx->i_ctx.engine)
                        return FIPS_hmac_init_ex(ctx, key, len, md, NULL);
                }
#endif

        if (md != NULL)
                {
                reset=1;
                ctx->md=md;
                }
        else
                md=ctx->md;

        if (key != NULL)
                {
                reset=1;
                j=EVP_MD_block_size(md);
                OPENSSL_assert(j <= (int)sizeof(ctx->key));
                if (j < len)
                        {
                        if (!EVP_DigestInit_ex(&ctx->md_ctx,md, impl))
                                goto err;
                        if (!EVP_DigestUpdate(&ctx->md_ctx,key,len))
                                goto err;
                        if (!EVP_DigestFinal_ex(&(ctx->md_ctx),ctx->key,
                                &ctx->key_length))
                                goto err;
                        }
                else
                        {
                        OPENSSL_assert(len>=0 && len<=(int)sizeof(ctx->key));
                        memcpy(ctx->key,key,len);
                        ctx->key_length=len;
                        }
                if(ctx->key_length != HMAC_MAX_MD_CBLOCK)
                        memset(&ctx->key[ctx->key_length], 0,
                                HMAC_MAX_MD_CBLOCK - ctx->key_length);
                }

        if (reset)
                {
```

```
128                 for (i=0; i<HMAC_MAX_MD_CBLOCK; i++)
129                         pad[i]=0x36^ctx->key[i];
130                 if (!EVP_DigestInit_ex(&ctx->i_ctx,md, impl))
131                         goto err;
132                 if (!EVP_DigestUpdate(&ctx->i_ctx,pad,EVP_MD_block_size(md)))
133                         goto err;

135                 for (i=0; i<HMAC_MAX_MD_CBLOCK; i++)
136                         pad[i]=0x5c^ctx->key[i];
137                 if (!EVP_DigestInit_ex(&ctx->o_ctx,md, impl))
138                         goto err;
139                 if (!EVP_DigestUpdate(&ctx->o_ctx,pad,EVP_MD_block_size(md)))
140                         goto err;
141                 }
142         if (!EVP_MD_CTX_copy_ex(&ctx->md_ctx,&ctx->i_ctx))
143                         goto err;
144         return 1;
145         err:
146         return 0;
147         }

149 int HMAC_Init(HMAC_CTX *ctx, const void *key, int len, const EVP_MD *md)
150         {
151         if(key && md)
152             HMAC_CTX_init(ctx);
153         return HMAC_Init_ex(ctx,key,len,md, NULL);
154         }

156 int HMAC_Update(HMAC_CTX *ctx, const unsigned char *data, size_t len)
157         {
158 #ifdef OPENSSL_FIPS
159         if (FIPS_mode() && !ctx->i_ctx.engine)
160                 return FIPS_hmac_update(ctx, data, len);
161 #endif
162         return EVP_DigestUpdate(&ctx->md_ctx,data,len);
163         }

165 int HMAC_Final(HMAC_CTX *ctx, unsigned char *md, unsigned int *len)
166         {
167         unsigned int i;
168         unsigned char buf[EVP_MAX_MD_SIZE];
169 #ifdef OPENSSL_FIPS
170         if (FIPS_mode() && !ctx->i_ctx.engine)
171                 return FIPS_hmac_final(ctx, md, len);
172 #endif

174         if (!EVP_DigestFinal_ex(&ctx->md_ctx,buf,&i))
175                 goto err;
176         if (!EVP_MD_CTX_copy_ex(&ctx->md_ctx,&ctx->o_ctx))
177                 goto err;
178         if (!EVP_DigestUpdate(&ctx->md_ctx,buf,i))
179                 goto err;
180         if (!EVP_DigestFinal_ex(&ctx->md_ctx,md,len))
181                 goto err;
182         return 1;
183         err:
184         return 0;
185         }

187 void HMAC_CTX_init(HMAC_CTX *ctx)
188         {
189         EVP_MD_CTX_init(&ctx->i_ctx);
190         EVP_MD_CTX_init(&ctx->o_ctx);
191         EVP_MD_CTX_init(&ctx->md_ctx);
192         }
```



```
194 int HMAC_CTX_copy(HMAC_CTX *dctx, HMAC_CTX *sctx)
195         {
196         if (!EVP_MD_CTX_copy(&dctx->i_ctx, &sctx->i_ctx))
197                 goto err;
198         if (!EVP_MD_CTX_copy(&dctx->o_ctx, &sctx->o_ctx))
199                 goto err;
200         if (!EVP_MD_CTX_copy(&dctx->md_ctx, &sctx->md_ctx))
201                 goto err;
202         memcpy(dctx->key, sctx->key, HMAC_MAX_MD_CBLOCK);
203         dctx->key_length = sctx->key_length;
204         dctx->md = sctx->md;
205         return 1;
206         err:
207         return 0;
208         }

210 void HMAC_CTX_cleanup(HMAC_CTX *ctx)
211         {
212 #ifdef OPENSSL_FIPS
213         if (FIPS_mode() && !ctx->i_ctx.engine)
214                 {
215                 FIPS_hmac_ctx_cleanup(ctx);
216                 return;
217                 }
218 #endif
219         EVP_MD_CTX_cleanup(&ctx->i_ctx);
220         EVP_MD_CTX_cleanup(&ctx->o_ctx);
221         EVP_MD_CTX_cleanup(&ctx->md_ctx);
222         memset(ctx,0,sizeof *ctx);
223         }

225 unsigned char *HMAC(const EVP_MD *evp_md, const void *key, int key_len,
226                     const unsigned char *d, size_t n, unsigned char *md,
227                     unsigned int *md_len)
228         {
229         HMAC_CTX c;
230         static unsigned char m[EVP_MAX_MD_SIZE];

232         if (md == NULL) md=m;
233         HMAC_CTX_init(&c);
234         if (!HMAC_Init(&c,key,key_len,evp_md))
235                 goto err;
236         if (!HMAC_Update(&c,d,n))
237                 goto err;
238         if (!HMAC_Final(&c,md,md_len))
239                 goto err;
240         HMAC_CTX_cleanup(&c);
241         return md;
242         err:
243         return NULL;
244         }

246 void HMAC_CTX_set_flags(HMAC_CTX *ctx, unsigned long flags)
247         {
248         EVP_MD_CTX_set_flags(&ctx->i_ctx, flags);
249         EVP_MD_CTX_set_flags(&ctx->o_ctx, flags);
250         EVP_MD_CTX_set_flags(&ctx->md_ctx, flags);
251         }
252 #endif /* ! codereview */
```

```
**********************************************************
    1813 Wed Aug 13 19:52:52 2014
new/usr/src/lib/openssl/libsunw_crypto/i386/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2014 Alexander Pyhalov
# Use is subject to license terms.
#

# aes/*.s
ASM_SOURCES =   x86cpuid.s \
                aes-586.s \
                aesni-x86.s \
                vpaes-x86.s \
                bf-586.s \
                bn-586.s \
                co-586.s \
                x86-gf2m.s \
                x86-mont.s \
                cmll-x86.s \
                crypt586.s \
                des-586.s \
                md5-586.s \
                ghash-x86.s \
                rc4-586.s \
                rmd-586.s \
                sha1-586.s \
                sha256-586.s \
                sha512-586.s

OBJECTS +=      $(ASM_SOURCES:%.s=%.o)

CLEANFILES +=   $(ASM_SOURCES)

include ../Makefile.com

CPPFLAGS += -DL_ENDIAN
CPPFLAGS += -DOPENSSL_NO_INLINE_ASM
CPPFLAGS += -DOPENSSL_BN_ASM_PART_WORDS
CPPFLAGS += -DOPENSSL_IA32_SSE2
CPPFLAGS += -DRMD160_ASM
CPPFLAGS += -DAES_ASM
CPPFLAGS += -DPK11_LIB_LOCATION=\"/usr/lib/libpkcs11.so.1\"
```



```
CPPFLAGS += -DOPENSSL_IA32_SSE2
CPPFLAGS += -fPIC

.KEEP_STATE:

all:  $(ROOTLIBDIR) $(LIBS) $(LIBLINKS)

$(LIBLINKS):    FRC
        $(RM) $@; $(SYMLINK) $(DYNLIB) $@

$(ROOTLIBDIR):
        $(INS.dir)

install: all $(ROOTLIBS) $(ROOTLINKS)

FRC:

#endif /* ! codereview */
```

```
**********************************************************
    6201 Wed Aug 13 19:52:52 2014
new/usr/src/lib/openssl/libsunw_crypto/krb5/krb5_asn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* krb5_asn.c */
/* Written by Vern Staats <staatsvr@asc.hpc.mil> for the OpenSSL project,
** using ocsp/{*.h,*asn*.c} as a starting point
*/
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/krb5_asn.h>
```



```
ASN1_SEQUENCE(KRB5_ENCDATA) = {
        ASN1_EXP(KRB5_ENCDATA, etype,           ASN1_INTEGER,     0),
        ASN1_EXP_OPT(KRB5_ENCDATA, kvno,        ASN1_INTEGER,     1),
        ASN1_EXP(KRB5_ENCDATA, cipher,          ASN1_OCTET_STRING,2)
} ASN1_SEQUENCE_END(KRB5_ENCDATA)

IMPLEMENT_ASN1_FUNCTIONS(KRB5_ENCDATA)


ASN1_SEQUENCE(KRB5_PRINCNAME) = {
        ASN1_EXP(KRB5_PRINCNAME, nametype,      ASN1_INTEGER,     0),
        ASN1_EXP_SEQUENCE_OF(KRB5_PRINCNAME, namestring, ASN1_GENERALSTRING, 1)
} ASN1_SEQUENCE_END(KRB5_PRINCNAME)

IMPLEMENT_ASN1_FUNCTIONS(KRB5_PRINCNAME)


/* [APPLICATION 1] = 0x61 */
ASN1_SEQUENCE(KRB5_TKTBODY) = {
        ASN1_EXP(KRB5_TKTBODY, tktvno,          ASN1_INTEGER,     0),
        ASN1_EXP(KRB5_TKTBODY, realm,           ASN1_GENERALSTRING, 1),
        ASN1_EXP(KRB5_TKTBODY, sname,           KRB5_PRINCNAME,   2),
        ASN1_EXP(KRB5_TKTBODY, encdata,         KRB5_ENCDATA,     3)
} ASN1_SEQUENCE_END(KRB5_TKTBODY)

IMPLEMENT_ASN1_FUNCTIONS(KRB5_TKTBODY)


ASN1_ITEM_TEMPLATE(KRB5_TICKET) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_EXPTAG|ASN1_TFLG_APPLICATION, 1,
                        KRB5_TICKET, KRB5_TKTBODY)
ASN1_ITEM_TEMPLATE_END(KRB5_TICKET)

IMPLEMENT_ASN1_FUNCTIONS(KRB5_TICKET)


/* [APPLICATION 14] = 0x6e */
ASN1_SEQUENCE(KRB5_APREQBODY) = {
        ASN1_EXP(KRB5_APREQBODY, pvno,          ASN1_INTEGER,     0),
        ASN1_EXP(KRB5_APREQBODY, msgtype,       ASN1_INTEGER,     1),
        ASN1_EXP(KRB5_APREQBODY, apoptions,     ASN1_BIT_STRING,  2),
        ASN1_EXP(KRB5_APREQBODY, ticket,        KRB5_TICKET,      3),
        ASN1_EXP(KRB5_APREQBODY, authenticator, KRB5_ENCDATA,     4),
} ASN1_SEQUENCE_END(KRB5_APREQBODY)

IMPLEMENT_ASN1_FUNCTIONS(KRB5_APREQBODY)

ASN1_ITEM_TEMPLATE(KRB5_APREQ) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_EXPTAG|ASN1_TFLG_APPLICATION, 14,
                        KRB5_APREQ, KRB5_APREQBODY)
ASN1_ITEM_TEMPLATE_END(KRB5_APREQ)

IMPLEMENT_ASN1_FUNCTIONS(KRB5_APREQ)


/*  Authenticator stuff         */

ASN1_SEQUENCE(KRB5_CHECKSUM) = {
        ASN1_EXP(KRB5_CHECKSUM, ctype,          ASN1_INTEGER,     0),
        ASN1_EXP(KRB5_CHECKSUM, checksum,       ASN1_OCTET_STRING,1)
} ASN1_SEQUENCE_END(KRB5_CHECKSUM)

IMPLEMENT_ASN1_FUNCTIONS(KRB5_CHECKSUM)
```

```
128 ASN1_SEQUENCE(KRB5_ENCKEY) = {
129         ASN1_EXP(KRB5_ENCKEY,   ktype,          ASN1_INTEGER,     0),
130         ASN1_EXP(KRB5_ENCKEY,   keyvalue,       ASN1_OCTET_STRING,1)
131 } ASN1_SEQUENCE_END(KRB5_ENCKEY)

133 IMPLEMENT_ASN1_FUNCTIONS(KRB5_ENCKEY)


136 /* SEQ OF SEQ; see ASN1_EXP_SEQUENCE_OF_OPT() below */
137 ASN1_SEQUENCE(KRB5_AUTHDATA) = {
138         ASN1_EXP(KRB5_AUTHDATA, adtype,         ASN1_INTEGER,     0),
139         ASN1_EXP(KRB5_AUTHDATA, addata,         ASN1_OCTET_STRING,1)
140 } ASN1_SEQUENCE_END(KRB5_AUTHDATA)

142 IMPLEMENT_ASN1_FUNCTIONS(KRB5_AUTHDATA)


145 /* [APPLICATION 2] = 0x62 */
146 ASN1_SEQUENCE(KRB5_AUTHENTBODY) = {
147         ASN1_EXP(KRB5_AUTHENTBODY,      avno,   ASN1_INTEGER,     0),
148         ASN1_EXP(KRB5_AUTHENTBODY,      crealm, ASN1_GENERALSTRING, 1),
149         ASN1_EXP(KRB5_AUTHENTBODY,      cname,  KRB5_PRINCNAME,   2),
150         ASN1_EXP_OPT(KRB5_AUTHENTBODY,  cksum,  KRB5_CHECKSUM,    3),
151         ASN1_EXP(KRB5_AUTHENTBODY,      cusec,  ASN1_INTEGER,     4),
152         ASN1_EXP(KRB5_AUTHENTBODY,      ctime,  ASN1_GENERALIZEDTIME, 5),
153         ASN1_EXP_OPT(KRB5_AUTHENTBODY,  subkey, KRB5_ENCKEY,      6),
154         ASN1_EXP_OPT(KRB5_AUTHENTBODY,  seqnum, ASN1_INTEGER,     7),
155         ASN1_EXP_SEQUENCE_OF_OPT
156                     (KRB5_AUTHENTBODY,  authorization,  KRB5_AUTHDATA, 8),
157 } ASN1_SEQUENCE_END(KRB5_AUTHENTBODY)

159 IMPLEMENT_ASN1_FUNCTIONS(KRB5_AUTHENTBODY)

161 ASN1_ITEM_TEMPLATE(KRB5_AUTHENT) =
162         ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_EXPTAG|ASN1_TFLG_APPLICATION, 2,
163                         KRB5_AUTHENT, KRB5_AUTHENTBODY)
164 ASN1_ITEM_TEMPLATE_END(KRB5_AUTHENT)

166 IMPLEMENT_ASN1_FUNCTIONS(KRB5_AUTHENT)
167 #endif /* ! codereview */
```

```
**********************************************************
    8595 Wed Aug 13 19:52:52 2014
new/usr/src/lib/openssl/libsunw_crypto/lhash/lh_stats.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/lhash/lh_stats.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```



```
/* If you wish to build this outside of SSLeay, remove the following lines
 * and things should work as expected */
#include "cryptlib.h"

#ifndef OPENSSL_NO_BIO
#include <openssl/bio.h>
#endif
#include <openssl/lhash.h>

#ifdef OPENSSL_NO_BIO

void lh_stats(LHASH *lh, FILE *out)
        {
        fprintf(out,"num_items             = %lu\n",lh->num_items);
        fprintf(out,"num_nodes             = %u\n",lh->num_nodes);
        fprintf(out,"num_alloc_nodes       = %u\n",lh->num_alloc_nodes);
        fprintf(out,"num_expands           = %lu\n",lh->num_expands);
        fprintf(out,"num_expand_reallocs   = %lu\n",lh->num_expand_reallocs);
        fprintf(out,"num_contracts         = %lu\n",lh->num_contracts);
        fprintf(out,"num_contract_reallocs = %lu\n",lh->num_contract_reallocs);
        fprintf(out,"num_hash_calls        = %lu\n",lh->num_hash_calls);
        fprintf(out,"num_comp_calls        = %lu\n",lh->num_comp_calls);
        fprintf(out,"num_insert            = %lu\n",lh->num_insert);
        fprintf(out,"num_replace           = %lu\n",lh->num_replace);
        fprintf(out,"num_delete            = %lu\n",lh->num_delete);
        fprintf(out,"num_no_delete         = %lu\n",lh->num_no_delete);
        fprintf(out,"num_retrieve          = %lu\n",lh->num_retrieve);
        fprintf(out,"num_retrieve_miss     = %lu\n",lh->num_retrieve_miss);
        fprintf(out,"num_hash_comps        = %lu\n",lh->num_hash_comps);
#if 0
        fprintf(out,"p                     = %u\n",lh->p);
        fprintf(out,"pmax                  = %u\n",lh->pmax);
        fprintf(out,"up_load               = %lu\n",lh->up_load);
        fprintf(out,"down_load             = %lu\n",lh->down_load);
#endif
        }

void lh_node_stats(LHASH *lh, FILE *out)
        {
        LHASH_NODE *n;
        unsigned int i,num;

        for (i=0; i<lh->num_nodes; i++)
                {
                for (n=lh->b[i],num=0; n != NULL; n=n->next)
                        num++;
                fprintf(out,"node %6u -> %3u\n",i,num);
                }
        }

void lh_node_usage_stats(LHASH *lh, FILE *out)
        {
        LHASH_NODE *n;
        unsigned long num;
        unsigned int i;
        unsigned long total=0,n_used=0;

        for (i=0; i<lh->num_nodes; i++)
                {
                for (n=lh->b[i],num=0; n != NULL; n=n->next)
                        num++;
                if (num != 0)
                        {
                        n_used++;
                        total+=num;
                        }
```

```
128                 }
129         fprintf(out,"%lu nodes used out of %u\n",n_used,lh->num_nodes);
130         fprintf(out,"%lu items\n",total);
131         if (n_used == 0) return;
132         fprintf(out,"load %d.%02d  actual load %d.%02d\n",
133                 (int)(total/lh->num_nodes),
134                 (int)((total%lh->num_nodes)*100/lh->num_nodes),
135                 (int)(total/n_used),
136                 (int)((total%n_used)*100/n_used));
137         }

139 #else

141 #ifndef OPENSSL_NO_FP_API
142 void lh_stats(const _LHASH *lh, FILE *fp)
143         {
144         BIO *bp;

146         bp=BIO_new(BIO_s_file());
147         if (bp == NULL) goto end;
148         BIO_set_fp(bp,fp,BIO_NOCLOSE);
149         lh_stats_bio(lh,bp);
150         BIO_free(bp);
151 end:;
152         }

154 void lh_node_stats(const _LHASH *lh, FILE *fp)
155         {
156         BIO *bp;

158         bp=BIO_new(BIO_s_file());
159         if (bp == NULL) goto end;
160         BIO_set_fp(bp,fp,BIO_NOCLOSE);
161         lh_node_stats_bio(lh,bp);
162         BIO_free(bp);
163 end:;
164         }

166 void lh_node_usage_stats(const _LHASH *lh, FILE *fp)
167         {
168         BIO *bp;

170         bp=BIO_new(BIO_s_file());
171         if (bp == NULL) goto end;
172         BIO_set_fp(bp,fp,BIO_NOCLOSE);
173         lh_node_usage_stats_bio(lh,bp);
174         BIO_free(bp);
175 end:;
176         }

178 #endif

180 void lh_stats_bio(const _LHASH *lh, BIO *out)
181         {
182         BIO_printf(out,"num_items             = %lu\n",lh->num_items);
183         BIO_printf(out,"num_nodes             = %u\n",lh->num_nodes);
184         BIO_printf(out,"num_alloc_nodes       = %u\n",lh->num_alloc_nodes);
185         BIO_printf(out,"num_expands           = %lu\n",lh->num_expands);
186         BIO_printf(out,"num_expand_reallocs   = %lu\n",
187                    lh->num_expand_reallocs);
188         BIO_printf(out,"num_contracts         = %lu\n",lh->num_contracts);
189         BIO_printf(out,"num_contract_reallocs = %lu\n",
190                    lh->num_contract_reallocs);
191         BIO_printf(out,"num_hash_calls        = %lu\n",lh->num_hash_calls);
192         BIO_printf(out,"num_comp_calls        = %lu\n",lh->num_comp_calls);
193         BIO_printf(out,"num_insert            = %lu\n",lh->num_insert);
```



```
194         BIO_printf(out,"num_replace           = %lu\n",lh->num_replace);
195         BIO_printf(out,"num_delete            = %lu\n",lh->num_delete);
196         BIO_printf(out,"num_no_delete         = %lu\n",lh->num_no_delete);
197         BIO_printf(out,"num_retrieve          = %lu\n",lh->num_retrieve);
198         BIO_printf(out,"num_retrieve_miss     = %lu\n",lh->num_retrieve_miss);
199         BIO_printf(out,"num_hash_comps        = %lu\n",lh->num_hash_comps);
200 #if 0
201         BIO_printf(out,"p                     = %u\n",lh->p);
202         BIO_printf(out,"pmax                  = %u\n",lh->pmax);
203         BIO_printf(out,"up_load               = %lu\n",lh->up_load);
204         BIO_printf(out,"down_load             = %lu\n",lh->down_load);
205 #endif
206         }

208 void lh_node_stats_bio(const _LHASH *lh, BIO *out)
209         {
210         LHASH_NODE *n;
211         unsigned int i,num;

213         for (i=0; i<lh->num_nodes; i++)
214                 {
215                 for (n=lh->b[i],num=0; n != NULL; n=n->next)
216                         num++;
217                 BIO_printf(out,"node %6u -> %3u\n",i,num);
218                 }
219         }

221 void lh_node_usage_stats_bio(const _LHASH *lh, BIO *out)
222         {
223         LHASH_NODE *n;
224         unsigned long num;
225         unsigned int i;
226         unsigned long total=0,n_used=0;

228         for (i=0; i<lh->num_nodes; i++)
229                 {
230                 for (n=lh->b[i],num=0; n != NULL; n=n->next)
231                         num++;
232                 if (num != 0)
233                         {
234                         n_used++;
235                         total+=num;
236                         }
237                 }
238         BIO_printf(out,"%lu nodes used out of %u\n",n_used,lh->num_nodes);
239         BIO_printf(out,"%lu items\n",total);
240         if (n_used == 0) return;
241         BIO_printf(out,"load %d.%02d  actual load %d.%02d\n",
242                     (int)(total/lh->num_nodes),
243                     (int)((total%lh->num_nodes)*100/lh->num_nodes),
244                     (int)(total/n_used),
245                     (int)((total%n_used)*100/n_used));
246         }

248 #endif
249 #endif /* ! codereview */
```

```
**********************************************************
   11520 Wed Aug 13 19:52:52 2014
new/usr/src/lib/openssl/libsunw_crypto/lhash/lhash.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/lhash/lhash.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Code for dynamic hash table routines
 * Author - Eric Young v 2.0
 *
```



```
 * 2.2 eay - added #include "crypto.h" so the memory leak checking code is
 *           present. eay 18-Jun-98
 *
 * 2.1 eay - Added an 'error in last operation' flag. eay 6-May-98
 *
 * 2.0 eay - Fixed a bug that occurred when using lh_delete
 *           from inside lh_doall().  As entries were deleted,
 *           the 'table' was 'contract()ed', making some entries
 *           jump from the end of the table to the start, there by
 *           skipping the lh_doall() processing. eay - 4/12/95
 *
 * 1.9 eay - Fixed a memory leak in lh_free, the LHASH_NODEs
 *           were not being free()ed. 21/11/95
 *
 * 1.8 eay - Put the stats routines into a separate file, lh_stats.c
 *           19/09/95
 *
 * 1.7 eay - Removed the fputs() for realloc failures - the code
 *           should silently tolerate them.  I have also fixed things
 *           lint complained about 04/05/95
 *
 * 1.6 eay - Fixed an invalid pointers in contract/expand 27/07/92
 *
 * 1.5 eay - Fixed a misuse of realloc in expand 02/03/1992
 *
 * 1.4 eay - Fixed lh_doall so the function can call lh_delete 28/05/91
 *
 * 1.3 eay - Fixed a few lint problems 19/3/1991
 *
 * 1.2 eay - Fixed lh_doall problem 13/3/1991
 *
 * 1.1 eay - Added lh_doall
 *
 * 1.0 eay - First version
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <openssl/crypto.h>
#include <openssl/lhash.h>

const char lh_version[]="lhash" OPENSSL_VERSION_PTEXT;

#undef MIN_NODES
#define MIN_NODES       16
#define UP_LOAD         (2*LH_LOAD_MULT) /* load times 256  (default 2) */
#define DOWN_LOAD       (LH_LOAD_MULT)   /* load times 256  (default 1) */

static void expand(_LHASH *lh);
static void contract(_LHASH *lh);
static LHASH_NODE **getrn(_LHASH *lh, const void *data, unsigned long *rhash);

_LHASH *lh_new(LHASH_HASH_FN_TYPE h, LHASH_COMP_FN_TYPE c)
        {
        _LHASH *ret;
        int i;

        if ((ret=OPENSSL_malloc(sizeof(_LHASH))) == NULL)
                goto err0;
        if ((ret->b=OPENSSL_malloc(sizeof(LHASH_NODE *)*MIN_NODES)) == NULL)
                goto err1;
        for (i=0; i<MIN_NODES; i++)
                ret->b[i]=NULL;
        ret->comp=((c == NULL)?(LHASH_COMP_FN_TYPE)strcmp:c);
        ret->hash=((h == NULL)?(LHASH_HASH_FN_TYPE)lh_strhash:h);
        ret->num_nodes=MIN_NODES/2;
```

```
128         ret->num_alloc_nodes=MIN_NODES;
129         ret->p=0;
130         ret->pmax=MIN_NODES/2;
131         ret->up_load=UP_LOAD;
132         ret->down_load=DOWN_LOAD;
133         ret->num_items=0;

135         ret->num_expands=0;
136         ret->num_expand_reallocs=0;
137         ret->num_contracts=0;
138         ret->num_contract_reallocs=0;
139         ret->num_hash_calls=0;
140         ret->num_comp_calls=0;
141         ret->num_insert=0;
142         ret->num_replace=0;
143         ret->num_delete=0;
144         ret->num_no_delete=0;
145         ret->num_retrieve=0;
146         ret->num_retrieve_miss=0;
147         ret->num_hash_comps=0;

149         ret->error=0;
150         return(ret);
151 err1:
152         OPENSSL_free(ret);
153 err0:
154         return(NULL);
155         }

157 void lh_free(_LHASH *lh)
158         {
159         unsigned int i;
160         LHASH_NODE *n,*nn;

162         if (lh == NULL)
163             return;

165         for (i=0; i<lh->num_nodes; i++)
166                 {
167                 n=lh->b[i];
168                 while (n != NULL)
169                         {
170                         nn=n->next;
171                         OPENSSL_free(n);
172                         n=nn;
173                         }
174                 }
175         OPENSSL_free(lh->b);
176         OPENSSL_free(lh);
177         }

179 void *lh_insert(_LHASH *lh, void *data)
180         {
181         unsigned long hash;
182         LHASH_NODE *nn,**rn;
183         void *ret;

185         lh->error=0;
186         if (lh->up_load <= (lh->num_items*LH_LOAD_MULT/lh->num_nodes))
187                 expand(lh);

189         rn=getrn(lh,data,&hash);

191         if (*rn == NULL)
192                 {
193                 if ((nn=(LHASH_NODE *)OPENSSL_malloc(sizeof(LHASH_NODE))) == NUL
```



```
194                         {
195                         lh->error++;
196                         return(NULL);
197                         }
198                 nn->data=data;
199                 nn->next=NULL;
200 #ifndef OPENSSL_NO_HASH_COMP
201                 nn->hash=hash;
202 #endif
203                 *rn=nn;
204                 ret=NULL;
205                 lh->num_insert++;
206                 lh->num_items++;
207                 }
208         else /* replace same key */
209                 {
210                 ret= (*rn)->data;
211                 (*rn)->data=data;
212                 lh->num_replace++;
213                 }
214         return(ret);
215         }

217 void *lh_delete(_LHASH *lh, const void *data)
218         {
219         unsigned long hash;
220         LHASH_NODE *nn,**rn;
221         void *ret;

223         lh->error=0;
224         rn=getrn(lh,data,&hash);

226         if (*rn == NULL)
227                 {
228                 lh->num_no_delete++;
229                 return(NULL);
230                 }
231         else
232                 {
233                 nn= *rn;
234                 *rn=nn->next;
235                 ret=nn->data;
236                 OPENSSL_free(nn);
237                 lh->num_delete++;
238                 }

240         lh->num_items--;
241         if ((lh->num_nodes > MIN_NODES) &&
242                 (lh->down_load >= (lh->num_items*LH_LOAD_MULT/lh->num_nodes)))
243                 contract(lh);

245         return(ret);
246         }

248 void *lh_retrieve(_LHASH *lh, const void *data)
249         {
250         unsigned long hash;
251         LHASH_NODE **rn;
252         void *ret;

254         lh->error=0;
255         rn=getrn(lh,data,&hash);

257         if (*rn == NULL)
258                 {
259                 lh->num_retrieve_miss++;
```

```
260                 return(NULL);
261                 }
262         else
263                 {
264                 ret= (*rn)->data;
265                 lh->num_retrieve++;
266                 }
267         return(ret);
268         }

270 static void doall_util_fn(_LHASH *lh, int use_arg, LHASH_DOALL_FN_TYPE func,
271                           LHASH_DOALL_ARG_FN_TYPE func_arg, void *arg)
272         {
273         int i;
274         LHASH_NODE *a,*n;

276         if (lh == NULL)
277                 return;

279         /* reverse the order so we search from 'top to bottom'
280          * We were having memory leaks otherwise */
281         for (i=lh->num_nodes-1; i>=0; i--)
282                 {
283                 a=lh->b[i];
284                 while (a != NULL)
285                         {
286                         /* 28/05/91 - eay - n added so items can be deleted
287                          * via lh_doall */
288                         /* 22/05/08 - ben - eh? since a is not passed,
289                          * this should not be needed */
290                         n=a->next;
291                         if(use_arg)
292                                 func_arg(a->data,arg);
293                         else
294                                 func(a->data);
295                         a=n;
296                         }
297                 }
298         }

300 void lh_doall(_LHASH *lh, LHASH_DOALL_FN_TYPE func)
301         {
302         doall_util_fn(lh, 0, func, (LHASH_DOALL_ARG_FN_TYPE)0, NULL);
303         }

305 void lh_doall_arg(_LHASH *lh, LHASH_DOALL_ARG_FN_TYPE func, void *arg)
306         {
307         doall_util_fn(lh, 1, (LHASH_DOALL_FN_TYPE)0, func, arg);
308         }

310 static void expand(_LHASH *lh)
311         {
312         LHASH_NODE **n,**n1,**n2,*np;
313         unsigned int p,i,j;
314         unsigned long hash,nni;

316         lh->num_nodes++;
317         lh->num_expands++;
318         p=(int)lh->p++;
319         n1= &(lh->b[p]);
320         n2= &(lh->b[p+(int)lh->pmax]);
321         *n2=NULL;               /* 27/07/92 - eay - undefined pointer bug */
322         nni=lh->num_alloc_nodes;

324         for (np= *n1; np != NULL; )
325                 {
```



```
326 #ifndef OPENSSL_NO_HASH_COMP
327                 hash=np->hash;
328 #else
329                 hash=lh->hash(np->data);
330                 lh->num_hash_calls++;
331 #endif
332                 if ((hash%nni) != p)
333                         { /* move it */
334                         *n1= (*n1)->next;
335                         np->next= *n2;
336                         *n2=np;
337                         }
338                 else
339                         n1= &((*n1)->next);
340                 np= *n1;
341                 }

343         if ((lh->p) >= lh->pmax)
344                 {
345                 j=(int)lh->num_alloc_nodes*2;
346                 n=(LHASH_NODE **)OPENSSL_realloc(lh->b,
347                         (int)(sizeof(LHASH_NODE *)*j));
348                 if (n == NULL)
349                         {
350 /*                      fputs("realloc error in lhash",stderr); */
351                         lh->error++;
352                         lh->p=0;
353                         return;
354                         }
355                 /* else */
356                 for (i=(int)lh->num_alloc_nodes; i<j; i++)/* 26/02/92 eay */
357                         n[i]=NULL;                      /* 02/03/92 eay */
358                 lh->pmax=lh->num_alloc_nodes;
359                 lh->num_alloc_nodes=j;
360                 lh->num_expand_reallocs++;
361                 lh->p=0;
362                 lh->b=n;
363                 }
364         }

366 static void contract(_LHASH *lh)
367         {
368         LHASH_NODE **n,*n1,*np;

370         np=lh->b[lh->p+lh->pmax-1];
371         lh->b[lh->p+lh->pmax-1]=NULL; /* 24/07-92 - eay - weird but :-( */
372         if (lh->p == 0)
373                 {
374                 n=(LHASH_NODE **)OPENSSL_realloc(lh->b,
375                         (unsigned int)(sizeof(LHASH_NODE *)*lh->pmax));
376                 if (n == NULL)
377                         {
378 /*                      fputs("realloc error in lhash",stderr); */
379                         lh->error++;
380                         return;
381                         }
382                 lh->num_contract_reallocs++;
383                 lh->num_alloc_nodes/=2;
384                 lh->pmax/=2;
385                 lh->p=lh->pmax-1;
386                 lh->b=n;
387                 }
388         else
389                 lh->p--;

391         lh->num_nodes--;
```

```
392         lh->num_contracts++;

394         n1=lh->b[(int)lh->p];
395         if (n1 == NULL)
396                 lh->b[(int)lh->p]=np;
397         else
398                 {
399                 while (n1->next != NULL)
400                         n1=n1->next;
401                 n1->next=np;
402                 }
403         }

405 static LHASH_NODE **getrn(_LHASH *lh, const void *data, unsigned long *rhash)
406         {
407         LHASH_NODE **ret,*n1;
408         unsigned long hash,nn;
409         LHASH_COMP_FN_TYPE cf;

411         hash=(*(lh->hash))(data);
412         lh->num_hash_calls++;
413         *rhash=hash;

415         nn=hash%lh->pmax;
416         if (nn < lh->p)
417                 nn=hash%lh->num_alloc_nodes;

419         cf=lh->comp;
420         ret= &(lh->b[(int)nn]);
421         for (n1= *ret; n1 != NULL; n1=n1->next)
422                 {
423 #ifndef OPENSSL_NO_HASH_COMP
424                 lh->num_hash_comps++;
425                 if (n1->hash != hash)
426                         {
427                         ret= &(n1->next);
428                         continue;
429                         }
430 #endif
431                 lh->num_comp_calls++;
432                 if(cf(n1->data,data) == 0)
433                         break;
434                 ret= &(n1->next);
435                 }
436         return(ret);
437         }

439 /* The following hash seems to work very well on normal text strings
440  * no collisions on /usr/dict/words and it distributes on %2^n quite
441  * well, not as good as MD5, but still good.
442  */
443 unsigned long lh_strhash(const char *c)
444         {
445         unsigned long ret=0;
446         long n;
447         unsigned long v;
448         int r;

450         if ((c == NULL) || (*c == '\0'))
451                 return(ret);
452 /*
453         unsigned char b[16];
454         MD5(c,strlen(c),b);
455         return(b[0]|(b[1]<<8)|(b[2]<<16)|(b[3]<<24));
456 */
```



```
458         n=0x100;
459         while (*c)
460                 {
461                 v=n|(*c);
462                 n+=0x100;
463                 r= (int)((v>>2)^v)&0x0f;
464                 ret=(ret<<r)|(ret>>(32-r));
465                 ret&=0xFFFFFFFFL;
466                 ret^=v*v;
467                 c++;
468                 }
469         return((ret>>16)^ret);
470         }

472 unsigned long lh_num_items(const _LHASH *lh)
473         {
474         return lh ? lh->num_items : 0;
475         }
476 #endif /* ! codereview */
```

```
**********************************************************
   95893 Wed Aug 13 19:52:52 2014
new/usr/src/lib/openssl/libsunw_crypto/mapfile-vers
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
$mapfile_version 2

SYMBOL_VERSION ILLUMOS_0.1 {
    global:
        sunw__CONF_add_string;
        sunw__CONF_free_data;
        sunw__CONF_get_section;
        sunw__CONF_get_section_values;
        sunw__CONF_get_string;
        sunw__CONF_new_data;
        sunw__CONF_new_section;
        sunw__des_crypt;
        sunw__ossl_096_des_random_seed;
        sunw__ossl_old_crypt;
        sunw__ossl_old_des_cbc_cksum;
        sunw__ossl_old_des_cbc_encrypt;
        sunw__ossl_old_des_cfb_encrypt;
        sunw__ossl_old_des_cfb64_encrypt;
        sunw__ossl_old_des_crypt;
        sunw__ossl_old_des_decrypt3;
        sunw__ossl_old_des_ecb_encrypt;
        sunw__ossl_old_des_ecb3_encrypt;
        sunw__ossl_old_des_ede3_cbc_encrypt;
        sunw__ossl_old_des_ede3_cfb64_encrypt;
        sunw__ossl_old_des_ede3_ofb64_encrypt;
        sunw__ossl_old_des_enc_read;
        sunw__ossl_old_des_enc_write;
        sunw__ossl_old_des_encrypt;
        sunw__ossl_old_des_encrypt2;
        sunw__ossl_old_des_encrypt3;
        sunw__ossl_old_des_fcrypt;
        sunw__ossl_old_des_is_weak_key;
        sunw__ossl_old_des_key_sched;
        sunw__ossl_old_des_ncbc_encrypt;
        sunw__ossl_old_des_ofb_encrypt;
        sunw__ossl_old_des_ofb64_encrypt;
        sunw__ossl_old_des_options;
        sunw__ossl_old_des_pcbc_encrypt;
        sunw__ossl_old_des_quad_cksum;
        sunw__ossl_old_des_random_key;
        sunw__ossl_old_des_random_seed;
        sunw__ossl_old_des_read_2passwords;
        sunw__ossl_old_des_read_password;
        sunw__ossl_old_des_read_pw;
        sunw__ossl_old_des_read_pw_string;
        sunw__ossl_old_des_set_key;
        sunw__ossl_old_des_set_odd_parity;
        sunw__ossl_old_des_string_to_2keys;
        sunw__ossl_old_des_string_to_key;
        sunw__ossl_old_des_xcbc_encrypt;
        sunw__shadow_DES_rw_mode;
        sunw_a2d_ASN1_OBJECT;
        sunw_a2i_ASN1_ENUMERATED;
        sunw_a2i_ASN1_INTEGER;
        sunw_a2i_ASN1_STRING;
        sunw_a2i_GENERAL_NAME;
        sunw_a2i_ipadd;
        sunw_a2i_IPADDRESS;
        sunw_a2i_IPADDRESS_NC;
        sunw_ACCESS_DESCRIPTION_free;
        sunw_ACCESS_DESCRIPTION_it;
```



```
        sunw_ACCESS_DESCRIPTION_new;
        sunw_AES_bi_ige_encrypt;
        sunw_AES_cbc_encrypt;
        sunw_AES_cfb1_encrypt;
        sunw_AES_cfb128_encrypt;
        sunw_AES_cfb8_encrypt;
        sunw_AES_ctr128_encrypt;
        sunw_AES_decrypt;
        sunw_AES_ecb_encrypt;
        sunw_AES_encrypt;
        sunw_AES_ige_encrypt;
        sunw_AES_ofb128_encrypt;
        sunw_AES_options;
        sunw_AES_set_decrypt_key;
        sunw_AES_set_encrypt_key;
        sunw_AES_unwrap_key;
        sunw_AES_version;
        sunw_AES_wrap_key;
        sunw_aesni_cbc_encrypt;
        sunw_aesni_ccm64_decrypt_blocks;
        sunw_aesni_ccm64_encrypt_blocks;
        sunw_aesni_ctr32_encrypt_blocks;
        sunw_aesni_decrypt;
        sunw_aesni_ecb_encrypt;
        sunw_aesni_encrypt;
        sunw_aesni_set_decrypt_key;
        sunw_aesni_set_encrypt_key;
        sunw_aesni_xts_decrypt;
        sunw_aesni_xts_encrypt;
        sunw_asn1_add_error;
        sunw_ASN1_add_oid_module;
        sunw_ASN1_ANY_it;
        sunw_ASN1_BIT_STRING_check;
        sunw_ASN1_BIT_STRING_free;
        sunw_ASN1_BIT_STRING_get_bit;
        sunw_ASN1_BIT_STRING_it;
        sunw_ASN1_BIT_STRING_name_print;
        sunw_ASN1_BIT_STRING_new;
        sunw_ASN1_BIT_STRING_num_asc;
        sunw_ASN1_BIT_STRING_set;
        sunw_ASN1_BIT_STRING_set_asc;
        sunw_ASN1_BIT_STRING_set_bit;
        sunw_ASN1_BMPSTRING_free;
        sunw_ASN1_BMPSTRING_it;
        sunw_ASN1_BMPSTRING_new;
        sunw_ASN1_bn_print;
        sunw_ASN1_BOOLEAN_it;
        sunw_ASN1_check_infinite_end;
        sunw_ASN1_const_check_infinite_end;
        sunw_asn1_const_Finish;
        sunw_ASN1_d2i_bio;
        sunw_ASN1_d2i_fp;
        sunw_ASN1_digest;
        sunw_asn1_do_adb;
        sunw_asn1_do_lock;
        sunw_ASN1_dup;
        sunw_asn1_enc_free;
        sunw_asn1_enc_init;
        sunw_asn1_enc_restore;
        sunw_asn1_enc_save;
        sunw_ASN1_ENUMERATED_free;
        sunw_ASN1_ENUMERATED_get;
        sunw_ASN1_ENUMERATED_it;
        sunw_ASN1_ENUMERATED_new;
        sunw_ASN1_ENUMERATED_set;
        sunw_ASN1_ENUMERATED_to_BN;
```

```
128         sunw_asn1_ex_c2i;
129         sunw_asn1_ex_i2c;
130         sunw_ASN1_FBOOLEAN_it;
131         sunw_asn1_Finish;
132         sunw_ASN1_GENERALIZEDTIME_adj;
133         sunw_ASN1_GENERALIZEDTIME_check;
134         sunw_ASN1_GENERALIZEDTIME_free;
135         sunw_ASN1_GENERALIZEDTIME_it;
136         sunw_ASN1_GENERALIZEDTIME_new;
137         sunw_ASN1_GENERALIZEDTIME_print;
138         sunw_ASN1_GENERALIZEDTIME_set;
139         sunw_ASN1_GENERALIZEDTIME_set_string;
140         sunw_ASN1_GENERALSTRING_free;
141         sunw_ASN1_GENERALSTRING_it;
142         sunw_ASN1_GENERALSTRING_new;
143         sunw_ASN1_generate_nconf;
144         sunw_ASN1_generate_v3;
145         sunw_asn1_get_choice_selector;
146         sunw_asn1_get_field_ptr;
147         sunw_ASN1_get_object;
148         sunw_asn1_GetSequence;
149         sunw_ASN1_i2d_bio;
150         sunw_ASN1_i2d_fp;
151         sunw_ASN1_IA5STRING_free;
152         sunw_ASN1_IA5STRING_it;
153         sunw_ASN1_IA5STRING_new;
154         sunw_ASN1_INTEGER_cmp;
155         sunw_ASN1_INTEGER_dup;
156         sunw_ASN1_INTEGER_free;
157         sunw_ASN1_INTEGER_get;
158         sunw_ASN1_INTEGER_it;
159         sunw_ASN1_INTEGER_new;
160         sunw_ASN1_INTEGER_set;
161         sunw_ASN1_INTEGER_to_BN;
162         sunw_ASN1_item_d2i;
163         sunw_ASN1_item_d2i_bio;
164         sunw_ASN1_item_d2i_fp;
165         sunw_ASN1_item_digest;
166         sunw_ASN1_item_dup;
167         sunw_ASN1_item_ex_d2i;
168         sunw_ASN1_item_ex_free;
169         sunw_ASN1_item_ex_i2d;
170         sunw_ASN1_item_ex_new;
171         sunw_ASN1_item_free;
172         sunw_ASN1_item_i2d;
173         sunw_ASN1_item_i2d_bio;
174         sunw_ASN1_item_i2d_fp;
175         sunw_ASN1_item_ndef_i2d;
176         sunw_ASN1_item_new;
177         sunw_ASN1_item_pack;
178         sunw_ASN1_item_print;
179         sunw_ASN1_item_sign;
180         sunw_ASN1_item_sign_ctx;
181         sunw_ASN1_item_unpack;
182         sunw_ASN1_item_verify;
183         sunw_ASN1_mbstring_copy;
184         sunw_ASN1_mbstring_ncopy;
185         sunw_ASN1_NULL_free;
186         sunw_ASN1_NULL_it;
187         sunw_ASN1_NULL_new;
188         sunw_ASN1_OBJECT_create;
189         sunw_ASN1_OBJECT_free;
190         sunw_ASN1_OBJECT_it;
191         sunw_ASN1_OBJECT_new;
192         sunw_ASN1_object_size;
193         sunw_ASN1_OCTET_STRING_cmp;
```



```
194         sunw_ASN1_OCTET_STRING_dup;
195         sunw_ASN1_OCTET_STRING_free;
196         sunw_ASN1_OCTET_STRING_it;
197         sunw_ASN1_OCTET_STRING_NDEF_it;
198         sunw_ASN1_OCTET_STRING_new;
199         sunw_ASN1_OCTET_STRING_set;
200         sunw_ASN1_pack_string;
201         sunw_ASN1_parse;
202         sunw_ASN1_parse_dump;
203         sunw_ASN1_PCTX_free;
204         sunw_ASN1_PCTX_get_cert_flags;
205         sunw_ASN1_PCTX_get_flags;
206         sunw_ASN1_PCTX_get_nm_flags;
207         sunw_ASN1_PCTX_get_oid_flags;
208         sunw_ASN1_PCTX_get_str_flags;
209         sunw_ASN1_PCTX_new;
210         sunw_ASN1_PCTX_set_cert_flags;
211         sunw_ASN1_PCTX_set_flags;
212         sunw_ASN1_PCTX_set_nm_flags;
213         sunw_ASN1_PCTX_set_oid_flags;
214         sunw_ASN1_PCTX_set_str_flags;
215         sunw_ASN1_primitive_free;
216         sunw_ASN1_primitive_new;
217         sunw_ASN1_PRINTABLE_free;
218         sunw_ASN1_PRINTABLE_it;
219         sunw_ASN1_PRINTABLE_new;
220         sunw_ASN1_PRINTABLE_type;
221         sunw_ASN1_PRINTABLESTRING_free;
222         sunw_ASN1_PRINTABLESTRING_it;
223         sunw_ASN1_PRINTABLESTRING_new;
224         sunw_ASN1_put_eoc;
225         sunw_ASN1_put_object;
226         sunw_ASN1_seq_pack;
227         sunw_ASN1_seq_unpack;
228         sunw_ASN1_SEQUENCE_ANY_it;
229         sunw_ASN1_SEQUENCE_it;
230         sunw_ASN1_SET_ANY_it;
231         sunw_asn1_set_choice_selector;
232         sunw_ASN1_sign;
233         sunw_ASN1_STRING_cmp;
234         sunw_ASN1_STRING_copy;
235         sunw_ASN1_STRING_data;
236         sunw_ASN1_STRING_dup;
237         sunw_ASN1_STRING_free;
238         sunw_ASN1_STRING_get_default_mask;
239         sunw_ASN1_STRING_length;
240         sunw_ASN1_STRING_length_set;
241         sunw_ASN1_STRING_new;
242         sunw_ASN1_STRING_print;
243         sunw_ASN1_STRING_print_ex;
244         sunw_ASN1_STRING_print_ex_fp;
245         sunw_ASN1_STRING_set;
246         sunw_ASN1_STRING_set_by_NID;
247         sunw_ASN1_STRING_set_default_mask;
248         sunw_ASN1_STRING_set_default_mask_asc;
249         sunw_ASN1_STRING_set0;
250         sunw_ASN1_STRING_TABLE_add;
251         sunw_ASN1_STRING_TABLE_cleanup;
252         sunw_ASN1_STRING_TABLE_get;
253         sunw_ASN1_STRING_to_UTF8;
254         sunw_ASN1_STRING_type;
255         sunw_ASN1_STRING_type_new;
256         sunw_ASN1_T61STRING_free;
257         sunw_ASN1_T61STRING_it;
258         sunw_ASN1_T61STRING_new;
259         sunw_ASN1_tag2bit;
```

```
260         sunw_ASN1_tag2str;
261         sunw_ASN1_TBOOLEAN_it;
262         sunw_ASN1_template_d2i;
263         sunw_ASN1_template_free;
264         sunw_ASN1_template_i2d;
265         sunw_ASN1_template_new;
266         sunw_asn1_template_print_ctx;
267         sunw_ASN1_TIME_adj;
268         sunw_ASN1_TIME_check;
269         sunw_ASN1_TIME_free;
270         sunw_ASN1_TIME_it;
271         sunw_ASN1_TIME_new;
272         sunw_ASN1_TIME_print;
273         sunw_ASN1_TIME_set;
274         sunw_ASN1_TIME_set_string;
275         sunw_ASN1_TIME_to_generalizedtime;
276         sunw_ASN1_TYPE_cmp;
277         sunw_ASN1_TYPE_free;
278         sunw_ASN1_TYPE_get;
279         sunw_ASN1_TYPE_get_int_octetstring;
280         sunw_ASN1_TYPE_get_octetstring;
281         sunw_ASN1_TYPE_new;
282         sunw_ASN1_TYPE_set;
283         sunw_ASN1_TYPE_set_int_octetstring;
284         sunw_ASN1_TYPE_set_octetstring;
285         sunw_ASN1_TYPE_set1;
286         sunw_ASN1_UNIVERSALSTRING_free;
287         sunw_ASN1_UNIVERSALSTRING_it;
288         sunw_ASN1_UNIVERSALSTRING_new;
289         sunw_ASN1_UNIVERSALSTRING_to_string;
290         sunw_ASN1_unpack_string;
291         sunw_ASN1_UTCTIME_adj;
292         sunw_ASN1_UTCTIME_check;
293         sunw_ASN1_UTCTIME_cmp_time_t;
294         sunw_ASN1_UTCTIME_free;
295         sunw_ASN1_UTCTIME_it;
296         sunw_ASN1_UTCTIME_new;
297         sunw_ASN1_UTCTIME_print;
298         sunw_ASN1_UTCTIME_set;
299         sunw_ASN1_UTCTIME_set_string;
300         sunw_ASN1_UTF8STRING_free;
301         sunw_ASN1_UTF8STRING_it;
302         sunw_ASN1_UTF8STRING_new;
303         sunw_ASN1_verify;
304         sunw_ASN1_version;
305         sunw_ASN1_VISIBLESTRING_free;
306         sunw_ASN1_VISIBLESTRING_it;
307         sunw_ASN1_VISIBLESTRING_new;
308         sunw_AUTHORITY_INFO_ACCESS_free;
309         sunw_AUTHORITY_INFO_ACCESS_it;
310         sunw_AUTHORITY_INFO_ACCESS_new;
311         sunw_AUTHORITY_KEYID_free;
312         sunw_AUTHORITY_KEYID_it;
313         sunw_AUTHORITY_KEYID_new;
314         sunw_b2i_PrivateKey;
315         sunw_b2i_PrivateKey_bio;
316         sunw_b2i_PublicKey;
317         sunw_b2i_PublicKey_bio;
318         sunw_b2i_PVK_bio;
319         sunw_BASIC_CONSTRAINTS_free;
320         sunw_BASIC_CONSTRAINTS_it;
321         sunw_BASIC_CONSTRAINTS_new;
322         sunw_BF_cbc_encrypt;
323         sunw_BF_cfb64_encrypt;
324         sunw_BF_decrypt;
325         sunw_BF_ecb_encrypt;
```



```
326         sunw_BF_encrypt;
327         sunw_BF_ofb64_encrypt;
328         sunw_BF_options;
329         sunw_BF_set_key;
330         sunw_BF_version;
331         sunw_BIGNUM_it;
332         sunw_BIO_accept;
333         sunw_BIO_asn1_get_prefix;
334         sunw_BIO_asn1_get_suffix;
335         sunw_BIO_asn1_set_prefix;
336         sunw_BIO_asn1_set_suffix;
337         sunw_BIO_callback_ctrl;
338         sunw_BIO_clear_flags;
339         sunw_BIO_CONNECT_free;
340         sunw_BIO_CONNECT_new;
341         sunw_BIO_copy_next_retry;
342         sunw_BIO_ctrl;
343         sunw_BIO_ctrl_get_read_request;
344         sunw_BIO_ctrl_get_write_guarantee;
345         sunw_BIO_ctrl_pending;
346         sunw_BIO_ctrl_reset_read_request;
347         sunw_BIO_ctrl_wpending;
348         sunw_BIO_debug_callback;
349         sunw_BIO_dgram_non_fatal_error;
350         sunw_BIO_dump;
351         sunw_BIO_dump_cb;
352         sunw_BIO_dump_fp;
353         sunw_BIO_dump_indent;
354         sunw_BIO_dump_indent_cb;
355         sunw_BIO_dump_indent_fp;
356         sunw_BIO_dup_chain;
357         sunw_BIO_f_asn1;
358         sunw_BIO_f_base64;
359         sunw_BIO_f_buffer;
360         sunw_BIO_f_cipher;
361         sunw_BIO_f_md;
362         sunw_BIO_f_nbio_test;
363         sunw_BIO_f_null;
364         sunw_BIO_f_reliable;
365         sunw_BIO_fd_non_fatal_error;
366         sunw_BIO_fd_should_retry;
367         sunw_BIO_find_type;
368         sunw_BIO_free;
369         sunw_BIO_free_all;
370         sunw_BIO_get_accept_socket;
371         sunw_BIO_get_callback;
372         sunw_BIO_get_callback_arg;
373         sunw_BIO_get_ex_data;
374         sunw_BIO_get_ex_new_index;
375         sunw_BIO_get_host_ip;
376         sunw_BIO_get_port;
377         sunw_BIO_get_retry_BIO;
378         sunw_BIO_get_retry_reason;
379         sunw_BIO_gethostbyname;
380         sunw_BIO_gets;
381         sunw_BIO_indent;
382         sunw_BIO_int_ctrl;
383         sunw_BIO_method_name;
384         sunw_BIO_method_type;
385         sunw_BIO_new;
386         sunw_BIO_new_accept;
387         sunw_BIO_new_bio_pair;
388         sunw_BIO_new_CMS;
389         sunw_BIO_new_connect;
390         sunw_BIO_new_dgram;
391         sunw_BIO_new_fd;
```

```
392         sunw_BIO_new_file;
393         sunw_BIO_new_fp;
394         sunw_BIO_new_mem_buf;
395         sunw_BIO_new_NDEF;
396         sunw_BIO_new_PKCS7;
397         sunw_BIO_new_socket;
398         sunw_BIO_next;
399         sunw_BIO_nread;
400         sunw_BIO_nread0;
401         sunw_BIO_number_read;
402         sunw_BIO_number_written;
403         sunw_BIO_nwrite;
404         sunw_BIO_nwrite0;
405         sunw_BIO_pop;
406         sunw_BIO_printf;
407         sunw_BIO_ptr_ctrl;
408         sunw_BIO_push;
409         sunw_BIO_puts;
410         sunw_BIO_read;
411         sunw_BIO_s_accept;
412         sunw_BIO_s_bio;
413         sunw_BIO_s_connect;
414         sunw_BIO_s_datagram;
415         sunw_BIO_s_fd;
416         sunw_BIO_s_file;
417         sunw_BIO_s_log;
418         sunw_BIO_s_mem;
419         sunw_BIO_s_null;
420         sunw_BIO_s_socket;
421         sunw_BIO_set;
422         sunw_BIO_set_callback;
423         sunw_BIO_set_callback_arg;
424         sunw_BIO_set_cipher;
425         sunw_BIO_set_ex_data;
426         sunw_BIO_set_flags;
427         sunw_BIO_set_tcp_ndelay;
428         sunw_BIO_snprintf;
429         sunw_BIO_sock_cleanup;
430         sunw_BIO_sock_error;
431         sunw_BIO_sock_init;
432         sunw_BIO_sock_non_fatal_error;
433         sunw_BIO_sock_should_retry;
434         sunw_BIO_socket_ioctl;
435         sunw_BIO_socket_nbio;
436         sunw_BIO_test_flags;
437         sunw_BIO_vfree;
438         sunw_BIO_vprintf;
439         sunw_BIO_vsnprintf;
440         sunw_BIO_write;
441         sunw_BN_add;
442         sunw_bn_add_part_words;
443         sunw_BN_add_word;
444         sunw_bn_add_words;
445         sunw_BN_asc2bn;
446         sunw_BN_bin2bn;
447         sunw_BN_BLINDING_convert;
448         sunw_BN_BLINDING_convert_ex;
449         sunw_BN_BLINDING_create_param;
450         sunw_BN_BLINDING_free;
451         sunw_BN_BLINDING_get_flags;
452         sunw_BN_BLINDING_get_thread_id;
453         sunw_BN_BLINDING_invert;
454         sunw_BN_BLINDING_invert_ex;
455         sunw_BN_BLINDING_new;
456         sunw_BN_BLINDING_set_flags;
457         sunw_BN_BLINDING_set_thread_id;
```



```
458         sunw_BN_BLINDING_thread_id;
459         sunw_BN_BLINDING_update;
460         sunw_BN_bn2bin;
461         sunw_BN_bn2dec;
462         sunw_BN_bn2hex;
463         sunw_BN_bn2mpi;
464         sunw_BN_bntest_rand;
465         sunw_BN_clear;
466         sunw_BN_clear_bit;
467         sunw_BN_clear_free;
468         sunw_BN_cmp;
469         sunw_bn_cmp_part_words;
470         sunw_bn_cmp_words;
471         sunw_BN_consttime_swap;
472         sunw_BN_copy;
473         sunw_BN_CTX_end;
474         sunw_BN_CTX_free;
475         sunw_BN_CTX_get;
476         sunw_BN_CTX_init;
477         sunw_BN_CTX_new;
478         sunw_BN_CTX_start;
479         sunw_BN_dec2bn;
480         sunw_BN_div;
481         sunw_BN_div_recp;
482         sunw_BN_div_word;
483         sunw_bn_div_words;
484         sunw_BN_dup;
485         sunw_bn_dup_expand;
486         sunw_BN_exp;
487         sunw_bn_expand2;
488         sunw_BN_free;
489         sunw_BN_from_montgomery;
490         sunw_BN_gcd;
491         sunw_BN_GENCB_call;
492         sunw_BN_generate_prime;
493         sunw_BN_generate_prime_ex;
494         sunw_BN_get_params;
495         sunw_BN_get_word;
496         sunw_BN_get0_nist_prime_192;
497         sunw_BN_get0_nist_prime_224;
498         sunw_BN_get0_nist_prime_256;
499         sunw_BN_get0_nist_prime_384;
500         sunw_BN_get0_nist_prime_521;
501         sunw_BN_GF2m_add;
502         sunw_BN_GF2m_arr2poly;
503         sunw_BN_GF2m_mod;
504         sunw_BN_GF2m_mod_arr;
505         sunw_BN_GF2m_mod_div;
506         sunw_BN_GF2m_mod_div_arr;
507         sunw_BN_GF2m_mod_exp;
508         sunw_BN_GF2m_mod_exp_arr;
509         sunw_BN_GF2m_mod_inv;
510         sunw_BN_GF2m_mod_inv_arr;
511         sunw_BN_GF2m_mod_mul;
512         sunw_BN_GF2m_mod_mul_arr;
513         sunw_BN_GF2m_mod_solve_quad;
514         sunw_BN_GF2m_mod_solve_quad_arr;
515         sunw_BN_GF2m_mod_sqr;
516         sunw_BN_GF2m_mod_sqr_arr;
517         sunw_BN_GF2m_mod_sqrt;
518         sunw_BN_GF2m_mod_sqrt_arr;
519         sunw_bn_GF2m_mul_2x2;
520         sunw_BN_GF2m_poly2arr;
521         sunw_BN_hex2bn;
522         sunw_BN_init;
523         sunw_BN_is_bit_set;
```

```
524         sunw_BN_is_prime;
525         sunw_BN_is_prime_ex;
526         sunw_BN_is_prime_fasttest;
527         sunw_BN_is_prime_fasttest_ex;
528         sunw_BN_kronecker;
529         sunw_BN_lshift;
530         sunw_BN_lshift1;
531         sunw_BN_mask_bits;
532         sunw_BN_mod_add;
533         sunw_BN_mod_add_quick;
534         sunw_BN_mod_exp;
535         sunw_BN_mod_exp_mont;
536         sunw_BN_mod_exp_mont_consttime;
537         sunw_BN_mod_exp_mont_word;
538         sunw_BN_mod_exp_recp;
539         sunw_BN_mod_exp_simple;
540         sunw_BN_mod_exp2_mont;
541         sunw_BN_mod_inverse;
542         sunw_BN_mod_lshift;
543         sunw_BN_mod_lshift_quick;
544         sunw_BN_mod_lshift1;
545         sunw_BN_mod_lshift1_quick;
546         sunw_BN_mod_mul;
547         sunw_BN_mod_mul_montgomery;
548         sunw_BN_mod_mul_reciprocal;
549         sunw_BN_mod_sqr;
550         sunw_BN_mod_sqrt;
551         sunw_BN_mod_sub;
552         sunw_BN_mod_sub_quick;
553         sunw_BN_mod_word;
554         sunw_BN_MONT_CTX_copy;
555         sunw_BN_MONT_CTX_free;
556         sunw_BN_MONT_CTX_init;
557         sunw_BN_MONT_CTX_new;
558         sunw_BN_MONT_CTX_set;
559         sunw_BN_MONT_CTX_set_locked;
560         sunw_BN_mpi2bn;
561         sunw_BN_mul;
562         sunw_bn_mul_add_words;
563         sunw_bn_mul_comba4;
564         sunw_bn_mul_comba8;
565         sunw_bn_mul_high;
566         sunw_bn_mul_low_normal;
567         sunw_bn_mul_low_recursive;
568         sunw_bn_mul_mont;
569         sunw_bn_mul_normal;
570         sunw_bn_mul_part_recursive;
571         sunw_bn_mul_recursive;
572         sunw_BN_mul_word;
573         sunw_bn_mul_words;
574         sunw_BN_new;
575         sunw_BN_nist_mod_192;
576         sunw_BN_nist_mod_224;
577         sunw_BN_nist_mod_256;
578         sunw_BN_nist_mod_384;
579         sunw_BN_nist_mod_521;
580         sunw_BN_nnmod;
581         sunw_BN_num_bits;
582         sunw_BN_num_bits_word;
583         sunw_BN_options;
584         sunw_BN_print;
585         sunw_BN_print_fp;
586         sunw_BN_pseudo_rand;
587         sunw_BN_pseudo_rand_range;
588         sunw_BN_rand;
589         sunw_BN_rand_range;
```



```
590         sunw_BN_reciprocal;
591         sunw_BN_RECP_CTX_free;
592         sunw_BN_RECP_CTX_init;
593         sunw_BN_RECP_CTX_new;
594         sunw_BN_RECP_CTX_set;
595         sunw_BN_rshift;
596         sunw_BN_rshift1;
597         sunw_BN_set_bit;
598         sunw_BN_set_negative;
599         sunw_BN_set_params;
600         sunw_BN_set_word;
601         sunw_BN_sqr;
602         sunw_bn_sqr_comba4;
603         sunw_bn_sqr_comba8;
604         sunw_bn_sqr_normal;
605         sunw_bn_sqr_recursive;
606         sunw_bn_sqr_words;
607         sunw_BN_sub;
608         sunw_bn_sub_part_words;
609         sunw_BN_sub_word;
610         sunw_bn_sub_words;
611         sunw_BN_swap;
612         sunw_BN_to_ASN1_ENUMERATED;
613         sunw_BN_to_ASN1_INTEGER;
614         sunw_BN_uadd;
615         sunw_BN_ucmp;
616         sunw_BN_usub;
617         sunw_BN_value_one;
618         sunw_BN_version;
619         sunw_BN_X931_derive_prime_ex;
620         sunw_BN_X931_generate_prime_ex;
621         sunw_BN_X931_generate_Xpq;
622         sunw_BUF_MEM_free;
623         sunw_BUF_MEM_grow;
624         sunw_BUF_MEM_grow_clean;
625         sunw_BUF_MEM_new;
626         sunw_BUF_memdup;
627         sunw_BUF_reverse;
628         sunw_BUF_strdup;
629         sunw_BUF_strlcat;
630         sunw_BUF_strlcpy;
631         sunw_BUF_strndup;
632         sunw_c2i_ASN1_BIT_STRING;
633         sunw_c2i_ASN1_INTEGER;
634         sunw_c2i_ASN1_OBJECT;
635         sunw_Camellia_cbc_encrypt;
636         sunw_Camellia_cfb1_encrypt;
637         sunw_Camellia_cfb128_encrypt;
638         sunw_Camellia_cfb8_encrypt;
639         sunw_Camellia_ctr128_encrypt;
640         sunw_Camellia_decrypt;
641         sunw_Camellia_DecryptBlock;
642         sunw_Camellia_DecryptBlock_Rounds;
643         sunw_Camellia_ecb_encrypt;
644         sunw_Camellia_Ekeygen;
645         sunw_Camellia_encrypt;
646         sunw_Camellia_EncryptBlock;
647         sunw_Camellia_EncryptBlock_Rounds;
648         sunw_Camellia_ofb128_encrypt;
649         sunw_Camellia_set_key;
650         sunw_CAST_cbc_encrypt;
651         sunw_CAST_cfb64_encrypt;
652         sunw_CAST_decrypt;
653         sunw_CAST_ecb_encrypt;
654         sunw_CAST_encrypt;
655         sunw_CAST_ofb64_encrypt;
```

```
656         sunw_CAST_S_table0;
657         sunw_CAST_S_table1;
658         sunw_CAST_S_table2;
659         sunw_CAST_S_table3;
660         sunw_CAST_S_table4;
661         sunw_CAST_S_table5;
662         sunw_CAST_S_table6;
663         sunw_CAST_S_table7;
664         sunw_CAST_set_key;
665         sunw_CAST_version;
666         sunw_CBIGNUM_it;
667         sunw_CERTIFICATEPOLICIES_free;
668         sunw_CERTIFICATEPOLICIES_it;
669         sunw_CERTIFICATEPOLICIES_new;
670         sunw_check_defer;
671         sunw_cmac_asn1_meth;
672         sunw_CMAC_CTX_cleanup;
673         sunw_CMAC_CTX_copy;
674         sunw_CMAC_CTX_free;
675         sunw_CMAC_CTX_get0_cipher_ctx;
676         sunw_CMAC_CTX_new;
677         sunw_CMAC_Final;
678         sunw_CMAC_Init;
679         sunw_cmac_pkey_meth;
680         sunw_CMAC_resume;
681         sunw_CMAC_Update;
682         sunw_CMS_add_simple_smimecap;
683         sunw_CMS_add_smimecap;
684         sunw_CMS_add_standard_smimecap;
685         sunw_CMS_add0_cert;
686         sunw_CMS_add0_CertificateChoices;
687         sunw_CMS_add0_crl;
688         sunw_CMS_add0_recipient_key;
689         sunw_CMS_add0_recipient_password;
690         sunw_CMS_add0_RevocationInfoChoice;
691         sunw_CMS_add1_cert;
692         sunw_CMS_add1_crl;
693         sunw_CMS_add1_ReceiptRequest;
694         sunw_CMS_add1_recipient_cert;
695         sunw_CMS_add1_signer;
696         sunw_CMS_Attributes_Sign_it;
697         sunw_CMS_Attributes_Verify_it;
698         sunw_CMS_AuthenticatedData_it;
699         sunw_CMS_CertificateChoices_it;
700         sunw_CMS_compress;
701         sunw_CMS_CompressedData_it;
702         sunw_cms_content_bio;
703         sunw_CMS_ContentInfo_free;
704         sunw_CMS_ContentInfo_it;
705         sunw_CMS_ContentInfo_new;
706         sunw_CMS_ContentInfo_print_ctx;
707         sunw_CMS_data;
708         sunw_cms_Data_create;
709         sunw_CMS_data_create;
710         sunw_CMS_dataFinal;
711         sunw_CMS_dataInit;
712         sunw_CMS_decrypt;
713         sunw_CMS_decrypt_set1_key;
714         sunw_CMS_decrypt_set1_password;
715         sunw_CMS_decrypt_set1_pkey;
716         sunw_CMS_digest_create;
717         sunw_CMS_digest_verify;
718         sunw_cms_DigestAlgorithm_find_ctx;
719         sunw_cms_DigestAlgorithm_init_bio;
720         sunw_cms_DigestAlgorithm_set;
721         sunw_cms_DigestedData_create;
```



```
722         sunw_cms_DigestedData_do_final;
723         sunw_cms_DigestedData_init_bio;
724         sunw_CMS_DigestedData_it;
725         sunw_CMS_EncapsulatedContentInfo_it;
726         sunw_cms_encode_Receipt;
727         sunw_CMS_encrypt;
728         sunw_cms_EncryptedContent_init;
729         sunw_cms_EncryptedContent_init_bio;
730         sunw_CMS_EncryptedContentInfo_it;
731         sunw_CMS_EncryptedData_decrypt;
732         sunw_CMS_EncryptedData_encrypt;
733         sunw_cms_EncryptedData_init_bio;
734         sunw_CMS_EncryptedData_it;
735         sunw_CMS_EncryptedData_set1_key;
736         sunw_CMS_EnvelopedData_create;
737         sunw_cms_EnvelopedData_init_bio;
738         sunw_CMS_EnvelopedData_it;
739         sunw_CMS_final;
740         sunw_CMS_get0_content;
741         sunw_CMS_get0_eContentType;
742         sunw_cms_get0_enveloped;
743         sunw_CMS_get0_RecipientInfos;
744         sunw_CMS_get0_SignerInfos;
745         sunw_CMS_get0_signers;
746         sunw_CMS_get0_type;
747         sunw_CMS_get1_certs;
748         sunw_CMS_get1_crls;
749         sunw_CMS_get1_ReceiptRequest;
750         sunw_CMS_is_detached;
751         sunw_CMS_IssuerAndSerialNumber_it;
752         sunw_CMS_KEKIdentifier_it;
753         sunw_CMS_KEKRecipientInfo_it;
754         sunw_CMS_KeyAgreeRecipientIdentifier_it;
755         sunw_CMS_KeyAgreeRecipientInfo_it;
756         sunw_CMS_KeyTransRecipientInfo_it;
757         sunw_cms_msgSigDigest_add1;
758         sunw_CMS_OriginatorIdentifierOrKey_it;
759         sunw_CMS_OriginatorInfo_it;
760         sunw_CMS_OriginatorPublicKey_it;
761         sunw_CMS_OtherCertificateFormat_it;
762         sunw_CMS_OtherKeyAttribute_it;
763         sunw_CMS_OtherRecipientInfo_it;
764         sunw_CMS_OtherRevocationInfoFormat_it;
765         sunw_CMS_PasswordRecipientInfo_it;
766         sunw_CMS_Receipt_it;
767         sunw_cms_Receipt_verify;
768         sunw_CMS_ReceiptRequest_create0;
769         sunw_CMS_ReceiptRequest_free;
770         sunw_CMS_ReceiptRequest_get0_values;
771         sunw_CMS_ReceiptRequest_it;
772         sunw_CMS_ReceiptRequest_new;
773         sunw_CMS_ReceiptsFrom_it;
774         sunw_CMS_RecipientEncryptedKey_it;
775         sunw_CMS_RecipientInfo_decrypt;
776         sunw_CMS_RecipientInfo_it;
777         sunw_CMS_RecipientInfo_kekri_get0_id;
778         sunw_CMS_RecipientInfo_kekri_id_cmp;
779         sunw_CMS_RecipientInfo_ktri_cert_cmp;
780         sunw_CMS_RecipientInfo_ktri_get0_algs;
781         sunw_CMS_RecipientInfo_ktri_get0_signer_id;
782         sunw_cms_RecipientInfo_pwri_crypt;
783         sunw_CMS_RecipientInfo_set0_key;
784         sunw_CMS_RecipientInfo_set0_password;
785         sunw_CMS_RecipientInfo_set0_pkey;
786         sunw_CMS_RecipientInfo_type;
787         sunw_CMS_RecipientKeyIdentifier_it;
```

```
788         sunw_CMS_RevocationInfoChoice_it;
789         sunw_CMS_set_detached;
790         sunw_CMS_set1_eContentType;
791         sunw_cms_set1_SignerIdentifier;
792         sunw_CMS_set1_signers_certs;
793         sunw_CMS_sign;
794         sunw_CMS_sign_receipt;
795         sunw_CMS_signed_add1_attr;
796         sunw_CMS_signed_add1_attr_by_NID;
797         sunw_CMS_signed_add1_attr_by_OBJ;
798         sunw_CMS_signed_add1_attr_by_txt;
799         sunw_CMS_signed_delete_attr;
800         sunw_CMS_signed_get_attr;
801         sunw_CMS_signed_get_attr_by_NID;
802         sunw_CMS_signed_get_attr_by_OBJ;
803         sunw_CMS_signed_get_attr_count;
804         sunw_CMS_signed_get0_data_by_OBJ;
805         sunw_cms_SignedData_final;
806         sunw_CMS_SignedData_init;
807         sunw_cms_SignedData_init_bio;
808         sunw_CMS_SignedData_it;
809         sunw_cms_SignerIdentifier_cert_cmp;
810         sunw_cms_SignerIdentifier_get0_signer_id;
811         sunw_CMS_SignerIdentifier_it;
812         sunw_CMS_SignerInfo_cert_cmp;
813         sunw_CMS_SignerInfo_get0_algs;
814         sunw_CMS_SignerInfo_get0_signer_id;
815         sunw_CMS_SignerInfo_it;
816         sunw_CMS_SignerInfo_set1_signer_cert;
817         sunw_CMS_SignerInfo_sign;
818         sunw_CMS_SignerInfo_verify;
819         sunw_CMS_SignerInfo_verify_content;
820         sunw_CMS_stream;
821         sunw_CMS_uncompress;
822         sunw_CMS_unsigned_add1_attr;
823         sunw_CMS_unsigned_add1_attr_by_NID;
824         sunw_CMS_unsigned_add1_attr_by_OBJ;
825         sunw_CMS_unsigned_add1_attr_by_txt;
826         sunw_CMS_unsigned_delete_attr;
827         sunw_CMS_unsigned_get_attr;
828         sunw_CMS_unsigned_get_attr_by_NID;
829         sunw_CMS_unsigned_get_attr_by_OBJ;
830         sunw_CMS_unsigned_get_attr_count;
831         sunw_CMS_unsigned_get0_data_by_OBJ;
832         sunw_CMS_verify;
833         sunw_CMS_verify_receipt;
834         sunw_COMP_compress_block;
835         sunw_COMP_CTX_free;
836         sunw_COMP_CTX_new;
837         sunw_COMP_expand_block;
838         sunw_COMP_rle;
839         sunw_COMP_zlib;
840         sunw_COMP_zlib_cleanup;
841         sunw_CONF_def_version;
842         sunw_CONF_dump_bio;
843         sunw_CONF_dump_fp;
844         sunw_CONF_free;
845         sunw_CONF_get_number;
846         sunw_CONF_get_section;
847         sunw_CONF_get_string;
848         sunw_CONF_get1_default_config_file;
849         sunw_CONF_imodule_get_flags;
850         sunw_CONF_imodule_get_module;
851         sunw_CONF_imodule_get_name;
852         sunw_CONF_imodule_get_usr_data;
853         sunw_CONF_imodule_get_value;
```



```
854         sunw_CONF_imodule_set_flags;
855         sunw_CONF_imodule_set_usr_data;
856         sunw_CONF_load;
857         sunw_CONF_load_bio;
858         sunw_CONF_load_fp;
859         sunw_CONF_module_add;
860         sunw_CONF_module_get_usr_data;
861         sunw_CONF_module_set_usr_data;
862         sunw_CONF_modules_finish;
863         sunw_CONF_modules_free;
864         sunw_CONF_modules_load;
865         sunw_CONF_modules_load_file;
866         sunw_CONF_modules_unload;
867         sunw_CONF_parse_list;
868         sunw_CONF_set_default_method;
869         sunw_CONF_set_nconf;
870         sunw_CONF_version;
871         sunw_CRL_DIST_POINTS_free;
872         sunw_CRL_DIST_POINTS_it;
873         sunw_CRL_DIST_POINTS_new;
874         sunw_CRYPTO_add_lock;
875         sunw_CRYPTO_cbc128_decrypt;
876         sunw_CRYPTO_cbc128_encrypt;
877         sunw_CRYPTO_ccm128_aad;
878         sunw_CRYPTO_ccm128_decrypt;
879         sunw_CRYPTO_ccm128_decrypt_ccm64;
880         sunw_CRYPTO_ccm128_encrypt;
881         sunw_CRYPTO_ccm128_encrypt_ccm64;
882         sunw_CRYPTO_ccm128_init;
883         sunw_CRYPTO_ccm128_setiv;
884         sunw_CRYPTO_ccm128_tag;
885         sunw_CRYPTO_cfb128_1_encrypt;
886         sunw_CRYPTO_cfb128_8_encrypt;
887         sunw_CRYPTO_cfb128_encrypt;
888         sunw_CRYPTO_cleanup_all_ex_data;
889         sunw_CRYPTO_ctr128_encrypt;
890         sunw_CRYPTO_ctr128_encrypt_ctr32;
891         sunw_CRYPTO_cts128_decrypt;
892         sunw_CRYPTO_cts128_decrypt_block;
893         sunw_CRYPTO_cts128_encrypt;
894         sunw_CRYPTO_cts128_encrypt_block;
895         sunw_CRYPTO_dbg_free;
896         sunw_CRYPTO_dbg_get_options;
897         sunw_CRYPTO_dbg_malloc;
898         sunw_CRYPTO_dbg_realloc;
899         sunw_CRYPTO_dbg_set_options;
900         sunw_CRYPTO_destroy_dynlockid;
901         sunw_CRYPTO_dup_ex_data;
902         sunw_CRYPTO_ex_data_new_class;
903         sunw_CRYPTO_free;
904         sunw_CRYPTO_free_ex_data;
905         sunw_CRYPTO_free_locked;
906         sunw_CRYPTO_gcm128_aad;
907         sunw_CRYPTO_gcm128_decrypt;
908         sunw_CRYPTO_gcm128_decrypt_ctr32;
909         sunw_CRYPTO_gcm128_encrypt;
910         sunw_CRYPTO_gcm128_encrypt_ctr32;
911         sunw_CRYPTO_gcm128_finish;
912         sunw_CRYPTO_gcm128_init;
913         sunw_CRYPTO_gcm128_new;
914         sunw_CRYPTO_gcm128_release;
915         sunw_CRYPTO_gcm128_setiv;
916         sunw_CRYPTO_gcm128_tag;
917         sunw_CRYPTO_get_add_lock_callback;
918         sunw_CRYPTO_get_dynlock_create_callback;
919         sunw_CRYPTO_get_dynlock_destroy_callback;
```

```
920         sunw_CRYPTO_get_dynlock_lock_callback;
921         sunw_CRYPTO_get_dynlock_value;
922         sunw_CRYPTO_get_ex_data;
923         sunw_CRYPTO_get_ex_data_implementation;
924         sunw_CRYPTO_get_ex_new_index;
925         sunw_CRYPTO_get_id_callback;
926         sunw_CRYPTO_get_lock_name;
927         sunw_CRYPTO_get_locked_mem_ex_functions;
928         sunw_CRYPTO_get_locked_mem_functions;
929         sunw_CRYPTO_get_locking_callback;
930         sunw_CRYPTO_get_mem_debug_functions;
931         sunw_CRYPTO_get_mem_debug_options;
932         sunw_CRYPTO_get_mem_ex_functions;
933         sunw_CRYPTO_get_mem_functions;
934         sunw_CRYPTO_get_new_dynlockid;
935         sunw_CRYPTO_get_new_lockid;
936         sunw_CRYPTO_is_mem_check_on;
937         sunw_CRYPTO_lock;
938         sunw_CRYPTO_malloc;
939         sunw_CRYPTO_malloc_locked;
940         sunw_CRYPTO_mem_ctrl;
941         sunw_CRYPTO_mem_leaks;
942         sunw_CRYPTO_mem_leaks_cb;
943         sunw_CRYPTO_mem_leaks_fp;
944         sunw_CRYPTO_memcmp;
945         sunw_CRYPTO_new_ex_data;
946         sunw_CRYPTO_nistcts128_decrypt;
947         sunw_CRYPTO_nistcts128_decrypt_block;
948         sunw_CRYPTO_nistcts128_encrypt;
949         sunw_CRYPTO_nistcts128_encrypt_block;
950         sunw_CRYPTO_num_locks;
951         sunw_CRYPTO_ofb128_encrypt;
952         sunw_CRYPTO_pop_info;
953         sunw_CRYPTO_push_info_;
954         sunw_CRYPTO_realloc;
955         sunw_CRYPTO_realloc_clean;
956         sunw_CRYPTO_remalloc;
957         sunw_CRYPTO_remove_all_info;
958         sunw_CRYPTO_set_add_lock_callback;
959         sunw_CRYPTO_set_dynlock_create_callback;
960         sunw_CRYPTO_set_dynlock_destroy_callback;
961         sunw_CRYPTO_set_dynlock_lock_callback;
962         sunw_CRYPTO_set_ex_data;
963         sunw_CRYPTO_set_ex_data_implementation;
964         sunw_CRYPTO_set_id_callback;
965         sunw_CRYPTO_set_locked_mem_ex_functions;
966         sunw_CRYPTO_set_locked_mem_functions;
967         sunw_CRYPTO_set_locking_callback;
968         sunw_CRYPTO_set_mem_debug_functions;
969         sunw_CRYPTO_set_mem_debug_options;
970         sunw_CRYPTO_set_mem_ex_functions;
971         sunw_CRYPTO_set_mem_functions;
972         sunw_CRYPTO_strdup;
973         sunw_CRYPTO_thread_id;
974         sunw_CRYPTO_THREADID_cmp;
975         sunw_CRYPTO_THREADID_cpy;
976         sunw_CRYPTO_THREADID_current;
977         sunw_CRYPTO_THREADID_get_callback;
978         sunw_CRYPTO_THREADID_hash;
979         sunw_CRYPTO_THREADID_set_callback;
980         sunw_CRYPTO_THREADID_set_numeric;
981         sunw_CRYPTO_THREADID_set_pointer;
982         sunw_CRYPTO_xts128_encrypt;
983         sunw_d2i_ACCESS_DESCRIPTION;
984         sunw_d2i_ASN1_BIT_STRING;
985         sunw_d2i_ASN1_BMPSTRING;
```



```
986         sunw_d2i_ASN1_BOOLEAN;
987         sunw_d2i_ASN1_bytes;
988         sunw_d2i_ASN1_ENUMERATED;
989         sunw_d2i_ASN1_GENERALIZEDTIME;
990         sunw_d2i_ASN1_GENERALSTRING;
991         sunw_d2i_ASN1_IA5STRING;
992         sunw_d2i_ASN1_INTEGER;
993         sunw_d2i_ASN1_NULL;
994         sunw_d2i_ASN1_OBJECT;
995         sunw_d2i_ASN1_OCTET_STRING;
996         sunw_d2i_ASN1_PRINTABLE;
997         sunw_d2i_ASN1_PRINTABLESTRING;
998         sunw_d2i_ASN1_SEQUENCE_ANY;
999         sunw_d2i_ASN1_SET;
1000        sunw_d2i_ASN1_SET_ANY;
1001        sunw_d2i_ASN1_T61STRING;
1002        sunw_d2i_ASN1_TIME;
1003        sunw_d2i_ASN1_TYPE;
1004        sunw_d2i_ASN1_type_bytes;
1005        sunw_d2i_ASN1_UINTEGER;
1006        sunw_d2i_ASN1_UNIVERSALSTRING;
1007        sunw_d2i_ASN1_UTCTIME;
1008        sunw_d2i_ASN1_UTF8STRING;
1009        sunw_d2i_ASN1_VISIBLESTRING;
1010        sunw_d2i_AUTHORITY_INFO_ACCESS;
1011        sunw_d2i_AUTHORITY_KEYID;
1012        sunw_d2i_AutoPrivateKey;
1013        sunw_d2i_BASIC_CONSTRAINTS;
1014        sunw_d2i_CERTIFICATEPOLICIES;
1015        sunw_d2i_CMS_bio;
1016        sunw_d2i_CMS_ContentInfo;
1017        sunw_d2i_CMS_ReceiptRequest;
1018        sunw_d2i_CRL_DIST_POINTS;
1019        sunw_d2i_DHparams;
1020        sunw_d2i_DIRECTORYSTRING;
1021        sunw_d2i_DISPLAYTEXT;
1022        sunw_d2i_DIST_POINT;
1023        sunw_d2i_DIST_POINT_NAME;
1024        sunw_d2i_DSA_PUBKEY;
1025        sunw_d2i_DSA_PUBKEY_bio;
1026        sunw_d2i_DSA_PUBKEY_fp;
1027        sunw_d2i_DSA_SIG;
1028        sunw_d2i_DSAparams;
1029        sunw_d2i_DSAPrivateKey;
1030        sunw_d2i_DSAPrivateKey_bio;
1031        sunw_d2i_DSAPrivateKey_fp;
1032        sunw_d2i_DSAPublicKey;
1033        sunw_d2i_EDIPARTYNAME;
1034        sunw_d2i_ESS_CERT_ID;
1035        sunw_d2i_ESS_ISSUER_SERIAL;
1036        sunw_d2i_ESS_SIGNING_CERT;
1037        sunw_d2i_EXTENDED_KEY_USAGE;
1038        sunw_d2i_GENERAL_NAME;
1039        sunw_d2i_GENERAL_NAMES;
1040        sunw_d2i_ISSUING_DIST_POINT;
1041        sunw_d2i_KRB5_APREQ;
1042        sunw_d2i_KRB5_APREQBODY;
1043        sunw_d2i_KRB5_AUTHDATA;
1044        sunw_d2i_KRB5_AUTHENT;
1045        sunw_d2i_KRB5_AUTHENTBODY;
1046        sunw_d2i_KRB5_CHECKSUM;
1047        sunw_d2i_KRB5_ENCDATA;
1048        sunw_d2i_KRB5_ENCKEY;
1049        sunw_d2i_KRB5_PRINCNAME;
1050        sunw_d2i_KRB5_TICKET;
1051        sunw_d2i_KRB5_TKTBODY;
```

```
1052         sunw_d2i_NETSCAPE_CERT_SEQUENCE;
1053         sunw_d2i_NETSCAPE_ENCRYPTED_PKEY;
1054         sunw_d2i_NETSCAPE_PKEY;
1055         sunw_d2i_Netscape_RSA;
1056         sunw_d2i_NETSCAPE_SPKAC;
1057         sunw_d2i_NETSCAPE_SPKI;
1058         sunw_d2i_NETSCAPE_X509;
1059         sunw_d2i_NOTICEREF;
1060         sunw_d2i_OCSP_BASICRESP;
1061         sunw_d2i_OCSP_CERTID;
1062         sunw_d2i_OCSP_CERTSTATUS;
1063         sunw_d2i_OCSP_CRLID;
1064         sunw_d2i_OCSP_ONEREQ;
1065         sunw_d2i_OCSP_REQINFO;
1066         sunw_d2i_OCSP_REQUEST;
1067         sunw_d2i_OCSP_RESPBYTES;
1068         sunw_d2i_OCSP_RESPDATA;
1069         sunw_d2i_OCSP_RESPID;
1070         sunw_d2i_OCSP_RESPONSE;
1071         sunw_d2i_OCSP_REVOKEDINFO;
1072         sunw_d2i_OCSP_SERVICELOC;
1073         sunw_d2i_OCSP_SIGNATURE;
1074         sunw_d2i_OCSP_SINGLERESP;
1075         sunw_d2i_OTHERNAME;
1076         sunw_d2i_PBE2PARAM;
1077         sunw_d2i_PBEPARAM;
1078         sunw_d2i_PBKDF2PARAM;
1079         sunw_d2i_PKCS12;
1080         sunw_d2i_PKCS12_BAGS;
1081         sunw_d2i_PKCS12_bio;
1082         sunw_d2i_PKCS12_fp;
1083         sunw_d2i_PKCS12_MAC_DATA;
1084         sunw_d2i_PKCS12_SAFEBAG;
1085         sunw_d2i_PKCS7;
1086         sunw_d2i_PKCS7_bio;
1087         sunw_d2i_PKCS7_DIGEST;
1088         sunw_d2i_PKCS7_ENC_CONTENT;
1089         sunw_d2i_PKCS7_ENCRYPT;
1090         sunw_d2i_PKCS7_ENVELOPE;
1091         sunw_d2i_PKCS7_fp;
1092         sunw_d2i_PKCS7_ISSUER_AND_SERIAL;
1093         sunw_d2i_PKCS7_RECIP_INFO;
1094         sunw_d2i_PKCS7_SIGN_ENVELOPE;
1095         sunw_d2i_PKCS7_SIGNED;
1096         sunw_d2i_PKCS7_SIGNER_INFO;
1097         sunw_d2i_PKCS8_bio;
1098         sunw_d2i_PKCS8_fp;
1099         sunw_d2i_PKCS8_PRIV_KEY_INFO;
1100         sunw_d2i_PKCS8_PRIV_KEY_INFO_bio;
1101         sunw_d2i_PKCS8_PRIV_KEY_INFO_fp;
1102         sunw_d2i_PKCS8PrivateKey_bio;
1103         sunw_d2i_PKCS8PrivateKey_fp;
1104         sunw_d2i_PKEY_USAGE_PERIOD;
1105         sunw_d2i_POLICYINFO;
1106         sunw_d2i_POLICYQUALINFO;
1107         sunw_d2i_PrivateKey;
1108         sunw_d2i_PrivateKey_bio;
1109         sunw_d2i_PrivateKey_fp;
1110         sunw_d2i_PROXY_CERT_INFO_EXTENSION;
1111         sunw_d2i_PROXY_POLICY;
1112         sunw_d2i_PUBKEY;
1113         sunw_d2i_PUBKEY_bio;
1114         sunw_d2i_PUBKEY_fp;
1115         sunw_d2i_PublicKey;
1116         sunw_d2i_RSA_NET;
1117         sunw_d2i_RSA_PSS_PARAMS;
```



```
1118         sunw_d2i_RSA_PUBKEY;
1119         sunw_d2i_RSA_PUBKEY_bio;
1120         sunw_d2i_RSA_PUBKEY_fp;
1121         sunw_d2i_RSAPrivateKey;
1122         sunw_d2i_RSAPrivateKey_bio;
1123         sunw_d2i_RSAPrivateKey_fp;
1124         sunw_d2i_RSAPublicKey;
1125         sunw_d2i_RSAPublicKey_bio;
1126         sunw_d2i_RSAPublicKey_fp;
1127         sunw_d2i_SXNET;
1128         sunw_d2i_SXNETID;
1129         sunw_d2i_TS_ACCURACY;
1130         sunw_d2i_TS_MSG_IMPRINT;
1131         sunw_d2i_TS_MSG_IMPRINT_bio;
1132         sunw_d2i_TS_MSG_IMPRINT_fp;
1133         sunw_d2i_TS_REQ;
1134         sunw_d2i_TS_REQ_bio;
1135         sunw_d2i_TS_REQ_fp;
1136         sunw_d2i_TS_RESP;
1137         sunw_d2i_TS_RESP_bio;
1138         sunw_d2i_TS_RESP_fp;
1139         sunw_d2i_TS_STATUS_INFO;
1140         sunw_d2i_TS_TST_INFO;
1141         sunw_d2i_TS_TST_INFO_bio;
1142         sunw_d2i_TS_TST_INFO_fp;
1143         sunw_d2i_USERNOTICE;
1144         sunw_d2i_X509;
1145         sunw_d2i_X509_ALGOR;
1146         sunw_d2i_X509_ALGORS;
1147         sunw_d2i_X509_ATTRIBUTE;
1148         sunw_d2i_X509_AUX;
1149         sunw_d2i_X509_bio;
1150         sunw_d2i_X509_CERT_AUX;
1151         sunw_d2i_X509_CERT_PAIR;
1152         sunw_d2i_X509_CINF;
1153         sunw_d2i_X509_CRL;
1154         sunw_d2i_X509_CRL_bio;
1155         sunw_d2i_X509_CRL_fp;
1156         sunw_d2i_X509_CRL_INFO;
1157         sunw_d2i_X509_EXTENSION;
1158         sunw_d2i_X509_EXTENSIONS;
1159         sunw_d2i_X509_fp;
1160         sunw_d2i_X509_NAME;
1161         sunw_d2i_X509_NAME_ENTRY;
1162         sunw_d2i_X509_PKEY;
1163         sunw_d2i_X509_PUBKEY;
1164         sunw_d2i_X509_REQ;
1165         sunw_d2i_X509_REQ_bio;
1166         sunw_d2i_X509_REQ_fp;
1167         sunw_d2i_X509_REQ_INFO;
1168         sunw_d2i_X509_REVOKED;
1169         sunw_d2i_X509_SIG;
1170         sunw_d2i_X509_VAL;
1171         sunw_default_pctx;
1172         sunw_DES_cbc_cksum;
1173         sunw_DES_cbc_encrypt;
1174         sunw_DES_cfb_encrypt;
1175         sunw_DES_cfb64_encrypt;
1176         sunw_DES_check_key_parity;
1177         sunw_DES_crypt;
1178         sunw_DES_decrypt3;
1179         sunw_DES_ecb_encrypt;
1180         sunw_DES_ecb3_encrypt;
1181         sunw_DES_ede3_cbc_encrypt;
1182         sunw_DES_ede3_cbcm_encrypt;
1183         sunw_DES_ede3_cfb_encrypt;
```

```
1184         sunw_DES_ede3_cfb64_encrypt;
1185         sunw_DES_ede3_ofb64_encrypt;
1186         sunw_DES_enc_read;
1187         sunw_DES_enc_write;
1188         sunw_DES_encrypt1;
1189         sunw_DES_encrypt2;
1190         sunw_DES_encrypt3;
1191         sunw_DES_fcrypt;
1192         sunw_DES_is_weak_key;
1193         sunw_DES_key_sched;
1194         sunw_DES_ncbc_encrypt;
1195         sunw_DES_ofb_encrypt;
1196         sunw_DES_ofb64_encrypt;
1197         sunw_DES_options;
1198         sunw_DES_pcbc_encrypt;
1199         sunw_DES_quad_cksum;
1200         sunw_DES_random_key;
1201         sunw_DES_read_2passwords;
1202         sunw_DES_read_password;
1203         sunw_DES_set_key;
1204         sunw_DES_set_key_checked;
1205         sunw_DES_set_key_unchecked;
1206         sunw_DES_set_odd_parity;
1207         sunw_DES_SPtrans;
1208         sunw_DES_string_to_2keys;
1209         sunw_DES_string_to_key;
1210         sunw_DES_xcbc_encrypt;
1211         sunw_dh_asn1_meth;
1212         sunw_DH_check;
1213         sunw_DH_check_pub_key;
1214         sunw_DH_compute_key;
1215         sunw_DH_free;
1216         sunw_DH_generate_key;
1217         sunw_DH_generate_parameters;
1218         sunw_DH_generate_parameters_ex;
1219         sunw_DH_get_default_method;
1220         sunw_DH_get_ex_data;
1221         sunw_DH_get_ex_new_index;
1222         sunw_DH_new;
1223         sunw_DH_new_method;
1224         sunw_DH_OpenSSL;
1225         sunw_dh_pkey_meth;
1226         sunw_DH_set_default_method;
1227         sunw_DH_set_ex_data;
1228         sunw_DH_set_method;
1229         sunw_DH_size;
1230         sunw_DH_up_ref;
1231         sunw_DH_version;
1232         sunw_DHparams_dup;
1233         sunw_DHparams_it;
1234         sunw_DHparams_print;
1235         sunw_DHparams_print_fp;
1236         sunw_DIRECTORYSTRING_free;
1237         sunw_DIRECTORYSTRING_it;
1238         sunw_DIRECTORYSTRING_new;
1239         sunw_DISPLAYTEXT_free;
1240         sunw_DISPLAYTEXT_it;
1241         sunw_DISPLAYTEXT_new;
1242         sunw_DIST_POINT_free;
1243         sunw_DIST_POINT_it;
1244         sunw_DIST_POINT_NAME_free;
1245         sunw_DIST_POINT_NAME_it;
1246         sunw_DIST_POINT_NAME_new;
1247         sunw_DIST_POINT_new;
1248         sunw_DIST_POINT_set_dpname;
1249         sunw_dsa_asn1_meths;
```



```
1250         sunw_dsa_builtin_paramgen;
1251         sunw_DSA_do_sign;
1252         sunw_DSA_do_verify;
1253         sunw_DSA_dup_DH;
1254         sunw_DSA_free;
1255         sunw_DSA_generate_key;
1256         sunw_DSA_generate_parameters;
1257         sunw_DSA_generate_parameters_ex;
1258         sunw_DSA_get_default_method;
1259         sunw_DSA_get_ex_data;
1260         sunw_DSA_get_ex_new_index;
1261         sunw_DSA_new;
1262         sunw_DSA_new_method;
1263         sunw_DSA_OpenSSL;
1264         sunw_dsa_pkey_meth;
1265         sunw_DSA_print;
1266         sunw_DSA_print_fp;
1267         sunw_dsa_pub_internal_it;
1268         sunw_DSA_set_default_method;
1269         sunw_DSA_set_ex_data;
1270         sunw_DSA_set_method;
1271         sunw_DSA_SIG_free;
1272         sunw_DSA_SIG_it;
1273         sunw_DSA_SIG_new;
1274         sunw_DSA_sign;
1275         sunw_DSA_sign_setup;
1276         sunw_DSA_size;
1277         sunw_DSA_up_ref;
1278         sunw_DSA_verify;
1279         sunw_DSA_version;
1280         sunw_DSAparams_dup;
1281         sunw_DSAparams_it;
1282         sunw_DSAparams_print;
1283         sunw_DSAparams_print_fp;
1284         sunw_DSAPrivateKey_it;
1285         sunw_DSAPublicKey_it;
1286         sunw_DSO_bind_func;
1287         sunw_DSO_bind_var;
1288         sunw_DSO_convert_filename;
1289         sunw_DSO_ctrl;
1290         sunw_DSO_flags;
1291         sunw_DSO_free;
1292         sunw_DSO_get_default_method;
1293         sunw_DSO_get_filename;
1294         sunw_DSO_get_loaded_filename;
1295         sunw_DSO_get_method;
1296         sunw_DSO_global_lookup;
1297         sunw_DSO_load;
1298         sunw_DSO_merge;
1299         sunw_DSO_METHOD_beos;
1300         sunw_DSO_METHOD_dl;
1301         sunw_DSO_METHOD_dlfcn;
1302         sunw_DSO_METHOD_null;
1303         sunw_DSO_METHOD_openssl;
1304         sunw_DSO_METHOD_vms;
1305         sunw_DSO_METHOD_win32;
1306         sunw_DSO_new;
1307         sunw_DSO_new_method;
1308         sunw_DSO_pathbyaddr;
1309         sunw_DSO_set_default_method;
1310         sunw_DSO_set_filename;
1311         sunw_DSO_set_method;
1312         sunw_DSO_set_name_converter;
1313         sunw_DSO_up_ref;
1314         sunw_EDIPARTYNAME_free;
1315         sunw_EDIPARTYNAME_it;
```

```
1316         sunw_EDIPARTYNAME_new;
1317         sunw_ENGINE_add;
1318         sunw_ENGINE_add_conf_module;
1319         sunw_ENGINE_by_id;
1320         sunw_ENGINE_cleanup;
1321         sunw_engine_cleanup_add_first;
1322         sunw_engine_cleanup_add_last;
1323         sunw_ENGINE_cmd_is_executable;
1324         sunw_ENGINE_ctrl;
1325         sunw_ENGINE_ctrl_cmd;
1326         sunw_ENGINE_ctrl_cmd_string;
1327         sunw_ENGINE_finish;
1328         sunw_ENGINE_free;
1329         sunw_engine_free_util;
1330         sunw_ENGINE_get_cipher;
1331         sunw_ENGINE_get_cipher_engine;
1332         sunw_ENGINE_get_ciphers;
1333         sunw_ENGINE_get_cmd_defns;
1334         sunw_ENGINE_get_ctrl_function;
1335         sunw_ENGINE_get_default_DH;
1336         sunw_ENGINE_get_default_DSA;
1337         sunw_ENGINE_get_default_ECDH;
1338         sunw_ENGINE_get_default_ECDSA;
1339         sunw_ENGINE_get_default_RAND;
1340         sunw_ENGINE_get_default_RSA;
1341         sunw_ENGINE_get_destroy_function;
1342         sunw_ENGINE_get_DH;
1343         sunw_ENGINE_get_digest;
1344         sunw_ENGINE_get_digest_engine;
1345         sunw_ENGINE_get_digests;
1346         sunw_ENGINE_get_DSA;
1347         sunw_ENGINE_get_ECDH;
1348         sunw_ENGINE_get_ECDSA;
1349         sunw_ENGINE_get_ex_data;
1350         sunw_ENGINE_get_ex_new_index;
1351         sunw_ENGINE_get_finish_function;
1352         sunw_ENGINE_get_first;
1353         sunw_ENGINE_get_flags;
1354         sunw_ENGINE_get_id;
1355         sunw_ENGINE_get_init_function;
1356         sunw_ENGINE_get_last;
1357         sunw_ENGINE_get_load_privkey_function;
1358         sunw_ENGINE_get_load_pubkey_function;
1359         sunw_ENGINE_get_name;
1360         sunw_ENGINE_get_next;
1361         sunw_ENGINE_get_pkey_asn1_meth;
1362         sunw_ENGINE_get_pkey_asn1_meth_engine;
1363         sunw_ENGINE_get_pkey_asn1_meth_str;
1364         sunw_ENGINE_get_pkey_asn1_meths;
1365         sunw_ENGINE_get_pkey_meth;
1366         sunw_ENGINE_get_pkey_meth_engine;
1367         sunw_ENGINE_get_pkey_meths;
1368         sunw_ENGINE_get_prev;
1369         sunw_ENGINE_get_RAND;
1370         sunw_ENGINE_get_RSA;
1371         sunw_ENGINE_get_ssl_client_cert_function;
1372         sunw_ENGINE_get_static_state;
1373         sunw_ENGINE_get_STORE;
1374         sunw_ENGINE_get_table_flags;
1375         sunw_ENGINE_init;
1376         sunw_ENGINE_load_builtin_engines;
1377         sunw_ENGINE_load_cryptodev;
1378         sunw_ENGINE_load_dynamic;
1379         sunw_ENGINE_load_openssl;
1380         sunw_ENGINE_load_pk11;
1381         sunw_ENGINE_load_private_key;
```



```
1382         sunw_ENGINE_load_public_key;
1383         sunw_ENGINE_load_rdrand;
1384         sunw_ENGINE_load_rsax;
1385         sunw_ENGINE_load_ssl_client_cert;
1386         sunw_ENGINE_new;
1387         sunw_ENGINE_pkey_asn1_find_str;
1388         sunw_engine_pkey_asn1_meths_free;
1389         sunw_engine_pkey_meths_free;
1390         sunw_ENGINE_register_all_ciphers;
1391         sunw_ENGINE_register_all_complete;
1392         sunw_ENGINE_register_all_DH;
1393         sunw_ENGINE_register_all_digests;
1394         sunw_ENGINE_register_all_DSA;
1395         sunw_ENGINE_register_all_ECDH;
1396         sunw_ENGINE_register_all_ECDSA;
1397         sunw_ENGINE_register_all_pkey_asn1_meths;
1398         sunw_ENGINE_register_all_pkey_meths;
1399         sunw_ENGINE_register_all_RAND;
1400         sunw_ENGINE_register_all_RSA;
1401         sunw_ENGINE_register_all_STORE;
1402         sunw_ENGINE_register_ciphers;
1403         sunw_ENGINE_register_complete;
1404         sunw_ENGINE_register_DH;
1405         sunw_ENGINE_register_digests;
1406         sunw_ENGINE_register_DSA;
1407         sunw_ENGINE_register_ECDH;
1408         sunw_ENGINE_register_ECDSA;
1409         sunw_ENGINE_register_pkey_asn1_meths;
1410         sunw_ENGINE_register_pkey_meths;
1411         sunw_ENGINE_register_RAND;
1412         sunw_ENGINE_register_RSA;
1413         sunw_ENGINE_register_STORE;
1414         sunw_ENGINE_remove;
1415         sunw_engine_set_all_null;
1416         sunw_ENGINE_set_ciphers;
1417         sunw_ENGINE_set_cmd_defns;
1418         sunw_ENGINE_set_ctrl_function;
1419         sunw_ENGINE_set_default;
1420         sunw_ENGINE_set_default_ciphers;
1421         sunw_ENGINE_set_default_DH;
1422         sunw_ENGINE_set_default_digests;
1423         sunw_ENGINE_set_default_DSA;
1424         sunw_ENGINE_set_default_ECDH;
1425         sunw_ENGINE_set_default_ECDSA;
1426         sunw_ENGINE_set_default_pkey_asn1_meths;
1427         sunw_ENGINE_set_default_pkey_meths;
1428         sunw_ENGINE_set_default_RAND;
1429         sunw_ENGINE_set_default_RSA;
1430         sunw_ENGINE_set_default_string;
1431         sunw_ENGINE_set_destroy_function;
1432         sunw_ENGINE_set_DH;
1433         sunw_ENGINE_set_digests;
1434         sunw_ENGINE_set_DSA;
1435         sunw_ENGINE_set_ECDH;
1436         sunw_ENGINE_set_ECDSA;
1437         sunw_ENGINE_set_ex_data;
1438         sunw_ENGINE_set_finish_function;
1439         sunw_ENGINE_set_flags;
1440         sunw_ENGINE_set_id;
1441         sunw_ENGINE_set_init_function;
1442         sunw_ENGINE_set_load_privkey_function;
1443         sunw_ENGINE_set_load_pubkey_function;
1444         sunw_ENGINE_set_load_ssl_client_cert_function;
1445         sunw_ENGINE_set_name;
1446         sunw_ENGINE_set_pkey_asn1_meths;
1447         sunw_ENGINE_set_pkey_meths;
```

```
1448         sunw_ENGINE_set_RAND;
1449         sunw_ENGINE_set_RSA;
1450         sunw_ENGINE_set_STORE;
1451         sunw_ENGINE_set_table_flags;
1452         sunw_engine_table_cleanup;
1453         sunw_engine_table_doall;
1454         sunw_engine_table_register;
1455         sunw_engine_table_select;
1456         sunw_engine_table_unregister;
1457         sunw_engine_unlocked_finish;
1458         sunw_engine_unlocked_init;
1459         sunw_ENGINE_unregister_ciphers;
1460         sunw_ENGINE_unregister_DH;
1461         sunw_ENGINE_unregister_digests;
1462         sunw_ENGINE_unregister_DSA;
1463         sunw_ENGINE_unregister_ECDH;
1464         sunw_ENGINE_unregister_ECDSA;
1465         sunw_ENGINE_unregister_pkey_asn1_meths;
1466         sunw_ENGINE_unregister_pkey_meths;
1467         sunw_ENGINE_unregister_RAND;
1468         sunw_ENGINE_unregister_RSA;
1469         sunw_ENGINE_unregister_STORE;
1470         sunw_ENGINE_up_ref;
1471         sunw_ERR_add_error_data;
1472         sunw_ERR_add_error_vdata;
1473         sunw_ERR_clear_error;
1474         sunw_ERR_error_string;
1475         sunw_ERR_error_string_n;
1476         sunw_ERR_free_strings;
1477         sunw_ERR_func_error_string;
1478         sunw_ERR_get_err_state_table;
1479         sunw_ERR_get_error;
1480         sunw_ERR_get_error_line;
1481         sunw_ERR_get_error_line_data;
1482         sunw_ERR_get_implementation;
1483         sunw_ERR_get_next_error_library;
1484         sunw_ERR_get_state;
1485         sunw_ERR_get_string_table;
1486         sunw_ERR_lib_error_string;
1487         sunw_ERR_load_ASN1_strings;
1488         sunw_ERR_load_BIO_strings;
1489         sunw_ERR_load_BN_strings;
1490         sunw_ERR_load_BUF_strings;
1491         sunw_ERR_load_CMS_strings;
1492         sunw_ERR_load_COMP_strings;
1493         sunw_ERR_load_CONF_strings;
1494         sunw_ERR_load_crypto_strings;
1495         sunw_ERR_load_CRYPTO_strings;
1496         sunw_ERR_load_DH_strings;
1497         sunw_ERR_load_DSA_strings;
1498         sunw_ERR_load_DSO_strings;
1499         sunw_ERR_load_ENGINE_strings;
1500         sunw_ERR_load_ERR_strings;
1501         sunw_ERR_load_EVP_strings;
1502         sunw_ERR_load_OBJ_strings;
1503         sunw_ERR_load_OCSP_strings;
1504         sunw_ERR_load_PEM_strings;
1505         sunw_ERR_load_PKCS12_strings;
1506         sunw_ERR_load_PKCS7_strings;
1507         sunw_ERR_load_RAND_strings;
1508         sunw_ERR_load_RSA_strings;
1509         sunw_ERR_load_strings;
1510         sunw_ERR_load_TS_strings;
1511         sunw_ERR_load_UI_strings;
1512         sunw_ERR_load_X509_strings;
1513         sunw_ERR_load_X509V3_strings;
```



```
1514         sunw_ERR_peek_error;
1515         sunw_ERR_peek_error_line;
1516         sunw_ERR_peek_error_line_data;
1517         sunw_ERR_peek_last_error;
1518         sunw_ERR_peek_last_error_line;
1519         sunw_ERR_peek_last_error_line_data;
1520         sunw_ERR_pk11_error;
1521         sunw_ERR_pop_to_mark;
1522         sunw_ERR_print_errors;
1523         sunw_ERR_print_errors_cb;
1524         sunw_ERR_print_errors_fp;
1525         sunw_ERR_put_error;
1526         sunw_ERR_reason_error_string;
1527         sunw_ERR_release_err_state_table;
1528         sunw_ERR_remove_state;
1529         sunw_ERR_remove_thread_state;
1530         sunw_ERR_set_error_data;
1531         sunw_ERR_set_implementation;
1532         sunw_ERR_set_mark;
1533         sunw_ERR_unload_strings;
1534         sunw_ESS_CERT_ID_dup;
1535         sunw_ESS_CERT_ID_free;
1536         sunw_ESS_CERT_ID_it;
1537         sunw_ESS_CERT_ID_new;
1538         sunw_ESS_ISSUER_SERIAL_dup;
1539         sunw_ESS_ISSUER_SERIAL_free;
1540         sunw_ESS_ISSUER_SERIAL_it;
1541         sunw_ESS_ISSUER_SERIAL_new;
1542         sunw_ESS_SIGNING_CERT_dup;
1543         sunw_ESS_SIGNING_CERT_free;
1544         sunw_ESS_SIGNING_CERT_it;
1545         sunw_ESS_SIGNING_CERT_new;
1546         sunw_EVP_add_alg_module;
1547         sunw_EVP_add_cipher;
1548         sunw_EVP_add_digest;
1549         sunw_EVP_aes_128_cbc;
1550         sunw_EVP_aes_128_cbc_hmac_sha1;
1551         sunw_EVP_aes_128_ccm;
1552         sunw_EVP_aes_128_cfb;
1553         sunw_EVP_aes_128_cfb1;
1554         sunw_EVP_aes_128_cfb128;
1555         sunw_EVP_aes_128_cfb8;
1556         sunw_EVP_aes_128_ctr;
1557         sunw_EVP_aes_128_ecb;
1558         sunw_EVP_aes_128_gcm;
1559         sunw_EVP_aes_128_ofb;
1560         sunw_EVP_aes_128_xts;
1561         sunw_EVP_aes_192_cbc;
1562         sunw_EVP_aes_192_ccm;
1563         sunw_EVP_aes_192_cfb;
1564         sunw_EVP_aes_192_cfb1;
1565         sunw_EVP_aes_192_cfb128;
1566         sunw_EVP_aes_192_cfb8;
1567         sunw_EVP_aes_192_ctr;
1568         sunw_EVP_aes_192_ecb;
1569         sunw_EVP_aes_192_gcm;
1570         sunw_EVP_aes_192_ofb;
1571         sunw_EVP_aes_256_cbc;
1572         sunw_EVP_aes_256_cbc_hmac_sha1;
1573         sunw_EVP_aes_256_ccm;
1574         sunw_EVP_aes_256_cfb;
1575         sunw_EVP_aes_256_cfb1;
1576         sunw_EVP_aes_256_cfb128;
1577         sunw_EVP_aes_256_cfb8;
1578         sunw_EVP_aes_256_ctr;
1579         sunw_EVP_aes_256_ecb;
```

```
1580         sunw_EVP_aes_256_gcm;
1581         sunw_EVP_aes_256_ofb;
1582         sunw_EVP_aes_256_xts;
1583         sunw_EVP_bf_cbc;
1584         sunw_EVP_bf_cfb;
1585         sunw_EVP_bf_cfb64;
1586         sunw_EVP_bf_ecb;
1587         sunw_EVP_bf_ofb;
1588         sunw_EVP_BytesToKey;
1589         sunw_EVP_camellia_128_cbc;
1590         sunw_EVP_camellia_128_cfb1;
1591         sunw_EVP_camellia_128_cfb128;
1592         sunw_EVP_camellia_128_cfb8;
1593         sunw_EVP_camellia_128_ecb;
1594         sunw_EVP_camellia_128_ofb;
1595         sunw_EVP_camellia_192_cbc;
1596         sunw_EVP_camellia_192_cfb1;
1597         sunw_EVP_camellia_192_cfb128;
1598         sunw_EVP_camellia_192_cfb8;
1599         sunw_EVP_camellia_192_ecb;
1600         sunw_EVP_camellia_192_ofb;
1601         sunw_EVP_camellia_256_cbc;
1602         sunw_EVP_camellia_256_cfb1;
1603         sunw_EVP_camellia_256_cfb128;
1604         sunw_EVP_camellia_256_cfb8;
1605         sunw_EVP_camellia_256_ecb;
1606         sunw_EVP_camellia_256_ofb;
1607         sunw_EVP_cast5_cbc;
1608         sunw_EVP_cast5_cfb;
1609         sunw_EVP_cast5_cfb64;
1610         sunw_EVP_cast5_ecb;
1611         sunw_EVP_cast5_ofb;
1612         sunw_EVP_Cipher;
1613         sunw_EVP_CIPHER_asn1_to_param;
1614         sunw_EVP_CIPHER_block_size;
1615         sunw_EVP_CIPHER_CTX_block_size;
1616         sunw_EVP_CIPHER_CTX_cipher;
1617         sunw_EVP_CIPHER_CTX_cleanup;
1618         sunw_EVP_CIPHER_CTX_clear_flags;
1619         sunw_EVP_CIPHER_CTX_copy;
1620         sunw_EVP_CIPHER_CTX_ctrl;
1621         sunw_EVP_CIPHER_CTX_flags;
1622         sunw_EVP_CIPHER_CTX_free;
1623         sunw_EVP_CIPHER_CTX_get_app_data;
1624         sunw_EVP_CIPHER_CTX_init;
1625         sunw_EVP_CIPHER_CTX_iv_length;
1626         sunw_EVP_CIPHER_CTX_key_length;
1627         sunw_EVP_CIPHER_CTX_new;
1628         sunw_EVP_CIPHER_CTX_nid;
1629         sunw_EVP_CIPHER_CTX_rand_key;
1630         sunw_EVP_CIPHER_CTX_set_app_data;
1631         sunw_EVP_CIPHER_CTX_set_flags;
1632         sunw_EVP_CIPHER_CTX_set_key_length;
1633         sunw_EVP_CIPHER_CTX_set_padding;
1634         sunw_EVP_CIPHER_CTX_test_flags;
1635         sunw_EVP_CIPHER_do_all;
1636         sunw_EVP_CIPHER_do_all_sorted;
1637         sunw_EVP_CIPHER_flags;
1638         sunw_EVP_CIPHER_get_asn1_iv;
1639         sunw_EVP_CIPHER_iv_length;
1640         sunw_EVP_CIPHER_key_length;
1641         sunw_EVP_CIPHER_nid;
1642         sunw_EVP_CIPHER_param_to_asn1;
1643         sunw_EVP_CIPHER_set_asn1_iv;
1644         sunw_EVP_CIPHER_type;
1645         sunw_EVP_CipherFinal;
```



```
1646         sunw_EVP_CipherFinal_ex;
1647         sunw_EVP_CipherInit;
1648         sunw_EVP_CipherInit_ex;
1649         sunw_EVP_CipherUpdate;
1650         sunw_EVP_cleanup;
1651         sunw_EVP_DecodeBlock;
1652         sunw_EVP_DecodeFinal;
1653         sunw_EVP_DecodeInit;
1654         sunw_EVP_DecodeUpdate;
1655         sunw_EVP_DecryptFinal;
1656         sunw_EVP_DecryptFinal_ex;
1657         sunw_EVP_DecryptInit;
1658         sunw_EVP_DecryptInit_ex;
1659         sunw_EVP_DecryptUpdate;
1660         sunw_EVP_des_cbc;
1661         sunw_EVP_des_cfb;
1662         sunw_EVP_des_cfb1;
1663         sunw_EVP_des_cfb64;
1664         sunw_EVP_des_cfb8;
1665         sunw_EVP_des_ecb;
1666         sunw_EVP_des_ede;
1667         sunw_EVP_des_ede_cbc;
1668         sunw_EVP_des_ede_cfb;
1669         sunw_EVP_des_ede_cfb64;
1670         sunw_EVP_des_ede_ecb;
1671         sunw_EVP_des_ede_ofb;
1672         sunw_EVP_des_ede3;
1673         sunw_EVP_des_ede3_cbc;
1674         sunw_EVP_des_ede3_cfb;
1675         sunw_EVP_des_ede3_cfb1;
1676         sunw_EVP_des_ede3_cfb64;
1677         sunw_EVP_des_ede3_cfb8;
1678         sunw_EVP_des_ede3_ecb;
1679         sunw_EVP_des_ede3_ofb;
1680         sunw_EVP_des_ofb;
1681         sunw_EVP_desx_cbc;
1682         sunw_EVP_Digest;
1683         sunw_EVP_DigestFinal;
1684         sunw_EVP_DigestFinal_ex;
1685         sunw_EVP_DigestInit;
1686         sunw_EVP_DigestInit_ex;
1687         sunw_EVP_DigestSignFinal;
1688         sunw_EVP_DigestSignInit;
1689         sunw_EVP_DigestUpdate;
1690         sunw_EVP_DigestVerifyFinal;
1691         sunw_EVP_DigestVerifyInit;
1692         sunw_EVP_dss;
1693         sunw_EVP_dss1;
1694         sunw_EVP_ecdsa;
1695         sunw_EVP_enc_null;
1696         sunw_EVP_EncodeBlock;
1697         sunw_EVP_EncodeFinal;
1698         sunw_EVP_EncodeInit;
1699         sunw_EVP_EncodeUpdate;
1700         sunw_EVP_EncryptFinal;
1701         sunw_EVP_EncryptFinal_ex;
1702         sunw_EVP_EncryptInit;
1703         sunw_EVP_EncryptInit_ex;
1704         sunw_EVP_EncryptUpdate;
1705         sunw_EVP_get_cipherbyname;
1706         sunw_EVP_get_digestbyname;
1707         sunw_EVP_get_pw_prompt;
1708         sunw_EVP_MD_block_size;
1709         sunw_EVP_MD_CTX_cleanup;
1710         sunw_EVP_MD_CTX_clear_flags;
1711         sunw_EVP_MD_CTX_copy;
```

```
1712         sunw_EVP_MD_CTX_copy_ex;
1713         sunw_EVP_MD_CTX_create;
1714         sunw_EVP_MD_CTX_destroy;
1715         sunw_EVP_MD_CTX_init;
1716         sunw_EVP_MD_CTX_md;
1717         sunw_EVP_MD_CTX_set_flags;
1718         sunw_EVP_MD_CTX_test_flags;
1719         sunw_EVP_MD_do_all;
1720         sunw_EVP_MD_do_all_sorted;
1721         sunw_EVP_MD_flags;
1722         sunw_EVP_md_null;
1723         sunw_EVP_MD_pkey_type;
1724         sunw_EVP_MD_size;
1725         sunw_EVP_MD_type;
1726         sunw_EVP_md2;
1727         sunw_EVP_md4;
1728         sunw_EVP_md5;
1729         sunw_EVP_OpenFinal;
1730         sunw_EVP_OpenInit;
1731         sunw_EVP_PBE_alg_add;
1732         sunw_EVP_PBE_alg_add_type;
1733         sunw_EVP_PBE_CipherInit;
1734         sunw_EVP_PBE_cleanup;
1735         sunw_EVP_PBE_find;
1736         sunw_EVP_PKCS82PKEY;
1737         sunw_EVP_PKEY_add1_attr;
1738         sunw_EVP_PKEY_add1_attr_by_NID;
1739         sunw_EVP_PKEY_add1_attr_by_OBJ;
1740         sunw_EVP_PKEY_add1_attr_by_txt;
1741         sunw_EVP_PKEY_asn1_add_alias;
1742         sunw_EVP_PKEY_asn1_add0;
1743         sunw_EVP_PKEY_asn1_copy;
1744         sunw_EVP_PKEY_asn1_find;
1745         sunw_EVP_PKEY_asn1_find_str;
1746         sunw_EVP_PKEY_asn1_free;
1747         sunw_EVP_PKEY_asn1_get_count;
1748         sunw_EVP_PKEY_asn1_get0;
1749         sunw_EVP_PKEY_asn1_get0_info;
1750         sunw_EVP_PKEY_asn1_new;
1751         sunw_EVP_PKEY_asn1_set_ctrl;
1752         sunw_EVP_PKEY_asn1_set_free;
1753         sunw_EVP_PKEY_asn1_set_param;
1754         sunw_EVP_PKEY_asn1_set_private;
1755         sunw_EVP_PKEY_asn1_set_public;
1756         sunw_EVP_PKEY_assign;
1757         sunw_EVP_PKEY_base_id;
1758         sunw_EVP_PKEY_bits;
1759         sunw_EVP_PKEY_cmp;
1760         sunw_EVP_PKEY_cmp_parameters;
1761         sunw_EVP_PKEY_copy_parameters;
1762         sunw_EVP_PKEY_CTX_ctrl;
1763         sunw_EVP_PKEY_CTX_ctrl_str;
1764         sunw_EVP_PKEY_CTX_dup;
1765         sunw_EVP_PKEY_CTX_free;
1766         sunw_EVP_PKEY_CTX_get_app_data;
1767         sunw_EVP_PKEY_CTX_get_cb;
1768         sunw_EVP_PKEY_CTX_get_data;
1769         sunw_EVP_PKEY_CTX_get_keygen_info;
1770         sunw_EVP_PKEY_CTX_get_operation;
1771         sunw_EVP_PKEY_CTX_get0_peerkey;
1772         sunw_EVP_PKEY_CTX_get0_pkey;
1773         sunw_EVP_PKEY_CTX_new;
1774         sunw_EVP_PKEY_CTX_new_id;
1775         sunw_EVP_PKEY_CTX_set_app_data;
1776         sunw_EVP_PKEY_CTX_set_cb;
1777         sunw_EVP_PKEY_CTX_set_data;
```



```
1778         sunw_EVP_PKEY_CTX_set0_keygen_info;
1779         sunw_EVP_PKEY_decrypt;
1780         sunw_EVP_PKEY_decrypt_init;
1781         sunw_EVP_PKEY_decrypt_old;
1782         sunw_EVP_PKEY_delete_attr;
1783         sunw_EVP_PKEY_derive;
1784         sunw_EVP_PKEY_derive_init;
1785         sunw_EVP_PKEY_derive_set_peer;
1786         sunw_EVP_PKEY_encrypt;
1787         sunw_EVP_PKEY_encrypt_init;
1788         sunw_EVP_PKEY_encrypt_old;
1789         sunw_EVP_PKEY_free;
1790         sunw_EVP_PKEY_get_attr;
1791         sunw_EVP_PKEY_get_attr_by_NID;
1792         sunw_EVP_PKEY_get_attr_by_OBJ;
1793         sunw_EVP_PKEY_get_attr_count;
1794         sunw_EVP_PKEY_get_default_digest_nid;
1795         sunw_EVP_PKEY_get0;
1796         sunw_EVP_PKEY_get0_asn1;
1797         sunw_EVP_PKEY_get1_DH;
1798         sunw_EVP_PKEY_get1_DSA;
1799         sunw_EVP_PKEY_get1_RSA;
1800         sunw_EVP_PKEY_id;
1801         sunw_EVP_PKEY_keygen;
1802         sunw_EVP_PKEY_keygen_init;
1803         sunw_EVP_PKEY_meth_add0;
1804         sunw_EVP_PKEY_meth_copy;
1805         sunw_EVP_PKEY_meth_find;
1806         sunw_EVP_PKEY_meth_free;
1807         sunw_EVP_PKEY_meth_get0_info;
1808         sunw_EVP_PKEY_meth_new;
1809         sunw_EVP_PKEY_meth_set_cleanup;
1810         sunw_EVP_PKEY_meth_set_copy;
1811         sunw_EVP_PKEY_meth_set_ctrl;
1812         sunw_EVP_PKEY_meth_set_decrypt;
1813         sunw_EVP_PKEY_meth_set_derive;
1814         sunw_EVP_PKEY_meth_set_encrypt;
1815         sunw_EVP_PKEY_meth_set_init;
1816         sunw_EVP_PKEY_meth_set_keygen;
1817         sunw_EVP_PKEY_meth_set_paramgen;
1818         sunw_EVP_PKEY_meth_set_sign;
1819         sunw_EVP_PKEY_meth_set_signctx;
1820         sunw_EVP_PKEY_meth_set_verify;
1821         sunw_EVP_PKEY_meth_set_verify_recover;
1822         sunw_EVP_PKEY_meth_set_verifyctx;
1823         sunw_EVP_PKEY_missing_parameters;
1824         sunw_EVP_PKEY_new;
1825         sunw_EVP_PKEY_new_mac_key;
1826         sunw_EVP_PKEY_paramgen;
1827         sunw_EVP_PKEY_paramgen_init;
1828         sunw_EVP_PKEY_print_params;
1829         sunw_EVP_PKEY_print_private;
1830         sunw_EVP_PKEY_print_public;
1831         sunw_EVP_PKEY_save_parameters;
1832         sunw_evp_pkey_set_cb_translate;
1833         sunw_EVP_PKEY_set_type;
1834         sunw_EVP_PKEY_set_type_str;
1835         sunw_EVP_PKEY_set1_DH;
1836         sunw_EVP_PKEY_set1_DSA;
1837         sunw_EVP_PKEY_set1_RSA;
1838         sunw_EVP_PKEY_sign;
1839         sunw_EVP_PKEY_sign_init;
1840         sunw_EVP_PKEY_size;
1841         sunw_EVP_PKEY_type;
1842         sunw_EVP_PKEY_verify;
1843         sunw_EVP_PKEY_verify_init;
```

```
1844         sunw_EVP_PKEY_verify_recover;
1845         sunw_EVP_PKEY_verify_recover_init;
1846         sunw_EVP_PKEY2PKCS8;
1847         sunw_EVP_PKEY2PKCS8_broken;
1848         sunw_EVP_rc2_40_cbc;
1849         sunw_EVP_rc2_64_cbc;
1850         sunw_EVP_rc2_cbc;
1851         sunw_EVP_rc2_cfb;
1852         sunw_EVP_rc2_cfb64;
1853         sunw_EVP_rc2_ecb;
1854         sunw_EVP_rc2_ofb;
1855         sunw_EVP_rc4;
1856         sunw_EVP_rc4_40;
1857         sunw_EVP_rc4_hmac_md5;
1858         sunw_EVP_read_pw_string;
1859         sunw_EVP_read_pw_string_min;
1860         sunw_EVP_ripemd160;
1861         sunw_EVP_SealFinal;
1862         sunw_EVP_SealInit;
1863         sunw_EVP_set_pw_prompt;
1864         sunw_EVP_sha;
1865         sunw_EVP_sha1;
1866         sunw_EVP_sha224;
1867         sunw_EVP_sha256;
1868         sunw_EVP_sha384;
1869         sunw_EVP_sha512;
1870         sunw_EVP_SignFinal;
1871         sunw_EVP_VerifyFinal;
1872         sunw_EVP_version;
1873         sunw_EXTENDED_KEY_USAGE_free;
1874         sunw_EXTENDED_KEY_USAGE_it;
1875         sunw_EXTENDED_KEY_USAGE_new;
1876         sunw_fcrypt_body;
1877         sunw_FIPS_mode;
1878         sunw_FIPS_mode_set;
1879         sunw_gcm_ghash_clmul;
1880         sunw_gcm_gmult_clmul;
1881         sunw_gcm_init_clmul;
1882         sunw_GENERAL_NAME_cmp;
1883         sunw_GENERAL_NAME_dup;
1884         sunw_GENERAL_NAME_free;
1885         sunw_GENERAL_NAME_get0_otherName;
1886         sunw_GENERAL_NAME_get0_value;
1887         sunw_GENERAL_NAME_it;
1888         sunw_GENERAL_NAME_new;
1889         sunw_GENERAL_NAME_print;
1890         sunw_GENERAL_NAME_set0_othername;
1891         sunw_GENERAL_NAME_set0_value;
1892         sunw_GENERAL_NAMES_free;
1893         sunw_GENERAL_NAMES_it;
1894         sunw_GENERAL_NAMES_new;
1895         sunw_GENERAL_SUBTREE_free;
1896         sunw_GENERAL_SUBTREE_it;
1897         sunw_GENERAL_SUBTREE_new;
1898         sunw_get_rfc2409_prime_1024;
1899         sunw_get_rfc2409_prime_768;
1900         sunw_get_rfc3526_prime_1536;
1901         sunw_get_rfc3526_prime_2048;
1902         sunw_get_rfc3526_prime_3072;
1903         sunw_get_rfc3526_prime_4096;
1904         sunw_get_rfc3526_prime_6144;
1905         sunw_get_rfc3526_prime_8192;
1906         sunw_hex_to_string;
1907         sunw_HMAC;
1908         sunw_hmac_asn1_meth;
1909         sunw_HMAC_CTX_cleanup;
```



```
1910         sunw_HMAC_CTX_copy;
1911         sunw_HMAC_CTX_init;
1912         sunw_HMAC_CTX_set_flags;
1913         sunw_HMAC_Final;
1914         sunw_HMAC_Init;
1915         sunw_HMAC_Init_ex;
1916         sunw_hmac_pkey_meth;
1917         sunw_HMAC_Update;
1918         sunw_i2a_ACCESS_DESCRIPTION;
1919         sunw_i2a_ASN1_ENUMERATED;
1920         sunw_i2a_ASN1_INTEGER;
1921         sunw_i2a_ASN1_OBJECT;
1922         sunw_i2a_ASN1_STRING;
1923         sunw_i2b_PrivateKey_bio;
1924         sunw_i2b_PublicKey_bio;
1925         sunw_i2b_PVK_bio;
1926         sunw_i2c_ASN1_BIT_STRING;
1927         sunw_i2c_ASN1_INTEGER;
1928         sunw_i2d_ACCESS_DESCRIPTION;
1929         sunw_i2d_ASN1_bio_stream;
1930         sunw_i2d_ASN1_BIT_STRING;
1931         sunw_i2d_ASN1_BMPSTRING;
1932         sunw_i2d_ASN1_BOOLEAN;
1933         sunw_i2d_ASN1_bytes;
1934         sunw_i2d_ASN1_ENUMERATED;
1935         sunw_i2d_ASN1_GENERALIZEDTIME;
1936         sunw_i2d_ASN1_GENERALSTRING;
1937         sunw_i2d_ASN1_IA5STRING;
1938         sunw_i2d_ASN1_INTEGER;
1939         sunw_i2d_ASN1_NULL;
1940         sunw_i2d_ASN1_OBJECT;
1941         sunw_i2d_ASN1_OCTET_STRING;
1942         sunw_i2d_ASN1_PRINTABLE;
1943         sunw_i2d_ASN1_PRINTABLESTRING;
1944         sunw_i2d_ASN1_SEQUENCE_ANY;
1945         sunw_i2d_ASN1_SET;
1946         sunw_i2d_ASN1_SET_ANY;
1947         sunw_i2d_ASN1_T61STRING;
1948         sunw_i2d_ASN1_TIME;
1949         sunw_i2d_ASN1_TYPE;
1950         sunw_i2d_ASN1_UNIVERSALSTRING;
1951         sunw_i2d_ASN1_UTCTIME;
1952         sunw_i2d_ASN1_UTF8STRING;
1953         sunw_i2d_ASN1_VISIBLESTRING;
1954         sunw_i2d_AUTHORITY_INFO_ACCESS;
1955         sunw_i2d_AUTHORITY_KEYID;
1956         sunw_i2d_BASIC_CONSTRAINTS;
1957         sunw_i2d_CERTIFICATEPOLICIES;
1958         sunw_i2d_CMS_bio;
1959         sunw_i2d_CMS_bio_stream;
1960         sunw_i2d_CMS_ContentInfo;
1961         sunw_i2d_CMS_ReceiptRequest;
1962         sunw_i2d_CRL_DIST_POINTS;
1963         sunw_i2d_DHparams;
1964         sunw_i2d_DIRECTORYSTRING;
1965         sunw_i2d_DISPLAYTEXT;
1966         sunw_i2d_DIST_POINT;
1967         sunw_i2d_DIST_POINT_NAME;
1968         sunw_i2d_DSA_PUBKEY;
1969         sunw_i2d_DSA_PUBKEY_bio;
1970         sunw_i2d_DSA_PUBKEY_fp;
1971         sunw_i2d_DSA_SIG;
1972         sunw_i2d_DSAparams;
1973         sunw_i2d_DSAPrivateKey;
1974         sunw_i2d_DSAPrivateKey_bio;
1975         sunw_i2d_DSAPrivateKey_fp;
```

```
1976         sunw_i2d_DSAPublicKey;
1977         sunw_i2d_EDIPARTYNAME;
1978         sunw_i2d_ESS_CERT_ID;
1979         sunw_i2d_ESS_ISSUER_SERIAL;
1980         sunw_i2d_ESS_SIGNING_CERT;
1981         sunw_i2d_EXTENDED_KEY_USAGE;
1982         sunw_i2d_GENERAL_NAME;
1983         sunw_i2d_GENERAL_NAMES;
1984         sunw_i2d_ISSUING_DIST_POINT;
1985         sunw_i2d_KRB5_APREQ;
1986         sunw_i2d_KRB5_APREQBODY;
1987         sunw_i2d_KRB5_AUTHDATA;
1988         sunw_i2d_KRB5_AUTHENT;
1989         sunw_i2d_KRB5_AUTHENTBODY;
1990         sunw_i2d_KRB5_CHECKSUM;
1991         sunw_i2d_KRB5_ENCDATA;
1992         sunw_i2d_KRB5_ENCKEY;
1993         sunw_i2d_KRB5_PRINCNAME;
1994         sunw_i2d_KRB5_TICKET;
1995         sunw_i2d_KRB5_TKTBODY;
1996         sunw_i2d_NETSCAPE_CERT_SEQUENCE;
1997         sunw_i2d_NETSCAPE_ENCRYPTED_PKEY;
1998         sunw_i2d_NETSCAPE_PKEY;
1999         sunw_i2d_Netscape_RSA;
2000         sunw_i2d_NETSCAPE_SPKAC;
2001         sunw_i2d_NETSCAPE_SPKI;
2002         sunw_i2d_NETSCAPE_X509;
2003         sunw_i2d_NOTICEREF;
2004         sunw_i2d_OCSP_BASICRESP;
2005         sunw_i2d_OCSP_CERTID;
2006         sunw_i2d_OCSP_CERTSTATUS;
2007         sunw_i2d_OCSP_CRLID;
2008         sunw_i2d_OCSP_ONEREQ;
2009         sunw_i2d_OCSP_REQINFO;
2010         sunw_i2d_OCSP_REQUEST;
2011         sunw_i2d_OCSP_RESPBYTES;
2012         sunw_i2d_OCSP_RESPDATA;
2013         sunw_i2d_OCSP_RESPID;
2014         sunw_i2d_OCSP_RESPONSE;
2015         sunw_i2d_OCSP_REVOKEDINFO;
2016         sunw_i2d_OCSP_SERVICELOC;
2017         sunw_i2d_OCSP_SIGNATURE;
2018         sunw_i2d_OCSP_SINGLERESP;
2019         sunw_i2d_OTHERNAME;
2020         sunw_i2d_PBE2PARAM;
2021         sunw_i2d_PBEPARAM;
2022         sunw_i2d_PBKDF2PARAM;
2023         sunw_i2d_PKCS12;
2024         sunw_i2d_PKCS12_BAGS;
2025         sunw_i2d_PKCS12_bio;
2026         sunw_i2d_PKCS12_fp;
2027         sunw_i2d_PKCS12_MAC_DATA;
2028         sunw_i2d_PKCS12_SAFEBAG;
2029         sunw_i2d_PKCS7;
2030         sunw_i2d_PKCS7_bio;
2031         sunw_i2d_PKCS7_bio_stream;
2032         sunw_i2d_PKCS7_DIGEST;
2033         sunw_i2d_PKCS7_ENC_CONTENT;
2034         sunw_i2d_PKCS7_ENCRYPT;
2035         sunw_i2d_PKCS7_ENVELOPE;
2036         sunw_i2d_PKCS7_fp;
2037         sunw_i2d_PKCS7_ISSUER_AND_SERIAL;
2038         sunw_i2d_PKCS7_NDEF;
2039         sunw_i2d_PKCS7_RECIP_INFO;
2040         sunw_i2d_PKCS7_SIGN_ENVELOPE;
2041         sunw_i2d_PKCS7_SIGNED;
```



```
2042         sunw_i2d_PKCS7_SIGNER_INFO;
2043         sunw_i2d_PKCS8_bio;
2044         sunw_i2d_PKCS8_fp;
2045         sunw_i2d_PKCS8_PRIV_KEY_INFO;
2046         sunw_i2d_PKCS8_PRIV_KEY_INFO_bio;
2047         sunw_i2d_PKCS8_PRIV_KEY_INFO_fp;
2048         sunw_i2d_PKCS8PrivateKey_bio;
2049         sunw_i2d_PKCS8PrivateKey_fp;
2050         sunw_i2d_PKCS8PrivateKey_nid_bio;
2051         sunw_i2d_PKCS8PrivateKey_nid_fp;
2052         sunw_i2d_PKCS8PrivateKeyInfo_bio;
2053         sunw_i2d_PKCS8PrivateKeyInfo_fp;
2054         sunw_i2d_PKEY_USAGE_PERIOD;
2055         sunw_i2d_POLICYINFO;
2056         sunw_i2d_POLICYQUALINFO;
2057         sunw_i2d_PrivateKey;
2058         sunw_i2d_PrivateKey_bio;
2059         sunw_i2d_PrivateKey_fp;
2060         sunw_i2d_PROXY_CERT_INFO_EXTENSION;
2061         sunw_i2d_PROXY_POLICY;
2062         sunw_i2d_PUBKEY;
2063         sunw_i2d_PUBKEY_bio;
2064         sunw_i2d_PUBKEY_fp;
2065         sunw_i2d_PublicKey;
2066         sunw_i2d_RSA_NET;
2067         sunw_i2d_RSA_PSS_PARAMS;
2068         sunw_i2d_RSA_PUBKEY;
2069         sunw_i2d_RSA_PUBKEY_bio;
2070         sunw_i2d_RSA_PUBKEY_fp;
2071         sunw_i2d_RSAPrivateKey;
2072         sunw_i2d_RSAPrivateKey_bio;
2073         sunw_i2d_RSAPrivateKey_fp;
2074         sunw_i2d_RSAPublicKey;
2075         sunw_i2d_RSAPublicKey_bio;
2076         sunw_i2d_RSAPublicKey_fp;
2077         sunw_i2d_SXNET;
2078         sunw_i2d_SXNETID;
2079         sunw_i2d_TS_ACCURACY;
2080         sunw_i2d_TS_MSG_IMPRINT;
2081         sunw_i2d_TS_MSG_IMPRINT_bio;
2082         sunw_i2d_TS_MSG_IMPRINT_fp;
2083         sunw_i2d_TS_REQ;
2084         sunw_i2d_TS_REQ_bio;
2085         sunw_i2d_TS_REQ_fp;
2086         sunw_i2d_TS_RESP;
2087         sunw_i2d_TS_RESP_bio;
2088         sunw_i2d_TS_RESP_fp;
2089         sunw_i2d_TS_STATUS_INFO;
2090         sunw_i2d_TS_TST_INFO;
2091         sunw_i2d_TS_TST_INFO_bio;
2092         sunw_i2d_TS_TST_INFO_fp;
2093         sunw_i2d_USERNOTICE;
2094         sunw_i2d_X509;
2095         sunw_i2d_X509_ALGOR;
2096         sunw_i2d_X509_ALGORS;
2097         sunw_i2d_X509_ATTRIBUTE;
2098         sunw_i2d_X509_AUX;
2099         sunw_i2d_X509_bio;
2100         sunw_i2d_X509_CERT_AUX;
2101         sunw_i2d_X509_CERT_PAIR;
2102         sunw_i2d_X509_CINF;
2103         sunw_i2d_X509_CRL;
2104         sunw_i2d_X509_CRL_bio;
2105         sunw_i2d_X509_CRL_fp;
2106         sunw_i2d_X509_CRL_INFO;
2107         sunw_i2d_X509_EXTENSION;
```

```
2108         sunw_i2d_X509_EXTENSIONS;
2109         sunw_i2d_X509_fp;
2110         sunw_i2d_X509_NAME;
2111         sunw_i2d_X509_NAME_ENTRY;
2112         sunw_i2d_X509_PKEY;
2113         sunw_i2d_X509_PUBKEY;
2114         sunw_i2d_X509_REQ;
2115         sunw_i2d_X509_REQ_bio;
2116         sunw_i2d_X509_REQ_fp;
2117         sunw_i2d_X509_REQ_INFO;
2118         sunw_i2d_X509_REVOKED;
2119         sunw_i2d_X509_SIG;
2120         sunw_i2d_X509_VAL;
2121         sunw_i2s_ASN1_ENUMERATED;
2122         sunw_i2s_ASN1_ENUMERATED_TABLE;
2123         sunw_i2s_ASN1_INTEGER;
2124         sunw_i2s_ASN1_OCTET_STRING;
2125         sunw_i2t_ASN1_OBJECT;
2126         sunw_i2v_ASN1_BIT_STRING;
2127         sunw_i2v_GENERAL_NAME;
2128         sunw_i2v_GENERAL_NAMES;
2129         sunw_int_rsa_verify;
2130         sunw_ISSUING_DIST_POINT_free;
2131         sunw_ISSUING_DIST_POINT_it;
2132         sunw_ISSUING_DIST_POINT_new;
2133         sunw_KRB5_APREQ_free;
2134         sunw_KRB5_APREQ_it;
2135         sunw_KRB5_APREQ_new;
2136         sunw_KRB5_APREQBODY_free;
2137         sunw_KRB5_APREQBODY_it;
2138         sunw_KRB5_APREQBODY_new;
2139         sunw_KRB5_AUTHDATA_free;
2140         sunw_KRB5_AUTHDATA_it;
2141         sunw_KRB5_AUTHDATA_new;
2142         sunw_KRB5_AUTHENT_free;
2143         sunw_KRB5_AUTHENT_it;
2144         sunw_KRB5_AUTHENT_new;
2145         sunw_KRB5_AUTHENTBODY_free;
2146         sunw_KRB5_AUTHENTBODY_it;
2147         sunw_KRB5_AUTHENTBODY_new;
2148         sunw_KRB5_CHECKSUM_free;
2149         sunw_KRB5_CHECKSUM_it;
2150         sunw_KRB5_CHECKSUM_new;
2151         sunw_KRB5_ENCDATA_free;
2152         sunw_KRB5_ENCDATA_it;
2153         sunw_KRB5_ENCDATA_new;
2154         sunw_KRB5_ENCKEY_free;
2155         sunw_KRB5_ENCKEY_it;
2156         sunw_KRB5_ENCKEY_new;
2157         sunw_KRB5_PRINCNAME_free;
2158         sunw_KRB5_PRINCNAME_it;
2159         sunw_KRB5_PRINCNAME_new;
2160         sunw_KRB5_TICKET_free;
2161         sunw_KRB5_TICKET_it;
2162         sunw_KRB5_TICKET_new;
2163         sunw_KRB5_TKTBODY_free;
2164         sunw_KRB5_TKTBODY_it;
2165         sunw_KRB5_TKTBODY_new;
2166         sunw_level_add_node;
2167         sunw_level_find_node;
2168         sunw_lh_delete;
2169         sunw_lh_doall;
2170         sunw_lh_doall_arg;
2171         sunw_lh_free;
2172         sunw_lh_insert;
2173         sunw_lh_new;
```



```
2174         sunw_lh_node_stats;
2175         sunw_lh_node_stats_bio;
2176         sunw_lh_node_usage_stats;
2177         sunw_lh_node_usage_stats_bio;
2178         sunw_lh_num_items;
2179         sunw_lh_retrieve;
2180         sunw_lh_stats;
2181         sunw_lh_stats_bio;
2182         sunw_lh_strhash;
2183         sunw_lh_version;
2184         sunw_LONG_it;
2185         sunw_MD2;
2186         sunw_MD2_Final;
2187         sunw_MD2_Init;
2188         sunw_MD2_options;
2189         sunw_MD2_Update;
2190         sunw_MD2_version;
2191         sunw_MD4;
2192         sunw_md4_block_data_order;
2193         sunw_MD4_Final;
2194         sunw_MD4_Init;
2195         sunw_MD4_Transform;
2196         sunw_MD4_Update;
2197         sunw_MD4_version;
2198         sunw_MD5;
2199         sunw_md5_block_asm_data_order;
2200         sunw_MD5_Final;
2201         sunw_MD5_Init;
2202         sunw_MD5_Transform;
2203         sunw_MD5_Update;
2204         sunw_MD5_version;
2205         sunw_name_cmp;
2206         sunw_NAME_CONSTRAINTS_check;
2207         sunw_NAME_CONSTRAINTS_free;
2208         sunw_NAME_CONSTRAINTS_it;
2209         sunw_NAME_CONSTRAINTS_new;
2210         sunw_NCONF_default;
2211         sunw_NCONF_dump_bio;
2212         sunw_NCONF_dump_fp;
2213         sunw_NCONF_free;
2214         sunw_NCONF_free_data;
2215         sunw_NCONF_get_number_e;
2216         sunw_NCONF_get_section;
2217         sunw_NCONF_get_string;
2218         sunw_NCONF_load;
2219         sunw_NCONF_load_bio;
2220         sunw_NCONF_load_fp;
2221         sunw_NCONF_new;
2222         sunw_NCONF_WIN32;
2223         sunw_NETSCAPE_CERT_SEQUENCE_free;
2224         sunw_NETSCAPE_CERT_SEQUENCE_it;
2225         sunw_NETSCAPE_CERT_SEQUENCE_new;
2226         sunw_NETSCAPE_ENCRYPTED_PKEY_free;
2227         sunw_NETSCAPE_ENCRYPTED_PKEY_it;
2228         sunw_NETSCAPE_ENCRYPTED_PKEY_new;
2229         sunw_NETSCAPE_PKEY_free;
2230         sunw_NETSCAPE_PKEY_it;
2231         sunw_NETSCAPE_PKEY_new;
2232         sunw_NETSCAPE_SPKAC_free;
2233         sunw_NETSCAPE_SPKAC_it;
2234         sunw_NETSCAPE_SPKAC_new;
2235         sunw_NETSCAPE_SPKI_b64_decode;
2236         sunw_NETSCAPE_SPKI_b64_encode;
2237         sunw_NETSCAPE_SPKI_free;
2238         sunw_NETSCAPE_SPKI_get_pubkey;
2239         sunw_NETSCAPE_SPKI_it;
```

```
2240         sunw_NETSCAPE_SPKI_new;
2241         sunw_NETSCAPE_SPKI_print;
2242         sunw_NETSCAPE_SPKI_set_pubkey;
2243         sunw_NETSCAPE_SPKI_sign;
2244         sunw_NETSCAPE_SPKI_verify;
2245         sunw_NETSCAPE_X509_free;
2246         sunw_NETSCAPE_X509_it;
2247         sunw_NETSCAPE_X509_new;
2248         sunw_NOTICEREF_free;
2249         sunw_NOTICEREF_it;
2250         sunw_NOTICEREF_new;
2251         sunw_OBJ_add_object;
2252         sunw_OBJ_add_sigid;
2253         sunw_OBJ_bsearch_;
2254         sunw_OBJ_bsearch_ex_;
2255         sunw_OBJ_cleanup;
2256         sunw_OBJ_cmp;
2257         sunw_OBJ_create;
2258         sunw_OBJ_create_objects;
2259         sunw_OBJ_dup;
2260         sunw_OBJ_find_sigid_algs;
2261         sunw_OBJ_find_sigid_by_algs;
2262         sunw_OBJ_ln2nid;
2263         sunw_OBJ_NAME_add;
2264         sunw_OBJ_NAME_cleanup;
2265         sunw_OBJ_NAME_do_all;
2266         sunw_OBJ_NAME_do_all_sorted;
2267         sunw_OBJ_NAME_get;
2268         sunw_OBJ_NAME_init;
2269         sunw_OBJ_NAME_new_index;
2270         sunw_OBJ_NAME_remove;
2271         sunw_OBJ_new_nid;
2272         sunw_OBJ_nid2ln;
2273         sunw_OBJ_nid2obj;
2274         sunw_OBJ_nid2sn;
2275         sunw_OBJ_obj2nid;
2276         sunw_OBJ_obj2txt;
2277         sunw_OBJ_sigid_free;
2278         sunw_OBJ_sn2nid;
2279         sunw_OBJ_txt2nid;
2280         sunw_OBJ_txt2obj;
2281         sunw_OCSP_accept_responses_new;
2282         sunw_OCSP_archive_cutoff_new;
2283         sunw_OCSP_basic_add1_cert;
2284         sunw_OCSP_basic_add1_nonce;
2285         sunw_OCSP_basic_add1_status;
2286         sunw_OCSP_basic_sign;
2287         sunw_OCSP_basic_verify;
2288         sunw_OCSP_BASICRESP_add_ext;
2289         sunw_OCSP_BASICRESP_add1_ext_i2d;
2290         sunw_OCSP_BASICRESP_delete_ext;
2291         sunw_OCSP_BASICRESP_free;
2292         sunw_OCSP_BASICRESP_get_ext;
2293         sunw_OCSP_BASICRESP_get_ext_by_critical;
2294         sunw_OCSP_BASICRESP_get_ext_by_NID;
2295         sunw_OCSP_BASICRESP_get_ext_by_OBJ;
2296         sunw_OCSP_BASICRESP_get_ext_count;
2297         sunw_OCSP_BASICRESP_get1_ext_d2i;
2298         sunw_OCSP_BASICRESP_it;
2299         sunw_OCSP_BASICRESP_new;
2300         sunw_OCSP_cert_id_new;
2301         sunw_OCSP_cert_status_str;
2302         sunw_OCSP_cert_to_id;
2303         sunw_OCSP_CERTID_dup;
2304         sunw_OCSP_CERTID_free;
2305         sunw_OCSP_CERTID_it;
```



```
2306         sunw_OCSP_CERTID_new;
2307         sunw_OCSP_CERTSTATUS_free;
2308         sunw_OCSP_CERTSTATUS_it;
2309         sunw_OCSP_CERTSTATUS_new;
2310         sunw_OCSP_check_nonce;
2311         sunw_OCSP_check_validity;
2312         sunw_OCSP_copy_nonce;
2313         sunw_OCSP_crl_reason_str;
2314         sunw_OCSP_CRLID_free;
2315         sunw_OCSP_CRLID_it;
2316         sunw_OCSP_crlID_new;
2317         sunw_OCSP_CRLID_new;
2318         sunw_OCSP_id_cmp;
2319         sunw_OCSP_id_get0_info;
2320         sunw_OCSP_id_issuer_cmp;
2321         sunw_OCSP_ONEREQ_add_ext;
2322         sunw_OCSP_ONEREQ_add1_ext_i2d;
2323         sunw_OCSP_ONEREQ_delete_ext;
2324         sunw_OCSP_ONEREQ_free;
2325         sunw_OCSP_ONEREQ_get_ext;
2326         sunw_OCSP_ONEREQ_get_ext_by_critical;
2327         sunw_OCSP_ONEREQ_get_ext_by_NID;
2328         sunw_OCSP_ONEREQ_get_ext_by_OBJ;
2329         sunw_OCSP_ONEREQ_get_ext_count;
2330         sunw_OCSP_onereq_get0_id;
2331         sunw_OCSP_ONEREQ_get1_ext_d2i;
2332         sunw_OCSP_ONEREQ_it;
2333         sunw_OCSP_ONEREQ_new;
2334         sunw_OCSP_parse_url;
2335         sunw_OCSP_REQ_CTX_add1_header;
2336         sunw_OCSP_REQ_CTX_free;
2337         sunw_OCSP_REQ_CTX_set1_req;
2338         sunw_OCSP_REQINFO_free;
2339         sunw_OCSP_REQINFO_it;
2340         sunw_OCSP_REQINFO_new;
2341         sunw_OCSP_REQUEST_add_ext;
2342         sunw_OCSP_request_add0_id;
2343         sunw_OCSP_request_add1_cert;
2344         sunw_OCSP_REQUEST_add1_ext_i2d;
2345         sunw_OCSP_request_add1_nonce;
2346         sunw_OCSP_REQUEST_delete_ext;
2347         sunw_OCSP_REQUEST_free;
2348         sunw_OCSP_REQUEST_get_ext;
2349         sunw_OCSP_REQUEST_get_ext_by_critical;
2350         sunw_OCSP_REQUEST_get_ext_by_NID;
2351         sunw_OCSP_REQUEST_get_ext_by_OBJ;
2352         sunw_OCSP_REQUEST_get_ext_count;
2353         sunw_OCSP_REQUEST_get1_ext_d2i;
2354         sunw_OCSP_request_is_signed;
2355         sunw_OCSP_REQUEST_it;
2356         sunw_OCSP_REQUEST_new;
2357         sunw_OCSP_request_onereq_count;
2358         sunw_OCSP_request_onereq_get0;
2359         sunw_OCSP_REQUEST_print;
2360         sunw_OCSP_request_set1_name;
2361         sunw_OCSP_request_sign;
2362         sunw_OCSP_request_verify;
2363         sunw_OCSP_resp_count;
2364         sunw_OCSP_resp_find;
2365         sunw_OCSP_resp_find_status;
2366         sunw_OCSP_resp_get0;
2367         sunw_OCSP_RESPBYTES_free;
2368         sunw_OCSP_RESPBYTES_it;
2369         sunw_OCSP_RESPBYTES_new;
2370         sunw_OCSP_RESPDATA_free;
2371         sunw_OCSP_RESPDATA_it;
```

```
        sunw_OCSP_RESPDATA_new;
        sunw_OCSP_RESPID_free;
        sunw_OCSP_RESPID_it;
        sunw_OCSP_RESPID_new;
        sunw_OCSP_response_create;
        sunw_OCSP_RESPONSE_free;
        sunw_OCSP_response_get1_basic;
        sunw_OCSP_RESPONSE_it;
        sunw_OCSP_RESPONSE_new;
        sunw_OCSP_RESPONSE_print;
        sunw_OCSP_response_status;
        sunw_OCSP_response_status_str;
        sunw_OCSP_REVOKEDINFO_free;
        sunw_OCSP_REVOKEDINFO_it;
        sunw_OCSP_REVOKEDINFO_new;
        sunw_OCSP_sendreq_bio;
        sunw_OCSP_sendreq_nbio;
        sunw_OCSP_sendreq_new;
        sunw_OCSP_SERVICELOC_free;
        sunw_OCSP_SERVICELOC_it;
        sunw_OCSP_SERVICELOC_new;
        sunw_OCSP_SIGNATURE_free;
        sunw_OCSP_SIGNATURE_it;
        sunw_OCSP_SIGNATURE_new;
        sunw_OCSP_single_get0_status;
        sunw_OCSP_SINGLERESP_add_ext;
        sunw_OCSP_SINGLERESP_add1_ext_i2d;
        sunw_OCSP_SINGLERESP_delete_ext;
        sunw_OCSP_SINGLERESP_free;
        sunw_OCSP_SINGLERESP_get_ext;
        sunw_OCSP_SINGLERESP_get_ext_by_critical;
        sunw_OCSP_SINGLERESP_get_ext_by_NID;
        sunw_OCSP_SINGLERESP_get_ext_by_OBJ;
        sunw_OCSP_SINGLERESP_get_ext_count;
        sunw_OCSP_SINGLERESP_get1_ext_d2i;
        sunw_OCSP_SINGLERESP_it;
        sunw_OCSP_SINGLERESP_new;
        sunw_OCSP_url_svcloc_new;
        sunw_OPENSSL_add_all_algorithms_conf;
        sunw_OPENSSL_add_all_algorithms_noconf;
        sunw_OpenSSL_add_all_ciphers;
        sunw_OpenSSL_add_all_digests;
        sunw_OPENSSL_asc2uni;
        sunw_OPENSSL_atomic_add;
        sunw_OPENSSL_cleanse;
        sunw_OPENSSL_config;
        sunw_OPENSSL_DIR_end;
        sunw_OPENSSL_DIR_read;
        sunw_OPENSSL_gmtime;
        sunw_OPENSSL_gmtime_adj;
        sunw_OPENSSL_ia32_cpuid;
        sunw_OPENSSL_ia32_rdrand;
        sunw_OPENSSL_ia32cap_loc;
        sunw_OPENSSL_init;
        sunw_OPENSSL_isservice;
        sunw_OPENSSL_issetugid;
        sunw_OPENSSL_load_builtin_modules;
        sunw_OPENSSL_memcmp;
        sunw_OPENSSL_no_config;
        sunw_OPENSSL_rdtsc;
        sunw_OPENSSL_showfatal;
        sunw_OPENSSL_stderr;
        sunw_OPENSSL_strcasecmp;
        sunw_OPENSSL_strncasecmp;
        sunw_OPENSSL_uni2asc;
        sunw_OPENSSL_wipe_cpu;
```



```
        sunw_OpenSSLDie;
        sunw_OSSL_DES_version;
        sunw_OSSL_libdes_version;
        sunw_OTHERNAME_cmp;
        sunw_OTHERNAME_free;
        sunw_OTHERNAME_it;
        sunw_OTHERNAME_new;
        sunw_PBE2PARAM_free;
        sunw_PBE2PARAM_it;
        sunw_PBE2PARAM_new;
        sunw_PBEPARAM_free;
        sunw_PBEPARAM_it;
        sunw_PBEPARAM_new;
        sunw_PBKDF2PARAM_free;
        sunw_PBKDF2PARAM_it;
        sunw_PBKDF2PARAM_new;
        sunw_PEM_ASN1_read;
        sunw_PEM_ASN1_read_bio;
        sunw_PEM_ASN1_write;
        sunw_PEM_ASN1_write_bio;
        sunw_PEM_bytes_read_bio;
        sunw_pem_check_suffix;
        sunw_PEM_def_callback;
        sunw_PEM_dek_info;
        sunw_PEM_do_header;
        sunw_PEM_get_EVP_CIPHER_INFO;
        sunw_PEM_proc_type;
        sunw_PEM_read;
        sunw_PEM_read_bio;
        sunw_PEM_read_bio_CMS;
        sunw_PEM_read_bio_DHparams;
        sunw_PEM_read_bio_DSA_PUBKEY;
        sunw_PEM_read_bio_DSAparams;
        sunw_PEM_read_bio_DSAPrivateKey;
        sunw_PEM_read_bio_NETSCAPE_CERT_SEQUENCE;
        sunw_PEM_read_bio_Parameters;
        sunw_PEM_read_bio_PKCS7;
        sunw_PEM_read_bio_PKCS8;
        sunw_PEM_read_bio_PKCS8_PRIV_KEY_INFO;
        sunw_PEM_read_bio_PrivateKey;
        sunw_PEM_read_bio_PUBKEY;
        sunw_PEM_read_bio_RSA_PUBKEY;
        sunw_PEM_read_bio_RSAPrivateKey;
        sunw_PEM_read_bio_RSAPublicKey;
        sunw_PEM_read_bio_X509;
        sunw_PEM_read_bio_X509_AUX;
        sunw_PEM_read_bio_X509_CERT_PAIR;
        sunw_PEM_read_bio_X509_CRL;
        sunw_PEM_read_bio_X509_REQ;
        sunw_PEM_read_CMS;
        sunw_PEM_read_DHparams;
        sunw_PEM_read_DSA_PUBKEY;
        sunw_PEM_read_DSAparams;
        sunw_PEM_read_DSAPrivateKey;
        sunw_PEM_read_NETSCAPE_CERT_SEQUENCE;
        sunw_PEM_read_PKCS7;
        sunw_PEM_read_PKCS8;
        sunw_PEM_read_PKCS8_PRIV_KEY_INFO;
        sunw_PEM_read_PrivateKey;
        sunw_PEM_read_PUBKEY;
        sunw_PEM_read_RSA_PUBKEY;
        sunw_PEM_read_RSAPrivateKey;
        sunw_PEM_read_RSAPublicKey;
        sunw_PEM_read_X509;
        sunw_PEM_read_X509_AUX;
        sunw_PEM_read_X509_CERT_PAIR;
```

```
2504         sunw_PEM_read_X509_CRL;
2505         sunw_PEM_read_X509_REQ;
2506         sunw_PEM_SealFinal;
2507         sunw_PEM_SealInit;
2508         sunw_PEM_SealUpdate;
2509         sunw_PEM_SignFinal;
2510         sunw_PEM_SignInit;
2511         sunw_PEM_SignUpdate;
2512         sunw_PEM_version;
2513         sunw_PEM_write;
2514         sunw_PEM_write_bio;
2515         sunw_PEM_write_bio_ASN1_stream;
2516         sunw_PEM_write_bio_CMS;
2517         sunw_PEM_write_bio_CMS_stream;
2518         sunw_PEM_write_bio_DHparams;
2519         sunw_PEM_write_bio_DSA_PUBKEY;
2520         sunw_PEM_write_bio_DSAparams;
2521         sunw_PEM_write_bio_DSAPrivateKey;
2522         sunw_PEM_write_bio_NETSCAPE_CERT_SEQUENCE;
2523         sunw_PEM_write_bio_Parameters;
2524         sunw_PEM_write_bio_PKCS7;
2525         sunw_PEM_write_bio_PKCS7_stream;
2526         sunw_PEM_write_bio_PKCS8;
2527         sunw_PEM_write_bio_PKCS8_PRIV_KEY_INFO;
2528         sunw_PEM_write_bio_PKCS8PrivateKey;
2529         sunw_PEM_write_bio_PKCS8PrivateKey_nid;
2530         sunw_PEM_write_bio_PrivateKey;
2531         sunw_PEM_write_bio_PUBKEY;
2532         sunw_PEM_write_bio_RSA_PUBKEY;
2533         sunw_PEM_write_bio_RSAPrivateKey;
2534         sunw_PEM_write_bio_RSAPublicKey;
2535         sunw_PEM_write_bio_X509;
2536         sunw_PEM_write_bio_X509_AUX;
2537         sunw_PEM_write_bio_X509_CERT_PAIR;
2538         sunw_PEM_write_bio_X509_CRL;
2539         sunw_PEM_write_bio_X509_REQ;
2540         sunw_PEM_write_bio_X509_REQ_NEW;
2541         sunw_PEM_write_CMS;
2542         sunw_PEM_write_DHparams;
2543         sunw_PEM_write_DSA_PUBKEY;
2544         sunw_PEM_write_DSAparams;
2545         sunw_PEM_write_DSAPrivateKey;
2546         sunw_PEM_write_NETSCAPE_CERT_SEQUENCE;
2547         sunw_PEM_write_PKCS7;
2548         sunw_PEM_write_PKCS8;
2549         sunw_PEM_write_PKCS8_PRIV_KEY_INFO;
2550         sunw_PEM_write_PKCS8PrivateKey;
2551         sunw_PEM_write_PKCS8PrivateKey_nid;
2552         sunw_PEM_write_PrivateKey;
2553         sunw_PEM_write_PUBKEY;
2554         sunw_PEM_write_RSA_PUBKEY;
2555         sunw_PEM_write_RSAPrivateKey;
2556         sunw_PEM_write_RSAPublicKey;
2557         sunw_PEM_write_X509;
2558         sunw_PEM_write_X509_AUX;
2559         sunw_PEM_write_X509_CERT_PAIR;
2560         sunw_PEM_write_X509_CRL;
2561         sunw_PEM_write_X509_REQ;
2562         sunw_PEM_write_X509_REQ_NEW;
2563         sunw_PEM_X509_INFO_read;
2564         sunw_PEM_X509_INFO_read_bio;
2565         sunw_PEM_X509_INFO_write_bio;
2566         sunw_pitem_free;
2567         sunw_pitem_new;
2568         sunw_pk11_active_add;
2569         sunw_pk11_active_delete;
```



```
2570         sunw_pk11_active_remove;
2571         sunw_pk11_destroy_dh_key_objects;
2572         sunw_pk11_destroy_dh_object;
2573         sunw_pk11_destroy_dsa_key_objects;
2574         sunw_pk11_destroy_dsa_object_priv;
2575         sunw_pk11_destroy_dsa_object_pub;
2576         sunw_pk11_destroy_rsa_key_objects;
2577         sunw_pk11_destroy_rsa_object_priv;
2578         sunw_pk11_destroy_rsa_object_pub;
2579         sunw_PK11_DH;
2580         sunw_PK11_DSA;
2581         sunw_pk11_free_active_list;
2582         sunw_pk11_get_session;
2583         sunw_pk11_load_privkey;
2584         sunw_pk11_load_pubkey;
2585         sunw_pk11_return_session;
2586         sunw_PK11_RSA;
2587         sunw_PK11err_add_data;
2588         sunw_PKCS1_MGF1;
2589         sunw_PKCS12_add_cert;
2590         sunw_PKCS12_add_CSPName_asc;
2591         sunw_PKCS12_add_friendlyname_asc;
2592         sunw_PKCS12_add_friendlyname_uni;
2593         sunw_PKCS12_add_key;
2594         sunw_PKCS12_add_localkeyid;
2595         sunw_PKCS12_add_safe;
2596         sunw_PKCS12_add_safes;
2597         sunw_PKCS12_AUTHSAFES_it;
2598         sunw_PKCS12_BAGS_free;
2599         sunw_PKCS12_BAGS_it;
2600         sunw_PKCS12_BAGS_new;
2601         sunw_PKCS12_certbag2x509;
2602         sunw_PKCS12_certbag2x509crl;
2603         sunw_PKCS12_create;
2604         sunw_PKCS12_decrypt_skey;
2605         sunw_PKCS12_free;
2606         sunw_PKCS12_gen_mac;
2607         sunw_PKCS12_get_attr_gen;
2608         sunw_PKCS12_get_friendlyname;
2609         sunw_PKCS12_init;
2610         sunw_PKCS12_it;
2611         sunw_PKCS12_item_decrypt_d2i;
2612         sunw_PKCS12_item_i2d_encrypt;
2613         sunw_PKCS12_item_pack_safebag;
2614         sunw_PKCS12_key_gen_asc;
2615         sunw_PKCS12_key_gen_uni;
2616         sunw_PKCS12_MAC_DATA_free;
2617         sunw_PKCS12_MAC_DATA_it;
2618         sunw_PKCS12_MAC_DATA_new;
2619         sunw_PKCS12_MAKE_KEYBAG;
2620         sunw_PKCS12_MAKE_SHKEYBAG;
2621         sunw_PKCS12_new;
2622         sunw_PKCS12_newpass;
2623         sunw_PKCS12_pack_authsafes;
2624         sunw_PKCS12_pack_p7data;
2625         sunw_PKCS12_pack_p7encdata;
2626         sunw_PKCS12_parse;
2627         sunw_PKCS12_PBE_add;
2628         sunw_PKCS12_pbe_crypt;
2629         sunw_PKCS12_PBE_keyivgen;
2630         sunw_PKCS12_SAFEBAG_free;
2631         sunw_PKCS12_SAFEBAG_it;
2632         sunw_PKCS12_SAFEBAG_new;
2633         sunw_PKCS12_SAFEBAGS_it;
2634         sunw_PKCS12_set_mac;
2635         sunw_PKCS12_setup_mac;
```

```
2636         sunw_PKCS12_unpack_authsafes;
2637         sunw_PKCS12_unpack_p7data;
2638         sunw_PKCS12_unpack_p7encdata;
2639         sunw_PKCS12_verify_mac;
2640         sunw_PKCS12_x5092certbag;
2641         sunw_PKCS12_x509crl2certbag;
2642         sunw_PKCS5_PBE_add;
2643         sunw_PKCS5_PBE_keyivgen;
2644         sunw_PKCS5_pbe_set;
2645         sunw_PKCS5_pbe_set0_algor;
2646         sunw_PKCS5_pbe2_set;
2647         sunw_PKCS5_pbe2_set_iv;
2648         sunw_PKCS5_PBKDF2_HMAC;
2649         sunw_PKCS5_PBKDF2_HMAC_SHA1;
2650         sunw_PKCS5_pbkdf2_set;
2651         sunw_PKCS5_v2_PBE_keyivgen;
2652         sunw_PKCS5_v2_PBKDF2_keyivgen;
2653         sunw_PKCS7_add_attrib_content_type;
2654         sunw_PKCS7_add_attrib_smimecap;
2655         sunw_PKCS7_add_attribute;
2656         sunw_PKCS7_add_certificate;
2657         sunw_PKCS7_add_crl;
2658         sunw_PKCS7_add_recipient;
2659         sunw_PKCS7_add_recipient_info;
2660         sunw_PKCS7_add_signature;
2661         sunw_PKCS7_add_signed_attribute;
2662         sunw_PKCS7_add_signer;
2663         sunw_PKCS7_add0_attrib_signing_time;
2664         sunw_PKCS7_add1_attrib_digest;
2665         sunw_PKCS7_ATTR_SIGN_it;
2666         sunw_PKCS7_ATTR_VERIFY_it;
2667         sunw_PKCS7_cert_from_signer_info;
2668         sunw_PKCS7_content_new;
2669         sunw_PKCS7_ctrl;
2670         sunw_PKCS7_dataDecode;
2671         sunw_PKCS7_dataFinal;
2672         sunw_PKCS7_dataInit;
2673         sunw_PKCS7_dataVerify;
2674         sunw_PKCS7_decrypt;
2675         sunw_PKCS7_DIGEST_free;
2676         sunw_PKCS7_digest_from_attributes;
2677         sunw_PKCS7_DIGEST_it;
2678         sunw_PKCS7_DIGEST_new;
2679         sunw_PKCS7_dup;
2680         sunw_PKCS7_ENC_CONTENT_free;
2681         sunw_PKCS7_ENC_CONTENT_it;
2682         sunw_PKCS7_ENC_CONTENT_new;
2683         sunw_PKCS7_encrypt;
2684         sunw_PKCS7_ENCRYPT_free;
2685         sunw_PKCS7_ENCRYPT_it;
2686         sunw_PKCS7_ENCRYPT_new;
2687         sunw_PKCS7_ENVELOPE_free;
2688         sunw_PKCS7_ENVELOPE_it;
2689         sunw_PKCS7_ENVELOPE_new;
2690         sunw_PKCS7_final;
2691         sunw_PKCS7_free;
2692         sunw_PKCS7_get_attribute;
2693         sunw_PKCS7_get_issuer_and_serial;
2694         sunw_PKCS7_get_signed_attribute;
2695         sunw_PKCS7_get_signer_info;
2696         sunw_PKCS7_get_smimecap;
2697         sunw_PKCS7_get0_signers;
2698         sunw_PKCS7_ISSUER_AND_SERIAL_digest;
2699         sunw_PKCS7_ISSUER_AND_SERIAL_free;
2700         sunw_PKCS7_ISSUER_AND_SERIAL_it;
2701         sunw_PKCS7_ISSUER_AND_SERIAL_new;
```



```
2702         sunw_PKCS7_it;
2703         sunw_PKCS7_new;
2704         sunw_PKCS7_print_ctx;
2705         sunw_PKCS7_RECIP_INFO_free;
2706         sunw_PKCS7_RECIP_INFO_get0_alg;
2707         sunw_PKCS7_RECIP_INFO_it;
2708         sunw_PKCS7_RECIP_INFO_new;
2709         sunw_PKCS7_RECIP_INFO_set;
2710         sunw_PKCS7_set_attributes;
2711         sunw_PKCS7_set_cipher;
2712         sunw_PKCS7_set_content;
2713         sunw_PKCS7_set_digest;
2714         sunw_PKCS7_set_signed_attributes;
2715         sunw_PKCS7_set_type;
2716         sunw_PKCS7_set0_type_other;
2717         sunw_PKCS7_sign;
2718         sunw_PKCS7_sign_add_signer;
2719         sunw_PKCS7_SIGN_ENVELOPE_free;
2720         sunw_PKCS7_SIGN_ENVELOPE_it;
2721         sunw_PKCS7_SIGN_ENVELOPE_new;
2722         sunw_PKCS7_signatureVerify;
2723         sunw_PKCS7_SIGNED_free;
2724         sunw_PKCS7_SIGNED_it;
2725         sunw_PKCS7_SIGNED_new;
2726         sunw_PKCS7_SIGNER_INFO_free;
2727         sunw_PKCS7_SIGNER_INFO_get0_algs;
2728         sunw_PKCS7_SIGNER_INFO_it;
2729         sunw_PKCS7_SIGNER_INFO_new;
2730         sunw_PKCS7_SIGNER_INFO_set;
2731         sunw_PKCS7_SIGNER_INFO_sign;
2732         sunw_PKCS7_simple_smimecap;
2733         sunw_PKCS7_stream;
2734         sunw_PKCS7_to_TS_TST_INFO;
2735         sunw_PKCS7_verify;
2736         sunw_PKCS8_add_keyusage;
2737         sunw_PKCS8_decrypt;
2738         sunw_PKCS8_encrypt;
2739         sunw_PKCS8_pkey_get0;
2740         sunw_PKCS8_pkey_set0;
2741         sunw_PKCS8_PRIV_KEY_INFO_free;
2742         sunw_PKCS8_PRIV_KEY_INFO_it;
2743         sunw_PKCS8_PRIV_KEY_INFO_new;
2744         sunw_PKCS8_set_broken;
2745         sunw_PKEY_USAGE_PERIOD_free;
2746         sunw_PKEY_USAGE_PERIOD_it;
2747         sunw_PKEY_USAGE_PERIOD_new;
2748         sunw_policy_cache_find_data;
2749         sunw_policy_cache_free;
2750         sunw_policy_cache_set;
2751         sunw_policy_cache_set_mapping;
2752         sunw_POLICY_CONSTRAINTS_free;
2753         sunw_POLICY_CONSTRAINTS_it;
2754         sunw_POLICY_CONSTRAINTS_new;
2755         sunw_policy_data_free;
2756         sunw_policy_data_new;
2757         sunw_POLICY_MAPPING_free;
2758         sunw_POLICY_MAPPING_it;
2759         sunw_POLICY_MAPPING_new;
2760         sunw_POLICY_MAPPINGS_it;
2761         sunw_policy_node_cmp_new;
2762         sunw_policy_node_free;
2763         sunw_policy_node_match;
2764         sunw_POLICYINFO_free;
2765         sunw_POLICYINFO_it;
2766         sunw_POLICYINFO_new;
2767         sunw_POLICYQUALINFO_free;
```

```
2768         sunw_POLICYQUALINFO_it;
2769         sunw_POLICYQUALINFO_new;
2770         sunw_pqueue_find;
2771         sunw_pqueue_free;
2772         sunw_pqueue_insert;
2773         sunw_pqueue_iterator;
2774         sunw_pqueue_new;
2775         sunw_pqueue_next;
2776         sunw_pqueue_peek;
2777         sunw_pqueue_pop;
2778         sunw_pqueue_print;
2779         sunw_pqueue_size;
2780         sunw_private_AES_set_decrypt_key;
2781         sunw_private_AES_set_encrypt_key;
2782         sunw_private_Camellia_set_key;
2783         sunw_private_RC4_set_key;
2784         sunw_PROXY_CERT_INFO_EXTENSION_free;
2785         sunw_PROXY_CERT_INFO_EXTENSION_it;
2786         sunw_PROXY_CERT_INFO_EXTENSION_new;
2787         sunw_PROXY_POLICY_free;
2788         sunw_PROXY_POLICY_it;
2789         sunw_PROXY_POLICY_new;
2790         sunw_RAND_add;
2791         sunw_RAND_bytes;
2792         sunw_RAND_cleanup;
2793         sunw_RAND_egd;
2794         sunw_RAND_egd_bytes;
2795         sunw_RAND_file_name;
2796         sunw_RAND_get_rand_method;
2797         sunw_RAND_load_file;
2798         sunw_RAND_poll;
2799         sunw_RAND_pseudo_bytes;
2800         sunw_RAND_query_egd_bytes;
2801         sunw_RAND_seed;
2802         sunw_RAND_set_rand_engine;
2803         sunw_RAND_set_rand_method;
2804         sunw_RAND_SSLeay;
2805         sunw_rand_ssleay_meth;
2806         sunw_RAND_status;
2807         sunw_RAND_version;
2808         sunw_RAND_write_file;
2809         sunw_RC2_cbc_encrypt;
2810         sunw_RC2_cfb64_encrypt;
2811         sunw_RC2_decrypt;
2812         sunw_RC2_ecb_encrypt;
2813         sunw_RC2_encrypt;
2814         sunw_RC2_ofb64_encrypt;
2815         sunw_RC2_set_key;
2816         sunw_RC2_version;
2817         sunw_RC4;
2818         sunw_RC4_options;
2819         sunw_RC4_set_key;
2820         sunw_RIPEMD160;
2821         sunw_RIPEMD160_Final;
2822         sunw_RIPEMD160_Init;
2823         sunw_RIPEMD160_Transform;
2824         sunw_RIPEMD160_Update;
2825         sunw_RMD160_version;
2826         sunw_rsa_asn1_meths;
2827         sunw_RSA_blinding_off;
2828         sunw_RSA_blinding_on;
2829         sunw_RSA_check_key;
2830         sunw_RSA_flags;
2831         sunw_RSA_free;
2832         sunw_RSA_generate_key;
2833         sunw_RSA_generate_key_ex;
```



```
2834         sunw_RSA_get_default_method;
2835         sunw_RSA_get_ex_data;
2836         sunw_RSA_get_ex_new_index;
2837         sunw_RSA_get_method;
2838         sunw_RSA_memory_lock;
2839         sunw_RSA_new;
2840         sunw_RSA_new_method;
2841         sunw_RSA_null_method;
2842         sunw_RSA_padding_add_none;
2843         sunw_RSA_padding_add_PKCS1_OAEP;
2844         sunw_RSA_padding_add_PKCS1_PSS;
2845         sunw_RSA_padding_add_PKCS1_PSS_mgf1;
2846         sunw_RSA_padding_add_PKCS1_type_1;
2847         sunw_RSA_padding_add_PKCS1_type_2;
2848         sunw_RSA_padding_add_SSLv23;
2849         sunw_RSA_padding_add_X931;
2850         sunw_RSA_padding_check_none;
2851         sunw_RSA_padding_check_PKCS1_OAEP;
2852         sunw_RSA_padding_check_PKCS1_type_1;
2853         sunw_RSA_padding_check_PKCS1_type_2;
2854         sunw_RSA_padding_check_SSLv23;
2855         sunw_RSA_padding_check_X931;
2856         sunw_RSA_PKCS1_SSLeay;
2857         sunw_rsa_pkey_meth;
2858         sunw_RSA_print;
2859         sunw_RSA_print_fp;
2860         sunw_RSA_private_decrypt;
2861         sunw_RSA_private_encrypt;
2862         sunw_RSA_PSS_PARAMS_free;
2863         sunw_RSA_PSS_PARAMS_it;
2864         sunw_RSA_PSS_PARAMS_new;
2865         sunw_RSA_public_decrypt;
2866         sunw_RSA_public_encrypt;
2867         sunw_RSA_set_default_method;
2868         sunw_RSA_set_ex_data;
2869         sunw_RSA_set_method;
2870         sunw_RSA_setup_blinding;
2871         sunw_RSA_sign;
2872         sunw_RSA_sign_ASN1_OCTET_STRING;
2873         sunw_RSA_size;
2874         sunw_RSA_up_ref;
2875         sunw_RSA_verify;
2876         sunw_RSA_verify_ASN1_OCTET_STRING;
2877         sunw_RSA_verify_PKCS1_PSS;
2878         sunw_RSA_verify_PKCS1_PSS_mgf1;
2879         sunw_RSA_version;
2880         sunw_RSA_X931_hash_id;
2881         sunw_RSAPrivateKey_dup;
2882         sunw_RSAPrivateKey_it;
2883         sunw_RSAPublicKey_dup;
2884         sunw_RSAPublicKey_it;
2885         sunw_s2i_ASN1_INTEGER;
2886         sunw_s2i_ASN1_OCTET_STRING;
2887         sunw_SHA;
2888         sunw_SHA_Final;
2889         sunw_SHA_Init;
2890         sunw_SHA_Transform;
2891         sunw_SHA_Update;
2892         sunw_SHA_version;
2893         sunw_SHA1;
2894         sunw_sha1_block_data_order;
2895         sunw_SHA1_Final;
2896         sunw_SHA1_Init;
2897         sunw_SHA1_Transform;
2898         sunw_SHA1_Update;
2899         sunw_SHA1_version;
```

```
2900         sunw_SHA224;
2901         sunw_SHA224_Final;
2902         sunw_SHA224_Init;
2903         sunw_SHA224_Update;
2904         sunw_SHA256;
2905         sunw_sha256_block_data_order;
2906         sunw_SHA256_Final;
2907         sunw_SHA256_Init;
2908         sunw_SHA256_Transform;
2909         sunw_SHA256_Update;
2910         sunw_SHA256_version;
2911         sunw_SHA384;
2912         sunw_SHA384_Final;
2913         sunw_SHA384_Init;
2914         sunw_SHA384_Update;
2915         sunw_SHA512;
2916         sunw_sha512_block_data_order;
2917         sunw_SHA512_Final;
2918         sunw_SHA512_Init;
2919         sunw_SHA512_Transform;
2920         sunw_SHA512_Update;
2921         sunw_SHA512_version;
2922         sunw_sk_delete;
2923         sunw_sk_delete_ptr;
2924         sunw_sk_dup;
2925         sunw_sk_find;
2926         sunw_sk_find_ex;
2927         sunw_sk_free;
2928         sunw_sk_insert;
2929         sunw_sk_is_sorted;
2930         sunw_sk_new;
2931         sunw_sk_new_null;
2932         sunw_sk_num;
2933         sunw_sk_pop;
2934         sunw_sk_pop_free;
2935         sunw_sk_push;
2936         sunw_sk_set;
2937         sunw_sk_set_cmp_func;
2938         sunw_sk_shift;
2939         sunw_sk_sort;
2940         sunw_sk_unshift;
2941         sunw_sk_value;
2942         sunw_sk_zero;
2943         sunw_SMIME_crlf_copy;
2944         sunw_SMIME_read_ASN1;
2945         sunw_SMIME_read_CMS;
2946         sunw_SMIME_read_PKCS7;
2947         sunw_SMIME_text;
2948         sunw_SMIME_write_ASN1;
2949         sunw_SMIME_write_CMS;
2950         sunw_SMIME_write_PKCS7;
2951         sunw_SRP_Calc_A;
2952         sunw_SRP_Calc_B;
2953         sunw_SRP_Calc_client_key;
2954         sunw_SRP_Calc_server_key;
2955         sunw_SRP_Calc_u;
2956         sunw_SRP_Calc_x;
2957         sunw_SRP_check_known_gN_param;
2958         sunw_SRP_create_verifier;
2959         sunw_SRP_create_verifier_BN;
2960         sunw_SRP_get_default_gN;
2961         sunw_SRP_VBASE_free;
2962         sunw_SRP_VBASE_get_by_user;
2963         sunw_SRP_VBASE_init;
2964         sunw_SRP_VBASE_new;
2965         sunw_SRP_Verify_A_mod_N;
```



```
2966         sunw_SRP_Verify_B_mod_N;
2967         sunw_SSLeay;
2968         sunw_SSLeay_version;
2969         sunw_STACK_version;
2970         sunw_string_to_hex;
2971         sunw_SXNET_add_id_asc;
2972         sunw_SXNET_add_id_INTEGER;
2973         sunw_SXNET_add_id_ulong;
2974         sunw_SXNET_free;
2975         sunw_SXNET_get_id_asc;
2976         sunw_SXNET_get_id_INTEGER;
2977         sunw_SXNET_get_id_ulong;
2978         sunw_SXNET_it;
2979         sunw_SXNET_new;
2980         sunw_SXNETID_free;
2981         sunw_SXNETID_it;
2982         sunw_SXNETID_new;
2983         sunw_tree_find_sk;
2984         sunw_TS_ACCURACY_dup;
2985         sunw_TS_ACCURACY_free;
2986         sunw_TS_ACCURACY_get_micros;
2987         sunw_TS_ACCURACY_get_millis;
2988         sunw_TS_ACCURACY_get_seconds;
2989         sunw_TS_ACCURACY_it;
2990         sunw_TS_ACCURACY_new;
2991         sunw_TS_ACCURACY_set_micros;
2992         sunw_TS_ACCURACY_set_millis;
2993         sunw_TS_ACCURACY_set_seconds;
2994         sunw_TS_ASN1_INTEGER_print_bio;
2995         sunw_TS_CONF_get_tsa_section;
2996         sunw_TS_CONF_load_cert;
2997         sunw_TS_CONF_load_certs;
2998         sunw_TS_CONF_load_key;
2999         sunw_TS_CONF_set_accuracy;
3000         sunw_TS_CONF_set_certs;
3001         sunw_TS_CONF_set_clock_precision_digits;
3002         sunw_TS_CONF_set_crypto_device;
3003         sunw_TS_CONF_set_def_policy;
3004         sunw_TS_CONF_set_default_engine;
3005         sunw_TS_CONF_set_digests;
3006         sunw_TS_CONF_set_ess_cert_id_chain;
3007         sunw_TS_CONF_set_ordering;
3008         sunw_TS_CONF_set_policies;
3009         sunw_TS_CONF_set_serial;
3010         sunw_TS_CONF_set_signer_cert;
3011         sunw_TS_CONF_set_signer_key;
3012         sunw_TS_CONF_set_tsa_name;
3013         sunw_TS_ext_print_bio;
3014         sunw_TS_MSG_IMPRINT_dup;
3015         sunw_TS_MSG_IMPRINT_free;
3016         sunw_TS_MSG_IMPRINT_get_algo;
3017         sunw_TS_MSG_IMPRINT_get_msg;
3018         sunw_TS_MSG_IMPRINT_it;
3019         sunw_TS_MSG_IMPRINT_new;
3020         sunw_TS_MSG_IMPRINT_print_bio;
3021         sunw_TS_MSG_IMPRINT_set_algo;
3022         sunw_TS_MSG_IMPRINT_set_msg;
3023         sunw_TS_OBJ_print_bio;
3024         sunw_TS_REQ_add_ext;
3025         sunw_TS_REQ_delete_ext;
3026         sunw_TS_REQ_dup;
3027         sunw_TS_REQ_ext_free;
3028         sunw_TS_REQ_free;
3029         sunw_TS_REQ_get_cert_req;
3030         sunw_TS_REQ_get_ext;
3031         sunw_TS_REQ_get_ext_by_critical;
```

```
3032         sunw_TS_REQ_get_ext_by_NID;
3033         sunw_TS_REQ_get_ext_by_OBJ;
3034         sunw_TS_REQ_get_ext_count;
3035         sunw_TS_REQ_get_ext_d2i;
3036         sunw_TS_REQ_get_exts;
3037         sunw_TS_REQ_get_msg_imprint;
3038         sunw_TS_REQ_get_nonce;
3039         sunw_TS_REQ_get_policy_id;
3040         sunw_TS_REQ_get_version;
3041         sunw_TS_REQ_it;
3042         sunw_TS_REQ_new;
3043         sunw_TS_REQ_print_bio;
3044         sunw_TS_REQ_set_cert_req;
3045         sunw_TS_REQ_set_msg_imprint;
3046         sunw_TS_REQ_set_nonce;
3047         sunw_TS_REQ_set_policy_id;
3048         sunw_TS_REQ_set_version;
3049         sunw_TS_REQ_to_TS_VERIFY_CTX;
3050         sunw_TS_RESP_create_response;
3051         sunw_TS_RESP_CTX_add_failure_info;
3052         sunw_TS_RESP_CTX_add_flags;
3053         sunw_TS_RESP_CTX_add_md;
3054         sunw_TS_RESP_CTX_add_policy;
3055         sunw_TS_RESP_CTX_free;
3056         sunw_TS_RESP_CTX_get_request;
3057         sunw_TS_RESP_CTX_get_tst_info;
3058         sunw_TS_RESP_CTX_new;
3059         sunw_TS_RESP_CTX_set_accuracy;
3060         sunw_TS_RESP_CTX_set_certs;
3061         sunw_TS_RESP_CTX_set_clock_precision_digits;
3062         sunw_TS_RESP_CTX_set_def_policy;
3063         sunw_TS_RESP_CTX_set_extension_cb;
3064         sunw_TS_RESP_CTX_set_serial_cb;
3065         sunw_TS_RESP_CTX_set_signer_cert;
3066         sunw_TS_RESP_CTX_set_signer_key;
3067         sunw_TS_RESP_CTX_set_status_info;
3068         sunw_TS_RESP_CTX_set_status_info_cond;
3069         sunw_TS_RESP_CTX_set_time_cb;
3070         sunw_TS_RESP_dup;
3071         sunw_TS_RESP_free;
3072         sunw_TS_RESP_get_status_info;
3073         sunw_TS_RESP_get_token;
3074         sunw_TS_RESP_get_tst_info;
3075         sunw_TS_RESP_it;
3076         sunw_TS_RESP_new;
3077         sunw_TS_RESP_print_bio;
3078         sunw_TS_RESP_set_status_info;
3079         sunw_TS_RESP_set_tst_info;
3080         sunw_TS_RESP_verify_response;
3081         sunw_TS_RESP_verify_signature;
3082         sunw_TS_RESP_verify_token;
3083         sunw_TS_STATUS_INFO_dup;
3084         sunw_TS_STATUS_INFO_free;
3085         sunw_TS_STATUS_INFO_it;
3086         sunw_TS_STATUS_INFO_new;
3087         sunw_TS_STATUS_INFO_print_bio;
3088         sunw_TS_TST_INFO_add_ext;
3089         sunw_TS_TST_INFO_delete_ext;
3090         sunw_TS_TST_INFO_dup;
3091         sunw_TS_TST_INFO_ext_free;
3092         sunw_TS_TST_INFO_free;
3093         sunw_TS_TST_INFO_get_accuracy;
3094         sunw_TS_TST_INFO_get_ext;
3095         sunw_TS_TST_INFO_get_ext_by_critical;
3096         sunw_TS_TST_INFO_get_ext_by_NID;
3097         sunw_TS_TST_INFO_get_ext_by_OBJ;
```



```
3098         sunw_TS_TST_INFO_get_ext_count;
3099         sunw_TS_TST_INFO_get_ext_d2i;
3100         sunw_TS_TST_INFO_get_exts;
3101         sunw_TS_TST_INFO_get_msg_imprint;
3102         sunw_TS_TST_INFO_get_nonce;
3103         sunw_TS_TST_INFO_get_ordering;
3104         sunw_TS_TST_INFO_get_policy_id;
3105         sunw_TS_TST_INFO_get_serial;
3106         sunw_TS_TST_INFO_get_time;
3107         sunw_TS_TST_INFO_get_tsa;
3108         sunw_TS_TST_INFO_get_version;
3109         sunw_TS_TST_INFO_it;
3110         sunw_TS_TST_INFO_new;
3111         sunw_TS_TST_INFO_print_bio;
3112         sunw_TS_TST_INFO_set_accuracy;
3113         sunw_TS_TST_INFO_set_msg_imprint;
3114         sunw_TS_TST_INFO_set_nonce;
3115         sunw_TS_TST_INFO_set_ordering;
3116         sunw_TS_TST_INFO_set_policy_id;
3117         sunw_TS_TST_INFO_set_serial;
3118         sunw_TS_TST_INFO_set_time;
3119         sunw_TS_TST_INFO_set_tsa;
3120         sunw_TS_TST_INFO_set_version;
3121         sunw_TS_VERIFY_CTX_cleanup;
3122         sunw_TS_VERIFY_CTX_free;
3123         sunw_TS_VERIFY_CTX_init;
3124         sunw_TS_VERIFY_CTX_new;
3125         sunw_TS_X509_ALGOR_print_bio;
3126         sunw_TXT_DB_create_index;
3127         sunw_TXT_DB_free;
3128         sunw_TXT_DB_get_by_index;
3129         sunw_TXT_DB_insert;
3130         sunw_TXT_DB_read;
3131         sunw_TXT_DB_version;
3132         sunw_TXT_DB_write;
3133         sunw_UI_add_error_string;
3134         sunw_UI_add_info_string;
3135         sunw_UI_add_input_boolean;
3136         sunw_UI_add_input_string;
3137         sunw_UI_add_user_data;
3138         sunw_UI_add_verify_string;
3139         sunw_UI_construct_prompt;
3140         sunw_UI_create_method;
3141         sunw_UI_ctrl;
3142         sunw_UI_destroy_method;
3143         sunw_UI_dup_error_string;
3144         sunw_UI_dup_info_string;
3145         sunw_UI_dup_input_boolean;
3146         sunw_UI_dup_input_string;
3147         sunw_UI_dup_verify_string;
3148         sunw_UI_free;
3149         sunw_UI_get_default_method;
3150         sunw_UI_get_ex_data;
3151         sunw_UI_get_ex_new_index;
3152         sunw_UI_get_input_flags;
3153         sunw_UI_get_method;
3154         sunw_UI_get_result_maxsize;
3155         sunw_UI_get_result_minsize;
3156         sunw_UI_get_string_type;
3157         sunw_UI_get0_action_string;
3158         sunw_UI_get0_output_string;
3159         sunw_UI_get0_result;
3160         sunw_UI_get0_result_string;
3161         sunw_UI_get0_test_string;
3162         sunw_UI_get0_user_data;
3163         sunw_UI_method_get_closer;
```

```
3164         sunw_UI_method_get_flusher;
3165         sunw_UI_method_get_opener;
3166         sunw_UI_method_get_prompt_constructor;
3167         sunw_UI_method_get_reader;
3168         sunw_UI_method_get_writer;
3169         sunw_UI_method_set_closer;
3170         sunw_UI_method_set_flusher;
3171         sunw_UI_method_set_opener;
3172         sunw_UI_method_set_prompt_constructor;
3173         sunw_UI_method_set_reader;
3174         sunw_UI_method_set_writer;
3175         sunw_UI_new;
3176         sunw_UI_new_method;
3177         sunw_UI_OpenSSL;
3178         sunw_UI_process;
3179         sunw_UI_set_default_method;
3180         sunw_UI_set_ex_data;
3181         sunw_UI_set_method;
3182         sunw_UI_set_result;
3183         sunw_UI_UTIL_read_pw;
3184         sunw_UI_UTIL_read_pw_string;
3185         sunw_USERNOTICE_free;
3186         sunw_USERNOTICE_it;
3187         sunw_USERNOTICE_new;
3188         sunw_UTF8_getc;
3189         sunw_UTF8_putc;
3190         sunw_v2i_ASN1_BIT_STRING;
3191         sunw_v2i_GENERAL_NAME;
3192         sunw_v2i_GENERAL_NAME_ex;
3193         sunw_v2i_GENERAL_NAMES;
3194         sunw_v3_akey_id;
3195         sunw_v3_alt;
3196         sunw_v3_bcons;
3197         sunw_v3_cpols;
3198         sunw_v3_crl_hold;
3199         sunw_v3_crl_invdate;
3200         sunw_v3_crl_num;
3201         sunw_v3_crl_reason;
3202         sunw_v3_crld;
3203         sunw_v3_delta_crl;
3204         sunw_v3_ext_ku;
3205         sunw_v3_freshest_crl;
3206         sunw_v3_idp;
3207         sunw_v3_info;
3208         sunw_v3_inhibit_anyp;
3209         sunw_v3_key_usage;
3210         sunw_v3_name_constraints;
3211         sunw_v3_ns_ia5_list;
3212         sunw_v3_nscert;
3213         sunw_v3_ocsp_accresp;
3214         sunw_v3_ocsp_acutoff;
3215         sunw_v3_ocsp_crlid;
3216         sunw_v3_ocsp_nocheck;
3217         sunw_v3_ocsp_nonce;
3218         sunw_v3_ocsp_serviceloc;
3219         sunw_v3_pci;
3220         sunw_v3_pkey_usage_period;
3221         sunw_v3_policy_constraints;
3222         sunw_v3_policy_mappings;
3223         sunw_v3_sinfo;
3224         sunw_v3_skey_id;
3225         sunw_v3_sxnet;
3226         sunw_vpaes_cbc_encrypt;
3227         sunw_vpaes_decrypt;
3228         sunw_vpaes_encrypt;
3229         sunw_vpaes_set_decrypt_key;
```



```
3230         sunw_vpaes_set_encrypt_key;
3231         sunw_X509_add_ext;
3232         sunw_X509_add1_ext_i2d;
3233         sunw_X509_add1_reject_object;
3234         sunw_X509_add1_trust_object;
3235         sunw_X509_ALGOR_dup;
3236         sunw_X509_ALGOR_free;
3237         sunw_X509_ALGOR_get0;
3238         sunw_X509_ALGOR_it;
3239         sunw_X509_ALGOR_new;
3240         sunw_X509_ALGOR_set_md;
3241         sunw_X509_ALGOR_set0;
3242         sunw_X509_ALGORS_it;
3243         sunw_X509_alias_get0;
3244         sunw_X509_alias_set1;
3245         sunw_X509_ATTRIBUTE_count;
3246         sunw_X509_ATTRIBUTE_create;
3247         sunw_X509_ATTRIBUTE_create_by_NID;
3248         sunw_X509_ATTRIBUTE_create_by_OBJ;
3249         sunw_X509_ATTRIBUTE_create_by_txt;
3250         sunw_X509_ATTRIBUTE_dup;
3251         sunw_X509_ATTRIBUTE_free;
3252         sunw_X509_ATTRIBUTE_get0_data;
3253         sunw_X509_ATTRIBUTE_get0_object;
3254         sunw_X509_ATTRIBUTE_get0_type;
3255         sunw_X509_ATTRIBUTE_it;
3256         sunw_X509_ATTRIBUTE_new;
3257         sunw_X509_ATTRIBUTE_SET_it;
3258         sunw_X509_ATTRIBUTE_set1_data;
3259         sunw_X509_ATTRIBUTE_set1_object;
3260         sunw_X509_CERT_AUX_free;
3261         sunw_X509_CERT_AUX_it;
3262         sunw_X509_CERT_AUX_new;
3263         sunw_X509_CERT_AUX_print;
3264         sunw_X509_CERT_PAIR_free;
3265         sunw_X509_CERT_PAIR_it;
3266         sunw_X509_CERT_PAIR_new;
3267         sunw_X509_certificate_type;
3268         sunw_X509_check_akid;
3269         sunw_X509_check_ca;
3270         sunw_X509_check_issued;
3271         sunw_X509_check_private_key;
3272         sunw_X509_check_purpose;
3273         sunw_X509_check_trust;
3274         sunw_X509_CINF_free;
3275         sunw_X509_CINF_it;
3276         sunw_X509_CINF_new;
3277         sunw_X509_cmp;
3278         sunw_X509_cmp_current_time;
3279         sunw_X509_cmp_time;
3280         sunw_X509_CRL_add_ext;
3281         sunw_X509_CRL_add0_revoked;
3282         sunw_X509_CRL_add1_ext_i2d;
3283         sunw_X509_CRL_cmp;
3284         sunw_X509_CRL_delete_ext;
3285         sunw_X509_CRL_digest;
3286         sunw_X509_CRL_dup;
3287         sunw_X509_CRL_free;
3288         sunw_X509_CRL_get_ext;
3289         sunw_X509_CRL_get_ext_by_critical;
3290         sunw_X509_CRL_get_ext_by_NID;
3291         sunw_X509_CRL_get_ext_by_OBJ;
3292         sunw_X509_CRL_get_ext_count;
3293         sunw_X509_CRL_get_ext_d2i;
3294         sunw_X509_CRL_get_meth_data;
3295         sunw_X509_CRL_get0_by_cert;
```

```
3296         sunw_X509_CRL_get0_by_serial;
3297         sunw_X509_CRL_INFO_free;
3298         sunw_X509_CRL_INFO_it;
3299         sunw_X509_CRL_INFO_new;
3300         sunw_X509_CRL_it;
3301         sunw_X509_CRL_match;
3302         sunw_X509_CRL_METHOD_free;
3303         sunw_X509_CRL_METHOD_new;
3304         sunw_X509_CRL_new;
3305         sunw_X509_CRL_print;
3306         sunw_X509_CRL_print_fp;
3307         sunw_X509_CRL_set_default_method;
3308         sunw_X509_CRL_set_issuer_name;
3309         sunw_X509_CRL_set_lastUpdate;
3310         sunw_X509_CRL_set_meth_data;
3311         sunw_X509_CRL_set_nextUpdate;
3312         sunw_X509_CRL_set_version;
3313         sunw_X509_CRL_sign;
3314         sunw_X509_CRL_sign_ctx;
3315         sunw_X509_CRL_sort;
3316         sunw_X509_CRL_verify;
3317         sunw_X509_delete_ext;
3318         sunw_X509_digest;
3319         sunw_x509_dir_lookup;
3320         sunw_X509_dup;
3321         sunw_X509_email_free;
3322         sunw_X509_EXTENSION_create_by_NID;
3323         sunw_X509_EXTENSION_create_by_OBJ;
3324         sunw_X509_EXTENSION_dup;
3325         sunw_X509_EXTENSION_free;
3326         sunw_X509_EXTENSION_get_critical;
3327         sunw_X509_EXTENSION_get_data;
3328         sunw_X509_EXTENSION_get_object;
3329         sunw_X509_EXTENSION_it;
3330         sunw_X509_EXTENSION_new;
3331         sunw_X509_EXTENSION_set_critical;
3332         sunw_X509_EXTENSION_set_data;
3333         sunw_X509_EXTENSION_set_object;
3334         sunw_X509_EXTENSIONS_it;
3335         sunw_x509_file_lookup;
3336         sunw_X509_find_by_issuer_and_serial;
3337         sunw_X509_find_by_subject;
3338         sunw_X509_free;
3339         sunw_X509_get_default_cert_area;
3340         sunw_X509_get_default_cert_dir;
3341         sunw_X509_get_default_cert_dir_env;
3342         sunw_X509_get_default_cert_file;
3343         sunw_X509_get_default_cert_file_env;
3344         sunw_X509_get_default_private_dir;
3345         sunw_X509_get_ex_data;
3346         sunw_X509_get_ex_new_index;
3347         sunw_X509_get_ext;
3348         sunw_X509_get_ext_by_critical;
3349         sunw_X509_get_ext_by_NID;
3350         sunw_X509_get_ext_by_OBJ;
3351         sunw_X509_get_ext_count;
3352         sunw_X509_get_ext_d2i;
3353         sunw_X509_get_issuer_name;
3354         sunw_X509_get_pubkey;
3355         sunw_X509_get_pubkey_parameters;
3356         sunw_X509_get_serialNumber;
3357         sunw_X509_get_subject_name;
3358         sunw_X509_get0_pubkey_bitstr;
3359         sunw_X509_get1_email;
3360         sunw_X509_get1_ocsp;
3361         sunw_X509_gmtime_adj;
```



```
3362         sunw_X509_INFO_free;
3363         sunw_X509_INFO_new;
3364         sunw_X509_issuer_and_serial_cmp;
3365         sunw_X509_issuer_and_serial_hash;
3366         sunw_X509_issuer_name_cmp;
3367         sunw_X509_issuer_name_hash;
3368         sunw_X509_issuer_name_hash_old;
3369         sunw_X509_it;
3370         sunw_X509_keyid_get0;
3371         sunw_X509_keyid_set1;
3372         sunw_X509_load_cert_crl_file;
3373         sunw_X509_load_cert_file;
3374         sunw_X509_load_crl_file;
3375         sunw_X509_LOOKUP_by_alias;
3376         sunw_X509_LOOKUP_by_fingerprint;
3377         sunw_X509_LOOKUP_by_issuer_serial;
3378         sunw_X509_LOOKUP_by_subject;
3379         sunw_X509_LOOKUP_ctrl;
3380         sunw_X509_LOOKUP_file;
3381         sunw_X509_LOOKUP_free;
3382         sunw_X509_LOOKUP_hash_dir;
3383         sunw_X509_LOOKUP_init;
3384         sunw_X509_LOOKUP_new;
3385         sunw_X509_LOOKUP_shutdown;
3386         sunw_X509_NAME_add_entry;
3387         sunw_X509_NAME_add_entry_by_NID;
3388         sunw_X509_NAME_add_entry_by_OBJ;
3389         sunw_X509_NAME_add_entry_by_txt;
3390         sunw_X509_NAME_cmp;
3391         sunw_X509_NAME_delete_entry;
3392         sunw_X509_NAME_digest;
3393         sunw_X509_NAME_dup;
3394         sunw_X509_NAME_ENTRIES_it;
3395         sunw_X509_NAME_entry_count;
3396         sunw_X509_NAME_ENTRY_create_by_NID;
3397         sunw_X509_NAME_ENTRY_create_by_OBJ;
3398         sunw_X509_NAME_ENTRY_create_by_txt;
3399         sunw_X509_NAME_ENTRY_dup;
3400         sunw_X509_NAME_ENTRY_free;
3401         sunw_X509_NAME_ENTRY_get_data;
3402         sunw_X509_NAME_ENTRY_get_object;
3403         sunw_X509_NAME_ENTRY_it;
3404         sunw_X509_NAME_ENTRY_new;
3405         sunw_X509_NAME_ENTRY_set_data;
3406         sunw_X509_NAME_ENTRY_set_object;
3407         sunw_x509_name_ff;
3408         sunw_X509_NAME_free;
3409         sunw_X509_NAME_get_entry;
3410         sunw_X509_NAME_get_index_by_NID;
3411         sunw_X509_NAME_get_index_by_OBJ;
3412         sunw_X509_NAME_get_text_by_NID;
3413         sunw_X509_NAME_get_text_by_OBJ;
3414         sunw_X509_NAME_hash;
3415         sunw_X509_NAME_hash_old;
3416         sunw_X509_NAME_INTERNAL_it;
3417         sunw_X509_NAME_it;
3418         sunw_X509_NAME_new;
3419         sunw_X509_NAME_oneline;
3420         sunw_X509_NAME_print;
3421         sunw_X509_NAME_print_ex;
3422         sunw_X509_NAME_print_ex_fp;
3423         sunw_X509_NAME_set;
3424         sunw_X509_new;
3425         sunw_X509_OBJECT_free_contents;
3426         sunw_X509_OBJECT_idx_by_subject;
3427         sunw_X509_OBJECT_retrieve_by_subject;
```

```
3428         sunw_X509_OBJECT_retrieve_match;
3429         sunw_X509_OBJECT_up_ref_count;
3430         sunw_X509_ocspid_print;
3431         sunw_X509_PKEY_free;
3432         sunw_X509_PKEY_new;
3433         sunw_X509_policy_check;
3434         sunw_X509_policy_level_get0_node;
3435         sunw_X509_policy_level_node_count;
3436         sunw_X509_policy_node_get0_parent;
3437         sunw_X509_policy_node_get0_policy;
3438         sunw_X509_policy_node_get0_qualifiers;
3439         sunw_X509_POLICY_NODE_print;
3440         sunw_X509_policy_tree_free;
3441         sunw_X509_policy_tree_get0_level;
3442         sunw_X509_policy_tree_get0_policies;
3443         sunw_X509_policy_tree_get0_user_policies;
3444         sunw_X509_policy_tree_level_count;
3445         sunw_X509_print;
3446         sunw_X509_print_ex;
3447         sunw_X509_print_ex_fp;
3448         sunw_X509_print_fp;
3449         sunw_X509_pubkey_digest;
3450         sunw_X509_PUBKEY_free;
3451         sunw_X509_PUBKEY_get;
3452         sunw_X509_PUBKEY_get0_param;
3453         sunw_X509_PUBKEY_it;
3454         sunw_X509_PUBKEY_new;
3455         sunw_X509_PUBKEY_set;
3456         sunw_X509_PUBKEY_set0_param;
3457         sunw_X509_PURPOSE_add;
3458         sunw_X509_PURPOSE_cleanup;
3459         sunw_X509_PURPOSE_get_by_id;
3460         sunw_X509_PURPOSE_get_by_sname;
3461         sunw_X509_PURPOSE_get_count;
3462         sunw_X509_PURPOSE_get_id;
3463         sunw_X509_PURPOSE_get_trust;
3464         sunw_X509_PURPOSE_get0;
3465         sunw_X509_PURPOSE_get0_name;
3466         sunw_X509_PURPOSE_get0_sname;
3467         sunw_X509_PURPOSE_set;
3468         sunw_X509_reject_clear;
3469         sunw_X509_REQ_add_extensions;
3470         sunw_X509_REQ_add_extensions_nid;
3471         sunw_X509_REQ_add1_attr;
3472         sunw_X509_REQ_add1_attr_by_NID;
3473         sunw_X509_REQ_add1_attr_by_OBJ;
3474         sunw_X509_REQ_add1_attr_by_txt;
3475         sunw_X509_REQ_check_private_key;
3476         sunw_X509_REQ_delete_attr;
3477         sunw_X509_REQ_digest;
3478         sunw_X509_REQ_dup;
3479         sunw_X509_REQ_extension_nid;
3480         sunw_X509_REQ_free;
3481         sunw_X509_REQ_get_attr;
3482         sunw_X509_REQ_get_attr_by_NID;
3483         sunw_X509_REQ_get_attr_by_OBJ;
3484         sunw_X509_REQ_get_attr_count;
3485         sunw_X509_REQ_get_extension_nids;
3486         sunw_X509_REQ_get_extensions;
3487         sunw_X509_REQ_get_pubkey;
3488         sunw_X509_REQ_get1_email;
3489         sunw_X509_REQ_INFO_free;
3490         sunw_X509_REQ_INFO_it;
3491         sunw_X509_REQ_INFO_new;
3492         sunw_X509_REQ_it;
3493         sunw_X509_REQ_new;
```



```
3494         sunw_X509_REQ_print;
3495         sunw_X509_REQ_print_ex;
3496         sunw_X509_REQ_print_fp;
3497         sunw_X509_REQ_set_extension_nids;
3498         sunw_X509_REQ_set_pubkey;
3499         sunw_X509_REQ_set_subject_name;
3500         sunw_X509_REQ_set_version;
3501         sunw_X509_REQ_sign;
3502         sunw_X509_REQ_sign_ctx;
3503         sunw_X509_REQ_to_X509;
3504         sunw_X509_REQ_verify;
3505         sunw_X509_REVOKED_add_ext;
3506         sunw_X509_REVOKED_add1_ext_i2d;
3507         sunw_X509_REVOKED_delete_ext;
3508         sunw_X509_REVOKED_free;
3509         sunw_X509_REVOKED_get_ext;
3510         sunw_X509_REVOKED_get_ext_by_critical;
3511         sunw_X509_REVOKED_get_ext_by_NID;
3512         sunw_X509_REVOKED_get_ext_by_OBJ;
3513         sunw_X509_REVOKED_get_ext_count;
3514         sunw_X509_REVOKED_get_ext_d2i;
3515         sunw_X509_REVOKED_it;
3516         sunw_X509_REVOKED_new;
3517         sunw_X509_REVOKED_set_revocationDate;
3518         sunw_X509_REVOKED_set_serialNumber;
3519         sunw_X509_set_ex_data;
3520         sunw_X509_set_issuer_name;
3521         sunw_X509_set_notAfter;
3522         sunw_X509_set_notBefore;
3523         sunw_X509_set_pubkey;
3524         sunw_X509_set_serialNumber;
3525         sunw_X509_set_subject_name;
3526         sunw_X509_set_version;
3527         sunw_X509_SIG_free;
3528         sunw_X509_SIG_it;
3529         sunw_X509_SIG_new;
3530         sunw_X509_sign;
3531         sunw_X509_sign_ctx;
3532         sunw_X509_signature_dump;
3533         sunw_X509_signature_print;
3534         sunw_X509_STORE_add_cert;
3535         sunw_X509_STORE_add_crl;
3536         sunw_X509_STORE_add_lookup;
3537         sunw_X509_STORE_CTX_cleanup;
3538         sunw_X509_STORE_CTX_free;
3539         sunw_X509_STORE_CTX_get_chain;
3540         sunw_X509_STORE_CTX_get_current_cert;
3541         sunw_X509_STORE_CTX_get_error;
3542         sunw_X509_STORE_CTX_get_error_depth;
3543         sunw_X509_STORE_CTX_get_ex_data;
3544         sunw_X509_STORE_CTX_get_ex_new_index;
3545         sunw_X509_STORE_CTX_get_explicit_policy;
3546         sunw_X509_STORE_CTX_get0_current_crl;
3547         sunw_X509_STORE_CTX_get0_current_issuer;
3548         sunw_X509_STORE_CTX_get0_param;
3549         sunw_X509_STORE_CTX_get0_parent_ctx;
3550         sunw_X509_STORE_CTX_get0_policy_tree;
3551         sunw_X509_STORE_CTX_get1_chain;
3552         sunw_X509_STORE_CTX_get1_issuer;
3553         sunw_X509_STORE_CTX_init;
3554         sunw_X509_STORE_CTX_new;
3555         sunw_X509_STORE_CTX_purpose_inherit;
3556         sunw_X509_STORE_CTX_set_cert;
3557         sunw_X509_STORE_CTX_set_chain;
3558         sunw_X509_STORE_CTX_set_default;
3559         sunw_X509_STORE_CTX_set_depth;
```

```
3560         sunw_X509_STORE_CTX_set_error;
3561         sunw_X509_STORE_CTX_set_ex_data;
3562         sunw_X509_STORE_CTX_set_flags;
3563         sunw_X509_STORE_CTX_set_purpose;
3564         sunw_X509_STORE_CTX_set_time;
3565         sunw_X509_STORE_CTX_set_trust;
3566         sunw_X509_STORE_CTX_set_verify_cb;
3567         sunw_X509_STORE_CTX_set0_crls;
3568         sunw_X509_STORE_CTX_set0_param;
3569         sunw_X509_STORE_CTX_trusted_stack;
3570         sunw_X509_STORE_free;
3571         sunw_X509_STORE_get_by_subject;
3572         sunw_X509_STORE_get1_certs;
3573         sunw_X509_STORE_get1_crls;
3574         sunw_X509_STORE_load_locations;
3575         sunw_X509_STORE_new;
3576         sunw_X509_STORE_set_default_paths;
3577         sunw_X509_STORE_set_depth;
3578         sunw_X509_STORE_set_flags;
3579         sunw_X509_STORE_set_purpose;
3580         sunw_X509_STORE_set_trust;
3581         sunw_X509_STORE_set_verify_cb;
3582         sunw_X509_STORE_set1_param;
3583         sunw_X509_subject_name_cmp;
3584         sunw_X509_subject_name_hash;
3585         sunw_X509_subject_name_hash_old;
3586         sunw_X509_supported_extension;
3587         sunw_X509_time_adj;
3588         sunw_X509_time_adj_ex;
3589         sunw_X509_to_X509_REQ;
3590         sunw_X509_TRUST_add;
3591         sunw_X509_TRUST_cleanup;
3592         sunw_X509_trust_clear;
3593         sunw_X509_TRUST_get_by_id;
3594         sunw_X509_TRUST_get_count;
3595         sunw_X509_TRUST_get_flags;
3596         sunw_X509_TRUST_get_trust;
3597         sunw_X509_TRUST_get0;
3598         sunw_X509_TRUST_get0_name;
3599         sunw_X509_TRUST_set;
3600         sunw_X509_TRUST_set_default;
3601         sunw_X509_VAL_free;
3602         sunw_X509_VAL_it;
3603         sunw_X509_VAL_new;
3604         sunw_X509_verify;
3605         sunw_X509_verify_cert;
3606         sunw_X509_verify_cert_error_string;
3607         sunw_X509_VERIFY_PARAM_add0_policy;
3608         sunw_X509_VERIFY_PARAM_add0_table;
3609         sunw_X509_VERIFY_PARAM_clear_flags;
3610         sunw_X509_VERIFY_PARAM_free;
3611         sunw_X509_VERIFY_PARAM_get_depth;
3612         sunw_X509_VERIFY_PARAM_get_flags;
3613         sunw_X509_VERIFY_PARAM_inherit;
3614         sunw_X509_VERIFY_PARAM_lookup;
3615         sunw_X509_VERIFY_PARAM_new;
3616         sunw_X509_VERIFY_PARAM_set_depth;
3617         sunw_X509_VERIFY_PARAM_set_flags;
3618         sunw_X509_VERIFY_PARAM_set_purpose;
3619         sunw_X509_VERIFY_PARAM_set_time;
3620         sunw_X509_VERIFY_PARAM_set_trust;
3621         sunw_X509_VERIFY_PARAM_set1;
3622         sunw_X509_VERIFY_PARAM_set1_name;
3623         sunw_X509_VERIFY_PARAM_set1_policies;
3624         sunw_X509_VERIFY_PARAM_table_cleanup;
3625         sunw_X509_version;
```



```
3626         sunw_X509at_add1_attr;
3627         sunw_X509at_add1_attr_by_NID;
3628         sunw_X509at_add1_attr_by_OBJ;
3629         sunw_X509at_add1_attr_by_txt;
3630         sunw_X509at_delete_attr;
3631         sunw_X509at_get_attr;
3632         sunw_X509at_get_attr_by_NID;
3633         sunw_X509at_get_attr_by_OBJ;
3634         sunw_X509at_get_attr_count;
3635         sunw_X509at_get0_data_by_OBJ;
3636         sunw_X509v3_add_ext;
3637         sunw_X509V3_add_standard_extensions;
3638         sunw_X509V3_add_value;
3639         sunw_X509V3_add_value_bool;
3640         sunw_X509V3_add_value_bool_nf;
3641         sunw_X509V3_add_value_int;
3642         sunw_X509V3_add_value_uchar;
3643         sunw_X509V3_add1_i2d;
3644         sunw_X509V3_conf_free;
3645         sunw_X509v3_delete_ext;
3646         sunw_X509V3_EXT_add;
3647         sunw_X509V3_EXT_add_alias;
3648         sunw_X509V3_EXT_add_conf;
3649         sunw_X509V3_EXT_add_list;
3650         sunw_X509V3_EXT_add_nconf;
3651         sunw_X509V3_EXT_add_nconf_sk;
3652         sunw_X509V3_EXT_cleanup;
3653         sunw_X509V3_EXT_conf;
3654         sunw_X509V3_EXT_conf_nid;
3655         sunw_X509V3_EXT_CRL_add_conf;
3656         sunw_X509V3_EXT_CRL_add_nconf;
3657         sunw_X509V3_EXT_d2i;
3658         sunw_X509V3_EXT_get;
3659         sunw_X509V3_EXT_get_nid;
3660         sunw_X509V3_EXT_i2d;
3661         sunw_X509V3_EXT_nconf;
3662         sunw_X509V3_EXT_nconf_nid;
3663         sunw_X509V3_EXT_print;
3664         sunw_X509V3_EXT_print_fp;
3665         sunw_X509V3_EXT_REQ_add_conf;
3666         sunw_X509V3_EXT_REQ_add_nconf;
3667         sunw_X509V3_EXT_val_prn;
3668         sunw_X509V3_extensions_print;
3669         sunw_X509V3_get_d2i;
3670         sunw_X509v3_get_ext;
3671         sunw_X509v3_get_ext_by_critical;
3672         sunw_X509v3_get_ext_by_NID;
3673         sunw_X509v3_get_ext_by_OBJ;
3674         sunw_X509v3_get_ext_count;
3675         sunw_X509V3_get_section;
3676         sunw_X509V3_get_string;
3677         sunw_X509V3_get_value_bool;
3678         sunw_X509V3_get_value_int;
3679         sunw_X509V3_NAME_from_section;
3680         sunw_X509V3_parse_list;
3681         sunw_X509V3_section_free;
3682         sunw_X509V3_set_conf_lhash;
3683         sunw_X509V3_set_ctx;
3684         sunw_X509V3_set_nconf;
3685         sunw_X509V3_string_free;
3686         sunw_ZLONG_it;
3687     local:
3688         *;
3689 };
3690 #endif /* ! codereview */
```

```
**********************************************************
    7342 Wed Aug 13 19:52:53 2014
new/usr/src/lib/openssl/libsunw_crypto/md2/md2_dgst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md2/md2_dgst.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```



```
#include <openssl/md2.h>
#include <openssl/opensslv.h>
#include <openssl/crypto.h>

const char MD2_version[]="MD2" OPENSSL_VERSION_PTEXT;

/* Implemented from RFC1319 The MD2 Message-Digest Algorithm
 */

#define UCHAR   unsigned char

static void md2_block(MD2_CTX *c, const unsigned char *d);
/* The magic S table - I have converted it to hex since it is
 * basically just a random byte string. */
static const MD2_INT S[256]={
        0x29, 0x2E, 0x43, 0xC9, 0xA2, 0xD8, 0x7C, 0x01,
        0x3D, 0x36, 0x54, 0xA1, 0xEC, 0xF0, 0x06, 0x13,
        0x62, 0xA7, 0x05, 0xF3, 0xC0, 0xC7, 0x73, 0x8C,
        0x98, 0x93, 0x2B, 0xD9, 0xBC, 0x4C, 0x82, 0xCA,
        0x1E, 0x9B, 0x57, 0x3C, 0xFD, 0xD4, 0xE0, 0x16,
        0x67, 0x42, 0x6F, 0x18, 0x8A, 0x17, 0xE5, 0x12,
        0xBE, 0x4E, 0xC4, 0xD6, 0xDA, 0x9E, 0xDE, 0x49,
        0xA0, 0xFB, 0xF5, 0x8E, 0xBB, 0x2F, 0xEE, 0x7A,
        0xA9, 0x68, 0x79, 0x91, 0x15, 0xB2, 0x07, 0x3F,
        0x94, 0xC2, 0x10, 0x89, 0x0B, 0x22, 0x5F, 0x21,
        0x80, 0x7F, 0x5D, 0x9A, 0x5A, 0x90, 0x32, 0x27,
        0x35, 0x3E, 0xCC, 0xE7, 0xBF, 0xF7, 0x97, 0x03,
        0xFF, 0x19, 0x30, 0xB3, 0x48, 0xA5, 0xB5, 0xD1,
        0xD7, 0x5E, 0x92, 0x2A, 0xAC, 0x56, 0xAA, 0xC6,
        0x4F, 0xB8, 0x38, 0xD2, 0x96, 0xA4, 0x7D, 0xB6,
        0x76, 0xFC, 0x6B, 0xE2, 0x9C, 0x74, 0x04, 0xF1,
        0x45, 0x9D, 0x70, 0x59, 0x64, 0x71, 0x87, 0x20,
        0x86, 0x5B, 0xCF, 0x65, 0xE6, 0x2D, 0xA8, 0x02,
        0x1B, 0x60, 0x25, 0xAD, 0xAE, 0xB0, 0xB9, 0xF6,
        0x1C, 0x46, 0x61, 0x69, 0x34, 0x40, 0x7E, 0x0F,
        0x55, 0x47, 0xA3, 0x23, 0xDD, 0x51, 0xAF, 0x3A,
        0xC3, 0x5C, 0xF9, 0xCE, 0xBA, 0xC5, 0xEA, 0x26,
        0x2C, 0x53, 0x0D, 0x6E, 0x85, 0x28, 0x84, 0x09,
        0xD3, 0xDF, 0xCD, 0xF4, 0x41, 0x81, 0x4D, 0x52,
        0x6A, 0xDC, 0x37, 0xC8, 0x6C, 0xC1, 0xAB, 0xFA,
        0x24, 0xE1, 0x7B, 0x08, 0x0C, 0xBD, 0xB1, 0x4A,
        0x78, 0x88, 0x95, 0x8B, 0xE3, 0x63, 0xE8, 0x6D,
        0xE9, 0xCB, 0xD5, 0xFE, 0x3B, 0x00, 0x1D, 0x39,
        0xF2, 0xEF, 0xB7, 0x0E, 0x66, 0x58, 0xD0, 0xE4,
        0xA6, 0x77, 0x72, 0xF8, 0xEB, 0x75, 0x4B, 0x0A,
        0x31, 0x44, 0x50, 0xB4, 0x8F, 0xED, 0x1F, 0x1A,
        0xDB, 0x99, 0x8D, 0x33, 0x9F, 0x11, 0x83, 0x14,
        };

const char *MD2_options(void)
        {
        if (sizeof(MD2_INT) == 1)
                return("md2(char)");
        else
                return("md2(int)");
        }

fips_md_init(MD2)
        {
        c->num=0;
        memset(c->state,0,sizeof c->state);
        memset(c->cksm,0,sizeof c->cksm);
        memset(c->data,0,sizeof c->data);
        return 1;
        }
```

```
128 int MD2_Update(MD2_CTX *c, const unsigned char *data, size_t len)
129         {
130         register UCHAR *p;

132         if (len == 0) return 1;

134         p=c->data;
135         if (c->num != 0)
136                 {
137                 if ((c->num+len) >= MD2_BLOCK)
138                         {
139                         memcpy(&(p[c->num]),data,MD2_BLOCK-c->num);
140                         md2_block(c,c->data);
141                         data+=(MD2_BLOCK - c->num);
142                         len-=(MD2_BLOCK - c->num);
143                         c->num=0;
144                         /* drop through and do the rest */
145                         }
146                 else
147                         {
148                         memcpy(&(p[c->num]),data,len);
149                         /* data+=len; */
150                         c->num+=(int)len;
151                         return 1;
152                         }
153                 }
154         /* we now can process the input data in blocks of MD2_BLOCK
155          * chars and save the leftovers to c->data. */
156         while (len >= MD2_BLOCK)
157                 {
158                 md2_block(c,data);
159                 data+=MD2_BLOCK;
160                 len-=MD2_BLOCK;
161                 }
162         memcpy(p,data,len);
163         c->num=(int)len;
164         return 1;
165         }

167 static void md2_block(MD2_CTX *c, const unsigned char *d)
168         {
169         register MD2_INT t,*sp1,*sp2;
170         register int i,j;
171         MD2_INT state[48];

173         sp1=c->state;
174         sp2=c->cksm;
175         j=sp2[MD2_BLOCK-1];
176         for (i=0; i<16; i++)
177                 {
178                 state[i]=sp1[i];
179                 state[i+16]=t=d[i];
180                 state[i+32]=(t^sp1[i]);
181                 j=sp2[i]^=S[t^j];
182                 }
183         t=0;
184         for (i=0; i<18; i++)
185                 {
186                 for (j=0; j<48; j+=8)
187                         {
188                         t= state[j+ 0]^=S[t];
189                         t= state[j+ 1]^=S[t];
190                         t= state[j+ 2]^=S[t];
191                         t= state[j+ 3]^=S[t];
192                         t= state[j+ 4]^=S[t];
193                         t= state[j+ 5]^=S[t];
```



```
194                         t= state[j+ 6]^=S[t];
195                         t= state[j+ 7]^=S[t];
196                         }
197                 t=(t+i)&0xff;
198                 }
199         memcpy(sp1,state,16*sizeof(MD2_INT));
200         OPENSSL_cleanse(state,48*sizeof(MD2_INT));
201         }

203 int MD2_Final(unsigned char *md, MD2_CTX *c)
204         {
205         int i,v;
206         register UCHAR *cp;
207         register MD2_INT *p1,*p2;

209         cp=c->data;
210         p1=c->state;
211         p2=c->cksm;
212         v=MD2_BLOCK-c->num;
213         for (i=c->num; i<MD2_BLOCK; i++)
214                 cp[i]=(UCHAR)v;

216         md2_block(c,cp);

218         for (i=0; i<MD2_BLOCK; i++)
219                 cp[i]=(UCHAR)p2[i];
220         md2_block(c,cp);

222         for (i=0; i<16; i++)
223                 md[i]=(UCHAR)(p1[i]&0xff);
224         memset((char *)&c,0,sizeof(c));
225         return 1;
226         }
227 #endif /* ! codereview */
```

```
**********************************************************
    3925 Wed Aug 13 19:52:53 2014
new/usr/src/lib/openssl/libsunw_crypto/md2/md2_one.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md2/md2_one.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/md2.h>
```



```
/* This is a separate file so that #defines in cryptlib.h can
 * map my MD functions to different names */

unsigned char *MD2(const unsigned char *d, size_t n, unsigned char *md)
        {
        MD2_CTX c;
        static unsigned char m[MD2_DIGEST_LENGTH];

        if (md == NULL) md=m;
        if (!MD2_Init(&c))
                return NULL;
#ifndef CHARSET_EBCDIC
        MD2_Update(&c,d,n);
#else
        {
                char temp[1024];
                unsigned long chunk;

                while (n > 0)
                {
                        chunk = (n > sizeof(temp)) ? sizeof(temp) : n;
                        ebcdic2ascii(temp, d, chunk);
                        MD2_Update(&c,temp,chunk);
                        n -= chunk;
                        d += chunk;
                }
        }
#endif
        MD2_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c));  /* Security consideration */
        return(md);
        }
#endif /* ! codereview */
```

```
**********************************************************
    6434 Wed Aug 13 19:52:53 2014
new/usr/src/lib/openssl/libsunw_crypto/md4/md4_dgst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md4/md4_dgst.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/opensslv.h>
#include <openssl/crypto.h>
```



```
#include "md4_locl.h"

const char MD4_version[]="MD4" OPENSSL_VERSION_PTEXT;

/* Implemented from RFC1186 The MD4 Message-Digest Algorithm
 */

#define INIT_DATA_A (unsigned long)0x67452301L
#define INIT_DATA_B (unsigned long)0xefcdab89L
#define INIT_DATA_C (unsigned long)0x98badcfeL
#define INIT_DATA_D (unsigned long)0x10325476L

fips_md_init(MD4)
        {
        memset (c,0,sizeof(*c));
        c->A=INIT_DATA_A;
        c->B=INIT_DATA_B;
        c->C=INIT_DATA_C;
        c->D=INIT_DATA_D;
        return 1;
        }

#ifndef md4_block_data_order
#ifdef X
#undef X
#endif
void md4_block_data_order (MD4_CTX *c, const void *data_, size_t num)
        {
        const unsigned char *data=data_;
        register unsigned MD32_REG_T A,B,C,D,l;
#ifndef MD32_XARRAY
        /* See comment in crypto/sha/sha_locl.h for details. */
        unsigned MD32_REG_T     XX0, XX1, XX2, XX3, XX4, XX5, XX6, XX7,
                                XX8, XX9,XX10,XX11,XX12,XX13,XX14,XX15;
# define X(i)   XX##i
#else
        MD4_LONG XX[MD4_LBLOCK];
# define X(i)   XX[i]
#endif

        A=c->A;
        B=c->B;
        C=c->C;
        D=c->D;

        for (;num--;)
                {
        (void)HOST_c2l(data,l); X( 0)=l;
        (void)HOST_c2l(data,l); X( 1)=l;
        /* Round 0 */
        R0(A,B,C,D,X( 0), 3,0); (void)HOST_c2l(data,l); X( 2)=l;
        R0(D,A,B,C,X( 1), 7,0); (void)HOST_c2l(data,l); X( 3)=l;
        R0(C,D,A,B,X( 2),11,0); (void)HOST_c2l(data,l); X( 4)=l;
        R0(B,C,D,A,X( 3),19,0); (void)HOST_c2l(data,l); X( 5)=l;
        R0(A,B,C,D,X( 4), 3,0); (void)HOST_c2l(data,l); X( 6)=l;
        R0(D,A,B,C,X( 5), 7,0); (void)HOST_c2l(data,l); X( 7)=l;
        R0(C,D,A,B,X( 6),11,0); (void)HOST_c2l(data,l); X( 8)=l;
        R0(B,C,D,A,X( 7),19,0); (void)HOST_c2l(data,l); X( 9)=l;
        R0(A,B,C,D,X( 8), 3,0); (void)HOST_c2l(data,l); X(10)=l;
        R0(D,A,B,C,X( 9), 7,0); (void)HOST_c2l(data,l); X(11)=l;
        R0(C,D,A,B,X(10),11,0); (void)HOST_c2l(data,l); X(12)=l;
        R0(B,C,D,A,X(11),19,0); (void)HOST_c2l(data,l); X(13)=l;
        R0(A,B,C,D,X(12), 3,0); (void)HOST_c2l(data,l); X(14)=l;
        R0(D,A,B,C,X(13), 7,0); (void)HOST_c2l(data,l); X(15)=l;
        R0(C,D,A,B,X(14),11,0);
        R0(B,C,D,A,X(15),19,0);
```

```
128         /* Round 1 */
129         R1(A,B,C,D,X( 0), 3,0x5A827999L);
130         R1(D,A,B,C,X( 4), 5,0x5A827999L);
131         R1(C,D,A,B,X( 8), 9,0x5A827999L);
132         R1(B,C,D,A,X(12),13,0x5A827999L);
133         R1(A,B,C,D,X( 1), 3,0x5A827999L);
134         R1(D,A,B,C,X( 5), 5,0x5A827999L);
135         R1(C,D,A,B,X( 9), 9,0x5A827999L);
136         R1(B,C,D,A,X(13),13,0x5A827999L);
137         R1(A,B,C,D,X( 2), 3,0x5A827999L);
138         R1(D,A,B,C,X( 6), 5,0x5A827999L);
139         R1(C,D,A,B,X(10), 9,0x5A827999L);
140         R1(B,C,D,A,X(14),13,0x5A827999L);
141         R1(A,B,C,D,X( 3), 3,0x5A827999L);
142         R1(D,A,B,C,X( 7), 5,0x5A827999L);
143         R1(C,D,A,B,X(11), 9,0x5A827999L);
144         R1(B,C,D,A,X(15),13,0x5A827999L);
145         /* Round 2 */
146         R2(A,B,C,D,X( 0), 3,0x6ED9EBA1L);
147         R2(D,A,B,C,X( 8), 9,0x6ED9EBA1L);
148         R2(C,D,A,B,X( 4),11,0x6ED9EBA1L);
149         R2(B,C,D,A,X(12),15,0x6ED9EBA1L);
150         R2(A,B,C,D,X( 2), 3,0x6ED9EBA1L);
151         R2(D,A,B,C,X(10), 9,0x6ED9EBA1L);
152         R2(C,D,A,B,X( 6),11,0x6ED9EBA1L);
153         R2(B,C,D,A,X(14),15,0x6ED9EBA1L);
154         R2(A,B,C,D,X( 1), 3,0x6ED9EBA1L);
155         R2(D,A,B,C,X( 9), 9,0x6ED9EBA1L);
156         R2(C,D,A,B,X( 5),11,0x6ED9EBA1L);
157         R2(B,C,D,A,X(13),15,0x6ED9EBA1L);
158         R2(A,B,C,D,X( 3), 3,0x6ED9EBA1L);
159         R2(D,A,B,C,X(11), 9,0x6ED9EBA1L);
160         R2(C,D,A,B,X( 7),11,0x6ED9EBA1L);
161         R2(B,C,D,A,X(15),15,0x6ED9EBA1L);

163         A = c->A += A;
164         B = c->B += B;
165         C = c->C += C;
166         D = c->D += D;
167                 }
168         }
169 #endif
170 #endif /* ! codereview */
```

```
**********************************************************
    3901 Wed Aug 13 19:52:53 2014
new/usr/src/lib/openssl/libsunw_crypto/md4/md4_one.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/md4/md4_one.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <openssl/md4.h>
```



```
#include <openssl/crypto.h>

#ifdef CHARSET_EBCDIC
#include <openssl/ebcdic.h>
#endif

unsigned char *MD4(const unsigned char *d, size_t n, unsigned char *md)
        {
        MD4_CTX c;
        static unsigned char m[MD4_DIGEST_LENGTH];

        if (md == NULL) md=m;
        if (!MD4_Init(&c))
                return NULL;
#ifndef CHARSET_EBCDIC
        MD4_Update(&c,d,n);
#else
        {
                char temp[1024];
                unsigned long chunk;

                while (n > 0)
                {
                        chunk = (n > sizeof(temp)) ? sizeof(temp) : n;
                        ebcdic2ascii(temp, d, chunk);
                        MD4_Update(&c,temp,chunk);
                        n -= chunk;
                        d += chunk;
                }
        }
#endif
        MD4_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c)); /* security consideration */
        return(md);
        }
#endif /* ! codereview */
```

```
**********************************************************
    7073 Wed Aug 13 19:52:53 2014
new/usr/src/lib/openssl/libsunw_crypto/md5/md5_dgst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md5/md5_dgst.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "md5_locl.h"
#include <openssl/opensslv.h>
```



```
#include <openssl/crypto.h>

const char MD5_version[]="MD5" OPENSSL_VERSION_PTEXT;

/* Implemented from RFC1321 The MD5 Message-Digest Algorithm
 */

#define INIT_DATA_A (unsigned long)0x67452301L
#define INIT_DATA_B (unsigned long)0xefcdab89L
#define INIT_DATA_C (unsigned long)0x98badcfeL
#define INIT_DATA_D (unsigned long)0x10325476L

fips_md_init(MD5)
        {
        memset (c,0,sizeof(*c));
        c->A=INIT_DATA_A;
        c->B=INIT_DATA_B;
        c->C=INIT_DATA_C;
        c->D=INIT_DATA_D;
        return 1;
        }

#ifndef md5_block_data_order
#ifdef X
#undef X
#endif
void md5_block_data_order (MD5_CTX *c, const void *data_, size_t num)
        {
        const unsigned char *data=data_;
        register unsigned MD32_REG_T A,B,C,D,l;
#ifndef MD32_XARRAY
        /* See comment in crypto/sha/sha_locl.h for details. */
        unsigned MD32_REG_T     XX0, XX1, XX2, XX3, XX4, XX5, XX6, XX7,
                                XX8, XX9,XX10,XX11,XX12,XX13,XX14,XX15;
# define X(i)   XX##i
#else
        MD5_LONG XX[MD5_LBLOCK];
# define X(i)   XX[i]
#endif

        A=c->A;
        B=c->B;
        C=c->C;
        D=c->D;

        for (;num--;)
                {
        HOST_c2l(data,l); X( 0)=l;              HOST_c2l(data,l); X( 1)=l;
        /* Round 0 */
        R0(A,B,C,D,X( 0), 7,0xd76aa478L);       HOST_c2l(data,l); X( 2)=l;
        R0(D,A,B,C,X( 1),12,0xe8c7b756L);       HOST_c2l(data,l); X( 3)=l;
        R0(C,D,A,B,X( 2),17,0x242070dbL);       HOST_c2l(data,l); X( 4)=l;
        R0(B,C,D,A,X( 3),22,0xc1bdceeeL);       HOST_c2l(data,l); X( 5)=l;
        R0(A,B,C,D,X( 4), 7,0xf57c0fafL);       HOST_c2l(data,l); X( 6)=l;
        R0(D,A,B,C,X( 5),12,0x4787c62aL);       HOST_c2l(data,l); X( 7)=l;
        R0(C,D,A,B,X( 6),17,0xa8304613L);       HOST_c2l(data,l); X( 8)=l;
        R0(B,C,D,A,X( 7),22,0xfd469501L);       HOST_c2l(data,l); X( 9)=l;
        R0(A,B,C,D,X( 8), 7,0x698098d8L);       HOST_c2l(data,l); X(10)=l;
        R0(D,A,B,C,X( 9),12,0x8b44f7afL);       HOST_c2l(data,l); X(11)=l;
        R0(C,D,A,B,X(10),17,0xffff5bb1L);       HOST_c2l(data,l); X(12)=l;
        R0(B,C,D,A,X(11),22,0x895cd7beL);       HOST_c2l(data,l); X(13)=l;
        R0(A,B,C,D,X(12), 7,0x6b901122L);       HOST_c2l(data,l); X(14)=l;
        R0(D,A,B,C,X(13),12,0xfd987193L);       HOST_c2l(data,l); X(15)=l;
        R0(C,D,A,B,X(14),17,0xa679438eL);
        R0(B,C,D,A,X(15),22,0x49b40821L);
        /* Round 1 */
```

```
128         R1(A,B,C,D,X( 1), 5,0xf61e2562L);
129         R1(D,A,B,C,X( 6), 9,0xc040b340L);
130         R1(C,D,A,B,X(11),14,0x265e5a51L);
131         R1(B,C,D,A,X( 0),20,0xe9b6c7aaL);
132         R1(A,B,C,D,X( 5), 5,0xd62f105dL);
133         R1(D,A,B,C,X(10), 9,0x02441453L);
134         R1(C,D,A,B,X(15),14,0xd8a1e681L);
135         R1(B,C,D,A,X( 4),20,0xe7d3fbc8L);
136         R1(A,B,C,D,X( 9), 5,0x21e1cde6L);
137         R1(D,A,B,C,X(14), 9,0xc33707d6L);
138         R1(C,D,A,B,X( 3),14,0xf4d50d87L);
139         R1(B,C,D,A,X( 8),20,0x455a14edL);
140         R1(A,B,C,D,X(13), 5,0xa9e3e905L);
141         R1(D,A,B,C,X( 2), 9,0xfcefa3f8L);
142         R1(C,D,A,B,X( 7),14,0x676f02d9L);
143         R1(B,C,D,A,X(12),20,0x8d2a4c8aL);
144         /* Round 2 */
145         R2(A,B,C,D,X( 5), 4,0xfffa3942L);
146         R2(D,A,B,C,X( 8),11,0x8771f681L);
147         R2(C,D,A,B,X(11),16,0x6d9d6122L);
148         R2(B,C,D,A,X(14),23,0xfde5380cL);
149         R2(A,B,C,D,X( 1), 4,0xa4beea44L);
150         R2(D,A,B,C,X( 4),11,0x4bdecfa9L);
151         R2(C,D,A,B,X( 7),16,0xf6bb4b60L);
152         R2(B,C,D,A,X(10),23,0xbebfbc70L);
153         R2(A,B,C,D,X(13), 4,0x289b7ec6L);
154         R2(D,A,B,C,X( 0),11,0xeaa127faL);
155         R2(C,D,A,B,X( 3),16,0xd4ef3085L);
156         R2(B,C,D,A,X( 6),23,0x04881d05L);
157         R2(A,B,C,D,X( 9), 4,0xd9d4d039L);
158         R2(D,A,B,C,X(12),11,0xe6db99e5L);
159         R2(C,D,A,B,X(15),16,0x1fa27cf8L);
160         R2(B,C,D,A,X( 2),23,0xc4ac5665L);
161         /* Round 3 */
162         R3(A,B,C,D,X( 0), 6,0xf4292244L);
163         R3(D,A,B,C,X( 7),10,0x432aff97L);
164         R3(C,D,A,B,X(14),15,0xab9423a7L);
165         R3(B,C,D,A,X( 5),21,0xfc93a039L);
166         R3(A,B,C,D,X(12), 6,0x655b59c3L);
167         R3(D,A,B,C,X( 3),10,0x8f0ccc92L);
168         R3(C,D,A,B,X(10),15,0xffeff47dL);
169         R3(B,C,D,A,X( 1),21,0x85845dd1L);
170         R3(A,B,C,D,X( 8), 6,0x6fa87e4fL);
171         R3(D,A,B,C,X(15),10,0xfe2ce6e0L);
172         R3(C,D,A,B,X( 6),15,0xa3014314L);
173         R3(B,C,D,A,X(13),21,0x4e0811a1L);
174         R3(A,B,C,D,X( 4), 6,0xf7537e82L);
175         R3(D,A,B,C,X(11),10,0xbd3af235L);
176         R3(C,D,A,B,X( 2),15,0x2ad7d2bbL);
177         R3(B,C,D,A,X( 9),21,0xeb86d391L);

179         A = c->A += A;
180         B = c->B += B;
181         C = c->C += C;
182         D = c->D += D;
183               }
184         }
185 #endif
186 #endif /* ! codereview */
```

```
**********************************************************
    3901 Wed Aug 13 19:52:53 2014
new/usr/src/lib/openssl/libsunw_crypto/md5/md5_one.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/md5/md5_one.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <openssl/md5.h>
```



```
#include <openssl/crypto.h>

#ifdef CHARSET_EBCDIC
#include <openssl/ebcdic.h>
#endif

unsigned char *MD5(const unsigned char *d, size_t n, unsigned char *md)
        {
        MD5_CTX c;
        static unsigned char m[MD5_DIGEST_LENGTH];

        if (md == NULL) md=m;
        if (!MD5_Init(&c))
                return NULL;
#ifndef CHARSET_EBCDIC
        MD5_Update(&c,d,n);
#else
        {
                char temp[1024];
                unsigned long chunk;

                while (n > 0)
                {
                        chunk = (n > sizeof(temp)) ? sizeof(temp) : n;
                        ebcdic2ascii(temp, d, chunk);
                        MD5_Update(&c,temp,chunk);
                        n -= chunk;
                        d += chunk;
                }
        }
#endif
        MD5_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c)); /* security consideration */
        return(md);
        }
#endif /* ! codereview */
```

```
**********************************************************
   13462 Wed Aug 13 19:52:54 2014
new/usr/src/lib/openssl/libsunw_crypto/mem.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/mem.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <stdlib.h>
#include <openssl/crypto.h>
```



```
#include "cryptlib.h"


static int allow_customize = 1;      /* we provide flexible functions for */
static int allow_customize_debug = 1;/* exchanging memory-related functions at
                                      * run-time, but this must be done
                                      * before any blocks are actually
                                      * allocated; or we'll run into huge
                                      * problems when malloc/free pairs
                                      * don't match etc. */



/* the following pointers may be changed as long as 'allow_customize' is set */

static void *(*malloc_func)(size_t)         = malloc;
static void *default_malloc_ex(size_t num, const char *file, int line)
        { return malloc_func(num); }
static void *(*malloc_ex_func)(size_t, const char *file, int line)
        = default_malloc_ex;

static void *(*realloc_func)(void *, size_t)= realloc;
static void *default_realloc_ex(void *str, size_t num,
        const char *file, int line)
        { return realloc_func(str,num); }
static void *(*realloc_ex_func)(void *, size_t, const char *file, int line)
        = default_realloc_ex;

static void (*free_func)(void *)            = free;

static void *(*malloc_locked_func)(size_t)  = malloc;
static void *default_malloc_locked_ex(size_t num, const char *file, int line)
        { return malloc_locked_func(num); }
static void *(*malloc_locked_ex_func)(size_t, const char *file, int line)
        = default_malloc_locked_ex;

static void (*free_locked_func)(void *)     = free;



/* may be changed as long as 'allow_customize_debug' is set */
/* XXX use correct function pointer types */
#ifdef CRYPTO_MDEBUG
/* use default functions from mem_dbg.c */
static void (*malloc_debug_func)(void *,int,const char *,int,int)
        = CRYPTO_dbg_malloc;
static void (*realloc_debug_func)(void *,void *,int,const char *,int,int)
        = CRYPTO_dbg_realloc;
static void (*free_debug_func)(void *,int) = CRYPTO_dbg_free;
static void (*set_debug_options_func)(long) = CRYPTO_dbg_set_options;
static long (*get_debug_options_func)(void) = CRYPTO_dbg_get_options;
#else
/* applications can use CRYPTO_malloc_debug_init() to select above case
 * at run-time */
static void (*malloc_debug_func)(void *,int,const char *,int,int) = NULL;
static void (*realloc_debug_func)(void *,void *,int,const char *,int,int)
        = NULL;
static void (*free_debug_func)(void *,int) = NULL;
static void (*set_debug_options_func)(long) = NULL;
static long (*get_debug_options_func)(void) = NULL;
#endif

int CRYPTO_set_mem_functions(void *(*m)(size_t), void *(*r)(void *, size_t),
        void (*f)(void *))
        {
        /* Dummy call just to ensure OPENSSL_init() gets linked in */
```

```
128         OPENSSL_init();
129         if (!allow_customize)
130                 return 0;
131         if ((m == 0) || (r == 0) || (f == 0))
132                 return 0;
133         malloc_func=m; malloc_ex_func=default_malloc_ex;
134         realloc_func=r; realloc_ex_func=default_realloc_ex;
135         free_func=f;
136         malloc_locked_func=m; malloc_locked_ex_func=default_malloc_locked_ex;
137         free_locked_func=f;
138         return 1;
139         }

141 int CRYPTO_set_mem_ex_functions(
142         void *(*m)(size_t,const char *,int),
143         void *(*r)(void *, size_t,const char *,int),
144         void (*f)(void *))
145         {
146         if (!allow_customize)
147                 return 0;
148         if ((m == 0) || (r == 0) || (f == 0))
149                 return 0;
150         malloc_func=0; malloc_ex_func=m;
151         realloc_func=0; realloc_ex_func=r;
152         free_func=f;
153         malloc_locked_func=0; malloc_locked_ex_func=m;
154         free_locked_func=f;
155         return 1;
156         }

158 int CRYPTO_set_locked_mem_functions(void *(*m)(size_t), void (*f)(void *))
159         {
160         if (!allow_customize)
161                 return 0;
162         if ((m == NULL) || (f == NULL))
163                 return 0;
164         malloc_locked_func=m; malloc_locked_ex_func=default_malloc_locked_ex;
165         free_locked_func=f;
166         return 1;
167         }

169 int CRYPTO_set_locked_mem_ex_functions(
170         void *(*m)(size_t,const char *,int),
171         void (*f)(void *))
172         {
173         if (!allow_customize)
174                 return 0;
175         if ((m == NULL) || (f == NULL))
176                 return 0;
177         malloc_locked_func=0; malloc_locked_ex_func=m;
178         free_func=f;
179         return 1;
180         }

182 int CRYPTO_set_mem_debug_functions(void (*m)(void *,int,const char *,int,int),
183                                    void (*r)(void *,void *,int,const char *,int,
184                                    void (*f)(void *,int),
185                                    void (*so)(long),
186                                    long (*go)(void))
187         {
188         if (!allow_customize_debug)
189                 return 0;
190         OPENSSL_init();
191         malloc_debug_func=m;
192         realloc_debug_func=r;
193         free_debug_func=f;
```



```
194         set_debug_options_func=so;
195         get_debug_options_func=go;
196         return 1;
197         }


200 void CRYPTO_get_mem_functions(void *(**m)(size_t), void *(**r)(void *, size_t),
201         void (**f)(void *))
202         {
203         if (m != NULL) *m = (malloc_ex_func == default_malloc_ex) ?
204                              malloc_func : 0;
205         if (r != NULL) *r = (realloc_ex_func == default_realloc_ex) ?
206                              realloc_func : 0;
207         if (f != NULL) *f=free_func;
208         }

210 void CRYPTO_get_mem_ex_functions(
211         void *(**m)(size_t,const char *,int),
212         void *(**r)(void *, size_t,const char *,int),
213         void (**f)(void *))
214         {
215         if (m != NULL) *m = (malloc_ex_func != default_malloc_ex) ?
216                             malloc_ex_func : 0;
217         if (r != NULL) *r = (realloc_ex_func != default_realloc_ex) ?
218                             realloc_ex_func : 0;
219         if (f != NULL) *f=free_func;
220         }

222 void CRYPTO_get_locked_mem_functions(void *(**m)(size_t), void (**f)(void *))
223         {
224         if (m != NULL) *m = (malloc_locked_ex_func == default_malloc_locked_ex)
225                              malloc_locked_func : 0;
226         if (f != NULL) *f=free_locked_func;
227         }

229 void CRYPTO_get_locked_mem_ex_functions(
230         void *(**m)(size_t,const char *,int),
231         void (**f)(void *))
232         {
233         if (m != NULL) *m = (malloc_locked_ex_func != default_malloc_locked_ex)
234                             malloc_locked_ex_func : 0;
235         if (f != NULL) *f=free_locked_func;
236         }

238 void CRYPTO_get_mem_debug_functions(void (**m)(void *,int,const char *,int,int),
239                                     void (**r)(void *,void *,int,const char *,in
240                                     void (**f)(void *,int),
241                                     void (**so)(long),
242                                     long (**go)(void))
243         {
244         if (m != NULL) *m=malloc_debug_func;
245         if (r != NULL) *r=realloc_debug_func;
246         if (f != NULL) *f=free_debug_func;
247         if (so != NULL) *so=set_debug_options_func;
248         if (go != NULL) *go=get_debug_options_func;
249         }


252 void *CRYPTO_malloc_locked(int num, const char *file, int line)
253         {
254         void *ret = NULL;

256         if (num <= 0) return NULL;

258         allow_customize = 0;
259         if (malloc_debug_func != NULL)
```

```
260                 {
261                 allow_customize_debug = 0;
262                 malloc_debug_func(NULL, num, file, line, 0);
263                 }
264         ret = malloc_locked_ex_func(num,file,line);
265 #ifdef LEVITTE_DEBUG_MEM
266         fprintf(stderr, "LEVITTE_DEBUG_MEM:         > 0x%p (%d)\n", ret, num);
267 #endif
268         if (malloc_debug_func != NULL)
269                 malloc_debug_func(ret, num, file, line, 1);

271 #ifndef OPENSSL_CPUID_OBJ
272         /* Create a dependency on the value of 'cleanse_ctr' so our memory
273          * sanitisation function can't be optimised out. NB: We only do
274          * this for >2Kb so the overhead doesn't bother us. */
275         if(ret && (num > 2048))
276         {       extern unsigned char cleanse_ctr;
277                 ((unsigned char *)ret)[0] = cleanse_ctr;
278         }
279 #endif

281         return ret;
282         }

284 void CRYPTO_free_locked(void *str)
285         {
286         if (free_debug_func != NULL)
287                 free_debug_func(str, 0);
288 #ifdef LEVITTE_DEBUG_MEM
289         fprintf(stderr, "LEVITTE_DEBUG_MEM:         < 0x%p\n", str);
290 #endif
291         free_locked_func(str);
292         if (free_debug_func != NULL)
293                 free_debug_func(NULL, 1);
294         }

296 void *CRYPTO_malloc(int num, const char *file, int line)
297         {
298         void *ret = NULL;

300         if (num <= 0) return NULL;

302         allow_customize = 0;
303         if (malloc_debug_func != NULL)
304                 {
305                 allow_customize_debug = 0;
306                 malloc_debug_func(NULL, num, file, line, 0);
307                 }
308         ret = malloc_ex_func(num,file,line);
309 #ifdef LEVITTE_DEBUG_MEM
310         fprintf(stderr, "LEVITTE_DEBUG_MEM:         > 0x%p (%d)\n", ret, num);
311 #endif
312         if (malloc_debug_func != NULL)
313                 malloc_debug_func(ret, num, file, line, 1);

315 #ifndef OPENSSL_CPUID_OBJ
316         /* Create a dependency on the value of 'cleanse_ctr' so our memory
317          * sanitisation function can't be optimised out. NB: We only do
318          * this for >2Kb so the overhead doesn't bother us. */
319         if(ret && (num > 2048))
320         {       extern unsigned char cleanse_ctr;
321                 ((unsigned char *)ret)[0] = cleanse_ctr;
322         }
323 #endif

325         return ret;
```



```
326         }
327 char *CRYPTO_strdup(const char *str, const char *file, int line)
328         {
329         char *ret = CRYPTO_malloc(strlen(str)+1, file, line);

331         strcpy(ret, str);
332         return ret;
333         }

335 void *CRYPTO_realloc(void *str, int num, const char *file, int line)
336         {
337         void *ret = NULL;

339         if (str == NULL)
340                 return CRYPTO_malloc(num, file, line);

342         if (num <= 0) return NULL;

344         if (realloc_debug_func != NULL)
345                 realloc_debug_func(str, NULL, num, file, line, 0);
346         ret = realloc_ex_func(str,num,file,line);
347 #ifdef LEVITTE_DEBUG_MEM
348         fprintf(stderr, "LEVITTE_DEBUG_MEM:         | 0x%p -> 0x%p (%d)\n", str,
349 #endif
350         if (realloc_debug_func != NULL)
351                 realloc_debug_func(str, ret, num, file, line, 1);

353         return ret;
354         }

356 void *CRYPTO_realloc_clean(void *str, int old_len, int num, const char *file,
357                            int line)
358         {
359         void *ret = NULL;

361         if (str == NULL)
362                 return CRYPTO_malloc(num, file, line);

364         if (num <= 0) return NULL;

366         /* We don't support shrinking the buffer. Note the memcpy that copies
367          * |old_len| bytes to the new buffer, below. */
368         if (num < old_len) return NULL;

370         if (realloc_debug_func != NULL)
371                 realloc_debug_func(str, NULL, num, file, line, 0);
372         ret=malloc_ex_func(num,file,line);
373         if(ret)
374                 {
375                 memcpy(ret,str,old_len);
376                 OPENSSL_cleanse(str,old_len);
377                 free_func(str);
378                 }
379 #ifdef LEVITTE_DEBUG_MEM
380         fprintf(stderr,
381                 "LEVITTE_DEBUG_MEM:         | 0x%p -> 0x%p (%d)\n",
382                 str, ret, num);
383 #endif
384         if (realloc_debug_func != NULL)
385                 realloc_debug_func(str, ret, num, file, line, 1);

387         return ret;
388         }

390 void CRYPTO_free(void *str)
391         {
```

```
392         if (free_debug_func != NULL)
393                 free_debug_func(str, 0);
394 #ifdef LEVITTE_DEBUG_MEM
395         fprintf(stderr, "LEVITTE_DEBUG_MEM:         < 0x%p\n", str);
396 #endif
397         free_func(str);
398         if (free_debug_func != NULL)
399                 free_debug_func(NULL, 1);
400         }

402 void *CRYPTO_remalloc(void *a, int num, const char *file, int line)
403         {
404         if (a != NULL) OPENSSL_free(a);
405         a=(char *)OPENSSL_malloc(num);
406         return(a);
407         }

409 void CRYPTO_set_mem_debug_options(long bits)
410         {
411         if (set_debug_options_func != NULL)
412                 set_debug_options_func(bits);
413         }

415 long CRYPTO_get_mem_debug_options(void)
416         {
417         if (get_debug_options_func != NULL)
418                 return get_debug_options_func();
419         return 0;
420         }
421 #endif /* ! codereview */
```

```
**********************************************************
   23509 Wed Aug 13 19:52:54 2014
new/usr/src/lib/openssl/libsunw_crypto/mem_dbg.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/mem_dbg.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <cryptlib.h>
#include <openssl/crypto.h>
#include <openssl/buffer.h>
#include <openssl/bio.h>
#include <openssl/lhash.h>

static int mh_mode=CRYPTO_MEM_CHECK_OFF;
/* The state changes to CRYPTO_MEM_CHECK_ON | CRYPTO_MEM_CHECK_ENABLE
 * when the application asks for it (usually after library initialisation
 * for which no book-keeping is desired).
 *
 * State CRYPTO_MEM_CHECK_ON exists only temporarily when the library
 * thinks that certain allocations should not be checked (e.g. the data
```

```
 * structures used for memory checking).  It is not suitable as an initial
 * state: the library will unexpectedly enable memory checking when it
 * executes one of those sections that want to disable checking
 * temporarily.
 *
 * State CRYPTO_MEM_CHECK_ENABLE without ..._ON makes no sense whatsoever.
 */

static unsigned long order = 0; /* number of memory requests */

DECLARE_LHASH_OF(MEM);
static LHASH_OF(MEM) *mh=NULL; /* hash-table of memory requests
                                * (address as key); access requires
                                * MALLOC2 lock */


typedef struct app_mem_info_st
/* For application-defined information (static C-string 'info')
 * to be displayed in memory leak list.
 * Each thread has its own stack.  For applications, there is
 *   CRYPTO_push_info("...")     to push an entry,
 *   CRYPTO_pop_info()           to pop an entry,
 *   CRYPTO_remove_all_info()    to pop all entries.
 */
        {
        CRYPTO_THREADID threadid;
        const char *file;
        int line;
        const char *info;
        struct app_mem_info_st *next; /* tail of thread's stack */
        int references;
        } APP_INFO;

static void app_info_free(APP_INFO *);

DECLARE_LHASH_OF(APP_INFO);
static LHASH_OF(APP_INFO) *amih=NULL; /* hash-table with those
                                       * app_mem_info_st's that are at
                                       * the top of their thread's
                                       * stack (with 'thread' as key);
                                       * access requires MALLOC2
                                       * lock */

typedef struct mem_st
/* memory-block description */
        {
        void *addr;
        int num;
        const char *file;
        int line;
        CRYPTO_THREADID threadid;
        unsigned long order;
        time_t time;
        APP_INFO *app_info;
        } MEM;

static long options =             /* extra information to be recorded */
#if defined(CRYPTO_MDEBUG_TIME) || defined(CRYPTO_MDEBUG_ALL)
        V_CRYPTO_MDEBUG_TIME |
#endif
#if defined(CRYPTO_MDEBUG_THREAD) || defined(CRYPTO_MDEBUG_ALL)
        V_CRYPTO_MDEBUG_THREAD |
#endif
        0;
```



```
static unsigned int num_disable = 0; /* num_disable > 0
                                      *     iff
                                      * mh_mode == CRYPTO_MEM_CHECK_ON (w/o ..._
                                      */

/* Valid iff num_disable > 0.  CRYPTO_LOCK_MALLOC2 is locked exactly in this
 * case (by the thread named in disabling_thread).
 */
static CRYPTO_THREADID disabling_threadid;

static void app_info_free(APP_INFO *inf)
        {
        if (--(inf->references) <= 0)
                {
                if (inf->next != NULL)
                        {
                        app_info_free(inf->next);
                        }
                OPENSSL_free(inf);
                }
        }

int CRYPTO_mem_ctrl(int mode)
        {
        int ret=mh_mode;

        CRYPTO_w_lock(CRYPTO_LOCK_MALLOC);
        switch (mode)
                {
        /* for applications (not to be called while multiple threads
         * use the library): */
        case CRYPTO_MEM_CHECK_ON: /* aka MemCheck_start() */
                mh_mode = CRYPTO_MEM_CHECK_ON|CRYPTO_MEM_CHECK_ENABLE;
                num_disable = 0;
                break;
        case CRYPTO_MEM_CHECK_OFF: /* aka MemCheck_stop() */
                mh_mode = 0;
                num_disable = 0; /* should be true *before* MemCheck_stop is use
                                    or there'll be a lot of confusion */
                break;

        /* switch off temporarily (for library-internal use): */
        case CRYPTO_MEM_CHECK_DISABLE: /* aka MemCheck_off() */
                if (mh_mode & CRYPTO_MEM_CHECK_ON)
                        {
                        CRYPTO_THREADID cur;
                        CRYPTO_THREADID_current(&cur);
                        if (!num_disable || CRYPTO_THREADID_cmp(&disabling_threa
                                {
                                /* Long-time lock CRYPTO_LOCK_MALLOC2 must not b
                                 * we're holding CRYPTO_LOCK_MALLOC, or we'll de
                                 * somebody else holds CRYPTO_LOCK_MALLOC2 (and
                                 * it because we block entry to this function).
                                 * Give them a chance, first, and then claim the
                                 * appropriate order (long-time lock first).
                                 */
                                CRYPTO_w_unlock(CRYPTO_LOCK_MALLOC);
                                /* Note that after we have waited for CRYPTO_LOC
                                 * and CRYPTO_LOCK_MALLOC, we'll still be in the
                                 * "case" and "if" branch because MemCheck_start
                                 * MemCheck_stop may never be used while there a
                                 * OpenSSL threads. */
                                CRYPTO_w_lock(CRYPTO_LOCK_MALLOC2);
                                CRYPTO_w_lock(CRYPTO_LOCK_MALLOC);
                                mh_mode &= ~CRYPTO_MEM_CHECK_ENABLE;
                                CRYPTO_THREADID_cpy(&disabling_threadid, &cur);
```

```
260                             }
261                         num_disable++;
262                         }
263                 break;
264         case CRYPTO_MEM_CHECK_ENABLE: /* aka MemCheck_on() */
265                 if (mh_mode & CRYPTO_MEM_CHECK_ON)
266                         {
267                         if (num_disable) /* always true, or something is going w
268                                 {
269                                 num_disable--;
270                                 if (num_disable == 0)
271                                         {
272                                         mh_mode|=CRYPTO_MEM_CHECK_ENABLE;
273                                         CRYPTO_w_unlock(CRYPTO_LOCK_MALLOC2);
274                                         }
275                                 }
276                         }
277                 break;

279         default:
280                 break;
281                 }
282         CRYPTO_w_unlock(CRYPTO_LOCK_MALLOC);
283         return(ret);
284         }

286 int CRYPTO_is_mem_check_on(void)
287         {
288         int ret = 0;

290         if (mh_mode & CRYPTO_MEM_CHECK_ON)
291                 {
292                 CRYPTO_THREADID cur;
293                 CRYPTO_THREADID_current(&cur);
294                 CRYPTO_r_lock(CRYPTO_LOCK_MALLOC);

296                 ret = (mh_mode & CRYPTO_MEM_CHECK_ENABLE)
297                         || CRYPTO_THREADID_cmp(&disabling_threadid, &cur);

299                 CRYPTO_r_unlock(CRYPTO_LOCK_MALLOC);
300                 }
301         return(ret);
302         }


305 void CRYPTO_dbg_set_options(long bits)
306         {
307         options = bits;
308         }

310 long CRYPTO_dbg_get_options(void)
311         {
312         return options;
313         }

315 static int mem_cmp(const MEM *a, const MEM *b)
316         {
317 #ifdef _WIN64
318         const char *ap=(const char *)a->addr,
319                    *bp=(const char *)b->addr;
320         if (ap==bp)     return 0;
321         else if (ap>bp) return 1;
322         else            return -1;
323 #else
324         return (const char *)a->addr - (const char *)b->addr;
325 #endif
```



```
326         }
327 static IMPLEMENT_LHASH_COMP_FN(mem, MEM)

329 static unsigned long mem_hash(const MEM *a)
330         {
331         unsigned long ret;

333         ret=(unsigned long)a->addr;

335         ret=ret*17851+(ret>>14)*7+(ret>>4)*251;
336         return(ret);
337         }
338 static IMPLEMENT_LHASH_HASH_FN(mem, MEM)

340 /* static int app_info_cmp(APP_INFO *a, APP_INFO *b) */
341 static int app_info_cmp(const void *a_void, const void *b_void)
342         {
343         return CRYPTO_THREADID_cmp(&((const APP_INFO *)a_void)->threadid,
344                                    &((const APP_INFO *)b_void)->threadid);
345         }
346 static IMPLEMENT_LHASH_COMP_FN(app_info, APP_INFO)

348 static unsigned long app_info_hash(const APP_INFO *a)
349         {
350         unsigned long ret;

352         ret = CRYPTO_THREADID_hash(&a->threadid);
353         /* This is left in as a "who am I to question legacy?" measure */
354         ret=ret*17851+(ret>>14)*7+(ret>>4)*251;
355         return(ret);
356         }
357 static IMPLEMENT_LHASH_HASH_FN(app_info, APP_INFO)

359 static APP_INFO *pop_info(void)
360         {
361         APP_INFO tmp;
362         APP_INFO *ret = NULL;

364         if (amih != NULL)
365                 {
366                 CRYPTO_THREADID_current(&tmp.threadid);
367                 if ((ret=lh_APP_INFO_delete(amih,&tmp)) != NULL)
368                         {
369                         APP_INFO *next=ret->next;

371                         if (next != NULL)
372                                 {
373                                 next->references++;
374                                 (void)lh_APP_INFO_insert(amih,next);
375                                 }
376 #ifdef LEVITTE_DEBUG_MEM
377                         if (CRYPTO_THREADID_cmp(&ret->threadid, &tmp.threadid))
378                                 {
379                                 fprintf(stderr, "pop_info(): deleted info has ot
380                                         CRYPTO_THREADID_hash(&ret->threadid),
381                                         CRYPTO_THREADID_hash(&tmp.threadid));
382                                 abort();
383                                 }
384 #endif
385                         if (--(ret->references) <= 0)
386                                 {
387                                 ret->next = NULL;
388                                 if (next != NULL)
389                                         next->references--;
390                                 OPENSSL_free(ret);
391                                 }
```

```
392                         }
393                 }
394         return(ret);
395         }

397 int CRYPTO_push_info_(const char *info, const char *file, int line)
398         {
399         APP_INFO *ami, *amim;
400         int ret=0;

402         if (is_MemCheck_on())
403                 {
404                 MemCheck_off(); /* obtain MALLOC2 lock */

406                 if ((ami = (APP_INFO *)OPENSSL_malloc(sizeof(APP_INFO))) == NULL
407                         {
408                         ret=0;
409                         goto err;
410                         }
411                 if (amih == NULL)
412                         {
413                         if ((amih=lh_APP_INFO_new()) == NULL)
414                                 {
415                                 OPENSSL_free(ami);
416                                 ret=0;
417                                 goto err;
418                                 }
419                         }

421                 CRYPTO_THREADID_current(&ami->threadid);
422                 ami->file=file;
423                 ami->line=line;
424                 ami->info=info;
425                 ami->references=1;
426                 ami->next=NULL;

428                 if ((amim=lh_APP_INFO_insert(amih,ami)) != NULL)
429                         {
430 #ifdef LEVITTE_DEBUG_MEM
431                         if (CRYPTO_THREADID_cmp(&ami->threadid, &amim->threadid)
432                                 {
433                                 fprintf(stderr, "CRYPTO_push_info(): previous in
434                                         CRYPTO_THREADID_hash(&amim->threadid),
435                                         CRYPTO_THREADID_hash(&ami->threadid));
436                                 abort();
437                                 }
438 #endif
439                         ami->next=amim;
440                         }
441  err:
442                 MemCheck_on(); /* release MALLOC2 lock */
443                 }

445         return(ret);
446         }

448 int CRYPTO_pop_info(void)
449         {
450         int ret=0;

452         if (is_MemCheck_on()) /* _must_ be true, or something went severely wron
453                 {
454                 MemCheck_off(); /* obtain MALLOC2 lock */

456                 ret=(pop_info() != NULL);
```



```
458                 MemCheck_on(); /* release MALLOC2 lock */
459                 }
460         return(ret);
461         }

463 int CRYPTO_remove_all_info(void)
464         {
465         int ret=0;

467         if (is_MemCheck_on()) /* _must_ be true */
468                 {
469                 MemCheck_off(); /* obtain MALLOC2 lock */

471                 while(pop_info() != NULL)
472                         ret++;

474                 MemCheck_on(); /* release MALLOC2 lock */
475                 }
476         return(ret);
477         }


480 static unsigned long break_order_num=0;
481 void CRYPTO_dbg_malloc(void *addr, int num, const char *file, int line,
482         int before_p)
483         {
484         MEM *m,*mm;
485         APP_INFO tmp,*amim;

487         switch(before_p & 127)
488                 {
489         case 0:
490                 break;
491         case 1:
492                 if (addr == NULL)
493                         break;

495                 if (is_MemCheck_on())
496                         {
497                         MemCheck_off(); /* make sure we hold MALLOC2 lock */
498                         if ((m=(MEM *)OPENSSL_malloc(sizeof(MEM))) == NULL)
499                                 {
500                                 OPENSSL_free(addr);
501                                 MemCheck_on(); /* release MALLOC2 lock
502                                                 * if num_disabled drops to 0 */
503                                 return;
504                                 }
505                         if (mh == NULL)
506                                 {
507                                 if ((mh=lh_MEM_new()) == NULL)
508                                         {
509                                         OPENSSL_free(addr);
510                                         OPENSSL_free(m);
511                                         addr=NULL;
512                                         goto err;
513                                         }
514                                 }

516                         m->addr=addr;
517                         m->file=file;
518                         m->line=line;
519                         m->num=num;
520                         if (options & V_CRYPTO_MDEBUG_THREAD)
521                                 CRYPTO_THREADID_current(&m->threadid);
522                         else
523                                 memset(&m->threadid, 0, sizeof(m->threadid));
```

```
525                         if (order == break_order_num)
526                                 {
527                                 /* BREAK HERE */
528                                 m->order=order;
529                                 }
530                         m->order=order++;
531 #ifdef LEVITTE_DEBUG_MEM
532                         fprintf(stderr, "LEVITTE_DEBUG_MEM: [%5ld] %c 0x%p (%d)\
533                                 m->order,
534                                 (before_p & 128) ? '*' : '+',
535                                 m->addr, m->num);
536 #endif
537                         if (options & V_CRYPTO_MDEBUG_TIME)
538                                 m->time=time(NULL);
539                         else
540                                 m->time=0;

542                         CRYPTO_THREADID_current(&tmp.threadid);
543                         m->app_info=NULL;
544                         if (amih != NULL
545                             && (amim=lh_APP_INFO_retrieve(amih,&tmp)) != NULL)
546                                 {
547                                 m->app_info = amim;
548                                 amim->references++;
549                                 }

551                         if ((mm=lh_MEM_insert(mh, m)) != NULL)
552                                 {
553                                 /* Not good, but don't sweat it */
554                                 if (mm->app_info != NULL)
555                                         {
556                                         mm->app_info->references--;
557                                         }
558                                 OPENSSL_free(mm);
559                                 }
560                 err:
561                         MemCheck_on(); /* release MALLOC2 lock
562                                         * if num_disabled drops to 0 */
563                         }
564                 break;
565                 }
566         return;
567         }

569 void CRYPTO_dbg_free(void *addr, int before_p)
570         {
571         MEM m,*mp;

573         switch(before_p)
574                 {
575         case 0:
576                 if (addr == NULL)
577                         break;

579                 if (is_MemCheck_on() && (mh != NULL))
580                         {
581                         MemCheck_off(); /* make sure we hold MALLOC2 lock */

583                         m.addr=addr;
584                         mp=lh_MEM_delete(mh,&m);
585                         if (mp != NULL)
586                                 {
587 #ifdef LEVITTE_DEBUG_MEM
588                         fprintf(stderr, "LEVITTE_DEBUG_MEM: [%5ld] - 0x%p (%d)\n
589                                 mp->order, mp->addr, mp->num);
```



```
590 #endif
591                                 if (mp->app_info != NULL)
592                                         app_info_free(mp->app_info);
593                                 OPENSSL_free(mp);
594                                 }

596                         MemCheck_on(); /* release MALLOC2 lock
597                                         * if num_disabled drops to 0 */
598                         }
599                 break;
600         case 1:
601                 break;
602                 }
603         }

605 void CRYPTO_dbg_realloc(void *addr1, void *addr2, int num,
606         const char *file, int line, int before_p)
607         {
608         MEM m,*mp;

610 #ifdef LEVITTE_DEBUG_MEM
611         fprintf(stderr, "LEVITTE_DEBUG_MEM: --> CRYPTO_dbg_malloc(addr1 = %p, ad
612                 addr1, addr2, num, file, line, before_p);
613 #endif

615         switch(before_p)
616                 {
617         case 0:
618                 break;
619         case 1:
620                 if (addr2 == NULL)
621                         break;

623                 if (addr1 == NULL)
624                         {
625                         CRYPTO_dbg_malloc(addr2, num, file, line, 128 | before_p
626                         break;
627                         }

629                 if (is_MemCheck_on())
630                         {
631                         MemCheck_off(); /* make sure we hold MALLOC2 lock */

633                         m.addr=addr1;
634                         mp=lh_MEM_delete(mh,&m);
635                         if (mp != NULL)
636                                 {
637 #ifdef LEVITTE_DEBUG_MEM
638                                 fprintf(stderr, "LEVITTE_DEBUG_MEM: [%5ld] * 0x%
639                                         mp->order,
640                                         mp->addr, mp->num,
641                                         addr2, num);
642 #endif
643                                 mp->addr=addr2;
644                                 mp->num=num;
645                                 (void)lh_MEM_insert(mh,mp);
646                                 }

648                         MemCheck_on(); /* release MALLOC2 lock
649                                         * if num_disabled drops to 0 */
650                         }
651                 break;
652                 }
653         return;
654         }
```

```
typedef struct mem_leak_st
        {
        BIO *bio;
        int chunks;
        long bytes;
        } MEM_LEAK;

static void print_leak_doall_arg(const MEM *m, MEM_LEAK *l)
        {
        char buf[1024];
        char *bufp = buf;
        APP_INFO *amip;
        int ami_cnt;
        struct tm *lcl = NULL;
        CRYPTO_THREADID ti;

#define BUF_REMAIN (sizeof buf - (size_t)(bufp - buf))

        if(m->addr == (char *)l->bio)
            return;

        if (options & V_CRYPTO_MDEBUG_TIME)
                {
                lcl = localtime(&m->time);

                BIO_snprintf(bufp, BUF_REMAIN, "[%02d:%02d:%02d] ",
                        lcl->tm_hour,lcl->tm_min,lcl->tm_sec);
                bufp += strlen(bufp);
                }

        BIO_snprintf(bufp, BUF_REMAIN, "%5lu file=%s, line=%d, ",
                m->order,m->file,m->line);
        bufp += strlen(bufp);

        if (options & V_CRYPTO_MDEBUG_THREAD)
                {
                BIO_snprintf(bufp, BUF_REMAIN, "thread=%lu, ",
                        CRYPTO_THREADID_hash(&m->threadid));
                bufp += strlen(bufp);
                }

        BIO_snprintf(bufp, BUF_REMAIN, "number=%d, address=%08lX\n",
                m->num,(unsigned long)m->addr);
        bufp += strlen(bufp);

        BIO_puts(l->bio,buf);

        l->chunks++;
        l->bytes+=m->num;

        amip=m->app_info;
        ami_cnt=0;
        if (!amip)
                return;
        CRYPTO_THREADID_cpy(&ti, &amip->threadid);

        do
                {
                int buf_len;
                int info_len;

                ami_cnt++;
                memset(buf,'>',ami_cnt);
                BIO_snprintf(buf + ami_cnt, sizeof buf - ami_cnt,
                        " thread=%lu, file=%s, line=%d, info=\"",
```



```
                        CRYPTO_THREADID_hash(&amip->threadid), amip->file,
                        amip->line);
                buf_len=strlen(buf);
                info_len=strlen(amip->info);
                if (128 - buf_len - 3 < info_len)
                        {
                        memcpy(buf + buf_len, amip->info, 128 - buf_len - 3);
                        buf_len = 128 - 3;
                        }
                else
                        {
                        BUF_strlcpy(buf + buf_len, amip->info,
                                        sizeof buf - buf_len);
                        buf_len = strlen(buf);
                        }
                BIO_snprintf(buf + buf_len, sizeof buf - buf_len, "\"\n");

                BIO_puts(l->bio,buf);

                amip = amip->next;
                }
        while(amip && !CRYPTO_THREADID_cmp(&amip->threadid, &ti));

#ifdef LEVITTE_DEBUG_MEM
        if (amip)
                {
                fprintf(stderr, "Thread switch detected in backtrace!!!!\n");
                abort();
                }
#endif
        }

static IMPLEMENT_LHASH_DOALL_ARG_FN(print_leak, const MEM, MEM_LEAK)

void CRYPTO_mem_leaks(BIO *b)
        {
        MEM_LEAK ml;

        if (mh == NULL && amih == NULL)
                return;

        MemCheck_off(); /* obtain MALLOC2 lock */

        ml.bio=b;
        ml.bytes=0;
        ml.chunks=0;
        if (mh != NULL)
                lh_MEM_doall_arg(mh, LHASH_DOALL_ARG_FN(print_leak), MEM_LEAK,
                                 &ml);
        if (ml.chunks != 0)
                {
                BIO_printf(b,"%ld bytes leaked in %d chunks\n",
                                ml.bytes,ml.chunks);
#ifdef CRYPTO_MDEBUG_ABORT
                abort();
#endif
                }
        else
                {
                /* Make sure that, if we found no leaks, memory-leak debugging i
                 * does not introduce memory leaks (which might irritate
                 * external debugging tools).
                 * (When someone enables leak checking, but does not call
                 * this function, we declare it to be their fault.)
                 *
                 * XXX    This should be in CRYPTO_mem_leaks_cb,
```

```
788                  * and CRYPTO_mem_leaks should be implemented by
789                  * using CRYPTO_mem_leaks_cb.
790                  * (Also there should be a variant of lh_doall_arg
791                  * that takes a function pointer instead of a void *;
792                  * this would obviate the ugly and illegal
793                  * void_fn_to_char kludge in CRYPTO_mem_leaks_cb.
794                  * Otherwise the code police will come and get us.)
795                  */
796                 int old_mh_mode;

798                 CRYPTO_w_lock(CRYPTO_LOCK_MALLOC);

800                 /* avoid deadlock when lh_free() uses CRYPTO_dbg_free(),
801                  * which uses CRYPTO_is_mem_check_on */
802                 old_mh_mode = mh_mode;
803                 mh_mode = CRYPTO_MEM_CHECK_OFF;

805                 if (mh != NULL)
806                         {
807                         lh_MEM_free(mh);
808                         mh = NULL;
809                         }
810                 if (amih != NULL)
811                         {
812                         if (lh_APP_INFO_num_items(amih) == 0)
813                                 {
814                                 lh_APP_INFO_free(amih);
815                                 amih = NULL;
816                                 }
817                         }

819                 mh_mode = old_mh_mode;
820                 CRYPTO_w_unlock(CRYPTO_LOCK_MALLOC);
821                 }
822         MemCheck_on(); /* release MALLOC2 lock */
823         }

825 #ifndef OPENSSL_NO_FP_API
826 void CRYPTO_mem_leaks_fp(FILE *fp)
827         {
828         BIO *b;

830         if (mh == NULL) return;
831         /* Need to turn off memory checking when allocated BIOs ... especially
832          * as we're creating them at a time when we're trying to check we've not
833          * left anything un-free()'d!! */
834         MemCheck_off();
835         b = BIO_new(BIO_s_file());
836         MemCheck_on();
837         if(!b) return;
838         BIO_set_fp(b,fp,BIO_NOCLOSE);
839         CRYPTO_mem_leaks(b);
840         BIO_free(b);
841         }
842 #endif



846 /* FIXME: We really don't allow much to the callback.  For example, it has
847    no chance of reaching the info stack for the item it processes.  Should
848    it really be this way?  -- Richard Levitte */
849 /* NB: The prototypes have been typedef'd to CRYPTO_MEM_LEAK_CB inside crypto.h
850  * If this code is restructured, remove the callback type if it is no longer
851  * needed. -- Geoff Thorpe */

853 /* Can't pass CRYPTO_MEM_LEAK_CB directly to lh_MEM_doall_arg because it
```



```
854  * is a function pointer and conversion to void * is prohibited. Instead
855  * pass its address
856  */

858 typedef CRYPTO_MEM_LEAK_CB *PCRYPTO_MEM_LEAK_CB;

860 static void cb_leak_doall_arg(const MEM *m, PCRYPTO_MEM_LEAK_CB *cb)
861         {
862         (*cb)(m->order,m->file,m->line,m->num,m->addr);
863         }

865 static IMPLEMENT_LHASH_DOALL_ARG_FN(cb_leak, const MEM, PCRYPTO_MEM_LEAK_CB)

867 void CRYPTO_mem_leaks_cb(CRYPTO_MEM_LEAK_CB *cb)
868         {
869         if (mh == NULL) return;
870         CRYPTO_w_lock(CRYPTO_LOCK_MALLOC2);
871         lh_MEM_doall_arg(mh, LHASH_DOALL_ARG_FN(cb_leak), PCRYPTO_MEM_LEAK_CB,
872                          &cb);
873         CRYPTO_w_unlock(CRYPTO_LOCK_MALLOC2);
874         }
875 #endif /* ! codereview */
```

```
**********************************************************
    5590 Wed Aug 13 19:52:54 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/cbc128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>
```



```
#ifndef STRICT_ALIGNMENT
#  define STRICT_ALIGNMENT 0
#endif

void CRYPTO_cbc128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block)
{
        size_t n;
        const unsigned char *iv = ivec;

        assert(in && out && key && ivec);

#if !defined(OPENSSL_SMALL_FOOTPRINT)
        if (STRICT_ALIGNMENT &&
            ((size_t)in|(size_t)out|(size_t)ivec)%sizeof(size_t) != 0) {
                while (len>=16) {
                        for(n=0; n<16; ++n)
                                out[n] = in[n] ^ iv[n];
                        (*block)(out, out, key);
                        iv = out;
                        len -= 16;
                        in  += 16;
                        out += 16;
                }
        } else {
                while (len>=16) {
                        for(n=0; n<16; n+=sizeof(size_t))
                                *(size_t*)(out+n) =
                                *(size_t*)(in+n) ^ *(size_t*)(iv+n);
                        (*block)(out, out, key);
                        iv = out;
                        len -= 16;
                        in  += 16;
                        out += 16;
                }
        }
#endif
        while (len) {
                for(n=0; n<16 && n<len; ++n)
                        out[n] = in[n] ^ iv[n];
                for(; n<16; ++n)
                        out[n] = iv[n];
                (*block)(out, out, key);
                iv = out;
                if (len<=16) break;
                len -= 16;
                in  += 16;
                out += 16;
        }
        memcpy(ivec,iv,16);
}

void CRYPTO_cbc128_decrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block)
{
        size_t n;
        union { size_t t[16/sizeof(size_t)]; unsigned char c[16]; } tmp;

        assert(in && out && key && ivec);

#if !defined(OPENSSL_SMALL_FOOTPRINT)
        if (in != out) {
                const unsigned char *iv = ivec;
```

```
128                 if (STRICT_ALIGNMENT &&
129                     ((size_t)in|(size_t)out|(size_t)ivec)%sizeof(size_t) != 0) {
130                         while (len>=16) {
131                                 (*block)(in, out, key);
132                                 for(n=0; n<16; ++n)
133                                         out[n] ^= iv[n];
134                                 iv = in;
135                                 len -= 16;
136                                 in  += 16;
137                                 out += 16;
138                         }
139                 }
140                 else  if (16%sizeof(size_t) == 0) { /* always true */
141                         while (len>=16) {
142                                 size_t *out_t=(size_t *)out, *iv_t=(size_t *)iv;

144                                 (*block)(in, out, key);
145                                 for(n=0; n<16/sizeof(size_t); n++)
146                                         out_t[n] ^= iv_t[n];
147                                 iv = in;
148                                 len -= 16;
149                                 in  += 16;
150                                 out += 16;
151                         }
152                 }
153                 memcpy(ivec,iv,16);
154         } else {
155                 if (STRICT_ALIGNMENT &&
156                     ((size_t)in|(size_t)out|(size_t)ivec)%sizeof(size_t) != 0) {
157                         unsigned char c;
158                         while (len>=16) {
159                                 (*block)(in, tmp.c, key);
160                                 for(n=0; n<16; ++n) {
161                                         c = in[n];
162                                         out[n] = tmp.c[n] ^ ivec[n];
163                                         ivec[n] = c;
164                                 }
165                                 len -= 16;
166                                 in  += 16;
167                                 out += 16;
168                         }
169                 }
170                 else if (16%sizeof(size_t) == 0) { /* always true */
171                         while (len>=16) {
172                                 size_t c, *out_t=(size_t *)out, *ivec_t=(size_t
173                                 const size_t *in_t=(const size_t *)in;

175                                 (*block)(in, tmp.c, key);
176                                 for(n=0; n<16/sizeof(size_t); n++) {
177                                         c = in_t[n];
178                                         out_t[n] = tmp.t[n] ^ ivec_t[n];
179                                         ivec_t[n] = c;
180                                 }
181                                 len -= 16;
182                                 in  += 16;
183                                 out += 16;
184                         }
185                 }
186         }
187 #endif
188         while (len) {
189                 unsigned char c;
190                 (*block)(in, tmp.c, key);
191                 for(n=0; n<16 && n<len; ++n) {
192                         c = in[n];
193                         out[n] = tmp.c[n] ^ ivec[n];
```



```
194                         ivec[n] = c;
195                 }
196                 if (len<=16) {
197                         for (; n<16; ++n)
198                                 ivec[n] = in[n];
199                         break;
200                 }
201                 len -= 16;
202                 in  += 16;
203                 out += 16;
204         }
205 }
206 #endif /* ! codereview */
```

```
**********************************************************
   11635 Wed Aug 13 19:52:54 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/ccm128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>

/* First you setup M and L parameters and pass the key schedule.
```



```
 * This is called once per session setup... */
void CRYPTO_ccm128_init(CCM128_CONTEXT *ctx,
        unsigned int M,unsigned int L,void *key,block128_f block)
{
        memset(ctx->nonce.c,0,sizeof(ctx->nonce.c));
        ctx->nonce.c[0] = ((u8)(L-1)&7) | (u8)(((M-2)/2)&7)<<3;
        ctx->blocks = 0;
        ctx->block = block;
        ctx->key = key;
}

/* !!! Following interfaces are to be called *once* per packet !!! */

/* Then you setup per-message nonce and pass the length of the message */
int CRYPTO_ccm128_setiv(CCM128_CONTEXT *ctx,
        const unsigned char *nonce,size_t nlen,size_t mlen)
{
        unsigned int L = ctx->nonce.c[0]&7;     /* the L parameter */

        if (nlen<(14-L)) return -1;             /* nonce is too short */

        if (sizeof(mlen)==8 && L>=3) {
                ctx->nonce.c[8]  = (u8)(mlen>>(56%(sizeof(mlen)*8)));
                ctx->nonce.c[9]  = (u8)(mlen>>(48%(sizeof(mlen)*8)));
                ctx->nonce.c[10] = (u8)(mlen>>(40%(sizeof(mlen)*8)));
                ctx->nonce.c[11] = (u8)(mlen>>(32%(sizeof(mlen)*8)));
        }
        else
                ctx->nonce.u[1] = 0;

        ctx->nonce.c[12] = (u8)(mlen>>24);
        ctx->nonce.c[13] = (u8)(mlen>>16);
        ctx->nonce.c[14] = (u8)(mlen>>8);
        ctx->nonce.c[15] = (u8)mlen;

        ctx->nonce.c[0] &= ~0x40;       /* clear Adata flag */
        memcpy(&ctx->nonce.c[1],nonce,14-L);

        return 0;
}

/* Then you pass additional authentication data, this is optional */
void CRYPTO_ccm128_aad(CCM128_CONTEXT *ctx,
        const unsigned char *aad,size_t alen)
{       unsigned int i;
        block128_f block = ctx->block;

        if (alen==0) return;

        ctx->nonce.c[0] |= 0x40;        /* set Adata flag */
        (*block)(ctx->nonce.c,ctx->cmac.c,ctx->key),
        ctx->blocks++;

        if (alen<(0x10000-0x100)) {
                ctx->cmac.c[0] ^= (u8)(alen>>8);
                ctx->cmac.c[1] ^= (u8)alen;
                i=2;
        }
        else if (sizeof(alen)==8 && alen>=(size_t)1<<(32%(sizeof(alen)*8))) {
                ctx->cmac.c[0] ^= 0xFF;
                ctx->cmac.c[1] ^= 0xFF;
                ctx->cmac.c[2] ^= (u8)(alen>>(56%(sizeof(alen)*8)));
                ctx->cmac.c[3] ^= (u8)(alen>>(48%(sizeof(alen)*8)));
                ctx->cmac.c[4] ^= (u8)(alen>>(40%(sizeof(alen)*8)));
                ctx->cmac.c[5] ^= (u8)(alen>>(32%(sizeof(alen)*8)));
                ctx->cmac.c[6] ^= (u8)(alen>>24);
```

```
128                 ctx->cmac.c[7] ^= (u8)(alen>>16);
129                 ctx->cmac.c[8] ^= (u8)(alen>>8);
130                 ctx->cmac.c[9] ^= (u8)alen;
131                 i=10;
132         }
133         else {
134                 ctx->cmac.c[0] ^= 0xFF;
135                 ctx->cmac.c[1] ^= 0xFE;
136                 ctx->cmac.c[2] ^= (u8)(alen>>24);
137                 ctx->cmac.c[3] ^= (u8)(alen>>16);
138                 ctx->cmac.c[4] ^= (u8)(alen>>8);
139                 ctx->cmac.c[5] ^= (u8)alen;
140                 i=6;
141         }
142
143         do {
144                 for(;i<16 && alen;++i,++aad,--alen)
145                         ctx->cmac.c[i] ^= *aad;
146                 (*block)(ctx->cmac.c,ctx->cmac.c,ctx->key),
147                 ctx->blocks++;
148                 i=0;
149         } while (alen);
150 }
151
152 /* Finally you encrypt or decrypt the message */
153
154 /* counter part of nonce may not be larger than L*8 bits,
155  * L is not larger than 8, therefore 64-bit counter... */
156 static void ctr64_inc(unsigned char *counter) {
157         unsigned int n=8;
158         u8  c;
159
160         counter += 8;
161         do {
162                 --n;
163                 c = counter[n];
164                 ++c;
165                 counter[n] = c;
166                 if (c) return;
167         } while (n);
168 }
169
170 int CRYPTO_ccm128_encrypt(CCM128_CONTEXT *ctx,
171         const unsigned char *inp, unsigned char *out,
172         size_t len)
173 {
174         size_t          n;
175         unsigned int    i,L;
176         unsigned char   flags0  = ctx->nonce.c[0];
177         block128_f      block   = ctx->block;
178         void *          key     = ctx->key;
179         union { u64 u[2]; u8 c[16]; } scratch;
180
181         if (!(flags0&0x40))
182                 (*block)(ctx->nonce.c,ctx->cmac.c,key),
183                 ctx->blocks++;
184
185         ctx->nonce.c[0] = L = flags0&7;
186         for (n=0,i=15-L;i<15;++i) {
187                 n |= ctx->nonce.c[i];
188                 ctx->nonce.c[i]=0;
189                 n <<= 8;
190         }
191         n |= ctx->nonce.c[15];  /* reconstructed length */
192         ctx->nonce.c[15]=1;
```



```
194         if (n!=len) return -1;  /* length mismatch */
195
196         ctx->blocks += ((len+15)>>3)|1;
197         if (ctx->blocks > (U64(1)<<61)) return -2; /* too much data */
198
199         while (len>=16) {
200 #if defined(STRICT_ALIGNMENT)
201                 union { u64 u[2]; u8 c[16]; } temp;
202
203                 memcpy (temp.c,inp,16);
204                 ctx->cmac.u[0] ^= temp.u[0];
205                 ctx->cmac.u[1] ^= temp.u[1];
206 #else
207                 ctx->cmac.u[0] ^= ((u64*)inp)[0];
208                 ctx->cmac.u[1] ^= ((u64*)inp)[1];
209 #endif
210                 (*block)(ctx->cmac.c,ctx->cmac.c,key);
211                 (*block)(ctx->nonce.c,scratch.c,key);
212                 ctr64_inc(ctx->nonce.c);
213 #if defined(STRICT_ALIGNMENT)
214                 temp.u[0] ^= scratch.u[0];
215                 temp.u[1] ^= scratch.u[1];
216                 memcpy(out,temp.c,16);
217 #else
218                 ((u64*)out)[0] = scratch.u[0]^((u64*)inp)[0];
219                 ((u64*)out)[1] = scratch.u[1]^((u64*)inp)[1];
220 #endif
221                 inp += 16;
222                 out += 16;
223                 len -= 16;
224         }
225
226         if (len) {
227                 for (i=0; i<len; ++i) ctx->cmac.c[i] ^= inp[i];
228                 (*block)(ctx->cmac.c,ctx->cmac.c,key);
229                 (*block)(ctx->nonce.c,scratch.c,key);
230                 for (i=0; i<len; ++i) out[i] = scratch.c[i]^inp[i];
231         }
232
233         for (i=15-L;i<16;++i)
234                 ctx->nonce.c[i]=0;
235
236         (*block)(ctx->nonce.c,scratch.c,key);
237         ctx->cmac.u[0] ^= scratch.u[0];
238         ctx->cmac.u[1] ^= scratch.u[1];
239
240         ctx->nonce.c[0] = flags0;
241
242         return 0;
243 }
244
245 int CRYPTO_ccm128_decrypt(CCM128_CONTEXT *ctx,
246         const unsigned char *inp, unsigned char *out,
247         size_t len)
248 {
249         size_t          n;
250         unsigned int    i,L;
251         unsigned char   flags0  = ctx->nonce.c[0];
252         block128_f      block   = ctx->block;
253         void *          key     = ctx->key;
254         union { u64 u[2]; u8 c[16]; } scratch;
255
256         if (!(flags0&0x40))
257                 (*block)(ctx->nonce.c,ctx->cmac.c,key);
258
259         ctx->nonce.c[0] = L = flags0&7;
```

```
260         for (n=0,i=15-L;i<15;++i) {
261                 n |= ctx->nonce.c[i];
262                 ctx->nonce.c[i]=0;
263                 n <<= 8;
264         }
265         n |= ctx->nonce.c[15];  /* reconstructed length */
266         ctx->nonce.c[15]=1;

268         if (n!=len) return -1;

270         while (len>=16) {
271 #if defined(STRICT_ALIGNMENT)
272                 union { u64 u[2]; u8 c[16]; } temp;
273 #endif
274                 (*block)(ctx->nonce.c,scratch.c,key);
275                 ctr64_inc(ctx->nonce.c);
276 #if defined(STRICT_ALIGNMENT)
277                 memcpy (temp.c,inp,16);
278                 ctx->cmac.u[0] ^= (scratch.u[0] ^= temp.u[0]);
279                 ctx->cmac.u[1] ^= (scratch.u[1] ^= temp.u[1]);
280                 memcpy (out,scratch.c,16);
281 #else
282                 ctx->cmac.u[0] ^= (((u64*)out)[0] = scratch.u[0]^((u64*)inp)[0])
283                 ctx->cmac.u[1] ^= (((u64*)out)[1] = scratch.u[1]^((u64*)inp)[1])
284 #endif
285                 (*block)(ctx->cmac.c,ctx->cmac.c,key);

287                 inp += 16;
288                 out += 16;
289                 len -= 16;
290         }

292         if (len) {
293                 (*block)(ctx->nonce.c,scratch.c,key);
294                 for (i=0; i<len; ++i)
295                         ctx->cmac.c[i] ^= (out[i] = scratch.c[i]^inp[i]);
296                 (*block)(ctx->cmac.c,ctx->cmac.c,key);
297         }

299         for (i=15-L;i<16;++i)
300                 ctx->nonce.c[i]=0;

302         (*block)(ctx->nonce.c,scratch.c,key);
303         ctx->cmac.u[0] ^= scratch.u[0];
304         ctx->cmac.u[1] ^= scratch.u[1];

306         ctx->nonce.c[0] = flags0;

308         return 0;
309 }

311 static void ctr64_add (unsigned char *counter,size_t inc)
312 {       size_t n=8, val=0;

314         counter += 8;
315         do {
316                 --n;
317                 val += counter[n] + (inc&0xff);
318                 counter[n] = (unsigned char)val;
319                 val >>= 8;      /* carry bit */
320                 inc >>= 8;
321         } while(n && (inc || val));
322 }

324 int CRYPTO_ccm128_encrypt_ccm64(CCM128_CONTEXT *ctx,
325         const unsigned char *inp, unsigned char *out,
```



```
326         size_t len,ccm128_f stream)
327 {
328         size_t          n;
329         unsigned int    i,L;
330         unsigned char   flags0  = ctx->nonce.c[0];
331         block128_f      block   = ctx->block;
332         void *          key     = ctx->key;
333         union { u64 u[2]; u8 c[16]; } scratch;

335         if (!(flags0&0x40))
336                 (*block)(ctx->nonce.c,ctx->cmac.c,key),
337                 ctx->blocks++;

339         ctx->nonce.c[0] = L = flags0&7;
340         for (n=0,i=15-L;i<15;++i) {
341                 n |= ctx->nonce.c[i];
342                 ctx->nonce.c[i]=0;
343                 n <<= 8;
344         }
345         n |= ctx->nonce.c[15];  /* reconstructed length */
346         ctx->nonce.c[15]=1;

348         if (n!=len) return -1;  /* length mismatch */

350         ctx->blocks += ((len+15)>>3)|1;
351         if (ctx->blocks > (U64(1)<<61)) return -2; /* too much data */

353         if ((n=len/16)) {
354                 (*stream)(inp,out,n,key,ctx->nonce.c,ctx->cmac.c);
355                 n   *= 16;
356                 inp += n;
357                 out += n;
358                 len -= n;
359                 if (len) ctr64_add(ctx->nonce.c,n/16);
360         }

362         if (len) {
363                 for (i=0; i<len; ++i) ctx->cmac.c[i] ^= inp[i];
364                 (*block)(ctx->cmac.c,ctx->cmac.c,key);
365                 (*block)(ctx->nonce.c,scratch.c,key);
366                 for (i=0; i<len; ++i) out[i] = scratch.c[i]^inp[i];
367         }

369         for (i=15-L;i<16;++i)
370                 ctx->nonce.c[i]=0;

372         (*block)(ctx->nonce.c,scratch.c,key);
373         ctx->cmac.u[0] ^= scratch.u[0];
374         ctx->cmac.u[1] ^= scratch.u[1];

376         ctx->nonce.c[0] = flags0;

378         return 0;
379 }

381 int CRYPTO_ccm128_decrypt_ccm64(CCM128_CONTEXT *ctx,
382         const unsigned char *inp, unsigned char *out,
383         size_t len,ccm128_f stream)
384 {
385         size_t          n;
386         unsigned int    i,L;
387         unsigned char   flags0  = ctx->nonce.c[0];
388         block128_f      block   = ctx->block;
389         void *          key     = ctx->key;
390         union { u64 u[2]; u8 c[16]; } scratch;
```

```
392         if (!(flags0&0x40))
393                 (*block)(ctx->nonce.c,ctx->cmac.c,key);

395         ctx->nonce.c[0] = L = flags0&7;
396         for (n=0,i=15-L;i<15;++i) {
397                 n |= ctx->nonce.c[i];
398                 ctx->nonce.c[i]=0;
399                 n <<= 8;
400         }
401         n |= ctx->nonce.c[15];  /* reconstructed length */
402         ctx->nonce.c[15]=1;

404         if (n!=len) return -1;

406         if ((n=len/16)) {
407                 (*stream)(inp,out,n,key,ctx->nonce.c,ctx->cmac.c);
408                 n   *= 16;
409                 inp += n;
410                 out += n;
411                 len -= n;
412                 if (len) ctr64_add(ctx->nonce.c,n/16);
413         }

415         if (len) {
416                 (*block)(ctx->nonce.c,scratch.c,key);
417                 for (i=0; i<len; ++i)
418                         ctx->cmac.c[i] ^= (out[i] = scratch.c[i]^inp[i]);
419                 (*block)(ctx->cmac.c,ctx->cmac.c,key);
420         }

422         for (i=15-L;i<16;++i)
423                 ctx->nonce.c[i]=0;

425         (*block)(ctx->nonce.c,scratch.c,key);
426         ctx->cmac.u[0] ^= scratch.u[0];
427         ctx->cmac.u[1] ^= scratch.u[1];

429         ctx->nonce.c[0] = flags0;

431         return 0;
432 }

434 size_t CRYPTO_ccm128_tag(CCM128_CONTEXT *ctx,unsigned char *tag,size_t len)
435 {       unsigned int M = (ctx->nonce.c[0]>>3)&7;        /* the M parameter */

437         M *= 2; M += 2;
438         if (len<M)      return 0;
439         memcpy(tag,ctx->cmac.c,M);
440         return M;
441 }
442 #endif /* ! codereview */
```

```
**********************************************************
    7034 Wed Aug 13 19:52:54 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/cfb128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>
```



```
/* The input and output encrypted as though 128bit cfb mode is being
 * used.  The extra state information to record how much of the
 * 128bit block we have used is contained in *num;
 */
void CRYPTO_cfb128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], int *num,
                        int enc, block128_f block)
{
    unsigned int n;
    size_t l = 0;

    assert(in && out && key && ivec && num);

    n = *num;

    if (enc) {
#if !defined(OPENSSL_SMALL_FOOTPRINT)
        if (16%sizeof(size_t) == 0) do {        /* always true actually */
                while (n && len) {
                        *(out++) = ivec[n] ^= *(in++);
                        --len;
                        n = (n+1) % 16;
                }
#if defined(STRICT_ALIGNMENT)
                if (((size_t)in|(size_t)out|(size_t)ivec)%sizeof(size_t) != 0)
                        break;
#endif
                while (len>=16) {
                        (*block)(ivec, ivec, key);
                        for (; n<16; n+=sizeof(size_t)) {
                                *(size_t*)(out+n) =
                                *(size_t*)(ivec+n) ^= *(size_t*)(in+n);
                        }
                        len -= 16;
                        out += 16;
                        in  += 16;
                        n = 0;
                }
                if (len) {
                        (*block)(ivec, ivec, key);
                        while (len--) {
                                out[n] = ivec[n] ^= in[n];
                                ++n;
                        }
                }
                *num = n;
                return;
        } while (0);
        /* the rest would be commonly eliminated by x86* compiler */
#endif
        while (l<len) {
                if (n == 0) {
                        (*block)(ivec, ivec, key);
                }
                out[l] = ivec[n] ^= in[l];
                ++l;
                n = (n+1) % 16;
        }
        *num = n;
    } else {
#if !defined(OPENSSL_SMALL_FOOTPRINT)
        if (16%sizeof(size_t) == 0) do {        /* always true actually */
                while (n && len) {
                        unsigned char c;
                        *(out++) = ivec[n] ^ (c = *(in++)); ivec[n] = c;
```

```
128                         --len;
129                         n = (n+1) % 16;
130                 }
131 #if defined(STRICT_ALIGNMENT)
132                 if (((size_t)in|(size_t)out|(size_t)ivec)%sizeof(size_t) != 0)
133                         break;
134 #endif
135                 while (len>=16) {
136                         (*block)(ivec, ivec, key);
137                         for (; n<16; n+=sizeof(size_t)) {
138                                 size_t t = *(size_t*)(in+n);
139                                 *(size_t*)(out+n) = *(size_t*)(ivec+n) ^ t;
140                                 *(size_t*)(ivec+n) = t;
141                         }
142                         len -= 16;
143                         out += 16;
144                         in  += 16;
145                         n = 0;
146                 }
147                 if (len) {
148                         (*block)(ivec, ivec, key);
149                         while (len--) {
150                                 unsigned char c;
151                                 out[n] = ivec[n] ^ (c = in[n]); ivec[n] = c;
152                                 ++n;
153                         }
154                 }
155                 *num = n;
156                 return;
157         } while (0);
158         /* the rest would be commonly eliminated by x86* compiler */
159 #endif
160         while (l<len) {
161                 unsigned char c;
162                 if (n == 0) {
163                         (*block)(ivec, ivec, key);
164                 }
165                 out[l] = ivec[n] ^ (c = in[l]); ivec[n] = c;
166                 ++l;
167                 n = (n+1) % 16;
168         }
169         *num=n;
170     }
171 }

173 /* This expects a single block of size nbits for both in and out. Note that
174    it corrupts any extra bits in the last byte of out */
175 static void cfbr_encrypt_block(const unsigned char *in,unsigned char *out,
176                                int nbits,const void *key,
177                                unsigned char ivec[16],int enc,
178                                block128_f block)
179 {
180     int n,rem,num;
181     unsigned char ovec[16*2 + 1];  /* +1 because we dererefence (but don't use)

183     if (nbits<=0 || nbits>128) return;

185         /* fill in the first half of the new IV with the current IV */
186         memcpy(ovec,ivec,16);
187         /* construct the new IV */
188         (*block)(ivec,ivec,key);
189         num = (nbits+7)/8;
190         if (enc)        /* encrypt the input */
191             for(n=0 ; n < num ; ++n)
192                 out[n] = (ovec[16+n] = in[n] ^ ivec[n]);
193         else            /* decrypt the input */
```



```
194             for(n=0 ; n < num ; ++n)
195                 out[n] = (ovec[16+n] = in[n]) ^ ivec[n];
196         /* shift ovec left... */
197         rem = nbits%8;
198         num = nbits/8;
199         if(rem==0)
200             memcpy(ivec,ovec+num,16);
201         else
202             for(n=0 ; n < 16 ; ++n)
203                 ivec[n] = ovec[n+num]<<rem | ovec[n+num+1]>>(8-rem);

205     /* it is not necessary to cleanse ovec, since the IV is not secret */
206 }

208 /* N.B. This expects the input to be packed, MS bit first */
209 void CRYPTO_cfb128_1_encrypt(const unsigned char *in, unsigned char *out,
210                         size_t bits, const void *key,
211                         unsigned char ivec[16], int *num,
212                         int enc, block128_f block)
213 {
214     size_t n;
215     unsigned char c[1],d[1];

217     assert(in && out && key && ivec && num);
218     assert(*num == 0);

220     for(n=0 ; n<bits ; ++n)
221         {
222         c[0]=(in[n/8]&(1 << (7-n%8))) ? 0x80 : 0;
223         cfbr_encrypt_block(c,d,1,key,ivec,enc,block);
224         out[n/8]=(out[n/8]&~(1 << (unsigned int)(7-n%8))) |
225                  ((d[0]&0x80) >> (unsigned int)(n%8));
226         }
227 }

229 void CRYPTO_cfb128_8_encrypt(const unsigned char *in, unsigned char *out,
230                         size_t length, const void *key,
231                         unsigned char ivec[16], int *num,
232                         int enc, block128_f block)
233 {
234     size_t n;

236     assert(in && out && key && ivec && num);
237     assert(*num == 0);

239     for(n=0 ; n<length ; ++n)
240         cfbr_encrypt_block(&in[n],&out[n],8,key,ivec,enc,block);
241 }
242 #endif /* ! codereview */
```

```
**********************************************************
    6985 Wed Aug 13 19:52:54 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/ctr128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>
```



```
/* NOTE: the IV/counter CTR mode is big-endian.  The code itself
 * is endian-neutral. */

/* increment counter (128-bit int) by 1 */
static void ctr128_inc(unsigned char *counter) {
        u32 n=16;
        u8  c;

        do {
                --n;
                c = counter[n];
                ++c;
                counter[n] = c;
                if (c) return;
        } while (n);
}

#if !defined(OPENSSL_SMALL_FOOTPRINT)
static void ctr128_inc_aligned(unsigned char *counter) {
        size_t *data,c,n;
        const union { long one; char little; } is_endian = {1};

        if (is_endian.little) {
                ctr128_inc(counter);
                return;
        }

        data = (size_t *)counter;
        n = 16/sizeof(size_t);
        do {
                --n;
                c = data[n];
                ++c;
                data[n] = c;
                if (c) return;
        } while (n);
}
#endif

/* The input encrypted as though 128bit counter mode is being
 * used.  The extra state information to record how much of the
 * 128bit block we have used is contained in *num, and the
 * encrypted counter is kept in ecount_buf.  Both *num and
 * ecount_buf must be initialised with zeros before the first
 * call to CRYPTO_ctr128_encrypt().
 *
 * This algorithm assumes that the counter is in the x lower bits
 * of the IV (ivec), and that the application has full control over
 * overflow and the rest of the IV.  This implementation takes NO
 * responsability for checking that the counter doesn't overflow
 * into the rest of the IV when incremented.
 */
void CRYPTO_ctr128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], unsigned char ecount_buf[16],
                        unsigned int *num, block128_f block)
{
        unsigned int n;
        size_t l=0;

        assert(in && out && key && ecount_buf && num);
        assert(*num < 16);

        n = *num;

#if !defined(OPENSSL_SMALL_FOOTPRINT)
```

```
128         if (16%sizeof(size_t) == 0) do { /* always true actually */
129                 while (n && len) {
130                         *(out++) = *(in++) ^ ecount_buf[n];
131                         --len;
132                         n = (n+1) % 16;
133                 }
135 #if defined(STRICT_ALIGNMENT)
136                 if (((size_t)in|(size_t)out|(size_t)ivec)%sizeof(size_t) != 0)
137                         break;
138 #endif
139                 while (len>=16) {
140                         (*block)(ivec, ecount_buf, key);
141                         ctr128_inc_aligned(ivec);
142                         for (; n<16; n+=sizeof(size_t))
143                                 *(size_t *)(out+n) =
144                                 *(size_t *)(in+n) ^ *(size_t *)(ecount_buf+n);
145                         len -= 16;
146                         out += 16;
147                         in  += 16;
148                         n = 0;
149                 }
150                 if (len) {
151                         (*block)(ivec, ecount_buf, key);
152                         ctr128_inc_aligned(ivec);
153                         while (len--) {
154                                 out[n] = in[n] ^ ecount_buf[n];
155                                 ++n;
156                         }
157                 }
158                 *num = n;
159                 return;
160         } while(0);
161         /* the rest would be commonly eliminated by x86* compiler */
162 #endif
163         while (l<len) {
164                 if (n==0) {
165                         (*block)(ivec, ecount_buf, key);
166                         ctr128_inc(ivec);
167                 }
168                 out[l] = in[l] ^ ecount_buf[n];
169                 ++l;
170                 n = (n+1) % 16;
171         }
173         *num=n;
174 }
176 /* increment upper 96 bits of 128-bit counter by 1 */
177 static void ctr96_inc(unsigned char *counter) {
178         u32 n=12;
179         u8  c;
181         do {
182                 --n;
183                 c = counter[n];
184                 ++c;
185                 counter[n] = c;
186                 if (c) return;
187         } while (n);
188 }
190 void CRYPTO_ctr128_encrypt_ctr32(const unsigned char *in, unsigned char *out,
191                         size_t len, const void *key,
192                         unsigned char ivec[16], unsigned char ecount_buf[16],
193                         unsigned int *num, ctr128_f func)
```



```
194 {
195         unsigned int n,ctr32;
197         assert(in && out && key && ecount_buf && num);
198         assert(*num < 16);
200         n = *num;
202         while (n && len) {
203                 *(out++) = *(in++) ^ ecount_buf[n];
204                 --len;
205                 n = (n+1) % 16;
206         }
208         ctr32 = GETU32(ivec+12);
209         while (len>=16) {
210                 size_t blocks = len/16;
211                 /*
212                  * 1<<28 is just a not-so-small yet not-so-large number...
213                  * Below condition is practically never met, but it has to
214                  * be checked for code correctness.
215                  */
216                 if (sizeof(size_t)>sizeof(unsigned int) && blocks>(1U<<28))
217                         blocks = (1U<<28);
218                 /*
219                  * As (*func) operates on 32-bit counter, caller
220                  * has to handle overflow. 'if' below detects the
221                  * overflow, which is then handled by limiting the
222                  * amount of blocks to the exact overflow point...
223                  */
224                 ctr32 += (u32)blocks;
225                 if (ctr32 < blocks) {
226                         blocks -= ctr32;
227                         ctr32   = 0;
228                 }
229                 (*func)(in,out,blocks,key,ivec);
230                 /* (*ctr) does not update ivec, caller does: */
231                 PUTU32(ivec+12,ctr32);
232                 /* ... overflow was detected, propogate carry. */
233                 if (ctr32 == 0) ctr96_inc(ivec);
234                 blocks *= 16;
235                 len -= blocks;
236                 out += blocks;
237                 in  += blocks;
238         }
239         if (len) {
240                 memset(ecount_buf,0,16);
241                 (*func)(ecount_buf,ecount_buf,1,key,ivec);
242                 ++ctr32;
243                 PUTU32(ivec+12,ctr32);
244                 if (ctr32 == 0) ctr96_inc(ivec);
245                 while (len--) {
246                         out[n] = in[n] ^ ecount_buf[n];
247                         ++n;
248                 }
249         }
251         *num=n;
252 }
253 #endif /* ! codereview */
```

```
**********************************************************
   12795 Wed Aug 13 19:52:55 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/cts128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project. All rights reserved.
 *
 * Rights for redistribution and usage in source and binary
 * forms are granted according to the OpenSSL license.
 */

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>

/*
 * Trouble with Ciphertext Stealing, CTS, mode is that there is no
 * common official specification, but couple of cipher/application
 * specific ones: RFC2040 and RFC3962. Then there is 'Proposal to
 * Extend CBC Mode By "Ciphertext Stealing"' at NIST site, which
 * deviates from mentioned RFCs. Most notably it allows input to be
 * of block length and it doesn't flip the order of the last two
 * blocks. CTS is being discussed even in ECB context, but it's not
 * adopted for any known application. This implementation provides
 * two interfaces: one compliant with above mentioned RFCs and one
 * compliant with the NIST proposal, both extending CBC mode.
 */

size_t CRYPTO_cts128_encrypt_block(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block)
{       size_t residue, n;

        assert (in && out && key && ivec);

        if (len <= 16) return 0;

        if ((residue=len%16) == 0) residue = 16;

        len -= residue;

        CRYPTO_cbc128_encrypt(in,out,len,key,ivec,block);

        in  += len;
        out += len;

        for (n=0; n<residue; ++n)
                ivec[n] ^= in[n];
        (*block)(ivec,ivec,key);
        memcpy(out,out-16,residue);
        memcpy(out-16,ivec,16);

        return len+residue;
}

size_t CRYPTO_nistcts128_encrypt_block(const unsigned char *in, unsigned char *o
                        size_t len, const void *key,
                        unsigned char ivec[16], block128_f block)
```



```
{       size_t residue, n;

        assert (in && out && key && ivec);

        if (len < 16) return 0;

        residue=len%16;

        len -= residue;

        CRYPTO_cbc128_encrypt(in,out,len,key,ivec,block);

        if (residue==0) return len;

        in  += len;
        out += len;

        for (n=0; n<residue; ++n)
                ivec[n] ^= in[n];
        (*block)(ivec,ivec,key);
        memcpy(out-16+residue,ivec,16);

        return len+residue;
}

size_t CRYPTO_cts128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], cbc128_f cbc)
{       size_t residue;
        union { size_t align; unsigned char c[16]; } tmp;

        assert (in && out && key && ivec);

        if (len <= 16) return 0;

        if ((residue=len%16) == 0) residue = 16;

        len -= residue;

        (*cbc)(in,out,len,key,ivec,1);

        in  += len;
        out += len;

#if defined(CBC_HANDLES_TRUNCATED_IO)
        memcpy(tmp.c,out-16,16);
        (*cbc)(in,out-16,residue,key,ivec,1);
        memcpy(out,tmp.c,residue);
#else
        memset(tmp.c,0,sizeof(tmp));
        memcpy(tmp.c,in,residue);
        memcpy(out,out-16,residue);
        (*cbc)(tmp.c,out-16,16,key,ivec,1);
#endif
        return len+residue;
}

size_t CRYPTO_nistcts128_encrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], cbc128_f cbc)
{       size_t residue;
        union { size_t align; unsigned char c[16]; } tmp;

        assert (in && out && key && ivec);

        if (len < 16) return 0;
```

```
129         residue=len%16;

131         len -= residue;

133         (*cbc)(in,out,len,key,ivec,1);

135         if (residue==0) return len;

137         in  += len;
138         out += len;

140 #if defined(CBC_HANDLES_TRUNCATED_IO)
141         (*cbc)(in,out-16+residue,residue,key,ivec,1);
142 #else
143         memset(tmp.c,0,sizeof(tmp));
144         memcpy(tmp.c,in,residue);
145         (*cbc)(tmp.c,out-16+residue,16,key,ivec,1);
146 #endif
147         return len+residue;
148 }

150 size_t CRYPTO_cts128_decrypt_block(const unsigned char *in, unsigned char *out,
151                         size_t len, const void *key,
152                         unsigned char ivec[16], block128_f block)
153 {       size_t residue, n;
154         union { size_t align; unsigned char c[32]; } tmp;

156         assert (in && out && key && ivec);

158         if (len<=16) return 0;

160         if ((residue=len%16) == 0) residue = 16;

162         len -= 16+residue;

164         if (len) {
165                 CRYPTO_cbc128_decrypt(in,out,len,key,ivec,block);
166                 in  += len;
167                 out += len;
168         }

170         (*block)(in,tmp.c+16,key);

172         memcpy(tmp.c,tmp.c+16,16);
173         memcpy(tmp.c,in+16,residue);
174         (*block)(tmp.c,tmp.c,key);

176         for(n=0; n<16; ++n) {
177                 unsigned char c = in[n];
178                 out[n] = tmp.c[n] ^ ivec[n];
179                 ivec[n] = c;
180         }
181         for(residue+=16; n<residue; ++n)
182                 out[n] = tmp.c[n] ^ in[n];

184         return 16+len+residue;
185 }

187 size_t CRYPTO_nistcts128_decrypt_block(const unsigned char *in, unsigned char *o
188                         size_t len, const void *key,
189                         unsigned char ivec[16], block128_f block)
190 {       size_t residue, n;
191         union { size_t align; unsigned char c[32]; } tmp;

193         assert (in && out && key && ivec);
```



```
195         if (len<16) return 0;

197         residue=len%16;

199         if (residue==0) {
200                 CRYPTO_cbc128_decrypt(in,out,len,key,ivec,block);
201                 return len;
202         }

204         len -= 16+residue;

206         if (len) {
207                 CRYPTO_cbc128_decrypt(in,out,len,key,ivec,block);
208                 in  += len;
209                 out += len;
210         }

212         (*block)(in+residue,tmp.c+16,key);

214         memcpy(tmp.c,tmp.c+16,16);
215         memcpy(tmp.c,in,residue);
216         (*block)(tmp.c,tmp.c,key);

218         for(n=0; n<16; ++n) {
219                 unsigned char c = in[n];
220                 out[n] = tmp.c[n] ^ ivec[n];
221                 ivec[n] = in[n+residue];
222                 tmp.c[n] = c;
223         }
224         for(residue+=16; n<residue; ++n)
225                 out[n] = tmp.c[n] ^ tmp.c[n-16];

227         return 16+len+residue;
228 }

230 size_t CRYPTO_cts128_decrypt(const unsigned char *in, unsigned char *out,
231                         size_t len, const void *key,
232                         unsigned char ivec[16], cbc128_f cbc)
233 {       size_t residue;
234         union { size_t align; unsigned char c[32]; } tmp;

236         assert (in && out && key && ivec);

238         if (len<=16) return 0;

240         if ((residue=len%16) == 0) residue = 16;

242         len -= 16+residue;

244         if (len) {
245                 (*cbc)(in,out,len,key,ivec,0);
246                 in  += len;
247                 out += len;
248         }

250         memset(tmp.c,0,sizeof(tmp));
251         /* this places in[16] at &tmp.c[16] and decrypted block at &tmp.c[0] */
252         (*cbc)(in,tmp.c,16,key,tmp.c+16,0);

254         memcpy(tmp.c,in+16,residue);
255 #if defined(CBC_HANDLES_TRUNCATED_IO)
256         (*cbc)(tmp.c,out,16+residue,key,ivec,0);
257 #else
258         (*cbc)(tmp.c,tmp.c,32,key,ivec,0);
259         memcpy(out,tmp.c,16+residue);
```

```
#endif
        return 16+len+residue;
}

size_t CRYPTO_nistcts128_decrypt(const unsigned char *in, unsigned char *out,
                        size_t len, const void *key,
                        unsigned char ivec[16], cbc128_f cbc)
{       size_t residue;
        union { size_t align; unsigned char c[32]; } tmp;

        assert (in && out && key && ivec);

        if (len<16) return 0;

        residue=len%16;

        if (residue==0) {
                (*cbc)(in,out,len,key,ivec,0);
                return len;
        }

        len -= 16+residue;

        if (len) {
                (*cbc)(in,out,len,key,ivec,0);
                in  += len;
                out += len;
        }

        memset(tmp.c,0,sizeof(tmp));
        /* this places in[16] at &tmp.c[16] and decrypted block at &tmp.c[0] */
        (*cbc)(in+residue,tmp.c,16,key,tmp.c+16,0);

        memcpy(tmp.c,in,residue);
#if defined(CBC_HANDLES_TRUNCATED_IO)
        (*cbc)(tmp.c,out,16+residue,key,ivec,0);
#else
        (*cbc)(tmp.c,tmp.c,32,key,ivec,0);
        memcpy(out,tmp.c,16+residue);
#endif
        return 16+len+residue;
}

#if defined(SELFTEST)
#include <stdio.h>
#include <openssl/aes.h>

/* test vectors from RFC 3962 */
static const unsigned char test_key[16] = "chicken teriyaki";
static const unsigned char test_input[64] =
                "I would like the" " General Gau's C"
                "hicken, please, " "and wonton soup.";
static const unsigned char test_iv[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

static const unsigned char vector_17[17] =
{0xc6,0x35,0x35,0x68,0xf2,0xbf,0x8c,0xb4, 0xd8,0xa5,0x80,0x36,0x2d,0xa7,0xff,0x7
 0x97};
static const unsigned char vector_31[31] =
{0xfc,0x00,0x78,0x3e,0x0e,0xfd,0xb2,0xc1, 0xd4,0x45,0xd4,0xc8,0xef,0xf7,0xed,0x2
 0x97,0x68,0x72,0x68,0xd6,0xec,0xcc,0xc0, 0xc0,0x7b,0x25,0xe2,0x5e,0xcf,0xe5};
static const unsigned char vector_32[32] =
{0x39,0x31,0x25,0x23,0xa7,0x86,0x62,0xd5, 0xbe,0x7f,0xcb,0xcc,0x98,0xeb,0xf5,0xa
 0x97,0x68,0x72,0x68,0xd6,0xec,0xcc,0xc0, 0xc0,0x7b,0x25,0xe2,0x5e,0xcf,0xe5,0x8
static const unsigned char vector_47[47] =
{0x97,0x68,0x72,0x68,0xd6,0xec,0xcc,0xc0, 0xc0,0x7b,0x25,0xe2,0x5e,0xcf,0xe5,0x8
 0xb3,0xff,0xfd,0x94,0x0c,0x16,0xa1,0x8c, 0x1b,0x55,0x49,0xd2,0xf8,0x38,0x02,0x9
```



```
 0x39,0x31,0x25,0x23,0xa7,0x86,0x62,0xd5, 0xbe,0x7f,0xcb,0xcc,0x98,0xeb,0xf5};
static const unsigned char vector_48[48] =
{0x97,0x68,0x72,0x68,0xd6,0xec,0xcc,0xc0, 0xc0,0x7b,0x25,0xe2,0x5e,0xcf,0xe5,0x8
 0x9d,0xad,0x8b,0xbb,0x96,0xc4,0xcd,0xc0, 0x3b,0xc1,0x03,0xe1,0xa1,0x94,0xbb,0xd
 0x39,0x31,0x25,0x23,0xa7,0x86,0x62,0xd5, 0xbe,0x7f,0xcb,0xcc,0x98,0xeb,0xf5,0xa
static const unsigned char vector_64[64] =
{0x97,0x68,0x72,0x68,0xd6,0xec,0xcc,0xc0, 0xc0,0x7b,0x25,0xe2,0x5e,0xcf,0xe5,0x8
 0x39,0x31,0x25,0x23,0xa7,0x86,0x62,0xd5, 0xbe,0x7f,0xcb,0xcc,0x98,0xeb,0xf5,0xa
 0x48,0x07,0xef,0xe8,0x36,0xee,0x89,0xa5, 0x26,0x73,0x0d,0xbc,0x2f,0x7b,0xc8,0x4
 0x9d,0xad,0x8b,0xbb,0x96,0xc4,0xcd,0xc0, 0x3b,0xc1,0x03,0xe1,0xa1,0x94,0xbb,0xd

static AES_KEY encks, decks;

void test_vector(const unsigned char *vector,size_t len)
{       unsigned char iv[sizeof(test_iv)];
        unsigned char cleartext[64],ciphertext[64];
        size_t tail;

        printf("vector_%d\n",len); fflush(stdout);

        if ((tail=len%16) == 0) tail = 16;
        tail += 16;

        /* test block-based encryption */
        memcpy(iv,test_iv,sizeof(test_iv));
        CRYPTO_cts128_encrypt_block(test_input,ciphertext,len,&encks,iv,(block12
        if (memcmp(ciphertext,vector,len))
                fprintf(stderr,"output_%d mismatch\n",len), exit(1);
        if (memcmp(iv,vector+len-tail,sizeof(iv)))
                fprintf(stderr,"iv_%d mismatch\n",len), exit(1);

        /* test block-based decryption */
        memcpy(iv,test_iv,sizeof(test_iv));
        CRYPTO_cts128_decrypt_block(ciphertext,cleartext,len,&decks,iv,(block128
        if (memcmp(cleartext,test_input,len))
                fprintf(stderr,"input_%d mismatch\n",len), exit(2);
        if (memcmp(iv,vector+len-tail,sizeof(iv)))
                fprintf(stderr,"iv_%d mismatch\n",len), exit(2);

        /* test streamed encryption */
        memcpy(iv,test_iv,sizeof(test_iv));
        CRYPTO_cts128_encrypt(test_input,ciphertext,len,&encks,iv,(cbc128_f)AES_
        if (memcmp(ciphertext,vector,len))
                fprintf(stderr,"output_%d mismatch\n",len), exit(3);
        if (memcmp(iv,vector+len-tail,sizeof(iv)))
                fprintf(stderr,"iv_%d mismatch\n",len), exit(3);

        /* test streamed decryption */
        memcpy(iv,test_iv,sizeof(test_iv));
        CRYPTO_cts128_decrypt(ciphertext,cleartext,len,&decks,iv,(cbc128_f)AES_c
        if (memcmp(cleartext,test_input,len))
                fprintf(stderr,"input_%d mismatch\n",len), exit(4);
        if (memcmp(iv,vector+len-tail,sizeof(iv)))
                fprintf(stderr,"iv_%d mismatch\n",len), exit(4);
}

void test_nistvector(const unsigned char *vector,size_t len)
{       unsigned char iv[sizeof(test_iv)];
        unsigned char cleartext[64],ciphertext[64],nistvector[64];
        size_t tail;

        printf("nistvector_%d\n",len); fflush(stdout);

        if ((tail=len%16) == 0) tail = 16;

        len -= 16 + tail;
```

```
392         memcpy(nistvector,vector,len);
393         /* flip two last blocks */
394         memcpy(nistvector+len,vector+len+16,tail);
395         memcpy(nistvector+len+tail,vector+len,16);
396         len += 16 + tail;
397         tail = 16;

399         /* test block-based encryption */
400         memcpy(iv,test_iv,sizeof(test_iv));
401         CRYPTO_nistcts128_encrypt_block(test_input,ciphertext,len,&encks,iv,(blo
402         if (memcmp(ciphertext,nistvector,len))
403                 fprintf(stderr,"output_%d mismatch\n",len), exit(1);
404         if (memcmp(iv,nistvector+len-tail,sizeof(iv)))
405                 fprintf(stderr,"iv_%d mismatch\n",len), exit(1);

407         /* test block-based decryption */
408         memcpy(iv,test_iv,sizeof(test_iv));
409         CRYPTO_nistcts128_decrypt_block(ciphertext,cleartext,len,&decks,iv,(bloc
410         if (memcmp(cleartext,test_input,len))
411                 fprintf(stderr,"input_%d mismatch\n",len), exit(2);
412         if (memcmp(iv,nistvector+len-tail,sizeof(iv)))
413                 fprintf(stderr,"iv_%d mismatch\n",len), exit(2);

415         /* test streamed encryption */
416         memcpy(iv,test_iv,sizeof(test_iv));
417         CRYPTO_nistcts128_encrypt(test_input,ciphertext,len,&encks,iv,(cbc128_f)
418         if (memcmp(ciphertext,nistvector,len))
419                 fprintf(stderr,"output_%d mismatch\n",len), exit(3);
420         if (memcmp(iv,nistvector+len-tail,sizeof(iv)))
421                 fprintf(stderr,"iv_%d mismatch\n",len), exit(3);

423         /* test streamed decryption */
424         memcpy(iv,test_iv,sizeof(test_iv));
425         CRYPTO_nistcts128_decrypt(ciphertext,cleartext,len,&decks,iv,(cbc128_f)A
426         if (memcmp(cleartext,test_input,len))
427                 fprintf(stderr,"input_%d mismatch\n",len), exit(4);
428         if (memcmp(iv,nistvector+len-tail,sizeof(iv)))
429                 fprintf(stderr,"iv_%d mismatch\n",len), exit(4);
430 }

432 int main()
433 {
434         AES_set_encrypt_key(test_key,128,&encks);
435         AES_set_decrypt_key(test_key,128,&decks);

437         test_vector(vector_17,sizeof(vector_17));
438         test_vector(vector_31,sizeof(vector_31));
439         test_vector(vector_32,sizeof(vector_32));
440         test_vector(vector_47,sizeof(vector_47));
441         test_vector(vector_48,sizeof(vector_48));
442         test_vector(vector_64,sizeof(vector_64));

444         test_nistvector(vector_17,sizeof(vector_17));
445         test_nistvector(vector_31,sizeof(vector_31));
446         test_nistvector(vector_32,sizeof(vector_32));
447         test_nistvector(vector_47,sizeof(vector_47));
448         test_nistvector(vector_48,sizeof(vector_48));
449         test_nistvector(vector_64,sizeof(vector_64));

451         return 0;
452 }
453 #endif
454 #endif /* ! codereview */
```

```
**********************************************************
   55945 Wed Aug 13 19:52:55 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/gcm128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ====================================================================
 * Copyright (c) 2010 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#define OPENSSL_FIPSAPI

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>
```



```
#if defined(BSWAP4) && defined(STRICT_ALIGNMENT)
/* redefine, because alignment is ensured */
#undef  GETU32
#define GETU32(p)       BSWAP4(*(const u32 *)(p))
#undef  PUTU32
#define PUTU32(p,v)     *(u32 *)(p) = BSWAP4(v)
#endif

#define PACK(s)         ((size_t)(s)<<(sizeof(size_t)*8-16))
#define REDUCE1BIT(V)   do { \
        if (sizeof(size_t)==8) { \
                u64 T = U64(0xe100000000000000) & (0-(V.lo&1)); \
                V.lo  = (V.hi<<63)|(V.lo>>1); \
                V.hi  = (V.hi>>1 )^T; \
        } \
        else { \
                u32 T = 0xe1000000U & (0-(u32)(V.lo&1)); \
                V.lo  = (V.hi<<63)|(V.lo>>1); \
                V.hi  = (V.hi>>1 )^((u64)T<<32); \
        } \
} while(0)

/*
 * Even though permitted values for TABLE_BITS are 8, 4 and 1, it should
 * never be set to 8. 8 is effectively reserved for testing purposes.
 * TABLE_BITS>1 are lookup-table-driven implementations referred to as
 * "Shoup's" in GCM specification. In other words OpenSSL does not cover
 * whole spectrum of possible table driven implementations. Why? In
 * non-"Shoup's" case memory access pattern is segmented in such manner,
 * that it's trivial to see that cache timing information can reveal
 * fair portion of intermediate hash value. Given that ciphertext is
 * always available to attacker, it's possible for him to attempt to
 * deduce secret parameter H and if successful, tamper with messages
 * [which is nothing but trivial in CTR mode]. In "Shoup's" case it's
 * not as trivial, but there is no reason to believe that it's resistant
 * to cache-timing attack. And the thing about "8-bit" implementation is
 * that it consumes 16 (sixteen) times more memory, 4KB per individual
 * key + 1KB shared. Well, on pros side it should be twice as fast as
 * "4-bit" version. And for gcc-generated x86[_64] code, "8-bit" version
 * was observed to run ~75% faster, closer to 100% for commercial
 * compilers... Yet "4-bit" procedure is preferred, because it's
 * believed to provide better security-performance balance and adequate
 * all-round performance. "All-round" refers to things like:
 *
 * - shorter setup time effectively improves overall timing for
 *   handling short messages;
 * - larger table allocation can become unbearable because of VM
 *   subsystem penalties (for example on Windows large enough free
 *   results in VM working set trimming, meaning that consequent
 *   malloc would immediately incur working set expansion);
 * - larger table has larger cache footprint, which can affect
 *   performance of other code paths (not necessarily even from same
 *   thread in Hyper-Threading world);
 *
 * Value of 1 is not appropriate for performance reasons.
 */
#if     TABLE_BITS==8

static void gcm_init_8bit(u128 Htable[256], u64 H[2])
{
        int  i, j;
        u128 V;

        Htable[0].hi = 0;
        Htable[0].lo = 0;
```

```
128         V.hi = H[0];
129         V.lo = H[1];
130
131         for (Htable[128]=V, i=64; i>0; i>>=1) {
132                 REDUCE1BIT(V);
133                 Htable[i] = V;
134         }
135
136         for (i=2; i<256; i<<=1) {
137                 u128 *Hi = Htable+i, H0 = *Hi;
138                 for (j=1; j<i; ++j) {
139                         Hi[j].hi = H0.hi^Htable[j].hi;
140                         Hi[j].lo = H0.lo^Htable[j].lo;
141                 }
142         }
143 }
144
145 static void gcm_gmult_8bit(u64 Xi[2], const u128 Htable[256])
146 {
147         u128 Z = { 0, 0};
148         const u8 *xi = (const u8 *)Xi+15;
149         size_t rem, n = *xi;
150         const union { long one; char little; } is_endian = {1};
151         static const size_t rem_8bit[256] = {
152                 PACK(0x0000), PACK(0x01C2), PACK(0x0384), PACK(0x0246),
153                 PACK(0x0708), PACK(0x06CA), PACK(0x048C), PACK(0x054E),
154                 PACK(0x0E10), PACK(0x0FD2), PACK(0x0D94), PACK(0x0C56),
155                 PACK(0x0918), PACK(0x08DA), PACK(0x0A9C), PACK(0x0B5E),
156                 PACK(0x1C20), PACK(0x1DE2), PACK(0x1FA4), PACK(0x1E66),
157                 PACK(0x1B28), PACK(0x1AEA), PACK(0x18AC), PACK(0x196E),
158                 PACK(0x1230), PACK(0x13F2), PACK(0x11B4), PACK(0x1076),
159                 PACK(0x1538), PACK(0x14FA), PACK(0x16BC), PACK(0x177E),
160                 PACK(0x3840), PACK(0x3982), PACK(0x3BC4), PACK(0x3A06),
161                 PACK(0x3F48), PACK(0x3E8A), PACK(0x3CCC), PACK(0x3D0E),
162                 PACK(0x3650), PACK(0x3792), PACK(0x35D4), PACK(0x3416),
163                 PACK(0x3158), PACK(0x309A), PACK(0x32DC), PACK(0x331E),
164                 PACK(0x2460), PACK(0x25A2), PACK(0x27E4), PACK(0x2626),
165                 PACK(0x2368), PACK(0x22AA), PACK(0x20EC), PACK(0x212E),
166                 PACK(0x2A70), PACK(0x2BB2), PACK(0x29F4), PACK(0x2836),
167                 PACK(0x2D78), PACK(0x2CBA), PACK(0x2EFC), PACK(0x2F3E),
168                 PACK(0x7080), PACK(0x7142), PACK(0x7304), PACK(0x72C6),
169                 PACK(0x7788), PACK(0x764A), PACK(0x740C), PACK(0x75CE),
170                 PACK(0x7E90), PACK(0x7F52), PACK(0x7D14), PACK(0x7CD6),
171                 PACK(0x7998), PACK(0x785A), PACK(0x7A1C), PACK(0x7BDE),
172                 PACK(0x6CA0), PACK(0x6D62), PACK(0x6F24), PACK(0x6EE6),
173                 PACK(0x6BA8), PACK(0x6A6A), PACK(0x682C), PACK(0x69EE),
174                 PACK(0x62B0), PACK(0x6372), PACK(0x6134), PACK(0x60F6),
175                 PACK(0x65B8), PACK(0x647A), PACK(0x663C), PACK(0x67FE),
176                 PACK(0x48C0), PACK(0x4902), PACK(0x4B44), PACK(0x4A86),
177                 PACK(0x4FC8), PACK(0x4E0A), PACK(0x4C4C), PACK(0x4D8E),
178                 PACK(0x46D0), PACK(0x4712), PACK(0x4554), PACK(0x4496),
179                 PACK(0x41D8), PACK(0x401A), PACK(0x425C), PACK(0x439E),
180                 PACK(0x54E0), PACK(0x5522), PACK(0x5764), PACK(0x56A6),
181                 PACK(0x53E8), PACK(0x522A), PACK(0x506C), PACK(0x51AE),
182                 PACK(0x5AF0), PACK(0x5B32), PACK(0x5974), PACK(0x58B6),
183                 PACK(0x5DF8), PACK(0x5C3A), PACK(0x5E7C), PACK(0x5FBE),
184                 PACK(0xE100), PACK(0xE0C2), PACK(0xE284), PACK(0xE346),
185                 PACK(0xE608), PACK(0xE7CA), PACK(0xE58C), PACK(0xE44E),
186                 PACK(0xEF10), PACK(0xEED2), PACK(0xEC94), PACK(0xED56),
187                 PACK(0xE818), PACK(0xE9DA), PACK(0xEB9C), PACK(0xEA5E),
188                 PACK(0xFD20), PACK(0xFCE2), PACK(0xFEA4), PACK(0xFF66),
189                 PACK(0xFA28), PACK(0xFBEA), PACK(0xF9AC), PACK(0xF86E),
190                 PACK(0xF330), PACK(0xF2F2), PACK(0xF0B4), PACK(0xF176),
191                 PACK(0xF438), PACK(0xF5FA), PACK(0xF7BC), PACK(0xF67E),
192                 PACK(0xD940), PACK(0xD882), PACK(0xDAC4), PACK(0xDB06),
193                 PACK(0xDE48), PACK(0xDF8A), PACK(0xDDCC), PACK(0xDC0E),
```



```
194                 PACK(0xD750), PACK(0xD692), PACK(0xD4D4), PACK(0xD516),
195                 PACK(0xD058), PACK(0xD19A), PACK(0xD3DC), PACK(0xD21E),
196                 PACK(0xC560), PACK(0xC4A2), PACK(0xC6E4), PACK(0xC726),
197                 PACK(0xC268), PACK(0xC3AA), PACK(0xC1EC), PACK(0xC02E),
198                 PACK(0xCB70), PACK(0xCAB2), PACK(0xC8F4), PACK(0xC936),
199                 PACK(0xCC78), PACK(0xCDBA), PACK(0xCFFC), PACK(0xCE3E),
200                 PACK(0x9180), PACK(0x9042), PACK(0x9204), PACK(0x93C6),
201                 PACK(0x9688), PACK(0x974A), PACK(0x950C), PACK(0x94CE),
202                 PACK(0x9F90), PACK(0x9E52), PACK(0x9C14), PACK(0x9DD6),
203                 PACK(0x9898), PACK(0x995A), PACK(0x9B1C), PACK(0x9ADE),
204                 PACK(0x8DA0), PACK(0x8C62), PACK(0x8E24), PACK(0x8FE6),
205                 PACK(0x8AA8), PACK(0x8B6A), PACK(0x892C), PACK(0x88EE),
206                 PACK(0x83B0), PACK(0x8272), PACK(0x8034), PACK(0x81F6),
207                 PACK(0x84B8), PACK(0x857A), PACK(0x873C), PACK(0x86FE),
208                 PACK(0xA9C0), PACK(0xA802), PACK(0xAA44), PACK(0xAB86),
209                 PACK(0xAEC8), PACK(0xAF0A), PACK(0xAD4C), PACK(0xAC8E),
210                 PACK(0xA7D0), PACK(0xA612), PACK(0xA454), PACK(0xA596),
211                 PACK(0xA0D8), PACK(0xA11A), PACK(0xA35C), PACK(0xA29E),
212                 PACK(0xB5E0), PACK(0xB422), PACK(0xB664), PACK(0xB7A6),
213                 PACK(0xB2E8), PACK(0xB32A), PACK(0xB16C), PACK(0xB0AE),
214                 PACK(0xBBF0), PACK(0xBA32), PACK(0xB874), PACK(0xB9B6),
215                 PACK(0xBCF8), PACK(0xBD3A), PACK(0xBF7C), PACK(0xBEBE) };
216
217         while (1) {
218                 Z.hi ^= Htable[n].hi;
219                 Z.lo ^= Htable[n].lo;
220
221                 if ((u8 *)Xi==xi)       break;
222
223                 n = *(--xi);
224
225                 rem  = (size_t)Z.lo&0xff;
226                 Z.lo = (Z.hi<<56)|(Z.lo>>8);
227                 Z.hi = (Z.hi>>8);
228                 if (sizeof(size_t)==8)
229                         Z.hi ^= rem_8bit[rem];
230                 else
231                         Z.hi ^= (u64)rem_8bit[rem]<<32;
232         }
233
234         if (is_endian.little) {
235 #ifdef BSWAP8
236                 Xi[0] = BSWAP8(Z.hi);
237                 Xi[1] = BSWAP8(Z.lo);
238 #else
239                 u8 *p = (u8 *)Xi;
240                 u32 v;
241                 v = (u32)(Z.hi>>32);    PUTU32(p,v);
242                 v = (u32)(Z.hi);        PUTU32(p+4,v);
243                 v = (u32)(Z.lo>>32);    PUTU32(p+8,v);
244                 v = (u32)(Z.lo);        PUTU32(p+12,v);
245 #endif
246         }
247         else {
248                 Xi[0] = Z.hi;
249                 Xi[1] = Z.lo;
250         }
251 }
252 #define GCM_MUL(ctx,Xi)   gcm_gmult_8bit(ctx->Xi.u,ctx->Htable)
253
254 #elif   TABLE_BITS==4
255
256 static void gcm_init_4bit(u128 Htable[16], u64 H[2])
257 {
258         u128 V;
259 #if defined(OPENSSL_SMALL_FOOTPRINT)
```

```
260         int  i;
261 #endif

263         Htable[0].hi = 0;
264         Htable[0].lo = 0;
265         V.hi = H[0];
266         V.lo = H[1];

268 #if defined(OPENSSL_SMALL_FOOTPRINT)
269         for (Htable[8]=V, i=4; i>0; i>>=1) {
270                 REDUCE1BIT(V);
271                 Htable[i] = V;
272         }

274         for (i=2; i<16; i<<=1) {
275                 u128 *Hi = Htable+i;
276                 int   j;
277                 for (V=*Hi, j=1; j<i; ++j) {
278                         Hi[j].hi = V.hi^Htable[j].hi;
279                         Hi[j].lo = V.lo^Htable[j].lo;
280                 }
281         }
282 #else
283         Htable[8] = V;
284         REDUCE1BIT(V);
285         Htable[4] = V;
286         REDUCE1BIT(V);
287         Htable[2] = V;
288         REDUCE1BIT(V);
289         Htable[1] = V;
290         Htable[3].hi  = V.hi^Htable[2].hi, Htable[3].lo  = V.lo^Htable[2].lo;
291         V=Htable[4];
292         Htable[5].hi  = V.hi^Htable[1].hi, Htable[5].lo  = V.lo^Htable[1].lo;
293         Htable[6].hi  = V.hi^Htable[2].hi, Htable[6].lo  = V.lo^Htable[2].lo;
294         Htable[7].hi  = V.hi^Htable[3].hi, Htable[7].lo  = V.lo^Htable[3].lo;
295         V=Htable[8];
296         Htable[9].hi  = V.hi^Htable[1].hi, Htable[9].lo  = V.lo^Htable[1].lo;
297         Htable[10].hi = V.hi^Htable[2].hi, Htable[10].lo = V.lo^Htable[2].lo;
298         Htable[11].hi = V.hi^Htable[3].hi, Htable[11].lo = V.lo^Htable[3].lo;
299         Htable[12].hi = V.hi^Htable[4].hi, Htable[12].lo = V.lo^Htable[4].lo;
300         Htable[13].hi = V.hi^Htable[5].hi, Htable[13].lo = V.lo^Htable[5].lo;
301         Htable[14].hi = V.hi^Htable[6].hi, Htable[14].lo = V.lo^Htable[6].lo;
302         Htable[15].hi = V.hi^Htable[7].hi, Htable[15].lo = V.lo^Htable[7].lo;
303 #endif
304 #if defined(GHASH_ASM) && (defined(__arm__) || defined(__arm))
305         /*
306          * ARM assembler expects specific dword order in Htable.
307          */
308         {
309         int j;
310         const union { long one; char little; } is_endian = {1};

312         if (is_endian.little)
313                 for (j=0;j<16;++j) {
314                         V = Htable[j];
315                         Htable[j].hi = V.lo;
316                         Htable[j].lo = V.hi;
317                 }
318         else
319                 for (j=0;j<16;++j) {
320                         V = Htable[j];
321                         Htable[j].hi = V.lo<<32|V.lo>>32;
322                         Htable[j].lo = V.hi<<32|V.hi>>32;
323                 }
324         }
325 #endif
```



```
326 }

328 #ifndef GHASH_ASM
329 static const size_t rem_4bit[16] = {
330         PACK(0x0000), PACK(0x1C20), PACK(0x3840), PACK(0x2460),
331         PACK(0x7080), PACK(0x6CA0), PACK(0x48C0), PACK(0x54E0),
332         PACK(0xE100), PACK(0xFD20), PACK(0xD940), PACK(0xC560),
333         PACK(0x9180), PACK(0x8DA0), PACK(0xA9C0), PACK(0xB5E0) };

335 static void gcm_gmult_4bit(u64 Xi[2], const u128 Htable[16])
336 {
337         u128 Z;
338         int cnt = 15;
339         size_t rem, nlo, nhi;
340         const union { long one; char little; } is_endian = {1};

342         nlo  = ((const u8 *)Xi)[15];
343         nhi  = nlo>>4;
344         nlo &= 0xf;

346         Z.hi = Htable[nlo].hi;
347         Z.lo = Htable[nlo].lo;

349         while (1) {
350                 rem  = (size_t)Z.lo&0xf;
351                 Z.lo = (Z.hi<<60)|(Z.lo>>4);
352                 Z.hi = (Z.hi>>4);
353                 if (sizeof(size_t)==8)
354                         Z.hi ^= rem_4bit[rem];
355                 else
356                         Z.hi ^= (u64)rem_4bit[rem]<<32;

358                 Z.hi ^= Htable[nhi].hi;
359                 Z.lo ^= Htable[nhi].lo;

361                 if (--cnt<0)            break;

363                 nlo  = ((const u8 *)Xi)[cnt];
364                 nhi  = nlo>>4;
365                 nlo &= 0xf;

367                 rem  = (size_t)Z.lo&0xf;
368                 Z.lo = (Z.hi<<60)|(Z.lo>>4);
369                 Z.hi = (Z.hi>>4);
370                 if (sizeof(size_t)==8)
371                         Z.hi ^= rem_4bit[rem];
372                 else
373                         Z.hi ^= (u64)rem_4bit[rem]<<32;

375                 Z.hi ^= Htable[nlo].hi;
376                 Z.lo ^= Htable[nlo].lo;
377         }

379         if (is_endian.little) {
380 #ifdef BSWAP8
381                 Xi[0] = BSWAP8(Z.hi);
382                 Xi[1] = BSWAP8(Z.lo);
383 #else
384                 u8 *p = (u8 *)Xi;
385                 u32 v;
386                 v = (u32)(Z.hi>>32);    PUTU32(p,v);
387                 v = (u32)(Z.hi);        PUTU32(p+4,v);
388                 v = (u32)(Z.lo>>32);    PUTU32(p+8,v);
389                 v = (u32)(Z.lo);        PUTU32(p+12,v);
390 #endif
391         }
```

```
392         else {
393                 Xi[0] = Z.hi;
394                 Xi[1] = Z.lo;
395         }
396 }

398 #if !defined(OPENSSL_SMALL_FOOTPRINT)
399 /*
400  * Streamed gcm_mult_4bit, see CRYPTO_gcm128_[en|de]crypt for
401  * details... Compiler-generated code doesn't seem to give any
402  * performance improvement, at least not on x86[_64]. It's here
403  * mostly as reference and a placeholder for possible future
404  * non-trivial optimization[s]...
405  */
406 static void gcm_ghash_4bit(u64 Xi[2],const u128 Htable[16],
407                                 const u8 *inp,size_t len)
408 {
409     u128 Z;
410     int cnt;
411     size_t rem, nlo, nhi;
412     const union { long one; char little; } is_endian = {1};

414 #if 1
415     do {
416         cnt  = 15;
417         nlo  = ((const u8 *)Xi)[15];
418         nlo ^= inp[15];
419         nhi  = nlo>>4;
420         nlo &= 0xf;

422         Z.hi = Htable[nlo].hi;
423         Z.lo = Htable[nlo].lo;

425         while (1) {
426                 rem  = (size_t)Z.lo&0xf;
427                 Z.lo = (Z.hi<<60)|(Z.lo>>4);
428                 Z.hi = (Z.hi>>4);
429                 if (sizeof(size_t)==8)
430                         Z.hi ^= rem_4bit[rem];
431                 else
432                         Z.hi ^= (u64)rem_4bit[rem]<<32;

434                 Z.hi ^= Htable[nhi].hi;
435                 Z.lo ^= Htable[nhi].lo;

437                 if (--cnt<0)            break;

439                 nlo  = ((const u8 *)Xi)[cnt];
440                 nlo ^= inp[cnt];
441                 nhi  = nlo>>4;
442                 nlo &= 0xf;

444                 rem  = (size_t)Z.lo&0xf;
445                 Z.lo = (Z.hi<<60)|(Z.lo>>4);
446                 Z.hi = (Z.hi>>4);
447                 if (sizeof(size_t)==8)
448                         Z.hi ^= rem_4bit[rem];
449                 else
450                         Z.hi ^= (u64)rem_4bit[rem]<<32;

452                 Z.hi ^= Htable[nlo].hi;
453                 Z.lo ^= Htable[nlo].lo;
454         }
455 #else
456     /*
457      * Extra 256+16 bytes per-key plus 512 bytes shared tables
```



```
458      * [should] give ~50% improvement... One could have PACK()-ed
459      * the rem_8bit even here, but the priority is to minimize
460      * cache footprint...
461      */
462     u128 Hshr4[16];     /* Htable shifted right by 4 bits */
463     u8   Hshl4[16];     /* Htable shifted left  by 4 bits */
464     static const unsigned short rem_8bit[256] = {
465         0x0000, 0x01C2, 0x0384, 0x0246, 0x0708, 0x06CA, 0x048C, 0x054E,
466         0x0E10, 0x0FD2, 0x0D94, 0x0C56, 0x0918, 0x08DA, 0x0A9C, 0x0B5E,
467         0x1C20, 0x1DE2, 0x1FA4, 0x1E66, 0x1B28, 0x1AEA, 0x18AC, 0x196E,
468         0x1230, 0x13F2, 0x11B4, 0x1076, 0x1538, 0x14FA, 0x16BC, 0x177E,
469         0x3840, 0x3982, 0x3BC4, 0x3A06, 0x3F48, 0x3E8A, 0x3CCC, 0x3D0E,
470         0x3650, 0x3792, 0x35D4, 0x3416, 0x3158, 0x309A, 0x32DC, 0x331E,
471         0x2460, 0x25A2, 0x27E4, 0x2626, 0x2368, 0x22AA, 0x20EC, 0x212E,
472         0x2A70, 0x2BB2, 0x29F4, 0x2836, 0x2D78, 0x2CBA, 0x2EFC, 0x2F3E,
473         0x7080, 0x7142, 0x7304, 0x72C6, 0x7788, 0x764A, 0x740C, 0x75CE,
474         0x7E90, 0x7F52, 0x7D14, 0x7CD6, 0x7998, 0x785A, 0x7A1C, 0x7BDE,
475         0x6CA0, 0x6D62, 0x6F24, 0x6EE6, 0x6BA8, 0x6A6A, 0x682C, 0x69EE,
476         0x62B0, 0x6372, 0x6134, 0x60F6, 0x65B8, 0x647A, 0x663C, 0x67FE,
477         0x48C0, 0x4902, 0x4B44, 0x4A86, 0x4FC8, 0x4E0A, 0x4C4C, 0x4D8E,
478         0x46D0, 0x4712, 0x4554, 0x4496, 0x41D8, 0x401A, 0x425C, 0x439E,
479         0x54E0, 0x5522, 0x5764, 0x56A6, 0x53E8, 0x522A, 0x506C, 0x51AE,
480         0x5AF0, 0x5B32, 0x5974, 0x58B6, 0x5DF8, 0x5C3A, 0x5E7C, 0x5FBE,
481         0xE100, 0xE0C2, 0xE284, 0xE346, 0xE608, 0xE7CA, 0xE58C, 0xE44E,
482         0xEF10, 0xEED2, 0xEC94, 0xED56, 0xE818, 0xE9DA, 0xEB9C, 0xEA5E,
483         0xFD20, 0xFCE2, 0xFEA4, 0xFF66, 0xFA28, 0xFBEA, 0xF9AC, 0xF86E,
484         0xF330, 0xF2F2, 0xF0B4, 0xF176, 0xF438, 0xF5FA, 0xF7BC, 0xF67E,
485         0xD940, 0xD882, 0xDAC4, 0xDB06, 0xDE48, 0xDF8A, 0xDDCC, 0xDC0E,
486         0xD750, 0xD692, 0xD4D4, 0xD516, 0xD058, 0xD19A, 0xD3DC, 0xD21E,
487         0xC560, 0xC4A2, 0xC6E4, 0xC726, 0xC268, 0xC3AA, 0xC1EC, 0xC02E,
488         0xCB70, 0xCAB2, 0xC8F4, 0xC936, 0xCC78, 0xCDBA, 0xCFFC, 0xCE3E,
489         0x9180, 0x9042, 0x9204, 0x93C6, 0x9688, 0x974A, 0x950C, 0x94CE,
490         0x9F90, 0x9E52, 0x9C14, 0x9DD6, 0x9898, 0x995A, 0x9B1C, 0x9ADE,
491         0x8DA0, 0x8C62, 0x8E24, 0x8FE6, 0x8AA8, 0x8B6A, 0x892C, 0x88EE,
492         0x83B0, 0x8272, 0x8034, 0x81F6, 0x84B8, 0x857A, 0x873C, 0x86FE,
493         0xA9C0, 0xA802, 0xAA44, 0xAB86, 0xAEC8, 0xAF0A, 0xAD4C, 0xAC8E,
494         0xA7D0, 0xA612, 0xA454, 0xA596, 0xA0D8, 0xA11A, 0xA35C, 0xA29E,
495         0xB5E0, 0xB422, 0xB664, 0xB7A6, 0xB2E8, 0xB32A, 0xB16C, 0xB0AE,
496         0xBBF0, 0xBA32, 0xB874, 0xB9B6, 0xBCF8, 0xBD3A, 0xBF7C, 0xBEBE };
497     /*
498      * This pre-processing phase slows down procedure by approximately
499      * same time as it makes each loop spin faster. In other words
500      * single block performance is approximately same as straightforward
501      * "4-bit" implementation, and then it goes only faster...
502      */
503     for (cnt=0; cnt<16; ++cnt) {
504         Z.hi = Htable[cnt].hi;
505         Z.lo = Htable[cnt].lo;
506         Hshr4[cnt].lo = (Z.hi<<60)|(Z.lo>>4);
507         Hshr4[cnt].hi = (Z.hi>>4);
508         Hshl4[cnt]    = (u8)(Z.lo<<4);
509     }

511     do {
512         for (Z.lo=0, Z.hi=0, cnt=15; cnt; --cnt) {
513                 nlo  = ((const u8 *)Xi)[cnt];
514                 nlo ^= inp[cnt];
515                 nhi  = nlo>>4;
516                 nlo &= 0xf;

518                 Z.hi ^= Htable[nlo].hi;
519                 Z.lo ^= Htable[nlo].lo;

521                 rem = (size_t)Z.lo&0xff;

523                 Z.lo = (Z.hi<<56)|(Z.lo>>8);
```

```
524                 Z.hi = (Z.hi>>8);

526                 Z.hi ^= Hshr4[nhi].hi;
527                 Z.lo ^= Hshr4[nhi].lo;
528                 Z.hi ^= (u64)rem_8bit[rem^Hshl4[nhi]]<<48;
529         }

531         nlo  = ((const u8 *)Xi)[0];
532         nlo ^= inp[0];
533         nhi  = nlo>>4;
534         nlo &= 0xf;

536         Z.hi ^= Htable[nlo].hi;
537         Z.lo ^= Htable[nlo].lo;

539         rem = (size_t)Z.lo&0xf;

541         Z.lo = (Z.hi<<60)|(Z.lo>>4);
542         Z.hi = (Z.hi>>4);

544         Z.hi ^= Htable[nhi].hi;
545         Z.lo ^= Htable[nhi].lo;
546         Z.hi ^= ((u64)rem_8bit[rem<<4])<<48;
547 #endif

549         if (is_endian.little) {
550 #ifdef BSWAP8
551                 Xi[0] = BSWAP8(Z.hi);
552                 Xi[1] = BSWAP8(Z.lo);
553 #else
554                 u8 *p = (u8 *)Xi;
555                 u32 v;
556                 v = (u32)(Z.hi>>32);    PUTU32(p,v);
557                 v = (u32)(Z.hi);        PUTU32(p+4,v);
558                 v = (u32)(Z.lo>>32);    PUTU32(p+8,v);
559                 v = (u32)(Z.lo);        PUTU32(p+12,v);
560 #endif
561         }
562         else {
563                 Xi[0] = Z.hi;
564                 Xi[1] = Z.lo;
565         }
566     } while (inp+=16, len-=16);
567 }
568 #endif
569 #else
570 void gcm_gmult_4bit(u64 Xi[2],const u128 Htable[16]);
571 void gcm_ghash_4bit(u64 Xi[2],const u128 Htable[16],const u8 *inp,size_t len);
572 #endif

574 #define GCM_MUL(ctx,Xi)   gcm_gmult_4bit(ctx->Xi.u,ctx->Htable)
575 #if defined(GHASH_ASM) || !defined(OPENSSL_SMALL_FOOTPRINT)
576 #define GHASH(ctx,in,len) gcm_ghash_4bit((ctx)->Xi.u,(ctx)->Htable,in,len)
577 /* GHASH_CHUNK is "stride parameter" missioned to mitigate cache
578  * trashing effect. In other words idea is to hash data while it's
579  * still in L1 cache after encryption pass... */
580 #define GHASH_CHUNK       (3*1024)
581 #endif

583 #else   /* TABLE_BITS */

585 static void gcm_gmult_1bit(u64 Xi[2],const u64 H[2])
586 {
587         u128 V,Z = { 0,0 };
588         long X;
589         int  i,j;
```



```
590         const long *xi = (const long *)Xi;
591         const union { long one; char little; } is_endian = {1};

593         V.hi = H[0];    /* H is in host byte order, no byte swapping */
594         V.lo = H[1];

596         for (j=0; j<16/sizeof(long); ++j) {
597                 if (is_endian.little) {
598                         if (sizeof(long)==8) {
599 #ifdef BSWAP8
600                                 X = (long)(BSWAP8(xi[j]));
601 #else
602                                 const u8 *p = (const u8 *)(xi+j);
603                                 X = (long)((u64)GETU32(p)<<32|GETU32(p+4));
604 #endif
605                         }
606                         else {
607                                 const u8 *p = (const u8 *)(xi+j);
608                                 X = (long)GETU32(p);
609                         }
610                 }
611                 else
612                         X = xi[j];

614                 for (i=0; i<8*sizeof(long); ++i, X<<=1) {
615                         u64 M = (u64)(X>>(8*sizeof(long)-1));
616                         Z.hi ^= V.hi&M;
617                         Z.lo ^= V.lo&M;

619                         REDUCE1BIT(V);
620                 }
621         }

623         if (is_endian.little) {
624 #ifdef BSWAP8
625                 Xi[0] = BSWAP8(Z.hi);
626                 Xi[1] = BSWAP8(Z.lo);
627 #else
628                 u8 *p = (u8 *)Xi;
629                 u32 v;
630                 v = (u32)(Z.hi>>32);    PUTU32(p,v);
631                 v = (u32)(Z.hi);        PUTU32(p+4,v);
632                 v = (u32)(Z.lo>>32);    PUTU32(p+8,v);
633                 v = (u32)(Z.lo);        PUTU32(p+12,v);
634 #endif
635         }
636         else {
637                 Xi[0] = Z.hi;
638                 Xi[1] = Z.lo;
639         }
640 }
641 #define GCM_MUL(ctx,Xi)   gcm_gmult_1bit(ctx->Xi.u,ctx->H.u)

643 #endif

645 #if     TABLE_BITS==4 && defined(GHASH_ASM)
646 # if    !defined(I386_ONLY) && \
647         (defined(__i386)        || defined(__i386__)    || \
648          defined(__x86_64)      || defined(__x86_64__)  || \
649          defined(_M_IX86)       || defined(_M_AMD64)    || defined(_M_X64))
650 #  define GHASH_ASM_X86_OR_64
651 #  define GCM_FUNCREF_4BIT
652 extern unsigned int OPENSSL_ia32cap_P[2];

654 void gcm_init_clmul(u128 Htable[16],const u64 Xi[2]);
655 void gcm_gmult_clmul(u64 Xi[2],const u128 Htable[16]);
```

```
656 void gcm_ghash_clmul(u64 Xi[2],const u128 Htable[16],const u8 *inp,size_t len);

658 #  if   defined(__i386) || defined(__i386__) || defined(_M_IX86)
659 #   define GHASH_ASM_X86
660 void gcm_gmult_4bit_mmx(u64 Xi[2],const u128 Htable[16]);
661 void gcm_ghash_4bit_mmx(u64 Xi[2],const u128 Htable[16],const u8 *inp,size_t len

663 void gcm_gmult_4bit_x86(u64 Xi[2],const u128 Htable[16]);
664 void gcm_ghash_4bit_x86(u64 Xi[2],const u128 Htable[16],const u8 *inp,size_t len
665 #  endif
666 # elif defined(__arm__) || defined(__arm)
667 #  include "arm_arch.h"
668 #  if __ARM_ARCH__>=7
669 #   define GHASH_ASM_ARM
670 #   define GCM_FUNCREF_4BIT
671 void gcm_gmult_neon(u64 Xi[2],const u128 Htable[16]);
672 void gcm_ghash_neon(u64 Xi[2],const u128 Htable[16],const u8 *inp,size_t len);
673 #  endif
674 # endif
675 #endif

677 #ifdef GCM_FUNCREF_4BIT
678 # undef  GCM_MUL
679 # define GCM_MUL(ctx,Xi)        (*gcm_gmult_p)(ctx->Xi.u,ctx->Htable)
680 # ifdef GHASH
681 #  undef  GHASH
682 #  define GHASH(ctx,in,len)     (*gcm_ghash_p)(ctx->Xi.u,ctx->Htable,in,len)
683 # endif
684 #endif

686 void CRYPTO_gcm128_init(GCM128_CONTEXT *ctx,void *key,block128_f block)
687 {
688         const union { long one; char little; } is_endian = {1};

690         memset(ctx,0,sizeof(*ctx));
691         ctx->block = block;
692         ctx->key   = key;

694         (*block)(ctx->H.c,ctx->H.c,key);

696         if (is_endian.little) {
697                 /* H is stored in host byte order */
698 #ifdef BSWAP8
699                 ctx->H.u[0] = BSWAP8(ctx->H.u[0]);
700                 ctx->H.u[1] = BSWAP8(ctx->H.u[1]);
701 #else
702                 u8 *p = ctx->H.c;
703                 u64 hi,lo;
704                 hi = (u64)GETU32(p)  <<32|GETU32(p+4);
705                 lo = (u64)GETU32(p+8)<<32|GETU32(p+12);
706                 ctx->H.u[0] = hi;
707                 ctx->H.u[1] = lo;
708 #endif
709         }

711 #if     TABLE_BITS==8
712         gcm_init_8bit(ctx->Htable,ctx->H.u);
713 #elif   TABLE_BITS==4
714 # if    defined(GHASH_ASM_X86_OR_64)
715 #  if   !defined(GHASH_ASM_X86) || defined(OPENSSL_IA32_SSE2)
716         if (OPENSSL_ia32cap_P[0]&(1<<24) &&     /* check FXSR bit */
717             OPENSSL_ia32cap_P[1]&(1<<1) ) {     /* check PCLMULQDQ bit */
718                 gcm_init_clmul(ctx->Htable,ctx->H.u);
719                 ctx->gmult = gcm_gmult_clmul;
720                 ctx->ghash = gcm_ghash_clmul;
721                 return;
```



```
722         }
723 #  endif
724         gcm_init_4bit(ctx->Htable,ctx->H.u);
725 #  if   defined(GHASH_ASM_X86)                  /* x86 only */
726 #   if  defined(OPENSSL_IA32_SSE2)
727         if (OPENSSL_ia32cap_P[0]&(1<<25)) {     /* check SSE bit */
728 #   else
729         if (OPENSSL_ia32cap_P[0]&(1<<23)) {     /* check MMX bit */
730 #   endif
731                 ctx->gmult = gcm_gmult_4bit_mmx;
732                 ctx->ghash = gcm_ghash_4bit_mmx;
733         } else {
734                 ctx->gmult = gcm_gmult_4bit_x86;
735                 ctx->ghash = gcm_ghash_4bit_x86;
736         }
737 #  else
738         ctx->gmult = gcm_gmult_4bit;
739         ctx->ghash = gcm_ghash_4bit;
740 #  endif
741 # elif  defined(GHASH_ASM_ARM)
742         if (OPENSSL_armcap_P & ARMV7_NEON) {
743                 ctx->gmult = gcm_gmult_neon;
744                 ctx->ghash = gcm_ghash_neon;
745         } else {
746                 gcm_init_4bit(ctx->Htable,ctx->H.u);
747                 ctx->gmult = gcm_gmult_4bit;
748                 ctx->ghash = gcm_ghash_4bit;
749         }
750 # else
751         gcm_init_4bit(ctx->Htable,ctx->H.u);
752 # endif
753 #endif
754 }

756 void CRYPTO_gcm128_setiv(GCM128_CONTEXT *ctx,const unsigned char *iv,size_t len)
757 {
758         const union { long one; char little; } is_endian = {1};
759         unsigned int ctr;
760 #ifdef GCM_FUNCREF_4BIT
761         void (*gcm_gmult_p)(u64 Xi[2],const u128 Htable[16])    = ctx->gmult;
762 #endif

764         ctx->Yi.u[0]  = 0;
765         ctx->Yi.u[1]  = 0;
766         ctx->Xi.u[0]  = 0;
767         ctx->Xi.u[1]  = 0;
768         ctx->len.u[0] = 0;      /* AAD length */
769         ctx->len.u[1] = 0;      /* message length */
770         ctx->ares = 0;
771         ctx->mres = 0;

773         if (len==12) {
774                 memcpy(ctx->Yi.c,iv,12);
775                 ctx->Yi.c[15]=1;
776                 ctr=1;
777         }
778         else {
779                 size_t i;
780                 u64 len0 = len;

782                 while (len>=16) {
783                         for (i=0; i<16; ++i) ctx->Yi.c[i] ^= iv[i];
784                         GCM_MUL(ctx,Yi);
785                         iv += 16;
786                         len -= 16;
787                 }
```

```
788                 if (len) {
789                         for (i=0; i<len; ++i) ctx->Yi.c[i] ^= iv[i];
790                         GCM_MUL(ctx,Yi);
791                 }
792                 len0 <<= 3;
793                 if (is_endian.little) {
794 #ifdef BSWAP8
795                         ctx->Yi.u[1]  ^= BSWAP8(len0);
796 #else
797                         ctx->Yi.c[8]  ^= (u8)(len0>>56);
798                         ctx->Yi.c[9]  ^= (u8)(len0>>48);
799                         ctx->Yi.c[10] ^= (u8)(len0>>40);
800                         ctx->Yi.c[11] ^= (u8)(len0>>32);
801                         ctx->Yi.c[12] ^= (u8)(len0>>24);
802                         ctx->Yi.c[13] ^= (u8)(len0>>16);
803                         ctx->Yi.c[14] ^= (u8)(len0>>8);
804                         ctx->Yi.c[15] ^= (u8)(len0);
805 #endif
806                 }
807                 else
808                         ctx->Yi.u[1]  ^= len0;

810                 GCM_MUL(ctx,Yi);

812                 if (is_endian.little)
813 #ifdef BSWAP4
814                         ctr = BSWAP4(ctx->Yi.d[3]);
815 #else
816                         ctr = GETU32(ctx->Yi.c+12);
817 #endif
818                 else
819                         ctr = ctx->Yi.d[3];
820         }

822         (*ctx->block)(ctx->Yi.c,ctx->EK0.c,ctx->key);
823         ++ctr;
824         if (is_endian.little)
825 #ifdef BSWAP4
826                 ctx->Yi.d[3] = BSWAP4(ctr);
827 #else
828                 PUTU32(ctx->Yi.c+12,ctr);
829 #endif
830         else
831                 ctx->Yi.d[3] = ctr;
832 }

834 int CRYPTO_gcm128_aad(GCM128_CONTEXT *ctx,const unsigned char *aad,size_t len)
835 {
836         size_t i;
837         unsigned int n;
838         u64 alen = ctx->len.u[0];
839 #ifdef GCM_FUNCREF_4BIT
840         void (*gcm_gmult_p)(u64 Xi[2],const u128 Htable[16])    = ctx->gmult;
841 # ifdef GHASH
842         void (*gcm_ghash_p)(u64 Xi[2],const u128 Htable[16],
843                             const u8 *inp,size_t len)       = ctx->ghash;
844 # endif
845 #endif

847         if (ctx->len.u[1]) return -2;

849         alen += len;
850         if (alen>(U64(1)<<61) || (sizeof(len)==8 && alen<len))
851                 return -1;
852         ctx->len.u[0] = alen;
```



```
854         n = ctx->ares;
855         if (n) {
856                 while (n && len) {
857                         ctx->Xi.c[n] ^= *(aad++);
858                         --len;
859                         n = (n+1)%16;
860                 }
861                 if (n==0) GCM_MUL(ctx,Xi);
862                 else {
863                         ctx->ares = n;
864                         return 0;
865                 }
866         }

868 #ifdef GHASH
869         if ((i = (len&(size_t)-16))) {
870                 GHASH(ctx,aad,i);
871                 aad += i;
872                 len -= i;
873         }
874 #else
875         while (len>=16) {
876                 for (i=0; i<16; ++i) ctx->Xi.c[i] ^= aad[i];
877                 GCM_MUL(ctx,Xi);
878                 aad += 16;
879                 len -= 16;
880         }
881 #endif
882         if (len) {
883                 n = (unsigned int)len;
884                 for (i=0; i<len; ++i) ctx->Xi.c[i] ^= aad[i];
885         }

887         ctx->ares = n;
888         return 0;
889 }

891 int CRYPTO_gcm128_encrypt(GCM128_CONTEXT *ctx,
892                 const unsigned char *in, unsigned char *out,
893                 size_t len)
894 {
895         const union { long one; char little; } is_endian = {1};
896         unsigned int n, ctr;
897         size_t i;
898         u64       mlen  = ctx->len.u[1];
899         block128_f block = ctx->block;
900         void      *key   = ctx->key;
901 #ifdef GCM_FUNCREF_4BIT
902         void (*gcm_gmult_p)(u64 Xi[2],const u128 Htable[16])    = ctx->gmult;
903 # ifdef GHASH
904         void (*gcm_ghash_p)(u64 Xi[2],const u128 Htable[16],
905                             const u8 *inp,size_t len)       = ctx->ghash;
906 # endif
907 #endif

909 #if 0
910         n = (unsigned int)mlen%16; /* alternative to ctx->mres */
911 #endif
912         mlen += len;
913         if (mlen>((U64(1)<<36)-32) || (sizeof(len)==8 && mlen<len))
914                 return -1;
915         ctx->len.u[1] = mlen;

917         if (ctx->ares) {
918                 /* First call to encrypt finalizes GHASH(AAD) */
919                 GCM_MUL(ctx,Xi);
```

```
920                 ctx->ares = 0;
921         }

923         if (is_endian.little)
924 #ifdef BSWAP4
925                 ctr = BSWAP4(ctx->Yi.d[3]);
926 #else
927                 ctr = GETU32(ctx->Yi.c+12);
928 #endif
929         else
930                 ctr = ctx->Yi.d[3];

932         n = ctx->mres;
933 #if !defined(OPENSSL_SMALL_FOOTPRINT)
934         if (16%sizeof(size_t) == 0) do {        /* always true actually */
935                 if (n) {
936                         while (n && len) {
937                                 ctx->Xi.c[n] ^= *(out++) = *(in++)^ctx->EKi.c[n];
938                                 --len;
939                                 n = (n+1)%16;
940                         }
941                         if (n==0) GCM_MUL(ctx,Xi);
942                         else {
943                                 ctx->mres = n;
944                                 return 0;
945                         }
946                 }
947 #if defined(STRICT_ALIGNMENT)
948                 if (((size_t)in|(size_t)out)%sizeof(size_t) != 0)
949                         break;
950 #endif
951 #if defined(GHASH) && defined(GHASH_CHUNK)
952                 while (len>=GHASH_CHUNK) {
953                     size_t j=GHASH_CHUNK;

955                     while (j) {
956                         size_t *out_t=(size_t *)out;
957                         const size_t *in_t=(const size_t *)in;

959                         (*block)(ctx->Yi.c,ctx->EKi.c,key);
960                         ++ctr;
961                         if (is_endian.little)
962 #ifdef BSWAP4
963                                 ctx->Yi.d[3] = BSWAP4(ctr);
964 #else
965                                 PUTU32(ctx->Yi.c+12,ctr);
966 #endif
967                         else
968                                 ctx->Yi.d[3] = ctr;
969                         for (i=0; i<16/sizeof(size_t); ++i)
970                                 out_t[i] = in_t[i] ^ ctx->EKi.t[i];
971                         out += 16;
972                         in  += 16;
973                         j   -= 16;
974                     }
975                     GHASH(ctx,out-GHASH_CHUNK,GHASH_CHUNK);
976                     len -= GHASH_CHUNK;
977                 }
978                 if ((i = (len&(size_t)-16))) {
979                     size_t j=i;

981                     while (len>=16) {
982                         size_t *out_t=(size_t *)out;
983                         const size_t *in_t=(const size_t *)in;

985                         (*block)(ctx->Yi.c,ctx->EKi.c,key);
```



```
986                         ++ctr;
987                         if (is_endian.little)
988 #ifdef BSWAP4
989                                 ctx->Yi.d[3] = BSWAP4(ctr);
990 #else
991                                 PUTU32(ctx->Yi.c+12,ctr);
992 #endif
993                         else
994                                 ctx->Yi.d[3] = ctr;
995                         for (i=0; i<16/sizeof(size_t); ++i)
996                                 out_t[i] = in_t[i] ^ ctx->EKi.t[i];
997                         out += 16;
998                         in  += 16;
999                         len -= 16;
1000                    }
1001                    GHASH(ctx,out-j,j);
1002                }
1003 #else
1004                 while (len>=16) {
1005                         size_t *out_t=(size_t *)out;
1006                         const size_t *in_t=(const size_t *)in;

1008                         (*block)(ctx->Yi.c,ctx->EKi.c,key);
1009                         ++ctr;
1010                         if (is_endian.little)
1011 #ifdef BSWAP4
1012                                 ctx->Yi.d[3] = BSWAP4(ctr);
1013 #else
1014                                 PUTU32(ctx->Yi.c+12,ctr);
1015 #endif
1016                         else
1017                                 ctx->Yi.d[3] = ctr;
1018                         for (i=0; i<16/sizeof(size_t); ++i)
1019                                 ctx->Xi.t[i] ^=
1020                                 out_t[i] = in_t[i]^ctx->EKi.t[i];
1021                         GCM_MUL(ctx,Xi);
1022                         out += 16;
1023                         in  += 16;
1024                         len -= 16;
1025                 }
1026 #endif
1027                 if (len) {
1028                         (*block)(ctx->Yi.c,ctx->EKi.c,key);
1029                         ++ctr;
1030                         if (is_endian.little)
1031 #ifdef BSWAP4
1032                                 ctx->Yi.d[3] = BSWAP4(ctr);
1033 #else
1034                                 PUTU32(ctx->Yi.c+12,ctr);
1035 #endif
1036                         else
1037                                 ctx->Yi.d[3] = ctr;
1038                         while (len--) {
1039                                 ctx->Xi.c[n] ^= out[n] = in[n]^ctx->EKi.c[n];
1040                                 ++n;
1041                         }
1042                 }

1044                 ctx->mres = n;
1045                 return 0;
1046         } while(0);
1047 #endif
1048         for (i=0;i<len;++i) {
1049                 if (n==0) {
1050                         (*block)(ctx->Yi.c,ctx->EKi.c,key);
1051                         ++ctr;
```

```
1052                         if (is_endian.little)
1053 #ifdef BSWAP4
1054                                 ctx->Yi.d[3] = BSWAP4(ctr);
1055 #else
1056                                 PUTU32(ctx->Yi.c+12,ctr);
1057 #endif
1058                         else
1059                                 ctx->Yi.d[3] = ctr;
1060                 }
1061                 ctx->Xi.c[n] ^= out[i] = in[i]^ctx->EKi.c[n];
1062                 n = (n+1)%16;
1063                 if (n==0)
1064                         GCM_MUL(ctx,Xi);
1065         }

1067         ctx->mres = n;
1068         return 0;
1069 }

1071 int CRYPTO_gcm128_decrypt(GCM128_CONTEXT *ctx,
1072                 const unsigned char *in, unsigned char *out,
1073                 size_t len)
1074 {
1075         const union { long one; char little; } is_endian = {1};
1076         unsigned int n, ctr;
1077         size_t i;
1078         u64        mlen  = ctx->len.u[1];
1079         block128_f block = ctx->block;
1080         void      *key   = ctx->key;
1081 #ifdef GCM_FUNCREF_4BIT
1082         void (*gcm_gmult_p)(u64 Xi[2],const u128 Htable[16])    = ctx->gmult;
1083 # ifdef GHASH
1084         void (*gcm_ghash_p)(u64 Xi[2],const u128 Htable[16],
1085                             const u8 *inp,size_t len)       = ctx->ghash;
1086 # endif
1087 #endif

1089         mlen += len;
1090         if (mlen>((U64(1)<<36)-32) || (sizeof(len)==8 && mlen<len))
1091                 return -1;
1092         ctx->len.u[1] = mlen;

1094         if (ctx->ares) {
1095                 /* First call to decrypt finalizes GHASH(AAD) */
1096                 GCM_MUL(ctx,Xi);
1097                 ctx->ares = 0;
1098         }

1100         if (is_endian.little)
1101 #ifdef BSWAP4
1102                 ctr = BSWAP4(ctx->Yi.d[3]);
1103 #else
1104                 ctr = GETU32(ctx->Yi.c+12);
1105 #endif
1106         else
1107                 ctr = ctx->Yi.d[3];

1109         n = ctx->mres;
1110 #if !defined(OPENSSL_SMALL_FOOTPRINT)
1111         if (16%sizeof(size_t) == 0) do {        /* always true actually */
1112                 if (n) {
1113                         while (n && len) {
1114                                 u8 c = *(in++);
1115                                 *(out++) = c^ctx->EKi.c[n];
1116                                 ctx->Xi.c[n] ^= c;
1117                                 --len;
```



```
1118                                 n = (n+1)%16;
1119                         }
1120                         if (n==0) GCM_MUL (ctx,Xi);
1121                         else {
1122                                 ctx->mres = n;
1123                                 return 0;
1124                         }
1125                 }
1126 #if defined(STRICT_ALIGNMENT)
1127                 if (((size_t)in|(size_t)out)%sizeof(size_t) != 0)
1128                         break;
1129 #endif
1130 #if defined(GHASH) && defined(GHASH_CHUNK)
1131                 while (len>=GHASH_CHUNK) {
1132                     size_t j=GHASH_CHUNK;

1134                     GHASH(ctx,in,GHASH_CHUNK);
1135                     while (j) {
1136                         size_t *out_t=(size_t *)out;
1137                         const size_t *in_t=(const size_t *)in;

1139                         (*block)(ctx->Yi.c,ctx->EKi.c,key);
1140                         ++ctr;
1141                         if (is_endian.little)
1142 #ifdef BSWAP4
1143                                 ctx->Yi.d[3] = BSWAP4(ctr);
1144 #else
1145                                 PUTU32(ctx->Yi.c+12,ctr);
1146 #endif
1147                         else
1148                                 ctx->Yi.d[3] = ctr;
1149                         for (i=0; i<16/sizeof(size_t); ++i)
1150                                 out_t[i] = in_t[i]^ctx->EKi.t[i];
1151                         out += 16;
1152                         in  += 16;
1153                         j   -= 16;
1154                     }
1155                     len -= GHASH_CHUNK;
1156                 }
1157                 if ((i = (len&(size_t)-16))) {
1158                     GHASH(ctx,in,i);
1159                     while (len>=16) {
1160                         size_t *out_t=(size_t *)out;
1161                         const size_t *in_t=(const size_t *)in;

1163                         (*block)(ctx->Yi.c,ctx->EKi.c,key);
1164                         ++ctr;
1165                         if (is_endian.little)
1166 #ifdef BSWAP4
1167                                 ctx->Yi.d[3] = BSWAP4(ctr);
1168 #else
1169                                 PUTU32(ctx->Yi.c+12,ctr);
1170 #endif
1171                         else
1172                                 ctx->Yi.d[3] = ctr;
1173                         for (i=0; i<16/sizeof(size_t); ++i)
1174                                 out_t[i] = in_t[i]^ctx->EKi.t[i];
1175                         out += 16;
1176                         in  += 16;
1177                         len -= 16;
1178                     }
1179                 }
1180 #else
1181                 while (len>=16) {
1182                         size_t *out_t=(size_t *)out;
1183                         const size_t *in_t=(const size_t *)in;
```

```
1185                        (*block)(ctx->Yi.c,ctx->EKi.c,key);
1186                        ++ctr;
1187                        if (is_endian.little)
1188 #ifdef BSWAP4
1189                                ctx->Yi.d[3] = BSWAP4(ctr);
1190 #else
1191                                PUTU32(ctx->Yi.c+12,ctr);
1192 #endif
1193                        else
1194                                ctx->Yi.d[3] = ctr;
1195                        for (i=0; i<16/sizeof(size_t); ++i) {
1196                                size_t c = in[i];
1197                                out[i] = c^ctx->EKi.t[i];
1198                                ctx->Xi.t[i] ^= c;
1199                        }
1200                        GCM_MUL(ctx,Xi);
1201                        out += 16;
1202                        in  += 16;
1203                        len -= 16;
1204                }
1205 #endif
1206                if (len) {
1207                        (*block)(ctx->Yi.c,ctx->EKi.c,key);
1208                        ++ctr;
1209                        if (is_endian.little)
1210 #ifdef BSWAP4
1211                                ctx->Yi.d[3] = BSWAP4(ctr);
1212 #else
1213                                PUTU32(ctx->Yi.c+12,ctr);
1214 #endif
1215                        else
1216                                ctx->Yi.d[3] = ctr;
1217                        while (len--) {
1218                                u8 c = in[n];
1219                                ctx->Xi.c[n] ^= c;
1220                                out[n] = c^ctx->EKi.c[n];
1221                                ++n;
1222                        }
1223                }

1225                ctx->mres = n;
1226                return 0;
1227        } while(0);
1228 #endif
1229        for (i=0;i<len;++i) {
1230                u8 c;
1231                if (n==0) {
1232                        (*block)(ctx->Yi.c,ctx->EKi.c,key);
1233                        ++ctr;
1234                        if (is_endian.little)
1235 #ifdef BSWAP4
1236                                ctx->Yi.d[3] = BSWAP4(ctr);
1237 #else
1238                                PUTU32(ctx->Yi.c+12,ctr);
1239 #endif
1240                        else
1241                                ctx->Yi.d[3] = ctr;
1242                }
1243                c = in[i];
1244                out[i] = c^ctx->EKi.c[n];
1245                ctx->Xi.c[n] ^= c;
1246                n = (n+1)%16;
1247                if (n==0)
1248                        GCM_MUL(ctx,Xi);
1249        }
```



```
1251        ctx->mres = n;
1252        return 0;
1253 }

1255 int CRYPTO_gcm128_encrypt_ctr32(GCM128_CONTEXT *ctx,
1256                const unsigned char *in, unsigned char *out,
1257                size_t len, ctr128_f stream)
1258 {
1259        const union { long one; char little; } is_endian = {1};
1260        unsigned int n, ctr;
1261        size_t i;
1262        u64   mlen = ctx->len.u[1];
1263        void *key  = ctx->key;
1264 #ifdef GCM_FUNCREF_4BIT
1265        void (*gcm_gmult_p)(u64 Xi[2],const u128 Htable[16])    = ctx->gmult;
1266 # ifdef GHASH
1267        void (*gcm_ghash_p)(u64 Xi[2],const u128 Htable[16],
1268                            const u8 *inp,size_t len)       = ctx->ghash;
1269 # endif
1270 #endif

1272        mlen += len;
1273        if (mlen>((U64(1)<<36)-32) || (sizeof(len)==8 && mlen<len))
1274                return -1;
1275        ctx->len.u[1] = mlen;

1277        if (ctx->ares) {
1278                /* First call to encrypt finalizes GHASH(AAD) */
1279                GCM_MUL(ctx,Xi);
1280                ctx->ares = 0;
1281        }

1283        if (is_endian.little)
1284 #ifdef BSWAP4
1285                ctr = BSWAP4(ctx->Yi.d[3]);
1286 #else
1287                ctr = GETU32(ctx->Yi.c+12);
1288 #endif
1289        else
1290                ctr = ctx->Yi.d[3];

1292        n = ctx->mres;
1293        if (n) {
1294                while (n && len) {
1295                        ctx->Xi.c[n] ^= *(out++) = *(in++)^ctx->EKi.c[n];
1296                        --len;
1297                        n = (n+1)%16;
1298                }
1299                if (n==0) GCM_MUL(ctx,Xi);
1300                else {
1301                        ctx->mres = n;
1302                        return 0;
1303                }
1304        }
1305 #if defined(GHASH) && !defined(OPENSSL_SMALL_FOOTPRINT)
1306        while (len>=GHASH_CHUNK) {
1307                (*stream)(in,out,GHASH_CHUNK/16,key,ctx->Yi.c);
1308                ctr += GHASH_CHUNK/16;
1309                if (is_endian.little)
1310 #ifdef BSWAP4
1311                        ctx->Yi.d[3] = BSWAP4(ctr);
1312 #else
1313                        PUTU32(ctx->Yi.c+12,ctr);
1314 #endif
1315                else
```

```
1316                        ctx->Yi.d[3] = ctr;
1317                GHASH(ctx,out,GHASH_CHUNK);
1318                out += GHASH_CHUNK;
1319                in  += GHASH_CHUNK;
1320                len -= GHASH_CHUNK;
1321        }
1322 #endif
1323        if ((i = (len&(size_t)-16))) {
1324                size_t j=i/16;

1326                (*stream)(in,out,j,key,ctx->Yi.c);
1327                ctr += (unsigned int)j;
1328                if (is_endian.little)
1329 #ifdef BSWAP4
1330                        ctx->Yi.d[3] = BSWAP4(ctr);
1331 #else
1332                        PUTU32(ctx->Yi.c+12,ctr);
1333 #endif
1334                else
1335                        ctx->Yi.d[3] = ctr;
1336                in  += i;
1337                len -= i;
1338 #if defined(GHASH)
1339                GHASH(ctx,out,i);
1340                out += i;
1341 #else
1342                while (j--) {
1343                        for (i=0;i<16;++i) ctx->Xi.c[i] ^= out[i];
1344                        GCM_MUL(ctx,Xi);
1345                        out += 16;
1346                }
1347 #endif
1348        }
1349        if (len) {
1350                (*ctx->block)(ctx->Yi.c,ctx->EKi.c,key);
1351                ++ctr;
1352                if (is_endian.little)
1353 #ifdef BSWAP4
1354                        ctx->Yi.d[3] = BSWAP4(ctr);
1355 #else
1356                        PUTU32(ctx->Yi.c+12,ctr);
1357 #endif
1358                else
1359                        ctx->Yi.d[3] = ctr;
1360                while (len--) {
1361                        ctx->Xi.c[n] ^= out[n] = in[n]^ctx->EKi.c[n];
1362                        ++n;
1363                }
1364        }

1366        ctx->mres = n;
1367        return 0;
1368 }

1370 int CRYPTO_gcm128_decrypt_ctr32(GCM128_CONTEXT *ctx,
1371                const unsigned char *in, unsigned char *out,
1372                size_t len,ctr128_f stream)
1373 {
1374        const union { long one; char little; } is_endian = {1};
1375        unsigned int n, ctr;
1376        size_t i;
1377        u64   mlen = ctx->len.u[1];
1378        void *key  = ctx->key;
1379 #ifdef GCM_FUNCREF_4BIT
1380        void (*gcm_gmult_p)(u64 Xi[2],const u128 Htable[16])    = ctx->gmult;
1381 # ifdef GHASH
```



```
1382        void (*gcm_ghash_p)(u64 Xi[2],const u128 Htable[16],
1383                            const u8 *inp,size_t len)       = ctx->ghash;
1384 # endif
1385 #endif

1387        mlen += len;
1388        if (mlen>((U64(1)<<36)-32) || (sizeof(len)==8 && mlen<len))
1389                return -1;
1390        ctx->len.u[1] = mlen;

1392        if (ctx->ares) {
1393                /* First call to decrypt finalizes GHASH(AAD) */
1394                GCM_MUL(ctx,Xi);
1395                ctx->ares = 0;
1396        }

1398        if (is_endian.little)
1399 #ifdef BSWAP4
1400                ctr = BSWAP4(ctx->Yi.d[3]);
1401 #else
1402                ctr = GETU32(ctx->Yi.c+12);
1403 #endif
1404        else
1405                ctr = ctx->Yi.d[3];

1407        n = ctx->mres;
1408        if (n) {
1409                while (n && len) {
1410                        u8 c = *(in++);
1411                        *(out++) = c^ctx->EKi.c[n];
1412                        ctx->Xi.c[n] ^= c;
1413                        --len;
1414                        n = (n+1)%16;
1415                }
1416                if (n==0) GCM_MUL (ctx,Xi);
1417                else {
1418                        ctx->mres = n;
1419                        return 0;
1420                }
1421        }
1422 #if defined(GHASH) && !defined(OPENSSL_SMALL_FOOTPRINT)
1423        while (len>=GHASH_CHUNK) {
1424                GHASH(ctx,in,GHASH_CHUNK);
1425                (*stream)(in,out,GHASH_CHUNK/16,key,ctx->Yi.c);
1426                ctr += GHASH_CHUNK/16;
1427                if (is_endian.little)
1428 #ifdef BSWAP4
1429                        ctx->Yi.d[3] = BSWAP4(ctr);
1430 #else
1431                        PUTU32(ctx->Yi.c+12,ctr);
1432 #endif
1433                else
1434                        ctx->Yi.d[3] = ctr;
1435                out += GHASH_CHUNK;
1436                in  += GHASH_CHUNK;
1437                len -= GHASH_CHUNK;
1438        }
1439 #endif
1440        if ((i = (len&(size_t)-16))) {
1441                size_t j=i/16;

1443 #if defined(GHASH)
1444                GHASH(ctx,in,i);
1445 #else
1446                while (j--) {
1447                        size_t k;
```

```
1448                        for (k=0;k<16;++k) ctx->Xi.c[k] ^= in[k];
1449                        GCM_MUL(ctx,Xi);
1450                        in += 16;
1451                }
1452                j   = i/16;
1453                in -= i;
1454 #endif
1455                (*stream)(in,out,j,key,ctx->Yi.c);
1456                ctr += (unsigned int)j;
1457                if (is_endian.little)
1458 #ifdef BSWAP4
1459                        ctx->Yi.d[3] = BSWAP4(ctr);
1460 #else
1461                        PUTU32(ctx->Yi.c+12,ctr);
1462 #endif
1463                else
1464                        ctx->Yi.d[3] = ctr;
1465                out += i;
1466                in  += i;
1467                len -= i;
1468        }
1469        if (len) {
1470                (*ctx->block)(ctx->Yi.c,ctx->EKi.c,key);
1471                ++ctr;
1472                if (is_endian.little)
1473 #ifdef BSWAP4
1474                        ctx->Yi.d[3] = BSWAP4(ctr);
1475 #else
1476                        PUTU32(ctx->Yi.c+12,ctr);
1477 #endif
1478                else
1479                        ctx->Yi.d[3] = ctr;
1480                while (len--) {
1481                        u8 c = in[n];
1482                        ctx->Xi.c[n] ^= c;
1483                        out[n] = c^ctx->EKi.c[n];
1484                        ++n;
1485                }
1486        }

1488        ctx->mres = n;
1489        return 0;
1490 }

1492 int CRYPTO_gcm128_finish(GCM128_CONTEXT *ctx,const unsigned char *tag,
1493                        size_t len)
1494 {
1495        const union { long one; char little; } is_endian = {1};
1496        u64 alen = ctx->len.u[0]<<3;
1497        u64 clen = ctx->len.u[1]<<3;
1498 #ifdef GCM_FUNCREF_4BIT
1499        void (*gcm_gmult_p)(u64 Xi[2],const u128 Htable[16])    = ctx->gmult;
1500 #endif

1502        if (ctx->mres || ctx->ares)
1503                GCM_MUL(ctx,Xi);

1505        if (is_endian.little) {
1506 #ifdef BSWAP8
1507                alen = BSWAP8(alen);
1508                clen = BSWAP8(clen);
1509 #else
1510                u8 *p = ctx->len.c;

1512                ctx->len.u[0] = alen;
1513                ctx->len.u[1] = clen;
```



```
1515                alen = (u64)GETU32(p)  <<32|GETU32(p+4);
1516                clen = (u64)GETU32(p+8)<<32|GETU32(p+12);
1517 #endif
1518        }

1520        ctx->Xi.u[0] ^= alen;
1521        ctx->Xi.u[1] ^= clen;
1522        GCM_MUL(ctx,Xi);

1524        ctx->Xi.u[0] ^= ctx->EK0.u[0];
1525        ctx->Xi.u[1] ^= ctx->EK0.u[1];

1527        if (tag && len<=sizeof(ctx->Xi))
1528                return memcmp(ctx->Xi.c,tag,len);
1529        else
1530                return -1;
1531 }

1533 void CRYPTO_gcm128_tag(GCM128_CONTEXT *ctx, unsigned char *tag, size_t len)
1534 {
1535        CRYPTO_gcm128_finish(ctx, NULL, 0);
1536        memcpy(tag, ctx->Xi.c, len<=sizeof(ctx->Xi.c)?len:sizeof(ctx->Xi.c));
1537 }

1539 GCM128_CONTEXT *CRYPTO_gcm128_new(void *key, block128_f block)
1540 {
1541        GCM128_CONTEXT *ret;

1543        if ((ret = (GCM128_CONTEXT *)OPENSSL_malloc(sizeof(GCM128_CONTEXT))))
1544                CRYPTO_gcm128_init(ret,key,block);

1546        return ret;
1547 }

1549 void CRYPTO_gcm128_release(GCM128_CONTEXT *ctx)
1550 {
1551        if (ctx) {
1552                OPENSSL_cleanse(ctx,sizeof(*ctx));
1553                OPENSSL_free(ctx);
1554        }
1555 }

1557 #if defined(SELFTEST)
1558 #include <stdio.h>
1559 #include <openssl/aes.h>

1561 /* Test Case 1 */
1562 static const u8 K1[16],
1563                *P1=NULL,
1564                *A1=NULL,
1565                IV1[12],
1566                *C1=NULL,
1567                T1[]=  {0x58,0xe2,0xfc,0xce,0xfa,0x7e,0x30,0x61,0x36,0x7f,0x1d,0

1569 /* Test Case 2 */
1570 #define K2 K1
1571 #define A2 A1
1572 #define IV2 IV1
1573 static const u8 P2[16],
1574                C2[]=  {0x03,0x88,0xda,0xce,0x60,0xb6,0xa3,0x92,0xf3,0x28,0xc2,0
1575                T2[]=  {0xab,0x6e,0x47,0xd4,0x2c,0xec,0x13,0xbd,0xf5,0x3a,0x67,0

1577 /* Test Case 3 */
1578 #define A3 A2
1579 static const u8 K3[]=  {0xfe,0xff,0xe9,0x92,0x86,0x65,0x73,0x1c,0x6d,0x6a,0x8f,0
```

```
1580                P3[]=  {0xd9,0x31,0x32,0x25,0xf8,0x84,0x06,0xe5,0xa5,0x59,0x09,0
1581                        0x86,0xa7,0xa9,0x53,0x15,0x34,0xf7,0xda,0x2e,0x4c,0x30,0
1582                        0x1c,0x3c,0x0c,0x95,0x95,0x68,0x09,0x53,0x2f,0xcf,0x0e,0
1583                        0xb1,0x6a,0xed,0xf5,0xaa,0x0d,0xe6,0x57,0xba,0x63,0x7b,0
1584                IV3[]= {0xca,0xfe,0xba,0xbe,0xfa,0xce,0xdb,0xad,0xde,0xca,0xf8,0
1585                C3[]=  {0x42,0x83,0x1e,0xc2,0x21,0x77,0x74,0x24,0x4b,0x72,0x21,0
1586                        0xe3,0xaa,0x21,0x2f,0x2c,0x02,0xa4,0xe0,0x35,0xc1,0x7e,0
1587                        0x21,0xd5,0x14,0xb2,0x54,0x66,0x93,0x1c,0x7d,0x8f,0x6a,0
1588                        0x1b,0xa3,0x0b,0x39,0x6a,0x0a,0xac,0x97,0x3d,0x58,0xe0,0
1589                T3[]=  {0x4d,0x5c,0x2a,0xf3,0x27,0xcd,0x64,0xa6,0x2c,0xf3,0x5a,0

1591 /* Test Case 4 */
1592 #define K4 K3
1593 #define IV4 IV3
1594 static const u8 P4[]=  {0xd9,0x31,0x32,0x25,0xf8,0x84,0x06,0xe5,0xa5,0x59,0x09,0
1595                         0x86,0xa7,0xa9,0x53,0x15,0x34,0xf7,0xda,0x2e,0x4c,0x30,0
1596                         0x1c,0x3c,0x0c,0x95,0x95,0x68,0x09,0x53,0x2f,0xcf,0x0e,0
1597                         0xb1,0x6a,0xed,0xf5,0xaa,0x0d,0xe6,0x57,0xba,0x63,0x7b,0
1598                A4[]=  {0xfe,0xed,0xfa,0xce,0xde,0xad,0xbe,0xef,0xfe,0xed,0xfa,0
1599                         0xab,0xad,0xda,0xd2},
1600                C4[]=  {0x42,0x83,0x1e,0xc2,0x21,0x77,0x74,0x24,0x4b,0x72,0x21,0
1601                        0xe3,0xaa,0x21,0x2f,0x2c,0x02,0xa4,0xe0,0x35,0xc1,0x7e,0
1602                        0x21,0xd5,0x14,0xb2,0x54,0x66,0x93,0x1c,0x7d,0x8f,0x6a,0
1603                        0x1b,0xa3,0x0b,0x39,0x6a,0x0a,0xac,0x97,0x3d,0x58,0xe0,0
1604                T4[]=  {0x5b,0xc9,0x4f,0xbc,0x32,0x21,0xa5,0xdb,0x94,0xfa,0xe9,0

1606 /* Test Case 5 */
1607 #define K5 K4
1608 #define P5 P4
1609 #define A5 A4
1610 static const u8 IV5[]= {0xca,0xfe,0xba,0xbe,0xfa,0xce,0xdb,0xad},
1611                 C5[]=  {0x61,0x35,0x3b,0x4c,0x28,0x06,0x93,0x4a,0x77,0x7f,0xf5,0
1612                         0x69,0x9b,0x2a,0x71,0x4f,0xcd,0xc6,0xf8,0x37,0x66,0xe5,0
1613                         0x73,0x80,0x69,0x00,0xe4,0x9f,0x24,0xb2,0x2b,0x09,0x75,0
1614                         0x49,0x89,0xb5,0xe1,0xeb,0xac,0x0f,0x07,0xc2,0x3f,0x45,0
1615                T5[]=  {0x36,0x12,0xd2,0xe7,0x9e,0x3b,0x07,0x85,0x56,0x1b,0xe1,0

1617 /* Test Case 6 */
1618 #define K6 K5
1619 #define P6 P5
1620 #define A6 A5
1621 static const u8 IV6[]= {0x93,0x13,0x22,0x5d,0xf8,0x84,0x06,0xe5,0x55,0x90,0x9c,0
1622                         0x6a,0x7a,0x95,0x38,0x53,0x4f,0x7d,0xa1,0xe4,0xc3,0x03,0
1623                         0xc3,0xc0,0xc9,0x51,0x56,0x80,0x95,0x39,0xfc,0xf0,0xe2,0
1624                         0x16,0xae,0xdb,0xf5,0xa0,0xde,0x6a,0x57,0xa6,0x37,0xb3,0
1625                C6[]=  {0x8c,0xe2,0x49,0x98,0x62,0x56,0x15,0xb6,0x03,0xa0,0x33,0
1626                         0xbe,0x91,0x12,0xa5,0xc3,0xa2,0x11,0xa8,0xba,0x26,0x2a,0
1627                         0x01,0xe4,0xa9,0xa4,0xfb,0xa4,0x3c,0x90,0xcc,0xdc,0xb2,0
1628                         0xd6,0x28,0x75,0xd2,0xac,0xa4,0x17,0x03,0x4c,0x34,0xae,0
1629                T6[]=  {0x61,0x9c,0xc5,0xae,0xff,0xfe,0x0b,0xfa,0x46,0x2a,0xf4,0

1631 /* Test Case 7 */
1632 static const u8 K7[24],
1633                 *P7=NULL,
1634                 *A7=NULL,
1635                 IV7[12],
1636                 *C7=NULL,
1637                 T7[]=  {0xcd,0x33,0xb2,0x8a,0xc7,0x73,0xf7,0x4b,0xa0,0x0e,0xd1,0

1639 /* Test Case 8 */
1640 #define K8 K7
1641 #define IV8 IV7
1642 #define A8 A7
1643 static const u8 P8[16],
1644                 C8[]=  {0x98,0xe7,0x24,0x7c,0x07,0xf0,0xfe,0x41,0x1c,0x26,0x7e,0
1645                 T8[]=  {0x2f,0xf5,0x8d,0x80,0x03,0x39,0x27,0xab,0x8e,0xf4,0xd4,0
```



```
1647 /* Test Case 9 */
1648 #define A9 A8
1649 static const u8 K9[]=  {0xfe,0xff,0xe9,0x92,0x86,0x65,0x73,0x1c,0x6d,0x6a,0x8f,0
1650                         0xfe,0xff,0xe9,0x92,0x86,0x65,0x73,0x1c},
1651                 P9[]=  {0xd9,0x31,0x32,0x25,0xf8,0x84,0x06,0xe5,0xa5,0x59,0x09,0
1652                         0x86,0xa7,0xa9,0x53,0x15,0x34,0xf7,0xda,0x2e,0x4c,0x30,0
1653                         0x1c,0x3c,0x0c,0x95,0x95,0x68,0x09,0x53,0x2f,0xcf,0x0e,0
1654                         0xb1,0x6a,0xed,0xf5,0xaa,0x0d,0xe6,0x57,0xba,0x63,0x7b,0
1655                 IV9[]= {0xca,0xfe,0xba,0xbe,0xfa,0xce,0xdb,0xad,0xde,0xca,0xf8,0
1656                 C9[]=  {0x39,0x80,0xca,0x0b,0x3c,0x00,0xe8,0x41,0xeb,0x06,0xfa,0
1657                         0x85,0x9e,0x1c,0xea,0xa6,0xef,0xd9,0x84,0x62,0x85,0x93,0
1658                         0x7d,0x77,0x3d,0x00,0xc1,0x44,0xc5,0x25,0xac,0x61,0x9d,0
1659                         0x18,0xe2,0x44,0x8b,0x2f,0xe3,0x24,0xd9,0xcc,0xda,0x27,0
1660                 T9[]=  {0x99,0x24,0xa7,0xc8,0x58,0x73,0x36,0xbf,0xb1,0x18,0x02,0

1662 /* Test Case 10 */
1663 #define K10 K9
1664 #define IV10 IV9
1665 static const u8 P10[]= {0xd9,0x31,0x32,0x25,0xf8,0x84,0x06,0xe5,0xa5,0x59,0x09,0
1666                         0x86,0xa7,0xa9,0x53,0x15,0x34,0xf7,0xda,0x2e,0x4c,0x30,0
1667                         0x1c,0x3c,0x0c,0x95,0x95,0x68,0x09,0x53,0x2f,0xcf,0x0e,0
1668                         0xb1,0x6a,0xed,0xf5,0xaa,0x0d,0xe6,0x57,0xba,0x63,0x7b,0
1669                 A10[]= {0xfe,0xed,0xfa,0xce,0xde,0xad,0xbe,0xef,0xfe,0xed,0xfa,0
1670                         0xab,0xad,0xda,0xd2},
1671                 C10[]= {0x39,0x80,0xca,0x0b,0x3c,0x00,0xe8,0x41,0xeb,0x06,0xfa,0
1672                         0x85,0x9e,0x1c,0xea,0xa6,0xef,0xd9,0x84,0x62,0x85,0x93,0
1673                         0x7d,0x77,0x3d,0x00,0xc1,0x44,0xc5,0x25,0xac,0x61,0x9d,0
1674                         0x18,0xe2,0x44,0x8b,0x2f,0xe3,0x24,0xd9,0xcc,0xda,0x27,0
1675                 T10[]= {0x25,0x19,0x49,0x8e,0x80,0xf1,0x47,0x8f,0x37,0xba,0x55,0

1677 /* Test Case 11 */
1678 #define K11 K10
1679 #define P11 P10
1680 #define A11 A10
1681 static const u8 IV11[]={0xca,0xfe,0xba,0xbe,0xfa,0xce,0xdb,0xad},
1682                 C11[]= {0x0f,0x10,0xf5,0x99,0xae,0x14,0xa1,0x54,0xed,0x24,0xb3,0
1683                         0xc5,0x66,0x63,0x2e,0xf2,0xbb,0xb3,0x4f,0x83,0x47,0x28,0
1684                         0xfd,0xdc,0x29,0xdf,0x9a,0x47,0x1f,0x75,0xc6,0x65,0x41,0
1685                         0xe9,0x3a,0x19,0xa5,0x8e,0x8b,0x47,0x3f,0xa0,0xf0,0x62,0
1686                 T11[]= {0x65,0xdc,0xc5,0x7f,0xcf,0x62,0x3a,0x24,0x09,0x4f,0xcc,0

1688 /* Test Case 12 */
1689 #define K12 K11
1690 #define P12 P11
1691 #define A12 A11
1692 static const u8 IV12[]={0x93,0x13,0x22,0x5d,0xf8,0x84,0x06,0xe5,0x55,0x90,0x9c,0
1693                         0x6a,0x7a,0x95,0x38,0x53,0x4f,0x7d,0xa1,0xe4,0xc3,0x03,0
1694                         0xc3,0xc0,0xc9,0x51,0x56,0x80,0x95,0x39,0xfc,0xf0,0xe2,0
1695                         0x16,0xae,0xdb,0xf5,0xa0,0xde,0x6a,0x57,0xa6,0x37,0xb3,0
1696                 C12[]= {0xd2,0x7e,0x88,0x68,0x1c,0xe3,0x24,0x3c,0x48,0x30,0x16,0
1697                         0x1d,0xe9,0xa1,0xd8,0xe6,0xb4,0x47,0xef,0x6e,0xf7,0xb7,0
1698                         0x81,0xe7,0x90,0x12,0xaf,0x34,0xdd,0xd9,0xe2,0xf0,0x37,0
1699                         0xe6,0x7c,0x03,0x67,0x45,0xfa,0x22,0xe7,0xe9,0xb7,0x37,0
1700                 T12[]= {0xdc,0xf5,0x66,0xff,0x29,0x1c,0x25,0xbb,0xb8,0x56,0x8f,0

1702 /* Test Case 13 */
1703 static const u8 K13[32],
1704                 *P13=NULL,
1705                 *A13=NULL,
1706                 IV13[12],
1707                 *C13=NULL,
1708                 T13[]={0x53,0x0f,0x8a,0xfb,0xc7,0x45,0x36,0xb9,0xa9,0x63,0xb4,0x

1710 /* Test Case 14 */
1711 #define K14 K13
```

```
#define A14 A13
static const u8 P14[16],
                IV14[12],
                C14[]= {0xce,0xa7,0x40,0x3d,0x4d,0x60,0x6b,0x6e,0x07,0x4e,0xc5,0
                T14[]= {0xd0,0xd1,0xc8,0xa7,0x99,0x99,0x6b,0xf0,0x26,0x5b,0x98,0

/* Test Case 15 */
#define A15 A14
static const u8 K15[]= {0xfe,0xff,0xe9,0x92,0x86,0x65,0x73,0x1c,0x6d,0x6a,0x8f,0
                        0xfe,0xff,0xe9,0x92,0x86,0x65,0x73,0x1c,0x6d,0x6a,0x8f,0
                P15[]= {0xd9,0x31,0x32,0x25,0xf8,0x84,0x06,0xe5,0xa5,0x59,0x09,0
                        0x86,0xa7,0xa9,0x53,0x15,0x34,0xf7,0xda,0x2e,0x4c,0x30,0
                        0x1c,0x3c,0x0c,0x95,0x95,0x68,0x09,0x53,0x2f,0xcf,0x0e,0
                        0xb1,0x6a,0xed,0xf5,0xaa,0x0d,0xe6,0x57,0xba,0x63,0x7b,0
                IV15[]={0xca,0xfe,0xba,0xbe,0xfa,0xce,0xdb,0xad,0xde,0xca,0xf8,0
                C15[]= {0x52,0x2d,0xc1,0xf0,0x99,0x56,0x7d,0x07,0xf4,0x7f,0x37,0
                        0x64,0x3a,0x8c,0xdc,0xbf,0xe5,0xc0,0xc9,0x75,0x98,0xa2,0
                        0x8c,0xb0,0x8e,0x48,0x59,0x0d,0xbb,0x3d,0xa7,0xb0,0x8b,0
                        0xc5,0xf6,0x1e,0x63,0x93,0xba,0x7a,0x0a,0xbc,0xc9,0xf6,0
                T15[]= {0xb0,0x94,0xda,0xc5,0xd9,0x34,0x71,0xbd,0xec,0x1a,0x50,0

/* Test Case 16 */
#define K16 K15
#define IV16 IV15
static const u8 P16[]= {0xd9,0x31,0x32,0x25,0xf8,0x84,0x06,0xe5,0xa5,0x59,0x09,0
                        0x86,0xa7,0xa9,0x53,0x15,0x34,0xf7,0xda,0x2e,0x4c,0x30,0
                        0x1c,0x3c,0x0c,0x95,0x95,0x68,0x09,0x53,0x2f,0xcf,0x0e,0
                        0xb1,0x6a,0xed,0xf5,0xaa,0x0d,0xe6,0x57,0xba,0x63,0x7b,0
                A16[]= {0xfe,0xed,0xfa,0xce,0xde,0xad,0xbe,0xef,0xfe,0xed,0xfa,0
                        0xab,0xad,0xda,0xd2},
                C16[]= {0x52,0x2d,0xc1,0xf0,0x99,0x56,0x7d,0x07,0xf4,0x7f,0x37,0
                        0x64,0x3a,0x8c,0xdc,0xbf,0xe5,0xc0,0xc9,0x75,0x98,0xa2,0
                        0x8c,0xb0,0x8e,0x48,0x59,0x0d,0xbb,0x3d,0xa7,0xb0,0x8b,0
                        0xc5,0xf6,0x1e,0x63,0x93,0xba,0x7a,0x0a,0xbc,0xc9,0xf6,0
                T16[]= {0x76,0xfc,0x6e,0xce,0x0f,0x4e,0x17,0x68,0xcd,0xdf,0x88,0

/* Test Case 17 */
#define K17 K16
#define P17 P16
#define A17 A16
static const u8 IV17[]={0xca,0xfe,0xba,0xbe,0xfa,0xce,0xdb,0xad},
                C17[]= {0xc3,0x76,0x2d,0xf1,0xca,0x78,0x7d,0x32,0xae,0x47,0xc1,0
                        0xaf,0x1a,0xe1,0x4d,0x0b,0x97,0x6a,0xfa,0xc5,0x2f,0xf7,0
                        0xfe,0xb5,0x82,0xd3,0x39,0x34,0xa4,0xf0,0x95,0x4c,0xc2,0
                        0x62,0xac,0x43,0x0e,0x64,0xab,0xe4,0x99,0xf4,0x7c,0x9b,0
                T17[]= {0x3a,0x33,0x7d,0xbf,0x46,0xa7,0x92,0xc4,0x5e,0x45,0x49,0

/* Test Case 18 */
#define K18 K17
#define P18 P17
#define A18 A17
static const u8 IV18[]={0x93,0x13,0x22,0x5d,0xf8,0x84,0x06,0xe5,0x55,0x90,0x9c,0
                        0x6a,0x7a,0x95,0x38,0x53,0x4f,0x7d,0xa1,0xe4,0xc3,0x03,0
                        0xc3,0xc0,0xc9,0x51,0x56,0x80,0x95,0x39,0xfc,0xf0,0xe2,0
                        0x16,0xae,0xdb,0xf5,0xa0,0xde,0x6a,0x57,0xa6,0x37,0xb3,0
                C18[]= {0x5a,0x8d,0xef,0x2f,0x0c,0x9e,0x53,0xf1,0xf7,0x5d,0x78,0
                        0xee,0xb2,0xb2,0x2a,0xaf,0xde,0x64,0x19,0xa0,0x58,0xab,0
                        0x0f,0xc0,0xc3,0xb7,0x80,0xf2,0x44,0x45,0x2d,0xa3,0xeb,0
                        0xa2,0x41,0x89,0x97,0x20,0x0e,0xf8,0x2e,0x44,0xae,0x7e,0
                T18[]= {0xa4,0x4a,0x82,0x66,0xee,0x1c,0x8e,0xb0,0xc8,0xb5,0xd4,0

/* Test Case 19 */
#define K19 K1
#define P19 P1
#define IV19 IV1
#define C19 C1
```



```
static const u8 A19[]= {0xd9,0x31,0x32,0x25,0xf8,0x84,0x06,0xe5,0xa5,0x59,0x09,0
                        0x86,0xa7,0xa9,0x53,0x15,0x34,0xf7,0xda,0x2e,0x4c,0x30,0
                        0x1c,0x3c,0x0c,0x95,0x95,0x68,0x09,0x53,0x2f,0xcf,0x0e,0
                        0xb1,0x6a,0xed,0xf5,0xaa,0x0d,0xe6,0x57,0xba,0x63,0x7b,0
                        0x52,0x2d,0xc1,0xf0,0x99,0x56,0x7d,0x07,0xf4,0x7f,0x37,0
                        0x64,0x3a,0x8c,0xdc,0xbf,0xe5,0xc0,0xc9,0x75,0x98,0xa2,0
                        0x8c,0xb0,0x8e,0x48,0x59,0x0d,0xbb,0x3d,0xa7,0xb0,0x8b,0
                        0xc5,0xf6,0x1e,0x63,0x93,0xba,0x7a,0x0a,0xbc,0xc9,0xf6,0
                T19[]= {0x5f,0xea,0x79,0x3a,0x2d,0x6f,0x97,0x4d,0x37,0xe6,0x8e,0

/* Test Case 20 */
#define K20 K1
#define A20 A1
static const u8 IV20[64]={0xff,0xff,0xff,0xff}, /* this results in 0xff in count
                P20[288],
                C20[]= {0x56,0xb3,0x37,0x3c,0xa9,0xef,0x6e,0x4a,0x2b,0x64,0xfe,0
                        0x25,0xf1,0x0d,0x47,0xa7,0x5a,0x5f,0xce,0x13,0xef,0xc6,0
                        0x41,0x41,0xbd,0xd4,0x8c,0xf7,0xc7,0x70,0x88,0x7a,0xfd,0
                        0xa9,0xae,0xff,0xcd,0x7c,0x5c,0xed,0xdf,0xc6,0xa7,0x83,0
                        0x9d,0xa5,0x58,0x25,0x72,0x67,0xca,0xab,0x2a,0xd0,0xb2,0
                        0xb1,0x7f,0xb4,0x1c,0x4b,0x8b,0x47,0x5c,0xb4,0xf3,0xf7,0
                        0xc9,0xe8,0xc4,0xdc,0x0a,0x2a,0x5f,0xf1,0x90,0x3e,0x50,0
                        0xa1,0xcd,0xb8,0x36,0x4c,0x50,0x61,0xa2,0x0c,0xae,0x74,0
                        0xb0,0xab,0xc9,0xfd,0x32,0x17,0xef,0x9f,0x8c,0x90,0xbe,0
                        0x97,0xf4,0xf8,0x80,0xdf,0xf1,0x5b,0xfb,0x7a,0x6b,0x28,0
                        0x3c,0x2d,0x59,0xe3,0xf9,0xdf,0xff,0x65,0x3c,0x71,0x26,0
                        0x11,0xf4,0x2b,0xae,0x12,0xaf,0x46,0x2b,0x10,0x70,0xbe,0
                        0x87,0x2c,0xa1,0x0d,0xee,0x15,0xb3,0x24,0x9b,0x1a,0x1b,0
                        0x4b,0xcc,0xb7,0xd0,0x32,0x00,0xbc,0xe4,0x20,0xa2,0xf8,0
                        0x4d,0x14,0x23,0xc1,0xb5,0x69,0x90,0x03,0xc1,0x3e,0xce,0
                        0x0e,0xed,0xc3,0x40,0x33,0xba,0xc1,0x90,0x27,0x83,0xdc,0
                        0x18,0x8a,0x43,0x9c,0x7e,0xbc,0xc0,0x67,0x2d,0xbd,0xa4,0
                        0x13,0xb0,0xbe,0x41,0x31,0x5e,0xf7,0x78,0x70,0x8a,0x70,0
                T20[]= {0x8b,0x30,0x7f,0x6b,0x33,0x28,0x6d,0x0a,0xb0,0x26,0xa9,0

#define TEST_CASE(n)    do {                                            \
        u8 out[sizeof(P##n)];                                           \
        AES_set_encrypt_key(K##n,sizeof(K##n)*8,&key);                  \
        CRYPTO_gcm128_init(&ctx,&key,(block128_f)AES_encrypt);          \
        CRYPTO_gcm128_setiv(&ctx,IV##n,sizeof(IV##n));                  \
        memset(out,0,sizeof(out));                                      \
        if (A##n) CRYPTO_gcm128_aad(&ctx,A##n,sizeof(A##n));            \
        if (P##n) CRYPTO_gcm128_encrypt(&ctx,P##n,out,sizeof(out));     \
        if (CRYPTO_gcm128_finish(&ctx,T##n,16) ||                       \
            (C##n && memcmp(out,C##n,sizeof(out))))                     \
                ret++, printf ("encrypt test#%d failed.\n",n);          \
        CRYPTO_gcm128_setiv(&ctx,IV##n,sizeof(IV##n));                  \
        memset(out,0,sizeof(out));                                      \
        if (A##n) CRYPTO_gcm128_aad(&ctx,A##n,sizeof(A##n));            \
        if (C##n) CRYPTO_gcm128_decrypt(&ctx,C##n,out,sizeof(out));     \
        if (CRYPTO_gcm128_finish(&ctx,T##n,16) ||                       \
            (P##n && memcmp(out,P##n,sizeof(out))))                     \
                ret++, printf ("decrypt test#%d failed.\n",n);          \
        } while(0)

int main()
{
        GCM128_CONTEXT ctx;
        AES_KEY key;
        int ret=0;

        TEST_CASE(1);
        TEST_CASE(2);
        TEST_CASE(3);
        TEST_CASE(4);
        TEST_CASE(5);
```

```
1844         TEST_CASE(6);
1845         TEST_CASE(7);
1846         TEST_CASE(8);
1847         TEST_CASE(9);
1848         TEST_CASE(10);
1849         TEST_CASE(11);
1850         TEST_CASE(12);
1851         TEST_CASE(13);
1852         TEST_CASE(14);
1853         TEST_CASE(15);
1854         TEST_CASE(16);
1855         TEST_CASE(17);
1856         TEST_CASE(18);
1857         TEST_CASE(19);
1858         TEST_CASE(20);

1860 #ifdef OPENSSL_CPUID_OBJ
1861         {
1862         size_t start,stop,gcm_t,ctr_t,OPENSSL_rdtsc();
1863         union { u64 u; u8 c[1024]; } buf;
1864         int i;

1866         AES_set_encrypt_key(K1,sizeof(K1)*8,&key);
1867         CRYPTO_gcm128_init(&ctx,&key,(block128_f)AES_encrypt);
1868         CRYPTO_gcm128_setiv(&ctx,IV1,sizeof(IV1));

1870         CRYPTO_gcm128_encrypt(&ctx,buf.c,buf.c,sizeof(buf));
1871         start = OPENSSL_rdtsc();
1872         CRYPTO_gcm128_encrypt(&ctx,buf.c,buf.c,sizeof(buf));
1873         gcm_t = OPENSSL_rdtsc() - start;

1875         CRYPTO_ctr128_encrypt(buf.c,buf.c,sizeof(buf),
1876                         &key,ctx.Yi.c,ctx.EKi.c,&ctx.mres,
1877                         (block128_f)AES_encrypt);
1878         start = OPENSSL_rdtsc();
1879         CRYPTO_ctr128_encrypt(buf.c,buf.c,sizeof(buf),
1880                         &key,ctx.Yi.c,ctx.EKi.c,&ctx.mres,
1881                         (block128_f)AES_encrypt);
1882         ctr_t = OPENSSL_rdtsc() - start;

1884         printf("%.2f-%.2f=%.2f\n",
1885                         gcm_t/(double)sizeof(buf),
1886                         ctr_t/(double)sizeof(buf),
1887                         (gcm_t-ctr_t)/(double)sizeof(buf));
1888 #ifdef GHASH
1889         {
1890         void (*gcm_ghash_p)(u64 Xi[2],const u128 Htable[16],
1891                             const u8 *inp,size_t len)       = ctx.ghash;

1893         GHASH((&ctx),buf.c,sizeof(buf));
1894         start = OPENSSL_rdtsc();
1895         for (i=0;i<100;++i) GHASH((&ctx),buf.c,sizeof(buf));
1896         gcm_t = OPENSSL_rdtsc() - start;
1897         printf("%.2f\n",gcm_t/(double)sizeof(buf)/(double)i);
1898         }
1899 #endif
1900         }
1901 #endif

1903         return ret;
1904 }
1905 #endif
1906 #endif /* ! codereview */
```

****
3906 Wed Aug 13 19:52:55 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/ofb128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>
```



```
/* The input and output encrypted as though 128bit ofb mode is being
 * used.  The extra state information to record how much of the
 * 128bit block we have used is contained in *num;
 */
void CRYPTO_ofb128_encrypt(const unsigned char *in, unsigned char *out,
                           size_t len, const void *key,
                           unsigned char ivec[16], int *num,
                           block128_f block)
{
        unsigned int n;
        size_t l=0;

        assert(in && out && key && ivec && num);

        n = *num;

#if !defined(OPENSSL_SMALL_FOOTPRINT)
        if (16%sizeof(size_t) == 0) do { /* always true actually */
                while (n && len) {
                        *(out++) = *(in++) ^ ivec[n];
                        --len;
                        n = (n+1) % 16;
                }
#if defined(STRICT_ALIGNMENT)
                if (((size_t)in|(size_t)out|(size_t)ivec)%sizeof(size_t) != 0)
                        break;
#endif
                while (len>=16) {
                        (*block)(ivec, ivec, key);
                        for (; n<16; n+=sizeof(size_t))
                                *(size_t*)(out+n) =
                                *(size_t*)(in+n) ^ *(size_t*)(ivec+n);
                        len -= 16;
                        out += 16;
                        in  += 16;
                        n = 0;
                }
                if (len) {
                        (*block)(ivec, ivec, key);
                        while (len--) {
                                out[n] = in[n] ^ ivec[n];
                                ++n;
                        }
                }
                *num = n;
                return;
        } while(0);
        /* the rest would be commonly eliminated by x86* compiler */
#endif
        while (l<len) {
                if (n==0) {
                        (*block)(ivec, ivec, key);
                }
                out[l] = in[l] ^ ivec[n];
                ++l;
                n = (n+1) % 16;
        }

        *num=n;
}
#endif /* ! codereview */
```

****
5607 Wed Aug 13 19:52:55 2014
new/usr/src/lib/openssl/libsunw_crypto/modes/xts128.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/crypto.h>
#include "modes_lcl.h"
#include <string.h>

#ifndef MODES_DEBUG
# ifndef NDEBUG
#  define NDEBUG
# endif
#endif
#include <assert.h>

int CRYPTO_xts128_encrypt(const XTS128_CONTEXT *ctx, const unsigned char iv[16],
```



```
        const unsigned char *inp, unsigned char *out,
        size_t len, int enc)
{
        const union { long one; char little; } is_endian = {1};
        union { u64 u[2]; u32 d[4]; u8 c[16]; } tweak, scratch;
        unsigned int i;

        if (len<16) return -1;

        memcpy(tweak.c, iv, 16);

        (*ctx->block2)(tweak.c,tweak.c,ctx->key2);

        if (!enc && (len%16)) len-=16;

        while (len>=16) {
#if defined(STRICT_ALIGNMENT)
                memcpy(scratch.c,inp,16);
                scratch.u[0] ^= tweak.u[0];
                scratch.u[1] ^= tweak.u[1];
#else
                scratch.u[0] = ((u64*)inp)[0]^tweak.u[0];
                scratch.u[1] = ((u64*)inp)[1]^tweak.u[1];
#endif
                (*ctx->block1)(scratch.c,scratch.c,ctx->key1);
#if defined(STRICT_ALIGNMENT)
                scratch.u[0] ^= tweak.u[0];
                scratch.u[1] ^= tweak.u[1];
                memcpy(out,scratch.c,16);
#else
                ((u64*)out)[0] = scratch.u[0]^=tweak.u[0];
                ((u64*)out)[1] = scratch.u[1]^=tweak.u[1];
#endif
                inp += 16;
                out += 16;
                len -= 16;

                if (len==0)     return 0;

                if (is_endian.little) {
                        unsigned int carry,res;

                        res = 0x87&(((int)tweak.d[3])>>31);
                        carry = (unsigned int)(tweak.u[0]>>63);
                        tweak.u[0] = (tweak.u[0]<<1)^res;
                        tweak.u[1] = (tweak.u[1]<<1)|carry;
                }
                else {
                        size_t c;

                        for (c=0,i=0;i<16;++i) {
                                /*+ substitutes for |, because c is 1 bit */
                                c += ((size_t)tweak.c[i])<<1;
                                tweak.c[i] = (u8)c;
                                c = c>>8;
                        }
                        tweak.c[0] ^= (u8)(0x87&(0-c));
                }
        }
        if (enc) {
                for (i=0;i<len;++i) {
                        u8 c = inp[i];
                        out[i] = scratch.c[i];
                        scratch.c[i] = c;
                }
                scratch.u[0] ^= tweak.u[0];
```

```
128                 scratch.u[1] ^= tweak.u[1];
129                 (*ctx->block1)(scratch.c,scratch.c,ctx->key1);
130                 scratch.u[0] ^= tweak.u[0];
131                 scratch.u[1] ^= tweak.u[1];
132                 memcpy(out-16,scratch.c,16);
133         }
134         else {
135                 union { u64 u[2]; u8 c[16]; } tweak1;

137                 if (is_endian.little) {
138                         unsigned int carry,res;

140                         res = 0x87&(((int)tweak.d[3])>>31);
141                         carry = (unsigned int)(tweak.u[0]>>63);
142                         tweak1.u[0] = (tweak.u[0]<<1)^res;
143                         tweak1.u[1] = (tweak.u[1]<<1)|carry;
144                 }
145                 else {
146                         size_t c;

148                         for (c=0,i=0;i<16;++i) {
149                                 /*+ substitutes for |, because c is 1 bit */
150                                 c += ((size_t)tweak.c[i])<<1;
151                                 tweak1.c[i] = (u8)c;
152                                 c = c>>8;
153                         }
154                         tweak1.c[0] ^= (u8)(0x87&(0-c));
155                 }
156 #if defined(STRICT_ALIGNMENT)
157                 memcpy(scratch.c,inp,16);
158                 scratch.u[0] ^= tweak1.u[0];
159                 scratch.u[1] ^= tweak1.u[1];
160 #else
161                 scratch.u[0] = ((u64*)inp)[0]^tweak1.u[0];
162                 scratch.u[1] = ((u64*)inp)[1]^tweak1.u[1];
163 #endif
164                 (*ctx->block1)(scratch.c,scratch.c,ctx->key1);
165                 scratch.u[0] ^= tweak1.u[0];
166                 scratch.u[1] ^= tweak1.u[1];

168                 for (i=0;i<len;++i) {
169                         u8 c = inp[16+i];
170                         out[16+i] = scratch.c[i];
171                         scratch.c[i] = c;
172                 }
173                 scratch.u[0] ^= tweak.u[0];
174                 scratch.u[1] ^= tweak.u[1];
175                 (*ctx->block1)(scratch.c,scratch.c,ctx->key1);
176 #if defined(STRICT_ALIGNMENT)
177                 scratch.u[0] ^= tweak.u[0];
178                 scratch.u[1] ^= tweak.u[1];
179                 memcpy (out,scratch.c,16);
180 #else
181                 ((u64*)out)[0] = scratch.u[0]^tweak.u[0];
182                 ((u64*)out)[1] = scratch.u[1]^tweak.u[1];
183 #endif
184         }

186         return 0;
187 }
188 #endif /* ! codereview */
```

```
**********************************************************
    3450 Wed Aug 13 19:52:55 2014
new/usr/src/lib/openssl/libsunw_crypto/o_dir.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/o_dir.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <errno.h>
#include <e_os.h>
```



```
/* The routines really come from the Levitte Programming, so to make
   life simple, let's just use the raw files and hack the symbols to
   fit our namespace.  */
#define LP_DIR_CTX OPENSSL_DIR_CTX
#define LP_dir_context_st OPENSSL_dir_context_st
#define LP_find_file OPENSSL_DIR_read
#define LP_find_file_end OPENSSL_DIR_end

#include "o_dir.h"

#define LPDIR_H
#if defined OPENSSL_SYS_UNIX || defined DJGPP
#include "LPdir_unix.c"
#elif defined OPENSSL_SYS_VMS
#include "LPdir_vms.c"
#elif defined OPENSSL_SYS_WIN32
#include "LPdir_win32.c"
#elif defined OPENSSL_SYS_WINCE
#include "LPdir_wince.c"
#else
#include "LPdir_nyi.c"
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3417 Wed Aug 13 19:52:55 2014
new/usr/src/lib/openssl/libsunw_crypto/o_fips.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Stephen henson (steve@openssl.org) for the OpenSSL
 * project 2011.
 */
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#include <openssl/fips_rand.h>
```



```
#include <openssl/rand.h>
#endif

int FIPS_mode(void)
        {
        OPENSSL_init();
#ifdef OPENSSL_FIPS
        return FIPS_module_mode();
#else
        return 0;
#endif
        }

int FIPS_mode_set(int r)
        {
        OPENSSL_init();
#ifdef OPENSSL_FIPS
#ifndef FIPS_AUTH_USER_PASS
#define FIPS_AUTH_USER_PASS     "Default FIPS Crypto User Password"
#endif
        if (!FIPS_module_mode_set(r, FIPS_AUTH_USER_PASS))
                return 0;
        if (r)
                RAND_set_rand_method(FIPS_rand_get_method());
        else
                RAND_set_rand_method(NULL);
        return 1;
#else
        if (r == 0)
                return 1;
        CRYPTOerr(CRYPTO_F_FIPS_MODE_SET, CRYPTO_R_FIPS_MODE_NOT_SUPPORTED);
        return 0;
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    3145 Wed Aug 13 19:52:56 2014
new/usr/src/lib/openssl/libsunw_crypto/o_init.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* o_init.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <e_os.h>
#include <openssl/err.h>
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#include <openssl/rand.h>
#endif
```



```
/* Perform any essential OpenSSL initialization operations.
 * Currently only sets FIPS callbacks
 */

void OPENSSL_init(void)
        {
        static int done = 0;
        if (done)
                return;
        done = 1;
#ifdef OPENSSL_FIPS
        FIPS_set_locking_callbacks(CRYPTO_lock, CRYPTO_add_lock);
        FIPS_set_error_callbacks(ERR_put_error, ERR_add_error_vdata);
        FIPS_set_malloc_callbacks(CRYPTO_malloc, CRYPTO_free);
        RAND_init_fips();
#endif
#if 0
        fprintf(stderr, "Called OPENSSL_init\n");
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    3914 Wed Aug 13 19:52:56 2014
new/usr/src/lib/openssl/libsunw_crypto/o_str.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/o_str.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2003.
 */
/* ====================================================================
 * Copyright (c) 2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <ctype.h>
#include <e_os.h>
#include <o_str.h>
```



```
#if !defined(OPENSSL_IMPLEMENTS_strncasecmp) && \
    !defined(OPENSSL_SYSNAME_WIN32) && \
    !defined(NETWARE_CLIB)
# include <strings.h>
#endif

int OPENSSL_strncasecmp(const char *str1, const char *str2, size_t n)
        {
#if defined(OPENSSL_IMPLEMENTS_strncasecmp)
        while (*str1 && *str2 && n)
                {
                int res = toupper(*str1) - toupper(*str2);
                if (res) return res < 0 ? -1 : 1;
                str1++;
                str2++;
                n--;
                }
        if (n == 0)
                return 0;
        if (*str1)
                return 1;
        if (*str2)
                return -1;
        return 0;
#else
        /* Recursion hazard warning! Whenever strncasecmp is #defined as
         * OPENSSL_strncasecmp, OPENSSL_IMPLEMENTS_strncasecmp must be
         * defined as well. */
        return strncasecmp(str1, str2, n);
#endif
        }
int OPENSSL_strcasecmp(const char *str1, const char *str2)
        {
#if defined(OPENSSL_IMPLEMENTS_strncasecmp)
        return OPENSSL_strncasecmp(str1, str2, (size_t)-1);
#else
        return strcasecmp(str1, str2);
#endif
        }

int OPENSSL_memcmp(const void *v1,const void *v2,size_t n)
        {
        const unsigned char *c1=v1,*c2=v2;
        int ret=0;

        while(n && (ret=*c1-*c2)==0) n--,c1++,c2++;

        return ret;
        }
#endif /* ! codereview */
```

```
**********************************************************
   11238 Wed Aug 13 19:52:56 2014
new/usr/src/lib/openssl/libsunw_crypto/o_time.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/o_time.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2008.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
```



```
#include <openssl/e_os2.h>
#include <string.h>
#include <o_time.h>

#ifdef OPENSSL_SYS_VMS
# if __CRTL_VER >= 70000000 && \
     (defined _POSIX_C_SOURCE || !defined _ANSI_C_SOURCE)
#  define VMS_GMTIME_OK
# endif
# ifndef VMS_GMTIME_OK
#  include <libdtdef.h>
#  include <lib$routines.h>
#  include <lnmdef.h>
#  include <starlet.h>
#  include <descrip.h>
#  include <stdlib.h>
# endif /* ndef VMS_GMTIME_OK */
#endif

struct tm *OPENSSL_gmtime(const time_t *timer, struct tm *result)
        {
        struct tm *ts = NULL;

#if defined(OPENSSL_THREADS) && !defined(OPENSSL_SYS_WIN32) && !defined(OPENSSL_
        /* should return &data, but doesn't on some systems,
           so we don't even look at the return value */
        gmtime_r(timer,result);
        ts = result;
#elif !defined(OPENSSL_SYS_VMS) || defined(VMS_GMTIME_OK)
        ts = gmtime(timer);
        if (ts == NULL)
                return NULL;

        memcpy(result, ts, sizeof(struct tm));
        ts = result;
#endif
#if defined( OPENSSL_SYS_VMS) && !defined( VMS_GMTIME_OK)
        if (ts == NULL)
                {
                static $DESCRIPTOR(tabnam,"LNM$DCL_LOGICAL");
                static $DESCRIPTOR(lognam,"SYS$TIMEZONE_DIFFERENTIAL");
                char logvalue[256];
                unsigned int reslen = 0;
                struct {
                        short buflen;
                        short code;
                        void *bufaddr;
                        unsigned int *reslen;
                } itemlist[] = {
                        { 0, LNM$_STRING, 0, 0 },
                        { 0, 0, 0, 0 },
                };
                int status;
                time_t t;

                /* Get the value for SYS$TIMEZONE_DIFFERENTIAL */
                itemlist[0].buflen = sizeof(logvalue);
                itemlist[0].bufaddr = logvalue;
                itemlist[0].reslen = &reslen;
                status = sys$trnlnm(0, &tabnam, &lognam, 0, itemlist);
                if (!(status & 1))
                        return NULL;
                logvalue[reslen] = '\0';

                t = *timer;
```

```
/* The following is extracted from the DEC C header time.h */
/*
**  Beginning in OpenVMS Version 7.0 mktime, time, ctime, strftime
**  have two implementations.  One implementation is provided
**  for compatibility and deals with time in terms of local time,
**  the other __utc_* deals with time in terms of UTC.
*/
/* We use the same conditions as in said time.h to check if we should
   assume that t contains local time (and should therefore be adjusted)
   or UTC (and should therefore be left untouched). */
#if __CRTL_VER < 70000000 || defined _VMS_V6_SOURCE
                /* Get the numerical value of the equivalence string */
                status = atoi(logvalue);

                /* and use it to move time to GMT */
                t -= status;
#endif

                /* then convert the result to the time structure */

                /* Since there was no gmtime_r() to do this stuff for us,
                   we have to do it the hard way. */
                {
                /* The VMS epoch is the astronomical Smithsonian date,
                   if I remember correctly, which is November 17, 1858.
                   Furthermore, time is measure in thenths of microseconds
                   and stored in quadwords (64 bit integers).  unix_epoch
                   below is January 1st 1970 expressed as a VMS time.  The
                   following code was used to get this number:

                   #include <stdio.h>
                   #include <stdlib.h>
                   #include <lib$routines.h>
                   #include <starlet.h>

                   main()
                   {
                     unsigned long systime[2];
                     unsigned short epoch_values[7] =
                       { 1970, 1, 1, 0, 0, 0, 0 };

                     lib$cvt_vectim(epoch_values, systime);

                     printf("%u %u", systime[0], systime[1]);
                   }
                */
                unsigned long unix_epoch[2] = { 1273708544, 8164711 };
                unsigned long deltatime[2];
                unsigned long systime[2];
                struct vms_vectime
                       {
                       short year, month, day, hour, minute, second,
                              centi_second;
                       } time_values;
                long operation;

                /* Turn the number of seconds since January 1st 1970 to
                   an internal delta time.
                   Note that lib$cvt_to_internal_time() will assume
                   that t is signed, and will therefore break on 32-bit
                   systems some time in 2038.
                */
                operation = LIB$K_DELTA_SECONDS;
                status = lib$cvt_to_internal_time(&operation,
                       &t, deltatime);
```



```
                /* Add the delta time with the Unix epoch and we have
                   the current UTC time in internal format */
                status = lib$add_times(unix_epoch, deltatime, systime);

                /* Turn the internal time into a time vector */
                status = sys$numtim(&time_values, systime);

                /* Fill in the struct tm with the result */
                result->tm_sec = time_values.second;
                result->tm_min = time_values.minute;
                result->tm_hour = time_values.hour;
                result->tm_mday = time_values.day;
                result->tm_mon = time_values.month - 1;
                result->tm_year = time_values.year - 1900;

                operation = LIB$K_DAY_OF_WEEK;
                status = lib$cvt_from_internal_time(&operation,
                       &result->tm_wday, systime);
                result->tm_wday %= 7;

                operation = LIB$K_DAY_OF_YEAR;
                status = lib$cvt_from_internal_time(&operation,
                       &result->tm_yday, systime);
                result->tm_yday--;

                result->tm_isdst = 0; /* There's no way to know... */

                ts = result;
                }
                }
#endif
        return ts;
        }

/* Take a tm structure and add an offset to it. This avoids any OS issues
 * with restricted date types and overflows which cause the year 2038
 * problem.
 */

#define SECS_PER_DAY (24 * 60 * 60)

static long date_to_julian(int y, int m, int d);
static void julian_to_date(long jd, int *y, int *m, int *d);

int OPENSSL_gmtime_adj(struct tm *tm, int off_day, long offset_sec)
        {
        int offset_hms, offset_day;
        long time_jd;
        int time_year, time_month, time_day;
        /* split offset into days and day seconds */
        offset_day = offset_sec / SECS_PER_DAY;
        /* Avoid sign issues with % operator */
        offset_hms  = offset_sec - (offset_day * SECS_PER_DAY);
        offset_day += off_day;
        /* Add current time seconds to offset */
        offset_hms += tm->tm_hour * 3600 + tm->tm_min * 60 + tm->tm_sec;
        /* Adjust day seconds if overflow */
        if (offset_hms >= SECS_PER_DAY)
                {
                offset_day++;
                offset_hms -= SECS_PER_DAY;
                }
        else if (offset_hms < 0)
                {
                offset_day--;
                offset_hms += SECS_PER_DAY;
```

```
260                 }

262         /* Convert date of time structure into a Julian day number.
263          */

265         time_year = tm->tm_year + 1900;
266         time_month = tm->tm_mon + 1;
267         time_day = tm->tm_mday;

269         time_jd = date_to_julian(time_year, time_month, time_day);

271         /* Work out Julian day of new date */
272         time_jd += offset_day;

274         if (time_jd < 0)
275                 return 0;

277         /* Convert Julian day back to date */

279         julian_to_date(time_jd, &time_year, &time_month, &time_day);

281         if (time_year < 1900 || time_year > 9999)
282                 return 0;

284         /* Update tm structure */

286         tm->tm_year = time_year - 1900;
287         tm->tm_mon = time_month - 1;
288         tm->tm_mday = time_day;

290         tm->tm_hour = offset_hms / 3600;
291         tm->tm_min = (offset_hms / 60) % 60;
292         tm->tm_sec = offset_hms % 60;

294         return 1;

296 }

298 /* Convert date to and from julian day
299  * Uses Fliegel & Van Flandern algorithm
300  */
301 static long date_to_julian(int y, int m, int d)
302 {
303         return (1461 * (y + 4800 + (m - 14) / 12)) / 4 +
304                 (367 * (m - 2 - 12 * ((m - 14) / 12))) / 12 -
305                 (3 * ((y + 4900 + (m - 14) / 12) / 100)) / 4 +
306                 d - 32075;
307 }

309 static void julian_to_date(long jd, int *y, int *m, int *d)
310         {
311         long  L = jd + 68569;
312         long  n = (4 * L) / 146097;
313         long  i, j;

315         L = L - (146097 * n + 3) / 4;
316         i = (4000 * (L + 1)) / 1461001;
317         L = L - (1461 * i) / 4 + 31;
318         j = (80 * L) / 2447;
319         *d = L - (2447 * j) / 80;
320         L = j / 11;
321         *m = j + 2 - (12 * L);
322         *y = 100 * (n - 49) + i + L;
323         }

325 #ifdef OPENSSL_TIME_TEST
```



```
327 #include <stdio.h>

329 /* Time checking test code. Check times are identical for a wide range of
330  * offsets. This should be run on a machine with 64 bit time_t or it will
331  * trigger the very errors the routines fix.
332  */

334 int main(int argc, char **argv)
335         {
336         long offset;
337         for (offset = 0; offset < 1000000; offset++)
338                 {
339                 check_time(offset);
340                 check_time(-offset);
341                 check_time(offset * 1000);
342                 check_time(-offset * 1000);
343                 }
344         }

346 int check_time(long offset)
347         {
348         struct tm tm1, tm2;
349         time_t t1, t2;
350         time(&t1);
351         t2 = t1 + offset;
352         OPENSSL_gmtime(&t2, &tm2);
353         OPENSSL_gmtime(&t1, &tm1);
354         OPENSSL_gmtime_adj(&tm1, 0, offset);
355         if ((tm1.tm_year == tm2.tm_year) &&
356             (tm1.tm_mon == tm2.tm_mon) &&
357             (tm1.tm_mday == tm2.tm_mday) &&
358             (tm1.tm_hour == tm2.tm_hour) &&
359             (tm1.tm_min == tm2.tm_min) &&
360             (tm1.tm_sec == tm2.tm_sec))
361                 return 1;
362         fprintf(stderr, "TIME ERROR!!\n");
363         fprintf(stderr, "Time1: %d/%d/%d, %d:%02d:%02d\n",
364                         tm2.tm_mday, tm2.tm_mon + 1, tm2.tm_year + 1900,
365                         tm2.tm_hour, tm2.tm_min, tm2.tm_sec);
366         fprintf(stderr, "Time2: %d/%d/%d, %d:%02d:%02d\n",
367                         tm1.tm_mday, tm1.tm_mon + 1, tm1.tm_year + 1900,
368                         tm1.tm_hour, tm1.tm_min, tm1.tm_sec);
369         return 0;
370         }

372 #endif
373 #endif /* ! codereview */
```

```
**********************************************************
    8303 Wed Aug 13 19:52:56 2014
new/usr/src/lib/openssl/libsunw_crypto/objects/o_names.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <openssl/err.h>
#include <openssl/lhash.h>
#include <openssl/objects.h>
#include <openssl/safestack.h>
#include <openssl/e_os2.h>

/* Later versions of DEC C has started to add lnkage information to certain
 * functions, which makes it tricky to use them as values to regular function
 * pointers.  One way is to define a macro that takes care of casting them
 * correctly.
 */
#ifdef OPENSSL_SYS_VMS_DECC
# define OPENSSL_strcmp (int (*)(const char *,const char *))strcmp
#else
# define OPENSSL_strcmp strcmp
#endif

/* I use the ex_data stuff to manage the identifiers for the obj_name_types
 * that applications may define.  I only really use the free function field.
 */
DECLARE_LHASH_OF(OBJ_NAME);
static LHASH_OF(OBJ_NAME) *names_lh=NULL;
static int names_type_num=OBJ_NAME_TYPE_NUM;

typedef struct name_funcs_st
        {
        unsigned long (*hash_func)(const char *name);
        int (*cmp_func)(const char *a,const char *b);
        void (*free_func)(const char *, int, const char *);
        } NAME_FUNCS;

DECLARE_STACK_OF(NAME_FUNCS)
IMPLEMENT_STACK_OF(NAME_FUNCS)

static STACK_OF(NAME_FUNCS) *name_funcs_stack;

/* The LHASH callbacks now use the raw "void *" prototypes and do per-variable
 * casting in the functions. This prevents function pointer casting without the
 * need for macro-generated wrapper functions. */

/* static unsigned long obj_name_hash(OBJ_NAME *a); */
static unsigned long obj_name_hash(const void *a_void);
/* static int obj_name_cmp(OBJ_NAME *a,OBJ_NAME *b); */
static int obj_name_cmp(const void *a_void,const void *b_void);

static IMPLEMENT_LHASH_HASH_FN(obj_name, OBJ_NAME)
static IMPLEMENT_LHASH_COMP_FN(obj_name, OBJ_NAME)

int OBJ_NAME_init(void)
        {
        if (names_lh != NULL) return(1);
        MemCheck_off();
        names_lh=lh_OBJ_NAME_new();
        MemCheck_on();
        return(names_lh != NULL);
        }
```



```
int OBJ_NAME_new_index(unsigned long (*hash_func)(const char *),
        int (*cmp_func)(const char *, const char *),
        void (*free_func)(const char *, int, const char *))
        {
        int ret;
        int i;
        NAME_FUNCS *name_funcs;

        if (name_funcs_stack == NULL)
                {
                MemCheck_off();
                name_funcs_stack=sk_NAME_FUNCS_new_null();
                MemCheck_on();
                }
        if (name_funcs_stack == NULL)
                {
                /* ERROR */
                return(0);
                }
        ret=names_type_num;
        names_type_num++;
        for (i=sk_NAME_FUNCS_num(name_funcs_stack); i<names_type_num; i++)
                {
                MemCheck_off();
                name_funcs = OPENSSL_malloc(sizeof(NAME_FUNCS));
                MemCheck_on();
                if (!name_funcs)
                        {
                        OBJerr(OBJ_F_OBJ_NAME_NEW_INDEX,ERR_R_MALLOC_FAILURE);
                        return(0);
                        }
                name_funcs->hash_func = lh_strhash;
                name_funcs->cmp_func = OPENSSL_strcmp;
                name_funcs->free_func = 0; /* NULL is often declared to
                                            * ((void *)0), which according
                                            * to Compaq C is not really
                                            * compatible with a function
                                            * pointer.      -- Richard Levit
                MemCheck_off();
                sk_NAME_FUNCS_push(name_funcs_stack,name_funcs);
                MemCheck_on();
                }
        name_funcs = sk_NAME_FUNCS_value(name_funcs_stack, ret);
        if (hash_func != NULL)
                name_funcs->hash_func = hash_func;
        if (cmp_func != NULL)
                name_funcs->cmp_func = cmp_func;
        if (free_func != NULL)
                name_funcs->free_func = free_func;
        return(ret);
        }

/* static int obj_name_cmp(OBJ_NAME *a, OBJ_NAME *b) */
static int obj_name_cmp(const void *a_void, const void *b_void)
        {
        int ret;
        const OBJ_NAME *a = (const OBJ_NAME *)a_void;
        const OBJ_NAME *b = (const OBJ_NAME *)b_void;

        ret=a->type-b->type;
        if (ret == 0)
                {
                if ((name_funcs_stack != NULL)
                        && (sk_NAME_FUNCS_num(name_funcs_stack) > a->type))
                        {
                        ret=sk_NAME_FUNCS_value(name_funcs_stack,
```

```
128                                 a->type)->cmp_func(a->name,b->name);
129                         }
130                 else
131                         ret=strcmp(a->name,b->name);
132                 }
133         return(ret);
134         }

136 /* static unsigned long obj_name_hash(OBJ_NAME *a) */
137 static unsigned long obj_name_hash(const void *a_void)
138         {
139         unsigned long ret;
140         const OBJ_NAME *a = (const OBJ_NAME *)a_void;

142         if ((name_funcs_stack != NULL) && (sk_NAME_FUNCS_num(name_funcs_stack) >
143                 {
144                 ret=sk_NAME_FUNCS_value(name_funcs_stack,
145                         a->type)->hash_func(a->name);
146                 }
147         else
148                 {
149                 ret=lh_strhash(a->name);
150                 }
151         ret^=a->type;
152         return(ret);
153         }

155 const char *OBJ_NAME_get(const char *name, int type)
156         {
157         OBJ_NAME on,*ret;
158         int num=0,alias;

160         if (name == NULL) return(NULL);
161         if ((names_lh == NULL) && !OBJ_NAME_init()) return(NULL);

163         alias=type&OBJ_NAME_ALIAS;
164         type&= ~OBJ_NAME_ALIAS;

166         on.name=name;
167         on.type=type;

169         for (;;)
170         {
171                 ret=lh_OBJ_NAME_retrieve(names_lh,&on);
172                 if (ret == NULL) return(NULL);
173                 if ((ret->alias) && !alias)
174                         {
175                         if (++num > 10) return(NULL);
176                         on.name=ret->data;
177                         }
178                 else
179                         {
180                         return(ret->data);
181                         }
182                 }
183         }

185 int OBJ_NAME_add(const char *name, int type, const char *data)
186         {
187         OBJ_NAME *onp,*ret;
188         int alias;

190         if ((names_lh == NULL) && !OBJ_NAME_init()) return(0);

192         alias=type&OBJ_NAME_ALIAS;
193         type&= ~OBJ_NAME_ALIAS;
```



```
195         onp=(OBJ_NAME *)OPENSSL_malloc(sizeof(OBJ_NAME));
196         if (onp == NULL)
197                 {
198                 /* ERROR */
199                 return(0);
200                 }

202         onp->name=name;
203         onp->alias=alias;
204         onp->type=type;
205         onp->data=data;

207         ret=lh_OBJ_NAME_insert(names_lh,onp);
208         if (ret != NULL)
209                 {
210                 /* free things */
211                 if ((name_funcs_stack != NULL) && (sk_NAME_FUNCS_num(name_funcs_
212                         {
213                         /* XXX: I'm not sure I understand why the free
214                          * function should get three arguments...
215                          * -- Richard Levitte
216                          */
217                         sk_NAME_FUNCS_value(name_funcs_stack,
218                                 ret->type)->free_func(ret->name,ret->type,ret->d
219                         }
220                 OPENSSL_free(ret);
221                 }
222         else
223                 {
224                 if (lh_OBJ_NAME_error(names_lh))
225                         {
226                         /* ERROR */
227                         return(0);
228                         }
229                 }
230         return(1);
231         }

233 int OBJ_NAME_remove(const char *name, int type)
234         {
235         OBJ_NAME on,*ret;

237         if (names_lh == NULL) return(0);

239         type&= ~OBJ_NAME_ALIAS;
240         on.name=name;
241         on.type=type;
242         ret=lh_OBJ_NAME_delete(names_lh,&on);
243         if (ret != NULL)
244                 {
245                 /* free things */
246                 if ((name_funcs_stack != NULL) && (sk_NAME_FUNCS_num(name_funcs_
247                         {
248                         /* XXX: I'm not sure I understand why the free
249                          * function should get three arguments...
250                          * -- Richard Levitte
251                          */
252                         sk_NAME_FUNCS_value(name_funcs_stack,
253                                 ret->type)->free_func(ret->name,ret->type,ret->d
254                         }
255                 OPENSSL_free(ret);
256                 return(1);
257                 }
258         else
259                 return(0);
```

```
260         }

262 struct doall
263         {
264         int type;
265         void (*fn)(const OBJ_NAME *,void *arg);
266         void *arg;
267         };

269 static void do_all_fn_doall_arg(const OBJ_NAME *name,struct doall *d)
270         {
271         if(name->type == d->type)
272                 d->fn(name,d->arg);
273         }

275 static IMPLEMENT_LHASH_DOALL_ARG_FN(do_all_fn, const OBJ_NAME, struct doall)

277 void OBJ_NAME_do_all(int type,void (*fn)(const OBJ_NAME *,void *arg),void *arg)
278         {
279         struct doall d;

281         d.type=type;
282         d.fn=fn;
283         d.arg=arg;

285         lh_OBJ_NAME_doall_arg(names_lh, LHASH_DOALL_ARG_FN(do_all_fn),
286                               struct doall, &d);
287         }

289 struct doall_sorted
290         {
291         int type;
292         int n;
293         const OBJ_NAME **names;
294         };

296 static void do_all_sorted_fn(const OBJ_NAME *name,void *d_)
297         {
298         struct doall_sorted *d=d_;

300         if(name->type != d->type)
301                 return;

303         d->names[d->n++]=name;
304         }

306 static int do_all_sorted_cmp(const void *n1_,const void *n2_)
307         {
308         const OBJ_NAME * const *n1=n1_;
309         const OBJ_NAME * const *n2=n2_;

311         return strcmp((*n1)->name,(*n2)->name);
312         }

314 void OBJ_NAME_do_all_sorted(int type,void (*fn)(const OBJ_NAME *,void *arg),
315                             void *arg)
316         {
317         struct doall_sorted d;
318         int n;

320         d.type=type;
321         d.names=OPENSSL_malloc(lh_OBJ_NAME_num_items(names_lh)*sizeof *d.names);
322         d.n=0;
323         OBJ_NAME_do_all(type,do_all_sorted_fn,&d);

325         qsort((void *)d.names,d.n,sizeof *d.names,do_all_sorted_cmp);
```



```
327         for(n=0 ; n < d.n ; ++n)
328                 fn(d.names[n],arg);

330         OPENSSL_free((void *)d.names);
331         }

333 static int free_type;

335 static void names_lh_free_doall(OBJ_NAME *onp)
336         {
337         if (onp == NULL)
338                 return;

340         if (free_type < 0 || free_type == onp->type)
341                 OBJ_NAME_remove(onp->name,onp->type);
342         }

344 static IMPLEMENT_LHASH_DOALL_FN(names_lh_free, OBJ_NAME)

346 static void name_funcs_free(NAME_FUNCS *ptr)
347         {
348         OPENSSL_free(ptr);
349         }

351 void OBJ_NAME_cleanup(int type)
352         {
353         unsigned long down_load;

355         if (names_lh == NULL) return;

357         free_type=type;
358         down_load=lh_OBJ_NAME_down_load(names_lh);
359         lh_OBJ_NAME_down_load(names_lh)=0;

361         lh_OBJ_NAME_doall(names_lh,LHASH_DOALL_FN(names_lh_free));
362         if (type < 0)
363                 {
364                 lh_OBJ_NAME_free(names_lh);
365                 sk_NAME_FUNCS_pop_free(name_funcs_stack,name_funcs_free);
366                 names_lh=NULL;
367                 name_funcs_stack = NULL;
368                 }
369         else
370                 lh_OBJ_NAME_down_load(names_lh)=down_load;
371         }
372 #endif /* ! codereview */
```

```
**********************************************************
   18076 Wed Aug 13 19:52:56 2014
new/usr/src/lib/openssl/libsunw_crypto/objects/obj_dat.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/objects/obj_dat.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <ctype.h>
#include <limits.h>
```



```
#include "cryptlib.h"
#include <openssl/lhash.h>
#include <openssl/asn1.h>
#include <openssl/objects.h>
#include <openssl/bn.h>

/* obj_dat.h is generated from objects.h by obj_dat.pl */
#ifndef OPENSSL_NO_OBJECT
#include "obj_dat.h"
#else
/* You will have to load all the objects needed manually in the application */
#define NUM_NID 0
#define NUM_SN 0
#define NUM_LN 0
#define NUM_OBJ 0
static const unsigned char lvalues[1];
static const ASN1_OBJECT nid_objs[1];
static const unsigned int sn_objs[1];
static const unsigned int ln_objs[1];
static const unsigned int obj_objs[1];
#endif

DECLARE_OBJ_BSEARCH_CMP_FN(const ASN1_OBJECT *, unsigned int, sn);
DECLARE_OBJ_BSEARCH_CMP_FN(const ASN1_OBJECT *, unsigned int, ln);
DECLARE_OBJ_BSEARCH_CMP_FN(const ASN1_OBJECT *, unsigned int, obj);

#define ADDED_DATA      0
#define ADDED_SNAME     1
#define ADDED_LNAME     2
#define ADDED_NID       3

typedef struct added_obj_st
        {
        int type;
        ASN1_OBJECT *obj;
        } ADDED_OBJ;
DECLARE_LHASH_OF(ADDED_OBJ);

static int new_nid=NUM_NID;
static LHASH_OF(ADDED_OBJ) *added=NULL;

static int sn_cmp(const ASN1_OBJECT * const *a, const unsigned int *b)
        { return(strcmp((*a)->sn,nid_objs[*b].sn)); }

IMPLEMENT_OBJ_BSEARCH_CMP_FN(const ASN1_OBJECT *, unsigned int, sn);

static int ln_cmp(const ASN1_OBJECT * const *a, const unsigned int *b)
        { return(strcmp((*a)->ln,nid_objs[*b].ln)); }

IMPLEMENT_OBJ_BSEARCH_CMP_FN(const ASN1_OBJECT *, unsigned int, ln);

static unsigned long added_obj_hash(const ADDED_OBJ *ca)
        {
        const ASN1_OBJECT *a;
        int i;
        unsigned long ret=0;
        unsigned char *p;

        a=ca->obj;
        switch (ca->type)
                {
        case ADDED_DATA:
                ret=a->length<<20L;
                p=(unsigned char *)a->data;
                for (i=0; i<a->length; i++)
                        ret^=p[i]<<((i*3)%24);
```

```
128                 break;
129         case ADDED_SNAME:
130                 ret=lh_strhash(a->sn);
131                 break;
132         case ADDED_LNAME:
133                 ret=lh_strhash(a->ln);
134                 break;
135         case ADDED_NID:
136                 ret=a->nid;
137                 break;
138         default:
139                 /* abort(); */
140                 return 0;
141                 }
142         ret&=0x3fffffffL;
143         ret|=ca->type<<30L;
144         return(ret);
145         }
146 static IMPLEMENT_LHASH_HASH_FN(added_obj, ADDED_OBJ)

148 static int added_obj_cmp(const ADDED_OBJ *ca, const ADDED_OBJ *cb)
149         {
150         ASN1_OBJECT *a,*b;
151         int i;

153         i=ca->type-cb->type;
154         if (i) return(i);
155         a=ca->obj;
156         b=cb->obj;
157         switch (ca->type)
158                 {
159         case ADDED_DATA:
160                 i=(a->length - b->length);
161                 if (i) return(i);
162                 return(memcmp(a->data,b->data,(size_t)a->length));
163         case ADDED_SNAME:
164                 if (a->sn == NULL) return(-1);
165                 else if (b->sn == NULL) return(1);
166                 else return(strcmp(a->sn,b->sn));
167         case ADDED_LNAME:
168                 if (a->ln == NULL) return(-1);
169                 else if (b->ln == NULL) return(1);
170                 else return(strcmp(a->ln,b->ln));
171         case ADDED_NID:
172                 return(a->nid-b->nid);
173         default:
174                 /* abort(); */
175                 return 0;
176                 }
177         }
178 static IMPLEMENT_LHASH_COMP_FN(added_obj, ADDED_OBJ)

180 static int init_added(void)
181         {
182         if (added != NULL) return(1);
183         added=lh_ADDED_OBJ_new();
184         return(added != NULL);
185         }

187 static void cleanup1_doall(ADDED_OBJ *a)
188         {
189         a->obj->nid=0;
190         a->obj->flags|=ASN1_OBJECT_FLAG_DYNAMIC|
191                        ASN1_OBJECT_FLAG_DYNAMIC_STRINGS|
192                        ASN1_OBJECT_FLAG_DYNAMIC_DATA;
193         }
```



```
195 static void cleanup2_doall(ADDED_OBJ *a)
196         { a->obj->nid++; }

198 static void cleanup3_doall(ADDED_OBJ *a)
199         {
200         if (--a->obj->nid == 0)
201                 ASN1_OBJECT_free(a->obj);
202         OPENSSL_free(a);
203         }

205 static IMPLEMENT_LHASH_DOALL_FN(cleanup1, ADDED_OBJ)
206 static IMPLEMENT_LHASH_DOALL_FN(cleanup2, ADDED_OBJ)
207 static IMPLEMENT_LHASH_DOALL_FN(cleanup3, ADDED_OBJ)

209 /* The purpose of obj_cleanup_defer is to avoid EVP_cleanup() attempting
210  * to use freed up OIDs. If neccessary the actual freeing up of OIDs is
211  * delayed.
212  */

214 int obj_cleanup_defer = 0;

216 void check_defer(int nid)
217         {
218         if (!obj_cleanup_defer && nid >= NUM_NID)
219                         obj_cleanup_defer = 1;
220         }

222 void OBJ_cleanup(void)
223         {
224         if (obj_cleanup_defer)
225                 {
226                 obj_cleanup_defer = 2;
227                 return ;
228                 }
229         if (added == NULL) return;
230         lh_ADDED_OBJ_down_load(added) = 0;
231         lh_ADDED_OBJ_doall(added,LHASH_DOALL_FN(cleanup1)); /* zero counters */
232         lh_ADDED_OBJ_doall(added,LHASH_DOALL_FN(cleanup2)); /* set counters */
233         lh_ADDED_OBJ_doall(added,LHASH_DOALL_FN(cleanup3)); /* free objects */
234         lh_ADDED_OBJ_free(added);
235         added=NULL;
236         }

238 int OBJ_new_nid(int num)
239         {
240         int i;

242         i=new_nid;
243         new_nid+=num;
244         return(i);
245         }

247 int OBJ_add_object(const ASN1_OBJECT *obj)
248         {
249         ASN1_OBJECT *o;
250         ADDED_OBJ *ao[4]={NULL,NULL,NULL,NULL},*aop;
251         int i;

253         if (added == NULL)
254                 if (!init_added()) return(0);
255         if ((o=OBJ_dup(obj)) == NULL) goto err;
256         if (!(ao[ADDED_NID]=(ADDED_OBJ *)OPENSSL_malloc(sizeof(ADDED_OBJ)))) got
257         if ((o->length != 0) && (obj->data != NULL))
258                 if (!(ao[ADDED_DATA]=(ADDED_OBJ *)OPENSSL_malloc(sizeof(ADDED_OB
259         if (o->sn != NULL)
```

```
260                 if (!(ao[ADDED_SNAME]=(ADDED_OBJ *)OPENSSL_malloc(sizeof(ADDED_O
261         if (o->ln != NULL)
262                 if (!(ao[ADDED_LNAME]=(ADDED_OBJ *)OPENSSL_malloc(sizeof(ADDED_O

264         for (i=ADDED_DATA; i<=ADDED_NID; i++)
265                 {
266                 if (ao[i] != NULL)
267                         {
268                         ao[i]->type=i;
269                         ao[i]->obj=o;
270                         aop=lh_ADDED_OBJ_insert(added,ao[i]);
271                         /* memory leak, buit should not normally matter */
272                         if (aop != NULL)
273                                 OPENSSL_free(aop);
274                         }
275                 }
276         o->flags&= ~(ASN1_OBJECT_FLAG_DYNAMIC|ASN1_OBJECT_FLAG_DYNAMIC_STRINGS|
277                         ASN1_OBJECT_FLAG_DYNAMIC_DATA);

279         return(o->nid);
280 err2:
281         OBJerr(OBJ_F_OBJ_ADD_OBJECT,ERR_R_MALLOC_FAILURE);
282 err:
283         for (i=ADDED_DATA; i<=ADDED_NID; i++)
284                 if (ao[i] != NULL) OPENSSL_free(ao[i]);
285         if (o != NULL) OPENSSL_free(o);
286         return(NID_undef);
287         }

289 ASN1_OBJECT *OBJ_nid2obj(int n)
290         {
291         ADDED_OBJ ad,*adp;
292         ASN1_OBJECT ob;

294         if ((n >= 0) && (n < NUM_NID))
295                 {
296                 if ((n != NID_undef) && (nid_objs[n].nid == NID_undef))
297                         {
298                         OBJerr(OBJ_F_OBJ_NID2OBJ,OBJ_R_UNKNOWN_NID);
299                         return(NULL);
300                         }
301                 return((ASN1_OBJECT *)&(nid_objs[n]));
302                 }
303         else if (added == NULL)
304                 return(NULL);
305         else
306                 {
307                 ad.type=ADDED_NID;
308                 ad.obj= &ob;
309                 ob.nid=n;
310                 adp=lh_ADDED_OBJ_retrieve(added,&ad);
311                 if (adp != NULL)
312                         return(adp->obj);
313                 else
314                         {
315                         OBJerr(OBJ_F_OBJ_NID2OBJ,OBJ_R_UNKNOWN_NID);
316                         return(NULL);
317                         }
318                 }
319         }

321 const char *OBJ_nid2sn(int n)
322         {
323         ADDED_OBJ ad,*adp;
324         ASN1_OBJECT ob;
```



```
326         if ((n >= 0) && (n < NUM_NID))
327                 {
328                 if ((n != NID_undef) && (nid_objs[n].nid == NID_undef))
329                         {
330                         OBJerr(OBJ_F_OBJ_NID2SN,OBJ_R_UNKNOWN_NID);
331                         return(NULL);
332                         }
333                 return(nid_objs[n].sn);
334                 }
335         else if (added == NULL)
336                 return(NULL);
337         else
338                 {
339                 ad.type=ADDED_NID;
340                 ad.obj= &ob;
341                 ob.nid=n;
342                 adp=lh_ADDED_OBJ_retrieve(added,&ad);
343                 if (adp != NULL)
344                         return(adp->obj->sn);
345                 else
346                         {
347                         OBJerr(OBJ_F_OBJ_NID2SN,OBJ_R_UNKNOWN_NID);
348                         return(NULL);
349                         }
350                 }
351         }

353 const char *OBJ_nid2ln(int n)
354         {
355         ADDED_OBJ ad,*adp;
356         ASN1_OBJECT ob;

358         if ((n >= 0) && (n < NUM_NID))
359                 {
360                 if ((n != NID_undef) && (nid_objs[n].nid == NID_undef))
361                         {
362                         OBJerr(OBJ_F_OBJ_NID2LN,OBJ_R_UNKNOWN_NID);
363                         return(NULL);
364                         }
365                 return(nid_objs[n].ln);
366                 }
367         else if (added == NULL)
368                 return(NULL);
369         else
370                 {
371                 ad.type=ADDED_NID;
372                 ad.obj= &ob;
373                 ob.nid=n;
374                 adp=lh_ADDED_OBJ_retrieve(added,&ad);
375                 if (adp != NULL)
376                         return(adp->obj->ln);
377                 else
378                         {
379                         OBJerr(OBJ_F_OBJ_NID2LN,OBJ_R_UNKNOWN_NID);
380                         return(NULL);
381                         }
382                 }
383         }

385 static int obj_cmp(const ASN1_OBJECT * const *ap, const unsigned int *bp)
386         {
387         int j;
388         const ASN1_OBJECT *a= *ap;
389         const ASN1_OBJECT *b= &nid_objs[*bp];

391         j=(a->length - b->length);
```

```
392         if (j) return(j);
393         return(memcmp(a->data,b->data,a->length));
394         }

396 IMPLEMENT_OBJ_BSEARCH_CMP_FN(const ASN1_OBJECT *, unsigned int, obj);

398 int OBJ_obj2nid(const ASN1_OBJECT *a)
399         {
400         const unsigned int *op;
401         ADDED_OBJ ad,*adp;

403         if (a == NULL)
404                 return(NID_undef);
405         if (a->nid != 0)
406                 return(a->nid);

408         if (added != NULL)
409                 {
410                 ad.type=ADDED_DATA;
411                 ad.obj=(ASN1_OBJECT *)a; /* XXX: ugly but harmless */
412                 adp=lh_ADDED_OBJ_retrieve(added,&ad);
413                 if (adp != NULL) return (adp->obj->nid);
414                 }
415         op=OBJ_bsearch_obj(&a, obj_objs, NUM_OBJ);
416         if (op == NULL)
417                 return(NID_undef);
418         return(nid_objs[*op].nid);
419         }

421 /* Convert an object name into an ASN1_OBJECT
422  * if "noname" is not set then search for short and long names first.
423  * This will convert the "dotted" form into an object: unlike OBJ_txt2nid
424  * it can be used with any objects, not just registered ones.
425  */

427 ASN1_OBJECT *OBJ_txt2obj(const char *s, int no_name)
428         {
429         int nid = NID_undef;
430         ASN1_OBJECT *op=NULL;
431         unsigned char *buf;
432         unsigned char *p;
433         const unsigned char *cp;
434         int i, j;

436         if(!no_name) {
437                 if( ((nid = OBJ_sn2nid(s)) != NID_undef) ||
438                         ((nid = OBJ_ln2nid(s)) != NID_undef) )
439                                         return OBJ_nid2obj(nid);
440         }

442         /* Work out size of content octets */
443         i=a2d_ASN1_OBJECT(NULL,0,s,-1);
444         if (i <= 0) {
445                 /* Don't clear the error */
446                 /*ERR_clear_error();*/
447                 return NULL;
448         }
449         /* Work out total size */
450         j = ASN1_object_size(0,i,V_ASN1_OBJECT);

452         if((buf=(unsigned char *)OPENSSL_malloc(j)) == NULL) return NULL;

454         p = buf;
455         /* Write out tag+length */
456         ASN1_put_object(&p,0,i,V_ASN1_OBJECT,V_ASN1_UNIVERSAL);
457         /* Write out contents */
```



```
458         a2d_ASN1_OBJECT(p,i,s,-1);

460         cp=buf;
461         op=d2i_ASN1_OBJECT(NULL,&cp,j);
462         OPENSSL_free(buf);
463         return op;
464         }

466 int OBJ_obj2txt(char *buf, int buf_len, const ASN1_OBJECT *a, int no_name)
467 {
468         int i,n=0,len,nid, first, use_bn;
469         BIGNUM *bl;
470         unsigned long l;
471         const unsigned char *p;
472         char tbuf[DECIMAL_SIZE(i)+DECIMAL_SIZE(l)+2];

474         /* Ensure that, at every state, |buf| is NUL-terminated. */
475         if (buf && buf_len > 0)
476                 buf[0]='\0';

478         if ((a == NULL) || (a->data == NULL))
479                 return(0);

481         if (!no_name && (nid=OBJ_obj2nid(a)) != NID_undef)
482                 {
483                 const char *s;
484                 s=OBJ_nid2ln(nid);
485                 if (s == NULL)
486                         s=OBJ_nid2sn(nid);
487                 if (s)
488                         {
489                         if (buf)
490                                 BUF_strlcpy(buf,s,buf_len);
491                         n=strlen(s);
492                         return n;
493                         }
494                 }


497         len=a->length;
498         p=a->data;

500         first = 1;
501         bl = NULL;

503         while (len > 0)
504                 {
505                 l=0;
506                 use_bn = 0;
507                 for (;;)
508                         {
509                         unsigned char c = *p++;
510                         len--;
511                         if ((len == 0) && (c & 0x80))
512                                 goto err;
513                         if (use_bn)
514                                 {
515                                 if (!BN_add_word(bl, c & 0x7f))
516                                         goto err;
517                                 }
518                         else
519                                 l |= c  & 0x7f;
520                         if (!(c & 0x80))
521                                 break;
522                         if (!use_bn && (l > (ULONG_MAX >> 7L)))
523                                 {
```

```
524                                 if (!bl && !(bl = BN_new()))
525                                         goto err;
526                                 if (!BN_set_word(bl, l))
527                                         goto err;
528                                 use_bn = 1;
529                                 }
530                         if (use_bn)
531                                 {
532                                 if (!BN_lshift(bl, bl, 7))
533                                         goto err;
534                                 }
535                         else
536                                 l<<=7L;
537                         }

539                 if (first)
540                         {
541                         first = 0;
542                         if (l >= 80)
543                                 {
544                                 i = 2;
545                                 if (use_bn)
546                                         {
547                                         if (!BN_sub_word(bl, 80))
548                                                 goto err;
549                                         }
550                                 else
551                                         l -= 80;
552                                 }
553                         else
554                                 {
555                                 i=(int)(l/40);
556                                 l-=(long)(i*40);
557                                 }
558                         if (buf && (buf_len > 1))
559                                 {
560                                 *buf++ = i + '0';
561                                 *buf = '\0';
562                                 buf_len--;
563                                 }
564                         n++;
565                         }

567                 if (use_bn)
568                         {
569                         char *bndec;
570                         bndec = BN_bn2dec(bl);
571                         if (!bndec)
572                                 goto err;
573                         i = strlen(bndec);
574                         if (buf)
575                                 {
576                                 if (buf_len > 1)
577                                         {
578                                         *buf++ = '.';
579                                         *buf = '\0';
580                                         buf_len--;
581                                         }
582                                 BUF_strlcpy(buf,bndec,buf_len);
583                                 if (i > buf_len)
584                                         {
585                                         buf += buf_len;
586                                         buf_len = 0;
587                                         }
588                                 else
589                                         {
```



```
590                                         buf+=i;
591                                         buf_len-=i;
592                                         }
593                                 }
594                         n++;
595                         n += i;
596                         OPENSSL_free(bndec);
597                         }
598                 else
599                         {
600                         BIO_snprintf(tbuf,sizeof tbuf,".%lu",l);
601                         i=strlen(tbuf);
602                         if (buf && (buf_len > 0))
603                                 {
604                                 BUF_strlcpy(buf,tbuf,buf_len);
605                                 if (i > buf_len)
606                                         {
607                                         buf += buf_len;
608                                         buf_len = 0;
609                                         }
610                                 else
611                                         {
612                                         buf+=i;
613                                         buf_len-=i;
614                                         }
615                                 }
616                         n+=i;
617                         l=0;
618                         }
619                 }

621         if (bl)
622                 BN_free(bl);
623         return n;

625         err:
626         if (bl)
627                 BN_free(bl);
628         return -1;
629 }

631 int OBJ_txt2nid(const char *s)
632 {
633         ASN1_OBJECT *obj;
634         int nid;
635         obj = OBJ_txt2obj(s, 0);
636         nid = OBJ_obj2nid(obj);
637         ASN1_OBJECT_free(obj);
638         return nid;
639 }

641 int OBJ_ln2nid(const char *s)
642         {
643         ASN1_OBJECT o;
644         const ASN1_OBJECT *oo= &o;
645         ADDED_OBJ ad,*adp;
646         const unsigned int *op;

648         o.ln=s;
649         if (added != NULL)
650                 {
651                 ad.type=ADDED_LNAME;
652                 ad.obj= &o;
653                 adp=lh_ADDED_OBJ_retrieve(added,&ad);
654                 if (adp != NULL) return (adp->obj->nid);
655                 }
```

```
656         op=OBJ_bsearch_ln(&oo, ln_objs, NUM_LN);
657         if (op == NULL) return(NID_undef);
658         return(nid_objs[*op].nid);
659         }

661 int OBJ_sn2nid(const char *s)
662         {
663         ASN1_OBJECT o;
664         const ASN1_OBJECT *oo= &o;
665         ADDED_OBJ ad,*adp;
666         const unsigned int *op;

668         o.sn=s;
669         if (added != NULL)
670                 {
671                 ad.type=ADDED_SNAME;
672                 ad.obj= &o;
673                 adp=lh_ADDED_OBJ_retrieve(added,&ad);
674                 if (adp != NULL) return (adp->obj->nid);
675                 }
676         op=OBJ_bsearch_sn(&oo, sn_objs, NUM_SN);
677         if (op == NULL) return(NID_undef);
678         return(nid_objs[*op].nid);
679         }

681 const void *OBJ_bsearch_(const void *key, const void *base, int num, int size,
682                          int (*cmp)(const void *, const void *))
683         {
684         return OBJ_bsearch_ex_(key, base, num, size, cmp, 0);
685         }

687 const void *OBJ_bsearch_ex_(const void *key, const void *base_, int num,
688                             int size,
689                             int (*cmp)(const void *, const void *),
690                             int flags)
691         {
692         const char *base=base_;
693         int l,h,i=0,c=0;
694         const char *p = NULL;

696         if (num == 0) return(NULL);
697         l=0;
698         h=num;
699         while (l < h)
700                 {
701                 i=(l+h)/2;
702                 p= &(base[i*size]);
703                 c=(*cmp)(key,p);
704                 if (c < 0)
705                         h=i;
706                 else if (c > 0)
707                         l=i+1;
708                 else
709                         break;
710                 }
711 #ifdef CHARSET_EBCDIC
712 /* THIS IS A KLUDGE - Because the *_obj is sorted in ASCII order, and
713  * I don't have perl (yet), we revert to a *LINEAR* search
714  * when the object wasn't found in the binary search.
715  */
716         if (c != 0)
717                 {
718                 for (i=0; i<num; ++i)
719                         {
720                         p= &(base[i*size]);
721                         c = (*cmp)(key,p);
```



```
722                         if (c == 0 || (c < 0 && (flags & OBJ_BSEARCH_VALUE_ON_NO
723                                 return p;
724                         }
725                 }
726 #endif
727         if (c != 0 && !(flags & OBJ_BSEARCH_VALUE_ON_NOMATCH))
728                 p = NULL;
729         else if (c == 0 && (flags & OBJ_BSEARCH_FIRST_VALUE_ON_MATCH))
730                 {
731                 while(i > 0 && (*cmp)(key,&(base[(i-1)*size])) == 0)
732                         i--;
733                 p = &(base[i*size]);
734                 }
735         return(p);
736         }

738 int OBJ_create_objects(BIO *in)
739         {
740         MS_STATIC char buf[512];
741         int i,num=0;
742         char *o,*s,*l=NULL;

744         for (;;)
745                 {
746                 s=o=NULL;
747                 i=BIO_gets(in,buf,512);
748                 if (i <= 0) return(num);
749                 buf[i-1]='\0';
750                 if (!isalnum((unsigned char)buf[0])) return(num);
751                 o=s=buf;
752                 while (isdigit((unsigned char)*s) || (*s == '.'))
753                         s++;
754                 if (*s != '\0')
755                         {
756                         *(s++)='\0';
757                         while (isspace((unsigned char)*s))
758                                 s++;
759                         if (*s == '\0')
760                                 s=NULL;
761                         else
762                                 {
763                                 l=s;
764                                 while ((*l != '\0') && !isspace((unsigned char)*
765                                         l++;
766                                 if (*l != '\0')
767                                         {
768                                         *(l++)='\0';
769                                         while (isspace((unsigned char)*l))
770                                                 l++;
771                                         if (*l == '\0') l=NULL;
772                                         }
773                                 else
774                                         l=NULL;
775                                 }
776                         }
777                 else
778                         s=NULL;
779                 if ((o == NULL) || (*o == '\0')) return(num);
780                 if (!OBJ_create(o,s,l)) return(num);
781                 num++;
782                 }
783         /* return(num); */
784         }

786 int OBJ_create(const char *oid, const char *sn, const char *ln)
787         {
```

```
788         int ok=0;
789         ASN1_OBJECT *op=NULL;
790         unsigned char *buf;
791         int i;

793         i=a2d_ASN1_OBJECT(NULL,0,oid,-1);
794         if (i <= 0) return(0);

796         if ((buf=(unsigned char *)OPENSSL_malloc(i)) == NULL)
797                 {
798                 OBJerr(OBJ_F_OBJ_CREATE,ERR_R_MALLOC_FAILURE);
799                 return(0);
800                 }
801         i=a2d_ASN1_OBJECT(buf,i,oid,-1);
802         if (i == 0)
803                 goto err;
804         op=(ASN1_OBJECT *)ASN1_OBJECT_create(OBJ_new_nid(1),buf,i,sn,ln);
805         if (op == NULL)
806                 goto err;
807         ok=OBJ_add_object(op);
808 err:
809         ASN1_OBJECT_free(op);
810         OPENSSL_free(buf);
811         return(ok);
812         }
813 #endif /* ! codereview */
```

```
**********************************************************
    3880 Wed Aug 13 19:52:56 2014
new/usr/src/lib/openssl/libsunw_crypto/objects/obj_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/objects/obj_err.c */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/objects.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_OBJ,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_OBJ,0,reason)

static ERR_STRING_DATA OBJ_str_functs[]=
        {
{ERR_FUNC(OBJ_F_OBJ_ADD_OBJECT),        "OBJ_add_object"},
{ERR_FUNC(OBJ_F_OBJ_CREATE),    "OBJ_create"},
{ERR_FUNC(OBJ_F_OBJ_DUP),       "OBJ_dup"},
{ERR_FUNC(OBJ_F_OBJ_NAME_NEW_INDEX),    "OBJ_NAME_new_index"},
{ERR_FUNC(OBJ_F_OBJ_NID2LN),    "OBJ_nid2ln"},
{ERR_FUNC(OBJ_F_OBJ_NID2OBJ),   "OBJ_nid2obj"},
{ERR_FUNC(OBJ_F_OBJ_NID2SN),    "OBJ_nid2sn"},
{0,NULL}
        };

static ERR_STRING_DATA OBJ_str_reasons[]=
        {
{ERR_REASON(OBJ_R_MALLOC_FAILURE)        ,"malloc failure"},
{ERR_REASON(OBJ_R_UNKNOWN_NID)           ,"unknown nid"},
{0,NULL}
        };

#endif

void ERR_load_OBJ_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(OBJ_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,OBJ_str_functs);
                ERR_load_strings(0,OBJ_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    4743 Wed Aug 13 19:52:57 2014
new/usr/src/lib/openssl/libsunw_crypto/objects/obj_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/objects/obj_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/lhash.h>
```



```
#include <openssl/objects.h>
#include <openssl/buffer.h>

ASN1_OBJECT *OBJ_dup(const ASN1_OBJECT *o)
        {
        ASN1_OBJECT *r;
        int i;
        char *ln=NULL,*sn=NULL;
        unsigned char *data=NULL;

        if (o == NULL) return(NULL);
        if (!(o->flags & ASN1_OBJECT_FLAG_DYNAMIC))
                return((ASN1_OBJECT *)o); /* XXX: ugh! Why? What kind of
                                             duplication is this??? */

        r=ASN1_OBJECT_new();
        if (r == NULL)
                {
                OBJerr(OBJ_F_OBJ_DUP,ERR_R_ASN1_LIB);
                return(NULL);
                }
        data=OPENSSL_malloc(o->length);
        if (data == NULL)
                goto err;
        if (o->data != NULL)
                memcpy(data,o->data,o->length);
        /* once data attached to object it remains const */
        r->data = data;
        r->length=o->length;
        r->nid=o->nid;
        r->ln=r->sn=NULL;
        if (o->ln != NULL)
                {
                i=strlen(o->ln)+1;
                ln=OPENSSL_malloc(i);
                if (ln == NULL) goto err;
                memcpy(ln,o->ln,i);
                r->ln=ln;
                }

        if (o->sn != NULL)
                {
                i=strlen(o->sn)+1;
                sn=OPENSSL_malloc(i);
                if (sn == NULL) goto err;
                memcpy(sn,o->sn,i);
                r->sn=sn;
                }
        r->flags=o->flags|(ASN1_OBJECT_FLAG_DYNAMIC|
                ASN1_OBJECT_FLAG_DYNAMIC_STRINGS|ASN1_OBJECT_FLAG_DYNAMIC_DATA);
        return(r);
err:
        OBJerr(OBJ_F_OBJ_DUP,ERR_R_MALLOC_FAILURE);
        if (ln != NULL)         OPENSSL_free(ln);
        if (sn != NULL)         OPENSSL_free(sn);
        if (data != NULL)       OPENSSL_free(data);
        if (r != NULL)          OPENSSL_free(r);
        return(NULL);
        }

int OBJ_cmp(const ASN1_OBJECT *a, const ASN1_OBJECT *b)
        {
        int ret;

        ret=(a->length-b->length);
        if (ret) return(ret);
```

```
128         return(memcmp(a->data,b->data,a->length));
129         }
130 #endif /* ! codereview */
```

```
**********************************************************
    6180 Wed Aug 13 19:52:57 2014
new/usr/src/lib/openssl/libsunw_crypto/objects/obj_xref.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/objects/obj_xref.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/objects.h>
#include "obj_xref.h"
```



```
DECLARE_STACK_OF(nid_triple)
STACK_OF(nid_triple) *sig_app, *sigx_app;

static int sig_cmp(const nid_triple *a, const nid_triple *b)
        {
        return a->sign_id - b->sign_id;
        }

DECLARE_OBJ_BSEARCH_CMP_FN(nid_triple, nid_triple, sig);
IMPLEMENT_OBJ_BSEARCH_CMP_FN(nid_triple, nid_triple, sig);

static int sig_sk_cmp(const nid_triple * const *a, const nid_triple * const *b)
        {
        return (*a)->sign_id - (*b)->sign_id;
        }

DECLARE_OBJ_BSEARCH_CMP_FN(const nid_triple *, const nid_triple *, sigx);

static int sigx_cmp(const nid_triple * const *a, const nid_triple * const *b)
        {
        int ret;
        ret = (*a)->hash_id - (*b)->hash_id;
        if (ret)
                return ret;
        return (*a)->pkey_id - (*b)->pkey_id;
        }

IMPLEMENT_OBJ_BSEARCH_CMP_FN(const nid_triple *, const nid_triple *, sigx);

int OBJ_find_sigid_algs(int signid, int *pdig_nid, int *ppkey_nid)
        {
        nid_triple tmp;
        const nid_triple *rv = NULL;
        tmp.sign_id = signid;

        if (sig_app)
                {
                int idx = sk_nid_triple_find(sig_app, &tmp);
                if (idx >= 0)
                        rv = sk_nid_triple_value(sig_app, idx);
                }

#ifndef OBJ_XREF_TEST2
        if (rv == NULL)
                {
                rv = OBJ_bsearch_sig(&tmp, sigoid_srt,
                                sizeof(sigoid_srt) / sizeof(nid_triple));
                }
#endif
        if (rv == NULL)
                return 0;
        if (pdig_nid)
                *pdig_nid = rv->hash_id;
        if (ppkey_nid)
                *ppkey_nid = rv->pkey_id;
        return 1;
        }

int OBJ_find_sigid_by_algs(int *psignid, int dig_nid, int pkey_nid)
        {
        nid_triple tmp;
        const nid_triple *t=&tmp;
        const nid_triple **rv = NULL;

        tmp.hash_id = dig_nid;
        tmp.pkey_id = pkey_nid;
```

```
129         if (sigx_app)
130                 {
131                 int idx = sk_nid_triple_find(sigx_app, &tmp);
132                 if (idx >= 0)
133                         {
134                         t = sk_nid_triple_value(sigx_app, idx);
135                         rv = &t;
136                         }
137                 }

139 #ifndef OBJ_XREF_TEST2
140         if (rv == NULL)
141                 {
142                 rv = OBJ_bsearch_sigx(&t, sigoid_srt_xref,
143                                 sizeof(sigoid_srt_xref) / sizeof(nid_triple *)
144                                 );
145                 }
146 #endif
147         if (rv == NULL)
148                 return 0;
149         if (psignid)
150                 *psignid = (*rv)->sign_id;
151         return 1;
152         }

154 int OBJ_add_sigid(int signid, int dig_id, int pkey_id)
155         {
156         nid_triple *ntr;
157         if (!sig_app)
158                 sig_app = sk_nid_triple_new(sig_sk_cmp);
159         if (!sig_app)
160                 return 0;
161         if (!sigx_app)
162                 sigx_app = sk_nid_triple_new(sigx_cmp);
163         if (!sigx_app)
164                 return 0;
165         ntr = OPENSSL_malloc(sizeof(int) * 3);
166         if (!ntr)
167                 return 0;
168         ntr->sign_id = signid;
169         ntr->hash_id = dig_id;
170         ntr->pkey_id = pkey_id;

172         if (!sk_nid_triple_push(sig_app, ntr))
173                 {
174                 OPENSSL_free(ntr);
175                 return 0;
176                 }

178         if (!sk_nid_triple_push(sigx_app, ntr))
179                 return 0;

181         sk_nid_triple_sort(sig_app);
182         sk_nid_triple_sort(sigx_app);

184         return 1;
185         }

187 static void sid_free(nid_triple *tt)
188         {
189         OPENSSL_free(tt);
190         }

192 void OBJ_sigid_free(void)
193         {
```



```
194         if (sig_app)
195                 {
196                 sk_nid_triple_pop_free(sig_app, sid_free);
197                 sig_app = NULL;
198                 }
199         if (sigx_app)
200                 {
201                 sk_nid_triple_free(sigx_app);
202                 sigx_app = NULL;
203                 }
204         }

206 #ifdef OBJ_XREF_TEST

208 main()
209         {
210         int n1, n2, n3;

212         int i, rv;
213 #ifdef OBJ_XREF_TEST2
214         for (i = 0; i < sizeof(sigoid_srt) / sizeof(nid_triple); i++)
215                 {
216                 OBJ_add_sigid(sigoid_srt[i][0], sigoid_srt[i][1],
217                                 sigoid_srt[i][2]);
218                 }
219 #endif

221         for (i = 0; i < sizeof(sigoid_srt) / sizeof(nid_triple); i++)
222                 {
223                 n1 = sigoid_srt[i][0];
224                 rv = OBJ_find_sigid_algs(n1, &n2, &n3);
225                 printf("Forward: %d, %s %s %s\n", rv,
226                         OBJ_nid2ln(n1), OBJ_nid2ln(n2), OBJ_nid2ln(n3));
227                 n1=0;
228                 rv = OBJ_find_sigid_by_algs(&n1, n2, n3);
229                 printf("Reverse: %d, %s %s %s\n", rv,
230                         OBJ_nid2ln(n1), OBJ_nid2ln(n2), OBJ_nid2ln(n3));
231                 }
232         }

234 #endif
235 #endif /* ! codereview */
```

```
**********************************************************
    7297 Wed Aug 13 19:52:57 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_asn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_asn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/ocsp.h>
```



```
ASN1_SEQUENCE(OCSP_SIGNATURE) = {
        ASN1_SIMPLE(OCSP_SIGNATURE, signatureAlgorithm, X509_ALGOR),
        ASN1_SIMPLE(OCSP_SIGNATURE, signature, ASN1_BIT_STRING),
        ASN1_EXP_SEQUENCE_OF_OPT(OCSP_SIGNATURE, certs, X509, 0)
} ASN1_SEQUENCE_END(OCSP_SIGNATURE)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_SIGNATURE)

ASN1_SEQUENCE(OCSP_CERTID) = {
        ASN1_SIMPLE(OCSP_CERTID, hashAlgorithm, X509_ALGOR),
        ASN1_SIMPLE(OCSP_CERTID, issuerNameHash, ASN1_OCTET_STRING),
        ASN1_SIMPLE(OCSP_CERTID, issuerKeyHash, ASN1_OCTET_STRING),
        ASN1_SIMPLE(OCSP_CERTID, serialNumber, ASN1_INTEGER)
} ASN1_SEQUENCE_END(OCSP_CERTID)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_CERTID)

ASN1_SEQUENCE(OCSP_ONEREQ) = {
        ASN1_SIMPLE(OCSP_ONEREQ, reqCert, OCSP_CERTID),
        ASN1_EXP_SEQUENCE_OF_OPT(OCSP_ONEREQ, singleRequestExtensions, X509_EXTE
} ASN1_SEQUENCE_END(OCSP_ONEREQ)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_ONEREQ)

ASN1_SEQUENCE(OCSP_REQINFO) = {
        ASN1_EXP_OPT(OCSP_REQINFO, version, ASN1_INTEGER, 0),
        ASN1_EXP_OPT(OCSP_REQINFO, requestorName, GENERAL_NAME, 1),
        ASN1_SEQUENCE_OF(OCSP_REQINFO, requestList, OCSP_ONEREQ),
        ASN1_EXP_SEQUENCE_OF_OPT(OCSP_REQINFO, requestExtensions, X509_EXTENSION
} ASN1_SEQUENCE_END(OCSP_REQINFO)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_REQINFO)

ASN1_SEQUENCE(OCSP_REQUEST) = {
        ASN1_SIMPLE(OCSP_REQUEST, tbsRequest, OCSP_REQINFO),
        ASN1_EXP_OPT(OCSP_REQUEST, optionalSignature, OCSP_SIGNATURE, 0)
} ASN1_SEQUENCE_END(OCSP_REQUEST)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_REQUEST)

/* OCSP_RESPONSE templates */

ASN1_SEQUENCE(OCSP_RESPBYTES) = {
            ASN1_SIMPLE(OCSP_RESPBYTES, responseType, ASN1_OBJECT),
            ASN1_SIMPLE(OCSP_RESPBYTES, response, ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END(OCSP_RESPBYTES)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_RESPBYTES)

ASN1_SEQUENCE(OCSP_RESPONSE) = {
        ASN1_SIMPLE(OCSP_RESPONSE, responseStatus, ASN1_ENUMERATED),
        ASN1_EXP_OPT(OCSP_RESPONSE, responseBytes, OCSP_RESPBYTES, 0)
} ASN1_SEQUENCE_END(OCSP_RESPONSE)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_RESPONSE)

ASN1_CHOICE(OCSP_RESPID) = {
          ASN1_EXP(OCSP_RESPID, value.byName, X509_NAME, 1),
          ASN1_EXP(OCSP_RESPID, value.byKey, ASN1_OCTET_STRING, 2)
} ASN1_CHOICE_END(OCSP_RESPID)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_RESPID)

ASN1_SEQUENCE(OCSP_REVOKEDINFO) = {
        ASN1_SIMPLE(OCSP_REVOKEDINFO, revocationTime, ASN1_GENERALIZEDTIME),
        ASN1_EXP_OPT(OCSP_REVOKEDINFO, revocationReason, ASN1_ENUMERATED, 0)
```

```
} ASN1_SEQUENCE_END(OCSP_REVOKEDINFO)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_REVOKEDINFO)

ASN1_CHOICE(OCSP_CERTSTATUS) = {
        ASN1_IMP(OCSP_CERTSTATUS, value.good, ASN1_NULL, 0),
        ASN1_IMP(OCSP_CERTSTATUS, value.revoked, OCSP_REVOKEDINFO, 1),
        ASN1_IMP(OCSP_CERTSTATUS, value.unknown, ASN1_NULL, 2)
} ASN1_CHOICE_END(OCSP_CERTSTATUS)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_CERTSTATUS)

ASN1_SEQUENCE(OCSP_SINGLERESP) = {
           ASN1_SIMPLE(OCSP_SINGLERESP, certId, OCSP_CERTID),
           ASN1_SIMPLE(OCSP_SINGLERESP, certStatus, OCSP_CERTSTATUS),
           ASN1_SIMPLE(OCSP_SINGLERESP, thisUpdate, ASN1_GENERALIZEDTIME),
           ASN1_EXP_OPT(OCSP_SINGLERESP, nextUpdate, ASN1_GENERALIZEDTIME, 0),
           ASN1_EXP_SEQUENCE_OF_OPT(OCSP_SINGLERESP, singleExtensions, X509_EXTE
} ASN1_SEQUENCE_END(OCSP_SINGLERESP)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_SINGLERESP)

ASN1_SEQUENCE(OCSP_RESPDATA) = {
           ASN1_EXP_OPT(OCSP_RESPDATA, version, ASN1_INTEGER, 0),
           ASN1_SIMPLE(OCSP_RESPDATA, responderId, OCSP_RESPID),
           ASN1_SIMPLE(OCSP_RESPDATA, producedAt, ASN1_GENERALIZEDTIME),
           ASN1_SEQUENCE_OF(OCSP_RESPDATA, responses, OCSP_SINGLERESP),
           ASN1_EXP_SEQUENCE_OF_OPT(OCSP_RESPDATA, responseExtensions, X509_EXTE
} ASN1_SEQUENCE_END(OCSP_RESPDATA)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_RESPDATA)

ASN1_SEQUENCE(OCSP_BASICRESP) = {
           ASN1_SIMPLE(OCSP_BASICRESP, tbsResponseData, OCSP_RESPDATA),
           ASN1_SIMPLE(OCSP_BASICRESP, signatureAlgorithm, X509_ALGOR),
           ASN1_SIMPLE(OCSP_BASICRESP, signature, ASN1_BIT_STRING),
           ASN1_EXP_SEQUENCE_OF_OPT(OCSP_BASICRESP, certs, X509, 0)
} ASN1_SEQUENCE_END(OCSP_BASICRESP)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_BASICRESP)

ASN1_SEQUENCE(OCSP_CRLID) = {
           ASN1_EXP_OPT(OCSP_CRLID, crlUrl, ASN1_IA5STRING, 0),
           ASN1_EXP_OPT(OCSP_CRLID, crlNum, ASN1_INTEGER, 1),
           ASN1_EXP_OPT(OCSP_CRLID, crlTime, ASN1_GENERALIZEDTIME, 2)
} ASN1_SEQUENCE_END(OCSP_CRLID)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_CRLID)

ASN1_SEQUENCE(OCSP_SERVICELOC) = {
        ASN1_SIMPLE(OCSP_SERVICELOC, issuer, X509_NAME),
        ASN1_SEQUENCE_OF_OPT(OCSP_SERVICELOC, locator, ACCESS_DESCRIPTION)
} ASN1_SEQUENCE_END(OCSP_SERVICELOC)

IMPLEMENT_ASN1_FUNCTIONS(OCSP_SERVICELOC)
#endif /* ! codereview */
```

```
**********************************************************
   10775 Wed Aug 13 19:52:57 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_cl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_cl.c */
/* Written by Tom Titchener <Tom_Titchener@groove.net> for the OpenSSL
 * project. */

/* History:
   This file was transfered to Richard Levitte from CertCo by Kathy
   Weinhold in mid-spring 2000 to be included in OpenSSL or released
   as a patch kit. */

/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
```



```
 */

#include <stdio.h>
#include <time.h>
#include <cryptlib.h>
#include <openssl/objects.h>
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/ocsp.h>

/* Utility functions related to sending OCSP requests and extracting
 * relevant information from the response.
 */

/* Add an OCSP_CERTID to an OCSP request. Return new OCSP_ONEREQ
 * pointer: useful if we want to add extensions.
 */

OCSP_ONEREQ *OCSP_request_add0_id(OCSP_REQUEST *req, OCSP_CERTID *cid)
        {
        OCSP_ONEREQ *one = NULL;

        if (!(one = OCSP_ONEREQ_new())) goto err;
        if (one->reqCert) OCSP_CERTID_free(one->reqCert);
        one->reqCert = cid;
        if (req &&
                !sk_OCSP_ONEREQ_push(req->tbsRequest->requestList, one))
                                goto err;
        return one;
err:
        OCSP_ONEREQ_free(one);
        return NULL;
        }

/* Set requestorName from an X509_NAME structure */

int OCSP_request_set1_name(OCSP_REQUEST *req, X509_NAME *nm)
        {
        GENERAL_NAME *gen;
        gen = GENERAL_NAME_new();
        if (gen == NULL)
                return 0;
        if (!X509_NAME_set(&gen->d.directoryName, nm))
                {
                GENERAL_NAME_free(gen);
                return 0;
                }
        gen->type = GEN_DIRNAME;
        if (req->tbsRequest->requestorName)
                GENERAL_NAME_free(req->tbsRequest->requestorName);
        req->tbsRequest->requestorName = gen;
        return 1;
        }


/* Add a certificate to an OCSP request */

int OCSP_request_add1_cert(OCSP_REQUEST *req, X509 *cert)
        {
        OCSP_SIGNATURE *sig;
        if (!req->optionalSignature)
                req->optionalSignature = OCSP_SIGNATURE_new();
        sig = req->optionalSignature;
        if (!sig) return 0;
```

```
128         if (!cert) return 1;
129         if (!sig->certs && !(sig->certs = sk_X509_new_null()))
130                 return 0;

132         if(!sk_X509_push(sig->certs, cert)) return 0;
133         CRYPTO_add(&cert->references, 1, CRYPTO_LOCK_X509);
134         return 1;
135         }

137 /* Sign an OCSP request set the requestorName to the subjec
138  * name of an optional signers certificate and include one
139  * or more optional certificates in the request. Behaves
140  * like PKCS7_sign().
141  */

143 int OCSP_request_sign(OCSP_REQUEST   *req,
144                       X509           *signer,
145                       EVP_PKEY       *key,
146                       const EVP_MD   *dgst,
147                       STACK_OF(X509) *certs,
148                       unsigned long flags)
149         {
150         int i;
151         OCSP_SIGNATURE *sig;
152         X509 *x;

154         if (!OCSP_request_set1_name(req, X509_get_subject_name(signer)))
155                         goto err;

157         if (!(req->optionalSignature = sig = OCSP_SIGNATURE_new())) goto err;
158         if (key)
159                 {
160                 if (!X509_check_private_key(signer, key))
161                         {
162                         OCSPerr(OCSP_F_OCSP_REQUEST_SIGN, OCSP_R_PRIVATE_KEY_DOE
163                         goto err;
164                         }
165                 if (!OCSP_REQUEST_sign(req, key, dgst)) goto err;
166                 }

168         if (!(flags & OCSP_NOCERTS))
169                 {
170                 if(!OCSP_request_add1_cert(req, signer)) goto err;
171                 for (i = 0; i < sk_X509_num(certs); i++)
172                         {
173                         x = sk_X509_value(certs, i);
174                         if (!OCSP_request_add1_cert(req, x)) goto err;
175                         }
176                 }

178         return 1;
179 err:
180         OCSP_SIGNATURE_free(req->optionalSignature);
181         req->optionalSignature = NULL;
182         return 0;
183         }

185 /* Get response status */

187 int OCSP_response_status(OCSP_RESPONSE *resp)
188         {
189         return ASN1_ENUMERATED_get(resp->responseStatus);
190         }

192 /* Extract basic response from OCSP_RESPONSE or NULL if
193  * no basic response present.
```



```
194  */


197 OCSP_BASICRESP *OCSP_response_get1_basic(OCSP_RESPONSE *resp)
198         {
199         OCSP_RESPBYTES *rb;
200         rb = resp->responseBytes;
201         if (!rb)
202                 {
203                 OCSPerr(OCSP_F_OCSP_RESPONSE_GET1_BASIC, OCSP_R_NO_RESPONSE_DATA
204                 return NULL;
205                 }
206         if (OBJ_obj2nid(rb->responseType) != NID_id_pkix_OCSP_basic)
207                 {
208                 OCSPerr(OCSP_F_OCSP_RESPONSE_GET1_BASIC, OCSP_R_NOT_BASIC_RESPON
209                 return NULL;
210                 }

212         return ASN1_item_unpack(rb->response, ASN1_ITEM_rptr(OCSP_BASICRESP));
213         }

215 /* Return number of OCSP_SINGLERESP reponses present in
216  * a basic response.
217  */

219 int OCSP_resp_count(OCSP_BASICRESP *bs)
220         {
221         if (!bs) return -1;
222         return sk_OCSP_SINGLERESP_num(bs->tbsResponseData->responses);
223         }

225 /* Extract an OCSP_SINGLERESP response with a given index */

227 OCSP_SINGLERESP *OCSP_resp_get0(OCSP_BASICRESP *bs, int idx)
228         {
229         if (!bs) return NULL;
230         return sk_OCSP_SINGLERESP_value(bs->tbsResponseData->responses, idx);
231         }

233 /* Look single response matching a given certificate ID */

235 int OCSP_resp_find(OCSP_BASICRESP *bs, OCSP_CERTID *id, int last)
236         {
237         int i;
238         STACK_OF(OCSP_SINGLERESP) *sresp;
239         OCSP_SINGLERESP *single;
240         if (!bs) return -1;
241         if (last < 0) last = 0;
242         else last++;
243         sresp = bs->tbsResponseData->responses;
244         for (i = last; i < sk_OCSP_SINGLERESP_num(sresp); i++)
245                 {
246                 single = sk_OCSP_SINGLERESP_value(sresp, i);
247                 if (!OCSP_id_cmp(id, single->certId)) return i;
248                 }
249         return -1;
250         }

252 /* Extract status information from an OCSP_SINGLERESP structure.
253  * Note: the revtime and reason values are only set if the
254  * certificate status is revoked. Returns numerical value of
255  * status.
256  */

258 int OCSP_single_get0_status(OCSP_SINGLERESP *single, int *reason,
259                             ASN1_GENERALIZEDTIME **revtime,
```

```
260                                 ASN1_GENERALIZEDTIME **thisupd,
261                                 ASN1_GENERALIZEDTIME **nextupd)
262         {
263         int ret;
264         OCSP_CERTSTATUS *cst;
265         if(!single) return -1;
266         cst = single->certStatus;
267         ret = cst->type;
268         if (ret == V_OCSP_CERTSTATUS_REVOKED)
269                 {
270                 OCSP_REVOKEDINFO *rev = cst->value.revoked;
271                 if (revtime) *revtime = rev->revocationTime;
272                 if (reason)
273                         {
274                         if(rev->revocationReason)
275                                 *reason = ASN1_ENUMERATED_get(rev->revocationRea
276                         else *reason = -1;
277                         }
278                 }
279         if(thisupd) *thisupd = single->thisUpdate;
280         if(nextupd) *nextupd = single->nextUpdate;
281         return ret;
282         }

284 /* This function combines the previous ones: look up a certificate ID and
285  * if found extract status information. Return 0 is successful.
286  */

288 int OCSP_resp_find_status(OCSP_BASICRESP *bs, OCSP_CERTID *id, int *status,
289                                 int *reason,
290                                 ASN1_GENERALIZEDTIME **revtime,
291                                 ASN1_GENERALIZEDTIME **thisupd,
292                                 ASN1_GENERALIZEDTIME **nextupd)
293         {
294         int i;
295         OCSP_SINGLERESP *single;
296         i = OCSP_resp_find(bs, id, -1);
297         /* Maybe check for multiple responses and give an error? */
298         if(i < 0) return 0;
299         single = OCSP_resp_get0(bs, i);
300         i = OCSP_single_get0_status(single, reason, revtime, thisupd, nextupd);
301         if(status) *status = i;
302         return 1;
303         }

305 /* Check validity of thisUpdate and nextUpdate fields. It is possible that the r
306  * take a few seconds to process and/or the time wont be totally accurate. There
307  * rejecting otherwise valid time we allow the times to be within 'nsec' of the
308  * Also to avoid accepting very old responses without a nextUpdate field an opti
309  * parameter specifies the maximum age the thisUpdate field can be.
310  */

312 int OCSP_check_validity(ASN1_GENERALIZEDTIME *thisupd, ASN1_GENERALIZEDTIME *nex
313         {
314         int ret = 1;
315         time_t t_now, t_tmp;
316         time(&t_now);
317         /* Check thisUpdate is valid and not more than nsec in the future */
318         if (!ASN1_GENERALIZEDTIME_check(thisupd))
319                 {
320                 OCSPerr(OCSP_F_OCSP_CHECK_VALIDITY, OCSP_R_ERROR_IN_THISUPDATE_F
321                 ret = 0;
322                 }
323         else
324                 {
325                         t_tmp = t_now + nsec;
```



```
326                         if (X509_cmp_time(thisupd, &t_tmp) > 0)
327                         {
328                         OCSPerr(OCSP_F_OCSP_CHECK_VALIDITY, OCSP_R_STATUS_NOT_YE
329                         ret = 0;
330                         }

332                 /* If maxsec specified check thisUpdate is not more than maxsec
333                 if (maxsec >= 0)
334                         {
335                         t_tmp = t_now - maxsec;
336                         if (X509_cmp_time(thisupd, &t_tmp) < 0)
337                                 {
338                                 OCSPerr(OCSP_F_OCSP_CHECK_VALIDITY, OCSP_R_STATU
339                                 ret = 0;
340                                 }
341                         }
342                 }


345         if (!nextupd) return ret;

347         /* Check nextUpdate is valid and not more than nsec in the past */
348         if (!ASN1_GENERALIZEDTIME_check(nextupd))
349                 {
350                 OCSPerr(OCSP_F_OCSP_CHECK_VALIDITY, OCSP_R_ERROR_IN_NEXTUPDATE_F
351                 ret = 0;
352                 }
353         else
354                 {
355                 t_tmp = t_now - nsec;
356                 if (X509_cmp_time(nextupd, &t_tmp) < 0)
357                         {
358                         OCSPerr(OCSP_F_OCSP_CHECK_VALIDITY, OCSP_R_STATUS_EXPIRE
359                         ret = 0;
360                         }
361                 }

363         /* Also don't allow nextUpdate to precede thisUpdate */
364         if (ASN1_STRING_cmp(nextupd, thisupd) < 0)
365                 {
366                 OCSPerr(OCSP_F_OCSP_CHECK_VALIDITY, OCSP_R_NEXTUPDATE_BEFORE_THI
367                 ret = 0;
368                 }

370         return ret;
371         }
372 #endif /* ! codereview */
```

```
**********************************************************
    6665 Wed Aug 13 19:52:57 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ocsp/ocsp_err.c */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/ocsp.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_OCSP,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_OCSP,0,reason)

static ERR_STRING_DATA OCSP_str_functs[]=
        {
{ERR_FUNC(OCSP_F_ASN1_STRING_ENCODE),   "ASN1_STRING_encode"},
{ERR_FUNC(OCSP_F_D2I_OCSP_NONCE),       "D2I_OCSP_NONCE"},
{ERR_FUNC(OCSP_F_OCSP_BASIC_ADD1_STATUS),       "OCSP_basic_add1_status"},
{ERR_FUNC(OCSP_F_OCSP_BASIC_SIGN),      "OCSP_basic_sign"},
{ERR_FUNC(OCSP_F_OCSP_BASIC_VERIFY),    "OCSP_basic_verify"},
{ERR_FUNC(OCSP_F_OCSP_CERT_ID_NEW),     "OCSP_cert_id_new"},
{ERR_FUNC(OCSP_F_OCSP_CHECK_DELEGATED), "OCSP_CHECK_DELEGATED"},
{ERR_FUNC(OCSP_F_OCSP_CHECK_IDS),       "OCSP_CHECK_IDS"},
{ERR_FUNC(OCSP_F_OCSP_CHECK_ISSUER),    "OCSP_CHECK_ISSUER"},
{ERR_FUNC(OCSP_F_OCSP_CHECK_VALIDITY),  "OCSP_check_validity"},
{ERR_FUNC(OCSP_F_OCSP_MATCH_ISSUERID),  "OCSP_MATCH_ISSUERID"},
{ERR_FUNC(OCSP_F_OCSP_PARSE_URL),       "OCSP_parse_url"},
{ERR_FUNC(OCSP_F_OCSP_REQUEST_SIGN),    "OCSP_request_sign"},
{ERR_FUNC(OCSP_F_OCSP_REQUEST_VERIFY),  "OCSP_request_verify"},
{ERR_FUNC(OCSP_F_OCSP_RESPONSE_GET1_BASIC),     "OCSP_response_get1_basic"},
{ERR_FUNC(OCSP_F_OCSP_SENDREQ_BIO),     "OCSP_sendreq_bio"},
{ERR_FUNC(OCSP_F_OCSP_SENDREQ_NBIO),    "OCSP_sendreq_nbio"},
{ERR_FUNC(OCSP_F_PARSE_HTTP_LINE1),     "PARSE_HTTP_LINE1"},
{ERR_FUNC(OCSP_F_REQUEST_VERIFY),       "REQUEST_VERIFY"},
{0,NULL}
        };

static ERR_STRING_DATA OCSP_str_reasons[]=
        {
{ERR_REASON(OCSP_R_BAD_DATA)              ,"bad data"},
{ERR_REASON(OCSP_R_CERTIFICATE_VERIFY_ERROR),"certificate verify error"},
{ERR_REASON(OCSP_R_DIGEST_ERR)            ,"digest err"},
{ERR_REASON(OCSP_R_ERROR_IN_NEXTUPDATE_FIELD),"error in nextupdate field"},
{ERR_REASON(OCSP_R_ERROR_IN_THISUPDATE_FIELD),"error in thisupdate field"},
{ERR_REASON(OCSP_R_ERROR_PARSING_URL)     ,"error parsing url"},
{ERR_REASON(OCSP_R_MISSING_OCSPSIGNING_USAGE),"missing ocspsigning usage"},
{ERR_REASON(OCSP_R_NEXTUPDATE_BEFORE_THISUPDATE),"nextupdate before thisupdate"},
{ERR_REASON(OCSP_R_NOT_BASIC_RESPONSE)    ,"not basic response"},
{ERR_REASON(OCSP_R_NO_CERTIFICATES_IN_CHAIN),"no certificates in chain"},
{ERR_REASON(OCSP_R_NO_CONTENT)            ,"no content"},
{ERR_REASON(OCSP_R_NO_PUBLIC_KEY)         ,"no public key"},
{ERR_REASON(OCSP_R_NO_RESPONSE_DATA)      ,"no response data"},
{ERR_REASON(OCSP_R_NO_REVOKED_TIME)       ,"no revoked time"},
{ERR_REASON(OCSP_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE),"private key does not
{ERR_REASON(OCSP_R_REQUEST_NOT_SIGNED)    ,"request not signed"},
{ERR_REASON(OCSP_R_RESPONSE_CONTAINS_NO_REVOCATION_DATA),"response contains no r
{ERR_REASON(OCSP_R_ROOT_CA_NOT_TRUSTED)   ,"root ca not trusted"},
{ERR_REASON(OCSP_R_SERVER_READ_ERROR)     ,"server read error"},
{ERR_REASON(OCSP_R_SERVER_RESPONSE_ERROR),"server response error"},
{ERR_REASON(OCSP_R_SERVER_RESPONSE_PARSE_ERROR),"server response parse error"},
{ERR_REASON(OCSP_R_SERVER_WRITE_ERROR)    ,"server write error"},
{ERR_REASON(OCSP_R_SIGNATURE_FAILURE)     ,"signature failure"},
{ERR_REASON(OCSP_R_SIGNER_CERTIFICATE_NOT_FOUND),"signer certificate not found"}
{ERR_REASON(OCSP_R_STATUS_EXPIRED)        ,"status expired"},
{ERR_REASON(OCSP_R_STATUS_NOT_YET_VALID) ,"status not yet valid"},
{ERR_REASON(OCSP_R_STATUS_TOO_OLD)        ,"status too old"},
{ERR_REASON(OCSP_R_UNKNOWN_MESSAGE_DIGEST),"unknown message digest"},
{ERR_REASON(OCSP_R_UNKNOWN_NID)           ,"unknown nid"},
{ERR_REASON(OCSP_R_UNSUPPORTED_REQUESTORNAME_TYPE),"unsupported requestorname ty
{0,NULL}
```

```
128         };

130 #endif

132 void ERR_load_OCSP_strings(void)
133         {
134 #ifndef OPENSSL_NO_ERR

136         if (ERR_func_error_string(OCSP_str_functs[0].error) == NULL)
137                 {
138                 ERR_load_strings(0,OCSP_str_functs);
139                 ERR_load_strings(0,OCSP_str_reasons);
140                 }
141 #endif
142         }
143 #endif /* ! codereview */
```

```
**********************************************************
   16212 Wed Aug 13 19:52:57 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_ext.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_ext.c */
/* Written by Tom Titchener <Tom_Titchener@groove.net> for the OpenSSL
 * project. */

/* History:
   This file was transfered to Richard Levitte from CertCo by Kathy
   Weinhold in mid-spring 2000 to be included in OpenSSL or released
   as a patch kit. */

/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
```



```
 */

#include <stdio.h>
#include <cryptlib.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/ocsp.h>
#include <openssl/rand.h>
#include <openssl/x509v3.h>

/* Standard wrapper functions for extensions */

/* OCSP request extensions */

int OCSP_REQUEST_get_ext_count(OCSP_REQUEST *x)
        {
        return(X509v3_get_ext_count(x->tbsRequest->requestExtensions));
        }

int OCSP_REQUEST_get_ext_by_NID(OCSP_REQUEST *x, int nid, int lastpos)
        {
        return(X509v3_get_ext_by_NID(x->tbsRequest->requestExtensions,nid,lastpo
        }

int OCSP_REQUEST_get_ext_by_OBJ(OCSP_REQUEST *x, ASN1_OBJECT *obj, int lastpos)
        {
        return(X509v3_get_ext_by_OBJ(x->tbsRequest->requestExtensions,obj,lastpo
        }

int OCSP_REQUEST_get_ext_by_critical(OCSP_REQUEST *x, int crit, int lastpos)
        {
        return(X509v3_get_ext_by_critical(x->tbsRequest->requestExtensions,crit,
        }

X509_EXTENSION *OCSP_REQUEST_get_ext(OCSP_REQUEST *x, int loc)
        {
        return(X509v3_get_ext(x->tbsRequest->requestExtensions,loc));
        }

X509_EXTENSION *OCSP_REQUEST_delete_ext(OCSP_REQUEST *x, int loc)
        {
        return(X509v3_delete_ext(x->tbsRequest->requestExtensions,loc));
        }

void *OCSP_REQUEST_get1_ext_d2i(OCSP_REQUEST *x, int nid, int *crit, int *idx)
        {
        return X509V3_get_d2i(x->tbsRequest->requestExtensions, nid, crit, idx);
        }

int OCSP_REQUEST_add1_ext_i2d(OCSP_REQUEST *x, int nid, void *value, int crit,
                                                        unsigned long flags)
        {
        return X509V3_add1_i2d(&x->tbsRequest->requestExtensions, nid, value, cr
        }

int OCSP_REQUEST_add_ext(OCSP_REQUEST *x, X509_EXTENSION *ex, int loc)
        {
        return(X509v3_add_ext(&(x->tbsRequest->requestExtensions),ex,loc) != NUL
        }

/* Single extensions */

int OCSP_ONEREQ_get_ext_count(OCSP_ONEREQ *x)
        {
        return(X509v3_get_ext_count(x->singleRequestExtensions));
        }
```

```
129 int OCSP_ONEREQ_get_ext_by_NID(OCSP_ONEREQ *x, int nid, int lastpos)
130         {
131         return(X509v3_get_ext_by_NID(x->singleRequestExtensions,nid,lastpos));
132         }
133
134 int OCSP_ONEREQ_get_ext_by_OBJ(OCSP_ONEREQ *x, ASN1_OBJECT *obj, int lastpos)
135         {
136         return(X509v3_get_ext_by_OBJ(x->singleRequestExtensions,obj,lastpos));
137         }
138
139 int OCSP_ONEREQ_get_ext_by_critical(OCSP_ONEREQ *x, int crit, int lastpos)
140         {
141         return(X509v3_get_ext_by_critical(x->singleRequestExtensions,crit,lastpo
142         }
143
144 X509_EXTENSION *OCSP_ONEREQ_get_ext(OCSP_ONEREQ *x, int loc)
145         {
146         return(X509v3_get_ext(x->singleRequestExtensions,loc));
147         }
148
149 X509_EXTENSION *OCSP_ONEREQ_delete_ext(OCSP_ONEREQ *x, int loc)
150         {
151         return(X509v3_delete_ext(x->singleRequestExtensions,loc));
152         }
153
154 void *OCSP_ONEREQ_get1_ext_d2i(OCSP_ONEREQ *x, int nid, int *crit, int *idx)
155         {
156         return X509V3_get_d2i(x->singleRequestExtensions, nid, crit, idx);
157         }
158
159 int OCSP_ONEREQ_add1_ext_i2d(OCSP_ONEREQ *x, int nid, void *value, int crit,
160                                                         unsigned long flags)
161         {
162         return X509V3_add1_i2d(&x->singleRequestExtensions, nid, value, crit, fl
163         }
164
165 int OCSP_ONEREQ_add_ext(OCSP_ONEREQ *x, X509_EXTENSION *ex, int loc)
166         {
167         return(X509v3_add_ext(&(x->singleRequestExtensions),ex,loc) != NULL);
168         }
169
170 /* OCSP Basic response */
171
172 int OCSP_BASICRESP_get_ext_count(OCSP_BASICRESP *x)
173         {
174         return(X509v3_get_ext_count(x->tbsResponseData->responseExtensions));
175         }
176
177 int OCSP_BASICRESP_get_ext_by_NID(OCSP_BASICRESP *x, int nid, int lastpos)
178         {
179         return(X509v3_get_ext_by_NID(x->tbsResponseData->responseExtensions,nid,
180         }
181
182 int OCSP_BASICRESP_get_ext_by_OBJ(OCSP_BASICRESP *x, ASN1_OBJECT *obj, int lastp
183         {
184         return(X509v3_get_ext_by_OBJ(x->tbsResponseData->responseExtensions,obj,
185         }
186
187 int OCSP_BASICRESP_get_ext_by_critical(OCSP_BASICRESP *x, int crit, int lastpos)
188         {
189         return(X509v3_get_ext_by_critical(x->tbsResponseData->responseExtensions
190         }
191
192 X509_EXTENSION *OCSP_BASICRESP_get_ext(OCSP_BASICRESP *x, int loc)
193         {
```



```
194         return(X509v3_get_ext(x->tbsResponseData->responseExtensions,loc));
195         }
196
197 X509_EXTENSION *OCSP_BASICRESP_delete_ext(OCSP_BASICRESP *x, int loc)
198         {
199         return(X509v3_delete_ext(x->tbsResponseData->responseExtensions,loc));
200         }
201
202 void *OCSP_BASICRESP_get1_ext_d2i(OCSP_BASICRESP *x, int nid, int *crit, int *id
203         {
204         return X509V3_get_d2i(x->tbsResponseData->responseExtensions, nid, crit,
205         }
206
207 int OCSP_BASICRESP_add1_ext_i2d(OCSP_BASICRESP *x, int nid, void *value, int cri
208                                                         unsigned long flags)
209         {
210         return X509V3_add1_i2d(&x->tbsResponseData->responseExtensions, nid, val
211         }
212
213 int OCSP_BASICRESP_add_ext(OCSP_BASICRESP *x, X509_EXTENSION *ex, int loc)
214         {
215         return(X509v3_add_ext(&(x->tbsResponseData->responseExtensions),ex,loc)
216         }
217
218 /* OCSP single response extensions */
219
220 int OCSP_SINGLERESP_get_ext_count(OCSP_SINGLERESP *x)
221         {
222         return(X509v3_get_ext_count(x->singleExtensions));
223         }
224
225 int OCSP_SINGLERESP_get_ext_by_NID(OCSP_SINGLERESP *x, int nid, int lastpos)
226         {
227         return(X509v3_get_ext_by_NID(x->singleExtensions,nid,lastpos));
228         }
229
230 int OCSP_SINGLERESP_get_ext_by_OBJ(OCSP_SINGLERESP *x, ASN1_OBJECT *obj, int las
231         {
232         return(X509v3_get_ext_by_OBJ(x->singleExtensions,obj,lastpos));
233         }
234
235 int OCSP_SINGLERESP_get_ext_by_critical(OCSP_SINGLERESP *x, int crit, int lastpo
236         {
237         return(X509v3_get_ext_by_critical(x->singleExtensions,crit,lastpos));
238         }
239
240 X509_EXTENSION *OCSP_SINGLERESP_get_ext(OCSP_SINGLERESP *x, int loc)
241         {
242         return(X509v3_get_ext(x->singleExtensions,loc));
243         }
244
245 X509_EXTENSION *OCSP_SINGLERESP_delete_ext(OCSP_SINGLERESP *x, int loc)
246         {
247         return(X509v3_delete_ext(x->singleExtensions,loc));
248         }
249
250 void *OCSP_SINGLERESP_get1_ext_d2i(OCSP_SINGLERESP *x, int nid, int *crit, int *
251         {
252         return X509V3_get_d2i(x->singleExtensions, nid, crit, idx);
253         }
254
255 int OCSP_SINGLERESP_add1_ext_i2d(OCSP_SINGLERESP *x, int nid, void *value, int c
256                                                         unsigned long flags)
257         {
258         return X509V3_add1_i2d(&x->singleExtensions, nid, value, crit, flags);
259         }
```

```
int OCSP_SINGLERESP_add_ext(OCSP_SINGLERESP *x, X509_EXTENSION *ex, int loc)
        {
        return(X509v3_add_ext(&(x->singleExtensions),ex,loc) != NULL);
        }

/* also CRL Entry Extensions */
#if 0
ASN1_STRING *ASN1_STRING_encode(ASN1_STRING *s, i2d_of_void *i2d,
                                void *data, STACK_OF(ASN1_OBJECT) *sk)
        {
        int i;
        unsigned char *p, *b = NULL;

        if (data)
                {
                if ((i=i2d(data,NULL)) <= 0) goto err;
                if (!(b=p=OPENSSL_malloc((unsigned int)i)))
                        goto err;
                if (i2d(data, &p) <= 0) goto err;
                }
        else if (sk)
                {
                if ((i=i2d_ASN1_SET_OF_ASN1_OBJECT(sk,NULL,
                                                   (I2D_OF(ASN1_OBJECT))i2d,
                                                   V_ASN1_SEQUENCE,
                                                   V_ASN1_UNIVERSAL,
                                                   IS_SEQUENCE))<=0) goto err;
                if (!(b=p=OPENSSL_malloc((unsigned int)i)))
                        goto err;
                if (i2d_ASN1_SET_OF_ASN1_OBJECT(sk,&p,(I2D_OF(ASN1_OBJECT))i2d,
                                                V_ASN1_SEQUENCE,
                                                V_ASN1_UNIVERSAL,
                                                IS_SEQUENCE)<=0) goto err;
                }
        else
                {
                OCSPerr(OCSP_F_ASN1_STRING_ENCODE,OCSP_R_BAD_DATA);
                goto err;
                }
        if (!s && !(s = ASN1_STRING_new())) goto err;
        if (!(ASN1_STRING_set(s, b, i))) goto err;
        OPENSSL_free(b);
        return s;
err:
        if (b) OPENSSL_free(b);
        return NULL;
        }
#endif

/* Nonce handling functions */

/* Add a nonce to an extension stack. A nonce can be specificed or if NULL
 * a random nonce will be generated.
 * Note: OpenSSL 0.9.7d and later create an OCTET STRING containing the
 * nonce, previous versions used the raw nonce.
 */

static int ocsp_add1_nonce(STACK_OF(X509_EXTENSION) **exts, unsigned char *val,
        {
        unsigned char *tmpval;
        ASN1_OCTET_STRING os;
        int ret = 0;
        if (len <= 0) len = OCSP_DEFAULT_NONCE_LENGTH;
        /* Create the OCTET STRING manually by writing out the header and
         * appending the content octets. This avoids an extra memory allocation
```



```
         * operation in some cases. Applications should *NOT* do this because
         * it relies on library internals.
         */
        os.length = ASN1_object_size(0, len, V_ASN1_OCTET_STRING);
        os.data = OPENSSL_malloc(os.length);
        if (os.data == NULL)
                goto err;
        tmpval = os.data;
        ASN1_put_object(&tmpval, 0, len, V_ASN1_OCTET_STRING, V_ASN1_UNIVERSAL);
        if (val)
                memcpy(tmpval, val, len);
        else
                RAND_pseudo_bytes(tmpval, len);
        if(!X509V3_add1_i2d(exts, NID_id_pkix_OCSP_Nonce,
                        &os, 0, X509V3_ADD_REPLACE))
                        goto err;
        ret = 1;
        err:
        if (os.data)
                OPENSSL_free(os.data);
        return ret;
        }


/* Add nonce to an OCSP request */

int OCSP_request_add1_nonce(OCSP_REQUEST *req, unsigned char *val, int len)
        {
        return ocsp_add1_nonce(&req->tbsRequest->requestExtensions, val, len);
        }

/* Same as above but for a response */

int OCSP_basic_add1_nonce(OCSP_BASICRESP *resp, unsigned char *val, int len)
        {
        return ocsp_add1_nonce(&resp->tbsResponseData->responseExtensions, val,
        }

/* Check nonce validity in a request and response.
 * Return value reflects result:
 *  1: nonces present and equal.
 *  2: nonces both absent.
 *  3: nonce present in response only.
 *  0: nonces both present and not equal.
 * -1: nonce in request only.
 *
 *  For most responders clients can check return > 0.
 *  If responder doesn't handle nonces return != 0 may be
 *  necessary. return == 0 is always an error.
 */

int OCSP_check_nonce(OCSP_REQUEST *req, OCSP_BASICRESP *bs)
        {
        /*
         * Since we are only interested in the presence or absence of
         * the nonce and comparing its value there is no need to use
         * the X509V3 routines: this way we can avoid them allocating an
         * ASN1_OCTET_STRING structure for the value which would be
         * freed immediately anyway.
         */

        int req_idx, resp_idx;
        X509_EXTENSION *req_ext, *resp_ext;
        req_idx = OCSP_REQUEST_get_ext_by_NID(req, NID_id_pkix_OCSP_Nonce, -1);
        resp_idx = OCSP_BASICRESP_get_ext_by_NID(bs, NID_id_pkix_OCSP_Nonce, -1)
        /* Check both absent */
```

```
392         if((req_idx < 0) && (resp_idx < 0))
393                 return 2;
394         /* Check in request only */
395         if((req_idx >= 0) && (resp_idx < 0))
396                 return -1;
397         /* Check in response but not request */
398         if((req_idx < 0) && (resp_idx >= 0))
399                 return 3;
400         /* Otherwise nonce in request and response so retrieve the extensions */
401         req_ext = OCSP_REQUEST_get_ext(req, req_idx);
402         resp_ext = OCSP_BASICRESP_get_ext(bs, resp_idx);
403         if(ASN1_OCTET_STRING_cmp(req_ext->value, resp_ext->value))
404                 return 0;
405         return 1;
406         }

408 /* Copy the nonce value (if any) from an OCSP request to
409  * a response.
410  */

412 int OCSP_copy_nonce(OCSP_BASICRESP *resp, OCSP_REQUEST *req)
413         {
414         X509_EXTENSION *req_ext;
415         int req_idx;
416         /* Check for nonce in request */
417         req_idx = OCSP_REQUEST_get_ext_by_NID(req, NID_id_pkix_OCSP_Nonce, -1);
418         /* If no nonce that's OK */
419         if (req_idx < 0) return 2;
420         req_ext = OCSP_REQUEST_get_ext(req, req_idx);
421         return OCSP_BASICRESP_add_ext(resp, req_ext, -1);
422         }

424 X509_EXTENSION *OCSP_crlID_new(char *url, long *n, char *tim)
425         {
426         X509_EXTENSION *x = NULL;
427         OCSP_CRLID *cid = NULL;

429         if (!(cid = OCSP_CRLID_new())) goto err;
430         if (url)
431                 {
432                 if (!(cid->crlUrl = ASN1_IA5STRING_new())) goto err;
433                 if (!(ASN1_STRING_set(cid->crlUrl, url, -1))) goto err;
434                 }
435         if (n)
436                 {
437                 if (!(cid->crlNum = ASN1_INTEGER_new())) goto err;
438                 if (!(ASN1_INTEGER_set(cid->crlNum, *n))) goto err;
439                 }
440         if (tim)
441                 {
442                 if (!(cid->crlTime = ASN1_GENERALIZEDTIME_new())) goto err;
443                 if (!(ASN1_GENERALIZEDTIME_set_string(cid->crlTime, tim)))
444                         goto err;
445                 }
446         x = X509V3_EXT_i2d(NID_id_pkix_OCSP_CrlID, 0, cid);
447 err:
448         if (cid) OCSP_CRLID_free(cid);
449         return x;
450         }

452 /*   AcceptableResponses ::= SEQUENCE OF OBJECT IDENTIFIER */
453 X509_EXTENSION *OCSP_accept_responses_new(char **oids)
454         {
455         int nid;
456         STACK_OF(ASN1_OBJECT) *sk = NULL;
457         ASN1_OBJECT *o = NULL;
```



```
458         X509_EXTENSION *x = NULL;

460         if (!(sk = sk_ASN1_OBJECT_new_null())) goto err;
461         while (oids && *oids)
462                 {
463                 if ((nid=OBJ_txt2nid(*oids))!=NID_undef&&(o=OBJ_nid2obj(nid)))
464                         sk_ASN1_OBJECT_push(sk, o);
465                 oids++;
466                 }
467         x = X509V3_EXT_i2d(NID_id_pkix_OCSP_acceptableResponses, 0, sk);
468 err:
469         if (sk) sk_ASN1_OBJECT_pop_free(sk, ASN1_OBJECT_free);
470         return x;
471         }

473 /*  ArchiveCutoff ::= GeneralizedTime */
474 X509_EXTENSION *OCSP_archive_cutoff_new(char* tim)
475         {
476         X509_EXTENSION *x=NULL;
477         ASN1_GENERALIZEDTIME *gt = NULL;

479         if (!(gt = ASN1_GENERALIZEDTIME_new())) goto err;
480         if (!(ASN1_GENERALIZEDTIME_set_string(gt, tim))) goto err;
481         x = X509V3_EXT_i2d(NID_id_pkix_OCSP_archiveCutoff, 0, gt);
482 err:
483         if (gt) ASN1_GENERALIZEDTIME_free(gt);
484         return x;
485         }

487 /* per ACCESS_DESCRIPTION parameter are oids, of which there are currently
488  * two--NID_ad_ocsp, NID_id_ad_caIssuers--and GeneralName value.  This
489  * method forces NID_ad_ocsp and uniformResourceLocator [6] IA5String.
490  */
491 X509_EXTENSION *OCSP_url_svcloc_new(X509_NAME* issuer, char **urls)
492         {
493         X509_EXTENSION *x = NULL;
494         ASN1_IA5STRING *ia5 = NULL;
495         OCSP_SERVICELOC *sloc = NULL;
496         ACCESS_DESCRIPTION *ad = NULL;

498         if (!(sloc = OCSP_SERVICELOC_new())) goto err;
499         if (!(sloc->issuer = X509_NAME_dup(issuer))) goto err;
500         if (urls && *urls && !(sloc->locator = sk_ACCESS_DESCRIPTION_new_null())
501         while (urls && *urls)
502                 {
503                 if (!(ad = ACCESS_DESCRIPTION_new())) goto err;
504                 if (!(ad->method=OBJ_nid2obj(NID_ad_OCSP))) goto err;
505                 if (!(ad->location = GENERAL_NAME_new())) goto err;
506                 if (!(ia5 = ASN1_IA5STRING_new())) goto err;
507                 if (!ASN1_STRING_set((ASN1_STRING*)ia5, *urls, -1)) goto err;
508                 ad->location->type = GEN_URI;
509                 ad->location->d.ia5 = ia5;
510                 if (!sk_ACCESS_DESCRIPTION_push(sloc->locator, ad)) goto err;
511                 urls++;
512                 }
513         x = X509V3_EXT_i2d(NID_id_pkix_OCSP_serviceLocator, 0, sloc);
514 err:
515         if (sloc) OCSP_SERVICELOC_free(sloc);
516         return x;
517         }
518 #endif /* ! codereview */
```

```
**********************************************************
   11641 Wed Aug 13 19:52:58 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_ht.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_ht.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
```



```
#include <string.h>
#include "e_os.h"
#include <openssl/asn1.h>
#include <openssl/ocsp.h>
#include <openssl/err.h>
#include <openssl/buffer.h>
#ifdef OPENSSL_SYS_SUNOS
#define strtoul (unsigned long)strtol
#endif /* OPENSSL_SYS_SUNOS */

/* Stateful OCSP request code, supporting non-blocking I/O */

/* Opaque OCSP request status structure */

struct ocsp_req_ctx_st {
        int state;              /* Current I/O state */
        unsigned char *iobuf;   /* Line buffer */
        int iobuflen;           /* Line buffer length */
        BIO *io;                /* BIO to perform I/O with */
        BIO *mem;               /* Memory BIO response is built into */
        unsigned long asn1_len; /* ASN1 length of response */
        };

#define OCSP_MAX_REQUEST_LENGTH (100 * 1024)
#define OCSP_MAX_LINE_LEN       4096;

/* OCSP states */

/* If set no reading should be performed */
#define OHS_NOREAD              0x1000
/* Error condition */
#define OHS_ERROR               (0 | OHS_NOREAD)
/* First line being read */
#define OHS_FIRSTLINE           1
/* MIME headers being read */
#define OHS_HEADERS             2
/* OCSP initial header (tag + length) being read */
#define OHS_ASN1_HEADER         3
/* OCSP content octets being read */
#define OHS_ASN1_CONTENT        4
/* Request being sent */
#define OHS_ASN1_WRITE          (6 | OHS_NOREAD)
/* Request being flushed */
#define OHS_ASN1_FLUSH          (7 | OHS_NOREAD)
/* Completed */
#define OHS_DONE                (8 | OHS_NOREAD)


static int parse_http_line1(char *line);

void OCSP_REQ_CTX_free(OCSP_REQ_CTX *rctx)
        {
        if (rctx->mem)
                BIO_free(rctx->mem);
        if (rctx->iobuf)
                OPENSSL_free(rctx->iobuf);
        OPENSSL_free(rctx);
        }

int OCSP_REQ_CTX_set1_req(OCSP_REQ_CTX *rctx, OCSP_REQUEST *req)
        {
        static const char req_hdr[] =
        "Content-Type: application/ocsp-request\r\n"
        "Content-Length: %d\r\n\r\n";
        if (BIO_printf(rctx->mem, req_hdr, i2d_OCSP_REQUEST(req, NULL)) <= 0)
                return 0;
```

```
128         if (i2d_OCSP_REQUEST_bio(rctx->mem, req) <= 0)
129                 return 0;
130         rctx->state = OHS_ASN1_WRITE;
131         rctx->asn1_len = BIO_get_mem_data(rctx->mem, NULL);
132         return 1;
133         }

135 int OCSP_REQ_CTX_add1_header(OCSP_REQ_CTX *rctx,
136                 const char *name, const char *value)
137         {
138         if (!name)
139                 return 0;
140         if (BIO_puts(rctx->mem, name) <= 0)
141                 return 0;
142         if (value)
143                 {
144                 if (BIO_write(rctx->mem, ": ", 2) != 2)
145                         return 0;
146                 if (BIO_puts(rctx->mem, value) <= 0)
147                         return 0;
148                 }
149         if (BIO_write(rctx->mem, "\r\n", 2) != 2)
150                 return 0;
151         return 1;
152         }

154 OCSP_REQ_CTX *OCSP_sendreq_new(BIO *io, char *path, OCSP_REQUEST *req,
155                                                                 int maxline)
156         {
157         static const char post_hdr[] = "POST %s HTTP/1.0\r\n";

159         OCSP_REQ_CTX *rctx;
160         rctx = OPENSSL_malloc(sizeof(OCSP_REQ_CTX));
161         if (!rctx)
162                 return NULL;
163         rctx->state = OHS_ERROR;
164         rctx->mem = BIO_new(BIO_s_mem());
165         rctx->io = io;
166         rctx->asn1_len = 0;
167         if (maxline > 0)
168                 rctx->iobuflen = maxline;
169         else
170                 rctx->iobuflen = OCSP_MAX_LINE_LEN;
171         rctx->iobuf = OPENSSL_malloc(rctx->iobuflen);
172         if (!rctx->mem || !rctx->iobuf)
173                 goto err;
174         if (!path)
175                 path = "/";

177         if (BIO_printf(rctx->mem, post_hdr, path) <= 0)
178                 goto err;

180         if (req && !OCSP_REQ_CTX_set1_req(rctx, req))
181                 goto err;

183         return rctx;
184         err:
185         OCSP_REQ_CTX_free(rctx);
186         return NULL;
187         }

189 /* Parse the HTTP response. This will look like this:
190  * "HTTP/1.0 200 OK". We need to obtain the numeric code and
191  * (optional) informational message.
192  */
```



```
194 static int parse_http_line1(char *line)
195         {
196         int retcode;
197         char *p, *q, *r;
198         /* Skip to first white space (passed protocol info) */

200         for(p = line; *p && !isspace((unsigned char)*p); p++)
201                 continue;
202         if(!*p)
203                 {
204                 OCSPerr(OCSP_F_PARSE_HTTP_LINE1,
205                                         OCSP_R_SERVER_RESPONSE_PARSE_ERROR);
206                 return 0;
207                 }

209         /* Skip past white space to start of response code */
210         while(*p && isspace((unsigned char)*p))
211                 p++;

213         if(!*p)
214                 {
215                 OCSPerr(OCSP_F_PARSE_HTTP_LINE1,
216                                         OCSP_R_SERVER_RESPONSE_PARSE_ERROR);
217                 return 0;
218                 }

220         /* Find end of response code: first whitespace after start of code */
221         for(q = p; *q && !isspace((unsigned char)*q); q++)
222                 continue;

224         if(!*q)
225                 {
226                 OCSPerr(OCSP_F_PARSE_HTTP_LINE1,
227                                         OCSP_R_SERVER_RESPONSE_PARSE_ERROR);
228                 return 0;
229                 }

231         /* Set end of response code and start of message */
232         *q++ = 0;

234         /* Attempt to parse numeric code */
235         retcode = strtoul(p, &r, 10);

237         if(*r)
238                 return 0;

240         /* Skip over any leading white space in message */
241         while(*q && isspace((unsigned char)*q))
242                 q++;

244         if(*q)
245                 {
246                 /* Finally zap any trailing white space in message (include
247                  * CRLF) */

249                 /* We know q has a non white space character so this is OK */
250                 for(r = q + strlen(q) - 1; isspace((unsigned char)*r); r--)
251                         *r = 0;
252                 }
253         if(retcode != 200)
254                 {
255                 OCSPerr(OCSP_F_PARSE_HTTP_LINE1, OCSP_R_SERVER_RESPONSE_ERROR);
256                 if(!*q)
257                         ERR_add_error_data(2, "Code=", p);
258                 else
259                         ERR_add_error_data(4, "Code=", p, ",Reason=", q);
```

```
260                 return 0;
261                 }


264         return 1;

266         }

268 int OCSP_sendreq_nbio(OCSP_RESPONSE **presp, OCSP_REQ_CTX *rctx)
269         {
270         int i, n;
271         const unsigned char *p;
272         next_io:
273         if (!(rctx->state & OHS_NOREAD))
274                 {
275                 n = BIO_read(rctx->io, rctx->iobuf, rctx->iobuflen);

277                 if (n <= 0)
278                         {
279                         if (BIO_should_retry(rctx->io))
280                                 return -1;
281                         return 0;
282                         }

284                 /* Write data to memory BIO */

286                 if (BIO_write(rctx->mem, rctx->iobuf, n) != n)
287                         return 0;
288                 }

290         switch(rctx->state)
291                 {

293                 case OHS_ASN1_WRITE:
294                 n = BIO_get_mem_data(rctx->mem, &p);

296                 i = BIO_write(rctx->io,
297                         p + (n - rctx->asn1_len), rctx->asn1_len);

299                 if (i <= 0)
300                         {
301                         if (BIO_should_retry(rctx->io))
302                                 return -1;
303                         rctx->state = OHS_ERROR;
304                         return 0;
305                         }

307                 rctx->asn1_len -= i;

309                 if (rctx->asn1_len > 0)
310                         goto next_io;

312                 rctx->state = OHS_ASN1_FLUSH;

314                 (void)BIO_reset(rctx->mem);

316                 case OHS_ASN1_FLUSH:

318                 i = BIO_flush(rctx->io);

320                 if (i > 0)
321                         {
322                         rctx->state = OHS_FIRSTLINE;
323                         goto next_io;
324                         }
```



```
326                 if (BIO_should_retry(rctx->io))
327                         return -1;

329                 rctx->state = OHS_ERROR;
330                 return 0;

332                 case OHS_ERROR:
333                 return 0;

335                 case OHS_FIRSTLINE:
336                 case OHS_HEADERS:

338                 /* Attempt to read a line in */

340                 next_line:
341                 /* Due to &%^*$" memory BIO behaviour with BIO_gets we
342                  * have to check there's a complete line in there before
343                  * calling BIO_gets or we'll just get a partial read.
344                  */
345                 n = BIO_get_mem_data(rctx->mem, &p);
346                 if ((n <= 0) || !memchr(p, '\n', n))
347                         {
348                         if (n >= rctx->iobuflen)
349                                 {
350                                 rctx->state = OHS_ERROR;
351                                 return 0;
352                                 }
353                         goto next_io;
354                         }
355                 n = BIO_gets(rctx->mem, (char *)rctx->iobuf, rctx->iobuflen);

357                 if (n <= 0)
358                         {
359                         if (BIO_should_retry(rctx->mem))
360                                 goto next_io;
361                         rctx->state = OHS_ERROR;
362                         return 0;
363                         }

365                 /* Don't allow excessive lines */
366                 if (n == rctx->iobuflen)
367                         {
368                         rctx->state = OHS_ERROR;
369                         return 0;
370                         }

372                 /* First line */
373                 if (rctx->state == OHS_FIRSTLINE)
374                         {
375                         if (parse_http_line1((char *)rctx->iobuf))
376                                 {
377                                 rctx->state = OHS_HEADERS;
378                                 goto next_line;
379                                 }
380                         else
381                                 {
382                                 rctx->state = OHS_ERROR;
383                                 return 0;
384                                 }
385                         }
386                 else
387                         {
388                         /* Look for blank line: end of headers */
389                         for (p = rctx->iobuf; *p; p++)
390                                 {
391                                 if ((*p != '\r') && (*p != '\n'))
```

```
392                                     break;
393                             }
394                     if (*p)
395                             goto next_line;

397                     rctx->state = OHS_ASN1_HEADER;

399                     }

401             /* Fall thru */


404             case OHS_ASN1_HEADER:
405             /* Now reading ASN1 header: can read at least 2 bytes which
406              * is enough for ASN1 SEQUENCE header and either length field
407              * or at least the length of the length field.
408              */
409             n = BIO_get_mem_data(rctx->mem, &p);
410             if (n < 2)
411                     goto next_io;

413             /* Check it is an ASN1 SEQUENCE */
414             if (*p++ != (V_ASN1_SEQUENCE|V_ASN1_CONSTRUCTED))
415                     {
416                     rctx->state = OHS_ERROR;
417                     return 0;
418                     }

420             /* Check out length field */
421             if (*p & 0x80)
422                     {
423                     /* If MSB set on initial length octet we can now
424                      * always read 6 octets: make sure we have them.
425                      */
426                     if (n < 6)
427                             goto next_io;
428                     n = *p & 0x7F;
429                     /* Not NDEF or excessive length */
430                     if (!n || (n > 4))
431                             {
432                             rctx->state = OHS_ERROR;
433                             return 0;
434                             }
435                     p++;
436                     rctx->asn1_len = 0;
437                     for (i = 0; i < n; i++)
438                             {
439                             rctx->asn1_len <<= 8;
440                             rctx->asn1_len |= *p++;
441                             }

443                     if (rctx->asn1_len > OCSP_MAX_REQUEST_LENGTH)
444                             {
445                             rctx->state = OHS_ERROR;
446                             return 0;
447                             }

449                     rctx->asn1_len += n + 2;
450                     }
451             else
452                     rctx->asn1_len = *p + 2;

454             rctx->state = OHS_ASN1_CONTENT;

456             /* Fall thru */
```



```
458             case OHS_ASN1_CONTENT:
459             n = BIO_get_mem_data(rctx->mem, &p);
460             if (n < (int)rctx->asn1_len)
461                     goto next_io;


464             *presp = d2i_OCSP_RESPONSE(NULL, &p, rctx->asn1_len);
465             if (*presp)
466                     {
467                     rctx->state = OHS_DONE;
468                     return 1;
469                     }

471             rctx->state = OHS_ERROR;
472             return 0;

474             break;

476             case OHS_DONE:
477             return 1;

479             }



483         return 0;


486         }

488 /* Blocking OCSP request handler: now a special case of non-blocking I/O */

490 OCSP_RESPONSE *OCSP_sendreq_bio(BIO *b, char *path, OCSP_REQUEST *req)
491         {
492         OCSP_RESPONSE *resp = NULL;
493         OCSP_REQ_CTX *ctx;
494         int rv;

496         ctx = OCSP_sendreq_new(b, path, req, -1);

498         if (!ctx)
499                 return NULL;

501         do
502                 {
503                 rv = OCSP_sendreq_nbio(&resp, ctx);
504                 } while ((rv == -1) && BIO_should_retry(b));

506         OCSP_REQ_CTX_free(ctx);

508         if (rv)
509                 return resp;

511         return NULL;
512         }
513 #endif /* ! codereview */
```

```
**********************************************************
    7311 Wed Aug 13 19:52:58 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_lib.c */
/* Written by Tom Titchener <Tom_Titchener@groove.net> for the OpenSSL
 * project. */

/* History:
   This file was transfered to Richard Levitte from CertCo by Kathy
   Weinhold in mid-spring 2000 to be included in OpenSSL or released
   as a patch kit. */

/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
```



```
 */

#include <stdio.h>
#include <cryptlib.h>
#include <openssl/objects.h>
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/ocsp.h>
#include <openssl/asn1t.h>

/* Convert a certificate and its issuer to an OCSP_CERTID */

OCSP_CERTID *OCSP_cert_to_id(const EVP_MD *dgst, X509 *subject, X509 *issuer)
{
        X509_NAME *iname;
        ASN1_INTEGER *serial;
        ASN1_BIT_STRING *ikey;
#ifndef OPENSSL_NO_SHA1
        if(!dgst) dgst = EVP_sha1();
#endif
        if (subject)
                {
                iname = X509_get_issuer_name(subject);
                serial = X509_get_serialNumber(subject);
                }
        else
                {
                iname = X509_get_subject_name(issuer);
                serial = NULL;
                }
        ikey = X509_get0_pubkey_bitstr(issuer);
        return OCSP_cert_id_new(dgst, iname, ikey, serial);
}


OCSP_CERTID *OCSP_cert_id_new(const EVP_MD *dgst,
                              X509_NAME *issuerName,
                              ASN1_BIT_STRING* issuerKey,
                              ASN1_INTEGER *serialNumber)
        {
        int nid;
        unsigned int i;
        X509_ALGOR *alg;
        OCSP_CERTID *cid = NULL;
        unsigned char md[EVP_MAX_MD_SIZE];

        if (!(cid = OCSP_CERTID_new())) goto err;

        alg = cid->hashAlgorithm;
        if (alg->algorithm != NULL) ASN1_OBJECT_free(alg->algorithm);
        if ((nid = EVP_MD_type(dgst)) == NID_undef)
                {
                OCSPerr(OCSP_F_OCSP_CERT_ID_NEW,OCSP_R_UNKNOWN_NID);
                goto err;
                }
        if (!(alg->algorithm=OBJ_nid2obj(nid))) goto err;
        if ((alg->parameter=ASN1_TYPE_new()) == NULL) goto err;
        alg->parameter->type=V_ASN1_NULL;

        if (!X509_NAME_digest(issuerName, dgst, md, &i)) goto digerr;
        if (!(ASN1_OCTET_STRING_set(cid->issuerNameHash, md, i))) goto err;

        /* Calculate the issuerKey hash, excluding tag and length */
        if (!EVP_Digest(issuerKey->data, issuerKey->length, md, &i, dgst, NULL))
```

```
128                 goto err;

130         if (!(ASN1_OCTET_STRING_set(cid->issuerKeyHash, md, i))) goto err;

132         if (serialNumber)
133                 {
134                 ASN1_INTEGER_free(cid->serialNumber);
135                 if (!(cid->serialNumber = ASN1_INTEGER_dup(serialNumber))) goto
136                 }
137         return cid;
138 digerr:
139         OCSPerr(OCSP_F_OCSP_CERT_ID_NEW,OCSP_R_DIGEST_ERR);
140 err:
141         if (cid) OCSP_CERTID_free(cid);
142         return NULL;
143         }

145 int OCSP_id_issuer_cmp(OCSP_CERTID *a, OCSP_CERTID *b)
146         {
147         int ret;
148         ret = OBJ_cmp(a->hashAlgorithm->algorithm, b->hashAlgorithm->algorithm);
149         if (ret) return ret;
150         ret = ASN1_OCTET_STRING_cmp(a->issuerNameHash, b->issuerNameHash);
151         if (ret) return ret;
152         return ASN1_OCTET_STRING_cmp(a->issuerKeyHash, b->issuerKeyHash);
153         }

155 int OCSP_id_cmp(OCSP_CERTID *a, OCSP_CERTID *b)
156         {
157         int ret;
158         ret = OCSP_id_issuer_cmp(a, b);
159         if (ret) return ret;
160         return ASN1_INTEGER_cmp(a->serialNumber, b->serialNumber);
161         }


164 /* Parse a URL and split it up into host, port and path components and whether
165  * it is SSL.
166  */

168 int OCSP_parse_url(char *url, char **phost, char **pport, char **ppath, int *pss
169         {
170         char *p, *buf;

172         char *host, *port;

174         *phost = NULL;
175         *pport = NULL;
176         *ppath = NULL;

178         /* dup the buffer since we are going to mess with it */
179         buf = BUF_strdup(url);
180         if (!buf) goto mem_err;

182         /* Check for initial colon */
183         p = strchr(buf, ':');

185         if (!p) goto parse_err;

187         *(p++) = '\0';

189         if (!strcmp(buf, "http"))
190                 {
191                 *pssl = 0;
192                 port = "80";
193                 }
```



```
194         else if (!strcmp(buf, "https"))
195                 {
196                 *pssl = 1;
197                 port = "443";
198                 }
199         else
200                 goto parse_err;

202         /* Check for double slash */
203         if ((p[0] != '/') || (p[1] != '/'))
204                 goto parse_err;

206         p += 2;

208         host = p;

210         /* Check for trailing part of path */

212         p = strchr(p, '/');

214         if (!p)
215                 *ppath = BUF_strdup("/");
216         else
217                 {
218                 *ppath = BUF_strdup(p);
219                 /* Set start of path to 0 so hostname is valid */
220                 *p = '\0';
221                 }

223         if (!*ppath) goto mem_err;

225         p = host;
226         if(host[0] == '[')
227                 {
228                 /* ipv6 literal */
229                 host++;
230                 p = strchr(host, ']');
231                 if(!p) goto parse_err;
232                 *p = '\0';
233                 p++;
234                 }

236         /* Look for optional ':' for port number */
237         if ((p = strchr(p, ':')))
238                 {
239                 *p = 0;
240                 port = p + 1;
241                 }
242         else
243                 {
244                 /* Not found: set default port */
245                 if (*pssl) port = "443";
246                 else port = "80";
247                 }

249         *pport = BUF_strdup(port);
250         if (!*pport) goto mem_err;

252         *phost = BUF_strdup(host);

254         if (!*phost) goto mem_err;

256         OPENSSL_free(buf);

258         return 1;
```

```
260         mem_err:
261         OCSPerr(OCSP_F_OCSP_PARSE_URL, ERR_R_MALLOC_FAILURE);
262         goto err;

264         parse_err:
265         OCSPerr(OCSP_F_OCSP_PARSE_URL, OCSP_R_ERROR_PARSING_URL);


268         err:
269         if (buf) OPENSSL_free(buf);
270         if (*ppath) OPENSSL_free(*ppath);
271         if (*pport) OPENSSL_free(*pport);
272         if (*phost) OPENSSL_free(*phost);
273         return 0;

275         }

277 IMPLEMENT_ASN1_DUP_FUNCTION(OCSP_CERTID)
278 #endif /* ! codereview */
```

```
**********************************************************
   10190 Wed Aug 13 19:52:58 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_prn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_prn.c */
/* Written by Tom Titchener <Tom_Titchener@groove.net> for the OpenSSL
 * project. */

/* History:
   This file was originally part of ocsp.c and was transfered to Richard
   Levitte from CertCo by Kathy Weinhold in mid-spring 2000 to be included
   in OpenSSL or released as a patch kit. */

/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
```



```
 */

#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/ocsp.h>
#include <openssl/pem.h>

static int ocsp_certid_print(BIO *bp, OCSP_CERTID* a, int indent)
        {
        BIO_printf(bp, "%*sCertificate ID:\n", indent, "");
        indent += 2;
        BIO_printf(bp, "%*sHash Algorithm: ", indent, "");
        i2a_ASN1_OBJECT(bp, a->hashAlgorithm->algorithm);
        BIO_printf(bp, "\n%*sIssuer Name Hash: ", indent, "");
        i2a_ASN1_STRING(bp, a->issuerNameHash, V_ASN1_OCTET_STRING);
        BIO_printf(bp, "\n%*sIssuer Key Hash: ", indent, "");
        i2a_ASN1_STRING(bp, a->issuerKeyHash, V_ASN1_OCTET_STRING);
        BIO_printf(bp, "\n%*sSerial Number: ", indent, "");
        i2a_ASN1_INTEGER(bp, a->serialNumber);
        BIO_printf(bp, "\n");
        return 1;
        }

typedef struct
        {
        long t;
        const char *m;
        } OCSP_TBLSTR;

static const char *table2string(long s, const OCSP_TBLSTR *ts, int len)
{
        const OCSP_TBLSTR *p;
        for (p=ts; p < ts + len; p++)
                if (p->t == s)
                        return p->m;
        return "(UNKNOWN)";
}

const char *OCSP_response_status_str(long s)
        {
        static const OCSP_TBLSTR rstat_tbl[] = {
                { OCSP_RESPONSE_STATUS_SUCCESSFUL, "successful" },
                { OCSP_RESPONSE_STATUS_MALFORMEDREQUEST, "malformedrequest" },
                { OCSP_RESPONSE_STATUS_INTERNALERROR, "internalerror" },
                { OCSP_RESPONSE_STATUS_TRYLATER, "trylater" },
                { OCSP_RESPONSE_STATUS_SIGREQUIRED, "sigrequired" },
                { OCSP_RESPONSE_STATUS_UNAUTHORIZED, "unauthorized" } };
        return table2string(s, rstat_tbl, 6);
        }

const char *OCSP_cert_status_str(long s)
        {
        static const OCSP_TBLSTR cstat_tbl[] = {
                { V_OCSP_CERTSTATUS_GOOD, "good" },
                { V_OCSP_CERTSTATUS_REVOKED, "revoked" },
                { V_OCSP_CERTSTATUS_UNKNOWN, "unknown" } };
        return table2string(s, cstat_tbl, 3);
        }

const char *OCSP_crl_reason_str(long s)
        {
        static const OCSP_TBLSTR reason_tbl[] = {
          { OCSP_REVOKED_STATUS_UNSPECIFIED, "unspecified" },
          { OCSP_REVOKED_STATUS_KEYCOMPROMISE, "keyCompromise" },
          { OCSP_REVOKED_STATUS_CACOMPROMISE, "cACompromise" },
          { OCSP_REVOKED_STATUS_AFFILIATIONCHANGED, "affiliationChanged" },
```

```
128           { OCSP_REVOKED_STATUS_SUPERSEDED, "superseded" },
129           { OCSP_REVOKED_STATUS_CESSATIONOFOPERATION, "cessationOfOperation" },
130           { OCSP_REVOKED_STATUS_CERTIFICATEHOLD, "certificateHold" },
131           { OCSP_REVOKED_STATUS_REMOVEFROMCRL, "removeFromCRL" } };
132         return table2string(s, reason_tbl, 8);
133         }

135 int OCSP_REQUEST_print(BIO *bp, OCSP_REQUEST* o, unsigned long flags)
136         {
137         int i;
138         long l;
139         OCSP_CERTID* cid = NULL;
140         OCSP_ONEREQ *one = NULL;
141         OCSP_REQINFO *inf = o->tbsRequest;
142         OCSP_SIGNATURE *sig = o->optionalSignature;

144         if (BIO_write(bp,"OCSP Request Data:\n",19) <= 0) goto err;
145         l=ASN1_INTEGER_get(inf->version);
146         if (BIO_printf(bp,"    Version: %lu (0x%lx)",l+1,l) <= 0) goto err;
147         if (inf->requestorName != NULL)
148                 {
149                 if (BIO_write(bp,"\n    Requestor Name: ",21) <= 0)
150                         goto err;
151                 GENERAL_NAME_print(bp, inf->requestorName);
152                 }
153         if (BIO_write(bp,"\n    Requestor List:\n",21) <= 0) goto err;
154         for (i = 0; i < sk_OCSP_ONEREQ_num(inf->requestList); i++)
155                 {
156                 one = sk_OCSP_ONEREQ_value(inf->requestList, i);
157                 cid = one->reqCert;
158                 ocsp_certid_print(bp, cid, 8);
159                 if (!X509V3_extensions_print(bp,
160                                              "Request Single Extensions",
161                                              one->singleRequestExtensions, flags, 8))
162                                                         goto err;
163                 }
164         if (!X509V3_extensions_print(bp, "Request Extensions",
165                         inf->requestExtensions, flags, 4))
166                                                         goto err;
167         if (sig)
168                 {
169                 X509_signature_print(bp, sig->signatureAlgorithm, sig->signature
170                 for (i=0; i<sk_X509_num(sig->certs); i++)
171                         {
172                         X509_print(bp, sk_X509_value(sig->certs,i));
173                         PEM_write_bio_X509(bp,sk_X509_value(sig->certs,i));
174                         }
175                 }
176         return 1;
177 err:
178         return 0;
179         }

181 int OCSP_RESPONSE_print(BIO *bp, OCSP_RESPONSE* o, unsigned long flags)
182         {
183         int i, ret = 0;
184         long l;
185         OCSP_CERTID *cid = NULL;
186         OCSP_BASICRESP *br = NULL;
187         OCSP_RESPID *rid = NULL;
188         OCSP_RESPDATA  *rd = NULL;
189         OCSP_CERTSTATUS *cst = NULL;
190         OCSP_REVOKEDINFO *rev = NULL;
191         OCSP_SINGLERESP *single = NULL;
192         OCSP_RESPBYTES *rb = o->responseBytes;
```



```
194         if (BIO_puts(bp,"OCSP Response Data:\n") <= 0) goto err;
195         l=ASN1_ENUMERATED_get(o->responseStatus);
196         if (BIO_printf(bp,"    OCSP Response Status: %s (0x%lx)\n",
197                                 OCSP_response_status_str(l), l) <= 0) goto err;
198         if (rb == NULL) return 1;
199         if (BIO_puts(bp,"    Response Type: ") <= 0)
200                 goto err;
201         if(i2a_ASN1_OBJECT(bp, rb->responseType) <= 0)
202                 goto err;
203         if (OBJ_obj2nid(rb->responseType) != NID_id_pkix_OCSP_basic)
204                 {
205                 BIO_puts(bp," (unknown response type)\n");
206                 return 1;
207                 }

209         i = ASN1_STRING_length(rb->response);
210         if (!(br = OCSP_response_get1_basic(o))) goto err;
211         rd = br->tbsResponseData;
212         l=ASN1_INTEGER_get(rd->version);
213         if (BIO_printf(bp,"\n    Version: %lu (0x%lx)\n",
214                                 l+1,l) <= 0) goto err;
215         if (BIO_puts(bp,"    Responder Id: ") <= 0) goto err;

217         rid =  rd->responderId;
218         switch (rid->type)
219                 {
220                 case V_OCSP_RESPID_NAME:
221                         X509_NAME_print_ex(bp, rid->value.byName, 0, XN_FLAG_ONE
222                         break;
223                 case V_OCSP_RESPID_KEY:
224                         i2a_ASN1_STRING(bp, rid->value.byKey, V_ASN1_OCTET_STRIN
225                         break;
226                 }

228         if (BIO_printf(bp,"\n    Produced At: ")<=0) goto err;
229         if (!ASN1_GENERALIZEDTIME_print(bp, rd->producedAt)) goto err;
230         if (BIO_printf(bp,"\n    Responses:\n") <= 0) goto err;
231         for (i = 0; i < sk_OCSP_SINGLERESP_num(rd->responses); i++)
232                 {
233                 if (! sk_OCSP_SINGLERESP_value(rd->responses, i)) continue;
234                 single = sk_OCSP_SINGLERESP_value(rd->responses, i);
235                 cid = single->certId;
236                 if(ocsp_certid_print(bp, cid, 4) <= 0) goto err;
237                 cst = single->certStatus;
238                 if (BIO_printf(bp,"    Cert Status: %s",
239                                 OCSP_cert_status_str(cst->type)) <= 0)
240                         goto err;
241                 if (cst->type == V_OCSP_CERTSTATUS_REVOKED)
242                         {
243                         rev = cst->value.revoked;
244                         if (BIO_printf(bp, "\n    Revocation Time: ") <= 0)
245                                 goto err;
246                         if (!ASN1_GENERALIZEDTIME_print(bp,
247                                                         rev->revocationTime))
248                                 goto err;
249                         if (rev->revocationReason)
250                                 {
251                                 l=ASN1_ENUMERATED_get(rev->revocationReason);
252                                 if (BIO_printf(bp,
253                                         "\n    Revocation Reason: %s (0x%lx)",
254                                              OCSP_crl_reason_str(l), l) <= 0)
255                                         goto err;
256                                 }
257                         }
258                 if (BIO_printf(bp,"\n    This Update: ") <= 0) goto err;
259                 if (!ASN1_GENERALIZEDTIME_print(bp, single->thisUpdate))
```

```
260                         goto err;
261                 if (single->nextUpdate)
262                         {
263                         if (BIO_printf(bp,"\n    Next Update: ") <= 0)goto err;
264                         if (!ASN1_GENERALIZEDTIME_print(bp,single->nextUpdate))
265                                 goto err;
266                         }
267                 if (BIO_write(bp,"\n",1) <= 0) goto err;
268                 if (!X509V3_extensions_print(bp,
269                                              "Response Single Extensions",
270                                              single->singleExtensions, flags, 8))
271                                                         goto err;
272                 if (BIO_write(bp,"\n",1) <= 0) goto err;
273                 }
274         if (!X509V3_extensions_print(bp, "Response Extensions",
275                                              rd->responseExtensions, flags, 4))
276                                                         goto err;
277         if(X509_signature_print(bp, br->signatureAlgorithm, br->signature) <= 0)
278                                                         goto err;

280         for (i=0; i<sk_X509_num(br->certs); i++)
281                 {
282                 X509_print(bp, sk_X509_value(br->certs,i));
283                 PEM_write_bio_X509(bp,sk_X509_value(br->certs,i));
284                 }

286         ret = 1;
287 err:
288         OCSP_BASICRESP_free(br);
289         return ret;
290         }
291 #endif /* ! codereview */
```

```
**********************************************************
    7814 Wed Aug 13 19:52:58 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_srv.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_srv.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <cryptlib.h>
#include <openssl/objects.h>
```



```
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/x509v3.h>
#include <openssl/ocsp.h>

/* Utility functions related to sending OCSP responses and extracting
 * relevant information from the request.
 */

int OCSP_request_onereq_count(OCSP_REQUEST *req)
        {
        return sk_OCSP_ONEREQ_num(req->tbsRequest->requestList);
        }

OCSP_ONEREQ *OCSP_request_onereq_get0(OCSP_REQUEST *req, int i)
        {
        return sk_OCSP_ONEREQ_value(req->tbsRequest->requestList, i);
        }

OCSP_CERTID *OCSP_onereq_get0_id(OCSP_ONEREQ *one)
        {
        return one->reqCert;
        }

int OCSP_id_get0_info(ASN1_OCTET_STRING **piNameHash, ASN1_OBJECT **pmd,
                      ASN1_OCTET_STRING **pikeyHash,
                      ASN1_INTEGER **pserial, OCSP_CERTID *cid)
        {
        if (!cid) return 0;
        if (pmd) *pmd = cid->hashAlgorithm->algorithm;
        if(piNameHash) *piNameHash = cid->issuerNameHash;
        if (pikeyHash) *pikeyHash = cid->issuerKeyHash;
        if (pserial) *pserial = cid->serialNumber;
        return 1;
        }

int OCSP_request_is_signed(OCSP_REQUEST *req)
        {
        if(req->optionalSignature) return 1;
        return 0;
        }

/* Create an OCSP response and encode an optional basic response */
OCSP_RESPONSE *OCSP_response_create(int status, OCSP_BASICRESP *bs)
        {
        OCSP_RESPONSE *rsp = NULL;

        if (!(rsp = OCSP_RESPONSE_new())) goto err;
        if (!(ASN1_ENUMERATED_set(rsp->responseStatus, status))) goto err;
        if (!bs) return rsp;
        if (!(rsp->responseBytes = OCSP_RESPBYTES_new())) goto err;
        rsp->responseBytes->responseType = OBJ_nid2obj(NID_id_pkix_OCSP_basic);
        if (!ASN1_item_pack(bs, ASN1_ITEM_rptr(OCSP_BASICRESP), &rsp->responseBy
                                goto err;
        return rsp;
err:
        if (rsp) OCSP_RESPONSE_free(rsp);
        return NULL;
        }


OCSP_SINGLERESP *OCSP_basic_add1_status(OCSP_BASICRESP *rsp,
                                                OCSP_CERTID *cid,
                                                int status, int reason,
                                                ASN1_TIME *revtime,
```

```
128                                     ASN1_TIME *thisupd, ASN1_TIME *nextupd)
129         {
130         OCSP_SINGLERESP *single = NULL;
131         OCSP_CERTSTATUS *cs;
132         OCSP_REVOKEDINFO *ri;
133
134         if(!rsp->tbsResponseData->responses &&
135             !(rsp->tbsResponseData->responses = sk_OCSP_SINGLERESP_new_null()))
136                 goto err;
137
138         if (!(single = OCSP_SINGLERESP_new()))
139                 goto err;
140
141
142
143         if (!ASN1_TIME_to_generalizedtime(thisupd, &single->thisUpdate))
144                 goto err;
145         if (nextupd &&
146                 !ASN1_TIME_to_generalizedtime(nextupd, &single->nextUpdate))
147                 goto err;
148
149         OCSP_CERTID_free(single->certId);
150
151         if(!(single->certId = OCSP_CERTID_dup(cid)))
152                 goto err;
153
154         cs = single->certStatus;
155         switch(cs->type = status)
156                 {
157         case V_OCSP_CERTSTATUS_REVOKED:
158                 if (!revtime)
159                         {
160                         OCSPerr(OCSP_F_OCSP_BASIC_ADD1_STATUS,OCSP_R_NO_REVOKED_
161                         goto err;
162                         }
163                 if (!(cs->value.revoked = ri = OCSP_REVOKEDINFO_new())) goto err
164                 if (!ASN1_TIME_to_generalizedtime(revtime, &ri->revocationTime))
165                         goto err;
166                 if (reason != OCSP_REVOKED_STATUS_NOSTATUS)
167                         {
168                         if (!(ri->revocationReason = ASN1_ENUMERATED_new()))
169                                 goto err;
170                         if (!(ASN1_ENUMERATED_set(ri->revocationReason,
171                                                   reason)))
172                                 goto err;
173                         }
174                 break;
175
176         case V_OCSP_CERTSTATUS_GOOD:
177                 cs->value.good = ASN1_NULL_new();
178                 break;
179
180         case V_OCSP_CERTSTATUS_UNKNOWN:
181                 cs->value.unknown = ASN1_NULL_new();
182                 break;
183
184         default:
185                 goto err;
186
187                 }
188         if (!(sk_OCSP_SINGLERESP_push(rsp->tbsResponseData->responses, single)))
189                 goto err;
190         return single;
191 err:
192         OCSP_SINGLERESP_free(single);
193         return NULL;
```



```
194         }
195
196 /* Add a certificate to an OCSP request */
197
198 int OCSP_basic_add1_cert(OCSP_BASICRESP *resp, X509 *cert)
199         {
200         if (!resp->certs && !(resp->certs = sk_X509_new_null()))
201                 return 0;
202
203         if(!sk_X509_push(resp->certs, cert)) return 0;
204         CRYPTO_add(&cert->references, 1, CRYPTO_LOCK_X509);
205         return 1;
206         }
207
208 int OCSP_basic_sign(OCSP_BASICRESP *brsp,
209                         X509 *signer, EVP_PKEY *key, const EVP_MD *dgst,
210                         STACK_OF(X509) *certs, unsigned long flags)
211         {
212         int i;
213         OCSP_RESPID *rid;
214
215         if (!X509_check_private_key(signer, key))
216                 {
217                 OCSPerr(OCSP_F_OCSP_BASIC_SIGN, OCSP_R_PRIVATE_KEY_DOES_NOT_MATC
218                 goto err;
219                 }
220
221         if(!(flags & OCSP_NOCERTS))
222                 {
223                 if(!OCSP_basic_add1_cert(brsp, signer))
224                         goto err;
225                 for (i = 0; i < sk_X509_num(certs); i++)
226                         {
227                         X509 *tmpcert = sk_X509_value(certs, i);
228                         if(!OCSP_basic_add1_cert(brsp, tmpcert))
229                                 goto err;
230                         }
231                 }
232
233         rid = brsp->tbsResponseData->responderId;
234         if (flags & OCSP_RESPID_KEY)
235                 {
236                 unsigned char md[SHA_DIGEST_LENGTH];
237                 X509_pubkey_digest(signer, EVP_sha1(), md, NULL);
238                 if (!(rid->value.byKey = ASN1_OCTET_STRING_new()))
239                         goto err;
240                 if (!(ASN1_OCTET_STRING_set(rid->value.byKey, md, SHA_DIGEST_LEN
241                         goto err;
242                 rid->type = V_OCSP_RESPID_KEY;
243                 }
244         else
245                 {
246                 if (!X509_NAME_set(&rid->value.byName,
247                                    X509_get_subject_name(signer)))
248                         goto err;
249                 rid->type = V_OCSP_RESPID_NAME;
250                 }
251
252         if (!(flags & OCSP_NOTIME) &&
253                 !X509_gmtime_adj(brsp->tbsResponseData->producedAt, 0))
254                 goto err;
255
256         /* Right now, I think that not doing double hashing is the right
257            thing.       -- Richard Levitte */
258
259         if (!OCSP_BASICRESP_sign(brsp, key, dgst, 0)) goto err;
```

```
261         return 1;
262 err:
263         return 0;
264         }
265 #endif /* ! codereview */
```

```
**********************************************************
   12862 Wed Aug 13 19:52:58 2014
new/usr/src/lib/openssl/libsunw_crypto/ocsp/ocsp_vfy.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ocsp_vfy.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/ocsp.h>
#include <openssl/err.h>
#include <string.h>
```



```
static int ocsp_find_signer(X509 **psigner, OCSP_BASICRESP *bs, STACK_OF(X509) *
                                X509_STORE *st, unsigned long flags);
static X509 *ocsp_find_signer_sk(STACK_OF(X509) *certs, OCSP_RESPID *id);
static int ocsp_check_issuer(OCSP_BASICRESP *bs, STACK_OF(X509) *chain, unsigned
static int ocsp_check_ids(STACK_OF(OCSP_SINGLERESP) *sresp, OCSP_CERTID **ret);
static int ocsp_match_issuerid(X509 *cert, OCSP_CERTID *cid, STACK_OF(OCSP_SINGL
static int ocsp_check_delegated(X509 *x, int flags);
static int ocsp_req_find_signer(X509 **psigner, OCSP_REQUEST *req, X509_NAME *nm
                                X509_STORE *st, unsigned long flags);

/* Verify a basic response message */

int OCSP_basic_verify(OCSP_BASICRESP *bs, STACK_OF(X509) *certs,
                                X509_STORE *st, unsigned long flags)
        {
        X509 *signer, *x;
        STACK_OF(X509) *chain = NULL;
        X509_STORE_CTX ctx;
        int i, ret = 0;
        ret = ocsp_find_signer(&signer, bs, certs, st, flags);
        if (!ret)
                {
                OCSPerr(OCSP_F_OCSP_BASIC_VERIFY, OCSP_R_SIGNER_CERTIFICATE_NOT_
                goto end;
                }
        if ((ret == 2) && (flags & OCSP_TRUSTOTHER))
                flags |= OCSP_NOVERIFY;
        if (!(flags & OCSP_NOSIGS))
                {
                EVP_PKEY *skey;
                skey = X509_get_pubkey(signer);
                if (skey)
                        {
                        ret = OCSP_BASICRESP_verify(bs, skey, 0);
                        EVP_PKEY_free(skey);
                        }
                if(!skey || ret <= 0)
                        {
                        OCSPerr(OCSP_F_OCSP_BASIC_VERIFY, OCSP_R_SIGNATURE_FAILU
                        goto end;
                        }
                }
        if (!(flags & OCSP_NOVERIFY))
                {
                int init_res;
                if(flags & OCSP_NOCHAIN)
                        init_res = X509_STORE_CTX_init(&ctx, st, signer, NULL);
                else
                        init_res = X509_STORE_CTX_init(&ctx, st, signer, bs->cer
                if(!init_res)
                        {
                        ret = -1;
                        OCSPerr(OCSP_F_OCSP_BASIC_VERIFY,ERR_R_X509_LIB);
                        goto end;
                        }

                X509_STORE_CTX_set_purpose(&ctx, X509_PURPOSE_OCSP_HELPER);
                ret = X509_verify_cert(&ctx);
                chain = X509_STORE_CTX_get1_chain(&ctx);
                X509_STORE_CTX_cleanup(&ctx);
                if (ret <= 0)
                        {
                        i = X509_STORE_CTX_get_error(&ctx);
                        OCSPerr(OCSP_F_OCSP_BASIC_VERIFY,OCSP_R_CERTIFICATE_VERI
                        ERR_add_error_data(2, "Verify error:",
```

```
128                                         X509_verify_cert_error_string(i));
129                         goto end;
130                         }
131                 if(flags & OCSP_NOCHECKS)
132                         {
133                         ret = 1;
134                         goto end;
135                         }
136                 /* At this point we have a valid certificate chain
137                  * need to verify it against the OCSP issuer criteria.
138                  */
139                 ret = ocsp_check_issuer(bs, chain, flags);
140
141                 /* If fatal error or valid match then finish */
142                 if (ret != 0) goto end;
143
144                 /* Easy case: explicitly trusted. Get root CA and
145                  * check for explicit trust
146                  */
147                 if(flags & OCSP_NOEXPLICIT) goto end;
148
149                 x = sk_X509_value(chain, sk_X509_num(chain) - 1);
150                 if(X509_check_trust(x, NID_OCSP_sign, 0) != X509_TRUST_TRUSTED)
151                         {
152                         OCSPerr(OCSP_F_OCSP_BASIC_VERIFY,OCSP_R_ROOT_CA_NOT_TRUS
153                         goto end;
154                         }
155                 ret = 1;
156                 }
157
158
159
160         end:
161         if(chain) sk_X509_pop_free(chain, X509_free);
162         return ret;
163         }
164
165
166 static int ocsp_find_signer(X509 **psigner, OCSP_BASICRESP *bs, STACK_OF(X509) *
167                                 X509_STORE *st, unsigned long flags)
168         {
169         X509 *signer;
170         OCSP_RESPID *rid = bs->tbsResponseData->responderId;
171         if ((signer = ocsp_find_signer_sk(certs, rid)))
172                 {
173                 *psigner = signer;
174                 return 2;
175                 }
176         if(!(flags & OCSP_NOINTERN) &&
177             (signer = ocsp_find_signer_sk(bs->certs, rid)))
178                 {
179                 *psigner = signer;
180                 return 1;
181                 }
182         /* Maybe lookup from store if by subject name */
183
184         *psigner = NULL;
185         return 0;
186         }
187
188
189 static X509 *ocsp_find_signer_sk(STACK_OF(X509) *certs, OCSP_RESPID *id)
190         {
191         int i;
192         unsigned char tmphash[SHA_DIGEST_LENGTH], *keyhash;
193         X509 *x;
```



```
194
195         /* Easy if lookup by name */
196         if (id->type == V_OCSP_RESPID_NAME)
197                 return X509_find_by_subject(certs, id->value.byName);
198
199         /* Lookup by key hash */
200
201         /* If key hash isn't SHA1 length then forget it */
202         if (id->value.byKey->length != SHA_DIGEST_LENGTH) return NULL;
203         keyhash = id->value.byKey->data;
204         /* Calculate hash of each key and compare */
205         for (i = 0; i < sk_X509_num(certs); i++)
206                 {
207                 x = sk_X509_value(certs, i);
208                 X509_pubkey_digest(x, EVP_sha1(), tmphash, NULL);
209                 if(!memcmp(keyhash, tmphash, SHA_DIGEST_LENGTH))
210                         return x;
211                 }
212         return NULL;
213         }
214
215
216 static int ocsp_check_issuer(OCSP_BASICRESP *bs, STACK_OF(X509) *chain, unsigned
217         {
218         STACK_OF(OCSP_SINGLERESP) *sresp;
219         X509 *signer, *sca;
220         OCSP_CERTID *caid = NULL;
221         int i;
222         sresp = bs->tbsResponseData->responses;
223
224         if (sk_X509_num(chain) <= 0)
225                 {
226                 OCSPerr(OCSP_F_OCSP_CHECK_ISSUER, OCSP_R_NO_CERTIFICATES_IN_CHAI
227                 return -1;
228                 }
229
230         /* See if the issuer IDs match. */
231         i = ocsp_check_ids(sresp, &caid);
232
233         /* If ID mismatch or other error then return */
234         if (i <= 0) return i;
235
236         signer = sk_X509_value(chain, 0);
237         /* Check to see if OCSP responder CA matches request CA */
238         if (sk_X509_num(chain) > 1)
239                 {
240                 sca = sk_X509_value(chain, 1);
241                 i = ocsp_match_issuerid(sca, caid, sresp);
242                 if (i < 0) return i;
243                 if (i)
244                         {
245                         /* We have a match, if extensions OK then success */
246                         if (ocsp_check_delegated(signer, flags)) return 1;
247                         return 0;
248                         }
249                 }
250
251         /* Otherwise check if OCSP request signed directly by request CA */
252         return ocsp_match_issuerid(signer, caid, sresp);
253         }
254
255
256 /* Check the issuer certificate IDs for equality. If there is a mismatch with th
257  * algorithm then there's no point trying to match any certificates against the
258  * If the issuer IDs all match then we just need to check equality against one o
259  */
```

```
261 static int ocsp_check_ids(STACK_OF(OCSP_SINGLERESP) *sresp, OCSP_CERTID **ret)
262         {
263         OCSP_CERTID *tmpid, *cid;
264         int i, idcount;
265
266         idcount = sk_OCSP_SINGLERESP_num(sresp);
267         if (idcount <= 0)
268                 {
269                 OCSPerr(OCSP_F_OCSP_CHECK_IDS, OCSP_R_RESPONSE_CONTAINS_NO_REVOC
270                 return -1;
271                 }
272
273         cid = sk_OCSP_SINGLERESP_value(sresp, 0)->certId;
274
275         *ret = NULL;
276
277         for (i = 1; i < idcount; i++)
278                 {
279                 tmpid = sk_OCSP_SINGLERESP_value(sresp, i)->certId;
280                 /* Check to see if IDs match */
281                 if (OCSP_id_issuer_cmp(cid, tmpid))
282                         {
283                         /* If algoritm mismatch let caller deal with it */
284                         if (OBJ_cmp(tmpid->hashAlgorithm->algorithm,
285                                         cid->hashAlgorithm->algorithm))
286                                         return 2;
287                         /* Else mismatch */
288                         return 0;
289                         }
290                 }
291
292         /* All IDs match: only need to check one ID */
293         *ret = cid;
294         return 1;
295         }
296
297
298 static int ocsp_match_issuerid(X509 *cert, OCSP_CERTID *cid,
299                         STACK_OF(OCSP_SINGLERESP) *sresp)
300         {
301         /* If only one ID to match then do it */
302         if(cid)
303                 {
304                 const EVP_MD *dgst;
305                 X509_NAME *iname;
306                 int mdlen;
307                 unsigned char md[EVP_MAX_MD_SIZE];
308                 if (!(dgst = EVP_get_digestbyobj(cid->hashAlgorithm->algorithm))
309                         {
310                         OCSPerr(OCSP_F_OCSP_MATCH_ISSUERID, OCSP_R_UNKNOWN_MESSA
311                         return -1;
312                         }
313
314                 mdlen = EVP_MD_size(dgst);
315                 if (mdlen < 0)
316                  return -1;
317                 if ((cid->issuerNameHash->length != mdlen) ||
318                    (cid->issuerKeyHash->length != mdlen))
319                         return 0;
320                 iname = X509_get_subject_name(cert);
321                 if (!X509_NAME_digest(iname, dgst, md, NULL))
322                         return -1;
323                 if (memcmp(md, cid->issuerNameHash->data, mdlen))
324                         return 0;
325                 X509_pubkey_digest(cert, dgst, md, NULL);
```



```
326                 if (memcmp(md, cid->issuerKeyHash->data, mdlen))
327                         return 0;
328
329                 return 1;
330
331                 }
332         else
333                 {
334                 /* We have to match the whole lot */
335                 int i, ret;
336                 OCSP_CERTID *tmpid;
337                 for (i = 0; i < sk_OCSP_SINGLERESP_num(sresp); i++)
338                         {
339                         tmpid = sk_OCSP_SINGLERESP_value(sresp, i)->certId;
340                         ret = ocsp_match_issuerid(cert, tmpid, NULL);
341                         if (ret <= 0) return ret;
342                         }
343                 return 1;
344                 }
345
346         }
347
348 static int ocsp_check_delegated(X509 *x, int flags)
349         {
350         X509_check_purpose(x, -1, 0);
351         if ((x->ex_flags & EXFLAG_XKUSAGE) &&
352             (x->ex_xkusage & XKU_OCSP_SIGN))
353                 return 1;
354         OCSPerr(OCSP_F_OCSP_CHECK_DELEGATED, OCSP_R_MISSING_OCSPSIGNING_USAGE);
355         return 0;
356         }
357
358 /* Verify an OCSP request. This is fortunately much easier than OCSP
359  * response verify. Just find the signers certificate and verify it
360  * against a given trust value.
361  */
362
363 int OCSP_request_verify(OCSP_REQUEST *req, STACK_OF(X509) *certs, X509_STORE *st
364         {
365         X509 *signer;
366         X509_NAME *nm;
367         GENERAL_NAME *gen;
368         int ret;
369         X509_STORE_CTX ctx;
370         if (!req->optionalSignature)
371                 {
372                 OCSPerr(OCSP_F_OCSP_REQUEST_VERIFY, OCSP_R_REQUEST_NOT_SIGNED);
373                 return 0;
374                 }
375         gen = req->tbsRequest->requestorName;
376         if (!gen || gen->type != GEN_DIRNAME)
377                 {
378                 OCSPerr(OCSP_F_OCSP_REQUEST_VERIFY, OCSP_R_UNSUPPORTED_REQUESTOR
379                 return 0;
380                 }
381         nm = gen->d.directoryName;
382         ret = ocsp_req_find_signer(&signer, req, nm, certs, store, flags);
383         if (ret <= 0)
384                 {
385                 OCSPerr(OCSP_F_OCSP_REQUEST_VERIFY, OCSP_R_SIGNER_CERTIFICATE_NO
386                 return 0;
387                 }
388         if ((ret == 2) && (flags & OCSP_TRUSTOTHER))
389                 flags |= OCSP_NOVERIFY;
390         if (!(flags & OCSP_NOSIGS))
391                 {
```

```
392                 EVP_PKEY *skey;
393                 skey = X509_get_pubkey(signer);
394                 ret = OCSP_REQUEST_verify(req, skey);
395                 EVP_PKEY_free(skey);
396                 if(ret <= 0)
397                         {
398                         OCSPerr(OCSP_F_OCSP_REQUEST_VERIFY, OCSP_R_SIGNATURE_FAI
399                         return 0;
400                         }
401                 }
402         if (!(flags & OCSP_NOVERIFY))
403                 {
404                 int init_res;
405                 if(flags & OCSP_NOCHAIN)
406                         init_res = X509_STORE_CTX_init(&ctx, store, signer, NULL
407                 else
408                         init_res = X509_STORE_CTX_init(&ctx, store, signer,
409                                         req->optionalSignature->certs);
410                 if(!init_res)
411                         {
412                         OCSPerr(OCSP_F_OCSP_REQUEST_VERIFY,ERR_R_X509_LIB);
413                         return 0;
414                         }

416                 X509_STORE_CTX_set_purpose(&ctx, X509_PURPOSE_OCSP_HELPER);
417                 X509_STORE_CTX_set_trust(&ctx, X509_TRUST_OCSP_REQUEST);
418                 ret = X509_verify_cert(&ctx);
419                 X509_STORE_CTX_cleanup(&ctx);
420                 if (ret <= 0)
421                         {
422                         ret = X509_STORE_CTX_get_error(&ctx);
423                         OCSPerr(OCSP_F_OCSP_REQUEST_VERIFY,OCSP_R_CERTIFICATE_VE
424                         ERR_add_error_data(2, "Verify error:",
425                                         X509_verify_cert_error_string(ret));
426                         return 0;
427                         }
428                 }
429         return 1;
430         }

432 static int ocsp_req_find_signer(X509 **psigner, OCSP_REQUEST *req, X509_NAME *nm
433                                 X509_STORE *st, unsigned long flags)
434         {
435         X509 *signer;
436         if(!(flags & OCSP_NOINTERN))
437                 {
438                 signer = X509_find_by_subject(req->optionalSignature->certs, nm)
439                 *psigner = signer;
440                 return 1;
441                 }

443         signer = X509_find_by_subject(certs, nm);
444         if (signer)
445                 {
446                 *psigner = signer;
447                 return 2;
448                 }
449         return 0;
450         }
451 #endif /* ! codereview */
```

```
**********************************************************
   13739 Wed Aug 13 19:52:59 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_all.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_all.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bio.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pkcs7.h>
#include <openssl/pem.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
```

```
129 #ifndef OPENSSL_NO_RSA
130 static RSA *pkey_get_rsa(EVP_PKEY *key, RSA **rsa);
131 #endif
132 #ifndef OPENSSL_NO_DSA
133 static DSA *pkey_get_dsa(EVP_PKEY *key, DSA **dsa);
134 #endif

136 #ifndef OPENSSL_NO_EC
137 static EC_KEY *pkey_get_eckey(EVP_PKEY *key, EC_KEY **eckey);
138 #endif

140 IMPLEMENT_PEM_rw(X509_REQ, X509_REQ, PEM_STRING_X509_REQ, X509_REQ)

142 IMPLEMENT_PEM_write(X509_REQ_NEW, X509_REQ, PEM_STRING_X509_REQ_OLD, X509_REQ)

144 IMPLEMENT_PEM_rw(X509_CRL, X509_CRL, PEM_STRING_X509_CRL, X509_CRL)

146 IMPLEMENT_PEM_rw(PKCS7, PKCS7, PEM_STRING_PKCS7, PKCS7)

148 IMPLEMENT_PEM_rw(NETSCAPE_CERT_SEQUENCE, NETSCAPE_CERT_SEQUENCE,
149                                         PEM_STRING_X509, NETSCAPE_CERT_SEQUENCE)


152 #ifndef OPENSSL_NO_RSA

154 /* We treat RSA or DSA private keys as a special case.
155  *
156  * For private keys we read in an EVP_PKEY structure with
157  * PEM_read_bio_PrivateKey() and extract the relevant private
158  * key: this means can handle "traditional" and PKCS#8 formats
159  * transparently.
160  */

162 static RSA *pkey_get_rsa(EVP_PKEY *key, RSA **rsa)
163 {
164         RSA *rtmp;
165         if(!key) return NULL;
166         rtmp = EVP_PKEY_get1_RSA(key);
167         EVP_PKEY_free(key);
168         if(!rtmp) return NULL;
169         if(rsa) {
170                 RSA_free(*rsa);
171                 *rsa = rtmp;
172         }
173         return rtmp;
174 }

176 RSA *PEM_read_bio_RSAPrivateKey(BIO *bp, RSA **rsa, pem_password_cb *cb,
177                                                                 void *u)
178 {
179         EVP_PKEY *pktmp;
180         pktmp = PEM_read_bio_PrivateKey(bp, NULL, cb, u);
181         return pkey_get_rsa(pktmp, rsa);
182 }

184 #ifndef OPENSSL_NO_FP_API

186 RSA *PEM_read_RSAPrivateKey(FILE *fp, RSA **rsa, pem_password_cb *cb,
187                                                                 void *u)
188 {
189         EVP_PKEY *pktmp;
190         pktmp = PEM_read_PrivateKey(fp, NULL, cb, u);
191         return pkey_get_rsa(pktmp, rsa);
192 }
```



```
194 #endif

196 #ifdef OPENSSL_FIPS

198 int PEM_write_bio_RSAPrivateKey(BIO *bp, RSA *x, const EVP_CIPHER *enc,
199                                                 unsigned char *kstr, int klen,
200                                                 pem_password_cb *cb, void *u)
201 {
202         if (FIPS_mode())
203                 {
204                 EVP_PKEY *k;
205                 int ret;
206                 k = EVP_PKEY_new();
207                 if (!k)
208                         return 0;
209                 EVP_PKEY_set1_RSA(k, x);

211                 ret = PEM_write_bio_PrivateKey(bp, k, enc, kstr, klen, cb, u);
212                 EVP_PKEY_free(k);
213                 return ret;
214                 }
215         else
216                 return PEM_ASN1_write_bio((i2d_of_void *)i2d_RSAPrivateKey,
217                                         PEM_STRING_RSA,bp,x,enc,kstr,klen,cb,u);
218 }

220 #ifndef OPENSSL_NO_FP_API
221 int PEM_write_RSAPrivateKey(FILE *fp, RSA *x, const EVP_CIPHER *enc,
222                                                 unsigned char *kstr, int klen,
223                                                 pem_password_cb *cb, void *u)
224 {
225         if (FIPS_mode())
226                 {
227                 EVP_PKEY *k;
228                 int ret;
229                 k = EVP_PKEY_new();
230                 if (!k)
231                         return 0;

233                 EVP_PKEY_set1_RSA(k, x);

235                 ret = PEM_write_PrivateKey(fp, k, enc, kstr, klen, cb, u);
236                 EVP_PKEY_free(k);
237                 return ret;
238                 }
239         else
240                 return PEM_ASN1_write((i2d_of_void *)i2d_RSAPrivateKey,
241                                         PEM_STRING_RSA,fp,x,enc,kstr,klen,cb,u);
242 }
243 #endif

245 #else

247 IMPLEMENT_PEM_write_cb_const(RSAPrivateKey, RSA, PEM_STRING_RSA, RSAPrivateKey)

249 #endif

251 IMPLEMENT_PEM_rw_const(RSAPublicKey, RSA, PEM_STRING_RSA_PUBLIC, RSAPublicKey)
252 IMPLEMENT_PEM_rw(RSA_PUBKEY, RSA, PEM_STRING_PUBLIC, RSA_PUBKEY)

254 #endif

256 #ifndef OPENSSL_NO_DSA

258 static DSA *pkey_get_dsa(EVP_PKEY *key, DSA **dsa)
259 {
```

```
260         DSA *dtmp;
261         if(!key) return NULL;
262         dtmp = EVP_PKEY_get1_DSA(key);
263         EVP_PKEY_free(key);
264         if(!dtmp) return NULL;
265         if(dsa) {
266                 DSA_free(*dsa);
267                 *dsa = dtmp;
268         }
269         return dtmp;
270 }

272 DSA *PEM_read_bio_DSAPrivateKey(BIO *bp, DSA **dsa, pem_password_cb *cb,
273                                                                 void *u)
274 {
275         EVP_PKEY *pktmp;
276         pktmp = PEM_read_bio_PrivateKey(bp, NULL, cb, u);
277         return pkey_get_dsa(pktmp, dsa);        /* will free pktmp */
278 }

280 #ifdef OPENSSL_FIPS

282 int PEM_write_bio_DSAPrivateKey(BIO *bp, DSA *x, const EVP_CIPHER *enc,
283                                                 unsigned char *kstr, int klen,
284                                                 pem_password_cb *cb, void *u)
285 {
286         if (FIPS_mode())
287                 {
288                 EVP_PKEY *k;
289                 int ret;
290                 k = EVP_PKEY_new();
291                 if (!k)
292                                 return 0;
293                 EVP_PKEY_set1_DSA(k, x);

295                 ret = PEM_write_bio_PrivateKey(bp, k, enc, kstr, klen, cb, u);
296                 EVP_PKEY_free(k);
297                 return ret;
298                 }
299         else
300                 return PEM_ASN1_write_bio((i2d_of_void *)i2d_DSAPrivateKey,
301                                         PEM_STRING_DSA,bp,x,enc,kstr,klen,cb,u);
302 }

304 #ifndef OPENSSL_NO_FP_API
305 int PEM_write_DSAPrivateKey(FILE *fp, DSA *x, const EVP_CIPHER *enc,
306                                                 unsigned char *kstr, int klen,
307                                                 pem_password_cb *cb, void *u)
308 {
309         if (FIPS_mode())
310                 {
311                 EVP_PKEY *k;
312                 int ret;
313                 k = EVP_PKEY_new();
314                 if (!k)
315                                 return 0;
316                 EVP_PKEY_set1_DSA(k, x);
317                 ret = PEM_write_PrivateKey(fp, k, enc, kstr, klen, cb, u);
318                 EVP_PKEY_free(k);
319                 return ret;
320                 }
321         else
322                 return PEM_ASN1_write((i2d_of_void *)i2d_DSAPrivateKey,
323                                         PEM_STRING_DSA,fp,x,enc,kstr,klen,cb,u);
324 }
325 #endif
```



```
327 #else

329 IMPLEMENT_PEM_write_cb_const(DSAPrivateKey, DSA, PEM_STRING_DSA, DSAPrivateKey)

331 #endif

333 IMPLEMENT_PEM_rw(DSA_PUBKEY, DSA, PEM_STRING_PUBLIC, DSA_PUBKEY)

335 #ifndef OPENSSL_NO_FP_API

337 DSA *PEM_read_DSAPrivateKey(FILE *fp, DSA **dsa, pem_password_cb *cb,
338                                                                 void *u)
339 {
340         EVP_PKEY *pktmp;
341         pktmp = PEM_read_PrivateKey(fp, NULL, cb, u);
342         return pkey_get_dsa(pktmp, dsa);        /* will free pktmp */
343 }

345 #endif

347 IMPLEMENT_PEM_rw_const(DSAparams, DSA, PEM_STRING_DSAPARAMS, DSAparams)

349 #endif


352 #ifndef OPENSSL_NO_EC
353 static EC_KEY *pkey_get_eckey(EVP_PKEY *key, EC_KEY **eckey)
354 {
355         EC_KEY *dtmp;
356         if(!key) return NULL;
357         dtmp = EVP_PKEY_get1_EC_KEY(key);
358         EVP_PKEY_free(key);
359         if(!dtmp) return NULL;
360         if(eckey)
361         {
362                 EC_KEY_free(*eckey);
363                 *eckey = dtmp;
364         }
365         return dtmp;
366 }

368 EC_KEY *PEM_read_bio_ECPrivateKey(BIO *bp, EC_KEY **key, pem_password_cb *cb,
369                                                                 void *u)
370 {
371         EVP_PKEY *pktmp;
372         pktmp = PEM_read_bio_PrivateKey(bp, NULL, cb, u);
373         return pkey_get_eckey(pktmp, key);      /* will free pktmp */
374 }

376 IMPLEMENT_PEM_rw_const(ECPKParameters, EC_GROUP, PEM_STRING_ECPARAMETERS, ECPKPa



380 #ifdef OPENSSL_FIPS

382 int PEM_write_bio_ECPrivateKey(BIO *bp, EC_KEY *x, const EVP_CIPHER *enc,
383                                                 unsigned char *kstr, int klen,
384                                                 pem_password_cb *cb, void *u)
385 {
386         if (FIPS_mode())
387                 {
388                 EVP_PKEY *k;
389                 int ret;
390                 k = EVP_PKEY_new();
391                 if (!k)
```

```
392                         return 0;
393                 EVP_PKEY_set1_EC_KEY(k, x);

395                 ret = PEM_write_bio_PrivateKey(bp, k, enc, kstr, klen, cb, u);
396                 EVP_PKEY_free(k);
397                 return ret;
398                 }
399         else
400                 return PEM_ASN1_write_bio((i2d_of_void *)i2d_ECPrivateKey,
401                                           PEM_STRING_ECPRIVATEKEY,
402                                           bp,x,enc,kstr,klen,cb,u);
403 }

405 #ifndef OPENSSL_NO_FP_API
406 int PEM_write_ECPrivateKey(FILE *fp, EC_KEY *x, const EVP_CIPHER *enc,
407                                          unsigned char *kstr, int klen,
408                                          pem_password_cb *cb, void *u)
409 {
410         if (FIPS_mode())
411                 {
412                 EVP_PKEY *k;
413                 int ret;
414                 k = EVP_PKEY_new();
415                 if (!k)
416                         return 0;
417                 EVP_PKEY_set1_EC_KEY(k, x);
418                 ret = PEM_write_PrivateKey(fp, k, enc, kstr, klen, cb, u);
419                 EVP_PKEY_free(k);
420                 return ret;
421                 }
422         else
423                 return PEM_ASN1_write((i2d_of_void *)i2d_ECPrivateKey,
424                                       PEM_STRING_ECPRIVATEKEY,
425                                       fp,x,enc,kstr,klen,cb,u);
426 }
427 #endif

429 #else

431 IMPLEMENT_PEM_write_cb(ECPrivateKey, EC_KEY, PEM_STRING_ECPRIVATEKEY, ECPrivateK

433 #endif

435 IMPLEMENT_PEM_rw(EC_PUBKEY, EC_KEY, PEM_STRING_PUBLIC, EC_PUBKEY)

437 #ifndef OPENSSL_NO_FP_API

439 EC_KEY *PEM_read_ECPrivateKey(FILE *fp, EC_KEY **eckey, pem_password_cb *cb,
440                                                         void *u)
441 {
442         EVP_PKEY *pktmp;
443         pktmp = PEM_read_PrivateKey(fp, NULL, cb, u);
444         return pkey_get_eckey(pktmp, eckey);    /* will free pktmp */
445 }

447 #endif

449 #endif

451 #ifndef OPENSSL_NO_DH

453 IMPLEMENT_PEM_rw_const(DHparams, DH, PEM_STRING_DHPARAMS, DHparams)

455 #endif

457 IMPLEMENT_PEM_rw(PUBKEY, EVP_PKEY, PEM_STRING_PUBLIC, PUBKEY)
```



```
458 #endif /* ! codereview */
```

```
**********************************************************
    7386 Wed Aug 13 19:52:59 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_err.c */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/pem.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_PEM,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_PEM,0,reason)

static ERR_STRING_DATA PEM_str_functs[]=
        {
{ERR_FUNC(PEM_F_B2I_DSS),       "B2I_DSS"},
{ERR_FUNC(PEM_F_B2I_PVK_BIO),   "b2i_PVK_bio"},
{ERR_FUNC(PEM_F_B2I_RSA),       "B2I_RSA"},
{ERR_FUNC(PEM_F_CHECK_BITLEN_DSA),      "CHECK_BITLEN_DSA"},
{ERR_FUNC(PEM_F_CHECK_BITLEN_RSA),      "CHECK_BITLEN_RSA"},
{ERR_FUNC(PEM_F_D2I_PKCS8PRIVATEKEY_BIO),       "d2i_PKCS8PrivateKey_bio"},
{ERR_FUNC(PEM_F_D2I_PKCS8PRIVATEKEY_FP),        "d2i_PKCS8PrivateKey_fp"},
{ERR_FUNC(PEM_F_DO_B2I),        "DO_B2I"},
{ERR_FUNC(PEM_F_DO_B2I_BIO),    "DO_B2I_BIO"},
{ERR_FUNC(PEM_F_DO_BLOB_HEADER),        "DO_BLOB_HEADER"},
{ERR_FUNC(PEM_F_DO_PK8PKEY),    "DO_PK8PKEY"},
{ERR_FUNC(PEM_F_DO_PK8PKEY_FP), "DO_PK8PKEY_FP"},
{ERR_FUNC(PEM_F_DO_PVK_BODY),   "DO_PVK_BODY"},
{ERR_FUNC(PEM_F_DO_PVK_HEADER), "DO_PVK_HEADER"},
{ERR_FUNC(PEM_F_I2B_PVK),       "I2B_PVK"},
{ERR_FUNC(PEM_F_I2B_PVK_BIO),   "i2b_PVK_bio"},
{ERR_FUNC(PEM_F_LOAD_IV),       "LOAD_IV"},
{ERR_FUNC(PEM_F_PEM_ASN1_READ), "PEM_ASN1_read"},
{ERR_FUNC(PEM_F_PEM_ASN1_READ_BIO),     "PEM_ASN1_read_bio"},
{ERR_FUNC(PEM_F_PEM_ASN1_WRITE),        "PEM_ASN1_write"},
{ERR_FUNC(PEM_F_PEM_ASN1_WRITE_BIO),    "PEM_ASN1_write_bio"},
{ERR_FUNC(PEM_F_PEM_DEF_CALLBACK),      "PEM_def_callback"},
{ERR_FUNC(PEM_F_PEM_DO_HEADER), "PEM_do_header"},
{ERR_FUNC(PEM_F_PEM_F_PEM_WRITE_PKCS8PRIVATEKEY),       "PEM_F_PEM_WRITE_PKCS8PR
{ERR_FUNC(PEM_F_PEM_GET_EVP_CIPHER_INFO),       "PEM_get_EVP_CIPHER_INFO"},
{ERR_FUNC(PEM_F_PEM_PK8PKEY),   "PEM_PK8PKEY"},
{ERR_FUNC(PEM_F_PEM_READ),      "PEM_read"},
{ERR_FUNC(PEM_F_PEM_READ_BIO),  "PEM_read_bio"},
{ERR_FUNC(PEM_F_PEM_READ_BIO_PARAMETERS),       "PEM_read_bio_Parameters"},
{ERR_FUNC(PEM_F_PEM_READ_BIO_PRIVATEKEY),       "PEM_READ_BIO_PRIVATEKEY"},
{ERR_FUNC(PEM_F_PEM_READ_PRIVATEKEY),   "PEM_READ_PRIVATEKEY"},
{ERR_FUNC(PEM_F_PEM_SEALFINAL), "PEM_SealFinal"},
{ERR_FUNC(PEM_F_PEM_SEALINIT),  "PEM_SealInit"},
{ERR_FUNC(PEM_F_PEM_SIGNFINAL), "PEM_SignFinal"},
{ERR_FUNC(PEM_F_PEM_WRITE),     "PEM_write"},
{ERR_FUNC(PEM_F_PEM_WRITE_BIO), "PEM_write_bio"},
{ERR_FUNC(PEM_F_PEM_WRITE_PRIVATEKEY),  "PEM_WRITE_PRIVATEKEY"},
{ERR_FUNC(PEM_F_PEM_X509_INFO_READ),    "PEM_X509_INFO_read"},
{ERR_FUNC(PEM_F_PEM_X509_INFO_READ_BIO),        "PEM_X509_INFO_read_bio"},
{ERR_FUNC(PEM_F_PEM_X509_INFO_WRITE_BIO),       "PEM_X509_INFO_write_bio"},
{0,NULL}
        };

static ERR_STRING_DATA PEM_str_reasons[]=
        {
{ERR_REASON(PEM_R_BAD_BASE64_DECODE)     ,"bad base64 decode"},
{ERR_REASON(PEM_R_BAD_DECRYPT)           ,"bad decrypt"},
{ERR_REASON(PEM_R_BAD_END_LINE)          ,"bad end line"},
{ERR_REASON(PEM_R_BAD_IV_CHARS)          ,"bad iv chars"},
{ERR_REASON(PEM_R_BAD_MAGIC_NUMBER)      ,"bad magic number"},
{ERR_REASON(PEM_R_BAD_PASSWORD_READ)     ,"bad password read"},
{ERR_REASON(PEM_R_BAD_VERSION_NUMBER)    ,"bad version number"},
{ERR_REASON(PEM_R_BIO_WRITE_FAILURE)     ,"bio write failure"},
{ERR_REASON(PEM_R_CIPHER_IS_NULL)        ,"cipher is null"},
{ERR_REASON(PEM_R_ERROR_CONVERTING_PRIVATE_KEY),"error converting private key"},
```

```
128 {ERR_REASON(PEM_R_EXPECTING_PRIVATE_KEY_BLOB),"expecting private key blob"},
129 {ERR_REASON(PEM_R_EXPECTING_PUBLIC_KEY_BLOB),"expecting public key blob"},
130 {ERR_REASON(PEM_R_INCONSISTENT_HEADER)   ,"inconsistent header"},
131 {ERR_REASON(PEM_R_KEYBLOB_HEADER_PARSE_ERROR),"keyblob header parse error"},
132 {ERR_REASON(PEM_R_KEYBLOB_TOO_SHORT)     ,"keyblob too short"},
133 {ERR_REASON(PEM_R_NOT_DEK_INFO)          ,"not dek info"},
134 {ERR_REASON(PEM_R_NOT_ENCRYPTED)         ,"not encrypted"},
135 {ERR_REASON(PEM_R_NOT_PROC_TYPE)         ,"not proc type"},
136 {ERR_REASON(PEM_R_NO_START_LINE)         ,"no start line"},
137 {ERR_REASON(PEM_R_PROBLEMS_GETTING_PASSWORD),"problems getting password"},
138 {ERR_REASON(PEM_R_PUBLIC_KEY_NO_RSA)     ,"public key no rsa"},
139 {ERR_REASON(PEM_R_PVK_DATA_TOO_SHORT)    ,"pvk data too short"},
140 {ERR_REASON(PEM_R_PVK_TOO_SHORT)         ,"pvk too short"},
141 {ERR_REASON(PEM_R_READ_KEY)              ,"read key"},
142 {ERR_REASON(PEM_R_SHORT_HEADER)          ,"short header"},
143 {ERR_REASON(PEM_R_UNSUPPORTED_CIPHER)    ,"unsupported cipher"},
144 {ERR_REASON(PEM_R_UNSUPPORTED_ENCRYPTION),"unsupported encryption"},
145 {ERR_REASON(PEM_R_UNSUPPORTED_KEY_COMPONENTS),"unsupported key components"},
146 {0,NULL}
147         };

149 #endif

151 void ERR_load_PEM_strings(void)
152         {
153 #ifndef OPENSSL_NO_ERR

155         if (ERR_func_error_string(PEM_str_functs[0].error) == NULL)
156                 {
157                 ERR_load_strings(0,PEM_str_functs);
158                 ERR_load_strings(0,PEM_str_reasons);
159                 }
160 #endif
161         }
162 #endif /* ! codereview */
```

```
**********************************************************
   11044 Wed Aug 13 19:52:59 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_info.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_info.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif

#ifndef OPENSSL_NO_FP_API
STACK_OF(X509_INFO) *PEM_X509_INFO_read(FILE *fp, STACK_OF(X509_INFO) *sk, pem_p
        {
        BIO *b;
        STACK_OF(X509_INFO) *ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                PEMerr(PEM_F_PEM_X509_INFO_READ,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=PEM_X509_INFO_read_bio(b,sk,cb,u);
        BIO_free(b);
        return(ret);
        }
#endif

STACK_OF(X509_INFO) *PEM_X509_INFO_read_bio(BIO *bp, STACK_OF(X509_INFO) *sk, pe
        {
        X509_INFO *xi=NULL;
        char *name=NULL,*header=NULL;
        void *pp;
        unsigned char *data=NULL;
        const unsigned char *p;
        long len,error=0;
        int ok=0;
        STACK_OF(X509_INFO) *ret=NULL;
        unsigned int i,raw,ptype;
        d2i_of_void *d2i = 0;

        if (sk == NULL)
                {
                if ((ret=sk_X509_INFO_new_null()) == NULL)
                        {
                        PEMerr(PEM_F_PEM_X509_INFO_READ_BIO,ERR_R_MALLOC_FAILURE
                        goto err;
                        }
                }
        else
                ret=sk;

        if ((xi=X509_INFO_new()) == NULL) goto err;
        for (;;)
                {
                raw=0;
                ptype = 0;
                i=PEM_read_bio(bp,&name,&header,&data,&len);
                if (i == 0)
                        {
                        error=ERR_GET_REASON(ERR_peek_last_error());
                        if (error == PEM_R_NO_START_LINE)
                                {
                                ERR_clear_error();
                                break;
```

```
128                             }
129                         goto err;
130                         }
131 start:
132                 if (    (strcmp(name,PEM_STRING_X509) == 0) ||
133                         (strcmp(name,PEM_STRING_X509_OLD) == 0))
134                         {
135                         d2i=(D2I_OF(void))d2i_X509;
136                         if (xi->x509 != NULL)
137                                 {
138                                 if (!sk_X509_INFO_push(ret,xi)) goto err;
139                                 if ((xi=X509_INFO_new()) == NULL) goto err;
140                                 goto start;
141                                 }
142                         pp=&(xi->x509);
143                         }
144                 else if ((strcmp(name,PEM_STRING_X509_TRUSTED) == 0))
145                         {
146                         d2i=(D2I_OF(void))d2i_X509_AUX;
147                         if (xi->x509 != NULL)
148                                 {
149                                 if (!sk_X509_INFO_push(ret,xi)) goto err;
150                                 if ((xi=X509_INFO_new()) == NULL) goto err;
151                                 goto start;
152                                 }
153                         pp=&(xi->x509);
154                         }
155                 else if (strcmp(name,PEM_STRING_X509_CRL) == 0)
156                         {
157                         d2i=(D2I_OF(void))d2i_X509_CRL;
158                         if (xi->crl != NULL)
159                                 {
160                                 if (!sk_X509_INFO_push(ret,xi)) goto err;
161                                 if ((xi=X509_INFO_new()) == NULL) goto err;
162                                 goto start;
163                                 }
164                         pp=&(xi->crl);
165                         }
166                 else
167 #ifndef OPENSSL_NO_RSA
168                         if (strcmp(name,PEM_STRING_RSA) == 0)
169                         {
170                         d2i=(D2I_OF(void))d2i_RSAPrivateKey;
171                         if (xi->x_pkey != NULL)
172                                 {
173                                 if (!sk_X509_INFO_push(ret,xi)) goto err;
174                                 if ((xi=X509_INFO_new()) == NULL) goto err;
175                                 goto start;
176                                 }

178                         xi->enc_data=NULL;
179                         xi->enc_len=0;

181                         xi->x_pkey=X509_PKEY_new();
182                         ptype=EVP_PKEY_RSA;
183                         pp=&xi->x_pkey->dec_pkey;
184                         if ((int)strlen(header) > 10) /* assume encrypted */
185                                 raw=1;
186                         }
187                 else
188 #endif
189 #ifndef OPENSSL_NO_DSA
190                         if (strcmp(name,PEM_STRING_DSA) == 0)
191                         {
192                         d2i=(D2I_OF(void))d2i_DSAPrivateKey;
193                         if (xi->x_pkey != NULL)
```



```
194                                 {
195                                 if (!sk_X509_INFO_push(ret,xi)) goto err;
196                                 if ((xi=X509_INFO_new()) == NULL) goto err;
197                                 goto start;
198                                 }

200                         xi->enc_data=NULL;
201                         xi->enc_len=0;

203                         xi->x_pkey=X509_PKEY_new();
204                         ptype = EVP_PKEY_DSA;
205                         pp=&xi->x_pkey->dec_pkey;
206                         if ((int)strlen(header) > 10) /* assume encrypted */
207                                 raw=1;
208                         }
209                 else
210 #endif
211 #ifndef OPENSSL_NO_EC
212                         if (strcmp(name,PEM_STRING_ECPRIVATEKEY) == 0)
213                         {
214                                 d2i=(D2I_OF(void))d2i_ECPrivateKey;
215                                 if (xi->x_pkey != NULL)
216                                 {
217                                         if (!sk_X509_INFO_push(ret,xi)) goto err
218                                         if ((xi=X509_INFO_new()) == NULL) goto e
219                                                 goto start;
220                                 }

222                         xi->enc_data=NULL;
223                         xi->enc_len=0;

225                         xi->x_pkey=X509_PKEY_new();
226                         ptype = EVP_PKEY_EC;
227                         pp=&xi->x_pkey->dec_pkey;
228                         if ((int)strlen(header) > 10) /* assume encrypted */
229                                 raw=1;
230                         }
231                 else
232 #endif
233                         {
234                         d2i=NULL;
235                         pp=NULL;
236                         }

238                 if (d2i != NULL)
239                         {
240                         if (!raw)
241                                 {
242                                 EVP_CIPHER_INFO cipher;

244                                 if (!PEM_get_EVP_CIPHER_INFO(header,&cipher))
245                                         goto err;
246                                 if (!PEM_do_header(&cipher,data,&len,cb,u))
247                                         goto err;
248                                 p=data;
249                                 if (ptype)
250                                         {
251                                         if (!d2i_PrivateKey(ptype, pp, &p, len))
252                                                 {
253                                                 PEMerr(PEM_F_PEM_X509_INFO_READ_
254                                                 goto err;
255                                                 }
256                                         }
257                                 else if (d2i(pp,&p,len) == NULL)
258                                         {
259                                         PEMerr(PEM_F_PEM_X509_INFO_READ_BIO,ERR_
```

```
260                                     goto err;
261                                     }
262                             }
263                     else
264                             { /* encrypted RSA data */
265                             if (!PEM_get_EVP_CIPHER_INFO(header,
266                                     &xi->enc_cipher)) goto err;
267                             xi->enc_data=(char *)data;
268                             xi->enc_len=(int)len;
269                             data=NULL;
270                             }
271                     }
272             else    {
273                     /* unknown */
274                     }
275             if (name != NULL) OPENSSL_free(name);
276             if (header != NULL) OPENSSL_free(header);
277             if (data != NULL) OPENSSL_free(data);
278             name=NULL;
279             header=NULL;
280             data=NULL;
281             }

283     /* if the last one hasn't been pushed yet and there is anything
284      * in it then add it to the stack ...
285      */
286     if ((xi->x509 != NULL) || (xi->crl != NULL) ||
287             (xi->x_pkey != NULL) || (xi->enc_data != NULL))
288             {
289             if (!sk_X509_INFO_push(ret,xi)) goto err;
290             xi=NULL;
291             }
292     ok=1;
293 err:
294     if (xi != NULL) X509_INFO_free(xi);
295     if (!ok)
296             {
297             for (i=0; ((int)i)<sk_X509_INFO_num(ret); i++)
298                     {
299                     xi=sk_X509_INFO_value(ret,i);
300                     X509_INFO_free(xi);
301                     }
302             if (ret != sk) sk_X509_INFO_free(ret);
303             ret=NULL;
304             }

306     if (name != NULL) OPENSSL_free(name);
307     if (header != NULL) OPENSSL_free(header);
308     if (data != NULL) OPENSSL_free(data);
309     return(ret);
310     }


313 /* A TJH addition */
314 int PEM_X509_INFO_write_bio(BIO *bp, X509_INFO *xi, EVP_CIPHER *enc,
315              unsigned char *kstr, int klen, pem_password_cb *cb, void *u)
316     {
317     EVP_CIPHER_CTX ctx;
318     int i,ret=0;
319     unsigned char *data=NULL;
320     const char *objstr=NULL;
321     char buf[PEM_BUFSIZE];
322     unsigned char *iv=NULL;

324     if (enc != NULL)
325             {
```



```
326             objstr=OBJ_nid2sn(EVP_CIPHER_nid(enc));
327             if (objstr == NULL)
328                     {
329                     PEMerr(PEM_F_PEM_X509_INFO_WRITE_BIO,PEM_R_UNSUPPORTED_C
330                     goto err;
331                     }
332             }

334     /* now for the fun part ... if we have a private key then
335      * we have to be able to handle a not-yet-decrypted key
336      * being written out correctly ... if it is decrypted or
337      * it is non-encrypted then we use the base code
338      */
339     if (xi->x_pkey!=NULL)
340             {
341             if ( (xi->enc_data!=NULL) && (xi->enc_len>0) )
342                     {
343                     if (enc == NULL)
344                             {
345                             PEMerr(PEM_F_PEM_X509_INFO_WRITE_BIO,PEM_R_CIPHE
346                             goto err;
347                             }

349                     /* copy from weirdo names into more normal things */
350                     iv=xi->enc_cipher.iv;
351                     data=(unsigned char *)xi->enc_data;
352                     i=xi->enc_len;

354                     /* we take the encryption data from the
355                      * internal stuff rather than what the
356                      * user has passed us ... as we have to
357                      * match exactly for some strange reason
358                      */
359                     objstr=OBJ_nid2sn(
360                             EVP_CIPHER_nid(xi->enc_cipher.cipher));
361                     if (objstr == NULL)
362                             {
363                             PEMerr(PEM_F_PEM_X509_INFO_WRITE_BIO,PEM_R_UNSUP
364                             goto err;
365                             }

367                     /* create the right magic header stuff */
368                     OPENSSL_assert(strlen(objstr)+23+2*enc->iv_len+13 <= siz
369                     buf[0]='\0';
370                     PEM_proc_type(buf,PEM_TYPE_ENCRYPTED);
371                     PEM_dek_info(buf,objstr,enc->iv_len,(char *)iv);

373                     /* use the normal code to write things out */
374                     i=PEM_write_bio(bp,PEM_STRING_RSA,buf,data,i);
375                     if (i <= 0) goto err;
376                     }
377             else
378                     {
379                     /* Add DSA/DH */
380 #ifndef OPENSSL_NO_RSA
381                     /* normal optionally encrypted stuff */
382                     if (PEM_write_bio_RSAPrivateKey(bp,
383                             xi->x_pkey->dec_pkey->pkey.rsa,
384                             enc,kstr,klen,cb,u)<=0)
385                             goto err;
386 #endif
387                     }
388             }

390     /* if we have a certificate then write it out now */
391     if ((xi->x509 != NULL) && (PEM_write_bio_X509(bp,xi->x509) <= 0))
```

```
392                 goto err;

394         /* we are ignoring anything else that is loaded into the X509_INFO
395          * structure for the moment ... as I don't need it so I'm not
396          * coding it here and Eric can do it when this makes it into the
397          * base library --tjh
398          */

400         ret=1;

402 err:
403         OPENSSL_cleanse((char *)&ctx,sizeof(ctx));
404         OPENSSL_cleanse(buf,PEM_BUFSIZE);
405         return(ret);
406         }
407 #endif /* ! codereview */
```

```
**********************************************************
   21075 Wed Aug 13 19:52:59 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/pkcs12.h>
#include "asn1_locl.h"
#ifndef OPENSSL_NO_DES
#include <openssl/des.h>
#endif
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

const char PEM_version[]="PEM" OPENSSL_VERSION_PTEXT;

#define MIN_LENGTH      4

static int load_iv(char **fromp,unsigned char *to, int num);
static int check_pem(const char *nm, const char *name);
int pem_check_suffix(const char *pem_str, const char *suffix);

int PEM_def_callback(char *buf, int num, int w, void *key)
        {
#ifdef OPENSSL_NO_FP_API
        /* We should not ever call the default callback routine from
         * windows. */
        PEMerr(PEM_F_PEM_DEF_CALLBACK,ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
        return(-1);
#else
        int i,j;
        const char *prompt;
        if(key) {
                i=strlen(key);
                i=(i > num)?num:i;
                memcpy(buf,key,i);
                return(i);
        }

        prompt=EVP_get_pw_prompt();
        if (prompt == NULL)
                prompt="Enter PEM pass phrase:";

        for (;;)
                {
                i=EVP_read_pw_string_min(buf,MIN_LENGTH,num,prompt,w);
                if (i != 0)
                        {
                        PEMerr(PEM_F_PEM_DEF_CALLBACK,PEM_R_PROBLEMS_GETTING_PAS
                        memset(buf,0,(unsigned int)num);
                        return(-1);
                        }
                j=strlen(buf);
                if (j < MIN_LENGTH)
                        {
                        fprintf(stderr,"phrase is too short, needs to be at leas
                        }
                else
                        break;
                }
        return(j);
#endif
        }

void PEM_proc_type(char *buf, int type)
```

```
128         {
129         const char *str;

131         if (type == PEM_TYPE_ENCRYPTED)
132                 str="ENCRYPTED";
133         else if (type == PEM_TYPE_MIC_CLEAR)
134                 str="MIC-CLEAR";
135         else if (type == PEM_TYPE_MIC_ONLY)
136                 str="MIC-ONLY";
137         else
138                 str="BAD-TYPE";

140         BUF_strlcat(buf,"Proc-Type: 4,",PEM_BUFSIZE);
141         BUF_strlcat(buf,str,PEM_BUFSIZE);
142         BUF_strlcat(buf,"\n",PEM_BUFSIZE);
143         }

145 void PEM_dek_info(char *buf, const char *type, int len, char *str)
146         {
147         static const unsigned char map[17]="0123456789ABCDEF";
148         long i;
149         int j;

151         BUF_strlcat(buf,"DEK-Info: ",PEM_BUFSIZE);
152         BUF_strlcat(buf,type,PEM_BUFSIZE);
153         BUF_strlcat(buf,",",PEM_BUFSIZE);
154         j=strlen(buf);
155         if (j + (len * 2) + 1 > PEM_BUFSIZE)
156                 return;
157         for (i=0; i<len; i++)
158                 {
159                 buf[j+i*2]  =map[(str[i]>>4)&0x0f];
160                 buf[j+i*2+1]=map[(str[i]   )&0x0f];
161                 }
162         buf[j+i*2]='\n';
163         buf[j+i*2+1]='\0';
164         }

166 #ifndef OPENSSL_NO_FP_API
167 void *PEM_ASN1_read(d2i_of_void *d2i, const char *name, FILE *fp, void **x,
168                     pem_password_cb *cb, void *u)
169         {
170         BIO *b;
171         void *ret;

173         if ((b=BIO_new(BIO_s_file())) == NULL)
174                 {
175                 PEMerr(PEM_F_PEM_ASN1_READ,ERR_R_BUF_LIB);
176                 return(0);
177                 }
178         BIO_set_fp(b,fp,BIO_NOCLOSE);
179         ret=PEM_ASN1_read_bio(d2i,name,b,x,cb,u);
180         BIO_free(b);
181         return(ret);
182         }
183 #endif

185 static int check_pem(const char *nm, const char *name)
186 {
187         /* Normal matching nm and name */
188         if (!strcmp(nm,name)) return 1;

190         /* Make PEM_STRING_EVP_PKEY match any private key */

192         if(!strcmp(name,PEM_STRING_EVP_PKEY))
193                 {
```



```
194                 int slen;
195                 const EVP_PKEY_ASN1_METHOD *ameth;
196                 if(!strcmp(nm,PEM_STRING_PKCS8))
197                         return 1;
198                 if(!strcmp(nm,PEM_STRING_PKCS8INF))
199                         return 1;
200                 slen = pem_check_suffix(nm, "PRIVATE KEY");
201                 if (slen > 0)
202                         {
203                         /* NB: ENGINE implementations wont contain
204                          * a deprecated old private key decode function
205                          * so don't look for them.
206                          */
207                         ameth = EVP_PKEY_asn1_find_str(NULL, nm, slen);
208                         if (ameth && ameth->old_priv_decode)
209                                 return 1;
210                         }
211                 return 0;
212                 }

214         if(!strcmp(name,PEM_STRING_PARAMETERS))
215                 {
216                 int slen;
217                 const EVP_PKEY_ASN1_METHOD *ameth;
218                 slen = pem_check_suffix(nm, "PARAMETERS");
219                 if (slen > 0)
220                         {
221                         ENGINE *e;
222                         ameth = EVP_PKEY_asn1_find_str(&e, nm, slen);
223                         if (ameth)
224                                 {
225                                 int r;
226                                 if (ameth->param_decode)
227                                         r = 1;
228                                 else
229                                         r = 0;
230 #ifndef OPENSSL_NO_ENGINE
231                                 if (e)
232                                         ENGINE_finish(e);
233 #endif
234                                 return r;
235                                 }
236                         }
237                 return 0;
238                 }

240         /* Permit older strings */

242         if(!strcmp(nm,PEM_STRING_X509_OLD) &&
243                 !strcmp(name,PEM_STRING_X509)) return 1;

245         if(!strcmp(nm,PEM_STRING_X509_REQ_OLD) &&
246                 !strcmp(name,PEM_STRING_X509_REQ)) return 1;

248         /* Allow normal certs to be read as trusted certs */
249         if(!strcmp(nm,PEM_STRING_X509) &&
250                 !strcmp(name,PEM_STRING_X509_TRUSTED)) return 1;

252         if(!strcmp(nm,PEM_STRING_X509_OLD) &&
253                 !strcmp(name,PEM_STRING_X509_TRUSTED)) return 1;

255         /* Some CAs use PKCS#7 with CERTIFICATE headers */
256         if(!strcmp(nm, PEM_STRING_X509) &&
257                 !strcmp(name, PEM_STRING_PKCS7)) return 1;

259         if(!strcmp(nm, PEM_STRING_PKCS7_SIGNED) &&
```

```
260                 !strcmp(name, PEM_STRING_PKCS7)) return 1;

262 #ifndef OPENSSL_NO_CMS
263         if(!strcmp(nm, PEM_STRING_X509) &&
264                 !strcmp(name, PEM_STRING_CMS)) return 1;
265         /* Allow CMS to be read from PKCS#7 headers */
266         if(!strcmp(nm, PEM_STRING_PKCS7) &&
267                 !strcmp(name, PEM_STRING_CMS)) return 1;
268 #endif

270         return 0;
271 }

273 int PEM_bytes_read_bio(unsigned char **pdata, long *plen, char **pnm, const char
274             pem_password_cb *cb, void *u)
275         {
276         EVP_CIPHER_INFO cipher;
277         char *nm=NULL,*header=NULL;
278         unsigned char *data=NULL;
279         long len;
280         int ret = 0;

282         for (;;)
283                 {
284                 if (!PEM_read_bio(bp,&nm,&header,&data,&len)) {
285                         if(ERR_GET_REASON(ERR_peek_error()) ==
286                                 PEM_R_NO_START_LINE)
287                                 ERR_add_error_data(2, "Expecting: ", name);
288                         return 0;
289                 }
290                 if(check_pem(nm, name)) break;
291                 OPENSSL_free(nm);
292                 OPENSSL_free(header);
293                 OPENSSL_free(data);
294                 }
295         if (!PEM_get_EVP_CIPHER_INFO(header,&cipher)) goto err;
296         if (!PEM_do_header(&cipher,data,&len,cb,u)) goto err;

298         *pdata = data;
299         *plen = len;

301         if (pnm)
302                 *pnm = nm;

304         ret = 1;

306 err:
307         if (!ret || !pnm) OPENSSL_free(nm);
308         OPENSSL_free(header);
309         if (!ret) OPENSSL_free(data);
310         return ret;
311         }

313 #ifndef OPENSSL_NO_FP_API
314 int PEM_ASN1_write(i2d_of_void *i2d, const char *name, FILE *fp,
315                    void *x, const EVP_CIPHER *enc, unsigned char *kstr,
316                    int klen, pem_password_cb *callback, void *u)
317         {
318         BIO *b;
319         int ret;

321         if ((b=BIO_new(BIO_s_file())) == NULL)
322                 {
323                 PEMerr(PEM_F_PEM_ASN1_WRITE,ERR_R_BUF_LIB);
324                 return(0);
325                 }
```



```
326         BIO_set_fp(b,fp,BIO_NOCLOSE);
327         ret=PEM_ASN1_write_bio(i2d,name,b,x,enc,kstr,klen,callback,u);
328         BIO_free(b);
329         return(ret);
330         }
331 #endif

333 int PEM_ASN1_write_bio(i2d_of_void *i2d, const char *name, BIO *bp,
334                        void *x, const EVP_CIPHER *enc, unsigned char *kstr,
335                        int klen, pem_password_cb *callback, void *u)
336         {
337         EVP_CIPHER_CTX ctx;
338         int dsize=0,i,j,ret=0;
339         unsigned char *p,*data=NULL;
340         const char *objstr=NULL;
341         char buf[PEM_BUFSIZE];
342         unsigned char key[EVP_MAX_KEY_LENGTH];
343         unsigned char iv[EVP_MAX_IV_LENGTH];

345         if (enc != NULL)
346                 {
347                 objstr=OBJ_nid2sn(EVP_CIPHER_nid(enc));
348                 if (objstr == NULL)
349                         {
350                         PEMerr(PEM_F_PEM_ASN1_WRITE_BIO,PEM_R_UNSUPPORTED_CIPHER
351                         goto err;
352                         }
353                 }

355         if ((dsize=i2d(x,NULL)) < 0)
356                 {
357                 PEMerr(PEM_F_PEM_ASN1_WRITE_BIO,ERR_R_ASN1_LIB);
358                 dsize=0;
359                 goto err;
360                 }
361         /* dzise + 8 bytes are needed */
362         /* actually it needs the cipher block size extra... */
363         data=(unsigned char *)OPENSSL_malloc((unsigned int)dsize+20);
364         if (data == NULL)
365                 {
366                 PEMerr(PEM_F_PEM_ASN1_WRITE_BIO,ERR_R_MALLOC_FAILURE);
367                 goto err;
368                 }
369         p=data;
370         i=i2d(x,&p);

372         if (enc != NULL)
373                 {
374                 if (kstr == NULL)
375                         {
376                         if (callback == NULL)
377                                 klen=PEM_def_callback(buf,PEM_BUFSIZE,1,u);
378                         else
379                                 klen=(*callback)(buf,PEM_BUFSIZE,1,u);
380                         if (klen <= 0)
381                                 {
382                                 PEMerr(PEM_F_PEM_ASN1_WRITE_BIO,PEM_R_READ_KEY);
383                                 goto err;
384                                 }
385 #ifdef CHARSET_EBCDIC
386                         /* Convert the pass phrase from EBCDIC */
387                         ebcdic2ascii(buf, buf, klen);
388 #endif
389                         kstr=(unsigned char *)buf;
390                         }
391                 RAND_add(data,i,0);/* put in the RSA key. */
```

```
392                 OPENSSL_assert(enc->iv_len <= (int)sizeof(iv));
393                 if (RAND_pseudo_bytes(iv,enc->iv_len) < 0) /* Generate a salt */
394                         goto err;
395                 /* The 'iv' is used as the iv and as a salt.  It is
396                  * NOT taken from the BytesToKey function */
397                 if (!EVP_BytesToKey(enc,EVP_md5(),iv,kstr,klen,1,key,NULL))
398                         goto err;

400                 if (kstr == (unsigned char *)buf) OPENSSL_cleanse(buf,PEM_BUFSIZ

402                 OPENSSL_assert(strlen(objstr)+23+2*enc->iv_len+13 <= sizeof buf)

404                 buf[0]='\0';
405                 PEM_proc_type(buf,PEM_TYPE_ENCRYPTED);
406                 PEM_dek_info(buf,objstr,enc->iv_len,(char *)iv);
407                 /* k=strlen(buf); */

409                 EVP_CIPHER_CTX_init(&ctx);
410                 ret = 1;
411                 if (!EVP_EncryptInit_ex(&ctx,enc,NULL,key,iv)
412                         || !EVP_EncryptUpdate(&ctx,data,&j,data,i)
413                         || !EVP_EncryptFinal_ex(&ctx,&(data[j]),&i))
414                         ret = 0;
415                 EVP_CIPHER_CTX_cleanup(&ctx);
416                 if (ret == 0)
417                         goto err;
418                 i+=j;
419                 }
420         else
421                 {
422                 ret=1;
423                 buf[0]='\0';
424                 }
425         i=PEM_write_bio(bp,name,buf,data,i);
426         if (i <= 0) ret=0;
427 err:
428         OPENSSL_cleanse(key,sizeof(key));
429         OPENSSL_cleanse(iv,sizeof(iv));
430         OPENSSL_cleanse((char *)&ctx,sizeof(ctx));
431         OPENSSL_cleanse(buf,PEM_BUFSIZE);
432         if (data != NULL)
433                 {
434                 OPENSSL_cleanse(data,(unsigned int)dsize);
435                 OPENSSL_free(data);
436                 }
437         return(ret);
438         }

440 int PEM_do_header(EVP_CIPHER_INFO *cipher, unsigned char *data, long *plen,
441             pem_password_cb *callback,void *u)
442         {
443         int i,j,o,klen;
444         long len;
445         EVP_CIPHER_CTX ctx;
446         unsigned char key[EVP_MAX_KEY_LENGTH];
447         char buf[PEM_BUFSIZE];

449         len= *plen;

451         if (cipher->cipher == NULL) return(1);
452         if (callback == NULL)
453                 klen=PEM_def_callback(buf,PEM_BUFSIZE,0,u);
454         else
455                 klen=callback(buf,PEM_BUFSIZE,0,u);
456         if (klen <= 0)
457                 {
```



```
458                 PEMerr(PEM_F_PEM_DO_HEADER,PEM_R_BAD_PASSWORD_READ);
459                 return(0);
460                 }
461 #ifdef CHARSET_EBCDIC
462         /* Convert the pass phrase from EBCDIC */
463         ebcdic2ascii(buf, buf, klen);
464 #endif

466         if (!EVP_BytesToKey(cipher->cipher,EVP_md5(),&(cipher->iv[0]),
467                 (unsigned char *)buf,klen,1,key,NULL))
468                 return 0;

470         j=(int)len;
471         EVP_CIPHER_CTX_init(&ctx);
472         o = EVP_DecryptInit_ex(&ctx,cipher->cipher,NULL, key,&(cipher->iv[0]));
473         if (o)
474                 o = EVP_DecryptUpdate(&ctx,data,&i,data,j);
475         if (o)
476                 o = EVP_DecryptFinal_ex(&ctx,&(data[i]),&j);
477         EVP_CIPHER_CTX_cleanup(&ctx);
478         OPENSSL_cleanse((char *)buf,sizeof(buf));
479         OPENSSL_cleanse((char *)key,sizeof(key));
480         j+=i;
481         if (!o)
482                 {
483                 PEMerr(PEM_F_PEM_DO_HEADER,PEM_R_BAD_DECRYPT);
484                 return(0);
485                 }
486         *plen=j;
487         return(1);
488         }

490 int PEM_get_EVP_CIPHER_INFO(char *header, EVP_CIPHER_INFO *cipher)
491         {
492         const EVP_CIPHER *enc=NULL;
493         char *p,c;
494         char **header_pp = &header;

496         cipher->cipher=NULL;
497         if ((header == NULL) || (*header == '\0') || (*header == '\n'))
498                 return(1);
499         if (strncmp(header,"Proc-Type: ",11) != 0)
500                 { PEMerr(PEM_F_PEM_GET_EVP_CIPHER_INFO,PEM_R_NOT_PROC_TYPE); ret
501         header+=11;
502         if (*header != '4') return(0); header++;
503         if (*header != ',') return(0); header++;
504         if (strncmp(header,"ENCRYPTED",9) != 0)
505                 { PEMerr(PEM_F_PEM_GET_EVP_CIPHER_INFO,PEM_R_NOT_ENCRYPTED); ret
506         for (; (*header != '\n') && (*header != '\0'); header++)
507                 ;
508         if (*header == '\0')
509                 { PEMerr(PEM_F_PEM_GET_EVP_CIPHER_INFO,PEM_R_SHORT_HEADER); retu
510         header++;
511         if (strncmp(header,"DEK-Info: ",10) != 0)
512                 { PEMerr(PEM_F_PEM_GET_EVP_CIPHER_INFO,PEM_R_NOT_DEK_INFO); retu
513         header+=10;

515         p=header;
516         for (;;)
517                 {
518                 c= *header;
519 #ifndef CHARSET_EBCDIC
520                 if (!(  ((c >= 'A') && (c <= 'Z')) || (c == '-') ||
521                         ((c >= '0') && (c <= '9'))))
522                         break;
523 #else
```

```
524                 if (!(  isupper(c) || (c == '-') ||
525                         isdigit(c)))
526                         break;
527 #endif
528                 header++;
529                 }
530         *header='\0';
531         cipher->cipher=enc=EVP_get_cipherbyname(p);
532         *header=c;
533         header++;

535         if (enc == NULL)
536                 {
537                 PEMerr(PEM_F_PEM_GET_EVP_CIPHER_INFO,PEM_R_UNSUPPORTED_ENCRYPTIO
538                 return(0);
539                 }
540         if (!load_iv(header_pp,&(cipher->iv[0]),enc->iv_len))
541                 return(0);

543         return(1);
544         }

546 static int load_iv(char **fromp, unsigned char *to, int num)
547         {
548         int v,i;
549         char *from;

551         from= *fromp;
552         for (i=0; i<num; i++) to[i]=0;
553         num*=2;
554         for (i=0; i<num; i++)
555                 {
556                 if ((*from >= '0') && (*from <= '9'))
557                         v= *from-'0';
558                 else if ((*from >= 'A') && (*from <= 'F'))
559                         v= *from-'A'+10;
560                 else if ((*from >= 'a') && (*from <= 'f'))
561                         v= *from-'a'+10;
562                 else
563                         {
564                         PEMerr(PEM_F_LOAD_IV,PEM_R_BAD_IV_CHARS);
565                         return(0);
566                         }
567                 from++;
568                 to[i/2]|=v<<(long)((!(i&1))*4);
569                 }

571         *fromp=from;
572         return(1);
573         }

575 #ifndef OPENSSL_NO_FP_API
576 int PEM_write(FILE *fp, char *name, char *header, unsigned char *data,
577               long len)
578         {
579         BIO *b;
580         int ret;

582         if ((b=BIO_new(BIO_s_file())) == NULL)
583                 {
584                 PEMerr(PEM_F_PEM_WRITE,ERR_R_BUF_LIB);
585                 return(0);
586                 }
587         BIO_set_fp(b,fp,BIO_NOCLOSE);
588         ret=PEM_write_bio(b, name, header, data,len);
589         BIO_free(b);
```



```
590         return(ret);
591         }
592 #endif

594 int PEM_write_bio(BIO *bp, const char *name, char *header, unsigned char *data,
595                   long len)
596         {
597         int nlen,n,i,j,outl;
598         unsigned char *buf = NULL;
599         EVP_ENCODE_CTX ctx;
600         int reason=ERR_R_BUF_LIB;

602         EVP_EncodeInit(&ctx);
603         nlen=strlen(name);

605         if (    (BIO_write(bp,"-----BEGIN ",11) != 11) ||
606                 (BIO_write(bp,name,nlen) != nlen) ||
607                 (BIO_write(bp,"-----\n",6) != 6))
608                 goto err;

610         i=strlen(header);
611         if (i > 0)
612                 {
613                 if (    (BIO_write(bp,header,i) != i) ||
614                         (BIO_write(bp,"\n",1) != 1))
615                         goto err;
616                 }

618         buf = OPENSSL_malloc(PEM_BUFSIZE*8);
619         if (buf == NULL)
620                 {
621                 reason=ERR_R_MALLOC_FAILURE;
622                 goto err;
623                 }

625         i=j=0;
626         while (len > 0)
627                 {
628                 n=(int)((len>(PEM_BUFSIZE*5))?(PEM_BUFSIZE*5):len);
629                 EVP_EncodeUpdate(&ctx,buf,&outl,&(data[j]),n);
630                 if ((outl) && (BIO_write(bp,(char *)buf,outl) != outl))
631                         goto err;
632                 i+=outl;
633                 len-=n;
634                 j+=n;
635                 }
636         EVP_EncodeFinal(&ctx,buf,&outl);
637         if ((outl > 0) && (BIO_write(bp,(char *)buf,outl) != outl)) goto err;
638         OPENSSL_cleanse(buf, PEM_BUFSIZE*8);
639         OPENSSL_free(buf);
640         buf = NULL;
641         if (    (BIO_write(bp,"-----END ",9) != 9) ||
642                 (BIO_write(bp,name,nlen) != nlen) ||
643                 (BIO_write(bp,"-----\n",6) != 6))
644                 goto err;
645         return(i+outl);
646 err:
647         if (buf) {
648                 OPENSSL_cleanse(buf, PEM_BUFSIZE*8);
649                 OPENSSL_free(buf);
650         }
651         PEMerr(PEM_F_PEM_WRITE_BIO,reason);
652         return(0);
653         }

655 #ifndef OPENSSL_NO_FP_API
```

```
656 int PEM_read(FILE *fp, char **name, char **header, unsigned char **data,
657             long *len)
658         {
659         BIO *b;
660         int ret;

662         if ((b=BIO_new(BIO_s_file())) == NULL)
663                 {
664                 PEMerr(PEM_F_PEM_READ,ERR_R_BUF_LIB);
665                 return(0);
666                 }
667         BIO_set_fp(b,fp,BIO_NOCLOSE);
668         ret=PEM_read_bio(b, name, header, data,len);
669         BIO_free(b);
670         return(ret);
671         }
672 #endif

674 int PEM_read_bio(BIO *bp, char **name, char **header, unsigned char **data,
675             long *len)
676         {
677         EVP_ENCODE_CTX ctx;
678         int end=0,i,k,bl=0,hl=0,nohead=0;
679         char buf[256];
680         BUF_MEM *nameB;
681         BUF_MEM *headerB;
682         BUF_MEM *dataB,*tmpB;

684         nameB=BUF_MEM_new();
685         headerB=BUF_MEM_new();
686         dataB=BUF_MEM_new();
687         if ((nameB == NULL) || (headerB == NULL) || (dataB == NULL))
688                 {
689                 BUF_MEM_free(nameB);
690                 BUF_MEM_free(headerB);
691                 BUF_MEM_free(dataB);
692                 PEMerr(PEM_F_PEM_READ_BIO,ERR_R_MALLOC_FAILURE);
693                 return(0);
694                 }

696         buf[254]='\0';
697         for (;;)
698                 {
699                 i=BIO_gets(bp,buf,254);

701                 if (i <= 0)
702                         {
703                         PEMerr(PEM_F_PEM_READ_BIO,PEM_R_NO_START_LINE);
704                         goto err;
705                         }

707                 while ((i >= 0) && (buf[i] <= ' ')) i--;
708                 buf[++i]='\n'; buf[++i]='\0';

710                 if (strncmp(buf,"-----BEGIN ",11) == 0)
711                         {
712                         i=strlen(&(buf[11]));

714                         if (strncmp(&(buf[11+i-6]),"-----\n",6) != 0)
715                                 continue;
716                         if (!BUF_MEM_grow(nameB,i+9))
717                                 {
718                                 PEMerr(PEM_F_PEM_READ_BIO,ERR_R_MALLOC_FAILURE);
719                                 goto err;
720                                 }
721                         memcpy(nameB->data,&(buf[11]),i-6);
```



```
722                         nameB->data[i-6]='\0';
723                         break;
724                         }
725                 }
726         hl=0;
727         if (!BUF_MEM_grow(headerB,256))
728                 { PEMerr(PEM_F_PEM_READ_BIO,ERR_R_MALLOC_FAILURE); goto err; }
729         headerB->data[0]='\0';
730         for (;;)
731                 {
732                 i=BIO_gets(bp,buf,254);
733                 if (i <= 0) break;

735                 while ((i >= 0) && (buf[i] <= ' ')) i--;
736                 buf[++i]='\n'; buf[++i]='\0';

738                 if (buf[0] == '\n') break;
739                 if (!BUF_MEM_grow(headerB,hl+i+9))
740                         { PEMerr(PEM_F_PEM_READ_BIO,ERR_R_MALLOC_FAILURE); goto
741                 if (strncmp(buf,"-----END ",9) == 0)
742                         {
743                         nohead=1;
744                         break;
745                         }
746                 memcpy(&(headerB->data[hl]),buf,i);
747                 headerB->data[hl+i]='\0';
748                 hl+=i;
749                 }

751         bl=0;
752         if (!BUF_MEM_grow(dataB,1024))
753                 { PEMerr(PEM_F_PEM_READ_BIO,ERR_R_MALLOC_FAILURE); goto err; }
754         dataB->data[0]='\0';
755         if (!nohead)
756                 {
757                 for (;;)
758                         {
759                         i=BIO_gets(bp,buf,254);
760                         if (i <= 0) break;

762                         while ((i >= 0) && (buf[i] <= ' ')) i--;
763                         buf[++i]='\n'; buf[++i]='\0';

765                         if (i != 65) end=1;
766                         if (strncmp(buf,"-----END ",9) == 0)
767                                 break;
768                         if (i > 65) break;
769                         if (!BUF_MEM_grow_clean(dataB,i+bl+9))
770                                 {
771                                 PEMerr(PEM_F_PEM_READ_BIO,ERR_R_MALLOC_FAILURE);
772                                 goto err;
773                                 }
774                         memcpy(&(dataB->data[bl]),buf,i);
775                         dataB->data[bl+i]='\0';
776                         bl+=i;
777                         if (end)
778                                 {
779                                 buf[0]='\0';
780                                 i=BIO_gets(bp,buf,254);
781                                 if (i <= 0) break;

783                                 while ((i >= 0) && (buf[i] <= ' ')) i--;
784                                 buf[++i]='\n'; buf[++i]='\0';

786                                 break;
787                                 }
```

```
788                         }
789                 }
790         else
791                 {
792                 tmpB=headerB;
793                 headerB=dataB;
794                 dataB=tmpB;
795                 bl=hl;
796                 }
797         i=strlen(nameB->data);
798         if (    (strncmp(buf,"-----END ",9) != 0) ||
799                 (strncmp(nameB->data,&(buf[9]),i) != 0) ||
800                 (strncmp(&(buf[9+i]),"-----\n",6) != 0))
801                 {
802                 PEMerr(PEM_F_PEM_READ_BIO,PEM_R_BAD_END_LINE);
803                 goto err;
804                 }

806         EVP_DecodeInit(&ctx);
807         i=EVP_DecodeUpdate(&ctx,
808                 (unsigned char *)dataB->data,&bl,
809                 (unsigned char *)dataB->data,bl);
810         if (i < 0)
811                 {
812                 PEMerr(PEM_F_PEM_READ_BIO,PEM_R_BAD_BASE64_DECODE);
813                 goto err;
814                 }
815         i=EVP_DecodeFinal(&ctx,(unsigned char *)&(dataB->data[bl]),&k);
816         if (i < 0)
817                 {
818                 PEMerr(PEM_F_PEM_READ_BIO,PEM_R_BAD_BASE64_DECODE);
819                 goto err;
820                 }
821         bl+=k;

823         if (bl == 0) goto err;
824         *name=nameB->data;
825         *header=headerB->data;
826         *data=(unsigned char *)dataB->data;
827         *len=bl;
828         OPENSSL_free(nameB);
829         OPENSSL_free(headerB);
830         OPENSSL_free(dataB);
831         return(1);
832 err:
833         BUF_MEM_free(nameB);
834         BUF_MEM_free(headerB);
835         BUF_MEM_free(dataB);
836         return(0);
837         }

839 /* Check pem string and return prefix length.
840  * If for example the pem_str == "RSA PRIVATE KEY" and suffix = "PRIVATE KEY"
841  * the return value is 3 for the string "RSA".
842  */

844 int pem_check_suffix(const char *pem_str, const char *suffix)
845         {
846         int pem_len = strlen(pem_str);
847         int suffix_len = strlen(suffix);
848         const char *p;
849         if (suffix_len + 1 >= pem_len)
850                 return 0;
851         p = pem_str + pem_len - suffix_len;
852         if (strcmp(p, suffix))
853                 return 0;
```



```
854         p--;
855         if (*p != ' ')
856                 return 0;
857         return p - pem_str;
858         }
859 #endif /* ! codereview */
```

```
**********************************************************
    3852 Wed Aug 13 19:52:59 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_oth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_oth.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/pem.h>

/* Handle 'other' PEMs: not private keys */

void *PEM_ASN1_read_bio(d2i_of_void *d2i, const char *name, BIO *bp, void **x,
                        pem_password_cb *cb, void *u)
        {
        const unsigned char *p=NULL;
        unsigned char *data=NULL;
        long len;
        char *ret=NULL;

        if (!PEM_bytes_read_bio(&data, &len, NULL, name, bp, cb, u))
                return NULL;
        p = data;
        ret=d2i(x,&p,len);
        if (ret == NULL)
                PEMerr(PEM_F_PEM_ASN1_READ_BIO,ERR_R_ASN1_LIB);
        OPENSSL_free(data);
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    8333 Wed Aug 13 19:52:59 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_pk8.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_pkey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/pkcs12.h>
#include <openssl/pem.h>

static int do_pk8pkey(BIO *bp, EVP_PKEY *x, int isder,
                                int nid, const EVP_CIPHER *enc,
                                char *kstr, int klen,
                                pem_password_cb *cb, void *u);
static int do_pk8pkey_fp(FILE *bp, EVP_PKEY *x, int isder,
                                int nid, const EVP_CIPHER *enc,
                                char *kstr, int klen,
                                pem_password_cb *cb, void *u);

/* These functions write a private key in PKCS#8 format: it is a "drop in"
 * replacement for PEM_write_bio_PrivateKey() and friends. As usual if 'enc'
 * is NULL then it uses the unencrypted private key form. The 'nid' versions
 * uses PKCS#5 v1.5 PBE algorithms whereas the others use PKCS#5 v2.0.
 */

int PEM_write_bio_PKCS8PrivateKey_nid(BIO *bp, EVP_PKEY *x, int nid,
                                  char *kstr, int klen,
                                  pem_password_cb *cb, void *u)
{
        return do_pk8pkey(bp, x, 0, nid, NULL, kstr, klen, cb, u);
}

int PEM_write_bio_PKCS8PrivateKey(BIO *bp, EVP_PKEY *x, const EVP_CIPHER *enc,
                                  char *kstr, int klen,
                                  pem_password_cb *cb, void *u)
{
        return do_pk8pkey(bp, x, 0, -1, enc, kstr, klen, cb, u);
}

int i2d_PKCS8PrivateKey_bio(BIO *bp, EVP_PKEY *x, const EVP_CIPHER *enc,
                                  char *kstr, int klen,
                                  pem_password_cb *cb, void *u)
{
        return do_pk8pkey(bp, x, 1, -1, enc, kstr, klen, cb, u);
}

int i2d_PKCS8PrivateKey_nid_bio(BIO *bp, EVP_PKEY *x, int nid,
                                  char *kstr, int klen,
                                  pem_password_cb *cb, void *u)
{
        return do_pk8pkey(bp, x, 1, nid, NULL, kstr, klen, cb, u);
}

static int do_pk8pkey(BIO *bp, EVP_PKEY *x, int isder, int nid, const EVP_CIPHER
                                  char *kstr, int klen,
                                  pem_password_cb *cb, void *u)
{
        X509_SIG *p8;
        PKCS8_PRIV_KEY_INFO *p8inf;
        char buf[PEM_BUFSIZE];
        int ret;
        if(!(p8inf = EVP_PKEY2PKCS8(x))) {
                PEMerr(PEM_F_DO_PK8PKEY,
                                        PEM_R_ERROR_CONVERTING_PRIVATE_KEY);
                return 0;
        }
        if(enc || (nid != -1)) {
                if(!kstr) {
                        if(!cb) klen = PEM_def_callback(buf, PEM_BUFSIZE, 1, u);
```

```
128                         else klen = cb(buf, PEM_BUFSIZE, 1, u);
129                         if(klen <= 0) {
130                                 PEMerr(PEM_F_DO_PK8PKEY,PEM_R_READ_KEY);
131                                 PKCS8_PRIV_KEY_INFO_free(p8inf);
132                                 return 0;
133                         }

135                         kstr = buf;
136                 }
137                 p8 = PKCS8_encrypt(nid, enc, kstr, klen, NULL, 0, 0, p8inf);
138                 if(kstr == buf) OPENSSL_cleanse(buf, klen);
139                 PKCS8_PRIV_KEY_INFO_free(p8inf);
140                 if(isder) ret = i2d_PKCS8_bio(bp, p8);
141                 else ret = PEM_write_bio_PKCS8(bp, p8);
142                 X509_SIG_free(p8);
143                 return ret;
144         } else {
145                 if(isder) ret = i2d_PKCS8_PRIV_KEY_INFO_bio(bp, p8inf);
146                 else ret = PEM_write_bio_PKCS8_PRIV_KEY_INFO(bp, p8inf);
147                 PKCS8_PRIV_KEY_INFO_free(p8inf);
148                 return ret;
149         }
150 }

152 EVP_PKEY *d2i_PKCS8PrivateKey_bio(BIO *bp, EVP_PKEY **x, pem_password_cb *cb, vo
153 {
154         PKCS8_PRIV_KEY_INFO *p8inf = NULL;
155         X509_SIG *p8 = NULL;
156         int klen;
157         EVP_PKEY *ret;
158         char psbuf[PEM_BUFSIZE];
159         p8 = d2i_PKCS8_bio(bp, NULL);
160         if(!p8) return NULL;
161         if (cb) klen=cb(psbuf,PEM_BUFSIZE,0,u);
162         else klen=PEM_def_callback(psbuf,PEM_BUFSIZE,0,u);
163         if (klen <= 0) {
164                 PEMerr(PEM_F_D2I_PKCS8PRIVATEKEY_BIO, PEM_R_BAD_PASSWORD_READ);
165                 X509_SIG_free(p8);
166                 return NULL;
167         }
168         p8inf = PKCS8_decrypt(p8, psbuf, klen);
169         X509_SIG_free(p8);
170         if(!p8inf) return NULL;
171         ret = EVP_PKCS82PKEY(p8inf);
172         PKCS8_PRIV_KEY_INFO_free(p8inf);
173         if(!ret) return NULL;
174         if(x) {
175                 if(*x) EVP_PKEY_free(*x);
176                 *x = ret;
177         }
178         return ret;
179 }

181 #ifndef OPENSSL_NO_FP_API

183 int i2d_PKCS8PrivateKey_fp(FILE *fp, EVP_PKEY *x, const EVP_CIPHER *enc,
184                                 char *kstr, int klen,
185                                 pem_password_cb *cb, void *u)
186 {
187         return do_pk8pkey_fp(fp, x, 1, -1, enc, kstr, klen, cb, u);
188 }

190 int i2d_PKCS8PrivateKey_nid_fp(FILE *fp, EVP_PKEY *x, int nid,
191                                 char *kstr, int klen,
192                                 pem_password_cb *cb, void *u)
193 {
```



```
194         return do_pk8pkey_fp(fp, x, 1, nid, NULL, kstr, klen, cb, u);
195 }

197 int PEM_write_PKCS8PrivateKey_nid(FILE *fp, EVP_PKEY *x, int nid,
198                                   char *kstr, int klen,
199                                   pem_password_cb *cb, void *u)
200 {
201         return do_pk8pkey_fp(fp, x, 0, nid, NULL, kstr, klen, cb, u);
202 }

204 int PEM_write_PKCS8PrivateKey(FILE *fp, EVP_PKEY *x, const EVP_CIPHER *enc,
205                               char *kstr, int klen, pem_password_cb *cb, void *u
206 {
207         return do_pk8pkey_fp(fp, x, 0, -1, enc, kstr, klen, cb, u);
208 }

210 static int do_pk8pkey_fp(FILE *fp, EVP_PKEY *x, int isder, int nid, const EVP_CI
211                                   char *kstr, int klen,
212                                   pem_password_cb *cb, void *u)
213 {
214         BIO *bp;
215         int ret;
216         if(!(bp = BIO_new_fp(fp, BIO_NOCLOSE))) {
217                 PEMerr(PEM_F_DO_PK8PKEY_FP,ERR_R_BUF_LIB);
218                 return(0);
219         }
220         ret = do_pk8pkey(bp, x, isder, nid, enc, kstr, klen, cb, u);
221         BIO_free(bp);
222         return ret;
223 }

225 EVP_PKEY *d2i_PKCS8PrivateKey_fp(FILE *fp, EVP_PKEY **x, pem_password_cb *cb, vo
226 {
227         BIO *bp;
228         EVP_PKEY *ret;
229         if(!(bp = BIO_new_fp(fp, BIO_NOCLOSE))) {
230                 PEMerr(PEM_F_D2I_PKCS8PRIVATEKEY_FP,ERR_R_BUF_LIB);
231                 return NULL;
232         }
233         ret = d2i_PKCS8PrivateKey_bio(bp, x, cb, u);
234         BIO_free(bp);
235         return ret;
236 }

238 #endif

240 IMPLEMENT_PEM_rw(PKCS8, X509_SIG, PEM_STRING_PKCS8, X509_SIG)
241 IMPLEMENT_PEM_rw(PKCS8_PRIV_KEY_INFO, PKCS8_PRIV_KEY_INFO, PEM_STRING_PKCS8INF,
242                                                         PKCS8_PRIV_KEY_INFO)
243 #endif /* ! codereview */
```

```
**********************************************************
    7821 Wed Aug 13 19:53:00 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_pkey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_pkey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/buffer.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/pkcs12.h>
#include <openssl/pem.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include "asn1_locl.h"

int pem_check_suffix(const char *pem_str, const char *suffix);

EVP_PKEY *PEM_read_bio_PrivateKey(BIO *bp, EVP_PKEY **x, pem_password_cb *cb, vo
        {
        char *nm=NULL;
        const unsigned char *p=NULL;
        unsigned char *data=NULL;
        long len;
        int slen;
        EVP_PKEY *ret=NULL;

        if (!PEM_bytes_read_bio(&data, &len, &nm, PEM_STRING_EVP_PKEY, bp, cb, u
                return NULL;
        p = data;

        if (strcmp(nm,PEM_STRING_PKCS8INF) == 0) {
                PKCS8_PRIV_KEY_INFO *p8inf;
                p8inf=d2i_PKCS8_PRIV_KEY_INFO(NULL, &p, len);
                if(!p8inf) goto p8err;
                ret = EVP_PKCS82PKEY(p8inf);
                if(x) {
                        if(*x) EVP_PKEY_free((EVP_PKEY *)*x);
                        *x = ret;
                }
                PKCS8_PRIV_KEY_INFO_free(p8inf);
        } else if (strcmp(nm,PEM_STRING_PKCS8) == 0) {
                PKCS8_PRIV_KEY_INFO *p8inf;
                X509_SIG *p8;
                int klen;
                char psbuf[PEM_BUFSIZE];
                p8 = d2i_X509_SIG(NULL, &p, len);
                if(!p8) goto p8err;
                if (cb) klen=cb(psbuf,PEM_BUFSIZE,0,u);
                else klen=PEM_def_callback(psbuf,PEM_BUFSIZE,0,u);
                if (klen <= 0) {
                        PEMerr(PEM_F_PEM_READ_BIO_PRIVATEKEY,
                                                PEM_R_BAD_PASSWORD_READ);
                        X509_SIG_free(p8);
                        goto err;
                }
                p8inf = PKCS8_decrypt(p8, psbuf, klen);
                X509_SIG_free(p8);
                if(!p8inf) goto p8err;
                ret = EVP_PKCS82PKEY(p8inf);
                if(x) {
                        if(*x) EVP_PKEY_free((EVP_PKEY *)*x);
                        *x = ret;
                }
                PKCS8_PRIV_KEY_INFO_free(p8inf);
        } else if ((slen = pem_check_suffix(nm, "PRIVATE KEY")) > 0)
                {
                const EVP_PKEY_ASN1_METHOD *ameth;
                ameth = EVP_PKEY_asn1_find_str(NULL, nm, slen);
                if (!ameth || !ameth->old_priv_decode)
                        goto p8err;
```

```
128                 ret=d2i_PrivateKey(ameth->pkey_id,x,&p,len);
129                 }
130 p8err:
131         if (ret == NULL)
132                 PEMerr(PEM_F_PEM_READ_BIO_PRIVATEKEY,ERR_R_ASN1_LIB);
133 err:
134         OPENSSL_free(nm);
135         OPENSSL_cleanse(data, len);
136         OPENSSL_free(data);
137         return(ret);
138         }

140 int PEM_write_bio_PrivateKey(BIO *bp, EVP_PKEY *x, const EVP_CIPHER *enc,
141                                         unsigned char *kstr, int klen,
142                                         pem_password_cb *cb, void *u)
143         {
144         char pem_str[80];
145         if (!x->ameth || x->ameth->priv_encode)
146                 return PEM_write_bio_PKCS8PrivateKey(bp, x, enc,
147                                                 (char *)kstr, klen,
148                                                 cb, u);

150         BIO_snprintf(pem_str, 80, "%s PRIVATE KEY", x->ameth->pem_str);
151         return PEM_ASN1_write_bio((i2d_of_void *)i2d_PrivateKey,
152                                 pem_str,bp,x,enc,kstr,klen,cb,u);
153         }

155 EVP_PKEY *PEM_read_bio_Parameters(BIO *bp, EVP_PKEY **x)
156         {
157         char *nm=NULL;
158         const unsigned char *p=NULL;
159         unsigned char *data=NULL;
160         long len;
161         int slen;
162         EVP_PKEY *ret=NULL;

164         if (!PEM_bytes_read_bio(&data, &len, &nm, PEM_STRING_PARAMETERS,
165                                                                 bp, 0, NULL))
166                 return NULL;
167         p = data;

169         if ((slen = pem_check_suffix(nm, "PARAMETERS")) > 0)
170                 {
171                 ret = EVP_PKEY_new();
172                 if (!ret)
173                         goto err;
174                 if (!EVP_PKEY_set_type_str(ret, nm, slen)
175                         || !ret->ameth->param_decode
176                         || !ret->ameth->param_decode(ret, &p, len))
177                         {
178                         EVP_PKEY_free(ret);
179                         ret = NULL;
180                         goto err;
181                         }
182                 if(x)
183                         {
184                         if(*x) EVP_PKEY_free((EVP_PKEY *)*x);
185                         *x = ret;
186                         }
187                 }
188 err:
189         if (ret == NULL)
190                 PEMerr(PEM_F_PEM_READ_BIO_PARAMETERS,ERR_R_ASN1_LIB);
191         OPENSSL_free(nm);
192         OPENSSL_free(data);
193         return(ret);
```



```
194         }

196 int PEM_write_bio_Parameters(BIO *bp, EVP_PKEY *x)
197         {
198         char pem_str[80];
199         if (!x->ameth || !x->ameth->param_encode)
200                 return 0;

202         BIO_snprintf(pem_str, 80, "%s PARAMETERS", x->ameth->pem_str);
203         return PEM_ASN1_write_bio(
204                 (i2d_of_void *)x->ameth->param_encode,
205                                 pem_str,bp,x,NULL,NULL,0,0,NULL);
206         }

208 #ifndef OPENSSL_NO_FP_API
209 EVP_PKEY *PEM_read_PrivateKey(FILE *fp, EVP_PKEY **x, pem_password_cb *cb, void
210         {
211         BIO *b;
212         EVP_PKEY *ret;

214         if ((b=BIO_new(BIO_s_file())) == NULL)
215                 {
216                 PEMerr(PEM_F_PEM_READ_PRIVATEKEY,ERR_R_BUF_LIB);
217                 return(0);
218                 }
219         BIO_set_fp(b,fp,BIO_NOCLOSE);
220         ret=PEM_read_bio_PrivateKey(b,x,cb,u);
221         BIO_free(b);
222         return(ret);
223         }

225 int PEM_write_PrivateKey(FILE *fp, EVP_PKEY *x, const EVP_CIPHER *enc,
226                                                 unsigned char *kstr, int klen,
227                                                 pem_password_cb *cb, void *u)
228         {
229         BIO *b;
230         int ret;

232         if ((b=BIO_new_fp(fp, BIO_NOCLOSE)) == NULL)
233                 {
234                 PEMerr(PEM_F_PEM_WRITE_PRIVATEKEY,ERR_R_BUF_LIB);
235                 return 0;
236                 }
237         ret=PEM_write_bio_PrivateKey(b, x, enc, kstr, klen, cb, u);
238         BIO_free(b);
239         return ret;
240         }

242 #endif
243 #endif /* ! codereview */
```

```
**********************************************************
    5960 Wed Aug 13 19:53:00 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_seal.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_seal.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/opensslconf.h>        /* for OPENSSL_NO_RSA */
#ifndef OPENSSL_NO_RSA
#include <stdio.h>
```



```
#include "cryptlib.h"
#include <openssl/evp.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/pem.h>
#include <openssl/rsa.h>

int PEM_SealInit(PEM_ENCODE_SEAL_CTX *ctx, EVP_CIPHER *type, EVP_MD *md_type,
             unsigned char **ek, int *ekl, unsigned char *iv, EVP_PKEY **pubk,
             int npubk)
        {
        unsigned char key[EVP_MAX_KEY_LENGTH];
        int ret= -1;
        int i,j,max=0;
        char *s=NULL;

        for (i=0; i<npubk; i++)
                {
                if (pubk[i]->type != EVP_PKEY_RSA)
                        {
                        PEMerr(PEM_F_PEM_SEALINIT,PEM_R_PUBLIC_KEY_NO_RSA);
                        goto err;
                        }
                j=RSA_size(pubk[i]->pkey.rsa);
                if (j > max) max=j;
                }
        s=(char *)OPENSSL_malloc(max*2);
        if (s == NULL)
                {
                PEMerr(PEM_F_PEM_SEALINIT,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        EVP_EncodeInit(&ctx->encode);

        EVP_MD_CTX_init(&ctx->md);
        if (!EVP_SignInit(&ctx->md,md_type))
                goto err;

        EVP_CIPHER_CTX_init(&ctx->cipher);
        ret=EVP_SealInit(&ctx->cipher,type,ek,ekl,iv,pubk,npubk);
        if (ret <= 0) goto err;

        /* base64 encode the keys */
        for (i=0; i<npubk; i++)
                {
                j=EVP_EncodeBlock((unsigned char *)s,ek[i],
                        RSA_size(pubk[i]->pkey.rsa));
                ekl[i]=j;
                memcpy(ek[i],s,j+1);
                }

        ret=npubk;
err:
        if (s != NULL) OPENSSL_free(s);
        OPENSSL_cleanse(key,EVP_MAX_KEY_LENGTH);
        return(ret);
        }

void PEM_SealUpdate(PEM_ENCODE_SEAL_CTX *ctx, unsigned char *out, int *outl,
             unsigned char *in, int inl)
        {
        unsigned char buffer[1600];
        int i,j;
```

```
128         *outl=0;
129         EVP_SignUpdate(&ctx->md,in,inl);
130         for (;;)
131                 {
132                 if (inl <= 0) break;
133                 if (inl > 1200)
134                         i=1200;
135                 else
136                         i=inl;
137                 EVP_EncryptUpdate(&ctx->cipher,buffer,&j,in,i);
138                 EVP_EncodeUpdate(&ctx->encode,out,&j,buffer,j);
139                 *outl+=j;
140                 out+=j;
141                 in+=i;
142                 inl-=i;
143                 }
144         }

146 int PEM_SealFinal(PEM_ENCODE_SEAL_CTX *ctx, unsigned char *sig, int *sigl,
147              unsigned char *out, int *outl, EVP_PKEY *priv)
148         {
149         unsigned char *s=NULL;
150         int ret=0,j;
151         unsigned int i;

153         if (priv->type != EVP_PKEY_RSA)
154                 {
155                 PEMerr(PEM_F_PEM_SEALFINAL,PEM_R_PUBLIC_KEY_NO_RSA);
156                 goto err;
157                 }
158         i=RSA_size(priv->pkey.rsa);
159         if (i < 100) i=100;
160         s=(unsigned char *)OPENSSL_malloc(i*2);
161         if (s == NULL)
162                 {
163                 PEMerr(PEM_F_PEM_SEALFINAL,ERR_R_MALLOC_FAILURE);
164                 goto err;
165                 }

167         if (!EVP_EncryptFinal_ex(&ctx->cipher,s,(int *)&i))
168                 goto err;
169         EVP_EncodeUpdate(&ctx->encode,out,&j,s,i);
170         *outl=j;
171         out+=j;
172         EVP_EncodeFinal(&ctx->encode,out,&j);
173         *outl+=j;

175         if (!EVP_SignFinal(&ctx->md,s,&i,priv)) goto err;
176         *sigl=EVP_EncodeBlock(sig,s,i);

178         ret=1;
179 err:
180         EVP_MD_CTX_cleanup(&ctx->md);
181         EVP_CIPHER_CTX_cleanup(&ctx->cipher);
182         if (s != NULL) OPENSSL_free(s);
183         return(ret);
184         }
185 #else /* !OPENSSL_NO_RSA */

187 # if PEDANTIC
188 static void *dummy=&dummy;
189 # endif

191 #endif
192 #endif /* ! codereview */
```

```
**********************************************************
    4104 Wed Aug 13 19:53:00 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_sign.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pem/pem_sign.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/rand.h>
```



```
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/pem.h>

void PEM_SignInit(EVP_MD_CTX *ctx, EVP_MD *type)
        {
        EVP_DigestInit_ex(ctx, type, NULL);
        }

void PEM_SignUpdate(EVP_MD_CTX *ctx, unsigned char *data,
             unsigned int count)
        {
        EVP_DigestUpdate(ctx,data,count);
        }

int PEM_SignFinal(EVP_MD_CTX *ctx, unsigned char *sigret, unsigned int *siglen,
             EVP_PKEY *pkey)
        {
        unsigned char *m;
        int i,ret=0;
        unsigned int m_len;

        m=(unsigned char *)OPENSSL_malloc(EVP_PKEY_size(pkey)+2);
        if (m == NULL)
                {
                PEMerr(PEM_F_PEM_SIGNFINAL,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        if (EVP_SignFinal(ctx,m,&m_len,pkey) <= 0) goto err;

        i=EVP_EncodeBlock(sigret,m,m_len);
        *siglen=i;
        ret=1;
err:
        /* ctx has been zeroed by EVP_SignFinal() */
        if (m != NULL) OPENSSL_free(m);
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    2956 Wed Aug 13 19:53:00 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_x509.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pem_x509.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bio.h>
```



```
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pkcs7.h>
#include <openssl/pem.h>

IMPLEMENT_PEM_rw(X509, X509, PEM_STRING_X509, X509)
#endif /* ! codereview */
```

```
**********************************************************
    3059 Wed Aug 13 19:53:00 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pem_xaux.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pem_xaux.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bio.h>
```



```
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pkcs7.h>
#include <openssl/pem.h>

IMPLEMENT_PEM_rw(X509_AUX, X509, PEM_STRING_X509_TRUSTED, X509_AUX)
IMPLEMENT_PEM_rw(X509_CERT_PAIR, X509_CERT_PAIR, PEM_STRING_X509_PAIR, X509_CERT
#endif /* ! codereview */
```

```
**********************************************************
   21712 Wed Aug 13 19:53:00 2014
new/usr/src/lib/openssl/libsunw_crypto/pem/pvkfmt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2005.
 */
/* ====================================================================
 * Copyright (c) 2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* Support for PVK format keys and related structures (such a PUBLICKEYBLOB
 * and PRIVATEKEYBLOB).
 */
```



```
#include "cryptlib.h"
#include <openssl/pem.h>
#include <openssl/rand.h>
#include <openssl/bn.h>
#if !defined(OPENSSL_NO_RSA) && !defined(OPENSSL_NO_DSA)
#include <openssl/dsa.h>
#include <openssl/rsa.h>

/* Utility function: read a DWORD (4 byte unsigned integer) in little endian
 * format
 */

static unsigned int read_ledword(const unsigned char **in)
        {
        const unsigned char *p = *in;
        unsigned int ret;
        ret = *p++;
        ret |= (*p++ << 8);
        ret |= (*p++ << 16);
        ret |= (*p++ << 24);
        *in = p;
        return ret;
        }

/* Read a BIGNUM in little endian format. The docs say that this should take up
 * bitlen/8 bytes.
 */

static int read_lebn(const unsigned char **in, unsigned int nbyte, BIGNUM **r)
        {
        const unsigned char *p;
        unsigned char *tmpbuf, *q;
        unsigned int i;
        p = *in + nbyte - 1;
        tmpbuf = OPENSSL_malloc(nbyte);
        if (!tmpbuf)
                return 0;
        q = tmpbuf;
        for (i = 0; i < nbyte; i++)
                *q++ = *p--;
        *r = BN_bin2bn(tmpbuf, nbyte, NULL);
        OPENSSL_free(tmpbuf);
        if (*r)
                {
                *in += nbyte;
                return 1;
                }
        else
                return 0;
        }


/* Convert private key blob to EVP_PKEY: RSA and DSA keys supported */

#define MS_PUBLICKEYBLOB        0x6
#define MS_PRIVATEKEYBLOB       0x7
#define MS_RSA1MAGIC            0x31415352L
#define MS_RSA2MAGIC            0x32415352L
#define MS_DSS1MAGIC            0x31535344L
#define MS_DSS2MAGIC            0x32535344L

#define MS_KEYALG_RSA_KEYX      0xa400
#define MS_KEYALG_DSS_SIGN      0x2200

#define MS_KEYTYPE_KEYX         0x1
#define MS_KEYTYPE_SIGN         0x2
```

```
/* The PVK file magic number: seems to spell out "bobsfile", who is Bob? */
#define MS_PVKMAGIC             0xb0b5f11eL
/* Salt length for PVK files */
#define PVK_SALTLEN             0x10

static EVP_PKEY *b2i_rsa(const unsigned char **in, unsigned int length,
                                                unsigned int bitlen, int ispub);
static EVP_PKEY *b2i_dss(const unsigned char **in, unsigned int length,
                                                unsigned int bitlen, int ispub);

static int do_blob_header(const unsigned char **in, unsigned int length,
                                unsigned int *pmagic, unsigned int *pbitlen,
                                int *pisdss, int *pispub)
        {
        const unsigned char *p = *in;
        if (length < 16)
                return 0;
        /* bType */
        if (*p == MS_PUBLICKEYBLOB)
                {
                if (*pispub == 0)
                        {
                        PEMerr(PEM_F_DO_BLOB_HEADER,
                                        PEM_R_EXPECTING_PRIVATE_KEY_BLOB);
                        return 0;
                        }
                *pispub = 1;
                }
        else if (*p == MS_PRIVATEKEYBLOB)
                {
                if (*pispub == 1)
                        {
                        PEMerr(PEM_F_DO_BLOB_HEADER,
                                        PEM_R_EXPECTING_PUBLIC_KEY_BLOB);
                        return 0;
                        }
                *pispub = 0;
                }
        else
                return 0;
        p++;
        /* Version */
        if (*p++ != 0x2)
                {
                PEMerr(PEM_F_DO_BLOB_HEADER, PEM_R_BAD_VERSION_NUMBER);
                return 0;
                }
        /* Ignore reserved, aiKeyAlg */
        p+= 6;
        *pmagic = read_ledword(&p);
        *pbitlen = read_ledword(&p);
        *pisdss = 0;
        switch (*pmagic)
                {

                case MS_DSS1MAGIC:
                *pisdss = 1;
                case MS_RSA1MAGIC:
                if (*pispub == 0)
                        {
                        PEMerr(PEM_F_DO_BLOB_HEADER,
                                        PEM_R_EXPECTING_PRIVATE_KEY_BLOB);
                        return 0;
                        }
                break;
```



```
                case MS_DSS2MAGIC:
                *pisdss = 1;
                case MS_RSA2MAGIC:
                if (*pispub == 1)
                        {
                        PEMerr(PEM_F_DO_BLOB_HEADER,
                                        PEM_R_EXPECTING_PUBLIC_KEY_BLOB);
                        return 0;
                        }
                break;

                default:
                PEMerr(PEM_F_DO_BLOB_HEADER, PEM_R_BAD_MAGIC_NUMBER);
                return -1;
                }
        *in = p;
        return 1;
        }

static unsigned int blob_length(unsigned bitlen, int isdss, int ispub)
        {
        unsigned int nbyte, hnbyte;
        nbyte = (bitlen + 7) >> 3;
        hnbyte = (bitlen + 15) >> 4;
        if (isdss)
                {

                /* Expected length: 20 for q + 3 components bitlen each + 24
                 * for seed structure.
                 */
                if (ispub)
                        return  44 + 3 * nbyte;
                /* Expected length: 20 for q, priv, 2 bitlen components + 24
                 * for seed structure.
                 */
                else
                        return 64 + 2 * nbyte;
                }
        else
                {
                /* Expected length: 4 for 'e' + 'n' */
                if (ispub)
                        return 4 + nbyte;
                else
                /* Expected length: 4 for 'e' and 7 other components.
                 * 2 components are bitlen size, 5 are bitlen/2
                 */
                        return 4 + 2*nbyte + 5*hnbyte;
                }

        }

static EVP_PKEY *do_b2i(const unsigned char **in, unsigned int length,
                                                                int ispub)
        {
        const unsigned char *p = *in;
        unsigned int bitlen, magic;
        int isdss;
        if (do_blob_header(&p, length, &magic, &bitlen, &isdss, &ispub) <= 0)
                {
                PEMerr(PEM_F_DO_B2I, PEM_R_KEYBLOB_HEADER_PARSE_ERROR);
                return NULL;
                }
        length -= 16;
        if (length < blob_length(bitlen, isdss, ispub))
```

```
260                 {
261                 PEMerr(PEM_F_DO_B2I, PEM_R_KEYBLOB_TOO_SHORT);
262                 return NULL;
263                 }
264         if (isdss)
265                 return b2i_dss(&p, length, bitlen, ispub);
266         else
267                 return b2i_rsa(&p, length, bitlen, ispub);
268         }

270 static EVP_PKEY *do_b2i_bio(BIO *in, int ispub)
271         {
272         const unsigned char *p;
273         unsigned char hdr_buf[16], *buf = NULL;
274         unsigned int bitlen, magic, length;
275         int isdss;
276         EVP_PKEY *ret = NULL;
277         if (BIO_read(in, hdr_buf, 16) != 16)
278                 {
279                 PEMerr(PEM_F_DO_B2I_BIO, PEM_R_KEYBLOB_TOO_SHORT);
280                 return NULL;
281                 }
282         p = hdr_buf;
283         if (do_blob_header(&p, 16, &magic, &bitlen, &isdss, &ispub) <= 0)
284                 return NULL;

286         length = blob_length(bitlen, isdss, ispub);
287         buf = OPENSSL_malloc(length);
288         if (!buf)
289                 {
290                 PEMerr(PEM_F_DO_B2I_BIO, ERR_R_MALLOC_FAILURE);
291                 goto err;
292                 }
293         p = buf;
294         if (BIO_read(in, buf, length) != (int)length)
295                 {
296                 PEMerr(PEM_F_DO_B2I_BIO, PEM_R_KEYBLOB_TOO_SHORT);
297                 goto err;
298                 }

300         if (isdss)
301                 ret = b2i_dss(&p, length, bitlen, ispub);
302         else
303                 ret = b2i_rsa(&p, length, bitlen, ispub);

305         err:
306         if (buf)
307                 OPENSSL_free(buf);
308         return ret;
309         }

311 static EVP_PKEY *b2i_dss(const unsigned char **in, unsigned int length,
312                                                 unsigned int bitlen, int ispub)
313         {
314         const unsigned char *p = *in;
315         EVP_PKEY *ret = NULL;
316         DSA *dsa = NULL;
317         BN_CTX *ctx = NULL;
318         unsigned int nbyte;
319         nbyte = (bitlen + 7) >> 3;

321         dsa = DSA_new();
322         ret = EVP_PKEY_new();
323         if (!dsa || !ret)
324                 goto memerr;
325         if (!read_lebn(&p, nbyte, &dsa->p))
```



```
326                 goto memerr;
327         if (!read_lebn(&p, 20, &dsa->q))
328                 goto memerr;
329         if (!read_lebn(&p, nbyte, &dsa->g))
330                 goto memerr;
331         if (ispub)
332                 {
333                 if (!read_lebn(&p, nbyte, &dsa->pub_key))
334                         goto memerr;
335                 }
336         else
337                 {
338                 if (!read_lebn(&p, 20, &dsa->priv_key))
339                         goto memerr;
340                 /* Calculate public key */
341                 if (!(dsa->pub_key = BN_new()))
342                         goto memerr;
343                 if (!(ctx = BN_CTX_new()))
344                         goto memerr;

346                 if (!BN_mod_exp(dsa->pub_key, dsa->g,
347                                                 dsa->priv_key, dsa->p, ctx))

349                         goto memerr;
350                 BN_CTX_free(ctx);
351                 }

353         EVP_PKEY_set1_DSA(ret, dsa);
354         DSA_free(dsa);
355         *in = p;
356         return ret;

358         memerr:
359         PEMerr(PEM_F_B2I_DSS, ERR_R_MALLOC_FAILURE);
360         if (dsa)
361                 DSA_free(dsa);
362         if (ret)
363                 EVP_PKEY_free(ret);
364         if (ctx)
365                 BN_CTX_free(ctx);
366         return NULL;
367         }

369 static EVP_PKEY *b2i_rsa(const unsigned char **in, unsigned int length,
370                                                 unsigned int bitlen, int ispub)

372         {
373         const unsigned char *p = *in;
374         EVP_PKEY *ret = NULL;
375         RSA *rsa = NULL;
376         unsigned int nbyte, hnbyte;
377         nbyte = (bitlen + 7) >> 3;
378         hnbyte = (bitlen + 15) >> 4;
379         rsa = RSA_new();
380         ret = EVP_PKEY_new();
381         if (!rsa || !ret)
382                 goto memerr;
383         rsa->e = BN_new();
384         if (!rsa->e)
385                 goto memerr;
386         if (!BN_set_word(rsa->e, read_ledword(&p)))
387                 goto memerr;
388         if (!read_lebn(&p, nbyte, &rsa->n))
389                 goto memerr;
390         if (!ispub)
391                 {
```

```
392                 if (!read_lebn(&p, hnbyte, &rsa->p))
393                         goto memerr;
394                 if (!read_lebn(&p, hnbyte, &rsa->q))
395                         goto memerr;
396                 if (!read_lebn(&p, hnbyte, &rsa->dmp1))
397                         goto memerr;
398                 if (!read_lebn(&p, hnbyte, &rsa->dmq1))
399                         goto memerr;
400                 if (!read_lebn(&p, hnbyte, &rsa->iqmp))
401                         goto memerr;
402                 if (!read_lebn(&p, nbyte, &rsa->d))
403                         goto memerr;
404                 }

406         EVP_PKEY_set1_RSA(ret, rsa);
407         RSA_free(rsa);
408         *in = p;
409         return ret;
410         memerr:
411         PEMerr(PEM_F_B2I_RSA, ERR_R_MALLOC_FAILURE);
412         if (rsa)
413                 RSA_free(rsa);
414         if (ret)
415                 EVP_PKEY_free(ret);
416         return NULL;
417         }

419 EVP_PKEY *b2i_PrivateKey(const unsigned char **in, long length)
420         {
421         return do_b2i(in, length, 0);
422         }

424 EVP_PKEY *b2i_PublicKey(const unsigned char **in, long length)
425         {
426         return do_b2i(in, length, 1);
427         }


430 EVP_PKEY *b2i_PrivateKey_bio(BIO *in)
431         {
432         return do_b2i_bio(in, 0);
433         }

435 EVP_PKEY *b2i_PublicKey_bio(BIO *in)
436         {
437         return do_b2i_bio(in, 1);
438         }

440 static void write_ledword(unsigned char **out, unsigned int dw)
441         {
442         unsigned char *p = *out;
443         *p++ = dw & 0xff;
444         *p++ = (dw>>8) & 0xff;
445         *p++ = (dw>>16) & 0xff;
446         *p++ = (dw>>24) & 0xff;
447         *out = p;
448         }

450 static void write_lebn(unsigned char **out, const BIGNUM *bn, int len)
451         {
452         int nb, i;
453         unsigned char *p = *out, *q, c;
454         nb = BN_num_bytes(bn);
455         BN_bn2bin(bn, p);
456         q = p + nb - 1;
457         /* In place byte order reversal */
```



```
458         for (i = 0; i < nb/2; i++)
459                 {
460                 c = *p;
461                 *p++ = *q;
462                 *q-- = c;
463                 }
464         *out += nb;
465         /* Pad with zeroes if we have to */
466         if (len > 0)
467                 {
468                 len -= nb;
469                 if (len > 0)
470                         {
471                         memset(*out, 0, len);
472                         *out += len;
473                         }
474                 }
475         }


478 static int check_bitlen_rsa(RSA *rsa, int ispub, unsigned int *magic);
479 static int check_bitlen_dsa(DSA *dsa, int ispub, unsigned int *magic);

481 static void write_rsa(unsigned char **out, RSA *rsa, int ispub);
482 static void write_dsa(unsigned char **out, DSA *dsa, int ispub);

484 static int do_i2b(unsigned char **out, EVP_PKEY *pk, int ispub)
485         {
486         unsigned char *p;
487         unsigned int bitlen, magic = 0, keyalg;
488         int outlen, noinc = 0;
489         if (pk->type == EVP_PKEY_DSA)
490                 {
491                 bitlen = check_bitlen_dsa(pk->pkey.dsa, ispub, &magic);
492                 keyalg = MS_KEYALG_DSS_SIGN;
493                 }
494         else if (pk->type == EVP_PKEY_RSA)
495                 {
496                 bitlen = check_bitlen_rsa(pk->pkey.rsa, ispub, &magic);
497                 keyalg = MS_KEYALG_RSA_KEYX;
498                 }
499         else
500                 return -1;
501         if (bitlen == 0)
502                 return -1;
503         outlen = 16 + blob_length(bitlen,
504                         keyalg == MS_KEYALG_DSS_SIGN ? 1 : 0, ispub);
505         if (out == NULL)
506                 return outlen;
507         if (*out)
508                 p = *out;
509         else
510                 {
511                 p = OPENSSL_malloc(outlen);
512                 if (!p)
513                         return -1;
514                 *out = p;
515                 noinc = 1;
516                 }
517         if (ispub)
518                 *p++ = MS_PUBLICKEYBLOB;
519         else
520                 *p++ = MS_PRIVATEKEYBLOB;
521         *p++ = 0x2;
522         *p++ = 0;
523         *p++ = 0;
```

```
524         write_ledword(&p, keyalg);
525         write_ledword(&p, magic);
526         write_ledword(&p, bitlen);
527         if (keyalg == MS_KEYALG_DSS_SIGN)
528                 write_dsa(&p, pk->pkey.dsa, ispub);
529         else
530                 write_rsa(&p, pk->pkey.rsa, ispub);
531         if (!noinc)
532                 *out += outlen;
533         return outlen;
534         }

536 static int do_i2b_bio(BIO *out, EVP_PKEY *pk, int ispub)
537         {
538         unsigned char *tmp = NULL;
539         int outlen, wrlen;
540         outlen = do_i2b(&tmp, pk, ispub);
541         if (outlen < 0)
542                 return -1;
543         wrlen = BIO_write(out, tmp, outlen);
544         OPENSSL_free(tmp);
545         if (wrlen == outlen)
546                 return outlen;
547         return -1;
548         }

550 static int check_bitlen_dsa(DSA *dsa, int ispub, unsigned int *pmagic)
551         {
552         int bitlen;
553         bitlen = BN_num_bits(dsa->p);
554         if ((bitlen & 7) || (BN_num_bits(dsa->q) != 160)
555                 || (BN_num_bits(dsa->g) > bitlen))
556                 goto badkey;
557         if (ispub)
558                 {
559                 if (BN_num_bits(dsa->pub_key) > bitlen)
560                         goto badkey;
561                 *pmagic = MS_DSS1MAGIC;
562                 }
563         else
564                 {
565                 if (BN_num_bits(dsa->priv_key) > 160)
566                         goto badkey;
567                 *pmagic = MS_DSS2MAGIC;
568                 }

570         return bitlen;
571         badkey:
572         PEMerr(PEM_F_CHECK_BITLEN_DSA, PEM_R_UNSUPPORTED_KEY_COMPONENTS);
573         return 0;
574         }

576 static int check_bitlen_rsa(RSA *rsa, int ispub, unsigned int *pmagic)
577         {
578         int nbyte, hnbyte, bitlen;
579         if (BN_num_bits(rsa->e) > 32)
580                 goto badkey;
581         bitlen = BN_num_bits(rsa->n);
582         nbyte = BN_num_bytes(rsa->n);
583         hnbyte = (BN_num_bits(rsa->n) + 15) >> 4;
584         if (ispub)
585                 {
586                 *pmagic = MS_RSA1MAGIC;
587                 return bitlen;
588                 }
589         else
```



```
590         {
591                 *pmagic = MS_RSA2MAGIC;
592                 /* For private key each component must fit within nbyte or
593                  * hnbyte.
594                  */
595                 if (BN_num_bytes(rsa->d) > nbyte)
596                         goto badkey;
597                 if ((BN_num_bytes(rsa->iqmp) > hnbyte)
598                         || (BN_num_bytes(rsa->p) > hnbyte)
599                         || (BN_num_bytes(rsa->q) > hnbyte)
600                         || (BN_num_bytes(rsa->dmp1) > hnbyte)
601                         || (BN_num_bytes(rsa->dmq1) > hnbyte))
602                         goto badkey;
603         }
604         return bitlen;
605         badkey:
606         PEMerr(PEM_F_CHECK_BITLEN_RSA, PEM_R_UNSUPPORTED_KEY_COMPONENTS);
607         return 0;
608         }


611 static void write_rsa(unsigned char **out, RSA *rsa, int ispub)
612         {
613         int nbyte, hnbyte;
614         nbyte = BN_num_bytes(rsa->n);
615         hnbyte = (BN_num_bits(rsa->n) + 15) >> 4;
616         write_lebn(out, rsa->e, 4);
617         write_lebn(out, rsa->n, -1);
618         if (ispub)
619                 return;
620         write_lebn(out, rsa->p, hnbyte);
621         write_lebn(out, rsa->q, hnbyte);
622         write_lebn(out, rsa->dmp1, hnbyte);
623         write_lebn(out, rsa->dmq1, hnbyte);
624         write_lebn(out, rsa->iqmp, hnbyte);
625         write_lebn(out, rsa->d, nbyte);
626         }


629 static void write_dsa(unsigned char **out, DSA *dsa, int ispub)
630         {
631         int nbyte;
632         nbyte = BN_num_bytes(dsa->p);
633         write_lebn(out, dsa->p, nbyte);
634         write_lebn(out, dsa->q, 20);
635         write_lebn(out, dsa->g, nbyte);
636         if (ispub)
637                 write_lebn(out, dsa->pub_key, nbyte);
638         else
639                 write_lebn(out, dsa->priv_key, 20);
640         /* Set "invalid" for seed structure values */
641         memset(*out, 0xff, 24);
642         *out += 24;
643         return;
644         }


647 int i2b_PrivateKey_bio(BIO *out, EVP_PKEY *pk)
648         {
649         return do_i2b_bio(out, pk, 0);
650         }

652 int i2b_PublicKey_bio(BIO *out, EVP_PKEY *pk)
653         {
654         return do_i2b_bio(out, pk, 1);
655         }
```

```
657 #ifndef OPENSSL_NO_RC4

659 static int do_PVK_header(const unsigned char **in, unsigned int length,
660                  int skip_magic,
661                  unsigned int *psaltlen, unsigned int *pkeylen)

663         {
664         const unsigned char *p = *in;
665         unsigned int pvk_magic, is_encrypted;
666         if (skip_magic)
667                 {
668                 if (length < 20)
669                         {
670                         PEMerr(PEM_F_DO_PVK_HEADER, PEM_R_PVK_TOO_SHORT);
671                         return 0;
672                         }
673                 length -= 20;
674                 }
675         else
676                 {
677                 if (length < 24)
678                         {
679                         PEMerr(PEM_F_DO_PVK_HEADER, PEM_R_PVK_TOO_SHORT);
680                         return 0;
681                         }
682                 length -= 24;
683                 pvk_magic = read_ledword(&p);
684                 if (pvk_magic != MS_PVKMAGIC)
685                         {
686                         PEMerr(PEM_F_DO_PVK_HEADER, PEM_R_BAD_MAGIC_NUMBER);
687                         return 0;
688                         }
689                 }
690         /* Skip reserved */
691         p += 4;
692         /*keytype = */read_ledword(&p);
693         is_encrypted = read_ledword(&p);
694         *psaltlen = read_ledword(&p);
695         *pkeylen = read_ledword(&p);

697         if (is_encrypted && !*psaltlen)
698                 {
699                 PEMerr(PEM_F_DO_PVK_HEADER, PEM_R_INCONSISTENT_HEADER);
700                 return 0;
701                 }

703         *in = p;
704         return 1;
705         }

707 static int derive_pvk_key(unsigned char *key,
708                           const unsigned char *salt, unsigned int saltlen,
709                           const unsigned char *pass, int passlen)
710         {
711         EVP_MD_CTX mctx;
712         int rv = 1;
713         EVP_MD_CTX_init(&mctx);
714         if (!EVP_DigestInit_ex(&mctx, EVP_sha1(), NULL)
715                 || !EVP_DigestUpdate(&mctx, salt, saltlen)
716                 || !EVP_DigestUpdate(&mctx, pass, passlen)
717                 || !EVP_DigestFinal_ex(&mctx, key, NULL))
718                         rv = 0;

720         EVP_MD_CTX_cleanup(&mctx);
721         return rv;
```



```
722         }


725 static EVP_PKEY *do_PVK_body(const unsigned char **in,
726                  unsigned int saltlen, unsigned int keylen,
727                  pem_password_cb *cb, void *u)
728         {
729         EVP_PKEY *ret = NULL;
730         const unsigned char *p = *in;
731         unsigned int magic;
732         unsigned char *enctmp = NULL, *q;
733         EVP_CIPHER_CTX cctx;
734         EVP_CIPHER_CTX_init(&cctx);
735         if (saltlen)
736                 {
737                 char psbuf[PEM_BUFSIZE];
738                 unsigned char keybuf[20];
739                 int enctmplen, inlen;
740                 if (cb)
741                         inlen=cb(psbuf,PEM_BUFSIZE,0,u);
742                 else
743                         inlen=PEM_def_callback(psbuf,PEM_BUFSIZE,0,u);
744                 if (inlen <= 0)
745                         {
746                         PEMerr(PEM_F_DO_PVK_BODY,PEM_R_BAD_PASSWORD_READ);
747                         return NULL;
748                         }
749                 enctmp = OPENSSL_malloc(keylen + 8);
750                 if (!enctmp)
751                         {
752                         PEMerr(PEM_F_DO_PVK_BODY, ERR_R_MALLOC_FAILURE);
753                         return NULL;
754                         }
755                 if (!derive_pvk_key(keybuf, p, saltlen,
756                             (unsigned char *)psbuf, inlen))
757                         return NULL;
758                 p += saltlen;
759                 /* Copy BLOBHEADER across, decrypt rest */
760                 memcpy(enctmp, p, 8);
761                 p += 8;
762                 if (keylen < 8)
763                         {
764                         PEMerr(PEM_F_DO_PVK_BODY, PEM_R_PVK_TOO_SHORT);
765                         return NULL;
766                         }
767                 inlen = keylen - 8;
768                 q = enctmp + 8;
769                 if (!EVP_DecryptInit_ex(&cctx, EVP_rc4(), NULL, keybuf, NULL))
770                         goto err;
771                 if (!EVP_DecryptUpdate(&cctx, q, &enctmplen, p, inlen))
772                         goto err;
773                 if (!EVP_DecryptFinal_ex(&cctx, q + enctmplen, &enctmplen))
774                         goto err;
775                 magic = read_ledword((const unsigned char **)&q);
776                 if (magic != MS_RSA2MAGIC && magic != MS_DSS2MAGIC)
777                         {
778                         q = enctmp + 8;
779                         memset(keybuf + 5, 0, 11);
780                         if (!EVP_DecryptInit_ex(&cctx, EVP_rc4(), NULL, keybuf,
781                                                                         NULL))
782                                 goto err;
783                         OPENSSL_cleanse(keybuf, 20);
784                         if (!EVP_DecryptUpdate(&cctx, q, &enctmplen, p, inlen))
785                                 goto err;
786                         if (!EVP_DecryptFinal_ex(&cctx, q + enctmplen,
787                                                                         &enctmplen))
```

```
788                             goto err;
789                     magic = read_ledword((const unsigned char **)&q);
790                     if (magic != MS_RSA2MAGIC && magic != MS_DSS2MAGIC)
791                             {
792                             PEMerr(PEM_F_DO_PVK_BODY, PEM_R_BAD_DECRYPT);
793                             goto err;
794                             }
795                     }
796             else
797                     OPENSSL_cleanse(keybuf, 20);
798             p = enctmp;
799             }

801         ret = b2i_PrivateKey(&p, keylen);
802         err:
803         EVP_CIPHER_CTX_cleanup(&cctx);
804         if (enctmp && saltlen)
805                 OPENSSL_free(enctmp);
806         return ret;
807         }


810 EVP_PKEY *b2i_PVK_bio(BIO *in, pem_password_cb *cb, void *u)
811         {
812         unsigned char pvk_hdr[24], *buf = NULL;
813         const unsigned char *p;
814         int buflen;
815         EVP_PKEY *ret = NULL;
816         unsigned int saltlen, keylen;
817         if (BIO_read(in, pvk_hdr, 24) != 24)
818                 {
819                 PEMerr(PEM_F_B2I_PVK_BIO, PEM_R_PVK_DATA_TOO_SHORT);
820                 return NULL;
821                 }
822         p = pvk_hdr;

824         if (!do_PVK_header(&p, 24, 0, &saltlen, &keylen))
825                 return 0;
826         buflen = (int) keylen + saltlen;
827         buf = OPENSSL_malloc(buflen);
828         if (!buf)
829                 {
830                 PEMerr(PEM_F_B2I_PVK_BIO, ERR_R_MALLOC_FAILURE);
831                 return 0;
832                 }
833         p = buf;
834         if (BIO_read(in, buf, buflen) != buflen)
835                 {
836                 PEMerr(PEM_F_B2I_PVK_BIO, PEM_R_PVK_DATA_TOO_SHORT);
837                 goto err;
838                 }
839         ret = do_PVK_body(&p, saltlen, keylen, cb, u);

841         err:
842         if (buf)
843                 {
844                 OPENSSL_cleanse(buf, buflen);
845                 OPENSSL_free(buf);
846                 }
847         return ret;
848         }



852 static int i2b_PVK(unsigned char **out, EVP_PKEY*pk, int enclevel,
853                 pem_password_cb *cb, void *u)
```



```
854         {
855         int outlen = 24, pklen;
856         unsigned char *p, *salt = NULL;
857         EVP_CIPHER_CTX cctx;
858         EVP_CIPHER_CTX_init(&cctx);
859         if (enclevel)
860                 outlen += PVK_SALTLEN;
861         pklen = do_i2b(NULL, pk, 0);
862         if (pklen < 0)
863                 return -1;
864         outlen += pklen;
865         if (!out)
866                 return outlen;
867         if (*out)
868                 p = *out;
869         else
870                 {
871                 p = OPENSSL_malloc(outlen);
872                 if (!p)
873                         {
874                         PEMerr(PEM_F_I2B_PVK,ERR_R_MALLOC_FAILURE);
875                         return -1;
876                         }
877                 *out = p;
878                 }

880         write_ledword(&p, MS_PVKMAGIC);
881         write_ledword(&p, 0);
882         if (pk->type == EVP_PKEY_DSA)
883                 write_ledword(&p, MS_KEYTYPE_SIGN);
884         else
885                 write_ledword(&p, MS_KEYTYPE_KEYX);
886         write_ledword(&p, enclevel ? 1 : 0);
887         write_ledword(&p, enclevel ? PVK_SALTLEN: 0);
888         write_ledword(&p, pklen);
889         if (enclevel)
890                 {
891                 if (RAND_bytes(p, PVK_SALTLEN) <= 0)
892                         goto error;
893                 salt = p;
894                 p += PVK_SALTLEN;
895                 }
896         do_i2b(&p, pk, 0);
897         if (enclevel == 0)
898                 return outlen;
899         else
900                 {
901                 char psbuf[PEM_BUFSIZE];
902                 unsigned char keybuf[20];
903                 int enctmplen, inlen;
904                 if (cb)
905                         inlen=cb(psbuf,PEM_BUFSIZE,1,u);
906                 else
907                         inlen=PEM_def_callback(psbuf,PEM_BUFSIZE,1,u);
908                 if (inlen <= 0)
909                         {
910                         PEMerr(PEM_F_I2B_PVK,PEM_R_BAD_PASSWORD_READ);
911                         goto error;
912                         }
913                 if (!derive_pvk_key(keybuf, salt, PVK_SALTLEN,
914                             (unsigned char *)psbuf, inlen))
915                         goto error;
916                 if (enclevel == 1)
917                         memset(keybuf + 5, 0, 11);
918                 p = salt + PVK_SALTLEN + 8;
919                 if (!EVP_EncryptInit_ex(&cctx, EVP_rc4(), NULL, keybuf, NULL))
```

```
920                         goto error;
921                 OPENSSL_cleanse(keybuf, 20);
922                 if (!EVP_DecryptUpdate(&cctx, p, &enctmplen, p, pklen - 8))
923                         goto error;
924                 if (!EVP_DecryptFinal_ex(&cctx, p + enctmplen, &enctmplen))
925                         goto error;
926                 }
927         EVP_CIPHER_CTX_cleanup(&cctx);
928         return outlen;

930         error:
931         EVP_CIPHER_CTX_cleanup(&cctx);
932         return -1;
933         }

935 int i2b_PVK_bio(BIO *out, EVP_PKEY *pk, int enclevel,
936                 pem_password_cb *cb, void *u)
937         {
938         unsigned char *tmp = NULL;
939         int outlen, wrlen;
940         outlen = i2b_PVK(&tmp, pk, enclevel, cb, u);
941         if (outlen < 0)
942                 return -1;
943         wrlen = BIO_write(out, tmp, outlen);
944         OPENSSL_free(tmp);
945         if (wrlen == outlen)
946                 {
947                 PEMerr(PEM_F_I2B_PVK_BIO, PEM_R_BIO_WRITE_FAILURE);
948                 return outlen;
949                 }
950         return -1;
951         }

953 #endif

955 #endif
956 #endif /* ! codereview */
```

```
**********************************************************
    7735 Wed Aug 13 19:53:01 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_add.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_add.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
/* Pack an object into an OCTET STRING and turn into a safebag */

PKCS12_SAFEBAG *PKCS12_item_pack_safebag(void *obj, const ASN1_ITEM *it, int nid
             int nid2)
{
        PKCS12_BAGS *bag;
        PKCS12_SAFEBAG *safebag;
        if (!(bag = PKCS12_BAGS_new())) {
                PKCS12err(PKCS12_F_PKCS12_ITEM_PACK_SAFEBAG, ERR_R_MALLOC_FAILUR
                return NULL;
        }
        bag->type = OBJ_nid2obj(nid1);
        if (!ASN1_item_pack(obj, it, &bag->value.octet)) {
                PKCS12err(PKCS12_F_PKCS12_ITEM_PACK_SAFEBAG, ERR_R_MALLOC_FAILUR
                return NULL;
        }
        if (!(safebag = PKCS12_SAFEBAG_new())) {
                PKCS12err(PKCS12_F_PKCS12_ITEM_PACK_SAFEBAG, ERR_R_MALLOC_FAILUR
                return NULL;
        }
        safebag->value.bag = bag;
        safebag->type = OBJ_nid2obj(nid2);
        return safebag;
}

/* Turn PKCS8 object into a keybag */

PKCS12_SAFEBAG *PKCS12_MAKE_KEYBAG(PKCS8_PRIV_KEY_INFO *p8)
{
        PKCS12_SAFEBAG *bag;
        if (!(bag = PKCS12_SAFEBAG_new())) {
                PKCS12err(PKCS12_F_PKCS12_MAKE_KEYBAG,ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        bag->type = OBJ_nid2obj(NID_keyBag);
        bag->value.keybag = p8;
        return bag;
}

/* Turn PKCS8 object into a shrouded keybag */

PKCS12_SAFEBAG *PKCS12_MAKE_SHKEYBAG(int pbe_nid, const char *pass,
             int passlen, unsigned char *salt, int saltlen, int iter,
             PKCS8_PRIV_KEY_INFO *p8)
{
        PKCS12_SAFEBAG *bag;
        const EVP_CIPHER *pbe_ciph;

        /* Set up the safe bag */
        if (!(bag = PKCS12_SAFEBAG_new())) {
                PKCS12err(PKCS12_F_PKCS12_MAKE_SHKEYBAG, ERR_R_MALLOC_FAILURE);
                return NULL;
        }

        bag->type = OBJ_nid2obj(NID_pkcs8ShroudedKeyBag);

        pbe_ciph = EVP_get_cipherbynid(pbe_nid);

        if (pbe_ciph)
                pbe_nid = -1;

        if (!(bag->value.shkeybag =
          PKCS8_encrypt(pbe_nid, pbe_ciph, pass, passlen, salt, saltlen, iter,
                                                                      p8))) {
                PKCS12err(PKCS12_F_PKCS12_MAKE_SHKEYBAG, ERR_R_MALLOC_FAILURE);
```

```
128                 return NULL;
129         }

131         return bag;
132 }

134 /* Turn a stack of SAFEBAGS into a PKCS#7 data Contentinfo */
135 PKCS7 *PKCS12_pack_p7data(STACK_OF(PKCS12_SAFEBAG) *sk)
136 {
137         PKCS7 *p7;
138         if (!(p7 = PKCS7_new())) {
139                 PKCS12err(PKCS12_F_PKCS12_PACK_P7DATA, ERR_R_MALLOC_FAILURE);
140                 return NULL;
141         }
142         p7->type = OBJ_nid2obj(NID_pkcs7_data);
143         if (!(p7->d.data = M_ASN1_OCTET_STRING_new())) {
144                 PKCS12err(PKCS12_F_PKCS12_PACK_P7DATA, ERR_R_MALLOC_FAILURE);
145                 return NULL;
146         }

148         if (!ASN1_item_pack(sk, ASN1_ITEM_rptr(PKCS12_SAFEBAGS), &p7->d.data)) {
149                 PKCS12err(PKCS12_F_PKCS12_PACK_P7DATA, PKCS12_R_CANT_PACK_STRUCT
150                 return NULL;
151         }
152         return p7;
153 }

155 /* Unpack SAFEBAGS from PKCS#7 data ContentInfo */
156 STACK_OF(PKCS12_SAFEBAG) *PKCS12_unpack_p7data(PKCS7 *p7)
157 {
158         if(!PKCS7_type_is_data(p7))
159                 {
160                 PKCS12err(PKCS12_F_PKCS12_UNPACK_P7DATA,PKCS12_R_CONTENT_TYPE_NO
161                 return NULL;
162                 }
163         return ASN1_item_unpack(p7->d.data, ASN1_ITEM_rptr(PKCS12_SAFEBAGS));
164 }

166 /* Turn a stack of SAFEBAGS into a PKCS#7 encrypted data ContentInfo */

168 PKCS7 *PKCS12_pack_p7encdata(int pbe_nid, const char *pass, int passlen,
169                              unsigned char *salt, int saltlen, int iter,
170                              STACK_OF(PKCS12_SAFEBAG) *bags)
171 {
172         PKCS7 *p7;
173         X509_ALGOR *pbe;
174         const EVP_CIPHER *pbe_ciph;
175         if (!(p7 = PKCS7_new())) {
176                 PKCS12err(PKCS12_F_PKCS12_PACK_P7ENCDATA, ERR_R_MALLOC_FAILURE);
177                 return NULL;
178         }
179         if(!PKCS7_set_type(p7, NID_pkcs7_encrypted)) {
180                 PKCS12err(PKCS12_F_PKCS12_PACK_P7ENCDATA,
181                                 PKCS12_R_ERROR_SETTING_ENCRYPTED_DATA_TYPE);
182                 return NULL;
183         }

185         pbe_ciph = EVP_get_cipherbynid(pbe_nid);

187         if (pbe_ciph)
188                 pbe = PKCS5_pbe2_set(pbe_ciph, iter, salt, saltlen);
189         else
190                 pbe = PKCS5_pbe_set(pbe_nid, iter, salt, saltlen);

192         if (!pbe) {
193                 PKCS12err(PKCS12_F_PKCS12_PACK_P7ENCDATA, ERR_R_MALLOC_FAILURE);
```



```
194                 return NULL;
195         }
196         X509_ALGOR_free(p7->d.encrypted->enc_data->algorithm);
197         p7->d.encrypted->enc_data->algorithm = pbe;
198         M_ASN1_OCTET_STRING_free(p7->d.encrypted->enc_data->enc_data);
199         if (!(p7->d.encrypted->enc_data->enc_data =
200         PKCS12_item_i2d_encrypt(pbe, ASN1_ITEM_rptr(PKCS12_SAFEBAGS), pass, pass
201                                 bags, 1))) {
202                 PKCS12err(PKCS12_F_PKCS12_PACK_P7ENCDATA, PKCS12_R_ENCRYPT_ERROR
203                 return NULL;
204         }

206         return p7;
207 }

209 STACK_OF(PKCS12_SAFEBAG) *PKCS12_unpack_p7encdata(PKCS7 *p7, const char *pass, i
210 {
211         if(!PKCS7_type_is_encrypted(p7)) return NULL;
212         return PKCS12_item_decrypt_d2i(p7->d.encrypted->enc_data->algorithm,
213                                 ASN1_ITEM_rptr(PKCS12_SAFEBAGS),
214                                 pass, passlen,
215                                 p7->d.encrypted->enc_data->enc_data, 1);
216 }

218 PKCS8_PRIV_KEY_INFO *PKCS12_decrypt_skey(PKCS12_SAFEBAG *bag, const char *pass,
219                                                                 int passlen)
220 {
221         return PKCS8_decrypt(bag->value.shkeybag, pass, passlen);
222 }

224 int PKCS12_pack_authsafes(PKCS12 *p12, STACK_OF(PKCS7) *safes)
225 {
226         if(ASN1_item_pack(safes, ASN1_ITEM_rptr(PKCS12_AUTHSAFES),
227                 &p12->authsafes->d.data))
228                         return 1;
229         return 0;
230 }

232 STACK_OF(PKCS7) *PKCS12_unpack_authsafes(PKCS12 *p12)
233 {
234         if (!PKCS7_type_is_data(p12->authsafes))
235                 {
236                 PKCS12err(PKCS12_F_PKCS12_UNPACK_AUTHSAFES,PKCS12_R_CONTENT_TYPE
237                 return NULL;
238                 }
239         return ASN1_item_unpack(p12->authsafes->d.data, ASN1_ITEM_rptr(PKCS12_AU
240 }
241 #endif /* ! codereview */
```

```
**********************************************************
    5295 Wed Aug 13 19:53:01 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_asn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_asn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/pkcs12.h>

/* PKCS#12 ASN1 module */

ASN1_SEQUENCE(PKCS12) = {
        ASN1_SIMPLE(PKCS12, version, ASN1_INTEGER),
        ASN1_SIMPLE(PKCS12, authsafes, PKCS7),
        ASN1_OPT(PKCS12, mac, PKCS12_MAC_DATA)
} ASN1_SEQUENCE_END(PKCS12)

IMPLEMENT_ASN1_FUNCTIONS(PKCS12)

ASN1_SEQUENCE(PKCS12_MAC_DATA) = {
        ASN1_SIMPLE(PKCS12_MAC_DATA, dinfo, X509_SIG),
        ASN1_SIMPLE(PKCS12_MAC_DATA, salt, ASN1_OCTET_STRING),
        ASN1_OPT(PKCS12_MAC_DATA, iter, ASN1_INTEGER)
} ASN1_SEQUENCE_END(PKCS12_MAC_DATA)

IMPLEMENT_ASN1_FUNCTIONS(PKCS12_MAC_DATA)

ASN1_ADB_TEMPLATE(bag_default) = ASN1_EXP(PKCS12_BAGS, value.other, ASN1_ANY, 0)

ASN1_ADB(PKCS12_BAGS) = {
        ADB_ENTRY(NID_x509Certificate, ASN1_EXP(PKCS12_BAGS, value.x509cert, ASN
        ADB_ENTRY(NID_x509Crl, ASN1_EXP(PKCS12_BAGS, value.x509crl, ASN1_OCTET_S
        ADB_ENTRY(NID_sdsiCertificate, ASN1_EXP(PKCS12_BAGS, value.sdsicert, ASN
} ASN1_ADB_END(PKCS12_BAGS, 0, type, 0, &bag_default_tt, NULL);

ASN1_SEQUENCE(PKCS12_BAGS) = {
        ASN1_SIMPLE(PKCS12_BAGS, type, ASN1_OBJECT),
        ASN1_ADB_OBJECT(PKCS12_BAGS),
} ASN1_SEQUENCE_END(PKCS12_BAGS)

IMPLEMENT_ASN1_FUNCTIONS(PKCS12_BAGS)

ASN1_ADB_TEMPLATE(safebag_default) = ASN1_EXP(PKCS12_SAFEBAG, value.other, ASN1_

ASN1_ADB(PKCS12_SAFEBAG) = {
        ADB_ENTRY(NID_keyBag, ASN1_EXP(PKCS12_SAFEBAG, value.keybag, PKCS8_PRIV_
        ADB_ENTRY(NID_pkcs8ShroudedKeyBag, ASN1_EXP(PKCS12_SAFEBAG, value.shkeyb
        ADB_ENTRY(NID_safeContentsBag, ASN1_EXP_SET_OF(PKCS12_SAFEBAG, value.saf
        ADB_ENTRY(NID_certBag, ASN1_EXP(PKCS12_SAFEBAG, value.bag, PKCS12_BAGS,
        ADB_ENTRY(NID_crlBag, ASN1_EXP(PKCS12_SAFEBAG, value.bag, PKCS12_BAGS, 0
        ADB_ENTRY(NID_secretBag, ASN1_EXP(PKCS12_SAFEBAG, value.bag, PKCS12_BAGS
} ASN1_ADB_END(PKCS12_SAFEBAG, 0, type, 0, &safebag_default_tt, NULL);

ASN1_SEQUENCE(PKCS12_SAFEBAG) = {
        ASN1_SIMPLE(PKCS12_SAFEBAG, type, ASN1_OBJECT),
        ASN1_ADB_OBJECT(PKCS12_SAFEBAG),
        ASN1_SET_OF_OPT(PKCS12_SAFEBAG, attrib, X509_ATTRIBUTE)
} ASN1_SEQUENCE_END(PKCS12_SAFEBAG)

IMPLEMENT_ASN1_FUNCTIONS(PKCS12_SAFEBAG)

/* SEQUENCE OF SafeBag */
ASN1_ITEM_TEMPLATE(PKCS12_SAFEBAGS) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, PKCS12_SAFEBAGS, PKCS12_
ASN1_ITEM_TEMPLATE_END(PKCS12_SAFEBAGS)

/* Authsafes: SEQUENCE OF PKCS7 */
ASN1_ITEM_TEMPLATE(PKCS12_AUTHSAFES) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, PKCS12_AUTHSAFES, PKCS7)
ASN1_ITEM_TEMPLATE_END(PKCS12_AUTHSAFES)
#endif /* ! codereview */
```

```
**********************************************************
    4809 Wed Aug 13 19:53:01 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_attr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_attr.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
/* Add a local keyid to a safebag */

int PKCS12_add_localkeyid(PKCS12_SAFEBAG *bag, unsigned char *name,
             int namelen)
{
        if (X509at_add1_attr_by_NID(&bag->attrib, NID_localKeyID,
                                    V_ASN1_OCTET_STRING, name, namelen))
                return 1;
        else
                return 0;
}

/* Add key usage to PKCS#8 structure */

int PKCS8_add_keyusage(PKCS8_PRIV_KEY_INFO *p8, int usage)
{
        unsigned char us_val;
        us_val = (unsigned char) usage;
        if (X509at_add1_attr_by_NID(&p8->attributes, NID_key_usage,
                                    V_ASN1_BIT_STRING, &us_val, 1))
                return 1;
        else
                return 0;
}

/* Add a friendlyname to a safebag */

int PKCS12_add_friendlyname_asc(PKCS12_SAFEBAG *bag, const char *name,
                                int namelen)
{
        if (X509at_add1_attr_by_NID(&bag->attrib, NID_friendlyName,
                                    MBSTRING_ASC, (unsigned char *)name, namelen))
                return 1;
        else
                return 0;
}


int PKCS12_add_friendlyname_uni(PKCS12_SAFEBAG *bag,
                                const unsigned char *name, int namelen)
{
        if (X509at_add1_attr_by_NID(&bag->attrib, NID_friendlyName,
                                    MBSTRING_BMP, name, namelen))
                return 1;
        else
                return 0;
}

int PKCS12_add_CSPName_asc(PKCS12_SAFEBAG *bag, const char *name,
                                int namelen)
{
        if (X509at_add1_attr_by_NID(&bag->attrib, NID_ms_csp_name,
                                    MBSTRING_ASC, (unsigned char *)name, namelen))
                return 1;
        else
                return 0;
}

ASN1_TYPE *PKCS12_get_attr_gen(STACK_OF(X509_ATTRIBUTE) *attrs, int attr_nid)
{
        X509_ATTRIBUTE *attrib;
        int i;
        if (!attrs) return NULL;
        for (i = 0; i < sk_X509_ATTRIBUTE_num (attrs); i++) {
                attrib = sk_X509_ATTRIBUTE_value (attrs, i);
```

```
128                 if (OBJ_obj2nid (attrib->object) == attr_nid) {
129                         if (sk_ASN1_TYPE_num (attrib->value.set))
130                             return sk_ASN1_TYPE_value(attrib->value.set, 0);
131                         else return NULL;
132                 }
133         }
134         return NULL;
135 }

137 char *PKCS12_get_friendlyname(PKCS12_SAFEBAG *bag)
138 {
139         ASN1_TYPE *atype;
140         if (!(atype = PKCS12_get_attr(bag, NID_friendlyName))) return NULL;
141         if (atype->type != V_ASN1_BMPSTRING) return NULL;
142         return OPENSSL_uni2asc(atype->value.bmpstring->data,
143                                atype->value.bmpstring->length);
144 }
145 #endif /* ! codereview */
```

```
**********************************************************
    4354 Wed Aug 13 19:53:01 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_crpt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_crpt.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
/* PKCS#12 PBE algorithms now in static table */

void PKCS12_PBE_add(void)
{
}

int PKCS12_PBE_keyivgen(EVP_CIPHER_CTX *ctx, const char *pass, int passlen,
                ASN1_TYPE *param, const EVP_CIPHER *cipher, const EVP_MD *md, in
{
        PBEPARAM *pbe;
        int saltlen, iter, ret;
        unsigned char *salt;
        const unsigned char *pbuf;
        unsigned char key[EVP_MAX_KEY_LENGTH], iv[EVP_MAX_IV_LENGTH];

        /* Extract useful info from parameter */
        if (param == NULL || param->type != V_ASN1_SEQUENCE ||
            param->value.sequence == NULL) {
                PKCS12err(PKCS12_F_PKCS12_PBE_KEYIVGEN,PKCS12_R_DECODE_ERROR);
                return 0;
        }

        pbuf = param->value.sequence->data;
        if (!(pbe = d2i_PBEPARAM(NULL, &pbuf, param->value.sequence->length))) {
                PKCS12err(PKCS12_F_PKCS12_PBE_KEYIVGEN,PKCS12_R_DECODE_ERROR);
                return 0;
        }

        if (!pbe->iter) iter = 1;
        else iter = ASN1_INTEGER_get (pbe->iter);
        salt = pbe->salt->data;
        saltlen = pbe->salt->length;
        if (!PKCS12_key_gen (pass, passlen, salt, saltlen, PKCS12_KEY_ID,
                             iter, EVP_CIPHER_key_length(cipher), key, md)) {
                PKCS12err(PKCS12_F_PKCS12_PBE_KEYIVGEN,PKCS12_R_KEY_GEN_ERROR);
                PBEPARAM_free(pbe);
                return 0;
        }
        if (!PKCS12_key_gen (pass, passlen, salt, saltlen, PKCS12_IV_ID,
                                iter, EVP_CIPHER_iv_length(cipher), iv, md)) {
                PKCS12err(PKCS12_F_PKCS12_PBE_KEYIVGEN,PKCS12_R_IV_GEN_ERROR);
                PBEPARAM_free(pbe);
                return 0;
        }
        PBEPARAM_free(pbe);
        ret = EVP_CipherInit_ex(ctx, cipher, NULL, key, iv, en_de);
        OPENSSL_cleanse(key, EVP_MAX_KEY_LENGTH);
        OPENSSL_cleanse(iv, EVP_MAX_IV_LENGTH);
        return ret;
}
#endif /* ! codereview */
```

```
**********************************************************
    8518 Wed Aug 13 19:53:01 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_crt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_crt.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 1999-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
static int pkcs12_add_bag(STACK_OF(PKCS12_SAFEBAG) **pbags, PKCS12_SAFEBAG *bag)

static int copy_bag_attr(PKCS12_SAFEBAG *bag, EVP_PKEY *pkey, int nid)
        {
        int idx;
        X509_ATTRIBUTE *attr;
        idx = EVP_PKEY_get_attr_by_NID(pkey, nid, -1);
        if (idx < 0)
                return 1;
        attr = EVP_PKEY_get_attr(pkey, idx);
        if (!X509at_add1_attr(&bag->attrib, attr))
                return 0;
        return 1;
        }

PKCS12 *PKCS12_create(char *pass, char *name, EVP_PKEY *pkey, X509 *cert,
             STACK_OF(X509) *ca, int nid_key, int nid_cert, int iter, int mac_it
             int keytype)
{
        PKCS12 *p12 = NULL;
        STACK_OF(PKCS7) *safes = NULL;
        STACK_OF(PKCS12_SAFEBAG) *bags = NULL;
        PKCS12_SAFEBAG *bag = NULL;
        int i;
        unsigned char keyid[EVP_MAX_MD_SIZE];
        unsigned int keyidlen = 0;

        /* Set defaults */
        if (!nid_cert)
                {
#ifdef OPENSSL_FIPS
                if (FIPS_mode())
                        nid_cert = NID_pbe_WithSHA1And3_Key_TripleDES_CBC;
                else
#endif
#ifdef OPENSSL_NO_RC2
                nid_cert = NID_pbe_WithSHA1And3_Key_TripleDES_CBC;
#else
                nid_cert = NID_pbe_WithSHA1And40BitRC2_CBC;
#endif
                }
        if (!nid_key)
                nid_key = NID_pbe_WithSHA1And3_Key_TripleDES_CBC;
        if (!iter)
                iter = PKCS12_DEFAULT_ITER;
        if (!mac_iter)
                mac_iter = 1;

        if(!pkey && !cert && !ca)
                {
                PKCS12err(PKCS12_F_PKCS12_CREATE,PKCS12_R_INVALID_NULL_ARGUMENT)
                return NULL;
                }

        if (pkey && cert)
                {
                if(!X509_check_private_key(cert, pkey))
                        return NULL;
                X509_digest(cert, EVP_sha1(), keyid, &keyidlen);
                }

        if (cert)
                {
                bag = PKCS12_add_cert(&bags, cert);
```

```
128                 if(name && !PKCS12_add_friendlyname(bag, name, -1))
129                         goto err;
130                 if(keyidlen && !PKCS12_add_localkeyid(bag, keyid, keyidlen))
131                         goto err;
132                 }

134         /* Add all other certificates */
135         for(i = 0; i < sk_X509_num(ca); i++)
136                 {
137                 if (!PKCS12_add_cert(&bags, sk_X509_value(ca, i)))
138                         goto err;
139                 }

141         if (bags && !PKCS12_add_safe(&safes, bags, nid_cert, iter, pass))
142                         goto err;

144         sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
145         bags = NULL;

147         if (pkey)
148                 {
149                 bag = PKCS12_add_key(&bags, pkey, keytype, iter, nid_key, pass);

151                 if (!bag)
152                         goto err;

154                 if (!copy_bag_attr(bag, pkey, NID_ms_csp_name))
155                         goto err;
156                 if (!copy_bag_attr(bag, pkey, NID_LocalKeySet))
157                         goto err;

159                 if(name && !PKCS12_add_friendlyname(bag, name, -1))
160                         goto err;
161                 if(keyidlen && !PKCS12_add_localkeyid(bag, keyid, keyidlen))
162                         goto err;
163                 }

165         if (bags && !PKCS12_add_safe(&safes, bags, -1, 0, NULL))
166                         goto err;

168         sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
169         bags = NULL;

171         p12 = PKCS12_add_safes(safes, 0);

173         if (!p12)
174                 goto err;

176         sk_PKCS7_pop_free(safes, PKCS7_free);

178         safes = NULL;

180         if ((mac_iter != -1) &&
181                 !PKCS12_set_mac(p12, pass, -1, NULL, 0, mac_iter, NULL))
182             goto err;

184         return p12;

186         err:

188         if (p12)
189                 PKCS12_free(p12);
190         if (safes)
191                 sk_PKCS7_pop_free(safes, PKCS7_free);
192         if (bags)
193                 sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
```



```
194         return NULL;

196 }

198 PKCS12_SAFEBAG *PKCS12_add_cert(STACK_OF(PKCS12_SAFEBAG) **pbags, X509 *cert)
199         {
200         PKCS12_SAFEBAG *bag = NULL;
201         char *name;
202         int namelen = -1;
203         unsigned char *keyid;
204         int keyidlen = -1;

206         /* Add user certificate */
207         if(!(bag = PKCS12_x5092certbag(cert)))
208                 goto err;

210         /* Use friendlyName and localKeyID in certificate.
211          * (if present)
212          */

214         name = (char *)X509_alias_get0(cert, &namelen);

216         if(name && !PKCS12_add_friendlyname(bag, name, namelen))
217                 goto err;

219         keyid = X509_keyid_get0(cert, &keyidlen);

221         if(keyid && !PKCS12_add_localkeyid(bag, keyid, keyidlen))
222                 goto err;

224         if (!pkcs12_add_bag(pbags, bag))
225                 goto err;

227         return bag;

229         err:

231         if (bag)
232                 PKCS12_SAFEBAG_free(bag);

234         return NULL;

236         }

238 PKCS12_SAFEBAG *PKCS12_add_key(STACK_OF(PKCS12_SAFEBAG) **pbags, EVP_PKEY *key,
239                                                 int key_usage, int iter,
240                                                 int nid_key, char *pass)
241         {

243         PKCS12_SAFEBAG *bag = NULL;
244         PKCS8_PRIV_KEY_INFO *p8 = NULL;

246         /* Make a PKCS#8 structure */
247         if(!(p8 = EVP_PKEY2PKCS8(key)))
248                 goto err;
249         if(key_usage && !PKCS8_add_keyusage(p8, key_usage))
250                 goto err;
251         if (nid_key != -1)
252                 {
253                 bag = PKCS12_MAKE_SHKEYBAG(nid_key, pass, -1, NULL, 0, iter, p8)
254                 PKCS8_PRIV_KEY_INFO_free(p8);
255                 }
256         else
257                 bag = PKCS12_MAKE_KEYBAG(p8);

259         if(!bag)
```

```
260                 goto err;

262         if (!pkcs12_add_bag(pbags, bag))
263                 goto err;

265         return bag;

267         err:

269         if (bag)
270                 PKCS12_SAFEBAG_free(bag);

272         return NULL;

274         }

276 int PKCS12_add_safe(STACK_OF(PKCS7) **psafes, STACK_OF(PKCS12_SAFEBAG) *bags,
277                                                 int nid_safe, int iter, char *pa
278         {
279         PKCS7 *p7 = NULL;
280         int free_safes = 0;

282         if (!*psafes)
283                 {
284                 *psafes = sk_PKCS7_new_null();
285                 if (!*psafes)
286                         return 0;
287                 free_safes = 1;
288                 }
289         else
290                 free_safes = 0;

292         if (nid_safe == 0)
293 #ifdef OPENSSL_NO_RC2
294                 nid_safe = NID_pbe_WithSHA1And3_Key_TripleDES_CBC;
295 #else
296                 nid_safe = NID_pbe_WithSHA1And40BitRC2_CBC;
297 #endif

299         if (nid_safe == -1)
300                 p7 = PKCS12_pack_p7data(bags);
301         else
302                 p7 = PKCS12_pack_p7encdata(nid_safe, pass, -1, NULL, 0,
303                                         iter, bags);
304         if (!p7)
305                 goto err;

307         if (!sk_PKCS7_push(*psafes, p7))
308                 goto err;

310         return 1;

312         err:
313         if (free_safes)
314                 {
315                 sk_PKCS7_free(*psafes);
316                 *psafes = NULL;
317                 }

319         if (p7)
320                 PKCS7_free(p7);

322         return 0;

324         }
```



```
326 static int pkcs12_add_bag(STACK_OF(PKCS12_SAFEBAG) **pbags, PKCS12_SAFEBAG *bag)
327         {
328         int free_bags;
329         if (!pbags)
330                 return 1;
331         if (!*pbags)
332                 {
333                 *pbags = sk_PKCS12_SAFEBAG_new_null();
334                 if (!*pbags)
335                         return 0;
336                 free_bags = 1;
337                 }
338         else
339                 free_bags = 0;

341         if (!sk_PKCS12_SAFEBAG_push(*pbags, bag))
342                 {
343                 if (free_bags)
344                         {
345                         sk_PKCS12_SAFEBAG_free(*pbags);
346                         *pbags = NULL;
347                         }
348                 return 0;
349                 }

351         return 1;

353         }


356 PKCS12 *PKCS12_add_safes(STACK_OF(PKCS7) *safes, int nid_p7)
357         {
358         PKCS12 *p12;
359         if (nid_p7 <= 0)
360                 nid_p7 = NID_pkcs7_data;
361         p12 = PKCS12_init(nid_p7);

363         if (!p12)
364                 return NULL;

366         if(!PKCS12_pack_authsafes(p12, safes))
367                 {
368                 PKCS12_free(p12);
369                 return NULL;
370                 }

372         return p12;

374         }
375 #endif /* ! codereview */
```

```
**********************************************************
    5900 Wed Aug 13 19:53:01 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_decr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_decr.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
/* Define this to dump decrypted output to files called DERnnn */
/*#define DEBUG_DECRYPT*/


/* Encrypt/Decrypt a buffer based on password and algor, result in a
 * OPENSSL_malloc'ed buffer
 */

unsigned char * PKCS12_pbe_crypt(X509_ALGOR *algor, const char *pass,
             int passlen, unsigned char *in, int inlen, unsigned char **data,
             int *datalen, int en_de)
{
        unsigned char *out;
        int outlen, i;
        EVP_CIPHER_CTX ctx;

        EVP_CIPHER_CTX_init(&ctx);
        /* Decrypt data */
        if (!EVP_PBE_CipherInit(algor->algorithm, pass, passlen,
                                algor->parameter, &ctx, en_de)) {
                PKCS12err(PKCS12_F_PKCS12_PBE_CRYPT,PKCS12_R_PKCS12_ALGOR_CIPHER
                return NULL;
        }

        if(!(out = OPENSSL_malloc(inlen + EVP_CIPHER_CTX_block_size(&ctx)))) {
                PKCS12err(PKCS12_F_PKCS12_PBE_CRYPT,ERR_R_MALLOC_FAILURE);
                goto err;
        }

        if (!EVP_CipherUpdate(&ctx, out, &i, in, inlen))
                {
                OPENSSL_free(out);
                out = NULL;
                PKCS12err(PKCS12_F_PKCS12_PBE_CRYPT,ERR_R_EVP_LIB);
                goto err;
                }

        outlen = i;
        if(!EVP_CipherFinal_ex(&ctx, out + i, &i)) {
                OPENSSL_free(out);
                out = NULL;
                PKCS12err(PKCS12_F_PKCS12_PBE_CRYPT,PKCS12_R_PKCS12_CIPHERFINAL_
                goto err;
        }
        outlen += i;
        if (datalen) *datalen = outlen;
        if (data) *data = out;
        err:
        EVP_CIPHER_CTX_cleanup(&ctx);
        return out;

}

/* Decrypt an OCTET STRING and decode ASN1 structure
 * if zbuf set zero buffer after use.
 */

void * PKCS12_item_decrypt_d2i(X509_ALGOR *algor, const ASN1_ITEM *it,
             const char *pass, int passlen, ASN1_OCTET_STRING *oct, int zbuf)
{
        unsigned char *out;
        const unsigned char *p;
        void *ret;
        int outlen;
```

```
128         if (!PKCS12_pbe_crypt(algor, pass, passlen, oct->data, oct->length,
129                               &out, &outlen, 0)) {
130                 PKCS12err(PKCS12_F_PKCS12_ITEM_DECRYPT_D2I,PKCS12_R_PKCS12_PBE_C
131                 return NULL;
132         }
133         p = out;
134 #ifdef DEBUG_DECRYPT
135         {
136                 FILE *op;

138                 char fname[30];
139                 static int fnm = 1;
140                 sprintf(fname, "DER%d", fnm++);
141                 op = fopen(fname, "wb");
142                 fwrite (p, 1, outlen, op);
143                 fclose(op);
144         }
145 #endif
146         ret = ASN1_item_d2i(NULL, &p, outlen, it);
147         if (zbuf) OPENSSL_cleanse(out, outlen);
148         if(!ret) PKCS12err(PKCS12_F_PKCS12_ITEM_DECRYPT_D2I,PKCS12_R_DECODE_ERRO
149         OPENSSL_free(out);
150         return ret;
151 }

153 /* Encode ASN1 structure and encrypt, return OCTET STRING
154  * if zbuf set zero encoding.
155  */

157 ASN1_OCTET_STRING *PKCS12_item_i2d_encrypt(X509_ALGOR *algor, const ASN1_ITEM *i
158                                            const char *pass, int passlen,
159                                            void *obj, int zbuf)
160 {
161         ASN1_OCTET_STRING *oct;
162         unsigned char *in = NULL;
163         int inlen;
164         if (!(oct = M_ASN1_OCTET_STRING_new ())) {
165                 PKCS12err(PKCS12_F_PKCS12_ITEM_I2D_ENCRYPT,ERR_R_MALLOC_FAILURE)
166                 return NULL;
167         }
168         inlen = ASN1_item_i2d(obj, &in, it);
169         if (!in) {
170                 PKCS12err(PKCS12_F_PKCS12_ITEM_I2D_ENCRYPT,PKCS12_R_ENCODE_ERROR
171                 return NULL;
172         }
173         if (!PKCS12_pbe_crypt(algor, pass, passlen, in, inlen, &oct->data,
174                               &oct->length, 1)) {
175                 PKCS12err(PKCS12_F_PKCS12_ITEM_I2D_ENCRYPT,PKCS12_R_ENCRYPT_ERRO
176                 OPENSSL_free(in);
177                 return NULL;
178         }
179         if (zbuf) OPENSSL_cleanse(in, inlen);
180         OPENSSL_free(in);
181         return oct;
182 }

184 IMPLEMENT_PKCS12_STACK_OF(PKCS7)
185 #endif /* ! codereview */
```

```
**********************************************************
    3471 Wed Aug 13 19:53:02 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_init.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_init.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
/* Initialise a PKCS12 structure to take data */

PKCS12 *PKCS12_init(int mode)
{
        PKCS12 *pkcs12;
        if (!(pkcs12 = PKCS12_new())) {
                PKCS12err(PKCS12_F_PKCS12_INIT,ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        ASN1_INTEGER_set(pkcs12->version, 3);
        pkcs12->authsafes->type = OBJ_nid2obj(mode);
        switch (mode) {
                case NID_pkcs7_data:
                        if (!(pkcs12->authsafes->d.data =
                                M_ASN1_OCTET_STRING_new())) {
                        PKCS12err(PKCS12_F_PKCS12_INIT,ERR_R_MALLOC_FAILURE);
                        goto err;
                }
                break;
                default:
                        PKCS12err(PKCS12_F_PKCS12_INIT,
                                PKCS12_R_UNSUPPORTED_PKCS12_MODE);
                        goto err;
        }

        return pkcs12;
err:
        if (pkcs12 != NULL) PKCS12_free(pkcs12);
        return NULL;
}
#endif /* ! codereview */
```

```
**********************************************************
    6895 Wed Aug 13 19:53:02 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_key.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_key.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
#include <openssl/bn.h>

/* Uncomment out this line to get debugging info about key generation */
/*#define DEBUG_KEYGEN*/
#ifdef DEBUG_KEYGEN
#include <openssl/bio.h>
extern BIO *bio_err;
void h__dump (unsigned char *p, int len);
#endif

/* PKCS12 compatible key/IV generation */
#ifndef min
#define min(a,b) ((a) < (b) ? (a) : (b))
#endif

int PKCS12_key_gen_asc(const char *pass, int passlen, unsigned char *salt,
             int saltlen, int id, int iter, int n, unsigned char *out,
             const EVP_MD *md_type)
{
        int ret;
        unsigned char *unipass;
        int uniplen;

        if(!pass) {
                unipass = NULL;
                uniplen = 0;
        } else if (!OPENSSL_asc2uni(pass, passlen, &unipass, &uniplen)) {
                PKCS12err(PKCS12_F_PKCS12_KEY_GEN_ASC,ERR_R_MALLOC_FAILURE);
                return 0;
        }
        ret = PKCS12_key_gen_uni(unipass, uniplen, salt, saltlen,
                                                id, iter, n, out, md_type);
        if (ret <= 0)
            return 0;
        if(unipass) {
                OPENSSL_cleanse(unipass, uniplen);      /* Clear password from m
                OPENSSL_free(unipass);
        }
        return ret;
}

int PKCS12_key_gen_uni(unsigned char *pass, int passlen, unsigned char *salt,
             int saltlen, int id, int iter, int n, unsigned char *out,
             const EVP_MD *md_type)
{
        unsigned char *B, *D, *I, *p, *Ai;
        int Slen, Plen, Ilen, Ijlen;
        int i, j, u, v;
        int ret = 0;
        BIGNUM *Ij, *Bpl1;      /* These hold Ij and B + 1 */
        EVP_MD_CTX ctx;
#ifdef  DEBUG_KEYGEN
        unsigned char *tmpout = out;
        int tmpn = n;
#endif

#if 0
        if (!pass) {
                PKCS12err(PKCS12_F_PKCS12_KEY_GEN_UNI,ERR_R_PASSED_NULL_PARAMETE
                return 0;
        }
#endif

        EVP_MD_CTX_init(&ctx);
#ifdef  DEBUG_KEYGEN
        fprintf(stderr, "KEYGEN DEBUG\n");
```

```
128         fprintf(stderr, "ID %d, ITER %d\n", id, iter);
129         fprintf(stderr, "Password (length %d):\n", passlen);
130         h__dump(pass, passlen);
131         fprintf(stderr, "Salt (length %d):\n", saltlen);
132         h__dump(salt, saltlen);
133 #endif
134         v = EVP_MD_block_size (md_type);
135         u = EVP_MD_size (md_type);
136         if (u < 0)
137             return 0;
138         D = OPENSSL_malloc (v);
139         Ai = OPENSSL_malloc (u);
140         B = OPENSSL_malloc (v + 1);
141         Slen = v * ((saltlen+v-1)/v);
142         if(passlen) Plen = v * ((passlen+v-1)/v);
143         else Plen = 0;
144         Ilen = Slen + Plen;
145         I = OPENSSL_malloc (Ilen);
146         Ij = BN_new();
147         Bpl1 = BN_new();
148         if (!D || !Ai || !B || !I || !Ij || !Bpl1)
149                 goto err;
150         for (i = 0; i < v; i++) D[i] = id;
151         p = I;
152         for (i = 0; i < Slen; i++) *p++ = salt[i % saltlen];
153         for (i = 0; i < Plen; i++) *p++ = pass[i % passlen];
154         for (;;) {
155                 if (!EVP_DigestInit_ex(&ctx, md_type, NULL)
156                         || !EVP_DigestUpdate(&ctx, D, v)
157                         || !EVP_DigestUpdate(&ctx, I, Ilen)
158                         || !EVP_DigestFinal_ex(&ctx, Ai, NULL))
159                         goto err;
160                 for (j = 1; j < iter; j++) {
161                         if (!EVP_DigestInit_ex(&ctx, md_type, NULL)
162                                 || !EVP_DigestUpdate(&ctx, Ai, u)
163                                 || !EVP_DigestFinal_ex(&ctx, Ai, NULL))
164                                 goto err;
165                 }
166                 memcpy (out, Ai, min (n, u));
167                 if (u >= n) {
168 #ifdef DEBUG_KEYGEN
169                         fprintf(stderr, "Output KEY (length %d)\n", tmpn);
170                         h__dump(tmpout, tmpn);
171 #endif
172                         ret = 1;
173                         goto end;
174                 }
175                 n -= u;
176                 out += u;
177                 for (j = 0; j < v; j++) B[j] = Ai[j % u];
178                 /* Work out B + 1 first then can use B as tmp space */
179                 if (!BN_bin2bn (B, v, Bpl1))
180                         goto err;
181                 if (!BN_add_word (Bpl1, 1))
182                         goto err;
183                 for (j = 0; j < Ilen ; j+=v) {
184                         if (!BN_bin2bn(I + j, v, Ij))
185                                 goto err;
186                         if (!BN_add(Ij, Ij, Bpl1))
187                                 goto err;
188                         if (!BN_bn2bin(Ij, B))
189                                 goto err;
190                         Ijlen = BN_num_bytes (Ij);
191                         /* If more than 2^(v*8) - 1 cut off MSB */
192                         if (Ijlen > v) {
193                                 if (!BN_bn2bin (Ij, B))
```



```
194                                         goto err;
195                                 memcpy (I + j, B + 1, v);
196 #ifndef PKCS12_BROKEN_KEYGEN
197                         /* If less than v bytes pad with zeroes */
198                         } else if (Ijlen < v) {
199                                 memset(I + j, 0, v - Ijlen);
200                                 if (!BN_bn2bin(Ij, I + j + v - Ijlen))
201                                         goto err;
202 #endif
203                         } else if (!BN_bn2bin (Ij, I + j))
204                                 goto err;
205                 }
206         }

208 err:
209         PKCS12err(PKCS12_F_PKCS12_KEY_GEN_UNI,ERR_R_MALLOC_FAILURE);

211 end:
212         OPENSSL_free (Ai);
213         OPENSSL_free (B);
214         OPENSSL_free (D);
215         OPENSSL_free (I);
216         BN_free (Ij);
217         BN_free (Bpl1);
218         EVP_MD_CTX_cleanup(&ctx);
219         return ret;
220 }
221 #ifdef DEBUG_KEYGEN
222 void h__dump (unsigned char *p, int len)
223 {
224         for (; len --; p++) fprintf(stderr, "%02X", *p);
225         fprintf(stderr, "\n");
226 }
227 #endif
228 #endif /* ! codereview */
```

```
**********************************************************
    8193 Wed Aug 13 19:53:02 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_kiss.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_kiss.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
/* Simplified PKCS#12 routines */

static int parse_pk12( PKCS12 *p12, const char *pass, int passlen,
                EVP_PKEY **pkey, STACK_OF(X509) *ocerts);

static int parse_bags( STACK_OF(PKCS12_SAFEBAG) *bags, const char *pass,
                       int passlen, EVP_PKEY **pkey, STACK_OF(X509) *ocerts);

static int parse_bag( PKCS12_SAFEBAG *bag, const char *pass, int passlen,
                        EVP_PKEY **pkey, STACK_OF(X509) *ocerts);

/* Parse and decrypt a PKCS#12 structure returning user key, user cert
 * and other (CA) certs. Note either ca should be NULL, *ca should be NULL,
 * or it should point to a valid STACK structure. pkey and cert can be
 * passed unitialised.
 */

int PKCS12_parse(PKCS12 *p12, const char *pass, EVP_PKEY **pkey, X509 **cert,
             STACK_OF(X509) **ca)
{
        STACK_OF(X509) *ocerts = NULL;
        X509 *x = NULL;
        /* Check for NULL PKCS12 structure */

        if(!p12)
                {
                PKCS12err(PKCS12_F_PKCS12_PARSE,PKCS12_R_INVALID_NULL_PKCS12_POI
                return 0;
                }

        if(pkey)
                *pkey = NULL;
        if(cert)
                *cert = NULL;

        /* Check the mac */

        /* If password is zero length or NULL then try verifying both cases
         * to determine which password is correct. The reason for this is that
         * under PKCS#12 password based encryption no password and a zero length
         * password are two different things...
         */

        if(!pass || !*pass) {
                if(PKCS12_verify_mac(p12, NULL, 0)) pass = NULL;
                else if(PKCS12_verify_mac(p12, "", 0)) pass = "";
                else {
                        PKCS12err(PKCS12_F_PKCS12_PARSE,PKCS12_R_MAC_VERIFY_FAIL
                        goto err;
                }
        } else if (!PKCS12_verify_mac(p12, pass, -1)) {
                PKCS12err(PKCS12_F_PKCS12_PARSE,PKCS12_R_MAC_VERIFY_FAILURE);
                goto err;
        }

        /* Allocate stack for other certificates */
        ocerts = sk_X509_new_null();

        if (!ocerts)
                {
                PKCS12err(PKCS12_F_PKCS12_PARSE,ERR_R_MALLOC_FAILURE);
                return 0;
                }

        if (!parse_pk12 (p12, pass, -1, pkey, ocerts))
```

```
128                 {
129                 PKCS12err(PKCS12_F_PKCS12_PARSE,PKCS12_R_PARSE_ERROR);
130                 goto err;
131                 }

133         while ((x = sk_X509_pop(ocerts)))
134                 {
135                 if (pkey && *pkey && cert && !*cert)
136                         {
137                         if (X509_check_private_key(x, *pkey))
138                                 {
139                                 *cert = x;
140                                 x = NULL;
141                                 }
142                         }

144                 if (ca && x)
145                         {
146                         if (!*ca)
147                                 *ca = sk_X509_new_null();
148                         if (!*ca)
149                                 goto err;
150                         if (!sk_X509_push(*ca, x))
151                                 goto err;
152                         x = NULL;
153                         }
154                 if (x)
155                         X509_free(x);
156                 }

158         if (ocerts)
159                 sk_X509_pop_free(ocerts, X509_free);

161         return 1;

163  err:

165         if (pkey && *pkey)
166                 EVP_PKEY_free(*pkey);
167         if (cert && *cert)
168                 X509_free(*cert);
169         if (x)
170                 X509_free(x);
171         if (ocerts)
172                 sk_X509_pop_free(ocerts, X509_free);
173         return 0;

175 }

177 /* Parse the outer PKCS#12 structure */

179 static int parse_pk12(PKCS12 *p12, const char *pass, int passlen,
180                 EVP_PKEY **pkey, STACK_OF(X509) *ocerts)
181 {
182         STACK_OF(PKCS7) *asafes;
183         STACK_OF(PKCS12_SAFEBAG) *bags;
184         int i, bagnid;
185         PKCS7 *p7;

187         if (!(asafes = PKCS12_unpack_authsafes (p12))) return 0;
188         for (i = 0; i < sk_PKCS7_num (asafes); i++) {
189                 p7 = sk_PKCS7_value (asafes, i);
190                 bagnid = OBJ_obj2nid (p7->type);
191                 if (bagnid == NID_pkcs7_data) {
192                         bags = PKCS12_unpack_p7data(p7);
193                 } else if (bagnid == NID_pkcs7_encrypted) {
```



```
194                         bags = PKCS12_unpack_p7encdata(p7, pass, passlen);
195                 } else continue;
196                 if (!bags) {
197                         sk_PKCS7_pop_free(asafes, PKCS7_free);
198                         return 0;
199                 }
200                 if (!parse_bags(bags, pass, passlen, pkey, ocerts)) {
201                         sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
202                         sk_PKCS7_pop_free(asafes, PKCS7_free);
203                         return 0;
204                 }
205                 sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
206         }
207         sk_PKCS7_pop_free(asafes, PKCS7_free);
208         return 1;
209 }


212 static int parse_bags(STACK_OF(PKCS12_SAFEBAG) *bags, const char *pass,
213                       int passlen, EVP_PKEY **pkey, STACK_OF(X509) *ocerts)
214 {
215         int i;
216         for (i = 0; i < sk_PKCS12_SAFEBAG_num(bags); i++) {
217                 if (!parse_bag(sk_PKCS12_SAFEBAG_value (bags, i),
218                                pass, passlen, pkey, ocerts))
219                         return 0;
220         }
221         return 1;
222 }

224 static int parse_bag(PKCS12_SAFEBAG *bag, const char *pass, int passlen,
225                      EVP_PKEY **pkey, STACK_OF(X509) *ocerts)
226 {
227         PKCS8_PRIV_KEY_INFO *p8;
228         X509 *x509;
229         ASN1_TYPE *attrib;
230         ASN1_BMPSTRING *fname = NULL;
231         ASN1_OCTET_STRING *lkid = NULL;

233         if ((attrib = PKCS12_get_attr (bag, NID_friendlyName)))
234                 fname = attrib->value.bmpstring;

236         if ((attrib = PKCS12_get_attr (bag, NID_localKeyID)))
237                 lkid = attrib->value.octet_string;

239         switch (M_PKCS12_bag_type(bag))
240         {
241         case NID_keyBag:
242                 if (!pkey || *pkey)
243                         return 1;
244                 if (!(*pkey = EVP_PKCS82PKEY(bag->value.keybag)))
245                         return 0;
246         break;

248         case NID_pkcs8ShroudedKeyBag:
249                 if (!pkey || *pkey)
250                         return 1;
251                 if (!(p8 = PKCS12_decrypt_skey(bag, pass, passlen)))
252                                 return 0;
253                 *pkey = EVP_PKCS82PKEY(p8);
254                 PKCS8_PRIV_KEY_INFO_free(p8);
255                 if (!(*pkey)) return 0;
256         break;

258         case NID_certBag:
259                 if (M_PKCS12_cert_bag_type(bag) != NID_x509Certificate )
```

```
260                     return 1;
261             if (!(x509 = PKCS12_certbag2x509(bag)))
262                     return 0;
263             if(lkid && !X509_keyid_set1(x509, lkid->data, lkid->length))
264                     {
265                     X509_free(x509);
266                     return 0;
267                     }
268             if(fname) {
269                     int len, r;
270                     unsigned char *data;
271                     len = ASN1_STRING_to_UTF8(&data, fname);
272                     if(len >= 0) {
273                             r = X509_alias_set1(x509, data, len);
274                             OPENSSL_free(data);
275                             if (!r)
276                                     {
277                                     X509_free(x509);
278                                     return 0;
279                                     }
280                     }
281             }

283             if(!sk_X509_push(ocerts, x509))
284                     {
285                     X509_free(x509);
286                     return 0;
287                     }

289         break;

291         case NID_safeContentsBag:
292                 return parse_bags(bag->value.safes, pass, passlen,
293                                   pkey, ocerts);
294         break;

296         default:
297                 return 1;
298         break;
299         }
300         return 1;
301 }
302 #endif /* ! codereview */
```

```
**********************************************************
    6506 Wed Aug 13 19:53:02 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_mutl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_mutl.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef OPENSSL_NO_HMAC
#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/hmac.h>
#include <openssl/rand.h>
#include <openssl/pkcs12.h>

/* Generate a MAC */
int PKCS12_gen_mac(PKCS12 *p12, const char *pass, int passlen,
                   unsigned char *mac, unsigned int *maclen)
{
        const EVP_MD *md_type;
        HMAC_CTX hmac;
        unsigned char key[EVP_MAX_MD_SIZE], *salt;
        int saltlen, iter;
        int md_size;

        if (!PKCS7_type_is_data(p12->authsafes))
                {
                PKCS12err(PKCS12_F_PKCS12_GEN_MAC,PKCS12_R_CONTENT_TYPE_NOT_DATA
                return 0;
                }

        salt = p12->mac->salt->data;
        saltlen = p12->mac->salt->length;
        if (!p12->mac->iter) iter = 1;
        else iter = ASN1_INTEGER_get (p12->mac->iter);
        if(!(md_type =
                 EVP_get_digestbyobj (p12->mac->dinfo->algor->algorithm))) {
                PKCS12err(PKCS12_F_PKCS12_GEN_MAC,PKCS12_R_UNKNOWN_DIGEST_ALGORI
                return 0;
        }
        md_size = EVP_MD_size(md_type);
        if (md_size < 0)
            return 0;
        if(!PKCS12_key_gen (pass, passlen, salt, saltlen, PKCS12_MAC_ID, iter,
                                md_size, key, md_type)) {
                PKCS12err(PKCS12_F_PKCS12_GEN_MAC,PKCS12_R_KEY_GEN_ERROR);
                return 0;
        }
        HMAC_CTX_init(&hmac);
        if (!HMAC_Init_ex(&hmac, key, md_size, md_type, NULL)
                || !HMAC_Update(&hmac, p12->authsafes->d.data->data,
                                       p12->authsafes->d.data->length)
                || !HMAC_Final(&hmac, mac, maclen))
                {
                HMAC_CTX_cleanup(&hmac);
                return 0;
                }
        HMAC_CTX_cleanup(&hmac);
        return 1;
}

/* Verify the mac */
int PKCS12_verify_mac(PKCS12 *p12, const char *pass, int passlen)
{
        unsigned char mac[EVP_MAX_MD_SIZE];
        unsigned int maclen;
        if(p12->mac == NULL) {
                PKCS12err(PKCS12_F_PKCS12_VERIFY_MAC,PKCS12_R_MAC_ABSENT);
                return 0;
        }
        if (!PKCS12_gen_mac (p12, pass, passlen, mac, &maclen)) {
                PKCS12err(PKCS12_F_PKCS12_VERIFY_MAC,PKCS12_R_MAC_GENERATION_ERR
                return 0;
        }
        if ((maclen != (unsigned int)p12->mac->dinfo->digest->length)
        || memcmp (mac, p12->mac->dinfo->digest->data, maclen)) return 0;
        return 1;
```

```
128 }

130 /* Set a mac */

132 int PKCS12_set_mac(PKCS12 *p12, const char *pass, int passlen,
133              unsigned char *salt, int saltlen, int iter, const EVP_MD *md_type)
134 {
135         unsigned char mac[EVP_MAX_MD_SIZE];
136         unsigned int maclen;

138         if (!md_type) md_type = EVP_sha1();
139         if (PKCS12_setup_mac (p12, iter, salt, saltlen, md_type) ==
140                                         PKCS12_ERROR) {
141                 PKCS12err(PKCS12_F_PKCS12_SET_MAC,PKCS12_R_MAC_SETUP_ERROR);
142                 return 0;
143         }
144         if (!PKCS12_gen_mac (p12, pass, passlen, mac, &maclen)) {
145                 PKCS12err(PKCS12_F_PKCS12_SET_MAC,PKCS12_R_MAC_GENERATION_ERROR)
146                 return 0;
147         }
148         if (!(M_ASN1_OCTET_STRING_set (p12->mac->dinfo->digest, mac, maclen))) {
149                 PKCS12err(PKCS12_F_PKCS12_SET_MAC,PKCS12_R_MAC_STRING_SET_ERROR)
150                                                 return 0;
151         }
152         return 1;
153 }

155 /* Set up a mac structure */
156 int PKCS12_setup_mac(PKCS12 *p12, int iter, unsigned char *salt, int saltlen,
157              const EVP_MD *md_type)
158 {
159         if (!(p12->mac = PKCS12_MAC_DATA_new())) return PKCS12_ERROR;
160         if (iter > 1) {
161                 if(!(p12->mac->iter = M_ASN1_INTEGER_new())) {
162                         PKCS12err(PKCS12_F_PKCS12_SETUP_MAC, ERR_R_MALLOC_FAILUR
163                         return 0;
164                 }
165                 if (!ASN1_INTEGER_set(p12->mac->iter, iter)) {
166                         PKCS12err(PKCS12_F_PKCS12_SETUP_MAC, ERR_R_MALLOC_FAILUR
167                         return 0;
168                 }
169         }
170         if (!saltlen) saltlen = PKCS12_SALT_LEN;
171         p12->mac->salt->length = saltlen;
172         if (!(p12->mac->salt->data = OPENSSL_malloc (saltlen))) {
173                 PKCS12err(PKCS12_F_PKCS12_SETUP_MAC, ERR_R_MALLOC_FAILURE);
174                 return 0;
175         }
176         if (!salt) {
177                 if (RAND_pseudo_bytes (p12->mac->salt->data, saltlen) < 0)
178                         return 0;
179         }
180         else memcpy (p12->mac->salt->data, salt, saltlen);
181         p12->mac->dinfo->algor->algorithm = OBJ_nid2obj(EVP_MD_type(md_type));
182         if (!(p12->mac->dinfo->algor->parameter = ASN1_TYPE_new())) {
183                 PKCS12err(PKCS12_F_PKCS12_SETUP_MAC, ERR_R_MALLOC_FAILURE);
184                 return 0;
185         }
186         p12->mac->dinfo->algor->parameter->type = V_ASN1_NULL;

188         return 1;
189 }
190 #endif
191 #endif /* ! codereview */
```

```
**********************************************************
    7465 Wed Aug 13 19:53:02 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_npas.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_npas.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```



```
#include <openssl/pem.h>
#include <openssl/err.h>
#include <openssl/pkcs12.h>

/* PKCS#12 password change routine */

static int newpass_p12(PKCS12 *p12, char *oldpass, char *newpass);
static int newpass_bags(STACK_OF(PKCS12_SAFEBAG) *bags, char *oldpass,
                        char *newpass);
static int newpass_bag(PKCS12_SAFEBAG *bag, char *oldpass, char *newpass);
static int alg_get(X509_ALGOR *alg, int *pnid, int *piter, int *psaltlen);

/*
 * Change the password on a PKCS#12 structure.
 */

int PKCS12_newpass(PKCS12 *p12, char *oldpass, char *newpass)
{
        /* Check for NULL PKCS12 structure */

        if(!p12) {
                PKCS12err(PKCS12_F_PKCS12_NEWPASS,PKCS12_R_INVALID_NULL_PKCS12_P
                return 0;
        }

        /* Check the mac */

        if (!PKCS12_verify_mac(p12, oldpass, -1)) {
                PKCS12err(PKCS12_F_PKCS12_NEWPASS,PKCS12_R_MAC_VERIFY_FAILURE);
                return 0;
        }

        if (!newpass_p12(p12, oldpass, newpass)) {
                PKCS12err(PKCS12_F_PKCS12_NEWPASS,PKCS12_R_PARSE_ERROR);
                return 0;
        }

        return 1;
}

/* Parse the outer PKCS#12 structure */

static int newpass_p12(PKCS12 *p12, char *oldpass, char *newpass)
{
        STACK_OF(PKCS7) *asafes, *newsafes;
        STACK_OF(PKCS12_SAFEBAG) *bags;
        int i, bagnid, pbe_nid = 0, pbe_iter = 0, pbe_saltlen = 0;
        PKCS7 *p7, *p7new;
        ASN1_OCTET_STRING *p12_data_tmp = NULL, *macnew = NULL;
        unsigned char mac[EVP_MAX_MD_SIZE];
        unsigned int maclen;

        if (!(asafes = PKCS12_unpack_authsafes(p12))) return 0;
        if(!(newsafes = sk_PKCS7_new_null())) return 0;
        for (i = 0; i < sk_PKCS7_num (asafes); i++) {
                p7 = sk_PKCS7_value(asafes, i);
                bagnid = OBJ_obj2nid(p7->type);
                if (bagnid == NID_pkcs7_data) {
                        bags = PKCS12_unpack_p7data(p7);
                } else if (bagnid == NID_pkcs7_encrypted) {
                        bags = PKCS12_unpack_p7encdata(p7, oldpass, -1);
                        if (!alg_get(p7->d.encrypted->enc_data->algorithm,
                                &pbe_nid, &pbe_iter, &pbe_saltlen))
                                {
                                sk_PKCS12_SAFEBAG_pop_free(bags,
                                                PKCS12_SAFEBAG_free);
```

```
128                                 bags = NULL;
129                                 }
130                 } else continue;
131                 if (!bags) {
132                         sk_PKCS7_pop_free(asafes, PKCS7_free);
133                         return 0;
134                 }
135                 if (!newpass_bags(bags, oldpass, newpass)) {
136                         sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
137                         sk_PKCS7_pop_free(asafes, PKCS7_free);
138                         return 0;
139                 }
140                 /* Repack bag in same form with new password */
141                 if (bagnid == NID_pkcs7_data) p7new = PKCS12_pack_p7data(bags);
142                 else p7new = PKCS12_pack_p7encdata(pbe_nid, newpass, -1, NULL,
143                                                    pbe_saltlen, pbe_iter, bags);
144                 sk_PKCS12_SAFEBAG_pop_free(bags, PKCS12_SAFEBAG_free);
145                 if(!p7new) {
146                         sk_PKCS7_pop_free(asafes, PKCS7_free);
147                         return 0;
148                 }
149                 sk_PKCS7_push(newsafes, p7new);
150         }
151         sk_PKCS7_pop_free(asafes, PKCS7_free);

153         /* Repack safe: save old safe in case of error */

155         p12_data_tmp = p12->authsafes->d.data;
156         if(!(p12->authsafes->d.data = ASN1_OCTET_STRING_new())) goto saferr;
157         if(!PKCS12_pack_authsafes(p12, newsafes)) goto saferr;

159         if(!PKCS12_gen_mac(p12, newpass, -1, mac, &maclen)) goto saferr;
160         if(!(macnew = ASN1_OCTET_STRING_new())) goto saferr;
161         if(!ASN1_OCTET_STRING_set(macnew, mac, maclen)) goto saferr;
162         ASN1_OCTET_STRING_free(p12->mac->dinfo->digest);
163         p12->mac->dinfo->digest = macnew;
164         ASN1_OCTET_STRING_free(p12_data_tmp);

166         return 1;

168         saferr:
169         /* Restore old safe */
170         ASN1_OCTET_STRING_free(p12->authsafes->d.data);
171         ASN1_OCTET_STRING_free(macnew);
172         p12->authsafes->d.data = p12_data_tmp;
173         return 0;

175 }


178 static int newpass_bags(STACK_OF(PKCS12_SAFEBAG) *bags, char *oldpass,
179                         char *newpass)
180 {
181         int i;
182         for (i = 0; i < sk_PKCS12_SAFEBAG_num(bags); i++) {
183                 if (!newpass_bag(sk_PKCS12_SAFEBAG_value(bags, i),
184                                  oldpass, newpass))
185                     return 0;
186         }
187         return 1;
188 }

190 /* Change password of safebag: only needs handle shrouded keybags */

192 static int newpass_bag(PKCS12_SAFEBAG *bag, char *oldpass, char *newpass)
193 {
```



```
194         PKCS8_PRIV_KEY_INFO *p8;
195         X509_SIG *p8new;
196         int p8_nid, p8_saltlen, p8_iter;

198         if(M_PKCS12_bag_type(bag) != NID_pkcs8ShroudedKeyBag) return 1;

200         if (!(p8 = PKCS8_decrypt(bag->value.shkeybag, oldpass, -1))) return 0;
201         if (!alg_get(bag->value.shkeybag->algor, &p8_nid, &p8_iter,
202                                                  &p8_saltlen))
203                 return 0;
204         if(!(p8new = PKCS8_encrypt(p8_nid, NULL, newpass, -1, NULL, p8_saltlen,
205                                                         p8_iter, p8))) return 0;
206         X509_SIG_free(bag->value.shkeybag);
207         bag->value.shkeybag = p8new;
208         return 1;
209 }

211 static int alg_get(X509_ALGOR *alg, int *pnid, int *piter, int *psaltlen)
212 {
213         PBEPARAM *pbe;
214         const unsigned char *p;

216         p = alg->parameter->value.sequence->data;
217         pbe = d2i_PBEPARAM(NULL, &p, alg->parameter->value.sequence->length);
218         if (!pbe)
219                 return 0;
220         *pnid = OBJ_obj2nid(alg->algorithm);
221         *piter = ASN1_INTEGER_get(pbe->iter);
222         *psaltlen = pbe->salt->length;
223         PBEPARAM_free(pbe);
224         return 1;
225 }
226 #endif /* ! codereview */
```

```
**********************************************************
    3003 Wed Aug 13 19:53:02 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_p8d.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_p8d.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
PKCS8_PRIV_KEY_INFO *PKCS8_decrypt(X509_SIG *p8, const char *pass, int passlen)
{
        return PKCS12_item_decrypt_d2i(p8->algor, ASN1_ITEM_rptr(PKCS8_PRIV_KEY_
                                       passlen, p8->digest, 1);
}
#endif /* ! codereview */
```

```
**********************************************************
    3687 Wed Aug 13 19:53:03 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_p8e.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_p8e.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
X509_SIG *PKCS8_encrypt(int pbe_nid, const EVP_CIPHER *cipher,
                        const char *pass, int passlen,
                        unsigned char *salt, int saltlen, int iter,
                                             PKCS8_PRIV_KEY_INFO *p8inf)
{
        X509_SIG *p8 = NULL;
        X509_ALGOR *pbe;

        if (!(p8 = X509_SIG_new())) {
                PKCS12err(PKCS12_F_PKCS8_ENCRYPT, ERR_R_MALLOC_FAILURE);
                goto err;
        }

        if(pbe_nid == -1) pbe = PKCS5_pbe2_set(cipher, iter, salt, saltlen);
        else pbe = PKCS5_pbe_set(pbe_nid, iter, salt, saltlen);
        if(!pbe) {
                PKCS12err(PKCS12_F_PKCS8_ENCRYPT, ERR_R_ASN1_LIB);
                goto err;
        }
        X509_ALGOR_free(p8->algor);
        p8->algor = pbe;
        M_ASN1_OCTET_STRING_free(p8->digest);
        p8->digest = PKCS12_item_i2d_encrypt(pbe, ASN1_ITEM_rptr(PKCS8_PRIV_KEY_
                                             pass, passlen, p8inf, 1);
        if(!p8->digest) {
                PKCS12err(PKCS12_F_PKCS8_ENCRYPT, PKCS12_R_ENCRYPT_ERROR);
                goto err;
        }

        return p8;

        err:
        X509_SIG_free(p8);
        return NULL;
}
#endif /* ! codereview */
```

```
**********************************************************
    5015 Wed Aug 13 19:53:03 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/p12_utl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* p12_utl.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/pkcs12.h>
```



```
/* Cheap and nasty Unicode stuff */

unsigned char *OPENSSL_asc2uni(const char *asc, int asclen, unsigned char **uni,
{
        int ulen, i;
        unsigned char *unitmp;
        if (asclen == -1) asclen = strlen(asc);
        ulen = asclen*2  + 2;
        if (!(unitmp = OPENSSL_malloc(ulen))) return NULL;
        for (i = 0; i < ulen - 2; i+=2) {
                unitmp[i] = 0;
                unitmp[i + 1] = asc[i>>1];
        }
        /* Make result double null terminated */
        unitmp[ulen - 2] = 0;
        unitmp[ulen - 1] = 0;
        if (unilen) *unilen = ulen;
        if (uni) *uni = unitmp;
        return unitmp;
}

char *OPENSSL_uni2asc(unsigned char *uni, int unilen)
{
        int asclen, i;
        char *asctmp;
        asclen = unilen / 2;
        /* If no terminating zero allow for one */
        if (!unilen || uni[unilen - 1]) asclen++;
        uni++;
        if (!(asctmp = OPENSSL_malloc(asclen))) return NULL;
        for (i = 0; i < unilen; i+=2) asctmp[i>>1] = uni[i];
        asctmp[asclen - 1] = 0;
        return asctmp;
}

int i2d_PKCS12_bio(BIO *bp, PKCS12 *p12)
{
        return ASN1_item_i2d_bio(ASN1_ITEM_rptr(PKCS12), bp, p12);
}

#ifndef OPENSSL_NO_FP_API
int i2d_PKCS12_fp(FILE *fp, PKCS12 *p12)
{
        return ASN1_item_i2d_fp(ASN1_ITEM_rptr(PKCS12), fp, p12);
}
#endif

PKCS12 *d2i_PKCS12_bio(BIO *bp, PKCS12 **p12)
{
        return ASN1_item_d2i_bio(ASN1_ITEM_rptr(PKCS12), bp, p12);
}
#ifndef OPENSSL_NO_FP_API
PKCS12 *d2i_PKCS12_fp(FILE *fp, PKCS12 **p12)
{
        return ASN1_item_d2i_fp(ASN1_ITEM_rptr(PKCS12), fp, p12);
}
#endif

PKCS12_SAFEBAG *PKCS12_x5092certbag(X509 *x509)
{
        return PKCS12_item_pack_safebag(x509, ASN1_ITEM_rptr(X509),
                        NID_x509Certificate, NID_certBag);
}

PKCS12_SAFEBAG *PKCS12_x509crl2certbag(X509_CRL *crl)
```

```
128 {
129         return PKCS12_item_pack_safebag(crl, ASN1_ITEM_rptr(X509_CRL),
130                         NID_x509Crl, NID_crlBag);
131 }

133 X509 *PKCS12_certbag2x509(PKCS12_SAFEBAG *bag)
134 {
135         if(M_PKCS12_bag_type(bag) != NID_certBag) return NULL;
136         if(M_PKCS12_cert_bag_type(bag) != NID_x509Certificate) return NULL;
137         return ASN1_item_unpack(bag->value.bag->value.octet, ASN1_ITEM_rptr(X509
138 }

140 X509_CRL *PKCS12_certbag2x509crl(PKCS12_SAFEBAG *bag)
141 {
142         if(M_PKCS12_bag_type(bag) != NID_crlBag) return NULL;
143         if(M_PKCS12_cert_bag_type(bag) != NID_x509Crl) return NULL;
144         return ASN1_item_unpack(bag->value.bag->value.octet,
145                                                         ASN1_ITEM_rptr(X509_CRL)
146 }
147 #endif /* ! codereview */
```

```
**********************************************************
    6782 Wed Aug 13 19:53:03 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs12/pk12err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pkcs12/pk12err.c */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/pkcs12.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_PKCS12,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_PKCS12,0,reason)

static ERR_STRING_DATA PKCS12_str_functs[]=
        {
{ERR_FUNC(PKCS12_F_PARSE_BAG),  "PARSE_BAG"},
{ERR_FUNC(PKCS12_F_PARSE_BAGS), "PARSE_BAGS"},
{ERR_FUNC(PKCS12_F_PKCS12_ADD_FRIENDLYNAME),    "PKCS12_ADD_FRIENDLYNAME"},
{ERR_FUNC(PKCS12_F_PKCS12_ADD_FRIENDLYNAME_ASC),        "PKCS12_add_friendlyname
{ERR_FUNC(PKCS12_F_PKCS12_ADD_FRIENDLYNAME_UNI),        "PKCS12_add_friendlyname
{ERR_FUNC(PKCS12_F_PKCS12_ADD_LOCALKEYID),      "PKCS12_add_localkeyid"},
{ERR_FUNC(PKCS12_F_PKCS12_CREATE),      "PKCS12_create"},
{ERR_FUNC(PKCS12_F_PKCS12_GEN_MAC),     "PKCS12_gen_mac"},
{ERR_FUNC(PKCS12_F_PKCS12_INIT),        "PKCS12_init"},
{ERR_FUNC(PKCS12_F_PKCS12_ITEM_DECRYPT_D2I),    "PKCS12_item_decrypt_d2i"},
{ERR_FUNC(PKCS12_F_PKCS12_ITEM_I2D_ENCRYPT),    "PKCS12_item_i2d_encrypt"},
{ERR_FUNC(PKCS12_F_PKCS12_ITEM_PACK_SAFEBAG),   "PKCS12_item_pack_safebag"},
{ERR_FUNC(PKCS12_F_PKCS12_KEY_GEN_ASC), "PKCS12_key_gen_asc"},
{ERR_FUNC(PKCS12_F_PKCS12_KEY_GEN_UNI), "PKCS12_key_gen_uni"},
{ERR_FUNC(PKCS12_F_PKCS12_MAKE_KEYBAG), "PKCS12_MAKE_KEYBAG"},
{ERR_FUNC(PKCS12_F_PKCS12_MAKE_SHKEYBAG),       "PKCS12_MAKE_SHKEYBAG"},
{ERR_FUNC(PKCS12_F_PKCS12_NEWPASS),     "PKCS12_newpass"},
{ERR_FUNC(PKCS12_F_PKCS12_PACK_P7DATA), "PKCS12_pack_p7data"},
{ERR_FUNC(PKCS12_F_PKCS12_PACK_P7ENCDATA),      "PKCS12_pack_p7encdata"},
{ERR_FUNC(PKCS12_F_PKCS12_PARSE),       "PKCS12_parse"},
{ERR_FUNC(PKCS12_F_PKCS12_PBE_CRYPT),   "PKCS12_pbe_crypt"},
{ERR_FUNC(PKCS12_F_PKCS12_PBE_KEYIVGEN),        "PKCS12_PBE_keyivgen"},
{ERR_FUNC(PKCS12_F_PKCS12_SETUP_MAC),   "PKCS12_setup_mac"},
{ERR_FUNC(PKCS12_F_PKCS12_SET_MAC),     "PKCS12_set_mac"},
{ERR_FUNC(PKCS12_F_PKCS12_UNPACK_AUTHSAFES),    "PKCS12_unpack_authsafes"},
{ERR_FUNC(PKCS12_F_PKCS12_UNPACK_P7DATA),       "PKCS12_unpack_p7data"},
{ERR_FUNC(PKCS12_F_PKCS12_VERIFY_MAC),  "PKCS12_verify_mac"},
{ERR_FUNC(PKCS12_F_PKCS8_ADD_KEYUSAGE), "PKCS8_add_keyusage"},
{ERR_FUNC(PKCS12_F_PKCS8_ENCRYPT),      "PKCS8_encrypt"},
{0,NULL}
        };

static ERR_STRING_DATA PKCS12_str_reasons[]=
        {
{ERR_REASON(PKCS12_R_CANT_PACK_STRUCTURE),"cant pack structure"},
{ERR_REASON(PKCS12_R_CONTENT_TYPE_NOT_DATA),"content type not data"},
{ERR_REASON(PKCS12_R_DECODE_ERROR)       ,"decode error"},
{ERR_REASON(PKCS12_R_ENCODE_ERROR)       ,"encode error"},
{ERR_REASON(PKCS12_R_ENCRYPT_ERROR)      ,"encrypt error"},
{ERR_REASON(PKCS12_R_ERROR_SETTING_ENCRYPTED_DATA_TYPE),"error setting encrypted
{ERR_REASON(PKCS12_R_INVALID_NULL_ARGUMENT),"invalid null argument"},
{ERR_REASON(PKCS12_R_INVALID_NULL_PKCS12_POINTER),"invalid null pkcs12 pointer"}
{ERR_REASON(PKCS12_R_IV_GEN_ERROR)       ,"iv gen error"},
{ERR_REASON(PKCS12_R_KEY_GEN_ERROR)      ,"key gen error"},
{ERR_REASON(PKCS12_R_MAC_ABSENT)         ,"mac absent"},
{ERR_REASON(PKCS12_R_MAC_GENERATION_ERROR),"mac generation error"},
{ERR_REASON(PKCS12_R_MAC_SETUP_ERROR)    ,"mac setup error"},
{ERR_REASON(PKCS12_R_MAC_STRING_SET_ERROR),"mac string set error"},
{ERR_REASON(PKCS12_R_MAC_VERIFY_ERROR)   ,"mac verify error"},
{ERR_REASON(PKCS12_R_MAC_VERIFY_FAILURE) ,"mac verify failure"},
{ERR_REASON(PKCS12_R_PARSE_ERROR)        ,"parse error"},
{ERR_REASON(PKCS12_R_PKCS12_ALGOR_CIPHERINIT_ERROR),"pkcs12 algor cipherinit err
{ERR_REASON(PKCS12_R_PKCS12_CIPHERFINAL_ERROR),"pkcs12 cipherfinal error"},
{ERR_REASON(PKCS12_R_PKCS12_PBE_CRYPT_ERROR),"pkcs12 pbe crypt error"},
{ERR_REASON(PKCS12_R_UNKNOWN_DIGEST_ALGORITHM),"unknown digest algorithm"},
```

```
128 {ERR_REASON(PKCS12_R_UNSUPPORTED_PKCS12_MODE),"unsupported pkcs12 mode"},
129 {0,NULL}
130         };

132 #endif

134 void ERR_load_PKCS12_strings(void)
135         {
136 #ifndef OPENSSL_NO_ERR

138         if (ERR_func_error_string(PKCS12_str_functs[0].error) == NULL)
139                 {
140                 ERR_load_strings(0,PKCS12_str_functs);
141                 ERR_load_strings(0,PKCS12_str_reasons);
142                 }
143 #endif
144         }
145 #endif /* ! codereview */
```

```
**********************************************************
    2911 Wed Aug 13 19:53:03 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/bio_pk7.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* bio_pk7.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/asn1.h>
#include <openssl/pkcs7.h>
#include <openssl/bio.h>

#if !defined(OPENSSL_SYSNAME_NETWARE) && !defined(OPENSSL_SYSNAME_VXWORKS)
#include <memory.h>
#endif
```



```
#include <stdio.h>

/* Streaming encode support for PKCS#7 */

BIO *BIO_new_PKCS7(BIO *out, PKCS7 *p7)
        {
        return BIO_new_NDEF(out, (ASN1_VALUE *)p7, ASN1_ITEM_rptr(PKCS7));
        }
#endif /* ! codereview */
```

```
**********************************************************
    9350 Wed Aug 13 19:53:03 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/pk7_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pk7_asn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/pkcs7.h>
#include <openssl/x509.h>

/* PKCS#7 ASN1 module */

/* This is the ANY DEFINED BY table for the top level PKCS#7 structure */

ASN1_ADB_TEMPLATE(p7default) = ASN1_EXP_OPT(PKCS7, d.other, ASN1_ANY, 0);

ASN1_ADB(PKCS7) = {
        ADB_ENTRY(NID_pkcs7_data, ASN1_NDEF_EXP_OPT(PKCS7, d.data, ASN1_OCTET_ST
        ADB_ENTRY(NID_pkcs7_signed, ASN1_NDEF_EXP_OPT(PKCS7, d.sign, PKCS7_SIGNE
        ADB_ENTRY(NID_pkcs7_enveloped, ASN1_NDEF_EXP_OPT(PKCS7, d.enveloped, PKC
        ADB_ENTRY(NID_pkcs7_signedAndEnveloped, ASN1_NDEF_EXP_OPT(PKCS7, d.signe
        ADB_ENTRY(NID_pkcs7_digest, ASN1_NDEF_EXP_OPT(PKCS7, d.digest, PKCS7_DIG
        ADB_ENTRY(NID_pkcs7_encrypted, ASN1_NDEF_EXP_OPT(PKCS7, d.encrypted, PKC
} ASN1_ADB_END(PKCS7, 0, type, 0, &p7default_tt, NULL);

/* PKCS#7 streaming support */
static int pk7_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                        void *exarg)
{
        ASN1_STREAM_ARG *sarg = exarg;
        PKCS7 **pp7 = (PKCS7 **)pval;

        switch(operation)
                {

                case ASN1_OP_STREAM_PRE:
                if (PKCS7_stream(&sarg->boundary, *pp7) <= 0)
                        return 0;
                case ASN1_OP_DETACHED_PRE:
                sarg->ndef_bio = PKCS7_dataInit(*pp7, sarg->out);
                if (!sarg->ndef_bio)
                        return 0;
                break;

                case ASN1_OP_STREAM_POST:
                case ASN1_OP_DETACHED_POST:
                if (PKCS7_dataFinal(*pp7, sarg->ndef_bio) <= 0)
                        return 0;
                break;

                }
        return 1;
}

ASN1_NDEF_SEQUENCE_cb(PKCS7, pk7_cb) = {
        ASN1_SIMPLE(PKCS7, type, ASN1_OBJECT),
        ASN1_ADB_OBJECT(PKCS7)
}ASN1_NDEF_SEQUENCE_END_cb(PKCS7, PKCS7)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7)
IMPLEMENT_ASN1_NDEF_FUNCTION(PKCS7)
IMPLEMENT_ASN1_DUP_FUNCTION(PKCS7)

ASN1_NDEF_SEQUENCE(PKCS7_SIGNED) = {
        ASN1_SIMPLE(PKCS7_SIGNED, version, ASN1_INTEGER),
        ASN1_SET_OF(PKCS7_SIGNED, md_algs, X509_ALGOR),
        ASN1_SIMPLE(PKCS7_SIGNED, contents, PKCS7),
        ASN1_IMP_SEQUENCE_OF_OPT(PKCS7_SIGNED, cert, X509, 0),
        ASN1_IMP_SET_OF_OPT(PKCS7_SIGNED, crl, X509_CRL, 1),
        ASN1_SET_OF(PKCS7_SIGNED, signer_info, PKCS7_SIGNER_INFO)
} ASN1_NDEF_SEQUENCE_END(PKCS7_SIGNED)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_SIGNED)
```

```
/* Minor tweak to operation: free up EVP_PKEY */
static int si_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                        void *exarg)
{
        if(operation == ASN1_OP_FREE_POST) {
                PKCS7_SIGNER_INFO *si = (PKCS7_SIGNER_INFO *)*pval;
                EVP_PKEY_free(si->pkey);
        }
        return 1;
}

ASN1_SEQUENCE_cb(PKCS7_SIGNER_INFO, si_cb) = {
        ASN1_SIMPLE(PKCS7_SIGNER_INFO, version, ASN1_INTEGER),
        ASN1_SIMPLE(PKCS7_SIGNER_INFO, issuer_and_serial, PKCS7_ISSUER_AND_SERIA
        ASN1_SIMPLE(PKCS7_SIGNER_INFO, digest_alg, X509_ALGOR),
        /* NB this should be a SET OF but we use a SEQUENCE OF so the
         * original order * is retained when the structure is reencoded.
         * Since the attributes are implicitly tagged this will not affect
         * the encoding.
         */
        ASN1_IMP_SEQUENCE_OF_OPT(PKCS7_SIGNER_INFO, auth_attr, X509_ATTRIBUTE, 0
        ASN1_SIMPLE(PKCS7_SIGNER_INFO, digest_enc_alg, X509_ALGOR),
        ASN1_SIMPLE(PKCS7_SIGNER_INFO, enc_digest, ASN1_OCTET_STRING),
        ASN1_IMP_SET_OF_OPT(PKCS7_SIGNER_INFO, unauth_attr, X509_ATTRIBUTE, 1)
} ASN1_SEQUENCE_END_cb(PKCS7_SIGNER_INFO, PKCS7_SIGNER_INFO)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_SIGNER_INFO)

ASN1_SEQUENCE(PKCS7_ISSUER_AND_SERIAL) = {
        ASN1_SIMPLE(PKCS7_ISSUER_AND_SERIAL, issuer, X509_NAME),
        ASN1_SIMPLE(PKCS7_ISSUER_AND_SERIAL, serial, ASN1_INTEGER)
} ASN1_SEQUENCE_END(PKCS7_ISSUER_AND_SERIAL)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_ISSUER_AND_SERIAL)

ASN1_NDEF_SEQUENCE(PKCS7_ENVELOPE) = {
        ASN1_SIMPLE(PKCS7_ENVELOPE, version, ASN1_INTEGER),
        ASN1_SET_OF(PKCS7_ENVELOPE, recipientinfo, PKCS7_RECIP_INFO),
        ASN1_SIMPLE(PKCS7_ENVELOPE, enc_data, PKCS7_ENC_CONTENT)
} ASN1_NDEF_SEQUENCE_END(PKCS7_ENVELOPE)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_ENVELOPE)

/* Minor tweak to operation: free up X509 */
static int ri_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                            void *exarg)
{
        if(operation == ASN1_OP_FREE_POST) {
                PKCS7_RECIP_INFO *ri = (PKCS7_RECIP_INFO *)*pval;
                X509_free(ri->cert);
        }
        return 1;
}

ASN1_SEQUENCE_cb(PKCS7_RECIP_INFO, ri_cb) = {
        ASN1_SIMPLE(PKCS7_RECIP_INFO, version, ASN1_INTEGER),
        ASN1_SIMPLE(PKCS7_RECIP_INFO, issuer_and_serial, PKCS7_ISSUER_AND_SERIAL
        ASN1_SIMPLE(PKCS7_RECIP_INFO, key_enc_algor, X509_ALGOR),
        ASN1_SIMPLE(PKCS7_RECIP_INFO, enc_key, ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END_cb(PKCS7_RECIP_INFO, PKCS7_RECIP_INFO)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_RECIP_INFO)

ASN1_NDEF_SEQUENCE(PKCS7_ENC_CONTENT) = {
        ASN1_SIMPLE(PKCS7_ENC_CONTENT, content_type, ASN1_OBJECT),
```



```
        ASN1_SIMPLE(PKCS7_ENC_CONTENT, algorithm, X509_ALGOR),
        ASN1_IMP_OPT(PKCS7_ENC_CONTENT, enc_data, ASN1_OCTET_STRING_NDEF, 0)
} ASN1_NDEF_SEQUENCE_END(PKCS7_ENC_CONTENT)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_ENC_CONTENT)

ASN1_NDEF_SEQUENCE(PKCS7_SIGN_ENVELOPE) = {
        ASN1_SIMPLE(PKCS7_SIGN_ENVELOPE, version, ASN1_INTEGER),
        ASN1_SET_OF(PKCS7_SIGN_ENVELOPE, recipientinfo, PKCS7_RECIP_INFO),
        ASN1_SET_OF(PKCS7_SIGN_ENVELOPE, md_algs, X509_ALGOR),
        ASN1_SIMPLE(PKCS7_SIGN_ENVELOPE, enc_data, PKCS7_ENC_CONTENT),
        ASN1_IMP_SET_OF_OPT(PKCS7_SIGN_ENVELOPE, cert, X509, 0),
        ASN1_IMP_SET_OF_OPT(PKCS7_SIGN_ENVELOPE, crl, X509_CRL, 1),
        ASN1_SET_OF(PKCS7_SIGN_ENVELOPE, signer_info, PKCS7_SIGNER_INFO)
} ASN1_NDEF_SEQUENCE_END(PKCS7_SIGN_ENVELOPE)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_SIGN_ENVELOPE)

ASN1_NDEF_SEQUENCE(PKCS7_ENCRYPT) = {
        ASN1_SIMPLE(PKCS7_ENCRYPT, version, ASN1_INTEGER),
        ASN1_SIMPLE(PKCS7_ENCRYPT, enc_data, PKCS7_ENC_CONTENT)
} ASN1_NDEF_SEQUENCE_END(PKCS7_ENCRYPT)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_ENCRYPT)

ASN1_NDEF_SEQUENCE(PKCS7_DIGEST) = {
        ASN1_SIMPLE(PKCS7_DIGEST, version, ASN1_INTEGER),
        ASN1_SIMPLE(PKCS7_DIGEST, md, X509_ALGOR),
        ASN1_SIMPLE(PKCS7_DIGEST, contents, PKCS7),
        ASN1_SIMPLE(PKCS7_DIGEST, digest, ASN1_OCTET_STRING)
} ASN1_NDEF_SEQUENCE_END(PKCS7_DIGEST)

IMPLEMENT_ASN1_FUNCTIONS(PKCS7_DIGEST)

/* Specials for authenticated attributes */

/* When signing attributes we want to reorder them to match the sorted
 * encoding.
 */

ASN1_ITEM_TEMPLATE(PKCS7_ATTR_SIGN) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SET_ORDER, 0, PKCS7_ATTRIBUTES, X509_ATT
ASN1_ITEM_TEMPLATE_END(PKCS7_ATTR_SIGN)

/* When verifying attributes we need to use the received order. So
 * we use SEQUENCE OF and tag it to SET OF
 */

ASN1_ITEM_TEMPLATE(PKCS7_ATTR_VERIFY) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF | ASN1_TFLG_IMPTAG | ASN1_TF
                              V_ASN1_SET, PKCS7_ATTRIBUTES, X509_ATTRIBUTE)
ASN1_ITEM_TEMPLATE_END(PKCS7_ATTR_VERIFY)

IMPLEMENT_ASN1_PRINT_FUNCTION(PKCS7)
#endif /* ! codereview */
```

```
**********************************************************
    5596 Wed Aug 13 19:53:03 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/pk7_attr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pk7_attr.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <openssl/bio.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/pem.h>
#include <openssl/pkcs7.h>
#include <openssl/x509.h>
#include <openssl/err.h>

int PKCS7_add_attrib_smimecap(PKCS7_SIGNER_INFO *si, STACK_OF(X509_ALGOR) *cap)
{
        ASN1_STRING *seq;
        if(!(seq = ASN1_STRING_new())) {
                PKCS7err(PKCS7_F_PKCS7_ADD_ATTRIB_SMIMECAP,ERR_R_MALLOC_FAILURE)
                return 0;
        }
        seq->length = ASN1_item_i2d((ASN1_VALUE *)cap,&seq->data,
                                ASN1_ITEM_rptr(X509_ALGORS));
        return PKCS7_add_signed_attribute(si, NID_SMIMECapabilities,
                                                V_ASN1_SEQUENCE, seq);
}

STACK_OF(X509_ALGOR) *PKCS7_get_smimecap(PKCS7_SIGNER_INFO *si)
        {
        ASN1_TYPE *cap;
        const unsigned char *p;

        cap = PKCS7_get_signed_attribute(si, NID_SMIMECapabilities);
        if (!cap || (cap->type != V_ASN1_SEQUENCE))
                return NULL;
        p = cap->value.sequence->data;
        return (STACK_OF(X509_ALGOR) *)
                ASN1_item_d2i(NULL, &p, cap->value.sequence->length,
                                ASN1_ITEM_rptr(X509_ALGORS));
        }

/* Basic smime-capabilities OID and optional integer arg */
int PKCS7_simple_smimecap(STACK_OF(X509_ALGOR) *sk, int nid, int arg)
{
        X509_ALGOR *alg;

        if(!(alg = X509_ALGOR_new())) {
                PKCS7err(PKCS7_F_PKCS7_SIMPLE_SMIMECAP,ERR_R_MALLOC_FAILURE);
                return 0;
        }
        ASN1_OBJECT_free(alg->algorithm);
        alg->algorithm = OBJ_nid2obj (nid);
        if (arg > 0) {
                ASN1_INTEGER *nbit;
                if(!(alg->parameter = ASN1_TYPE_new())) {
                        PKCS7err(PKCS7_F_PKCS7_SIMPLE_SMIMECAP,ERR_R_MALLOC_FAIL
                        return 0;
                }
                if(!(nbit = ASN1_INTEGER_new())) {
                        PKCS7err(PKCS7_F_PKCS7_SIMPLE_SMIMECAP,ERR_R_MALLOC_FAIL
                        return 0;
                }
                if(!ASN1_INTEGER_set (nbit, arg)) {
                        PKCS7err(PKCS7_F_PKCS7_SIMPLE_SMIMECAP,ERR_R_MALLOC_FAIL
                        return 0;
                }
                alg->parameter->value.integer = nbit;
                alg->parameter->type = V_ASN1_INTEGER;
        }
        sk_X509_ALGOR_push (sk, alg);
        return 1;
}
```

```
128 int PKCS7_add_attrib_content_type(PKCS7_SIGNER_INFO *si, ASN1_OBJECT *coid)
129         {
130         if (PKCS7_get_signed_attribute(si, NID_pkcs9_contentType))
131                 return 0;
132         if (!coid)
133                 coid = OBJ_nid2obj(NID_pkcs7_data);
134         return PKCS7_add_signed_attribute(si, NID_pkcs9_contentType,
135                                 V_ASN1_OBJECT, coid);
136         }

138 int PKCS7_add0_attrib_signing_time(PKCS7_SIGNER_INFO *si, ASN1_TIME *t)
139         {
140         if (!t && !(t=X509_gmtime_adj(NULL,0)))
141                 {
142                 PKCS7err(PKCS7_F_PKCS7_ADD0_ATTRIB_SIGNING_TIME,
143                                 ERR_R_MALLOC_FAILURE);
144                 return 0;
145                 }
146         return PKCS7_add_signed_attribute(si, NID_pkcs9_signingTime,
147                                                 V_ASN1_UTCTIME, t);
148         }

150 int PKCS7_add1_attrib_digest(PKCS7_SIGNER_INFO *si,
151                                 const unsigned char *md, int mdlen)
152         {
153         ASN1_OCTET_STRING *os;
154         os = ASN1_OCTET_STRING_new();
155         if (!os)
156                 return 0;
157         if (!ASN1_STRING_set(os, md, mdlen)
158                 || !PKCS7_add_signed_attribute(si, NID_pkcs9_messageDigest,
159                                                 V_ASN1_OCTET_STRING, os))
160                 {
161                 ASN1_OCTET_STRING_free(os);
162                 return 0;
163                 }
164         return 1;
165         }
166 #endif /* ! codereview */
```

```
**********************************************************
   30846 Wed Aug 13 19:53:04 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/pk7_doit.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pkcs7/pk7_doit.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/rand.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/err.h>

static int add_attribute(STACK_OF(X509_ATTRIBUTE) **sk, int nid, int atrtype,
                         void *value);
static ASN1_TYPE *get_attribute(STACK_OF(X509_ATTRIBUTE) *sk, int nid);

static int PKCS7_type_is_other(PKCS7* p7)
        {
        int isOther=1;

        int nid=OBJ_obj2nid(p7->type);

        switch( nid )
                {
        case NID_pkcs7_data:
        case NID_pkcs7_signed:
        case NID_pkcs7_enveloped:
        case NID_pkcs7_signedAndEnveloped:
        case NID_pkcs7_digest:
        case NID_pkcs7_encrypted:
                isOther=0;
                break;
        default:
                isOther=1;
                }

        return isOther;

        }

static ASN1_OCTET_STRING *PKCS7_get_octet_string(PKCS7 *p7)
        {
        if ( PKCS7_type_is_data(p7))
                return p7->d.data;
        if ( PKCS7_type_is_other(p7) && p7->d.other
                && (p7->d.other->type == V_ASN1_OCTET_STRING))
                return p7->d.other->value.octet_string;
        return NULL;
        }

static int PKCS7_bio_add_digest(BIO **pbio, X509_ALGOR *alg)
        {
        BIO *btmp;
        const EVP_MD *md;
        if ((btmp=BIO_new(BIO_f_md())) == NULL)
                {
                PKCS7err(PKCS7_F_PKCS7_BIO_ADD_DIGEST,ERR_R_BIO_LIB);
                goto err;
                }

        md=EVP_get_digestbyobj(alg->algorithm);
        if (md == NULL)
                {
                PKCS7err(PKCS7_F_PKCS7_BIO_ADD_DIGEST,PKCS7_R_UNKNOWN_DIGEST_TYP
                goto err;
                }

        BIO_set_md(btmp,md);
        if (*pbio == NULL)
                *pbio=btmp;
        else if (!BIO_push(*pbio,btmp))
                {
                PKCS7err(PKCS7_F_PKCS7_BIO_ADD_DIGEST,ERR_R_BIO_LIB);
```

```
128                 goto err;
129                 }
130         btmp=NULL;

132         return 1;

134         err:
135         if (btmp)
136                 BIO_free(btmp);
137         return 0;

139         }

141 static int pkcs7_encode_rinfo(PKCS7_RECIP_INFO *ri,
142                                       unsigned char *key, int keylen)
143         {
144         EVP_PKEY_CTX *pctx = NULL;
145         EVP_PKEY *pkey = NULL;
146         unsigned char *ek = NULL;
147         int ret = 0;
148         size_t eklen;

150         pkey = X509_get_pubkey(ri->cert);

152         if (!pkey)
153                 return 0;

155         pctx = EVP_PKEY_CTX_new(pkey, NULL);
156         if (!pctx)
157                 return 0;

159         if (EVP_PKEY_encrypt_init(pctx) <= 0)
160                 goto err;

162         if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_ENCRYPT,
163                               EVP_PKEY_CTRL_PKCS7_ENCRYPT, 0, ri) <= 0)
164                 {
165                 PKCS7err(PKCS7_F_PKCS7_ENCODE_RINFO, PKCS7_R_CTRL_ERROR);
166                 goto err;
167                 }

169         if (EVP_PKEY_encrypt(pctx, NULL, &eklen, key, keylen) <= 0)
170                 goto err;

172         ek = OPENSSL_malloc(eklen);

174         if (ek == NULL)
175                 {
176                 PKCS7err(PKCS7_F_PKCS7_ENCODE_RINFO, ERR_R_MALLOC_FAILURE);
177                 goto err;
178                 }

180         if (EVP_PKEY_encrypt(pctx, ek, &eklen, key, keylen) <= 0)
181                 goto err;

183         ASN1_STRING_set0(ri->enc_key, ek, eklen);
184         ek = NULL;

186         ret = 1;

188         err:
189         if (pkey)
190                 EVP_PKEY_free(pkey);
191         if (pctx)
192                 EVP_PKEY_CTX_free(pctx);
193         if (ek)
```



```
194                 OPENSSL_free(ek);
195         return ret;

197         }


200 static int pkcs7_decrypt_rinfo(unsigned char **pek, int *peklen,
201                                 PKCS7_RECIP_INFO *ri, EVP_PKEY *pkey)
202         {
203         EVP_PKEY_CTX *pctx = NULL;
204         unsigned char *ek = NULL;
205         size_t eklen;

207         int ret = -1;

209         pctx = EVP_PKEY_CTX_new(pkey, NULL);
210         if (!pctx)
211                 return -1;

213         if (EVP_PKEY_decrypt_init(pctx) <= 0)
214                 goto err;

216         if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_DECRYPT,
217                               EVP_PKEY_CTRL_PKCS7_DECRYPT, 0, ri) <= 0)
218                 {
219                 PKCS7err(PKCS7_F_PKCS7_DECRYPT_RINFO, PKCS7_R_CTRL_ERROR);
220                 goto err;
221                 }

223         if (EVP_PKEY_decrypt(pctx, NULL, &eklen,
224                              ri->enc_key->data, ri->enc_key->length) <= 0)
225                 goto err;

227         ek = OPENSSL_malloc(eklen);

229         if (ek == NULL)
230                 {
231                 PKCS7err(PKCS7_F_PKCS7_DECRYPT_RINFO, ERR_R_MALLOC_FAILURE);
232                 goto err;
233                 }

235         if (EVP_PKEY_decrypt(pctx, ek, &eklen,
236                              ri->enc_key->data, ri->enc_key->length) <= 0)
237                 {
238                 ret = 0;
239                 PKCS7err(PKCS7_F_PKCS7_DECRYPT_RINFO, ERR_R_EVP_LIB);
240                 goto err;
241                 }

243         ret = 1;

245         if (*pek)
246                 {
247                 OPENSSL_cleanse(*pek, *peklen);
248                 OPENSSL_free(*pek);
249                 }

251         *pek = ek;
252         *peklen = eklen;

254         err:
255         if (pctx)
256                 EVP_PKEY_CTX_free(pctx);
257         if (!ret && ek)
258                 OPENSSL_free(ek);
```

```
260         return ret;
261         }

263 BIO *PKCS7_dataInit(PKCS7 *p7, BIO *bio)
264         {
265         int i;
266         BIO *out=NULL,*btmp=NULL;
267         X509_ALGOR *xa = NULL;
268         const EVP_CIPHER *evp_cipher=NULL;
269         STACK_OF(X509_ALGOR) *md_sk=NULL;
270         STACK_OF(PKCS7_RECIP_INFO) *rsk=NULL;
271         X509_ALGOR *xalg=NULL;
272         PKCS7_RECIP_INFO *ri=NULL;
273         ASN1_OCTET_STRING *os=NULL;

275         i=OBJ_obj2nid(p7->type);
276         p7->state=PKCS7_S_HEADER;

278         switch (i)
279                 {
280         case NID_pkcs7_signed:
281                 md_sk=p7->d.sign->md_algs;
282                 os = PKCS7_get_octet_string(p7->d.sign->contents);
283                 break;
284         case NID_pkcs7_signedAndEnveloped:
285                 rsk=p7->d.signed_and_enveloped->recipientinfo;
286                 md_sk=p7->d.signed_and_enveloped->md_algs;
287                 xalg=p7->d.signed_and_enveloped->enc_data->algorithm;
288                 evp_cipher=p7->d.signed_and_enveloped->enc_data->cipher;
289                 if (evp_cipher == NULL)
290                         {
291                         PKCS7err(PKCS7_F_PKCS7_DATAINIT,
292                                                 PKCS7_R_CIPHER_NOT_INITIALIZED);
293                         goto err;
294                         }
295                 break;
296         case NID_pkcs7_enveloped:
297                 rsk=p7->d.enveloped->recipientinfo;
298                 xalg=p7->d.enveloped->enc_data->algorithm;
299                 evp_cipher=p7->d.enveloped->enc_data->cipher;
300                 if (evp_cipher == NULL)
301                         {
302                         PKCS7err(PKCS7_F_PKCS7_DATAINIT,
303                                                 PKCS7_R_CIPHER_NOT_INITIALIZED);
304                         goto err;
305                         }
306                 break;
307         case NID_pkcs7_digest:
308                 xa = p7->d.digest->md;
309                 os = PKCS7_get_octet_string(p7->d.digest->contents);
310                 break;
311         case NID_pkcs7_data:
312                 break;
313         default:
314                 PKCS7err(PKCS7_F_PKCS7_DATAINIT,PKCS7_R_UNSUPPORTED_CONTENT_TYPE
315                 goto err;
316                 }

318         for (i=0; i<sk_X509_ALGOR_num(md_sk); i++)
319                 if (!PKCS7_bio_add_digest(&out, sk_X509_ALGOR_value(md_sk, i)))
320                         goto err;

322         if (xa && !PKCS7_bio_add_digest(&out, xa))
323                         goto err;

325         if (evp_cipher != NULL)
```



```
326                 {
327                 unsigned char key[EVP_MAX_KEY_LENGTH];
328                 unsigned char iv[EVP_MAX_IV_LENGTH];
329                 int keylen,ivlen;
330                 EVP_CIPHER_CTX *ctx;

332                 if ((btmp=BIO_new(BIO_f_cipher())) == NULL)
333                         {
334                         PKCS7err(PKCS7_F_PKCS7_DATAINIT,ERR_R_BIO_LIB);
335                         goto err;
336                         }
337                 BIO_get_cipher_ctx(btmp, &ctx);
338                 keylen=EVP_CIPHER_key_length(evp_cipher);
339                 ivlen=EVP_CIPHER_iv_length(evp_cipher);
340                 xalg->algorithm = OBJ_nid2obj(EVP_CIPHER_type(evp_cipher));
341                 if (ivlen > 0)
342                         if (RAND_pseudo_bytes(iv,ivlen) <= 0)
343                                 goto err;
344                 if (EVP_CipherInit_ex(ctx, evp_cipher, NULL, NULL, NULL, 1)<=0)
345                         goto err;
346                 if (EVP_CIPHER_CTX_rand_key(ctx, key) <= 0)
347                         goto err;
348                 if (EVP_CipherInit_ex(ctx, NULL, NULL, key, iv, 1) <= 0)
349                         goto err;

351                 if (ivlen > 0) {
352                         if (xalg->parameter == NULL) {
353                                 xalg->parameter = ASN1_TYPE_new();
354                                 if (xalg->parameter == NULL)
355                                         goto err;
356                         }
357                         if(EVP_CIPHER_param_to_asn1(ctx, xalg->parameter) < 0)
358                                 goto err;
359                 }

361                 /* Lets do the pub key stuff :-) */
362                 for (i=0; i<sk_PKCS7_RECIP_INFO_num(rsk); i++)
363                         {
364                         ri=sk_PKCS7_RECIP_INFO_value(rsk,i);
365                         if (pkcs7_encode_rinfo(ri, key, keylen) <= 0)
366                                 goto err;
367                         }
368                 OPENSSL_cleanse(key, keylen);

370                 if (out == NULL)
371                         out=btmp;
372                 else
373                         BIO_push(out,btmp);
374                 btmp=NULL;
375                 }

377         if (bio == NULL)
378                 {
379                 if (PKCS7_is_detached(p7))
380                         bio=BIO_new(BIO_s_null());
381                 else if (os && os->length > 0)
382                         bio = BIO_new_mem_buf(os->data, os->length);
383                 if(bio == NULL)
384                         {
385                         bio=BIO_new(BIO_s_mem());
386                         if (bio == NULL)
387                                 goto err;
388                         BIO_set_mem_eof_return(bio,0);
389                         }
390                 }
391         if (out)
```

```
392                 BIO_push(out,bio);
393         else
394                 out = bio;
395         bio=NULL;
396         if (0)
397                 {
398 err:
399                 if (out != NULL)
400                         BIO_free_all(out);
401                 if (btmp != NULL)
402                         BIO_free_all(btmp);
403                 out=NULL;
404                 }
405         return(out);
406         }

408 static int pkcs7_cmp_ri(PKCS7_RECIP_INFO *ri, X509 *pcert)
409         {
410         int ret;
411         ret = X509_NAME_cmp(ri->issuer_and_serial->issuer,
412                             pcert->cert_info->issuer);
413         if (ret)
414                 return ret;
415         return M_ASN1_INTEGER_cmp(pcert->cert_info->serialNumber,
416                                   ri->issuer_and_serial->serial);
417         }

419 /* int */
420 BIO *PKCS7_dataDecode(PKCS7 *p7, EVP_PKEY *pkey, BIO *in_bio, X509 *pcert)
421         {
422         int i,j;
423         BIO *out=NULL,*btmp=NULL,*etmp=NULL,*bio=NULL;
424         X509_ALGOR *xa;
425         ASN1_OCTET_STRING *data_body=NULL;
426         const EVP_MD *evp_md;
427         const EVP_CIPHER *evp_cipher=NULL;
428         EVP_CIPHER_CTX *evp_ctx=NULL;
429         X509_ALGOR *enc_alg=NULL;
430         STACK_OF(X509_ALGOR) *md_sk=NULL;
431         STACK_OF(PKCS7_RECIP_INFO) *rsk=NULL;
432         PKCS7_RECIP_INFO *ri=NULL;
433        unsigned char *ek = NULL, *tkey = NULL;
434        int eklen = 0, tkeylen = 0;

436         i=OBJ_obj2nid(p7->type);
437         p7->state=PKCS7_S_HEADER;

439         switch (i)
440                 {
441         case NID_pkcs7_signed:
442                 data_body=PKCS7_get_octet_string(p7->d.sign->contents);
443                 if (!PKCS7_is_detached(p7) && data_body == NULL)
444                         {
445                         PKCS7err(PKCS7_F_PKCS7_DATADECODE,PKCS7_R_INVALID_SIGNED
446                         goto err;
447                         }
448                 md_sk=p7->d.sign->md_algs;
449                 break;
450         case NID_pkcs7_signedAndEnveloped:
451                 rsk=p7->d.signed_and_enveloped->recipientinfo;
452                 md_sk=p7->d.signed_and_enveloped->md_algs;
453                 data_body=p7->d.signed_and_enveloped->enc_data->enc_data;
454                 enc_alg=p7->d.signed_and_enveloped->enc_data->algorithm;
455                 evp_cipher=EVP_get_cipherbyobj(enc_alg->algorithm);
456                 if (evp_cipher == NULL)
457                         {
```



```
458                         PKCS7err(PKCS7_F_PKCS7_DATADECODE,PKCS7_R_UNSUPPORTED_CI
459                         goto err;
460                         }
461                 break;
462         case NID_pkcs7_enveloped:
463                 rsk=p7->d.enveloped->recipientinfo;
464                 enc_alg=p7->d.enveloped->enc_data->algorithm;
465                 data_body=p7->d.enveloped->enc_data->enc_data;
466                 evp_cipher=EVP_get_cipherbyobj(enc_alg->algorithm);
467                 if (evp_cipher == NULL)
468                         {
469                         PKCS7err(PKCS7_F_PKCS7_DATADECODE,PKCS7_R_UNSUPPORTED_CI
470                         goto err;
471                         }
472                 break;
473         default:
474                 PKCS7err(PKCS7_F_PKCS7_DATADECODE,PKCS7_R_UNSUPPORTED_CONTENT_TY
475                 goto err;
476                 }

478         /* We will be checking the signature */
479         if (md_sk != NULL)
480                 {
481                 for (i=0; i<sk_X509_ALGOR_num(md_sk); i++)
482                         {
483                         xa=sk_X509_ALGOR_value(md_sk,i);
484                         if ((btmp=BIO_new(BIO_f_md())) == NULL)
485                                 {
486                                 PKCS7err(PKCS7_F_PKCS7_DATADECODE,ERR_R_BIO_LIB)
487                                 goto err;
488                                 }

490                         j=OBJ_obj2nid(xa->algorithm);
491                         evp_md=EVP_get_digestbynid(j);
492                         if (evp_md == NULL)
493                                 {
494                                 PKCS7err(PKCS7_F_PKCS7_DATADECODE,PKCS7_R_UNKNOW
495                                 goto err;
496                                 }

498                         BIO_set_md(btmp,evp_md);
499                         if (out == NULL)
500                                 out=btmp;
501                         else
502                                 BIO_push(out,btmp);
503                         btmp=NULL;
504                         }
505                 }

507         if (evp_cipher != NULL)
508                 {
509 #if 0
510                 unsigned char key[EVP_MAX_KEY_LENGTH];
511                 unsigned char iv[EVP_MAX_IV_LENGTH];
512                 unsigned char *p;
513                 int keylen,ivlen;
514                 int max;
515                 X509_OBJECT ret;
516 #endif

518                 if ((etmp=BIO_new(BIO_f_cipher())) == NULL)
519                         {
520                         PKCS7err(PKCS7_F_PKCS7_DATADECODE,ERR_R_BIO_LIB);
521                         goto err;
522                         }
```

```
524                 /* It was encrypted, we need to decrypt the secret key
525                  * with the private key */
526
527                 /* Find the recipientInfo which matches the passed certificate
528                  * (if any)
529                  */
530
531                 if (pcert)
532                         {
533                         for (i=0; i<sk_PKCS7_RECIP_INFO_num(rsk); i++)
534                                 {
535                                 ri=sk_PKCS7_RECIP_INFO_value(rsk,i);
536                                 if (!pkcs7_cmp_ri(ri, pcert))
537                                         break;
538                                 ri=NULL;
539                                 }
540                         if (ri == NULL)
541                                 {
542                                 PKCS7err(PKCS7_F_PKCS7_DATADECODE,
543                                       PKCS7_R_NO_RECIPIENT_MATCHES_CERTIFICATE);
544                                 goto err;
545                                 }
546                         }
547
548                 /* If we haven't got a certificate try each ri in turn */
549                 if (pcert == NULL)
550                         {
551                         /* Always attempt to decrypt all rinfo even
552                          * after sucess as a defence against MMA timing
553                          * attacks.
554                          */
555                         for (i=0; i<sk_PKCS7_RECIP_INFO_num(rsk); i++)
556                                 {
557                                 ri=sk_PKCS7_RECIP_INFO_value(rsk,i);
558
559                                 if (pkcs7_decrypt_rinfo(&ek, &eklen,
560                                                         ri, pkey) < 0)
561                                         goto err;
562                                 ERR_clear_error();
563                                 }
564                         }
565                 else
566                         {
567                         /* Only exit on fatal errors, not decrypt failure */
568                         if (pkcs7_decrypt_rinfo(&ek, &eklen, ri, pkey) < 0)
569                                 goto err;
570                         ERR_clear_error();
571                         }
572
573                 evp_ctx=NULL;
574                 BIO_get_cipher_ctx(etmp,&evp_ctx);
575                 if (EVP_CipherInit_ex(evp_ctx,evp_cipher,NULL,NULL,NULL,0) <= 0)
576                         goto err;
577                 if (EVP_CIPHER_asn1_to_param(evp_ctx,enc_alg->parameter) < 0)
578                         goto err;
579                 /* Generate random key as MMA defence */
580                 tkeylen = EVP_CIPHER_CTX_key_length(evp_ctx);
581                 tkey = OPENSSL_malloc(tkeylen);
582                 if (!tkey)
583                         goto err;
584                 if (EVP_CIPHER_CTX_rand_key(evp_ctx, tkey) <= 0)
585                         goto err;
586                 if (ek == NULL)
587                         {
588                         ek = tkey;
589                         eklen = tkeylen;
```



```
590                         tkey = NULL;
591                         }
592
593                 if (eklen != EVP_CIPHER_CTX_key_length(evp_ctx)) {
594                         /* Some S/MIME clients don't use the same key
595                          * and effective key length. The key length is
596                          * determined by the size of the decrypted RSA key.
597                          */
598                         if(!EVP_CIPHER_CTX_set_key_length(evp_ctx, eklen))
599                                 {
600                                 /* Use random key as MMA defence */
601                                 OPENSSL_cleanse(ek, eklen);
602                                 OPENSSL_free(ek);
603                                 ek = tkey;
604                                 eklen = tkeylen;
605                                 tkey = NULL;
606                                 }
607                 }
608                 /* Clear errors so we don't leak information useful in MMA */
609                 ERR_clear_error();
610                 if (EVP_CipherInit_ex(evp_ctx,NULL,NULL,ek,NULL,0) <= 0)
611                         goto err;
612
613                 if (ek)
614                         {
615                         OPENSSL_cleanse(ek,eklen);
616                         OPENSSL_free(ek);
617                         ek = NULL;
618                         }
619                 if (tkey)
620                         {
621                         OPENSSL_cleanse(tkey,tkeylen);
622                         OPENSSL_free(tkey);
623                         tkey = NULL;
624                         }
625
626                 if (out == NULL)
627                         out=etmp;
628                 else
629                         BIO_push(out,etmp);
630                 etmp=NULL;
631                 }
632
633 #if 1
634         if (PKCS7_is_detached(p7) || (in_bio != NULL))
635                 {
636                 bio=in_bio;
637                 }
638         else
639                 {
640 #if 0
641                 bio=BIO_new(BIO_s_mem());
642                 /* We need to set this so that when we have read all
643                  * the data, the encrypt BIO, if present, will read
644                  * EOF and encode the last few bytes */
645                 BIO_set_mem_eof_return(bio,0);
646
647                 if (data_body->length > 0)
648                         BIO_write(bio,(char *)data_body->data,data_body->length)
649 #else
650                 if (data_body->length > 0)
651                       bio = BIO_new_mem_buf(data_body->data,data_body->length);
652                 else {
653                         bio=BIO_new(BIO_s_mem());
654                         BIO_set_mem_eof_return(bio,0);
655                 }
```

```
656                 if (bio == NULL)
657                         goto err;
658 #endif
659                 }
660         BIO_push(out,bio);
661         bio=NULL;
662 #endif
663         if (0)
664                 {
665 err:
666                 if (ek)
667                         {
668                         OPENSSL_cleanse(ek,eklen);
669                         OPENSSL_free(ek);
670                         }
671                 if (tkey)
672                         {
673                         OPENSSL_cleanse(tkey,tkeylen);
674                         OPENSSL_free(tkey);
675                         }
676                 if (out != NULL) BIO_free_all(out);
677                 if (btmp != NULL) BIO_free_all(btmp);
678                 if (etmp != NULL) BIO_free_all(etmp);
679                 if (bio != NULL) BIO_free_all(bio);
680                 out=NULL;
681                 }
682         return(out);
683         }

685 static BIO *PKCS7_find_digest(EVP_MD_CTX **pmd, BIO *bio, int nid)
686         {
687         for (;;)
688                 {
689                 bio=BIO_find_type(bio,BIO_TYPE_MD);
690                 if (bio == NULL)
691                         {
692                         PKCS7err(PKCS7_F_PKCS7_FIND_DIGEST,PKCS7_R_UNABLE_TO_FIN
693                         return NULL;
694                         }
695                 BIO_get_md_ctx(bio,pmd);
696                 if (*pmd == NULL)
697                         {
698                         PKCS7err(PKCS7_F_PKCS7_FIND_DIGEST,ERR_R_INTERNAL_ERROR)
699                         return NULL;
700                         }
701                 if (EVP_MD_CTX_type(*pmd) == nid)
702                         return bio;
703                 bio=BIO_next(bio);
704                 }
705         return NULL;
706         }

708 static int do_pkcs7_signed_attrib(PKCS7_SIGNER_INFO *si, EVP_MD_CTX *mctx)
709         {
710         unsigned char md_data[EVP_MAX_MD_SIZE];
711         unsigned int md_len;

713         /* Add signing time if not already present */
714         if (!PKCS7_get_signed_attribute(si, NID_pkcs9_signingTime))
715                 {
716                 if (!PKCS7_add0_attrib_signing_time(si, NULL))
717                         {
718                         PKCS7err(PKCS7_F_DO_PKCS7_SIGNED_ATTRIB,
719                                         ERR_R_MALLOC_FAILURE);
720                         return 0;
721                         }
```



```
722                 }

724         /* Add digest */
725         if (!EVP_DigestFinal_ex(mctx, md_data,&md_len))
726                 {
727                 PKCS7err(PKCS7_F_DO_PKCS7_SIGNED_ATTRIB, ERR_R_EVP_LIB);
728                 return 0;
729                 }
730         if (!PKCS7_add1_attrib_digest(si, md_data, md_len))
731                 {
732                 PKCS7err(PKCS7_F_DO_PKCS7_SIGNED_ATTRIB, ERR_R_MALLOC_FAILURE);
733                 return 0;
734                 }

736         /* Now sign the attributes */
737         if (!PKCS7_SIGNER_INFO_sign(si))
738                         return 0;

740         return 1;
741         }


744 int PKCS7_dataFinal(PKCS7 *p7, BIO *bio)
745         {
746         int ret=0;
747         int i,j;
748         BIO *btmp;
749         PKCS7_SIGNER_INFO *si;
750         EVP_MD_CTX *mdc,ctx_tmp;
751         STACK_OF(X509_ATTRIBUTE) *sk;
752         STACK_OF(PKCS7_SIGNER_INFO) *si_sk=NULL;
753         ASN1_OCTET_STRING *os=NULL;

755         EVP_MD_CTX_init(&ctx_tmp);
756         i=OBJ_obj2nid(p7->type);
757         p7->state=PKCS7_S_HEADER;

759         switch (i)
760                 {
761         case NID_pkcs7_data:
762                 os = p7->d.data;
763                 break;
764         case NID_pkcs7_signedAndEnveloped:
765                 /* XXXXXXXXXXXXXXX */
766                 si_sk=p7->d.signed_and_enveloped->signer_info;
767                 os = p7->d.signed_and_enveloped->enc_data->enc_data;
768                 if (!os)
769                         {
770                         os=M_ASN1_OCTET_STRING_new();
771                         if (!os)
772                                 {
773                                 PKCS7err(PKCS7_F_PKCS7_DATAFINAL,ERR_R_MALLOC_FA
774                                 goto err;
775                                 }
776                         p7->d.signed_and_enveloped->enc_data->enc_data=os;
777                         }
778                 break;
779         case NID_pkcs7_enveloped:
780                 /* XXXXXXXXXXXXXXX */
781                 os = p7->d.enveloped->enc_data->enc_data;
782                 if (!os)
783                         {
784                         os=M_ASN1_OCTET_STRING_new();
785                         if (!os)
786                                 {
787                                 PKCS7err(PKCS7_F_PKCS7_DATAFINAL,ERR_R_MALLOC_FA
```

```
788                                 goto err;
789                                 }
790                         p7->d.enveloped->enc_data->enc_data=os;
791                         }
792                 break;
793         case NID_pkcs7_signed:
794                 si_sk=p7->d.sign->signer_info;
795                 os=PKCS7_get_octet_string(p7->d.sign->contents);
796                 /* If detached data then the content is excluded */
797                 if(PKCS7_type_is_data(p7->d.sign->contents) && p7->detached) {
798                         M_ASN1_OCTET_STRING_free(os);
799                         p7->d.sign->contents->d.data = NULL;
800                 }
801                 break;

803         case NID_pkcs7_digest:
804                 os=PKCS7_get_octet_string(p7->d.digest->contents);
805                 /* If detached data then the content is excluded */
806                 if(PKCS7_type_is_data(p7->d.digest->contents) && p7->detached)
807                         {
808                         M_ASN1_OCTET_STRING_free(os);
809                         p7->d.digest->contents->d.data = NULL;
810                         }
811                 break;

813         default:
814                 PKCS7err(PKCS7_F_PKCS7_DATAFINAL,PKCS7_R_UNSUPPORTED_CONTENT_TYP
815                 goto err;
816                 }

818         if (si_sk != NULL)
819                 {
820                 for (i=0; i<sk_PKCS7_SIGNER_INFO_num(si_sk); i++)
821                         {
822                         si=sk_PKCS7_SIGNER_INFO_value(si_sk,i);
823                         if (si->pkey == NULL)
824                                 continue;

826                         j = OBJ_obj2nid(si->digest_alg->algorithm);

828                         btmp=bio;

830                         btmp = PKCS7_find_digest(&mdc, btmp, j);

832                         if (btmp == NULL)
833                                 goto err;

835                         /* We now have the EVP_MD_CTX, lets do the
836                          * signing. */
837                         if (!EVP_MD_CTX_copy_ex(&ctx_tmp,mdc))
838                                 goto err;

840                         sk=si->auth_attr;

842                         /* If there are attributes, we add the digest
843                          * attribute and only sign the attributes */
844                         if (sk_X509_ATTRIBUTE_num(sk) > 0)
845                                 {
846                                 if (!do_pkcs7_signed_attrib(si, &ctx_tmp))
847                                         goto err;
848                                 }
849                         else
850                                 {
851                                 unsigned char *abuf = NULL;
852                                 unsigned int abuflen;
853                                 abuflen = EVP_PKEY_size(si->pkey);
```



```
854                                 abuf = OPENSSL_malloc(abuflen);
855                                 if (!abuf)
856                                         goto err;

858                                 if (!EVP_SignFinal(&ctx_tmp, abuf, &abuflen,
859                                                         si->pkey))
860                                         {
861                                         PKCS7err(PKCS7_F_PKCS7_DATAFINAL,
862                                                         ERR_R_EVP_LIB);
863                                         goto err;
864                                         }
865                                 ASN1_STRING_set0(si->enc_digest, abuf, abuflen);
866                                 }
867                         }
868                 }
869         else if (i == NID_pkcs7_digest)
870                 {
871                 unsigned char md_data[EVP_MAX_MD_SIZE];
872                 unsigned int md_len;
873                 if (!PKCS7_find_digest(&mdc, bio,
874                                 OBJ_obj2nid(p7->d.digest->md->algorithm)))
875                         goto err;
876                 if (!EVP_DigestFinal_ex(mdc,md_data,&md_len))
877                         goto err;
878                 M_ASN1_OCTET_STRING_set(p7->d.digest->digest, md_data, md_len);
879                 }

881         if (!PKCS7_is_detached(p7) && !(os->flags & ASN1_STRING_FLAG_NDEF))
882                 {
883                 char *cont;
884                 long contlen;
885                 btmp=BIO_find_type(bio,BIO_TYPE_MEM);
886                 if (btmp == NULL)
887                         {
888                         PKCS7err(PKCS7_F_PKCS7_DATAFINAL,PKCS7_R_UNABLE_TO_FIND_
889                         goto err;
890                         }
891                 contlen = BIO_get_mem_data(btmp, &cont);
892                 /* Mark the BIO read only then we can use its copy of the data
893                  * instead of making an extra copy.
894                  */
895                 BIO_set_flags(btmp, BIO_FLAGS_MEM_RDONLY);
896                 BIO_set_mem_eof_return(btmp, 0);
897                 ASN1_STRING_set0(os, (unsigned char *)cont, contlen);
898                 }
899         ret=1;
900 err:
901         EVP_MD_CTX_cleanup(&ctx_tmp);
902         return(ret);
903         }

905 int PKCS7_SIGNER_INFO_sign(PKCS7_SIGNER_INFO *si)
906         {
907         EVP_MD_CTX mctx;
908         EVP_PKEY_CTX *pctx;
909         unsigned char *abuf = NULL;
910         int alen;
911         size_t siglen;
912         const EVP_MD *md = NULL;

914         md = EVP_get_digestbyobj(si->digest_alg->algorithm);
915         if (md == NULL)
916                 return 0;

918         EVP_MD_CTX_init(&mctx);
919         if (EVP_DigestSignInit(&mctx, &pctx, md,NULL, si->pkey) <= 0)
```

```
920                 goto err;

922         if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_SIGN,
923                               EVP_PKEY_CTRL_PKCS7_SIGN, 0, si) <= 0)
924                 {
925                 PKCS7err(PKCS7_F_PKCS7_SIGNER_INFO_SIGN, PKCS7_R_CTRL_ERROR);
926                 goto err;
927                 }

929         alen = ASN1_item_i2d((ASN1_VALUE *)si->auth_attr,&abuf,
930                              ASN1_ITEM_rptr(PKCS7_ATTR_SIGN));
931         if(!abuf)
932                 goto err;
933         if (EVP_DigestSignUpdate(&mctx,abuf,alen) <= 0)
934                 goto err;
935         OPENSSL_free(abuf);
936         abuf = NULL;
937         if (EVP_DigestSignFinal(&mctx, NULL, &siglen) <= 0)
938                 goto err;
939         abuf = OPENSSL_malloc(siglen);
940         if(!abuf)
941                 goto err;
942         if (EVP_DigestSignFinal(&mctx, abuf, &siglen) <= 0)
943                 goto err;

945         if (EVP_PKEY_CTX_ctrl(pctx, -1, EVP_PKEY_OP_SIGN,
946                               EVP_PKEY_CTRL_PKCS7_SIGN, 1, si) <= 0)
947                 {
948                 PKCS7err(PKCS7_F_PKCS7_SIGNER_INFO_SIGN, PKCS7_R_CTRL_ERROR);
949                 goto err;
950                 }

952         EVP_MD_CTX_cleanup(&mctx);

954         ASN1_STRING_set0(si->enc_digest, abuf, siglen);

956         return 1;

958         err:
959         if (abuf)
960                 OPENSSL_free(abuf);
961         EVP_MD_CTX_cleanup(&mctx);
962         return 0;

964         }

966 int PKCS7_dataVerify(X509_STORE *cert_store, X509_STORE_CTX *ctx, BIO *bio,
967                      PKCS7 *p7, PKCS7_SIGNER_INFO *si)
968         {
969         PKCS7_ISSUER_AND_SERIAL *ias;
970         int ret=0,i;
971         STACK_OF(X509) *cert;
972         X509 *x509;

974         if (PKCS7_type_is_signed(p7))
975                 {
976                 cert=p7->d.sign->cert;
977                 }
978         else if (PKCS7_type_is_signedAndEnveloped(p7))
979                 {
980                 cert=p7->d.signed_and_enveloped->cert;
981                 }
982         else
983                 {
984                 PKCS7err(PKCS7_F_PKCS7_DATAVERIFY,PKCS7_R_WRONG_PKCS7_TYPE);
985                 goto err;
```



```
986                 }
987         /* XXXXXXXXXXXXXXXXXXXXXX */
988         ias=si->issuer_and_serial;

990         x509=X509_find_by_issuer_and_serial(cert,ias->issuer,ias->serial);

992         /* were we able to find the cert in passed to us */
993         if (x509 == NULL)
994                 {
995                 PKCS7err(PKCS7_F_PKCS7_DATAVERIFY,PKCS7_R_UNABLE_TO_FIND_CERTIFI
996                 goto err;
997                 }

999         /* Lets verify */
1000        if(!X509_STORE_CTX_init(ctx,cert_store,x509,cert))
1001                {
1002                PKCS7err(PKCS7_F_PKCS7_DATAVERIFY,ERR_R_X509_LIB);
1003                goto err;
1004                }
1005        X509_STORE_CTX_set_purpose(ctx, X509_PURPOSE_SMIME_SIGN);
1006        i=X509_verify_cert(ctx);
1007        if (i <= 0)
1008                {
1009                PKCS7err(PKCS7_F_PKCS7_DATAVERIFY,ERR_R_X509_LIB);
1010                X509_STORE_CTX_cleanup(ctx);
1011                goto err;
1012                }
1013        X509_STORE_CTX_cleanup(ctx);

1015        return PKCS7_signatureVerify(bio, p7, si, x509);
1016        err:
1017        return ret;
1018        }

1020 int PKCS7_signatureVerify(BIO *bio, PKCS7 *p7, PKCS7_SIGNER_INFO *si,
1021                                                                X509 *x509)
1022        {
1023        ASN1_OCTET_STRING *os;
1024        EVP_MD_CTX mdc_tmp,*mdc;
1025        int ret=0,i;
1026        int md_type;
1027        STACK_OF(X509_ATTRIBUTE) *sk;
1028        BIO *btmp;
1029        EVP_PKEY *pkey;

1031        EVP_MD_CTX_init(&mdc_tmp);

1033        if (!PKCS7_type_is_signed(p7) &&
1034                                !PKCS7_type_is_signedAndEnveloped(p7)) {
1035                PKCS7err(PKCS7_F_PKCS7_SIGNATUREVERIFY,
1036                                                PKCS7_R_WRONG_PKCS7_TYPE);
1037                goto err;
1038        }

1040        md_type=OBJ_obj2nid(si->digest_alg->algorithm);

1042        btmp=bio;
1043        for (;;)
1044                {
1045                if ((btmp == NULL) ||
1046                        ((btmp=BIO_find_type(btmp,BIO_TYPE_MD)) == NULL))
1047                        {
1048                        PKCS7err(PKCS7_F_PKCS7_SIGNATUREVERIFY,
1049                                                PKCS7_R_UNABLE_TO_FIND_MESSAGE_DIGEST);
1050                        goto err;
1051                        }
```

```
1052                 BIO_get_md_ctx(btmp,&mdc);
1053                 if (mdc == NULL)
1054                         {
1055                         PKCS7err(PKCS7_F_PKCS7_SIGNATUREVERIFY,
1056                                                         ERR_R_INTERNAL_ERROR);
1057                         goto err;
1058                         }
1059                 if (EVP_MD_CTX_type(mdc) == md_type)
1060                         break;
1061                 /* Workaround for some broken clients that put the signature
1062                  * OID instead of the digest OID in digest_alg->algorithm
1063                  */
1064                 if (EVP_MD_pkey_type(EVP_MD_CTX_md(mdc)) == md_type)
1065                         break;
1066                 btmp=BIO_next(btmp);
1067                 }
1069         /* mdc is the digest ctx that we want, unless there are attributes,
1070          * in which case the digest is the signed attributes */
1071         if (!EVP_MD_CTX_copy_ex(&mdc_tmp,mdc))
1072                 goto err;
1074         sk=si->auth_attr;
1075         if ((sk != NULL) && (sk_X509_ATTRIBUTE_num(sk) != 0))
1076                 {
1077                 unsigned char md_dat[EVP_MAX_MD_SIZE], *abuf = NULL;
1078                 unsigned int md_len;
1079                 int alen;
1080                 ASN1_OCTET_STRING *message_digest;
1082                 if (!EVP_DigestFinal_ex(&mdc_tmp,md_dat,&md_len))
1083                         goto err;
1084                 message_digest=PKCS7_digest_from_attributes(sk);
1085                 if (!message_digest)
1086                         {
1087                         PKCS7err(PKCS7_F_PKCS7_SIGNATUREVERIFY,
1088                                         PKCS7_R_UNABLE_TO_FIND_MESSAGE_DIGEST);
1089                         goto err;
1090                         }
1091                 if ((message_digest->length != (int)md_len) ||
1092                         (memcmp(message_digest->data,md_dat,md_len)))
1093                         {
1094 #if 0
1095 {
1096 int ii;
1097 for (ii=0; ii<message_digest->length; ii++)
1098         printf("%02X",message_digest->data[ii]); printf(" sent\n");
1099 for (ii=0; ii<md_len; ii++) printf("%02X",md_dat[ii]); printf(" calc\n");
1100 }
1101 #endif
1102                         PKCS7err(PKCS7_F_PKCS7_SIGNATUREVERIFY,
1103                                                         PKCS7_R_DIGEST_FAILURE);
1104                         ret= -1;
1105                         goto err;
1106                         }
1108                 if (!EVP_VerifyInit_ex(&mdc_tmp,EVP_get_digestbynid(md_type), NU
1109                         goto err;
1111                 alen = ASN1_item_i2d((ASN1_VALUE *)sk, &abuf,
1112                                                 ASN1_ITEM_rptr(PKCS7_ATTR_VERIFY
1113                 if (alen <= 0)
1114                         {
1115                         PKCS7err(PKCS7_F_PKCS7_SIGNATUREVERIFY,ERR_R_ASN1_LIB);
1116                         ret = -1;
1117                         goto err;
```



```
1118                         }
1119                 if (!EVP_VerifyUpdate(&mdc_tmp, abuf, alen))
1120                         goto err;
1122                 OPENSSL_free(abuf);
1123                 }
1125         os=si->enc_digest;
1126         pkey = X509_get_pubkey(x509);
1127         if (!pkey)
1128                 {
1129                 ret = -1;
1130                 goto err;
1131                 }
1133         i=EVP_VerifyFinal(&mdc_tmp,os->data,os->length, pkey);
1134         EVP_PKEY_free(pkey);
1135         if (i <= 0)
1136                 {
1137                 PKCS7err(PKCS7_F_PKCS7_SIGNATUREVERIFY,
1138                                                 PKCS7_R_SIGNATURE_FAILURE);
1139                 ret= -1;
1140                 goto err;
1141                 }
1142         else
1143                 ret=1;
1144 err:
1145         EVP_MD_CTX_cleanup(&mdc_tmp);
1146         return(ret);
1147         }
1149 PKCS7_ISSUER_AND_SERIAL *PKCS7_get_issuer_and_serial(PKCS7 *p7, int idx)
1150         {
1151         STACK_OF(PKCS7_RECIP_INFO) *rsk;
1152         PKCS7_RECIP_INFO *ri;
1153         int i;
1155         i=OBJ_obj2nid(p7->type);
1156         if (i != NID_pkcs7_signedAndEnveloped)
1157                 return NULL;
1158         if (p7->d.signed_and_enveloped == NULL)
1159                 return NULL;
1160         rsk=p7->d.signed_and_enveloped->recipientinfo;
1161         if (rsk == NULL)
1162                 return NULL;
1163         ri=sk_PKCS7_RECIP_INFO_value(rsk,0);
1164         if (sk_PKCS7_RECIP_INFO_num(rsk) <= idx) return(NULL);
1165         ri=sk_PKCS7_RECIP_INFO_value(rsk,idx);
1166         return(ri->issuer_and_serial);
1167         }
1169 ASN1_TYPE *PKCS7_get_signed_attribute(PKCS7_SIGNER_INFO *si, int nid)
1170         {
1171         return(get_attribute(si->auth_attr,nid));
1172         }
1174 ASN1_TYPE *PKCS7_get_attribute(PKCS7_SIGNER_INFO *si, int nid)
1175         {
1176         return(get_attribute(si->unauth_attr,nid));
1177         }
1179 static ASN1_TYPE *get_attribute(STACK_OF(X509_ATTRIBUTE) *sk, int nid)
1180         {
1181         int i;
1182         X509_ATTRIBUTE *xa;
1183         ASN1_OBJECT *o;
```

```
1185         o=OBJ_nid2obj(nid);
1186         if (!o || !sk) return(NULL);
1187         for (i=0; i<sk_X509_ATTRIBUTE_num(sk); i++)
1188                 {
1189                 xa=sk_X509_ATTRIBUTE_value(sk,i);
1190                 if (OBJ_cmp(xa->object,o) == 0)
1191                         {
1192                         if (!xa->single && sk_ASN1_TYPE_num(xa->value.set))
1193                                 return(sk_ASN1_TYPE_value(xa->value.set,0));
1194                         else
1195                                 return(NULL);
1196                         }
1197                 }
1198         return(NULL);
1199         }

1201 ASN1_OCTET_STRING *PKCS7_digest_from_attributes(STACK_OF(X509_ATTRIBUTE) *sk)
1202 {
1203         ASN1_TYPE *astype;
1204         if(!(astype = get_attribute(sk, NID_pkcs9_messageDigest))) return NULL;
1205         return astype->value.octet_string;
1206 }

1208 int PKCS7_set_signed_attributes(PKCS7_SIGNER_INFO *p7si,
1209                                 STACK_OF(X509_ATTRIBUTE) *sk)
1210         {
1211         int i;

1213         if (p7si->auth_attr != NULL)
1214                 sk_X509_ATTRIBUTE_pop_free(p7si->auth_attr,X509_ATTRIBUTE_free);
1215         p7si->auth_attr=sk_X509_ATTRIBUTE_dup(sk);
1216         if (p7si->auth_attr == NULL)
1217                 return 0;
1218         for (i=0; i<sk_X509_ATTRIBUTE_num(sk); i++)
1219                 {
1220                 if ((sk_X509_ATTRIBUTE_set(p7si->auth_attr,i,
1221                         X509_ATTRIBUTE_dup(sk_X509_ATTRIBUTE_value(sk,i))))
1222                     == NULL)
1223                         return(0);
1224                 }
1225         return(1);
1226         }

1228 int PKCS7_set_attributes(PKCS7_SIGNER_INFO *p7si, STACK_OF(X509_ATTRIBUTE) *sk)
1229         {
1230         int i;

1232         if (p7si->unauth_attr != NULL)
1233                 sk_X509_ATTRIBUTE_pop_free(p7si->unauth_attr,
1234                                            X509_ATTRIBUTE_free);
1235         p7si->unauth_attr=sk_X509_ATTRIBUTE_dup(sk);
1236         if (p7si->unauth_attr == NULL)
1237                 return 0;
1238         for (i=0; i<sk_X509_ATTRIBUTE_num(sk); i++)
1239                 {
1240                 if ((sk_X509_ATTRIBUTE_set(p7si->unauth_attr,i,
1241                         X509_ATTRIBUTE_dup(sk_X509_ATTRIBUTE_value(sk,i))))
1242                     == NULL)
1243                         return(0);
1244                 }
1245         return(1);
1246         }

1248 int PKCS7_add_signed_attribute(PKCS7_SIGNER_INFO *p7si, int nid, int atrtype,
1249              void *value)
```



```
1250         {
1251         return(add_attribute(&(p7si->auth_attr),nid,atrtype,value));
1252         }

1254 int PKCS7_add_attribute(PKCS7_SIGNER_INFO *p7si, int nid, int atrtype,
1255              void *value)
1256         {
1257         return(add_attribute(&(p7si->unauth_attr),nid,atrtype,value));
1258         }

1260 static int add_attribute(STACK_OF(X509_ATTRIBUTE) **sk, int nid, int atrtype,
1261                          void *value)
1262         {
1263         X509_ATTRIBUTE *attr=NULL;

1265         if (*sk == NULL)
1266                 {
1267                 *sk = sk_X509_ATTRIBUTE_new_null();
1268                 if (*sk == NULL)
1269                         return 0;
1270 new_attrib:
1271                 if (!(attr=X509_ATTRIBUTE_create(nid,atrtype,value)))
1272                         return 0;
1273                 if (!sk_X509_ATTRIBUTE_push(*sk,attr))
1274                         {
1275                         X509_ATTRIBUTE_free(attr);
1276                         return 0;
1277                         }
1278                 }
1279         else
1280                 {
1281                 int i;

1283                 for (i=0; i<sk_X509_ATTRIBUTE_num(*sk); i++)
1284                         {
1285                         attr=sk_X509_ATTRIBUTE_value(*sk,i);
1286                         if (OBJ_obj2nid(attr->object) == nid)
1287                                 {
1288                                 X509_ATTRIBUTE_free(attr);
1289                                 attr=X509_ATTRIBUTE_create(nid,atrtype,value);
1290                                 if (attr == NULL)
1291                                         return 0;
1292                                 if (!sk_X509_ATTRIBUTE_set(*sk,i,attr))
1293                                         {
1294                                         X509_ATTRIBUTE_free(attr);
1295                                         return 0;
1296                                         }
1297                                 goto end;
1298                                 }
1299                         }
1300                 goto new_attrib;
1301                 }
1302 end:
1303         return(1);
1304         }
1305 #endif /* ! codereview */
```

```
**********************************************************
   16050 Wed Aug 13 19:53:04 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/pk7_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pkcs7/pk7_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/x509.h>
#include "asn1_locl.h"

long PKCS7_ctrl(PKCS7 *p7, int cmd, long larg, char *parg)
        {
        int nid;
        long ret;

        nid=OBJ_obj2nid(p7->type);

        switch (cmd)
                {
        case PKCS7_OP_SET_DETACHED_SIGNATURE:
                if (nid == NID_pkcs7_signed)
                        {
                        ret=p7->detached=(int)larg;
                        if (ret && PKCS7_type_is_data(p7->d.sign->contents))
                                {
                                ASN1_OCTET_STRING *os;
                                os=p7->d.sign->contents->d.data;
                                ASN1_OCTET_STRING_free(os);
                                p7->d.sign->contents->d.data = NULL;
                                }
                        }
                else
                        {
                        PKCS7err(PKCS7_F_PKCS7_CTRL,PKCS7_R_OPERATION_NOT_SUPPOR
                        ret=0;
                        }
                break;
        case PKCS7_OP_GET_DETACHED_SIGNATURE:
                if (nid == NID_pkcs7_signed)
                        {
                        if(!p7->d.sign  || !p7->d.sign->contents->d.ptr)
                                ret = 1;
                        else ret = 0;

                        p7->detached = ret;
                        }
                else
                        {
                        PKCS7err(PKCS7_F_PKCS7_CTRL,PKCS7_R_OPERATION_NOT_SUPPOR
                        ret=0;
                        }

                break;
        default:
                PKCS7err(PKCS7_F_PKCS7_CTRL,PKCS7_R_UNKNOWN_OPERATION);
                ret=0;
                }
        return(ret);
        }

int PKCS7_content_new(PKCS7 *p7, int type)
        {
        PKCS7 *ret=NULL;

        if ((ret=PKCS7_new()) == NULL) goto err;
        if (!PKCS7_set_type(ret,type)) goto err;
        if (!PKCS7_set_content(p7,ret)) goto err;

        return(1);
err:
        if (ret != NULL) PKCS7_free(ret);
        return(0);
        }
```

```
int PKCS7_set_content(PKCS7 *p7, PKCS7 *p7_data)
        {
        int i;

        i=OBJ_obj2nid(p7->type);
        switch (i)
                {
        case NID_pkcs7_signed:
                if (p7->d.sign->contents != NULL)
                        PKCS7_free(p7->d.sign->contents);
                p7->d.sign->contents=p7_data;
                break;
        case NID_pkcs7_digest:
                if (p7->d.digest->contents != NULL)
                        PKCS7_free(p7->d.digest->contents);
                p7->d.digest->contents=p7_data;
                break;
        case NID_pkcs7_data:
        case NID_pkcs7_enveloped:
        case NID_pkcs7_signedAndEnveloped:
        case NID_pkcs7_encrypted:
        default:
                PKCS7err(PKCS7_F_PKCS7_SET_CONTENT,PKCS7_R_UNSUPPORTED_CONTENT_T
                goto err;
                }
        return(1);
err:
        return(0);
        }

int PKCS7_set_type(PKCS7 *p7, int type)
        {
        ASN1_OBJECT *obj;

        /*PKCS7_content_free(p7);*/
        obj=OBJ_nid2obj(type); /* will not fail */

        switch (type)
                {
        case NID_pkcs7_signed:
                p7->type=obj;
                if ((p7->d.sign=PKCS7_SIGNED_new()) == NULL)
                        goto err;
                if (!ASN1_INTEGER_set(p7->d.sign->version,1))
                        {
                        PKCS7_SIGNED_free(p7->d.sign);
                        p7->d.sign=NULL;
                        goto err;
                        }
                break;
        case NID_pkcs7_data:
                p7->type=obj;
                if ((p7->d.data=M_ASN1_OCTET_STRING_new()) == NULL)
                        goto err;
                break;
        case NID_pkcs7_signedAndEnveloped:
                p7->type=obj;
                if ((p7->d.signed_and_enveloped=PKCS7_SIGN_ENVELOPE_new())
                        == NULL) goto err;
                ASN1_INTEGER_set(p7->d.signed_and_enveloped->version,1);
                if (!ASN1_INTEGER_set(p7->d.signed_and_enveloped->version,1))
                        goto err;
                p7->d.signed_and_enveloped->enc_data->content_type
                                                = OBJ_nid2obj(NID_pkcs7_data);
                break;
```



```
        case NID_pkcs7_enveloped:
                p7->type=obj;
                if ((p7->d.enveloped=PKCS7_ENVELOPE_new())
                        == NULL) goto err;
                if (!ASN1_INTEGER_set(p7->d.enveloped->version,0))
                        goto err;
                p7->d.enveloped->enc_data->content_type
                                                = OBJ_nid2obj(NID_pkcs7_data);
                break;
        case NID_pkcs7_encrypted:
                p7->type=obj;
                if ((p7->d.encrypted=PKCS7_ENCRYPT_new())
                        == NULL) goto err;
                if (!ASN1_INTEGER_set(p7->d.encrypted->version,0))
                        goto err;
                p7->d.encrypted->enc_data->content_type
                                                = OBJ_nid2obj(NID_pkcs7_data);
                break;

        case NID_pkcs7_digest:
                p7->type=obj;
                if ((p7->d.digest=PKCS7_DIGEST_new())
                        == NULL) goto err;
                if (!ASN1_INTEGER_set(p7->d.digest->version,0))
                        goto err;
                break;
        default:
                PKCS7err(PKCS7_F_PKCS7_SET_TYPE,PKCS7_R_UNSUPPORTED_CONTENT_TYPE
                goto err;
                }
        return(1);
err:
        return(0);
        }

int PKCS7_set0_type_other(PKCS7 *p7, int type, ASN1_TYPE *other)
        {
        p7->type = OBJ_nid2obj(type);
        p7->d.other = other;
        return 1;
        }

int PKCS7_add_signer(PKCS7 *p7, PKCS7_SIGNER_INFO *psi)
        {
        int i,j,nid;
        X509_ALGOR *alg;
        STACK_OF(PKCS7_SIGNER_INFO) *signer_sk;
        STACK_OF(X509_ALGOR) *md_sk;

        i=OBJ_obj2nid(p7->type);
        switch (i)
                {
        case NID_pkcs7_signed:
                signer_sk=      p7->d.sign->signer_info;
                md_sk=          p7->d.sign->md_algs;
                break;
        case NID_pkcs7_signedAndEnveloped:
                signer_sk=      p7->d.signed_and_enveloped->signer_info;
                md_sk=          p7->d.signed_and_enveloped->md_algs;
                break;
        default:
                PKCS7err(PKCS7_F_PKCS7_ADD_SIGNER,PKCS7_R_WRONG_CONTENT_TYPE);
                return(0);
                }

        nid=OBJ_obj2nid(psi->digest_alg->algorithm);
```

```
261         /* If the digest is not currently listed, add it */
262         j=0;
263         for (i=0; i<sk_X509_ALGOR_num(md_sk); i++)
264                 {
265                 alg=sk_X509_ALGOR_value(md_sk,i);
266                 if (OBJ_obj2nid(alg->algorithm) == nid)
267                         {
268                         j=1;
269                         break;
270                         }
271                 }
272         if (!j) /* we need to add another algorithm */
273                 {
274                 if(!(alg=X509_ALGOR_new())
275                         || !(alg->parameter = ASN1_TYPE_new()))
276                         {
277                         X509_ALGOR_free(alg);
278                         PKCS7err(PKCS7_F_PKCS7_ADD_SIGNER,ERR_R_MALLOC_FAILURE);
279                         return(0);
280                         }
281                 alg->algorithm=OBJ_nid2obj(nid);
282                 alg->parameter->type = V_ASN1_NULL;
283                 if (!sk_X509_ALGOR_push(md_sk,alg))
284                         {
285                         X509_ALGOR_free(alg);
286                         return 0;
287                         }
288                 }

290         if (!sk_PKCS7_SIGNER_INFO_push(signer_sk,psi))
291                 return 0;
292         return(1);
293         }

295 int PKCS7_add_certificate(PKCS7 *p7, X509 *x509)
296         {
297         int i;
298         STACK_OF(X509) **sk;

300         i=OBJ_obj2nid(p7->type);
301         switch (i)
302                 {
303         case NID_pkcs7_signed:
304                 sk= &(p7->d.sign->cert);
305                 break;
306         case NID_pkcs7_signedAndEnveloped:
307                 sk= &(p7->d.signed_and_enveloped->cert);
308                 break;
309         default:
310                 PKCS7err(PKCS7_F_PKCS7_ADD_CERTIFICATE,PKCS7_R_WRONG_CONTENT_TYP
311                 return(0);
312                 }

314         if (*sk == NULL)
315                 *sk=sk_X509_new_null();
316         if (*sk == NULL)
317                 {
318                 PKCS7err(PKCS7_F_PKCS7_ADD_CERTIFICATE, ERR_R_MALLOC_FAILURE);
319                 return 0;
320                 }
321         CRYPTO_add(&x509->references,1,CRYPTO_LOCK_X509);
322         if (!sk_X509_push(*sk,x509))
323                 {
324                 X509_free(x509);
325                 return 0;
```



```
326                 }
327         return(1);
328         }

330 int PKCS7_add_crl(PKCS7 *p7, X509_CRL *crl)
331         {
332         int i;
333         STACK_OF(X509_CRL) **sk;

335         i=OBJ_obj2nid(p7->type);
336         switch (i)
337                 {
338         case NID_pkcs7_signed:
339                 sk= &(p7->d.sign->crl);
340                 break;
341         case NID_pkcs7_signedAndEnveloped:
342                 sk= &(p7->d.signed_and_enveloped->crl);
343                 break;
344         default:
345                 PKCS7err(PKCS7_F_PKCS7_ADD_CRL,PKCS7_R_WRONG_CONTENT_TYPE);
346                 return(0);
347                 }

349         if (*sk == NULL)
350                 *sk=sk_X509_CRL_new_null();
351         if (*sk == NULL)
352                 {
353                 PKCS7err(PKCS7_F_PKCS7_ADD_CRL,ERR_R_MALLOC_FAILURE);
354                 return 0;
355                 }

357         CRYPTO_add(&crl->references,1,CRYPTO_LOCK_X509_CRL);
358         if (!sk_X509_CRL_push(*sk,crl))
359                 {
360                 X509_CRL_free(crl);
361                 return 0;
362                 }
363         return(1);
364         }

366 int PKCS7_SIGNER_INFO_set(PKCS7_SIGNER_INFO *p7i, X509 *x509, EVP_PKEY *pkey,
367                 const EVP_MD *dgst)
368         {
369         int ret;

371         /* We now need to add another PKCS7_SIGNER_INFO entry */
372         if (!ASN1_INTEGER_set(p7i->version,1))
373                 goto err;
374         if (!X509_NAME_set(&p7i->issuer_and_serial->issuer,
375                         X509_get_issuer_name(x509)))
376                 goto err;

378         /* because ASN1_INTEGER_set is used to set a 'long' we will do
379          * things the ugly way. */
380         M_ASN1_INTEGER_free(p7i->issuer_and_serial->serial);
381         if (!(p7i->issuer_and_serial->serial=
382                         M_ASN1_INTEGER_dup(X509_get_serialNumber(x509))))
383                 goto err;

385         /* lets keep the pkey around for a while */
386         CRYPTO_add(&pkey->references,1,CRYPTO_LOCK_EVP_PKEY);
387         p7i->pkey=pkey;

389         /* Set the algorithms */

391         X509_ALGOR_set0(p7i->digest_alg, OBJ_nid2obj(EVP_MD_type(dgst)),
```

```
392                                 V_ASN1_NULL, NULL);

394         if (pkey->ameth && pkey->ameth->pkey_ctrl)
395                 {
396                 ret = pkey->ameth->pkey_ctrl(pkey, ASN1_PKEY_CTRL_PKCS7_SIGN,
397                                              0, p7i);
398                 if (ret > 0)
399                         return 1;
400                 if (ret != -2)
401                         {
402                         PKCS7err(PKCS7_F_PKCS7_SIGNER_INFO_SET,
403                                  PKCS7_R_SIGNING_CTRL_FAILURE);
404                         return 0;
405                         }
406                 }
407         PKCS7err(PKCS7_F_PKCS7_SIGNER_INFO_SET,
408                  PKCS7_R_SIGNING_NOT_SUPPORTED_FOR_THIS_KEY_TYPE);
409 err:
410         return 0;
411         }

413 PKCS7_SIGNER_INFO *PKCS7_add_signature(PKCS7 *p7, X509 *x509, EVP_PKEY *pkey,
414              const EVP_MD *dgst)
415         {
416         PKCS7_SIGNER_INFO *si = NULL;

418         if (dgst == NULL)
419                 {
420                 int def_nid;
421                 if (EVP_PKEY_get_default_digest_nid(pkey, &def_nid) <= 0)
422                         goto err;
423                 dgst = EVP_get_digestbynid(def_nid);
424                 if (dgst == NULL)
425                         {
426                         PKCS7err(PKCS7_F_PKCS7_ADD_SIGNATURE,
427                                  PKCS7_R_NO_DEFAULT_DIGEST);
428                         goto err;
429                         }
430                 }

432         if ((si=PKCS7_SIGNER_INFO_new()) == NULL) goto err;
433         if (!PKCS7_SIGNER_INFO_set(si,x509,pkey,dgst)) goto err;
434         if (!PKCS7_add_signer(p7,si)) goto err;
435         return(si);
436 err:
437         if (si)
438                 PKCS7_SIGNER_INFO_free(si);
439         return(NULL);
440         }

442 int PKCS7_set_digest(PKCS7 *p7, const EVP_MD *md)
443         {
444         if (PKCS7_type_is_digest(p7))
445                 {
446                 if(!(p7->d.digest->md->parameter = ASN1_TYPE_new()))
447                         {
448                         PKCS7err(PKCS7_F_PKCS7_SET_DIGEST,ERR_R_MALLOC_FAILURE);
449                         return 0;
450                         }
451                 p7->d.digest->md->parameter->type = V_ASN1_NULL;
452                 p7->d.digest->md->algorithm = OBJ_nid2obj(EVP_MD_nid(md));
453                 return 1;
454                 }

456         PKCS7err(PKCS7_F_PKCS7_SET_DIGEST,PKCS7_R_WRONG_CONTENT_TYPE);
457         return 1;
```



```
458         }

460 STACK_OF(PKCS7_SIGNER_INFO) *PKCS7_get_signer_info(PKCS7 *p7)
461         {
462         if (PKCS7_type_is_signed(p7))
463                 {
464                 return(p7->d.sign->signer_info);
465                 }
466         else if (PKCS7_type_is_signedAndEnveloped(p7))
467                 {
468                 return(p7->d.signed_and_enveloped->signer_info);
469                 }
470         else
471                 return(NULL);
472         }

474 void PKCS7_SIGNER_INFO_get0_algs(PKCS7_SIGNER_INFO *si, EVP_PKEY **pk,
475                                  X509_ALGOR **pdig, X509_ALGOR **psig)
476         {
477         if (pk)
478                 *pk = si->pkey;
479         if (pdig)
480                 *pdig = si->digest_alg;
481         if (psig)
482                 *psig = si->digest_enc_alg;
483         }

485 void PKCS7_RECIP_INFO_get0_alg(PKCS7_RECIP_INFO *ri, X509_ALGOR **penc)
486         {
487         if (penc)
488                 *penc = ri->key_enc_algor;
489         }

491 PKCS7_RECIP_INFO *PKCS7_add_recipient(PKCS7 *p7, X509 *x509)
492         {
493         PKCS7_RECIP_INFO *ri;

495         if ((ri=PKCS7_RECIP_INFO_new()) == NULL) goto err;
496         if (!PKCS7_RECIP_INFO_set(ri,x509)) goto err;
497         if (!PKCS7_add_recipient_info(p7,ri)) goto err;
498         return ri;
499 err:
500         if (ri)
501                 PKCS7_RECIP_INFO_free(ri);
502         return NULL;
503         }

505 int PKCS7_add_recipient_info(PKCS7 *p7, PKCS7_RECIP_INFO *ri)
506         {
507         int i;
508         STACK_OF(PKCS7_RECIP_INFO) *sk;

510         i=OBJ_obj2nid(p7->type);
511         switch (i)
512                 {
513         case NID_pkcs7_signedAndEnveloped:
514                 sk=     p7->d.signed_and_enveloped->recipientinfo;
515                 break;
516         case NID_pkcs7_enveloped:
517                 sk=     p7->d.enveloped->recipientinfo;
518                 break;
519         default:
520                 PKCS7err(PKCS7_F_PKCS7_ADD_RECIPIENT_INFO,PKCS7_R_WRONG_CONTENT_
521                 return(0);
522                 }
```

```
524         if (!sk_PKCS7_RECIP_INFO_push(sk,ri))
525                 return 0;
526         return(1);
527         }
528
529 int PKCS7_RECIP_INFO_set(PKCS7_RECIP_INFO *p7i, X509 *x509)
530         {
531         int ret;
532         EVP_PKEY *pkey = NULL;
533         if (!ASN1_INTEGER_set(p7i->version,0))
534                 return 0;
535         if (!X509_NAME_set(&p7i->issuer_and_serial->issuer,
536                 X509_get_issuer_name(x509)))
537                 return 0;
538
539         M_ASN1_INTEGER_free(p7i->issuer_and_serial->serial);
540         if (!(p7i->issuer_and_serial->serial=
541                 M_ASN1_INTEGER_dup(X509_get_serialNumber(x509))))
542                 return 0;
543
544         pkey = X509_get_pubkey(x509);
545
546         if (!pkey || !pkey->ameth || !pkey->ameth->pkey_ctrl)
547                 {
548                 PKCS7err(PKCS7_F_PKCS7_RECIP_INFO_SET,
549                         PKCS7_R_ENCRYPTION_NOT_SUPPORTED_FOR_THIS_KEY_TYPE);
550                 goto err;
551                 }
552
553         ret = pkey->ameth->pkey_ctrl(pkey, ASN1_PKEY_CTRL_PKCS7_ENCRYPT,
554                                                 0, p7i);
555         if (ret == -2)
556                 {
557                 PKCS7err(PKCS7_F_PKCS7_RECIP_INFO_SET,
558                         PKCS7_R_ENCRYPTION_NOT_SUPPORTED_FOR_THIS_KEY_TYPE);
559                 goto err;
560                 }
561         if (ret <= 0)
562                 {
563                 PKCS7err(PKCS7_F_PKCS7_RECIP_INFO_SET,
564                                 PKCS7_R_ENCRYPTION_CTRL_FAILURE);
565                 goto err;
566                 }
567
568         EVP_PKEY_free(pkey);
569
570         CRYPTO_add(&x509->references,1,CRYPTO_LOCK_X509);
571         p7i->cert=x509;
572
573         return 1;
574
575         err:
576         if (pkey)
577                 EVP_PKEY_free(pkey);
578         return 0;
579         }
580
581 X509 *PKCS7_cert_from_signer_info(PKCS7 *p7, PKCS7_SIGNER_INFO *si)
582         {
583         if (PKCS7_type_is_signed(p7))
584                 return(X509_find_by_issuer_and_serial(p7->d.sign->cert,
585                         si->issuer_and_serial->issuer,
586                         si->issuer_and_serial->serial));
587         else
588                 return(NULL);
589         }
```



```
590
591 int PKCS7_set_cipher(PKCS7 *p7, const EVP_CIPHER *cipher)
592         {
593         int i;
594         PKCS7_ENC_CONTENT *ec;
595
596         i=OBJ_obj2nid(p7->type);
597         switch (i)
598                 {
599         case NID_pkcs7_signedAndEnveloped:
600                 ec=p7->d.signed_and_enveloped->enc_data;
601                 break;
602         case NID_pkcs7_enveloped:
603                 ec=p7->d.enveloped->enc_data;
604                 break;
605         default:
606                 PKCS7err(PKCS7_F_PKCS7_SET_CIPHER,PKCS7_R_WRONG_CONTENT_TYPE);
607                 return(0);
608                 }
609
610         /* Check cipher OID exists and has data in it*/
611         i = EVP_CIPHER_type(cipher);
612         if(i == NID_undef) {
613                 PKCS7err(PKCS7_F_PKCS7_SET_CIPHER,PKCS7_R_CIPHER_HAS_NO_OBJECT_I
614                 return(0);
615         }
616
617         ec->cipher = cipher;
618         return 1;
619         }
620
621 int PKCS7_stream(unsigned char ***boundary, PKCS7 *p7)
622         {
623         ASN1_OCTET_STRING *os = NULL;
624
625         switch (OBJ_obj2nid(p7->type))
626                 {
627                 case NID_pkcs7_data:
628                 os = p7->d.data;
629                 break;
630
631                 case NID_pkcs7_signedAndEnveloped:
632                 os = p7->d.signed_and_enveloped->enc_data->enc_data;
633                 if (os == NULL)
634                         {
635                         os=M_ASN1_OCTET_STRING_new();
636                         p7->d.signed_and_enveloped->enc_data->enc_data=os;
637                         }
638                 break;
639
640                 case NID_pkcs7_enveloped:
641                 os = p7->d.enveloped->enc_data->enc_data;
642                 if (os == NULL)
643                         {
644                         os=M_ASN1_OCTET_STRING_new();
645                         p7->d.enveloped->enc_data->enc_data=os;
646                         }
647                 break;
648
649                 case NID_pkcs7_signed:
650                 os=p7->d.sign->contents->d.data;
651                 break;
652
653                 default:
654                 os = NULL;
655                 break;
```

```
656                 }

658         if (os == NULL)
659                 return 0;

661         os->flags |= ASN1_STRING_FLAG_NDEF;
662         *boundary = &os->data;

664         return 1;
665         }
666 #endif /* ! codereview */
```

```
**********************************************************
    3630 Wed Aug 13 19:53:04 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/pk7_mime.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pk7_mime.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
#include <openssl/rand.h>
#include <openssl/x509.h>
#include <openssl/asn1.h>
```



```
/* PKCS#7 wrappers round generalised stream and MIME routines */

int i2d_PKCS7_bio_stream(BIO *out, PKCS7 *p7, BIO *in, int flags)
        {
        return i2d_ASN1_bio_stream(out, (ASN1_VALUE *)p7, in, flags,
                                   ASN1_ITEM_rptr(PKCS7));
        }

int PEM_write_bio_PKCS7_stream(BIO *out, PKCS7 *p7, BIO *in, int flags)
        {
        return PEM_write_bio_ASN1_stream(out, (ASN1_VALUE *) p7, in, flags,
                                         "PKCS7",
                                         ASN1_ITEM_rptr(PKCS7));
        }

int SMIME_write_PKCS7(BIO *bio, PKCS7 *p7, BIO *data, int flags)
        {
        STACK_OF(X509_ALGOR) *mdalgs;
        int ctype_nid = OBJ_obj2nid(p7->type);
        if (ctype_nid == NID_pkcs7_signed)
                mdalgs = p7->d.sign->md_algs;
        else
                mdalgs = NULL;

        flags ^= SMIME_OLDMIME;


        return SMIME_write_ASN1(bio, (ASN1_VALUE *)p7, data, flags,
                                ctype_nid, NID_undef, mdalgs,
                                ASN1_ITEM_rptr(PKCS7));
        }

PKCS7 *SMIME_read_PKCS7(BIO *bio, BIO **bcont)
        {
        return (PKCS7 *)SMIME_read_ASN1(bio, bcont, ASN1_ITEM_rptr(PKCS7));
        }
#endif /* ! codereview */
```

```
**********************************************************
   15349 Wed Aug 13 19:53:04 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/pk7_smime.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pk7_smime.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 1999-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* Simple PKCS#7 processing functions */

#include <stdio.h>
```



```
#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/x509v3.h>

static int pkcs7_copy_existing_digest(PKCS7 *p7, PKCS7_SIGNER_INFO *si);

PKCS7 *PKCS7_sign(X509 *signcert, EVP_PKEY *pkey, STACK_OF(X509) *certs,
                  BIO *data, int flags)
{
        PKCS7 *p7;
        int i;

        if(!(p7 = PKCS7_new()))
                {
                PKCS7err(PKCS7_F_PKCS7_SIGN,ERR_R_MALLOC_FAILURE);
                return NULL;
                }

        if (!PKCS7_set_type(p7, NID_pkcs7_signed))
                goto err;

        if (!PKCS7_content_new(p7, NID_pkcs7_data))
                goto err;

        if (pkey && !PKCS7_sign_add_signer(p7, signcert, pkey, NULL, flags))
                {
                PKCS7err(PKCS7_F_PKCS7_SIGN,PKCS7_R_PKCS7_ADD_SIGNER_ERROR);
                goto err;
                }

        if(!(flags & PKCS7_NOCERTS))
                {
                for(i = 0; i < sk_X509_num(certs); i++)
                        {
                        if (!PKCS7_add_certificate(p7, sk_X509_value(certs, i)))
                                goto err;
                        }
                }

        if(flags & PKCS7_DETACHED)
                PKCS7_set_detached(p7, 1);

        if (flags & (PKCS7_STREAM|PKCS7_PARTIAL))
                return p7;

        if (PKCS7_final(p7, data, flags))
                return p7;

        err:
        PKCS7_free(p7);
        return NULL;
}

int PKCS7_final(PKCS7 *p7, BIO *data, int flags)
        {
        BIO *p7bio;
        int ret = 0;
        if (!(p7bio = PKCS7_dataInit(p7, NULL)))
                {
                PKCS7err(PKCS7_F_PKCS7_FINAL,ERR_R_MALLOC_FAILURE);
                return 0;
                }

        SMIME_crlf_copy(data, p7bio, flags);

        (void)BIO_flush(p7bio);
```

```
130         if (!PKCS7_dataFinal(p7,p7bio))
131                 {
132                 PKCS7err(PKCS7_F_PKCS7_FINAL,PKCS7_R_PKCS7_DATASIGN);
133                 goto err;
134                 }

136         ret = 1;

138         err:
139         BIO_free_all(p7bio);

141         return ret;

143         }

145 /* Check to see if a cipher exists and if so add S/MIME capabilities */

147 static int add_cipher_smcap(STACK_OF(X509_ALGOR) *sk, int nid, int arg)
148         {
149         if (EVP_get_cipherbynid(nid))
150                 return PKCS7_simple_smimecap(sk, nid, arg);
151         return 1;
152         }

154 static int add_digest_smcap(STACK_OF(X509_ALGOR) *sk, int nid, int arg)
155         {
156         if (EVP_get_digestbynid(nid))
157                 return PKCS7_simple_smimecap(sk, nid, arg);
158         return 1;
159         }

161 PKCS7_SIGNER_INFO *PKCS7_sign_add_signer(PKCS7 *p7, X509 *signcert,
162                                          EVP_PKEY *pkey, const EVP_MD *md,
163                                          int flags)
164         {
165         PKCS7_SIGNER_INFO *si = NULL;
166         STACK_OF(X509_ALGOR) *smcap = NULL;
167         if(!X509_check_private_key(signcert, pkey))
168                 {
169                 PKCS7err(PKCS7_F_PKCS7_SIGN_ADD_SIGNER,
170                         PKCS7_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE);
171                 return NULL;
172                 }

174         if (!(si = PKCS7_add_signature(p7,signcert,pkey, md)))
175                 {
176                 PKCS7err(PKCS7_F_PKCS7_SIGN_ADD_SIGNER,
177                                 PKCS7_R_PKCS7_ADD_SIGNATURE_ERROR);
178                 return NULL;
179                 }

181         if(!(flags & PKCS7_NOCERTS))
182                 {
183                 if (!PKCS7_add_certificate(p7, signcert))
184                         goto err;
185                 }

187         if(!(flags & PKCS7_NOATTR))
188                 {
189                 if (!PKCS7_add_attrib_content_type(si, NULL))
190                         goto err;
191                 /* Add SMIMECapabilities */
192                 if(!(flags & PKCS7_NOSMIMECAP))
193                         {
```



```
194                         if(!(smcap = sk_X509_ALGOR_new_null()))
195                                 {
196                                 PKCS7err(PKCS7_F_PKCS7_SIGN_ADD_SIGNER,
197                                         ERR_R_MALLOC_FAILURE);
198                                 goto err;
199                                 }
200                         if (!add_cipher_smcap(smcap, NID_aes_256_cbc, -1)
201                         || !add_digest_smcap(smcap, NID_id_GostR3411_94, -1)
202                         || !add_cipher_smcap(smcap, NID_id_Gost28147_89, -1)
203                                 || !add_cipher_smcap(smcap, NID_aes_192_cbc, -1)
204                                 || !add_cipher_smcap(smcap, NID_aes_128_cbc, -1)
205                         || !add_cipher_smcap(smcap, NID_des_ede3_cbc, -1)
206                                 || !add_cipher_smcap(smcap, NID_rc2_cbc, 128)
207                                 || !add_cipher_smcap(smcap, NID_rc2_cbc, 64)
208                                 || !add_cipher_smcap(smcap, NID_des_cbc, -1)
209                                 || !add_cipher_smcap(smcap, NID_rc2_cbc, 40)
210                                 || !PKCS7_add_attrib_smimecap (si, smcap))
211                                 goto err;
212                         sk_X509_ALGOR_pop_free(smcap, X509_ALGOR_free);
213                         smcap = NULL;
214                         }
215                 if (flags & PKCS7_REUSE_DIGEST)
216                         {
217                         if (!pkcs7_copy_existing_digest(p7, si))
218                                 goto err;
219                         if (!(flags & PKCS7_PARTIAL) &&
220                                         !PKCS7_SIGNER_INFO_sign(si))
221                                 goto err;
222                         }
223                 }
224         return si;
225         err:
226         if (smcap)
227                 sk_X509_ALGOR_pop_free(smcap, X509_ALGOR_free);
228         return NULL;
229         }

231 /* Search for a digest matching SignerInfo digest type and if found
232  * copy across.
233  */

235 static int pkcs7_copy_existing_digest(PKCS7 *p7, PKCS7_SIGNER_INFO *si)
236         {
237         int i;
238         STACK_OF(PKCS7_SIGNER_INFO) *sinfos;
239         PKCS7_SIGNER_INFO *sitmp;
240         ASN1_OCTET_STRING *osdig = NULL;
241         sinfos = PKCS7_get_signer_info(p7);
242         for (i = 0; i < sk_PKCS7_SIGNER_INFO_num(sinfos); i++)
243                 {
244                 sitmp = sk_PKCS7_SIGNER_INFO_value(sinfos, i);
245                 if (si == sitmp)
246                         break;
247                 if (sk_X509_ATTRIBUTE_num(sitmp->auth_attr) <= 0)
248                         continue;
249                 if (!OBJ_cmp(si->digest_alg->algorithm,
250                                 sitmp->digest_alg->algorithm))
251                         {
252                         osdig = PKCS7_digest_from_attributes(sitmp->auth_attr);
253                         break;
254                         }

256                 }

258         if (osdig)
259                 return PKCS7_add1_attrib_digest(si, osdig->data, osdig->length);
```

```
261         PKCS7err(PKCS7_F_PKCS7_COPY_EXISTING_DIGEST,
262                  PKCS7_R_NO_MATCHING_DIGEST_TYPE_FOUND);
263         return 0;
264         }

266 int PKCS7_verify(PKCS7 *p7, STACK_OF(X509) *certs, X509_STORE *store,
267                                         BIO *indata, BIO *out, int flags)
268 {
269         STACK_OF(X509) *signers;
270         X509 *signer;
271         STACK_OF(PKCS7_SIGNER_INFO) *sinfos;
272         PKCS7_SIGNER_INFO *si;
273         X509_STORE_CTX cert_ctx;
274         char buf[4096];
275         int i, j=0, k, ret = 0;
276         BIO *p7bio;
277         BIO *tmpin, *tmpout;

279         if(!p7) {
280                 PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_INVALID_NULL_POINTER);
281                 return 0;
282         }

284         if(!PKCS7_type_is_signed(p7)) {
285                 PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_WRONG_CONTENT_TYPE);
286                 return 0;
287         }

289         /* Check for no data and no content: no data to verify signature */
290         if(PKCS7_get_detached(p7) && !indata) {
291                 PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_NO_CONTENT);
292                 return 0;
293         }
294 #if 0
295         /* NB: this test commented out because some versions of Netscape
296          * illegally include zero length content when signing data.
297          */

299         /* Check for data and content: two sets of data */
300         if(!PKCS7_get_detached(p7) && indata) {
301                                 PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_CONTENT_AN
302                 return 0;
303         }
304 #endif

306         sinfos = PKCS7_get_signer_info(p7);

308         if(!sinfos || !sk_PKCS7_SIGNER_INFO_num(sinfos)) {
309                 PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_NO_SIGNATURES_ON_DATA);
310                 return 0;
311         }


314         signers = PKCS7_get0_signers(p7, certs, flags);

316         if(!signers) return 0;

318         /* Now verify the certificates */

320         if (!(flags & PKCS7_NOVERIFY)) for (k = 0; k < sk_X509_num(signers); k++
321                 signer = sk_X509_value (signers, k);
322                 if (!(flags & PKCS7_NOCHAIN)) {
323                         if(!X509_STORE_CTX_init(&cert_ctx, store, signer,
324                                                 p7->d.sign->cert))
325                                 {
```



```
326                                 PKCS7err(PKCS7_F_PKCS7_VERIFY,ERR_R_X509_LIB);
327                                 sk_X509_free(signers);
328                                 return 0;
329                                 }
330                         X509_STORE_CTX_set_default(&cert_ctx, "smime_sign");
331                 } else if(!X509_STORE_CTX_init (&cert_ctx, store, signer, NULL))
332                         PKCS7err(PKCS7_F_PKCS7_VERIFY,ERR_R_X509_LIB);
333                         sk_X509_free(signers);
334                         return 0;
335                 }
336                 if (!(flags & PKCS7_NOCRL))
337                         X509_STORE_CTX_set0_crls(&cert_ctx, p7->d.sign->crl);
338                 i = X509_verify_cert(&cert_ctx);
339                 if (i <= 0) j = X509_STORE_CTX_get_error(&cert_ctx);
340                 X509_STORE_CTX_cleanup(&cert_ctx);
341                 if (i <= 0) {
342                         PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_CERTIFICATE_VERIFY
343                         ERR_add_error_data(2, "Verify error:",
344                                            X509_verify_cert_error_string(j));
345                         sk_X509_free(signers);
346                         return 0;
347                 }
348                 /* Check for revocation status here */
349         }

351         /* Performance optimization: if the content is a memory BIO then
352          * store its contents in a temporary read only memory BIO. This
353          * avoids potentially large numbers of slow copies of data which will
354          * occur when reading from a read write memory BIO when signatures
355          * are calculated.
356          */

358         if (indata && (BIO_method_type(indata) == BIO_TYPE_MEM))
359                 {
360                 char *ptr;
361                 long len;
362                 len = BIO_get_mem_data(indata, &ptr);
363                 tmpin = BIO_new_mem_buf(ptr, len);
364                 if (tmpin == NULL)
365                         {
366                         PKCS7err(PKCS7_F_PKCS7_VERIFY,ERR_R_MALLOC_FAILURE);
367                         return 0;
368                         }
369                 }
370         else
371                 tmpin = indata;


374         if (!(p7bio=PKCS7_dataInit(p7,tmpin)))
375                 goto err;

377         if(flags & PKCS7_TEXT) {
378                 if(!(tmpout = BIO_new(BIO_s_mem()))) {
379                         PKCS7err(PKCS7_F_PKCS7_VERIFY,ERR_R_MALLOC_FAILURE);
380                         goto err;
381                 }
382                 BIO_set_mem_eof_return(tmpout, 0);
383         } else tmpout = out;

385         /* We now have to 'read' from p7bio to calculate digests etc. */
386         for (;;)
387         {
388                 i=BIO_read(p7bio,buf,sizeof(buf));
389                 if (i <= 0) break;
390                 if (tmpout) BIO_write(tmpout, buf, i);
391         }
```

```
393         if(flags & PKCS7_TEXT) {
394                 if(!SMIME_text(tmpout, out)) {
395                         PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_SMIME_TEXT_ERROR);
396                         BIO_free(tmpout);
397                         goto err;
398                 }
399                 BIO_free(tmpout);
400         }

402         /* Now Verify All Signatures */
403         if (!(flags & PKCS7_NOSIGS))
404             for (i=0; i<sk_PKCS7_SIGNER_INFO_num(sinfos); i++)
405                 {
406                 si=sk_PKCS7_SIGNER_INFO_value(sinfos,i);
407                 signer = sk_X509_value (signers, i);
408                 j=PKCS7_signatureVerify(p7bio,p7,si, signer);
409                 if (j <= 0) {
410                         PKCS7err(PKCS7_F_PKCS7_VERIFY,PKCS7_R_SIGNATURE_FAILURE)
411                         goto err;
412                 }
413         }

415         ret = 1;

417         err:

419         if (tmpin == indata)
420                 {
421                 if (indata) BIO_pop(p7bio);
422                 }
423         BIO_free_all(p7bio);

425         sk_X509_free(signers);

427         return ret;
428 }

430 STACK_OF(X509) *PKCS7_get0_signers(PKCS7 *p7, STACK_OF(X509) *certs, int flags)
431 {
432         STACK_OF(X509) *signers;
433         STACK_OF(PKCS7_SIGNER_INFO) *sinfos;
434         PKCS7_SIGNER_INFO *si;
435         PKCS7_ISSUER_AND_SERIAL *ias;
436         X509 *signer;
437         int i;

439         if(!p7) {
440                 PKCS7err(PKCS7_F_PKCS7_GET0_SIGNERS,PKCS7_R_INVALID_NULL_POINTER
441                 return NULL;
442         }

444         if(!PKCS7_type_is_signed(p7)) {
445                 PKCS7err(PKCS7_F_PKCS7_GET0_SIGNERS,PKCS7_R_WRONG_CONTENT_TYPE);
446                 return NULL;
447         }

449         /* Collect all the signers together */

451         sinfos = PKCS7_get_signer_info(p7);

453         if(sk_PKCS7_SIGNER_INFO_num(sinfos) <= 0) {
454                 PKCS7err(PKCS7_F_PKCS7_GET0_SIGNERS,PKCS7_R_NO_SIGNERS);
455                 return 0;
456         }
```



```
458         if(!(signers = sk_X509_new_null())) {
459                 PKCS7err(PKCS7_F_PKCS7_GET0_SIGNERS,ERR_R_MALLOC_FAILURE);
460                 return NULL;
461         }

463         for (i = 0; i < sk_PKCS7_SIGNER_INFO_num(sinfos); i++)
464         {
465             si = sk_PKCS7_SIGNER_INFO_value(sinfos, i);
466             ias = si->issuer_and_serial;
467             signer = NULL;
468                 /* If any certificates passed they take priority */
469             if (certs) signer = X509_find_by_issuer_and_serial (certs,
470                                                 ias->issuer, ias->serial);
471             if (!signer && !(flags & PKCS7_NOINTERN)
472                         && p7->d.sign->cert) signer =
473                               X509_find_by_issuer_and_serial (p7->d.sign->cert,
474                                                 ias->issuer, ias->serial);
475             if (!signer) {
476                         PKCS7err(PKCS7_F_PKCS7_GET0_SIGNERS,PKCS7_R_SIGNER_CERTI
477                         sk_X509_free(signers);
478                         return 0;
479             }

481             if (!sk_X509_push(signers, signer)) {
482                 sk_X509_free(signers);
483                 return NULL;
484             }
485         }
486         return signers;
487 }


490 /* Build a complete PKCS#7 enveloped data */

492 PKCS7 *PKCS7_encrypt(STACK_OF(X509) *certs, BIO *in, const EVP_CIPHER *cipher,
493                                                                 int flags)
494 {
495         PKCS7 *p7;
496         BIO *p7bio = NULL;
497         int i;
498         X509 *x509;
499         if(!(p7 = PKCS7_new())) {
500                 PKCS7err(PKCS7_F_PKCS7_ENCRYPT,ERR_R_MALLOC_FAILURE);
501                 return NULL;
502         }

504         if (!PKCS7_set_type(p7, NID_pkcs7_enveloped))
505                 goto err;
506         if (!PKCS7_set_cipher(p7, cipher)) {
507                 PKCS7err(PKCS7_F_PKCS7_ENCRYPT,PKCS7_R_ERROR_SETTING_CIPHER);
508                 goto err;
509         }

511         for(i = 0; i < sk_X509_num(certs); i++) {
512                 x509 = sk_X509_value(certs, i);
513                 if(!PKCS7_add_recipient(p7, x509)) {
514                         PKCS7err(PKCS7_F_PKCS7_ENCRYPT,
515                                                 PKCS7_R_ERROR_ADDING_RECIPIENT);
516                         goto err;
517                 }
518         }

520         if (flags & PKCS7_STREAM)
521                 return p7;

523         if (PKCS7_final(p7, in, flags))
```

```
524                 return p7;

526         err:

528         BIO_free_all(p7bio);
529         PKCS7_free(p7);
530         return NULL;

532 }

534 int PKCS7_decrypt(PKCS7 *p7, EVP_PKEY *pkey, X509 *cert, BIO *data, int flags)
535 {
536         BIO *tmpmem;
537         int ret, i;
538         char buf[4096];

540         if(!p7) {
541                 PKCS7err(PKCS7_F_PKCS7_DECRYPT,PKCS7_R_INVALID_NULL_POINTER);
542                 return 0;
543         }

545         if(!PKCS7_type_is_enveloped(p7)) {
546                 PKCS7err(PKCS7_F_PKCS7_DECRYPT,PKCS7_R_WRONG_CONTENT_TYPE);
547                 return 0;
548         }

550         if(cert && !X509_check_private_key(cert, pkey)) {
551                 PKCS7err(PKCS7_F_PKCS7_DECRYPT,
552                                 PKCS7_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE);
553                 return 0;
554         }

556         if(!(tmpmem = PKCS7_dataDecode(p7, pkey, NULL, cert))) {
557                 PKCS7err(PKCS7_F_PKCS7_DECRYPT, PKCS7_R_DECRYPT_ERROR);
558                 return 0;
559         }

561         if (flags & PKCS7_TEXT) {
562                 BIO *tmpbuf, *bread;
563                 /* Encrypt BIOs can't do BIO_gets() so add a buffer BIO */
564                 if(!(tmpbuf = BIO_new(BIO_f_buffer()))) {
565                         PKCS7err(PKCS7_F_PKCS7_DECRYPT, ERR_R_MALLOC_FAILURE);
566                         BIO_free_all(tmpmem);
567                         return 0;
568                 }
569                 if(!(bread = BIO_push(tmpbuf, tmpmem))) {
570                         PKCS7err(PKCS7_F_PKCS7_DECRYPT, ERR_R_MALLOC_FAILURE);
571                         BIO_free_all(tmpbuf);
572                         BIO_free_all(tmpmem);
573                         return 0;
574                 }
575                 ret = SMIME_text(bread, data);
576                 if (ret > 0 && BIO_method_type(tmpmem) == BIO_TYPE_CIPHER)
577                         {
578                         if (!BIO_get_cipher_status(tmpmem))
579                                 ret = 0;
580                         }
581                 BIO_free_all(bread);
582                 return ret;
583         } else {
584                 for(;;) {
585                         i = BIO_read(tmpmem, buf, sizeof(buf));
586                         if(i <= 0)
587                                 {
588                                 ret = 1;
589                                 if (BIO_method_type(tmpmem) == BIO_TYPE_CIPHER)
```



```
590                                     {
591                                     if (!BIO_get_cipher_status(tmpmem))
592                                             ret = 0;
593                                     }

595                                 break;
596                                 }
597                         if (BIO_write(data, buf, i) != i)
598                                 {
599                                 ret = 0;
600                                 break;
601                                 }
602                 }
603                 BIO_free_all(tmpmem);
604                 return ret;
605         }
606 }
607 #endif /* ! codereview */
```

```
**********************************************************
    9843 Wed Aug 13 19:53:04 2014
new/usr/src/lib/openssl/libsunw_crypto/pkcs7/pkcs7err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pkcs7/pkcs7err.c */
/* ====================================================================
 * Copyright (c) 1999-2014 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/pkcs7.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_PKCS7,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_PKCS7,0,reason)

static ERR_STRING_DATA PKCS7_str_functs[]=
        {
{ERR_FUNC(PKCS7_F_B64_READ_PKCS7),      "B64_READ_PKCS7"},
{ERR_FUNC(PKCS7_F_B64_WRITE_PKCS7),     "B64_WRITE_PKCS7"},
{ERR_FUNC(PKCS7_F_DO_PKCS7_SIGNED_ATTRIB),      "DO_PKCS7_SIGNED_ATTRIB"},
{ERR_FUNC(PKCS7_F_I2D_PKCS7_BIO_STREAM),        "i2d_PKCS7_bio_stream"},
{ERR_FUNC(PKCS7_F_PKCS7_ADD0_ATTRIB_SIGNING_TIME),      "PKCS7_add0_attrib_signi
{ERR_FUNC(PKCS7_F_PKCS7_ADD_ATTRIB_SMIMECAP),   "PKCS7_add_attrib_smimecap"},
{ERR_FUNC(PKCS7_F_PKCS7_ADD_CERTIFICATE),       "PKCS7_add_certificate"},
{ERR_FUNC(PKCS7_F_PKCS7_ADD_CRL),       "PKCS7_add_crl"},
{ERR_FUNC(PKCS7_F_PKCS7_ADD_RECIPIENT_INFO),    "PKCS7_add_recipient_info"},
{ERR_FUNC(PKCS7_F_PKCS7_ADD_SIGNATURE), "PKCS7_add_signature"},
{ERR_FUNC(PKCS7_F_PKCS7_ADD_SIGNER),    "PKCS7_add_signer"},
{ERR_FUNC(PKCS7_F_PKCS7_BIO_ADD_DIGEST),        "PKCS7_BIO_ADD_DIGEST"},
{ERR_FUNC(PKCS7_F_PKCS7_COPY_EXISTING_DIGEST),  "PKCS7_COPY_EXISTING_DIGEST"},
{ERR_FUNC(PKCS7_F_PKCS7_CTRL),  "PKCS7_ctrl"},
{ERR_FUNC(PKCS7_F_PKCS7_DATADECODE),    "PKCS7_dataDecode"},
{ERR_FUNC(PKCS7_F_PKCS7_DATAFINAL),     "PKCS7_dataFinal"},
{ERR_FUNC(PKCS7_F_PKCS7_DATAINIT),      "PKCS7_dataInit"},
{ERR_FUNC(PKCS7_F_PKCS7_DATASIGN),      "PKCS7_DATASIGN"},
{ERR_FUNC(PKCS7_F_PKCS7_DATAVERIFY),    "PKCS7_dataVerify"},
{ERR_FUNC(PKCS7_F_PKCS7_DECRYPT),       "PKCS7_decrypt"},
{ERR_FUNC(PKCS7_F_PKCS7_DECRYPT_RINFO), "PKCS7_DECRYPT_RINFO"},
{ERR_FUNC(PKCS7_F_PKCS7_ENCODE_RINFO),  "PKCS7_ENCODE_RINFO"},
{ERR_FUNC(PKCS7_F_PKCS7_ENCRYPT),       "PKCS7_encrypt"},
{ERR_FUNC(PKCS7_F_PKCS7_FINAL), "PKCS7_final"},
{ERR_FUNC(PKCS7_F_PKCS7_FIND_DIGEST),   "PKCS7_FIND_DIGEST"},
{ERR_FUNC(PKCS7_F_PKCS7_GET0_SIGNERS),  "PKCS7_get0_signers"},
{ERR_FUNC(PKCS7_F_PKCS7_RECIP_INFO_SET),        "PKCS7_RECIP_INFO_set"},
{ERR_FUNC(PKCS7_F_PKCS7_SET_CIPHER),    "PKCS7_set_cipher"},
{ERR_FUNC(PKCS7_F_PKCS7_SET_CONTENT),   "PKCS7_set_content"},
{ERR_FUNC(PKCS7_F_PKCS7_SET_DIGEST),    "PKCS7_set_digest"},
{ERR_FUNC(PKCS7_F_PKCS7_SET_TYPE),      "PKCS7_set_type"},
{ERR_FUNC(PKCS7_F_PKCS7_SIGN),  "PKCS7_sign"},
{ERR_FUNC(PKCS7_F_PKCS7_SIGNATUREVERIFY),       "PKCS7_signatureVerify"},
{ERR_FUNC(PKCS7_F_PKCS7_SIGNER_INFO_SET),       "PKCS7_SIGNER_INFO_set"},
{ERR_FUNC(PKCS7_F_PKCS7_SIGNER_INFO_SIGN),      "PKCS7_SIGNER_INFO_sign"},
{ERR_FUNC(PKCS7_F_PKCS7_SIGN_ADD_SIGNER),       "PKCS7_sign_add_signer"},
{ERR_FUNC(PKCS7_F_PKCS7_SIMPLE_SMIMECAP),       "PKCS7_simple_smimecap"},
{ERR_FUNC(PKCS7_F_PKCS7_VERIFY),        "PKCS7_verify"},
{ERR_FUNC(PKCS7_F_SMIME_READ_PKCS7),    "SMIME_read_PKCS7"},
{ERR_FUNC(PKCS7_F_SMIME_TEXT),  "SMIME_text"},
{0,NULL}
        };

static ERR_STRING_DATA PKCS7_str_reasons[]=
        {
{ERR_REASON(PKCS7_R_CERTIFICATE_VERIFY_ERROR),"certificate verify error"},
{ERR_REASON(PKCS7_R_CIPHER_HAS_NO_OBJECT_IDENTIFIER),"cipher has no object ident
{ERR_REASON(PKCS7_R_CIPHER_NOT_INITIALIZED),"cipher not initialized"},
{ERR_REASON(PKCS7_R_CONTENT_AND_DATA_PRESENT),"content and data present"},
{ERR_REASON(PKCS7_R_CTRL_ERROR)          ,"ctrl error"},
{ERR_REASON(PKCS7_R_DECODE_ERROR)        ,"decode error"},
{ERR_REASON(PKCS7_R_DECRYPTED_KEY_IS_WRONG_LENGTH),"decrypted key is wrong lengt
{ERR_REASON(PKCS7_R_DECRYPT_ERROR)       ,"decrypt error"},
{ERR_REASON(PKCS7_R_DIGEST_FAILURE)      ,"digest failure"},
{ERR_REASON(PKCS7_R_ENCRYPTION_CTRL_FAILURE),"encryption ctrl failure"},
```

```
{ERR_REASON(PKCS7_R_ENCRYPTION_NOT_SUPPORTED_FOR_THIS_KEY_TYPE),"encryption not
{ERR_REASON(PKCS7_R_ERROR_ADDING_RECIPIENT),"error adding recipient"},
{ERR_REASON(PKCS7_R_ERROR_SETTING_CIPHER),"error setting cipher"},
{ERR_REASON(PKCS7_R_INVALID_MIME_TYPE)   ,"invalid mime type"},
{ERR_REASON(PKCS7_R_INVALID_NULL_POINTER),"invalid null pointer"},
{ERR_REASON(PKCS7_R_INVALID_SIGNED_DATA_TYPE),"invalid signed data type"},
{ERR_REASON(PKCS7_R_MIME_NO_CONTENT_TYPE),"mime no content type"},
{ERR_REASON(PKCS7_R_MIME_PARSE_ERROR)    ,"mime parse error"},
{ERR_REASON(PKCS7_R_MIME_SIG_PARSE_ERROR),"mime sig parse error"},
{ERR_REASON(PKCS7_R_MISSING_CERIPEND_INFO),"missing ceripend info"},
{ERR_REASON(PKCS7_R_NO_CONTENT)          ,"no content"},
{ERR_REASON(PKCS7_R_NO_CONTENT_TYPE)     ,"no content type"},
{ERR_REASON(PKCS7_R_NO_DEFAULT_DIGEST)   ,"no default digest"},
{ERR_REASON(PKCS7_R_NO_MATCHING_DIGEST_TYPE_FOUND),"no matching digest type foun
{ERR_REASON(PKCS7_R_NO_MULTIPART_BODY_FAILURE),"no multipart body failure"},
{ERR_REASON(PKCS7_R_NO_MULTIPART_BOUNDARY),"no multipart boundary"},
{ERR_REASON(PKCS7_R_NO_RECIPIENT_MATCHES_CERTIFICATE),"no recipient matches cert
{ERR_REASON(PKCS7_R_NO_RECIPIENT_MATCHES_KEY),"no recipient matches key"},
{ERR_REASON(PKCS7_R_NO_SIGNATURES_ON_DATA),"no signatures on data"},
{ERR_REASON(PKCS7_R_NO_SIGNERS)          ,"no signers"},
{ERR_REASON(PKCS7_R_NO_SIG_CONTENT_TYPE) ,"no sig content type"},
{ERR_REASON(PKCS7_R_OPERATION_NOT_SUPPORTED_ON_THIS_TYPE),"operation not support
{ERR_REASON(PKCS7_R_PKCS7_ADD_SIGNATURE_ERROR),"pkcs7 add signature error"},
{ERR_REASON(PKCS7_R_PKCS7_ADD_SIGNER_ERROR),"pkcs7 add signer error"},
{ERR_REASON(PKCS7_R_PKCS7_DATAFINAL)     ,"pkcs7 datafinal"},
{ERR_REASON(PKCS7_R_PKCS7_DATAFINAL_ERROR),"pkcs7 datafinal error"},
{ERR_REASON(PKCS7_R_PKCS7_DATASIGN)      ,"pkcs7 datasign"},
{ERR_REASON(PKCS7_R_PKCS7_PARSE_ERROR)   ,"pkcs7 parse error"},
{ERR_REASON(PKCS7_R_PKCS7_SIG_PARSE_ERROR),"pkcs7 sig parse error"},
{ERR_REASON(PKCS7_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE),"private key does no
{ERR_REASON(PKCS7_R_SIGNATURE_FAILURE)   ,"signature failure"},
{ERR_REASON(PKCS7_R_SIGNER_CERTIFICATE_NOT_FOUND),"signer certificate not found"
{ERR_REASON(PKCS7_R_SIGNING_CTRL_FAILURE),"signing ctrl failure"},
{ERR_REASON(PKCS7_R_SIGNING_NOT_SUPPORTED_FOR_THIS_KEY_TYPE),"signing not suppor
{ERR_REASON(PKCS7_R_SIG_INVALID_MIME_TYPE),"sig invalid mime type"},
{ERR_REASON(PKCS7_R_SMIME_TEXT_ERROR)    ,"smime text error"},
{ERR_REASON(PKCS7_R_UNABLE_TO_FIND_CERTIFICATE),"unable to find certificate"},
{ERR_REASON(PKCS7_R_UNABLE_TO_FIND_MEM_BIO),"unable to find mem bio"},
{ERR_REASON(PKCS7_R_UNABLE_TO_FIND_MESSAGE_DIGEST),"unable to find message diges
{ERR_REASON(PKCS7_R_UNKNOWN_DIGEST_TYPE) ,"unknown digest type"},
{ERR_REASON(PKCS7_R_UNKNOWN_OPERATION)   ,"unknown operation"},
{ERR_REASON(PKCS7_R_UNSUPPORTED_CIPHER_TYPE),"unsupported cipher type"},
{ERR_REASON(PKCS7_R_UNSUPPORTED_CONTENT_TYPE),"unsupported content type"},
{ERR_REASON(PKCS7_R_WRONG_CONTENT_TYPE)  ,"wrong content type"},
{ERR_REASON(PKCS7_R_WRONG_PKCS7_TYPE)    ,"wrong pkcs7 type"},
{0,NULL}
        };

#endif

void ERR_load_PKCS7_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(PKCS7_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,PKCS7_str_functs);
                ERR_load_strings(0,PKCS7_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
  103475 Wed Aug 13 19:53:04 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/aes-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# Version 4.3.
#
# You might fail to appreciate this module performance from the first
# try. If compared to "vanilla" linux-ia32-icc target, i.e. considered
# to be *the* best Intel C compiler without -KPIC, performance appears
# to be virtually identical... But try to re-configure with shared
# library support... Aha! Intel compiler "suddenly" lags behind by 30%
# [on P4, more on others]:-) And if compared to position-independent
# code generated by GNU C, this code performs *more* than *twice* as
# fast! Yes, all this buzz about PIC means that unlike other hand-
# coded implementations, this one was explicitly designed to be safe
# to use even in shared library context... This also means that this
# code isn't necessarily absolutely fastest "ever," because in order
# to achieve position independence an extra register has to be
# off-loaded to stack, which affects the benchmark result.
#
# Special note about instruction choice. Do you recall RC4_INT code
# performing poorly on P4? It might be the time to figure out why.
# RC4_INT code implies effective address calculations in base+offset*4
# form. Trouble is that it seems that offset scaling turned to be
# critical path... At least eliminating scaling resulted in 2.8x RC4
# performance improvement [as you might recall]. As AES code is hungry
# for scaling too, I [try to] avoid the latter by favoring off-by-2
# shifts and masking the result with 0xFF<<2 instead of "boring" 0xFF.
#
# As was shown by Dean Gaudet <dean@arctic.org>, the above note turned
# void. Performance improvement with off-by-2 shifts was observed on
# intermediate implementation, which was spilling yet another register
# to stack... Final offset*4 code below runs just a tad faster on P4,
# but exhibits up to 10% improvement on other cores.
#
# Second version is "monolithic" replacement for aes_core.c, which in
# addition to AES_[de|en]crypt implements private_AES_set_[de|en]cryption_key.
# This made it possible to implement little-endian variant of the
# algorithm without modifying the base C code. Motivating factor for
# the undertaken effort was that it appeared that in tight IA-32
# register window little-endian flavor could achieve slightly higher
# Instruction Level Parallelism, and it indeed resulted in up to 15%
# better performance on most recent µ-archs...
#
# Third version adds AES_cbc_encrypt implementation, which resulted in
# up to 40% performance imrovement of CBC benchmark results. 40% was
# observed on P4 core, where "overall" imrovement coefficient, i.e. if
# compared to PIC generated by GCC and in CBC mode, was observed to be
# as large as 4x:-) CBC performance is virtually identical to ECB now
# and on some platforms even better, e.g. 17.6 "small" cycles/byte on
# Opteron, because certain function prologues and epilogues are
# effectively taken out of the loop...
#
# Version 3.2 implements compressed tables and prefetch of these tables
# in CBC[!] mode. Former means that 3/4 of table references are now
# misaligned, which unfortunately has negative impact on elder IA-32
```



```
# implementations, Pentium suffered 30% penalty, PIII - 10%.
#
# Version 3.3 avoids L1 cache aliasing between stack frame and
# S-boxes, and 3.4 - L1 cache aliasing even between key schedule. The
# latter is achieved by copying the key schedule to controlled place in
# stack. This unfortunately has rather strong impact on small block CBC
# performance, ~2x deterioration on 16-byte block if compared to 3.3.
#
# Version 3.5 checks if there is L1 cache aliasing between user-supplied
# key schedule and S-boxes and abstains from copying the former if
# there is no. This allows end-user to consciously retain small block
# performance by aligning key schedule in specific manner.
#
# Version 3.6 compresses Td4 to 256 bytes and prefetches it in ECB.
#
# Current ECB performance numbers for 128-bit key in CPU cycles per
# processed byte [measure commonly used by AES benchmarkers] are:
#
#               small footprint         fully unrolled
# P4            24                      22
# AMD K8        20                      19
# PIII          25                      23
# Pentium       81                      78
#
# Version 3.7 reimplements outer rounds as "compact." Meaning that
# first and last rounds reference compact 256 bytes S-box. This means
# that first round consumes a lot more CPU cycles and that encrypt
# and decrypt performance becomes asymmetric. Encrypt performance
# drops by 10-12%, while decrypt - by 20-25%:-( 256 bytes S-box is
# aggressively pre-fetched.
#
# Version 4.0 effectively rolls back to 3.6 and instead implements
# additional set of functions, _[x86|sse]_AES_[en|de]crypt_compact,
# which use exclusively 256 byte S-box. These functions are to be
# called in modes not concealing plain text, such as ECB, or when
# we're asked to process smaller amount of data [or unconditionally
# on hyper-threading CPU]. Currently it's called unconditionally from
# AES_[en|de]crypt, which affects all modes, but CBC. CBC routine
# still needs to be modified to switch between slower and faster
# mode when appropriate... But in either case benchmark landscape
# changes dramatically and below numbers are CPU cycles per processed
# byte for 128-bit key.
#
#               ECB encrypt     ECB decrypt     CBC large chunk
# P4            56[60]          84[100]         23
# AMD K8        48[44]          70[79]          18
# PIII          41[50]          61[91]          24
# Core 2        32[38]          45[70]          18.5
# Pentium       120             160             77
#
# Version 4.1 switches to compact S-box even in key schedule setup.
#
# Version 4.2 prefetches compact S-box in every SSE round or in other
# words every cache-line is *guaranteed* to be accessed within ~50
# cycles window. Why just SSE? Because it's needed on hyper-threading
# CPU! Which is also why it's prefetched with 64 byte stride. Best
# part is that it has no negative effect on performance:-)
#
# Version 4.3 implements switch between compact and non-compact block
# functions in AES_cbc_encrypt depending on how much data was asked
# to be processed in one stroke.
#
##############################################################################
# Timing attacks are classified in two classes: synchronous when
# attacker consciously initiates cryptographic operation and collects
# timing data of various character afterwards, and asynchronous when
```

```
# malicious code is executed on same CPU simultaneously with AES,
# instruments itself and performs statistical analysis of this data.
#
# As far as synchronous attacks go the root to the AES timing
# vulnerability is twofold. Firstly, of 256 S-box elements at most 160
# are referred to in single 128-bit block operation. Well, in C
# implementation with 4 distinct tables it's actually as little as 40
# references per 256 elements table, but anyway... Secondly, even
# though S-box elements are clustered into smaller amount of cache-
# lines, smaller than 160 and even 40, it turned out that for certain
# plain-text pattern[s] or simply put chosen plain-text and given key
# few cache-lines remain unaccessed during block operation. Now, if
# attacker can figure out this access pattern, he can deduct the key
# [or at least part of it]. The natural way to mitigate this kind of
# attacks is to minimize the amount of cache-lines in S-box and/or
# prefetch them to ensure that every one is accessed for more uniform
# timing. But note that *if* plain-text was concealed in such way that
# input to block function is distributed *uniformly*, then attack
# wouldn't apply. Now note that some encryption modes, most notably
# CBC, do mask the plain-text in this exact way [secure cipher output
# is distributed uniformly]. Yes, one still might find input that
# would reveal the information about given key, but if amount of
# candidate inputs to be tried is larger than amount of possible key
# combinations then attack becomes infeasible. This is why revised
# AES_cbc_encrypt "dares" to switch to larger S-box when larger chunk
# of data is to be processed in one stroke. The current size limit of
# 512 bytes is chosen to provide same [diminishigly low] probability
# for cache-line to remain untouched in large chunk operation with
# large S-box as for single block operation with compact S-box and
# surely needs more careful consideration...
#
# As for asynchronous attacks. There are two flavours: attacker code
# being interleaved with AES on hyper-threading CPU at *instruction*
# level, and two processes time sharing single core. As for latter.
# Two vectors. 1. Given that attacker process has higher priority,
# yield execution to process performing AES just before timer fires
# off the scheduler, immediately regain control of CPU and analyze the
# cache state. For this attack to be efficient attacker would have to
# effectively slow down the operation by several *orders* of magnitute,
# by ratio of time slice to duration of handful of AES rounds, which
# unlikely to remain unnoticed. Not to mention that this also means
# that he would spend correspondigly more time to collect enough
# statistical data to mount the attack. It's probably appropriate to
# say that if adeversary reckons that this attack is beneficial and
# risks to be noticed, you probably have larger problems having him
# mere opportunity. In other words suggested code design expects you
# to preclude/mitigate this attack by overall system security design.
# 2. Attacker manages to make his code interrupt driven. In order for
# this kind of attack to be feasible, interrupt rate has to be high
# enough, again comparable to duration of handful of AES rounds. But
# is there interrupt source of such rate? Hardly, not even 1Gbps NIC
# generates interrupts at such raging rate...
#
# And now back to the former, hyper-threading CPU or more specifically
# Intel P4. Recall that asynchronous attack implies that malicious
# code instruments itself. And naturally instrumentation granularity
# has be noticeably lower than duration of codepath accessing S-box.
# Given that all cache-lines are accessed during that time that is.
# Current implementation accesses *all* cache-lines within ~50 cycles
# window, which is actually *less* than RDTSC latency on Intel P4!

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],"aes-586.pl",$x86only = $ARGV[$#ARGV] eq "386");
```



```
&static_label("AES_Te");
&static_label("AES_Td");

$s0="eax";
$s1="ebx";
$s2="ecx";
$s3="edx";
$key="edi";
$acc="esi";
$tbl="ebp";

# stack frame layout in _[x86|sse]_AES_* routines, frame is allocated
# by caller
$__ra=&DWP(0,"esp");    # return address
$__s0=&DWP(4,"esp");    # s0 backing store
$__s1=&DWP(8,"esp");    # s1 backing store
$__s2=&DWP(12,"esp");   # s2 backing store
$__s3=&DWP(16,"esp");   # s3 backing store
$__key=&DWP(20,"esp");  # pointer to key schedule
$__end=&DWP(24,"esp");  # pointer to end of key schedule
$__tbl=&DWP(28,"esp");  # %ebp backing store

# stack frame layout in AES_[en|crypt] routines, which differs from
# above by 4 and overlaps by %ebp backing store
$_tbl=&DWP(24,"esp");
$_esp=&DWP(28,"esp");

sub _data_word() { my $i; while(defined($i=shift)) { &data_word($i,$i); } }

$speed_limit=512;       # chunks smaller than $speed_limit are
                        # processed with compact routine in CBC mode
$small_footprint=1;     # $small_footprint=1 code is ~5% slower [on
                        # recent µ-archs], but ~5 times smaller!
                        # I favor compact code to minimize cache
                        # contention and in hope to "collect" 5% back
                        # in real-life applications...

$vertical_spin=0;       # shift "verticaly" defaults to 0, because of
                        # its proof-of-concept status...
# Note that there is no decvert(), as well as last encryption round is
# performed with "horizontal" shifts. This is because this "vertical"
# implementation [one which groups shifts on a given $s[i] to form a
# "column," unlike "horizontal" one, which groups shifts on different
# $s[i] to form a "row"] is work in progress. It was observed to run
# few percents faster on Intel cores, but not AMD. On AMD K8 core it's
# whole 12% slower:-( So we face a trade-off... Shall it be resolved
# some day? Till then the code is considered experimental and by
# default remains dormant...

sub encvert()
{ my ($te,@s) = @_;
  my $v0 = $acc, $v1 = $key;

        &mov    ($v0,$s[3]);                            # copy s3
        &mov    (&DWP(4,"esp"),$s[2]);                  # save s2
        &mov    ($v1,$s[0]);                            # copy s0
        &mov    (&DWP(8,"esp"),$s[1]);                  # save s1

        &movz   ($s[2],&HB($s[0]));
        &and    ($s[0],0xFF);
        &mov    ($s[0],&DWP(0,$te,$s[0],8));            # s0>>0
        &shr    ($v1,16);
        &mov    ($s[3],&DWP(3,$te,$s[2],8));            # s0>>8
        &movz   ($s[1],&HB($v1));
        &and    ($v1,0xFF);
        &mov    ($s[2],&DWP(2,$te,$v1,8));              # s0>>16
```

```
260          &mov   ($v1,$v0);
261         &mov    ($s[1],&DWP(1,$te,$s[1],8));            # s0>>24

263         &and    ($v0,0xFF);
264         &xor    ($s[3],&DWP(0,$te,$v0,8));              # s3>>0
265         &movz   ($v0,&HB($v1));
266         &shr    ($v1,16);
267         &xor    ($s[2],&DWP(3,$te,$v0,8));              # s3>>8
268         &movz   ($v0,&HB($v1));
269         &and    ($v1,0xFF);
270         &xor    ($s[1],&DWP(2,$te,$v1,8));              # s3>>16
271          &mov   ($v1,&DWP(4,"esp"));                    # restore s2
272         &xor    ($s[0],&DWP(1,$te,$v0,8));              # s3>>24

274         &mov    ($v0,$v1);
275         &and    ($v1,0xFF);
276         &xor    ($s[2],&DWP(0,$te,$v1,8));              # s2>>0
277         &movz   ($v1,&HB($v0));
278         &shr    ($v0,16);
279         &xor    ($s[1],&DWP(3,$te,$v1,8));              # s2>>8
280         &movz   ($v1,&HB($v0));
281         &and    ($v0,0xFF);
282         &xor    ($s[0],&DWP(2,$te,$v0,8));              # s2>>16
283          &mov   ($v0,&DWP(8,"esp"));                    # restore s1
284         &xor    ($s[3],&DWP(1,$te,$v1,8));              # s2>>24

286         &mov    ($v1,$v0);
287         &and    ($v0,0xFF);
288         &xor    ($s[1],&DWP(0,$te,$v0,8));              # s1>>0
289         &movz   ($v0,&HB($v1));
290         &shr    ($v1,16);
291         &xor    ($s[0],&DWP(3,$te,$v0,8));              # s1>>8
292         &movz   ($v0,&HB($v1));
293         &and    ($v1,0xFF);
294         &xor    ($s[3],&DWP(2,$te,$v1,8));              # s1>>16
295          &mov   ($key,$__key);                          # reincarnate v1 as key
296         &xor    ($s[2],&DWP(1,$te,$v0,8));              # s1>>24
297 }

299 # Another experimental routine, which features "horizontal spin," but
300 # eliminates one reference to stack. Strangely enough runs slower...
301 sub enchoriz()
302 { my $v0 = $key, $v1 = $acc;

304         &movz   ($v0,&LB($s0));                 #  3, 2, 1, 0*
305         &rotr   ($s2,8);                        #  8,11,10, 9
306         &mov    ($v1,&DWP(0,$te,$v0,8));        #  0
307         &movz   ($v0,&HB($s1));                 #  7, 6, 5*, 4
308         &rotr   ($s3,16);                       # 13,12,15,14
309         &xor    ($v1,&DWP(3,$te,$v0,8));        #  5
310         &movz   ($v0,&HB($s2));                 #  8,11,10*, 9
311         &rotr   ($s0,16);                       #  1, 0, 3, 2
312         &xor    ($v1,&DWP(2,$te,$v0,8));        # 10
313         &movz   ($v0,&HB($s3));                 # 13,12,15*,14
314         &xor    ($v1,&DWP(1,$te,$v0,8));        # 15, t[0] collected
315         &mov    ($__s0,$v1);                    # t[0] saved

317         &movz   ($v0,&LB($s1));                 #  7, 6, 5, 4*
318         &shr    ($s1,16);                       #  -, -, 7, 6
319         &mov    ($v1,&DWP(0,$te,$v0,8));        #  4
320         &movz   ($v0,&LB($s3));                 # 13,12,15,14*
321         &xor    ($v1,&DWP(2,$te,$v0,8));        # 14
322         &movz   ($v0,&HB($s0));                 #  1, 0, 3*, 2
323         &and    ($s3,0xffff0000);               # 13,12, -, -
324         &xor    ($v1,&DWP(1,$te,$v0,8));        #  3
325         &movz   ($v0,&LB($s2));                 #  8,11,10, 9*
```



```
326         &or     ($s3,$s1);                      # 13,12, 7, 6
327         &xor    ($v1,&DWP(3,$te,$v0,8));        #  9, t[1] collected
328         &mov    ($s1,$v1);                      #  s[1]=t[1]

330         &movz   ($v0,&LB($s0));                 #  1, 0, 3, 2*
331         &shr    ($s2,16);                       #  -, -, 8,11
332         &mov    ($v1,&DWP(2,$te,$v0,8));        #  2
333         &movz   ($v0,&HB($s3));                 # 13,12, 7*, 6
334         &xor    ($v1,&DWP(1,$te,$v0,8));        #  7
335         &movz   ($v0,&HB($s2));                 #  -, -, 8*,11
336         &xor    ($v1,&DWP(0,$te,$v0,8));        #  8
337         &mov    ($v0,$s3);
338         &shr    ($v0,24);                       # 13
339         &xor    ($v1,&DWP(3,$te,$v0,8));        # 13, t[2] collected

341         &movz   ($v0,&LB($s2));                 #  -, -, 8,11*
342         &shr    ($s0,24);                       #  1*
343         &mov    ($s2,&DWP(1,$te,$v0,8));        # 11
344         &xor    ($s2,&DWP(3,$te,$s0,8));        #  1
345         &mov    ($s0,$__s0);                    # s[0]=t[0]
346         &movz   ($v0,&LB($s3));                 # 13,12, 7, 6*
347         &shr    ($s3,16);                       #  ,  ,13,12
348         &xor    ($s2,&DWP(2,$te,$v0,8));        #  6
349         &mov    ($key,$__key);                  # reincarnate v0 as key
350         &and    ($s3,0xff);                     #  ,  ,13,12*
351         &mov    ($s3,&DWP(0,$te,$s3,8));        # 12
352         &xor    ($s3,$s2);                      # s[2]=t[3] collected
353         &mov    ($s2,$v1);                      # s[2]=t[2]
354 }

356 # More experimental code... SSE one... Even though this one eliminates
357 # *all* references to stack, it's not faster...
358 sub sse_encbody()
359 {
360         &movz   ($acc,&LB("eax"));              #  0
361         &mov    ("ecx",&DWP(0,$tbl,$acc,8));    #  0
362         &pshufw ("mm2","mm0",0x0d);             #  7, 6, 3, 2
363         &movz   ("edx",&HB("eax"));             #  1
364         &mov    ("edx",&DWP(3,$tbl,"edx",8));   #  1
365         &shr    ("eax",16);                     #  5, 4

367         &movz   ($acc,&LB("ebx"));              # 10
368         &xor    ("ecx",&DWP(2,$tbl,$acc,8));    # 10
369         &pshufw ("mm6","mm4",0x08);             # 13,12, 9, 8
370         &movz   ($acc,&HB("ebx"));              # 11
371         &xor    ("edx",&DWP(1,$tbl,$acc,8));    # 11
372         &shr    ("ebx",16);                     # 15,14

374         &movz   ($acc,&HB("eax"));              #  5
375         &xor    ("ecx",&DWP(3,$tbl,$acc,8));    #  5
376         &movq   ("mm3",QWP(16,$key));
377         &movz   ($acc,&HB("ebx"));              # 15
378         &xor    ("ecx",&DWP(1,$tbl,$acc,8));    # 15
379         &movd   ("mm0","ecx");                  # t[0] collected

381         &movz   ($acc,&LB("eax"));              #  4
382         &mov    ("ecx",&DWP(0,$tbl,$acc,8));    #  4
383         &movd   ("eax","mm2");                  #  7, 6, 3, 2
384         &movz   ($acc,&LB("ebx"));              # 14
385         &xor    ("ecx",&DWP(2,$tbl,$acc,8));    # 14
386         &movd   ("ebx","mm6");                  # 13,12, 9, 8

388         &movz   ($acc,&HB("eax"));              #  3
389         &xor    ("ecx",&DWP(1,$tbl,$acc,8));    #  3
390         &movz   ($acc,&HB("ebx"));              #  9
391         &xor    ("ecx",&DWP(3,$tbl,$acc,8));    #  9
```

```
392         &movd   ("mm1","ecx");                 # t[1] collected

394         &movz   ($acc,&LB("eax"));             #  2
395         &mov    ("ecx",&DWP(2,$tbl,$acc,8));   #  2
396         &shr    ("eax",16);                    #  7, 6
397         &punpckldq      ("mm0","mm1");         # t[0,1] collected
398         &movz   ($acc,&LB("ebx"));             #  8
399         &xor    ("ecx",&DWP(0,$tbl,$acc,8));   #  8
400         &shr    ("ebx",16);                    # 13,12

402         &movz   ($acc,&HB("eax"));             #  7
403         &xor    ("ecx",&DWP(1,$tbl,$acc,8));   #  7
404         &pxor   ("mm0","mm3");
405         &movz   ("eax",&LB("eax"));            #  6
406         &xor    ("edx",&DWP(2,$tbl,"eax",8));  #  6
407         &pshufw ("mm1","mm0",0x08);            #  5, 4, 1, 0
408         &movz   ($acc,&HB("ebx"));             # 13
409         &xor    ("ecx",&DWP(3,$tbl,$acc,8));   # 13
410         &xor    ("ecx",&DWP(24,$key));         # t[2]
411         &movd   ("mm4","ecx");                 # t[2] collected
412         &movz   ("ebx",&LB("ebx"));            # 12
413         &xor    ("edx",&DWP(0,$tbl,"ebx",8));  # 12
414         &shr    ("ecx",16);
415         &movd   ("eax","mm1");                 #  5, 4, 1, 0
416         &mov    ("ebx",&DWP(28,$key));         # t[3]
417         &xor    ("ebx","edx");
418         &movd   ("mm5","ebx");                 # t[3] collected
419         &and    ("ebx",0xffff0000);
420         &or     ("ebx","ecx");

422         &punpckldq      ("mm4","mm5");         # t[2,3] collected
423 }

425 ######################################################################
426 # "Compact" block function
427 ######################################################################

429 sub enccompact()
430 { my $Fn = mov;
431   while ($#_>5) { pop(@_); $Fn=sub{}; }
432   my ($i,$te,@s)=@_;
433   my $tmp = $key;
434   my $out = $i==3?$s[0]:$acc;

436         # $Fn is used in first compact round and its purpose is to
437         # void restoration of some values from stack, so that after
438         # 4xenccompact with extra argument $key value is left there...
439         if ($i==3)  {   &$Fn    ($key,$__key);                  }##%edx
440         else        {   &mov    ($out,$s[0]);                   }
441                         &and    ($out,0xFF);
442         if ($i==1)  {   &shr    ($s[0],16);                     }#%ebx[1]
443         if ($i==2)  {   &shr    ($s[0],24);                     }#%ecx[2]
444                         &movz   ($out,&BP(-128,$te,$out,1));

446         if ($i==3)  {   $tmp=$s[1];                             }##%eax
447                         &movz   ($tmp,&HB($s[1]));
448                         &movz   ($tmp,&BP(-128,$te,$tmp,1));
449                         &shl    ($tmp,8);
450                         &xor    ($out,$tmp);

452         if ($i==3)  {   $tmp=$s[2]; &mov ($s[1],$__s0);         }##%ebx
453         else        {   &mov    ($tmp,$s[2]);
454                         &shr    ($tmp,16);                      }
455         if ($i==2)  {   &and    ($s[1],0xFF);                   }#%edx[2]
456                         &and    ($tmp,0xFF);
457                         &movz   ($tmp,&BP(-128,$te,$tmp,1));
```



```
458                         &shl    ($tmp,16);
459                         &xor    ($out,$tmp);

461         if ($i==3)  {   $tmp=$s[3]; &mov ($s[2],$__s1);         }##%ecx
462         elsif($i==2){   &movz   ($tmp,&HB($s[3]));              }#%ebx[2]
463         else        {   &mov    ($tmp,$s[3]);
464                         &shr    ($tmp,24);                      }
465                         &movz   ($tmp,&BP(-128,$te,$tmp,1));
466                         &shl    ($tmp,24);
467                         &xor    ($out,$tmp);
468         if ($i<2)   {   &mov    (&DWP(4+4*$i,"esp"),$out);      }
469         if ($i==3)  {   &mov    ($s[3],$acc);                   }
470         &comment();
471 }

473 sub enctransform()
474 { my @s = ($s0,$s1,$s2,$s3);
475   my $i = shift;
476   my $tmp = $tbl;
477   my $r2  = $key ;

479         &mov    ($acc,$s[$i]);
480         &and    ($acc,0x80808080);
481         &mov    ($tmp,$acc);
482         &shr    ($tmp,7);
483         &lea    ($r2,&DWP(0,$s[$i],$s[$i]));
484         &sub    ($acc,$tmp);
485         &and    ($r2,0xfefefefe);
486         &and    ($acc,0x1b1b1b1b);
487         &mov    ($tmp,$s[$i]);
488         &xor    ($acc,$r2);     # r2

490         &xor    ($s[$i],$acc);  # r0 ^ r2
491         &rotl   ($s[$i],24);
492         &xor    ($s[$i],$acc)   # ROTATE(r2^r0,24) ^ r2
493         &rotr   ($tmp,16);
494         &xor    ($s[$i],$tmp);
495         &rotr   ($tmp,8);
496         &xor    ($s[$i],$tmp);
497 }

499 &function_begin_B("_x86_AES_encrypt_compact");
500         # note that caller is expected to allocate stack frame for me!
501         &mov    ($__key,$key);                  # save key

503         &xor    ($s0,&DWP(0,$key));             # xor with key
504         &xor    ($s1,&DWP(4,$key));
505         &xor    ($s2,&DWP(8,$key));
506         &xor    ($s3,&DWP(12,$key));

508         &mov    ($acc,&DWP(240,$key));          # load key->rounds
509         &lea    ($acc,&DWP(-2,$acc,$acc));
510         &lea    ($acc,&DWP(0,$key,$acc,8));
511         &mov    ($__end,$acc);                  # end of key schedule

513         # prefetch Te4
514         &mov    ($key,&DWP(0-128,$tbl));
515         &mov    ($acc,&DWP(32-128,$tbl));
516         &mov    ($key,&DWP(64-128,$tbl));
517         &mov    ($acc,&DWP(96-128,$tbl));
518         &mov    ($key,&DWP(128-128,$tbl));
519         &mov    ($acc,&DWP(160-128,$tbl));
520         &mov    ($key,&DWP(192-128,$tbl));
521         &mov    ($acc,&DWP(224-128,$tbl));

523         &set_label("loop",16);
```

```
525                 &enccompact(0,$tbl,$s0,$s1,$s2,$s3,1);
526                 &enccompact(1,$tbl,$s1,$s2,$s3,$s0,1);
527                 &enccompact(2,$tbl,$s2,$s3,$s0,$s1,1);
528                 &enccompact(3,$tbl,$s3,$s0,$s1,$s2,1);
529                 &enctransform(2);
530                 &enctransform(3);
531                 &enctransform(0);
532                 &enctransform(1);
533                 &mov    ($key,$__key);
534                 &mov    ($tbl,$__tbl);
535                 &add    ($key,16);              # advance rd_key
536                 &xor    ($s0,&DWP(0,$key));
537                 &xor    ($s1,&DWP(4,$key));
538                 &xor    ($s2,&DWP(8,$key));
539                 &xor    ($s3,&DWP(12,$key));
540
541         &cmp    ($key,$__end);
542         &mov    ($__key,$key);
543         &jb     (&label("loop"));
544
545         &enccompact(0,$tbl,$s0,$s1,$s2,$s3);
546         &enccompact(1,$tbl,$s1,$s2,$s3,$s0);
547         &enccompact(2,$tbl,$s2,$s3,$s0,$s1);
548         &enccompact(3,$tbl,$s3,$s0,$s1,$s2);
549
550         &xor    ($s0,&DWP(16,$key));
551         &xor    ($s1,&DWP(20,$key));
552         &xor    ($s2,&DWP(24,$key));
553         &xor    ($s3,&DWP(28,$key));
554
555         &ret    ();
556 &function_end_B("_x86_AES_encrypt_compact");
557
558 ########################################################################
559 # "Compact" SSE block function.
560 ########################################################################
561 #
562 # Performance is not actually extraordinary in comparison to pure
563 # x86 code. In particular encrypt performance is virtually the same.
564 # Decrypt performance on the other hand is 15-20% better on newer
565 # µ-archs [but we're thankful for *any* improvement here], and ~50%
566 # better on PIII:-) And additionally on the pros side this code
567 # eliminates redundant references to stack and thus relieves/
568 # minimizes the pressure on the memory bus.
569 #
570 # MMX register layout                           lsb
571 # +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
572 # |          mm4          |          mm0          |
573 # +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
574 # |     s3    |     s2    |     s1    |     s0    |
575 # +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
576 # |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
577 # +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
578 #
579 # Indexes translate as s[N/4]>>(8*(N%4)), e.g. 5 means s1>>8.
580 # In this terms encryption and decryption "compact" permutation
581 # matrices can be depicted as following:
582 #
583 # encryption              lsb   # decryption              lsb
584 # +----++----+----+----+----+   # +----++----+----+----+----+
585 # | t0 || 15 | 10 |  5 |  0 |   # | t0 ||  7 | 10 | 13 |  0 |
586 # +----++----+----+----+----+   # +----++----+----+----+----+
587 # | t1 ||  3 | 14 |  9 |  4 |   # | t1 || 11 | 14 |  1 |  4 |
588 # +----++----+----+----+----+   # +----++----+----+----+----+
589 # | t2 ||  7 |  2 | 13 |  8 |   # | t2 || 15 |  2 |  5 |  8 |
```



```
590 # +----++----+----+----+----+   # +----++----+----+----+----+
591 # | t3 || 11 |  6 |  1 | 12 |   # | t3 ||  3 |  6 |  9 | 12 |
592 # +----++----+----+----+----+   # +----++----+----+----+----+
593 #
594 ########################################################################
595 # Why not xmm registers? Short answer. It was actually tested and
596 # was not any faster, but *contrary*, most notably on Intel CPUs.
597 # Longer answer. Main advantage of using mm registers is that movd
598 # latency is lower, especially on Intel P4. While arithmetic
599 # instructions are twice as many, they can be scheduled every cycle
600 # and not every second one when they are operating on xmm register,
601 # so that "arithmetic throughput" remains virtually the same. And
602 # finally the code can be executed even on elder SSE-only CPUs:-)
603
604 sub sse_enccompact()
605 {
606         &pshufw ("mm1","mm0",0x08);             #  5, 4, 1, 0
607         &pshufw ("mm5","mm4",0x0d);             # 15,14,11,10
608         &movd   ("eax","mm1");                  #  5, 4, 1, 0
609         &movd   ("ebx","mm5");                  # 15,14,11,10
610
611         &movz   ($acc,&LB("eax"));              #  0
612         &movz   ("ecx",&BP(-128,$tbl,$acc,1));  #  0
613         &pshufw ("mm2","mm0",0x0d);             #  7, 6, 3, 2
614         &movz   ("edx",&HB("eax"));             #  1
615         &movz   ("edx",&BP(-128,$tbl,"edx",1)); #  1
616         &shl    ("edx",8);                      #  1
617         &shr    ("eax",16);                     #  5, 4
618
619         &movz   ($acc,&LB("ebx"));              # 10
620         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 10
621         &shl    ($acc,16);                      # 10
622         &or     ("ecx",$acc);                   # 10
623         &pshufw ("mm6","mm4",0x08);             # 13,12, 9, 8
624         &movz   ($acc,&HB("ebx"));              # 11
625         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 11
626         &shl    ($acc,24);                      # 11
627         &or     ("edx",$acc);                   # 11
628         &shr    ("ebx",16);                     # 15,14
629
630         &movz   ($acc,&HB("eax"));              #  5
631         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  5
632         &shl    ($acc,8);                       #  5
633         &or     ("ecx",$acc);                   #  5
634         &movz   ($acc,&HB("ebx"));              # 15
635         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 15
636         &shl    ($acc,24);                      # 15
637         &or     ("ecx",$acc);                   # 15
638         &movd   ("mm0","ecx");                  # t[0] collected
639
640         &movz   ($acc,&LB("eax"));              #  4
641         &movz   ("ecx",&BP(-128,$tbl,$acc,1));  #  4
642         &movd   ("eax","mm2");                  #  7, 6, 3, 2
643         &movz   ($acc,&LB("ebx"));              # 14
644         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 14
645         &shl    ($acc,16);                      # 14
646         &or     ("ecx",$acc);                   # 14
647
648         &movd   ("ebx","mm6");                  # 13,12, 9, 8
649         &movz   ($acc,&HB("eax"));              #  3
650         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  3
651         &shl    ($acc,24);                      #  3
652         &or     ("ecx",$acc);                   #  3
653         &movz   ($acc,&HB("ebx"));              #  9
654         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  9
655         &shl    ($acc,8);                       #  9
```

```
656         &or     ("ecx",$acc);                   #  9
657         &movd   ("mm1","ecx");                  # t[1] collected

659         &movz   ($acc,&LB("ebx"));              #  8
660         &movz   ("ecx",&BP(-128,$tbl,$acc,1));  #  8
661         &shr    ("ebx",16);                     # 13,12
662         &movz   ($acc,&LB("eax"));              #  2
663         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  2
664         &shl    ($acc,16);                      #  2
665         &or     ("ecx",$acc);                   #  2
666         &shr    ("eax",16);                     #  7, 6

668         &punpckldq      ("mm0","mm1");          # t[0,1] collected

670         &movz   ($acc,&HB("eax"));              #  7
671         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  7
672         &shl    ($acc,24);                      #  7
673         &or     ("ecx",$acc);                   #  7
674         &and    ("eax",0xff);                   #  6
675         &movz   ("eax",&BP(-128,$tbl,"eax",1)); #  6
676         &shl    ("eax",16);                     #  6
677         &or     ("edx","eax");                  #  6
678         &movz   ($acc,&HB("ebx"));              # 13
679         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 13
680         &shl    ($acc,8);                       # 13
681         &or     ("ecx",$acc);                   # 13
682         &movd   ("mm4","ecx");                  # t[2] collected
683         &and    ("ebx",0xff);                   # 12
684         &movz   ("ebx",&BP(-128,$tbl,"ebx",1)); # 12
685         &or     ("edx","ebx");                  # 12
686         &movd   ("mm5","edx");                  # t[3] collected

688         &punpckldq      ("mm4","mm5");          # t[2,3] collected
689 }

691                                     if (!$x86only) {
692 &function_begin_B("_sse_AES_encrypt_compact");
693         &pxor   ("mm0",&QWP(0,$key));   #  7, 6, 5, 4, 3, 2, 1, 0
694         &pxor   ("mm4",&QWP(8,$key));   # 15,14,13,12,11,10, 9, 8

696         # note that caller is expected to allocate stack frame for me!
697         &mov    ($acc,&DWP(240,$key));          # load key->rounds
698         &lea    ($acc,&DWP(-2,$acc,$acc));
699         &lea    ($acc,&DWP(0,$key,$acc,8));
700         &mov    ($__end,$acc);                  # end of key schedule

702         &mov    ($s0,0x1b1b1b1b);               # magic constant
703         &mov    (&DWP(8,"esp"),$s0);
704         &mov    (&DWP(12,"esp"),$s0);

706         # prefetch Te4
707         &mov    ($s0,&DWP(0-128,$tbl));
708         &mov    ($s1,&DWP(32-128,$tbl));
709         &mov    ($s2,&DWP(64-128,$tbl));
710         &mov    ($s3,&DWP(96-128,$tbl));
711         &mov    ($s0,&DWP(128-128,$tbl));
712         &mov    ($s1,&DWP(160-128,$tbl));
713         &mov    ($s2,&DWP(192-128,$tbl));
714         &mov    ($s3,&DWP(224-128,$tbl));

716         &set_label("loop",16);
717                 &sse_enccompact();
718                 &add    ($key,16);
719                 &cmp    ($key,$__end);
720                 &ja     (&label("out"));
```



```
722                 &movq   ("mm2",&QWP(8,"esp"));
723                 &pxor   ("mm3","mm3");          &pxor   ("mm7","mm7");
724                 &movq   ("mm1","mm0");          &movq   ("mm5","mm4");  # r0
725                 &pcmpgtb("mm3","mm0");          &pcmpgtb("mm7","mm4");
726                 &pand   ("mm3","mm2");          &pand   ("mm7","mm2");
727                 &pshufw ("mm2","mm0",0xb1);     &pshufw ("mm6","mm4",0xb1);# ROT
728                 &paddb  ("mm0","mm0");          &paddb  ("mm4","mm4");
729                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # = r2
730                 &pshufw ("mm3","mm2",0xb1);     &pshufw ("mm7","mm6",0xb1);# r0
731                 &pxor   ("mm1","mm0");          &pxor   ("mm5","mm4");  # r0^r2
732                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");  # ^= ROT

734                 &movq   ("mm2","mm3");          &movq   ("mm6","mm7");
735                 &pslld  ("mm3",8);              &pslld  ("mm7",8);
736                 &psrld  ("mm2",24);             &psrld  ("mm6",24);
737                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= r0<
738                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");  # ^= r0>

740                 &movq   ("mm3","mm1");          &movq   ("mm7","mm5");
741                 &movq   ("mm2",&QWP(0,$key));   &movq   ("mm6",&QWP(8,$key));
742                 &psrld  ("mm1",8);              &psrld  ("mm5",8);
743                 &mov    ($s0,&DWP(0-128,$tbl));
744                 &pslld  ("mm3",24);             &pslld  ("mm7",24);
745                 &mov    ($s1,&DWP(64-128,$tbl));
746                 &pxor   ("mm0","mm1");          &pxor   ("mm4","mm5");  # ^= (r2
747                 &mov    ($s2,&DWP(128-128,$tbl));
748                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= (r2
749                 &mov    ($s3,&DWP(192-128,$tbl));

751                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");
752         &jmp    (&label("loop"));

754         &set_label("out",16);
755         &pxor   ("mm0",&QWP(0,$key));
756         &pxor   ("mm4",&QWP(8,$key));

758         &ret    ();
759 &function_end_B("_sse_AES_encrypt_compact");
760                                     }

762 ######################################################################
763 # Vanilla block function.
764 ######################################################################

766 sub encstep()
767 { my ($i,$te,@s) = @_;
768   my $tmp = $key;
769   my $out = $i==3?$s[0]:$acc;

771         # lines marked with #%e?x[i] denote "reordered" instructions...
772         if ($i==3)  {   &mov    ($key,$__key);                  }##%edx
773         else        {   &mov    ($out,$s[0]);
774                         &and    ($out,0xFF);                    }
775         if ($i==1)  {   &shr    ($s[0],16);                     }#%ebx[1]
776         if ($i==2)  {   &shr    ($s[0],24);                     }#%ecx[2]
777                         &mov    ($out,&DWP(0,$te,$out,8));

779         if ($i==3)  {   $tmp=$s[1];                             }##%eax
780                         &movz   ($tmp,&HB($s[1]));
781                         &xor    ($out,&DWP(3,$te,$tmp,8));

783         if ($i==3)  {   $tmp=$s[2]; &mov ($s[1],$__s0);         }##%ebx
784         else        {   &mov    ($tmp,$s[2]);
785                         &shr    ($tmp,16);                      }
786         if ($i==2)  {   &and    ($s[1],0xFF);                   }#%edx[2]
787                         &and    ($tmp,0xFF);
```

```
788                         &xor    ($out,&DWP(2,$te,$tmp,8));

790         if ($i==3)  {   $tmp=$s[3]; &mov ($s[2],$__s1);         }##%ecx
791         elsif($i==2){   &movz   ($tmp,&HB($s[3]));              }#%ebx[2]
792         else        {   &mov    ($tmp,$s[3]);
793                         &shr    ($tmp,24)                       }
794                         &xor    ($out,&DWP(1,$te,$tmp,8));
795         if ($i<2)   {   &mov    (&DWP(4+4*$i,"esp"),$out);      }
796         if ($i==3)  {   &mov    ($s[3],$acc);                   }
797                         &comment();
798 }

800 sub enclast()
801 { my ($i,$te,@s)=@_;
802   my $tmp = $key;
803   my $out = $i==3?$s[0]:$acc;

805         if ($i==3)  {   &mov    ($key,$__key);                  }##%edx
806         else        {   &mov    ($out,$s[0]);                   }
807                         &and    ($out,0xFF);
808         if ($i==1)  {   &shr    ($s[0],16);                     }#%ebx[1]
809         if ($i==2)  {   &shr    ($s[0],24);                     }#%ecx[2]
810                         &mov    ($out,&DWP(2,$te,$out,8));
811                         &and    ($out,0x000000ff);

813         if ($i==3)  {   $tmp=$s[1];                             }##%eax
814                         &movz   ($tmp,&HB($s[1]));
815                         &mov    ($tmp,&DWP(0,$te,$tmp,8));
816                         &and    ($tmp,0x0000ff00);
817                         &xor    ($out,$tmp);

819         if ($i==3)  {   $tmp=$s[2]; &mov ($s[1],$__s0);         }##%ebx
820         else        {   &mov    ($tmp,$s[2]);
821                         &shr    ($tmp,16);                      }
822         if ($i==2)  {   &and    ($s[1],0xFF);                   }#%edx[2]
823                         &and    ($tmp,0xFF);
824                         &mov    ($tmp,&DWP(0,$te,$tmp,8));
825                         &and    ($tmp,0x00ff0000);
826                         &xor    ($out,$tmp);

828         if ($i==3)  {   $tmp=$s[3]; &mov ($s[2],$__s1);         }##%ecx
829         elsif($i==2){   &movz   ($tmp,&HB($s[3]));              }#%ebx[2]
830         else        {   &mov    ($tmp,$s[3]);
831                         &shr    ($tmp,24);                      }
832                         &mov    ($tmp,&DWP(2,$te,$tmp,8));
833                         &and    ($tmp,0xff000000);
834                         &xor    ($out,$tmp);
835         if ($i<2)   {   &mov    (&DWP(4+4*$i,"esp"),$out);      }
836         if ($i==3)  {   &mov    ($s[3],$acc);                   }
837 }

839 &function_begin_B("_x86_AES_encrypt");
840         if ($vertical_spin) {
841                 # I need high parts of volatile registers to be accessible...
842                 &exch   ($s1="edi",$key="ebx");
843                 &mov    ($s2="esi",$acc="ecx");
844         }

846         # note that caller is expected to allocate stack frame for me!
847         &mov    ($__key,$key);                  # save key

849         &xor    ($s0,&DWP(0,$key));             # xor with key
850         &xor    ($s1,&DWP(4,$key));
851         &xor    ($s2,&DWP(8,$key));
852         &xor    ($s3,&DWP(12,$key));
```



```
854         &mov    ($acc,&DWP(240,$key));          # load key->rounds

856         if ($small_footprint) {
857             &lea        ($acc,&DWP(-2,$acc,$acc));
858             &lea        ($acc,&DWP(0,$key,$acc,8));
859             &mov        ($__end,$acc);          # end of key schedule

861             &set_label("loop",16);
862                 if ($vertical_spin) {
863                     &encvert($tbl,$s0,$s1,$s2,$s3);
864                 } else {
865                     &encstep(0,$tbl,$s0,$s1,$s2,$s3);
866                     &encstep(1,$tbl,$s1,$s2,$s3,$s0);
867                     &encstep(2,$tbl,$s2,$s3,$s0,$s1);
868                     &encstep(3,$tbl,$s3,$s0,$s1,$s2);
869                 }
870                 &add    ($key,16);              # advance rd_key
871                 &xor    ($s0,&DWP(0,$key));
872                 &xor    ($s1,&DWP(4,$key));
873                 &xor    ($s2,&DWP(8,$key));
874                 &xor    ($s3,&DWP(12,$key));
875             &cmp        ($key,$__end);
876             &mov        ($__key,$key);
877             &jb         (&label("loop"));
878         }
879         else {
880             &cmp        ($acc,10);
881             &jle        (&label("10rounds"));
882             &cmp        ($acc,12);
883             &jle        (&label("12rounds"));

885         &set_label("14rounds",4);
886             for ($i=1;$i<3;$i++) {
887                 if ($vertical_spin) {
888                     &encvert($tbl,$s0,$s1,$s2,$s3);
889                 } else {
890                     &encstep(0,$tbl,$s0,$s1,$s2,$s3);
891                     &encstep(1,$tbl,$s1,$s2,$s3,$s0);
892                     &encstep(2,$tbl,$s2,$s3,$s0,$s1);
893                     &encstep(3,$tbl,$s3,$s0,$s1,$s2);
894                 }
895                 &xor    ($s0,&DWP(16*$i+0,$key));
896                 &xor    ($s1,&DWP(16*$i+4,$key));
897                 &xor    ($s2,&DWP(16*$i+8,$key));
898                 &xor    ($s3,&DWP(16*$i+12,$key));
899             }
900             &add        ($key,32);
901             &mov        ($__key,$key);          # advance rd_key
902         &set_label("12rounds",4);
903             for ($i=1;$i<3;$i++) {
904                 if ($vertical_spin) {
905                     &encvert($tbl,$s0,$s1,$s2,$s3);
906                 } else {
907                     &encstep(0,$tbl,$s0,$s1,$s2,$s3);
908                     &encstep(1,$tbl,$s1,$s2,$s3,$s0);
909                     &encstep(2,$tbl,$s2,$s3,$s0,$s1);
910                     &encstep(3,$tbl,$s3,$s0,$s1,$s2);
911                 }
912                 &xor    ($s0,&DWP(16*$i+0,$key));
913                 &xor    ($s1,&DWP(16*$i+4,$key));
914                 &xor    ($s2,&DWP(16*$i+8,$key));
915                 &xor    ($s3,&DWP(16*$i+12,$key));
916             }
917             &add        ($key,32);
918             &mov        ($__key,$key);          # advance rd_key
919         &set_label("10rounds",4);
```

```
            for ($i=1;$i<10;$i++) {
                if ($vertical_spin) {
                    &encvert($tbl,$s0,$s1,$s2,$s3);
                } else {
                    &encstep(0,$tbl,$s0,$s1,$s2,$s3);
                    &encstep(1,$tbl,$s1,$s2,$s3,$s0);
                    &encstep(2,$tbl,$s2,$s3,$s0,$s1);
                    &encstep(3,$tbl,$s3,$s0,$s1,$s2);
                }
                &xor    ($s0,&DWP(16*$i+0,$key));
                &xor    ($s1,&DWP(16*$i+4,$key));
                &xor    ($s2,&DWP(16*$i+8,$key));
                &xor    ($s3,&DWP(16*$i+12,$key));
            }
        }

        if ($vertical_spin) {
            # "reincarnate" some registers for "horizontal" spin...
            &mov        ($s1="ebx",$key="edi");
            &mov        ($s2="ecx",$acc="esi");
        }
        &enclast(0,$tbl,$s0,$s1,$s2,$s3);
        &enclast(1,$tbl,$s1,$s2,$s3,$s0);
        &enclast(2,$tbl,$s2,$s3,$s0,$s1);
        &enclast(3,$tbl,$s3,$s0,$s1,$s2);

        &add    ($key,$small_footprint?16:160);
        &xor    ($s0,&DWP(0,$key));
        &xor    ($s1,&DWP(4,$key));
        &xor    ($s2,&DWP(8,$key));
        &xor    ($s3,&DWP(12,$key));

        &ret    ();

&set_label("AES_Te",64);        # Yes! I keep it in the code segment!
        &_data_word(0xa56363c6, 0x847c7cf8, 0x997777ee, 0x8d7b7bf6);
        &_data_word(0x0df2f2ff, 0xbd6b6bd6, 0xb16f6fde, 0x54c5c591);
        &_data_word(0x50303060, 0x03010102, 0xa96767ce, 0x7d2b2b56);
        &_data_word(0x19fefee7, 0x62d7d7b5, 0xe6ababd4, 0x9a7676ec);
        &_data_word(0x45caca8f, 0x9d82821f, 0x40c9c989, 0x877d7dfa);
        &_data_word(0x15fafaef, 0xeb5959b2, 0xc947478e, 0x0bf0f0fb);
        &_data_word(0xecadad41, 0x67d4d4b3, 0xfda2a25f, 0xeaafaf45);
        &_data_word(0xbf9c9c23, 0xf7a4a453, 0x967272e4, 0x5bc0c09b);
        &_data_word(0xc2b7b775, 0x1cfdfde1, 0xae93933d, 0x6a26264c);
        &_data_word(0x5a36366c, 0x413f3f7e, 0x02f7f7f5, 0x4fcccc83);
        &_data_word(0x5c343468, 0xf4a5a551, 0x34e5e5d1, 0x08f1f1f9);
        &_data_word(0x937171e2, 0x73d8d8ab, 0x53313162, 0x3f15152a);
        &_data_word(0x0c040408, 0x52c7c795, 0x65232346, 0x5ec3c39d);
        &_data_word(0x28181830, 0xa1969637, 0x0f05050a, 0xb59a9a2f);
        &_data_word(0x0907070e, 0x36121224, 0x9b80801b, 0x3de2e2df);
        &_data_word(0x26ebebcd, 0x6927274e, 0xcdb2b27f, 0x9f7575ea);
        &_data_word(0x1b090912, 0x9e83831d, 0x742c2c58, 0x2e1a1a34);
        &_data_word(0x2d1b1b36, 0xb26e6edc, 0xee5a5ab4, 0xfba0a05b);
        &_data_word(0xf65252a4, 0x4d3b3b76, 0x61d6d6b7, 0xceb3b37d);
        &_data_word(0x7b292952, 0x3ee3e3dd, 0x712f2f5e, 0x97848413);
        &_data_word(0xf55353a6, 0x68d1d1b9, 0x00000000, 0x2cededc1);
        &_data_word(0x60202040, 0x1ffcfce3, 0xc8b1b179, 0xed5b5bb6);
        &_data_word(0xbe6a6ad4, 0x46cbcb8d, 0xd9bebe67, 0x4b393972);
        &_data_word(0xde4a4a94, 0xd44c4c98, 0xe85858b0, 0x4acfcf85);
        &_data_word(0x6bd0d0bb, 0x2aefefc5, 0xe5aaaa4f, 0x16fbfbed);
        &_data_word(0xc5434386, 0xd74d4d9a, 0x55333366, 0x94858511);
        &_data_word(0xcf45458a, 0x10f9f9e9, 0x06020204, 0x817f7ffe);
        &_data_word(0xf05050a0, 0x443c3c78, 0xba9f9f25, 0xe3a8a84b);
        &_data_word(0xf35151a2, 0xfea3a35d, 0xc0404080, 0x8a8f8f05);
        &_data_word(0xad92923f, 0xbc9d9d21, 0x48383870, 0x04f5f5f1);
        &_data_word(0xdfbcbc63, 0xc1b6b677, 0x75dadaaf, 0x63212142);
```



```
        &_data_word(0x30101020, 0x1affffe5, 0x0ef3f3fd, 0x6dd2d2bf);
        &_data_word(0x4ccdcd81, 0x140c0c18, 0x35131326, 0x2fececc3);
        &_data_word(0xe15f5fbe, 0xa2979735, 0xcc444488, 0x3917172e);
        &_data_word(0x57c4c493, 0xf2a7a755, 0x827e7efc, 0x473d3d7a);
        &_data_word(0xac6464c8, 0xe75d5dba, 0x2b191932, 0x957373e6);
        &_data_word(0xa06060c0, 0x98818119, 0xd14f4f9e, 0x7fdcdca3);
        &_data_word(0x66222244, 0x7e2a2a54, 0xab90903b, 0x8388880b);
        &_data_word(0xca46468c, 0x29eeeec7, 0xd3b8b86b, 0x3c141428);
        &_data_word(0x79dedea7, 0xe25e5ebc, 0x1d0b0b16, 0x76dbdbad);
        &_data_word(0x3be0e0db, 0x56323264, 0x4e3a3a74, 0x1e0a0a14);
        &_data_word(0xdb494992, 0x0a06060c, 0x6c242448, 0xe45c5cb8);
        &_data_word(0x5dc2c29f, 0x6ed3d3bd, 0xefacac43, 0xa66262c4);
        &_data_word(0xa8919139, 0xa4959531, 0x37e4e4d3, 0x8b7979f2);
        &_data_word(0x32e7e7d5, 0x43c8c88b, 0x5937376e, 0xb76d6dda);
        &_data_word(0x8c8d8d01, 0x64d5d5b1, 0xd24e4e9c, 0xe0a9a949);
        &_data_word(0xb46c6cd8, 0xfa5656ac, 0x07f4f4f3, 0x25eaeacf);
        &_data_word(0xaf6565ca, 0x8e7a7af4, 0xe9aeae47, 0x18080810);
        &_data_word(0xd5baba6f, 0x887878f0, 0x6f25254a, 0x722e2e5c);
        &_data_word(0x241c1c38, 0xf1a6a657, 0xc7b4b473, 0x51c6c697);
        &_data_word(0x23e8e8cb, 0x7cdddda1, 0x9c7474e8, 0x211f1f3e);
        &_data_word(0xdd4b4b96, 0xdcbdbd61, 0x868b8b0d, 0x858a8a0f);
        &_data_word(0x907070e0, 0x423e3e7c, 0xc4b5b571, 0xaa6666cc);
        &_data_word(0xd8484890, 0x05030306, 0x01f6f6f7, 0x120e0e1c);
        &_data_word(0xa36161c2, 0x5f35356a, 0xf95757ae, 0xd0b9b969);
        &_data_word(0x91868617, 0x58c1c199, 0x271d1d3a, 0xb99e9e27);
        &_data_word(0x38e1e1d9, 0x13f8f8eb, 0xb398982b, 0x33111122);
        &_data_word(0xbb6969d2, 0x70d9d9a9, 0x898e8e07, 0xa7949433);
        &_data_word(0xb69b9b2d, 0x221e1e3c, 0x92878715, 0x20e9e9c9);
        &_data_word(0x49cece87, 0xff5555aa, 0x78282850, 0x7adfdfa5);
        &_data_word(0x8f8c8c03, 0xf8a1a159, 0x80898909, 0x170d0d1a);
        &_data_word(0xdabfbf65, 0x31e6e6d7, 0xc6424284, 0xb86868d0);
        &_data_word(0xc3414182, 0xb0999929, 0x772d2d5a, 0x110f0f1e);
        &_data_word(0xcbb0b07b, 0xfc5454a8, 0xd6bbbb6d, 0x3a16162c);

#Te4    # four copies of Te4 to choose from to avoid L1 aliasing
        &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
        &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
        &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
        &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
        &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
        &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
        &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
        &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
        &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
        &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
        &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
        &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
        &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
        &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
        &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
        &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
        &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
        &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
        &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
        &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
        &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
        &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
        &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
        &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
        &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
        &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
        &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
        &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
        &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
        &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
        &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
```

```
1052         &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);

1054         &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
1055         &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
1056         &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
1057         &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
1058         &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
1059         &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
1060         &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
1061         &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
1062         &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
1063         &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
1064         &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
1065         &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
1066         &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
1067         &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
1068         &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
1069         &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
1070         &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
1071         &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
1072         &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
1073         &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
1074         &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
1075         &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
1076         &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
1077         &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
1078         &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
1079         &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
1080         &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
1081         &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
1082         &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
1083         &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
1084         &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
1085         &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);

1087         &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
1088         &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
1089         &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
1090         &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
1091         &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
1092         &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
1093         &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
1094         &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
1095         &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
1096         &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
1097         &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
1098         &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
1099         &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
1100         &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
1101         &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
1102         &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
1103         &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
1104         &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
1105         &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
1106         &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
1107         &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
1108         &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
1109         &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
1110         &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
1111         &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
1112         &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
1113         &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
1114         &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
1115         &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
1116         &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
1117         &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
```



```
1118         &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);

1120         &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
1121         &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
1122         &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
1123         &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
1124         &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
1125         &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
1126         &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
1127         &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
1128         &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
1129         &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
1130         &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
1131         &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
1132         &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
1133         &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
1134         &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
1135         &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
1136         &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
1137         &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
1138         &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
1139         &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
1140         &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
1141         &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
1142         &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
1143         &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
1144         &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
1145         &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
1146         &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
1147         &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
1148         &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
1149         &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
1150         &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
1151         &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);
1152 #rcon:
1153         &data_word(0x00000001, 0x00000002, 0x00000004, 0x00000008);
1154         &data_word(0x00000010, 0x00000020, 0x00000040, 0x00000080);
1155         &data_word(0x0000001b, 0x00000036, 0x00000000, 0x00000000);
1156         &data_word(0x00000000, 0x00000000, 0x00000000, 0x00000000);
1157 &function_end_B("_x86_AES_encrypt");

1159 # void AES_encrypt (const void *inp,void *out,const AES_KEY *key);
1160 &function_begin("AES_encrypt");
1161         &mov    ($acc,&wparam(0));              # load inp
1162         &mov    ($key,&wparam(2));              # load key

1164         &mov    ($s0,"esp");
1165         &sub    ("esp",36);
1166         &and    ("esp",-64);                    # align to cache-line

1168         # place stack frame just "above" the key schedule
1169         &lea    ($s1,&DWP(-64-63,$key));
1170         &sub    ($s1,"esp");
1171         &neg    ($s1);
1172         &and    ($s1,0x3C0);    # modulo 1024, but aligned to cache-line
1173         &sub    ("esp",$s1);
1174         &add    ("esp",4);      # 4 is reserved for caller's return address
1175         &mov    ($_esp,$s0);                    # save stack pointer

1177         &call   (&label("pic_point"));          # make it PIC!
1178         &set_label("pic_point");
1179         &blindpop($tbl);
1180         &picmeup($s0,"OPENSSL_ia32cap_P",$tbl,&label("pic_point")) if (!$x86only
1181         &lea    ($tbl,&DWP(&label("AES_Te")."-".&label("pic_point"),$tbl));

1183         # pick Te4 copy which can't "overlap" with stack frame or key schedule
```

```
        &lea    ($s1,&DWP(768-4,"esp"));
        &sub    ($s1,$tbl);
        &and    ($s1,0x300);
        &lea    ($tbl,&DWP(2048+128,$tbl,$s1));

                                        if (!$x86only) {
        &bt     (&DWP(0,$s0),25);       # check for SSE bit
        &jnc    (&label("x86"));

        &movq   ("mm0",&QWP(0,$acc));
        &movq   ("mm4",&QWP(8,$acc));
        &call   ("_sse_AES_encrypt_compact");
        &mov    ("esp",$_esp);                  # restore stack pointer
        &mov    ($acc,&wparam(1));              # load out
        &movq   (&QWP(0,$acc),"mm0");           # write output data
        &movq   (&QWP(8,$acc),"mm4");
        &emms   ();
        &function_end_A();
                                        }
        &set_label("x86",16);
        &mov    ($_tbl,$tbl);
        &mov    ($s0,&DWP(0,$acc));             # load input data
        &mov    ($s1,&DWP(4,$acc));
        &mov    ($s2,&DWP(8,$acc));
        &mov    ($s3,&DWP(12,$acc));
        &call   ("_x86_AES_encrypt_compact");
        &mov    ("esp",$_esp);                  # restore stack pointer
        &mov    ($acc,&wparam(1));              # load out
        &mov    (&DWP(0,$acc),$s0);             # write output data
        &mov    (&DWP(4,$acc),$s1);
        &mov    (&DWP(8,$acc),$s2);
        &mov    (&DWP(12,$acc),$s3);
&function_end("AES_encrypt");

#------------------------------------------------------------------#

######################################################################
# "Compact" block function
######################################################################

sub deccompact()
{ my $Fn = mov;
  while ($#_>5) { pop(@_); $Fn=sub{}; }
  my ($i,$td,@s)=@_;
  my $tmp = $key;
  my $out = $i==3?$s[0]:$acc;

        # $Fn is used in first compact round and its purpose is to
        # void restoration of some values from stack, so that after
        # 4xdeccompact with extra argument $key, $s0 and $s1 values
        # are left there...
        if($i==3)   {   &$Fn    ($key,$__key);                  }
        else        {   &mov    ($out,$s[0]);                   }
                        &and    ($out,0xFF);
                        &movz   ($out,&BP(-128,$td,$out,1));

        if ($i==3)  {   $tmp=$s[1];                             }
                        &movz   ($tmp,&HB($s[1]));
                        &movz   ($tmp,&BP(-128,$td,$tmp,1));
                        &shl    ($tmp,8);
                        &xor    ($out,$tmp);

        if ($i==3)  {   $tmp=$s[2]; &mov ($s[1],$acc);          }
        else        {   mov     ($tmp,$s[2]);                   }
                        &shr    ($tmp,16);
                        &and    ($tmp,0xFF);
```



```
                        &movz   ($tmp,&BP(-128,$td,$tmp,1));
                        &shl    ($tmp,16);
                        &xor    ($out,$tmp);

        if ($i==3)  {   $tmp=$s[3]; &$Fn ($s[2],$__s1);         }
        else        {   &mov    ($tmp,$s[3]);                   }
                        &shr    ($tmp,24);
                        &movz   ($tmp,&BP(-128,$td,$tmp,1));
                        &shl    ($tmp,24);
                        &xor    ($out,$tmp);
        if ($i<2)   {   &mov    (&DWP(4+4*$i,"esp"),$out);      }
        if ($i==3)  {   &$Fn    ($s[3],$__s0);                  }
}

# must be called with 2,3,0,1 as argument sequence!!!
sub dectransform()
{ my @s = ($s0,$s1,$s2,$s3);
  my $i = shift;
  my $tmp = $key;
  my $tp2 = @s[($i+2)%4]; $tp2 = @s[2] if ($i==1);
  my $tp4 = @s[($i+3)%4]; $tp4 = @s[3] if ($i==1);
  my $tp8 = $tbl;

        &mov    ($acc,$s[$i]);
        &and    ($acc,0x80808080);
        &mov    ($tmp,$acc);
        &shr    ($tmp,7);
        &lea    ($tp2,&DWP(0,$s[$i],$s[$i]));
        &sub    ($acc,$tmp);
        &and    ($tp2,0xfefefefe);
        &and    ($acc,0x1b1b1b1b);
        &xor    ($acc,$tp2);
        &mov    ($tp2,$acc);

        &and    ($acc,0x80808080);
        &mov    ($tmp,$acc);
        &shr    ($tmp,7);
        &lea    ($tp4,&DWP(0,$tp2,$tp2));
        &sub    ($acc,$tmp);
        &and    ($tp4,0xfefefefe);
        &and    ($acc,0x1b1b1b1b);
         &xor   ($tp2,$s[$i]);  # tp2^tp1
        &xor    ($acc,$tp4);
        &mov    ($tp4,$acc);

        &and    ($acc,0x80808080);
        &mov    ($tmp,$acc);
        &shr    ($tmp,7);
        &lea    ($tp8,&DWP(0,$tp4,$tp4));
        &sub    ($acc,$tmp);
        &and    ($tp8,0xfefefefe);
        &and    ($acc,0x1b1b1b1b);
         &xor   ($tp4,$s[$i]);  # tp4^tp1
         &rotl  ($s[$i],8);     # = ROTATE(tp1,8)
        &xor    ($tp8,$acc);

        &xor    ($s[$i],$tp2);
        &xor    ($tp2,$tp8);
        &rotl   ($tp2,24);
        &xor    ($s[$i],$tp4);
        &xor    ($tp4,$tp8);
        &rotl   ($tp4,16);
        &xor    ($s[$i],$tp8);  # ^= tp8^(tp4^tp1)^(tp2^tp1)
        &rotl   ($tp8,8);
        &xor    ($s[$i],$tp2);  # ^= ROTATE(tp8^tp2^tp1,24)
        &xor    ($s[$i],$tp4);  # ^= ROTATE(tp8^tp4^tp1,16)
```

```
1316          &mov    ($s[0],$__s0)                   if($i==2); #prefetch $s0
1317          &mov    ($s[1],$__s1)                   if($i==3); #prefetch $s1
1318          &mov    ($s[2],$__s2)                   if($i==1);
1319         &xor     ($s[$i],$tp8);  # ^= ROTATE(tp8,8)

1321         &mov     ($s[3],$__s3)                   if($i==1);
1322         &mov     (&DWP(4+4*$i,"esp"),$s[$i])     if($i>=2);
1323 }

1325 &function_begin_B("_x86_AES_decrypt_compact");
1326         # note that caller is expected to allocate stack frame for me!
1327         &mov     ($__key,$key);                  # save key

1329         &xor     ($s0,&DWP(0,$key));             # xor with key
1330         &xor     ($s1,&DWP(4,$key));
1331         &xor     ($s2,&DWP(8,$key));
1332         &xor     ($s3,&DWP(12,$key));

1334         &mov     ($acc,&DWP(240,$key));          # load key->rounds

1336         &lea     ($acc,&DWP(-2,$acc,$acc));
1337         &lea     ($acc,&DWP(0,$key,$acc,8));
1338         &mov     ($__end,$acc);                  # end of key schedule

1340         # prefetch Td4
1341         &mov     ($key,&DWP(0-128,$tbl));
1342         &mov     ($acc,&DWP(32-128,$tbl));
1343         &mov     ($key,&DWP(64-128,$tbl));
1344         &mov     ($acc,&DWP(96-128,$tbl));
1345         &mov     ($key,&DWP(128-128,$tbl));
1346         &mov     ($acc,&DWP(160-128,$tbl));
1347         &mov     ($key,&DWP(192-128,$tbl));
1348         &mov     ($acc,&DWP(224-128,$tbl));

1350         &set_label("loop",16);

1352                 &deccompact(0,$tbl,$s0,$s3,$s2,$s1,1);
1353                 &deccompact(1,$tbl,$s1,$s0,$s3,$s2,1);
1354                 &deccompact(2,$tbl,$s2,$s1,$s0,$s3,1);
1355                 &deccompact(3,$tbl,$s3,$s2,$s1,$s0,1);
1356                 &dectransform(2);
1357                 &dectransform(3);
1358                 &dectransform(0);
1359                 &dectransform(1);
1360                 &mov    ($key,$__key);
1361                 &mov    ($tbl,$__tbl);
1362                 &add    ($key,16);              # advance rd_key
1363                 &xor    ($s0,&DWP(0,$key));
1364                 &xor    ($s1,&DWP(4,$key));
1365                 &xor    ($s2,&DWP(8,$key));
1366                 &xor    ($s3,&DWP(12,$key));

1368         &cmp     ($key,$__end);
1369         &mov     ($__key,$key);
1370         &jb      (&label("loop"));

1372         &deccompact(0,$tbl,$s0,$s3,$s2,$s1);
1373         &deccompact(1,$tbl,$s1,$s0,$s3,$s2);
1374         &deccompact(2,$tbl,$s2,$s1,$s0,$s3);
1375         &deccompact(3,$tbl,$s3,$s2,$s1,$s0);

1377         &xor     ($s0,&DWP(16,$key));
1378         &xor     ($s1,&DWP(20,$key));
1379         &xor     ($s2,&DWP(24,$key));
1380         &xor     ($s3,&DWP(28,$key));
```



```
1382         &ret    ();
1383 &function_end_B("_x86_AES_decrypt_compact");

1385 ########################################################################
1386 # "Compact" SSE block function.
1387 ########################################################################

1389 sub sse_deccompact()
1390 {
1391         &pshufw ("mm1","mm0",0x0c);             #  7, 6, 1, 0
1392         &movd   ("eax","mm1");                  #  7, 6, 1, 0

1394         &pshufw ("mm5","mm4",0x09);             # 13,12,11,10
1395         &movz   ($acc,&LB("eax"));              #  0
1396         &movz   ("ecx",&BP(-128,$tbl,$acc,1));  #  0
1397         &movd   ("ebx","mm5");                  # 13,12,11,10
1398         &movz   ("edx",&HB("eax"));             #  1
1399         &movz   ("edx",&BP(-128,$tbl,"edx",1)); #  1
1400         &shl    ("edx",8);                      #  1

1402         &pshufw ("mm2","mm0",0x06);             #  3, 2, 5, 4
1403         &movz   ($acc,&LB("ebx"));              # 10
1404         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 10
1405         &shl    ($acc,16);                      # 10
1406         &or     ("ecx",$acc);                   # 10
1407         &shr    ("eax",16);                     #  7, 6
1408         &movz   ($acc,&HB("ebx"));              # 11
1409         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 11
1410         &shl    ($acc,24);                      # 11
1411         &or     ("edx",$acc);                   # 11
1412         &shr    ("ebx",16);                     # 13,12

1414         &pshufw ("mm6","mm4",0x03);             # 9, 8,15,14
1415         &movz   ($acc,&HB("eax"));              #  7
1416         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  7
1417         &shl    ($acc,24);                      #  7
1418         &or     ("ecx",$acc);                   #  7
1419         &movz   ($acc,&HB("ebx"));              # 13
1420         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 13
1421         &shl    ($acc,8);                       # 13
1422         &or     ("ecx",$acc);                   # 13
1423         &movd   ("mm0","ecx");                  # t[0] collected

1425         &movz   ($acc,&LB("eax"));              #  6
1426         &movd   ("eax","mm2");                  #  3, 2, 5, 4
1427         &movz   ("ecx",&BP(-128,$tbl,$acc,1));  #  6
1428         &shl    ("ecx",16);                     #  6
1429         &movz   ($acc,&LB("ebx"));              # 12
1430         &movd   ("ebx","mm6");                  #  9, 8,15,14
1431         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 12
1432         &or     ("ecx",$acc);                   # 12

1434         &movz   ($acc,&LB("eax"));              #  4
1435         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  4
1436         &or     ("edx",$acc);                   #  4
1437         &movz   ($acc,&LB("ebx"));              # 14
1438         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 14
1439         &shl    ($acc,16);                      # 14
1440         &or     ("edx",$acc);                   # 14
1441         &movd   ("mm1","edx");                  # t[1] collected

1443         &movz   ($acc,&HB("eax"));              #  5
1444         &movz   ("edx",&BP(-128,$tbl,$acc,1));  #  5
1445         &shl    ("edx",8);                      #  5
1446         &movz   ($acc,&HB("ebx"));              # 15
1447         &shr    ("eax",16);                     #  3, 2
```

```
1448         &movz   ($acc,&BP(-128,$tbl,$acc,1));   # 15
1449         &shl    ($acc,24);                      # 15
1450         &or     ("edx",$acc);                   # 15
1451         &shr    ("ebx",16);                     #  9, 8

1453         &punpckldq      ("mm0","mm1");          # t[0,1] collected

1455         &movz   ($acc,&HB("ebx"));              #  9
1456         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  9
1457         &shl    ($acc,8);                       #  9
1458         &or     ("ecx",$acc);                   #  9
1459         &and    ("ebx",0xff);                   #  8
1460         &movz   ("ebx",&BP(-128,$tbl,"ebx",1)); #  8
1461         &or     ("edx","ebx");                  #  8
1462         &movz   ($acc,&LB("eax"));              #  2
1463         &movz   ($acc,&BP(-128,$tbl,$acc,1));   #  2
1464         &shl    ($acc,16);                      #  2
1465         &or     ("edx",$acc);                   #  2
1466         &movd   ("mm4","edx");                  # t[2] collected
1467         &movz   ("eax",&HB("eax"));             #  3
1468         &movz   ("eax",&BP(-128,$tbl,"eax",1)); #  3
1469         &shl    ("eax",24);                     #  3
1470         &or     ("ecx","eax");                  #  3
1471         &movd   ("mm5","ecx");                  # t[3] collected

1473         &punpckldq      ("mm4","mm5");          # t[2,3] collected
1474 }

1476                                     if (!$x86only) {
1477 &function_begin_B("_sse_AES_decrypt_compact");
1478         &pxor   ("mm0",&QWP(0,$key));   #  7, 6, 5, 4, 3, 2, 1, 0
1479         &pxor   ("mm4",&QWP(8,$key));   # 15,14,13,12,11,10, 9, 8

1481         # note that caller is expected to allocate stack frame for me!
1482         &mov    ($acc,&DWP(240,$key));          # load key->rounds
1483         &lea    ($acc,&DWP(-2,$acc,$acc));
1484         &lea    ($acc,&DWP(0,$key,$acc,8));
1485         &mov    ($__end,$acc);                  # end of key schedule

1487         &mov    ($s0,0x1b1b1b1b);               # magic constant
1488         &mov    (&DWP(8,"esp"),$s0);
1489         &mov    (&DWP(12,"esp"),$s0);

1491         # prefetch Td4
1492         &mov    ($s0,&DWP(0-128,$tbl));
1493         &mov    ($s1,&DWP(32-128,$tbl));
1494         &mov    ($s2,&DWP(64-128,$tbl));
1495         &mov    ($s3,&DWP(96-128,$tbl));
1496         &mov    ($s0,&DWP(128-128,$tbl));
1497         &mov    ($s1,&DWP(160-128,$tbl));
1498         &mov    ($s2,&DWP(192-128,$tbl));
1499         &mov    ($s3,&DWP(224-128,$tbl));

1501         &set_label("loop",16);
1502                 &sse_deccompact();
1503                 &add    ($key,16);
1504                 &cmp    ($key,$__end);
1505                 &ja     (&label("out"));

1507                 # ROTATE(x^y,N) == ROTATE(x,N)^ROTATE(y,N)
1508                 &movq   ("mm3","mm0");          &movq   ("mm7","mm4");
1509                 &movq   ("mm2","mm0",1);        &movq   ("mm6","mm4",1);
1510                 &movq   ("mm1","mm0");          &movq   ("mm5","mm4");
1511                 &pshufw ("mm0","mm0",0xb1);     &pshufw ("mm4","mm4",0xb1);# = R
1512                 &pslld  ("mm2",8);              &pslld  ("mm6",8);
1513                 &psrld  ("mm3",8);              &psrld  ("mm7",8);
```



```
1514                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");  # ^= tp0
1515                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= tp0
1516                 &pslld  ("mm2",16);             &pslld  ("mm6",16);
1517                 &psrld  ("mm3",16);             &psrld  ("mm7",16);
1518                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");  # ^= tp0
1519                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= tp0

1521                 &movq   ("mm3",&QWP(8,"esp"));
1522                 &pxor   ("mm2","mm2");          &pxor   ("mm6","mm6");
1523                 &pcmpgtb("mm2","mm1");          &pcmpgtb("mm6","mm5");
1524                 &pand   ("mm2","mm3");          &pand   ("mm6","mm3");
1525                 &paddb  ("mm1","mm1");          &paddb  ("mm5","mm5");
1526                 &pxor   ("mm1","mm2");          &pxor   ("mm5","mm6");  # tp2
1527                 &movq   ("mm3","mm1");          &movq   ("mm7","mm5");
1528                 &movq   ("mm2","mm1");          &movq   ("mm6","mm5");
1529                 &pxor   ("mm0","mm1");          &pxor   ("mm4","mm5");  # ^= tp2
1530                 &pslld  ("mm3",24);             &pslld  ("mm7",24);
1531                 &psrld  ("mm2",8);              &psrld  ("mm6",8);
1532                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= tp2
1533                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");  # ^= tp2

1535                 &movq   ("mm2",&QWP(8,"esp"));
1536                 &pxor   ("mm3","mm3");          &pxor   ("mm7","mm7");
1537                 &pcmpgtb("mm3","mm1");          &pcmpgtb("mm7","mm5");
1538                 &pand   ("mm3","mm2");          &pand   ("mm7","mm2");
1539                 &paddb  ("mm1","mm1");          &paddb  ("mm5","mm5");
1540                 &pxor   ("mm1","mm3");          &pxor   ("mm5","mm7");  # tp4
1541                 &pshufw ("mm3","mm1",0xb1);     &pshufw ("mm7","mm5",0xb1);
1542                 &pxor   ("mm0","mm1");          &pxor   ("mm4","mm5");  # ^= tp4
1543                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= ROT

1545                 &pxor   ("mm3","mm3");          &pxor   ("mm7","mm7");
1546                 &pcmpgtb("mm3","mm1");          &pcmpgtb("mm7","mm5");
1547                 &pand   ("mm3","mm2");          &pand   ("mm7","mm2");
1548                 &paddb  ("mm1","mm1");          &paddb  ("mm5","mm5");
1549                 &pxor   ("mm1","mm3");          &pxor   ("mm5","mm7");  # tp8
1550                 &pxor   ("mm0","mm1");          &pxor   ("mm4","mm5");  # ^= tp8
1551                 &movq   ("mm3","mm1");          &movq   ("mm7","mm5");
1552                 &pshufw ("mm2","mm1",0xb1);     &pshufw ("mm6","mm5",0xb1);
1553                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");  # ^= ROT
1554                 &pslld  ("mm1",8);              &pslld  ("mm5",8);
1555                 &psrld  ("mm3",8);              &psrld  ("mm7",8);
1556                 &movq   ("mm2",&QWP(0,$key));   &movq   ("mm6",&QWP(8,$key));
1557                 &pxor   ("mm0","mm1");          &pxor   ("mm4","mm5");  # ^= tp8
1558                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= tp8
1559                 &mov    ($s0,&DWP(0-128,$tbl));
1560                 &pslld  ("mm1",16);             &pslld  ("mm5",16);
1561                 &mov    ($s1,&DWP(64-128,$tbl));
1562                 &psrld  ("mm3",16);             &psrld  ("mm7",16);
1563                 &mov    ($s2,&DWP(128-128,$tbl));
1564                 &pxor   ("mm0","mm1");          &pxor   ("mm4","mm5");  # ^= tp8
1565                 &mov    ($s3,&DWP(192-128,$tbl));
1566                 &pxor   ("mm0","mm3");          &pxor   ("mm4","mm7");  # ^= tp8

1568                 &pxor   ("mm0","mm2");          &pxor   ("mm4","mm6");
1569         &jmp    (&label("loop"));

1571         &set_label("out",16);
1572         &pxor   ("mm0",&QWP(0,$key));
1573         &pxor   ("mm4",&QWP(8,$key));

1575         &ret    ();
1576 &function_end_B("_sse_AES_decrypt_compact");
1577                                     }

1579 ########################################################################
```

```
1580 # Vanilla block function.
1581 ########################################################################

1583 sub decstep()
1584 { my ($i,$td,@s) = @_;
1585   my $tmp = $key;
1586   my $out = $i==3?$s[0]:$acc;

1588         # no instructions are reordered, as performance appears
1589         # optimal... or rather that all attempts to reorder didn't
1590         # result in better performance [which by the way is not a
1591         # bit lower than ecryption].
1592         if($i==3)   {   &mov    ($key,$__key);                  }
1593         else        {   &mov    ($out,$s[0]);                   }
1594                         &and    ($out,0xFF);
1595                         &mov    ($out,&DWP(0,$td,$out,8));

1597         if ($i==3)  {   $tmp=$s[1];                             }
1598                         &movz   ($tmp,&HB($s[1]));
1599                         &xor    ($out,&DWP(3,$td,$tmp,8));

1601         if ($i==3)  {   $tmp=$s[2]; &mov ($s[1],$acc);          }
1602         else        {   &mov    ($tmp,$s[2]);                   }
1603                         &shr    ($tmp,16);
1604                         &and    ($tmp,0xFF);
1605                         &xor    ($out,&DWP(2,$td,$tmp,8));

1607         if ($i==3)  {   $tmp=$s[3]; &mov ($s[2],$__s1);         }
1608         else        {   &mov    ($tmp,$s[3]);                   }
1609                         &shr    ($tmp,24);
1610                         &xor    ($out,&DWP(1,$td,$tmp,8));
1611         if ($i<2)   {   &mov    (&DWP(4+4*$i,"esp"),$out);      }
1612         if ($i==3)  {   &mov    ($s[3],$__s0);                  }
1613                         &comment();
1614 }

1616 sub declast()
1617 { my ($i,$td,@s)=@_;
1618   my $tmp = $key;
1619   my $out = $i==3?$s[0]:$acc;

1621         if($i==0)   {   &lea    ($td,&DWP(2048+128,$td));
1622                         &mov    ($tmp,&DWP(0-128,$td));
1623                         &mov    ($acc,&DWP(32-128,$td));
1624                         &mov    ($tmp,&DWP(64-128,$td));
1625                         &mov    ($acc,&DWP(96-128,$td));
1626                         &mov    ($tmp,&DWP(128-128,$td));
1627                         &mov    ($acc,&DWP(160-128,$td));
1628                         &mov    ($tmp,&DWP(192-128,$td));
1629                         &mov    ($acc,&DWP(224-128,$td));
1630                         &lea    ($td,&DWP(-128,$td));           }
1631         if($i==3)   {   &mov    ($key,$__key);                  }
1632         else        {   &mov    ($out,$s[0]);                   }
1633                         &and    ($out,0xFF);
1634                         &movz   ($out,&BP(0,$td,$out,1));

1636         if ($i==3)  {   $tmp=$s[1];                             }
1637                         &movz   ($tmp,&HB($s[1]));
1638                         &movz   ($tmp,&BP(0,$td,$tmp,1));
1639                         &shl    ($tmp,8);
1640                         &xor    ($out,$tmp);

1642         if ($i==3)  {   $tmp=$s[2]; &mov ($s[1],$acc);          }
1643         else        {   mov     ($tmp,$s[2]);                   }
1644                         &shr    ($tmp,16);
1645                         &and    ($tmp,0xFF);
```



```
1646                         &movz   ($tmp,&BP(0,$td,$tmp,1));
1647                         &shl    ($tmp,16);
1648                         &xor    ($out,$tmp);

1650         if ($i==3)  {   $tmp=$s[3]; &mov ($s[2],$__s1);         }
1651         else        {   &mov    ($tmp,$s[3]);                   }
1652                         &shr    ($tmp,24);
1653                         &movz   ($tmp,&BP(0,$td,$tmp,1));
1654                         &shl    ($tmp,24);
1655                         &xor    ($out,$tmp);
1656         if ($i<2)   {   &mov    (&DWP(4+4*$i,"esp"),$out);      }
1657         if ($i==3)  {   &mov    ($s[3],$__s0);
1658                         &lea    ($td,&DWP(-2048,$td));          }
1659 }

1661 &function_begin_B("_x86_AES_decrypt");
1662         # note that caller is expected to allocate stack frame for me!
1663         &mov    ($__key,$key);                  # save key

1665         &xor    ($s0,&DWP(0,$key));             # xor with key
1666         &xor    ($s1,&DWP(4,$key));
1667         &xor    ($s2,&DWP(8,$key));
1668         &xor    ($s3,&DWP(12,$key));

1670         &mov    ($acc,&DWP(240,$key));          # load key->rounds

1672         if ($small_footprint) {
1673             &lea        ($acc,&DWP(-2,$acc,$acc));
1674             &lea        ($acc,&DWP(0,$key,$acc,8));
1675             &mov        ($__end,$acc);          # end of key schedule
1676             &set_label("loop",16);
1677                 &decstep(0,$tbl,$s0,$s3,$s2,$s1);
1678                 &decstep(1,$tbl,$s1,$s0,$s3,$s2);
1679                 &decstep(2,$tbl,$s2,$s1,$s0,$s3);
1680                 &decstep(3,$tbl,$s3,$s2,$s1,$s0);
1681                 &add    ($key,16);              # advance rd_key
1682                 &xor    ($s0,&DWP(0,$key));
1683                 &xor    ($s1,&DWP(4,$key));
1684                 &xor    ($s2,&DWP(8,$key));
1685                 &xor    ($s3,&DWP(12,$key));
1686             &cmp        ($key,$__end);
1687             &mov        ($__key,$key);
1688             &jb         (&label("loop"));
1689         }
1690         else {
1691             &cmp        ($acc,10);
1692             &jle        (&label("10rounds"));
1693             &cmp        ($acc,12);
1694             &jle        (&label("12rounds"));

1696         &set_label("14rounds",4);
1697             for ($i=1;$i<3;$i++) {
1698                 &decstep(0,$tbl,$s0,$s3,$s2,$s1);
1699                 &decstep(1,$tbl,$s1,$s0,$s3,$s2);
1700                 &decstep(2,$tbl,$s2,$s1,$s0,$s3);
1701                 &decstep(3,$tbl,$s3,$s2,$s1,$s0);
1702                 &xor    ($s0,&DWP(16*$i+0,$key));
1703                 &xor    ($s1,&DWP(16*$i+4,$key));
1704                 &xor    ($s2,&DWP(16*$i+8,$key));
1705                 &xor    ($s3,&DWP(16*$i+12,$key));
1706             }
1707             &add        ($key,32);
1708             &mov        ($__key,$key);          # advance rd_key
1709         &set_label("12rounds",4);
1710             for ($i=1;$i<3;$i++) {
1711                 &decstep(0,$tbl,$s0,$s3,$s2,$s1);
```

```
1712                &decstep(1,$tbl,$s1,$s0,$s3,$s2);
1713                &decstep(2,$tbl,$s2,$s1,$s0,$s3);
1714                &decstep(3,$tbl,$s3,$s2,$s1,$s0);
1715                &xor    ($s0,&DWP(16*$i+0,$key));
1716                &xor    ($s1,&DWP(16*$i+4,$key));
1717                &xor    ($s2,&DWP(16*$i+8,$key));
1718                &xor    ($s3,&DWP(16*$i+12,$key));
1719            }
1720            &add        ($key,32);
1721            &mov        ($__key,$key);          # advance rd_key
1722        &set_label("10rounds",4);
1723            for ($i=1;$i<10;$i++) {
1724                &decstep(0,$tbl,$s0,$s3,$s2,$s1);
1725                &decstep(1,$tbl,$s1,$s0,$s3,$s2);
1726                &decstep(2,$tbl,$s2,$s1,$s0,$s3);
1727                &decstep(3,$tbl,$s3,$s2,$s1,$s0);
1728                &xor    ($s0,&DWP(16*$i+0,$key));
1729                &xor    ($s1,&DWP(16*$i+4,$key));
1730                &xor    ($s2,&DWP(16*$i+8,$key));
1731                &xor    ($s3,&DWP(16*$i+12,$key));
1732            }
1733        }

1735        &declast(0,$tbl,$s0,$s3,$s2,$s1);
1736        &declast(1,$tbl,$s1,$s0,$s3,$s2);
1737        &declast(2,$tbl,$s2,$s1,$s0,$s3);
1738        &declast(3,$tbl,$s3,$s2,$s1,$s0);

1740        &add    ($key,$small_footprint?16:160);
1741        &xor    ($s0,&DWP(0,$key));
1742        &xor    ($s1,&DWP(4,$key));
1743        &xor    ($s2,&DWP(8,$key));
1744        &xor    ($s3,&DWP(12,$key));

1746        &ret    ();

1748 &set_label("AES_Td",64);       # Yes! I keep it in the code segment!
1749        &_data_word(0x50a7f451, 0x5365417e, 0xc3a4171a, 0x965e273a);
1750        &_data_word(0xcb6bab3b, 0xf1459d1f, 0xab58faac, 0x9303e34b);
1751        &_data_word(0x55fa3020, 0xf66d76ad, 0x9176cc88, 0x254c02f5);
1752        &_data_word(0xfcd7e54f, 0xd7cb2ac5, 0x80443526, 0x8fa362b5);
1753        &_data_word(0x495ab1de, 0x671bba25, 0x980eea45, 0xe1c0fe5d);
1754        &_data_word(0x02752fc3, 0x12f04c81, 0xa397468d, 0xc6f9d36b);
1755        &_data_word(0xe75f8f03, 0x959c9215, 0xeb7a6dbf, 0xda595295);
1756        &_data_word(0x2d83bed4, 0xd3217458, 0x2969e049, 0x44c8c98e);
1757        &_data_word(0x6a89c275, 0x78798ef4, 0x6b3e5899, 0xdd71b927);
1758        &_data_word(0xb64fe1be, 0x17ad88f0, 0x66ac20c9, 0xb43ace7d);
1759        &_data_word(0x184adf63, 0x82311ae5, 0x60335197, 0x457f5362);
1760        &_data_word(0xe07764b1, 0x84ae6bbb, 0x1ca081fe, 0x942b08f9);
1761        &_data_word(0x58684870, 0x19fd458f, 0x876cde94, 0xb7f87b52);
1762        &_data_word(0x23d373ab, 0xe2024b72, 0x578f1fe3, 0x2aab5566);
1763        &_data_word(0x0728ebb2, 0x03c2b52f, 0x9a7bc586, 0xa50837d3);
1764        &_data_word(0xf2872830, 0xb2a5bf23, 0xba6a0302, 0x5c8216ed);
1765        &_data_word(0x2b1ccf8a, 0x92b479a7, 0xf0f207f3, 0xa1e2694e);
1766        &_data_word(0xcdf4da65, 0xd5be0506, 0x1f6234d1, 0x8afea6c4);
1767        &_data_word(0x9d532e34, 0xa055f3a2, 0x32e18a05, 0x75ebf6a4);
1768        &_data_word(0x39ec830b, 0xaaef6040, 0x069f715e, 0x51106ebd);
1769        &_data_word(0xf98a213e, 0x3d06dd96, 0xae053edd, 0x46bde64d);
1770        &_data_word(0xb58d5491, 0x055dc471, 0x6fd40604, 0xff155060);
1771        &_data_word(0x24fb9819, 0x97e9bdd6, 0xcc434089, 0x779ed967);
1772        &_data_word(0xbd42e8b0, 0x888b8907, 0x385b19e7, 0xdbeec879);
1773        &_data_word(0x470a7ca1, 0xe90f427c, 0xc91e84f8, 0x00000000);
1774        &_data_word(0x83868009, 0x48ed2b32, 0xac70111e, 0x4e725a6c);
1775        &_data_word(0xfbff0efd, 0x5638850f, 0x1ed5ae3d, 0x27392d36);
1776        &_data_word(0x64d90f0a, 0x21a65c68, 0xd1545b9b, 0x3a2e3624);
1777        &_data_word(0xb1670a0c, 0x0fe75793, 0xd296eeb4, 0x9e919b1b);
```



```
1778        &_data_word(0x4fc5c080, 0xa220dc61, 0x694b775a, 0x161a121c);
1779        &_data_word(0x0aba93e2, 0xe52aa0c0, 0x43e0223c, 0x1d171b12);
1780        &_data_word(0x0b0d090e, 0xadc78bf2, 0xb9a8b62d, 0xc8a91e14);
1781        &_data_word(0x8519f157, 0x4c0775af, 0xbbdd99ee, 0xfd607fa3);
1782        &_data_word(0x9f2601f7, 0xbcf5725c, 0xc53b6644, 0x347efb5b);
1783        &_data_word(0x7629438b, 0xdcc623cb, 0x68fcedb6, 0x63f1e4b8);
1784        &_data_word(0xcadc31d7, 0x10856342, 0x40229713, 0x2011c684);
1785        &_data_word(0x7d244a85, 0xf83dbbd2, 0x1132f9ae, 0x6da129c7);
1786        &_data_word(0x4b2f9e1d, 0xf330b2dc, 0xec52860d, 0xd0e3c177);
1787        &_data_word(0x6c16b32b, 0x99b970a9, 0xfa489411, 0x2264e947);
1788        &_data_word(0xc48cfca8, 0x1a3ff0a0, 0xd82c7d56, 0xef903322);
1789        &_data_word(0xc74e4987, 0xc1d138d9, 0xfea2ca8c, 0x360bd498);
1790        &_data_word(0xcf81f5a6, 0x28de7aa5, 0x268eb7da, 0xa4bfad3f);
1791        &_data_word(0xe49d3a2c, 0x0d927850, 0x9bcc5f6a, 0x62467e54);
1792        &_data_word(0xc2138df6, 0xe8b8d890, 0x5ef7392e, 0xf5afc382);
1793        &_data_word(0xbe805d9f, 0x7c93d069, 0xa92dd56f, 0xb31225cf);
1794        &_data_word(0x3b99acc8, 0xa77d1810, 0x6e639ce8, 0x7bbb3bdb);
1795        &_data_word(0x097826cd, 0xf418596e, 0x01b79aec, 0xa89a4f83);
1796        &_data_word(0x656e95e6, 0x7ee6ffaa, 0x08cfbc21, 0xe6e815ef);
1797        &_data_word(0xd99be7ba, 0xce366f4a, 0xd4099fea, 0xd67cb029);
1798        &_data_word(0xafb2a431, 0x31233f2a, 0x3094a5c6, 0xc066a235);
1799        &_data_word(0x37bc4e74, 0xa6ca82fc, 0xb0d090e0, 0x15d8a733);
1800        &_data_word(0x4a9804f1, 0xf7daec41, 0x0e50cd7f, 0x2ff69117);
1801        &_data_word(0x8dd64d76, 0x4db0ef43, 0x544daacc, 0xdf0496e4);
1802        &_data_word(0xe3b5d19e, 0x1b886a4c, 0xb81f2cc1, 0x7f516546);
1803        &_data_word(0x04ea5e9d, 0x5d358c01, 0x737487fa, 0x2e410bfb);
1804        &_data_word(0x5a1d67b3, 0x52d2db92, 0x335610e9, 0x1347d66d);
1805        &_data_word(0x8c61d79a, 0x7a0ca137, 0x8e14f859, 0x893c13eb);
1806        &_data_word(0xee27a9ce, 0x35c961b7, 0xede51ce1, 0x3cb1477a);
1807        &_data_word(0x59dfd29c, 0x3f73f255, 0x79ce1418, 0xbf37c773);
1808        &_data_word(0xeacdf753, 0x5baafd5f, 0x146f3ddf, 0x86db4478);
1809        &_data_word(0x81f3afca, 0x3ec468b9, 0x2c342438, 0x5f40a3c2);
1810        &_data_word(0x72c31d16, 0x0c25e2bc, 0x8b493c28, 0x41950dff);
1811        &_data_word(0x7101a839, 0xdeb30c08, 0x9ce4b4d8, 0x90c15664);
1812        &_data_word(0x6184cb7b, 0x70b632d5, 0x745c6c48, 0x4257b8d0);

1814 #Td4:   # four copies of Td4 to choose from to avoid L1 aliasing
1815        &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
1816        &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
1817        &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
1818        &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
1819        &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
1820        &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
1821        &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
1822        &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
1823        &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
1824        &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
1825        &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
1826        &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
1827        &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
1828        &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
1829        &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
1830        &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
1831        &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
1832        &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
1833        &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
1834        &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
1835        &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
1836        &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
1837        &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
1838        &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
1839        &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
1840        &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
1841        &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
1842        &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
1843        &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
```

```
1844         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
1845         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
1846         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);

1848         &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
1849         &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
1850         &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
1851         &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
1852         &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
1853         &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
1854         &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
1855         &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
1856         &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
1857         &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
1858         &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
1859         &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
1860         &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
1861         &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
1862         &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
1863         &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
1864         &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
1865         &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
1866         &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
1867         &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
1868         &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
1869         &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
1870         &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
1871         &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
1872         &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
1873         &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
1874         &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
1875         &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
1876         &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
1877         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
1878         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
1879         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);

1881         &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
1882         &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
1883         &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
1884         &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
1885         &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
1886         &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
1887         &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
1888         &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
1889         &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
1890         &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
1891         &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
1892         &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
1893         &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
1894         &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
1895         &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
1896         &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
1897         &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
1898         &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
1899         &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
1900         &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
1901         &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
1902         &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
1903         &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
1904         &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
1905         &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
1906         &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
1907         &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
1908         &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
1909         &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
```



```
1910         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
1911         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
1912         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);

1914         &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
1915         &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
1916         &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
1917         &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
1918         &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
1919         &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
1920         &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
1921         &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
1922         &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
1923         &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
1924         &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
1925         &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
1926         &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
1927         &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
1928         &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
1929         &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
1930         &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
1931         &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
1932         &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
1933         &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
1934         &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
1935         &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
1936         &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
1937         &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
1938         &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
1939         &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
1940         &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
1941         &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
1942         &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
1943         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
1944         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
1945         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);
1946 &function_end_B("_x86_AES_decrypt");

1948 # void AES_decrypt (const void *inp,void *out,const AES_KEY *key);
1949 &function_begin("AES_decrypt");
1950         &mov    ($acc,&wparam(0));              # load inp
1951         &mov    ($key,&wparam(2));              # load key

1953         &mov    ($s0,"esp");
1954         &sub    ("esp",36);
1955         &and    ("esp",-64);                    # align to cache-line

1957         # place stack frame just "above" the key schedule
1958         &lea    ($s1,&DWP(-64-63,$key));
1959         &sub    ($s1,"esp");
1960         &neg    ($s1);
1961         &and    ($s1,0x3C0);    # modulo 1024, but aligned to cache-line
1962         &sub    ("esp",$s1);
1963         &add    ("esp",4);      # 4 is reserved for caller's return address
1964         &mov    ($_esp,$s0);    # save stack pointer

1966         &call   (&label("pic_point"));          # make it PIC!
1967         &set_label("pic_point");
1968         &blindpop($tbl);
1969         &picmeup($s0,"OPENSSL_ia32cap_P",$tbl,&label("pic_point")) if(!$x86only)
1970         &lea    ($tbl,&DWP(&label("AES_Td")."-".&label("pic_point"),$tbl));

1972         # pick Td4 copy which can't "overlap" with stack frame or key schedule
1973         &lea    ($s1,&DWP(768-4,"esp"));
1974         &sub    ($s1,$tbl);
1975         &and    ($s1,0x300);
```

```
1976         &lea    ($tbl,&DWP(2048+128,$tbl,$s1));

1978                                 if (!$x86only) {
1979         &bt     (&DWP(0,$s0),25);       # check for SSE bit
1980         &jnc    (&label("x86"));

1982         &movq   ("mm0",&QWP(0,$acc));
1983         &movq   ("mm4",&QWP(8,$acc));
1984         &call   ("_sse_AES_decrypt_compact");
1985         &mov    ("esp",$_esp);                  # restore stack pointer
1986         &mov    ($acc,&wparam(1));              # load out
1987         &movq   (&QWP(0,$acc),"mm0");           # write output data
1988         &movq   (&QWP(8,$acc),"mm4");
1989         &emms   ();
1990         &function_end_A();
1991                                 }
1992         &set_label("x86",16);
1993         &mov    ($_tbl,$tbl);
1994         &mov    ($s0,&DWP(0,$acc));             # load input data
1995         &mov    ($s1,&DWP(4,$acc));
1996         &mov    ($s2,&DWP(8,$acc));
1997         &mov    ($s3,&DWP(12,$acc));
1998         &call   ("_x86_AES_decrypt_compact");
1999         &mov    ("esp",$_esp);                  # restore stack pointer
2000         &mov    ($acc,&wparam(1));              # load out
2001         &mov    (&DWP(0,$acc),$s0);             # write output data
2002         &mov    (&DWP(4,$acc),$s1);
2003         &mov    (&DWP(8,$acc),$s2);
2004         &mov    (&DWP(12,$acc),$s3);
2005 &function_end("AES_decrypt");

2007 # void AES_cbc_encrypt (const void char *inp, unsigned char *out,
2008 #                       size_t length, const AES_KEY *key,
2009 #                       unsigned char *ivp,const int enc);
2010 {
2011 # stack frame layout
2012 #             -4(%esp)          # return address         0(%esp)
2013 #              0(%esp)          # s0 backing store       4(%esp)
2014 #              4(%esp)          # s1 backing store       8(%esp)
2015 #              8(%esp)          # s2 backing store      12(%esp)
2016 #             12(%esp)          # s3 backing store      16(%esp)
2017 #             16(%esp)          # key backup            20(%esp)
2018 #             20(%esp)          # end of key schedule   24(%esp)
2019 #             24(%esp)          # %ebp backup           28(%esp)
2020 #             28(%esp)          # %esp backup
2021 my $_inp=&DWP(32,"esp");        # copy of wparam(0)
2022 my $_out=&DWP(36,"esp");        # copy of wparam(1)
2023 my $_len=&DWP(40,"esp");        # copy of wparam(2)
2024 my $_key=&DWP(44,"esp");        # copy of wparam(3)
2025 my $_ivp=&DWP(48,"esp");        # copy of wparam(4)
2026 my $_tmp=&DWP(52,"esp");        # volatile variable
2027 #
2028 my $ivec=&DWP(60,"esp");        # ivec[16]
2029 my $aes_key=&DWP(76,"esp");     # copy of aes_key
2030 my $mark=&DWP(76+240,"esp");    # copy of aes_key->rounds

2032 &function_begin("AES_cbc_encrypt");
2033         &mov    ($s2 eq "ecx"? $s2 : "",&wparam(2));    # load len
2034         &cmp    ($s2,0);
2035         &je     (&label("drop_out"));

2037         &call   (&label("pic_point"));          # make it PIC!
2038         &set_label("pic_point");
2039         &blindpop($tbl);
2040         &picmeup($s0,"OPENSSL_ia32cap_P",$tbl,&label("pic_point")) if(!$x86only)
```



```
2042         &cmp    (&wparam(5),0);
2043         &lea    ($tbl,&DWP(&label("AES_Te")."-".&label("pic_point"),$tbl));
2044         &jne    (&label("picked_te"));
2045         &lea    ($tbl,&DWP(&label("AES_Td")."-".&label("AES_Te"),$tbl));
2046         &set_label("picked_te");

2048         # one can argue if this is required
2049         &pushf  ();
2050         &cld    ();

2052         &cmp    ($s2,$speed_limit);
2053         &jb     (&label("slow_way"));
2054         &test   ($s2,15);
2055         &jnz    (&label("slow_way"));
2056                                 if (!$x86only) {
2057         &bt     (&DWP(0,$s0),28);       # check for hyper-threading bit
2058         &jc     (&label("slow_way"));
2059                                 }
2060         # pre-allocate aligned stack frame...
2061         &lea    ($acc,&DWP(-80-244,"esp"));
2062         &and    ($acc,-64);

2064         # ... and make sure it doesn't alias with $tbl modulo 4096
2065         &mov    ($s0,$tbl);
2066         &lea    ($s1,&DWP(2048+256,$tbl));
2067         &mov    ($s3,$acc);
2068         &and    ($s0,0xfff);            # s = %ebp&0xfff
2069         &and    ($s1,0xfff);            # e = (%ebp+2048+256)&0xfff
2070         &and    ($s3,0xfff);            # p = %esp&0xfff

2072         &cmp    ($s3,$s1);              # if (p>=e) %esp =- (p-e);
2073         &jb     (&label("tbl_break_out"));
2074         &sub    ($s3,$s1);
2075         &sub    ($acc,$s3);
2076         &jmp    (&label("tbl_ok"));
2077         &set_label("tbl_break_out",4);  # else %esp -= (p-s)&0xfff + framesz;
2078         &sub    ($s3,$s0);
2079         &and    ($s3,0xfff);
2080         &add    ($s3,384);
2081         &sub    ($acc,$s3);
2082         &set_label("tbl_ok",4);

2084         &lea    ($s3,&wparam(0));       # obtain pointer to parameter block
2085         &exch   ("esp",$acc);           # allocate stack frame
2086         &add    ("esp",4);              # reserve for return address!
2087         &mov    ($_tbl,$tbl);           # save %ebp
2088         &mov    ($_esp,$acc);           # save %esp

2090         &mov    ($s0,&DWP(0,$s3));      # load inp
2091         &mov    ($s1,&DWP(4,$s3));      # load out
2092         #&mov   ($s2,&DWP(8,$s3));      # load len
2093         &mov    ($key,&DWP(12,$s3));    # load key
2094         &mov    ($acc,&DWP(16,$s3));    # load ivp
2095         &mov    ($s3,&DWP(20,$s3));     # load enc flag

2097         &mov    ($_inp,$s0);            # save copy of inp
2098         &mov    ($_out,$s1);            # save copy of out
2099         &mov    ($_len,$s2);            # save copy of len
2100         &mov    ($_key,$key);           # save copy of key
2101         &mov    ($_ivp,$acc);           # save copy of ivp

2103         &mov    ($mark,0);              # copy of aes_key->rounds = 0;
2104         # do we copy key schedule to stack?
2105         &mov    ($s1 eq "ebx" ? $s1 : "",$key);
2106         &mov    ($s2 eq "ecx" ? $s2 : "",244/4);
2107         &sub    ($s1,$tbl);
```

```
2108         &mov    ("esi",$key);
2109         &and    ($s1,0xfff);
2110         &lea    ("edi",$aes_key);
2111         &cmp    ($s1,2048+256);
2112         &jb     (&label("do_copy"));
2113         &cmp    ($s1,4096-244);
2114         &jb     (&label("skip_copy"));
2115         &set_label("do_copy",4);
2116                 &mov    ($_key,"edi");
2117                 &data_word(0xA5F3F689); # rep movsd
2118         &set_label("skip_copy");
2119
2120         &mov    ($key,16);
2121         &set_label("prefetch_tbl",4);
2122                 &mov    ($s0,&DWP(0,$tbl));
2123                 &mov    ($s1,&DWP(32,$tbl));
2124                 &mov    ($s2,&DWP(64,$tbl));
2125                 &mov    ($acc,&DWP(96,$tbl));
2126                 &lea    ($tbl,&DWP(128,$tbl));
2127                 &sub    ($key,1);
2128         &jnz    (&label("prefetch_tbl"));
2129         &sub    ($tbl,2048);
2130
2131         &mov    ($acc,$_inp);
2132         &mov    ($key,$_ivp);
2133
2134         &cmp    ($s3,0);
2135         &je     (&label("fast_decrypt"));
2136
2137 #---------------------------- ENCRYPT -----------------------------#
2138         &mov    ($s0,&DWP(0,$key));             # load iv
2139         &mov    ($s1,&DWP(4,$key));
2140
2141         &set_label("fast_enc_loop",16);
2142                 &mov    ($s2,&DWP(8,$key));
2143                 &mov    ($s3,&DWP(12,$key));
2144
2145                 &xor    ($s0,&DWP(0,$acc));     # xor input data
2146                 &xor    ($s1,&DWP(4,$acc));
2147                 &xor    ($s2,&DWP(8,$acc));
2148                 &xor    ($s3,&DWP(12,$acc));
2149
2150                 &mov    ($key,$_key);           # load key
2151                 &call   ("_x86_AES_encrypt");
2152
2153                 &mov    ($acc,$_inp);           # load inp
2154                 &mov    ($key,$_out);           # load out
2155
2156                 &mov    (&DWP(0,$key),$s0);     # save output data
2157                 &mov    (&DWP(4,$key),$s1);
2158                 &mov    (&DWP(8,$key),$s2);
2159                 &mov    (&DWP(12,$key),$s3);
2160
2161                 &lea    ($acc,&DWP(16,$acc));   # advance inp
2162                 &mov    ($s2,$_len);            # load len
2163                 &mov    ($_inp,$acc);           # save inp
2164                 &lea    ($s3,&DWP(16,$key));    # advance out
2165                 &mov    ($_out,$s3);            # save out
2166                 &sub    ($s2,16);               # decrease len
2167                 &mov    ($_len,$s2);            # save len
2168         &jnz    (&label("fast_enc_loop"));
2169         &mov    ($acc,$_ivp);           # load ivp
2170         &mov    ($s2,&DWP(8,$key));     # restore last 2 dwords
2171         &mov    ($s3,&DWP(12,$key));
2172         &mov    (&DWP(0,$acc),$s0);     # save ivec
2173         &mov    (&DWP(4,$acc),$s1);
```



```
2174         &mov    (&DWP(8,$acc),$s2);
2175         &mov    (&DWP(12,$acc),$s3);
2176
2177         &cmp    ($mark,0);              # was the key schedule copied?
2178         &mov    ("edi",$_key);
2179         &je     (&label("skip_ezero"));
2180         # zero copy of key schedule
2181         &mov    ("ecx",240/4);
2182         &xor    ("eax","eax");
2183         &align  (4);
2184         &data_word(0xABF3F689); # rep stosd
2185         &set_label("skip_ezero")
2186         &mov    ("esp",$_esp);
2187         &popf   ();
2188     &set_label("drop_out");
2189         &function_end_A();
2190         &pushf  ();                     # kludge, never executed
2191
2192 #---------------------------- DECRYPT -----------------------------#
2193 &set_label("fast_decrypt",16);
2194
2195         &cmp    ($acc,$_out);
2196         &je     (&label("fast_dec_in_place"));  # in-place processing...
2197
2198         &mov    ($_tmp,$key);
2199
2200         &align  (4);
2201         &set_label("fast_dec_loop",16);
2202                 &mov    ($s0,&DWP(0,$acc));     # read input
2203                 &mov    ($s1,&DWP(4,$acc));
2204                 &mov    ($s2,&DWP(8,$acc));
2205                 &mov    ($s3,&DWP(12,$acc));
2206
2207                 &mov    ($key,$_key);           # load key
2208                 &call   ("_x86_AES_decrypt");
2209
2210                 &mov    ($key,$_tmp);           # load ivp
2211                 &mov    ($acc,$_len);           # load len
2212                 &xor    ($s0,&DWP(0,$key));     # xor iv
2213                 &xor    ($s1,&DWP(4,$key));
2214                 &xor    ($s2,&DWP(8,$key));
2215                 &xor    ($s3,&DWP(12,$key));
2216
2217                 &mov    ($key,$_out);           # load out
2218                 &mov    ($acc,$_inp);           # load inp
2219
2220                 &mov    (&DWP(0,$key),$s0);     # write output
2221                 &mov    (&DWP(4,$key),$s1);
2222                 &mov    (&DWP(8,$key),$s2);
2223                 &mov    (&DWP(12,$key),$s3);
2224
2225                 &mov    ($s2,$_len);            # load len
2226                 &mov    ($_tmp,$acc);           # save ivp
2227                 &lea    ($acc,&DWP(16,$acc));   # advance inp
2228                 &mov    ($_inp,$acc);           # save inp
2229                 &lea    ($key,&DWP(16,$key));   # advance out
2230                 &mov    ($_out,$key);           # save out
2231                 &sub    ($s2,16);               # decrease len
2232                 &mov    ($_len,$s2);            # save len
2233         &jnz    (&label("fast_dec_loop"));
2234         &mov    ($key,$_tmp);           # load temp ivp
2235         &mov    ($acc,$_ivp);           # load user ivp
2236         &mov    ($s0,&DWP(0,$key));     # load iv
2237         &mov    ($s1,&DWP(4,$key));
2238         &mov    ($s2,&DWP(8,$key));
2239         &mov    ($s3,&DWP(12,$key));
```

```
2240         &mov    (&DWP(0,$acc),$s0);     # copy back to user
2241         &mov    (&DWP(4,$acc),$s1);
2242         &mov    (&DWP(8,$acc),$s2);
2243         &mov    (&DWP(12,$acc),$s3);
2244         &jmp    (&label("fast_dec_out"));

2246     &set_label("fast_dec_in_place",16);
2247         &set_label("fast_dec_in_place_loop");
2248                 &mov    ($s0,&DWP(0,$acc));     # read input
2249                 &mov    ($s1,&DWP(4,$acc));
2250                 &mov    ($s2,&DWP(8,$acc));
2251                 &mov    ($s3,&DWP(12,$acc));

2253                 &lea    ($key,$ivec);
2254                 &mov    (&DWP(0,$key),$s0);     # copy to temp
2255                 &mov    (&DWP(4,$key),$s1);
2256                 &mov    (&DWP(8,$key),$s2);
2257                 &mov    (&DWP(12,$key),$s3);

2259                 &mov    ($key,$_key);           # load key
2260                 &call   ("_x86_AES_decrypt");

2262                 &mov    ($key,$_ivp);           # load ivp
2263                 &mov    ($acc,$_out);           # load out
2264                 &xor    ($s0,&DWP(0,$key));     # xor iv
2265                 &xor    ($s1,&DWP(4,$key));
2266                 &xor    ($s2,&DWP(8,$key));
2267                 &xor    ($s3,&DWP(12,$key));

2269                 &mov    (&DWP(0,$acc),$s0);     # write output
2270                 &mov    (&DWP(4,$acc),$s1);
2271                 &mov    (&DWP(8,$acc),$s2);
2272                 &mov    (&DWP(12,$acc),$s3);

2274                 &lea    ($acc,&DWP(16,$acc));   # advance out
2275                 &mov    ($_out,$acc);           # save out

2277                 &lea    ($acc,$ivec);
2278                 &mov    ($s0,&DWP(0,$acc));     # read temp
2279                 &mov    ($s1,&DWP(4,$acc));
2280                 &mov    ($s2,&DWP(8,$acc));
2281                 &mov    ($s3,&DWP(12,$acc));

2283                 &mov    (&DWP(0,$key),$s0);     # copy iv
2284                 &mov    (&DWP(4,$key),$s1);
2285                 &mov    (&DWP(8,$key),$s2);
2286                 &mov    (&DWP(12,$key),$s3);

2288                 &mov    ($acc,$_inp);           # load inp
2289                 &mov    ($s2,$_len);            # load len
2290                 &lea    ($acc,&DWP(16,$acc));   # advance inp
2291                 &mov    ($_inp,$acc);           # save inp
2292                 &sub    ($s2,16);               # decrease len
2293                 &mov    ($_len,$s2);            # save len
2294         &jnz    (&label("fast_dec_in_place_loop"));

2296     &set_label("fast_dec_out",4);
2297         &cmp    ($mark,0);              # was the key schedule copied?
2298         &mov    ("edi",$_key);
2299         &je     (&label("skip_dzero"));
2300         # zero copy of key schedule
2301         &mov    ("ecx",240/4);
2302         &xor    ("eax","eax");
2303         &align  (4);
2304         &data_word(0xABF3F689); # rep stosd
2305         &set_label("skip_dzero")
```



```
2306         &mov    ("esp",$_esp);
2307         &popf   ();
2308         &function_end_A();
2309         &pushf  ();                     # kludge, never executed

2311 #------------------------- SLOW ROUTINE -------------------------#
2312 &set_label("slow_way",16);

2314         &mov    ($s0,&DWP(0,$s0)) if (!$x86only);# load OPENSSL_ia32cap
2315         &mov    ($key,&wparam(3));      # load key

2317         # pre-allocate aligned stack frame...
2318         &lea    ($acc,&DWP(-80,"esp"));
2319         &and    ($acc,-64);

2321         # ... and make sure it doesn't alias with $key modulo 1024
2322         &lea    ($s1,&DWP(-80-63,$key));
2323         &sub    ($s1,$acc);
2324         &neg    ($s1);
2325         &and    ($s1,0x3C0);    # modulo 1024, but aligned to cache-line
2326         &sub    ($acc,$s1);

2328         # pick S-box copy which can't overlap with stack frame or $key
2329         &lea    ($s1,&DWP(768,$acc));
2330         &sub    ($s1,$tbl);
2331         &and    ($s1,0x300);
2332         &lea    ($tbl,&DWP(2048+128,$tbl,$s1));

2334         &lea    ($s3,&wparam(0));       # pointer to parameter block

2336         &exch   ("esp",$acc);
2337         &add    ("esp",4);              # reserve for return address!
2338         &mov    ($_tbl,$tbl);           # save %ebp
2339         &mov    ($_esp,$acc);           # save %esp
2340         &mov    ($_tmp,$s0);            # save OPENSSL_ia32cap

2342         &mov    ($s0,&DWP(0,$s3));      # load inp
2343         &mov    ($s1,&DWP(4,$s3));      # load out
2344         #&mov   ($s2,&DWP(8,$s3));      # load len
2345         #&mov   ($key,&DWP(12,$s3));    # load key
2346         &mov    ($acc,&DWP(16,$s3));    # load ivp
2347         &mov    ($s3,&DWP(20,$s3));     # load enc flag

2349         &mov    ($_inp,$s0);            # save copy of inp
2350         &mov    ($_out,$s1);            # save copy of out
2351         &mov    ($_len,$s2);            # save copy of len
2352         &mov    ($_key,$key);           # save copy of key
2353         &mov    ($_ivp,$acc);           # save copy of ivp

2355         &mov    ($key,$acc);
2356         &mov    ($acc,$s0);

2358         &cmp    ($s3,0);
2359         &je     (&label("slow_decrypt"));

2361 #------------------------- SLOW ENCRYPT -------------------------#
2362         &cmp    ($s2,16);
2363         &mov    ($s3,$s1);
2364         &jb     (&label("slow_enc_tail"));

2366                                         if (!$x86only) {
2367         &bt     ($_tmp,25);             # check for SSE bit
2368         &jnc    (&label("slow_enc_x86"));

2370         &movq   ("mm0",&QWP(0,$key));   # load iv
2371         &movq   ("mm4",&QWP(8,$key));
```

```
2373         &set_label("slow_enc_loop_sse",16);
2374                 &pxor   ("mm0",&QWP(0,$acc));   # xor input data
2375                 &pxor   ("mm4",&QWP(8,$acc));

2377                 &mov    ($key,$_key);
2378                 &call   ("_sse_AES_encrypt_compact");

2380                 &mov    ($acc,$_inp);           # load inp
2381                 &mov    ($key,$_out);           # load out
2382                 &mov    ($s2,$_len);            # load len

2384                 &movq   (&QWP(0,$key),"mm0");   # save output data
2385                 &movq   (&QWP(8,$key),"mm4");

2387                 &lea    ($acc,&DWP(16,$acc));   # advance inp
2388                 &mov    ($_inp,$acc);           # save inp
2389                 &lea    ($s3,&DWP(16,$key));    # advance out
2390                 &mov    ($_out,$s3);            # save out
2391                 &sub    ($s2,16);               # decrease len
2392                 &cmp    ($s2,16);
2393                 &mov    ($_len,$s2);            # save len
2394         &jae    (&label("slow_enc_loop_sse"));
2395         &test   ($s2,15);
2396         &jnz    (&label("slow_enc_tail"));
2397         &mov    ($acc,$_ivp);           # load ivp
2398         &movq   (&QWP(0,$acc),"mm0");   # save ivec
2399         &movq   (&QWP(8,$acc),"mm4");
2400         &emms   ();
2401         &mov    ("esp",$_esp);
2402         &popf   ();
2403         &function_end_A();
2404         &pushf  ();                     # kludge, never executed
2405                                         }
2406     &set_label("slow_enc_x86",16);
2407         &mov    ($s0,&DWP(0,$key));     # load iv
2408         &mov    ($s1,&DWP(4,$key));

2410         &set_label("slow_enc_loop_x86",4);
2411                 &mov    ($s2,&DWP(8,$key));
2412                 &mov    ($s3,&DWP(12,$key));

2414                 &xor    ($s0,&DWP(0,$acc));     # xor input data
2415                 &xor    ($s1,&DWP(4,$acc));
2416                 &xor    ($s2,&DWP(8,$acc));
2417                 &xor    ($s3,&DWP(12,$acc));

2419                 &mov    ($key,$_key);           # load key
2420                 &call   ("_x86_AES_encrypt_compact");

2422                 &mov    ($acc,$_inp);           # load inp
2423                 &mov    ($key,$_out);           # load out

2425                 &mov    (&DWP(0,$key),$s0);     # save output data
2426                 &mov    (&DWP(4,$key),$s1);
2427                 &mov    (&DWP(8,$key),$s2);
2428                 &mov    (&DWP(12,$key),$s3);

2430                 &mov    ($s2,$_len);            # load len
2431                 &lea    ($acc,&DWP(16,$acc));   # advance inp
2432                 &mov    ($_inp,$acc);           # save inp
2433                 &lea    ($s3,&DWP(16,$key));    # advance out
2434                 &mov    ($_out,$s3);            # save out
2435                 &sub    ($s2,16);               # decrease len
2436                 &cmp    ($s2,16);
2437                 &mov    ($_len,$s2);            # save len
```



```
2438         &jae    (&label("slow_enc_loop_x86"));
2439         &test   ($s2,15);
2440         &jnz    (&label("slow_enc_tail"));
2441         &mov    ($acc,$_ivp);           # load ivp
2442         &mov    ($s2,&DWP(8,$key));     # restore last dwords
2443         &mov    ($s3,&DWP(12,$key));
2444         &mov    (&DWP(0,$acc),$s0);     # save ivec
2445         &mov    (&DWP(4,$acc),$s1);
2446         &mov    (&DWP(8,$acc),$s2);
2447         &mov    (&DWP(12,$acc),$s3);

2449         &mov    ("esp",$_esp);
2450         &popf   ();
2451         &function_end_A();
2452         &pushf  ();                     # kludge, never executed

2454     &set_label("slow_enc_tail",16);
2455         &emms   ()      if (!$x86only);
2456         &mov    ($key eq "edi"? $key:"",$s3);   # load out to edi
2457         &mov    ($s1,16);
2458         &sub    ($s1,$s2);
2459         &cmp    ($key,$acc eq "esi"? $acc:"");  # compare with inp
2460         &je     (&label("enc_in_place"));
2461         &align  (4);
2462         &data_word(0xA4F3F689); # rep movsb     # copy input
2463         &jmp    (&label("enc_skip_in_place"));
2464     &set_label("enc_in_place");
2465         &lea    ($key,&DWP(0,$key,$s2));
2466     &set_label("enc_skip_in_place");
2467         &mov    ($s2,$s1);
2468         &xor    ($s0,$s0);
2469         &align  (4);
2470         &data_word(0xAAF3F689); # rep stosb     # zero tail

2472         &mov    ($key,$_ivp);                   # restore ivp
2473         &mov    ($acc,$s3);                     # output as input
2474         &mov    ($s0,&DWP(0,$key));
2475         &mov    ($s1,&DWP(4,$key));
2476         &mov    ($_len,16);                     # len=16
2477         &jmp    (&label("slow_enc_loop_x86"));  # one more spin...

2479 #------------------------- SLOW DECRYPT -------------------------#
2480 &set_label("slow_decrypt",16);
2481                                         if (!$x86only) {
2482         &bt     ($_tmp,25);             # check for SSE bit
2483         &jnc    (&label("slow_dec_loop_x86"));

2485         &set_label("slow_dec_loop_sse",4);
2486                 &movq   ("mm0",&QWP(0,$acc));   # read input
2487                 &movq   ("mm4",&QWP(8,$acc));

2489                 &mov    ($key,$_key);
2490                 &call   ("_sse_AES_decrypt_compact");

2492                 &mov    ($acc,$_inp);           # load inp
2493                 &lea    ($s0,$ivec);
2494                 &mov    ($s1,$_out);            # load out
2495                 &mov    ($s2,$_len);            # load len
2496                 &mov    ($key,$_ivp);           # load ivp

2498                 &movq   ("mm1",&QWP(0,$acc));   # re-read input
2499                 &movq   ("mm5",&QWP(8,$acc));

2501                 &pxor   ("mm0",&QWP(0,$key));   # xor iv
2502                 &pxor   ("mm4",&QWP(8,$key));
```

```
2504                &movq   (&QWP(0,$key),"mm1");   # copy input to iv
2505                &movq   (&QWP(8,$key),"mm5");

2507                &sub    ($s2,16);               # decrease len
2508                &jc     (&label("slow_dec_partial_sse"));

2510                &movq   (&QWP(0,$s1),"mm0");    # write output
2511                &movq   (&QWP(8,$s1),"mm4");

2513                &lea    ($s1,&DWP(16,$s1));     # advance out
2514                &mov    ($_out,$s1);            # save out
2515                &lea    ($acc,&DWP(16,$acc));   # advance inp
2516                &mov    ($_inp,$acc);           # save inp
2517                &mov    ($_len,$s2);            # save len
2518        &jnz    (&label("slow_dec_loop_sse"));
2519        &emms   ();
2520        &mov    ("esp",$_esp);
2521        &popf   ();
2522        &function_end_A();
2523        &pushf  ();                     # kludge, never executed

2525    &set_label("slow_dec_partial_sse",16);
2526        &movq   (&QWP(0,$s0),"mm0");    # save output to temp
2527        &movq   (&QWP(8,$s0),"mm4");
2528        &emms   ();

2530        &add    ($s2 eq "ecx" ? "ecx":"",16);
2531        &mov    ("edi",$s1);            # out
2532        &mov    ("esi",$s0);            # temp
2533        &align  (4);
2534        &data_word(0xA4F3F689);         # rep movsb # copy partial output

2536        &mov    ("esp",$_esp);
2537        &popf   ();
2538        &function_end_A();
2539        &pushf  ();                     # kludge, never executed
2540                                        }
2541        &set_label("slow_dec_loop_x86",16);
2542                &mov    ($s0,&DWP(0,$acc));     # read input
2543                &mov    ($s1,&DWP(4,$acc));
2544                &mov    ($s2,&DWP(8,$acc));
2545                &mov    ($s3,&DWP(12,$acc));

2547                &lea    ($key,$ivec);
2548                &mov    (&DWP(0,$key),$s0);     # copy to temp
2549                &mov    (&DWP(4,$key),$s1);
2550                &mov    (&DWP(8,$key),$s2);
2551                &mov    (&DWP(12,$key),$s3);

2553                &mov    ($key,$_key);           # load key
2554                &call   ("_x86_AES_decrypt_compact");

2556                &mov    ($key,$_ivp);           # load ivp
2557                &mov    ($acc,$_len);           # load len
2558                &xor    ($s0,&DWP(0,$key));     # xor iv
2559                &xor    ($s1,&DWP(4,$key));
2560                &xor    ($s2,&DWP(8,$key));
2561                &xor    ($s3,&DWP(12,$key));

2563                &sub    ($acc,16);
2564                &jc     (&label("slow_dec_partial_x86"));

2566                &mov    ($_len,$acc);           # save len
2567                &mov    ($acc,$_out);           # load out

2569                &mov    (&DWP(0,$acc),$s0);     # write output
```



```
2570                &mov    (&DWP(4,$acc),$s1);
2571                &mov    (&DWP(8,$acc),$s2);
2572                &mov    (&DWP(12,$acc),$s3);

2574                &lea    ($acc,&DWP(16,$acc));   # advance out
2575                &mov    ($_out,$acc);           # save out

2577                &lea    ($acc,$ivec);
2578                &mov    ($s0,&DWP(0,$acc));     # read temp
2579                &mov    ($s1,&DWP(4,$acc));
2580                &mov    ($s2,&DWP(8,$acc));
2581                &mov    ($s3,&DWP(12,$acc));

2583                &mov    (&DWP(0,$key),$s0);     # copy it to iv
2584                &mov    (&DWP(4,$key),$s1);
2585                &mov    (&DWP(8,$key),$s2);
2586                &mov    (&DWP(12,$key),$s3);

2588                &mov    ($acc,$_inp);           # load inp
2589                &lea    ($acc,&DWP(16,$acc));   # advance inp
2590                &mov    ($_inp,$acc);           # save inp
2591        &jnz    (&label("slow_dec_loop_x86"));
2592        &mov    ("esp",$_esp);
2593        &popf   ();
2594        &function_end_A();
2595        &pushf  ();                     # kludge, never executed

2597    &set_label("slow_dec_partial_x86",16);
2598        &lea    ($acc,$ivec);
2599        &mov    (&DWP(0,$acc),$s0);     # save output to temp
2600        &mov    (&DWP(4,$acc),$s1);
2601        &mov    (&DWP(8,$acc),$s2);
2602        &mov    (&DWP(12,$acc),$s3);

2604        &mov    ($acc,$_inp);
2605        &mov    ($s0,&DWP(0,$acc));     # re-read input
2606        &mov    ($s1,&DWP(4,$acc));
2607        &mov    ($s2,&DWP(8,$acc));
2608        &mov    ($s3,&DWP(12,$acc));

2610        &mov    (&DWP(0,$key),$s0);     # copy it to iv
2611        &mov    (&DWP(4,$key),$s1);
2612        &mov    (&DWP(8,$key),$s2);
2613        &mov    (&DWP(12,$key),$s3);

2615        &mov    ("ecx",$_len);
2616        &mov    ("edi",$_out);
2617        &lea    ("esi",$ivec);
2618        &align  (4);
2619        &data_word(0xA4F3F689);         # rep movsb # copy partial output

2621        &mov    ("esp",$_esp);
2622        &popf   ();
2623 &function_end("AES_cbc_encrypt");
2624 }

2626 #------------------------------------------------------------------#

2628 sub enckey()
2629 {
2630        &movz   ("esi",&LB("edx"));             # rk[i]>>0
2631        &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2632        &movz   ("esi",&HB("edx"));             # rk[i]>>8
2633        &shl    ("ebx",24);
2634        &xor    ("eax","ebx");
```

```
2636         &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2637         &shr    ("edx",16);
2638         &movz   ("esi",&LB("edx"));             # rk[i]>>16
2639         &xor    ("eax","ebx");

2641         &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2642         &movz   ("esi",&HB("edx"));             # rk[i]>>24
2643         &shl    ("ebx",8);
2644         &xor    ("eax","ebx");

2646         &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2647         &shl    ("ebx",16);
2648         &xor    ("eax","ebx");

2650         &xor    ("eax",&DWP(1024-128,$tbl,"ecx",4));    # rcon
2651 }

2653 &function_begin("_x86_AES_set_encrypt_key");
2654         &mov    ("esi",&wparam(1));             # user supplied key
2655         &mov    ("edi",&wparam(3));             # private key schedule

2657         &test   ("esi",-1);
2658         &jz     (&label("badpointer"));
2659         &test   ("edi",-1);
2660         &jz     (&label("badpointer"));

2662         &call   (&label("pic_point"));
2663         &set_label("pic_point");
2664         &blindpop($tbl);
2665         &lea    ($tbl,&DWP(&label("AES_Te")."-".&label("pic_point"),$tbl));
2666         &lea    ($tbl,&DWP(2048+128,$tbl));

2668         # prefetch Te4
2669         &mov    ("eax",&DWP(0-128,$tbl));
2670         &mov    ("ebx",&DWP(32-128,$tbl));
2671         &mov    ("ecx",&DWP(64-128,$tbl));
2672         &mov    ("edx",&DWP(96-128,$tbl));
2673         &mov    ("eax",&DWP(128-128,$tbl));
2674         &mov    ("ebx",&DWP(160-128,$tbl));
2675         &mov    ("ecx",&DWP(192-128,$tbl));
2676         &mov    ("edx",&DWP(224-128,$tbl));

2678         &mov    ("ecx",&wparam(2));             # number of bits in key
2679         &cmp    ("ecx",128);
2680         &je     (&label("10rounds"));
2681         &cmp    ("ecx",192);
2682         &je     (&label("12rounds"));
2683         &cmp    ("ecx",256);
2684         &je     (&label("14rounds"));
2685         &mov    ("eax",-2);                     # invalid number of bits
2686         &jmp    (&label("exit"));

2688     &set_label("10rounds");
2689         &mov    ("eax",&DWP(0,"esi"));          # copy first 4 dwords
2690         &mov    ("ebx",&DWP(4,"esi"));
2691         &mov    ("ecx",&DWP(8,"esi"));
2692         &mov    ("edx",&DWP(12,"esi"));
2693         &mov    (&DWP(0,"edi"),"eax");
2694         &mov    (&DWP(4,"edi"),"ebx");
2695         &mov    (&DWP(8,"edi"),"ecx");
2696         &mov    (&DWP(12,"edi"),"edx");

2698         &xor    ("ecx","ecx");
2699         &jmp    (&label("10shortcut"));

2701         &align  (4);
```



```
2702         &set_label("10loop");
2703                 &mov    ("eax",&DWP(0,"edi"));          # rk[0]
2704                 &mov    ("edx",&DWP(12,"edi"));         # rk[3]
2705         &set_label("10shortcut");
2706                 &enckey ();

2708                 &mov    (&DWP(16,"edi"),"eax");         # rk[4]
2709                 &xor    ("eax",&DWP(4,"edi"));
2710                 &mov    (&DWP(20,"edi"),"eax");         # rk[5]
2711                 &xor    ("eax",&DWP(8,"edi"));
2712                 &mov    (&DWP(24,"edi"),"eax");         # rk[6]
2713                 &xor    ("eax",&DWP(12,"edi"));
2714                 &mov    (&DWP(28,"edi"),"eax");         # rk[7]
2715                 &inc    ("ecx");
2716                 &add    ("edi",16);
2717                 &cmp    ("ecx",10);
2718         &jl     (&label("10loop"));

2720         &mov    (&DWP(80,"edi"),10);            # setup number of rounds
2721         &xor    ("eax","eax");
2722         &jmp    (&label("exit"));

2724     &set_label("12rounds");
2725         &mov    ("eax",&DWP(0,"esi"));          # copy first 6 dwords
2726         &mov    ("ebx",&DWP(4,"esi"));
2727         &mov    ("ecx",&DWP(8,"esi"));
2728         &mov    ("edx",&DWP(12,"esi"));
2729         &mov    (&DWP(0,"edi"),"eax");
2730         &mov    (&DWP(4,"edi"),"ebx");
2731         &mov    (&DWP(8,"edi"),"ecx");
2732         &mov    (&DWP(12,"edi"),"edx");
2733         &mov    ("ecx",&DWP(16,"esi"));
2734         &mov    ("edx",&DWP(20,"esi"));
2735         &mov    (&DWP(16,"edi"),"ecx");
2736         &mov    (&DWP(20,"edi"),"edx");

2738         &xor    ("ecx","ecx");
2739         &jmp    (&label("12shortcut"));

2741         &align  (4);
2742         &set_label("12loop");
2743                 &mov    ("eax",&DWP(0,"edi"));          # rk[0]
2744                 &mov    ("edx",&DWP(20,"edi"));         # rk[5]
2745         &set_label("12shortcut");
2746                 &enckey ();

2748                 &mov    (&DWP(24,"edi"),"eax");         # rk[6]
2749                 &xor    ("eax",&DWP(4,"edi"));
2750                 &mov    (&DWP(28,"edi"),"eax");         # rk[7]
2751                 &xor    ("eax",&DWP(8,"edi"));
2752                 &mov    (&DWP(32,"edi"),"eax");         # rk[8]
2753                 &xor    ("eax",&DWP(12,"edi"));
2754                 &mov    (&DWP(36,"edi"),"eax");         # rk[9]

2756                 &cmp    ("ecx",7);
2757                 &je     (&label("12break"));
2758                 &inc    ("ecx");

2760                 &xor    ("eax",&DWP(16,"edi"));
2761                 &mov    (&DWP(40,"edi"),"eax");         # rk[10]
2762                 &xor    ("eax",&DWP(20,"edi"));
2763                 &mov    (&DWP(44,"edi"),"eax");         # rk[11]

2765                 &add    ("edi",24);
2766         &jmp    (&label("12loop"));
```

```
2768         &set_label("12break");
2769         &mov    (&DWP(72,"edi"),12);            # setup number of rounds
2770         &xor    ("eax","eax");
2771         &jmp    (&label("exit"));

2773     &set_label("14rounds");
2774         &mov    ("eax",&DWP(0,"esi"));          # copy first 8 dwords
2775         &mov    ("ebx",&DWP(4,"esi"));
2776         &mov    ("ecx",&DWP(8,"esi"));
2777         &mov    ("edx",&DWP(12,"esi"));
2778         &mov    (&DWP(0,"edi"),"eax");
2779         &mov    (&DWP(4,"edi"),"ebx");
2780         &mov    (&DWP(8,"edi"),"ecx");
2781         &mov    (&DWP(12,"edi"),"edx");
2782         &mov    ("eax",&DWP(16,"esi"));
2783         &mov    ("ebx",&DWP(20,"esi"));
2784         &mov    ("ecx",&DWP(24,"esi"));
2785         &mov    ("edx",&DWP(28,"esi"));
2786         &mov    (&DWP(16,"edi"),"eax");
2787         &mov    (&DWP(20,"edi"),"ebx");
2788         &mov    (&DWP(24,"edi"),"ecx");
2789         &mov    (&DWP(28,"edi"),"edx");

2791         &xor    ("ecx","ecx");
2792         &jmp    (&label("14shortcut"));

2794         &align  (4);
2795         &set_label("14loop");
2796                 &mov    ("edx",&DWP(28,"edi"));         # rk[7]
2797         &set_label("14shortcut");
2798                 &mov    ("eax",&DWP(0,"edi"));          # rk[0]

2800                 &enckey ();

2802                 &mov    (&DWP(32,"edi"),"eax");         # rk[8]
2803                 &xor    ("eax",&DWP(4,"edi"));
2804                 &mov    (&DWP(36,"edi"),"eax");         # rk[9]
2805                 &xor    ("eax",&DWP(8,"edi"));
2806                 &mov    (&DWP(40,"edi"),"eax");         # rk[10]
2807                 &xor    ("eax",&DWP(12,"edi"));
2808                 &mov    (&DWP(44,"edi"),"eax");         # rk[11]

2810                 &cmp    ("ecx",6);
2811                 &je     (&label("14break"));
2812                 &inc    ("ecx");

2814                 &mov    ("edx","eax");
2815                 &mov    ("eax",&DWP(16,"edi"));         # rk[4]
2816                 &movz   ("esi",&LB("edx"));             # rk[11]>>0
2817                 &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2818                 &movz   ("esi",&HB("edx"));             # rk[11]>>8
2819                 &xor    ("eax","ebx");

2821                 &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2822                 &shr    ("edx",16);
2823                 &shl    ("ebx",8);
2824                 &movz   ("esi",&LB("edx"));             # rk[11]>>16
2825                 &xor    ("eax","ebx");

2827                 &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2828                 &movz   ("esi",&HB("edx"));             # rk[11]>>24
2829                 &shl    ("ebx",16);
2830                 &xor    ("eax","ebx");

2832                 &movz   ("ebx",&BP(-128,$tbl,"esi",1));
2833                 &shl    ("ebx",24);
```



```
2834                 &xor    ("eax","ebx");

2836                 &mov    (&DWP(48,"edi"),"eax");         # rk[12]
2837                 &xor    ("eax",&DWP(20,"edi"));
2838                 &mov    (&DWP(52,"edi"),"eax");         # rk[13]
2839                 &xor    ("eax",&DWP(24,"edi"));
2840                 &mov    (&DWP(56,"edi"),"eax");         # rk[14]
2841                 &xor    ("eax",&DWP(28,"edi"));
2842                 &mov    (&DWP(60,"edi"),"eax");         # rk[15]

2844                 &add    ("edi",32);
2845         &jmp    (&label("14loop"));

2847         &set_label("14break");
2848         &mov    (&DWP(48,"edi"),14);            # setup number of rounds
2849         &xor    ("eax","eax");
2850         &jmp    (&label("exit"));

2852     &set_label("badpointer");
2853         &mov    ("eax",-1);
2854     &set_label("exit");
2855 &function_end("_x86_AES_set_encrypt_key");

2857 # int private_AES_set_encrypt_key(const unsigned char *userKey, const int bits,
2858 #                                 AES_KEY *key)
2859 &function_begin_B("private_AES_set_encrypt_key");
2860         &call   ("_x86_AES_set_encrypt_key");
2861         &ret    ();
2862 &function_end_B("private_AES_set_encrypt_key");

2864 sub deckey()
2865 { my ($i,$key,$tp1,$tp2,$tp4,$tp8) = @_;
2866   my $tmp = $tbl;

2868         &mov    ($acc,$tp1);
2869         &and    ($acc,0x80808080);
2870         &mov    ($tmp,$acc);
2871         &shr    ($tmp,7);
2872         &lea    ($tp2,&DWP(0,$tp1,$tp1));
2873         &sub    ($acc,$tmp);
2874         &and    ($tp2,0xfefefefe);
2875         &and    ($acc,0x1b1b1b1b);
2876         &xor    ($acc,$tp2);
2877         &mov    ($tp2,$acc);

2879         &and    ($acc,0x80808080);
2880         &mov    ($tmp,$acc);
2881         &shr    ($tmp,7);
2882         &lea    ($tp4,&DWP(0,$tp2,$tp2));
2883         &sub    ($acc,$tmp);
2884         &and    ($tp4,0xfefefefe);
2885         &and    ($acc,0x1b1b1b1b);
2886          &xor   ($tp2,$tp1);    # tp2^tp1
2887         &xor    ($acc,$tp4);
2888         &mov    ($tp4,$acc);

2890         &and    ($acc,0x80808080);
2891         &mov    ($tmp,$acc);
2892         &shr    ($tmp,7);
2893         &lea    ($tp8,&DWP(0,$tp4,$tp4));
2894          &xor   ($tp4,$tp1);    # tp4^tp1
2895         &sub    ($acc,$tmp);
2896         &and    ($tp8,0xfefefefe);
2897         &and    ($acc,0x1b1b1b1b);
2898          &rotl  ($tp1,8);       # = ROTATE(tp1,8)
2899         &xor    ($tp8,$acc);
```

```
2901         &mov    ($tmp,&DWP(4*($i+1),$key));     # modulo-scheduled load

2903         &xor    ($tp1,$tp2);
2904         &xor    ($tp2,$tp8);
2905         &xor    ($tp1,$tp4);
2906         &rotl   ($tp2,24);
2907         &xor    ($tp4,$tp8);
2908         &xor    ($tp1,$tp8);    # ^= tp8^(tp4^tp1)^(tp2^tp1)
2909         &rotl   ($tp4,16);
2910         &xor    ($tp1,$tp2);    # ^= ROTATE(tp8^tp2^tp1,24)
2911         &rotl   ($tp8,8);
2912         &xor    ($tp1,$tp4);    # ^= ROTATE(tp8^tp4^tp1,16)
2913         &mov    ($tp2,$tmp);
2914         &xor    ($tp1,$tp8);    # ^= ROTATE(tp8,8)

2916         &mov    (&DWP(4*$i,$key),$tp1);
2917 }

2919 # int private_AES_set_decrypt_key(const unsigned char *userKey, const int bits,
2920 #                       AES_KEY *key)
2921 &function_begin_B("private_AES_set_decrypt_key");
2922         &call   ("_x86_AES_set_encrypt_key");
2923         &cmp    ("eax",0);
2924         &je     (&label("proceed"));
2925         &ret    ();

2927     &set_label("proceed");
2928         &push   ("ebp");
2929         &push   ("ebx");
2930         &push   ("esi");
2931         &push   ("edi");

2933         &mov    ("esi",&wparam(2));
2934         &mov    ("ecx",&DWP(240,"esi"));        # pull number of rounds
2935         &lea    ("ecx",&DWP(0,"","ecx",4));
2936         &lea    ("edi",&DWP(0,"esi","ecx",4));  # pointer to last chunk

2938         &set_label("invert",4);                 # invert order of chunks
2939                 &mov    ("eax",&DWP(0,"esi"));
2940                 &mov    ("ebx",&DWP(4,"esi"));
2941                 &mov    ("ecx",&DWP(0,"edi"));
2942                 &mov    ("edx",&DWP(4,"edi"));
2943                 &mov    (&DWP(0,"edi"),"eax");
2944                 &mov    (&DWP(4,"edi"),"ebx");
2945                 &mov    (&DWP(0,"esi"),"ecx");
2946                 &mov    (&DWP(4,"esi"),"edx");
2947                 &mov    ("eax",&DWP(8,"esi"));
2948                 &mov    ("ebx",&DWP(12,"esi"));
2949                 &mov    ("ecx",&DWP(8,"edi"));
2950                 &mov    ("edx",&DWP(12,"edi"));
2951                 &mov    (&DWP(8,"edi"),"eax");
2952                 &mov    (&DWP(12,"edi"),"ebx");
2953                 &mov    (&DWP(8,"esi"),"ecx");
2954                 &mov    (&DWP(12,"esi"),"edx");
2955                 &add    ("esi",16);
2956                 &sub    ("edi",16);
2957                 &cmp    ("esi","edi");
2958         &jne    (&label("invert"));

2960         &mov    ($key,&wparam(2));
2961         &mov    ($acc,&DWP(240,$key));          # pull number of rounds
2962         &lea    ($acc,&DWP(-2,$acc,$acc));
2963         &lea    ($acc,&DWP(0,$key,$acc,8));
2964         &mov    (&wparam(2),$acc);
```



```
2966         &mov    ($s0,&DWP(16,$key));            # modulo-scheduled load
2967         &set_label("permute",4);                # permute the key schedule
2968                 &add    ($key,16);
2969                 &deckey (0,$key,$s0,$s1,$s2,$s3);
2970                 &deckey (1,$key,$s1,$s2,$s3,$s0);
2971                 &deckey (2,$key,$s2,$s3,$s0,$s1);
2972                 &deckey (3,$key,$s3,$s0,$s1,$s2);
2973                 &cmp    ($key,&wparam(2));
2974         &jb     (&label("permute"));

2976         &xor    ("eax","eax");                  # return success
2977 &function_end("private_AES_set_decrypt_key");
2978 &asciz("AES for x86, CRYPTOGAMS by <appro\@openssl.org>");

2980 &asm_finish();
2981 #endif /* ! codereview */
```

```
**********************************************************
   75019 Wed Aug 13 19:53:05 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/aes-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# Version 2.1.
#
# aes-*-cbc benchmarks are improved by >70% [compared to gcc 3.3.2 on
# Opteron 240 CPU] plus all the bells-n-whistles from 32-bit version
# [you'll notice a lot of resemblance], such as compressed S-boxes
# in little-endian byte order, prefetch of these tables in CBC mode,
# as well as avoiding L1 cache aliasing between stack frame and key
# schedule and already mentioned tables, compressed Td4...
#
# Performance in number of cycles per processed byte for 128-bit key:
#
#               ECB encrypt     ECB decrypt     CBC large chunk
# AMD64         33              41              13.0
# EM64T         38              59              18.6(*)
# Core 2        30              43              14.5(*)
#
# (*) with hyper-threading off

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

$verticalspin=1;        # unlike 32-bit version $verticalspin performs
                        # ~15% better on both AMD and Intel cores
$speed_limit=512;       # see aes-586.pl for details

$code=".text\n";

$s0="%eax";
$s1="%ebx";
$s2="%ecx";
$s3="%edx";
$acc0="%esi";   $mask80="%rsi";
$acc1="%edi";   $maskfe="%rdi";
$acc2="%ebp";   $mask1b="%rbp";
$inp="%r8";
$out="%r9";
$t0="%r10d";
$t1="%r11d";
$t2="%r12d";
$rnds="%r13d";
$sbox="%r14";
```



```
$key="%r15";

sub hi() { my $r=shift; $r =~ s/%[er]([a-d])x/%\1h/;    $r; }
sub lo() { my $r=shift; $r =~ s/%[er]([a-d])x/%\1l/;
                        $r =~ s/%[er]([sd]i)/%\1l/;
                        $r =~ s/%(r[0-9]+)[d]?/%\1b/;   $r; }
sub LO() { my $r=shift; $r =~ s/%r([a-z]+)/%e\1/;
                        $r =~ s/%r([0-9]+)/%r\1d/;      $r; }
sub _data_word()
{ my $i;
    while(defined($i=shift)) { $code.=sprintf".long\t0x%08x,0x%08x\n",$i,$i; }
}
sub data_word()
{ my $i;
  my $last=pop(@_);
    $code.=".long\t";
    while(defined($i=shift)) { $code.=sprintf"0x%08x,",$i; }
    $code.=sprintf"0x%08x\n",$last;
}

sub data_byte()
{ my $i;
  my $last=pop(@_);
    $code.=".byte\t";
    while(defined($i=shift)) { $code.=sprintf"0x%02x,",$i&0xff; }
    $code.=sprintf"0x%02x\n",$last&0xff;
}

sub encvert()
{ my $t3="%r8d";        # zaps $inp!

$code.=<<___;
        # favor 3-way issue Opteron pipeline...
        movzb   `&lo("$s0")`,$acc0
        movzb   `&lo("$s1")`,$acc1
        movzb   `&lo("$s2")`,$acc2
        mov     0($sbox,$acc0,8),$t0
        mov     0($sbox,$acc1,8),$t1
        mov     0($sbox,$acc2,8),$t2

        movzb   `&hi("$s1")`,$acc0
        movzb   `&hi("$s2")`,$acc1
        movzb   `&lo("$s3")`,$acc2
        xor     3($sbox,$acc0,8),$t0
        xor     3($sbox,$acc1,8),$t1
        mov     0($sbox,$acc2,8),$t3

        movzb   `&hi("$s3")`,$acc0
        shr     \$16,$s2
        movzb   `&hi("$s0")`,$acc2
        xor     3($sbox,$acc0,8),$t2
        shr     \$16,$s3
        xor     3($sbox,$acc2,8),$t3

        shr     \$16,$s1
        lea     16($key),$key
        shr     \$16,$s0

        movzb   `&lo("$s2")`,$acc0
        movzb   `&lo("$s3")`,$acc1
        movzb   `&lo("$s0")`,$acc2
        xor     2($sbox,$acc0,8),$t0
        xor     2($sbox,$acc1,8),$t1
        xor     2($sbox,$acc2,8),$t2

        movzb   `&hi("$s3")`,$acc0
```

```
128         movzb   `&hi("$s0")`,$acc1
129         movzb   `&lo("$s1")`,$acc2
130         xor     1($sbox,$acc0,8),$t0
131         xor     1($sbox,$acc1,8),$t1
132         xor     2($sbox,$acc2,8),$t3

134         mov     12($key),$s3
135         movzb   `&hi("$s1")`,$acc1
136         movzb   `&hi("$s2")`,$acc2
137         mov     0($key),$s0
138         xor     1($sbox,$acc1,8),$t2
139         xor     1($sbox,$acc2,8),$t3

141         mov     4($key),$s1
142         mov     8($key),$s2
143         xor     $t0,$s0
144         xor     $t1,$s1
145         xor     $t2,$s2
146         xor     $t3,$s3
147 ___
148 }

150 sub enclastvert()
151 { my $t3="%r8d";        # zaps $inp!

153 $code.=<<___;
154         movzb   `&lo("$s0")`,$acc0
155         movzb   `&lo("$s1")`,$acc1
156         movzb   `&lo("$s2")`,$acc2
157         movzb   2($sbox,$acc0,8),$t0
158         movzb   2($sbox,$acc1,8),$t1
159         movzb   2($sbox,$acc2,8),$t2

161         movzb   `&lo("$s3")`,$acc0
162         movzb   `&hi("$s1")`,$acc1
163         movzb   `&hi("$s2")`,$acc2
164         movzb   2($sbox,$acc0,8),$t3
165         mov     0($sbox,$acc1,8),$acc1  #$t0
166         mov     0($sbox,$acc2,8),$acc2  #$t1

168         and     \$0x0000ff00,$acc1
169         and     \$0x0000ff00,$acc2

171         xor     $acc1,$t0
172         xor     $acc2,$t1
173         shr     \$16,$s2

175         movzb   `&hi("$s3")`,$acc0
176         movzb   `&hi("$s0")`,$acc1
177         shr     \$16,$s3
178         mov     0($sbox,$acc0,8),$acc0  #$t2
179         mov     0($sbox,$acc1,8),$acc1  #$t3

181         and     \$0x0000ff00,$acc0
182         and     \$0x0000ff00,$acc1
183         shr     \$16,$s1
184         xor     $acc0,$t2
185         xor     $acc1,$t3
186         shr     \$16,$s0

188         movzb   `&lo("$s2")`,$acc0
189         movzb   `&lo("$s3")`,$acc1
190         movzb   `&lo("$s0")`,$acc2
191         mov     0($sbox,$acc0,8),$acc0  #$t0
192         mov     0($sbox,$acc1,8),$acc1  #$t1
193         mov     0($sbox,$acc2,8),$acc2  #$t2
```



```
195         and     \$0x00ff0000,$acc0
196         and     \$0x00ff0000,$acc1
197         and     \$0x00ff0000,$acc2

199         xor     $acc0,$t0
200         xor     $acc1,$t1
201         xor     $acc2,$t2

203         movzb   `&lo("$s1")`,$acc0
204         movzb   `&hi("$s3")`,$acc1
205         movzb   `&hi("$s0")`,$acc2
206         mov     0($sbox,$acc0,8),$acc0  #$t3
207         mov     2($sbox,$acc1,8),$acc1  #$t0
208         mov     2($sbox,$acc2,8),$acc2  #$t1

210         and     \$0x00ff0000,$acc0
211         and     \$0xff000000,$acc1
212         and     \$0xff000000,$acc2

214         xor     $acc0,$t3
215         xor     $acc1,$t0
216         xor     $acc2,$t1

218         movzb   `&hi("$s1")`,$acc0
219         movzb   `&hi("$s2")`,$acc1
220         mov     16+12($key),$s3
221         mov     2($sbox,$acc0,8),$acc0  #$t2
222         mov     2($sbox,$acc1,8),$acc1  #$t3
223         mov     16+0($key),$s0

225         and     \$0xff000000,$acc0
226         and     \$0xff000000,$acc1

228         xor     $acc0,$t2
229         xor     $acc1,$t3

231         mov     16+4($key),$s1
232         mov     16+8($key),$s2
233         xor     $t0,$s0
234         xor     $t1,$s1
235         xor     $t2,$s2
236         xor     $t3,$s3
237 ___
238 }

240 sub encstep()
241 { my ($i,@s) = @_;
242   my $tmp0=$acc0;
243   my $tmp1=$acc1;
244   my $tmp2=$acc2;
245   my $out=($t0,$t1,$t2,$s[0])[$i];

247         if ($i==3) {
248                 $tmp0=$s[1];
249                 $tmp1=$s[2];
250                 $tmp2=$s[3];
251         }
252         $code.="        movzb   ".&lo($s[0]).",$out\n";
253         $code.="        mov     $s[2],$tmp1\n"          if ($i!=3);
254         $code.="        lea     16($key),$key\n"        if ($i==0);

256         $code.="        movzb   ".&hi($s[1]).",$tmp0\n";
257         $code.="        mov     0($sbox,$out,8),$out\n";

259         $code.="        shr     \$16,$tmp1\n";
```

```
260         $code.="        mov     $s[3],$tmp2\n"          if ($i!=3);
261         $code.="        xor     3($sbox,$tmp0,8),$out\n";

263         $code.="        movzb   ".&lo($tmp1).",$tmp1\n";
264         $code.="        shr     \$24,$tmp2\n";
265         $code.="        xor     4*$i($key),$out\n";

267         $code.="        xor     2($sbox,$tmp1,8),$out\n";
268         $code.="        xor     1($sbox,$tmp2,8),$out\n";

270         $code.="        mov     $t0,$s[1]\n"            if ($i==3);
271         $code.="        mov     $t1,$s[2]\n"            if ($i==3);
272         $code.="        mov     $t2,$s[3]\n"            if ($i==3);
273         $code.="\n";
274 }

276 sub enclast()
277 { my ($i,@s)=@_;
278   my $tmp0=$acc0;
279   my $tmp1=$acc1;
280   my $tmp2=$acc2;
281   my $out=($t0,$t1,$t2,$s[0])[$i];

283         if ($i==3) {
284                 $tmp0=$s[1];
285                 $tmp1=$s[2];
286                 $tmp2=$s[3];
287         }
288         $code.="        movzb   ".&lo($s[0]).",$out\n";
289         $code.="        mov     $s[2],$tmp1\n"          if ($i!=3);

291         $code.="        mov     2($sbox,$out,8),$out\n";
292         $code.="        shr     \$16,$tmp1\n";
293         $code.="        mov     $s[3],$tmp2\n"          if ($i!=3);

295         $code.="        and     \$0x000000ff,$out\n";
296         $code.="        movzb   ".&hi($s[1]).",$tmp0\n";
297         $code.="        movzb   ".&lo($tmp1).",$tmp1\n";
298         $code.="        shr     \$24,$tmp2\n";

300         $code.="        mov     0($sbox,$tmp0,8),$tmp0\n";
301         $code.="        mov     0($sbox,$tmp1,8),$tmp1\n";
302         $code.="        mov     2($sbox,$tmp2,8),$tmp2\n";

304         $code.="        and     \$0x0000ff00,$tmp0\n";
305         $code.="        and     \$0x00ff0000,$tmp1\n";
306         $code.="        and     \$0xff000000,$tmp2\n";

308         $code.="        xor     $tmp0,$out\n";
309         $code.="        mov     $t0,$s[1]\n"            if ($i==3);
310         $code.="        xor     $tmp1,$out\n";
311         $code.="        mov     $t1,$s[2]\n"            if ($i==3);
312         $code.="        xor     $tmp2,$out\n";
313         $code.="        mov     $t2,$s[3]\n"            if ($i==3);
314         $code.="\n";
315 }

317 $code.=<<___;
318 .type   _x86_64_AES_encrypt,\@abi-omnipotent
319 .align  16
320 _x86_64_AES_encrypt:
321         xor     0($key),$s0                     # xor with key
322         xor     4($key),$s1
323         xor     8($key),$s2
324         xor     12($key),$s3
```



```
326         mov     240($key),$rnds                 # load key->rounds
327         sub     \$1,$rnds
328         jmp     .Lenc_loop
329 .align  16
330 .Lenc_loop:
331 ___
332         if ($verticalspin) { &encvert(); }
333         else {  &encstep(0,$s0,$s1,$s2,$s3);
334                 &encstep(1,$s1,$s2,$s3,$s0);
335                 &encstep(2,$s2,$s3,$s0,$s1);
336                 &encstep(3,$s3,$s0,$s1,$s2);
337         }
338 $code.=<<___;
339         sub     \$1,$rnds
340         jnz     .Lenc_loop
341 ___
342         if ($verticalspin) { &enclastvert(); }
343         else {  &enclast(0,$s0,$s1,$s2,$s3);
344                 &enclast(1,$s1,$s2,$s3,$s0);
345                 &enclast(2,$s2,$s3,$s0,$s1);
346                 &enclast(3,$s3,$s0,$s1,$s2);
347                 $code.=<<___;
348                 xor     16+0($key),$s0          # xor with key
349                 xor     16+4($key),$s1
350                 xor     16+8($key),$s2
351                 xor     16+12($key),$s3
352 ___
353         }
354 $code.=<<___;
355         .byte   0xf3,0xc3                       # rep ret
356 .size   _x86_64_AES_encrypt,.-_x86_64_AES_encrypt
357 ___

359 # it's possible to implement this by shifting tN by 8, filling least
360 # significant byte with byte load and finally bswap-ing at the end,
361 # but such partial register load kills Core 2...
362 sub enccompactvert()
363 { my ($t3,$t4,$t5)=("%r8d","%r9d","%r13d");

365 $code.=<<___;
366         movzb   `&lo("$s0")`,$t0
367         movzb   `&lo("$s1")`,$t1
368         movzb   `&lo("$s2")`,$t2
369         movzb   ($sbox,$t0,1),$t0
370         movzb   ($sbox,$t1,1),$t1
371         movzb   ($sbox,$t2,1),$t2

373         movzb   `&lo("$s3")`,$t3
374         movzb   `&hi("$s1")`,$acc0
375         movzb   `&hi("$s2")`,$acc1
376         movzb   ($sbox,$t3,1),$t3
377         movzb   ($sbox,$acc0,1),$t4     #$t0
378         movzb   ($sbox,$acc1,1),$t5     #$t1

380         movzb   `&hi("$s3")`,$acc2
381         movzb   `&hi("$s0")`,$acc0
382         shr     \$16,$s2
383         movzb   ($sbox,$acc2,1),$acc2   #$t2
384         movzb   ($sbox,$acc0,1),$acc0   #$t3
385         shr     \$16,$s3

387         movzb   `&lo("$s2")`,$acc1
388         shl     \$8,$t4
389         shl     \$8,$t5
390         movzb   ($sbox,$acc1,1),$acc1   #$t0
391         xor     $t4,$t0
```

```
392         xor     $t5,$t1

394         movzb   ‘&lo("$s3")‘,$t4
395         shr     \$16,$s0
396         shr     \$16,$s1
397         movzb   ‘&lo("$s0")‘,$t5
398         shl     \$8,$acc2
399         shl     \$8,$acc0
400         movzb   ($sbox,$t4,1),$t4       #$t1
401         movzb   ($sbox,$t5,1),$t5       #$t2
402         xor     $acc2,$t2
403         xor     $acc0,$t3

405         movzb   ‘&lo("$s1")‘,$acc2
406         movzb   ‘&hi("$s3")‘,$acc0
407         shl     \$16,$acc1
408         movzb   ($sbox,$acc2,1),$acc2   #$t3
409         movzb   ($sbox,$acc0,1),$acc0   #$t0
410         xor     $acc1,$t0

412         movzb   ‘&hi("$s0")‘,$acc1
413         shr     \$8,$s2
414         shr     \$8,$s1
415         movzb   ($sbox,$acc1,1),$acc1   #$t1
416         movzb   ($sbox,$s2,1),$s3       #$t3
417         movzb   ($sbox,$s1,1),$s2       #$t2
418         shl     \$16,$t4
419         shl     \$16,$t5
420         shl     \$16,$acc2
421         xor     $t4,$t1
422         xor     $t5,$t2
423         xor     $acc2,$t3

425         shl     \$24,$acc0
426         shl     \$24,$acc1
427         shl     \$24,$s3
428         xor     $acc0,$t0
429         shl     \$24,$s2
430         xor     $acc1,$t1
431         mov     $t0,$s0
432         mov     $t1,$s1
433         xor     $t2,$s2
434         xor     $t3,$s3
435 ___
436 }

438 sub enctransform_ref()
439 { my $sn = shift;
440   my ($acc,$r2,$tmp)=("%r8d","%r9d","%r13d");

442 $code.=<<___;
443         mov     $sn,$acc
444         and     \$0x80808080,$acc
445         mov     $acc,$tmp
446         shr     \$7,$tmp
447         lea     ($sn,$sn),$r2
448         sub     $tmp,$acc
449         and     \$0xfefefefe,$r2
450         and     \$0x1b1b1b1b,$acc
451         mov     $sn,$tmp
452         xor     $acc,$r2

454         xor     $r2,$sn
455         rol     \$24,$sn
456         xor     $r2,$sn
457         ror     \$16,$tmp
```



```
458         xor     $tmp,$sn
459         ror     \$8,$tmp
460         xor     $tmp,$sn
461 ___
462 }

464 # unlike decrypt case it does not pay off to parallelize enctransform
465 sub enctransform()
466 { my ($t3,$r20,$r21)=($acc2,"%r8d","%r9d");

468 $code.=<<___;
469         mov     $s0,$acc0
470         mov     $s1,$acc1
471         and     \$0x80808080,$acc0
472         and     \$0x80808080,$acc1
473         mov     $acc0,$t0
474         mov     $acc1,$t1
475         shr     \$7,$t0
476         lea     ($s0,$s0),$r20
477         shr     \$7,$t1
478         lea     ($s1,$s1),$r21
479         sub     $t0,$acc0
480         sub     $t1,$acc1
481         and     \$0xfefefefe,$r20
482         and     \$0xfefefefe,$r21
483         and     \$0x1b1b1b1b,$acc0
484         and     \$0x1b1b1b1b,$acc1
485         mov     $s0,$t0
486         mov     $s1,$t1
487         xor     $acc0,$r20
488         xor     $acc1,$r21

490         xor     $r20,$s0
491         xor     $r21,$s1
492          mov    $s2,$acc0
493          mov    $s3,$acc1
494         rol     \$24,$s0
495         rol     \$24,$s1
496          and    \$0x80808080,$acc0
497          and    \$0x80808080,$acc1
498         xor     $r20,$s0
499         xor     $r21,$s1
500          mov    $acc0,$t2
501          mov    $acc1,$t3
502         ror     \$16,$t0
503         ror     \$16,$t1
504          shr    \$7,$t2
505          lea    ($s2,$s2),$r20
506         xor     $t0,$s0
507         xor     $t1,$s1
508          shr    \$7,$t3
509          lea    ($s3,$s3),$r21
510         ror     \$8,$t0
511         ror     \$8,$t1
512          sub    $t2,$acc0
513          sub    $t3,$acc1
514         xor     $t0,$s0
515         xor     $t1,$s1

517         and     \$0xfefefefe,$r20
518         and     \$0xfefefefe,$r21
519         and     \$0x1b1b1b1b,$acc0
520         and     \$0x1b1b1b1b,$acc1
521         mov     $s2,$t2
522         mov     $s3,$t3
523         xor     $acc0,$r20
```

```
524         xor     $acc1,$r21

526         xor     $r20,$s2
527         xor     $r21,$s3
528         rol     \$24,$s2
529         rol     \$24,$s3
530         xor     $r20,$s2
531         xor     $r21,$s3
532         mov     0($sbox),$acc0                  # prefetch Te4
533         ror     \$16,$t2
534         ror     \$16,$t3
535         mov     64($sbox),$acc1
536         xor     $t2,$s2
537         xor     $t3,$s3
538         mov     128($sbox),$r20
539         ror     \$8,$t2
540         ror     \$8,$t3
541         mov     192($sbox),$r21
542         xor     $t2,$s2
543         xor     $t3,$s3
544 ___
545 }

547 $code.=<<___;
548 .type   _x86_64_AES_encrypt_compact,\@abi-omnipotent
549 .align  16
550 _x86_64_AES_encrypt_compact:
551         lea     128($sbox),$inp                 # size optimization
552         mov     0-128($inp),$acc1               # prefetch Te4
553         mov     32-128($inp),$acc2
554         mov     64-128($inp),$t0
555         mov     96-128($inp),$t1
556         mov     128-128($inp),$acc1
557         mov     160-128($inp),$acc2
558         mov     192-128($inp),$t0
559         mov     224-128($inp),$t1
560         jmp     .Lenc_loop_compact
561 .align  16
562 .Lenc_loop_compact:
563                 xor     0($key),$s0             # xor with key
564                 xor     4($key),$s1
565                 xor     8($key),$s2
566                 xor     12($key),$s3
567                 lea     16($key),$key
568 ___
569                 &enccompactvert();
570 $code.=<<___;
571                 cmp     16(%rsp),$key
572                 je      .Lenc_compact_done
573 ___
574                 &enctransform();
575 $code.=<<___;
576         jmp     .Lenc_loop_compact
577 .align  16
578 .Lenc_compact_done:
579         xor     0($key),$s0
580         xor     4($key),$s1
581         xor     8($key),$s2
582         xor     12($key),$s3
583         .byte   0xf3,0xc3                       # rep ret
584 .size   _x86_64_AES_encrypt_compact,.-_x86_64_AES_encrypt_compact
585 ___

587 # void AES_encrypt (const void *inp,void *out,const AES_KEY *key);
588 $code.=<<___;
589 .globl  AES_encrypt
```



```
590 .type   AES_encrypt,\@function,3
591 .align  16
592 .globl  asm_AES_encrypt
593 .hidden asm_AES_encrypt
594 asm_AES_encrypt:
595 AES_encrypt:
596         push    %rbx
597         push    %rbp
598         push    %r12
599         push    %r13
600         push    %r14
601         push    %r15

603         # allocate frame "above" key schedule
604         mov     %rsp,%r10
605         lea     -63(%rdx),%rcx  # %rdx is key argument
606         and     \$-64,%rsp
607         sub     %rsp,%rcx
608         neg     %rcx
609         and     \$0x3c0,%rcx
610         sub     %rcx,%rsp
611         sub     \$32,%rsp

613         mov     %rsi,16(%rsp)   # save out
614         mov     %r10,24(%rsp)   # save real stack pointer
615 .Lenc_prologue:

617         mov     %rdx,$key
618         mov     240($key),$rnds # load rounds

620         mov     0(%rdi),$s0     # load input vector
621         mov     4(%rdi),$s1
622         mov     8(%rdi),$s2
623         mov     12(%rdi),$s3

625         shl     \$4,$rnds
626         lea     ($key,$rnds),%rbp
627         mov     $key,(%rsp)     # key schedule
628         mov     %rbp,8(%rsp)    # end of key schedule

630         # pick Te4 copy which can't "overlap" with stack frame or key schedule
631         lea     .LAES_Te+2048(%rip),$sbox
632         lea     768(%rsp),%rbp
633         sub     $sbox,%rbp
634         and     \$0x300,%rbp
635         lea     ($sbox,%rbp),$sbox

637         call    _x86_64_AES_encrypt_compact

639         mov     16(%rsp),$out   # restore out
640         mov     24(%rsp),%rsi   # restore saved stack pointer
641         mov     $s0,0($out)     # write output vector
642         mov     $s1,4($out)
643         mov     $s2,8($out)
644         mov     $s3,12($out)

646         mov     (%rsi),%r15
647         mov     8(%rsi),%r14
648         mov     16(%rsi),%r13
649         mov     24(%rsi),%r12
650         mov     32(%rsi),%rbp
651         mov     40(%rsi),%rbx
652         lea     48(%rsi),%rsp
653 .Lenc_epilogue:
654         ret
655 .size   AES_encrypt,.-AES_encrypt
```

```
___

#------------------------------------------------------------------#

sub decvert()
{ my $t3="%r8d";        # zaps $inp!

$code.=<<___;
        # favor 3-way issue Opteron pipeline...
        movzb   `&lo("$s0")`,$acc0
        movzb   `&lo("$s1")`,$acc1
        movzb   `&lo("$s2")`,$acc2
        mov     0($sbox,$acc0,8),$t0
        mov     0($sbox,$acc1,8),$t1
        mov     0($sbox,$acc2,8),$t2

        movzb   `&hi("$s3")`,$acc0
        movzb   `&hi("$s0")`,$acc1
        movzb   `&lo("$s3")`,$acc2
        xor     3($sbox,$acc0,8),$t0
        xor     3($sbox,$acc1,8),$t1
        mov     0($sbox,$acc2,8),$t3

        movzb   `&hi("$s1")`,$acc0
        shr     \$16,$s0
        movzb   `&hi("$s2")`,$acc2
        xor     3($sbox,$acc0,8),$t2
        shr     \$16,$s3
        xor     3($sbox,$acc2,8),$t3

        shr     \$16,$s1
        lea     16($key),$key
        shr     \$16,$s2

        movzb   `&lo("$s2")`,$acc0
        movzb   `&lo("$s3")`,$acc1
        movzb   `&lo("$s0")`,$acc2
        xor     2($sbox,$acc0,8),$t0
        xor     2($sbox,$acc1,8),$t1
        xor     2($sbox,$acc2,8),$t2

        movzb   `&hi("$s1")`,$acc0
        movzb   `&hi("$s2")`,$acc1
        movzb   `&lo("$s1")`,$acc2
        xor     1($sbox,$acc0,8),$t0
        xor     1($sbox,$acc1,8),$t1
        xor     2($sbox,$acc2,8),$t3

        movzb   `&hi("$s3")`,$acc0
        mov     12($key),$s3
        movzb   `&hi("$s0")`,$acc2
        xor     1($sbox,$acc0,8),$t2
        mov     0($key),$s0
        xor     1($sbox,$acc2,8),$t3

        xor     $t0,$s0
        mov     4($key),$s1
        mov     8($key),$s2
        xor     $t2,$s2
        xor     $t1,$s1
        xor     $t3,$s3
___
}

sub declastvert()
{ my $t3="%r8d";        # zaps $inp!
```



```
$code.=<<___;
        lea     2048($sbox),$sbox       # size optimization
        movzb   `&lo("$s0")`,$acc0
        movzb   `&lo("$s1")`,$acc1
        movzb   `&lo("$s2")`,$acc2
        movzb   ($sbox,$acc0,1),$t0
        movzb   ($sbox,$acc1,1),$t1
        movzb   ($sbox,$acc2,1),$t2

        movzb   `&lo("$s3")`,$acc0
        movzb   `&hi("$s3")`,$acc1
        movzb   `&hi("$s0")`,$acc2
        movzb   ($sbox,$acc0,1),$t3
        movzb   ($sbox,$acc1,1),$acc1   #$t0
        movzb   ($sbox,$acc2,1),$acc2   #$t1

        shl     \$8,$acc1
        shl     \$8,$acc2

        xor     $acc1,$t0
        xor     $acc2,$t1
        shr     \$16,$s3

        movzb   `&hi("$s1")`,$acc0
        movzb   `&hi("$s2")`,$acc1
        shr     \$16,$s0
        movzb   ($sbox,$acc0,1),$acc0   #$t2
        movzb   ($sbox,$acc1,1),$acc1   #$t3

        shl     \$8,$acc0
        shl     \$8,$acc1
        shr     \$16,$s1
        xor     $acc0,$t2
        xor     $acc1,$t3
        shr     \$16,$s2

        movzb   `&lo("$s2")`,$acc0
        movzb   `&lo("$s3")`,$acc1
        movzb   `&lo("$s0")`,$acc2
        movzb   ($sbox,$acc0,1),$acc0   #$t0
        movzb   ($sbox,$acc1,1),$acc1   #$t1
        movzb   ($sbox,$acc2,1),$acc2   #$t2

        shl     \$16,$acc0
        shl     \$16,$acc1
        shl     \$16,$acc2

        xor     $acc0,$t0
        xor     $acc1,$t1
        xor     $acc2,$t2

        movzb   `&lo("$s1")`,$acc0
        movzb   `&hi("$s1")`,$acc1
        movzb   `&hi("$s2")`,$acc2
        movzb   ($sbox,$acc0,1),$acc0   #$t3
        movzb   ($sbox,$acc1,1),$acc1   #$t0
        movzb   ($sbox,$acc2,1),$acc2   #$t1

        shl     \$16,$acc0
        shl     \$24,$acc1
        shl     \$24,$acc2

        xor     $acc0,$t3
        xor     $acc1,$t0
        xor     $acc2,$t1
```

```
789         movzb   `&hi("$s3")`,$acc0
790         movzb   `&hi("$s0")`,$acc1
791         mov     16+12($key),$s3
792         movzb   ($sbox,$acc0,1),$acc0   #$t2
793         movzb   ($sbox,$acc1,1),$acc1   #$t3
794         mov     16+0($key),$s0

796         shl     \$24,$acc0
797         shl     \$24,$acc1

799         xor     $acc0,$t2
800         xor     $acc1,$t3

802         mov     16+4($key),$s1
803         mov     16+8($key),$s2
804         lea     -2048($sbox),$sbox
805         xor     $t0,$s0
806         xor     $t1,$s1
807         xor     $t2,$s2
808         xor     $t3,$s3
809 ___
810 }

812 sub decstep()
813 { my ($i,@s) = @_;
814   my $tmp0=$acc0;
815   my $tmp1=$acc1;
816   my $tmp2=$acc2;
817   my $out=($t0,$t1,$t2,$s[0])[$i];

819         $code.="        mov     $s[0],$out\n"           if ($i!=3);
820                         $tmp1=$s[2]                     if ($i==3);
821         $code.="        mov     $s[2],$tmp1\n"          if ($i!=3);
822         $code.="        and     \$0xFF,$out\n";

824         $code.="        mov     0($sbox,$out,8),$out\n";
825         $code.="        shr     \$16,$tmp1\n";
826                         $tmp2=$s[3]                     if ($i==3);
827         $code.="        mov     $s[3],$tmp2\n"          if ($i!=3);

829                         $tmp0=$s[1]                     if ($i==3);
830         $code.="        movzb   ".&hi($s[1])."",$tmp0\n";
831         $code.="        and     \$0xFF,$tmp1\n";
832         $code.="        shr     \$24,$tmp2\n";

834         $code.="        xor     3($sbox,$tmp0,8),$out\n";
835         $code.="        xor     2($sbox,$tmp1,8),$out\n";
836         $code.="        xor     1($sbox,$tmp2,8),$out\n";

838         $code.="        mov     $t2,$s[1]\n"            if ($i==3);
839         $code.="        mov     $t1,$s[2]\n"            if ($i==3);
840         $code.="        mov     $t0,$s[3]\n"            if ($i==3);
841         $code.="\n";
842 }

844 sub declast()
845 { my ($i,@s)=@_;
846   my $tmp0=$acc0;
847   my $tmp1=$acc1;
848   my $tmp2=$acc2;
849   my $out=($t0,$t1,$t2,$s[0])[$i];

851         $code.="        mov     $s[0],$out\n"           if ($i!=3);
852                         $tmp1=$s[2]                     if ($i==3);
853         $code.="        mov     $s[2],$tmp1\n"          if ($i!=3);
```



```
854         $code.="        and     \$0xFF,$out\n";

856         $code.="        movzb   2048($sbox,$out,1),$out\n";
857         $code.="        shr     \$16,$tmp1\n";
858                         $tmp2=$s[3]                     if ($i==3);
859         $code.="        mov     $s[3],$tmp2\n"          if ($i!=3);

861                         $tmp0=$s[1]                     if ($i==3);
862         $code.="        movzb   ".&hi($s[1])."",$tmp0\n";
863         $code.="        and     \$0xFF,$tmp1\n";
864         $code.="        shr     \$24,$tmp2\n";

866         $code.="        movzb   2048($sbox,$tmp0,1),$tmp0\n";
867         $code.="        movzb   2048($sbox,$tmp1,1),$tmp1\n";
868         $code.="        movzb   2048($sbox,$tmp2,1),$tmp2\n";

870         $code.="        shl     \$8,$tmp0\n";
871         $code.="        shl     \$16,$tmp1\n";
872         $code.="        shl     \$24,$tmp2\n";

874         $code.="        xor     $tmp0,$out\n";
875         $code.="        mov     $t2,$s[1]\n"            if ($i==3);
876         $code.="        xor     $tmp1,$out\n";
877         $code.="        mov     $t1,$s[2]\n"            if ($i==3);
878         $code.="        xor     $tmp2,$out\n";
879         $code.="        mov     $t0,$s[3]\n"            if ($i==3);
880         $code.="\n";
881 }

883 $code.=<<___;
884 .type   _x86_64_AES_decrypt,\@abi-omnipotent
885 .align  16
886 _x86_64_AES_decrypt:
887         xor     0($key),$s0                     # xor with key
888         xor     4($key),$s1
889         xor     8($key),$s2
890         xor     12($key),$s3

892         mov     240($key),$rnds                 # load key->rounds
893         sub     \$1,$rnds
894         jmp     .Ldec_loop
895 .align  16
896 .Ldec_loop:
897 ___
898         if ($verticalspin) { &decvert(); }
899         else {  &decstep(0,$s0,$s3,$s2,$s1);
900                 &decstep(1,$s1,$s0,$s3,$s2);
901                 &decstep(2,$s2,$s1,$s0,$s3);
902                 &decstep(3,$s3,$s2,$s1,$s0);
903                 $code.=<<___;
904                 lea     16($key),$key
905                 xor     0($key),$s0                     # xor with key
906                 xor     4($key),$s1
907                 xor     8($key),$s2
908                 xor     12($key),$s3
909 ___
910         }
911 $code.=<<___;
912         sub     \$1,$rnds
913         jnz     .Ldec_loop
914 ___
915         if ($verticalspin) { &declastvert(); }
916         else {  &declast(0,$s0,$s3,$s2,$s1);
917                 &declast(1,$s1,$s0,$s3,$s2);
918                 &declast(2,$s2,$s1,$s0,$s3);
919                 &declast(3,$s3,$s2,$s1,$s0);
```

```
920                 $code.=<<___;
921                 xor     16+0($key),$s0                  # xor with key
922                 xor     16+4($key),$s1
923                 xor     16+8($key),$s2
924                 xor     16+12($key),$s3
925 ___
926         }
927 $code.=<<___;
928         .byte   0xf3,0xc3                       # rep ret
929 .size   _x86_64_AES_decrypt,.-_x86_64_AES_decrypt
930 ___

932 sub deccompactvert()
933 { my ($t3,$t4,$t5)=("%r8d","%r9d","%r13d");

935 $code.=<<___;
936         movzb   `&lo("$s0")`,$t0
937         movzb   `&lo("$s1")`,$t1
938         movzb   `&lo("$s2")`,$t2
939         movzb   ($sbox,$t0,1),$t0
940         movzb   ($sbox,$t1,1),$t1
941         movzb   ($sbox,$t2,1),$t2

943         movzb   `&lo("$s3")`,$t3
944         movzb   `&hi("$s3")`,$acc0
945         movzb   `&hi("$s0")`,$acc1
946         movzb   ($sbox,$t3,1),$t3
947         movzb   ($sbox,$acc0,1),$t4     #$t0
948         movzb   ($sbox,$acc1,1),$t5     #$t1

950         movzb   `&hi("$s1")`,$acc2
951         movzb   `&hi("$s2")`,$acc0
952         shr     \$16,$s2
953         movzb   ($sbox,$acc2,1),$acc2   #$t2
954         movzb   ($sbox,$acc0,1),$acc0   #$t3
955         shr     \$16,$s3

957         movzb   `&lo("$s2")`,$acc1
958         shl     \$8,$t4
959         shl     \$8,$t5
960         movzb   ($sbox,$acc1,1),$acc1   #$t0
961         xor     $t4,$t0
962         xor     $t5,$t1

964         movzb   `&lo("$s3")`,$t4
965         shr     \$16,$s0
966         shr     \$16,$s1
967         movzb   `&lo("$s0")`,$t5
968         shl     \$8,$acc2
969         shl     \$8,$acc0
970         movzb   ($sbox,$t4,1),$t4       #$t1
971         movzb   ($sbox,$t5,1),$t5       #$t2
972         xor     $acc2,$t2
973         xor     $acc0,$t3

975         movzb   `&lo("$s1")`,$acc2
976         movzb   `&hi("$s1")`,$acc0
977         shl     \$16,$acc1
978         movzb   ($sbox,$acc2,1),$acc2   #$t3
979         movzb   ($sbox,$acc0,1),$acc0   #$t0
980         xor     $acc1,$t0

982         movzb   `&hi("$s2")`,$acc1
983         shl     \$16,$t4
984         shl     \$16,$t5
985         movzb   ($sbox,$acc1,1),$s1     #$t1
```



```
986         xor     $t4,$t1
987         xor     $t5,$t2

989         movzb   `&hi("$s3")`,$acc1
990         shr     \$8,$s0
991         shl     \$16,$acc2
992         movzb   ($sbox,$acc1,1),$s2     #$t2
993         movzb   ($sbox,$s0,1),$s3       #$t3
994         xor     $acc2,$t3

996         shl     \$24,$acc0
997         shl     \$24,$s1
998         shl     \$24,$s2
999         xor     $acc0,$t0
1000        shl     \$24,$s3
1001        xor     $t1,$s1
1002        mov     $t0,$s0
1003        xor     $t2,$s2
1004        xor     $t3,$s3
1005 ___
1006 }

1008 # parallelized version! input is pair of 64-bit values: %rax=s1.s0
1009 # and %rcx=s3.s2, output is four 32-bit values in %eax=s0, %ebx=s1,
1010 # %ecx=s2 and %edx=s3.
1011 sub dectransform()
1012 { my ($tp10,$tp20,$tp40,$tp80,$acc0)=("%rax","%r8", "%r9", "%r10","%rbx");
1013   my ($tp18,$tp28,$tp48,$tp88,$acc8)=("%rcx","%r11","%r12","%r13","%rdx");
1014   my $prefetch = shift;

1016 $code.=<<___;
1017         mov     $tp10,$acc0
1018         mov     $tp18,$acc8
1019         and     $mask80,$acc0
1020         and     $mask80,$acc8
1021         mov     $acc0,$tp40
1022         mov     $acc8,$tp48
1023         shr     \$7,$tp40
1024         lea     ($tp10,$tp10),$tp20
1025         shr     \$7,$tp48
1026         lea     ($tp18,$tp18),$tp28
1027         sub     $tp40,$acc0
1028         sub     $tp48,$acc8
1029         and     $maskfe,$tp20
1030         and     $maskfe,$tp28
1031         and     $mask1b,$acc0
1032         and     $mask1b,$acc8
1033         xor     $tp20,$acc0
1034         xor     $tp28,$acc8
1035         mov     $acc0,$tp20
1036         mov     $acc8,$tp28

1038         and     $mask80,$acc0
1039         and     $mask80,$acc8
1040         mov     $acc0,$tp80
1041         mov     $acc8,$tp88
1042         shr     \$7,$tp80
1043         lea     ($tp20,$tp20),$tp40
1044         shr     \$7,$tp88
1045         lea     ($tp28,$tp28),$tp48
1046         sub     $tp80,$acc0
1047         sub     $tp88,$acc8
1048         and     $maskfe,$tp40
1049         and     $maskfe,$tp48
1050         and     $mask1b,$acc0
1051         and     $mask1b,$acc8
```

```
1052         xor     $tp40,$acc0
1053         xor     $tp48,$acc8
1054         mov     $acc0,$tp40
1055         mov     $acc8,$tp48

1057         and     $mask80,$acc0
1058         and     $mask80,$acc8
1059         mov     $acc0,$tp80
1060         mov     $acc8,$tp88
1061         shr     \$7,$tp80
1062          xor    $tp10,$tp20             # tp2^=tp1
1063         shr     \$7,$tp88
1064          xor    $tp18,$tp28             # tp2^=tp1
1065         sub     $tp80,$acc0
1066         sub     $tp88,$acc8
1067         lea     ($tp40,$tp40),$tp80
1068         lea     ($tp48,$tp48),$tp88
1069          xor    $tp10,$tp40             # tp4^=tp1
1070          xor    $tp18,$tp48             # tp4^=tp1
1071         and     $maskfe,$tp80
1072         and     $maskfe,$tp88
1073         and     $mask1b,$acc0
1074         and     $mask1b,$acc8
1075         xor     $acc0,$tp80
1076         xor     $acc8,$tp88

1078         xor     $tp80,$tp10             # tp1^=tp8
1079         xor     $tp88,$tp18             # tp1^=tp8
1080         xor     $tp80,$tp20             # tp2^tp1^=tp8
1081         xor     $tp88,$tp28             # tp2^tp1^=tp8
1082         mov     $tp10,$acc0
1083         mov     $tp18,$acc8
1084         xor     $tp80,$tp40             # tp4^tp1^=tp8
1085         xor     $tp88,$tp48             # tp4^tp1^=tp8
1086         shr     \$32,$acc0
1087         shr     \$32,$acc8
1088         xor     $tp20,$tp80             # tp8^=tp8^tp2^tp1=tp2^tp1
1089         xor     $tp28,$tp88             # tp8^=tp8^tp2^tp1=tp2^tp1
1090         rol     \$8,‘&LO("$tp10")‘      # ROTATE(tp1^tp8,8)
1091         rol     \$8,‘&LO("$tp18")‘      # ROTATE(tp1^tp8,8)
1092         xor     $tp40,$tp80             # tp2^tp1^=tp8^tp4^tp1=tp8^tp4^tp2
1093         xor     $tp48,$tp88             # tp2^tp1^=tp8^tp4^tp1=tp8^tp4^tp2

1095         rol     \$8,‘&LO("$acc0")‘      # ROTATE(tp1^tp8,8)
1096         rol     \$8,‘&LO("$acc8")‘      # ROTATE(tp1^tp8,8)
1097         xor     ‘&LO("$tp80")‘,‘&LO("$tp10")‘
1098         xor     ‘&LO("$tp88")‘,‘&LO("$tp18")‘
1099         shr     \$32,$tp80
1100         shr     \$32,$tp88
1101         xor     ‘&LO("$tp80")‘,‘&LO("$acc0")‘
1102         xor     ‘&LO("$tp88")‘,‘&LO("$acc8")‘

1104         mov     $tp20,$tp80
1105         mov     $tp28,$tp88
1106         shr     \$32,$tp80
1107         shr     \$32,$tp88
1108         rol     \$24,‘&LO("$tp20")‘     # ROTATE(tp2^tp1^tp8,24)
1109         rol     \$24,‘&LO("$tp28")‘     # ROTATE(tp2^tp1^tp8,24)
1110         rol     \$24,‘&LO("$tp80")‘     # ROTATE(tp2^tp1^tp8,24)
1111         rol     \$24,‘&LO("$tp88")‘     # ROTATE(tp2^tp1^tp8,24)
1112         xor     ‘&LO("$tp20")‘,‘&LO("$tp10")‘
1113         xor     ‘&LO("$tp28")‘,‘&LO("$tp18")‘
1114         mov     $tp40,$tp20
1115         mov     $tp48,$tp28
1116         xor     ‘&LO("$tp80")‘,‘&LO("$acc0")‘
1117         xor     ‘&LO("$tp88")‘,‘&LO("$acc8")‘
```



```
1119         ‘"mov   0($sbox),$mask80"       if ($prefetch)‘
1120         shr     \$32,$tp20
1121         shr     \$32,$tp28
1122         ‘"mov   64($sbox),$maskfe"      if ($prefetch)‘
1123         rol     \$16,‘&LO("$tp40")‘     # ROTATE(tp4^tp1^tp8,16)
1124         rol     \$16,‘&LO("$tp48")‘     # ROTATE(tp4^tp1^tp8,16)
1125         ‘"mov   128($sbox),$mask1b"     if ($prefetch)‘
1126         rol     \$16,‘&LO("$tp20")‘     # ROTATE(tp4^tp1^tp8,16)
1127         rol     \$16,‘&LO("$tp28")‘     # ROTATE(tp4^tp1^tp8,16)
1128         ‘"mov   192($sbox),$tp80"       if ($prefetch)‘
1129         xor     ‘&LO("$tp40")‘,‘&LO("$tp10")‘
1130         xor     ‘&LO("$tp48")‘,‘&LO("$tp18")‘
1131         ‘"mov   256($sbox),$tp88"       if ($prefetch)‘
1132         xor     ‘&LO("$tp20")‘,‘&LO("$acc0")‘
1133         xor     ‘&LO("$tp28")‘,‘&LO("$acc8")‘
1134 ___
1135 }

1137 $code.=<<___;
1138 .type   _x86_64_AES_decrypt_compact,\@abi-omnipotent
1139 .align  16
1140 _x86_64_AES_decrypt_compact:
1141         lea     128($sbox),$inp                 # size optimization
1142         mov     0-128($inp),$acc1               # prefetch Td4
1143         mov     32-128($inp),$acc2
1144         mov     64-128($inp),$t0
1145         mov     96-128($inp),$t1
1146         mov     128-128($inp),$acc1
1147         mov     160-128($inp),$acc2
1148         mov     192-128($inp),$t0
1149         mov     224-128($inp),$t1
1150         jmp     .Ldec_loop_compact

1152 .align  16
1153 .Ldec_loop_compact:
1154                 xor     0($key),$s0             # xor with key
1155                 xor     4($key),$s1
1156                 xor     8($key),$s2
1157                 xor     12($key),$s3
1158                 lea     16($key),$key
1159 ___
1160                 &deccompactvert();
1161 $code.=<<___;
1162                 cmp     16(%rsp),$key
1163                 je      .Ldec_compact_done

1165                 mov     256+0($sbox),$mask80
1166                 shl     \$32,%rbx
1167                 shl     \$32,%rdx
1168                 mov     256+8($sbox),$maskfe
1169                 or      %rbx,%rax
1170                 or      %rdx,%rcx
1171                 mov     256+16($sbox),$mask1b
1172 ___
1173                 &dectransform(1);
1174 $code.=<<___;
1175         jmp     .Ldec_loop_compact
1176 .align  16
1177 .Ldec_compact_done:
1178         xor     0($key),$s0
1179         xor     4($key),$s1
1180         xor     8($key),$s2
1181         xor     12($key),$s3
1182         .byte   0xf3,0xc3                       # rep ret
1183 .size   _x86_64_AES_decrypt_compact,.-_x86_64_AES_decrypt_compact
```

```
___

# void AES_decrypt (const void *inp,void *out,const AES_KEY *key);
$code.=<<___;
.globl  AES_decrypt
.type   AES_decrypt,\@function,3
.align  16
.globl  asm_AES_decrypt
.hidden asm_AES_decrypt
asm_AES_decrypt:
AES_decrypt:
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15

        # allocate frame "above" key schedule
        mov     %rsp,%r10
        lea     -63(%rdx),%rcx  # %rdx is key argument
        and     \$-64,%rsp
        sub     %rsp,%rcx
        neg     %rcx
        and     \$0x3c0,%rcx
        sub     %rcx,%rsp
        sub     \$32,%rsp

        mov     %rsi,16(%rsp)   # save out
        mov     %r10,24(%rsp)   # save real stack pointer
.Ldec_prologue:

        mov     %rdx,$key
        mov     240($key),$rnds # load rounds

        mov     0(%rdi),$s0     # load input vector
        mov     4(%rdi),$s1
        mov     8(%rdi),$s2
        mov     12(%rdi),$s3

        shl     \$4,$rnds
        lea     ($key,$rnds),%rbp
        mov     $key,(%rsp)     # key schedule
        mov     %rbp,8(%rsp)    # end of key schedule

        # pick Td4 copy which can't "overlap" with stack frame or key schedule
        lea     .LAES_Td+2048(%rip),$sbox
        lea     768(%rsp),%rbp
        sub     $sbox,%rbp
        and     \$0x300,%rbp
        lea     ($sbox,%rbp),$sbox
        shr     \$3,%rbp        # recall "magic" constants!
        add     %rbp,$sbox

        call    _x86_64_AES_decrypt_compact

        mov     16(%rsp),$out   # restore out
        mov     24(%rsp),%rsi   # restore saved stack pointer
        mov     $s0,0($out)     # write output vector
        mov     $s1,4($out)
        mov     $s2,8($out)
        mov     $s3,12($out)

        mov     (%rsi),%r15
        mov     8(%rsi),%r14
        mov     16(%rsi),%r13
```



```
        mov     24(%rsi),%r12
        mov     32(%rsi),%rbp
        mov     40(%rsi),%rbx
        lea     48(%rsi),%rsp
.Ldec_epilogue:
        ret
.size   AES_decrypt,.-AES_decrypt
___
#------------------------------------------------------------------#

sub enckey()
{
$code.=<<___;
        movz    %dl,%esi                # rk[i]>>0
        movzb   -128(%rbp,%rsi),%ebx
        movz    %dh,%esi                # rk[i]>>8
        shl     \$24,%ebx
        xor     %ebx,%eax

        movzb   -128(%rbp,%rsi),%ebx
        shr     \$16,%edx
        movz    %dl,%esi                # rk[i]>>16
        xor     %ebx,%eax

        movzb   -128(%rbp,%rsi),%ebx
        movz    %dh,%esi                # rk[i]>>24
        shl     \$8,%ebx
        xor     %ebx,%eax

        movzb   -128(%rbp,%rsi),%ebx
        shl     \$16,%ebx
        xor     %ebx,%eax

        xor     1024-128(%rbp,%rcx,4),%eax              # rcon
___
}

# int private_AES_set_encrypt_key(const unsigned char *userKey, const int bits,
#                       AES_KEY *key)
$code.=<<___;
.globl  private_AES_set_encrypt_key
.type   private_AES_set_encrypt_key,\@function,3
.align  16
private_AES_set_encrypt_key:
        push    %rbx
        push    %rbp
        push    %r12                    # redundant, but allows to share
        push    %r13                    # exception handler...
        push    %r14
        push    %r15
        sub     \$8,%rsp
.Lenc_key_prologue:

        call    _x86_64_AES_set_encrypt_key

        mov     8(%rsp),%r15
        mov     16(%rsp),%r14
        mov     24(%rsp),%r13
        mov     32(%rsp),%r12
        mov     40(%rsp),%rbp
        mov     48(%rsp),%rbx
        add     \$56,%rsp
.Lenc_key_epilogue:
        ret
.size   private_AES_set_encrypt_key,.-private_AES_set_encrypt_key
```

```
.type   _x86_64_AES_set_encrypt_key,\@abi-omnipotent
.align  16
_x86_64_AES_set_encrypt_key:
        mov     %esi,%ecx                       # %ecx=bits
        mov     %rdi,%rsi                       # %rsi=userKey
        mov     %rdx,%rdi                       # %rdi=key

        test    \$-1,%rsi
        jz      .Lbadpointer
        test    \$-1,%rdi
        jz      .Lbadpointer

        lea     .LAES_Te(%rip),%rbp
        lea     2048+128(%rbp),%rbp

        # prefetch Te4
        mov     0-128(%rbp),%eax
        mov     32-128(%rbp),%ebx
        mov     64-128(%rbp),%r8d
        mov     96-128(%rbp),%edx
        mov     128-128(%rbp),%eax
        mov     160-128(%rbp),%ebx
        mov     192-128(%rbp),%r8d
        mov     224-128(%rbp),%edx

        cmp     \$128,%ecx
        je      .L10rounds
        cmp     \$192,%ecx
        je      .L12rounds
        cmp     \$256,%ecx
        je      .L14rounds
        mov     \$-2,%rax                       # invalid number of bits
        jmp     .Lexit

.L10rounds:
        mov     0(%rsi),%rax                    # copy first 4 dwords
        mov     8(%rsi),%rdx
        mov     %rax,0(%rdi)
        mov     %rdx,8(%rdi)

        shr     \$32,%rdx
        xor     %ecx,%ecx
        jmp     .L10shortcut
.align  4
.L10loop:
                mov     0(%rdi),%eax                    # rk[0]
                mov     12(%rdi),%edx                   # rk[3]
.L10shortcut:
___
                &enckey ();
$code.=<<___;
                mov     %eax,16(%rdi)                   # rk[4]
                xor     4(%rdi),%eax
                mov     %eax,20(%rdi)                   # rk[5]
                xor     8(%rdi),%eax
                mov     %eax,24(%rdi)                   # rk[6]
                xor     12(%rdi),%eax
                mov     %eax,28(%rdi)                   # rk[7]
                add     \$1,%ecx
                lea     16(%rdi),%rdi
                cmp     \$10,%ecx
        jl      .L10loop

        movl    \$10,80(%rdi)                   # setup number of rounds
        xor     %rax,%rax
        jmp     .Lexit
```



```
.L12rounds:
        mov     0(%rsi),%rax                    # copy first 6 dwords
        mov     8(%rsi),%rbx
        mov     16(%rsi),%rdx
        mov     %rax,0(%rdi)
        mov     %rbx,8(%rdi)
        mov     %rdx,16(%rdi)

        shr     \$32,%rdx
        xor     %ecx,%ecx
        jmp     .L12shortcut
.align  4
.L12loop:
                mov     0(%rdi),%eax                    # rk[0]
                mov     20(%rdi),%edx                   # rk[5]
.L12shortcut:
___
                &enckey ();
$code.=<<___;
                mov     %eax,24(%rdi)                   # rk[6]
                xor     4(%rdi),%eax
                mov     %eax,28(%rdi)                   # rk[7]
                xor     8(%rdi),%eax
                mov     %eax,32(%rdi)                   # rk[8]
                xor     12(%rdi),%eax
                mov     %eax,36(%rdi)                   # rk[9]

                cmp     \$7,%ecx
                je      .L12break
                add     \$1,%ecx

                xor     16(%rdi),%eax
                mov     %eax,40(%rdi)                   # rk[10]
                xor     20(%rdi),%eax
                mov     %eax,44(%rdi)                   # rk[11]

                lea     24(%rdi),%rdi
        jmp     .L12loop
.L12break:
        movl    \$12,72(%rdi)           # setup number of rounds
        xor     %rax,%rax
        jmp     .Lexit

.L14rounds:
        mov     0(%rsi),%rax                    # copy first 8 dwords
        mov     8(%rsi),%rbx
        mov     16(%rsi),%rcx
        mov     24(%rsi),%rdx
        mov     %rax,0(%rdi)
        mov     %rbx,8(%rdi)
        mov     %rcx,16(%rdi)
        mov     %rdx,24(%rdi)

        shr     \$32,%rdx
        xor     %ecx,%ecx
        jmp     .L14shortcut
.align  4
.L14loop:
                mov     0(%rdi),%eax                    # rk[0]
                mov     28(%rdi),%edx                   # rk[4]
.L14shortcut:
___
                &enckey ();
$code.=<<___;
                mov     %eax,32(%rdi)                   # rk[8]
```

```
1448                xor     4(%rdi),%eax
1449                mov     %eax,36(%rdi)                   # rk[9]
1450                xor     8(%rdi),%eax
1451                mov     %eax,40(%rdi)                   # rk[10]
1452                xor     12(%rdi),%eax
1453                mov     %eax,44(%rdi)                   # rk[11]

1455                cmp     \$6,%ecx
1456                je      .L14break
1457                add     \$1,%ecx

1459                mov     %eax,%edx
1460                mov     16(%rdi),%eax                   # rk[4]
1461                movz    %dl,%esi                        # rk[11]>>0
1462                movzb   -128(%rbp,%rsi),%ebx
1463                movz    %dh,%esi                        # rk[11]>>8
1464                xor     %ebx,%eax

1466                movzb   -128(%rbp,%rsi),%ebx
1467                shr     \$16,%edx
1468                shl     \$8,%ebx
1469                movz    %dl,%esi                        # rk[11]>>16
1470                xor     %ebx,%eax

1472                movzb   -128(%rbp,%rsi),%ebx
1473                movz    %dh,%esi                        # rk[11]>>24
1474                shl     \$16,%ebx
1475                xor     %ebx,%eax

1477                movzb   -128(%rbp,%rsi),%ebx
1478                shl     \$24,%ebx
1479                xor     %ebx,%eax

1481                mov     %eax,48(%rdi)                   # rk[12]
1482                xor     20(%rdi),%eax
1483                mov     %eax,52(%rdi)                   # rk[13]
1484                xor     24(%rdi),%eax
1485                mov     %eax,56(%rdi)                   # rk[14]
1486                xor     28(%rdi),%eax
1487                mov     %eax,60(%rdi)                   # rk[15]

1489                lea     32(%rdi),%rdi
1490        jmp     .L14loop
1491 .L14break:
1492        movl    \$14,48(%rdi)           # setup number of rounds
1493        xor     %rax,%rax
1494        jmp     .Lexit

1496 .Lbadpointer:
1497        mov     \$-1,%rax
1498 .Lexit:
1499        .byte   0xf3,0xc3                       # rep ret
1500 .size   _x86_64_AES_set_encrypt_key,.-_x86_64_AES_set_encrypt_key
1501 ___

1503 sub deckey_ref()
1504 { my ($i,$ptr,$te,$td) = @_;
1505   my ($tp1,$tp2,$tp4,$tp8,$acc)=("%eax","%ebx","%edi","%edx","%r8d");
1506 $code.=<<___;
1507        mov     $i($ptr),$tp1
1508        mov     $tp1,$acc
1509        and     \$0x80808080,$acc
1510        mov     $acc,$tp4
1511        shr     \$7,$tp4
1512        lea     0($tp1,$tp1),$tp2
1513        sub     $tp4,$acc
```



```
1514        and     \$0xfefefefe,$tp2
1515        and     \$0x1b1b1b1b,$acc
1516        xor     $tp2,$acc
1517        mov     $acc,$tp2

1519        and     \$0x80808080,$acc
1520        mov     $acc,$tp8
1521        shr     \$7,$tp8
1522        lea     0($tp2,$tp2),$tp4
1523        sub     $tp8,$acc
1524        and     \$0xfefefefe,$tp4
1525        and     \$0x1b1b1b1b,$acc
1526         xor    $tp1,$tp2               # tp2^tp1
1527        xor     $tp4,$acc
1528        mov     $acc,$tp4

1530        and     \$0x80808080,$acc
1531        mov     $acc,$tp8
1532        shr     \$7,$tp8
1533        sub     $tp8,$acc
1534        lea     0($tp4,$tp4),$tp8
1535         xor    $tp1,$tp4               # tp4^tp1
1536        and     \$0xfefefefe,$tp8
1537        and     \$0x1b1b1b1b,$acc
1538        xor     $acc,$tp8

1540        xor     $tp8,$tp1               # tp1^tp8
1541        rol     \$8,$tp1                # ROTATE(tp1^tp8,8)
1542        xor     $tp8,$tp2               # tp2^tp1^tp8
1543        xor     $tp8,$tp4               # tp4^tp1^tp8
1544        xor     $tp2,$tp8
1545        xor     $tp4,$tp8               # tp8^(tp8^tp4^tp1)^(tp8^tp2^tp1)=tp8^tp

1547        xor     $tp8,$tp1
1548        rol     \$24,$tp2               # ROTATE(tp2^tp1^tp8,24)
1549        xor     $tp2,$tp1
1550        rol     \$16,$tp4               # ROTATE(tp4^tp1^tp8,16)
1551        xor     $tp4,$tp1

1553        mov     $tp1,$i($ptr)
1554 ___
1555 }

1557 # int private_AES_set_decrypt_key(const unsigned char *userKey, const int bits,
1558 #                       AES_KEY *key)
1559 $code.=<<___;
1560 .globl  private_AES_set_decrypt_key
1561 .type   private_AES_set_decrypt_key,\@function,3
1562 .align  16
1563 private_AES_set_decrypt_key:
1564        push    %rbx
1565        push    %rbp
1566        push    %r12
1567        push    %r13
1568        push    %r14
1569        push    %r15
1570        push    %rdx                    # save key schedule
1571 .Ldec_key_prologue:

1573        call    _x86_64_AES_set_encrypt_key
1574        mov     (%rsp),%r8              # restore key schedule
1575        cmp     \$0,%eax
1576        jne     .Labort

1578        mov     240(%r8),%r14d          # pull number of rounds
1579        xor     %rdi,%rdi
```

```
1580         lea     (%rdi,%r14d,4),%rcx
1581         mov     %r8,%rsi
1582         lea     (%r8,%rcx,4),%rdi       # pointer to last chunk
1583 .align  4
1584 .Linvert:
1585                 mov     0(%rsi),%rax
1586                 mov     8(%rsi),%rbx
1587                 mov     0(%rdi),%rcx
1588                 mov     8(%rdi),%rdx
1589                 mov     %rax,0(%rdi)
1590                 mov     %rbx,8(%rdi)
1591                 mov     %rcx,0(%rsi)
1592                 mov     %rdx,8(%rsi)
1593                 lea     16(%rsi),%rsi
1594                 lea     -16(%rdi),%rdi
1595                 cmp     %rsi,%rdi
1596         jne     .Linvert

1598         lea     .LAES_Te+2048+1024(%rip),%rax   # rcon

1600         mov     40(%rax),$mask80
1601         mov     48(%rax),$maskfe
1602         mov     56(%rax),$mask1b

1604         mov     %r8,$key
1605         sub     \$1,%r14d
1606 .align  4
1607 .Lpermute:
1608                 lea     16($key),$key
1609                 mov     0($key),%rax
1610                 mov     8($key),%rcx
1611 ___
1612                 &dectransform ();
1613 $code.=<<___;
1614                 mov     %eax,0($key)
1615                 mov     %ebx,4($key)
1616                 mov     %ecx,8($key)
1617                 mov     %edx,12($key)
1618                 sub     \$1,%r14d
1619         jnz     .Lpermute

1621         xor     %rax,%rax
1622 .Labort:
1623         mov     8(%rsp),%r15
1624         mov     16(%rsp),%r14
1625         mov     24(%rsp),%r13
1626         mov     32(%rsp),%r12
1627         mov     40(%rsp),%rbp
1628         mov     48(%rsp),%rbx
1629         add     \$56,%rsp
1630 .Ldec_key_epilogue:
1631         ret
1632 .size   private_AES_set_decrypt_key,.-private_AES_set_decrypt_key
1633 ___

1635 # void AES_cbc_encrypt (const void char *inp, unsigned char *out,
1636 #                       size_t length, const AES_KEY *key,
1637 #                       unsigned char *ivp,const int enc);
1638 {
1639 # stack frame layout
1640 # -8(%rsp)              return address
1641 my $keyp="0(%rsp)";             # one to pass as $key
1642 my $keyend="8(%rsp)";           # &(keyp->rd_key[4*keyp->rounds])
1643 my $_rsp="16(%rsp)";            # saved %rsp
1644 my $_inp="24(%rsp)";            # copy of 1st parameter, inp
1645 my $_out="32(%rsp)";            # copy of 2nd parameter, out
```



```
1646 my $_len="40(%rsp)";            # copy of 3rd parameter, length
1647 my $_key="48(%rsp)";            # copy of 4th parameter, key
1648 my $_ivp="56(%rsp)";            # copy of 5th parameter, ivp
1649 my $ivec="64(%rsp)";            # ivec[16]
1650 my $aes_key="80(%rsp)";         # copy of aes_key
1651 my $mark="80+240(%rsp)";        # copy of aes_key->rounds

1653 $code.=<<___;
1654 .globl  AES_cbc_encrypt
1655 .type   AES_cbc_encrypt,\@function,6
1656 .align  16
1657 .extern OPENSSL_ia32cap_P
1658 .globl  asm_AES_cbc_encrypt
1659 .hidden asm_AES_cbc_encrypt
1660 asm_AES_cbc_encrypt:
1661 AES_cbc_encrypt:
1662         cmp     \$0,%rdx        # check length
1663         je      .Lcbc_epilogue
1664         pushfq
1665         push    %rbx
1666         push    %rbp
1667         push    %r12
1668         push    %r13
1669         push    %r14
1670         push    %r15
1671 .Lcbc_prologue:

1673         cld
1674         mov     %r9d,%r9d       # clear upper half of enc

1676         lea     .LAES_Te(%rip),$sbox
1677         cmp     \$0,%r9
1678         jne     .Lcbc_picked_te
1679         lea     .LAES_Td(%rip),$sbox
1680 .Lcbc_picked_te:

1682         mov     OPENSSL_ia32cap_P(%rip),%r10d
1683         cmp     \$$speed_limit,%rdx
1684         jb      .Lcbc_slow_prologue
1685         test    \$15,%rdx
1686         jnz     .Lcbc_slow_prologue
1687         bt      \$28,%r10d
1688         jc      .Lcbc_slow_prologue

1690         # allocate aligned stack frame...
1691         lea     -88-248(%rsp),$key
1692         and     \$-64,$key

1694         # ... and make sure it doesn't alias with AES_T[ed] modulo 4096
1695         mov     $sbox,%r10
1696         lea     2304($sbox),%r11
1697         mov     $key,%r12
1698         and     \$0xFFF,%r10    # s = $sbox&0xfff
1699         and     \$0xFFF,%r11    # e = ($sbox+2048)&0xfff
1700         and     \$0xFFF,%r12    # p = %rsp&0xfff

1702         cmp     %r11,%r12       # if (p=>e) %rsp =- (p-e);
1703         jb      .Lcbc_te_break_out
1704         sub     %r11,%r12
1705         sub     %r12,$key
1706         jmp     .Lcbc_te_ok
1707 .Lcbc_te_break_out:             # else %rsp -= (p-s)&0xfff + framesz
1708         sub     %r10,%r12
1709         and     \$0xFFF,%r12
1710         add     \$320,%r12
1711         sub     %r12,$key
```

```
.align  4
.Lcbc_te_ok:

        xchg    %rsp,$key
        #add    \$8,%rsp        # reserve for return address!
        mov     $key,$_rsp      # save %rsp
.Lcbc_fast_body:
        mov     %rdi,$_inp      # save copy of inp
        mov     %rsi,$_out      # save copy of out
        mov     %rdx,$_len      # save copy of len
        mov     %rcx,$_key      # save copy of key
        mov     %r8,$_ivp       # save copy of ivp
        movl    \$0,$mark       # copy of aes_key->rounds = 0;
        mov     %r8,%rbp        # rearrange input arguments
        mov     %r9,%rbx
        mov     %rsi,$out
        mov     %rdi,$inp
        mov     %rcx,$key

        mov     240($key),%eax          # key->rounds
        # do we copy key schedule to stack?
        mov     $key,%r10
        sub     $sbox,%r10
        and     \$0xfff,%r10
        cmp     \$2304,%r10
        jb      .Lcbc_do_ecopy
        cmp     \$4096-248,%r10
        jb      .Lcbc_skip_ecopy
.align  4
.Lcbc_do_ecopy:
                mov     $key,%rsi
                lea     $aes_key,%rdi
                lea     $aes_key,$key
                mov     \$240/8,%ecx
                .long   0x90A548F3      # rep movsq
                mov     %eax,(%rdi)     # copy aes_key->rounds
.Lcbc_skip_ecopy:
        mov     $key,$keyp      # save key pointer

        mov     \$18,%ecx
.align  4
.Lcbc_prefetch_te:
                mov     0($sbox),%r10
                mov     32($sbox),%r11
                mov     64($sbox),%r12
                mov     96($sbox),%r13
                lea     128($sbox),$sbox
                sub     \$1,%ecx
        jnz     .Lcbc_prefetch_te
        lea     -2304($sbox),$sbox

        cmp     \$0,%rbx
        je      .LFAST_DECRYPT

#---------------------------- ENCRYPT -----------------------------#
        mov     0(%rbp),$s0             # load iv
        mov     4(%rbp),$s1
        mov     8(%rbp),$s2
        mov     12(%rbp),$s3

.align  4
.Lcbc_fast_enc_loop:
                xor     0($inp),$s0
                xor     4($inp),$s1
                xor     8($inp),$s2
                xor     12($inp),$s3
```



```
                mov     $keyp,$key      # restore key
                mov     $inp,$_inp      # if ($verticalspin) save inp

                call    _x86_64_AES_encrypt

                mov     $_inp,$inp      # if ($verticalspin) restore inp
                mov     $_len,%r10
                mov     $s0,0($out)
                mov     $s1,4($out)
                mov     $s2,8($out)
                mov     $s3,12($out)

                lea     16($inp),$inp
                lea     16($out),$out
                sub     \$16,%r10
                test    \$-16,%r10
                mov     %r10,$_len
        jnz     .Lcbc_fast_enc_loop
        mov     $_ivp,%rbp      # restore ivp
        mov     $s0,0(%rbp)     # save ivec
        mov     $s1,4(%rbp)
        mov     $s2,8(%rbp)
        mov     $s3,12(%rbp)

        jmp     .Lcbc_fast_cleanup

#---------------------------- DECRYPT -----------------------------#
.align  16
.LFAST_DECRYPT:
        cmp     $inp,$out
        je      .Lcbc_fast_dec_in_place

        mov     %rbp,$ivec
.align  4
.Lcbc_fast_dec_loop:
                mov     0($inp),$s0     # read input
                mov     4($inp),$s1
                mov     8($inp),$s2
                mov     12($inp),$s3
                mov     $keyp,$key      # restore key
                mov     $inp,$_inp      # if ($verticalspin) save inp

                call    _x86_64_AES_decrypt

                mov     $ivec,%rbp      # load ivp
                mov     $_inp,$inp      # if ($verticalspin) restore inp
                mov     $_len,%r10      # load len
                xor     0(%rbp),$s0     # xor iv
                xor     4(%rbp),$s1
                xor     8(%rbp),$s2
                xor     12(%rbp),$s3
                mov     $inp,%rbp       # current input, next iv

                sub     \$16,%r10
                mov     %r10,$_len      # update len
                mov     %rbp,$ivec      # update ivp

                mov     $s0,0($out)     # write output
                mov     $s1,4($out)
                mov     $s2,8($out)
                mov     $s3,12($out)

                lea     16($inp),$inp
                lea     16($out),$out
        jnz     .Lcbc_fast_dec_loop
        mov     $_ivp,%r12              # load user ivp
```

```
1844         mov     0(%rbp),%r10            # load iv
1845         mov     8(%rbp),%r11
1846         mov     %r10,0(%r12)            # copy back to user
1847         mov     %r11,8(%r12)
1848         jmp     .Lcbc_fast_cleanup

1850 .align  16
1851 .Lcbc_fast_dec_in_place:
1852         mov     0(%rbp),%r10            # copy iv to stack
1853         mov     8(%rbp),%r11
1854         mov     %r10,0+$ivec
1855         mov     %r11,8+$ivec
1856 .align  4
1857 .Lcbc_fast_dec_in_place_loop:
1858                 mov     0($inp),$s0     # load input
1859                 mov     4($inp),$s1
1860                 mov     8($inp),$s2
1861                 mov     12($inp),$s3
1862                 mov     $keyp,$key      # restore key
1863                 mov     $inp,$_inp      # if ($verticalspin) save inp

1865                 call    _x86_64_AES_decrypt

1867                 mov     $_inp,$inp      # if ($verticalspin) restore inp
1868                 mov     $_len,%r10
1869                 xor     0+$ivec,$s0
1870                 xor     4+$ivec,$s1
1871                 xor     8+$ivec,$s2
1872                 xor     12+$ivec,$s3

1874                 mov     0($inp),%r11    # load input
1875                 mov     8($inp),%r12
1876                 sub     \$16,%r10
1877                 jz      .Lcbc_fast_dec_in_place_done

1879                 mov     %r11,0+$ivec    # copy input to iv
1880                 mov     %r12,8+$ivec

1882                 mov     $s0,0($out)     # save output [zaps input]
1883                 mov     $s1,4($out)
1884                 mov     $s2,8($out)
1885                 mov     $s3,12($out)

1887                 lea     16($inp),$inp
1888                 lea     16($out),$out
1889                 mov     %r10,$_len
1890         jmp     .Lcbc_fast_dec_in_place_loop
1891 .Lcbc_fast_dec_in_place_done:
1892         mov     $_ivp,%rdi
1893         mov     %r11,0(%rdi)    # copy iv back to user
1894         mov     %r12,8(%rdi)

1896         mov     $s0,0($out)     # save output [zaps input]
1897         mov     $s1,4($out)
1898         mov     $s2,8($out)
1899         mov     $s3,12($out)

1901 .align  4
1902 .Lcbc_fast_cleanup:
1903         cmpl    \$0,$mark       # was the key schedule copied?
1904         lea     $aes_key,%rdi
1905         je      .Lcbc_exit
1906                 mov     \$240/8,%ecx
1907                 xor     %rax,%rax
1908                 .long   0x90AB48F3      # rep stosq
```



```
1910         jmp     .Lcbc_exit

1912 #---------------------------- SLOW ROUTINE --------------------------#
1913 .align  16
1914 .Lcbc_slow_prologue:
1915         # allocate aligned stack frame...
1916         lea     -88(%rsp),%rbp
1917         and     \$-64,%rbp
1918         # ... just "above" key schedule
1919         lea     -88-63(%rcx),%r10
1920         sub     %rbp,%r10
1921         neg     %r10
1922         and     \$0x3c0,%r10
1923         sub     %r10,%rbp

1925         xchg    %rsp,%rbp
1926         #add    \$8,%rsp        # reserve for return address!
1927         mov     %rbp,$_rsp      # save %rsp
1928 .Lcbc_slow_body:
1929         #mov    %rdi,$_inp      # save copy of inp
1930         #mov    %rsi,$_out      # save copy of out
1931         #mov    %rdx,$_len      # save copy of len
1932         #mov    %rcx,$_key      # save copy of key
1933         mov     %r8,$_ivp       # save copy of ivp
1934         mov     %r8,%rbp        # rearrange input arguments
1935         mov     %r9,%rbx
1936         mov     %rsi,$out
1937         mov     %rdi,$inp
1938         mov     %rcx,$key
1939         mov     %rdx,%r10

1941         mov     240($key),%eax
1942         mov     $key,$keyp      # save key pointer
1943         shl     \$4,%eax
1944         lea     ($key,%rax),%rax
1945         mov     %rax,$keyend

1947         # pick Te4 copy which can't "overlap" with stack frame or key scdedule
1948         lea     2048($sbox),$sbox
1949         lea     768-8(%rsp),%rax
1950         sub     $sbox,%rax
1951         and     \$0x300,%rax
1952         lea     ($sbox,%rax),$sbox

1954         cmp     \$0,%rbx
1955         je      .LSLOW_DECRYPT

1957 #---------------------------- SLOW ENCRYPT --------------------------#
1958         test    \$-16,%r10              # check upon length
1959         mov     0(%rbp),$s0             # load iv
1960         mov     4(%rbp),$s1
1961         mov     8(%rbp),$s2
1962         mov     12(%rbp),$s3
1963         jz      .Lcbc_slow_enc_tail     # short input...

1965 .align  4
1966 .Lcbc_slow_enc_loop:
1967                 xor     0($inp),$s0
1968                 xor     4($inp),$s1
1969                 xor     8($inp),$s2
1970                 xor     12($inp),$s3
1971                 mov     $keyp,$key      # restore key
1972                 mov     $inp,$_inp      # save inp
1973                 mov     $out,$_out      # save out
1974                 mov     %r10,$_len      # save len
```

```
1976                call    _x86_64_AES_encrypt_compact

1978                mov     $_inp,$inp      # restore inp
1979                mov     $_out,$out      # restore out
1980                mov     $_len,%r10      # restore len
1981                mov     $s0,0($out)
1982                mov     $s1,4($out)
1983                mov     $s2,8($out)
1984                mov     $s3,12($out)

1986                lea     16($inp),$inp
1987                lea     16($out),$out
1988                sub     \$16,%r10
1989                test    \$-16,%r10
1990        jnz     .Lcbc_slow_enc_loop
1991        test    \$15,%r10
1992        jnz     .Lcbc_slow_enc_tail
1993        mov     $_ivp,%rbp      # restore ivp
1994        mov     $s0,0(%rbp)     # save ivec
1995        mov     $s1,4(%rbp)
1996        mov     $s2,8(%rbp)
1997        mov     $s3,12(%rbp)

1999        jmp     .Lcbc_exit

2001 .align  4
2002 .Lcbc_slow_enc_tail:
2003        mov     %rax,%r11
2004        mov     %rcx,%r12
2005        mov     %r10,%rcx
2006        mov     $inp,%rsi
2007        mov     $out,%rdi
2008        .long   0x9066A4F3              # rep movsb
2009        mov     \$16,%rcx               # zero tail
2010        sub     %r10,%rcx
2011        xor     %rax,%rax
2012        .long   0x9066AAF3              # rep stosb
2013        mov     $out,$inp               # this is not a mistake!
2014        mov     \$16,%r10               # len=16
2015        mov     %r11,%rax
2016        mov     %r12,%rcx
2017        jmp     .Lcbc_slow_enc_loop     # one more spin...
2018 #----------------------------- SLOW DECRYPT --------------------------#
2019 .align  16
2020 .LSLOW_DECRYPT:
2021        shr     \$3,%rax
2022        add     %rax,$sbox              # recall "magic" constants!

2024        mov     0(%rbp),%r11            # copy iv to stack
2025        mov     8(%rbp),%r12
2026        mov     %r11,0+$ivec
2027        mov     %r12,8+$ivec

2029 .align  4
2030 .Lcbc_slow_dec_loop:
2031                mov     0($inp),$s0     # load input
2032                mov     4($inp),$s1
2033                mov     8($inp),$s2
2034                mov     12($inp),$s3
2035                mov     $keyp,$key      # restore key
2036                mov     $inp,$_inp      # save inp
2037                mov     $out,$_out      # save out
2038                mov     %r10,$_len      # save len

2040                call    _x86_64_AES_decrypt_compact
```



```
2042                mov     $_inp,$inp      # restore inp
2043                mov     $_out,$out      # restore out
2044                mov     $_len,%r10
2045                xor     0+$ivec,$s0
2046                xor     4+$ivec,$s1
2047                xor     8+$ivec,$s2
2048                xor     12+$ivec,$s3

2050                mov     0($inp),%r11    # load input
2051                mov     8($inp),%r12
2052                sub     \$16,%r10
2053                jc      .Lcbc_slow_dec_partial
2054                jz      .Lcbc_slow_dec_done

2056                mov     %r11,0+$ivec    # copy input to iv
2057                mov     %r12,8+$ivec

2059                mov     $s0,0($out)     # save output [can zap input]
2060                mov     $s1,4($out)
2061                mov     $s2,8($out)
2062                mov     $s3,12($out)

2064                lea     16($inp),$inp
2065                lea     16($out),$out
2066        jmp     .Lcbc_slow_dec_loop
2067 .Lcbc_slow_dec_done:
2068        mov     $_ivp,%rdi
2069        mov     %r11,0(%rdi)            # copy iv back to user
2070        mov     %r12,8(%rdi)

2072        mov     $s0,0($out)             # save output [can zap input]
2073        mov     $s1,4($out)
2074        mov     $s2,8($out)
2075        mov     $s3,12($out)

2077        jmp     .Lcbc_exit

2079 .align  4
2080 .Lcbc_slow_dec_partial:
2081        mov     $_ivp,%rdi
2082        mov     %r11,0(%rdi)            # copy iv back to user
2083        mov     %r12,8(%rdi)

2085        mov     $s0,0+$ivec             # save output to stack
2086        mov     $s1,4+$ivec
2087        mov     $s2,8+$ivec
2088        mov     $s3,12+$ivec

2090        mov     $out,%rdi
2091        lea     $ivec,%rsi
2092        lea     16(%r10),%rcx
2093        .long   0x9066A4F3      # rep movsb
2094        jmp     .Lcbc_exit

2096 .align  16
2097 .Lcbc_exit:
2098        mov     $_rsp,%rsi
2099        mov     (%rsi),%r15
2100        mov     8(%rsi),%r14
2101        mov     16(%rsi),%r13
2102        mov     24(%rsi),%r12
2103        mov     32(%rsi),%rbp
2104        mov     40(%rsi),%rbx
2105        lea     48(%rsi),%rsp
2106 .Lcbc_popfq:
2107        popfq
```

```
.Lcbc_epilogue:
        ret
.size   AES_cbc_encrypt,.-AES_cbc_encrypt
___
}

$code.=<<___;
.align  64
.LAES_Te:
___
        &_data_word(0xa56363c6, 0x847c7cf8, 0x997777ee, 0x8d7b7bf6);
        &_data_word(0x0df2f2ff, 0xbd6b6bd6, 0xb16f6fde, 0x54c5c591);
        &_data_word(0x50303060, 0x03010102, 0xa96767ce, 0x7d2b2b56);
        &_data_word(0x19fefee7, 0x62d7d7b5, 0xe6abab4d, 0x9a7676ec);
        &_data_word(0x45caca8f, 0x9d82821f, 0x40c9c989, 0x877d7dfa);
        &_data_word(0x15fafaef, 0xeb5959b2, 0xc947478e, 0x0bf0f0fb);
        &_data_word(0xecadad41, 0x67d4d4b3, 0xfda2a25f, 0xeaafaf45);
        &_data_word(0xbf9c9c23, 0xf7a4a453, 0x967272e4, 0x5bc0c09b);
        &_data_word(0xc2b7b775, 0x1cfdfde1, 0xae93933d, 0x6a26264c);
        &_data_word(0x5a36366c, 0x413f3f7e, 0x02f7f7f5, 0x4fcccc83);
        &_data_word(0x5c343468, 0xf4a5a551, 0x34e5e5d1, 0x08f1f1f9);
        &_data_word(0x937171e2, 0x73d8d8ab, 0x53313162, 0x3f15152a);
        &_data_word(0x0c040408, 0x52c7c795, 0x65232346, 0x5ec3c39d);
        &_data_word(0x28181830, 0xa1969637, 0x0f05050a, 0xb59a9a2f);
        &_data_word(0x0907070e, 0x36121224, 0x9b80801b, 0x3de2e2df);
        &_data_word(0x26ebebcd, 0x6927274e, 0xcdb2b27f, 0x9f7575ea);
        &_data_word(0x1b090912, 0x9e83831d, 0x742c2c58, 0x2e1a1a34);
        &_data_word(0x2d1b1b36, 0xb26e6edc, 0xee5a5ab4, 0xfba0a05b);
        &_data_word(0xf65252a4, 0x4d3b3b76, 0x61d6d6b7, 0xceb3b37d);
        &_data_word(0x7b292952, 0x3ee3e3dd, 0x712f2f5e, 0x97848413);
        &_data_word(0xf55353a6, 0x68d1d1b9, 0x00000000, 0x2cededc1);
        &_data_word(0x60202040, 0x1ffcfce3, 0xc8b1b179, 0xed5b5bb6);
        &_data_word(0xbe6a6ad4, 0x46cbcb8d, 0xd9bebe67, 0x4b393972);
        &_data_word(0xde4a4a94, 0xd44c4c98, 0xe85858b0, 0x4acfcf85);
        &_data_word(0x6bd0d0bb, 0x2aefefc5, 0xe5aaaa4f, 0x16fbfbed);
        &_data_word(0xc5434386, 0xd74d4d9a, 0x55333366, 0x94858511);
        &_data_word(0xcf45458a, 0x10f9f9e9, 0x06020204, 0x817f7ffe);
        &_data_word(0xf05050a0, 0x443c3c78, 0xba9f9f25, 0xe3a8a84b);
        &_data_word(0xf35151a2, 0xfea3a35d, 0xc0404080, 0x8a8f8f05);
        &_data_word(0xad92923f, 0xbc9d9d21, 0x48383870, 0x04f5f5f1);
        &_data_word(0xdfbcbc63, 0xc1b6b677, 0x75dadaaf, 0x63212142);
        &_data_word(0x30101020, 0x1affffe5, 0x0ef3f3fd, 0x6dd2d2bf);
        &_data_word(0x4ccdcd81, 0x140c0c18, 0x35131326, 0x2fececc3);
        &_data_word(0xe15f5fbe, 0xa2979735, 0xcc444488, 0x3917172e);
        &_data_word(0x57c4c493, 0xf2a7a755, 0x827e7efc, 0x473d3d7a);
        &_data_word(0xac6464c8, 0xe75d5dba, 0x2b191932, 0x957373e6);
        &_data_word(0xa06060c0, 0x98818119, 0xd14f4f9e, 0x7fdcdca3);
        &_data_word(0x66222244, 0x7e2a2a54, 0xab90903b, 0x8388880b);
        &_data_word(0xca46468c, 0x29eeeec7, 0xd3b8b86b, 0x3c141428);
        &_data_word(0x79dedea7, 0xe25e5ebc, 0x1d0b0b16, 0x76dbdbad);
        &_data_word(0x3be0e0db, 0x56323264, 0x4e3a3a74, 0x1e0a0a14);
        &_data_word(0xdb494992, 0x0a06060c, 0x6c242448, 0xe45c5cb8);
        &_data_word(0x5dc2c29f, 0x6ed3d3bd, 0xefacac43, 0xa66262c4);
        &_data_word(0xa8919139, 0xa4959531, 0x37e4e4d3, 0x8b7979f2);
        &_data_word(0x32e7e7d5, 0x43c8c88b, 0x5937376e, 0xb76d6dda);
        &_data_word(0x8c8d8d01, 0x64d5d5b1, 0xd24e4e9c, 0xe0a9a949);
        &_data_word(0xb46c6cd8, 0xfa5656ac, 0x07f4f4f3, 0x25eaeacf);
        &_data_word(0xaf6565ca, 0x8e7a7af4, 0xe9aeae47, 0x18080810);
        &_data_word(0xd5baba6f, 0x887878f0, 0x6f25254a, 0x722e2e5c);
        &_data_word(0x241c1c38, 0xf1a6a657, 0xc7b4b473, 0x51c6c697);
        &_data_word(0x23e8e8cb, 0x7cdddda1, 0x9c7474e8, 0x211f1f3e);
        &_data_word(0xdd4b4b96, 0xdcbdbd61, 0x868b8b0d, 0x858a8a0f);
        &_data_word(0x907070e0, 0x423e3e7c, 0xc4b5b571, 0xaa6666cc);
        &_data_word(0xd8484890, 0x05030306, 0x01f6f6f7, 0x120e0e1c);
        &_data_word(0xa36161c2, 0x5f35356a, 0xf95757ae, 0xd0b9b969);
        &_data_word(0x91868617, 0x58c1c199, 0x271d1d3a, 0xb99e9e27);
```



```
        &_data_word(0x38e1e1d9, 0x13f8f8eb, 0xb398982b, 0x33111122);
        &_data_word(0xbb6969d2, 0x70d9d9a9, 0x898e8e07, 0xa7949433);
        &_data_word(0xb69b9b2d, 0x221e1e3c, 0x92878715, 0x20e9e9c9);
        &_data_word(0x49cece87, 0xff5555aa, 0x78282850, 0x7adfdfa5);
        &_data_word(0x8f8c8c03, 0xf8a1a159, 0x80898909, 0x170d0d1a);
        &_data_word(0xdabfbf65, 0x31e6e6d7, 0xc6424284, 0xb86868d0);
        &_data_word(0xc3414182, 0xb0999929, 0x772d2d5a, 0x110f0f1e);
        &_data_word(0xcbb0b07b, 0xfc5454a8, 0xd6bbbb6d, 0x3a16162c);

#Te4    # four copies of Te4 to choose from to avoid L1 aliasing
        &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
        &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
        &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
        &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
        &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
        &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
        &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
        &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
        &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
        &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
        &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
        &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
        &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
        &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
        &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
        &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
        &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
        &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
        &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
        &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
        &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
        &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
        &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
        &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
        &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
        &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
        &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
        &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
        &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
        &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
        &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
        &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);

        &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
        &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
        &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
        &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
        &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
        &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
        &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
        &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
        &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
        &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
        &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
        &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
        &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
        &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
        &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
        &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
        &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
        &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
        &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
        &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
        &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
        &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
        &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
```

```
2240         &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
2241         &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
2242         &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
2243         &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
2244         &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
2245         &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
2246         &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
2247         &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
2248         &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);

2250         &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
2251         &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
2252         &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
2253         &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
2254         &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
2255         &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
2256         &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
2257         &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
2258         &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
2259         &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
2260         &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
2261         &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
2262         &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
2263         &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
2264         &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
2265         &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
2266         &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
2267         &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
2268         &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
2269         &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
2270         &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
2271         &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
2272         &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
2273         &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
2274         &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
2275         &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
2276         &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
2277         &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
2278         &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
2279         &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
2280         &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
2281         &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);

2283         &data_byte(0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5);
2284         &data_byte(0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76);
2285         &data_byte(0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0);
2286         &data_byte(0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0);
2287         &data_byte(0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc);
2288         &data_byte(0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15);
2289         &data_byte(0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a);
2290         &data_byte(0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75);
2291         &data_byte(0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0);
2292         &data_byte(0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84);
2293         &data_byte(0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b);
2294         &data_byte(0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf);
2295         &data_byte(0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85);
2296         &data_byte(0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8);
2297         &data_byte(0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5);
2298         &data_byte(0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2);
2299         &data_byte(0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17);
2300         &data_byte(0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73);
2301         &data_byte(0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88);
2302         &data_byte(0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb);
2303         &data_byte(0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c);
2304         &data_byte(0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79);
2305         &data_byte(0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9);
```



```
2306         &data_byte(0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08);
2307         &data_byte(0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6);
2308         &data_byte(0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a);
2309         &data_byte(0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e);
2310         &data_byte(0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e);
2311         &data_byte(0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94);
2312         &data_byte(0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf);
2313         &data_byte(0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68);
2314         &data_byte(0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16);
2315 #rcon:
2316 $code.=<<___;
2317         .long   0x00000001, 0x00000002, 0x00000004, 0x00000008
2318         .long   0x00000010, 0x00000020, 0x00000040, 0x00000080
2319         .long   0x0000001b, 0x00000036, 0x80808080, 0x80808080
2320         .long   0xfefefefe, 0xfefefefe, 0x1b1b1b1b, 0x1b1b1b1b
2321 ___
2322 $code.=<<___;
2323 .align  64
2324 .LAES_Td:
2325 ___
2326         &_data_word(0x50a7f451, 0x5365417e, 0xc3a4171a, 0x965e273a);
2327         &_data_word(0xcb6bab3b, 0xf1459d1f, 0xab58faac, 0x9303e34b);
2328         &_data_word(0x55fa3020, 0xf66d76ad, 0x9176cc88, 0x254c02f5);
2329         &_data_word(0xfcd7e54f, 0xd7cb2ac5, 0x80443526, 0x8fa362b5);
2330         &_data_word(0x495ab1de, 0x671bba25, 0x980eea45, 0xe1c0fe5d);
2331         &_data_word(0x02752fc3, 0x12f04c81, 0xa397468d, 0xc6f9d36b);
2332         &_data_word(0xe75f8f03, 0x959c9215, 0xeb7a6dbf, 0xda595295);
2333         &_data_word(0x2d83bed4, 0xd3217458, 0x2969e049, 0x44c8c98e);
2334         &_data_word(0x6a89c275, 0x78798ef4, 0x6b3e5899, 0xdd71b927);
2335         &_data_word(0xb64fe1be, 0x17ad88f0, 0x66ac20c9, 0xb43ace7d);
2336         &_data_word(0x184adf63, 0x82311ae5, 0x60335197, 0x457f5362);
2337         &_data_word(0xe07764b1, 0x84ae6bbb, 0x1ca081fe, 0x942b08f9);
2338         &_data_word(0x58684870, 0x19fd458f, 0x876cde94, 0xb7f87b52);
2339         &_data_word(0x23d373ab, 0xe2024b72, 0x578f1fe3, 0x2aab5566);
2340         &_data_word(0x0728ebb2, 0x03c2b52f, 0x9a7bc586, 0xa50837d3);
2341         &_data_word(0xf2872830, 0xb2a5bf23, 0xba6a0302, 0x5c8216ed);
2342         &_data_word(0x2b1ccf8a, 0x92b479a7, 0xf0f207f3, 0xa1e2694e);
2343         &_data_word(0xcdf4da65, 0xd5be0506, 0x1f6234d1, 0x8afea6c4);
2344         &_data_word(0x9d532e34, 0xa055f3a2, 0x32e18a05, 0x75ebf6a4);
2345         &_data_word(0x39ec830b, 0xaaef6040, 0x069f715e, 0x51106ebd);
2346         &_data_word(0xf98a213e, 0x3d06dd96, 0xae053edd, 0x46bde64d);
2347         &_data_word(0xb58d5491, 0x055dc471, 0x6fd40604, 0xff155060);
2348         &_data_word(0x24fb9819, 0x97e9bdd6, 0xcc434089, 0x779ed967);
2349         &_data_word(0xbd42e8b0, 0x888b8907, 0x385b19e7, 0xdbeec879);
2350         &_data_word(0x470a7ca1, 0xe90f427c, 0xc91e84f8, 0x00000000);
2351         &_data_word(0x83868009, 0x48ed2b32, 0xac70111e, 0x4e725a6c);
2352         &_data_word(0xfbff0efd, 0x5638850f, 0x1ed5ae3d, 0x27392d36);
2353         &_data_word(0x64d90f0a, 0x21a65c68, 0xd1545b9b, 0x3a2e3624);
2354         &_data_word(0xb1670a0c, 0x0fe75793, 0xd296eeb4, 0x9e919b1b);
2355         &_data_word(0x4fc5c080, 0xa220dc61, 0x694b775a, 0x161a121c);
2356         &_data_word(0x0aba93e2, 0xe52aa0c0, 0x43e0223c, 0x1d171b12);
2357         &_data_word(0x0b0d090e, 0xadc78bf2, 0xb9a8b62d, 0xc8a91e14);
2358         &_data_word(0x8519f157, 0x4c0775af, 0xbbdd99ee, 0xfd607fa3);
2359         &_data_word(0x9f2601f7, 0xbcf5725c, 0xc53b6644, 0x347efb5b);
2360         &_data_word(0x7629438b, 0xdcc623cb, 0x68fcedb6, 0x63f1e4b8);
2361         &_data_word(0xcadc31d7, 0x10856342, 0x40229713, 0x2011c684);
2362         &_data_word(0x7d244a85, 0xf83dbbd2, 0x1132f9ae, 0x6da129c7);
2363         &_data_word(0x4b2f9e1d, 0xf330b2dc, 0xec52860d, 0xd0e3c177);
2364         &_data_word(0x6c16b32b, 0x99b970a9, 0xfa489411, 0x2264e947);
2365         &_data_word(0xc48cfca8, 0x1a3ff0a0, 0xd82c7d56, 0xef903322);
2366         &_data_word(0xc74e4987, 0xc1d138d9, 0xfea2ca8c, 0x360bd498);
2367         &_data_word(0xcf81f5a6, 0x28de7aa5, 0x268eb7da, 0xa4bfad3f);
2368         &_data_word(0xe49d3a2c, 0x0d927850, 0x9bcc5f6a, 0x62467e54);
2369         &_data_word(0xc2138df6, 0xe8b8d890, 0x5ef7392e, 0xf5afc382);
2370         &_data_word(0xbe805d9f, 0x7c93d069, 0xa92dd56f, 0xb31225cf);
2371         &_data_word(0x3b99acc8, 0xa77d1810, 0x6e639ce8, 0x7bbb3bdb);
```

```
2372         &_data_word(0x097826cd, 0xf418596e, 0x01b79aec, 0xa89a4f83);
2373         &_data_word(0x656e95e6, 0x7ee6ffaa, 0x08cfbc21, 0xe6e815ef);
2374         &_data_word(0xd99be7ba, 0xce366f4a, 0xd4099fea, 0xd67cb029);
2375         &_data_word(0xafb2a431, 0x31233f2a, 0x3094a5c6, 0xc066a235);
2376         &_data_word(0x37bc4e74, 0xa6ca82fc, 0xb0d090e0, 0x15d8a733);
2377         &_data_word(0x4a9804f1, 0xf7daec41, 0x0e50cd7f, 0x2ff69117);
2378         &_data_word(0x8dd64d76, 0x4db0ef43, 0x544daacc, 0xdf0496e4);
2379         &_data_word(0xe3b5d19e, 0x1b886a4c, 0xb81f2cc1, 0x7f516546);
2380         &_data_word(0x04ea5e9d, 0x5d358c01, 0x737487fa, 0x2e410bfb);
2381         &_data_word(0x5a1d67b3, 0x52d2db92, 0x335610e9, 0x1347d66d);
2382         &_data_word(0x8c61d79a, 0x7a0ca137, 0x8e14f859, 0x893c13eb);
2383         &_data_word(0xee27a9ce, 0x35c961b7, 0xede51ce1, 0x3cb1477a);
2384         &_data_word(0x59dfd29c, 0x3f73f255, 0x79ce1418, 0xbf37c773);
2385         &_data_word(0xeacdf753, 0x5baafd5f, 0x146f3ddf, 0x86db4478);
2386         &_data_word(0x81f3afca, 0x3ec468b9, 0x2c342438, 0x5f40a3c2);
2387         &_data_word(0x72c31d16, 0x0c25e2bc, 0x8b493c28, 0x41950dff);
2388         &_data_word(0x7101a839, 0xdeb30c08, 0x9ce4b4d8, 0x90c15664);
2389         &_data_word(0x6184cb7b, 0x70b632d5, 0x745c6c48, 0x4257b8d0);

2391 #Td4:   # four copies of Td4 to choose from to avoid L1 aliasing
2392         &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
2393         &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
2394         &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
2395         &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
2396         &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
2397         &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
2398         &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
2399         &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
2400         &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
2401         &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
2402         &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
2403         &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
2404         &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
2405         &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
2406         &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
2407         &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
2408         &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
2409         &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
2410         &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
2411         &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
2412         &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
2413         &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
2414         &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
2415         &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
2416         &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
2417         &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
2418         &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
2419         &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
2420         &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
2421         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
2422         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
2423         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);
2424 $code.=<<___;
2425         .long   0x80808080, 0x80808080, 0xfefefefe, 0xfefefefe
2426         .long   0x1b1b1b1b, 0x1b1b1b1b, 0, 0
2427 ___
2428         &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
2429         &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
2430         &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
2431         &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
2432         &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
2433         &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
2434         &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
2435         &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
2436         &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
2437         &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
```



```
2438         &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
2439         &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
2440         &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
2441         &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
2442         &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
2443         &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
2444         &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
2445         &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
2446         &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
2447         &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
2448         &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
2449         &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
2450         &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
2451         &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
2452         &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
2453         &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
2454         &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
2455         &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
2456         &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
2457         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
2458         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
2459         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);
2460 $code.=<<___;
2461         .long   0x80808080, 0x80808080, 0xfefefefe, 0xfefefefe
2462         .long   0x1b1b1b1b, 0x1b1b1b1b, 0, 0
2463 ___
2464         &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
2465         &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
2466         &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
2467         &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
2468         &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
2469         &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
2470         &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
2471         &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
2472         &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
2473         &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
2474         &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
2475         &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
2476         &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
2477         &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
2478         &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
2479         &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
2480         &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
2481         &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
2482         &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
2483         &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
2484         &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
2485         &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
2486         &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
2487         &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
2488         &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
2489         &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
2490         &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
2491         &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
2492         &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
2493         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
2494         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
2495         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);
2496 $code.=<<___;
2497         .long   0x80808080, 0x80808080, 0xfefefefe, 0xfefefefe
2498         .long   0x1b1b1b1b, 0x1b1b1b1b, 0, 0
2499 ___
2500         &data_byte(0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38);
2501         &data_byte(0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb);
2502         &data_byte(0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87);
2503         &data_byte(0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb);
```

```
2504         &data_byte(0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d);
2505         &data_byte(0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e);
2506         &data_byte(0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2);
2507         &data_byte(0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25);
2508         &data_byte(0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16);
2509         &data_byte(0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92);
2510         &data_byte(0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda);
2511         &data_byte(0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84);
2512         &data_byte(0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a);
2513         &data_byte(0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06);
2514         &data_byte(0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02);
2515         &data_byte(0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b);
2516         &data_byte(0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea);
2517         &data_byte(0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73);
2518         &data_byte(0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85);
2519         &data_byte(0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e);
2520         &data_byte(0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89);
2521         &data_byte(0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b);
2522         &data_byte(0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20);
2523         &data_byte(0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4);
2524         &data_byte(0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31);
2525         &data_byte(0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f);
2526         &data_byte(0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d);
2527         &data_byte(0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef);
2528         &data_byte(0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0);
2529         &data_byte(0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61);
2530         &data_byte(0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26);
2531         &data_byte(0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d);
2532 $code.=<<___;
2533         .long   0x80808080, 0x80808080, 0xfefefefe, 0xfefefefe
2534         .long   0x1b1b1b1b, 0x1b1b1b1b, 0, 0
2535 .asciz  "AES for x86_64, CRYPTOGAMS by <appro\@openssl.org>"
2536 .align  64
2537 ___

2539 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
2540 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
2541 if ($win64) {
2542 $rec="%rcx";
2543 $frame="%rdx";
2544 $context="%r8";
2545 $disp="%r9";

2547 $code.=<<___;
2548 .extern __imp_RtlVirtualUnwind
2549 .type   block_se_handler,\@abi-omnipotent
2550 .align  16
2551 block_se_handler:
2552         push    %rsi
2553         push    %rdi
2554         push    %rbx
2555         push    %rbp
2556         push    %r12
2557         push    %r13
2558         push    %r14
2559         push    %r15
2560         pushfq
2561         sub     \$64,%rsp

2563         mov     120($context),%rax      # pull context->Rax
2564         mov     248($context),%rbx      # pull context->Rip

2566         mov     8($disp),%rsi           # disp->ImageBase
2567         mov     56($disp),%r11          # disp->HandlerData

2569         mov     0(%r11),%r10d           # HandlerData[0]
```



```
2570         lea     (%rsi,%r10),%r10        # prologue label
2571         cmp     %r10,%rbx               # context->Rip<prologue label
2572         jb      .Lin_block_prologue

2574         mov     152($context),%rax      # pull context->Rsp

2576         mov     4(%r11),%r10d           # HandlerData[1]
2577         lea     (%rsi,%r10),%r10        # epilogue label
2578         cmp     %r10,%rbx               # context->Rip>=epilogue label
2579         jae     .Lin_block_prologue

2581         mov     24(%rax),%rax           # pull saved real stack pointer
2582         lea     48(%rax),%rax           # adjust...

2584         mov     -8(%rax),%rbx
2585         mov     -16(%rax),%rbp
2586         mov     -24(%rax),%r12
2587         mov     -32(%rax),%r13
2588         mov     -40(%rax),%r14
2589         mov     -48(%rax),%r15
2590         mov     %rbx,144($context)      # restore context->Rbx
2591         mov     %rbp,160($context)      # restore context->Rbp
2592         mov     %r12,216($context)      # restore context->R12
2593         mov     %r13,224($context)      # restore context->R13
2594         mov     %r14,232($context)      # restore context->R14
2595         mov     %r15,240($context)      # restore context->R15

2597 .Lin_block_prologue:
2598         mov     8(%rax),%rdi
2599         mov     16(%rax),%rsi
2600         mov     %rax,152($context)      # restore context->Rsp
2601         mov     %rsi,168($context)      # restore context->Rsi
2602         mov     %rdi,176($context)      # restore context->Rdi

2604         jmp     .Lcommon_seh_exit
2605 .size   block_se_handler,.-block_se_handler

2607 .type   key_se_handler,\@abi-omnipotent
2608 .align  16
2609 key_se_handler:
2610         push    %rsi
2611         push    %rdi
2612         push    %rbx
2613         push    %rbp
2614         push    %r12
2615         push    %r13
2616         push    %r14
2617         push    %r15
2618         pushfq
2619         sub     \$64,%rsp

2621         mov     120($context),%rax      # pull context->Rax
2622         mov     248($context),%rbx      # pull context->Rip

2624         mov     8($disp),%rsi           # disp->ImageBase
2625         mov     56($disp),%r11          # disp->HandlerData

2627         mov     0(%r11),%r10d           # HandlerData[0]
2628         lea     (%rsi,%r10),%r10        # prologue label
2629         cmp     %r10,%rbx               # context->Rip<prologue label
2630         jb      .Lin_key_prologue

2632         mov     152($context),%rax      # pull context->Rsp

2634         mov     4(%r11),%r10d           # HandlerData[1]
2635         lea     (%rsi,%r10),%r10        # epilogue label
```

```
2636         cmp     %r10,%rbx               # context->Rip>=epilogue label
2637         jae     .Lin_key_prologue

2639         lea     56(%rax),%rax

2641         mov     -8(%rax),%rbx
2642         mov     -16(%rax),%rbp
2643         mov     -24(%rax),%r12
2644         mov     -32(%rax),%r13
2645         mov     -40(%rax),%r14
2646         mov     -48(%rax),%r15
2647         mov     %rbx,144($context)      # restore context->Rbx
2648         mov     %rbp,160($context)      # restore context->Rbp
2649         mov     %r12,216($context)      # restore context->R12
2650         mov     %r13,224($context)      # restore context->R13
2651         mov     %r14,232($context)      # restore context->R14
2652         mov     %r15,240($context)      # restore context->R15

2654 .Lin_key_prologue:
2655         mov     8(%rax),%rdi
2656         mov     16(%rax),%rsi
2657         mov     %rax,152($context)      # restore context->Rsp
2658         mov     %rsi,168($context)      # restore context->Rsi
2659         mov     %rdi,176($context)      # restore context->Rdi

2661         jmp     .Lcommon_seh_exit
2662 .size   key_se_handler,.-key_se_handler

2664 .type   cbc_se_handler,\@abi-omnipotent
2665 .align  16
2666 cbc_se_handler:
2667         push    %rsi
2668         push    %rdi
2669         push    %rbx
2670         push    %rbp
2671         push    %r12
2672         push    %r13
2673         push    %r14
2674         push    %r15
2675         pushfq
2676         sub     \$64,%rsp

2678         mov     120($context),%rax      # pull context->Rax
2679         mov     248($context),%rbx      # pull context->Rip

2681         lea     .Lcbc_prologue(%rip),%r10
2682         cmp     %r10,%rbx               # context->Rip<.Lcbc_prologue
2683         jb      .Lin_cbc_prologue

2685         lea     .Lcbc_fast_body(%rip),%r10
2686         cmp     %r10,%rbx               # context->Rip<.Lcbc_fast_body
2687         jb      .Lin_cbc_frame_setup

2689         lea     .Lcbc_slow_prologue(%rip),%r10
2690         cmp     %r10,%rbx               # context->Rip<.Lcbc_slow_prologue
2691         jb      .Lin_cbc_body

2693         lea     .Lcbc_slow_body(%rip),%r10
2694         cmp     %r10,%rbx               # context->Rip<.Lcbc_slow_body
2695         jb      .Lin_cbc_frame_setup

2697 .Lin_cbc_body:
2698         mov     152($context),%rax      # pull context->Rsp

2700         lea     .Lcbc_epilogue(%rip),%r10
2701         cmp     %r10,%rbx               # context->Rip>=.Lcbc_epilogue
```



```
2702         jae     .Lin_cbc_prologue

2704         lea     8(%rax),%rax

2706         lea     .Lcbc_popfq(%rip),%r10
2707         cmp     %r10,%rbx               # context->Rip>=.Lcbc_popfq
2708         jae     .Lin_cbc_prologue

2710         mov     '16-8'(%rax),%rax       # biased $_rsp
2711         lea     56(%rax),%rax

2713 .Lin_cbc_frame_setup:
2714         mov     -16(%rax),%rbx
2715         mov     -24(%rax),%rbp
2716         mov     -32(%rax),%r12
2717         mov     -40(%rax),%r13
2718         mov     -48(%rax),%r14
2719         mov     -56(%rax),%r15
2720         mov     %rbx,144($context)      # restore context->Rbx
2721         mov     %rbp,160($context)      # restore context->Rbp
2722         mov     %r12,216($context)      # restore context->R12
2723         mov     %r13,224($context)      # restore context->R13
2724         mov     %r14,232($context)      # restore context->R14
2725         mov     %r15,240($context)      # restore context->R15

2727 .Lin_cbc_prologue:
2728         mov     8(%rax),%rdi
2729         mov     16(%rax),%rsi
2730         mov     %rax,152($context)      # restore context->Rsp
2731         mov     %rsi,168($context)      # restore context->Rsi
2732         mov     %rdi,176($context)      # restore context->Rdi

2734 .Lcommon_seh_exit:

2736         mov     40($disp),%rdi          # disp->ContextRecord
2737         mov     $context,%rsi           # context
2738         mov     \$'1232/8',%ecx         # sizeof(CONTEXT)
2739         .long   0xa548f3fc              # cld; rep movsq

2741         mov     $disp,%rsi
2742         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
2743         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
2744         mov     0(%rsi),%r8             # arg3, disp->ControlPc
2745         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
2746         mov     40(%rsi),%r10           # disp->ContextRecord
2747         lea     56(%rsi),%r11           # &disp->HandlerData
2748         lea     24(%rsi),%r12           # &disp->EstablisherFrame
2749         mov     %r10,32(%rsp)           # arg5
2750         mov     %r11,40(%rsp)           # arg6
2751         mov     %r12,48(%rsp)           # arg7
2752         mov     %rcx,56(%rsp)           # arg8, (NULL)
2753         call    *__imp_RtlVirtualUnwind(%rip)

2755         mov     \$1,%eax                # ExceptionContinueSearch
2756         add     \$64,%rsp
2757         popfq
2758         pop     %r15
2759         pop     %r14
2760         pop     %r13
2761         pop     %r12
2762         pop     %rbp
2763         pop     %rbx
2764         pop     %rdi
2765         pop     %rsi
2766         ret
2767 .size   cbc_se_handler,.-cbc_se_handler
```

```
.section        .pdata
.align  4
        .rva    .LSEH_begin_AES_encrypt
        .rva    .LSEH_end_AES_encrypt
        .rva    .LSEH_info_AES_encrypt

        .rva    .LSEH_begin_AES_decrypt
        .rva    .LSEH_end_AES_decrypt
        .rva    .LSEH_info_AES_decrypt

        .rva    .LSEH_begin_private_AES_set_encrypt_key
        .rva    .LSEH_end_private_AES_set_encrypt_key
        .rva    .LSEH_info_private_AES_set_encrypt_key

        .rva    .LSEH_begin_private_AES_set_decrypt_key
        .rva    .LSEH_end_private_AES_set_decrypt_key
        .rva    .LSEH_info_private_AES_set_decrypt_key

        .rva    .LSEH_begin_AES_cbc_encrypt
        .rva    .LSEH_end_AES_cbc_encrypt
        .rva    .LSEH_info_AES_cbc_encrypt

.section        .xdata
.align  8
.LSEH_info_AES_encrypt:
        .byte   9,0,0,0
        .rva    block_se_handler
        .rva    .Lenc_prologue,.Lenc_epilogue   # HandlerData[]
.LSEH_info_AES_decrypt:
        .byte   9,0,0,0
        .rva    block_se_handler
        .rva    .Ldec_prologue,.Ldec_epilogue   # HandlerData[]
.LSEH_info_private_AES_set_encrypt_key:
        .byte   9,0,0,0
        .rva    key_se_handler
        .rva    .Lenc_key_prologue,.Lenc_key_epilogue   # HandlerData[]
.LSEH_info_private_AES_set_decrypt_key:
        .byte   9,0,0,0
        .rva    key_se_handler
        .rva    .Ldec_key_prologue,.Ldec_key_epilogue   # HandlerData[]
.LSEH_info_AES_cbc_encrypt:
        .byte   9,0,0,0
        .rva    cbc_se_handler
___
}

$code =~ s/\`([^\`]*)\`/eval($1)/gem;

print $code;

close STDOUT;
#endif /* ! codereview */
```

```
**********************************************************
   31703 Wed Aug 13 19:53:05 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/aesni-sha1-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# June 2011
#
# This is AESNI-CBC+SHA1 "stitch" implementation. The idea, as spelled
# in http://download.intel.com/design/intarch/papers/323686.pdf, is
# that since AESNI-CBC encrypt exhibit *very* low instruction-level
# parallelism, interleaving it with another algorithm would allow to
# utilize processor resources better and achieve better performance.
# SHA1 instruction sequences(*) are taken from sha1-x86_64.pl and
# AESNI code is weaved into it. Below are performance numbers in
# cycles per processed byte, less is better, for standalone AESNI-CBC
# encrypt, sum of the latter and standalone SHA1, and "stitched"
# subroutine:
#
#               AES-128-CBC     +SHA1           stitch      gain
# Westmere      3.77[+5.6]      9.37            6.65        +41%
# Sandy Bridge  5.05[+5.2(6.3)] 10.25(11.35)    6.16(7.08)  +67%(+60%)
#
#               AES-192-CBC
# Westmere      4.51            10.11           6.97        +45%
# Sandy Bridge  6.05            11.25(12.35)    6.34(7.27)  +77%(+70%)
#
#               AES-256-CBC
# Westmere      5.25            10.85           7.25        +50%
# Sandy Bridge  7.05            12.25(13.35)    7.06(7.70)  +74%(+73%)
#
# (*)   There are two code paths: SSSE3 and AVX. See sha1-568.pl for
#       background information. Above numbers in parentheses are SSSE3
#       results collected on AVX-capable CPU, i.e. apply on OSes that
#       don't support AVX.
#
# Needless to mention that it makes no sense to implement "stitched"
# *decrypt* subroutine. Because *both* AESNI-CBC decrypt and SHA1
# fully utilize parallelism, so stitching would not give any gain
# anyway. Well, there might be some, e.g. because of better cache
# locality... For reference, here are performance results for
# standalone AESNI-CBC decrypt:
#
#               AES-128-CBC     AES-192-CBC     AES-256-CBC
# Westmere      1.31            1.55            1.80
# Sandy Bridge  0.93            1.06            1.22

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";
```



```
$avx=1 if ('$ENV{CC} -Wa,-v -c -o /dev/null -x assembler /dev/null 2>&1'
               =~ /GNU assembler version ([2-9]\.[0-9]+)/ &&
           $1>=2.19);
$avx=1 if (!$avx && $win64 && ($flavour =~ /nasm/ || $ENV{ASM} =~ /nasm/) &&
           `nasm -v 2>&1` =~ /NASM version ([2-9]\.[0-9]+)/ &&
           $1>=2.09);
$avx=1 if (!$avx && $win64 && ($flavour =~ /masm/ || $ENV{ASM} =~ /ml64/) &&
           `ml64 2>&1` =~ /Version ([0-9]+)\./ &&
           $1>=10);

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

# void aesni_cbc_sha1_enc(const void *inp,
#                         void *out,
#                         size_t length,
#                         const AES_KEY *key,
#                         unsigned char *iv,
#                         SHA_CTX *ctx,
#                         const void *in0);

$code.=<<___;
.text
.extern OPENSSL_ia32cap_P

.globl  aesni_cbc_sha1_enc
.type   aesni_cbc_sha1_enc,\@abi-omnipotent
.align  16
aesni_cbc_sha1_enc:
        # caller should check for SSSE3 and AES-NI bits
        mov     OPENSSL_ia32cap_P+0(%rip),%r10d
        mov     OPENSSL_ia32cap_P+4(%rip),%r11d
___
$code.=<<___ if ($avx);
        and     \$`1<<28`,%r11d         # mask AVX bit
        and     \$`1<<30`,%r10d         # mask "Intel CPU" bit
        or      %r11d,%r10d
        cmp     \$`1<<28|1<<30`,%r10d
        je      aesni_cbc_sha1_enc_avx
___
$code.=<<___;
        jmp     aesni_cbc_sha1_enc_ssse3
        ret
.size   aesni_cbc_sha1_enc,.-aesni_cbc_sha1_enc
___

my ($in0,$out,$len,$key,$ivp,$ctx,$inp)=("%rdi","%rsi","%rdx","%rcx","%r8","%r9"

my $Xi=4;
my @X=map("%xmm$_",(4..7,0..3));
my @Tx=map("%xmm$_",(8..10));
my @V=($A,$B,$C,$D,$E)=("%eax","%ebx","%ecx","%edx","%ebp");    # size optimizat
my @T=("%esi","%edi");
my $j=0; my $jj=0; my $r=0; my $sn=0;
my $K_XX_XX="%r11";
my ($iv,$in,$rndkey0)=map("%xmm$_",(11..13));
my @rndkey=("%xmm14","%xmm15");

sub AUTOLOAD()          # thunk [simplified] 32-bit style perlasm
{ my $opcode = $AUTOLOAD; $opcode =~ s/.*::/;
  my $arg = pop;
    $arg = "\$$arg" if ($arg*1 eq $arg);
    $code .= "\t$opcode\t".join(',',$arg,reverse @_)."\n";
}

my $_rol=sub { &rol(@_) };
```

```
128 my $_ror=sub { &ror(@_) };
129
130 $code.=<<___;
131 .type   aesni_cbc_sha1_enc_ssse3,\@function,6
132 .align  16
133 aesni_cbc_sha1_enc_ssse3:
134         mov     `($win64?56:8)`(%rsp),$inp      # load 7th argument
135         #shr    \$6,$len                        # debugging artefact
136         #jz     .Lepilogue_ssse3                # debugging artefact
137         push    %rbx
138         push    %rbp
139         push    %r12
140         push    %r13
141         push    %r14
142         push    %r15
143         lea     `-104-($win64?10*16:0)`(%rsp),%rsp
144         #mov    $in0,$inp                       # debugging artefact
145         #lea    64(%rsp),$ctx                   # debugging artefact
146 ___
147 $code.=<<___ if ($win64);
148         movaps  %xmm6,96+0(%rsp)
149         movaps  %xmm7,96+16(%rsp)
150         movaps  %xmm8,96+32(%rsp)
151         movaps  %xmm9,96+48(%rsp)
152         movaps  %xmm10,96+64(%rsp)
153         movaps  %xmm11,96+80(%rsp)
154         movaps  %xmm12,96+96(%rsp)
155         movaps  %xmm13,96+112(%rsp)
156         movaps  %xmm14,96+128(%rsp)
157         movaps  %xmm15,96+144(%rsp)
158 .Lprologue_ssse3:
159 ___
160 $code.=<<___;
161         mov     $in0,%r12                       # reassign arguments
162         mov     $out,%r13
163         mov     $len,%r14
164         mov     $key,%r15
165         movdqu  ($ivp),$iv                      # load IV
166         mov     $ivp,88(%rsp)                   # save $ivp
167 ___
168 my ($in0,$out,$len,$key)=map("%r$_",(12..15));  # reassign arguments
169 my $rounds="${ivp}d";
170 $code.=<<___;
171         shl     \$6,$len
172         sub     $in0,$out
173         mov     240($key),$rounds
174         add     $inp,$len               # end of input
175
176         lea     K_XX_XX(%rip),$K_XX_XX
177         mov     0($ctx),$A              # load context
178         mov     4($ctx),$B
179         mov     8($ctx),$C
180         mov     12($ctx),$D
181         mov     $B,@T[0]                # magic seed
182         mov     16($ctx),$E
183
184         movdqa  64($K_XX_XX),@X[2]      # pbswap mask
185         movdqa  0($K_XX_XX),@Tx[1]      # K_00_19
186         movdqu  0($inp),@X[-4&7]        # load input to %xmm[0-3]
187         movdqu  16($inp),@X[-3&7]
188         movdqu  32($inp),@X[-2&7]
189         movdqu  48($inp),@X[-1&7]
190         pshufb  @X[2],@X[-4&7]          # byte swap
191         add     \$64,$inp
192         pshufb  @X[2],@X[-3&7]
193         pshufb  @X[2],@X[-2&7]
```



```
194         pshufb  @X[2],@X[-1&7]
195         paddd   @Tx[1],@X[-4&7]         # add K_00_19
196         paddd   @Tx[1],@X[-3&7]
197         paddd   @Tx[1],@X[-2&7]
198         movdqa  @X[-4&7],0(%rsp)        # X[]+K xfer to IALU
199         psubd   @Tx[1],@X[-4&7]         # restore X[]
200         movdqa  @X[-3&7],16(%rsp)
201         psubd   @Tx[1],@X[-3&7]
202         movdqa  @X[-2&7],32(%rsp)
203         psubd   @Tx[1],@X[-2&7]
204         movups  ($key),$rndkey0         # $key[0]
205         movups  16($key),$rndkey[0]     # forward reference
206         jmp     .Loop_ssse3
207 ___
208
209 my $aesenc=sub {
210   use integer;
211   my ($n,$k)=($r/10,$r%10);
212     if ($k==0) {
213       $code.=<<___;
214         movups          `16*$n`($in0),$in               # load input
215         xorps           $rndkey0,$in
216 ___
217       $code.=<<___ if ($n);
218         movups          $iv,`16*($n-1)`($out,$in0)      # write output
219 ___
220       $code.=<<___;
221         xorps           $in,$iv
222         aesenc          $rndkey[0],$iv
223         movups          `32+16*$k`($key),$rndkey[1]
224 ___
225     } elsif ($k==9) {
226       $sn++;
227       $code.=<<___;
228         cmp             \$11,$rounds
229         jb              .Laesenclast$sn
230         movups          `32+16*($k+0)`($key),$rndkey[1]
231         aesenc          $rndkey[0],$iv
232         movups          `32+16*($k+1)`($key),$rndkey[0]
233         aesenc          $rndkey[1],$iv
234         je              .Laesenclast$sn
235         movups          `32+16*($k+2)`($key),$rndkey[1]
236         aesenc          $rndkey[0],$iv
237         movups          `32+16*($k+3)`($key),$rndkey[0]
238         aesenc          $rndkey[1],$iv
239 .Laesenclast$sn:
240         aesenclast      $rndkey[0],$iv
241         movups          16($key),$rndkey[1]             # forward reference
242 ___
243     } else {
244       $code.=<<___;
245         aesenc          $rndkey[0],$iv
246         movups          `32+16*$k`($key),$rndkey[1]
247 ___
248     }
249     $r++;       unshift(@rndkey,pop(@rndkey));
250 };
251
252 sub Xupdate_ssse3_16_31()               # recall that $Xi starts wtih 4
253 { use integer;
254   my $body = shift;
255   my @insns = (&$body,&$body,&$body,&$body);    # 40 instructions
256   my ($a,$b,$c,$d,$e);
257
258         &movdqa (@X[0],@X[-3&7]);
259          eval(shift(@insns));
```

```
260          eval(shift(@insns));
261         &movdqa (@Tx[0],@X[-1&7]);
262         &palignr(@X[0],@X[-4&7],8);     # compose "X[-14]" in "X[0]"
263          eval(shift(@insns));
264          eval(shift(@insns));
265
266           &paddd        (@Tx[1],@X[-1&7]);
267          eval(shift(@insns));
268          eval(shift(@insns));
269         &psrldq (@Tx[0],4);             # "X[-3]", 3 dwords
270          eval(shift(@insns));
271          eval(shift(@insns));
272         &pxor   (@X[0],@X[-4&7]);       # "X[0]"^="X[-16]"
273          eval(shift(@insns));
274          eval(shift(@insns));
275
276         &pxor   (@Tx[0],@X[-2&7]);      # "X[-3]"^"X[-8]"
277          eval(shift(@insns));
278          eval(shift(@insns));
279          eval(shift(@insns));
280          eval(shift(@insns));
281
282         &pxor   (@X[0],@Tx[0]);         # "X[0]"^="X[-3]"^"X[-8]"
283          eval(shift(@insns));
284          eval(shift(@insns));
285           &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
286          eval(shift(@insns));
287          eval(shift(@insns));
288
289         &movdqa (@Tx[2],@X[0]);
290         &movdqa (@Tx[0],@X[0]);
291          eval(shift(@insns));
292          eval(shift(@insns));
293          eval(shift(@insns));
294          eval(shift(@insns));
295
296         &pslldq (@Tx[2],12);            # "X[0]"<<96, extract one dword
297         &paddd  (@X[0],@X[0]);
298          eval(shift(@insns));
299          eval(shift(@insns));
300          eval(shift(@insns));
301          eval(shift(@insns));
302
303         &psrld  (@Tx[0],31);
304          eval(shift(@insns));
305          eval(shift(@insns));
306         &movdqa (@Tx[1],@Tx[2]);
307          eval(shift(@insns));
308          eval(shift(@insns));
309
310         &psrld  (@Tx[2],30);
311         &por    (@X[0],@Tx[0]);         # "X[0]"<<<<1
312          eval(shift(@insns));
313          eval(shift(@insns));
314          eval(shift(@insns));
315          eval(shift(@insns));
316
317         &pslld  (@Tx[1],2);
318         &pxor   (@X[0],@Tx[2]);
319          eval(shift(@insns));
320          eval(shift(@insns));
321           &movdqa       (@Tx[2],eval(16*(($Xi)/5))."($K_XX_XX)");       # K_XX_X
322          eval(shift(@insns));
323          eval(shift(@insns));
324
325         &pxor   (@X[0],@Tx[1]);         # "X[0]"^=("X[0]">>96)<<<2
```



```
326
327          foreach (@insns) { eval; }     # remaining instructions [if any]
328
329   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
330                 push(@Tx,shift(@Tx));
331 }
332
333 sub Xupdate_ssse3_32_79()
334 { use integer;
335   my $body = shift;
336   my @insns = (&$body,&$body,&$body,&$body);    # 32 to 48 instructions
337   my ($a,$b,$c,$d,$e);
338
339         &movdqa (@Tx[0],@X[-1&7])       if ($Xi==8);
340          eval(shift(@insns));           # body_20_39
341         &pxor   (@X[0],@X[-4&7]);       # "X[0]"="X[-32]"^"X[-16]"
342         &palignr(@Tx[0],@X[-2&7],8);    # compose "X[-6]"
343          eval(shift(@insns));
344          eval(shift(@insns));
345          eval(shift(@insns));           # rol
346
347         &pxor   (@X[0],@X[-7&7]);       # "X[0]"^="X[-28]"
348          eval(shift(@insns));
349          eval(shift(@insns))    if (@insns[0] !~ /&ro[rl]/);
350         if ($Xi%5) {
351           &movdqa       (@Tx[2],@Tx[1]);# "perpetuate" K_XX_XX...
352         } else {                        # ... or load next one
353           &movdqa       (@Tx[2],eval(16*($Xi/5))."($K_XX_XX)");
354         }
355           &paddd        (@Tx[1],@X[-1&7]);
356          eval(shift(@insns));           # ror
357          eval(shift(@insns));
358
359         &pxor   (@X[0],@Tx[0]);         # "X[0]"^="X[-6]"
360          eval(shift(@insns));           # body_20_39
361          eval(shift(@insns));
362          eval(shift(@insns));
363          eval(shift(@insns));           # rol
364
365         &movdqa (@Tx[0],@X[0]);
366           &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
367          eval(shift(@insns));
368          eval(shift(@insns));
369          eval(shift(@insns));           # ror
370          eval(shift(@insns));
371
372         &pslld  (@X[0],2);
373          eval(shift(@insns));           # body_20_39
374          eval(shift(@insns));
375         &psrld  (@Tx[0],30);
376          eval(shift(@insns));
377          eval(shift(@insns));           # rol
378          eval(shift(@insns));
379          eval(shift(@insns));
380          eval(shift(@insns));           # ror
381          eval(shift(@insns));
382
383         &por    (@X[0],@Tx[0]);         # "X[0]"<<<<2
384          eval(shift(@insns));           # body_20_39
385          eval(shift(@insns));
386           &movdqa       (@Tx[1],@X[0])  if ($Xi<19);
387          eval(shift(@insns));
388          eval(shift(@insns));           # rol
389          eval(shift(@insns));
390          eval(shift(@insns));
391          eval(shift(@insns));           # rol
```

```
392         eval(shift(@insns));
393
394         foreach (@insns) { eval; }     # remaining instructions
395
396   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
397                 push(@Tx,shift(@Tx));
398 }
399
400 sub Xuplast_ssse3_80()
401 { use integer;
402   my $body = shift;
403   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
404   my ($a,$b,$c,$d,$e);
405
406         eval(shift(@insns));
407          &paddd        (@Tx[1],@X[-1&7]);
408         eval(shift(@insns));
409         eval(shift(@insns));
410         eval(shift(@insns));
411         eval(shift(@insns));
412
413          &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer IAL
414
415         foreach (@insns) { eval; }             # remaining instructions
416
417        &cmp    ($inp,$len);
418        &je     (".Ldone_ssse3");
419
420        unshift(@Tx,pop(@Tx));
421
422        &movdqa (@X[2],"64($K_XX_XX)");         # pbswap mask
423        &movdqa (@Tx[1],"0($K_XX_XX)");         # K_00_19
424        &movdqu (@X[-4&7],"0($inp)");           # load input
425        &movdqu (@X[-3&7],"16($inp)");
426        &movdqu (@X[-2&7],"32($inp)");
427        &movdqu (@X[-1&7],"48($inp)");
428        &pshufb (@X[-4&7],@X[2]);               # byte swap
429        &add    ($inp,64);
430
431   $Xi=0;
432 }
433
434 sub Xloop_ssse3()
435 { use integer;
436   my $body = shift;
437   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
438   my ($a,$b,$c,$d,$e);
439
440         eval(shift(@insns));
441         eval(shift(@insns));
442        &pshufb (@X[($Xi-3)&7],@X[2]);
443         eval(shift(@insns));
444         eval(shift(@insns));
445        &paddd  (@X[($Xi-4)&7],@Tx[1]);
446         eval(shift(@insns));
447         eval(shift(@insns));
448         eval(shift(@insns));
449         eval(shift(@insns));
450        &movdqa (eval(16*$Xi)."(%rsp)",@X[($Xi-4)&7]);  # X[]+K xfer to IALU
451         eval(shift(@insns));
452         eval(shift(@insns));
453        &psubd  (@X[($Xi-4)&7],@Tx[1]);
454
455         foreach (@insns) { eval; }
456   $Xi++;
457 }
```



```
459 sub Xtail_ssse3()
460 { use integer;
461   my $body = shift;
462   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
463   my ($a,$b,$c,$d,$e);
464
465         foreach (@insns) { eval; }
466 }
467
468 sub body_00_19 () {
469   use integer;
470   my ($k,$n);
471   my @r=(
472         '($a,$b,$c,$d,$e)=@V;'.
473         '&add   ($e,eval(4*($j&15))."(%rsp)");',        # X[]+K xfer
474         '&xor   ($c,$d);',
475         '&mov   (@T[1],$a);',   # $b in next round
476         '&$_rol ($a,5);',
477         '&and   (@T[0],$c);',   # ($b&($c^$d))
478         '&xor   ($c,$d);',      # restore $c
479         '&xor   (@T[0],$d);',
480         '&add   ($e,$a);',
481         '&$_ror ($b,$j?7:2);',  # $b>>>2
482         '&add   ($e,@T[0]);'    .'$j++; unshift(@V,pop(@V)); unshift(@T,pop(@T))
483         );
484         $n = scalar(@r);
485         $k = (($jj+1)*12/20)*20*$n/12;  # 12 aesencs per these 20 rounds
486         @r[$k%$n].='&$aesenc();'        if ($jj==$k/$n);
487         $jj++;
488     return @r;
489 }
490
491 sub body_20_39 () {
492   use integer;
493   my ($k,$n);
494   my @r=(
495         '($a,$b,$c,$d,$e)=@V;'.
496         '&add   ($e,eval(4*($j++&15))."(%rsp)");',      # X[]+K xfer
497         '&xor   (@T[0],$d);',   # ($b^$d)
498         '&mov   (@T[1],$a);',   # $b in next round
499         '&$_rol ($a,5);',
500         '&xor   (@T[0],$c);',   # ($b^$d^$c)
501         '&add   ($e,$a);',
502         '&$_ror ($b,7);',       # $b>>>2
503         '&add   ($e,@T[0]);'    .'unshift(@V,pop(@V)); unshift(@T,pop(@T));'
504         );
505         $n = scalar(@r);
506         $k = (($jj+1)*8/20)*20*$n/8;    # 8 aesencs per these 20 rounds
507         @r[$k%$n].='&$aesenc();'        if ($jj==$k/$n);
508         $jj++;
509     return @r;
510 }
511
512 sub body_40_59 () {
513   use integer;
514   my ($k,$n);
515   my @r=(
516         '($a,$b,$c,$d,$e)=@V;'.
517         '&mov   (@T[1],$c);',
518         '&xor   ($c,$d);',
519         '&add   ($e,eval(4*($j++&15))."(%rsp)");',      # X[]+K xfer
520         '&and   (@T[1],$d);',
521         '&and   (@T[0],$c);',   # ($b&($c^$d))
522         '&$_ror ($b,7);',       # $b>>>2
523         '&add   ($e,@T[1]);',
```

```
524         '&mov   (@T[1],$a);',   # $b in next round
525         '&$_rol ($a,5);',
526         '&add   ($e,@T[0]);',
527         '&xor   ($c,$d);',      # restore $c
528         '&add   ($e,$a);'       .'unshift(@V,pop(@V)); unshift(@T,pop(@T));'
529         );
530         $n = scalar(@r);
531         $k=(($jj+1)*12/20)*20*$n/12;    # 12 aesencs per these 20 rounds
532         @r[$k%$n].='&$aesenc();'        if ($jj==$k/$n);
533         $jj++;
534     return @r;
535 }
536 $code.=<<___;
537 .align  16
538 .Loop_ssse3:
539 ___
540         &Xupdate_ssse3_16_31(\&body_00_19);
541         &Xupdate_ssse3_16_31(\&body_00_19);
542         &Xupdate_ssse3_16_31(\&body_00_19);
543         &Xupdate_ssse3_16_31(\&body_00_19);
544         &Xupdate_ssse3_32_79(\&body_00_19);
545         &Xupdate_ssse3_32_79(\&body_20_39);
546         &Xupdate_ssse3_32_79(\&body_20_39);
547         &Xupdate_ssse3_32_79(\&body_20_39);
548         &Xupdate_ssse3_32_79(\&body_20_39);
549         &Xupdate_ssse3_32_79(\&body_20_39);
550         &Xupdate_ssse3_32_79(\&body_40_59);
551         &Xupdate_ssse3_32_79(\&body_40_59);
552         &Xupdate_ssse3_32_79(\&body_40_59);
553         &Xupdate_ssse3_32_79(\&body_40_59);
554         &Xupdate_ssse3_32_79(\&body_40_59);
555         &Xupdate_ssse3_32_79(\&body_20_39);
556         &Xuplast_ssse3_80(\&body_20_39);        # can jump to "done"

558                                 $saved_j=$j; @saved_V=@V;
559                                 $saved_r=$r; @saved_rndkey=@rndkey;

561         &Xloop_ssse3(\&body_20_39);
562         &Xloop_ssse3(\&body_20_39);
563         &Xloop_ssse3(\&body_20_39);

565 $code.=<<___;
566         movups  $iv,48($out,$in0)               # write output
567         lea     64($in0),$in0

569         add     0($ctx),$A                      # update context
570         add     4($ctx),@T[0]
571         add     8($ctx),$C
572         add     12($ctx),$D
573         mov     $A,0($ctx)
574         add     16($ctx),$E
575         mov     @T[0],4($ctx)
576         mov     @T[0],$B                        # magic seed
577         mov     $C,8($ctx)
578         mov     $D,12($ctx)
579         mov     $E,16($ctx)
580         jmp     .Loop_ssse3

582 .align  16
583 .Ldone_ssse3:
584 ___
585                                 $jj=$j=$saved_j; @V=@saved_V;
586                                 $r=$saved_r;     @rndkey=@saved_rndkey;

588         &Xtail_ssse3(\&body_20_39);
589         &Xtail_ssse3(\&body_20_39);
```



```
590         &Xtail_ssse3(\&body_20_39);

592 $code.=<<___;
593         movups  $iv,48($out,$in0)               # write output
594         mov     88(%rsp),$ivp                   # restore $ivp

596         add     0($ctx),$A                      # update context
597         add     4($ctx),@T[0]
598         add     8($ctx),$C
599         mov     $A,0($ctx)
600         add     12($ctx),$D
601         mov     @T[0],4($ctx)
602         add     16($ctx),$E
603         mov     $C,8($ctx)
604         mov     $D,12($ctx)
605         mov     $E,16($ctx)
606         movups  $iv,($ivp)                      # write IV
607 ___
608 $code.=<<___ if ($win64);
609         movaps  96+0(%rsp),%xmm6
610         movaps  96+16(%rsp),%xmm7
611         movaps  96+32(%rsp),%xmm8
612         movaps  96+48(%rsp),%xmm9
613         movaps  96+64(%rsp),%xmm10
614         movaps  96+80(%rsp),%xmm11
615         movaps  96+96(%rsp),%xmm12
616         movaps  96+112(%rsp),%xmm13
617         movaps  96+128(%rsp),%xmm14
618         movaps  96+144(%rsp),%xmm15
619 ___
620 $code.=<<___;
621         lea     '104+($win64?10*16:0)'(%rsp),%rsi
622         mov     0(%rsi),%r15
623         mov     8(%rsi),%r14
624         mov     16(%rsi),%r13
625         mov     24(%rsi),%r12
626         mov     32(%rsi),%rbp
627         mov     40(%rsi),%rbx
628         lea     48(%rsi),%rsp
629 .Lepilogue_ssse3:
630         ret
631 .size   aesni_cbc_sha1_enc_ssse3,.-aesni_cbc_sha1_enc_ssse3
632 ___

634 $j=$jj=$r=$sn=0;

636 if ($avx) {
637 my ($in0,$out,$len,$key,$ivp,$ctx,$inp)=("%rdi","%rsi","%rdx","%rcx","%r8","%r9"

639 my $Xi=4;
640 my @X=map("%xmm$_",(4..7,0..3));
641 my @Tx=map("%xmm$_",(8..10));
642 my @V=($A,$B,$C,$D,$E)=("%eax","%ebx","%ecx","%edx","%ebp");    # size optimizat
643 my @T=("%esi","%edi");

645 my $_rol=sub { &shld(@_[0],@_) };
646 my $_ror=sub { &shrd(@_[0],@_) };

648 $code.=<<___;
649 .type   aesni_cbc_sha1_enc_avx,\@function,6
650 .align  16
651 aesni_cbc_sha1_enc_avx:
652         mov     '($win64?56:8)'(%rsp),$inp      # load 7th argument
653         #shr    \$6,$len                        # debugging artefact
654         #jz     .Lepilogue_avx                  # debugging artefact
655         push    %rbx
```

```
656         push    %rbp
657         push    %r12
658         push    %r13
659         push    %r14
660         push    %r15
661         lea     '-104-($win64?10*16:0)'(%rsp),%rsp
662         #mov    $in0,$inp                       # debugging artefact
663         #lea    64(%rsp),$ctx                   # debugging artefact
664 ___
665 $code.=<<___ if ($win64);
666         movaps  %xmm6,96+0(%rsp)
667         movaps  %xmm7,96+16(%rsp)
668         movaps  %xmm8,96+32(%rsp)
669         movaps  %xmm9,96+48(%rsp)
670         movaps  %xmm10,96+64(%rsp)
671         movaps  %xmm11,96+80(%rsp)
672         movaps  %xmm12,96+96(%rsp)
673         movaps  %xmm13,96+112(%rsp)
674         movaps  %xmm14,96+128(%rsp)
675         movaps  %xmm15,96+144(%rsp)
676 .Lprologue_avx:
677 ___
678 $code.=<<___;
679         vzeroall
680         mov     $in0,%r12                       # reassign arguments
681         mov     $out,%r13
682         mov     $len,%r14
683         mov     $key,%r15
684         vmovdqu ($ivp),$iv                      # load IV
685         mov     $ivp,88(%rsp)                   # save $ivp
686 ___
687 my ($in0,$out,$len,$key)=map("%r$_",(12..15));  # reassign arguments
688 my $rounds="${ivp}d";
689 $code.=<<___;
690         shl     \$6,$len
691         sub     $in0,$out
692         mov     240($key),$rounds
693         add     \$112,$key              # size optimization
694         add     $inp,$len               # end of input

696         lea     K_XX_XX(%rip),$K_XX_XX
697         mov     0($ctx),$A              # load context
698         mov     4($ctx),$B
699         mov     8($ctx),$C
700         mov     12($ctx),$D
701         mov     $B,@T[0]                # magic seed
702         mov     16($ctx),$E

704         vmovdqa 64($K_XX_XX),@X[2]      # pbswap mask
705         vmovdqa 0($K_XX_XX),@Tx[1]      # K_00_19
706         vmovdqu 0($inp),@X[-4&7]        # load input to %xmm[0-3]
707         vmovdqu 16($inp),@X[-3&7]
708         vmovdqu 32($inp),@X[-2&7]
709         vmovdqu 48($inp),@X[-1&7]
710         vpshufb @X[2],@X[-4&7],@X[-4&7] # byte swap
711         add     \$64,$inp
712         vpshufb @X[2],@X[-3&7],@X[-3&7]
713         vpshufb @X[2],@X[-2&7],@X[-2&7]
714         vpshufb @X[2],@X[-1&7],@X[-1&7]
715         vpaddd  @Tx[1],@X[-4&7],@X[0]   # add K_00_19
716         vpaddd  @Tx[1],@X[-3&7],@X[1]
717         vpaddd  @Tx[1],@X[-2&7],@X[2]
718         vmovdqa @X[0],0(%rsp)           # X[]+K xfer to IALU
719         vmovdqa @X[1],16(%rsp)
720         vmovdqa @X[2],32(%rsp)
721         vmovups -112($key),$rndkey0     # $key[0]
```



```
722         vmovups 16-112($key),$rndkey[0] # forward reference
723         jmp     .Loop_avx
724 ___

726 my $aesenc=sub {
727   use integer;
728   my ($n,$k)=($r/10,$r%10);
729     if ($k==0) {
730       $code.=<<___;
731         vmovups         '16*$n'($in0),$in               # load input
732         vxorps          $rndkey0,$in,$in
733 ___
734       $code.=<<___ if ($n);
735         vmovups         $iv,'16*($n-1)'($out,$in0)      # write output
736 ___
737       $code.=<<___;
738         vxorps          $in,$iv,$iv
739         vaesenc         $rndkey[0],$iv,$iv
740         vmovups         '32+16*$k-112'($key),$rndkey[1]
741 ___
742     } elsif ($k==9) {
743       $sn++;
744       $code.=<<___;
745         cmp             \$11,$rounds
746         jb              .Lvaesenclast$sn
747         vaesenc         $rndkey[0],$iv,$iv
748         vmovups         '32+16*($k+0)-112'($key),$rndkey[1]
749         vaesenc         $rndkey[1],$iv,$iv
750         vmovups         '32+16*($k+1)-112'($key),$rndkey[0]
751         je              .Lvaesenclast$sn
752         vaesenc         $rndkey[0],$iv,$iv
753         vmovups         '32+16*($k+2)-112'($key),$rndkey[1]
754         vaesenc         $rndkey[1],$iv,$iv
755         vmovups         '32+16*($k+3)-112'($key),$rndkey[0]
756 .Lvaesenclast$sn:
757         vaesenclast     $rndkey[0],$iv,$iv
758         vmovups         16-112($key),$rndkey[1]         # forward reference
759 ___
760     } else {
761       $code.=<<___;
762         vaesenc         $rndkey[0],$iv,$iv
763         vmovups         '32+16*$k-112'($key),$rndkey[1]
764 ___
765     }
766     $r++;       unshift(@rndkey,pop(@rndkey));
767 };

769 sub Xupdate_avx_16_31()         # recall that $Xi starts wtih 4
770 { use integer;
771   my $body = shift;
772   my @insns = (&$body,&$body,&$body,&$body);    # 40 instructions
773   my ($a,$b,$c,$d,$e);

775          eval(shift(@insns));
776          eval(shift(@insns));
777         &vpalignr(@X[0],@X[-3&7],@X[-4&7],8);   # compose "X[-14]" in "X[0]"
778          eval(shift(@insns));
779          eval(shift(@insns));

781           &vpaddd       (@Tx[1],@Tx[1],@X[-1&7]);
782          eval(shift(@insns));
783          eval(shift(@insns));
784         &vpsrldq(@Tx[0],@X[-1&7],4);    # "X[-3]", 3 dwords
785          eval(shift(@insns));
786          eval(shift(@insns));
787         &vpxor  (@X[0],@X[0],@X[-4&7]);         # "X[0]"^="X[-16]"
```

```
788          eval(shift(@insns));
789          eval(shift(@insns));

791         &vpxor  (@Tx[0],@Tx[0],@X[-2&7]);       # "X[-3]"^"X[-8]"
792          eval(shift(@insns));
793          eval(shift(@insns));
794          eval(shift(@insns));
795          eval(shift(@insns));

797         &vpxor  (@X[0],@X[0],@Tx[0]);           # "X[0]"^="X[-3]"^"X[-8]"
798          eval(shift(@insns));
799          eval(shift(@insns));
800           &vmovdqa      (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
801          eval(shift(@insns));
802          eval(shift(@insns));

804         &vpsrld (@Tx[0],@X[0],31);
805          eval(shift(@insns));
806          eval(shift(@insns));
807          eval(shift(@insns));
808          eval(shift(@insns));

810         &vpslldq(@Tx[2],@X[0],12);              # "X[0]"<<96, extract one dword
811         &vpaddd (@X[0],@X[0],@X[0]);
812          eval(shift(@insns));
813          eval(shift(@insns));
814          eval(shift(@insns));
815          eval(shift(@insns));

817         &vpsrld (@Tx[1],@Tx[2],30);
818         &vpor   (@X[0],@X[0],@Tx[0]);           # "X[0]"<<<=1
819          eval(shift(@insns));
820          eval(shift(@insns));
821          eval(shift(@insns));
822          eval(shift(@insns));

824         &vpslld (@Tx[2],@Tx[2],2);
825         &vpxor  (@X[0],@X[0],@Tx[1]);
826          eval(shift(@insns));
827          eval(shift(@insns));
828          eval(shift(@insns));
829          eval(shift(@insns));

831         &vpxor  (@X[0],@X[0],@Tx[2]);           # "X[0]"^=("X[0]">>96)<<<2
832          eval(shift(@insns));
833          eval(shift(@insns));
834           &vmovdqa      (@Tx[2],eval(16*(($Xi)/5))."($K_XX_XX)");       # K_XX_X
835          eval(shift(@insns));
836          eval(shift(@insns));


839          foreach (@insns) { eval; }     # remaining instructions [if any]

841   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
842                 push(@Tx,shift(@Tx));
843 }

845 sub Xupdate_avx_32_79()
846 { use integer;
847   my $body = shift;
848   my @insns = (&$body,&$body,&$body,&$body);    # 32 to 48 instructions
849   my ($a,$b,$c,$d,$e);

851         &vpalignr(@Tx[0],@X[-1&7],@X[-2&7],8);  # compose "X[-6]"
852         &vpxor  (@X[0],@X[0],@X[-4&7]);         # "X[0]"="X[-32]"^"X[-16]"
853          eval(shift(@insns));           # body_20_39
```



```
854          eval(shift(@insns));
855          eval(shift(@insns));
856          eval(shift(@insns));           # rol

858         &vpxor  (@X[0],@X[0],@X[-7&7]);         # "X[0]"^="X[-28]"
859          eval(shift(@insns));
860          eval(shift(@insns));    if (@insns[0] !~ /&ro[rl]/);
861         if ($Xi%5) {
862           &vmovdqa      (@Tx[2],@Tx[1]);# "perpetuate" K_XX_XX...
863         } else {                        # ... or load next one
864           &vmovdqa      (@Tx[2],eval(16*($Xi/5))."($K_XX_XX)");
865         }
866           &vpaddd       (@Tx[1],@Tx[1],@X[-1&7]);
867          eval(shift(@insns));           # ror
868          eval(shift(@insns));

870         &vpxor  (@X[0],@X[0],@Tx[0]);           # "X[0]"^="X[-6]"
871          eval(shift(@insns));           # body_20_39
872          eval(shift(@insns));
873          eval(shift(@insns));
874          eval(shift(@insns));           # rol

876         &vpsrld (@Tx[0],@X[0],30);
877           &vmovdqa      (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
878          eval(shift(@insns));
879          eval(shift(@insns));
880          eval(shift(@insns));           # ror
881          eval(shift(@insns));

883         &vpslld (@X[0],@X[0],2);
884          eval(shift(@insns));           # body_20_39
885          eval(shift(@insns));
886          eval(shift(@insns));
887          eval(shift(@insns));           # rol
888          eval(shift(@insns));
889          eval(shift(@insns));
890          eval(shift(@insns));           # ror
891          eval(shift(@insns));

893         &vpor   (@X[0],@X[0],@Tx[0]);           # "X[0]"<<<=2
894          eval(shift(@insns));           # body_20_39
895          eval(shift(@insns));
896           &vmovdqa      (@Tx[1],@X[0])  if ($Xi<19);
897          eval(shift(@insns));
898          eval(shift(@insns));           # rol
899          eval(shift(@insns));
900          eval(shift(@insns));
901          eval(shift(@insns));           # rol
902          eval(shift(@insns));

904          foreach (@insns) { eval; }     # remaining instructions

906   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
907                 push(@Tx,shift(@Tx));
908 }

910 sub Xuplast_avx_80()
911 { use integer;
912   my $body = shift;
913   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
914   my ($a,$b,$c,$d,$e);

916          eval(shift(@insns));
917           &vpaddd       (@Tx[1],@Tx[1],@X[-1&7]);
918          eval(shift(@insns));
919          eval(shift(@insns));
```

```
920          eval(shift(@insns));
921          eval(shift(@insns));

923           &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer IAL

925          foreach (@insns) { eval; }             # remaining instructions

927         &cmp    ($inp,$len);
928         &je     (".Ldone_avx");

930         unshift(@Tx,pop(@Tx));

932         &vmovdqa(@X[2],"64($K_XX_XX)");         # pbswap mask
933         &vmovdqa(@Tx[1],"0($K_XX_XX)");         # K_00_19
934         &vmovdqu(@X[-4&7],"0($inp)");           # load input
935         &vmovdqu(@X[-3&7],"16($inp)");
936         &vmovdqu(@X[-2&7],"32($inp)");
937         &vmovdqu(@X[-1&7],"48($inp)");
938         &vpshufb(@X[-4&7],@X[-4&7],@X[2]);      # byte swap
939         &add    ($inp,64);

941   $Xi=0;
942 }

944 sub Xloop_avx()
945 { use integer;
946   my $body = shift;
947   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
948   my ($a,$b,$c,$d,$e);

950          eval(shift(@insns));
951          eval(shift(@insns));
952         &vpshufb(@X[($Xi-3)&7],@X[($Xi-3)&7],@X[2]);
953          eval(shift(@insns));
954          eval(shift(@insns));
955         &vpaddd (@X[$Xi&7],@X[($Xi-4)&7],@Tx[1]);
956          eval(shift(@insns));
957          eval(shift(@insns));
958          eval(shift(@insns));
959          eval(shift(@insns));
960         &vmovdqa(eval(16*$Xi)."(%rsp)",@X[$Xi&7]);      # X[]+K xfer to IALU
961          eval(shift(@insns));
962          eval(shift(@insns));

964         foreach (@insns) { eval; }
965   $Xi++;
966 }

968 sub Xtail_avx()
969 { use integer;
970   my $body = shift;
971   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
972   my ($a,$b,$c,$d,$e);

974         foreach (@insns) { eval; }
975 }

977 $code.=<<___;
978 .align  16
979 .Loop_avx:
980 ___
981         &Xupdate_avx_16_31(\&body_00_19);
982         &Xupdate_avx_16_31(\&body_00_19);
983         &Xupdate_avx_16_31(\&body_00_19);
984         &Xupdate_avx_16_31(\&body_00_19);
985         &Xupdate_avx_32_79(\&body_00_19);
```



```
986         &Xupdate_avx_32_79(\&body_20_39);
987         &Xupdate_avx_32_79(\&body_20_39);
988         &Xupdate_avx_32_79(\&body_20_39);
989         &Xupdate_avx_32_79(\&body_20_39);
990         &Xupdate_avx_32_79(\&body_20_39);
991         &Xupdate_avx_32_79(\&body_40_59);
992         &Xupdate_avx_32_79(\&body_40_59);
993         &Xupdate_avx_32_79(\&body_40_59);
994         &Xupdate_avx_32_79(\&body_40_59);
995         &Xupdate_avx_32_79(\&body_40_59);
996         &Xupdate_avx_32_79(\&body_20_39);
997         &Xuplast_avx_80(\&body_20_39);  # can jump to "done"

999                                 $saved_j=$j; @saved_V=@V;
1000                                $saved_r=$r; @saved_rndkey=@rndkey;

1002        &Xloop_avx(\&body_20_39);
1003        &Xloop_avx(\&body_20_39);
1004        &Xloop_avx(\&body_20_39);

1006 $code.=<<___;
1007        vmovups $iv,48($out,$in0)               # write output
1008        lea     64($in0),$in0

1010        add     0($ctx),$A                      # update context
1011        add     4($ctx),@T[0]
1012        add     8($ctx),$C
1013        add     12($ctx),$D
1014        mov     $A,0($ctx)
1015        add     16($ctx),$E
1016        mov     @T[0],4($ctx)
1017        mov     @T[0],$B                        # magic seed
1018        mov     $C,8($ctx)
1019        mov     $D,12($ctx)
1020        mov     $E,16($ctx)
1021        jmp     .Loop_avx

1023 .align  16
1024 .Ldone_avx:
1025 ___
1026                                $jj=$j=$saved_j; @V=@saved_V;
1027                                $r=$saved_r;     @rndkey=@saved_rndkey;

1029        &Xtail_avx(\&body_20_39);
1030        &Xtail_avx(\&body_20_39);
1031        &Xtail_avx(\&body_20_39);

1033 $code.=<<___;
1034        vmovups $iv,48($out,$in0)               # write output
1035        mov     88(%rsp),$ivp                   # restore $ivp

1037        add     0($ctx),$A                      # update context
1038        add     4($ctx),@T[0]
1039        add     8($ctx),$C
1040        mov     $A,0($ctx)
1041        add     12($ctx),$D
1042        mov     @T[0],4($ctx)
1043        add     16($ctx),$E
1044        mov     $C,8($ctx)
1045        mov     $D,12($ctx)
1046        mov     $E,16($ctx)
1047        vmovups $iv,($ivp)                      # write IV
1048        vzeroall
1049 ___
1050 $code.=<<___ if ($win64);
1051        movaps  96+0(%rsp),%xmm6
```

```
1052         movaps  96+16(%rsp),%xmm7
1053         movaps  96+32(%rsp),%xmm8
1054         movaps  96+48(%rsp),%xmm9
1055         movaps  96+64(%rsp),%xmm10
1056         movaps  96+80(%rsp),%xmm11
1057         movaps  96+96(%rsp),%xmm12
1058         movaps  96+112(%rsp),%xmm13
1059         movaps  96+128(%rsp),%xmm14
1060         movaps  96+144(%rsp),%xmm15
1061 ___
1062 $code.=<<___;
1063         lea     `104+($win64?10*16:0)`(%rsp),%rsi
1064         mov     0(%rsi),%r15
1065         mov     8(%rsi),%r14
1066         mov     16(%rsi),%r13
1067         mov     24(%rsi),%r12
1068         mov     32(%rsi),%rbp
1069         mov     40(%rsi),%rbx
1070         lea     48(%rsi),%rsp
1071 .Lepilogue_avx:
1072         ret
1073 .size   aesni_cbc_sha1_enc_avx,.-aesni_cbc_sha1_enc_avx
1074 ___
1075 }
1076 $code.=<<___;
1077 .align  64
1078 K_XX_XX:
1079 .long   0x5a827999,0x5a827999,0x5a827999,0x5a827999     # K_00_19
1080 .long   0x6ed9eba1,0x6ed9eba1,0x6ed9eba1,0x6ed9eba1     # K_20_39
1081 .long   0x8f1bbcdc,0x8f1bbcdc,0x8f1bbcdc,0x8f1bbcdc     # K_40_59
1082 .long   0xca62c1d6,0xca62c1d6,0xca62c1d6,0xca62c1d6     # K_60_79
1083 .long   0x00010203,0x04050607,0x08090a0b,0x0c0d0e0f     # pbswap mask

1085 .asciz  "AESNI-CBC+SHA1 stitch for x86_64, CRYPTOGAMS by <appro\@openssl.org>"
1086 .align  64
1087 ___

1089 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
1090 #           CONTEXT *context,DISPATCHER_CONTEXT *disp)
1091 if ($win64) {
1092 $rec="%rcx";
1093 $frame="%rdx";
1094 $context="%r8";
1095 $disp="%r9";

1097 $code.=<<___;
1098 .extern __imp_RtlVirtualUnwind
1099 .type   ssse3_handler,\@abi-omnipotent
1100 .align  16
1101 ssse3_handler:
1102         push    %rsi
1103         push    %rdi
1104         push    %rbx
1105         push    %rbp
1106         push    %r12
1107         push    %r13
1108         push    %r14
1109         push    %r15
1110         pushfq
1111         sub     \$64,%rsp

1113         mov     120($context),%rax      # pull context->Rax
1114         mov     248($context),%rbx      # pull context->Rip

1116         mov     8($disp),%rsi           # disp->ImageBase
1117         mov     56($disp),%r11          # disp->HandlerData
```



```
1119         mov     0(%r11),%r10d           # HandlerData[0]
1120         lea     (%rsi,%r10),%r10        # prologue label
1121         cmp     %r10,%rbx               # context->Rip<prologue label
1122         jb      .Lcommon_seh_tail

1124         mov     152($context),%rax      # pull context->Rsp

1126         mov     4(%r11),%r10d           # HandlerData[1]
1127         lea     (%rsi,%r10),%r10        # epilogue label
1128         cmp     %r10,%rbx               # context->Rip>=epilogue label
1129         jae     .Lcommon_seh_tail

1131         lea     96(%rax),%rsi
1132         lea     512($context),%rdi      # &context.Xmm6
1133         mov     \$20,%ecx
1134         .long   0xa548f3fc              # cld; rep movsq
1135         lea     `104+10*16`(%rax),%rax  # adjust stack pointer

1137         mov     0(%rax),%r15
1138         mov     8(%rax),%r14
1139         mov     16(%rax),%r13
1140         mov     24(%rax),%r12
1141         mov     32(%rax),%rbp
1142         mov     40(%rax),%rbx
1143         lea     48(%rax),%rax
1144         mov     %rbx,144($context)      # restore context->Rbx
1145         mov     %rbp,160($context)      # restore context->Rbp
1146         mov     %r12,216($context)      # restore context->R12
1147         mov     %r13,224($context)      # restore context->R13
1148         mov     %r14,232($context)      # restore context->R14
1149         mov     %r15,240($context)      # restore context->R15

1151 .Lcommon_seh_tail:
1152         mov     8(%rax),%rdi
1153         mov     16(%rax),%rsi
1154         mov     %rax,152($context)      # restore context->Rsp
1155         mov     %rsi,168($context)      # restore context->Rsi
1156         mov     %rdi,176($context)      # restore context->Rdi

1158         mov     40($disp),%rdi          # disp->ContextRecord
1159         mov     $context,%rsi           # context
1160         mov     \$154,%ecx              # sizeof(CONTEXT)
1161         .long   0xa548f3fc              # cld; rep movsq

1163         mov     $disp,%rsi
1164         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
1165         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
1166         mov     0(%rsi),%r8             # arg3, disp->ControlPc
1167         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
1168         mov     40(%rsi),%r10           # disp->ContextRecord
1169         lea     56(%rsi),%r11           # &disp->HandlerData
1170         lea     24(%rsi),%r12           # &disp->EstablisherFrame
1171         mov     %r10,32(%rsp)           # arg5
1172         mov     %r11,40(%rsp)           # arg6
1173         mov     %r12,48(%rsp)           # arg7
1174         mov     %rcx,56(%rsp)           # arg8, (NULL)
1175         call    *__imp_RtlVirtualUnwind(%rip)

1177         mov     \$1,%eax                # ExceptionContinueSearch
1178         add     \$64,%rsp
1179         popfq
1180         pop     %r15
1181         pop     %r14
1182         pop     %r13
1183         pop     %r12
```

```
1184         pop     %rbp
1185         pop     %rbx
1186         pop     %rdi
1187         pop     %rsi
1188         ret
1189 .size   ssse3_handler,.-ssse3_handler

1191 .section        .pdata
1192 .align  4
1193         .rva    .LSEH_begin_aesni_cbc_sha1_enc_ssse3
1194         .rva    .LSEH_end_aesni_cbc_sha1_enc_ssse3
1195         .rva    .LSEH_info_aesni_cbc_sha1_enc_ssse3
1196 ___
1197 $code.=<<___ if ($avx);
1198         .rva    .LSEH_begin_aesni_cbc_sha1_enc_avx
1199         .rva    .LSEH_end_aesni_cbc_sha1_enc_avx
1200         .rva    .LSEH_info_aesni_cbc_sha1_enc_avx
1201 ___
1202 $code.=<<___;
1203 .section        .xdata
1204 .align  8
1205 .LSEH_info_aesni_cbc_sha1_enc_ssse3:
1206         .byte   9,0,0,0
1207         .rva    ssse3_handler
1208         .rva    .Lprologue_ssse3,.Lepilogue_ssse3       # HandlerData[]
1209 ___
1210 $code.=<<___ if ($avx);
1211 .LSEH_info_aesni_cbc_sha1_enc_avx:
1212         .byte   9,0,0,0
1213         .rva    ssse3_handler
1214         .rva    .Lprologue_avx,.Lepilogue_avx           # HandlerData[]
1215 ___
1216 }

1218 ####################################################################
1219 sub rex {
1220   local *opcode=shift;
1221   my ($dst,$src)=@_;
1222   my $rex=0;

1224     $rex|=0x04                  if($dst>=8);
1225     $rex|=0x01                  if($src>=8);
1226     push @opcode,$rex|0x40      if($rex);
1227 }

1229 sub aesni {
1230   my $line=shift;
1231   my @opcode=(0x66);

1233     if ($line=~/(aes[a-z]+)\s+%xmm([0-9]+),\s*%xmm([0-9]+)/) {
1234         my %opcodelet = (
1235                 "aesenc" => 0xdc,       "aesenclast" => 0xdd
1236         );
1237         return undef if (!defined($opcodelet{$1}));
1238         rex(\@opcode,$3,$2);
1239         push @opcode,0x0f,0x38,$opcodelet{$1};
1240         push @opcode,0xc0|($2&7)|(($3&7)<<3);   # ModR/M
1241         return ".byte\t".join(',',@opcode);
1242     }
1243     return $line;
1244 }

1246 $code =~ s/\`([^\`]*)\`/eval($1)/gem;
1247 $code =~ s/\b(aes.*%xmm[0-9]+).*$/aesni($1)/gem;

1249 print $code;
```



```
1250 close STDOUT;
1251 #endif /* ! codereview */
```

```
**********************************************************
   67129 Wed Aug 13 19:53:05 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/aesni-x86.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# This module implements support for Intel AES-NI extension. In
# OpenSSL context it’s used with Intel engine, but can also be used as
# drop-in replacement for crypto/aes/asm/aes-586.pl [see below for
# details].
#
# Performance.
#
# To start with see corresponding paragraph in aesni-x86_64.pl...
# Instead of filling table similar to one found there I’ve chosen to
# summarize *comparison* results for raw ECB, CTR and CBC benchmarks.
# The simplified table below represents 32-bit performance relative
# to 64-bit one in every given point. Ratios vary for different
# encryption modes, therefore interval values.
#
#       16-byte     64-byte     256-byte    1-KB        8-KB
#       53-67%      67-84%      91-94%      95-98%      97-99.5%
#
# Lower ratios for smaller block sizes are perfectly understandable,
# because function call overhead is higher in 32-bit mode. Largest
# 8-KB block performance is virtually same: 32-bit code is less than
# 1% slower for ECB, CBC and CCM, and ~3% slower otherwise.

# January 2011
#
# See aesni-x86_64.pl for details. Unlike x86_64 version this module
# interleaves at most 6 aes[enc|dec] instructions, because there are
# not enough registers for 8x interleave [which should be optimal for
# Sandy Bridge]. Actually, performance results for 6x interleave
# factor presented in aesni-x86_64.pl (except for CTR) are for this
# module.

# April 2011
#
# Add aesni_xts_[en|de]crypt. Westmere spends 1.50 cycles processing
# one byte out of 8KB with 128-bit key, Sandy Bridge - 1.09.

$PREFIX="aesni";        # if $PREFIX is set to "AES", the script
                        # generates drop-in replacement for
                        # crypto/aes/asm/aes-586.pl:-)
$inline=1;              # inline _aesni_[en|de]crypt

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],$0);

if ($PREFIX eq "aesni") { $movekey=*movups; }
else                    { $movekey=*movups; }

$len="eax";
$rounds="ecx";
```



```
$key="edx";
$inp="esi";
$out="edi";
$rounds_="ebx"; # backup copy for $rounds
$key_="ebp";    # backup copy for $key

$rndkey0="xmm0";
$rndkey1="xmm1";
$inout0="xmm2";
$inout1="xmm3";
$inout2="xmm4";
$inout3="xmm5"; $in1="xmm5";
$inout4="xmm6"; $in0="xmm6";
$inout5="xmm7"; $ivec="xmm7";

# AESNI extenstion
sub aeskeygenassist
{ my($dst,$src,$imm)=@_;
    if ("$dst:$src" =~ /xmm([0-7]):xmm([0-7])/)
    {   &data_byte(0x66,0x0f,0x3a,0xdf,0xc0|($1<<3)|$2,$imm);   }
}
sub aescommon
{ my($opcodelet,$dst,$src)=@_;
    if ("$dst:$src" =~ /xmm([0-7]):xmm([0-7])/)
    {   &data_byte(0x66,0x0f,0x38,$opcodelet,0xc0|($1<<3)|$2);}
}
sub aesimc      { aescommon(0xdb,@_); }
sub aesenc      { aescommon(0xdc,@_); }
sub aesenclast  { aescommon(0xdd,@_); }
sub aesdec      { aescommon(0xde,@_); }
sub aesdeclast  { aescommon(0xdf,@_); }
```

```
 93 # Inline version of internal aesni_[en|de]crypt1
 94 { my $sn;
 95 sub aesni_inline_generate1
 96 { my ($p,$inout,$ivec)=@_; $inout=$inout0 if (!defined($inout));
 97   $sn++;

 99     &$movekey           ($rndkey0,&QWP(0,$key));
100     &$movekey           ($rndkey1,&QWP(16,$key));
101     &xorps              ($ivec,$rndkey0)        if (defined($ivec));
102     &lea                ($key,&DWP(32,$key));
103     &xorps              ($inout,$ivec)          if (defined($ivec));
104     &xorps              ($inout,$rndkey0)       if (!defined($ivec));
105     &set_label("${p}1_loop_$sn");
106         eval"&aes${p}   ($inout,$rndkey1)";
107         &dec            ($rounds);
108         &$movekey       ($rndkey1,&QWP(0,$key));
109         &lea            ($key,&DWP(16,$key));
110     &jnz                (&label("${p}1_loop_$sn"));
111     eval"&aes${p}last   ($inout,$rndkey1)";
112 }}

114 sub aesni_generate1     # fully unrolled loop
115 { my ($p,$inout)=@_; $inout=$inout0 if (!defined($inout));

117     &function_begin_B("_aesni_${p}rypt1");
118         &movups         ($rndkey0,&QWP(0,$key));
119         &$movekey       ($rndkey1,&QWP(0x10,$key));
120         &xorps          ($inout,$rndkey0);
121         &$movekey       ($rndkey0,&QWP(0x20,$key));
122         &lea            ($key,&DWP(0x30,$key));
123         &cmp            ($rounds,11);
124         &jb             (&label("${p}128"));
125         &lea            ($key,&DWP(0x20,$key));
126         &je             (&label("${p}192"));
127         &lea            ($key,&DWP(0x20,$key));
128         eval"&aes${p}   ($inout,$rndkey1)";
129         &$movekey       ($rndkey1,&QWP(-0x40,$key));
130         eval"&aes${p}   ($inout,$rndkey0)";
131         &$movekey       ($rndkey0,&QWP(-0x30,$key));
132     &set_label("${p}192");
133         eval"&aes${p}   ($inout,$rndkey1)";
134         &$movekey       ($rndkey1,&QWP(-0x20,$key));
135         eval"&aes${p}   ($inout,$rndkey0)";
136         &$movekey       ($rndkey0,&QWP(-0x10,$key));
137     &set_label("${p}128");
138         eval"&aes${p}   ($inout,$rndkey1)";
139         &$movekey       ($rndkey1,&QWP(0,$key));
140         eval"&aes${p}   ($inout,$rndkey0)";
141         &$movekey       ($rndkey0,&QWP(0x10,$key));
142         eval"&aes${p}   ($inout,$rndkey1)";
143         &$movekey       ($rndkey1,&QWP(0x20,$key));
144         eval"&aes${p}   ($inout,$rndkey0)";
145         &$movekey       ($rndkey0,&QWP(0x30,$key));
146         eval"&aes${p}   ($inout,$rndkey1)";
147         &$movekey       ($rndkey1,&QWP(0x40,$key));
148         eval"&aes${p}   ($inout,$rndkey0)";
149         &$movekey       ($rndkey0,&QWP(0x50,$key));
150         eval"&aes${p}   ($inout,$rndkey1)";
151         &$movekey       ($rndkey1,&QWP(0x60,$key));
152         eval"&aes${p}   ($inout,$rndkey0)";
153         &$movekey       ($rndkey0,&QWP(0x70,$key));
154         eval"&aes${p}   ($inout,$rndkey1)";
155     eval"&aes${p}last   ($inout,$rndkey0)";
156     &ret();
157     &function_end_B("_aesni_${p}rypt1");
158 }
```

```
# void $PREFIX_encrypt (const void *inp,void *out,const AES_KEY *key);
&aesni_generate1("enc") if (!$inline);
&function_begin_B("${PREFIX}_encrypt");
        &mov    ("eax",&wparam(0));
        &mov    ($key,&wparam(2));
        &movups ($inout0,&QWP(0,"eax"));
        &mov    ($rounds,&DWP(240,$key));
        &mov    ("eax",&wparam(1));
        if ($inline)
        {   &aesni_inline_generate1("enc");     }
        else
        {   &call       ("_aesni_encrypt1");    }
        &movups (&QWP(0,"eax"),$inout0);
        &ret    ();
&function_end_B("${PREFIX}_encrypt");

# void $PREFIX_decrypt (const void *inp,void *out,const AES_KEY *key);
&aesni_generate1("dec") if(!$inline);
&function_begin_B("${PREFIX}_decrypt");
        &mov    ("eax",&wparam(0));
        &mov    ($key,&wparam(2));
        &movups ($inout0,&QWP(0,"eax"));
        &mov    ($rounds,&DWP(240,$key));
        &mov    ("eax",&wparam(1));
        if ($inline)
        {   &aesni_inline_generate1("dec");     }
        else
        {   &call       ("_aesni_decrypt1");    }
        &movups (&QWP(0,"eax"),$inout0);
        &ret    ();
&function_end_B("${PREFIX}_decrypt");

# _aesni_[en|de]cryptN are private interfaces, N denotes interleave
# factor. Why 3x subroutine were originally used in loops? Even though
# aes[enc|dec] latency was originally 6, it could be scheduled only
# every *2nd* cycle. Thus 3x interleave was the one providing optimal
# utilization, i.e. when subroutine's throughput is virtually same as
# of non-interleaved subroutine [for number of input blocks up to 3].
# This is why it makes no sense to implement 2x subroutine.
# aes[enc|dec] latency in next processor generation is 8, but the
# instructions can be scheduled every cycle. Optimal interleave for
# new processor is therefore 8x, but it's unfeasible to accommodate it
# in XMM registers addreassable in 32-bit mode and therefore 6x is
# used instead...

sub aesni_generate3
{ my $p=shift;

    &function_begin_B("_aesni_${p}rypt3");
        &$movekey       ($rndkey0,&QWP(0,$key));
        &shr            ($rounds,1);
        &$movekey       ($rndkey1,&QWP(16,$key));
        &lea            ($key,&DWP(32,$key));
        &xorps          ($inout0,$rndkey0);
        &pxor           ($inout1,$rndkey0);
        &pxor           ($inout2,$rndkey0);
        &$movekey       ($rndkey0,&QWP(0,$key));

    &set_label("${p}3_loop");
        eval"&aes${p}   ($inout0,$rndkey1)";
        eval"&aes${p}   ($inout1,$rndkey1)";
        &dec            ($rounds);
        eval"&aes${p}   ($inout2,$rndkey1)";
        &$movekey       ($rndkey1,&QWP(16,$key));
        eval"&aes${p}   ($inout0,$rndkey0)";
        eval"&aes${p}   ($inout1,$rndkey0)";
```



```
        &lea            ($key,&DWP(32,$key));
        eval"&aes${p}   ($inout2,$rndkey0)";
        &$movekey       ($rndkey0,&QWP(0,$key));
        &jnz            (&label("${p}3_loop"));
    eval"&aes${p}       ($inout0,$rndkey1)";
    eval"&aes${p}       ($inout1,$rndkey1)";
    eval"&aes${p}       ($inout2,$rndkey1)";
    eval"&aes${p}last   ($inout0,$rndkey0)";
    eval"&aes${p}last   ($inout1,$rndkey0)";
    eval"&aes${p}last   ($inout2,$rndkey0)";
    &ret();
    &function_end_B("_aesni_${p}rypt3");
}

# 4x interleave is implemented to improve small block performance,
# most notably [and naturally] 4 block by ~30%. One can argue that one
# should have implemented 5x as well, but improvement  would be <20%,
# so it's not worth it...
sub aesni_generate4
{ my $p=shift;

    &function_begin_B("_aesni_${p}rypt4");
        &$movekey       ($rndkey0,&QWP(0,$key));
        &$movekey       ($rndkey1,&QWP(16,$key));
        &shr            ($rounds,1);
        &lea            ($key,&DWP(32,$key));
        &xorps          ($inout0,$rndkey0);
        &pxor           ($inout1,$rndkey0);
        &pxor           ($inout2,$rndkey0);
        &pxor           ($inout3,$rndkey0);
        &$movekey       ($rndkey0,&QWP(0,$key));

    &set_label("${p}4_loop");
        eval"&aes${p}   ($inout0,$rndkey1)";
        eval"&aes${p}   ($inout1,$rndkey1)";
        &dec            ($rounds);
        eval"&aes${p}   ($inout2,$rndkey1)";
        eval"&aes${p}   ($inout3,$rndkey1)";
        &$movekey       ($rndkey1,&QWP(16,$key));
        eval"&aes${p}   ($inout0,$rndkey0)";
        eval"&aes${p}   ($inout1,$rndkey0)";
        &lea            ($key,&DWP(32,$key));
        eval"&aes${p}   ($inout2,$rndkey0)";
        eval"&aes${p}   ($inout3,$rndkey0)";
        &$movekey       ($rndkey0,&QWP(0,$key));
    &jnz                (&label("${p}4_loop"));

    eval"&aes${p}       ($inout0,$rndkey1)";
    eval"&aes${p}       ($inout1,$rndkey1)";
    eval"&aes${p}       ($inout2,$rndkey1)";
    eval"&aes${p}       ($inout3,$rndkey1)";
    eval"&aes${p}last   ($inout0,$rndkey0)";
    eval"&aes${p}last   ($inout1,$rndkey0)";
    eval"&aes${p}last   ($inout2,$rndkey0)";
    eval"&aes${p}last   ($inout3,$rndkey0)";
    &ret();
    &function_end_B("_aesni_${p}rypt4");
}

sub aesni_generate6
{ my $p=shift;

    &function_begin_B("_aesni_${p}rypt6");
    &static_label("_aesni_${p}rypt6_enter");
        &$movekey       ($rndkey0,&QWP(0,$key));
        &shr            ($rounds,1);
```

```
291         &$movekey       ($rndkey1,&QWP(16,$key));
292         &lea            ($key,&DWP(32,$key));
293         &xorps          ($inout0,$rndkey0);
294         &pxor           ($inout1,$rndkey0);     # pxor does better here
295         eval"&aes${p}   ($inout0,$rndkey1)";
296         &pxor           ($inout2,$rndkey0);
297         eval"&aes${p}   ($inout1,$rndkey1)";
298         &pxor           ($inout3,$rndkey0);
299         &dec            ($rounds);
300         eval"&aes${p}   ($inout2,$rndkey1)";
301         &pxor           ($inout4,$rndkey0);
302         eval"&aes${p}   ($inout3,$rndkey1)";
303         &pxor           ($inout5,$rndkey0);
304         eval"&aes${p}   ($inout4,$rndkey1)";
305         &$movekey       ($rndkey0,&QWP(0,$key));
306         eval"&aes${p}   ($inout5,$rndkey1)";
307         &jmp            (&label("_aesni_${p}rypt6_enter"));

309     &set_label("${p}6_loop",16);
310         eval"&aes${p}   ($inout0,$rndkey1)";
311         eval"&aes${p}   ($inout1,$rndkey1)";
312         &dec            ($rounds);
313         eval"&aes${p}   ($inout2,$rndkey1)";
314         eval"&aes${p}   ($inout3,$rndkey1)";
315         eval"&aes${p}   ($inout4,$rndkey1)";
316         eval"&aes${p}   ($inout5,$rndkey1)";
317     &set_label("_aesni_${p}rypt6_enter",16);
318         &$movekey       ($rndkey1,&QWP(16,$key));
319         eval"&aes${p}   ($inout0,$rndkey0)";
320         eval"&aes${p}   ($inout1,$rndkey0)";
321         &lea            ($key,&DWP(32,$key));
322         eval"&aes${p}   ($inout2,$rndkey0)";
323         eval"&aes${p}   ($inout3,$rndkey0)";
324         eval"&aes${p}   ($inout4,$rndkey0)";
325         eval"&aes${p}   ($inout5,$rndkey0)";
326         &$movekey       ($rndkey0,&QWP(0,$key));
327     &jnz                (&label("${p}6_loop"));

329     eval"&aes${p}       ($inout0,$rndkey1)";
330     eval"&aes${p}       ($inout1,$rndkey1)";
331     eval"&aes${p}       ($inout2,$rndkey1)";
332     eval"&aes${p}       ($inout3,$rndkey1)";
333     eval"&aes${p}       ($inout4,$rndkey1)";
334     eval"&aes${p}       ($inout5,$rndkey1)";
335     eval"&aes${p}last   ($inout0,$rndkey0)";
336     eval"&aes${p}last   ($inout1,$rndkey0)";
337     eval"&aes${p}last   ($inout2,$rndkey0)";
338     eval"&aes${p}last   ($inout3,$rndkey0)";
339     eval"&aes${p}last   ($inout4,$rndkey0)";
340     eval"&aes${p}last   ($inout5,$rndkey0)";
341     &ret();
342     &function_end_B("_aesni_${p}rypt6");
343 }
344 &aesni_generate3("enc") if ($PREFIX eq "aesni");
345 &aesni_generate3("dec");
346 &aesni_generate4("enc") if ($PREFIX eq "aesni");
347 &aesni_generate4("dec");
348 &aesni_generate6("enc") if ($PREFIX eq "aesni");
349 &aesni_generate6("dec");
```



```
350 if ($PREFIX eq "aesni") {
351 ########################################################################
352 # void aesni_ecb_encrypt (const void *in, void *out,
353 #                         size_t length, const AES_KEY *key,
354 #                         int enc);
355 &function_begin("aesni_ecb_encrypt");
356         &mov    ($inp,&wparam(0));
357         &mov    ($out,&wparam(1));
358         &mov    ($len,&wparam(2));
359         &mov    ($key,&wparam(3));
360         &mov    ($rounds_,&wparam(4));
361         &and    ($len,-16);
362         &jz     (&label("ecb_ret"));
363         &mov    ($rounds,&DWP(240,$key));
364         &test   ($rounds_,$rounds_);
365         &jz     (&label("ecb_decrypt"));

367         &mov    ($key_,$key);           # backup $key
368         &mov    ($rounds_,$rounds);     # backup $rounds
369         &cmp    ($len,0x60);
370         &jb     (&label("ecb_enc_tail"));

372         &movdqu ($inout0,&QWP(0,$inp));
373         &movdqu ($inout1,&QWP(0x10,$inp));
374         &movdqu ($inout2,&QWP(0x20,$inp));
375         &movdqu ($inout3,&QWP(0x30,$inp));
376         &movdqu ($inout4,&QWP(0x40,$inp));
377         &movdqu ($inout5,&QWP(0x50,$inp));
378         &lea    ($inp,&DWP(0x60,$inp));
379         &sub    ($len,0x60);
380         &jmp    (&label("ecb_enc_loop6_enter"));

382 &set_label("ecb_enc_loop6",16);
383         &movups (&QWP(0,$out),$inout0);
384         &movdqu ($inout0,&QWP(0,$inp));
385         &movups (&QWP(0x10,$out),$inout1);
386         &movdqu ($inout1,&QWP(0x10,$inp));
387         &movups (&QWP(0x20,$out),$inout2);
388         &movdqu ($inout2,&QWP(0x20,$inp));
389         &movups (&QWP(0x30,$out),$inout3);
390         &movdqu ($inout3,&QWP(0x30,$inp));
391         &movups (&QWP(0x40,$out),$inout4);
392         &movdqu ($inout4,&QWP(0x40,$inp));
393         &movups (&QWP(0x50,$out),$inout5);
394         &lea    ($out,&DWP(0x60,$out));
395         &movdqu ($inout5,&QWP(0x50,$inp));
396         &lea    ($inp,&DWP(0x60,$inp));
397 &set_label("ecb_enc_loop6_enter");

399         &call   ("_aesni_encrypt6");

401         &mov    ($key,$key_);           # restore $key
402         &mov    ($rounds,$rounds_);     # restore $rounds
403         &sub    ($len,0x60);
404         &jnc    (&label("ecb_enc_loop6"));

406         &movups (&QWP(0,$out),$inout0);
407         &movups (&QWP(0x10,$out),$inout1);
408         &movups (&QWP(0x20,$out),$inout2);
409         &movups (&QWP(0x30,$out),$inout3);
410         &movups (&QWP(0x40,$out),$inout4);
411         &movups (&QWP(0x50,$out),$inout5);
412         &lea    ($out,&DWP(0x60,$out));
413         &add    ($len,0x60);
414         &jz     (&label("ecb_ret"));
```

```
&set_label("ecb_enc_tail");
        &movups ($inout0,&QWP(0,$inp));
        &cmp    ($len,0x20);
        &jb     (&label("ecb_enc_one"));
        &movups ($inout1,&QWP(0x10,$inp));
        &je     (&label("ecb_enc_two"));
        &movups ($inout2,&QWP(0x20,$inp));
        &cmp    ($len,0x40);
        &jb     (&label("ecb_enc_three"));
        &movups ($inout3,&QWP(0x30,$inp));
        &je     (&label("ecb_enc_four"));
        &movups ($inout4,&QWP(0x40,$inp));
        &xorps  ($inout5,$inout5);
        &call   ("_aesni_encrypt6");
        &movups (&QWP(0,$out),$inout0);
        &movups (&QWP(0x10,$out),$inout1);
        &movups (&QWP(0x20,$out),$inout2);
        &movups (&QWP(0x30,$out),$inout3);
        &movups (&QWP(0x40,$out),$inout4);
        jmp     (&label("ecb_ret"));

&set_label("ecb_enc_one",16);
        if ($inline)
        {   &aesni_inline_generate1("enc");     }
        else
        {   &call       ("_aesni_encrypt1");    }
        &movups (&QWP(0,$out),$inout0);
        &jmp    (&label("ecb_ret"));

&set_label("ecb_enc_two",16);
        &xorps  ($inout2,$inout2);
        &call   ("_aesni_encrypt3");
        &movups (&QWP(0,$out),$inout0);
        &movups (&QWP(0x10,$out),$inout1);
        &jmp    (&label("ecb_ret"));

&set_label("ecb_enc_three",16);
        &call   ("_aesni_encrypt3");
        &movups (&QWP(0,$out),$inout0);
        &movups (&QWP(0x10,$out),$inout1);
        &movups (&QWP(0x20,$out),$inout2);
        &jmp    (&label("ecb_ret"));

&set_label("ecb_enc_four",16);
        &call   ("_aesni_encrypt4");
        &movups (&QWP(0,$out),$inout0);
        &movups (&QWP(0x10,$out),$inout1);
        &movups (&QWP(0x20,$out),$inout2);
        &movups (&QWP(0x30,$out),$inout3);
        &jmp    (&label("ecb_ret"));
######################################################################
&set_label("ecb_decrypt",16);
        &mov    ($key_,$key);           # backup $key
        &mov    ($rounds_,$rounds);     # backup $rounds
        &cmp    ($len,0x60);
        &jb     (&label("ecb_dec_tail"));

        &movdqu ($inout0,&QWP(0,$inp));
        &movdqu ($inout1,&QWP(0x10,$inp));
        &movdqu ($inout2,&QWP(0x20,$inp));
        &movdqu ($inout3,&QWP(0x30,$inp));
        &movdqu ($inout4,&QWP(0x40,$inp));
        &movdqu ($inout5,&QWP(0x50,$inp));
        &lea    ($inp,&DWP(0x60,$inp));
        &sub    ($len,0x60);
        &jmp    (&label("ecb_dec_loop6_enter"));
```



```
&set_label("ecb_dec_loop6",16);
        &movups (&QWP(0,$out),$inout0);
        &movdqu ($inout0,&QWP(0,$inp));
        &movups (&QWP(0x10,$out),$inout1);
        &movdqu ($inout1,&QWP(0x10,$inp));
        &movups (&QWP(0x20,$out),$inout2);
        &movdqu ($inout2,&QWP(0x20,$inp));
        &movups (&QWP(0x30,$out),$inout3);
        &movdqu ($inout3,&QWP(0x30,$inp));
        &movups (&QWP(0x40,$out),$inout4);
        &movdqu ($inout4,&QWP(0x40,$inp));
        &movups (&QWP(0x50,$out),$inout5);
        &lea    ($out,&DWP(0x60,$out));
        &movdqu ($inout5,&QWP(0x50,$inp));
        &lea    ($inp,&DWP(0x60,$inp));
&set_label("ecb_dec_loop6_enter");

        &call   ("_aesni_decrypt6");

        &mov    ($key,$key_);           # restore $key
        &mov    ($rounds,$rounds_);     # restore $rounds
        &sub    ($len,0x60);
        &jnc    (&label("ecb_dec_loop6"));

        &movups (&QWP(0,$out),$inout0);
        &movups (&QWP(0x10,$out),$inout1);
        &movups (&QWP(0x20,$out),$inout2);
        &movups (&QWP(0x30,$out),$inout3);
        &movups (&QWP(0x40,$out),$inout4);
        &movups (&QWP(0x50,$out),$inout5);
        &lea    ($out,&DWP(0x60,$out));
        &add    ($len,0x60);
        &jz     (&label("ecb_ret"));

&set_label("ecb_dec_tail");
        &movups ($inout0,&QWP(0,$inp));
        &cmp    ($len,0x20);
        &jb     (&label("ecb_dec_one"));
        &movups ($inout1,&QWP(0x10,$inp));
        &je     (&label("ecb_dec_two"));
        &movups ($inout2,&QWP(0x20,$inp));
        &cmp    ($len,0x40);
        &jb     (&label("ecb_dec_three"));
        &movups ($inout3,&QWP(0x30,$inp));
        &je     (&label("ecb_dec_four"));
        &movups ($inout4,&QWP(0x40,$inp));
        &xorps  ($inout5,$inout5);
        &call   ("_aesni_decrypt6");
        &movups (&QWP(0,$out),$inout0);
        &movups (&QWP(0x10,$out),$inout1);
        &movups (&QWP(0x20,$out),$inout2);
        &movups (&QWP(0x30,$out),$inout3);
        &movups (&QWP(0x40,$out),$inout4);
        &jmp    (&label("ecb_ret"));

&set_label("ecb_dec_one",16);
        if ($inline)
        {   &aesni_inline_generate1("dec");     }
        else
        {   &call       ("_aesni_decrypt1");    }
        &movups (&QWP(0,$out),$inout0);
        &jmp    (&label("ecb_ret"));

&set_label("ecb_dec_two",16);
        &xorps  ($inout2,$inout2);
```

```
548         &call   ("_aesni_decrypt3");
549         &movups (&QWP(0,$out),$inout0);
550         &movups (&QWP(0x10,$out),$inout1);
551         &jmp    (&label("ecb_ret"));

553 &set_label("ecb_dec_three",16);
554         &call   ("_aesni_decrypt3");
555         &movups (&QWP(0,$out),$inout0);
556         &movups (&QWP(0x10,$out),$inout1);
557         &movups (&QWP(0x20,$out),$inout2);
558         &jmp    (&label("ecb_ret"));

560 &set_label("ecb_dec_four",16);
561         &call   ("_aesni_decrypt4");
562         &movups (&QWP(0,$out),$inout0);
563         &movups (&QWP(0x10,$out),$inout1);
564         &movups (&QWP(0x20,$out),$inout2);
565         &movups (&QWP(0x30,$out),$inout3);

567 &set_label("ecb_ret");
568 &function_end("aesni_ecb_encrypt");
```



```
569 ######################################################################
570 # void aesni_ccm64_[en|de]crypt_blocks (const void *in, void *out,
571 #                         size_t blocks, const AES_KEY *key,
572 #                         const char *ivec,char *cmac);
573 #
574 # Handles only complete blocks, operates on 64-bit counter and
575 # does not update *ivec! Nor does it finalize CMAC value
576 # (see engine/eng_aesni.c for details)
577 #
578 { my $cmac=$inout1;
579 &function_begin("aesni_ccm64_encrypt_blocks");
580         &mov    ($inp,&wparam(0));
581         &mov    ($out,&wparam(1));
582         &mov    ($len,&wparam(2));
583         &mov    ($key,&wparam(3));
584         &mov    ($rounds_,&wparam(4));
585         &mov    ($rounds,&wparam(5));
586         &mov    ($key_,"esp");
587         &sub    ("esp",60);
588         &and    ("esp",-16);                    # align stack
589         &mov    (&DWP(48,"esp"),$key_);

591         &movdqu ($ivec,&QWP(0,$rounds_));       # load ivec
592         &movdqu ($cmac,&QWP(0,$rounds));        # load cmac
593         &mov    ($rounds,&DWP(240,$key));

595         # compose byte-swap control mask for pshufb on stack
596         &mov    (&DWP(0,"esp"),0x0c0d0e0f);
597         &mov    (&DWP(4,"esp"),0x08090a0b);
598         &mov    (&DWP(8,"esp"),0x04050607);
599         &mov    (&DWP(12,"esp"),0x00010203);

601         # compose counter increment vector on stack
602         &mov    ($rounds_,1);
603         &xor    ($key_,$key_);
604         &mov    (&DWP(16,"esp"),$rounds_);
605         &mov    (&DWP(20,"esp"),$key_);
606         &mov    (&DWP(24,"esp"),$key_);
607         &mov    (&DWP(28,"esp"),$key_);

609         &shr    ($rounds,1);
610         &lea    ($key_,&DWP(0,$key));
611         &movdqa ($inout3,&QWP(0,"esp"));
612         &movdqa ($inout0,$ivec);
613         &mov    ($rounds_,$rounds);
614         &pshufb ($ivec,$inout3);

616 &set_label("ccm64_enc_outer");
617         &$movekey       ($rndkey0,&QWP(0,$key_));
618         &mov            ($rounds,$rounds_);
619         &movups         ($in0,&QWP(0,$inp));

621         &xorps          ($inout0,$rndkey0);
622         &$movekey       ($rndkey1,&QWP(16,$key_));
623         &xorps          ($rndkey0,$in0);
624         &lea            ($key,&DWP(32,$key_));
625         &xorps          ($cmac,$rndkey0);               # cmac^=inp
626         &$movekey       ($rndkey0,&QWP(0,$key));

628 &set_label("ccm64_enc2_loop");
629         &aesenc         ($inout0,$rndkey1);
630         &dec            ($rounds);
631         &aesenc         ($cmac,$rndkey1);
632         &$movekey       ($rndkey1,&QWP(16,$key));
633         &aesenc         ($inout0,$rndkey0);
634         &lea            ($key,&DWP(32,$key));
```

```
635         &aesenc         ($cmac,$rndkey0);
636         &$movekey       ($rndkey0,&QWP(0,$key));
637         &jnz            (&label("ccm64_enc2_loop"));
638         &aesenc         ($inout0,$rndkey1);
639         &aesenc         ($cmac,$rndkey1);
640         &paddq          ($ivec,&QWP(16,"esp"));
641         &aesenclast     ($inout0,$rndkey0);
642         &aesenclast     ($cmac,$rndkey0);

644         &dec    ($len);
645         &lea    ($inp,&DWP(16,$inp));
646         &xorps  ($in0,$inout0);                 # inp^=E(ivec)
647         &movdqa ($inout0,$ivec);
648         &movups (&QWP(0,$out),$in0);            # save output
649         &lea    ($out,&DWP(16,$out));
650         &pshufb ($inout0,$inout3);
651         &jnz    (&label("ccm64_enc_outer"));

653         &mov    ("esp",&DWP(48,"esp"));
654         &mov    ($out,&wparam(5));
655         &movups (&QWP(0,$out),$cmac);
656 &function_end("aesni_ccm64_encrypt_blocks");

658 &function_begin("aesni_ccm64_decrypt_blocks");
659         &mov    ($inp,&wparam(0));
660         &mov    ($out,&wparam(1));
661         &mov    ($len,&wparam(2));
662         &mov    ($key,&wparam(3));
663         &mov    ($rounds_,&wparam(4));
664         &mov    ($rounds,&wparam(5));
665         &mov    ($key_,"esp");
666         &sub    ("esp",60);
667         &and    ("esp",-16);                    # align stack
668         &mov    (&DWP(48,"esp"),$key_);

670         &movdqu ($ivec,&QWP(0,$rounds_));       # load ivec
671         &movdqu ($cmac,&QWP(0,$rounds));        # load cmac
672         &mov    ($rounds,&DWP(240,$key));

674         # compose byte-swap control mask for pshufb on stack
675         &mov    (&DWP(0,"esp"),0x0c0d0e0f);
676         &mov    (&DWP(4,"esp"),0x08090a0b);
677         &mov    (&DWP(8,"esp"),0x04050607);
678         &mov    (&DWP(12,"esp"),0x00010203);

680         # compose counter increment vector on stack
681         &mov    ($rounds_,1);
682         &xor    ($key_,$key_);
683         &mov    (&DWP(16,"esp"),$rounds_);
684         &mov    (&DWP(20,"esp"),$key_);
685         &mov    (&DWP(24,"esp"),$key_);
686         &mov    (&DWP(28,"esp"),$key_);

688         &movdqa ($inout3,&QWP(0,"esp"));        # bswap mask
689         &movdqa ($inout0,$ivec);

691         &mov    ($key_,$key);
692         &mov    ($rounds_,$rounds);

694         &pshufb ($ivec,$inout3);
695         if ($inline)
696         {   &aesni_inline_generate1("enc");     }
697         else
698         {   &call       ("_aesni_encrypt1");    }
699         &movups ($in0,&QWP(0,$inp));            # load inp
700         &paddq  ($ivec,&QWP(16,"esp"));
```



```
701         &lea    ($inp,&QWP(16,$inp));
702         &jmp    (&label("ccm64_dec_outer"));

704 &set_label("ccm64_dec_outer",16);
705         &xorps  ($in0,$inout0);                 # inp ^= E(ivec)
706         &movdqa ($inout0,$ivec);
707         &mov    ($rounds,$rounds_);
708         &movups (&QWP(0,$out),$in0);            # save output
709         &lea    ($out,&DWP(16,$out));
710         &pshufb ($inout0,$inout3);

712         &sub    ($len,1);
713         &jz     (&label("ccm64_dec_break"));

715         &$movekey       ($rndkey0,&QWP(0,$key_));
716         &shr            ($rounds,1);
717         &$movekey       ($rndkey1,&QWP(16,$key_));
718         &xorps          ($in0,$rndkey0);
719         &lea            ($key,&DWP(32,$key_));
720         &xorps          ($inout0,$rndkey0);
721         &xorps          ($cmac,$in0);           # cmac^=out
722         &$movekey       ($rndkey0,&QWP(0,$key));

724 &set_label("ccm64_dec2_loop");
725         &aesenc         ($inout0,$rndkey1);
726         &dec            ($rounds);
727         &aesenc         ($cmac,$rndkey1);
728         &$movekey       ($rndkey1,&QWP(16,$key));
729         &aesenc         ($inout0,$rndkey0);
730         &lea            ($key,&DWP(32,$key));
731         &aesenc         ($cmac,$rndkey0);
732         &$movekey       ($rndkey0,&QWP(0,$key));
733         &jnz            (&label("ccm64_dec2_loop"));
734         &movups         ($in0,&QWP(0,$inp));    # load inp
735         &paddq          ($ivec,&QWP(16,"esp"));
736         &aesenc         ($inout0,$rndkey1);
737         &aesenc         ($cmac,$rndkey1);
738         &lea            ($inp,&QWP(16,$inp));
739         &aesenclast     ($inout0,$rndkey0);
740         &aesenclast     ($cmac,$rndkey0);
741         &jmp    (&label("ccm64_dec_outer"));

743 &set_label("ccm64_dec_break",16);
744         &mov    ($key,$key_);
745         if ($inline)
746         {   &aesni_inline_generate1("enc",$cmac,$in0);  }
747         else
748         {   &call       ("_aesni_encrypt1",$cmac);      }

750         &mov    ("esp",&DWP(48,"esp"));
751         &mov    ($out,&wparam(5));
752         &movups (&QWP(0,$out),$cmac);
753 &function_end("aesni_ccm64_decrypt_blocks");
754 }
```

```
755 ######################################################################
756 # void aesni_ctr32_encrypt_blocks (const void *in, void *out,
757 #                         size_t blocks, const AES_KEY *key,
758 #                         const char *ivec);
759 #
760 # Handles only complete blocks, operates on 32-bit counter and
761 # does not update *ivec! (see engine/eng_aesni.c for details)
762 #
763 # stack layout:
764 #       0       pshufb mask
765 #       16      vector addend: 0,6,6,6
766 #       32      counter-less ivec
767 #       48      1st triplet of counter vector
768 #       64      2nd triplet of counter vector
769 #       80      saved %esp

771 &function_begin("aesni_ctr32_encrypt_blocks");
772         &mov    ($inp,&wparam(0));
773         &mov    ($out,&wparam(1));
774         &mov    ($len,&wparam(2));
775         &mov    ($key,&wparam(3));
776         &mov    ($rounds_,&wparam(4));
777         &mov    ($key_,"esp");
778         &sub    ("esp",88);
779         &and    ("esp",-16);                    # align stack
780         &mov    (&DWP(80,"esp"),$key_);

782         &cmp    ($len,1);
783         &je     (&label("ctr32_one_shortcut"));

785         &movdqu ($inout5,&QWP(0,$rounds_));     # load ivec

787         # compose byte-swap control mask for pshufb on stack
788         &mov    (&DWP(0,"esp"),0x0c0d0e0f);
789         &mov    (&DWP(4,"esp"),0x08090a0b);
790         &mov    (&DWP(8,"esp"),0x04050607);
791         &mov    (&DWP(12,"esp"),0x00010203);

793         # compose counter increment vector on stack
794         &mov    ($rounds,6);
795         &xor    ($key_,$key_);
796         &mov    (&DWP(16,"esp"),$rounds);
797         &mov    (&DWP(20,"esp"),$rounds);
798         &mov    (&DWP(24,"esp"),$rounds);
799         &mov    (&DWP(28,"esp"),$key_);

801         &pextrd ($rounds_,$inout5,3);           # pull 32-bit counter
802         &pinsrd ($inout5,$key_,3);              # wipe 32-bit counter

804         &mov    ($rounds,&DWP(240,$key));       # key->rounds

806         # compose 2 vectors of 3x32-bit counters
807         &bswap  ($rounds_);
808         &pxor   ($rndkey1,$rndkey1);
809         &pxor   ($rndkey0,$rndkey0);
810         &movdqa ($inout0,&QWP(0,"esp"));        # load byte-swap mask
811         &pinsrd ($rndkey1,$rounds_,0);
812         &lea    ($key_,&DWP(3,$rounds_));
813         &pinsrd ($rndkey0,$key_,0);
814         &inc    ($rounds_);
815         &pinsrd ($rndkey1,$rounds_,1);
816         &inc    ($key_);
817         &pinsrd ($rndkey0,$key_,1);
818         &inc    ($rounds_);
819         &pinsrd ($rndkey1,$rounds_,2);
820         &inc    ($key_);
```



```
821         &pinsrd ($rndkey0,$key_,2);
822         &movdqa (&QWP(48,"esp"),$rndkey1);      # save 1st triplet
823         &pshufb ($rndkey1,$inout0);             # byte swap
824         &movdqa (&QWP(64,"esp"),$rndkey0);      # save 2nd triplet
825         &pshufb ($rndkey0,$inout0);             # byte swap

827         &pshufd ($inout0,$rndkey1,3<<6);        # place counter to upper dword
828         &pshufd ($inout1,$rndkey1,2<<6);
829         &cmp    ($len,6);
830         &jb     (&label("ctr32_tail"));
831         &movdqa (&QWP(32,"esp"),$inout5);       # save counter-less ivec
832         &shr    ($rounds,1);
833         &mov    ($key_,$key);                   # backup $key
834         &mov    ($rounds_,$rounds);             # backup $rounds
835         &sub    ($len,6);
836         &jmp    (&label("ctr32_loop6"));

838 &set_label("ctr32_loop6",16);
839         &pshufd ($inout2,$rndkey1,1<<6);
840         &movdqa ($rndkey1,&QWP(32,"esp"));      # pull counter-less ivec
841         &pshufd ($inout3,$rndkey0,3<<6);
842         &por    ($inout0,$rndkey1);             # merge counter-less ivec
843         &pshufd ($inout4,$rndkey0,2<<6);
844         &por    ($inout1,$rndkey1);
845         &pshufd ($inout5,$rndkey0,1<<6);
846         &por    ($inout2,$rndkey1);
847         &por    ($inout3,$rndkey1);
848         &por    ($inout4,$rndkey1);
849         &por    ($inout5,$rndkey1);

851         # inlining _aesni_encrypt6's prologue gives ~4% improvement...
852         &$movekey       ($rndkey0,&QWP(0,$key_));
853         &$movekey       ($rndkey1,&QWP(16,$key_));
854         &lea            ($key,&DWP(32,$key_));
855         &dec            ($rounds);
856         &pxor           ($inout0,$rndkey0);
857         &pxor           ($inout1,$rndkey0);
858         &aesenc         ($inout0,$rndkey1);
859         &pxor           ($inout2,$rndkey0);
860         &aesenc         ($inout1,$rndkey1);
861         &pxor           ($inout3,$rndkey0);
862         &aesenc         ($inout2,$rndkey1);
863         &pxor           ($inout4,$rndkey0);
864         &aesenc         ($inout3,$rndkey1);
865         &pxor           ($inout5,$rndkey0);
866         &aesenc         ($inout4,$rndkey1);
867         &$movekey       ($rndkey0,&QWP(0,$key));
868         &aesenc         ($inout5,$rndkey1);

870         &call           (&label("_aesni_encrypt6_enter"));

872         &movups ($rndkey1,&QWP(0,$inp));
873         &movups ($rndkey0,&QWP(0x10,$inp));
874         &xorps  ($inout0,$rndkey1);
875         &movups ($rndkey1,&QWP(0x20,$inp));
876         &xorps  ($inout1,$rndkey0);
877         &movups (&QWP(0,$out),$inout0);
878         &movdqa ($rndkey0,&QWP(16,"esp"));      # load increment
879         &xorps  ($inout2,$rndkey1);
880         &movdqa ($rndkey1,&QWP(48,"esp"));      # load 1st triplet
881         &movups (&QWP(0x10,$out),$inout1);
882         &movups (&QWP(0x20,$out),$inout2);

884         &paddd  ($rndkey1,$rndkey0);            # 1st triplet increment
885         &paddd  ($rndkey0,&QWP(64,"esp"));      # 2nd triplet increment
886         &movdqa ($inout0,&QWP(0,"esp"));        # load byte swap mask
```

```
888         &movups ($inout1,&QWP(0x30,$inp));
889         &movups ($inout2,&QWP(0x40,$inp));
890         &xorps  ($inout3,$inout1);
891         &movups ($inout1,&QWP(0x50,$inp));
892         &lea    ($inp,&DWP(0x60,$inp));
893         &movdqa (&QWP(48,"esp"),$rndkey1);      # save 1st triplet
894         &pshufb ($rndkey1,$inout0);             # byte swap
895         &xorps  ($inout4,$inout2);
896         &movups (&QWP(0x30,$out),$inout3);
897         &xorps  ($inout5,$inout1);
898         &movdqa (&QWP(64,"esp"),$rndkey0);      # save 2nd triplet
899         &pshufb ($rndkey0,$inout0);             # byte swap
900         &movups (&QWP(0x40,$out),$inout4);
901         &pshufd ($inout0,$rndkey1,3<<6);
902         &movups (&QWP(0x50,$out),$inout5);
903         &lea    ($out,&DWP(0x60,$out));

905         &mov    ($rounds,$rounds_);
906         &pshufd ($inout1,$rndkey1,2<<6);
907         &sub    ($len,6);
908         &jnc    (&label("ctr32_loop6"));

910         &add    ($len,6);
911         &jz     (&label("ctr32_ret"));
912         &mov    ($key,$key_);
913         &lea    ($rounds,&DWP(1,"",$rounds,2)); # restore $rounds
914         &movdqa ($inout5,&QWP(32,"esp"));       # pull count-less ivec

916 &set_label("ctr32_tail");
917         &por    ($inout0,$inout5);
918         &cmp    ($len,2);
919         &jb     (&label("ctr32_one"));

921         &pshufd ($inout2,$rndkey1,1<<6);
922         &por    ($inout1,$inout5);
923         &je     (&label("ctr32_two"));

925         &pshufd ($inout3,$rndkey0,3<<6);
926         &por    ($inout2,$inout5);
927         &cmp    ($len,4);
928         &jb     (&label("ctr32_three"));

930         &pshufd ($inout4,$rndkey0,2<<6);
931         &por    ($inout3,$inout5);
932         &je     (&label("ctr32_four"));

934         &por    ($inout4,$inout5);
935         &call   ("_aesni_encrypt6");
936         &movups ($rndkey1,&QWP(0,$inp));
937         &movups ($rndkey0,&QWP(0x10,$inp));
938         &xorps  ($inout0,$rndkey1);
939         &movups ($rndkey1,&QWP(0x20,$inp));
940         &xorps  ($inout1,$rndkey0);
941         &movups ($rndkey0,&QWP(0x30,$inp));
942         &xorps  ($inout2,$rndkey1);
943         &movups ($rndkey1,&QWP(0x40,$inp));
944         &xorps  ($inout3,$rndkey0);
945         &movups (&QWP(0,$out),$inout0);
946         &xorps  ($inout4,$rndkey1);
947         &movups (&QWP(0x10,$out),$inout1);
948         &movups (&QWP(0x20,$out),$inout2);
949         &movups (&QWP(0x30,$out),$inout3);
950         &movups (&QWP(0x40,$out),$inout4);
951         &jmp    (&label("ctr32_ret"));
```



```
953 &set_label("ctr32_one_shortcut",16);
954         &movups ($inout0,&QWP(0,$rounds_));     # load ivec
955         &mov    ($rounds,&DWP(240,$key));

957 &set_label("ctr32_one");
958         if ($inline)
959         {   &aesni_inline_generate1("enc");     }
960         else
961         {   &call       ("_aesni_encrypt1");    }
962         &movups ($in0,&QWP(0,$inp));
963         &xorps  ($in0,$inout0);
964         &movups (&QWP(0,$out),$in0);
965         &jmp    (&label("ctr32_ret"));

967 &set_label("ctr32_two",16);
968         &call   ("_aesni_encrypt3");
969         &movups ($inout3,&QWP(0,$inp));
970         &movups ($inout4,&QWP(0x10,$inp));
971         &xorps  ($inout0,$inout3);
972         &xorps  ($inout1,$inout4);
973         &movups (&QWP(0,$out),$inout0);
974         &movups (&QWP(0x10,$out),$inout1);
975         &jmp    (&label("ctr32_ret"));

977 &set_label("ctr32_three",16);
978         &call   ("_aesni_encrypt3");
979         &movups ($inout3,&QWP(0,$inp));
980         &movups ($inout4,&QWP(0x10,$inp));
981         &xorps  ($inout0,$inout3);
982         &movups ($inout5,&QWP(0x20,$inp));
983         &xorps  ($inout1,$inout4);
984         &movups (&QWP(0,$out),$inout0);
985         &xorps  ($inout2,$inout5);
986         &movups (&QWP(0x10,$out),$inout1);
987         &movups (&QWP(0x20,$out),$inout2);
988         &jmp    (&label("ctr32_ret"));

990 &set_label("ctr32_four",16);
991         &call   ("_aesni_encrypt4");
992         &movups ($inout4,&QWP(0,$inp));
993         &movups ($inout5,&QWP(0x10,$inp));
994         &movups ($rndkey1,&QWP(0x20,$inp));
995         &xorps  ($inout0,$inout4);
996         &movups ($rndkey0,&QWP(0x30,$inp));
997         &xorps  ($inout1,$inout5);
998         &movups (&QWP(0,$out),$inout0);
999         &xorps  ($inout2,$rndkey1);
1000        &movups (&QWP(0x10,$out),$inout1);
1001        &xorps  ($inout3,$rndkey0);
1002        &movups (&QWP(0x20,$out),$inout2);
1003        &movups (&QWP(0x30,$out),$inout3);

1005 &set_label("ctr32_ret");
1006        &mov    ("esp",&DWP(80,"esp"));
1007 &function_end("aesni_ctr32_encrypt_blocks");
```

```
1008 ########################################################################
1009 # void aesni_xts_[en|de]crypt(const char *inp,char *out,size_t len,
1010 #       const AES_KEY *key1, const AES_KEY *key2
1011 #       const unsigned char iv[16]);
1012 #
1013 { my ($tweak,$twtmp,$twres,$twmask)=($rndkey1,$rndkey0,$inout0,$inout1);

1015 &function_begin("aesni_xts_encrypt");
1016         &mov    ($key,&wparam(4));              # key2
1017         &mov    ($inp,&wparam(5));              # clear-text tweak

1019         &mov    ($rounds,&DWP(240,$key));       # key2->rounds
1020         &movups ($inout0,&QWP(0,$inp));
1021         if ($inline)
1022         {   &aesni_inline_generate1("enc");     }
1023         else
1024         {   &call       ("_aesni_encrypt1");    }

1026         &mov    ($inp,&wparam(0));
1027         &mov    ($out,&wparam(1));
1028         &mov    ($len,&wparam(2));
1029         &mov    ($key,&wparam(3));              # key1

1031         &mov    ($key_,"esp");
1032         &sub    ("esp",16*7+8);
1033         &mov    ($rounds,&DWP(240,$key));       # key1->rounds
1034         &and    ("esp",-16);                    # align stack

1036         &mov    (&DWP(16*6+0,"esp"),0x87);      # compose the magic constant
1037         &mov    (&DWP(16*6+4,"esp"),0);
1038         &mov    (&DWP(16*6+8,"esp"),1);
1039         &mov    (&DWP(16*6+12,"esp"),0);
1040         &mov    (&DWP(16*7+0,"esp"),$len);      # save original $len
1041         &mov    (&DWP(16*7+4,"esp"),$key_);     # save original %esp

1043         &movdqa ($tweak,$inout0);
1044         &pxor   ($twtmp,$twtmp);
1045         &movdqa ($twmask,&QWP(6*16,"esp"));     # 0x0...010...87
1046         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits

1048         &and    ($len,-16);
1049         &mov    ($key_,$key);                   # backup $key
1050         &mov    ($rounds_,$rounds);             # backup $rounds
1051         &sub    ($len,16*6);
1052         &jc     (&label("xts_enc_short"));

1054         &shr    ($rounds,1);
1055         &mov    ($rounds_,$rounds);
1056         &jmp    (&label("xts_enc_loop6"));

1058 &set_label("xts_enc_loop6",16);
1059         for ($i=0;$i<4;$i++) {
1060             &pshufd     ($twres,$twtmp,0x13);
1061             &pxor       ($twtmp,$twtmp);
1062             &movdqa     (&QWP(16*$i,"esp"),$tweak);
1063             &paddq      ($tweak,$tweak);        # &psllq($tweak,1);
1064             &pand       ($twres,$twmask);       # isolate carry and residue
1065             &pcmpgtd    ($twtmp,$tweak);        # broadcast upper bits
1066             &pxor       ($tweak,$twres);
1067         }
1068         &pshufd ($inout5,$twtmp,0x13);
1069         &movdqa (&QWP(16*$i++,"esp"),$tweak);
1070         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1071          &$movekey      ($rndkey0,&QWP(0,$key_));
1072         &pand   ($inout5,$twmask);              # isolate carry and residue
1073          &movups        ($inout0,&QWP(0,$inp)); # load input
```



```
1074         &pxor   ($inout5,$tweak);

1076         # inline _aesni_encrypt6 prologue and flip xor with tweak and key[0]
1077         &movdqu ($inout1,&QWP(16*1,$inp));
1078          &xorps         ($inout0,$rndkey0);     # input^=rndkey[0]
1079         &movdqu ($inout2,&QWP(16*2,$inp));
1080          &pxor          ($inout1,$rndkey0);
1081         &movdqu ($inout3,&QWP(16*3,$inp));
1082          &pxor          ($inout2,$rndkey0);
1083         &movdqu ($inout4,&QWP(16*4,$inp));
1084          &pxor          ($inout3,$rndkey0);
1085         &movdqu ($rndkey1,&QWP(16*5,$inp));
1086          &pxor          ($inout4,$rndkey0);
1087         &lea    ($inp,&DWP(16*6,$inp));
1088         &pxor   ($inout0,&QWP(16*0,"esp"));     # input^=tweak
1089         &movdqa (&QWP(16*$i,"esp"),$inout5);    # save last tweak
1090         &pxor   ($inout5,$rndkey1);

1092          &$movekey      ($rndkey1,&QWP(16,$key_));
1093          &lea           ($key,&DWP(32,$key_));
1094         &pxor   ($inout1,&QWP(16*1,"esp"));
1095          &aesenc        ($inout0,$rndkey1);
1096         &pxor   ($inout2,&QWP(16*2,"esp"));
1097          &aesenc        ($inout1,$rndkey1);
1098         &pxor   ($inout3,&QWP(16*3,"esp"));
1099          &dec           ($rounds);
1100          &aesenc        ($inout2,$rndkey1);
1101         &pxor   ($inout4,&QWP(16*4,"esp"));
1102          &aesenc        ($inout3,$rndkey1);
1103         &pxor           ($inout5,$rndkey0);
1104          &aesenc        ($inout4,$rndkey1);
1105          &$movekey      ($rndkey0,&QWP(0,$key));
1106          &aesenc        ($inout5,$rndkey1);
1107         &call           (&label("_aesni_encrypt6_enter"));

1109         &movdqa ($tweak,&QWP(16*5,"esp"));      # last tweak
1110        &pxor    ($twtmp,$twtmp);
1111         &xorps  ($inout0,&QWP(16*0,"esp"));     # output^=tweak
1112        &pcmpgtd ($twtmp,$tweak);                # broadcast upper bits
1113         &xorps  ($inout1,&QWP(16*1,"esp"));
1114         &movups (&QWP(16*0,$out),$inout0);      # write output
1115         &xorps  ($inout2,&QWP(16*2,"esp"));
1116         &movups (&QWP(16*1,$out),$inout1);
1117         &xorps  ($inout3,&QWP(16*3,"esp"));
1118         &movups (&QWP(16*2,$out),$inout2);
1119         &xorps  ($inout4,&QWP(16*4,"esp"));
1120         &movups (&QWP(16*3,$out),$inout3);
1121         &xorps  ($inout5,$tweak);
1122         &movups (&QWP(16*4,$out),$inout4);
1123        &pshufd  ($twres,$twtmp,0x13);
1124         &movups (&QWP(16*5,$out),$inout5);
1125         &lea    ($out,&DWP(16*6,$out));
1126        &movdqa  ($twmask,&QWP(16*6,"esp"));     # 0x0...010...87

1128         &pxor   ($twtmp,$twtmp);
1129         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1130         &pand   ($twres,$twmask);               # isolate carry and residue
1131         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1132         &mov    ($rounds,$rounds_);             # restore $rounds
1133         &pxor   ($tweak,$twres);

1135         &sub    ($len,16*6);
1136         &jnc    (&label("xts_enc_loop6"));

1138         &lea    ($rounds,&DWP(1,"",$rounds,2)); # restore $rounds
1139         &mov    ($key,$key_);                   # restore $key
```

```
1140         &mov    ($rounds_,$rounds);

1142 &set_label("xts_enc_short");
1143         &add    ($len,16*6);
1144         &jz     (&label("xts_enc_done6x"));

1146         &movdqa ($inout3,$tweak);               # put aside previous tweak
1147         &cmp    ($len,0x20);
1148         &jb     (&label("xts_enc_one"));

1150         &pshufd ($twres,$twtmp,0x13);
1151         &pxor   ($twtmp,$twtmp);
1152         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1153         &pand   ($twres,$twmask);               # isolate carry and residue
1154         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1155         &pxor   ($tweak,$twres);
1156         &je     (&label("xts_enc_two"));

1158         &pshufd ($twres,$twtmp,0x13);
1159         &pxor   ($twtmp,$twtmp);
1160         &movdqa ($inout4,$tweak);               # put aside previous tweak
1161         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1162         &pand   ($twres,$twmask);               # isolate carry and residue
1163         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1164         &pxor   ($tweak,$twres);
1165         &cmp    ($len,0x40);
1166         &jb     (&label("xts_enc_three"));

1168         &pshufd ($twres,$twtmp,0x13);
1169         &pxor   ($twtmp,$twtmp);
1170         &movdqa ($inout5,$tweak);               # put aside previous tweak
1171         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1172         &pand   ($twres,$twmask);               # isolate carry and residue
1173         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1174         &pxor   ($tweak,$twres);
1175         &movdqa (&QWP(16*0,"esp"),$inout3);
1176         &movdqa (&QWP(16*1,"esp"),$inout4);
1177         &je     (&label("xts_enc_four"));

1179         &movdqa (&QWP(16*2,"esp"),$inout5);
1180         &pshufd ($inout5,$twtmp,0x13);
1181         &movdqa (&QWP(16*3,"esp"),$tweak);
1182         &paddq  ($tweak,$tweak);                # &psllq($inout0,1);
1183         &pand   ($inout5,$twmask);              # isolate carry and residue
1184         &pxor   ($inout5,$tweak);

1186         &movdqu ($inout0,&QWP(16*0,$inp));      # load input
1187         &movdqu ($inout1,&QWP(16*1,$inp));
1188         &movdqu ($inout2,&QWP(16*2,$inp));
1189         &pxor   ($inout0,&QWP(16*0,"esp"));     # input^=tweak
1190         &movdqu ($inout3,&QWP(16*3,$inp));
1191         &pxor   ($inout1,&QWP(16*1,"esp"));
1192         &movdqu ($inout4,&QWP(16*4,$inp));
1193         &pxor   ($inout2,&QWP(16*2,"esp"));
1194         &lea    ($inp,&DWP(16*5,$inp));
1195         &pxor   ($inout3,&QWP(16*3,"esp"));
1196         &movdqa (&QWP(16*4,"esp"),$inout5);     # save last tweak
1197         &pxor   ($inout4,$inout5);

1199         &call   ("_aesni_encrypt6");

1201         &movaps ($tweak,&QWP(16*4,"esp"));      # last tweak
1202         &xorps  ($inout0,&QWP(16*0,"esp"));     # output^=tweak
1203         &xorps  ($inout1,&QWP(16*1,"esp"));
1204         &xorps  ($inout2,&QWP(16*2,"esp"));
1205         &movups (&QWP(16*0,$out),$inout0);      # write output
```



```
1206         &xorps  ($inout3,&QWP(16*3,"esp"));
1207         &movups (&QWP(16*1,$out),$inout1);
1208         &xorps  ($inout4,$tweak);
1209         &movups (&QWP(16*2,$out),$inout2);
1210         &movups (&QWP(16*3,$out),$inout3);
1211         &movups (&QWP(16*4,$out),$inout4);
1212         &lea    ($out,&DWP(16*5,$out));
1213         &jmp    (&label("xts_enc_done"));

1215 &set_label("xts_enc_one",16);
1216         &movups ($inout0,&QWP(16*0,$inp));      # load input
1217         &lea    ($inp,&DWP(16*1,$inp));
1218         &xorps  ($inout0,$inout3);              # input^=tweak
1219         if ($inline)
1220         {   &aesni_inline_generate1("enc");     }
1221         else
1222         {   &call       ("_aesni_encrypt1");    }
1223         &xorps  ($inout0,$inout3);              # output^=tweak
1224         &movups (&QWP(16*0,$out),$inout0);      # write output
1225         &lea    ($out,&DWP(16*1,$out));

1227         &movdqa ($tweak,$inout3);               # last tweak
1228         &jmp    (&label("xts_enc_done"));

1230 &set_label("xts_enc_two",16);
1231         &movaps ($inout4,$tweak);               # put aside last tweak

1233         &movups ($inout0,&QWP(16*0,$inp));      # load input
1234         &movups ($inout1,&QWP(16*1,$inp));
1235         &lea    ($inp,&DWP(16*2,$inp));
1236         &xorps  ($inout0,$inout3);              # input^=tweak
1237         &xorps  ($inout1,$inout4);
1238         &xorps  ($inout2,$inout2);

1240         &call   ("_aesni_encrypt3");

1242         &xorps  ($inout0,$inout3);              # output^=tweak
1243         &xorps  ($inout1,$inout4);
1244         &movups (&QWP(16*0,$out),$inout0);      # write output
1245         &movups (&QWP(16*1,$out),$inout1);
1246         &lea    ($out,&DWP(16*2,$out));

1248         &movdqa ($tweak,$inout4);               # last tweak
1249         &jmp    (&label("xts_enc_done"));

1251 &set_label("xts_enc_three",16);
1252         &movaps ($inout5,$tweak);               # put aside last tweak
1253         &movups ($inout0,&QWP(16*0,$inp));      # load input
1254         &movups ($inout1,&QWP(16*1,$inp));
1255         &movups ($inout2,&QWP(16*2,$inp));
1256         &lea    ($inp,&DWP(16*3,$inp));
1257         &xorps  ($inout0,$inout3);              # input^=tweak
1258         &xorps  ($inout1,$inout4);
1259         &xorps  ($inout2,$inout5);

1261         &call   ("_aesni_encrypt3");

1263         &xorps  ($inout0,$inout3);              # output^=tweak
1264         &xorps  ($inout1,$inout4);
1265         &xorps  ($inout2,$inout5);
1266         &movups (&QWP(16*0,$out),$inout0);      # write output
1267         &movups (&QWP(16*1,$out),$inout1);
1268         &movups (&QWP(16*2,$out),$inout2);
1269         &lea    ($out,&DWP(16*3,$out));

1271         &movdqa ($tweak,$inout5);               # last tweak
```

```
1272         &jmp    (&label("xts_enc_done"));

1274 &set_label("xts_enc_four",16);
1275         &movaps ($inout4,$tweak);               # put aside last tweak

1277         &movups ($inout0,&QWP(16*0,$inp));      # load input
1278         &movups ($inout1,&QWP(16*1,$inp));
1279         &movups ($inout2,&QWP(16*2,$inp));
1280         &xorps  ($inout0,&QWP(16*0,"esp"));     # input^=tweak
1281         &movups ($inout3,&QWP(16*3,$inp));
1282         &lea    ($inp,&DWP(16*4,$inp));
1283         &xorps  ($inout1,&QWP(16*1,"esp"));
1284         &xorps  ($inout2,$inout5);
1285         &xorps  ($inout3,$inout4);

1287         &call   ("_aesni_encrypt4");

1289         &xorps  ($inout0,&QWP(16*0,"esp"));     # output^=tweak
1290         &xorps  ($inout1,&QWP(16*1,"esp"));
1291         &xorps  ($inout2,$inout5);
1292         &movups (&QWP(16*0,$out),$inout0);      # write output
1293         &xorps  ($inout3,$inout4);
1294         &movups (&QWP(16*1,$out),$inout1);
1295         &movups (&QWP(16*2,$out),$inout2);
1296         &movups (&QWP(16*3,$out),$inout3);
1297         &lea    ($out,&DWP(16*4,$out));

1299         &movdqa ($tweak,$inout4);               # last tweak
1300         &jmp    (&label("xts_enc_done"));

1302 &set_label("xts_enc_done6x",16);               # $tweak is pre-calculated
1303         &mov    ($len,&DWP(16*7+0,"esp"));      # restore original $len
1304         &and    ($len,15);
1305         &jz     (&label("xts_enc_ret"));
1306         &movdqa ($inout3,$tweak);
1307         &mov    (&DWP(16*7+0,"esp"),$len);      # save $len%16
1308         &jmp    (&label("xts_enc_steal"));

1310 &set_label("xts_enc_done",16);
1311         &mov    ($len,&DWP(16*7+0,"esp"));      # restore original $len
1312         &pxor   ($twtmp,$twtmp);
1313         &and    ($len,15);
1314         &jz     (&label("xts_enc_ret"));

1316         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1317         &mov    (&DWP(16*7+0,"esp"),$len);      # save $len%16
1318         &pshufd ($inout3,$twtmp,0x13);
1319         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1320         &pand   ($inout3,&QWP(16*6,"esp"));     # isolate carry and residue
1321         &pxor   ($inout3,$tweak);

1323 &set_label("xts_enc_steal");
1324         &movz   ($rounds,&BP(0,$inp));
1325         &movz   ($key,&BP(-16,$out));
1326         &lea    ($inp,&DWP(1,$inp));
1327         &mov    (&BP(-16,$out),&LB($rounds));
1328         &mov    (&BP(0,$out),&LB($key));
1329         &lea    ($out,&DWP(1,$out));
1330         &sub    ($len,1);
1331         &jnz    (&label("xts_enc_steal"));

1333         &sub    ($out,&DWP(16*7+0,"esp"));      # rewind $out
1334         &mov    ($key,$key_);                   # restore $key
1335         &mov    ($rounds,$rounds_);             # restore $rounds

1337         &movups ($inout0,&QWP(-16,$out));       # load input
```



```
1338         &xorps  ($inout0,$inout3);              # input^=tweak
1339         if ($inline)
1340         {   &aesni_inline_generate1("enc");     }
1341         else
1342         {   &call       ("_aesni_encrypt1");    }
1343         &xorps  ($inout0,$inout3);              # output^=tweak
1344         &movups (&QWP(-16,$out),$inout0);       # write output

1346 &set_label("xts_enc_ret");
1347         &mov    ("esp",&DWP(16*7+4,"esp"));     # restore %esp
1348 &function_end("aesni_xts_encrypt");

1350 &function_begin("aesni_xts_decrypt");
1351         &mov    ($key,&wparam(4));              # key2
1352         &mov    ($inp,&wparam(5));              # clear-text tweak

1354         &mov    ($rounds,&DWP(240,$key));       # key2->rounds
1355         &movups ($inout0,&QWP(0,$inp));
1356         if ($inline)
1357         {   &aesni_inline_generate1("enc");     }
1358         else
1359         {   &call       ("_aesni_encrypt1");    }

1361         &mov    ($inp,&wparam(0));
1362         &mov    ($out,&wparam(1));
1363         &mov    ($len,&wparam(2));
1364         &mov    ($key,&wparam(3));              # key1

1366         &mov    ($key_,"esp");
1367         &sub    ("esp",16*7+8);
1368         &and    ("esp",-16);                    # align stack

1370         &xor    ($rounds_,$rounds_);            # if(len%16) len-=16;
1371         &test   ($len,15);
1372         &setnz  (&LB($rounds_));
1373         &shl    ($rounds_,4);
1374         &sub    ($len,$rounds_);

1376         &mov    (&DWP(16*6+0,"esp"),0x87);      # compose the magic constant
1377         &mov    (&DWP(16*6+4,"esp"),0);
1378         &mov    (&DWP(16*6+8,"esp"),1);
1379         &mov    (&DWP(16*6+12,"esp"),0);
1380         &mov    (&DWP(16*7+0,"esp"),$len);      # save original $len
1381         &mov    (&DWP(16*7+4,"esp"),$key_);     # save original %esp

1383         &mov    ($rounds,&DWP(240,$key));       # key1->rounds
1384         &mov    ($key_,$key);                   # backup $key
1385         &mov    ($rounds_,$rounds);             # backup $rounds

1387         &movdqa ($tweak,$inout0);
1388         &pxor   ($twtmp,$twtmp);
1389         &movdqa ($twmask,&QWP(6*16,"esp"));     # 0x0...010...87
1390         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits

1392         &and    ($len,-16);
1393         &sub    ($len,16*6);
1394         &jc     (&label("xts_dec_short"));

1396         &shr    ($rounds,1);
1397         &mov    ($rounds_,$rounds);
1398         &jmp    (&label("xts_dec_loop6"));

1400 &set_label("xts_dec_loop6",16);
1401         for ($i=0;$i<4;$i++) {
1402             &pshufd     ($twres,$twtmp,0x13);
1403             &pxor       ($twtmp,$twtmp);
```

```
1404            &movdqa     (&QWP(16*$i,"esp"),$tweak);
1405            &paddq      ($tweak,$tweak);        # &psllq($tweak,1);
1406            &pand       ($twres,$twmask);       # isolate carry and residue
1407            &pcmpgtd    ($twtmp,$tweak);        # broadcast upper bits
1408            &pxor       ($tweak,$twres);
1409        }
1410        &pshufd ($inout5,$twtmp,0x13);
1411        &movdqa (&QWP(16*$i++,"esp"),$tweak);
1412        &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1413         &$movekey      ($rndkey0,&QWP(0,$key_));
1414        &pand   ($inout5,$twmask);              # isolate carry and residue
1415         &movups        ($inout0,&QWP(0,$inp)); # load input
1416        &pxor   ($inout5,$tweak);

1418        # inline _aesni_encrypt6 prologue and flip xor with tweak and key[0]
1419        &movdqu ($inout1,&QWP(16*1,$inp));
1420         &xorps         ($inout0,$rndkey0);     # input^=rndkey[0]
1421        &movdqu ($inout2,&QWP(16*2,$inp));
1422         &pxor          ($inout1,$rndkey0);
1423        &movdqu ($inout3,&QWP(16*3,$inp));
1424         &pxor          ($inout2,$rndkey0);
1425        &movdqu ($inout4,&QWP(16*4,$inp));
1426         &pxor          ($inout3,$rndkey0);
1427        &movdqu ($rndkey1,&QWP(16*5,$inp));
1428         &pxor          ($inout4,$rndkey0);
1429        &lea    ($inp,&DWP(16*6,$inp));
1430        &pxor   ($inout0,&QWP(16*0,"esp"));     # input^=tweak
1431        &movdqa (&QWP(16*$i,"esp"),$inout5);    # save last tweak
1432        &pxor   ($inout5,$rndkey1);

1434         &$movekey      ($rndkey1,&QWP(16,$key_));
1435         &lea           ($key,&DWP(32,$key_));
1436        &pxor   ($inout1,&QWP(16*1,"esp"));
1437         &aesdec        ($inout0,$rndkey1);
1438        &pxor   ($inout2,&QWP(16*2,"esp"));
1439         &aesdec        ($inout1,$rndkey1);
1440        &pxor   ($inout3,&QWP(16*3,"esp"));
1441         &dec           ($rounds);
1442         &aesdec        ($inout2,$rndkey1);
1443        &pxor   ($inout4,&QWP(16*4,"esp"));
1444         &aesdec        ($inout3,$rndkey1);
1445        &pxor           ($inout5,$rndkey0);
1446         &aesdec        ($inout4,$rndkey1);
1447         &$movekey      ($rndkey0,&QWP(0,$key));
1448         &aesdec        ($inout5,$rndkey1);
1449        &call           (&label("_aesni_decrypt6_enter"));

1451        &movdqa ($tweak,&QWP(16*5,"esp"));      # last tweak
1452       &pxor    ($twtmp,$twtmp);
1453        &xorps  ($inout0,&QWP(16*0,"esp"));     # output^=tweak
1454       &pcmpgtd ($twtmp,$tweak);                # broadcast upper bits
1455        &xorps  ($inout1,&QWP(16*1,"esp"));
1456        &movups (&QWP(16*0,$out),$inout0);      # write output
1457        &xorps  ($inout2,&QWP(16*2,"esp"));
1458        &movups (&QWP(16*1,$out),$inout1);
1459        &xorps  ($inout3,&QWP(16*3,"esp"));
1460        &movups (&QWP(16*2,$out),$inout2);
1461        &xorps  ($inout4,&QWP(16*4,"esp"));
1462        &movups (&QWP(16*3,$out),$inout3);
1463        &xorps  ($inout5,$tweak);
1464        &movups (&QWP(16*4,$out),$inout4);
1465       &pshufd  ($twres,$twtmp,0x13);
1466        &movups (&QWP(16*5,$out),$inout5);
1467        &lea    ($out,&DWP(16*6,$out));
1468       &movdqa  ($twmask,&QWP(16*6,"esp"));     # 0x0...010...87
```



```
1470        &pxor   ($twtmp,$twtmp);
1471        &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1472        &pand   ($twres,$twmask);               # isolate carry and residue
1473        &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1474        &mov    ($rounds,$rounds_);             # restore $rounds
1475        &pxor   ($tweak,$twres);

1477        &sub    ($len,16*6);
1478        &jnc    (&label("xts_dec_loop6"));

1480        &lea    ($rounds,&DWP(1,"",$rounds,2)); # restore $rounds
1481        &mov    ($key,$key_);                   # restore $key
1482        &mov    ($rounds_,$rounds);

1484 &set_label("xts_dec_short");
1485        &add    ($len,16*6);
1486        &jz     (&label("xts_dec_done6x"));

1488        &movdqa ($inout3,$tweak);               # put aside previous tweak
1489        &cmp    ($len,0x20);
1490        &jb     (&label("xts_dec_one"));

1492        &pshufd ($twres,$twtmp,0x13);
1493        &pxor   ($twtmp,$twtmp);
1494        &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1495        &pand   ($twres,$twmask);               # isolate carry and residue
1496        &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1497        &pxor   ($tweak,$twres);
1498        &je     (&label("xts_dec_two"));

1500        &pshufd ($twres,$twtmp,0x13);
1501        &pxor   ($twtmp,$twtmp);
1502        &movdqa ($inout4,$tweak);               # put aside previous tweak
1503        &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1504        &pand   ($twres,$twmask);               # isolate carry and residue
1505        &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1506        &pxor   ($tweak,$twres);
1507        &cmp    ($len,0x40);
1508        &jb     (&label("xts_dec_three"));

1510        &pshufd ($twres,$twtmp,0x13);
1511        &pxor   ($twtmp,$twtmp);
1512        &movdqa ($inout5,$tweak);               # put aside previous tweak
1513        &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1514        &pand   ($twres,$twmask);               # isolate carry and residue
1515        &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1516        &pxor   ($tweak,$twres);
1517        &movdqa (&QWP(16*0,"esp"),$inout3);
1518        &movdqa (&QWP(16*1,"esp"),$inout4);
1519        &je     (&label("xts_dec_four"));

1521        &movdqa (&QWP(16*2,"esp"),$inout5);
1522        &pshufd ($inout5,$twtmp,0x13);
1523        &movdqa (&QWP(16*3,"esp"),$tweak);
1524        &paddq  ($tweak,$tweak);                # &psllq($inout0,1);
1525        &pand   ($inout5,$twmask);              # isolate carry and residue
1526        &pxor   ($inout5,$tweak);

1528        &movdqu ($inout0,&QWP(16*0,$inp));      # load input
1529        &movdqu ($inout1,&QWP(16*1,$inp));
1530        &movdqu ($inout2,&QWP(16*2,$inp));
1531        &pxor   ($inout0,&QWP(16*0,"esp"));     # input^=tweak
1532        &movdqu ($inout3,&QWP(16*3,$inp));
1533        &pxor   ($inout1,&QWP(16*1,"esp"));
1534        &movdqu ($inout4,&QWP(16*4,$inp));
1535        &pxor   ($inout2,&QWP(16*2,"esp"));
```

```
1536         &lea    ($inp,&DWP(16*5,$inp));
1537         &pxor   ($inout3,&QWP(16*3,"esp"));
1538         &movdqa (&QWP(16*4,"esp"),$inout5);     # save last tweak
1539         &pxor   ($inout4,$inout5);

1541         &call   ("_aesni_decrypt6");

1543         &movaps ($tweak,&QWP(16*4,"esp"));      # last tweak
1544         &xorps  ($inout0,&QWP(16*0,"esp"));     # output^=tweak
1545         &xorps  ($inout1,&QWP(16*1,"esp"));
1546         &xorps  ($inout2,&QWP(16*2,"esp"));
1547         &movups (&QWP(16*0,$out),$inout0);      # write output
1548         &xorps  ($inout3,&QWP(16*3,"esp"));
1549         &movups (&QWP(16*1,$out),$inout1);
1550         &xorps  ($inout4,$tweak);
1551         &movups (&QWP(16*2,$out),$inout2);
1552         &movups (&QWP(16*3,$out),$inout3);
1553         &movups (&QWP(16*4,$out),$inout4);
1554         &lea    ($out,&DWP(16*5,$out));
1555         &jmp    (&label("xts_dec_done"));

1557 &set_label("xts_dec_one",16);
1558         &movups ($inout0,&QWP(16*0,$inp));      # load input
1559         &lea    ($inp,&DWP(16*1,$inp));
1560         &xorps  ($inout0,$inout3);              # input^=tweak
1561         if ($inline)
1562         {   &aesni_inline_generate1("dec");     }
1563         else
1564         {   &call       ("_aesni_decrypt1");    }
1565         &xorps  ($inout0,$inout3);              # output^=tweak
1566         &movups (&QWP(16*0,$out),$inout0);      # write output
1567         &lea    ($out,&DWP(16*1,$out));

1569         &movdqa ($tweak,$inout3);               # last tweak
1570         &jmp    (&label("xts_dec_done"));

1572 &set_label("xts_dec_two",16);
1573         &movaps ($inout4,$tweak);               # put aside last tweak

1575         &movups ($inout0,&QWP(16*0,$inp));      # load input
1576         &movups ($inout1,&QWP(16*1,$inp));
1577         &lea    ($inp,&DWP(16*2,$inp));
1578         &xorps  ($inout0,$inout3);              # input^=tweak
1579         &xorps  ($inout1,$inout4);

1581         &call   ("_aesni_decrypt3");

1583         &xorps  ($inout0,$inout3);              # output^=tweak
1584         &xorps  ($inout1,$inout4);
1585         &movups (&QWP(16*0,$out),$inout0);      # write output
1586         &movups (&QWP(16*1,$out),$inout1);
1587         &lea    ($out,&DWP(16*2,$out));

1589         &movdqa ($tweak,$inout4);               # last tweak
1590         &jmp    (&label("xts_dec_done"));

1592 &set_label("xts_dec_three",16);
1593         &movaps ($inout5,$tweak);               # put aside last tweak
1594         &movups ($inout0,&QWP(16*0,$inp));      # load input
1595         &movups ($inout1,&QWP(16*1,$inp));
1596         &movups ($inout2,&QWP(16*2,$inp));
1597         &lea    ($inp,&DWP(16*3,$inp));
1598         &xorps  ($inout0,$inout3);              # input^=tweak
1599         &xorps  ($inout1,$inout4);
1600         &xorps  ($inout2,$inout5);
```



```
1602         &call   ("_aesni_decrypt3");

1604         &xorps  ($inout0,$inout3);              # output^=tweak
1605         &xorps  ($inout1,$inout4);
1606         &xorps  ($inout2,$inout5);
1607         &movups (&QWP(16*0,$out),$inout0);      # write output
1608         &movups (&QWP(16*1,$out),$inout1);
1609         &movups (&QWP(16*2,$out),$inout2);
1610         &lea    ($out,&DWP(16*3,$out));

1612         &movdqa ($tweak,$inout5);               # last tweak
1613         &jmp    (&label("xts_dec_done"));

1615 &set_label("xts_dec_four",16);
1616         &movaps ($inout4,$tweak);               # put aside last tweak

1618         &movups ($inout0,&QWP(16*0,$inp));      # load input
1619         &movups ($inout1,&QWP(16*1,$inp));
1620         &movups ($inout2,&QWP(16*2,$inp));
1621         &xorps  ($inout0,&QWP(16*0,"esp"));     # input^=tweak
1622         &movups ($inout3,&QWP(16*3,$inp));
1623         &lea    ($inp,&DWP(16*4,$inp));
1624         &xorps  ($inout1,&QWP(16*1,"esp"));
1625         &xorps  ($inout2,$inout5);
1626         &xorps  ($inout3,$inout4);

1628         &call   ("_aesni_decrypt4");

1630         &xorps  ($inout0,&QWP(16*0,"esp"));     # output^=tweak
1631         &xorps  ($inout1,&QWP(16*1,"esp"));
1632         &xorps  ($inout2,$inout5);
1633         &movups (&QWP(16*0,$out),$inout0);      # write output
1634         &xorps  ($inout3,$inout4);
1635         &movups (&QWP(16*1,$out),$inout1);
1636         &movups (&QWP(16*2,$out),$inout2);
1637         &movups (&QWP(16*3,$out),$inout3);
1638         &lea    ($out,&DWP(16*4,$out));

1640         &movdqa ($tweak,$inout4);               # last tweak
1641         &jmp    (&label("xts_dec_done"));

1643 &set_label("xts_dec_done6x",16);               # $tweak is pre-calculated
1644         &mov    ($len,&DWP(16*7+0,"esp"));      # restore original $len
1645         &and    ($len,15);
1646         &jz     (&label("xts_dec_ret"));
1647         &mov    (&DWP(16*7+0,"esp"),$len);      # save $len%16
1648         &jmp    (&label("xts_dec_only_one_more"));

1650 &set_label("xts_dec_done",16);
1651         &mov    ($len,&DWP(16*7+0,"esp"));      # restore original $len
1652         &pxor   ($twtmp,$twtmp);
1653         &and    ($len,15);
1654         &jz     (&label("xts_dec_ret"));

1656         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1657         &mov    (&DWP(16*7+0,"esp"),$len);      # save $len%16
1658         &pshufd ($twres,$twtmp,0x13);
1659         &pxor   ($twtmp,$twtmp);
1660         &movdqa ($twmask,&QWP(16*6,"esp"));
1661         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1662         &pand   ($twres,$twmask);               # isolate carry and residue
1663         &pcmpgtd($twtmp,$tweak);                # broadcast upper bits
1664         &pxor   ($tweak,$twres);

1666 &set_label("xts_dec_only_one_more");
1667         &pshufd ($inout3,$twtmp,0x13);
```

```
1668         &movdqa ($inout4,$tweak);               # put aside previous tweak
1669         &paddq  ($tweak,$tweak);                # &psllq($tweak,1);
1670         &pand   ($inout3,$twmask);              # isolate carry and residue
1671         &pxor   ($inout3,$tweak);

1673         &mov    ($key,$key_);                   # restore $key
1674         &mov    ($rounds,$rounds_);             # restore $rounds

1676         &movups ($inout0,&QWP(0,$inp));         # load input
1677         &xorps  ($inout0,$inout3);              # input^=tweak
1678         if ($inline)
1679         {   &aesni_inline_generate1("dec");     }
1680         else
1681         {   &call       ("_aesni_decrypt1");    }
1682         &xorps  ($inout0,$inout3);              # output^=tweak
1683         &movups (&QWP(0,$out),$inout0);         # write output

1685 &set_label("xts_dec_steal");
1686         &movz   ($rounds,&BP(16,$inp));
1687         &movz   ($key,&BP(0,$out));
1688         &lea    ($inp,&DWP(1,$inp));
1689         &mov    (&BP(0,$out),&LB($rounds));
1690         &mov    (&BP(16,$out),&LB($key));
1691         &lea    ($out,&DWP(1,$out));
1692         &sub    ($len,1);
1693         &jnz    (&label("xts_dec_steal"));

1695         &sub    ($out,&DWP(16*7+0,"esp"));      # rewind $out
1696         &mov    ($key,$key_);                   # restore $key
1697         &mov    ($rounds,$rounds_);             # restore $rounds

1699         &movups ($inout0,&QWP(0,$out));         # load input
1700         &xorps  ($inout0,$inout4);              # input^=tweak
1701         if ($inline)
1702         {   &aesni_inline_generate1("dec");     }
1703         else
1704         {   &call       ("_aesni_decrypt1");    }
1705         &xorps  ($inout0,$inout4);              # output^=tweak
1706         &movups (&QWP(0,$out),$inout0);         # write output

1708 &set_label("xts_dec_ret");
1709         &mov    ("esp",&DWP(16*7+4,"esp"));     # restore %esp
1710 &function_end("aesni_xts_decrypt");
1711 }
1712 }
```



```
1713 ######################################################################
1714 # void $PREFIX_cbc_encrypt (const void *inp, void *out,
1715 #                           size_t length, const AES_KEY *key,
1716 #                           unsigned char *ivp,const int enc);
1717 &function_begin("${PREFIX}_cbc_encrypt");
1718         &mov    ($inp,&wparam(0));
1719         &mov    ($rounds_,"esp");
1720         &mov    ($out,&wparam(1));
1721         &sub    ($rounds_,24);
1722         &mov    ($len,&wparam(2));
1723         &and    ($rounds_,-16);
1724         &mov    ($key,&wparam(3));
1725         &mov    ($key_,&wparam(4));
1726         &test   ($len,$len);
1727         &jz     (&label("cbc_abort"));

1729         &cmp    (&wparam(5),0);
1730         &xchg   ($rounds_,"esp");               # alloca
1731         &movups ($ivec,&QWP(0,$key_));          # load IV
1732         &mov    ($rounds,&DWP(240,$key));
1733         &mov    ($key_,$key);                   # backup $key
1734         &mov    (&DWP(16,"esp"),$rounds_);      # save original %esp
1735         &mov    ($rounds_,$rounds);             # backup $rounds
1736         &je     (&label("cbc_decrypt"));

1738         &movaps ($inout0,$ivec);
1739         &cmp    ($len,16);
1740         &jb     (&label("cbc_enc_tail"));
1741         &sub    ($len,16);
1742         &jmp    (&label("cbc_enc_loop"));

1744 &set_label("cbc_enc_loop",16);
1745         &movups ($ivec,&QWP(0,$inp));           # input actually
1746         &lea    ($inp,&DWP(16,$inp));
1747         if ($inline)
1748         {   &aesni_inline_generate1("enc",$inout0,$ivec);       }
1749         else
1750         {   &xorps($inout0,$ivec); &call("_aesni_encrypt1");    }
1751         &mov    ($rounds,$rounds_);     # restore $rounds
1752         &mov    ($key,$key_);           # restore $key
1753         &movups (&QWP(0,$out),$inout0); # store output
1754         &lea    ($out,&DWP(16,$out));
1755         &sub    ($len,16);
1756         &jnc    (&label("cbc_enc_loop"));
1757         &add    ($len,16);
1758         &jnz    (&label("cbc_enc_tail"));
1759         &movaps ($ivec,$inout0);
1760         &jmp    (&label("cbc_ret"));

1762 &set_label("cbc_enc_tail");
1763         &mov    ("ecx",$len);           # zaps $rounds
1764         &data_word(0xA4F3F689);         # rep movsb
1765         &mov    ("ecx",16);             # zero tail
1766         &sub    ("ecx",$len);
1767         &xor    ("eax","eax");          # zaps $len
1768         &data_word(0xAAF3F689);         # rep stosb
1769         &lea    ($out,&DWP(-16,$out));  # rewind $out by 1 block
1770         &mov    ($rounds,$rounds_);     # restore $rounds
1771         &mov    ($inp,$out);            # $inp and $out are the same
1772         &mov    ($key,$key_);           # restore $key
1773         &jmp    (&label("cbc_enc_loop"));
1774 ######################################################################
1775 &set_label("cbc_decrypt",16);
1776         &cmp    ($len,0x50);
1777         &jbe    (&label("cbc_dec_tail"));
1778         &movaps (&QWP(0,"esp"),$ivec);          # save IV
```

```
1779         &sub    ($len,0x50);
1780         &jmp    (&label("cbc_dec_loop6_enter"));

1782 &set_label("cbc_dec_loop6",16);
1783         &movaps (&QWP(0,"esp"),$rndkey0);       # save IV
1784         &movups ($inout5,&QWP(0,$out));
1785         &lea    ($out,&DWP(0x10,$out));
1786 &set_label("cbc_dec_loop6_enter");
1787         &movdqu ($inout0,&QWP(0,$inp));
1788         &movdqu ($inout1,&QWP(0x10,$inp));
1789         &movdqu ($inout2,&QWP(0x20,$inp));
1790         &movdqu ($inout3,&QWP(0x30,$inp));
1791         &movdqu ($inout4,&QWP(0x40,$inp));
1792         &movdqu ($inout5,&QWP(0x50,$inp));

1794         &call   ("_aesni_decrypt6");

1796         &movups ($rndkey1,&QWP(0,$inp));
1797         &movups ($rndkey0,&QWP(0x10,$inp));
1798         &xorps  ($inout0,&QWP(0,"esp"));        # ^=IV
1799         &xorps  ($inout1,$rndkey1);
1800         &movups ($rndkey1,&QWP(0x20,$inp));
1801         &xorps  ($inout2,$rndkey0);
1802         &movups ($rndkey0,&QWP(0x30,$inp));
1803         &xorps  ($inout3,$rndkey1);
1804         &movups ($rndkey1,&QWP(0x40,$inp));
1805         &xorps  ($inout4,$rndkey0);
1806         &movups ($rndkey0,&QWP(0x50,$inp));     # IV
1807         &xorps  ($inout5,$rndkey1);
1808         &movups (&QWP(0,$out),$inout0);
1809         &movups (&QWP(0x10,$out),$inout1);
1810         &lea    ($inp,&DWP(0x60,$inp));
1811         &movups (&QWP(0x20,$out),$inout2);
1812         &mov    ($rounds,$rounds_)              # restore $rounds
1813         &movups (&QWP(0x30,$out),$inout3);
1814         &mov    ($key,$key_);                   # restore $key
1815         &movups (&QWP(0x40,$out),$inout4);
1816         &lea    ($out,&DWP(0x50,$out));
1817         &sub    ($len,0x60);
1818         &ja     (&label("cbc_dec_loop6"));

1820         &movaps ($inout0,$inout5);
1821         &movaps ($ivec,$rndkey0);
1822         &add    ($len,0x50);
1823         &jle    (&label("cbc_dec_tail_collected"));
1824         &movups (&QWP(0,$out),$inout0);
1825         &lea    ($out,&DWP(0x10,$out));
1826 &set_label("cbc_dec_tail");
1827         &movups ($inout0,&QWP(0,$inp));
1828         &movaps ($in0,$inout0);
1829         &cmp    ($len,0x10);
1830         &jbe    (&label("cbc_dec_one"));

1832         &movups ($inout1,&QWP(0x10,$inp));
1833         &movaps ($in1,$inout1);
1834         &cmp    ($len,0x20);
1835         &jbe    (&label("cbc_dec_two"));

1837         &movups ($inout2,&QWP(0x20,$inp));
1838         &cmp    ($len,0x30);
1839         &jbe    (&label("cbc_dec_three"));

1841         &movups ($inout3,&QWP(0x30,$inp));
1842         &cmp    ($len,0x40);
1843         &jbe    (&label("cbc_dec_four"));
```



```
1845         &movups ($inout4,&QWP(0x40,$inp));
1846         &movaps (&QWP(0,"esp"),$ivec);          # save IV
1847         &movups ($inout0,&QWP(0,$inp));
1848         &xorps  ($inout5,$inout5);
1849         &call   ("_aesni_decrypt6");
1850         &movups ($rndkey1,&QWP(0,$inp));
1851         &movups ($rndkey0,&QWP(0x10,$inp));
1852         &xorps  ($inout0,&QWP(0,"esp"));        # ^= IV
1853         &xorps  ($inout1,$rndkey1);
1854         &movups ($rndkey1,&QWP(0x20,$inp));
1855         &xorps  ($inout2,$rndkey0);
1856         &movups ($rndkey0,&QWP(0x30,$inp));
1857         &xorps  ($inout3,$rndkey1);
1858         &movups ($ivec,&QWP(0x40,$inp));        # IV
1859         &xorps  ($inout4,$rndkey0);
1860         &movups (&QWP(0,$out),$inout0);
1861         &movups (&QWP(0x10,$out),$inout1);
1862         &movups (&QWP(0x20,$out),$inout2);
1863         &movups (&QWP(0x30,$out),$inout3);
1864         &lea    ($out,&DWP(0x40,$out));
1865         &movaps ($inout0,$inout4);
1866         &sub    ($len,0x50);
1867         &jmp    (&label("cbc_dec_tail_collected"));

1869 &set_label("cbc_dec_one",16);
1870         if ($inline)
1871         {   &aesni_inline_generate1("dec");     }
1872         else
1873         {   &call       ("_aesni_decrypt1");    }
1874         &xorps  ($inout0,$ivec);
1875         &movaps ($ivec,$in0);
1876         &sub    ($len,0x10);
1877         &jmp    (&label("cbc_dec_tail_collected"));

1879 &set_label("cbc_dec_two",16);
1880         &xorps  ($inout2,$inout2);
1881         &call   ("_aesni_decrypt3");
1882         &xorps  ($inout0,$ivec);
1883         &xorps  ($inout1,$in0);
1884         &movups (&QWP(0,$out),$inout0);
1885         &movaps ($inout0,$inout1);
1886         &lea    ($out,&DWP(0x10,$out));
1887         &movaps ($ivec,$in1);
1888         &sub    ($len,0x20);
1889         &jmp    (&label("cbc_dec_tail_collected"));

1891 &set_label("cbc_dec_three",16);
1892         &call   ("_aesni_decrypt3");
1893         &xorps  ($inout0,$ivec);
1894         &xorps  ($inout1,$in0);
1895         &xorps  ($inout2,$in1);
1896         &movups (&QWP(0,$out),$inout0);
1897         &movaps ($inout0,$inout2);
1898         &movups (&QWP(0x10,$out),$inout1);
1899         &lea    ($out,&DWP(0x20,$out));
1900         &movups ($ivec,&QWP(0x20,$inp));
1901         &sub    ($len,0x30);
1902         &jmp    (&label("cbc_dec_tail_collected"));

1904 &set_label("cbc_dec_four",16);
1905         &call   ("_aesni_decrypt4");
1906         &movups ($rndkey1,&QWP(0x10,$inp));
1907         &movups ($rndkey0,&QWP(0x20,$inp));
1908         &xorps  ($inout0,$ivec);
1909         &movups ($ivec,&QWP(0x30,$inp));
1910         &xorps  ($inout1,$in0);
```

```
1911         &movups (&QWP(0,$out),$inout0);
1912         &xorps  ($inout2,$rndkey1);
1913         &movups (&QWP(0x10,$out),$inout1);
1914         &xorps  ($inout3,$rndkey0);
1915         &movups (&QWP(0x20,$out),$inout2);
1916         &lea    ($out,&DWP(0x30,$out));
1917         &movaps ($inout0,$inout3);
1918         &sub    ($len,0x40);

1920 &set_label("cbc_dec_tail_collected");
1921         &and    ($len,15);
1922         &jnz    (&label("cbc_dec_tail_partial"));
1923         &movups (&QWP(0,$out),$inout0);
1924         &jmp    (&label("cbc_ret"));

1926 &set_label("cbc_dec_tail_partial",16);
1927         &movaps (&QWP(0,"esp"),$inout0);
1928         &mov    ("ecx",16);
1929         &mov    ($inp,"esp");
1930         &sub    ("ecx",$len);
1931         &data_word(0xA4F3F689);         # rep movsb

1933 &set_label("cbc_ret");
1934         &mov    ("esp",&DWP(16,"esp")); # pull original %esp
1935         &mov    ($key_,&wparam(4));
1936         &movups (&QWP(0,$key_),$ivec);  # output IV
1937 &set_label("cbc_abort");
1938 &function_end("${PREFIX}_cbc_encrypt");
```



```
1939 ######################################################################
1940 # Mechanical port from aesni-x86_64.pl.
1941 #
1942 # _aesni_set_encrypt_key is private interface,
1943 # input:
1944 #       "eax"   const unsigned char *userKey
1945 #       $rounds int bits
1946 #       $key    AES_KEY *key
1947 # output:
1948 #       "eax"   return code
1949 #       $round  rounds

1951 &function_begin_B("_aesni_set_encrypt_key");
1952         &test   ("eax","eax");
1953         &jz     (&label("bad_pointer"));
1954         &test   ($key,$key);
1955         &jz     (&label("bad_pointer"));

1957         &movups ("xmm0",&QWP(0,"eax")); # pull first 128 bits of *userKey
1958         &xorps  ("xmm4","xmm4");        # low dword of xmm4 is assumed 0
1959         &lea    ($key,&DWP(16,$key));
1960         &cmp    ($rounds,256);
1961         &je     (&label("14rounds"));
1962         &cmp    ($rounds,192);
1963         &je     (&label("12rounds"));
1964         &cmp    ($rounds,128);
1965         &jne    (&label("bad_keybits"));

1967 &set_label("10rounds",16);
1968         &mov            ($rounds,9);
1969         &$movekey       (&QWP(-16,$key),"xmm0");        # round 0
1970         &aeskeygenassist("xmm1","xmm0",0x01);           # round 1
1971         &call           (&label("key_128_cold"));
1972         &aeskeygenassist("xmm1","xmm0",0x2);            # round 2
1973         &call           (&label("key_128"));
1974         &aeskeygenassist("xmm1","xmm0",0x04);           # round 3
1975         &call           (&label("key_128"));
1976         &aeskeygenassist("xmm1","xmm0",0x08);           # round 4
1977         &call           (&label("key_128"));
1978         &aeskeygenassist("xmm1","xmm0",0x10);           # round 5
1979         &call           (&label("key_128"));
1980         &aeskeygenassist("xmm1","xmm0",0x20);           # round 6
1981         &call           (&label("key_128"));
1982         &aeskeygenassist("xmm1","xmm0",0x40);           # round 7
1983         &call           (&label("key_128"));
1984         &aeskeygenassist("xmm1","xmm0",0x80);           # round 8
1985         &call           (&label("key_128"));
1986         &aeskeygenassist("xmm1","xmm0",0x1b);           # round 9
1987         &call           (&label("key_128"));
1988         &aeskeygenassist("xmm1","xmm0",0x36);           # round 10
1989         &call           (&label("key_128"));
1990         &$movekey       (&QWP(0,$key),"xmm0");
1991         &mov            (&DWP(80,$key),$rounds);
1992         &xor            ("eax","eax");
1993         &ret();

1995 &set_label("key_128",16);
1996         &$movekey       (&QWP(0,$key),"xmm0");
1997         &lea            ($key,&DWP(16,$key));
1998 &set_label("key_128_cold");
1999         &shufps         ("xmm4","xmm0",0b00010000);
2000         &xorps          ("xmm0","xmm4");
2001         &shufps         ("xmm4","xmm0",0b10001100);
2002         &xorps          ("xmm0","xmm4");
2003         &shufps         ("xmm1","xmm1",0b11111111);     # critical path
2004         &xorps          ("xmm0","xmm1");
```

```
2005         &ret();
2006
2007 &set_label("12rounds",16);
2008         &movq           ("xmm2",&QWP(16,"eax"));        # remaining 1/3 of *user
2009         &mov            ($rounds,11);
2010         &$movekey       (&QWP(-16,$key),"xmm0")         # round 0
2011         &aeskeygenassist("xmm1","xmm2",0x01);           # round 1,2
2012         &call           (&label("key_192a_cold"));
2013         &aeskeygenassist("xmm1","xmm2",0x02);           # round 2,3
2014         &call           (&label("key_192b"));
2015         &aeskeygenassist("xmm1","xmm2",0x04);           # round 4,5
2016         &call           (&label("key_192a"));
2017         &aeskeygenassist("xmm1","xmm2",0x08);           # round 5,6
2018         &call           (&label("key_192b"));
2019         &aeskeygenassist("xmm1","xmm2",0x10);           # round 7,8
2020         &call           (&label("key_192a"));
2021         &aeskeygenassist("xmm1","xmm2",0x20);           # round 8,9
2022         &call           (&label("key_192b"));
2023         &aeskeygenassist("xmm1","xmm2",0x40);           # round 10,11
2024         &call           (&label("key_192a"));
2025         &aeskeygenassist("xmm1","xmm2",0x80);           # round 11,12
2026         &call           (&label("key_192b"));
2027         &$movekey       (&QWP(0,$key),"xmm0");
2028         &mov            (&DWP(48,$key),$rounds);
2029         &xor            ("eax","eax");
2030         &ret();
2031
2032 &set_label("key_192a",16);
2033         &$movekey       (&QWP(0,$key),"xmm0");
2034         &lea            ($key,&DWP(16,$key));
2035 &set_label("key_192a_cold",16);
2036         &movaps         ("xmm5","xmm2");
2037 &set_label("key_192b_warm");
2038         &shufps         ("xmm4","xmm0",0b00010000);
2039         &movdqa         ("xmm3","xmm2");
2040         &xorps          ("xmm0","xmm4");
2041         &shufps         ("xmm4","xmm0",0b10001100);
2042         &pslldq         ("xmm3",4);
2043         &xorps          ("xmm0","xmm4");
2044         &pshufd         ("xmm1","xmm1",0b01010101);     # critical path
2045         &pxor           ("xmm2","xmm3");
2046         &pxor           ("xmm0","xmm1");
2047         &pshufd         ("xmm3","xmm0",0b11111111);
2048         &pxor           ("xmm2","xmm3");
2049         &ret();
2050
2051 &set_label("key_192b",16);
2052         &movaps         ("xmm3","xmm0");
2053         &shufps         ("xmm5","xmm0",0b01000100);
2054         &$movekey       (&QWP(0,$key),"xmm5");
2055         &shufps         ("xmm3","xmm2",0b01001110);
2056         &$movekey       (&QWP(16,$key),"xmm3");
2057         &lea            ($key,&DWP(32,$key));
2058         &jmp            (&label("key_192b_warm"));
2059
2060 &set_label("14rounds",16);
2061         &movups         ("xmm2",&QWP(16,"eax"));        # remaining half of *use
2062         &mov            ($rounds,13);
2063         &lea            ($key,&DWP(16,$key));
2064         &$movekey       (&QWP(-32,$key),"xmm0");        # round 0
2065         &$movekey       (&QWP(-16,$key),"xmm2");        # round 1
2066         &aeskeygenassist("xmm1","xmm2",0x01);           # round 2
2067         &call           (&label("key_256a_cold"));
2068         &aeskeygenassist("xmm1","xmm0",0x01);           # round 3
2069         &call           (&label("key_256b"));
2070         &aeskeygenassist("xmm1","xmm2",0x02);           # round 4
```



```
2071         &call           (&label("key_256a"));
2072         &aeskeygenassist("xmm1","xmm0",0x02);           # round 5
2073         &call           (&label("key_256b"));
2074         &aeskeygenassist("xmm1","xmm2",0x04);           # round 6
2075         &call           (&label("key_256a"));
2076         &aeskeygenassist("xmm1","xmm0",0x04);           # round 7
2077         &call           (&label("key_256b"));
2078         &aeskeygenassist("xmm1","xmm2",0x08);           # round 8
2079         &call           (&label("key_256a"));
2080         &aeskeygenassist("xmm1","xmm0",0x08);           # round 9
2081         &call           (&label("key_256b"));
2082         &aeskeygenassist("xmm1","xmm2",0x10);           # round 10
2083         &call           (&label("key_256a"));
2084         &aeskeygenassist("xmm1","xmm0",0x10);           # round 11
2085         &call           (&label("key_256b"));
2086         &aeskeygenassist("xmm1","xmm2",0x20);           # round 12
2087         &call           (&label("key_256a"));
2088         &aeskeygenassist("xmm1","xmm0",0x20);           # round 13
2089         &call           (&label("key_256b"));
2090         &aeskeygenassist("xmm1","xmm2",0x40);           # round 14
2091         &call           (&label("key_256a"));
2092         &$movekey       (&QWP(0,$key),"xmm0");
2093         &mov            (&DWP(16,$key),$rounds);
2094         &xor            ("eax","eax");
2095         &ret();
2096
2097 &set_label("key_256a",16);
2098         &$movekey       (&QWP(0,$key),"xmm2");
2099         &lea            ($key,&DWP(16,$key));
2100 &set_label("key_256a_cold");
2101         &shufps         ("xmm4","xmm0",0b00010000);
2102         &xorps          ("xmm0","xmm4");
2103         &shufps         ("xmm4","xmm0",0b10001100);
2104         &xorps          ("xmm0","xmm4");
2105         &shufps         ("xmm1","xmm1",0b11111111);     # critical path
2106         &xorps          ("xmm0","xmm1");
2107         &ret();
2108
2109 &set_label("key_256b",16);
2110         &$movekey       (&QWP(0,$key),"xmm0");
2111         &lea            ($key,&DWP(16,$key));
2112
2113         &shufps         ("xmm4","xmm2",0b00010000);
2114         &xorps          ("xmm2","xmm4");
2115         &shufps         ("xmm4","xmm2",0b10001100);
2116         &xorps          ("xmm2","xmm4");
2117         &shufps         ("xmm1","xmm1",0b10101010);     # critical path
2118         &xorps          ("xmm2","xmm1");
2119         &ret();
2120
2121 &set_label("bad_pointer",4);
2122         &mov    ("eax",-1);
2123         &ret    ();
2124 &set_label("bad_keybits",4);
2125         &mov    ("eax",-2);
2126         &ret    ();
2127 &function_end_B("_aesni_set_encrypt_key");
2128
2129 # int $PREFIX_set_encrypt_key (const unsigned char *userKey, int bits,
2130 #                              AES_KEY *key)
2131 &function_begin_B("${PREFIX}_set_encrypt_key");
2132         &mov    ("eax",&wparam(0));
2133         &mov    ($rounds,&wparam(1));
2134         &mov    ($key,&wparam(2));
2135         &call   ("_aesni_set_encrypt_key");
2136         &ret    ();
```

```
2137 &function_end_B("${PREFIX}_set_encrypt_key");
2138
2139 # int $PREFIX_set_decrypt_key (const unsigned char *userKey, int bits,
2140 #                              AES_KEY *key)
2141 &function_begin_B("${PREFIX}_set_decrypt_key");
2142         &mov    ("eax",&wparam(0));
2143         &mov    ($rounds,&wparam(1));
2144         &mov    ($key,&wparam(2));
2145         &call   ("_aesni_set_encrypt_key");
2146         &mov    ($key,&wparam(2));
2147         &shl    ($rounds,4)     # rounds-1 after _aesni_set_encrypt_key
2148         &test   ("eax","eax");
2149         &jnz    (&label("dec_key_ret"));
2150         &lea    ("eax",&DWP(16,$key,$rounds));  # end of key schedule
2151
2152         &$movekey       ("xmm0",&QWP(0,$key));  # just swap
2153         &$movekey       ("xmm1",&QWP(0,"eax"));
2154         &$movekey       (&QWP(0,"eax"),"xmm0");
2155         &$movekey       (&QWP(0,$key),"xmm1");
2156         &lea            ($key,&DWP(16,$key));
2157         &lea            ("eax",&DWP(-16,"eax"));
2158
2159 &set_label("dec_key_inverse");
2160         &$movekey       ("xmm0",&QWP(0,$key));  # swap and inverse
2161         &$movekey       ("xmm1",&QWP(0,"eax"));
2162         &aesimc         ("xmm0","xmm0");
2163         &aesimc         ("xmm1","xmm1");
2164         &lea            ($key,&DWP(16,$key));
2165         &lea            ("eax",&DWP(-16,"eax"));
2166         &$movekey       (&QWP(16,"eax"),"xmm0");
2167         &$movekey       (&QWP(-16,$key),"xmm1");
2168         &cmp            ("eax",$key);
2169         &ja             (&label("dec_key_inverse"));
2170
2171         &$movekey       ("xmm0",&QWP(0,$key));  # inverse middle
2172         &aesimc         ("xmm0","xmm0");
2173         &$movekey       (&QWP(0,$key),"xmm0");
2174
2175         &xor            ("eax","eax");          # return success
2176 &set_label("dec_key_ret");
2177         &ret    ();
2178 &function_end_B("${PREFIX}_set_decrypt_key");
2179 &asciz("AES for Intel AES-NI, CRYPTOGAMS by <appro\@openssl.org>");
2180
2181 &asm_finish();
2182 #endif /* ! codereview */
```

```
**********************************************************
   77727 Wed Aug 13 19:53:05 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/aesni-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# This module implements support for Intel AES-NI extension. In
# OpenSSL context it's used with Intel engine, but can also be used as
# drop-in replacement for crypto/aes/asm/aes-x86_64.pl [see below for
# details].
#
# Performance.
#
# Given aes(enc|dec) instructions' latency asymptotic performance for
# non-parallelizable modes such as CBC encrypt is 3.75 cycles per byte
# processed with 128-bit key. And given their throughput asymptotic
# performance for parallelizable modes is 1.25 cycles per byte. Being
# asymptotic limit it's not something you commonly achieve in reality,
# but how close does one get? Below are results collected for
# different modes and block sized. Pairs of numbers are for en-/
# decryption.
#
#       16-byte     64-byte     256-byte    1-KB        8-KB
# ECB   4.25/4.25   1.38/1.38   1.28/1.28   1.26/1.26   1.26/1.26
# CTR   5.42/5.42   1.92/1.92   1.44/1.44   1.28/1.28   1.26/1.26
# CBC   4.38/4.43   4.15/1.43   4.07/1.32   4.07/1.29   4.06/1.28
# CCM   5.66/9.42   4.42/5.41   4.16/4.40   4.09/4.15   4.06/4.07
# OFB   5.42/5.42   4.64/4.64   4.44/4.44   4.39/4.39   4.38/4.38
# CFB   5.73/5.85   5.56/5.62   5.48/5.56   5.47/5.55   5.47/5.55
#
# ECB, CTR, CBC and CCM results are free from EVP overhead. This means
# that otherwise used 'openssl speed -evp aes-128-??? -engine aesni
# [-decrypt]' will exhibit 10-15% worse results for smaller blocks.
# The results were collected with specially crafted speed.c benchmark
# in order to compare them with results reported in "Intel Advanced
# Encryption Standard (AES) New Instruction Set" White Paper Revision
# 3.0 dated May 2010. All above results are consistently better. This
# module also provides better performance for block sizes smaller than
# 128 bytes in points *not* represented in the above table.
#
# Looking at the results for 8-KB buffer.
#
# CFB and OFB results are far from the limit, because implementation
# uses "generic" CRYPTO_[c|o]fb128_encrypt interfaces relying on
# single-block aesni_encrypt, which is not the most optimal way to go.
# CBC encrypt result is unexpectedly high and there is no documented
# explanation for it. Seemingly there is a small penalty for feeding
# the result back to AES unit the way it's done in CBC mode. There is
# nothing one can do and the result appears optimal. CCM result is
# identical to CBC, because CBC-MAC is essentially CBC encrypt without
# saving output. CCM CTR "stays invisible," because it's neatly
# interleaved wih CBC-MAC. This provides ~30% improvement over
# "straghtforward" CCM implementation with CTR and CBC-MAC performed
# disjointly. Parallelizable modes practically achieve the theoretical
# limit.
#
# Looking at how results vary with buffer size.
#
```



```
# Curves are practically saturated at 1-KB buffer size. In most cases
# "256-byte" performance is >95%, and "64-byte" is ~90% of "8-KB" one.
# CTR curve doesn't follow this pattern and is "slowest" changing one
# with "256-byte" result being 87% of "8-KB." This is because overhead
# in CTR mode is most computationally intensive. Small-block CCM
# decrypt is slower than encrypt, because first CTR and last CBC-MAC
# iterations can't be interleaved.
#
# Results for 192- and 256-bit keys.
#
# EVP-free results were observed to scale perfectly with number of
# rounds for larger block sizes, i.e. 192-bit result being 10/12 times
# lower and 256-bit one - 10/14. Well, in CBC encrypt case differences
# are a tad smaller, because the above mentioned penalty biases all
# results by same constant value. In similar way function call
# overhead affects small-block performance, as well as OFB and CFB
# results. Differences are not large, most common coefficients are
# 10/11.7 and 10/13.4 (as opposite to 10/12.0 and 10/14.0), but one
# observe even 10/11.2 and 10/12.4 (CTR, OFB, CFB)...

# January 2011
#
# While Westmere processor features 6 cycles latency for aes[enc|dec]
# instructions, which can be scheduled every second cycle, Sandy
# Bridge spends 8 cycles per instruction, but it can schedule them
# every cycle. This means that code targeting Westmere would perform
# suboptimally on Sandy Bridge. Therefore this update.
#
# In addition, non-parallelizable CBC encrypt (as well as CCM) is
# optimized. Relative improvement might appear modest, 8% on Westmere,
# but in absolute terms it's 3.77 cycles per byte encrypted with
# 128-bit key on Westmere, and 5.07 - on Sandy Bridge. These numbers
# should be compared to asymptotic limits of 3.75 for Westmere and
# 5.00 for Sandy Bridge. Actually, the fact that they get this close
# to asymptotic limits is quite amazing. Indeed, the limit is
# calculated as latency times number of rounds, 10 for 128-bit key,
# and divided by 16, the number of bytes in block, or in other words
# it accounts *solely* for aesenc instructions. But there are extra
# instructions, and numbers so close to the asymptotic limits mean
# that it's as if it takes as little as *one* additional cycle to
# execute all of them. How is it possible? It is possible thanks to
# out-of-order execution logic, which manages to overlap post-
# processing of previous block, things like saving the output, with
# actual encryption of current block, as well as pre-processing of
# current block, things like fetching input and xor-ing it with
# 0-round element of the key schedule, with actual encryption of
# previous block. Keep this in mind...
#
# For parallelizable modes, such as ECB, CBC decrypt, CTR, higher
# performance is achieved by interleaving instructions working on
# independent blocks. In which case asymptotic limit for such modes
# can be obtained by dividing above mentioned numbers by AES
# instructions' interleave factor. Westmere can execute at most 3
# instructions at a time, meaning that optimal interleave factor is 3,
# and that's where the "magic" number of 1.25 come from. "Optimal
# interleave factor" means that increase of interleave factor does
# not improve performance. The formula has proven to reflect reality
# pretty well on Westmere... Sandy Bridge on the other hand can
# execute up to 8 AES instructions at a time, so how does varying
# interleave factor affect the performance? Here is table for ECB
# (numbers are cycles per byte processed with 128-bit key):
#
# instruction interleave factor         3x      6x      8x
# theoretical asymptotic limit          1.67    0.83    0.625
# measured performance for 8KB block    1.05    0.86    0.84
#
```

```
# "as if" interleave factor             4.7x    5.8x    6.0x
#
# Further data for other parallelizable modes:
#
# CBC decrypt                           1.16    0.93    0.93
# CTR                                   1.14    0.91    n/a
#
# Well, given 3x column it’s probably inappropriate to call the limit
# asymptotic, if it can be surpassed, isn’t it? What happens there?
# Rewind to CBC paragraph for the answer. Yes, out-of-order execution
# magic is responsible for this. Processor overlaps not only the
# additional instructions with AES ones, but even AES instuctions
# processing adjacent triplets of independent blocks. In the 6x case
# additional instructions  still claim disproportionally small amount
# of additional cycles, but in 8x case number of instructions must be
# a tad too high for out-of-order logic to cope with, and AES unit
# remains underutilized... As you can see 8x interleave is hardly
# justifiable, so there no need to feel bad that 32-bit aesni-x86.pl
# utilizies 6x interleave because of limited register bank capacity.
#
# Higher interleave factors do have negative impact on Westmere
# performance. While for ECB mode it’s negligible ~1.5%, other
# parallelizables perform ~5% worse, which is outweighed by ~25%
# improvement on Sandy Bridge. To balance regression on Westmere
# CTR mode was implemented with 6x aesenc interleave factor.

# April 2011
#
# Add aesni_xts_[en|de]crypt. Westmere spends 1.33 cycles processing
# one byte out of 8KB with 128-bit key, Sandy Bridge - 0.97. Just like
# in CTR mode AES instruction interleave factor was chosen to be 6x.

$PREFIX="aesni";        # if $PREFIX is set to "AES", the script
                        # generates drop-in replacement for
                        # crypto/aes/asm/aes-x86_64.pl:-)

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can’t locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

$movkey = $PREFIX eq "aesni" ? "movups" : "movups";
@_4args=$win64? ("%rcx","%rdx","%r8", "%r9") :  # Win64 order
                ("%rdi","%rsi","%rdx","%rcx");  # Unix order

$code=".text\n";

$rounds="%eax"; # input to and changed by aesni_[en|de]cryptN !!!
# this is natural Unix argument order for public $PREFIX_[ecb|cbc]_encrypt ...
$inp="%rdi";
$out="%rsi";
$len="%rdx";
$key="%rcx";    # input to and changed by aesni_[en|de]cryptN !!!
$ivp="%r8";     # cbc, ctr, ...

$rnds_="%r10d"; # backup copy for $rounds
$key_="%r11";   # backup copy for $key
```



```
# %xmm register layout
$rndkey0="%xmm0";       $rndkey1="%xmm1";
$inout0="%xmm2";        $inout1="%xmm3";
$inout2="%xmm4";        $inout3="%xmm5";
$inout4="%xmm6";        $inout5="%xmm7";
$inout6="%xmm8";        $inout7="%xmm9";

$in2="%xmm6";           $in1="%xmm7";   # used in CBC decrypt, CTR, ...
$in0="%xmm8";           $iv="%xmm9";
```

```
# Inline version of internal aesni_[en|de]crypt1.
#
# Why folded loop? Because aes[enc|dec] is slow enough to accommodate
# cycles which take care of loop variables...
{ my $sn;
sub aesni_generate1 {
my ($p,$key,$rounds,$inout,$ivec)=@_;   $inout=$inout0 if (!defined($inout));
++$sn;
$code.=<<___;
        $movkey ($key),$rndkey0
        $movkey 16($key),$rndkey1
___
$code.=<<___ if (defined($ivec));
        xorps   $rndkey0,$ivec
        lea     32($key),$key
        xorps   $ivec,$inout
___
$code.=<<___ if (!defined($ivec));
        lea     32($key),$key
        xorps   $rndkey0,$inout
___
$code.=<<___;
.Loop_${p}1_$sn:
        aes${p} $rndkey1,$inout
        dec     $rounds
        $movkey ($key),$rndkey1
        lea     16($key),$key
        jnz     .Loop_${p}1_$sn # loop body is 16 bytes
        aes${p}last     $rndkey1,$inout
___
}}
# void $PREFIX_[en|de]crypt (const void *inp,void *out,const AES_KEY *key);
#
{ my ($inp,$out,$key) = @_4args;

$code.=<<___;
.globl  ${PREFIX}_encrypt
.type   ${PREFIX}_encrypt,\@abi-omnipotent
.align  16
${PREFIX}_encrypt:
        movups  ($inp),$inout0          # load input
        mov     240($key),$rounds       # key->rounds
___
        &aesni_generate1("enc",$key,$rounds);
$code.=<<___;
        movups  $inout0,($out)          # output
        ret
.size   ${PREFIX}_encrypt,.-${PREFIX}_encrypt

.globl  ${PREFIX}_decrypt
.type   ${PREFIX}_decrypt,\@abi-omnipotent
.align  16
${PREFIX}_decrypt:
        movups  ($inp),$inout0          # load input
        mov     240($key),$rounds       # key->rounds
___
        &aesni_generate1("dec",$key,$rounds);
$code.=<<___;
        movups  $inout0,($out)          # output
        ret
.size   ${PREFIX}_decrypt, .-${PREFIX}_decrypt
___
}
```



```
# _aesni_[en|de]cryptN are private interfaces, N denotes interleave
# factor. Why 3x subroutine were originally used in loops? Even though
# aes[enc|dec] latency was originally 6, it could be scheduled only
# every *2nd* cycle. Thus 3x interleave was the one providing optimal
# utilization, i.e. when subroutine's throughput is virtually same as
# of non-interleaved subroutine [for number of input blocks up to 3].
# This is why it makes no sense to implement 2x subroutine.
# aes[enc|dec] latency in next processor generation is 8, but the
# instructions can be scheduled every cycle. Optimal interleave for
# new processor is therefore 8x...
sub aesni_generate3 {
my $dir=shift;
# As already mentioned it takes in $key and $rounds, which are *not*
# preserved. $inout[0-2] is cipher/clear text...
$code.=<<___;
.type   _aesni_${dir}rypt3,\@abi-omnipotent
.align  16
_aesni_${dir}rypt3:
        $movkey ($key),$rndkey0
        shr     \$1,$rounds
        $movkey 16($key),$rndkey1
        lea     32($key),$key
        xorps   $rndkey0,$inout0
        xorps   $rndkey0,$inout1
        xorps   $rndkey0,$inout2
        $movkey         ($key),$rndkey0

.L${dir}_loop3:
        aes${dir}       $rndkey1,$inout0
        aes${dir}       $rndkey1,$inout1
        dec             $rounds
        aes${dir}       $rndkey1,$inout2
        $movkey         16($key),$rndkey1
        aes${dir}       $rndkey0,$inout0
        aes${dir}       $rndkey0,$inout1
        lea             32($key),$key
        aes${dir}       $rndkey0,$inout2
        $movkey         ($key),$rndkey0
        jnz             .L${dir}_loop3

        aes${dir}       $rndkey1,$inout0
        aes${dir}       $rndkey1,$inout1
        aes${dir}       $rndkey1,$inout2
        aes${dir}last   $rndkey0,$inout0
        aes${dir}last   $rndkey0,$inout1
        aes${dir}last   $rndkey0,$inout2
        ret
.size   _aesni_${dir}rypt3,.-_aesni_${dir}rypt3
___
}
# 4x interleave is implemented to improve small block performance,
# most notably [and naturally] 4 block by ~30%. One can argue that one
# should have implemented 5x as well, but improvement would be <20%,
# so it's not worth it...
sub aesni_generate4 {
my $dir=shift;
# As already mentioned it takes in $key and $rounds, which are *not*
# preserved. $inout[0-3] is cipher/clear text...
$code.=<<___;
.type   _aesni_${dir}rypt4,\@abi-omnipotent
.align  16
_aesni_${dir}rypt4:
        $movkey ($key),$rndkey0
        shr     \$1,$rounds
        $movkey 16($key),$rndkey1
        lea     32($key),$key
```

```
333         xorps   $rndkey0,$inout0
334         xorps   $rndkey0,$inout1
335         xorps   $rndkey0,$inout2
336         xorps   $rndkey0,$inout3
337         $movkey ($key),$rndkey0

339 .L${dir}_loop4:
340         aes${dir}       $rndkey1,$inout0
341         aes${dir}       $rndkey1,$inout1
342         dec             $rounds
343         aes${dir}       $rndkey1,$inout2
344         aes${dir}       $rndkey1,$inout3
345         $movkey         16($key),$rndkey1
346         aes${dir}       $rndkey0,$inout0
347         aes${dir}       $rndkey0,$inout1
348         lea             32($key),$key
349         aes${dir}       $rndkey0,$inout2
350         aes${dir}       $rndkey0,$inout3
351         $movkey         ($key),$rndkey0
352         jnz             .L${dir}_loop4

354         aes${dir}       $rndkey1,$inout0
355         aes${dir}       $rndkey1,$inout1
356         aes${dir}       $rndkey1,$inout2
357         aes${dir}       $rndkey1,$inout3
358         aes${dir}last   $rndkey0,$inout0
359         aes${dir}last   $rndkey0,$inout1
360         aes${dir}last   $rndkey0,$inout2
361         aes${dir}last   $rndkey0,$inout3
362         ret
363 .size   _aesni_${dir}rypt4,.-_aesni_${dir}rypt4
364 ___
365 }
366 sub aesni_generate6 {
367 my $dir=shift;
368 # As already mentioned it takes in $key and $rounds, which are *not*
369 # preserved. $inout[0-5] is cipher/clear text...
370 $code.=<<___;
371 .type   _aesni_${dir}rypt6,\@abi-omnipotent
372 .align  16
373 _aesni_${dir}rypt6:
374         $movkey         ($key),$rndkey0
375         shr             \$1,$rounds
376         $movkey         16($key),$rndkey1
377         lea             32($key),$key
378         xorps           $rndkey0,$inout0
379         pxor            $rndkey0,$inout1
380         aes${dir}       $rndkey1,$inout0
381         pxor            $rndkey0,$inout2
382         aes${dir}       $rndkey1,$inout1
383         pxor            $rndkey0,$inout3
384         aes${dir}       $rndkey1,$inout2
385         pxor            $rndkey0,$inout4
386         aes${dir}       $rndkey1,$inout3
387         pxor            $rndkey0,$inout5
388         dec             $rounds
389         aes${dir}       $rndkey1,$inout4
390         $movkey         ($key),$rndkey0
391         aes${dir}       $rndkey1,$inout5
392         jmp             .L${dir}_loop6_enter
393 .align  16
394 .L${dir}_loop6:
395         aes${dir}       $rndkey1,$inout0
396         aes${dir}       $rndkey1,$inout1
397         dec             $rounds
398         aes${dir}       $rndkey1,$inout2
```



```
399         aes${dir}       $rndkey1,$inout3
400         aes${dir}       $rndkey1,$inout4
401         aes${dir}       $rndkey1,$inout5
402 .L${dir}_loop6_enter:                           # happens to be 16-byte aligned
403         $movkey         16($key),$rndkey1
404         aes${dir}       $rndkey0,$inout0
405         aes${dir}       $rndkey0,$inout1
406         lea             32($key),$key
407         aes${dir}       $rndkey0,$inout2
408         aes${dir}       $rndkey0,$inout3
409         aes${dir}       $rndkey0,$inout4
410         aes${dir}       $rndkey0,$inout5
411         $movkey         ($key),$rndkey0
412         jnz             .L${dir}_loop6

414         aes${dir}       $rndkey1,$inout0
415         aes${dir}       $rndkey1,$inout1
416         aes${dir}       $rndkey1,$inout2
417         aes${dir}       $rndkey1,$inout3
418         aes${dir}       $rndkey1,$inout4
419         aes${dir}       $rndkey1,$inout5
420         aes${dir}last   $rndkey0,$inout0
421         aes${dir}last   $rndkey0,$inout1
422         aes${dir}last   $rndkey0,$inout2
423         aes${dir}last   $rndkey0,$inout3
424         aes${dir}last   $rndkey0,$inout4
425         aes${dir}last   $rndkey0,$inout5
426         ret
427 .size   _aesni_${dir}rypt6,.-_aesni_${dir}rypt6
428 ___
429 }
430 sub aesni_generate8 {
431 my $dir=shift;
432 # As already mentioned it takes in $key and $rounds, which are *not*
433 # preserved. $inout[0-7] is cipher/clear text...
434 $code.=<<___;
435 .type   _aesni_${dir}rypt8,\@abi-omnipotent
436 .align  16
437 _aesni_${dir}rypt8:
438         $movkey         ($key),$rndkey0
439         shr             \$1,$rounds
440         $movkey         16($key),$rndkey1
441         lea             32($key),$key
442         xorps           $rndkey0,$inout0
443         xorps           $rndkey0,$inout1
444         aes${dir}       $rndkey1,$inout0
445         pxor            $rndkey0,$inout2
446         aes${dir}       $rndkey1,$inout1
447         pxor            $rndkey0,$inout3
448         aes${dir}       $rndkey1,$inout2
449         pxor            $rndkey0,$inout4
450         aes${dir}       $rndkey1,$inout3
451         pxor            $rndkey0,$inout5
452         dec             $rounds
453         aes${dir}       $rndkey1,$inout4
454         pxor            $rndkey0,$inout6
455         aes${dir}       $rndkey1,$inout5
456         pxor            $rndkey0,$inout7
457         $movkey         ($key),$rndkey0
458         aes${dir}       $rndkey1,$inout6
459         aes${dir}       $rndkey1,$inout7
460         $movkey         16($key),$rndkey1
461         jmp             .L${dir}_loop8_enter
462 .align  16
463 .L${dir}_loop8:
464         aes${dir}       $rndkey1,$inout0
```

```
465         aes${dir}       $rndkey1,$inout1
466         dec             $rounds
467         aes${dir}       $rndkey1,$inout2
468         aes${dir}       $rndkey1,$inout3
469         aes${dir}       $rndkey1,$inout4
470         aes${dir}       $rndkey1,$inout5
471         aes${dir}       $rndkey1,$inout6
472         aes${dir}       $rndkey1,$inout7
473         $movkey         16($key),$rndkey1
474 .L${dir}_loop8_enter:                           # happens to be 16-byte aligned
475         aes${dir}       $rndkey0,$inout0
476         aes${dir}       $rndkey0,$inout1
477         lea             32($key),$key
478         aes${dir}       $rndkey0,$inout2
479         aes${dir}       $rndkey0,$inout3
480         aes${dir}       $rndkey0,$inout4
481         aes${dir}       $rndkey0,$inout5
482         aes${dir}       $rndkey0,$inout6
483         aes${dir}       $rndkey0,$inout7
484         $movkey         ($key),$rndkey0
485         jnz             .L${dir}_loop8

487         aes${dir}       $rndkey1,$inout0
488         aes${dir}       $rndkey1,$inout1
489         aes${dir}       $rndkey1,$inout2
490         aes${dir}       $rndkey1,$inout3
491         aes${dir}       $rndkey1,$inout4
492         aes${dir}       $rndkey1,$inout5
493         aes${dir}       $rndkey1,$inout6
494         aes${dir}       $rndkey1,$inout7
495         aes${dir}last   $rndkey0,$inout0
496         aes${dir}last   $rndkey0,$inout1
497         aes${dir}last   $rndkey0,$inout2
498         aes${dir}last   $rndkey0,$inout3
499         aes${dir}last   $rndkey0,$inout4
500         aes${dir}last   $rndkey0,$inout5
501         aes${dir}last   $rndkey0,$inout6
502         aes${dir}last   $rndkey0,$inout7
503         ret
504 .size   _aesni_${dir}rypt8,.-_aesni_${dir}rypt8
505 ___
506 }
507 &aesni_generate3("enc") if ($PREFIX eq "aesni");
508 &aesni_generate3("dec");
509 &aesni_generate4("enc") if ($PREFIX eq "aesni");
510 &aesni_generate4("dec");
511 &aesni_generate6("enc") if ($PREFIX eq "aesni");
512 &aesni_generate6("dec");
513 &aesni_generate8("enc") if ($PREFIX eq "aesni");
514 &aesni_generate8("dec");
```



```
515 if ($PREFIX eq "aesni") {
516 ########################################################################
517 # void aesni_ecb_encrypt (const void *in, void *out,
518 #                         size_t length, const AES_KEY *key,
519 #                         int enc);
520 $code.=<<___;
521 .globl  aesni_ecb_encrypt
522 .type   aesni_ecb_encrypt,\@function,5
523 .align  16
524 aesni_ecb_encrypt:
525         and     \$-16,$len
526         jz      .Lecb_ret

528         mov     240($key),$rounds       # key->rounds
529         $movkey ($key),$rndkey0
530         mov     $key,$key_              # backup $key
531         mov     $rounds,$rnds_          # backup $rounds
532         test    %r8d,%r8d               # 5th argument
533         jz      .Lecb_decrypt
534 #---------------------------- ECB ENCRYPT -----------------------------#
535         cmp     \$0x80,$len
536         jb      .Lecb_enc_tail

538         movdqu  ($inp),$inout0
539         movdqu  0x10($inp),$inout1
540         movdqu  0x20($inp),$inout2
541         movdqu  0x30($inp),$inout3
542         movdqu  0x40($inp),$inout4
543         movdqu  0x50($inp),$inout5
544         movdqu  0x60($inp),$inout6
545         movdqu  0x70($inp),$inout7
546         lea     0x80($inp),$inp
547         sub     \$0x80,$len
548         jmp     .Lecb_enc_loop8_enter
549 .align 16
550 .Lecb_enc_loop8:
551         movups  $inout0,($out)
552         mov     $key_,$key              # restore $key
553         movdqu  ($inp),$inout0
554         mov     $rnds_,$rounds          # restore $rounds
555         movups  $inout1,0x10($out)
556         movdqu  0x10($inp),$inout1
557         movups  $inout2,0x20($out)
558         movdqu  0x20($inp),$inout2
559         movups  $inout3,0x30($out)
560         movdqu  0x30($inp),$inout3
561         movups  $inout4,0x40($out)
562         movdqu  0x40($inp),$inout4
563         movups  $inout5,0x50($out)
564         movdqu  0x50($inp),$inout5
565         movups  $inout6,0x60($out)
566         movdqu  0x60($inp),$inout6
567         movups  $inout7,0x70($out)
568         lea     0x80($out),$out
569         movdqu  0x70($inp),$inout7
570         lea     0x80($inp),$inp
571 .Lecb_enc_loop8_enter:

573         call    _aesni_encrypt8

575         sub     \$0x80,$len
576         jnc     .Lecb_enc_loop8

578         movups  $inout0,($out)
579         mov     $key_,$key              # restore $key
580         movups  $inout1,0x10($out)
```

```
581         mov     $rnds_,$rounds          # restore $rounds
582         movups  $inout2,0x20($out)
583         movups  $inout3,0x30($out)
584         movups  $inout4,0x40($out)
585         movups  $inout5,0x50($out)
586         movups  $inout6,0x60($out)
587         movups  $inout7,0x70($out)
588         lea     0x80($out),$out
589         add     \$0x80,$len
590         jz      .Lecb_ret

592 .Lecb_enc_tail:
593         movups  ($inp),$inout0
594         cmp     \$0x20,$len
595         jb      .Lecb_enc_one
596         movups  0x10($inp),$inout1
597         je      .Lecb_enc_two
598         movups  0x20($inp),$inout2
599         cmp     \$0x40,$len
600         jb      .Lecb_enc_three
601         movups  0x30($inp),$inout3
602         je      .Lecb_enc_four
603         movups  0x40($inp),$inout4
604         cmp     \$0x60,$len
605         jb      .Lecb_enc_five
606         movups  0x50($inp),$inout5
607         je      .Lecb_enc_six
608         movdqu  0x60($inp),$inout6
609         call    _aesni_encrypt8
610         movups  $inout0,($out)
611         movups  $inout1,0x10($out)
612         movups  $inout2,0x20($out)
613         movups  $inout3,0x30($out)
614         movups  $inout4,0x40($out)
615         movups  $inout5,0x50($out)
616         movups  $inout6,0x60($out)
617         jmp     .Lecb_ret
618 .align  16
619 .Lecb_enc_one:
620 ___
621         &aesni_generate1("enc",$key,$rounds);
622 $code.=<<___;
623         movups  $inout0,($out)
624         jmp     .Lecb_ret
625 .align  16
626 .Lecb_enc_two:
627         xorps   $inout2,$inout2
628         call    _aesni_encrypt3
629         movups  $inout0,($out)
630         movups  $inout1,0x10($out)
631         jmp     .Lecb_ret
632 .align  16
633 .Lecb_enc_three:
634         call    _aesni_encrypt3
635         movups  $inout0,($out)
636         movups  $inout1,0x10($out)
637         movups  $inout2,0x20($out)
638         jmp     .Lecb_ret
639 .align  16
640 .Lecb_enc_four:
641         call    _aesni_encrypt4
642         movups  $inout0,($out)
643         movups  $inout1,0x10($out)
644         movups  $inout2,0x20($out)
645         movups  $inout3,0x30($out)
646         jmp     .Lecb_ret
```



```
647 .align  16
648 .Lecb_enc_five:
649         xorps   $inout5,$inout5
650         call    _aesni_encrypt6
651         movups  $inout0,($out)
652         movups  $inout1,0x10($out)
653         movups  $inout2,0x20($out)
654         movups  $inout3,0x30($out)
655         movups  $inout4,0x40($out)
656         jmp     .Lecb_ret
657 .align  16
658 .Lecb_enc_six:
659         call    _aesni_encrypt6
660         movups  $inout0,($out)
661         movups  $inout1,0x10($out)
662         movups  $inout2,0x20($out)
663         movups  $inout3,0x30($out)
664         movups  $inout4,0x40($out)
665         movups  $inout5,0x50($out)
666         jmp     .Lecb_ret
```

```
667 #--------------------------- ECB DECRYPT ------------------------------#
668 .align  16
669 .Lecb_decrypt:
670         cmp     \$0x80,$len
671         jb      .Lecb_dec_tail

673         movdqu  ($inp),$inout0
674         movdqu  0x10($inp),$inout1
675         movdqu  0x20($inp),$inout2
676         movdqu  0x30($inp),$inout3
677         movdqu  0x40($inp),$inout4
678         movdqu  0x50($inp),$inout5
679         movdqu  0x60($inp),$inout6
680         movdqu  0x70($inp),$inout7
681         lea     0x80($inp),$inp
682         sub     \$0x80,$len
683         jmp     .Lecb_dec_loop8_enter
684 .align 16
685 .Lecb_dec_loop8:
686         movups  $inout0,($out)
687         mov     $key_,$key              # restore $key
688         movdqu  ($inp),$inout0
689         mov     $rnds_,$rounds          # restore $rounds
690         movups  $inout1,0x10($out)
691         movdqu  0x10($inp),$inout1
692         movups  $inout2,0x20($out)
693         movdqu  0x20($inp),$inout2
694         movups  $inout3,0x30($out)
695         movdqu  0x30($inp),$inout3
696         movups  $inout4,0x40($out)
697         movdqu  0x40($inp),$inout4
698         movups  $inout5,0x50($out)
699         movdqu  0x50($inp),$inout5
700         movups  $inout6,0x60($out)
701         movdqu  0x60($inp),$inout6
702         movups  $inout7,0x70($out)
703         lea     0x80($out),$out
704         movdqu  0x70($inp),$inout7
705         lea     0x80($inp),$inp
706 .Lecb_dec_loop8_enter:

708         call    _aesni_decrypt8

710         $movkey ($key_),$rndkey0
711         sub     \$0x80,$len
712         jnc     .Lecb_dec_loop8

714         movups  $inout0,($out)
715         mov     $key_,$key              # restore $key
716         movups  $inout1,0x10($out)
717         mov     $rnds_,$rounds          # restore $rounds
718         movups  $inout2,0x20($out)
719         movups  $inout3,0x30($out)
720         movups  $inout4,0x40($out)
721         movups  $inout5,0x50($out)
722         movups  $inout6,0x60($out)
723         movups  $inout7,0x70($out)
724         lea     0x80($out),$out
725         add     \$0x80,$len
726         jz      .Lecb_ret

728 .Lecb_dec_tail:
729         movups  ($inp),$inout0
730         cmp     \$0x20,$len
731         jb      .Lecb_dec_one
732         movups  0x10($inp),$inout1
```



```
733         je      .Lecb_dec_two
734         movups  0x20($inp),$inout2
735         cmp     \$0x40,$len
736         jb      .Lecb_dec_three
737         movups  0x30($inp),$inout3
738         je      .Lecb_dec_four
739         movups  0x40($inp),$inout4
740         cmp     \$0x60,$len
741         jb      .Lecb_dec_five
742         movups  0x50($inp),$inout5
743         je      .Lecb_dec_six
744         movups  0x60($inp),$inout6
745         $movkey ($key),$rndkey0
746         call    _aesni_decrypt8
747         movups  $inout0,($out)
748         movups  $inout1,0x10($out)
749         movups  $inout2,0x20($out)
750         movups  $inout3,0x30($out)
751         movups  $inout4,0x40($out)
752         movups  $inout5,0x50($out)
753         movups  $inout6,0x60($out)
754         jmp     .Lecb_ret
755 .align  16
756 .Lecb_dec_one:
757 ___
758         &aesni_generate1("dec",$key,$rounds);
759 $code.=<<___;
760         movups  $inout0,($out)
761         jmp     .Lecb_ret
762 .align  16
763 .Lecb_dec_two:
764         xorps   $inout2,$inout2
765         call    _aesni_decrypt3
766         movups  $inout0,($out)
767         movups  $inout1,0x10($out)
768         jmp     .Lecb_ret
769 .align  16
770 .Lecb_dec_three:
771         call    _aesni_decrypt3
772         movups  $inout0,($out)
773         movups  $inout1,0x10($out)
774         movups  $inout2,0x20($out)
775         jmp     .Lecb_ret
776 .align  16
777 .Lecb_dec_four:
778         call    _aesni_decrypt4
779         movups  $inout0,($out)
780         movups  $inout1,0x10($out)
781         movups  $inout2,0x20($out)
782         movups  $inout3,0x30($out)
783         jmp     .Lecb_ret
784 .align  16
785 .Lecb_dec_five:
786         xorps   $inout5,$inout5
787         call    _aesni_decrypt6
788         movups  $inout0,($out)
789         movups  $inout1,0x10($out)
790         movups  $inout2,0x20($out)
791         movups  $inout3,0x30($out)
792         movups  $inout4,0x40($out)
793         jmp     .Lecb_ret
794 .align  16
795 .Lecb_dec_six:
796         call    _aesni_decrypt6
797         movups  $inout0,($out)
798         movups  $inout1,0x10($out)
```

```
799         movups  $inout2,0x20($out)
800         movups  $inout3,0x30($out)
801         movups  $inout4,0x40($out)
802         movups  $inout5,0x50($out)

804 .Lecb_ret:
805         ret
806 .size   aesni_ecb_encrypt,.-aesni_ecb_encrypt
807 ___
```



```
808 {
809 ######################################################################
810 # void aesni_ccm64_[en|de]crypt_blocks (const void *in, void *out,
811 #                         size_t blocks, const AES_KEY *key,
812 #                         const char *ivec,char *cmac);
813 #
814 # Handles only complete blocks, operates on 64-bit counter and
815 # does not update *ivec! Nor does it finalize CMAC value
816 # (see engine/eng_aesni.c for details)
817 #
818 {
819 my $cmac="%r9"; # 6th argument

821 my $increment="%xmm6";
822 my $bswap_mask="%xmm7";

824 $code.=<<___;
825 .globl  aesni_ccm64_encrypt_blocks
826 .type   aesni_ccm64_encrypt_blocks,\@function,6
827 .align  16
828 aesni_ccm64_encrypt_blocks:
829 ___
830 $code.=<<___ if ($win64);
831         lea     -0x58(%rsp),%rsp
832         movaps  %xmm6,(%rsp)
833         movaps  %xmm7,0x10(%rsp)
834         movaps  %xmm8,0x20(%rsp)
835         movaps  %xmm9,0x30(%rsp)
836 .Lccm64_enc_body:
837 ___
838 $code.=<<___;
839         mov     240($key),$rounds               # key->rounds
840         movdqu  ($ivp),$iv
841         movdqa  .Lincrement64(%rip),$increment
842         movdqa  .Lbswap_mask(%rip),$bswap_mask

844         shr     \$1,$rounds
845         lea     0($key),$key_
846         movdqu  ($cmac),$inout1
847         movdqa  $iv,$inout0
848         mov     $rounds,$rnds_
849         pshufb  $bswap_mask,$iv
850         jmp     .Lccm64_enc_outer
851 .align  16
852 .Lccm64_enc_outer:
853         $movkey ($key_),$rndkey0
854         mov     $rnds_,$rounds
855         movups  ($inp),$in0                     # load inp

857         xorps   $rndkey0,$inout0                # counter
858         $movkey 16($key_),$rndkey1
859         xorps   $in0,$rndkey0
860         lea     32($key_),$key
861         xorps   $rndkey0,$inout1                # cmac^=inp
862         $movkey ($key),$rndkey0

864 .Lccm64_enc2_loop:
865         aesenc  $rndkey1,$inout0
866         dec     $rounds
867         aesenc  $rndkey1,$inout1
868         $movkey 16($key),$rndkey1
869         aesenc  $rndkey0,$inout0
870         lea     32($key),$key
871         aesenc  $rndkey0,$inout1
872         $movkey 0($key),$rndkey0
873         jnz     .Lccm64_enc2_loop
```

```
874         aesenc  $rndkey1,$inout0
875         aesenc  $rndkey1,$inout1
876         paddq   $increment,$iv
877         aesenclast      $rndkey0,$inout0
878         aesenclast      $rndkey0,$inout1

880         dec     $len
881         lea     16($inp),$inp
882         xorps   $inout0,$in0                    # inp ^= E(iv)
883         movdqa  $iv,$inout0
884         movups  $in0,($out)                     # save output
885         lea     16($out),$out
886         pshufb  $bswap_mask,$inout0
887         jnz     .Lccm64_enc_outer

889         movups  $inout1,($cmac)
890 ___
891 $code.=<<___ if ($win64);
892         movaps  (%rsp),%xmm6
893         movaps  0x10(%rsp),%xmm7
894         movaps  0x20(%rsp),%xmm8
895         movaps  0x30(%rsp),%xmm9
896         lea     0x58(%rsp),%rsp
897 .Lccm64_enc_ret:
898 ___
899 $code.=<<___;
900         ret
901 .size   aesni_ccm64_encrypt_blocks,.-aesni_ccm64_encrypt_blocks
902 ___
903 ######################################################################
904 $code.=<<___;
905 .globl  aesni_ccm64_decrypt_blocks
906 .type   aesni_ccm64_decrypt_blocks,\@function,6
907 .align  16
908 aesni_ccm64_decrypt_blocks:
909 ___
910 $code.=<<___ if ($win64);
911         lea     -0x58(%rsp),%rsp
912         movaps  %xmm6,(%rsp)
913         movaps  %xmm7,0x10(%rsp)
914         movaps  %xmm8,0x20(%rsp)
915         movaps  %xmm9,0x30(%rsp)
916 .Lccm64_dec_body:
917 ___
918 $code.=<<___;
919         mov     240($key),$rounds               # key->rounds
920         movups  ($ivp),$iv
921         movdqu  ($cmac),$inout1
922         movdqa  .Lincrement64(%rip),$increment
923         movdqa  .Lbswap_mask(%rip),$bswap_mask

925         movaps  $iv,$inout0
926         mov     $rounds,$rnds_
927         mov     $key,$key_
928         pshufb  $bswap_mask,$iv
929 ___
930         &aesni_generate1("enc",$key,$rounds);
931 $code.=<<___;
932         movups  ($inp),$in0                     # load inp
933         paddq   $increment,$iv
934         lea     16($inp),$inp
935         jmp     .Lccm64_dec_outer
936 .align  16
937 .Lccm64_dec_outer:
938         xorps   $inout0,$in0                    # inp ^= E(iv)
939         movdqa  $iv,$inout0
```



```
940         mov     $rnds_,$rounds
941         movups  $in0,($out)                     # save output
942         lea     16($out),$out
943         pshufb  $bswap_mask,$inout0

945         sub     \$1,$len
946         jz      .Lccm64_dec_break

948         $movkey ($key_),$rndkey0
949         shr     \$1,$rounds
950         $movkey 16($key_),$rndkey1
951         xorps   $rndkey0,$in0
952         lea     32($key_),$key
953         xorps   $rndkey0,$inout0
954         xorps   $in0,$inout1                    # cmac^=out
955         $movkey ($key),$rndkey0

957 .Lccm64_dec2_loop:
958         aesenc  $rndkey1,$inout0
959         dec     $rounds
960         aesenc  $rndkey1,$inout1
961         $movkey 16($key),$rndkey1
962         aesenc  $rndkey0,$inout0
963         lea     32($key),$key
964         aesenc  $rndkey0,$inout1
965         $movkey 0($key),$rndkey0
966         jnz     .Lccm64_dec2_loop
967         movups  ($inp),$in0                     # load inp
968         paddq   $increment,$iv
969         aesenc  $rndkey1,$inout0
970         aesenc  $rndkey1,$inout1
971         lea     16($inp),$inp
972         aesenclast      $rndkey0,$inout0
973         aesenclast      $rndkey0,$inout1
974         jmp     .Lccm64_dec_outer

976 .align  16
977 .Lccm64_dec_break:
978         #xorps  $in0,$inout1                    # cmac^=out
979 ___
980         &aesni_generate1("enc",$key_,$rounds,$inout1,$in0);
981 $code.=<<___;
982         movups  $inout1,($cmac)
983 ___
984 $code.=<<___ if ($win64);
985         movaps  (%rsp),%xmm6
986         movaps  0x10(%rsp),%xmm7
987         movaps  0x20(%rsp),%xmm8
988         movaps  0x30(%rsp),%xmm9
989         lea     0x58(%rsp),%rsp
990 .Lccm64_dec_ret:
991 ___
992 $code.=<<___;
993         ret
994 .size   aesni_ccm64_decrypt_blocks,.-aesni_ccm64_decrypt_blocks
995 ___
996 }
```

```
 997 ########################################################################
 998 # void aesni_ctr32_encrypt_blocks (const void *in, void *out,
 999 #                         size_t blocks, const AES_KEY *key,
1000 #                         const char *ivec);
1001 #
1002 # Handles only complete blocks, operates on 32-bit counter and
1003 # does not update *ivec! (see engine/eng_aesni.c for details)
1004 #
1005 {
1006 my $reserved = $win64?0:-0x28;
1007 my ($in0,$in1,$in2,$in3)=map("%xmm$_",(8..11));
1008 my ($iv0,$iv1,$ivec)=("%xmm12","%xmm13","%xmm14");
1009 my $bswap_mask="%xmm15";

1011 $code.=<<___;
1012 .globl  aesni_ctr32_encrypt_blocks
1013 .type   aesni_ctr32_encrypt_blocks,\@function,5
1014 .align  16
1015 aesni_ctr32_encrypt_blocks:
1016 ___
1017 $code.=<<___ if ($win64);
1018         lea     -0xc8(%rsp),%rsp
1019         movaps  %xmm6,0x20(%rsp)
1020         movaps  %xmm7,0x30(%rsp)
1021         movaps  %xmm8,0x40(%rsp)
1022         movaps  %xmm9,0x50(%rsp)
1023         movaps  %xmm10,0x60(%rsp)
1024         movaps  %xmm11,0x70(%rsp)
1025         movaps  %xmm12,0x80(%rsp)
1026         movaps  %xmm13,0x90(%rsp)
1027         movaps  %xmm14,0xa0(%rsp)
1028         movaps  %xmm15,0xb0(%rsp)
1029 .Lctr32_body:
1030 ___
1031 $code.=<<___;
1032         cmp     \$1,$len
1033         je      .Lctr32_one_shortcut

1035         movdqu  ($ivp),$ivec
1036         movdqa  .Lbswap_mask(%rip),$bswap_mask
1037         xor     $rounds,$rounds
1038         pextrd  \$3,$ivec,$rnds_                # pull 32-bit counter
1039         pinsrd  \$3,$rounds,$ivec               # wipe 32-bit counter

1041         mov     240($key),$rounds               # key->rounds
1042         bswap   $rnds_
1043         pxor    $iv0,$iv0                       # vector of 3 32-bit counters
1044         pxor    $iv1,$iv1                       # vector of 3 32-bit counters
1045         pinsrd  \$0,$rnds_,$iv0
1046         lea     3($rnds_),$key_
1047         pinsrd  \$0,$key_,$iv1
1048         inc     $rnds_
1049         pinsrd  \$1,$rnds_,$iv0
1050         inc     $key_
1051         pinsrd  \$1,$key_,$iv1
1052         inc     $rnds_
1053         pinsrd  \$2,$rnds_,$iv0
1054         inc     $key_
1055         pinsrd  \$2,$key_,$iv1
1056         movdqa  $iv0,$reserved(%rsp)
1057         pshufb  $bswap_mask,$iv0
1058         movdqa  $iv1,`$reserved+0x10`(%rsp)
1059         pshufb  $bswap_mask,$iv1

1061         pshufd  \$`3<<6`,$iv0,$inout0           # place counter to upper dword
1062         pshufd  \$`2<<6`,$iv0,$inout1
```



```
1063         pshufd  \$`1<<6`,$iv0,$inout2
1064         cmp     \$6,$len
1065         jb      .Lctr32_tail
1066         shr     \$1,$rounds
1067         mov     $key,$key_                      # backup $key
1068         mov     $rounds,$rnds_                  # backup $rounds
1069         sub     \$6,$len
1070         jmp     .Lctr32_loop6

1072 .align  16
1073 .Lctr32_loop6:
1074         pshufd  \$`3<<6`,$iv1,$inout3
1075         por     $ivec,$inout0                   # merge counter-less ivec
1076          $movkey        ($key_),$rndkey0
1077         pshufd  \$`2<<6`,$iv1,$inout4
1078         por     $ivec,$inout1
1079          $movkey        16($key_),$rndkey1
1080         pshufd  \$`1<<6`,$iv1,$inout5
1081         por     $ivec,$inout2
1082         por     $ivec,$inout3
1083          xorps          $rndkey0,$inout0
1084         por     $ivec,$inout4
1085         por     $ivec,$inout5

1087         # inline _aesni_encrypt6 and interleave last rounds
1088         # with own code...

1090         pxor            $rndkey0,$inout1
1091         aesenc          $rndkey1,$inout0
1092         lea             32($key_),$key
1093         pxor            $rndkey0,$inout2
1094         aesenc          $rndkey1,$inout1
1095          movdqa         .Lincrement32(%rip),$iv1
1096         pxor            $rndkey0,$inout3
1097         aesenc          $rndkey1,$inout2
1098          movdqa         $reserved(%rsp),$iv0
1099         pxor            $rndkey0,$inout4
1100         aesenc          $rndkey1,$inout3
1101         pxor            $rndkey0,$inout5
1102         $movkey         ($key),$rndkey0
1103         dec             $rounds
1104         aesenc          $rndkey1,$inout4
1105         aesenc          $rndkey1,$inout5
1106         jmp             .Lctr32_enc_loop6_enter
1107 .align  16
1108 .Lctr32_enc_loop6:
1109         aesenc          $rndkey1,$inout0
1110         aesenc          $rndkey1,$inout1
1111         dec             $rounds
1112         aesenc          $rndkey1,$inout2
1113         aesenc          $rndkey1,$inout3
1114         aesenc          $rndkey1,$inout4
1115         aesenc          $rndkey1,$inout5
1116 .Lctr32_enc_loop6_enter:
1117         $movkey         16($key),$rndkey1
1118         aesenc          $rndkey0,$inout0
1119         aesenc          $rndkey0,$inout1
1120         lea             32($key),$key
1121         aesenc          $rndkey0,$inout2
1122         aesenc          $rndkey0,$inout3
1123         aesenc          $rndkey0,$inout4
1124         aesenc          $rndkey0,$inout5
1125         $movkey         ($key),$rndkey0
1126         jnz             .Lctr32_enc_loop6

1128         aesenc          $rndkey1,$inout0
```

```
1129          paddd           $iv1,$iv0               # increment counter vector
1130         aesenc           $rndkey1,$inout1
1131          paddd           ‘$reserved+0x10‘(%rsp),$iv1
1132         aesenc           $rndkey1,$inout2
1133          movdqa          $iv0,$reserved(%rsp)    # save counter vector
1134         aesenc           $rndkey1,$inout3
1135          movdqa          $iv1,‘$reserved+0x10‘(%rsp)
1136         aesenc           $rndkey1,$inout4
1137          pshufb          $bswap_mask,$iv0        # byte swap
1138         aesenc           $rndkey1,$inout5
1139          pshufb          $bswap_mask,$iv1

1141         aesenclast       $rndkey0,$inout0
1142          movups          ($inp),$in0             # load input
1143         aesenclast       $rndkey0,$inout1
1144          movups          0x10($inp),$in1
1145         aesenclast       $rndkey0,$inout2
1146          movups          0x20($inp),$in2
1147         aesenclast       $rndkey0,$inout3
1148          movups          0x30($inp),$in3
1149         aesenclast       $rndkey0,$inout4
1150          movups          0x40($inp),$rndkey1
1151         aesenclast       $rndkey0,$inout5
1152          movups          0x50($inp),$rndkey0
1153          lea     0x60($inp),$inp

1155         xorps    $inout0,$in0                    # xor
1156          pshufd  \$‘3<<6‘,$iv0,$inout0
1157         xorps    $inout1,$in1
1158          pshufd  \$‘2<<6‘,$iv0,$inout1
1159         movups   $in0,($out)                     # store output
1160         xorps    $inout2,$in2
1161          pshufd  \$‘1<<6‘,$iv0,$inout2
1162         movups   $in1,0x10($out)
1163         xorps    $inout3,$in3
1164         movups   $in2,0x20($out)
1165         xorps    $inout4,$rndkey1
1166         movups   $in3,0x30($out)
1167         xorps    $inout5,$rndkey0
1168         movups   $rndkey1,0x40($out)
1169         movups   $rndkey0,0x50($out)
1170         lea      0x60($out),$out
1171         mov      $rnds_,$rounds
1172         sub      \$6,$len
1173         jnc      .Lctr32_loop6

1175         add      \$6,$len
1176         jz       .Lctr32_done
1177         mov      $key_,$key                      # restore $key
1178         lea      1($rounds,$rounds),$rounds      # restore original value

1180 .Lctr32_tail:
1181         por      $ivec,$inout0
1182         movups   ($inp),$in0
1183         cmp      \$2,$len
1184         jb       .Lctr32_one

1186         por      $ivec,$inout1
1187         movups   0x10($inp),$in1
1188         je       .Lctr32_two

1190         pshufd   \$‘3<<6‘,$iv1,$inout3
1191         por      $ivec,$inout2
1192         movups   0x20($inp),$in2
1193         cmp      \$4,$len
1194         jb       .Lctr32_three
```



```
1196         pshufd   \$‘2<<6‘,$iv1,$inout4
1197         por      $ivec,$inout3
1198         movups   0x30($inp),$in3
1199         je       .Lctr32_four

1201         por      $ivec,$inout4
1202         xorps    $inout5,$inout5

1204         call     _aesni_encrypt6

1206         movups   0x40($inp),$rndkey1
1207         xorps    $inout0,$in0
1208         xorps    $inout1,$in1
1209         movups   $in0,($out)
1210         xorps    $inout2,$in2
1211         movups   $in1,0x10($out)
1212         xorps    $inout3,$in3
1213         movups   $in2,0x20($out)
1214         xorps    $inout4,$rndkey1
1215         movups   $in3,0x30($out)
1216         movups   $rndkey1,0x40($out)
1217         jmp      .Lctr32_done

1219 .align  16
1220 .Lctr32_one_shortcut:
1221         movups   ($ivp),$inout0
1222         movups   ($inp),$in0
1223         mov      240($key),$rounds               # key->rounds
1224 .Lctr32_one:
1225 ___
1226         &aesni_generate1("enc",$key,$rounds);
1227 $code.=<<___;
1228         xorps    $inout0,$in0
1229         movups   $in0,($out)
1230         jmp      .Lctr32_done

1232 .align  16
1233 .Lctr32_two:
1234         xorps    $inout2,$inout2
1235         call     _aesni_encrypt3
1236         xorps    $inout0,$in0
1237         xorps    $inout1,$in1
1238         movups   $in0,($out)
1239         movups   $in1,0x10($out)
1240         jmp      .Lctr32_done

1242 .align  16
1243 .Lctr32_three:
1244         call     _aesni_encrypt3
1245         xorps    $inout0,$in0
1246         xorps    $inout1,$in1
1247         movups   $in0,($out)
1248         xorps    $inout2,$in2
1249         movups   $in1,0x10($out)
1250         movups   $in2,0x20($out)
1251         jmp      .Lctr32_done

1253 .align  16
1254 .Lctr32_four:
1255         call     _aesni_encrypt4
1256         xorps    $inout0,$in0
1257         xorps    $inout1,$in1
1258         movups   $in0,($out)
1259         xorps    $inout2,$in2
1260         movups   $in1,0x10($out)
```

```
1261         xorps   $inout3,$in3
1262         movups  $in2,0x20($out)
1263         movups  $in3,0x30($out)

1265 .Lctr32_done:
1266 ___
1267 $code.=<<___ if ($win64);
1268         movaps  0x20(%rsp),%xmm6
1269         movaps  0x30(%rsp),%xmm7
1270         movaps  0x40(%rsp),%xmm8
1271         movaps  0x50(%rsp),%xmm9
1272         movaps  0x60(%rsp),%xmm10
1273         movaps  0x70(%rsp),%xmm11
1274         movaps  0x80(%rsp),%xmm12
1275         movaps  0x90(%rsp),%xmm13
1276         movaps  0xa0(%rsp),%xmm14
1277         movaps  0xb0(%rsp),%xmm15
1278         lea     0xc8(%rsp),%rsp
1279 .Lctr32_ret:
1280 ___
1281 $code.=<<___;
1282         ret
1283 .size   aesni_ctr32_encrypt_blocks,.-aesni_ctr32_encrypt_blocks
1284 ___
1285 }
```



```
1286 ######################################################################
1287 # void aesni_xts_[en|de]crypt(const char *inp,char *out,size_t len,
1288 #       const AES_KEY *key1, const AES_KEY *key2
1289 #       const unsigned char iv[16]);
1290 #
1291 {
1292 my @tweak=map("%xmm$_",(10..15));
1293 my ($twmask,$twres,$twtmp)=("%xmm8","%xmm9",@tweak[4]);
1294 my ($key2,$ivp,$len_)=("%r8","%r9","%r9");
1295 my $frame_size = 0x68 + ($win64?160:0);

1297 $code.=<<___;
1298 .globl  aesni_xts_encrypt
1299 .type   aesni_xts_encrypt,\@function,6
1300 .align  16
1301 aesni_xts_encrypt:
1302         lea     -$frame_size(%rsp),%rsp
1303 ___
1304 $code.=<<___ if ($win64);
1305         movaps  %xmm6,0x60(%rsp)
1306         movaps  %xmm7,0x70(%rsp)
1307         movaps  %xmm8,0x80(%rsp)
1308         movaps  %xmm9,0x90(%rsp)
1309         movaps  %xmm10,0xa0(%rsp)
1310         movaps  %xmm11,0xb0(%rsp)
1311         movaps  %xmm12,0xc0(%rsp)
1312         movaps  %xmm13,0xd0(%rsp)
1313         movaps  %xmm14,0xe0(%rsp)
1314         movaps  %xmm15,0xf0(%rsp)
1315 .Lxts_enc_body:
1316 ___
1317 $code.=<<___;
1318         movups  ($ivp),@tweak[5]                # load clear-text tweak
1319         mov     240(%r8),$rounds                # key2->rounds
1320         mov     240($key),$rnds_                # key1->rounds
1321 ___
1322         # generate the tweak
1323         &aesni_generate1("enc",$key2,$rounds,@tweak[5]);
1324 $code.=<<___;
1325         mov     $key,$key_                      # backup $key
1326         mov     $rnds_,$rounds                  # backup $rounds
1327         mov     $len,$len_                      # backup $len
1328         and     \$-16,$len

1330         movdqa  .Lxts_magic(%rip),$twmask
1331         pxor    $twtmp,$twtmp
1332         pcmpgtd @tweak[5],$twtmp                # broadcast upper bits
1333 ___
1334     for ($i=0;$i<4;$i++) {
1335     $code.=<<___;
1336         pshufd  \$0x13,$twtmp,$twres
1337         pxor    $twtmp,$twtmp
1338         movdqa  @tweak[5],@tweak[$i]
1339         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1340         pand    $twmask,$twres                  # isolate carry and residue
1341         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1342         pxor    $twres,@tweak[5]
1343 ___
1344     }
1345 $code.=<<___;
1346         sub     \$16*6,$len
1347         jc      .Lxts_enc_short

1349         shr     \$1,$rounds
1350         sub     \$1,$rounds
1351         mov     $rounds,$rnds_
```

```
1352         jmp     .Lxts_enc_grandloop

1354 .align  16
1355 .Lxts_enc_grandloop:
1356         pshufd  \$0x13,$twtmp,$twres
1357         movdqa  @tweak[5],@tweak[4]
1358         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1359         movdqu  ‘16*0‘($inp),$inout0            # load input
1360         pand    $twmask,$twres                  # isolate carry and residue
1361         movdqu  ‘16*1‘($inp),$inout1
1362         pxor    $twres,@tweak[5]

1364         movdqu  ‘16*2‘($inp),$inout2
1365         pxor    @tweak[0],$inout0               # input^=tweak
1366         movdqu  ‘16*3‘($inp),$inout3
1367         pxor    @tweak[1],$inout1
1368         movdqu  ‘16*4‘($inp),$inout4
1369         pxor    @tweak[2],$inout2
1370         movdqu  ‘16*5‘($inp),$inout5
1371         lea     ‘16*6‘($inp),$inp
1372         pxor    @tweak[3],$inout3
1373         $movkey         ($key_),$rndkey0
1374         pxor    @tweak[4],$inout4
1375         pxor    @tweak[5],$inout5

1377         # inline _aesni_encrypt6 and interleave first and last rounds
1378         # with own code...
1379         $movkey         16($key_),$rndkey1
1380         pxor            $rndkey0,$inout0
1381         pxor            $rndkey0,$inout1
1382          movdqa @tweak[0],‘16*0‘(%rsp)          # put aside tweaks
1383         aesenc          $rndkey1,$inout0
1384         lea             32($key_),$key
1385         pxor            $rndkey0,$inout2
1386          movdqa @tweak[1],‘16*1‘(%rsp)
1387         aesenc          $rndkey1,$inout1
1388         pxor            $rndkey0,$inout3
1389          movdqa @tweak[2],‘16*2‘(%rsp)
1390         aesenc          $rndkey1,$inout2
1391         pxor            $rndkey0,$inout4
1392          movdqa @tweak[3],‘16*3‘(%rsp)
1393         aesenc          $rndkey1,$inout3
1394         pxor            $rndkey0,$inout5
1395         $movkey         ($key),$rndkey0
1396         dec             $rounds
1397          movdqa @tweak[4],‘16*4‘(%rsp)
1398         aesenc          $rndkey1,$inout4
1399          movdqa @tweak[5],‘16*5‘(%rsp)
1400         aesenc          $rndkey1,$inout5
1401         pxor    $twtmp,$twtmp
1402         pcmpgtd @tweak[5],$twtmp
1403         jmp             .Lxts_enc_loop6_enter

1405 .align  16
1406 .Lxts_enc_loop6:
1407         aesenc          $rndkey1,$inout0
1408         aesenc          $rndkey1,$inout1
1409         dec             $rounds
1410         aesenc          $rndkey1,$inout2
1411         aesenc          $rndkey1,$inout3
1412         aesenc          $rndkey1,$inout4
1413         aesenc          $rndkey1,$inout5
1414 .Lxts_enc_loop6_enter:
1415         $movkey         16($key),$rndkey1
1416         aesenc          $rndkey0,$inout0
1417         aesenc          $rndkey0,$inout1
```



```
1418         lea             32($key),$key
1419         aesenc          $rndkey0,$inout2
1420         aesenc          $rndkey0,$inout3
1421         aesenc          $rndkey0,$inout4
1422         aesenc          $rndkey0,$inout5
1423         $movkey         ($key),$rndkey0
1424         jnz             .Lxts_enc_loop6

1426         pshufd  \$0x13,$twtmp,$twres
1427         pxor    $twtmp,$twtmp
1428         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1429          aesenc         $rndkey1,$inout0
1430         pand    $twmask,$twres                  # isolate carry and residue
1431          aesenc         $rndkey1,$inout1
1432         pcmpgtd @tweak[5],$twtmp                # broadcast upper bits
1433          aesenc         $rndkey1,$inout2
1434         pxor    $twres,@tweak[5]
1435          aesenc         $rndkey1,$inout3
1436          aesenc         $rndkey1,$inout4
1437          aesenc         $rndkey1,$inout5
1438          $movkey        16($key),$rndkey1

1440         pshufd  \$0x13,$twtmp,$twres
1441         pxor    $twtmp,$twtmp
1442         movdqa  @tweak[5],@tweak[0]
1443         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1444          aesenc         $rndkey0,$inout0
1445         pand    $twmask,$twres                  # isolate carry and residue
1446          aesenc         $rndkey0,$inout1
1447         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1448          aesenc         $rndkey0,$inout2
1449         pxor    $twres,@tweak[5]
1450          aesenc         $rndkey0,$inout3
1451          aesenc         $rndkey0,$inout4
1452          aesenc         $rndkey0,$inout5
1453          $movkey        32($key),$rndkey0

1455         pshufd  \$0x13,$twtmp,$twres
1456         pxor    $twtmp,$twtmp
1457         movdqa  @tweak[5],@tweak[1]
1458         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1459          aesenc         $rndkey1,$inout0
1460         pand    $twmask,$twres                  # isolate carry and residue
1461          aesenc         $rndkey1,$inout1
1462         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1463          aesenc         $rndkey1,$inout2
1464         pxor    $twres,@tweak[5]
1465          aesenc         $rndkey1,$inout3
1466          aesenc         $rndkey1,$inout4
1467          aesenc         $rndkey1,$inout5

1469         pshufd  \$0x13,$twtmp,$twres
1470         pxor    $twtmp,$twtmp
1471         movdqa  @tweak[5],@tweak[2]
1472         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1473          aesenclast     $rndkey0,$inout0
1474         pand    $twmask,$twres                  # isolate carry and residue
1475          aesenclast     $rndkey0,$inout1
1476         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1477          aesenclast     $rndkey0,$inout2
1478         pxor    $twres,@tweak[5]
1479          aesenclast     $rndkey0,$inout3
1480          aesenclast     $rndkey0,$inout4
1481          aesenclast     $rndkey0,$inout5

1483         pshufd  \$0x13,$twtmp,$twres
```

```
1484         pxor    $twtmp,$twtmp
1485         movdqa  @tweak[5],@tweak[3]
1486         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1487          xorps  '16*0'(%rsp),$inout0            # output^=tweak
1488         pand    $twmask,$twres                  # isolate carry and residue
1489          xorps  '16*1'(%rsp),$inout1
1490         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1491         pxor    $twres,@tweak[5]

1493         xorps   '16*2'(%rsp),$inout2
1494         movups  $inout0,'16*0'($out)            # write output
1495         xorps   '16*3'(%rsp),$inout3
1496         movups  $inout1,'16*1'($out)
1497         xorps   '16*4'(%rsp),$inout4
1498         movups  $inout2,'16*2'($out)
1499         xorps   '16*5'(%rsp),$inout5
1500         movups  $inout3,'16*3'($out)
1501         mov     $rnds_,$rounds                  # restore $rounds
1502         movups  $inout4,'16*4'($out)
1503         movups  $inout5,'16*5'($out)
1504         lea     '16*6'($out),$out
1505         sub     \$16*6,$len
1506         jnc     .Lxts_enc_grandloop

1508         lea     3($rounds,$rounds),$rounds      # restore original value
1509         mov     $key_,$key                      # restore $key
1510         mov     $rounds,$rnds_                  # backup $rounds

1512 .Lxts_enc_short:
1513         add     \$16*6,$len
1514         jz      .Lxts_enc_done

1516         cmp     \$0x20,$len
1517         jb      .Lxts_enc_one
1518         je      .Lxts_enc_two

1520         cmp     \$0x40,$len
1521         jb      .Lxts_enc_three
1522         je      .Lxts_enc_four

1524         pshufd  \$0x13,$twtmp,$twres
1525         movdqa  @tweak[5],@tweak[4]
1526         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1527          movdqu ($inp),$inout0
1528         pand    $twmask,$twres                  # isolate carry and residue
1529          movdqu 16*1($inp),$inout1
1530         pxor    $twres,@tweak[5]

1532         movdqu  16*2($inp),$inout2
1533         pxor    @tweak[0],$inout0
1534         movdqu  16*3($inp),$inout3
1535         pxor    @tweak[1],$inout1
1536         movdqu  16*4($inp),$inout4
1537         lea     16*5($inp),$inp
1538         pxor    @tweak[2],$inout2
1539         pxor    @tweak[3],$inout3
1540         pxor    @tweak[4],$inout4

1542         call    _aesni_encrypt6

1544         xorps   @tweak[0],$inout0
1545         movdqa  @tweak[5],@tweak[0]
1546         xorps   @tweak[1],$inout1
1547         xorps   @tweak[2],$inout2
1548         movdqu  $inout0,($out)
1549         xorps   @tweak[3],$inout3
```



```
1550         movdqu  $inout1,16*1($out)
1551         xorps   @tweak[4],$inout4
1552         movdqu  $inout2,16*2($out)
1553         movdqu  $inout3,16*3($out)
1554         movdqu  $inout4,16*4($out)
1555         lea     16*5($out),$out
1556         jmp     .Lxts_enc_done

1558 .align  16
1559 .Lxts_enc_one:
1560         movups  ($inp),$inout0
1561         lea     16*1($inp),$inp
1562         xorps   @tweak[0],$inout0
1563 ___
1564         &aesni_generate1("enc",$key,$rounds);
1565 $code.=<<___;
1566         xorps   @tweak[0],$inout0
1567         movdqa  @tweak[1],@tweak[0]
1568         movups  $inout0,($out)
1569         lea     16*1($out),$out
1570         jmp     .Lxts_enc_done

1572 .align  16
1573 .Lxts_enc_two:
1574         movups  ($inp),$inout0
1575         movups  16($inp),$inout1
1576         lea     32($inp),$inp
1577         xorps   @tweak[0],$inout0
1578         xorps   @tweak[1],$inout1

1580         call    _aesni_encrypt3

1582         xorps   @tweak[0],$inout0
1583         movdqa  @tweak[2],@tweak[0]
1584         xorps   @tweak[1],$inout1
1585         movups  $inout0,($out)
1586         movups  $inout1,16*1($out)
1587         lea     16*2($out),$out
1588         jmp     .Lxts_enc_done

1590 .align  16
1591 .Lxts_enc_three:
1592         movups  ($inp),$inout0
1593         movups  16*1($inp),$inout1
1594         movups  16*2($inp),$inout2
1595         lea     16*3($inp),$inp
1596         xorps   @tweak[0],$inout0
1597         xorps   @tweak[1],$inout1
1598         xorps   @tweak[2],$inout2

1600         call    _aesni_encrypt3

1602         xorps   @tweak[0],$inout0
1603         movdqa  @tweak[3],@tweak[0]
1604         xorps   @tweak[1],$inout1
1605         xorps   @tweak[2],$inout2
1606         movups  $inout0,($out)
1607         movups  $inout1,16*1($out)
1608         movups  $inout2,16*2($out)
1609         lea     16*3($out),$out
1610         jmp     .Lxts_enc_done

1612 .align  16
1613 .Lxts_enc_four:
1614         movups  ($inp),$inout0
1615         movups  16*1($inp),$inout1
```

```
1616         movups  16*2($inp),$inout2
1617         xorps   @tweak[0],$inout0
1618         movups  16*3($inp),$inout3
1619         lea     16*4($inp),$inp
1620         xorps   @tweak[1],$inout1
1621         xorps   @tweak[2],$inout2
1622         xorps   @tweak[3],$inout3

1624         call    _aesni_encrypt4

1626         xorps   @tweak[0],$inout0
1627         movdqa  @tweak[5],@tweak[0]
1628         xorps   @tweak[1],$inout1
1629         xorps   @tweak[2],$inout2
1630         movups  $inout0,($out)
1631         xorps   @tweak[3],$inout3
1632         movups  $inout1,16*1($out)
1633         movups  $inout2,16*2($out)
1634         movups  $inout3,16*3($out)
1635         lea     16*4($out),$out
1636         jmp     .Lxts_enc_done

1638 .align  16
1639 .Lxts_enc_done:
1640         and     \$15,$len_
1641         jz      .Lxts_enc_ret
1642         mov     $len_,$len

1644 .Lxts_enc_steal:
1645         movzb   ($inp),%eax                     # borrow $rounds ...
1646         movzb   -16($out),%ecx                  # ... and $key
1647         lea     1($inp),$inp
1648         mov     %al,-16($out)
1649         mov     %cl,0($out)
1650         lea     1($out),$out
1651         sub     \$1,$len
1652         jnz     .Lxts_enc_steal

1654         sub     $len_,$out                      # rewind $out
1655         mov     $key_,$key                      # restore $key
1656         mov     $rnds_,$rounds                  # restore $rounds

1658         movups  -16($out),$inout0
1659         xorps   @tweak[0],$inout0
1660 ___
1661         &aesni_generate1("enc",$key,$rounds);
1662 $code.=<<___;
1663         xorps   @tweak[0],$inout0
1664         movups  $inout0,-16($out)

1666 .Lxts_enc_ret:
1667 ___
1668 $code.=<<___ if ($win64);
1669         movaps  0x60(%rsp),%xmm6
1670         movaps  0x70(%rsp),%xmm7
1671         movaps  0x80(%rsp),%xmm8
1672         movaps  0x90(%rsp),%xmm9
1673         movaps  0xa0(%rsp),%xmm10
1674         movaps  0xb0(%rsp),%xmm11
1675         movaps  0xc0(%rsp),%xmm12
1676         movaps  0xd0(%rsp),%xmm13
1677         movaps  0xe0(%rsp),%xmm14
1678         movaps  0xf0(%rsp),%xmm15
1679 ___
1680 $code.=<<___;
1681         lea     $frame_size(%rsp),%rsp
```



```
1682 .Lxts_enc_epilogue:
1683         ret
1684 .size   aesni_xts_encrypt,.-aesni_xts_encrypt
1685 ___

1687 $code.=<<___;
1688 .globl  aesni_xts_decrypt
1689 .type   aesni_xts_decrypt,\@function,6
1690 .align  16
1691 aesni_xts_decrypt:
1692         lea     -$frame_size(%rsp),%rsp
1693 ___
1694 $code.=<<___ if ($win64);
1695         movaps  %xmm6,0x60(%rsp)
1696         movaps  %xmm7,0x70(%rsp)
1697         movaps  %xmm8,0x80(%rsp)
1698         movaps  %xmm9,0x90(%rsp)
1699         movaps  %xmm10,0xa0(%rsp)
1700         movaps  %xmm11,0xb0(%rsp)
1701         movaps  %xmm12,0xc0(%rsp)
1702         movaps  %xmm13,0xd0(%rsp)
1703         movaps  %xmm14,0xe0(%rsp)
1704         movaps  %xmm15,0xf0(%rsp)
1705 .Lxts_dec_body:
1706 ___
1707 $code.=<<___;
1708         movups  ($ivp),@tweak[5]                # load clear-text tweak
1709         mov     240($key2),$rounds              # key2->rounds
1710         mov     240($key),$rnds_                # key1->rounds
1711 ___
1712         # generate the tweak
1713         &aesni_generate1("enc",$key2,$rounds,@tweak[5]);
1714 $code.=<<___;
1715         xor     %eax,%eax                       # if ($len%16) len-=16;
1716         test    \$15,$len
1717         setnz   %al
1718         shl     \$4,%rax
1719         sub     %rax,$len

1721         mov     $key,$key_                      # backup $key
1722         mov     $rnds_,$rounds                  # backup $rounds
1723         mov     $len,$len_                      # backup $len
1724         and     \$-16,$len

1726         movdqa  .Lxts_magic(%rip),$twmask
1727         pxor    $twtmp,$twtmp
1728         pcmpgtd @tweak[5],$twtmp                # broadcast upper bits
1729 ___
1730     for ($i=0;$i<4;$i++) {
1731     $code.=<<___;
1732         pshufd  \$0x13,$twtmp,$twres
1733         pxor    $twtmp,$twtmp
1734         movdqa  @tweak[5],@tweak[$i]
1735         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1736         pand    $twmask,$twres                  # isolate carry and residue
1737         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1738         pxor    $twres,@tweak[5]
1739 ___
1740     }
1741 $code.=<<___;
1742         sub     \$16*6,$len
1743         jc      .Lxts_dec_short

1745         shr     \$1,$rounds
1746         sub     \$1,$rounds
1747         mov     $rounds,$rnds_
```

```
1748         jmp     .Lxts_dec_grandloop

1750 .align  16
1751 .Lxts_dec_grandloop:
1752         pshufd  \$0x13,$twtmp,$twres
1753         movdqa  @tweak[5],@tweak[4]
1754         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1755         movdqu  ‘16*0’($inp),$inout0            # load input
1756         pand    $twmask,$twres                  # isolate carry and residue
1757         movdqu  ‘16*1’($inp),$inout1
1758         pxor    $twres,@tweak[5]

1760         movdqu  ‘16*2’($inp),$inout2
1761         pxor    @tweak[0],$inout0               # input^=tweak
1762         movdqu  ‘16*3’($inp),$inout3
1763         pxor    @tweak[1],$inout1
1764         movdqu  ‘16*4’($inp),$inout4
1765         pxor    @tweak[2],$inout2
1766         movdqu  ‘16*5’($inp),$inout5
1767         lea     ‘16*6’($inp),$inp
1768         pxor    @tweak[3],$inout3
1769         $movkey         ($key_),$rndkey0
1770         pxor    @tweak[4],$inout4
1771         pxor    @tweak[5],$inout5

1773         # inline _aesni_decrypt6 and interleave first and last rounds
1774         # with own code...
1775         $movkey         16($key_),$rndkey1
1776         pxor            $rndkey0,$inout0
1777         pxor            $rndkey0,$inout1
1778          movdqa @tweak[0],‘16*0’(%rsp)          # put aside tweaks
1779         aesdec          $rndkey1,$inout0
1780         lea             32($key_),$key
1781         pxor            $rndkey0,$inout2
1782          movdqa @tweak[1],‘16*1’(%rsp)
1783         aesdec          $rndkey1,$inout1
1784         pxor            $rndkey0,$inout3
1785          movdqa @tweak[2],‘16*2’(%rsp)
1786         aesdec          $rndkey1,$inout2
1787         pxor            $rndkey0,$inout4
1788          movdqa @tweak[3],‘16*3’(%rsp)
1789         aesdec          $rndkey1,$inout3
1790         pxor            $rndkey0,$inout5
1791         $movkey         ($key),$rndkey0
1792         dec             $rounds
1793          movdqa @tweak[4],‘16*4’(%rsp)
1794         aesdec          $rndkey1,$inout4
1795          movdqa @tweak[5],‘16*5’(%rsp)
1796         aesdec          $rndkey1,$inout5
1797         pxor    $twtmp,$twtmp
1798         pcmpgtd @tweak[5],$twtmp
1799         jmp             .Lxts_dec_loop6_enter

1801 .align  16
1802 .Lxts_dec_loop6:
1803         aesdec          $rndkey1,$inout0
1804         aesdec          $rndkey1,$inout1
1805         dec             $rounds
1806         aesdec          $rndkey1,$inout2
1807         aesdec          $rndkey1,$inout3
1808         aesdec          $rndkey1,$inout4
1809         aesdec          $rndkey1,$inout5
1810 .Lxts_dec_loop6_enter:
1811         $movkey         16($key),$rndkey1
1812         aesdec          $rndkey0,$inout0
1813         aesdec          $rndkey0,$inout1
```



```
1814         lea             32($key),$key
1815         aesdec          $rndkey0,$inout2
1816         aesdec          $rndkey0,$inout3
1817         aesdec          $rndkey0,$inout4
1818         aesdec          $rndkey0,$inout5
1819         $movkey         ($key),$rndkey0
1820         jnz             .Lxts_dec_loop6

1822         pshufd  \$0x13,$twtmp,$twres
1823         pxor    $twtmp,$twtmp
1824         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1825          aesdec         $rndkey1,$inout0
1826         pand    $twmask,$twres                  # isolate carry and residue
1827          aesdec         $rndkey1,$inout1
1828         pcmpgtd @tweak[5],$twtmp                # broadcast upper bits
1829          aesdec         $rndkey1,$inout2
1830         pxor    $twres,@tweak[5]
1831          aesdec         $rndkey1,$inout3
1832          aesdec         $rndkey1,$inout4
1833          aesdec         $rndkey1,$inout5
1834          $movkey        16($key),$rndkey1

1836         pshufd  \$0x13,$twtmp,$twres
1837         pxor    $twtmp,$twtmp
1838         movdqa  @tweak[5],@tweak[0]
1839         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1840          aesdec         $rndkey0,$inout0
1841         pand    $twmask,$twres                  # isolate carry and residue
1842          aesdec         $rndkey0,$inout1
1843         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1844          aesdec         $rndkey0,$inout2
1845         pxor    $twres,@tweak[5]
1846          aesdec         $rndkey0,$inout3
1847          aesdec         $rndkey0,$inout4
1848          aesdec         $rndkey0,$inout5
1849          $movkey        32($key),$rndkey0

1851         pshufd  \$0x13,$twtmp,$twres
1852         pxor    $twtmp,$twtmp
1853         movdqa  @tweak[5],@tweak[1]
1854         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1855          aesdec         $rndkey1,$inout0
1856         pand    $twmask,$twres                  # isolate carry and residue
1857          aesdec         $rndkey1,$inout1
1858         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1859          aesdec         $rndkey1,$inout2
1860         pxor    $twres,@tweak[5]
1861          aesdec         $rndkey1,$inout3
1862          aesdec         $rndkey1,$inout4
1863          aesdec         $rndkey1,$inout5

1865         pshufd  \$0x13,$twtmp,$twres
1866         pxor    $twtmp,$twtmp
1867         movdqa  @tweak[5],@tweak[2]
1868         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1869          aesdeclast     $rndkey0,$inout0
1870         pand    $twmask,$twres                  # isolate carry and residue
1871          aesdeclast     $rndkey0,$inout1
1872         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1873          aesdeclast     $rndkey0,$inout2
1874         pxor    $twres,@tweak[5]
1875          aesdeclast     $rndkey0,$inout3
1876          aesdeclast     $rndkey0,$inout4
1877          aesdeclast     $rndkey0,$inout5

1879         pshufd  \$0x13,$twtmp,$twres
```

```
1880         pxor    $twtmp,$twtmp
1881         movdqa  @tweak[5],@tweak[3]
1882         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1883          xorps  '16*0'(%rsp),$inout0            # output^=tweak
1884         pand    $twmask,$twres                  # isolate carry and residue
1885          xorps  '16*1'(%rsp),$inout1
1886         pcmpgtd @tweak[5],$twtmp                # broadcat upper bits
1887         pxor    $twres,@tweak[5]

1889         xorps   '16*2'(%rsp),$inout2
1890         movups  $inout0,'16*0'($out)            # write output
1891         xorps   '16*3'(%rsp),$inout3
1892         movups  $inout1,'16*1'($out)
1893         xorps   '16*4'(%rsp),$inout4
1894         movups  $inout2,'16*2'($out)
1895         xorps   '16*5'(%rsp),$inout5
1896         movups  $inout3,'16*3'($out)
1897         mov     $rnds_,$rounds                  # restore $rounds
1898         movups  $inout4,'16*4'($out)
1899         movups  $inout5,'16*5'($out)
1900         lea     '16*6'($out),$out
1901         sub     \$16*6,$len
1902         jnc     .Lxts_dec_grandloop

1904         lea     3($rounds,$rounds),$rounds      # restore original value
1905         mov     $key_,$key                      # restore $key
1906         mov     $rounds,$rnds_                  # backup $rounds

1908 .Lxts_dec_short:
1909         add     \$16*6,$len
1910         jz      .Lxts_dec_done

1912         cmp     \$0x20,$len
1913         jb      .Lxts_dec_one
1914         je      .Lxts_dec_two

1916         cmp     \$0x40,$len
1917         jb      .Lxts_dec_three
1918         je      .Lxts_dec_four

1920         pshufd  \$0x13,$twtmp,$twres
1921         movdqa  @tweak[5],@tweak[4]
1922         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1923          movdqu ($inp),$inout0
1924         pand    $twmask,$twres                  # isolate carry and residue
1925          movdqu 16*1($inp),$inout1
1926         pxor    $twres,@tweak[5]

1928         movdqu  16*2($inp),$inout2
1929         pxor    @tweak[0],$inout0
1930         movdqu  16*3($inp),$inout3
1931         pxor    @tweak[1],$inout1
1932         movdqu  16*4($inp),$inout4
1933         lea     16*5($inp),$inp
1934         pxor    @tweak[2],$inout2
1935         pxor    @tweak[3],$inout3
1936         pxor    @tweak[4],$inout4

1938         call    _aesni_decrypt6

1940         xorps   @tweak[0],$inout0
1941         xorps   @tweak[1],$inout1
1942         xorps   @tweak[2],$inout2
1943         movdqu  $inout0,($out)
1944         xorps   @tweak[3],$inout3
1945         movdqu  $inout1,16*1($out)
```



```
1946         xorps   @tweak[4],$inout4
1947         movdqu  $inout2,16*2($out)
1948          pxor           $twtmp,$twtmp
1949         movdqu  $inout3,16*3($out)
1950          pcmpgtd        @tweak[5],$twtmp
1951         movdqu  $inout4,16*4($out)
1952         lea     16*5($out),$out
1953          pshufd         \$0x13,$twtmp,@tweak[1] # $twres
1954         and     \$15,$len_
1955         jz      .Lxts_dec_ret

1957         movdqa  @tweak[5],@tweak[0]
1958         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
1959         pand    $twmask,@tweak[1]               # isolate carry and residue
1960         pxor    @tweak[5],@tweak[1]
1961         jmp     .Lxts_dec_done2

1963 .align  16
1964 .Lxts_dec_one:
1965         movups  ($inp),$inout0
1966         lea     16*1($inp),$inp
1967         xorps   @tweak[0],$inout0
1968 ___
1969         &aesni_generate1("dec",$key,$rounds);
1970 $code.=<<___;
1971         xorps   @tweak[0],$inout0
1972         movdqa  @tweak[1],@tweak[0]
1973         movups  $inout0,($out)
1974         movdqa  @tweak[2],@tweak[1]
1975         lea     16*1($out),$out
1976         jmp     .Lxts_dec_done

1978 .align  16
1979 .Lxts_dec_two:
1980         movups  ($inp),$inout0
1981         movups  16($inp),$inout1
1982         lea     32($inp),$inp
1983         xorps   @tweak[0],$inout0
1984         xorps   @tweak[1],$inout1

1986         call    _aesni_decrypt3

1988         xorps   @tweak[0],$inout0
1989         movdqa  @tweak[2],@tweak[0]
1990         xorps   @tweak[1],$inout1
1991         movdqa  @tweak[3],@tweak[1]
1992         movups  $inout0,($out)
1993         movups  $inout1,16*1($out)
1994         lea     16*2($out),$out
1995         jmp     .Lxts_dec_done

1997 .align  16
1998 .Lxts_dec_three:
1999         movups  ($inp),$inout0
2000         movups  16*1($inp),$inout1
2001         movups  16*2($inp),$inout2
2002         lea     16*3($inp),$inp
2003         xorps   @tweak[0],$inout0
2004         xorps   @tweak[1],$inout1
2005         xorps   @tweak[2],$inout2

2007         call    _aesni_decrypt3

2009         xorps   @tweak[0],$inout0
2010         movdqa  @tweak[3],@tweak[0]
2011         xorps   @tweak[1],$inout1
```

```
2012         movdqa  @tweak[5],@tweak[1]
2013         xorps   @tweak[2],$inout2
2014         movups  $inout0,($out)
2015         movups  $inout1,16*1($out)
2016         movups  $inout2,16*2($out)
2017         lea     16*3($out),$out
2018         jmp     .Lxts_dec_done

2020 .align  16
2021 .Lxts_dec_four:
2022         pshufd  \$0x13,$twtmp,$twres
2023         movdqa  @tweak[5],@tweak[4]
2024         paddq   @tweak[5],@tweak[5]             # psllq 1,$tweak
2025          movups ($inp),$inout0
2026         pand    $twmask,$twres                  # isolate carry and residue
2027          movups 16*1($inp),$inout1
2028         pxor    $twres,@tweak[5]

2030         movups  16*2($inp),$inout2
2031         xorps   @tweak[0],$inout0
2032         movups  16*3($inp),$inout3
2033         lea     16*4($inp),$inp
2034         xorps   @tweak[1],$inout1
2035         xorps   @tweak[2],$inout2
2036         xorps   @tweak[3],$inout3

2038         call    _aesni_decrypt4

2040         xorps   @tweak[0],$inout0
2041         movdqa  @tweak[4],@tweak[0]
2042         xorps   @tweak[1],$inout1
2043         movdqa  @tweak[5],@tweak[1]
2044         xorps   @tweak[2],$inout2
2045         movups  $inout0,($out)
2046         xorps   @tweak[3],$inout3
2047         movups  $inout1,16*1($out)
2048         movups  $inout2,16*2($out)
2049         movups  $inout3,16*3($out)
2050         lea     16*4($out),$out
2051         jmp     .Lxts_dec_done

2053 .align  16
2054 .Lxts_dec_done:
2055         and     \$15,$len_
2056         jz      .Lxts_dec_ret
2057 .Lxts_dec_done2:
2058         mov     $len_,$len
2059         mov     $key_,$key                      # restore $key
2060         mov     $rnds_,$rounds                  # restore $rounds

2062         movups  ($inp),$inout0
2063         xorps   @tweak[1],$inout0
2064 ___
2065         &aesni_generate1("dec",$key,$rounds);
2066 $code.=<<___;
2067         xorps   @tweak[1],$inout0
2068         movups  $inout0,($out)

2070 .Lxts_dec_steal:
2071         movzb   16($inp),%eax                   # borrow $rounds ...
2072         movzb   ($out),%ecx                     # ... and $key
2073         lea     1($inp),$inp
2074         mov     %al,($out)
2075         mov     %cl,16($out)
2076         lea     1($out),$out
2077         sub     \$1,$len
```



```
2078         jnz     .Lxts_dec_steal

2080         sub     $len_,$out                      # rewind $out
2081         mov     $key_,$key                      # restore $key
2082         mov     $rnds_,$rounds                  # restore $rounds

2084         movups  ($out),$inout0
2085         xorps   @tweak[0],$inout0
2086 ___
2087         &aesni_generate1("dec",$key,$rounds);
2088 $code.=<<___;
2089         xorps   @tweak[0],$inout0
2090         movups  $inout0,($out)

2092 .Lxts_dec_ret:
2093 ___
2094 $code.=<<___ if ($win64);
2095         movaps  0x60(%rsp),%xmm6
2096         movaps  0x70(%rsp),%xmm7
2097         movaps  0x80(%rsp),%xmm8
2098         movaps  0x90(%rsp),%xmm9
2099         movaps  0xa0(%rsp),%xmm10
2100         movaps  0xb0(%rsp),%xmm11
2101         movaps  0xc0(%rsp),%xmm12
2102         movaps  0xd0(%rsp),%xmm13
2103         movaps  0xe0(%rsp),%xmm14
2104         movaps  0xf0(%rsp),%xmm15
2105 ___
2106 $code.=<<___;
2107         lea     $frame_size(%rsp),%rsp
2108 .Lxts_dec_epilogue:
2109         ret
2110 .size   aesni_xts_decrypt,.-aesni_xts_decrypt
2111 ___
2112 } }}
```

```
2113 ######################################################################
2114 # void $PREFIX_cbc_encrypt (const void *inp, void *out,
2115 #                           size_t length, const AES_KEY *key,
2116 #                           unsigned char *ivp,const int enc);
2117 {
2118 my $reserved = $win64?0x40:-0x18;       # used in decrypt
2119 $code.=<<___;
2120 .globl  ${PREFIX}_cbc_encrypt
2121 .type   ${PREFIX}_cbc_encrypt,\@function,6
2122 .align  16
2123 ${PREFIX}_cbc_encrypt:
2124         test    $len,$len               # check length
2125         jz      .Lcbc_ret

2127         mov     240($key),$rnds_        # key->rounds
2128         mov     $key,$key_              # backup $key
2129         test    %r9d,%r9d               # 6th argument
2130         jz      .Lcbc_decrypt
2131 #----------------------- CBC ENCRYPT -----------------------------#
2132         movups  ($ivp),$inout0          # load iv as initial state
2133         mov     $rnds_,$rounds
2134         cmp     \$16,$len
2135         jb      .Lcbc_enc_tail
2136         sub     \$16,$len
2137         jmp     .Lcbc_enc_loop
2138 .align  16
2139 .Lcbc_enc_loop:
2140         movups  ($inp),$inout1          # load input
2141         lea     16($inp),$inp
2142         #xorps  $inout1,$inout0
2143 ___
2144         &aesni_generate1("enc",$key,$rounds,$inout0,$inout1);
2145 $code.=<<___;
2146         mov     $rnds_,$rounds          # restore $rounds
2147         mov     $key_,$key              # restore $key
2148         movups  $inout0,0($out)         # store output
2149         lea     16($out),$out
2150         sub     \$16,$len
2151         jnc     .Lcbc_enc_loop
2152         add     \$16,$len
2153         jnz     .Lcbc_enc_tail
2154         movups  $inout0,($ivp)
2155         jmp     .Lcbc_ret

2157 .Lcbc_enc_tail:
2158         mov     $len,%rcx       # zaps $key
2159         xchg    $inp,$out       # $inp is %rsi and $out is %rdi now
2160         .long   0x9066A4F3      # rep movsb
2161         mov     \$16,%ecx       # zero tail
2162         sub     $len,%rcx
2163         xor     %eax,%eax
2164         .long   0x9066AAF3      # rep stosb
2165         lea     -16(%rdi),%rdi  # rewind $out by 1 block
2166         mov     $rnds_,$rounds  # restore $rounds
2167         mov     %rdi,%rsi       # $inp and $out are the same
2168         mov     $key_,$key      # restore $key
2169         xor     $len,$len       # len=16
2170         jmp     .Lcbc_enc_loop  # one more spin
```



```
2171 #----------------------- CBC DECRYPT -----------------------------#
2172 .align  16
2173 .Lcbc_decrypt:
2174 ___
2175 $code.=<<___ if ($win64);
2176         lea     -0x58(%rsp),%rsp
2177         movaps  %xmm6,(%rsp)
2178         movaps  %xmm7,0x10(%rsp)
2179         movaps  %xmm8,0x20(%rsp)
2180         movaps  %xmm9,0x30(%rsp)
2181 .Lcbc_decrypt_body:
2182 ___
2183 $code.=<<___;
2184         movups  ($ivp),$iv
2185         mov     $rnds_,$rounds
2186         cmp     \$0x70,$len
2187         jbe     .Lcbc_dec_tail
2188         shr     \$1,$rnds_
2189         sub     \$0x70,$len
2190         mov     $rnds_,$rounds
2191         movaps  $iv,$reserved(%rsp)
2192         jmp     .Lcbc_dec_loop8_enter
2193 .align  16
2194 .Lcbc_dec_loop8:
2195         movaps  $rndkey0,$reserved(%rsp)        # save IV
2196         movups  $inout7,($out)
2197         lea     0x10($out),$out
2198 .Lcbc_dec_loop8_enter:
2199         $movkey         ($key),$rndkey0
2200         movups  ($inp),$inout0                  # load input
2201         movups  0x10($inp),$inout1
2202         $movkey         16($key),$rndkey1

2204         lea             32($key),$key
2205         movdqu  0x20($inp),$inout2
2206         xorps           $rndkey0,$inout0
2207         movdqu  0x30($inp),$inout3
2208         xorps           $rndkey0,$inout1
2209         movdqu  0x40($inp),$inout4
2210         aesdec          $rndkey1,$inout0
2211         pxor            $rndkey0,$inout2
2212         movdqu  0x50($inp),$inout5
2213         aesdec          $rndkey1,$inout1
2214         pxor            $rndkey0,$inout3
2215         movdqu  0x60($inp),$inout6
2216         aesdec          $rndkey1,$inout2
2217         pxor            $rndkey0,$inout4
2218         movdqu  0x70($inp),$inout7
2219         aesdec          $rndkey1,$inout3
2220         pxor            $rndkey0,$inout5
2221         dec             $rounds
2222         aesdec          $rndkey1,$inout4
2223         pxor            $rndkey0,$inout6
2224         aesdec          $rndkey1,$inout5
2225         pxor            $rndkey0,$inout7
2226         $movkey         ($key),$rndkey0
2227         aesdec          $rndkey1,$inout6
2228         aesdec          $rndkey1,$inout7
2229         $movkey         16($key),$rndkey1

2231         call            .Ldec_loop8_enter

2233         movups  ($inp),$rndkey1         # re-load input
2234         movups  0x10($inp),$rndkey0
2235         xorps   $reserved(%rsp),$inout0 # ^= IV
2236         xorps   $rndkey1,$inout1
```

```
2237         movups  0x20($inp),$rndkey1
2238         xorps   $rndkey0,$inout2
2239         movups  0x30($inp),$rndkey0
2240         xorps   $rndkey1,$inout3
2241         movups  0x40($inp),$rndkey1
2242         xorps   $rndkey0,$inout4
2243         movups  0x50($inp),$rndkey0
2244         xorps   $rndkey1,$inout5
2245         movups  0x60($inp),$rndkey1
2246         xorps   $rndkey0,$inout6
2247         movups  0x70($inp),$rndkey0     # IV
2248         xorps   $rndkey1,$inout7
2249         movups  $inout0,($out)
2250         movups  $inout1,0x10($out)
2251         movups  $inout2,0x20($out)
2252         movups  $inout3,0x30($out)
2253         mov     $rnds_,$rounds          # restore $rounds
2254         movups  $inout4,0x40($out)
2255         mov     $key_,$key              # restore $key
2256         movups  $inout5,0x50($out)
2257         lea     0x80($inp),$inp
2258         movups  $inout6,0x60($out)
2259         lea     0x70($out),$out
2260         sub     \$0x80,$len
2261         ja      .Lcbc_dec_loop8

2263         movaps  $inout7,$inout0
2264         movaps  $rndkey0,$iv
2265         add     \$0x70,$len
2266         jle     .Lcbc_dec_tail_collected
2267         movups  $inout0,($out)
2268         lea     1($rnds_,$rnds_),$rounds
2269         lea     0x10($out),$out
2270 .Lcbc_dec_tail:
2271         movups  ($inp),$inout0
2272         movaps  $inout0,$in0
2273         cmp     \$0x10,$len
2274         jbe     .Lcbc_dec_one

2276         movups  0x10($inp),$inout1
2277         movaps  $inout1,$in1
2278         cmp     \$0x20,$len
2279         jbe     .Lcbc_dec_two

2281         movups  0x20($inp),$inout2
2282         movaps  $inout2,$in2
2283         cmp     \$0x30,$len
2284         jbe     .Lcbc_dec_three

2286         movups  0x30($inp),$inout3
2287         cmp     \$0x40,$len
2288         jbe     .Lcbc_dec_four

2290         movups  0x40($inp),$inout4
2291         cmp     \$0x50,$len
2292         jbe     .Lcbc_dec_five

2294         movups  0x50($inp),$inout5
2295         cmp     \$0x60,$len
2296         jbe     .Lcbc_dec_six

2298         movups  0x60($inp),$inout6
2299         movaps  $iv,$reserved(%rsp)     # save IV
2300         call    _aesni_decrypt8
2301         movups  ($inp),$rndkey1
2302         movups  0x10($inp),$rndkey0
```



```
2303         xorps   $reserved(%rsp),$inout0 # ^= IV
2304         xorps   $rndkey1,$inout1
2305         movups  0x20($inp),$rndkey1
2306         xorps   $rndkey0,$inout2
2307         movups  0x30($inp),$rndkey0
2308         xorps   $rndkey1,$inout3
2309         movups  0x40($inp),$rndkey1
2310         xorps   $rndkey0,$inout4
2311         movups  0x50($inp),$rndkey0
2312         xorps   $rndkey1,$inout5
2313         movups  0x60($inp),$iv          # IV
2314         xorps   $rndkey0,$inout6
2315         movups  $inout0,($out)
2316         movups  $inout1,0x10($out)
2317         movups  $inout2,0x20($out)
2318         movups  $inout3,0x30($out)
2319         movups  $inout4,0x40($out)
2320         movups  $inout5,0x50($out)
2321         lea     0x60($out),$out
2322         movaps  $inout6,$inout0
2323         sub     \$0x70,$len
2324         jmp     .Lcbc_dec_tail_collected
2325 .align  16
2326 .Lcbc_dec_one:
2327 ___
2328         &aesni_generate1("dec",$key,$rounds);
2329 $code.=<<___;
2330         xorps   $iv,$inout0
2331         movaps  $in0,$iv
2332         sub     \$0x10,$len
2333         jmp     .Lcbc_dec_tail_collected
2334 .align  16
2335 .Lcbc_dec_two:
2336         xorps   $inout2,$inout2
2337         call    _aesni_decrypt3
2338         xorps   $iv,$inout0
2339         xorps   $in0,$inout1
2340         movups  $inout0,($out)
2341         movaps  $in1,$iv
2342         movaps  $inout1,$inout0
2343         lea     0x10($out),$out
2344         sub     \$0x20,$len
2345         jmp     .Lcbc_dec_tail_collected
2346 .align  16
2347 .Lcbc_dec_three:
2348         call    _aesni_decrypt3
2349         xorps   $iv,$inout0
2350         xorps   $in0,$inout1
2351         movups  $inout0,($out)
2352         xorps   $in1,$inout2
2353         movups  $inout1,0x10($out)
2354         movaps  $in2,$iv
2355         movaps  $inout2,$inout0
2356         lea     0x20($out),$out
2357         sub     \$0x30,$len
2358         jmp     .Lcbc_dec_tail_collected
2359 .align  16
2360 .Lcbc_dec_four:
2361         call    _aesni_decrypt4
2362         xorps   $iv,$inout0
2363         movups  0x30($inp),$iv
2364         xorps   $in0,$inout1
2365         movups  $inout0,($out)
2366         xorps   $in1,$inout2
2367         movups  $inout1,0x10($out)
2368         xorps   $in2,$inout3
```

```
2369         movups  $inout2,0x20($out)
2370         movaps  $inout3,$inout0
2371         lea     0x30($out),$out
2372         sub     \$0x40,$len
2373         jmp     .Lcbc_dec_tail_collected
2374 .align  16
2375 .Lcbc_dec_five:
2376         xorps   $inout5,$inout5
2377         call    _aesni_decrypt6
2378         movups  0x10($inp),$rndkey1
2379         movups  0x20($inp),$rndkey0
2380         xorps   $iv,$inout0
2381         xorps   $in0,$inout1
2382         xorps   $rndkey1,$inout2
2383         movups  0x30($inp),$rndkey1
2384         xorps   $rndkey0,$inout3
2385         movups  0x40($inp),$iv
2386         xorps   $rndkey1,$inout4
2387         movups  $inout0,($out)
2388         movups  $inout1,0x10($out)
2389         movups  $inout2,0x20($out)
2390         movups  $inout3,0x30($out)
2391         lea     0x40($out),$out
2392         movaps  $inout4,$inout0
2393         sub     \$0x50,$len
2394         jmp     .Lcbc_dec_tail_collected
2395 .align  16
2396 .Lcbc_dec_six:
2397         call    _aesni_decrypt6
2398         movups  0x10($inp),$rndkey1
2399         movups  0x20($inp),$rndkey0
2400         xorps   $iv,$inout0
2401         xorps   $in0,$inout1
2402         xorps   $rndkey1,$inout2
2403         movups  0x30($inp),$rndkey1
2404         xorps   $rndkey0,$inout3
2405         movups  0x40($inp),$rndkey0
2406         xorps   $rndkey1,$inout4
2407         movups  0x50($inp),$iv
2408         xorps   $rndkey0,$inout5
2409         movups  $inout0,($out)
2410         movups  $inout1,0x10($out)
2411         movups  $inout2,0x20($out)
2412         movups  $inout3,0x30($out)
2413         movups  $inout4,0x40($out)
2414         lea     0x50($out),$out
2415         movaps  $inout5,$inout0
2416         sub     \$0x60,$len
2417         jmp     .Lcbc_dec_tail_collected
2418 .align  16
2419 .Lcbc_dec_tail_collected:
2420         and     \$15,$len
2421         movups  $iv,($ivp)
2422         jnz     .Lcbc_dec_tail_partial
2423         movups  $inout0,($out)
2424         jmp     .Lcbc_dec_ret
2425 .align  16
2426 .Lcbc_dec_tail_partial:
2427         movaps  $inout0,$reserved(%rsp)
2428         mov     \$16,%rcx
2429         mov     $out,%rdi
2430         sub     $len,%rcx
2431         lea     $reserved(%rsp),%rsi
2432         .long   0x9066A4F3      # rep movsb

2434 .Lcbc_dec_ret:
```



```
2435 ___
2436 $code.=<<___ if ($win64);
2437         movaps  (%rsp),%xmm6
2438         movaps  0x10(%rsp),%xmm7
2439         movaps  0x20(%rsp),%xmm8
2440         movaps  0x30(%rsp),%xmm9
2441         lea     0x58(%rsp),%rsp
2442 ___
2443 $code.=<<___;
2444 .Lcbc_ret:
2445         ret
2446 .size   ${PREFIX}_cbc_encrypt,.-${PREFIX}_cbc_encrypt
2447 ___
2448 }
```

```
2449 # int $PREFIX_set_[en|de]crypt_key (const unsigned char *userKey,
2450 #                                int bits, AES_KEY *key)
2451 { my ($inp,$bits,$key) = @_4args;
2452   $bits =~ s/%r/%e/;

2454 $code.=<<___;
2455 .globl  ${PREFIX}_set_decrypt_key
2456 .type   ${PREFIX}_set_decrypt_key,\@abi-omnipotent
2457 .align  16
2458 ${PREFIX}_set_decrypt_key:
2459         .byte   0x48,0x83,0xEC,0x08     # sub rsp,8
2460         call    __aesni_set_encrypt_key
2461         shl     \$4,$bits               # rounds-1 after _aesni_set_encrypt_key
2462         test    %eax,%eax
2463         jnz     .Ldec_key_ret
2464         lea     16($key,$bits),$inp     # points at the end of key schedule

2466         $movkey ($key),%xmm0            # just swap
2467         $movkey ($inp),%xmm1
2468         $movkey %xmm0,($inp)
2469         $movkey %xmm1,($key)
2470         lea     16($key),$key
2471         lea     -16($inp),$inp

2473 .Ldec_key_inverse:
2474         $movkey ($key),%xmm0            # swap and inverse
2475         $movkey ($inp),%xmm1
2476         aesimc  %xmm0,%xmm0
2477         aesimc  %xmm1,%xmm1
2478         lea     16($key),$key
2479         lea     -16($inp),$inp
2480         $movkey %xmm0,16($inp)
2481         $movkey %xmm1,-16($key)
2482         cmp     $key,$inp
2483         ja      .Ldec_key_inverse

2485         $movkey ($key),%xmm0            # inverse middle
2486         aesimc  %xmm0,%xmm0
2487         $movkey %xmm0,($inp)
2488 .Ldec_key_ret:
2489         add     \$8,%rsp
2490         ret
2491 .LSEH_end_set_decrypt_key:
2492 .size   ${PREFIX}_set_decrypt_key,.-${PREFIX}_set_decrypt_key
2493 ___
```



```
2494 # This is based on submission by
2495 #
2496 #       Huang Ying <ying.huang@intel.com>
2497 #       Vinodh Gopal <vinodh.gopal@intel.com>
2498 #       Kahraman Akdemir
2499 #
2500 # Agressively optimized in respect to aeskeygenassist's critical path
2501 # and is contained in %xmm0-5 to meet Win64 ABI requirement.
2502 #
2503 $code.=<<___;
2504 .globl  ${PREFIX}_set_encrypt_key
2505 .type   ${PREFIX}_set_encrypt_key,\@abi-omnipotent
2506 .align  16
2507 ${PREFIX}_set_encrypt_key:
2508 __aesni_set_encrypt_key:
2509         .byte   0x48,0x83,0xEC,0x08     # sub rsp,8
2510         mov     \$-1,%rax
2511         test    $inp,$inp
2512         jz      .Lenc_key_ret
2513         test    $key,$key
2514         jz      .Lenc_key_ret

2516         movups  ($inp),%xmm0            # pull first 128 bits of *userKey
2517         xorps   %xmm4,%xmm4             # low dword of xmm4 is assumed 0
2518         lea     16($key),%rax
2519         cmp     \$256,$bits
2520         je      .L14rounds
2521         cmp     \$192,$bits
2522         je      .L12rounds
2523         cmp     \$128,$bits
2524         jne     .Lbad_keybits

2526 .L10rounds:
2527         mov     \$9,$bits                       # 10 rounds for 128-bit key
2528         $movkey %xmm0,($key)                    # round 0
2529         aeskeygenassist \$0x1,%xmm0,%xmm1       # round 1
2530         call            .Lkey_expansion_128_cold
2531         aeskeygenassist \$0x2,%xmm0,%xmm1       # round 2
2532         call            .Lkey_expansion_128
2533         aeskeygenassist \$0x4,%xmm0,%xmm1       # round 3
2534         call            .Lkey_expansion_128
2535         aeskeygenassist \$0x8,%xmm0,%xmm1       # round 4
2536         call            .Lkey_expansion_128
2537         aeskeygenassist \$0x10,%xmm0,%xmm1      # round 5
2538         call            .Lkey_expansion_128
2539         aeskeygenassist \$0x20,%xmm0,%xmm1      # round 6
2540         call            .Lkey_expansion_128
2541         aeskeygenassist \$0x40,%xmm0,%xmm1      # round 7
2542         call            .Lkey_expansion_128
2543         aeskeygenassist \$0x80,%xmm0,%xmm1      # round 8
2544         call            .Lkey_expansion_128
2545         aeskeygenassist \$0x1b,%xmm0,%xmm1      # round 9
2546         call            .Lkey_expansion_128
2547         aeskeygenassist \$0x36,%xmm0,%xmm1      # round 10
2548         call            .Lkey_expansion_128
2549         $movkey %xmm0,(%rax)
2550         mov     $bits,80(%rax)  # 240(%rdx)
2551         xor     %eax,%eax
2552         jmp     .Lenc_key_ret

2554 .align  16
2555 .L12rounds:
2556         movq    16($inp),%xmm2                  # remaining 1/3 of *userKey
2557         mov     \$11,$bits                      # 12 rounds for 192
2558         $movkey %xmm0,($key)                    # round 0
2559         aeskeygenassist \$0x1,%xmm2,%xmm1       # round 1,2
```

```
2560         call            .Lkey_expansion_192a_cold
2561         aeskeygenassist \$0x2,%xmm2,%xmm1       # round 2,3
2562         call            .Lkey_expansion_192b
2563         aeskeygenassist \$0x4,%xmm2,%xmm1       # round 4,5
2564         call            .Lkey_expansion_192a
2565         aeskeygenassist \$0x8,%xmm2,%xmm1       # round 5,6
2566         call            .Lkey_expansion_192b
2567         aeskeygenassist \$0x10,%xmm2,%xmm1      # round 7,8
2568         call            .Lkey_expansion_192a
2569         aeskeygenassist \$0x20,%xmm2,%xmm1      # round 8,9
2570         call            .Lkey_expansion_192b
2571         aeskeygenassist \$0x40,%xmm2,%xmm1      # round 10,11
2572         call            .Lkey_expansion_192a
2573         aeskeygenassist \$0x80,%xmm2,%xmm1      # round 11,12
2574         call            .Lkey_expansion_192b
2575         $movkey %xmm0,(%rax)
2576         mov     $bits,48(%rax)  # 240(%rdx)
2577         xor     %rax, %rax
2578         jmp     .Lenc_key_ret

2580 .align  16
2581 .L14rounds:
2582         movups  16($inp),%xmm2                  # remaning half of *userKey
2583         mov     \$13,$bits                      # 14 rounds for 256
2584         lea     16(%rax),%rax
2585         $movkey %xmm0,($key)                    # round 0
2586         $movkey %xmm2,16($key)                  # round 1
2587         aeskeygenassist \$0x1,%xmm2,%xmm1       # round 2
2588         call            .Lkey_expansion_256a_cold
2589         aeskeygenassist \$0x1,%xmm0,%xmm1       # round 3
2590         call            .Lkey_expansion_256b
2591         aeskeygenassist \$0x2,%xmm2,%xmm1       # round 4
2592         call            .Lkey_expansion_256a
2593         aeskeygenassist \$0x2,%xmm0,%xmm1       # round 5
2594         call            .Lkey_expansion_256b
2595         aeskeygenassist \$0x4,%xmm2,%xmm1       # round 6
2596         call            .Lkey_expansion_256a
2597         aeskeygenassist \$0x4,%xmm0,%xmm1       # round 7
2598         call            .Lkey_expansion_256b
2599         aeskeygenassist \$0x8,%xmm2,%xmm1       # round 8
2600         call            .Lkey_expansion_256a
2601         aeskeygenassist \$0x8,%xmm0,%xmm1       # round 9
2602         call            .Lkey_expansion_256b
2603         aeskeygenassist \$0x10,%xmm2,%xmm1      # round 10
2604         call            .Lkey_expansion_256a
2605         aeskeygenassist \$0x10,%xmm0,%xmm1      # round 11
2606         call            .Lkey_expansion_256b
2607         aeskeygenassist \$0x20,%xmm2,%xmm1      # round 12
2608         call            .Lkey_expansion_256a
2609         aeskeygenassist \$0x20,%xmm0,%xmm1      # round 13
2610         call            .Lkey_expansion_256b
2611         aeskeygenassist \$0x40,%xmm2,%xmm1      # round 14
2612         call            .Lkey_expansion_256a
2613         $movkey %xmm0,(%rax)
2614         mov     $bits,16(%rax)  # 240(%rdx)
2615         xor     %rax,%rax
2616         jmp     .Lenc_key_ret

2618 .align  16
2619 .Lbad_keybits:
2620         mov     \$-2,%rax
2621 .Lenc_key_ret:
2622         add     \$8,%rsp
2623         ret
2624 .LSEH_end_set_encrypt_key:
```



```
2625 .align  16
2626 .Lkey_expansion_128:
2627         $movkey %xmm0,(%rax)
2628         lea     16(%rax),%rax
2629 .Lkey_expansion_128_cold:
2630         shufps  \$0b00010000,%xmm0,%xmm4
2631         xorps   %xmm4, %xmm0
2632         shufps  \$0b10001100,%xmm0,%xmm4
2633         xorps   %xmm4, %xmm0
2634         shufps  \$0b11111111,%xmm1,%xmm1        # critical path
2635         xorps   %xmm1,%xmm0
2636         ret

2638 .align 16
2639 .Lkey_expansion_192a:
2640         $movkey %xmm0,(%rax)
2641         lea     16(%rax),%rax
2642 .Lkey_expansion_192a_cold:
2643         movaps  %xmm2, %xmm5
2644 .Lkey_expansion_192b_warm:
2645         shufps  \$0b00010000,%xmm0,%xmm4
2646         movdqa  %xmm2,%xmm3
2647         xorps   %xmm4,%xmm0
2648         shufps  \$0b10001100,%xmm0,%xmm4
2649         pslldq  \$4,%xmm3
2650         xorps   %xmm4,%xmm0
2651         pshufd  \$0b01010101,%xmm1,%xmm1        # critical path
2652         pxor    %xmm3,%xmm2
2653         pxor    %xmm1,%xmm0
2654         pshufd  \$0b11111111,%xmm0,%xmm3
2655         pxor    %xmm3,%xmm2
2656         ret

2658 .align 16
2659 .Lkey_expansion_192b:
2660         movaps  %xmm0,%xmm3
2661         shufps  \$0b01000100,%xmm0,%xmm5
2662         $movkey %xmm5,(%rax)
2663         shufps  \$0b01001110,%xmm2,%xmm3
2664         $movkey %xmm3,16(%rax)
2665         lea     32(%rax),%rax
2666         jmp     .Lkey_expansion_192b_warm

2668 .align  16
2669 .Lkey_expansion_256a:
2670         $movkey %xmm2,(%rax)
2671         lea     16(%rax),%rax
2672 .Lkey_expansion_256a_cold:
2673         shufps  \$0b00010000,%xmm0,%xmm4
2674         xorps   %xmm4,%xmm0
2675         shufps  \$0b10001100,%xmm0,%xmm4
2676         xorps   %xmm4,%xmm0
2677         shufps  \$0b11111111,%xmm1,%xmm1        # critical path
2678         xorps   %xmm1,%xmm0
2679         ret

2681 .align 16
2682 .Lkey_expansion_256b:
2683         $movkey %xmm0,(%rax)
2684         lea     16(%rax),%rax

2686         shufps  \$0b00010000,%xmm2,%xmm4
2687         xorps   %xmm4,%xmm2
2688         shufps  \$0b10001100,%xmm2,%xmm4
2689         xorps   %xmm4,%xmm2
2690         shufps  \$0b10101010,%xmm1,%xmm1        # critical path
```

```
2691         xorps   %xmm1,%xmm2
2692         ret
2693 .size   ${PREFIX}_set_encrypt_key,.-${PREFIX}_set_encrypt_key
2694 .size   __aesni_set_encrypt_key,.-__aesni_set_encrypt_key
2695 ___
2696 }
```



```
2697 $code.=<<___;
2698 .align  64
2699 .Lbswap_mask:
2700         .byte   15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
2701 .Lincrement32:
2702         .long   6,6,6,0
2703 .Lincrement64:
2704         .long   1,0,0,0
2705 .Lxts_magic:
2706         .long   0x87,0,1,0
2707
2708 .asciz  "AES for Intel AES-NI, CRYPTOGAMS by <appro\@openssl.org>"
2709 .align  64
2710 ___
2711
2712 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
2713 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
2714 if ($win64) {
2715 $rec="%rcx";
2716 $frame="%rdx";
2717 $context="%r8";
2718 $disp="%r9";
2719
2720 $code.=<<___;
2721 .extern __imp_RtlVirtualUnwind
2722 ___
2723 $code.=<<___ if ($PREFIX eq "aesni");
2724 .type   ecb_se_handler,\@abi-omnipotent
2725 .align  16
2726 ecb_se_handler:
2727         push    %rsi
2728         push    %rdi
2729         push    %rbx
2730         push    %rbp
2731         push    %r12
2732         push    %r13
2733         push    %r14
2734         push    %r15
2735         pushfq
2736         sub     \$64,%rsp
2737
2738         mov     152($context),%rax      # pull context->Rsp
2739
2740         jmp     .Lcommon_seh_tail
2741 .size   ecb_se_handler,.-ecb_se_handler
2742
2743 .type   ccm64_se_handler,\@abi-omnipotent
2744 .align  16
2745 ccm64_se_handler:
2746         push    %rsi
2747         push    %rdi
2748         push    %rbx
2749         push    %rbp
2750         push    %r12
2751         push    %r13
2752         push    %r14
2753         push    %r15
2754         pushfq
2755         sub     \$64,%rsp
2756
2757         mov     120($context),%rax      # pull context->Rax
2758         mov     248($context),%rbx      # pull context->Rip
2759
2760         mov     8($disp),%rsi           # disp->ImageBase
2761         mov     56($disp),%r11          # disp->HandlerData
```

```
2763         mov     0(%r11),%r10d           # HandlerData[0]
2764         lea     (%rsi,%r10),%r10        # prologue label
2765         cmp     %r10,%rbx               # context->Rip<prologue label
2766         jb      .Lcommon_seh_tail

2768         mov     152($context),%rax      # pull context->Rsp

2770         mov     4(%r11),%r10d           # HandlerData[1]
2771         lea     (%rsi,%r10),%r10        # epilogue label
2772         cmp     %r10,%rbx               # context->Rip>=epilogue label
2773         jae     .Lcommon_seh_tail

2775         lea     0(%rax),%rsi            # %xmm save area
2776         lea     512($context),%rdi      # &context.Xmm6
2777         mov     \$8,%ecx                # 4*sizeof(%xmm0)/sizeof(%rax)
2778         .long   0xa548f3fc              # cld; rep movsq
2779         lea     0x58(%rax),%rax         # adjust stack pointer

2781         jmp     .Lcommon_seh_tail
2782 .size   ccm64_se_handler,.-ccm64_se_handler

2784 .type   ctr32_se_handler,\@abi-omnipotent
2785 .align  16
2786 ctr32_se_handler:
2787         push    %rsi
2788         push    %rdi
2789         push    %rbx
2790         push    %rbp
2791         push    %r12
2792         push    %r13
2793         push    %r14
2794         push    %r15
2795         pushfq
2796         sub     \$64,%rsp

2798         mov     120($context),%rax      # pull context->Rax
2799         mov     248($context),%rbx      # pull context->Rip

2801         lea     .Lctr32_body(%rip),%r10
2802         cmp     %r10,%rbx               # context->Rip<"prologue" label
2803         jb      .Lcommon_seh_tail

2805         mov     152($context),%rax      # pull context->Rsp

2807         lea     .Lctr32_ret(%rip),%r10
2808         cmp     %r10,%rbx
2809         jae     .Lcommon_seh_tail

2811         lea     0x20(%rax),%rsi         # %xmm save area
2812         lea     512($context),%rdi      # &context.Xmm6
2813         mov     \$20,%ecx               # 10*sizeof(%xmm0)/sizeof(%rax)
2814         .long   0xa548f3fc              # cld; rep movsq
2815         lea     0xc8(%rax),%rax         # adjust stack pointer

2817         jmp     .Lcommon_seh_tail
2818 .size   ctr32_se_handler,.-ctr32_se_handler

2820 .type   xts_se_handler,\@abi-omnipotent
2821 .align  16
2822 xts_se_handler:
2823         push    %rsi
2824         push    %rdi
2825         push    %rbx
2826         push    %rbp
2827         push    %r12
2828         push    %r13
```



```
2829         push    %r14
2830         push    %r15
2831         pushfq
2832         sub     \$64,%rsp

2834         mov     120($context),%rax      # pull context->Rax
2835         mov     248($context),%rbx      # pull context->Rip

2837         mov     8($disp),%rsi           # disp->ImageBase
2838         mov     56($disp),%r11          # disp->HandlerData

2840         mov     0(%r11),%r10d           # HandlerData[0]
2841         lea     (%rsi,%r10),%r10        # prologue lable
2842         cmp     %r10,%rbx               # context->Rip<prologue label
2843         jb      .Lcommon_seh_tail

2845         mov     152($context),%rax      # pull context->Rsp

2847         mov     4(%r11),%r10d           # HandlerData[1]
2848         lea     (%rsi,%r10),%r10        # epilogue label
2849         cmp     %r10,%rbx               # context->Rip>=epilogue label
2850         jae     .Lcommon_seh_tail

2852         lea     0x60(%rax),%rsi         # %xmm save area
2853         lea     512($context),%rdi      # & context.Xmm6
2854         mov     \$20,%ecx               # 10*sizeof(%xmm0)/sizeof(%rax)
2855         .long   0xa548f3fc              # cld; rep movsq
2856         lea     0x68+160(%rax),%rax     # adjust stack pointer

2858         jmp     .Lcommon_seh_tail
2859 .size   xts_se_handler,.-xts_se_handler
2860 ___
2861 $code.=<<___;
2862 .type   cbc_se_handler,\@abi-omnipotent
2863 .align  16
2864 cbc_se_handler:
2865         push    %rsi
2866         push    %rdi
2867         push    %rbx
2868         push    %rbp
2869         push    %r12
2870         push    %r13
2871         push    %r14
2872         push    %r15
2873         pushfq
2874         sub     \$64,%rsp

2876         mov     152($context),%rax      # pull context->Rsp
2877         mov     248($context),%rbx      # pull context->Rip

2879         lea     .Lcbc_decrypt(%rip),%r10
2880         cmp     %r10,%rbx               # context->Rip<"prologue" label
2881         jb      .Lcommon_seh_tail

2883         lea     .Lcbc_decrypt_body(%rip),%r10
2884         cmp     %r10,%rbx               # context->Rip<cbc_decrypt_body
2885         jb      .Lrestore_cbc_rax

2887         lea     .Lcbc_ret(%rip),%r10
2888         cmp     %r10,%rbx               # context->Rip>="epilogue" label
2889         jae     .Lcommon_seh_tail

2891         lea     0(%rax),%rsi            # top of stack
2892         lea     512($context),%rdi      # &context.Xmm6
2893         mov     \$8,%ecx                # 4*sizeof(%xmm0)/sizeof(%rax)
2894         .long   0xa548f3fc              # cld; rep movsq
```

```
2895         lea     0x58(%rax),%rax         # adjust stack pointer
2896         jmp     .Lcommon_seh_tail

2898 .Lrestore_cbc_rax:
2899         mov     120($context),%rax

2901 .Lcommon_seh_tail:
2902         mov     8(%rax),%rdi
2903         mov     16(%rax),%rsi
2904         mov     %rax,152($context)      # restore context->Rsp
2905         mov     %rsi,168($context)      # restore context->Rsi
2906         mov     %rdi,176($context)      # restore context->Rdi

2908         mov     40($disp),%rdi          # disp->ContextRecord
2909         mov     $context,%rsi           # context
2910         mov     \$154,%ecx              # sizeof(CONTEXT)
2911         .long   0xa548f3fc              # cld; rep movsq

2913         mov     $disp,%rsi
2914         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
2915         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
2916         mov     0(%rsi),%r8             # arg3, disp->ControlPc
2917         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
2918         mov     40(%rsi),%r10           # disp->ContextRecord
2919         lea     56(%rsi),%r11           # &disp->HandlerData
2920         lea     24(%rsi),%r12           # &disp->EstablisherFrame
2921         mov     %r10,32(%rsp)           # arg5
2922         mov     %r11,40(%rsp)           # arg6
2923         mov     %r12,48(%rsp)           # arg7
2924         mov     %rcx,56(%rsp)           # arg8, (NULL)
2925         call    *__imp_RtlVirtualUnwind(%rip)

2927         mov     \$1,%eax                # ExceptionContinueSearch
2928         add     \$64,%rsp
2929         popfq
2930         pop     %r15
2931         pop     %r14
2932         pop     %r13
2933         pop     %r12
2934         pop     %rbp
2935         pop     %rbx
2936         pop     %rdi
2937         pop     %rsi
2938         ret
2939 .size   cbc_se_handler,.-cbc_se_handler

2941 .section        .pdata
2942 .align  4
2943 ___
2944 $code.=<<___ if ($PREFIX eq "aesni");
2945         .rva    .LSEH_begin_aesni_ecb_encrypt
2946         .rva    .LSEH_end_aesni_ecb_encrypt
2947         .rva    .LSEH_info_ecb

2949         .rva    .LSEH_begin_aesni_ccm64_encrypt_blocks
2950         .rva    .LSEH_end_aesni_ccm64_encrypt_blocks
2951         .rva    .LSEH_info_ccm64_enc

2953         .rva    .LSEH_begin_aesni_ccm64_decrypt_blocks
2954         .rva    .LSEH_end_aesni_ccm64_decrypt_blocks
2955         .rva    .LSEH_info_ccm64_dec

2957         .rva    .LSEH_begin_aesni_ctr32_encrypt_blocks
2958         .rva    .LSEH_end_aesni_ctr32_encrypt_blocks
2959         .rva    .LSEH_info_ctr32
```



```
2961         .rva    .LSEH_begin_aesni_xts_encrypt
2962         .rva    .LSEH_end_aesni_xts_encrypt
2963         .rva    .LSEH_info_xts_enc

2965         .rva    .LSEH_begin_aesni_xts_decrypt
2966         .rva    .LSEH_end_aesni_xts_decrypt
2967         .rva    .LSEH_info_xts_dec
2968 ___
2969 $code.=<<___;
2970         .rva    .LSEH_begin_${PREFIX}_cbc_encrypt
2971         .rva    .LSEH_end_${PREFIX}_cbc_encrypt
2972         .rva    .LSEH_info_cbc

2974         .rva    ${PREFIX}_set_decrypt_key
2975         .rva    .LSEH_end_set_decrypt_key
2976         .rva    .LSEH_info_key

2978         .rva    ${PREFIX}_set_encrypt_key
2979         .rva    .LSEH_end_set_encrypt_key
2980         .rva    .LSEH_info_key
2981 .section        .xdata
2982 .align  8
2983 ___
2984 $code.=<<___ if ($PREFIX eq "aesni");
2985 .LSEH_info_ecb:
2986         .byte   9,0,0,0
2987         .rva    ecb_se_handler
2988 .LSEH_info_ccm64_enc:
2989         .byte   9,0,0,0
2990         .rva    ccm64_se_handler
2991         .rva    .Lccm64_enc_body,.Lccm64_enc_ret        # HandlerData[]
2992 .LSEH_info_ccm64_dec:
2993         .byte   9,0,0,0
2994         .rva    ccm64_se_handler
2995         .rva    .Lccm64_dec_body,.Lccm64_dec_ret        # HandlerData[]
2996 .LSEH_info_ctr32:
2997         .byte   9,0,0,0
2998         .rva    ctr32_se_handler
2999 .LSEH_info_xts_enc:
3000         .byte   9,0,0,0
3001         .rva    xts_se_handler
3002         .rva    .Lxts_enc_body,.Lxts_enc_epilogue       # HandlerData[]
3003 .LSEH_info_xts_dec:
3004         .byte   9,0,0,0
3005         .rva    xts_se_handler
3006         .rva    .Lxts_dec_body,.Lxts_dec_epilogue       # HandlerData[]
3007 ___
3008 $code.=<<___;
3009 .LSEH_info_cbc:
3010         .byte   9,0,0,0
3011         .rva    cbc_se_handler
3012 .LSEH_info_key:
3013         .byte   0x01,0x04,0x01,0x00
3014         .byte   0x04,0x02,0x00,0x00     # sub rsp,8
3015 ___
3016 }

3018 sub rex {
3019   local *opcode=shift;
3020   my ($dst,$src)=@_;
3021   my $rex=0;

3023     $rex|=0x04                  if($dst>=8);
3024     $rex|=0x01                  if($src>=8);
3025     push @opcode,$rex|0x40      if($rex);
3026 }
```

```
3028 sub aesni {
3029   my $line=shift;
3030   my @opcode=(0x66);

3032     if ($line=~/(aeskeygenassist)\s+\$([x0-9a-f]+),\s*%xmm([0-9]+),\s*%xmm([0-9]
3033         rex(\@opcode,$4,$3);
3034         push @opcode,0x0f,0x3a,0xdf;
3035         push @opcode,0xc0|($3&7)|(($4&7)<<3);   # ModR/M
3036         my $c=$2;
3037         push @opcode,$c=~/^0/?oct($c):$c;
3038         return ".byte\t".join(',',@opcode);
3039     }
3040     elsif ($line=~/(aes[a-z]+)\s+%xmm([0-9]+),\s*%xmm([0-9]+)/) {
3041         my %opcodelet = (
3042                 "aesimc" => 0xdb,
3043                 "aesenc" => 0xdc,       "aesenclast" => 0xdd,
3044                 "aesdec" => 0xde,       "aesdeclast" => 0xdf
3045         );
3046         return undef if (!defined($opcodelet{$1}));
3047         rex(\@opcode,$3,$2);
3048         push @opcode,0x0f,0x38,$opcodelet{$1};
3049         push @opcode,0xc0|($2&7)|(($3&7)<<3);   # ModR/M
3050         return ".byte\t".join(',',@opcode);
3051     }
3052     return $line;
3053 }

3055 $code =~ s/\`([^\`]*)\`/eval($1)/gem;
3056 $code =~ s/\b(aes.*%xmm[0-9]+).*$/aesni($1)/gem;

3058 print $code;

3060 close STDOUT;
3061 #endif /* ! codereview */
```

```
**********************************************************
    2664 Wed Aug 13 19:53:05 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/bf-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";
require "cbc.pl";

&asm_init($ARGV[0],"bf-586.pl",$ARGV[$#ARGV] eq "386");

$BF_ROUNDS=16;
$BF_OFF=($BF_ROUNDS+2)*4;
$L="edi";
$R="esi";
$P="ebp";
$tmp1="eax";
$tmp2="ebx";
$tmp3="ecx";
$tmp4="edx";

&BF_encrypt("BF_encrypt",1);
&BF_encrypt("BF_decrypt",0);
&cbc("BF_cbc_encrypt","BF_encrypt","BF_decrypt",1,4,5,3,-1,-1);
&asm_finish();

sub BF_encrypt
        {
        local($name,$enc)=@_;

        &function_begin_B($name,"");

        &comment("");

        &push("ebp");
        &push("ebx");
        &mov($tmp2,&wparam(0));
        &mov($P,&wparam(1));
        &push("esi");
        &push("edi");

        &comment("Load the 2 words");
        &mov($L,&DWP(0,$tmp2,"",0));
        &mov($R,&DWP(4,$tmp2,"",0));

        &xor(   $tmp1,  $tmp1);

        # encrypting part

        if ($enc)
                {
                 &mov($tmp2,&DWP(0,$P,"",0));
                &xor(   $tmp3,  $tmp3);

                &xor($L,$tmp2);
                for ($i=0; $i<$BF_ROUNDS; $i+=2)
                        {
                        &comment("");
                        &comment("Round $i");
                        &BF_ENCRYPT($i+1,$R,$L,$P,$tmp1,$tmp2,$tmp3,$tmp4,1);

                        &comment("");
                        &comment("Round ".sprintf("%d",$i+1));
```



```
                        &BF_ENCRYPT($i+2,$L,$R,$P,$tmp1,$tmp2,$tmp3,$tmp4,1);
                        }
                # &mov($tmp1,&wparam(0)); In last loop
                &mov($tmp4,&DWP(($BF_ROUNDS+1)*4,$P,"",0));
                }
        else
                {
                 &mov($tmp2,&DWP(($BF_ROUNDS+1)*4,$P,"",0));
                &xor(   $tmp3,  $tmp3);

                &xor($L,$tmp2);
                for ($i=$BF_ROUNDS; $i>0; $i-=2)
                        {
                        &comment("");
                        &comment("Round $i");
                        &BF_ENCRYPT($i,$R,$L,$P,$tmp1,$tmp2,$tmp3,$tmp4,0);
                        &comment("");
                        &comment("Round ".sprintf("%d",$i-1));
                        &BF_ENCRYPT($i-1,$L,$R,$P,$tmp1,$tmp2,$tmp3,$tmp4,0);
                        }
                # &mov($tmp1,&wparam(0)); In last loop
                &mov($tmp4,&DWP(0,$P,"",0));
                }

        &xor($R,$tmp4);
        &mov(&DWP(4,$tmp1,"",0),$L);

        &mov(&DWP(0,$tmp1,"",0),$R);
        &function_end($name);
        }

sub BF_ENCRYPT
        {
        local($i,$L,$R,$P,$tmp1,$tmp2,$tmp3,$tmp4,$enc)=@_;

        &mov(   $tmp4,          &DWP(&n2a($i*4),$P,"",0)); # for next round

        &mov(   $tmp2,          $R);
        &xor(   $L,             $tmp4);

        &shr(   $tmp2,          16);
        &mov(   $tmp4,          $R);

        &movb(  &LB($tmp1),     &HB($tmp2));    # A
        &and(   $tmp2,          0xff);          # B

        &movb(  &LB($tmp3),     &HB($tmp4));    # C
        &and(   $tmp4,          0xff);          # D

        &mov(   $tmp1,          &DWP(&n2a($BF_OFF+0x0000),$P,$tmp1,4));
        &mov(   $tmp2,          &DWP(&n2a($BF_OFF+0x0400),$P,$tmp2,4));

        &add(   $tmp2,          $tmp1);
        &mov(   $tmp1,          &DWP(&n2a($BF_OFF+0x0800),$P,$tmp3,4));

        &xor(   $tmp2,          $tmp1);
        &mov(   $tmp4,          &DWP(&n2a($BF_OFF+0x0C00),$P,$tmp4,4));

        &add(   $tmp2,          $tmp4);
        if (($enc && ($i != 16)) || ((!$enc) && ($i != 1)))
                { &xor( $tmp1,          $tmp1); }
        else
                {
                &comment("Load parameter 0 ($i) enc=$enc");
                &mov($tmp1,&wparam(0));
                } # In last loop
```

```
129         &xor(   $L,             $tmp2);
130         # delay
131         }

133 sub n2a
134         {
135         sprintf("%d",$_[0]);
136         }
137 #endif /* ! codereview */
```

```
**********************************************************
   16370 Wed Aug 13 19:53:06 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/bn-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],$0);

$sse2=0;
for (@ARGV) { $sse2=1 if (/-DOPENSSL_IA32_SSE2/); }

&external_label("OPENSSL_ia32cap_P") if ($sse2);

&bn_mul_add_words("bn_mul_add_words");
&bn_mul_words("bn_mul_words");
&bn_sqr_words("bn_sqr_words");
&bn_div_words("bn_div_words");
&bn_add_words("bn_add_words");
&bn_sub_words("bn_sub_words");
&bn_sub_part_words("bn_sub_part_words");

&asm_finish();

sub bn_mul_add_words
        {
        local($name)=@_;

        &function_begin_B($name,$sse2?"EXTRN\t_OPENSSL_ia32cap_P:DWORD":"");

        $r="eax";
        $a="edx";
        $c="ecx";

        if ($sse2) {
                &picmeup("eax","OPENSSL_ia32cap_P");
                &bt(&DWP(0,"eax"),26);
                &jnc(&label("maw_non_sse2"));

                &mov($r,&wparam(0));
                &mov($a,&wparam(1));
                &mov($c,&wparam(2));
                &movd("mm0",&wparam(3));        # mm0 = w
                &pxor("mm1","mm1");             # mm1 = carry_in
                &jmp(&label("maw_sse2_entry"));

        &set_label("maw_sse2_unrolled",16);
                &movd("mm3",&DWP(0,$r,"",0));   # mm3 = r[0]
                &paddq("mm1","mm3");            # mm1 = carry_in + r[0]
                &movd("mm2",&DWP(0,$a,"",0));   # mm2 = a[0]
                &pmuludq("mm2","mm0");          # mm2 = w*a[0]
                &movd("mm4",&DWP(4,$a,"",0));   # mm4 = a[1]
                &pmuludq("mm4","mm0");          # mm4 = w*a[1]
                &movd("mm6",&DWP(8,$a,"",0));   # mm6 = a[2]
                &pmuludq("mm6","mm0");          # mm6 = w*a[2]
                &movd("mm7",&DWP(12,$a,"",0));  # mm7 = a[3]
                &pmuludq("mm7","mm0");          # mm7 = w*a[3]
                &paddq("mm1","mm2");            # mm1 = carry_in + r[0] + w*a[0]
                &movd("mm3",&DWP(4,$r,"",0));   # mm3 = r[1]
                &paddq("mm3","mm4");            # mm3 = r[1] + w*a[1]
                &movd("mm5",&DWP(8,$r,"",0));   # mm5 = r[2]
                &paddq("mm5","mm6");            # mm5 = r[2] + w*a[2]
```



```
                &movd("mm4",&DWP(12,$r,"",0));  # mm4 = r[3]
                &paddq("mm7","mm4");            # mm7 = r[3] + w*a[3]
                &movd(&DWP(0,$r,"",0),"mm1");
                &movd("mm2",&DWP(16,$a,"",0));  # mm2 = a[4]
                &pmuludq("mm2","mm0");          # mm2 = w*a[4]
                &psrlq("mm1",32);               # mm1 = carry0
                &movd("mm4",&DWP(20,$a,"",0));  # mm4 = a[5]
                &pmuludq("mm4","mm0");          # mm4 = w*a[5]
                &paddq("mm1","mm3");            # mm1 = carry0 + r[1] + w*a[1]
                &movd("mm6",&DWP(24,$a,"",0));  # mm6 = a[6]
                &pmuludq("mm6","mm0");          # mm6 = w*a[6]
                &movd(&DWP(4,$r,"",0),"mm1");
                &psrlq("mm1",32);               # mm1 = carry1
                &movd("mm3",&DWP(28,$a,"",0));  # mm3 = a[7]
                &add($a,32);
                &pmuludq("mm3","mm0");          # mm3 = w*a[7]
                &paddq("mm1","mm5");            # mm1 = carry1 + r[2] + w*a[2]
                &movd("mm5",&DWP(16,$r,"",0));  # mm5 = r[4]
                &paddq("mm2","mm5");            # mm2 = r[4] + w*a[4]
                &movd(&DWP(8,$r,"",0),"mm1");
                &psrlq("mm1",32);               # mm1 = carry2
                &paddq("mm1","mm7");            # mm1 = carry2 + r[3] + w*a[3]
                &movd("mm5",&DWP(20,$r,"",0));  # mm5 = r[5]
                &paddq("mm4","mm5");            # mm4 = r[5] + w*a[5]
                &movd(&DWP(12,$r,"",0),"mm1");
                &psrlq("mm1",32);               # mm1 = carry3
                &paddq("mm1","mm2");            # mm1 = carry3 + r[4] + w*a[4]
                &movd("mm5",&DWP(24,$r,"",0));  # mm5 = r[6]
                &paddq("mm6","mm5");            # mm6 = r[6] + w*a[6]
                &movd(&DWP(16,$r,"",0),"mm1");
                &psrlq("mm1",32);               # mm1 = carry4
                &paddq("mm1","mm4");            # mm1 = carry4 + r[5] + w*a[5]
                &movd("mm5",&DWP(28,$r,"",0));  # mm5 = r[7]
                &paddq("mm3","mm5");            # mm3 = r[7] + w*a[7]
                &movd(&DWP(20,$r,"",0),"mm1");
                &psrlq("mm1",32);               # mm1 = carry5
                &paddq("mm1","mm6");            # mm1 = carry5 + r[6] + w*a[6]
                &movd(&DWP(24,$r,"",0),"mm1");
                &psrlq("mm1",32);               # mm1 = carry6
                &paddq("mm1","mm3");            # mm1 = carry6 + r[7] + w*a[7]
                &movd(&DWP(28,$r,"",0),"mm1");
                &lea($r,&DWP(32,$r));
                &psrlq("mm1",32);               # mm1 = carry_out

                &sub($c,8);
                &jz(&label("maw_sse2_exit"));
        &set_label("maw_sse2_entry");
                &test($c,0xfffffff8);
                &jnz(&label("maw_sse2_unrolled"));

        &set_label("maw_sse2_loop",4);
                &movd("mm2",&DWP(0,$a));        # mm2 = a[i]
                &movd("mm3",&DWP(0,$r));        # mm3 = r[i]
                &pmuludq("mm2","mm0");          # a[i] *= w
                &lea($a,&DWP(4,$a));
                &paddq("mm1","mm3");            # carry += r[i]
                &paddq("mm1","mm2");            # carry += a[i]*w
                &movd(&DWP(0,$r),"mm1");        # r[i] = carry_low
                &sub($c,1);
                &psrlq("mm1",32);               # carry = carry_high
                &lea($r,&DWP(4,$r));
                &jnz(&label("maw_sse2_loop"));
        &set_label("maw_sse2_exit");
                &movd("eax","mm1");             # c = carry_out
                &emms();
                &ret();
```

```
129         &set_label("maw_non_sse2",16);
130         }
131
132         # function_begin prologue
133         &push("ebp");
134         &push("ebx");
135         &push("esi");
136         &push("edi");
137
138         &comment("");
139         $Low="eax";
140         $High="edx";
141         $a="ebx";
142         $w="ebp";
143         $r="edi";
144         $c="esi";
145
146         &xor($c,$c);            # clear carry
147         &mov($r,&wparam(0));    #
148
149         &mov("ecx",&wparam(2)); #
150         &mov($a,&wparam(1));    #
151
152         &and("ecx",0xfffffff8); # num / 8
153         &mov($w,&wparam(3));    #
154
155         &push("ecx");           # Up the stack for a tmp variable
156
157         &jz(&label("maw_finish"));
158
159         &set_label("maw_loop",16);
160
161         for ($i=0; $i<32; $i+=4)
162                 {
163                 &comment("Round $i");
164
165                  &mov("eax",&DWP($i,$a));       # *a
166                 &mul($w);                       # *a * w
167                 &add("eax",$c);                 # L(t)+= c
168                 &adc("edx",0);                  # H(t)+=carry
169                  &add("eax",&DWP($i,$r));       # L(t)+= *r
170                 &adc("edx",0);                  # H(t)+=carry
171                  &mov(&DWP($i,$r),"eax");       # *r= L(t);
172                 &mov($c,"edx");                 # c=  H(t);
173                 }
174
175         &comment("");
176         &sub("ecx",8);
177         &lea($a,&DWP(32,$a));
178         &lea($r,&DWP(32,$r));
179         &jnz(&label("maw_loop"));
180
181         &set_label("maw_finish",0);
182         &mov("ecx",&wparam(2)); # get num
183         &and("ecx",7);
184         &jnz(&label("maw_finish2"));    # helps branch prediction
185         &jmp(&label("maw_end"));
186
187         &set_label("maw_finish2",1);
188         for ($i=0; $i<7; $i++)
189                 {
190                 &comment("Tail Round $i");
191                  &mov("eax",&DWP($i*4,$a));     # *a
192                 &mul($w);                       # *a * w
193                 &add("eax",$c);                 # L(t)+=c
```



```
194                 &adc("edx",0);                  # H(t)+=carry
195                  &add("eax",&DWP($i*4,$r));     # L(t)+= *r
196                 &adc("edx",0);                  # H(t)+=carry
197                  &dec("ecx") if ($i != 7-1);
198                 &mov(&DWP($i*4,$r),"eax");      # *r= L(t);
199                  &mov($c,"edx");                # c=  H(t);
200                 &jz(&label("maw_end")) if ($i != 7-1);
201                 }
202         &set_label("maw_end",0);
203         &mov("eax",$c);
204
205         &pop("ecx");    # clear variable from
206
207         &function_end($name);
208         }
209
210 sub bn_mul_words
211         {
212         local($name)=@_;
213
214         &function_begin_B($name,$sse2?"EXTRN\t_OPENSSL_ia32cap_P:DWORD":"");
215
216         $r="eax";
217         $a="edx";
218         $c="ecx";
219
220         if ($sse2) {
221                 &picmeup("eax","OPENSSL_ia32cap_P");
222                 &bt(&DWP(0,"eax"),26);
223                 &jnc(&label("mw_non_sse2"));
224
225                 &mov($r,&wparam(0));
226                 &mov($a,&wparam(1));
227                 &mov($c,&wparam(2));
228                 &movd("mm0",&wparam(3));        # mm0 = w
229                 &pxor("mm1","mm1");             # mm1 = carry = 0
230
231         &set_label("mw_sse2_loop",16);
232                 &movd("mm2",&DWP(0,$a));        # mm2 = a[i]
233                 &pmuludq("mm2","mm0");          # a[i] *= w
234                 &lea($a,&DWP(4,$a));
235                 &paddq("mm1","mm2");            # carry += a[i]*w
236                 &movd(&DWP(0,$r),"mm1");        # r[i] = carry_low
237                 &sub($c,1);
238                 &psrlq("mm1",32);               # carry = carry_high
239                 &lea($r,&DWP(4,$r));
240                 &jnz(&label("mw_sse2_loop"));
241
242                 &movd("eax","mm1");             # return carry
243                 &emms();
244                 &ret();
245         &set_label("mw_non_sse2",16);
246         }
247
248         # function_begin prologue
249         &push("ebp");
250         &push("ebx");
251         &push("esi");
252         &push("edi");
253
254         &comment("");
255         $Low="eax";
256         $High="edx";
257         $a="ebx";
258         $w="ecx";
259         $r="edi";
```

```
260         $c="esi";
261         $num="ebp";
263         &xor($c,$c);            # clear carry
264         &mov($r,&wparam(0));    #
265         &mov($a,&wparam(1));    #
266         &mov($num,&wparam(2));  #
267         &mov($w,&wparam(3));    #
269         &and($num,0xfffffff8);  # num / 8
270         &jz(&label("mw_finish"));
272         &set_label("mw_loop",0);
273         for ($i=0; $i<32; $i+=4)
274                 {
275                 &comment("Round $i");
277                  &mov("eax",&DWP($i,$a,"",0));  # *a
278                 &mul($w);                       # *a * w
279                 &add("eax",$c);                 # L(t)+=c
280                  # XXX
282                 &adc("edx",0);                  # H(t)+=carry
283                  &mov(&DWP($i,$r,"",0),"eax");  # *r= L(t);
285                 &mov($c,"edx");                 # c=  H(t);
286                 }
288         &comment("");
289         &add($a,32);
290         &add($r,32);
291         &sub($num,8);
292         &jz(&label("mw_finish"));
293         &jmp(&label("mw_loop"));
295         &set_label("mw_finish",0);
296         &mov($num,&wparam(2));  # get num
297         &and($num,7);
298         &jnz(&label("mw_finish2"));
299         &jmp(&label("mw_end"));
301         &set_label("mw_finish2",1);
302         for ($i=0; $i<7; $i++)
303                 {
304                 &comment("Tail Round $i");
305                  &mov("eax",&DWP($i*4,$a,"",0));# *a
306                 &mul($w);                       # *a * w
307                 &add("eax",$c);                 # L(t)+=c
308                  # XXX
309                 &adc("edx",0);                  # H(t)+=carry
310                  &mov(&DWP($i*4,$r,"",0),"eax");# *r= L(t);
311                 &mov($c,"edx");                 # c=  H(t);
312                  &dec($num) if ($i != 7-1);
313                 &jz(&label("mw_end")) if ($i != 7-1);
314                 }
315         &set_label("mw_end",0);
316         &mov("eax",$c);
318         &function_end($name);
319         }
321 sub bn_sqr_words
322         {
323         local($name)=@_;
325         &function_begin_B($name,$sse2?"EXTRN\t_OPENSSL_ia32cap_P:DWORD":"");
```



```
327         $r="eax";
328         $a="edx";
329         $c="ecx";
331         if ($sse2) {
332                 &picmeup("eax","OPENSSL_ia32cap_P");
333                 &bt(&DWP(0,"eax"),26);
334                 &jnc(&label("sqr_non_sse2"));
336                 &mov($r,&wparam(0));
337                 &mov($a,&wparam(1));
338                 &mov($c,&wparam(2));
340         &set_label("sqr_sse2_loop",16);
341                 &movd("mm0",&DWP(0,$a));        # mm0 = a[i]
342                 &pmuludq("mm0","mm0");          # a[i] *= a[i]
343                 &lea($a,&DWP(4,$a));            # a++
344                 &movq(&QWP(0,$r),"mm0");        # r[i] = a[i]*a[i]
345                 &sub($c,1);
346                 &lea($r,&DWP(8,$r));            # r += 2
347                 &jnz(&label("sqr_sse2_loop"));
349                 &emms();
350                 &ret();
351         &set_label("sqr_non_sse2",16);
352         }
354         # function_begin prologue
355         &push("ebp");
356         &push("ebx");
357         &push("esi");
358         &push("edi");
360         &comment("");
361         $r="esi";
362         $a="edi";
363         $num="ebx";
365         &mov($r,&wparam(0));    #
366         &mov($a,&wparam(1));    #
367         &mov($num,&wparam(2));  #
369         &and($num,0xfffffff8);  # num / 8
370         &jz(&label("sw_finish"));
372         &set_label("sw_loop",0);
373         for ($i=0; $i<32; $i+=4)
374                 {
375                 &comment("Round $i");
376                 &mov("eax",&DWP($i,$a,"",0));   # *a
377                  # XXX
378                 &mul("eax");                    # *a * *a
379                 &mov(&DWP($i*2,$r,"",0),"eax"); #
380                  &mov(&DWP($i*2+4,$r,"",0),"edx");#
381                 }
383         &comment("");
384         &add($a,32);
385         &add($r,64);
386         &sub($num,8);
387         &jnz(&label("sw_loop"));
389         &set_label("sw_finish",0);
390         &mov($num,&wparam(2));  # get num
391         &and($num,7);
```

```
392         &jz(&label("sw_end"));

394         for ($i=0; $i<7; $i++)
395                 {
396                 &comment("Tail Round $i");
397                 &mov("eax",&DWP($i*4,$a,"",0)); # *a
398                  # XXX
399                 &mul("eax");                    # *a * *a
400                 &mov(&DWP($i*8,$r,"",0),"eax"); #
401                  &dec($num) if ($i != 7-1);
402                 &mov(&DWP($i*8+4,$r,"",0),"edx");
403                  &jz(&label("sw_end")) if ($i != 7-1);
404                 }
405         &set_label("sw_end",0);

407         &function_end($name);
408         }

410 sub bn_div_words
411         {
412         local($name)=@_;

414         &function_begin_B($name,"");
415         &mov("edx",&wparam(0)); #
416         &mov("eax",&wparam(1)); #
417         &mov("ecx",&wparam(2)); #
418         &div("ecx");
419         &ret();
420         &function_end_B($name);
421         }

423 sub bn_add_words
424         {
425         local($name)=@_;

427         &function_begin($name,"");

429         &comment("");
430         $a="esi";
431         $b="edi";
432         $c="eax";
433         $r="ebx";
434         $tmp1="ecx";
435         $tmp2="edx";
436         $num="ebp";

438         &mov($r,&wparam(0));    # get r
439          &mov($a,&wparam(1));   # get a
440         &mov($b,&wparam(2));    # get b
441          &mov($num,&wparam(3)); # get num
442         &xor($c,$c);            # clear carry
443          &and($num,0xfffffff8); # num / 8

445         &jz(&label("aw_finish"));

447         &set_label("aw_loop",0);
448         for ($i=0; $i<8; $i++)
449                 {
450                 &comment("Round $i");

452                 &mov($tmp1,&DWP($i*4,$a,"",0));         # *a
453                  &mov($tmp2,&DWP($i*4,$b,"",0));        # *b
454                 &add($tmp1,$c);
455                  &mov($c,0);
456                 &adc($c,$c);
457                  &add($tmp1,$tmp2);
```



```
458                 &adc($c,0);
459                  &mov(&DWP($i*4,$r,"",0),$tmp1);        # *r
460                 }

462         &comment("");
463         &add($a,32);
464          &add($b,32);
465         &add($r,32);
466          &sub($num,8);
467         &jnz(&label("aw_loop"));

469         &set_label("aw_finish",0);
470         &mov($num,&wparam(3));  # get num
471         &and($num,7);
472          &jz(&label("aw_end"));

474         for ($i=0; $i<7; $i++)
475                 {
476                 &comment("Tail Round $i");
477                 &mov($tmp1,&DWP($i*4,$a,"",0)); # *a
478                  &mov($tmp2,&DWP($i*4,$b,"",0));# *b
479                 &add($tmp1,$c);
480                  &mov($c,0);
481                 &adc($c,$c);
482                  &add($tmp1,$tmp2);
483                 &adc($c,0);
484                  &dec($num) if ($i != 6);
485                 &mov(&DWP($i*4,$r,"",0),$tmp1); # *r
486                  &jz(&label("aw_end")) if ($i != 6);
487                 }
488         &set_label("aw_end",0);

490 #       &mov("eax",$c);         # $c is "eax"

492         &function_end($name);
493         }

495 sub bn_sub_words
496         {
497         local($name)=@_;

499         &function_begin($name,"");

501         &comment("");
502         $a="esi";
503         $b="edi";
504         $c="eax";
505         $r="ebx";
506         $tmp1="ecx";
507         $tmp2="edx";
508         $num="ebp";

510         &mov($r,&wparam(0));    # get r
511          &mov($a,&wparam(1));   # get a
512         &mov($b,&wparam(2));    # get b
513          &mov($num,&wparam(3)); # get num
514         &xor($c,$c);            # clear carry
515          &and($num,0xfffffff8); # num / 8

517         &jz(&label("aw_finish"));

519         &set_label("aw_loop",0);
520         for ($i=0; $i<8; $i++)
521                 {
522                 &comment("Round $i");
```

```
524                 &mov($tmp1,&DWP($i*4,$a,"",0));         # *a
525                  &mov($tmp2,&DWP($i*4,$b,"",0));        # *b
526                 &sub($tmp1,$c);
527                  &mov($c,0);
528                 &adc($c,$c);
529                  &sub($tmp1,$tmp2);
530                 &adc($c,0);
531                  &mov(&DWP($i*4,$r,"",0),$tmp1);        # *r
532                 }

534         &comment("");
535         &add($a,32);
536          &add($b,32);
537         &add($r,32);
538          &sub($num,8);
539         &jnz(&label("aw_loop"));

541         &set_label("aw_finish",0);
542         &mov($num,&wparam(3));  # get num
543         &and($num,7);
544          &jz(&label("aw_end"));

546         for ($i=0; $i<7; $i++)
547                 {
548                 &comment("Tail Round $i");
549                 &mov($tmp1,&DWP($i*4,$a,"",0)); # *a
550                  &mov($tmp2,&DWP($i*4,$b,"",0));# *b
551                 &sub($tmp1,$c);
552                  &mov($c,0);
553                 &adc($c,$c);
554                  &sub($tmp1,$tmp2);
555                 &adc($c,0);
556                  &dec($num) if ($i != 6);
557                 &mov(&DWP($i*4,$r,"",0),$tmp1); # *r
558                  &jz(&label("aw_end")) if ($i != 6);
559                 }
560         &set_label("aw_end",0);

562 #       &mov("eax",$c);         # $c is "eax"

564         &function_end($name);
565         }

567 sub bn_sub_part_words
568         {
569         local($name)=@_;

571         &function_begin($name,"");

573         &comment("");
574         $a="esi";
575         $b="edi";
576         $c="eax";
577         $r="ebx";
578         $tmp1="ecx";
579         $tmp2="edx";
580         $num="ebp";

582         &mov($r,&wparam(0));    # get r
583          &mov($a,&wparam(1));   # get a
584         &mov($b,&wparam(2));    # get b
585          &mov($num,&wparam(3)); # get num
586         &xor($c,$c);            # clear carry
587          &and($num,0xfffffff8); # num / 8

589         &jz(&label("aw_finish"));
```



```
591         &set_label("aw_loop",0);
592         for ($i=0; $i<8; $i++)
593                 {
594                 &comment("Round $i");

596                 &mov($tmp1,&DWP($i*4,$a,"",0));         # *a
597                  &mov($tmp2,&DWP($i*4,$b,"",0));        # *b
598                 &sub($tmp1,$c);
599                  &mov($c,0);
600                 &adc($c,$c);
601                  &sub($tmp1,$tmp2);
602                 &adc($c,0);
603                  &mov(&DWP($i*4,$r,"",0),$tmp1);        # *r
604                 }

606         &comment("");
607         &add($a,32);
608          &add($b,32);
609         &add($r,32);
610          &sub($num,8);
611         &jnz(&label("aw_loop"));

613         &set_label("aw_finish",0);
614         &mov($num,&wparam(3));  # get num
615         &and($num,7);
616          &jz(&label("aw_end"));

618         for ($i=0; $i<7; $i++)
619                 {
620                 &comment("Tail Round $i");
621                 &mov($tmp1,&DWP(0,$a,"",0));    # *a
622                  &mov($tmp2,&DWP(0,$b,"",0));# *b
623                 &sub($tmp1,$c);
624                  &mov($c,0);
625                 &adc($c,$c);
626                  &sub($tmp1,$tmp2);
627                 &adc($c,0);
628                 &mov(&DWP(0,$r,"",0),$tmp1);    # *r
629                 &add($a, 4);
630                 &add($b, 4);
631                 &add($r, 4);
632                  &dec($num) if ($i != 6);
633                  &jz(&label("aw_end")) if ($i != 6);
634                 }
635         &set_label("aw_end",0);

637         &cmp(&wparam(4),0);
638         &je(&label("pw_end"));

640         &mov($num,&wparam(4));  # get dl
641         &cmp($num,0);
642         &je(&label("pw_end"));
643         &jge(&label("pw_pos"));

645         &comment("pw_neg");
646         &mov($tmp2,0);
647         &sub($tmp2,$num);
648         &mov($num,$tmp2);
649         &and($num,0xfffffff8);  # num / 8
650         &jz(&label("pw_neg_finish"));

652         &set_label("pw_neg_loop",0);
653         for ($i=0; $i<8; $i++)
654         {
655             &comment("dl<0 Round $i");
```

```
657             &mov($tmp1,0);
658             &mov($tmp2,&DWP($i*4,$b,"",0));     # *b
659             &sub($tmp1,$c);
660             &mov($c,0);
661             &adc($c,$c);
662             &sub($tmp1,$tmp2);
663             &adc($c,0);
664             &mov(&DWP($i*4,$r,"",0),$tmp1);     # *r
665         }

667         &comment("");
668         &add($b,32);
669         &add($r,32);
670         &sub($num,8);
671         &jnz(&label("pw_neg_loop"));

673         &set_label("pw_neg_finish",0);
674         &mov($tmp2,&wparam(4)); # get dl
675         &mov($num,0);
676         &sub($num,$tmp2);
677         &and($num,7);
678         &jz(&label("pw_end"));

680         for ($i=0; $i<7; $i++)
681         {
682             &comment("dl<0 Tail Round $i");
683             &mov($tmp1,0);
684             &mov($tmp2,&DWP($i*4,$b,"",0));# *b
685             &sub($tmp1,$c);
686             &mov($c,0);
687             &adc($c,$c);
688             &sub($tmp1,$tmp2);
689             &adc($c,0);
690             &dec($num) if ($i != 6);
691             &mov(&DWP($i*4,$r,"",0),$tmp1);     # *r
692             &jz(&label("pw_end")) if ($i != 6);
693         }

695         &jmp(&label("pw_end"));

697         &set_label("pw_pos",0);

699         &and($num,0xfffffff8);  # num / 8
700         &jz(&label("pw_pos_finish"));

702         &set_label("pw_pos_loop",0);

704         for ($i=0; $i<8; $i++)
705         {
706             &comment("dl>0 Round $i");

708             &mov($tmp1,&DWP($i*4,$a,"",0));     # *a
709             &sub($tmp1,$c);
710             &mov(&DWP($i*4,$r,"",0),$tmp1);     # *r
711             &jnc(&label("pw_nc".$i));
712         }

714         &comment("");
715         &add($a,32);
716         &add($r,32);
717         &sub($num,8);
718         &jnz(&label("pw_pos_loop"));

720         &set_label("pw_pos_finish",0);
721         &mov($num,&wparam(4));  # get dl
```



```
722         &and($num,7);
723         &jz(&label("pw_end"));

725         for ($i=0; $i<7; $i++)
726         {
727             &comment("dl>0 Tail Round $i");
728             &mov($tmp1,&DWP($i*4,$a,"",0));     # *a
729             &sub($tmp1,$c);
730             &mov(&DWP($i*4,$r,"",0),$tmp1);     # *r
731             &jnc(&label("pw_tail_nc".$i));
732             &dec($num) if ($i != 6);
733             &jz(&label("pw_end")) if ($i != 6);
734         }
735         &mov($c,1);
736         &jmp(&label("pw_end"));

738         &set_label("pw_nc_loop",0);
739         for ($i=0; $i<8; $i++)
740         {
741             &mov($tmp1,&DWP($i*4,$a,"",0));     # *a
742             &mov(&DWP($i*4,$r,"",0),$tmp1);     # *r
743             &set_label("pw_nc".$i,0);
744         }

746         &comment("");
747         &add($a,32);
748         &add($r,32);
749         &sub($num,8);
750         &jnz(&label("pw_nc_loop"));

752         &mov($num,&wparam(4));  # get dl
753         &and($num,7);
754         &jz(&label("pw_nc_end"));

756         for ($i=0; $i<7; $i++)
757         {
758             &mov($tmp1,&DWP($i*4,$a,"",0));     # *a
759             &mov(&DWP($i*4,$r,"",0),$tmp1);     # *r
760             &set_label("pw_tail_nc".$i,0);
761             &dec($num) if ($i != 6);
762             &jz(&label("pw_nc_end")) if ($i != 6);
763         }

765         &set_label("pw_nc_end",0);
766         &mov($c,0);

768         &set_label("pw_end",0);

770 #       &mov("eax",$c);         # $c is "eax"

772         &function_end($name);
773         }
774 #endif /* ! codereview */
```

```
**********************************************************
   73457 Wed Aug 13 19:53:06 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/bsaes-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

######################################################################
### AES-128 [originally in CTR mode]                               ###
### bitsliced implementation for Intel Core 2 processors           ###
### requires support of SSE extensions up to SSSE3                 ###
### Author: Emilia K^/sper and Peter Schwabe                       ###
### Date: 2009-03-19                                               ###
### Public domain                                                  ###
###                                                                ###
### See http://homes.esat.kuleuven.be/~ekasper/#software for       ###
### further information.                                           ###
######################################################################
#
# September 2011.
#
# Started as transliteration to "perlasm" the original code has
# undergone following changes:
#
# - code was made position-independent;
# - rounds were folded into a loop resulting in >5x size reduction
#   from 12.5KB to 2.2KB;
# - above was possibile thanks to mixcolumns() modification that
#   allowed to feed its output back to aesenc[last], this was
#   achieved at cost of two additional inter-registers moves;
# - some instruction reordering and interleaving;
# - this module doesn't implement key setup subroutine, instead it
#   relies on conversion of "conventional" key schedule as returned
#   by AES_set_encrypt_key (see discussion below);
# - first and last round keys are treated differently, which allowed
#   to skip one shiftrows(), reduce bit-sliced key schedule and
#   speed-up conversion by 22%;
# - support for 192- and 256-bit keys was added;
#
# Resulting performance in CPU cycles spent to encrypt one byte out
# of 4096-byte buffer with 128-bit key is:
#
#               Emilia's        this(*)         difference
#
# Core 2        9.30            8.69            +7%
# Nehalem(**)   7.63            6.98            +9%
# Atom          17.1            17.4            -2%(***)
#
# (*)   Comparison is not completely fair, because "this" is ECB,
#       i.e. no extra processing such as counter values calculation
#       and xor-ing input as in Emilia's CTR implementation is
#       performed. However, the CTR calculations stand for not more
#       than 1% of total time, so comparison is *rather* fair.
#
# (**)  Results were collected on Westmere, which is considered to
#       be equivalent to Nehalem for this code.
#
# (***) Slowdown on Atom is rather strange per se, because original
#       implementation has a number of 9+-bytes instructions, which
#       are bad for Atom front-end, and which I eliminated completely.
#       In attempt to address deterioration sbox() was tested in FP
#       SIMD "domain" (movaps instead of movdqa, xorps instead of
#       pxor, etc.). While it resulted in nominal 4% improvement on
#       Atom, it hurted Westmere by more than 2x factor.
#
# As for key schedule conversion subroutine. Interface to OpenSSL
```



```
# relies on per-invocation on-the-fly conversion. This naturally
# has impact on performance, especially for short inputs. Conversion
# time in CPU cycles and its ratio to CPU cycles spent in 8x block
# function is:
#
#               conversion      conversion/8x block
# Core 2        240             0.22
# Nehalem       180             0.20
# Atom          430             0.19
#
# The ratio values mean that 128-byte blocks will be processed
# 16-18% slower, 256-byte blocks - 9-10%, 384-byte blocks - 6-7%,
# etc. Then keep in mind that input sizes not divisible by 128 are
# *effectively* slower, especially shortest ones, e.g. consecutive
# 144-byte blocks are processed 44% slower than one would expect,
# 272 - 29%, 400 - 22%, etc. Yet, despite all these "shortcomings"
# it's still faster than ["hyper-threading-safe" code path in]
# aes-x86_64.pl on all lengths above 64 bytes...
#
# October 2011.
#
# Add decryption procedure. Performance in CPU cycles spent to decrypt
# one byte out of 4096-byte buffer with 128-bit key is:
#
# Core 2        9.83
# Nehalem       7.74
# Atom          19.0
#
# November 2011.
#
# Add bsaes_xts_[en|de]crypt. Less-than-80-bytes-block performance is
# suboptimal, but XTS is meant to be used with larger blocks...
#
#                                               <appro@openssl.org>

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

my ($inp,$out,$len,$key,$ivp)=("%rdi","%rsi","%rdx","%rcx");
my @XMM=map("%xmm$_",(15,0..14));       # best on Atom, +10% over (0..15)
my $ecb=0;      # suppress unreferenced ECB subroutines, spare some space...

{
my ($key,$rounds,$const)=("%rax","%r10d","%r11");

sub Sbox {
# input in  lsb > [b0, b1, b2, b3, b4, b5, b6, b7] < msb
# output in lsb > [b0, b1, b4, b6, b3, b7, b2, b5] < msb
my @b=@_[0..7];
my @t=@_[8..11];
my @s=@_[12..15];
        &InBasisChange  (@b);
        &Inv_GF256      (@b[6,5,0,3,7,1,4,2],@t,@s);
        &OutBasisChange (@b[7,1,4,2,6,5,0,3]);
}
```

```
sub InBasisChange {
# input in  lsb > [b0, b1, b2, b3, b4, b5, b6, b7] < msb
# output in lsb > [b6, b5, b0, b3, b7, b1, b4, b2] < msb
my @b=@_[0..7];
$code.=<<___;
        pxor    @b[6], @b[5]
        pxor    @b[1], @b[2]
        pxor    @b[0], @b[3]
        pxor    @b[2], @b[6]
        pxor    @b[0], @b[5]

        pxor    @b[3], @b[6]
        pxor    @b[7], @b[3]
        pxor    @b[5], @b[7]
        pxor    @b[4], @b[3]
        pxor    @b[5], @b[4]
        pxor    @b[1], @b[3]

        pxor    @b[7], @b[2]
        pxor    @b[5], @b[1]
___
}

sub OutBasisChange {
# input in  lsb > [b0, b1, b2, b3, b4, b5, b6, b7] < msb
# output in lsb > [b6, b1, b2, b4, b7, b0, b3, b5] < msb
my @b=@_[0..7];
$code.=<<___;
        pxor    @b[6], @b[0]
        pxor    @b[4], @b[1]
        pxor    @b[0], @b[2]
        pxor    @b[6], @b[4]
        pxor    @b[1], @b[6]

        pxor    @b[5], @b[1]
        pxor    @b[3], @b[5]
        pxor    @b[7], @b[3]
        pxor    @b[5], @b[7]
        pxor    @b[5], @b[2]

        pxor    @b[7], @b[4]
___
}

sub InvSbox {
# input in lsb  > [b0, b1, b2, b3, b4, b5, b6, b7] < msb
# output in lsb > [b0, b1, b6, b4, b2, b7, b3, b5] < msb
my @b=@_[0..7];
my @t=@_[8..11];
my @s=@_[12..15];
        &InvInBasisChange       (@b);
        &Inv_GF256              (@b[5,1,2,6,3,7,0,4],@t,@s);
        &InvOutBasisChange      (@b[3,7,0,4,5,1,2,6]);
}

sub InvInBasisChange {          # OutBasisChange in reverse
my @b=@_[5,1,2,6,3,7,0,4];
$code.=<<___
        pxor    @b[7], @b[4]

        pxor    @b[5], @b[7]
        pxor    @b[5], @b[2]
        pxor    @b[7], @b[3]
        pxor    @b[3], @b[5]
        pxor    @b[5], @b[1]
```



```
        pxor    @b[1], @b[6]
        pxor    @b[0], @b[2]
        pxor    @b[6], @b[4]
        pxor    @b[6], @b[0]
        pxor    @b[4], @b[1]
___
}

sub InvOutBasisChange {         # InBasisChange in reverse
my @b=@_[2,5,7,3,6,1,0,4];
$code.=<<___;
        pxor    @b[5], @b[1]
        pxor    @b[7], @b[2]

        pxor    @b[1], @b[3]
        pxor    @b[5], @b[4]
        pxor    @b[5], @b[7]
        pxor    @b[4], @b[3]
         pxor   @b[0], @b[5]
        pxor    @b[7], @b[3]
         pxor   @b[2], @b[6]
         pxor   @b[1], @b[2]
        pxor    @b[3], @b[6]

        pxor    @b[0], @b[3]
        pxor    @b[6], @b[5]
___
}

sub Mul_GF4 {
#;*************************************************************
#;* Mul_GF4: Input x0-x1,y0-y1 Output x0-x1 Temp t0 (8) *
#;*************************************************************
my ($x0,$x1,$y0,$y1,$t0)=@_;
$code.=<<___;
        movdqa  $y0, $t0
        pxor    $y1, $t0
        pand    $x0, $t0
        pxor    $x1, $x0
        pand    $y0, $x1
        pand    $y1, $x0
        pxor    $x1, $x0
        pxor    $t0, $x1
___
}

sub Mul_GF4_N {                         # not used, see next subroutine
# multiply and scale by N
my ($x0,$x1,$y0,$y1,$t0)=@_;
$code.=<<___;
        movdqa  $y0, $t0
        pxor    $y1, $t0
        pand    $x0, $t0
        pxor    $x1, $x0
        pand    $y0, $x1
        pand    $y1, $x0
        pxor    $x0, $x1
        pxor    $t0, $x0
___
}

sub Mul_GF4_N_GF4 {
# interleaved Mul_GF4_N and Mul_GF4
my ($x0,$x1,$y0,$y1,$t0,
    $x2,$x3,$y2,$y3,$t1)=@_;
```

```
$code.=<<___;
        movdqa  $y0, $t0
         movdqa $y2, $t1
        pxor    $y1, $t0
         pxor   $y3, $t1
        pand    $x0, $t0
         pand   $x2, $t1
        pxor    $x1, $x0
         pxor   $x3, $x2
        pand    $y0, $x1
         pand   $y2, $x3
        pand    $y1, $x0
         pand   $y3, $x2
        pxor    $x0, $x1
         pxor   $x3, $x2
        pxor    $t0, $x0
         pxor   $t1, $x3
___
}
sub Mul_GF16_2 {
my @x=@_[0..7];
my @y=@_[8..11];
my @t=@_[12..15];
$code.=<<___;
        movdqa  @x[0], @t[0]
        movdqa  @x[1], @t[1]
___
        &Mul_GF4        (@x[0], @x[1], @y[0], @y[1], @t[2]);
$code.=<<___;
        pxor    @x[2], @t[0]
        pxor    @x[3], @t[1]
        pxor    @y[2], @y[0]
        pxor    @y[3], @y[1]
___
        Mul_GF4_N_GF4   (@t[0], @t[1], @y[0], @y[1], @t[3],
                         @x[2], @x[3], @y[2], @y[3], @t[2]);
$code.=<<___;
        pxor    @t[0], @x[0]
        pxor    @t[0], @x[2]
        pxor    @t[1], @x[1]
        pxor    @t[1], @x[3]

        movdqa  @x[4], @t[0]
        movdqa  @x[5], @t[1]
        pxor    @x[6], @t[0]
        pxor    @x[7], @t[1]
___
        &Mul_GF4_N_GF4  (@t[0], @t[1], @y[0], @y[1], @t[3],
                         @x[6], @x[7], @y[2], @y[3], @t[2]);
$code.=<<___;
        pxor    @y[2], @y[0]
        pxor    @y[3], @y[1]
___
        &Mul_GF4        (@x[4], @x[5], @y[0], @y[1], @t[3]);
$code.=<<___;
        pxor    @t[0], @x[4]
        pxor    @t[0], @x[6]
        pxor    @t[1], @x[5]
        pxor    @t[1], @x[7]
___
}
sub Inv_GF256 {
#;*********************************************************************
#;* Inv_GF256: Input x0-x7 Output x0-x7 Temp t0-t3,s0-s3 (144)       *
#;*********************************************************************
my @x=@_[0..7];
```



```
my @t=@_[8..11];
my @s=@_[12..15];
# direct optimizations from hardware
$code.=<<___;
        movdqa  @x[4], @t[3]
        movdqa  @x[5], @t[2]
        movdqa  @x[1], @t[1]
        movdqa  @x[7], @s[1]
        movdqa  @x[0], @s[0]

        pxor    @x[6], @t[3]
        pxor    @x[7], @t[2]
        pxor    @x[3], @t[1]
         movdqa @t[3], @s[2]
        pxor    @x[6], @s[1]
         movdqa @t[2], @t[0]
        pxor    @x[2], @s[0]
         movdqa @t[3], @s[3]

        por     @t[1], @t[2]
        por     @s[0], @t[3]
        pxor    @t[0], @s[3]
        pand    @s[0], @s[2]
        pxor    @t[1], @s[0]
        pand    @t[1], @t[0]
        pand    @s[0], @s[3]
        movdqa  @x[3], @s[0]
        pxor    @x[2], @s[0]
        pand    @s[0], @s[1]
        pxor    @s[1], @t[3]
        pxor    @s[1], @t[2]
        movdqa  @x[4], @s[1]
        movdqa  @x[1], @s[0]
        pxor    @x[5], @s[1]
        pxor    @x[0], @s[0]
        movdqa  @s[1], @t[1]
        pand    @s[0], @s[1]
        por     @s[0], @t[1]
        pxor    @s[1], @t[0]
        pxor    @s[3], @t[3]
        pxor    @s[2], @t[2]
        pxor    @s[3], @t[1]
        movdqa  @x[7], @s[0]
        pxor    @s[2], @t[0]
        movdqa  @x[6], @s[1]
        pxor    @s[2], @t[1]
        movdqa  @x[5], @s[2]
        pand    @x[3], @s[0]
        movdqa  @x[4], @s[3]
        pand    @x[2], @s[1]
        pand    @x[1], @s[2]
        por     @x[0], @s[3]
        pxor    @s[0], @t[3]
        pxor    @s[1], @t[2]
        pxor    @s[2], @t[1]
        pxor    @s[3], @t[0]

        #Inv_GF16 \t0, \t1, \t2, \t3, \s0, \s1, \s2, \s3

        # new smaller inversion

        movdqa  @t[3], @s[0]
        pand    @t[1], @t[3]
        pxor    @t[2], @s[0]

        movdqa  @t[0], @s[2]
```

```
392         movdqa  @s[0], @s[3]
393         pxor    @t[3], @s[2]
394         pand    @s[2], @s[3]

396         movdqa  @t[1], @s[1]
397         pxor    @t[2], @s[3]
398         pxor    @t[0], @s[1]

400         pxor    @t[2], @t[3]

402         pand    @t[3], @s[1]

404         movdqa  @s[2], @t[2]
405         pxor    @t[0], @s[1]

407         pxor    @s[1], @t[2]
408         pxor    @s[1], @t[1]

410         pand    @t[0], @t[2]

412         pxor    @t[2], @s[2]
413         pxor    @t[2], @t[1]

415         pand    @s[3], @s[2]

417         pxor    @s[0], @s[2]
418 ___
419 # output in s3, s2, s1, t1

421 # Mul_GF16_2 \x0, \x1, \x2, \x3, \x4, \x5, \x6, \x7, \t2, \t3, \t0, \t1, \s0, \s

423 # Mul_GF16_2 \x0, \x1, \x2, \x3, \x4, \x5, \x6, \x7, \s3, \s2, \s1, \t1, \s0, \t
424         &Mul_GF16_2(@x,@s[3,2,1],@t[1],@s[0],@t[0,2,3]);

426 ### output msb > [x3,x2,x1,x0,x7,x6,x5,x4] < lsb
427 }

429 # AES linear components

431 sub ShiftRows {
432 my @x=@_[0..7];
433 my $mask=pop;
434 $code.=<<___;
435         pxor    0x00($key),@x[0]
436         pxor    0x10($key),@x[1]
437         pshufb  $mask,@x[0]
438         pxor    0x20($key),@x[2]
439         pshufb  $mask,@x[1]
440         pxor    0x30($key),@x[3]
441         pshufb  $mask,@x[2]
442         pxor    0x40($key),@x[4]
443         pshufb  $mask,@x[3]
444         pxor    0x50($key),@x[5]
445         pshufb  $mask,@x[4]
446         pxor    0x60($key),@x[6]
447         pshufb  $mask,@x[5]
448         pxor    0x70($key),@x[7]
449         pshufb  $mask,@x[6]
450         lea     0x80($key),$key
451         pshufb  $mask,@x[7]
452 ___
453 }

455 sub MixColumns {
456 # modified to emit output in order suitable for feeding back to aesenc[last]
457 my @x=@_[0..7];
```



```
458 my @t=@_[8..15];
459 my $inv=@_[16]; # optional
460 $code.=<<___;
461         pshufd  \$0x93, @x[0], @t[0]    # x0 <<< 32
462         pshufd  \$0x93, @x[1], @t[1]
463          pxor   @t[0], @x[0]            # x0 ^ (x0 <<< 32)
464         pshufd  \$0x93, @x[2], @t[2]
465          pxor   @t[1], @x[1]
466         pshufd  \$0x93, @x[3], @t[3]
467          pxor   @t[2], @x[2]
468         pshufd  \$0x93, @x[4], @t[4]
469          pxor   @t[3], @x[3]
470         pshufd  \$0x93, @x[5], @t[5]
471          pxor   @t[4], @x[4]
472         pshufd  \$0x93, @x[6], @t[6]
473          pxor   @t[5], @x[5]
474         pshufd  \$0x93, @x[7], @t[7]
475          pxor   @t[6], @x[6]
476          pxor   @t[7], @x[7]

478         pxor    @x[0], @t[1]
479         pxor    @x[7], @t[0]
480         pxor    @x[7], @t[1]
481          pshufd \$0x4E, @x[0], @x[0]    # (x0 ^ (x0 <<< 32)) <<< 64)
482         pxor    @x[1], @t[2]
483          pshufd \$0x4E, @x[1], @x[1]
484         pxor    @x[4], @t[5]
485          pxor   @t[0], @x[0]
486         pxor    @x[5], @t[6]
487          pxor   @t[1], @x[1]
488         pxor    @x[3], @t[4]
489          pshufd \$0x4E, @x[4], @t[0]
490         pxor    @x[6], @t[7]
491          pshufd \$0x4E, @x[5], @t[1]
492         pxor    @x[2], @t[3]
493          pshufd \$0x4E, @x[3], @x[4]
494         pxor    @x[7], @t[3]
495          pshufd \$0x4E, @x[7], @x[5]
496         pxor    @x[7], @t[4]
497          pshufd \$0x4E, @x[6], @x[3]
498         pxor    @t[4], @t[0]
499          pshufd \$0x4E, @x[2], @x[6]
500         pxor    @t[5], @t[1]
501 ___
502 $code.=<<___ if (!$inv);
503         pxor    @t[3], @x[4]
504         pxor    @t[7], @x[5]
505         pxor    @t[6], @x[3]
506          movdqa @t[0], @x[2]
507         pxor    @t[2], @x[6]
508          movdqa @t[1], @x[7]
509 ___
510 $code.=<<___ if ($inv);
511         pxor    @x[4], @t[3]
512         pxor    @t[7], @x[5]
513         pxor    @x[3], @t[6]
514          movdqa @t[0], @x[3]
515         pxor    @t[2], @x[6]
516          movdqa @t[6], @x[2]
517          movdqa @t[1], @x[7]
518          movdqa @x[6], @x[4]
519          movdqa @t[3], @x[6]
520 ___
521 }

523 sub InvMixColumns_orig {
```

```
524 my @x=@_[0..7];
525 my @t=@_[8..15];

527 $code.=<<___;
528         # multiplication by 0x0e
529         pshufd  \$0x93, @x[7], @t[7]
530         movdqa  @x[2], @t[2]
531         pxor    @x[5], @x[7]            # 7 5
532         pxor    @x[5], @x[2]            # 2 5
533         pshufd  \$0x93, @x[0], @t[0]
534         movdqa  @x[5], @t[5]
535         pxor    @x[0], @x[5]            # 5 0           [1]
536         pxor    @x[1], @x[0]            # 0 1
537         pshufd  \$0x93, @x[1], @t[1]
538         pxor    @x[2], @x[1]            # 1 25
539         pxor    @x[6], @x[0]            # 01 6          [2]
540         pxor    @x[3], @x[1]            # 125 3         [4]
541         pshufd  \$0x93, @x[3], @t[3]
542         pxor    @x[0], @x[2]            # 25 016        [3]
543         pxor    @x[7], @x[3]            # 3 75
544         pxor    @x[6], @x[7]            # 75 6          [0]
545         pshufd  \$0x93, @x[6], @t[6]
546         movdqa  @x[4], @t[4]
547         pxor    @x[4], @x[6]            # 6 4
548         pxor    @x[3], @x[4]            # 4 375         [6]
549         pxor    @x[7], @x[3]            # 375 756=36
550         pxor    @t[5], @x[6]            # 64 5          [7]
551         pxor    @t[2], @x[3]            # 36 2
552         pxor    @t[4], @x[3]            # 362 4         [5]
553         pshufd  \$0x93, @t[5], @t[5]
554 ___
555                                         my @y = @x[7,5,0,2,1,3,4,6];
556 $code.=<<___;
557         # multiplication by 0x0b
558         pxor    @y[0], @y[1]
559         pxor    @t[0], @y[0]
560         pxor    @t[1], @y[1]
561         pshufd  \$0x93, @t[2], @t[2]
562         pxor    @t[5], @y[0]
563         pxor    @t[6], @y[1]
564         pxor    @t[7], @y[0]
565         pshufd  \$0x93, @t[4], @t[4]
566         pxor    @t[6], @t[7]            # clobber t[7]
567         pxor    @y[0], @y[1]

569         pxor    @t[0], @y[3]
570         pshufd  \$0x93, @t[0], @t[0]
571         pxor    @t[1], @y[2]
572         pxor    @t[1], @y[4]
573         pxor    @t[2], @y[2]
574         pshufd  \$0x93, @t[1], @t[1]
575         pxor    @t[2], @y[3]
576         pxor    @t[2], @y[5]
577         pxor    @t[7], @y[2]
578         pshufd  \$0x93, @t[2], @t[2]
579         pxor    @t[3], @y[3]
580         pxor    @t[3], @y[6]
581         pxor    @t[3], @y[4]
582         pshufd  \$0x93, @t[3], @t[3]
583         pxor    @t[4], @y[7]
584         pxor    @t[4], @y[5]
585         pxor    @t[7], @y[7]
586         pxor    @t[5], @y[3]
587         pxor    @t[4], @y[4]
588         pxor    @t[5], @t[7]            # clobber t[7] even more
```



```
590         pxor    @t[7], @y[5]
591         pshufd  \$0x93, @t[4], @t[4]
592         pxor    @t[7], @y[6]
593         pxor    @t[7], @y[4]

595         pxor    @t[5], @t[7]
596         pshufd  \$0x93, @t[5], @t[5]
597         pxor    @t[6], @t[7]            # restore t[7]

599         # multiplication by 0x0d
600         pxor    @y[7], @y[4]
601         pxor    @t[4], @y[7]
602         pshufd  \$0x93, @t[6], @t[6]
603         pxor    @t[0], @y[2]
604         pxor    @t[5], @y[7]
605         pxor    @t[2], @y[2]
606         pshufd  \$0x93, @t[7], @t[7]

608         pxor    @y[1], @y[3]
609         pxor    @t[1], @y[1]
610         pxor    @t[0], @y[0]
611         pxor    @t[0], @y[3]
612         pxor    @t[5], @y[1]
613         pxor    @t[5], @y[0]
614         pxor    @t[7], @y[1]
615         pshufd  \$0x93, @t[0], @t[0]
616         pxor    @t[6], @y[0]
617         pxor    @y[1], @y[3]
618         pxor    @t[1], @y[4]
619         pshufd  \$0x93, @t[1], @t[1]

621         pxor    @t[7], @y[7]
622         pxor    @t[2], @y[4]
623         pxor    @t[2], @y[5]
624         pshufd  \$0x93, @t[2], @t[2]
625         pxor    @t[6], @y[2]
626         pxor    @t[3], @t[6]            # clobber t[6]
627         pxor    @y[7], @y[4]
628         pxor    @t[6], @y[3]

630         pxor    @t[6], @y[6]
631         pxor    @t[5], @y[5]
632         pxor    @t[4], @y[6]
633         pshufd  \$0x93, @t[4], @t[4]
634         pxor    @t[6], @y[5]
635         pxor    @t[7], @y[6]
636         pxor    @t[3], @t[6]            # restore t[6]

638         pshufd  \$0x93, @t[5], @t[5]
639         pshufd  \$0x93, @t[6], @t[6]
640         pshufd  \$0x93, @t[7], @t[7]
641         pshufd  \$0x93, @t[3], @t[3]

643         # multiplication by 0x09
644         pxor    @y[1], @y[4]
645         pxor    @y[1], @t[1]            # t[1]=y[1]
646         pxor    @t[5], @t[0]            # clobber t[0]
647         pxor    @t[5], @t[1]
648         pxor    @t[0], @y[3]
649         pxor    @y[0], @t[0]            # t[0]=y[0]
650         pxor    @t[6], @t[1]
651         pxor    @t[7], @t[6]            # clobber t[6]
652         pxor    @t[1], @y[4]
653         pxor    @t[4], @y[7]
654         pxor    @y[4], @t[4]            # t[4]=y[4]
655         pxor    @t[3], @y[6]
```

```
656         pxor    @y[3], @t[3]            # t[3]=y[3]
657         pxor    @t[2], @y[5]
658         pxor    @y[2], @t[2]            # t[2]=y[2]
659         pxor    @t[7], @t[3]
660         pxor    @y[5], @t[5]            # t[5]=y[5]
661         pxor    @t[6], @t[2]
662         pxor    @t[6], @t[5]
663         pxor    @y[6], @t[6]            # t[6]=y[6]
664         pxor    @y[7], @t[7]            # t[7]=y[7]

666         movdqa  @t[0],@XMM[0]
667         movdqa  @t[1],@XMM[1]
668         movdqa  @t[2],@XMM[2]
669         movdqa  @t[3],@XMM[3]
670         movdqa  @t[4],@XMM[4]
671         movdqa  @t[5],@XMM[5]
672         movdqa  @t[6],@XMM[6]
673         movdqa  @t[7],@XMM[7]
674 ___
675 }

677 sub InvMixColumns {
678 my @x=@_[0..7];
679 my @t=@_[8..15];

681 # Thanks to Jussi Kivilinna for providing pointer to
682 #
683 # | 0e 0b 0d 09 |   | 02 03 01 01 |   | 05 00 04 00 |
684 # | 09 0e 0b 0d | = | 01 02 03 01 | x | 00 05 00 04 |
685 # | 0d 09 0e 0b |   | 01 01 02 03 |   | 04 00 05 00 |
686 # | 0b 0d 09 0e |   | 03 01 01 02 |   | 00 04 00 05 |

688 $code.=<<___;
689         # multiplication by 0x05-0x00-0x04-0x00
690         pshufd  \$0x4E, @x[0], @t[0]
691         pshufd  \$0x4E, @x[6], @t[6]
692         pxor    @x[0], @t[0]
693         pshufd  \$0x4E, @x[7], @t[7]
694         pxor    @x[6], @t[6]
695         pshufd  \$0x4E, @x[1], @t[1]
696         pxor    @x[7], @t[7]
697         pshufd  \$0x4E, @x[2], @t[2]
698         pxor    @x[1], @t[1]
699         pshufd  \$0x4E, @x[3], @t[3]
700         pxor    @x[2], @t[2]
701          pxor   @t[6], @x[0]
702          pxor   @t[6], @x[1]
703         pshufd  \$0x4E, @x[4], @t[4]
704         pxor    @x[3], @t[3]
705          pxor   @t[0], @x[2]
706          pxor   @t[1], @x[3]
707         pshufd  \$0x4E, @x[5], @t[5]
708         pxor    @x[4], @t[4]
709          pxor   @t[7], @x[1]
710          pxor   @t[2], @x[4]
711         pxor    @x[5], @t[5]

713          pxor   @t[7], @x[2]
714          pxor   @t[6], @x[3]
715          pxor   @t[6], @x[4]
716          pxor   @t[3], @x[5]
717          pxor   @t[4], @x[6]
718          pxor   @t[7], @x[4]
719          pxor   @t[7], @x[5]
720          pxor   @t[5], @x[7]
721 ___
```



```
722         &MixColumns     (@x,@t,1);      # flipped 2<->3 and 4<->6
723 }

725 sub aesenc {                            # not used
726 my @b=@_[0..7];
727 my @t=@_[8..15];
728 $code.=<<___;
729         movdqa  0x30($const),@t[0]      # .LSR
730 ___
731         &ShiftRows      (@b,@t[0]);
732         &Sbox           (@b,@t);
733         &MixColumns     (@b[0,1,4,6,3,7,2,5],@t);
734 }

736 sub aesenclast {                        # not used
737 my @b=@_[0..7];
738 my @t=@_[8..15];
739 $code.=<<___;
740         movdqa  0x40($const),@t[0]      # .LSRM0
741 ___
742         &ShiftRows      (@b,@t[0]);
743         &Sbox           (@b,@t);
744 $code.=<<___
745         pxor    0x00($key),@b[0]
746         pxor    0x10($key),@b[1]
747         pxor    0x20($key),@b[4]
748         pxor    0x30($key),@b[6]
749         pxor    0x40($key),@b[3]
750         pxor    0x50($key),@b[7]
751         pxor    0x60($key),@b[2]
752         pxor    0x70($key),@b[5]
753 ___
754 }

756 sub swapmove {
757 my ($a,$b,$n,$mask,$t)=@_;
758 $code.=<<___;
759         movdqa  $b,$t
760         psrlq   \$$n,$b
761         pxor    $a,$b
762         pand    $mask,$b
763         pxor    $b,$a
764         psllq   \$$n,$b
765         pxor    $t,$b
766 ___
767 }
768 sub swapmove2x {
769 my ($a0,$b0,$a1,$b1,$n,$mask,$t0,$t1)=@_;
770 $code.=<<___;
771         movdqa  $b0,$t0
772         psrlq   \$$n,$b0
773          movdqa $b1,$t1
774          psrlq  \$$n,$b1
775         pxor    $a0,$b0
776          pxor   $a1,$b1
777         pand    $mask,$b0
778          pand   $mask,$b1
779         pxor    $b0,$a0
780         psllq   \$$n,$b0
781          pxor   $b1,$a1
782          psllq  \$$n,$b1
783         pxor    $t0,$b0
784          pxor   $t1,$b1
785 ___
786 }
```

```
sub bitslice {
my @x=reverse(@_[0..7]);
my ($t0,$t1,$t2,$t3)=@_[8..11];
$code.=<<___;
        movdqa  0x00($const),$t0        # .LBS0
        movdqa  0x10($const),$t1        # .LBS1
___
        &swapmove2x(@x[0,1,2,3],1,$t0,$t2,$t3);
        &swapmove2x(@x[4,5,6,7],1,$t0,$t2,$t3);
$code.=<<___;
        movdqa  0x20($const),$t0        # .LBS2
___
        &swapmove2x(@x[0,2,1,3],2,$t1,$t2,$t3);
        &swapmove2x(@x[4,6,5,7],2,$t1,$t2,$t3);

        &swapmove2x(@x[0,4,1,5],4,$t0,$t2,$t3);
        &swapmove2x(@x[2,6,3,7],4,$t0,$t2,$t3);
}

$code.=<<___;
.text

.extern asm_AES_encrypt
.extern asm_AES_decrypt

.type   _bsaes_encrypt8,\@abi-omnipotent
.align  64
_bsaes_encrypt8:
        lea     .LBS0(%rip), $const     # constants table

        movdqa  ($key), @XMM[9]         # round 0 key
        lea     0x10($key), $key
        movdqa  0x50($const), @XMM[8]   # .LM0SR
        pxor    @XMM[9], @XMM[0]        # xor with round0 key
        pxor    @XMM[9], @XMM[1]
         pshufb @XMM[8], @XMM[0]
        pxor    @XMM[9], @XMM[2]
         pshufb @XMM[8], @XMM[1]
        pxor    @XMM[9], @XMM[3]
         pshufb @XMM[8], @XMM[2]
        pxor    @XMM[9], @XMM[4]
         pshufb @XMM[8], @XMM[3]
        pxor    @XMM[9], @XMM[5]
         pshufb @XMM[8], @XMM[4]
        pxor    @XMM[9], @XMM[6]
         pshufb @XMM[8], @XMM[5]
        pxor    @XMM[9], @XMM[7]
         pshufb @XMM[8], @XMM[6]
         pshufb @XMM[8], @XMM[7]
_bsaes_encrypt8_bitslice:
___
        &bitslice       (@XMM[0..7, 8..11]);
$code.=<<___;
        dec     $rounds
        jmp     .Lenc_sbox
.align  16
.Lenc_loop:
___
        &ShiftRows      (@XMM[0..7, 8]);
$code.=".Lenc_sbox:\n";
        &Sbox           (@XMM[0..7, 8..15]);
$code.=<<___;
        dec     $rounds
        jl      .Lenc_done
___
        &MixColumns     (@XMM[0,1,4,6,3,7,2,5, 8..15]);
```



```
$code.=<<___;
        movdqa  0x30($const), @XMM[8]   # .LSR
        jnz     .Lenc_loop
        movdqa  0x40($const), @XMM[8]   # .LSRM0
        jmp     .Lenc_loop
.align  16
.Lenc_done:
___
        # output in lsb > [t0, t1, t4, t6, t3, t7, t2, t5] < msb
        &bitslice       (@XMM[0,1,4,6,3,7,2,5, 8..11]);
$code.=<<___;
        movdqa  ($key), @XMM[8]         # last round key
        pxor    @XMM[8], @XMM[4]
        pxor    @XMM[8], @XMM[6]
        pxor    @XMM[8], @XMM[3]
        pxor    @XMM[8], @XMM[7]
        pxor    @XMM[8], @XMM[2]
        pxor    @XMM[8], @XMM[5]
        pxor    @XMM[8], @XMM[0]
        pxor    @XMM[8], @XMM[1]
        ret
.size   _bsaes_encrypt8,.-_bsaes_encrypt8

.type   _bsaes_decrypt8,\@abi-omnipotent
.align  64
_bsaes_decrypt8:
        lea     .LBS0(%rip), $const     # constants table

        movdqa  ($key), @XMM[9]         # round 0 key
        lea     0x10($key), $key
        movdqa  -0x30($const), @XMM[8]  # .LM0ISR
        pxor    @XMM[9], @XMM[0]        # xor with round0 key
        pxor    @XMM[9], @XMM[1]
         pshufb @XMM[8], @XMM[0]
        pxor    @XMM[9], @XMM[2]
         pshufb @XMM[8], @XMM[1]
        pxor    @XMM[9], @XMM[3]
         pshufb @XMM[8], @XMM[2]
        pxor    @XMM[9], @XMM[4]
         pshufb @XMM[8], @XMM[3]
        pxor    @XMM[9], @XMM[5]
         pshufb @XMM[8], @XMM[4]
        pxor    @XMM[9], @XMM[6]
         pshufb @XMM[8], @XMM[5]
        pxor    @XMM[9], @XMM[7]
         pshufb @XMM[8], @XMM[6]
         pshufb @XMM[8], @XMM[7]
___
        &bitslice       (@XMM[0..7, 8..11]);
$code.=<<___;
        dec     $rounds
        jmp     .Ldec_sbox
.align  16
.Ldec_loop:
___
        &ShiftRows      (@XMM[0..7, 8]);
$code.=".Ldec_sbox:\n";
        &InvSbox        (@XMM[0..7, 8..15]);
$code.=<<___;
        dec     $rounds
        jl      .Ldec_done
___
        &InvMixColumns  (@XMM[0,1,6,4,2,7,3,5, 8..15]);
$code.=<<___;
        movdqa  -0x10($const), @XMM[8]  # .LISR
        jnz     .Ldec_loop
```

```
920         movdqa  -0x20($const), @XMM[8]  # .LISRM0
921         jmp     .Ldec_loop
922 .align  16
923 .Ldec_done:
924 ___
925         &bitslice       (@XMM[0,1,6,4,2,7,3,5, 8..11]);
926 $code.=<<___;
927         movdqa  ($key), @XMM[8]         # last round key
928         pxor    @XMM[8], @XMM[6]
929         pxor    @XMM[8], @XMM[4]
930         pxor    @XMM[8], @XMM[2]
931         pxor    @XMM[8], @XMM[7]
932         pxor    @XMM[8], @XMM[3]
933         pxor    @XMM[8], @XMM[5]
934         pxor    @XMM[8], @XMM[0]
935         pxor    @XMM[8], @XMM[1]
936         ret
937 .size   _bsaes_decrypt8,.-_bsaes_decrypt8
938 ___
939 }
940 {
941 my ($out,$inp,$rounds,$const)=("%rax","%rcx","%r10d","%r11");

943 sub bitslice_key {
944 my @x=reverse(@_[0..7]);
945 my ($bs0,$bs1,$bs2,$t2,$t3)=@_[8..12];

947         &swapmove       (@x[0,1],1,$bs0,$t2,$t3);
948 $code.=<<___;
949         #&swapmove(@x[2,3],1,$t0,$t2,$t3);
950         movdqa  @x[0], @x[2]
951         movdqa  @x[1], @x[3]
952 ___
953         #&swapmove2x(@x[4,5,6,7],1,$t0,$t2,$t3);

955         &swapmove2x     (@x[0,2,1,3],2,$bs1,$t2,$t3);
956 $code.=<<___;
957         #&swapmove2x(@x[4,6,5,7],2,$t1,$t2,$t3);
958         movdqa  @x[0], @x[4]
959         movdqa  @x[2], @x[6]
960         movdqa  @x[1], @x[5]
961         movdqa  @x[3], @x[7]
962 ___
963         &swapmove2x     (@x[0,4,1,5],4,$bs2,$t2,$t3);
964         &swapmove2x     (@x[2,6,3,7],4,$bs2,$t2,$t3);
965 }

967 $code.=<<___;
968 .type   _bsaes_key_convert,\@abi-omnipotent
969 .align  16
970 _bsaes_key_convert:
971         lea     .Lmasks(%rip), $const
972         movdqu  ($inp), %xmm7           # load round 0 key
973         lea     0x10($inp), $inp
974         movdqa  0x00($const), %xmm0     # 0x01...
975         movdqa  0x10($const), %xmm1     # 0x02...
976         movdqa  0x20($const), %xmm2     # 0x04...
977         movdqa  0x30($const), %xmm3     # 0x08...
978         movdqa  0x40($const), %xmm4     # .LM0
979         pcmpeqd %xmm5, %xmm5            # .LNOT

981         movdqu  ($inp), %xmm6           # load round 1 key
982         movdqa  %xmm7, ($out)           # save round 0 key
983         lea     0x10($out), $out
984         dec     $rounds
985         jmp     .Lkey_loop
```



```
986 .align  16
987 .Lkey_loop:
988         pshufb  %xmm4, %xmm6            # .LM0

990         movdqa  %xmm0,  %xmm8
991         movdqa  %xmm1,  %xmm9

993         pand    %xmm6,  %xmm8
994         pand    %xmm6,  %xmm9
995         movdqa  %xmm2,  %xmm10
996         pcmpeqb %xmm0,  %xmm8
997         psllq   \$4,    %xmm0           # 0x10...
998         movdqa  %xmm3,  %xmm11
999         pcmpeqb %xmm1,  %xmm9
1000        psllq   \$4,    %xmm1           # 0x20...

1002        pand    %xmm6,  %xmm10
1003        pand    %xmm6,  %xmm11
1004        movdqa  %xmm0,  %xmm12
1005        pcmpeqb %xmm2,  %xmm10
1006        psllq   \$4,    %xmm2           # 0x40...
1007        movdqa  %xmm1,  %xmm13
1008        pcmpeqb %xmm3,  %xmm11
1009        psllq   \$4,    %xmm3           # 0x80...

1011        movdqa  %xmm2,  %xmm14
1012        movdqa  %xmm3,  %xmm15
1013         pxor   %xmm5,  %xmm8           # "pnot"
1014         pxor   %xmm5,  %xmm9

1016        pand    %xmm6,  %xmm12
1017        pand    %xmm6,  %xmm13
1018         movdqa %xmm8, 0x00($out)       # write bit-sliced round key
1019        pcmpeqb %xmm0,  %xmm12
1020        psrlq   \$4,    %xmm0           # 0x01...
1021         movdqa %xmm9, 0x10($out)
1022        pcmpeqb %xmm1,  %xmm13
1023        psrlq   \$4,    %xmm1           # 0x02...
1024         lea    0x10($inp), $inp

1026        pand    %xmm6,  %xmm14
1027        pand    %xmm6,  %xmm15
1028         movdqa %xmm10, 0x20($out)
1029        pcmpeqb %xmm2,  %xmm14
1030        psrlq   \$4,    %xmm2           # 0x04...
1031         movdqa %xmm11, 0x30($out)
1032        pcmpeqb %xmm3,  %xmm15
1033        psrlq   \$4,    %xmm3           # 0x08...
1034         movdqu ($inp), %xmm6           # load next round key

1036        pxor    %xmm5, %xmm13           # "pnot"
1037        pxor    %xmm5, %xmm14
1038        movdqa  %xmm12, 0x40($out)
1039        movdqa  %xmm13, 0x50($out)
1040        movdqa  %xmm14, 0x60($out)
1041        movdqa  %xmm15, 0x70($out)
1042        lea     0x80($out),$out
1043        dec     $rounds
1044        jnz     .Lkey_loop

1046        movdqa  0x50($const), %xmm7     # .L63
1047        #movdqa %xmm6, ($out)           # don't save last round key
1048        ret
1049 .size   _bsaes_key_convert,.-_bsaes_key_convert
1050 ___
1051 }
```

```
1053 if (0 && !$win64) {     # following four functions are unsupported interface
1054                         # used for benchmarking...
1055 $code.=<<___;
1056 .globl  bsaes_enc_key_convert
1057 .type   bsaes_enc_key_convert,\@function,2
1058 .align  16
1059 bsaes_enc_key_convert:
1060         mov     240($inp),%r10d         # pass rounds
1061         mov     $inp,%rcx               # pass key
1062         mov     $out,%rax               # pass key schedule
1063         call    _bsaes_key_convert
1064         pxor    %xmm6,%xmm7             # fix up last round key
1065         movdqa  %xmm7,(%rax)            # save last round key
1066         ret
1067 .size   bsaes_enc_key_convert,.-bsaes_enc_key_convert

1069 .globl  bsaes_encrypt_128
1070 .type   bsaes_encrypt_128,\@function,4
1071 .align  16
1072 bsaes_encrypt_128:
1073 .Lenc128_loop:
1074         movdqu  0x00($inp), @XMM[0]     # load input
1075         movdqu  0x10($inp), @XMM[1]
1076         movdqu  0x20($inp), @XMM[2]
1077         movdqu  0x30($inp), @XMM[3]
1078         movdqu  0x40($inp), @XMM[4]
1079         movdqu  0x50($inp), @XMM[5]
1080         movdqu  0x60($inp), @XMM[6]
1081         movdqu  0x70($inp), @XMM[7]
1082         mov     $key, %rax              # pass the $key
1083         lea     0x80($inp), $inp
1084         mov     \$10,%r10d

1086         call    _bsaes_encrypt8

1088         movdqu  @XMM[0], 0x00($out)     # write output
1089         movdqu  @XMM[1], 0x10($out)
1090         movdqu  @XMM[4], 0x20($out)
1091         movdqu  @XMM[6], 0x30($out)
1092         movdqu  @XMM[3], 0x40($out)
1093         movdqu  @XMM[7], 0x50($out)
1094         movdqu  @XMM[2], 0x60($out)
1095         movdqu  @XMM[5], 0x70($out)
1096         lea     0x80($out), $out
1097         sub     \$0x80,$len
1098         ja      .Lenc128_loop
1099         ret
1100 .size   bsaes_encrypt_128,.-bsaes_encrypt_128

1102 .globl  bsaes_dec_key_convert
1103 .type   bsaes_dec_key_convert,\@function,2
1104 .align  16
1105 bsaes_dec_key_convert:
1106         mov     240($inp),%r10d         # pass rounds
1107         mov     $inp,%rcx               # pass key
1108         mov     $out,%rax               # pass key schedule
1109         call    _bsaes_key_convert
1110         pxor    ($out),%xmm7            # fix up round 0 key
1111         movdqa  %xmm6,(%rax)            # save last round key
1112         movdqa  %xmm7,($out)
1113         ret
1114 .size   bsaes_dec_key_convert,.-bsaes_dec_key_convert

1116 .globl  bsaes_decrypt_128
1117 .type   bsaes_decrypt_128,\@function,4
```



```
1118 .align  16
1119 bsaes_decrypt_128:
1120 .Ldec128_loop:
1121         movdqu  0x00($inp), @XMM[0]     # load input
1122         movdqu  0x10($inp), @XMM[1]
1123         movdqu  0x20($inp), @XMM[2]
1124         movdqu  0x30($inp), @XMM[3]
1125         movdqu  0x40($inp), @XMM[4]
1126         movdqu  0x50($inp), @XMM[5]
1127         movdqu  0x60($inp), @XMM[6]
1128         movdqu  0x70($inp), @XMM[7]
1129         mov     $key, %rax              # pass the $key
1130         lea     0x80($inp), $inp
1131         mov     \$10,%r10d

1133         call    _bsaes_decrypt8

1135         movdqu  @XMM[0], 0x00($out)     # write output
1136         movdqu  @XMM[1], 0x10($out)
1137         movdqu  @XMM[6], 0x20($out)
1138         movdqu  @XMM[4], 0x30($out)
1139         movdqu  @XMM[2], 0x40($out)
1140         movdqu  @XMM[7], 0x50($out)
1141         movdqu  @XMM[3], 0x60($out)
1142         movdqu  @XMM[5], 0x70($out)
1143         lea     0x80($out), $out
1144         sub     \$0x80,$len
1145         ja      .Ldec128_loop
1146         ret
1147 .size   bsaes_decrypt_128,.-bsaes_decrypt_128
1148 ___
1149 }
1150 {
1151 ######################################################################
1152 #
1153 # OpenSSL interface
1154 #
1155 my ($arg1,$arg2,$arg3,$arg4,$arg5,$arg6)=$win64 ? ("%rcx","%rdx","%r8","%r9","%r
1156                                                 : ("%rdi","%rsi","%rdx","%rcx","
1157 my ($inp,$out,$len,$key)=("%r12","%r13","%r14","%r15");

1159 if ($ecb) {
1160 $code.=<<___;
1161 .globl  bsaes_ecb_encrypt_blocks
1162 .type   bsaes_ecb_encrypt_blocks,\@abi-omnipotent
1163 .align  16
1164 bsaes_ecb_encrypt_blocks:
1165         mov     %rsp, %rax
1166 .Lecb_enc_prologue:
1167         push    %rbp
1168         push    %rbx
1169         push    %r12
1170         push    %r13
1171         push    %r14
1172         push    %r15
1173         lea     -0x48(%rsp),%rsp
1174 ___
1175 $code.=<<___ if ($win64);
1176         lea     -0xa0(%rsp), %rsp
1177         movaps  %xmm6, 0x40(%rsp)
1178         movaps  %xmm7, 0x50(%rsp)
1179         movaps  %xmm8, 0x60(%rsp)
1180         movaps  %xmm9, 0x70(%rsp)
1181         movaps  %xmm10, 0x80(%rsp)
1182         movaps  %xmm11, 0x90(%rsp)
1183         movaps  %xmm12, 0xa0(%rsp)
```

```
1184         movaps  %xmm13, 0xb0(%rsp)
1185         movaps  %xmm14, 0xc0(%rsp)
1186         movaps  %xmm15, 0xd0(%rsp)
1187 .Lecb_enc_body:
1188 ___
1189 $code.=<<___;
1190         mov     %rsp,%rbp               # backup %rsp
1191         mov     240($arg4),%eax         # rounds
1192         mov     $arg1,$inp              # backup arguments
1193         mov     $arg2,$out
1194         mov     $arg3,$len
1195         mov     $arg4,$key
1196         cmp     \$8,$arg3
1197         jb      .Lecb_enc_short

1199         mov     %eax,%ebx               # backup rounds
1200         shl     \$7,%rax                # 128 bytes per inner round key
1201         sub     \$`128-32`,%rax         # size of bit-sliced key schedule
1202         sub     %rax,%rsp
1203         mov     %rsp,%rax               # pass key schedule
1204         mov     $key,%rcx               # pass key
1205         mov     %ebx,%r10d              # pass rounds
1206         call    _bsaes_key_convert
1207         pxor    %xmm6,%xmm7             # fix up last round key
1208         movdqa  %xmm7,(%rax)            # save last round key

1210         sub     \$8,$len
1211 .Lecb_enc_loop:
1212         movdqu  0x00($inp), @XMM[0]     # load input
1213         movdqu  0x10($inp), @XMM[1]
1214         movdqu  0x20($inp), @XMM[2]
1215         movdqu  0x30($inp), @XMM[3]
1216         movdqu  0x40($inp), @XMM[4]
1217         movdqu  0x50($inp), @XMM[5]
1218         mov     %rsp, %rax              # pass key schedule
1219         movdqu  0x60($inp), @XMM[6]
1220         mov     %ebx,%r10d              # pass rounds
1221         movdqu  0x70($inp), @XMM[7]
1222         lea     0x80($inp), $inp

1224         call    _bsaes_encrypt8

1226         movdqu  @XMM[0], 0x00($out)     # write output
1227         movdqu  @XMM[1], 0x10($out)
1228         movdqu  @XMM[4], 0x20($out)
1229         movdqu  @XMM[6], 0x30($out)
1230         movdqu  @XMM[3], 0x40($out)
1231         movdqu  @XMM[7], 0x50($out)
1232         movdqu  @XMM[2], 0x60($out)
1233         movdqu  @XMM[5], 0x70($out)
1234         lea     0x80($out), $out
1235         sub     \$8,$len
1236         jnc     .Lecb_enc_loop

1238         add     \$8,$len
1239         jz      .Lecb_enc_done

1241         movdqu  0x00($inp), @XMM[0]     # load input
1242         mov     %rsp, %rax              # pass key schedule
1243         mov     %ebx,%r10d              # pass rounds
1244         cmp     \$2,$len
1245         jb      .Lecb_enc_one
1246         movdqu  0x10($inp), @XMM[1]
1247         je      .Lecb_enc_two
1248         movdqu  0x20($inp), @XMM[2]
1249         cmp     \$4,$len
```



```
1250         jb      .Lecb_enc_three
1251         movdqu  0x30($inp), @XMM[3]
1252         je      .Lecb_enc_four
1253         movdqu  0x40($inp), @XMM[4]
1254         cmp     \$6,$len
1255         jb      .Lecb_enc_five
1256         movdqu  0x50($inp), @XMM[5]
1257         je      .Lecb_enc_six
1258         movdqu  0x60($inp), @XMM[6]
1259         call    _bsaes_encrypt8
1260         movdqu  @XMM[0], 0x00($out)     # write output
1261         movdqu  @XMM[1], 0x10($out)
1262         movdqu  @XMM[4], 0x20($out)
1263         movdqu  @XMM[6], 0x30($out)
1264         movdqu  @XMM[3], 0x40($out)
1265         movdqu  @XMM[7], 0x50($out)
1266         movdqu  @XMM[2], 0x60($out)
1267         jmp     .Lecb_enc_done
1268 .align  16
1269 .Lecb_enc_six:
1270         call    _bsaes_encrypt8
1271         movdqu  @XMM[0], 0x00($out)     # write output
1272         movdqu  @XMM[1], 0x10($out)
1273         movdqu  @XMM[4], 0x20($out)
1274         movdqu  @XMM[6], 0x30($out)
1275         movdqu  @XMM[3], 0x40($out)
1276         movdqu  @XMM[7], 0x50($out)
1277         jmp     .Lecb_enc_done
1278 .align  16
1279 .Lecb_enc_five:
1280         call    _bsaes_encrypt8
1281         movdqu  @XMM[0], 0x00($out)     # write output
1282         movdqu  @XMM[1], 0x10($out)
1283         movdqu  @XMM[4], 0x20($out)
1284         movdqu  @XMM[6], 0x30($out)
1285         movdqu  @XMM[3], 0x40($out)
1286         jmp     .Lecb_enc_done
1287 .align  16
1288 .Lecb_enc_four:
1289         call    _bsaes_encrypt8
1290         movdqu  @XMM[0], 0x00($out)     # write output
1291         movdqu  @XMM[1], 0x10($out)
1292         movdqu  @XMM[4], 0x20($out)
1293         movdqu  @XMM[6], 0x30($out)
1294         jmp     .Lecb_enc_done
1295 .align  16
1296 .Lecb_enc_three:
1297         call    _bsaes_encrypt8
1298         movdqu  @XMM[0], 0x00($out)     # write output
1299         movdqu  @XMM[1], 0x10($out)
1300         movdqu  @XMM[4], 0x20($out)
1301         jmp     .Lecb_enc_done
1302 .align  16
1303 .Lecb_enc_two:
1304         call    _bsaes_encrypt8
1305         movdqu  @XMM[0], 0x00($out)     # write output
1306         movdqu  @XMM[1], 0x10($out)
1307         jmp     .Lecb_enc_done
1308 .align  16
1309 .Lecb_enc_one:
1310         call    _bsaes_encrypt8
1311         movdqu  @XMM[0], 0x00($out)     # write output
1312         jmp     .Lecb_enc_done
1313 .align  16
1314 .Lecb_enc_short:
1315         lea     ($inp), $arg1
```

```
1316         lea     ($out), $arg2
1317         lea     ($key), $arg3
1318         call    asm_AES_encrypt
1319         lea     16($inp), $inp
1320         lea     16($out), $out
1321         dec     $len
1322         jnz     .Lecb_enc_short

1324 .Lecb_enc_done:
1325         lea     (%rsp),%rax
1326         pxor    %xmm0, %xmm0
1327 .Lecb_enc_bzero:                        # wipe key schedule [if any]
1328         movdqa  %xmm0, 0x00(%rax)
1329         movdqa  %xmm0, 0x10(%rax)
1330         lea     0x20(%rax), %rax
1331         cmp     %rax, %rbp
1332         jb      .Lecb_enc_bzero

1334         lea     (%rbp),%rsp             # restore %rsp
1335 ___
1336 $code.=<<___ if ($win64);
1337         movaps  0x40(%rbp), %xmm6
1338         movaps  0x50(%rbp), %xmm7
1339         movaps  0x60(%rbp), %xmm8
1340         movaps  0x70(%rbp), %xmm9
1341         movaps  0x80(%rbp), %xmm10
1342         movaps  0x90(%rbp), %xmm11
1343         movaps  0xa0(%rbp), %xmm12
1344         movaps  0xb0(%rbp), %xmm13
1345         movaps  0xc0(%rbp), %xmm14
1346         movaps  0xd0(%rbp), %xmm15
1347         lea     0xa0(%rbp), %rsp
1348 ___
1349 $code.=<<___;
1350         mov     0x48(%rsp), %r15
1351         mov     0x50(%rsp), %r14
1352         mov     0x58(%rsp), %r13
1353         mov     0x60(%rsp), %r12
1354         mov     0x68(%rsp), %rbx
1355         mov     0x70(%rsp), %rax
1356         lea     0x78(%rsp), %rsp
1357         mov     %rax, %rbp
1358 .Lecb_enc_epilogue:
1359         ret
1360 .size   bsaes_ecb_encrypt_blocks,.-bsaes_ecb_encrypt_blocks

1362 .globl  bsaes_ecb_decrypt_blocks
1363 .type   bsaes_ecb_decrypt_blocks,\@abi-omnipotent
1364 .align  16
1365 bsaes_ecb_decrypt_blocks:
1366         mov     %rsp, %rax
1367 .Lecb_dec_prologue:
1368         push    %rbp
1369         push    %rbx
1370         push    %r12
1371         push    %r13
1372         push    %r14
1373         push    %r15
1374         lea     -0x48(%rsp),%rsp
1375 ___
1376 $code.=<<___ if ($win64);
1377         lea     -0xa0(%rsp), %rsp
1378         movaps  %xmm6, 0x40(%rsp)
1379         movaps  %xmm7, 0x50(%rsp)
1380         movaps  %xmm8, 0x60(%rsp)
1381         movaps  %xmm9, 0x70(%rsp)
```



```
1382         movaps  %xmm10, 0x80(%rsp)
1383         movaps  %xmm11, 0x90(%rsp)
1384         movaps  %xmm12, 0xa0(%rsp)
1385         movaps  %xmm13, 0xb0(%rsp)
1386         movaps  %xmm14, 0xc0(%rsp)
1387         movaps  %xmm15, 0xd0(%rsp)
1388 .Lecb_dec_body:
1389 ___
1390 $code.=<<___;
1391         mov     %rsp,%rbp               # backup %rsp
1392         mov     240($arg4),%eax         # rounds
1393         mov     $arg1,$inp              # backup arguments
1394         mov     $arg2,$out
1395         mov     $arg3,$len
1396         mov     $arg4,$key
1397         cmp     \$8,$arg3
1398         jb      .Lecb_dec_short

1400         mov     %eax,%ebx               # backup rounds
1401         shl     \$7,%rax                # 128 bytes per inner round key
1402         sub     \$`128-32`,%rax         # size of bit-sliced key schedule
1403         sub     %rax,%rsp
1404         mov     %rsp,%rax               # pass key schedule
1405         mov     $key,%rcx               # pass key
1406         mov     %ebx,%r10d              # pass rounds
1407         call    _bsaes_key_convert
1408         pxor    (%rsp),%xmm7            # fix up 0 round key
1409         movdqa  %xmm6,(%rax)            # save last round key
1410         movdqa  %xmm7,(%rsp)

1412         sub     \$8,$len
1413 .Lecb_dec_loop:
1414         movdqu  0x00($inp), @XMM[0]     # load input
1415         movdqu  0x10($inp), @XMM[1]
1416         movdqu  0x20($inp), @XMM[2]
1417         movdqu  0x30($inp), @XMM[3]
1418         movdqu  0x40($inp), @XMM[4]
1419         movdqu  0x50($inp), @XMM[5]
1420         mov     %rsp, %rax              # pass key schedule
1421         movdqu  0x60($inp), @XMM[6]
1422         mov     %ebx,%r10d              # pass rounds
1423         movdqu  0x70($inp), @XMM[7]
1424         lea     0x80($inp), $inp

1426         call    _bsaes_decrypt8

1428         movdqu  @XMM[0], 0x00($out)     # write output
1429         movdqu  @XMM[1], 0x10($out)
1430         movdqu  @XMM[6], 0x20($out)
1431         movdqu  @XMM[4], 0x30($out)
1432         movdqu  @XMM[2], 0x40($out)
1433         movdqu  @XMM[7], 0x50($out)
1434         movdqu  @XMM[3], 0x60($out)
1435         movdqu  @XMM[5], 0x70($out)
1436         lea     0x80($out), $out
1437         sub     \$8,$len
1438         jnc     .Lecb_dec_loop

1440         add     \$8,$len
1441         jz      .Lecb_dec_done

1443         movdqu  0x00($inp), @XMM[0]     # load input
1444         mov     %rsp, %rax              # pass key schedule
1445         mov     %ebx,%r10d              # pass rounds
1446         cmp     \$2,$len
1447         jb      .Lecb_dec_one
```

```
1448         movdqu  0x10($inp), @XMM[1]
1449         je      .Lecb_dec_two
1450         movdqu  0x20($inp), @XMM[2]
1451         cmp     \$4,$len
1452         jb      .Lecb_dec_three
1453         movdqu  0x30($inp), @XMM[3]
1454         je      .Lecb_dec_four
1455         movdqu  0x40($inp), @XMM[4]
1456         cmp     \$6,$len
1457         jb      .Lecb_dec_five
1458         movdqu  0x50($inp), @XMM[5]
1459         je      .Lecb_dec_six
1460         movdqu  0x60($inp), @XMM[6]
1461         call    _bsaes_decrypt8
1462         movdqu  @XMM[0], 0x00($out)     # write output
1463         movdqu  @XMM[1], 0x10($out)
1464         movdqu  @XMM[6], 0x20($out)
1465         movdqu  @XMM[4], 0x30($out)
1466         movdqu  @XMM[2], 0x40($out)
1467         movdqu  @XMM[7], 0x50($out)
1468         movdqu  @XMM[3], 0x60($out)
1469         jmp     .Lecb_dec_done
1470 .align  16
1471 .Lecb_dec_six:
1472         call    _bsaes_decrypt8
1473         movdqu  @XMM[0], 0x00($out)     # write output
1474         movdqu  @XMM[1], 0x10($out)
1475         movdqu  @XMM[6], 0x20($out)
1476         movdqu  @XMM[4], 0x30($out)
1477         movdqu  @XMM[2], 0x40($out)
1478         movdqu  @XMM[7], 0x50($out)
1479         jmp     .Lecb_dec_done
1480 .align  16
1481 .Lecb_dec_five:
1482         call    _bsaes_decrypt8
1483         movdqu  @XMM[0], 0x00($out)     # write output
1484         movdqu  @XMM[1], 0x10($out)
1485         movdqu  @XMM[6], 0x20($out)
1486         movdqu  @XMM[4], 0x30($out)
1487         movdqu  @XMM[2], 0x40($out)
1488         jmp     .Lecb_dec_done
1489 .align  16
1490 .Lecb_dec_four:
1491         call    _bsaes_decrypt8
1492         movdqu  @XMM[0], 0x00($out)     # write output
1493         movdqu  @XMM[1], 0x10($out)
1494         movdqu  @XMM[6], 0x20($out)
1495         movdqu  @XMM[4], 0x30($out)
1496         jmp     .Lecb_dec_done
1497 .align  16
1498 .Lecb_dec_three:
1499         call    _bsaes_decrypt8
1500         movdqu  @XMM[0], 0x00($out)     # write output
1501         movdqu  @XMM[1], 0x10($out)
1502         movdqu  @XMM[6], 0x20($out)
1503         jmp     .Lecb_dec_done
1504 .align  16
1505 .Lecb_dec_two:
1506         call    _bsaes_decrypt8
1507         movdqu  @XMM[0], 0x00($out)     # write output
1508         movdqu  @XMM[1], 0x10($out)
1509         jmp     .Lecb_dec_done
1510 .align  16
1511 .Lecb_dec_one:
1512         call    _bsaes_decrypt8
1513         movdqu  @XMM[0], 0x00($out)     # write output
```



```
1514         jmp     .Lecb_dec_done
1515 .align  16
1516 .Lecb_dec_short:
1517         lea     ($inp), $arg1
1518         lea     ($out), $arg2
1519         lea     ($key), $arg3
1520         call    asm_AES_decrypt
1521         lea     16($inp), $inp
1522         lea     16($out), $out
1523         dec     $len
1524         jnz     .Lecb_dec_short

1526 .Lecb_dec_done:
1527         lea     (%rsp),%rax
1528         pxor    %xmm0, %xmm0
1529 .Lecb_dec_bzero:                        # wipe key schedule [if any]
1530         movdqa  %xmm0, 0x00(%rax)
1531         movdqa  %xmm0, 0x10(%rax)
1532         lea     0x20(%rax), %rax
1533         cmp     %rax, %rbp
1534         jb      .Lecb_dec_bzero

1536         lea     (%rbp),%rsp             # restore %rsp
1537 ___
1538 $code.=<<___ if ($win64);
1539         movaps  0x40(%rbp), %xmm6
1540         movaps  0x50(%rbp), %xmm7
1541         movaps  0x60(%rbp), %xmm8
1542         movaps  0x70(%rbp), %xmm9
1543         movaps  0x80(%rbp), %xmm10
1544         movaps  0x90(%rbp), %xmm11
1545         movaps  0xa0(%rbp), %xmm12
1546         movaps  0xb0(%rbp), %xmm13
1547         movaps  0xc0(%rbp), %xmm14
1548         movaps  0xd0(%rbp), %xmm15
1549         lea     0xa0(%rbp), %rsp
1550 ___
1551 $code.=<<___;
1552         mov     0x48(%rsp), %r15
1553         mov     0x50(%rsp), %r14
1554         mov     0x58(%rsp), %r13
1555         mov     0x60(%rsp), %r12
1556         mov     0x68(%rsp), %rbx
1557         mov     0x70(%rsp), %rax
1558         lea     0x78(%rsp), %rsp
1559         mov     %rax, %rbp
1560 .Lecb_dec_epilogue:
1561         ret
1562 .size   bsaes_ecb_decrypt_blocks,.-bsaes_ecb_decrypt_blocks
1563 ___
1564 }
1565 $code.=<<___;
1566 .extern asm_AES_cbc_encrypt
1567 .globl  bsaes_cbc_encrypt
1568 .type   bsaes_cbc_encrypt,\@abi-omnipotent
1569 .align  16
1570 bsaes_cbc_encrypt:
1571 ___
1572 $code.=<<___ if ($win64);
1573         mov     48(%rsp),$arg6          # pull direction flag
1574 ___
1575 $code.=<<___;
1576         cmp     \$0,$arg6
1577         jne     asm_AES_cbc_encrypt
1578         cmp     \$128,$arg3
1579         jb      asm_AES_cbc_encrypt
```

```
1581         mov     %rsp, %rax
1582 .Lcbc_dec_prologue:
1583         push    %rbp
1584         push    %rbx
1585         push    %r12
1586         push    %r13
1587         push    %r14
1588         push    %r15
1589         lea     -0x48(%rsp), %rsp
1590 ___
1591 $code.=<<___ if ($win64);
1592         mov     0xa0(%rsp),$arg5        # pull ivp
1593         lea     -0xa0(%rsp), %rsp
1594         movaps  %xmm6, 0x40(%rsp)
1595         movaps  %xmm7, 0x50(%rsp)
1596         movaps  %xmm8, 0x60(%rsp)
1597         movaps  %xmm9, 0x70(%rsp)
1598         movaps  %xmm10, 0x80(%rsp)
1599         movaps  %xmm11, 0x90(%rsp)
1600         movaps  %xmm12, 0xa0(%rsp)
1601         movaps  %xmm13, 0xb0(%rsp)
1602         movaps  %xmm14, 0xc0(%rsp)
1603         movaps  %xmm15, 0xd0(%rsp)
1604 .Lcbc_dec_body:
1605 ___
1606 $code.=<<___;
1607         mov     %rsp, %rbp              # backup %rsp
1608         mov     240($arg4), %eax        # rounds
1609         mov     $arg1, $inp             # backup arguments
1610         mov     $arg2, $out
1611         mov     $arg3, $len
1612         mov     $arg4, $key
1613         mov     $arg5, %rbx
1614         shr     \$4, $len               # bytes to blocks

1616         mov     %eax, %edx              # rounds
1617         shl     \$7, %rax               # 128 bytes per inner round key
1618         sub     \$`128-32`, %rax        # size of bit-sliced key schedule
1619         sub     %rax, %rsp

1621         mov     %rsp, %rax              # pass key schedule
1622         mov     $key, %rcx              # pass key
1623         mov     %edx, %r10d             # pass rounds
1624         call    _bsaes_key_convert
1625         pxor    (%rsp),%xmm7            # fix up 0 round key
1626         movdqa  %xmm6,(%rax)            # save last round key
1627         movdqa  %xmm7,(%rsp)

1629         movdqu  (%rbx), @XMM[15]        # load IV
1630         sub     \$8,$len
1631 .Lcbc_dec_loop:
1632         movdqu  0x00($inp), @XMM[0]     # load input
1633         movdqu  0x10($inp), @XMM[1]
1634         movdqu  0x20($inp), @XMM[2]
1635         movdqu  0x30($inp), @XMM[3]
1636         movdqu  0x40($inp), @XMM[4]
1637         movdqu  0x50($inp), @XMM[5]
1638         mov     %rsp, %rax              # pass key schedule
1639         movdqu  0x60($inp), @XMM[6]
1640         mov     %edx,%r10d              # pass rounds
1641         movdqu  0x70($inp), @XMM[7]
1642         movdqa  @XMM[15], 0x20(%rbp)    # put aside IV

1644         call    _bsaes_decrypt8
```



```
1646         pxor    0x20(%rbp), @XMM[0]     # ^= IV
1647         movdqu  0x00($inp), @XMM[8]     # re-load input
1648         movdqu  0x10($inp), @XMM[9]
1649         pxor    @XMM[8], @XMM[1]
1650         movdqu  0x20($inp), @XMM[10]
1651         pxor    @XMM[9], @XMM[6]
1652         movdqu  0x30($inp), @XMM[11]
1653         pxor    @XMM[10], @XMM[4]
1654         movdqu  0x40($inp), @XMM[12]
1655         pxor    @XMM[11], @XMM[2]
1656         movdqu  0x50($inp), @XMM[13]
1657         pxor    @XMM[12], @XMM[7]
1658         movdqu  0x60($inp), @XMM[14]
1659         pxor    @XMM[13], @XMM[3]
1660         movdqu  0x70($inp), @XMM[15]    # IV
1661         pxor    @XMM[14], @XMM[5]
1662         movdqu  @XMM[0], 0x00($out)     # write output
1663         lea     0x80($inp), $inp
1664         movdqu  @XMM[1], 0x10($out)
1665         movdqu  @XMM[6], 0x20($out)
1666         movdqu  @XMM[4], 0x30($out)
1667         movdqu  @XMM[2], 0x40($out)
1668         movdqu  @XMM[7], 0x50($out)
1669         movdqu  @XMM[3], 0x60($out)
1670         movdqu  @XMM[5], 0x70($out)
1671         lea     0x80($out), $out
1672         sub     \$8,$len
1673         jnc     .Lcbc_dec_loop

1675         add     \$8,$len
1676         jz      .Lcbc_dec_done

1678         movdqu  0x00($inp), @XMM[0]     # load input
1679         mov     %rsp, %rax              # pass key schedule
1680         mov     %edx, %r10d             # pass rounds
1681         cmp     \$2,$len
1682         jb      .Lcbc_dec_one
1683         movdqu  0x10($inp), @XMM[1]
1684         je      .Lcbc_dec_two
1685         movdqu  0x20($inp), @XMM[2]
1686         cmp     \$4,$len
1687         jb      .Lcbc_dec_three
1688         movdqu  0x30($inp), @XMM[3]
1689         je      .Lcbc_dec_four
1690         movdqu  0x40($inp), @XMM[4]
1691         cmp     \$6,$len
1692         jb      .Lcbc_dec_five
1693         movdqu  0x50($inp), @XMM[5]
1694         je      .Lcbc_dec_six
1695         movdqu  0x60($inp), @XMM[6]
1696         movdqa  @XMM[15], 0x20(%rbp)    # put aside IV
1697         call    _bsaes_decrypt8
1698         pxor    0x20(%rbp), @XMM[0]     # ^= IV
1699         movdqu  0x00($inp), @XMM[8]     # re-load input
1700         movdqu  0x10($inp), @XMM[9]
1701         pxor    @XMM[8], @XMM[1]
1702         movdqu  0x20($inp), @XMM[10]
1703         pxor    @XMM[9], @XMM[6]
1704         movdqu  0x30($inp), @XMM[11]
1705         pxor    @XMM[10], @XMM[4]
1706         movdqu  0x40($inp), @XMM[12]
1707         pxor    @XMM[11], @XMM[2]
1708         movdqu  0x50($inp), @XMM[13]
1709         pxor    @XMM[12], @XMM[7]
1710         movdqu  0x60($inp), @XMM[15]    # IV
1711         pxor    @XMM[13], @XMM[3]
```

```
1712         movdqu  @XMM[0], 0x00($out)     # write output
1713         movdqu  @XMM[1], 0x10($out)
1714         movdqu  @XMM[6], 0x20($out)
1715         movdqu  @XMM[4], 0x30($out)
1716         movdqu  @XMM[2], 0x40($out)
1717         movdqu  @XMM[7], 0x50($out)
1718         movdqu  @XMM[3], 0x60($out)
1719         jmp     .Lcbc_dec_done
1720 .align  16
1721 .Lcbc_dec_six:
1722         movdqa  @XMM[15], 0x20(%rbp)    # put aside IV
1723         call    _bsaes_decrypt8
1724         pxor    0x20(%rbp), @XMM[0]     # ^= IV
1725         movdqu  0x00($inp), @XMM[8]     # re-load input
1726         movdqu  0x10($inp), @XMM[9]
1727         pxor    @XMM[8], @XMM[1]
1728         movdqu  0x20($inp), @XMM[10]
1729         pxor    @XMM[9], @XMM[6]
1730         movdqu  0x30($inp), @XMM[11]
1731         pxor    @XMM[10], @XMM[4]
1732         movdqu  0x40($inp), @XMM[12]
1733         pxor    @XMM[11], @XMM[2]
1734         movdqu  0x50($inp), @XMM[15]    # IV
1735         pxor    @XMM[12], @XMM[7]
1736         movdqu  @XMM[0], 0x00($out)     # write output
1737         movdqu  @XMM[1], 0x10($out)
1738         movdqu  @XMM[6], 0x20($out)
1739         movdqu  @XMM[4], 0x30($out)
1740         movdqu  @XMM[2], 0x40($out)
1741         movdqu  @XMM[7], 0x50($out)
1742         jmp     .Lcbc_dec_done
1743 .align  16
1744 .Lcbc_dec_five:
1745         movdqa  @XMM[15], 0x20(%rbp)    # put aside IV
1746         call    _bsaes_decrypt8
1747         pxor    0x20(%rbp), @XMM[0]     # ^= IV
1748         movdqu  0x00($inp), @XMM[8]     # re-load input
1749         movdqu  0x10($inp), @XMM[9]
1750         pxor    @XMM[8], @XMM[1]
1751         movdqu  0x20($inp), @XMM[10]
1752         pxor    @XMM[9], @XMM[6]
1753         movdqu  0x30($inp), @XMM[11]
1754         pxor    @XMM[10], @XMM[4]
1755         movdqu  0x40($inp), @XMM[15]    # IV
1756         pxor    @XMM[11], @XMM[2]
1757         movdqu  @XMM[0], 0x00($out)     # write output
1758         movdqu  @XMM[1], 0x10($out)
1759         movdqu  @XMM[6], 0x20($out)
1760         movdqu  @XMM[4], 0x30($out)
1761         movdqu  @XMM[2], 0x40($out)
1762         jmp     .Lcbc_dec_done
1763 .align  16
1764 .Lcbc_dec_four:
1765         movdqa  @XMM[15], 0x20(%rbp)    # put aside IV
1766         call    _bsaes_decrypt8
1767         pxor    0x20(%rbp), @XMM[0]     # ^= IV
1768         movdqu  0x00($inp), @XMM[8]     # re-load input
1769         movdqu  0x10($inp), @XMM[9]
1770         pxor    @XMM[8], @XMM[1]
1771         movdqu  0x20($inp), @XMM[10]
1772         pxor    @XMM[9], @XMM[6]
1773         movdqu  0x30($inp), @XMM[15]    # IV
1774         pxor    @XMM[10], @XMM[4]
1775         movdqu  @XMM[0], 0x00($out)     # write output
1776         movdqu  @XMM[1], 0x10($out)
1777         movdqu  @XMM[6], 0x20($out)
```



```
1778         movdqu  @XMM[4], 0x30($out)
1779         jmp     .Lcbc_dec_done
1780 .align  16
1781 .Lcbc_dec_three:
1782         movdqa  @XMM[15], 0x20(%rbp)    # put aside IV
1783         call    _bsaes_decrypt8
1784         pxor    0x20(%rbp), @XMM[0]     # ^= IV
1785         movdqu  0x00($inp), @XMM[8]     # re-load input
1786         movdqu  0x10($inp), @XMM[9]
1787         pxor    @XMM[8], @XMM[1]
1788         movdqu  0x20($inp), @XMM[15]    # IV
1789         pxor    @XMM[9], @XMM[6]
1790         movdqu  @XMM[0], 0x00($out)     # write output
1791         movdqu  @XMM[1], 0x10($out)
1792         movdqu  @XMM[6], 0x20($out)
1793         jmp     .Lcbc_dec_done
1794 .align  16
1795 .Lcbc_dec_two:
1796         movdqa  @XMM[15], 0x20(%rbp)    # put aside IV
1797         call    _bsaes_decrypt8
1798         pxor    0x20(%rbp), @XMM[0]     # ^= IV
1799         movdqu  0x00($inp), @XMM[8]     # re-load input
1800         movdqu  0x10($inp), @XMM[15]    # IV
1801         pxor    @XMM[8], @XMM[1]
1802         movdqu  @XMM[0], 0x00($out)     # write output
1803         movdqu  @XMM[1], 0x10($out)
1804         jmp     .Lcbc_dec_done
1805 .align  16
1806 .Lcbc_dec_one:
1807         lea     ($inp), $arg1
1808         lea     0x20(%rbp), $arg2       # buffer output
1809         lea     ($key), $arg3
1810         call    asm_AES_decrypt         # doesn't touch %xmm
1811         pxor    0x20(%rbp), @XMM[15]    # ^= IV
1812         movdqu  @XMM[15], ($out)        # write output
1813         movdqa  @XMM[0], @XMM[15]       # IV

1815 .Lcbc_dec_done:
1816         movdqu  @XMM[15], (%rbx)        # return IV
1817         lea     (%rsp), %rax
1818         pxor    %xmm0, %xmm0
1819 .Lcbc_dec_bzero:                        # wipe key schedule [if any]
1820         movdqa  %xmm0, 0x00(%rax)
1821         movdqa  %xmm0, 0x10(%rax)
1822         lea     0x20(%rax), %rax
1823         cmp     %rax, %rbp
1824         ja      .Lcbc_dec_bzero

1826         lea     (%rbp),%rsp             # restore %rsp
1827 ___
1828 $code.=<<___ if ($win64);
1829         movaps  0x40(%rbp), %xmm6
1830         movaps  0x50(%rbp), %xmm7
1831         movaps  0x60(%rbp), %xmm8
1832         movaps  0x70(%rbp), %xmm9
1833         movaps  0x80(%rbp), %xmm10
1834         movaps  0x90(%rbp), %xmm11
1835         movaps  0xa0(%rbp), %xmm12
1836         movaps  0xb0(%rbp), %xmm13
1837         movaps  0xc0(%rbp), %xmm14
1838         movaps  0xd0(%rbp), %xmm15
1839         lea     0xa0(%rbp), %rsp
1840 ___
1841 $code.=<<___;
1842         mov     0x48(%rsp), %r15
1843         mov     0x50(%rsp), %r14
```

```
1844         mov     0x58(%rsp), %r13
1845         mov     0x60(%rsp), %r12
1846         mov     0x68(%rsp), %rbx
1847         mov     0x70(%rsp), %rax
1848         lea     0x78(%rsp), %rsp
1849         mov     %rax, %rbp
1850 .Lcbc_dec_epilogue:
1851         ret
1852 .size   bsaes_cbc_encrypt,.-bsaes_cbc_encrypt

1854 .globl  bsaes_ctr32_encrypt_blocks
1855 .type   bsaes_ctr32_encrypt_blocks,\@abi-omnipotent
1856 .align  16
1857 bsaes_ctr32_encrypt_blocks:
1858         mov     %rsp, %rax
1859 .Lctr_enc_prologue:
1860         push    %rbp
1861         push    %rbx
1862         push    %r12
1863         push    %r13
1864         push    %r14
1865         push    %r15
1866         lea     -0x48(%rsp), %rsp
1867 ___
1868 $code.=<<___ if ($win64);
1869         mov     0xa0(%rsp),$arg5        # pull ivp
1870         lea     -0xa0(%rsp), %rsp
1871         movaps  %xmm6, 0x40(%rsp)
1872         movaps  %xmm7, 0x50(%rsp)
1873         movaps  %xmm8, 0x60(%rsp)
1874         movaps  %xmm9, 0x70(%rsp)
1875         movaps  %xmm10, 0x80(%rsp)
1876         movaps  %xmm11, 0x90(%rsp)
1877         movaps  %xmm12, 0xa0(%rsp)
1878         movaps  %xmm13, 0xb0(%rsp)
1879         movaps  %xmm14, 0xc0(%rsp)
1880         movaps  %xmm15, 0xd0(%rsp)
1881 .Lctr_enc_body:
1882 ___
1883 $code.=<<___;
1884         mov     %rsp, %rbp              # backup %rsp
1885         movdqu  ($arg5), %xmm0          # load counter
1886         mov     240($arg4), %eax        # rounds
1887         mov     $arg1, $inp             # backup arguments
1888         mov     $arg2, $out
1889         mov     $arg3, $len
1890         mov     $arg4, $key
1891         movdqa  %xmm0, 0x20(%rbp)       # copy counter
1892         cmp     \$8, $arg3
1893         jb      .Lctr_enc_short

1895         mov     %eax, %ebx              # rounds
1896         shl     \$7, %rax               # 128 bytes per inner round key
1897         sub     \$`128-32`, %rax        # size of bit-sliced key schedule
1898         sub     %rax, %rsp

1900         mov     %rsp, %rax              # pass key schedule
1901         mov     $key, %rcx              # pass key
1902         mov     %ebx, %r10d             # pass rounds
1903         call    _bsaes_key_convert
1904         pxor    %xmm6,%xmm7             # fix up last round key
1905         movdqa  %xmm7,(%rax)            # save last round key

1907         movdqa  (%rsp), @XMM[9]         # load round0 key
1908         lea     .LADD1(%rip), %r11
1909         movdqa  0x20(%rbp), @XMM[0]     # counter copy
```



```
1910         movdqa  -0x20(%r11), @XMM[8]    # .LSWPUP
1911         pshufb  @XMM[8], @XMM[9]        # byte swap upper part
1912         pshufb  @XMM[8], @XMM[0]
1913         movdqa  @XMM[9], (%rsp)         # save adjusted round0 key
1914         jmp     .Lctr_enc_loop
1915 .align  16
1916 .Lctr_enc_loop:
1917         movdqa  @XMM[0], 0x20(%rbp)     # save counter
1918         movdqa  @XMM[0], @XMM[1]        # prepare 8 counter values
1919         movdqa  @XMM[0], @XMM[2]
1920         paddd   0x00(%r11), @XMM[1]     # .LADD1
1921         movdqa  @XMM[0], @XMM[3]
1922         paddd   0x10(%r11), @XMM[2]     # .LADD2
1923         movdqa  @XMM[0], @XMM[4]
1924         paddd   0x20(%r11), @XMM[3]     # .LADD3
1925         movdqa  @XMM[0], @XMM[5]
1926         paddd   0x30(%r11), @XMM[4]     # .LADD4
1927         movdqa  @XMM[0], @XMM[6]
1928         paddd   0x40(%r11), @XMM[5]     # .LADD5
1929         movdqa  @XMM[0], @XMM[7]
1930         paddd   0x50(%r11), @XMM[6]     # .LADD6
1931         paddd   0x60(%r11), @XMM[7]     # .LADD7

1933         # Borrow prologue from _bsaes_encrypt8 to use the opportunity
1934         # to flip byte order in 32-bit counter
1935         movdqa  (%rsp), @XMM[9]         # round 0 key
1936         lea     0x10(%rsp), %rax        # pass key schedule
1937         movdqa  -0x10(%r11), @XMM[8]    # .LSWPUPM0SR
1938         pxor    @XMM[9], @XMM[0]        # xor with round0 key
1939         pxor    @XMM[9], @XMM[1]
1940          pshufb @XMM[8], @XMM[0]
1941         pxor    @XMM[9], @XMM[2]
1942          pshufb @XMM[8], @XMM[1]
1943         pxor    @XMM[9], @XMM[3]
1944          pshufb @XMM[8], @XMM[2]
1945         pxor    @XMM[9], @XMM[4]
1946          pshufb @XMM[8], @XMM[3]
1947         pxor    @XMM[9], @XMM[5]
1948          pshufb @XMM[8], @XMM[4]
1949         pxor    @XMM[9], @XMM[6]
1950          pshufb @XMM[8], @XMM[5]
1951         pxor    @XMM[9], @XMM[7]
1952          pshufb @XMM[8], @XMM[6]
1953         lea     .LBS0(%rip), %r11       # constants table
1954          pshufb @XMM[8], @XMM[7]
1955         mov     %ebx,%r10d              # pass rounds

1957         call    _bsaes_encrypt8_bitslice

1959         sub     \$8,$len
1960         jc      .Lctr_enc_loop_done

1962         movdqu  0x00($inp), @XMM[8]     # load input
1963         movdqu  0x10($inp), @XMM[9]
1964         movdqu  0x20($inp), @XMM[10]
1965         movdqu  0x30($inp), @XMM[11]
1966         movdqu  0x40($inp), @XMM[12]
1967         movdqu  0x50($inp), @XMM[13]
1968         movdqu  0x60($inp), @XMM[14]
1969         movdqu  0x70($inp), @XMM[15]
1970         lea     0x80($inp),$inp
1971         pxor    @XMM[0], @XMM[8]
1972         movdqa  0x20(%rbp), @XMM[0]     # load counter
1973         pxor    @XMM[9], @XMM[1]
1974         movdqu  @XMM[8], 0x00($out)     # write output
1975         pxor    @XMM[10], @XMM[4]
```

```
1976         movdqu  @XMM[1], 0x10($out)
1977         pxor    @XMM[11], @XMM[6]
1978         movdqu  @XMM[4], 0x20($out)
1979         pxor    @XMM[12], @XMM[3]
1980         movdqu  @XMM[6], 0x30($out)
1981         pxor    @XMM[13], @XMM[7]
1982         movdqu  @XMM[3], 0x40($out)
1983         pxor    @XMM[14], @XMM[2]
1984         movdqu  @XMM[7], 0x50($out)
1985         pxor    @XMM[15], @XMM[5]
1986         movdqu  @XMM[2], 0x60($out)
1987         lea     .LADD1(%rip), %r11
1988         movdqu  @XMM[5], 0x70($out)
1989         lea     0x80($out), $out
1990         paddd   0x70(%r11), @XMM[0]     # .LADD8
1991         jnz     .Lctr_enc_loop

1993         jmp     .Lctr_enc_done
1994 .align  16
1995 .Lctr_enc_loop_done:
1996         add     \$8, $len
1997         movdqu  0x00($inp), @XMM[8]     # load input
1998         pxor    @XMM[8], @XMM[0]
1999         movdqu  @XMM[0], 0x00($out)     # write output
2000         cmp     \$2,$len
2001         jb      .Lctr_enc_done
2002         movdqu  0x10($inp), @XMM[9]
2003         pxor    @XMM[9], @XMM[1]
2004         movdqu  @XMM[1], 0x10($out)
2005         je      .Lctr_enc_done
2006         movdqu  0x20($inp), @XMM[10]
2007         pxor    @XMM[10], @XMM[4]
2008         movdqu  @XMM[4], 0x20($out)
2009         cmp     \$4,$len
2010         jb      .Lctr_enc_done
2011         movdqu  0x30($inp), @XMM[11]
2012         pxor    @XMM[11], @XMM[6]
2013         movdqu  @XMM[6], 0x30($out)
2014         je      .Lctr_enc_done
2015         movdqu  0x40($inp), @XMM[12]
2016         pxor    @XMM[12], @XMM[3]
2017         movdqu  @XMM[3], 0x40($out)
2018         cmp     \$6,$len
2019         jb      .Lctr_enc_done
2020         movdqu  0x50($inp), @XMM[13]
2021         pxor    @XMM[13], @XMM[7]
2022         movdqu  @XMM[7], 0x50($out)
2023         je      .Lctr_enc_done
2024         movdqu  0x60($inp), @XMM[14]
2025         pxor    @XMM[14], @XMM[2]
2026         movdqu  @XMM[2], 0x60($out)
2027         jmp     .Lctr_enc_done

2029 .align  16
2030 .Lctr_enc_short:
2031         lea     0x20(%rbp), $arg1
2032         lea     0x30(%rbp), $arg2
2033         lea     ($key), $arg3
2034         call    asm_AES_encrypt
2035         movdqu  ($inp), @XMM[1]
2036         lea     16($inp), $inp
2037         mov     0x2c(%rbp), %eax        # load 32-bit counter
2038         bswap   %eax
2039         pxor    0x30(%rbp), @XMM[1]
2040         inc     %eax                    # increment
2041         movdqu  @XMM[1], ($out)
```



```
2042         bswap   %eax
2043         lea     16($out), $out
2044         mov     %eax, 0x2c(%rsp)        # save 32-bit counter
2045         dec     $len
2046         jnz     .Lctr_enc_short

2048 .Lctr_enc_done:
2049         lea     (%rsp), %rax
2050         pxor    %xmm0, %xmm0
2051 .Lctr_enc_bzero:                        # wipe key schedule [if any]
2052         movdqa  %xmm0, 0x00(%rax)
2053         movdqa  %xmm0, 0x10(%rax)
2054         lea     0x20(%rax), %rax
2055         cmp     %rax, %rbp
2056         ja      .Lctr_enc_bzero

2058         lea     (%rbp),%rsp             # restore %rsp
2059 ___
2060 $code.=<<___ if ($win64);
2061         movaps  0x40(%rbp), %xmm6
2062         movaps  0x50(%rbp), %xmm7
2063         movaps  0x60(%rbp), %xmm8
2064         movaps  0x70(%rbp), %xmm9
2065         movaps  0x80(%rbp), %xmm10
2066         movaps  0x90(%rbp), %xmm11
2067         movaps  0xa0(%rbp), %xmm12
2068         movaps  0xb0(%rbp), %xmm13
2069         movaps  0xc0(%rbp), %xmm14
2070         movaps  0xd0(%rbp), %xmm15
2071         lea     0xa0(%rbp), %rsp
2072 ___
2073 $code.=<<___;
2074         mov     0x48(%rsp), %r15
2075         mov     0x50(%rsp), %r14
2076         mov     0x58(%rsp), %r13
2077         mov     0x60(%rsp), %r12
2078         mov     0x68(%rsp), %rbx
2079         mov     0x70(%rsp), %rax
2080         lea     0x78(%rsp), %rsp
2081         mov     %rax, %rbp
2082 .Lctr_enc_epilogue:
2083         ret
2084 .size   bsaes_ctr32_encrypt_blocks,.-bsaes_ctr32_encrypt_blocks
2085 ___
2086 ######################################################################
2087 # void bsaes_xts_[en|de]crypt(const char *inp,char *out,size_t len,
2088 #       const AES_KEY *key1, const AES_KEY *key2,
2089 #       const unsigned char iv[16]);
2090 #
2091 my ($twmask,$twres,$twtmp)=@XMM[13..15];
2092 $arg6=~s/d$//;

2094 $code.=<<___;
2095 .globl  bsaes_xts_encrypt
2096 .type   bsaes_xts_encrypt,\@abi-omnipotent
2097 .align  16
2098 bsaes_xts_encrypt:
2099         mov     %rsp, %rax
2100 .Lxts_enc_prologue:
2101         push    %rbp
2102         push    %rbx
2103         push    %r12
2104         push    %r13
2105         push    %r14
2106         push    %r15
2107         lea     -0x48(%rsp), %rsp
```

```
___
$code.=<<___ if ($win64);
        mov     0xa0(%rsp),$arg5        # pull key2
        mov     0xa8(%rsp),$arg6        # pull ivp
        lea     -0xa0(%rsp), %rsp
        movaps  %xmm6, 0x40(%rsp)
        movaps  %xmm7, 0x50(%rsp)
        movaps  %xmm8, 0x60(%rsp)
        movaps  %xmm9, 0x70(%rsp)
        movaps  %xmm10, 0x80(%rsp)
        movaps  %xmm11, 0x90(%rsp)
        movaps  %xmm12, 0xa0(%rsp)
        movaps  %xmm13, 0xb0(%rsp)
        movaps  %xmm14, 0xc0(%rsp)
        movaps  %xmm15, 0xd0(%rsp)
.Lxts_enc_body:
___
$code.=<<___;
        mov     %rsp, %rbp              # backup %rsp
        mov     $arg1, $inp             # backup arguments
        mov     $arg2, $out
        mov     $arg3, $len
        mov     $arg4, $key

        lea     ($arg6), $arg1
        lea     0x20(%rbp), $arg2
        lea     ($arg5), $arg3
        call    asm_AES_encrypt         # generate initial tweak

        mov     240($key), %eax         # rounds
        mov     $len, %rbx              # backup $len

        mov     %eax, %edx              # rounds
        shl     \$7, %rax               # 128 bytes per inner round key
        sub     \$`128-32`, %rax        # size of bit-sliced key schedule
        sub     %rax, %rsp

        mov     %rsp, %rax              # pass key schedule
        mov     $key, %rcx              # pass key
        mov     %edx, %r10d             # pass rounds
        call    _bsaes_key_convert
        pxor    %xmm6, %xmm7            # fix up last round key
        movdqa  %xmm7, (%rax)           # save last round key

        and     \$-16, $len
        sub     \$0x80, %rsp            # place for tweak[8]
        movdqa  0x20(%rbp), @XMM[7]     # initial tweak

        pxor    $twtmp, $twtmp
        movdqa  .Lxts_magic(%rip), $twmask
        pcmpgtd @XMM[7], $twtmp         # broadcast upper bits

        sub     \$0x80, $len
        jc      .Lxts_enc_short
        jmp     .Lxts_enc_loop

.align  16
.Lxts_enc_loop:
___
    for ($i=0;$i<7;$i++) {
    $code.=<<___;
        pshufd  \$0x13, $twtmp, $twres
        pxor    $twtmp, $twtmp
        movdqa  @XMM[7], @XMM[$i]
        movdqa  @XMM[7], `0x10*$i`(%rsp)# save tweak[$i]
        paddq   @XMM[7], @XMM[7]        # psllq 1,$tweak
```



```
        pand    $twmask, $twres         # isolate carry and residue
        pcmpgtd @XMM[7], $twtmp         # broadcast upper bits
        pxor    $twres, @XMM[7]
___
    $code.=<<___ if ($i>=1);
        movdqu  `0x10*($i-1)`($inp), @XMM[8+$i-1]
___
    $code.=<<___ if ($i>=2);
        pxor    @XMM[8+$i-2], @XMM[$i-2]# input[] ^ tweak[]
___
    }
$code.=<<___;
        movdqu  0x60($inp), @XMM[8+6]
        pxor    @XMM[8+5], @XMM[5]
        movdqu  0x70($inp), @XMM[8+7]
        lea     0x80($inp), $inp
        movdqa  @XMM[7], 0x70(%rsp)
        pxor    @XMM[8+6], @XMM[6]
        lea     0x80(%rsp), %rax        # pass key schedule
        pxor    @XMM[8+7], @XMM[7]
        mov     %edx, %r10d             # pass rounds

        call    _bsaes_encrypt8

        pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
        pxor    0x10(%rsp), @XMM[1]
        movdqu  @XMM[0], 0x00($out)     # write output
        pxor    0x20(%rsp), @XMM[4]
        movdqu  @XMM[1], 0x10($out)
        pxor    0x30(%rsp), @XMM[6]
        movdqu  @XMM[4], 0x20($out)
        pxor    0x40(%rsp), @XMM[3]
        movdqu  @XMM[6], 0x30($out)
        pxor    0x50(%rsp), @XMM[7]
        movdqu  @XMM[3], 0x40($out)
        pxor    0x60(%rsp), @XMM[2]
        movdqu  @XMM[7], 0x50($out)
        pxor    0x70(%rsp), @XMM[5]
        movdqu  @XMM[2], 0x60($out)
        movdqu  @XMM[5], 0x70($out)
        lea     0x80($out), $out

        movdqa  0x70(%rsp), @XMM[7]     # prepare next iteration tweak
        pxor    $twtmp, $twtmp
        movdqa  .Lxts_magic(%rip), $twmask
        pcmpgtd @XMM[7], $twtmp
        pshufd  \$0x13, $twtmp, $twres
        pxor    $twtmp, $twtmp
        paddq   @XMM[7], @XMM[7]        # psllq 1,$tweak
        pand    $twmask, $twres         # isolate carry and residue
        pcmpgtd @XMM[7], $twtmp         # broadcast upper bits
        pxor    $twres, @XMM[7]

        sub     \$0x80,$len
        jnc     .Lxts_enc_loop

.Lxts_enc_short:
        add     \$0x80, $len
        jz      .Lxts_enc_done
___
    for ($i=0;$i<7;$i++) {
    $code.=<<___;
        pshufd  \$0x13, $twtmp, $twres
        pxor    $twtmp, $twtmp
        movdqa  @XMM[7], @XMM[$i]
        movdqa  @XMM[7], `0x10*$i`(%rsp)# save tweak[$i]
```

```
2240         paddq   @XMM[7], @XMM[7]        # psllq 1,$tweak
2241         pand    $twmask, $twres         # isolate carry and residue
2242         pcmpgtd @XMM[7], $twtmp         # broadcast upper bits
2243         pxor    $twres, @XMM[7]
2244 ___
2245     $code.=<<___ if ($i>=1);
2246         movdqu  `0x10*($i-1)`($inp), @XMM[8+$i-1]
2247         cmp     \$`0x10*$i`,$len
2248         je      .Lxts_enc_$i
2249 ___
2250     $code.=<<___ if ($i>=2);
2251         pxor    @XMM[8+$i-2], @XMM[$i-2]# input[] ^ tweak[]
2252 ___
2253     }
2254 $code.=<<___;
2255         movdqu  0x60($inp), @XMM[8+6]
2256         pxor    @XMM[8+5], @XMM[5]
2257         movdqa  @XMM[7], 0x70(%rsp)
2258         lea     0x70($inp), $inp
2259         pxor    @XMM[8+6], @XMM[6]
2260         lea     0x80(%rsp), %rax        # pass key schedule
2261         mov     %edx, %r10d             # pass rounds

2263         call    _bsaes_encrypt8

2265         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2266         pxor    0x10(%rsp), @XMM[1]
2267         movdqu  @XMM[0], 0x00($out)     # write output
2268         pxor    0x20(%rsp), @XMM[4]
2269         movdqu  @XMM[1], 0x10($out)
2270         pxor    0x30(%rsp), @XMM[6]
2271         movdqu  @XMM[4], 0x20($out)
2272         pxor    0x40(%rsp), @XMM[3]
2273         movdqu  @XMM[6], 0x30($out)
2274         pxor    0x50(%rsp), @XMM[7]
2275         movdqu  @XMM[3], 0x40($out)
2276         pxor    0x60(%rsp), @XMM[2]
2277         movdqu  @XMM[7], 0x50($out)
2278         movdqu  @XMM[2], 0x60($out)
2279         lea     0x70($out), $out

2281         movdqa  0x70(%rsp), @XMM[7]     # next iteration tweak
2282         jmp     .Lxts_enc_done
2283 .align  16
2284 .Lxts_enc_6:
2285         pxor    @XMM[8+4], @XMM[4]
2286         lea     0x60($inp), $inp
2287         pxor    @XMM[8+5], @XMM[5]
2288         lea     0x80(%rsp), %rax        # pass key schedule
2289         mov     %edx, %r10d             # pass rounds

2291         call    _bsaes_encrypt8

2293         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2294         pxor    0x10(%rsp), @XMM[1]
2295         movdqu  @XMM[0], 0x00($out)     # write output
2296         pxor    0x20(%rsp), @XMM[4]
2297         movdqu  @XMM[1], 0x10($out)
2298         pxor    0x30(%rsp), @XMM[6]
2299         movdqu  @XMM[4], 0x20($out)
2300         pxor    0x40(%rsp), @XMM[3]
2301         movdqu  @XMM[6], 0x30($out)
2302         pxor    0x50(%rsp), @XMM[7]
2303         movdqu  @XMM[3], 0x40($out)
2304         movdqu  @XMM[7], 0x50($out)
2305         lea     0x60($out), $out
```



```
2307         movdqa  0x60(%rsp), @XMM[7]     # next iteration tweak
2308         jmp     .Lxts_enc_done
2309 .align  16
2310 .Lxts_enc_5:
2311         pxor    @XMM[8+3], @XMM[3]
2312         lea     0x50($inp), $inp
2313         pxor    @XMM[8+4], @XMM[4]
2314         lea     0x80(%rsp), %rax        # pass key schedule
2315         mov     %edx, %r10d             # pass rounds

2317         call    _bsaes_encrypt8

2319         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2320         pxor    0x10(%rsp), @XMM[1]
2321         movdqu  @XMM[0], 0x00($out)     # write output
2322         pxor    0x20(%rsp), @XMM[4]
2323         movdqu  @XMM[1], 0x10($out)
2324         pxor    0x30(%rsp), @XMM[6]
2325         movdqu  @XMM[4], 0x20($out)
2326         pxor    0x40(%rsp), @XMM[3]
2327         movdqu  @XMM[6], 0x30($out)
2328         movdqu  @XMM[3], 0x40($out)
2329         lea     0x50($out), $out

2331         movdqa  0x50(%rsp), @XMM[7]     # next iteration tweak
2332         jmp     .Lxts_enc_done
2333 .align  16
2334 .Lxts_enc_4:
2335         pxor    @XMM[8+2], @XMM[2]
2336         lea     0x40($inp), $inp
2337         pxor    @XMM[8+3], @XMM[3]
2338         lea     0x80(%rsp), %rax        # pass key schedule
2339         mov     %edx, %r10d             # pass rounds

2341         call    _bsaes_encrypt8

2343         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2344         pxor    0x10(%rsp), @XMM[1]
2345         movdqu  @XMM[0], 0x00($out)     # write output
2346         pxor    0x20(%rsp), @XMM[4]
2347         movdqu  @XMM[1], 0x10($out)
2348         pxor    0x30(%rsp), @XMM[6]
2349         movdqu  @XMM[4], 0x20($out)
2350         movdqu  @XMM[6], 0x30($out)
2351         lea     0x40($out), $out

2353         movdqa  0x40(%rsp), @XMM[7]     # next iteration tweak
2354         jmp     .Lxts_enc_done
2355 .align  16
2356 .Lxts_enc_3:
2357         pxor    @XMM[8+1], @XMM[1]
2358         lea     0x30($inp), $inp
2359         pxor    @XMM[8+2], @XMM[2]
2360         lea     0x80(%rsp), %rax        # pass key schedule
2361         mov     %edx, %r10d             # pass rounds

2363         call    _bsaes_encrypt8

2365         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2366         pxor    0x10(%rsp), @XMM[1]
2367         movdqu  @XMM[0], 0x00($out)     # write output
2368         pxor    0x20(%rsp), @XMM[4]
2369         movdqu  @XMM[1], 0x10($out)
2370         movdqu  @XMM[4], 0x20($out)
2371         lea     0x30($out), $out
```

```
2373         movdqa  0x30(%rsp), @XMM[7]     # next iteration tweak
2374         jmp     .Lxts_enc_done
2375 .align  16
2376 .Lxts_enc_2:
2377         pxor    @XMM[8+0], @XMM[0]
2378         lea     0x20($inp), $inp
2379         pxor    @XMM[8+1], @XMM[1]
2380         lea     0x80(%rsp), %rax        # pass key schedule
2381         mov     %edx, %r10d             # pass rounds

2383         call    _bsaes_encrypt8

2385         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2386         pxor    0x10(%rsp), @XMM[1]
2387         movdqu  @XMM[0], 0x00($out)     # write output
2388         movdqu  @XMM[1], 0x10($out)
2389         lea     0x20($out), $out

2391         movdqa  0x20(%rsp), @XMM[7]     # next iteration tweak
2392         jmp     .Lxts_enc_done
2393 .align  16
2394 .Lxts_enc_1:
2395         pxor    @XMM[0], @XMM[8]
2396         lea     0x10($inp), $inp
2397         movdqa  @XMM[8], 0x20(%rbp)
2398         lea     0x20(%rbp), $arg1
2399         lea     0x20(%rbp), $arg2
2400         lea     ($key), $arg3
2401         call    asm_AES_encrypt         # doesn't touch %xmm
2402         pxor    0x20(%rbp), @XMM[0]     # ^= tweak[]
2403         #pxor   @XMM[8], @XMM[0]
2404         #lea    0x80(%rsp), %rax        # pass key schedule
2405         #mov    %edx, %r10d             # pass rounds
2406         #call   _bsaes_encrypt8
2407         #pxor   0x00(%rsp), @XMM[0]     # ^= tweak[]
2408         movdqu  @XMM[0], 0x00($out)     # write output
2409         lea     0x10($out), $out

2411         movdqa  0x10(%rsp), @XMM[7]     # next iteration tweak

2413 .Lxts_enc_done:
2414         and     \$15, %ebx
2415         jz      .Lxts_enc_ret
2416         mov     $out, %rdx

2418 .Lxts_enc_steal:
2419         movzb   ($inp), %eax
2420         movzb   -16(%rdx), %ecx
2421         lea     1($inp), $inp
2422         mov     %al, -16(%rdx)
2423         mov     %cl, 0(%rdx)
2424         lea     1(%rdx), %rdx
2425         sub     \$1,%ebx
2426         jnz     .Lxts_enc_steal

2428         movdqu  -16($out), @XMM[0]
2429         lea     0x20(%rbp), $arg1
2430         pxor    @XMM[7], @XMM[0]
2431         lea     0x20(%rbp), $arg2
2432         movdqa  @XMM[0], 0x20(%rbp)
2433         lea     ($key), $arg3
2434         call    asm_AES_encrypt         # doesn't touch %xmm
2435         pxor    0x20(%rbp), @XMM[7]
2436         movdqu  @XMM[7], -16($out)
```



```
2438 .Lxts_enc_ret:
2439         lea     (%rsp), %rax
2440         pxor    %xmm0, %xmm0
2441 .Lxts_enc_bzero:                        # wipe key schedule [if any]
2442         movdqa  %xmm0, 0x00(%rax)
2443         movdqa  %xmm0, 0x10(%rax)
2444         lea     0x20(%rax), %rax
2445         cmp     %rax, %rbp
2446         ja      .Lxts_enc_bzero

2448         lea     (%rbp),%rsp             # restore %rsp
2449 ___
2450 $code.=<<___ if ($win64);
2451         movaps  0x40(%rbp), %xmm6
2452         movaps  0x50(%rbp), %xmm7
2453         movaps  0x60(%rbp), %xmm8
2454         movaps  0x70(%rbp), %xmm9
2455         movaps  0x80(%rbp), %xmm10
2456         movaps  0x90(%rbp), %xmm11
2457         movaps  0xa0(%rbp), %xmm12
2458         movaps  0xb0(%rbp), %xmm13
2459         movaps  0xc0(%rbp), %xmm14
2460         movaps  0xd0(%rbp), %xmm15
2461         lea     0xa0(%rbp), %rsp
2462 ___
2463 $code.=<<___;
2464         mov     0x48(%rsp), %r15
2465         mov     0x50(%rsp), %r14
2466         mov     0x58(%rsp), %r13
2467         mov     0x60(%rsp), %r12
2468         mov     0x68(%rsp), %rbx
2469         mov     0x70(%rsp), %rax
2470         lea     0x78(%rsp), %rsp
2471         mov     %rax, %rbp
2472 .Lxts_enc_epilogue:
2473         ret
2474 .size   bsaes_xts_encrypt,.-bsaes_xts_encrypt

2476 .globl  bsaes_xts_decrypt
2477 .type   bsaes_xts_decrypt,\@abi-omnipotent
2478 .align  16
2479 bsaes_xts_decrypt:
2480         mov     %rsp, %rax
2481 .Lxts_dec_prologue:
2482         push    %rbp
2483         push    %rbx
2484         push    %r12
2485         push    %r13
2486         push    %r14
2487         push    %r15
2488         lea     -0x48(%rsp), %rsp
2489 ___
2490 $code.=<<___ if ($win64);
2491         mov     0xa0(%rsp),$arg5        # pull key2
2492         mov     0xa8(%rsp),$arg6        # pull ivp
2493         lea     -0xa0(%rsp), %rsp
2494         movaps  %xmm6, 0x40(%rsp)
2495         movaps  %xmm7, 0x50(%rsp)
2496         movaps  %xmm8, 0x60(%rsp)
2497         movaps  %xmm9, 0x70(%rsp)
2498         movaps  %xmm10, 0x80(%rsp)
2499         movaps  %xmm11, 0x90(%rsp)
2500         movaps  %xmm12, 0xa0(%rsp)
2501         movaps  %xmm13, 0xb0(%rsp)
2502         movaps  %xmm14, 0xc0(%rsp)
2503         movaps  %xmm15, 0xd0(%rsp)
```

```
.Lxts_dec_body:
___
$code.=<<___;
        mov     %rsp, %rbp              # backup %rsp
        mov     $arg1, $inp             # backup arguments
        mov     $arg2, $out
        mov     $arg3, $len
        mov     $arg4, $key

        lea     ($arg6), $arg1
        lea     0x20(%rbp), $arg2
        lea     ($arg5), $arg3
        call    asm_AES_encrypt         # generate initial tweak

        mov     240($key), %eax         # rounds
        mov     $len, %rbx              # backup $len

        mov     %eax, %edx              # rounds
        shl     \$7, %rax               # 128 bytes per inner round key
        sub     \$`128-32`, %rax        # size of bit-sliced key schedule
        sub     %rax, %rsp

        mov     %rsp, %rax              # pass key schedule
        mov     $key, %rcx              # pass key
        mov     %edx, %r10d             # pass rounds
        call    _bsaes_key_convert
        pxor    (%rsp), %xmm7           # fix up round 0 key
        movdqa  %xmm6, (%rax)           # save last round key
        movdqa  %xmm7, (%rsp)

        xor     %eax, %eax              # if ($len%16) len-=16;
        and     \$-16, $len
        test    \$15, %ebx
        setnz   %al
        shl     \$4, %rax
        sub     %rax, $len

        sub     \$0x80, %rsp            # place for tweak[8]
        movdqa  0x20(%rbp), @XMM[7]     # initial tweak

        pxor    $twtmp, $twtmp
        movdqa  .Lxts_magic(%rip), $twmask
        pcmpgtd @XMM[7], $twtmp         # broadcast upper bits

        sub     \$0x80, $len
        jc      .Lxts_dec_short
        jmp     .Lxts_dec_loop

.align  16
.Lxts_dec_loop:
___
    for ($i=0;$i<7;$i++) {
    $code.=<<___;
        pshufd  \$0x13, $twtmp, $twres
        pxor    $twtmp, $twtmp
        movdqa  @XMM[7], @XMM[$i]
        movdqa  @XMM[7], `0x10*$i`(%rsp)# save tweak[$i]
        paddq   @XMM[7], @XMM[7]        # psllq 1,$tweak
        pand    $twmask, $twres         # isolate carry and residue
        pcmpgtd @XMM[7], $twtmp         # broadcast upper bits
        pxor    $twres, @XMM[7]
___
    $code.=<<___ if ($i>=1);
        movdqu  `0x10*($i-1)`($inp), @XMM[8+$i-1]
___
    $code.=<<___ if ($i>=2);
```



```
        pxor    @XMM[8+$i-2], @XMM[$i-2]# input[] ^ tweak[]
___
    }
$code.=<<___;
        movdqu  0x60($inp), @XMM[8+6]
        pxor    @XMM[8+5], @XMM[5]
        movdqu  0x70($inp), @XMM[8+7]
        lea     0x80($inp), $inp
        movdqa  @XMM[7], 0x70(%rsp)
        pxor    @XMM[8+6], @XMM[6]
        lea     0x80(%rsp), %rax        # pass key schedule
        pxor    @XMM[8+7], @XMM[7]
        mov     %edx, %r10d             # pass rounds

        call    _bsaes_decrypt8

        pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
        pxor    0x10(%rsp), @XMM[1]
        movdqu  @XMM[0], 0x00($out)     # write output
        pxor    0x20(%rsp), @XMM[6]
        movdqu  @XMM[1], 0x10($out)
        pxor    0x30(%rsp), @XMM[4]
        movdqu  @XMM[6], 0x20($out)
        pxor    0x40(%rsp), @XMM[2]
        movdqu  @XMM[4], 0x30($out)
        pxor    0x50(%rsp), @XMM[7]
        movdqu  @XMM[2], 0x40($out)
        pxor    0x60(%rsp), @XMM[3]
        movdqu  @XMM[7], 0x50($out)
        pxor    0x70(%rsp), @XMM[5]
        movdqu  @XMM[3], 0x60($out)
        movdqu  @XMM[5], 0x70($out)
        lea     0x80($out), $out

        movdqa  0x70(%rsp), @XMM[7]     # prepare next iteration tweak
        pxor    $twtmp, $twtmp
        movdqa  .Lxts_magic(%rip), $twmask
        pcmpgtd @XMM[7], $twtmp
        pshufd  \$0x13, $twtmp, $twres
        pxor    $twtmp, $twtmp
        paddq   @XMM[7], @XMM[7]        # psllq 1,$tweak
        pand    $twmask, $twres         # isolate carry and residue
        pcmpgtd @XMM[7], $twtmp         # broadcast upper bits
        pxor    $twres, @XMM[7]

        sub     \$0x80,$len
        jnc     .Lxts_dec_loop

.Lxts_dec_short:
        add     \$0x80, $len
        jz      .Lxts_dec_done
___
    for ($i=0;$i<7;$i++) {
    $code.=<<___;
        pshufd  \$0x13, $twtmp, $twres
        pxor    $twtmp, $twtmp
        movdqa  @XMM[7], @XMM[$i]
        movdqa  @XMM[7], `0x10*$i`(%rsp)# save tweak[$i]
        paddq   @XMM[7], @XMM[7]        # psllq 1,$tweak
        pand    $twmask, $twres         # isolate carry and residue
        pcmpgtd @XMM[7], $twtmp         # broadcast upper bits
        pxor    $twres, @XMM[7]
___
    $code.=<<___ if ($i>=1);
        movdqu  `0x10*($i-1)`($inp), @XMM[8+$i-1]
        cmp     \$`0x10*$i`,$len
```

```
2636         je      .Lxts_dec_$i
2637 ___
2638     $code.=<<___ if ($i>=2);
2639         pxor    @XMM[8+$i-2], @XMM[$i-2]# input[] ^ tweak[]
2640 ___
2641     }
2642 $code.=<<___;
2643         movdqu  0x60($inp), @XMM[8+6]
2644         pxor    @XMM[8+5], @XMM[5]
2645         movdqa  @XMM[7], 0x70(%rsp)
2646         lea     0x70($inp), $inp
2647         pxor    @XMM[8+6], @XMM[6]
2648         lea     0x80(%rsp), %rax        # pass key schedule
2649         mov     %edx, %r10d             # pass rounds

2651         call    _bsaes_decrypt8

2653         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2654         pxor    0x10(%rsp), @XMM[1]
2655         movdqu  @XMM[0], 0x00($out)     # write output
2656         pxor    0x20(%rsp), @XMM[6]
2657         movdqu  @XMM[1], 0x10($out)
2658         pxor    0x30(%rsp), @XMM[4]
2659         movdqu  @XMM[6], 0x20($out)
2660         pxor    0x40(%rsp), @XMM[2]
2661         movdqu  @XMM[4], 0x30($out)
2662         pxor    0x50(%rsp), @XMM[7]
2663         movdqu  @XMM[2], 0x40($out)
2664         pxor    0x60(%rsp), @XMM[3]
2665         movdqu  @XMM[7], 0x50($out)
2666         movdqu  @XMM[3], 0x60($out)
2667         lea     0x70($out), $out

2669         movdqa  0x70(%rsp), @XMM[7]     # next iteration tweak
2670         jmp     .Lxts_dec_done
2671 .align  16
2672 .Lxts_dec_6:
2673         pxor    @XMM[8+4], @XMM[4]
2674         lea     0x60($inp), $inp
2675         pxor    @XMM[8+5], @XMM[5]
2676         lea     0x80(%rsp), %rax        # pass key schedule
2677         mov     %edx, %r10d             # pass rounds

2679         call    _bsaes_decrypt8

2681         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2682         pxor    0x10(%rsp), @XMM[1]
2683         movdqu  @XMM[0], 0x00($out)     # write output
2684         pxor    0x20(%rsp), @XMM[6]
2685         movdqu  @XMM[1], 0x10($out)
2686         pxor    0x30(%rsp), @XMM[4]
2687         movdqu  @XMM[6], 0x20($out)
2688         pxor    0x40(%rsp), @XMM[2]
2689         movdqu  @XMM[4], 0x30($out)
2690         pxor    0x50(%rsp), @XMM[7]
2691         movdqu  @XMM[2], 0x40($out)
2692         movdqu  @XMM[7], 0x50($out)
2693         lea     0x60($out), $out

2695         movdqa  0x60(%rsp), @XMM[7]     # next iteration tweak
2696         jmp     .Lxts_dec_done
2697 .align  16
2698 .Lxts_dec_5:
2699         pxor    @XMM[8+3], @XMM[3]
2700         lea     0x50($inp), $inp
2701         pxor    @XMM[8+4], @XMM[4]
```



```
2702         lea     0x80(%rsp), %rax        # pass key schedule
2703         mov     %edx, %r10d             # pass rounds

2705         call    _bsaes_decrypt8

2707         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2708         pxor    0x10(%rsp), @XMM[1]
2709         movdqu  @XMM[0], 0x00($out)     # write output
2710         pxor    0x20(%rsp), @XMM[6]
2711         movdqu  @XMM[1], 0x10($out)
2712         pxor    0x30(%rsp), @XMM[4]
2713         movdqu  @XMM[6], 0x20($out)
2714         pxor    0x40(%rsp), @XMM[2]
2715         movdqu  @XMM[4], 0x30($out)
2716         movdqu  @XMM[2], 0x40($out)
2717         lea     0x50($out), $out

2719         movdqa  0x50(%rsp), @XMM[7]     # next iteration tweak
2720         jmp     .Lxts_dec_done
2721 .align  16
2722 .Lxts_dec_4:
2723         pxor    @XMM[8+2], @XMM[2]
2724         lea     0x40($inp), $inp
2725         pxor    @XMM[8+3], @XMM[3]
2726         lea     0x80(%rsp), %rax        # pass key schedule
2727         mov     %edx, %r10d             # pass rounds

2729         call    _bsaes_decrypt8

2731         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2732         pxor    0x10(%rsp), @XMM[1]
2733         movdqu  @XMM[0], 0x00($out)     # write output
2734         pxor    0x20(%rsp), @XMM[6]
2735         movdqu  @XMM[1], 0x10($out)
2736         pxor    0x30(%rsp), @XMM[4]
2737         movdqu  @XMM[6], 0x20($out)
2738         movdqu  @XMM[4], 0x30($out)
2739         lea     0x40($out), $out

2741         movdqa  0x40(%rsp), @XMM[7]     # next iteration tweak
2742         jmp     .Lxts_dec_done
2743 .align  16
2744 .Lxts_dec_3:
2745         pxor    @XMM[8+1], @XMM[1]
2746         lea     0x30($inp), $inp
2747         pxor    @XMM[8+2], @XMM[2]
2748         lea     0x80(%rsp), %rax        # pass key schedule
2749         mov     %edx, %r10d             # pass rounds

2751         call    _bsaes_decrypt8

2753         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2754         pxor    0x10(%rsp), @XMM[1]
2755         movdqu  @XMM[0], 0x00($out)     # write output
2756         pxor    0x20(%rsp), @XMM[6]
2757         movdqu  @XMM[1], 0x10($out)
2758         movdqu  @XMM[6], 0x20($out)
2759         lea     0x30($out), $out

2761         movdqa  0x30(%rsp), @XMM[7]     # next iteration tweak
2762         jmp     .Lxts_dec_done
2763 .align  16
2764 .Lxts_dec_2:
2765         pxor    @XMM[8+0], @XMM[0]
2766         lea     0x20($inp), $inp
2767         pxor    @XMM[8+1], @XMM[1]
```

```
2768         lea     0x80(%rsp), %rax        # pass key schedule
2769         mov     %edx, %r10d             # pass rounds

2771         call    _bsaes_decrypt8

2773         pxor    0x00(%rsp), @XMM[0]     # ^= tweak[]
2774         pxor    0x10(%rsp), @XMM[1]
2775         movdqu  @XMM[0], 0x00($out)     # write output
2776         movdqu  @XMM[1], 0x10($out)
2777         lea     0x20($out), $out

2779         movdqa  0x20(%rsp), @XMM[7]     # next iteration tweak
2780         jmp     .Lxts_dec_done
2781 .align  16
2782 .Lxts_dec_1:
2783         pxor    @XMM[0], @XMM[8]
2784         lea     0x10($inp), $inp
2785         movdqa  @XMM[8], 0x20(%rbp)
2786         lea     0x20(%rbp), $arg1
2787         lea     0x20(%rbp), $arg2
2788         lea     ($key), $arg3
2789         call    asm_AES_decrypt         # doesn't touch %xmm
2790         pxor    0x20(%rbp), @XMM[0]     # ^= tweak[]
2791         #pxor   @XMM[8], @XMM[0]
2792         #lea    0x80(%rsp), %rax        # pass key schedule
2793         #mov    %edx, %r10d             # pass rounds
2794         #call   _bsaes_decrypt8
2795         #pxor   0x00(%rsp), @XMM[0]     # ^= tweak[]
2796         movdqu  @XMM[0], 0x00($out)     # write output
2797         lea     0x10($out), $out

2799         movdqa  0x10(%rsp), @XMM[7]     # next iteration tweak

2801 .Lxts_dec_done:
2802         and     \$15, %ebx
2803         jz      .Lxts_dec_ret

2805         pxor    $twtmp, $twtmp
2806         movdqa  .Lxts_magic(%rip), $twmask
2807         pcmpgtd @XMM[7], $twtmp
2808         pshufd  \$0x13, $twtmp, $twres
2809         movdqa  @XMM[7], @XMM[6]
2810         paddq   @XMM[7], @XMM[7]        # psllq 1,$tweak
2811         pand    $twmask, $twres         # isolate carry and residue
2812         movdqu  ($inp), @XMM[0]
2813         pxor    $twres, @XMM[7]

2815         lea     0x20(%rbp), $arg1
2816         pxor    @XMM[7], @XMM[0]
2817         lea     0x20(%rbp), $arg2
2818         movdqa  @XMM[0], 0x20(%rbp)
2819         lea     ($key), $arg3
2820         call    asm_AES_decrypt         # doesn't touch %xmm
2821         pxor    0x20(%rbp), @XMM[7]
2822         mov     $out, %rdx
2823         movdqu  @XMM[7], ($out)

2825 .Lxts_dec_steal:
2826         movzb   16($inp), %eax
2827         movzb   (%rdx), %ecx
2828         lea     1($inp), $inp
2829         mov     %al, (%rdx)
2830         mov     %cl, 16(%rdx)
2831         lea     1(%rdx), %rdx
2832         sub     \$1,%ebx
2833         jnz     .Lxts_dec_steal
```



```
2835         movdqu  ($out), @XMM[0]
2836         lea     0x20(%rbp), $arg1
2837         pxor    @XMM[6], @XMM[0]
2838         lea     0x20(%rbp), $arg2
2839         movdqa  @XMM[0], 0x20(%rbp)
2840         lea     ($key), $arg3
2841         call    asm_AES_decrypt         # doesn't touch %xmm
2842         pxor    0x20(%rbp), @XMM[6]
2843         movdqu  @XMM[6], ($out)

2845 .Lxts_dec_ret:
2846         lea     (%rsp), %rax
2847         pxor    %xmm0, %xmm0
2848 .Lxts_dec_bzero:                        # wipe key schedule [if any]
2849         movdqa  %xmm0, 0x00(%rax)
2850         movdqa  %xmm0, 0x10(%rax)
2851         lea     0x20(%rax), %rax
2852         cmp     %rax, %rbp
2853         ja      .Lxts_dec_bzero

2855         lea     (%rbp),%rsp             # restore %rsp
2856 ___
2857 $code.=<<___ if ($win64);
2858         movaps  0x40(%rbp), %xmm6
2859         movaps  0x50(%rbp), %xmm7
2860         movaps  0x60(%rbp), %xmm8
2861         movaps  0x70(%rbp), %xmm9
2862         movaps  0x80(%rbp), %xmm10
2863         movaps  0x90(%rbp), %xmm11
2864         movaps  0xa0(%rbp), %xmm12
2865         movaps  0xb0(%rbp), %xmm13
2866         movaps  0xc0(%rbp), %xmm14
2867         movaps  0xd0(%rbp), %xmm15
2868         lea     0xa0(%rbp), %rsp
2869 ___
2870 $code.=<<___;
2871         mov     0x48(%rsp), %r15
2872         mov     0x50(%rsp), %r14
2873         mov     0x58(%rsp), %r13
2874         mov     0x60(%rsp), %r12
2875         mov     0x68(%rsp), %rbx
2876         mov     0x70(%rsp), %rax
2877         lea     0x78(%rsp), %rsp
2878         mov     %rax, %rbp
2879 .Lxts_dec_epilogue:
2880         ret
2881 .size   bsaes_xts_decrypt,.-bsaes_xts_decrypt
2882 ___
2883 }
2884 $code.=<<___;
2885 .type   _bsaes_const,\@object
2886 .align  64
2887 _bsaes_const:
2888 .LM0ISR:        # InvShiftRows constants
2889         .quad   0x0a0e0206070b0f03, 0x0004080c0d010509
2890 .LISRM0:
2891         .quad   0x01040b0e0205080f, 0x0306090c00070a0d
2892 .LISR:
2893         .quad   0x0504070602010003, 0x0f0e0d0c080b0a09
2894 .LBS0:          # bit-slice constants
2895         .quad   0x5555555555555555, 0x5555555555555555
2896 .LBS1:
2897         .quad   0x3333333333333333, 0x3333333333333333
2898 .LBS2:
2899         .quad   0x0f0f0f0f0f0f0f0f, 0x0f0f0f0f0f0f0f0f
```

```
.LSR:           # shiftrows constants
        .quad   0x0504070600030201, 0x0f0e0d0c0a09080b
.LSRM0:
        .quad   0x0304090e00050a0f, 0x01060b0c0207080d
.LM0SR:
        .quad   0x0a0e02060f03070b, 0x0004080c05090d01
.LSWPUP:        # byte-swap upper dword
        .quad   0x0706050403020100, 0x0c0d0e0f0b0a0908
.LSWPUPM0SR:
        .quad   0x0a0d02060c03070b, 0x0004080f05090e01
.LADD1:         # counter increment constants
        .quad   0x0000000000000000, 0x0000000100000000
.LADD2:
        .quad   0x0000000000000000, 0x0000000200000000
.LADD3:
        .quad   0x0000000000000000, 0x0000000300000000
.LADD4:
        .quad   0x0000000000000000, 0x0000000400000000
.LADD5:
        .quad   0x0000000000000000, 0x0000000500000000
.LADD6:
        .quad   0x0000000000000000, 0x0000000600000000
.LADD7:
        .quad   0x0000000000000000, 0x0000000700000000
.LADD8:
        .quad   0x0000000000000000, 0x0000000800000000
.Lxts_magic:
        .long   0x87,0,1,0
.Lmasks:
        .quad   0x0101010101010101, 0x0101010101010101
        .quad   0x0202020202020202, 0x0202020202020202
        .quad   0x0404040404040404, 0x0404040404040404
        .quad   0x0808080808080808, 0x0808080808080808
.LM0:
        .quad   0x02060a0e03070b0f, 0x0004080c0105090d
.L63:
        .quad   0x6363636363636363, 0x6363636363636363
.asciz  "Bit-sliced AES for x86_64/SSSE3, Emilia K^/sper, Peter Schwabe, Andy Po
.align  64
.size   _bsaes_const,.-_bsaes_const
___

# EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
#               CONTEXT *context,DISPATCHER_CONTEXT *disp)
if ($win64) {
$rec="%rcx";
$frame="%rdx";
$context="%r8";
$disp="%r9";

$code.=<<___;
.extern __imp_RtlVirtualUnwind
.type   se_handler,\@abi-omnipotent
.align  16
se_handler:
        push    %rsi
        push    %rdi
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
        pushfq
        sub     \$64,%rsp
```



```
        mov     120($context),%rax      # pull context->Rax
        mov     248($context),%rbx      # pull context->Rip

        mov     8($disp),%rsi           # disp->ImageBase
        mov     56($disp),%r11          # disp->HandlerData

        mov     0(%r11),%r10d           # HandlerData[0]
        lea     (%rsi,%r10),%r10        # prologue label
        cmp     %r10,%rbx               # context->Rip<prologue label
        jb      .Lin_prologue

        mov     152($context),%rax      # pull context->Rsp

        mov     4(%r11),%r10d           # HandlerData[1]
        lea     (%rsi,%r10),%r10        # epilogue label
        cmp     %r10,%rbx               # context->Rip>=epilogue label
        jae     .Lin_prologue

        mov     160($context),%rax      # pull context->Rbp

        lea     0x40(%rax),%rsi         # %xmm save area
        lea     512($context),%rdi      # &context.Xmm6
        mov     \$20,%ecx               # 10*sizeof(%xmm0)/sizeof(%rax)
        .long   0xa548f3fc              # cld; rep movsq
        lea     0xa0(%rax),%rax         # adjust stack pointer

        mov     0x70(%rax),%rbp
        mov     0x68(%rax),%rbx
        mov     0x60(%rax),%r12
        mov     0x58(%rax),%r13
        mov     0x50(%rax),%r14
        mov     0x48(%rax),%r15
        lea     0x78(%rax),%rax         # adjust stack pointer
        mov     %rbx,144($context)      # restore context->Rbx
        mov     %rbp,160($context)      # restore context->Rbp
        mov     %r12,216($context)      # restore context->R12
        mov     %r13,224($context)      # restore context->R13
        mov     %r14,232($context)      # restore context->R14
        mov     %r15,240($context)      # restore context->R15

.Lin_prologue:
        mov     %rax,152($context)      # restore context->Rsp

        mov     40($disp),%rdi          # disp->ContextRecord
        mov     $context,%rsi           # context
        mov     \$‘1232/8‘,%ecx         # sizeof(CONTEXT)
        .long   0xa548f3fc              # cld; rep movsq

        mov     $disp,%rsi
        xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
        mov     8(%rsi),%rdx            # arg2, disp->ImageBase
        mov     0(%rsi),%r8             # arg3, disp->ControlPc
        mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
        mov     40(%rsi),%r10           # disp->ContextRecord
        lea     56(%rsi),%r11           # &disp->HandlerData
        lea     24(%rsi),%r12           # &disp->EstablisherFrame
        mov     %r10,32(%rsp)           # arg5
        mov     %r11,40(%rsp)           # arg6
        mov     %r12,48(%rsp)           # arg7
        mov     %rcx,56(%rsp)           # arg8, (NULL)
        call    *__imp_RtlVirtualUnwind(%rip)

        mov     \$1,%eax                # ExceptionContinueSearch
        add     \$64,%rsp
        popfq
        pop     %r15
```

```
3032         pop     %r14
3033         pop     %r13
3034         pop     %r12
3035         pop     %rbp
3036         pop     %rbx
3037         pop     %rdi
3038         pop     %rsi
3039         ret
3040 .size   se_handler,.-se_handler

3042 .section        .pdata
3043 .align  4
3044 ___
3045 $code.=<<___ if ($ecb);
3046         .rva    .Lecb_enc_prologue
3047         .rva    .Lecb_enc_epilogue
3048         .rva    .Lecb_enc_info

3050         .rva    .Lecb_dec_prologue
3051         .rva    .Lecb_dec_epilogue
3052         .rva    .Lecb_dec_info
3053 ___
3054 $code.=<<___;
3055         .rva    .Lcbc_dec_prologue
3056         .rva    .Lcbc_dec_epilogue
3057         .rva    .Lcbc_dec_info

3059         .rva    .Lctr_enc_prologue
3060         .rva    .Lctr_enc_epilogue
3061         .rva    .Lctr_enc_info

3063         .rva    .Lxts_enc_prologue
3064         .rva    .Lxts_enc_epilogue
3065         .rva    .Lxts_enc_info

3067         .rva    .Lxts_dec_prologue
3068         .rva    .Lxts_dec_epilogue
3069         .rva    .Lxts_dec_info

3071 .section        .xdata
3072 .align  8
3073 ___
3074 $code.=<<___ if ($ecb);
3075 .Lecb_enc_info:
3076         .byte   9,0,0,0
3077         .rva    se_handler
3078         .rva    .Lecb_enc_body,.Lecb_enc_epilogue       # HandlerData[]
3079 .Lecb_dec_info:
3080         .byte   9,0,0,0
3081         .rva    se_handler
3082         .rva    .Lecb_dec_body,.Lecb_dec_epilogue       # HandlerData[]
3083 ___
3084 $code.=<<___;
3085 .Lcbc_dec_info:
3086         .byte   9,0,0,0
3087         .rva    se_handler
3088         .rva    .Lcbc_dec_body,.Lcbc_dec_epilogue       # HandlerData[]
3089 .Lctr_enc_info:
3090         .byte   9,0,0,0
3091         .rva    se_handler
3092         .rva    .Lctr_enc_body,.Lctr_enc_epilogue       # HandlerData[]
3093 .Lxts_enc_info:
3094         .byte   9,0,0,0
3095         .rva    se_handler
3096         .rva    .Lxts_enc_body,.Lxts_enc_epilogue       # HandlerData[]
3097 .Lxts_dec_info:
```



```
3098         .byte   9,0,0,0
3099         .rva    se_handler
3100         .rva    .Lxts_dec_body,.Lxts_dec_epilogue       # HandlerData[]
3101 ___
3102 }

3104 $code =~ s/\`([^\`]*)\`/eval($1)/gem;

3106 print $code;

3108 close STDOUT;
3109 #endif /* ! codereview */
```

```
**********************************************************
    9075 Wed Aug 13 19:53:06 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/cbc.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl

# void des_ncbc_encrypt(input, output, length, schedule, ivec, enc)
# des_cblock (*input);
# des_cblock (*output);
# long length;
# des_key_schedule schedule;
# des_cblock (*ivec);
# int enc;
#
# calls
# des_encrypt((DES_LONG *)tin,schedule,DES_ENCRYPT);
#

#&cbc("des_ncbc_encrypt","des_encrypt",0);
#&cbc("BF_cbc_encrypt","BF_encrypt","BF_encrypt",
#       1,4,5,3,5,-1);
#&cbc("des_ncbc_encrypt","des_encrypt","des_encrypt",
#       0,4,5,3,5,-1);
#&cbc("des_ede3_cbc_encrypt","des_encrypt3","des_decrypt3",
#       0,6,7,3,4,5);
#
# When doing a cipher that needs bigendian order,
# for encrypt, the iv is kept in bigendian form,
# while for decrypt, it is kept in little endian.
sub cbc
        {
        local($name,$enc_func,$dec_func,$swap,$iv_off,$enc_off,$p1,$p2,$p3)=@_;
        # name is the function name
        # enc_func and dec_func and the functions to call for encrypt/decrypt
        # swap is true if byte order needs to be reversed
        # iv_off is parameter number for the iv
        # enc_off is parameter number for the encrypt/decrypt flag
        # p1,p2,p3 are the offsets for parameters to be passed to the
        # underlying calls.

        &function_begin_B($name,"");
        &comment("");

        $in="esi";
        $out="edi";
        $count="ebp";

        &push("ebp");
        &push("ebx");
        &push("esi");
        &push("edi");

        $data_off=4;
        $data_off+=4 if ($p1 > 0);
        $data_off+=4 if ($p2 > 0);
        $data_off+=4 if ($p3 > 0);

        &mov($count,    &wparam(2));    # length

        &comment("getting iv ptr from parameter $iv_off");
        &mov("ebx",     &wparam($iv_off));      # Get iv ptr

        &mov($in,       &DWP(0,"ebx","",0));#   iv[0]
        &mov($out,      &DWP(4,"ebx","",0));#   iv[1]
```



```
        &push($out);
        &push($in);
        &push($out);    # used in decrypt for iv[1]
        &push($in);     # used in decrypt for iv[0]

        &mov("ebx",     "esp");         # This is the address of tin[2]

        &mov($in,       &wparam(0));    # in
        &mov($out,      &wparam(1));    # out

        # We have loaded them all, how lets push things
        &comment("getting encrypt flag from parameter $enc_off");
        &mov("ecx",     &wparam($enc_off));     # Get enc flag
        if ($p3 > 0)
                {
                &comment("get and push parameter $p3");
                if ($enc_off != $p3)
                        { &mov("eax",   &wparam($p3)); &push("eax"); }
                else    { &push("ecx"); }
                }
        if ($p2 > 0)
                {
                &comment("get and push parameter $p2");
                if ($enc_off != $p2)
                        { &mov("eax",   &wparam($p2)); &push("eax"); }
                else    { &push("ecx"); }
                }
        if ($p1 > 0)
                {
                &comment("get and push parameter $p1");
                if ($enc_off != $p1)
                        { &mov("eax",   &wparam($p1)); &push("eax"); }
                else    { &push("ecx"); }
                }
        &push("ebx");           # push data/iv

        &cmp("ecx",0);
        &jz(&label("decrypt"));

        &and($count,0xfffffff8);
        &mov("eax",     &DWP($data_off,"esp","",0));    # load iv[0]
        &mov("ebx",     &DWP($data_off+4,"esp","",0));  # load iv[1]

        &jz(&label("encrypt_finish"));

        ###############################################################

        &set_label("encrypt_loop");
        # encrypt start
        # "eax" and "ebx" hold iv (or the last cipher text)

        &mov("ecx",     &DWP(0,$in,"",0));      # load first 4 bytes
        &mov("edx",     &DWP(4,$in,"",0));      # second 4 bytes

        &xor("eax",     "ecx");
        &xor("ebx",     "edx");

        &bswap("eax")   if $swap;
        &bswap("ebx")   if $swap;

        &mov(&DWP($data_off,"esp","",0),        "eax"); # put in array for call
        &mov(&DWP($data_off+4,"esp","",0),      "ebx"); #

        &call($enc_func);

        &mov("eax",     &DWP($data_off,"esp","",0));
```

```
128         &mov("ebx",     &DWP($data_off+4,"esp","",0));

130         &bswap("eax")   if $swap;
131         &bswap("ebx")   if $swap;

133         &mov(&DWP(0,$out,"",0),"eax");
134         &mov(&DWP(4,$out,"",0),"ebx");

136         # eax and ebx are the next iv.

138         &add($in,       8);
139         &add($out,      8);

141         &sub($count,    8);
142         &jnz(&label("encrypt_loop"));

144 ####################################################################3
145         &set_label("encrypt_finish");
146         &mov($count,    &wparam(2));    # length
147         &and($count,    7);
148         &jz(&label("finish"));
149         &call(&label("PIC_point"));
150 &set_label("PIC_point");
151         &blindpop("edx");
152         &lea("ecx",&DWP(&label("cbc_enc_jmp_table")."-".&label("PIC_point"),"edx
153         &mov($count,&DWP(0,"ecx",$count,4));
154         &add($count,"edx");
155         &xor("ecx","ecx");
156         &xor("edx","edx");
157         #&mov($count,&DWP(&label("cbc_enc_jmp_table"),"",$count,4));
158         &jmp_ptr($count);

160 &set_label("ej7");
161         &movb(&HB("edx"),       &BP(6,$in,"",0));
162         &shl("edx",8);
163 &set_label("ej6");
164         &movb(&HB("edx"),       &BP(5,$in,"",0));
165 &set_label("ej5");
166         &movb(&LB("edx"),       &BP(4,$in,"",0));
167 &set_label("ej4");
168         &mov("ecx",             &DWP(0,$in,"",0));
169         &jmp(&label("ejend"));
170 &set_label("ej3");
171         &movb(&HB("ecx"),       &BP(2,$in,"",0));
172         &shl("ecx",8);
173 &set_label("ej2");
174         &movb(&HB("ecx"),       &BP(1,$in,"",0));
175 &set_label("ej1");
176         &movb(&LB("ecx"),       &BP(0,$in,"",0));
177 &set_label("ejend");

179         &xor("eax",     "ecx");
180         &xor("ebx",     "edx");

182         &bswap("eax")   if $swap;
183         &bswap("ebx")   if $swap;

185         &mov(&DWP($data_off,"esp","",0),        "eax"); # put in array for call
186         &mov(&DWP($data_off+4,"esp","",0),      "ebx"); #

188         &call($enc_func);

190         &mov("eax",     &DWP($data_off,"esp","",0));
191         &mov("ebx",     &DWP($data_off+4,"esp","",0));

193         &bswap("eax")   if $swap;
```



```
194         &bswap("ebx")   if $swap;

196         &mov(&DWP(0,$out,"",0),"eax");
197         &mov(&DWP(4,$out,"",0),"ebx");

199         &jmp(&label("finish"));

201         ####################################################################
202         ####################################################################
203         &set_label("decrypt",1);
204         # decrypt start
205         &and($count,0xfffffff8);
206         # The next 2 instructions are only for if the jz is taken
207         &mov("eax",     &DWP($data_off+8,"esp","",0));  # get iv[0]
208         &mov("ebx",     &DWP($data_off+12,"esp","",0)); # get iv[1]
209         &jz(&label("decrypt_finish"));

211         &set_label("decrypt_loop");
212         &mov("eax",     &DWP(0,$in,"",0));      # load first 4 bytes
213         &mov("ebx",     &DWP(4,$in,"",0));      # second 4 bytes

215         &bswap("eax")   if $swap;
216         &bswap("ebx")   if $swap;

218         &mov(&DWP($data_off,"esp","",0),        "eax"); # put back
219         &mov(&DWP($data_off+4,"esp","",0),      "ebx"); #

221         &call($dec_func);

223         &mov("eax",     &DWP($data_off,"esp","",0));    # get return
224         &mov("ebx",     &DWP($data_off+4,"esp","",0));  #

226         &bswap("eax")   if $swap;
227         &bswap("ebx")   if $swap;

229         &mov("ecx",     &DWP($data_off+8,"esp","",0));  # get iv[0]
230         &mov("edx",     &DWP($data_off+12,"esp","",0)); # get iv[1]

232         &xor("ecx",     "eax");
233         &xor("edx",     "ebx");

235         &mov("eax",     &DWP(0,$in,"",0));      # get old cipher text,
236         &mov("ebx",     &DWP(4,$in,"",0));      # next iv actually

238         &mov(&DWP(0,$out,"",0),"ecx");
239         &mov(&DWP(4,$out,"",0),"edx");

241         &mov(&DWP($data_off+8,"esp","",0),      "eax"); # save iv
242         &mov(&DWP($data_off+12,"esp","",0),     "ebx"); #

244         &add($in,       8);
245         &add($out,      8);

247         &sub($count,    8);
248         &jnz(&label("decrypt_loop"));
249 ############################ ENDIT ######################3
250         &set_label("decrypt_finish");
251         &mov($count,    &wparam(2));    # length
252         &and($count,    7);
253         &jz(&label("finish"));

255         &mov("eax",     &DWP(0,$in,"",0));      # load first 4 bytes
256         &mov("ebx",     &DWP(4,$in,"",0));      # second 4 bytes

258         &bswap("eax")   if $swap;
259         &bswap("ebx")   if $swap;
```

```
261         &mov(&DWP($data_off,"esp","",0),        "eax"); # put back
262         &mov(&DWP($data_off+4,"esp","",0),      "ebx"); #

264         &call($dec_func);

266         &mov("eax",     &DWP($data_off,"esp","",0));    # get return
267         &mov("ebx",     &DWP($data_off+4,"esp","",0));  #

269         &bswap("eax")   if $swap;
270         &bswap("ebx")   if $swap;

272         &mov("ecx",     &DWP($data_off+8,"esp","",0));  # get iv[0]
273         &mov("edx",     &DWP($data_off+12,"esp","",0)); # get iv[1]

275         &xor("ecx",     "eax");
276         &xor("edx",     "ebx");

278         # this is for when we exit
279         &mov("eax",     &DWP(0,$in,"",0));      # get old cipher text,
280         &mov("ebx",     &DWP(4,$in,"",0));      # next iv actually

282 &set_label("dj7");
283         &rotr("edx",    16);
284         &movb(&BP(6,$out,"",0), &LB("edx"));
285         &shr("edx",16);
286 &set_label("dj6");
287         &movb(&BP(5,$out,"",0), &HB("edx"));
288 &set_label("dj5");
289         &movb(&BP(4,$out,"",0), &LB("edx"));
290 &set_label("dj4");
291         &mov(&DWP(0,$out,"",0), "ecx");
292         &jmp(&label("djend"));
293 &set_label("dj3");
294         &rotr("ecx",    16);
295         &movb(&BP(2,$out,"",0), &LB("ecx"));
296         &shl("ecx",16);
297 &set_label("dj2");
298         &movb(&BP(1,$in,"",0),  &HB("ecx"));
299 &set_label("dj1");
300         &movb(&BP(0,$in,"",0),  &LB("ecx"));
301 &set_label("djend");

303         # final iv is still in eax:ebx
304         &jmp(&label("finish"));


307 ########################## FINISH ######################3
308         &set_label("finish",1);
309         &mov("ecx",     &wparam($iv_off));      # Get iv ptr

311         ################################################
312         $total=16+4;
313         $total+=4 if ($p1 > 0);
314         $total+=4 if ($p2 > 0);
315         $total+=4 if ($p3 > 0);
316         &add("esp",$total);

318         &mov(&DWP(0,"ecx","",0),        "eax"); # save iv
319         &mov(&DWP(4,"ecx","",0),        "ebx"); # save iv

321         &function_end_A($name);

323         &align(64);
324         &set_label("cbc_enc_jmp_table");
325         &data_word("0");
```



```
326         &data_word(&label("ej1")."-".&label("PIC_point"));
327         &data_word(&label("ej2")."-".&label("PIC_point"));
328         &data_word(&label("ej3")."-".&label("PIC_point"));
329         &data_word(&label("ej4")."-".&label("PIC_point"));
330         &data_word(&label("ej5")."-".&label("PIC_point"));
331         &data_word(&label("ej6")."-".&label("PIC_point"));
332         &data_word(&label("ej7")."-".&label("PIC_point"));
333         # not used
334         #&set_label("cbc_dec_jmp_table",1);
335         #&data_word("0");
336         #&data_word(&label("dj1")."-".&label("PIC_point"));
337         #&data_word(&label("dj2")."-".&label("PIC_point"));
338         #&data_word(&label("dj3")."-".&label("PIC_point"));
339         #&data_word(&label("dj4")."-".&label("PIC_point"));
340         #&data_word(&label("dj5")."-".&label("PIC_point"));
341         #&data_word(&label("dj6")."-".&label("PIC_point"));
342         #&data_word(&label("dj7")."-".&label("PIC_point"));
343         &align(64);

345         &function_end_B($name);

347         }

349 1;
350 #endif /* ! codereview */
```

```
**********************************************************
   33057 Wed Aug 13 19:53:06 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/cmll-x86.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #!/usr/bin/env perl

   3 # ====================================================================
   4 # Copyright (c) 2008 Andy Polyakov <appro@openssl.org>
   5 #
   6 # This module may be used under the terms of either the GNU General
   7 # Public License version 2 or later, the GNU Lesser General Public
   8 # License version 2.1 or later, the Mozilla Public License version
   9 # 1.1 or the BSD License. The exact terms of either license are
  10 # distributed along with this module. For further details see
  11 # http://www.openssl.org/~appro/camellia/.
  12 # ====================================================================

  14 # Performance in cycles per processed byte (less is better) in
  15 # 'openssl speed ...' benchmark:
  16 #
  17 #                       AMD K8  Core2   PIII    P4
  18 # -evp camellia-128-ecb 21.5    22.8    27.0    28.9
  19 # + over gcc 3.4.6      +90/11% +70/10% +53/4%  +160/64%
  20 # + over icc 8.0        +48/19% +21/15% +21/17% +55/37%
  21 #
  22 # camellia-128-cbc      17.3    21.1    23.9    25.9
  23 #
  24 # 128-bit key setup     196     280     256     240     cycles/key
  25 # + over gcc 3.4.6      +30/0%  +17/11% +11/0%  +63/40%
  26 # + over icc 8.0        +18/3%  +10/0%  +10/3%  +21/10%
  27 #
  28 # Pairs of numbers in "+" rows represent performance improvement over
  29 # compiler generated position-independent code, PIC, and non-PIC
  30 # respectively. PIC results are of greater relevance, as this module
  31 # is position-independent, i.e. suitable for a shared library or PIE.
  32 # Position independence "costs" one register, which is why compilers
  33 # are so close with non-PIC results, they have an extra register to
  34 # spare. CBC results are better than ECB ones thanks to "zero-copy"
  35 # private _x86_* interface, and are ~30-40% better than with compiler
  36 # generated cmll_cbc.o, and reach ~80-90% of x86_64 performance on
  37 # same CPU (where applicable).

  39 $0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
  40 push(@INC,"${dir}","${dir}../../perlasm");
  41 require "x86asm.pl";

  43 $OPENSSL=1;

  45 &asm_init($ARGV[0],"cmll-586.pl",$ARGV[$#ARGV] eq "386");

  47 @T=("eax","ebx","ecx","edx");
  48 $idx="esi";
  49 $key="edi";
  50 $Tbl="ebp";

  52 # stack frame layout in _x86_Camellia_* routines, frame is allocated
  53 # by caller
  54 $__ra=&DWP(0,"esp");    # return address
  55 $__s0=&DWP(4,"esp");    # s0 backing store
  56 $__s1=&DWP(8,"esp");    # s1 backing store
  57 $__s2=&DWP(12,"esp");   # s2 backing store
  58 $__s3=&DWP(16,"esp");   # s3 backing store
  59 $__end=&DWP(20,"esp");  # pointer to end/start of key schedule

  61 # stack frame layout in Camellia_[en|crypt] routines, which differs from
```



```
  62 # above by 4 and overlaps by pointer to end/start of key schedule
  63 $_end=&DWP(16,"esp");
  64 $_esp=&DWP(20,"esp");

  66 # const unsigned int Camellia_SBOX[4][256];
  67 # Well, sort of... Camellia_SBOX[0][] is interleaved with [1][],
  68 # and [2][] - with [3][]. This is done to optimize code size.
  69 $SBOX1_1110=0;          # Camellia_SBOX[0]
  70 $SBOX4_4404=4;          # Camellia_SBOX[1]
  71 $SBOX2_0222=2048;       # Camellia_SBOX[2]
  72 $SBOX3_3033=2052;       # Camellia_SBOX[3]
  73 &static_label("Camellia_SIGMA");
  74 &static_label("Camellia_SBOX");

  76 sub Camellia_Feistel {
  77 my $i=@_[0];
  78 my $seed=defined(@_[1])?@_[1]:0;
  79 my $scale=$seed<0?-8:8;
  80 my $frame=defined(@_[2])?@_[2]:0;
  81 my $j=($i&1)*2;
  82 my $t0=@T[($j)%4],$t1=@T[($j+1)%4],$t2=@T[($j+2)%4],$t3=@T[($j+3)%4];

  84         &xor    ($t0,$idx);                             # t0^=key[0]
  85         &xor    ($t1,&DWP($seed+$i*$scale+4,$key));     # t1^=key[1]
  86         &movz   ($idx,&HB($t0));                        # (t0>>8)&0xff
  87         &mov    ($t3,&DWP($SBOX3_3033,$Tbl,$idx,8));    # t3=SBOX3_3033[0]
  88         &movz   ($idx,&LB($t0));                        # (t0>>0)&0xff
  89         &xor    ($t3,&DWP($SBOX4_4404,$Tbl,$idx,8));    # t3^=SBOX4_4404[0]
  90         &shr    ($t0,16);
  91         &movz   ($idx,&LB($t1));                        # (t1>>0)&0xff
  92         &mov    ($t2,&DWP($SBOX1_1110,$Tbl,$idx,8));    # t2=SBOX1_1110[1]
  93         &movz   ($idx,&HB($t0));                        # (t0>>24)&0xff
  94         &xor    ($t3,&DWP($SBOX1_1110,$Tbl,$idx,8));    # t3^=SBOX1_1110[0]
  95         &movz   ($idx,&HB($t1));                        # (t1>>8)&0xff
  96         &xor    ($t2,&DWP($SBOX4_4404,$Tbl,$idx,8));    # t2^=SBOX4_4404[1]
  97         &shr    ($t1,16);
  98         &movz   ($t0,&LB($t0));                         # (t0>>16)&0xff
  99         &xor    ($t3,&DWP($SBOX2_0222,$Tbl,$t0,8));     # t3^=SBOX2_0222[0]
 100         &movz   ($idx,&HB($t1));                        # (t1>>24)&0xff
 101         &mov    ($t0,&DWP($frame+4*(($j+3)%4),"esp"));  # prefetch "s3"
 102         &xor    ($t2,$t3);                              # t2^=t3
 103         &rotr   ($t3,8);                                # t3=RightRotate(t3,8)
 104         &xor    ($t2,&DWP($SBOX2_0222,$Tbl,$idx,8));    # t2^=SBOX2_0222[1]
 105         &movz   ($idx,&LB($t1));                        # (t1>>16)&0xff
 106         &mov    ($t1,&DWP($frame+4*(($j+2)%4),"esp"));  # prefetch "s2"
 107         &xor    ($t3,$t0);                              # t3^=s3
 108         &xor    ($t2,&DWP($SBOX3_3033,$Tbl,$idx,8));    # t2^=SBOX3_3033[1]
 109         &mov    ($idx,&DWP($seed+($i+1)*$scale,$key));  # prefetch key[i+1]
 110         &xor    ($t3,$t2);                              # t3^=t2
 111         &mov    (&DWP($frame+4*(($j+3)%4),"esp"),$t3);  # s3=t3
 112         &xor    ($t2,$t1);                              # t2^=s2
 113         &mov    (&DWP($frame+4*(($j+2)%4),"esp"),$t2);  # s2=t2
 114 }

 116 # void Camellia_EncryptBlock_Rounds(
 117 #               int grandRounds,
 118 #               const Byte plaintext[],
 119 #               const KEY_TABLE_TYPE keyTable,
 120 #               Byte ciphertext[])
 121 &function_begin("Camellia_EncryptBlock_Rounds");
 122         &mov    ("eax",&wparam(0));     # load grandRounds
 123         &mov    ($idx,&wparam(1));      # load plaintext pointer
 124         &mov    ($key,&wparam(2));      # load key schedule pointer

 126         &mov    ("ebx","esp");
 127         &sub    ("esp",7*4);            # place for s[0-3],keyEnd,esp and ra
```

```
128         &and    ("esp",-64);
129
130         # place stack frame just "above mod 1024" the key schedule
131         # this ensures that cache associativity of 2 suffices
132         &lea    ("ecx",&DWP(-64-63,$key));
133         &sub    ("ecx","esp");
134         &neg    ("ecx");
135         &and    ("ecx",0x3C0);  # modulo 1024, but aligned to cache-line
136         &sub    ("esp","ecx");
137         &add    ("esp",4);      # 4 is reserved for callee's return address
138
139         &shl    ("eax",6);
140         &lea    ("eax",&DWP(0,$key,"eax"));
141         &mov    ($_esp,"ebx");  # save %esp
142         &mov    ($_end,"eax");  # save keyEnd
143
144         &call   (&label("pic_point"));
145         &set_label("pic_point");
146         &blindpop($Tbl);
147         &lea    ($Tbl,&DWP(&label("Camellia_SBOX")."-".&label("pic_point"),$Tbl));
148
149         &mov    (@T[0],&DWP(0,$idx));   # load plaintext
150         &mov    (@T[1],&DWP(4,$idx));
151         &mov    (@T[2],&DWP(8,$idx));
152         &bswap  (@T[0]);
153         &mov    (@T[3],&DWP(12,$idx));
154         &bswap  (@T[1]);
155         &bswap  (@T[2]);
156         &bswap  (@T[3]);
157
158         &call   ("_x86_Camellia_encrypt");
159
160         &mov    ("esp",$_esp);
161         &bswap  (@T[0]);
162         &mov    ($idx,&wparam(3));      # load ciphertext pointer
163         &bswap  (@T[1]);
164         &bswap  (@T[2]);
165         &bswap  (@T[3]);
166         &mov    (&DWP(0,$idx),@T[0]);   # write ciphertext
167         &mov    (&DWP(4,$idx),@T[1]);
168         &mov    (&DWP(8,$idx),@T[2]);
169         &mov    (&DWP(12,$idx),@T[3]);
170 &function_end("Camellia_EncryptBlock_Rounds");
171 # V1.x API
172 &function_begin_B("Camellia_EncryptBlock");
173         &mov    ("eax",128);
174         &sub    ("eax",&wparam(0));     # load keyBitLength
175         &mov    ("eax",3);
176         &adc    ("eax",0);              # keyBitLength==128?3:4
177         &mov    (&wparam(0),"eax");
178         &jmp    (&label("Camellia_EncryptBlock_Rounds"));
179 &function_end_B("Camellia_EncryptBlock");
180
181 if ($OPENSSL) {
182 # void Camellia_encrypt(
183 #               const unsigned char *in,
184 #               unsigned char *out,
185 #               const CAMELLIA_KEY *key)
186 &function_begin("Camellia_encrypt");
187         &mov    ($idx,&wparam(0));      # load plaintext pointer
188         &mov    ($key,&wparam(2));      # load key schedule pointer
189
190         &mov    ("ebx","esp");
191         &sub    ("esp",7*4);            # place for s[0-3],keyEnd,esp and ra
192         &and    ("esp",-64);
193         &mov    ("eax",&DWP(272,$key)); # load grandRounds counter
```



```
194
195         # place stack frame just "above mod 1024" the key schedule
196         # this ensures that cache associativity of 2 suffices
197         &lea    ("ecx",&DWP(-64-63,$key));
198         &sub    ("ecx","esp");
199         &neg    ("ecx");
200         &and    ("ecx",0x3C0);  # modulo 1024, but aligned to cache-line
201         &sub    ("esp","ecx");
202         &add    ("esp",4);      # 4 is reserved for callee's return address
203
204         &shl    ("eax",6);
205         &lea    ("eax",&DWP(0,$key,"eax"));
206         &mov    ($_esp,"ebx");  # save %esp
207         &mov    ($_end,"eax");  # save keyEnd
208
209         &call   (&label("pic_point"));
210         &set_label("pic_point");
211         &blindpop($Tbl);
212         &lea    ($Tbl,&DWP(&label("Camellia_SBOX")."-".&label("pic_point"),$Tbl));
213
214         &mov    (@T[0],&DWP(0,$idx));   # load plaintext
215         &mov    (@T[1],&DWP(4,$idx));
216         &mov    (@T[2],&DWP(8,$idx));
217         &bswap  (@T[0]);
218         &mov    (@T[3],&DWP(12,$idx));
219         &bswap  (@T[1]);
220         &bswap  (@T[2]);
221         &bswap  (@T[3]);
222
223         &call   ("_x86_Camellia_encrypt");
224
225         &mov    ("esp",$_esp);
226         &bswap  (@T[0]);
227         &mov    ($idx,&wparam(1));      # load ciphertext pointer
228         &bswap  (@T[1]);
229         &bswap  (@T[2]);
230         &bswap  (@T[3]);
231         &mov    (&DWP(0,$idx),@T[0]);   # write ciphertext
232         &mov    (&DWP(4,$idx),@T[1]);
233         &mov    (&DWP(8,$idx),@T[2]);
234         &mov    (&DWP(12,$idx),@T[3]);
235 &function_end("Camellia_encrypt");
236 }
237
238 &function_begin_B("_x86_Camellia_encrypt");
239         &xor    (@T[0],&DWP(0,$key));   # ^=key[0-3]
240         &xor    (@T[1],&DWP(4,$key));
241         &xor    (@T[2],&DWP(8,$key));
242         &xor    (@T[3],&DWP(12,$key));
243         &mov    ($idx,&DWP(16,$key));   # prefetch key[4]
244
245         &mov    ($__s0,@T[0]);          # save s[0-3]
246         &mov    ($__s1,@T[1]);
247         &mov    ($__s2,@T[2]);
248         &mov    ($__s3,@T[3]);
249
250 &set_label("loop",16);
251         for ($i=0;$i<6;$i++) { Camellia_Feistel($i,16,4); }
252
253         &add    ($key,16*4);
254         &cmp    ($key,$__end);
255         &je     (&label("done"));
256
257         # @T[0-1] are preloaded, $idx is preloaded with key[0]
258         &and    ($idx,@T[0]);
259          &mov    (@T[3],$__s3);
```

```
260         &rotl   ($idx,1);
261          &mov    (@T[2],@T[3]);
262         &xor    (@T[1],$idx);
263          &or     (@T[2],&DWP(12,$key));
264         &mov    ($__s1,@T[1]);          # s1^=LeftRotate(s0&key[0],1);
265          &xor    (@T[2],$__s2);

267         &mov    ($idx,&DWP(4,$key));
268          &mov    ($__s2,@T[2]);         # s2^=s3|key[3];
269         &or     ($idx,@T[1]);
270          &and    (@T[2],&DWP(8,$key));
271         &xor    (@T[0],$idx);
272          &rotl   (@T[2],1);
273         &mov    ($__s0,@T[0]);          # s0^=s1|key[1];
274          &xor    (@T[3],@T[2]);
275         &mov    ($idx,&DWP(16,$key));           # prefetch key[4]
276          &mov    ($__s3,@T[3]);         # s3^=LeftRotate(s2&key[2],1);
277         &jmp    (&label("loop"));

279 &set_label("done",8);
280         &mov    (@T[2],@T[0]);          # SwapHalf
281         &mov    (@T[3],@T[1]);
282         &mov    (@T[0],$__s2);
283         &mov    (@T[1],$__s3);
284         &xor    (@T[0],$idx);           # $idx is preloaded with key[0]
285         &xor    (@T[1],&DWP(4,$key));
286         &xor    (@T[2],&DWP(8,$key));
287         &xor    (@T[3],&DWP(12,$key));
288         &ret    ();
289 &function_end_B("_x86_Camellia_encrypt");

291 # void Camellia_DecryptBlock_Rounds(
292 #               int grandRounds,
293 #               const Byte ciphertext[],
294 #               const KEY_TABLE_TYPE keyTable,
295 #               Byte plaintext[])
296 &function_begin("Camellia_DecryptBlock_Rounds");
297         &mov    ("eax",&wparam(0));     # load grandRounds
298         &mov    ($idx,&wparam(1));      # load ciphertext pointer
299         &mov    ($key,&wparam(2));      # load key schedule pointer

301         &mov    ("ebx","esp");
302         &sub    ("esp",7*4);            # place for s[0-3],keyEnd,esp and ra
303         &and    ("esp",-64);

305         # place stack frame just "above mod 1024" the key schedule
306         # this ensures that cache associativity of 2 suffices
307         &lea    ("ecx",&DWP(-64-63,$key));
308         &sub    ("ecx","esp");
309         &neg    ("ecx");
310         &and    ("ecx",0x3C0);  # modulo 1024, but aligned to cache-line
311         &sub    ("esp","ecx");
312         &add    ("esp",4);      # 4 is reserved for callee's return address

314         &shl    ("eax",6);
315         &mov    (&DWP(4*4,"esp"),$key); # save keyStart
316         &lea    ($key,&DWP(0,$key,"eax"));
317         &mov    (&DWP(5*4,"esp"),"ebx");# save %esp

319         &call   (&label("pic_point"));
320         &set_label("pic_point");
321         &blindpop($Tbl);
322         &lea    ($Tbl,&DWP(&label("Camellia_SBOX")."-".&label("pic_point"),$Tbl)

324         &mov    (@T[0],&DWP(0,$idx));   # load ciphertext
325         &mov    (@T[1],&DWP(4,$idx));
```



```
326         &mov    (@T[2],&DWP(8,$idx));
327         &bswap  (@T[0]);
328         &mov    (@T[3],&DWP(12,$idx));
329         &bswap  (@T[1]);
330         &bswap  (@T[2]);
331         &bswap  (@T[3]);

333         &call   ("_x86_Camellia_decrypt");

335         &mov    ("esp",&DWP(5*4,"esp"));
336         &bswap  (@T[0]);
337         &mov    ($idx,&wparam(3));      # load plaintext pointer
338         &bswap  (@T[1]);
339         &bswap  (@T[2]);
340         &bswap  (@T[3]);
341         &mov    (&DWP(0,$idx),@T[0]);   # write plaintext
342         &mov    (&DWP(4,$idx),@T[1]);
343         &mov    (&DWP(8,$idx),@T[2]);
344         &mov    (&DWP(12,$idx),@T[3]);
345 &function_end("Camellia_DecryptBlock_Rounds");
346 # V1.x API
347 &function_begin_B("Camellia_DecryptBlock");
348         &mov    ("eax",128);
349         &sub    ("eax",&wparam(0));     # load keyBitLength
350         &mov    ("eax",3);
351         &adc    ("eax",0);              # keyBitLength==128?3:4
352         &mov    (&wparam(0),"eax");
353         &jmp    (&label("Camellia_DecryptBlock_Rounds"));
354 &function_end_B("Camellia_DecryptBlock");

356 if ($OPENSSL) {
357 # void Camellia_decrypt(
358 #               const unsigned char *in,
359 #               unsigned char *out,
360 #               const CAMELLIA_KEY *key)
361 &function_begin("Camellia_decrypt");
362         &mov    ($idx,&wparam(0));      # load ciphertext pointer
363         &mov    ($key,&wparam(2));      # load key schedule pointer

365         &mov    ("ebx","esp");
366         &sub    ("esp",7*4);            # place for s[0-3],keyEnd,esp and ra
367         &and    ("esp",-64);
368         &mov    ("eax",&DWP(272,$key)); # load grandRounds counter

370         # place stack frame just "above mod 1024" the key schedule
371         # this ensures that cache associativity of 2 suffices
372         &lea    ("ecx",&DWP(-64-63,$key));
373         &sub    ("ecx","esp");
374         &neg    ("ecx");
375         &and    ("ecx",0x3C0);  # modulo 1024, but aligned to cache-line
376         &sub    ("esp","ecx");
377         &add    ("esp",4);      # 4 is reserved for callee's return address

379         &shl    ("eax",6);
380         &mov    (&DWP(4*4,"esp"),$key); # save keyStart
381         &lea    ($key,&DWP(0,$key,"eax"));
382         &mov    (&DWP(5*4,"esp"),"ebx");# save %esp

384         &call   (&label("pic_point"));
385         &set_label("pic_point");
386         &blindpop($Tbl);
387         &lea    ($Tbl,&DWP(&label("Camellia_SBOX")."-".&label("pic_point"),$Tbl)

389         &mov    (@T[0],&DWP(0,$idx));   # load ciphertext
390         &mov    (@T[1],&DWP(4,$idx));
391         &mov    (@T[2],&DWP(8,$idx));
```

```
392         &bswap  (@T[0]);
393         &mov    (@T[3],&DWP(12,$idx));
394         &bswap  (@T[1]);
395         &bswap  (@T[2]);
396         &bswap  (@T[3]);

398         &call   ("_x86_Camellia_decrypt");

400         &mov    ("esp",&DWP(5*4,"esp"));
401         &bswap  (@T[0]);
402         &mov    ($idx,&wparam(1));      # load plaintext pointer
403         &bswap  (@T[1]);
404         &bswap  (@T[2]);
405         &bswap  (@T[3]);
406         &mov    (&DWP(0,$idx),@T[0]);   # write plaintext
407         &mov    (&DWP(4,$idx),@T[1]);
408         &mov    (&DWP(8,$idx),@T[2]);
409         &mov    (&DWP(12,$idx),@T[3]);
410 &function_end("Camellia_decrypt");
411 }

413 &function_begin_B("_x86_Camellia_decrypt");
414         &xor    (@T[0],&DWP(0,$key));   # ^=key[0-3]
415         &xor    (@T[1],&DWP(4,$key));
416         &xor    (@T[2],&DWP(8,$key));
417         &xor    (@T[3],&DWP(12,$key));
418         &mov    ($idx,&DWP(-8,$key));   # prefetch key[-2]

420         &mov    ($__s0,@T[0]);          # save s[0-3]
421         &mov    ($__s1,@T[1]);
422         &mov    ($__s2,@T[2]);
423         &mov    ($__s3,@T[3]);

425 &set_label("loop",16);
426         for ($i=0;$i<6;$i++) { Camellia_Feistel($i,-8,4); }

428         &sub    ($key,16*4);
429         &cmp    ($key,$__end);
430         &je     (&label("done"));

432         # @T[0-1] are preloaded, $idx is preloaded with key[2]
433         &and    ($idx,@T[0]);
434          &mov    (@T[3],$__s3);
435         &rotl   ($idx,1);
436          &mov    (@T[2],@T[3]);
437         &xor    (@T[1],$idx);
438          &or     (@T[2],&DWP(4,$key));
439         &mov    ($__s1,@T[1]);          # s1^=LeftRotate(s0&key[0],1);
440          &xor    (@T[2],$__s2);

442         &mov    ($idx,&DWP(12,$key));
443          &mov    ($__s2,@T[2]);         # s2^=s3|key[3];
444         &or     ($idx,@T[1]);
445          &and    (@T[2],&DWP(0,$key));
446         &xor    (@T[0],$idx);
447          &rotl   (@T[2],1);
448         &mov    ($__s0,@T[0]);          # s0^=s1|key[1];
449          &xor    (@T[3],@T[2]);
450         &mov    ($idx,&DWP(-8,$key));   # prefetch key[4]
451          &mov    ($__s3,@T[3]);         # s3^=LeftRotate(s2&key[2],1);
452         &jmp    (&label("loop"));

454 &set_label("done",8);
455         &mov    (@T[2],@T[0]);          # SwapHalf
456         &mov    (@T[3],@T[1]);
457         &mov    (@T[0],$__s2);
```



```
458         &mov    (@T[1],$__s3);
459         &xor    (@T[2],$idx);           # $idx is preloaded with key[2]
460         &xor    (@T[3],&DWP(12,$key));
461         &xor    (@T[0],&DWP(0,$key));
462         &xor    (@T[1],&DWP(4,$key));
463         &ret    ();
464 &function_end_B("_x86_Camellia_decrypt");

466 # shld is very slow on Intel P4 family. Even on AMD it limits
467 # instruction decode rate [because it's VectorPath] and consequently
468 # performance. PIII, PM and Core[2] seem to be the only ones which
469 # execute this code ~7% faster...
470 sub __rotl128 {
471   my ($i0,$i1,$i2,$i3,$rot,$rnd,@T)=@_;

473     $rnd *= 2;
474     if ($rot) {
475         &mov    ($idx,$i0);
476         &shld   ($i0,$i1,$rot);
477         &shld   ($i1,$i2,$rot);
478         &shld   ($i2,$i3,$rot);
479         &shld   ($i3,$idx,$rot);
480     }
481     &mov        (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i0 eq @T[0]);
482     &mov        (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i1 eq @T[0]);
483     &mov        (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i2 eq @T[0]);
484     &mov        (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i3 eq @T[0]);
485 }

487 # ... Implementing 128-bit rotate without shld gives >3x performance
488 # improvement on P4, only ~7% degradation on other Intel CPUs and
489 # not worse performance on AMD. This is therefore preferred.
490 sub _rotl128 {
491   my ($i0,$i1,$i2,$i3,$rot,$rnd,@T)=@_;

493     $rnd *= 2;
494     if ($rot) {
495         &mov    ($Tbl,$i0);
496         &shl    ($i0,$rot);
497         &mov    ($idx,$i1);
498         &shr    ($idx,32-$rot);
499         &shl    ($i1,$rot);
500         &or     ($i0,$idx);
501         &mov    ($idx,$i2);
502         &shl    ($i2,$rot);
503         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i0 eq @T[0]);
504         &shr    ($idx,32-$rot);
505         &or     ($i1,$idx);
506         &shr    ($Tbl,32-$rot);
507         &mov    ($idx,$i3);
508         &shr    ($idx,32-$rot);
509         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i1 eq @T[0]);
510         &shl    ($i3,$rot);
511         &or     ($i2,$idx);
512         &or     ($i3,$Tbl);
513         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i2 eq @T[0]);
514         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i3 eq @T[0]);
515     } else {
516         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i0 eq @T[0]);
517         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i1 eq @T[0]);
518         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i2 eq @T[0]);
519         &mov    (&DWP(-128+4*$rnd++,$key),shift(@T))    if ($i3 eq @T[0]);
520     }
521 }

523 sub _saveround {
```

```
524 my ($rnd,$key,@T)=@_;
525 my $bias=int(@T[0])?shift(@T):0;

527         &mov    (&DWP($bias+$rnd*8+0,$key),@T[0]);
528         &mov    (&DWP($bias+$rnd*8+4,$key),@T[1])       if ($#T>=1);
529         &mov    (&DWP($bias+$rnd*8+8,$key),@T[2])       if ($#T>=2);
530         &mov    (&DWP($bias+$rnd*8+12,$key),@T[3])      if ($#T>=3);
531 }

533 sub _loadround {
534 my ($rnd,$key,@T)=@_;
535 my $bias=int(@T[0])?shift(@T):0;

537         &mov    (@T[0],&DWP($bias+$rnd*8+0,$key));
538         &mov    (@T[1],&DWP($bias+$rnd*8+4,$key))       if ($#T>=1);
539         &mov    (@T[2],&DWP($bias+$rnd*8+8,$key))       if ($#T>=2);
540         &mov    (@T[3],&DWP($bias+$rnd*8+12,$key))      if ($#T>=3);
541 }

543 # void Camellia_Ekeygen(
544 #               const int keyBitLength,
545 #               const Byte *rawKey,
546 #               KEY_TABLE_TYPE keyTable)
547 &function_begin("Camellia_Ekeygen");
548 { my $step=0;

550         &stack_push(4);                         # place for s[0-3]

552         &mov    ($Tbl,&wparam(0));              # load arguments
553         &mov    ($idx,&wparam(1));
554         &mov    ($key,&wparam(2));

556         &mov    (@T[0],&DWP(0,$idx));           # load 0-127 bits
557         &mov    (@T[1],&DWP(4,$idx));
558         &mov    (@T[2],&DWP(8,$idx));
559         &mov    (@T[3],&DWP(12,$idx));

561         &bswap  (@T[0]);
562         &bswap  (@T[1]);
563         &bswap  (@T[2]);
564         &bswap  (@T[3]);

566         &_saveround     (0,$key,@T);            # KL<<<0

568         &cmp    ($Tbl,128);
569         &je     (&label("1st128"));

571         &mov    (@T[0],&DWP(16,$idx));          # load 128-191 bits
572         &mov    (@T[1],&DWP(20,$idx));
573         &cmp    ($Tbl,192);
574         &je     (&label("1st192"));
575         &mov    (@T[2],&DWP(24,$idx));          # load 192-255 bits
576         &mov    (@T[3],&DWP(28,$idx));
577         &jmp    (&label("1st256"));
578 &set_label("1st192",4);
579         &mov    (@T[2],@T[0]);
580         &mov    (@T[3],@T[1]);
581         &not    (@T[2]);
582         &not    (@T[3]);
583 &set_label("1st256",4);
584         &bswap  (@T[0]);
585         &bswap  (@T[1]);
586         &bswap  (@T[2]);
587         &bswap  (@T[3]);

589         &_saveround     (4,$key,@T);            # temporary storage for KR!
```



```
591         &xor    (@T[0],&DWP(0*8+0,$key));       # KR^KL
592         &xor    (@T[1],&DWP(0*8+4,$key));
593         &xor    (@T[2],&DWP(1*8+0,$key));
594         &xor    (@T[3],&DWP(1*8+4,$key));

596 &set_label("1st128",4);
597         &call   (&label("pic_point"));
598         &set_label("pic_point");
599         &blindpop($Tbl);
600         &lea    ($Tbl,&DWP(&label("Camellia_SBOX")."-".&label("pic_point"),$Tbl)
601         &lea    ($key,&DWP(&label("Camellia_SIGMA")."-".&label("Camellia_SBOX"),

603         &mov    ($idx,&DWP($step*8,$key));      # prefetch SIGMA[0]
604         &mov    (&swtmp(0),@T[0]);              # save s[0-3]
605         &mov    (&swtmp(1),@T[1]);
606         &mov    (&swtmp(2),@T[2]);
607         &mov    (&swtmp(3),@T[3]);
608         &Camellia_Feistel($step++);
609         &Camellia_Feistel($step++);
610         &mov    (@T[2],&swtmp(2));
611         &mov    (@T[3],&swtmp(3));

613         &mov    ($idx,&wparam(2));
614         &xor    (@T[0],&DWP(0*8+0,$idx));       # ^KL
615         &xor    (@T[1],&DWP(0*8+4,$idx));
616         &xor    (@T[2],&DWP(1*8+0,$idx));
617         &xor    (@T[3],&DWP(1*8+4,$idx));

619         &mov    ($idx,&DWP($step*8,$key));      # prefetch SIGMA[4]
620         &mov    (&swtmp(0),@T[0]);              # save s[0-3]
621         &mov    (&swtmp(1),@T[1]);
622         &mov    (&swtmp(2),@T[2]);
623         &mov    (&swtmp(3),@T[3]);
624         &Camellia_Feistel($step++);
625         &Camellia_Feistel($step++);
626         &mov    (@T[2],&swtmp(2));
627         &mov    (@T[3],&swtmp(3));

629         &mov    ($idx,&wparam(0));
630         &cmp    ($idx,128);
631         &jne    (&label("2nd256"));

633         &mov    ($key,&wparam(2));
634         &lea    ($key,&DWP(128,$key));          # size optimization

636         ####### process KA
637         &_saveround     (2,$key,-128,@T);       # KA<<<0
638         &_rotl128       (@T,15,6,@T);           # KA<<<15
639         &_rotl128       (@T,15,8,@T);           # KA<<<(15+15=30)
640         &_rotl128       (@T,15,12,@T[0],@T[1]); # KA<<<(30+15=45)
641         &_rotl128       (@T,15,14,@T);          # KA<<<(45+15=60)
642         push            (@T,shift(@T));         # rotl128(@T,32);
643         &_rotl128       (@T,2,20,@T);           # KA<<<(60+32+2=94)
644         &_rotl128       (@T,17,24,@T);          # KA<<<(94+17=111)

646         ####### process KL
647         &_loadround     (0,$key,-128,@T);       # load KL
648         &_rotl128       (@T,15,4,@T);           # KL<<<15
649         &_rotl128       (@T,30,10,@T);          # KL<<<(15+30=45)
650         &_rotl128       (@T,15,13,@T[2],@T[3]); # KL<<<(45+15=60)
651         &_rotl128       (@T,17,16,@T);          # KL<<<(60+17=77)
652         &_rotl128       (@T,17,18,@T);          # KL<<<(77+17=94)
653         &_rotl128       (@T,17,22,@T);          # KL<<<(94+17=111)

655         while (@T[0] ne "eax")                  # restore order
```

```
656         {   unshift     (@T,pop(@T));   }

658         &mov    ("eax",3);                      # 3 grandRounds
659         &jmp    (&label("done"));

661 &set_label("2nd256",16);
662         &mov    ($idx,&wparam(2));
663         &_saveround     (6,$idx,@T);            # temporary storage for KA!

665         &xor    (@T[0],&DWP(4*8+0,$idx));       # KA^KR
666         &xor    (@T[1],&DWP(4*8+4,$idx));
667         &xor    (@T[2],&DWP(5*8+0,$idx));
668         &xor    (@T[3],&DWP(5*8+4,$idx));

670         &mov    ($idx,&DWP($step*8,$key));      # prefetch SIGMA[8]
671         &mov    (&swtmp(0),@T[0]);              # save s[0-3]
672         &mov    (&swtmp(1),@T[1]);
673         &mov    (&swtmp(2),@T[2]);
674         &mov    (&swtmp(3),@T[3]);
675         &Camellia_Feistel($step++);
676         &Camellia_Feistel($step++);
677         &mov    (@T[2],&swtmp(2));
678         &mov    (@T[3],&swtmp(3));

680         &mov    ($key,&wparam(2));
681         &lea    ($key,&DWP(128,$key));          # size optimization

683         ####### process KB
684         &_saveround     (2,$key,-128,@T);       # KB<<<0
685         &_rotl128       (@T,30,10,@T);          # KB<<<30
686         &_rotl128       (@T,30,20,@T);          # KB<<<(30+30=60)
687         push            (@T,shift(@T));         # rotl128(@T,32);
688         &_rotl128       (@T,19,32,@T);          # KB<<<(60+32+19=111)

690         ####### process KR
691         &_loadround     (4,$key,-128,@T);       # load KR
692         &_rotl128       (@T,15,4,@T);           # KR<<<15
693         &_rotl128       (@T,15,8,@T);           # KR<<<(15+15=30)
694         &_rotl128       (@T,30,18,@T);          # KR<<<(30+30=60)
695         push            (@T,shift(@T));         # rotl128(@T,32);
696         &_rotl128       (@T,2,26,@T);           # KR<<<(60+32+2=94)

698         ####### process KA
699         &_loadround     (6,$key,-128,@T);       # load KA
700         &_rotl128       (@T,15,6,@T);           # KA<<<15
701         &_rotl128       (@T,30,14,@T);          # KA<<<(15+30=45)
702         push            (@T,shift(@T));         # rotl128(@T,32);
703         &_rotl128       (@T,0,24,@T);           # KA<<<(45+32+0=77)
704         &_rotl128       (@T,17,28,@T);          # KA<<<(77+17=94)

706         ####### process KL
707         &_loadround     (0,$key,-128,@T);       # load KL
708         push            (@T,shift(@T));         # rotl128(@T,32);
709         &_rotl128       (@T,13,12,@T);          # KL<<<(32+13=45)
710         &_rotl128       (@T,15,16,@T);          # KL<<<(45+15=60)
711         &_rotl128       (@T,17,22,@T);          # KL<<<(60+17=77)
712         push            (@T,shift(@T));         # rotl128(@T,32);
713         &_rotl128       (@T,2,30,@T);           # KL<<<(77+32+2=111)

715         while (@T[0] ne "eax")                  # restore order
716         {   unshift     (@T,pop(@T));   }

718         &mov    ("eax",4);                      # 4 grandRounds
719 &set_label("done");
720         &lea    ("edx",&DWP(272-128,$key));     # end of key schedule
721         &stack_pop(4);
```



```
722 }
723 &function_end("Camellia_Ekeygen");

725 if ($OPENSSL) {
726 # int private_Camellia_set_key (
727 #               const unsigned char *userKey,
728 #               int bits,
729 #               CAMELLIA_KEY *key)
730 &function_begin_B("private_Camellia_set_key");
731         &push   ("ebx");
732         &mov    ("ecx",&wparam(0));     # pull arguments
733         &mov    ("ebx",&wparam(1));
734         &mov    ("edx",&wparam(2));

736         &mov    ("eax",-1);
737         &test   ("ecx","ecx");
738         &jz     (&label("done"));       # userKey==NULL?
739         &test   ("edx","edx");
740         &jz     (&label("done"));       # key==NULL?

742         &mov    ("eax",-2);
743         &cmp    ("ebx",256);
744         &je     (&label("arg_ok"));     # bits==256?
745         &cmp    ("ebx",192);
746         &je     (&label("arg_ok"));     # bits==192?
747         &cmp    ("ebx",128);
748         &jne    (&label("done"));       # bits!=128?
749 &set_label("arg_ok",4);

751         &push   ("edx");                # push arguments
752         &push   ("ecx");
753         &push   ("ebx");
754         &call   ("Camellia_Ekeygen");
755         &stack_pop(3);

757         # eax holds grandRounds and edx points at where to put it
758         &mov    (&DWP(0,"edx"),"eax");
759         &xor    ("eax","eax");
760 &set_label("done",4);
761         &pop    ("ebx");
762         &ret    ();
763 &function_end_B("private_Camellia_set_key");
764 }

766 @SBOX=(
767 112,130, 44,236,179, 39,192,229,228,133, 87, 53,234, 12,174, 65,
768  35,239,107,147, 69, 25,165, 33,237, 14, 79, 78, 29,101,146,189,
769 134,184,175,143,124,235, 31,206, 62, 48,220, 95, 94,197, 11, 26,
770 166,225, 57,202,213, 71, 93, 61,217,  1, 90,214, 81, 86,108, 77,
771 139, 13,154,102,251,204,176, 45,116, 18, 43, 32,240,177,132,153,
772 223, 76,203,194, 52,126,118,  5,109,183,169, 49,209, 23,  4,215,
773  20, 88, 58, 97,222, 27, 17, 28, 50, 15,156, 22, 83, 24,242, 34,
774 254, 68,207,178,195,181,122,145, 36,  8,232,168, 96,252,105, 80,
775 170,208,160,125,161,137, 98,151, 84, 91, 30,149,224,255,100,210,
776  16,196,  0, 72,163,247,117,219,138,  3,230,218,  9, 63,221,148,
777 135, 92,131,  2,205, 74,144, 51,115,103,246,243,157,127,191,226,
778  82,155,216, 38,200, 55,198, 59,129,150,111, 75, 19,190, 99, 46,
779 233,121,167,140,159,110,188,142, 41,245,249,182, 47,253,180, 89,
780 120,152,  6,106,231, 70,113,186,212, 37,171, 66,136,162,141,250,
781 114,  7,185, 85,248,238,172, 10, 54, 73, 42,104, 60, 56,241,164,
782  64, 40,211,123,187,201, 67,193, 21,227,173,244,119,199,128,158);

784 sub S1110 { my $i=shift; $i=@SBOX[$i]; return $i<<24|$i<<16|$i<<8; }
785 sub S4404 { my $i=shift; $i=($i<<1|$i>>7)&0xff; $i=@SBOX[$i]; return $i<<24|$i<<
786 sub S0222 { my $i=shift; $i=@SBOX[$i]; $i=($i<<1|$i>>7)&0xff; return $i<<16|$i<<
787 sub S3033 { my $i=shift; $i=@SBOX[$i]; $i=($i>>1|$i<<7)&0xff; return $i<<24|$i<<
```

```
789 &set_label("Camellia_SIGMA",64);
790 &data_word(
791     0xa09e667f, 0x3bcc908b, 0xb67ae858, 0x4caa73b2,
792     0xc6ef372f, 0xe94f82be, 0x54ff53a5, 0xf1d36f1c,
793     0x10e527fa, 0xde682d1d, 0xb05688c2, 0xb3e6c1fd,
794     0,          0,          0,          0);
795 &set_label("Camellia_SBOX",64);
796 # tables are interleaved, remember?
797 for ($i=0;$i<256;$i++) { &data_word(&S1110($i),&S4404($i)); }
798 for ($i=0;$i<256;$i++) { &data_word(&S0222($i),&S3033($i)); }

800 # void Camellia_cbc_encrypt (const void char *inp, unsigned char *out,
801 #                       size_t length, const CAMELLIA_KEY *key,
802 #                       unsigned char *ivp,const int enc);
803 {
804 # stack frame layout
805 #             -4(%esp)          # return address         0(%esp)
806 #              0(%esp)          # s0                     4(%esp)
807 #              4(%esp)          # s1                     8(%esp)
808 #              8(%esp)          # s2                    12(%esp)
809 #             12(%esp)          # s3                    16(%esp)
810 #             16(%esp)          # end of key schedule   20(%esp)
811 #             20(%esp)          # %esp backup
812 my $_inp=&DWP(24,"esp");        #copy of wparam(0)
813 my $_out=&DWP(28,"esp");        #copy of wparam(1)
814 my $_len=&DWP(32,"esp");        #copy of wparam(2)
815 my $_key=&DWP(36,"esp");        #copy of wparam(3)
816 my $_ivp=&DWP(40,"esp");        #copy of wparam(4)
817 my $ivec=&DWP(44,"esp");        #ivec[16]
818 my $_tmp=&DWP(44,"esp");        #volatile variable [yes, aliases with ivec]
819 my ($s0,$s1,$s2,$s3) = @T;

821 &function_begin("Camellia_cbc_encrypt");
822         &mov    ($s2 eq "ecx"? $s2 : "",&wparam(2));    # load len
823         &cmp    ($s2,0);
824         &je     (&label("enc_out"));

826         &pushf  ();
827         &cld    ();

829         &mov    ($s0,&wparam(0));       # load inp
830         &mov    ($s1,&wparam(1));       # load out
831         #&mov   ($s2,&wparam(2));       # load len
832         &mov    ($s3,&wparam(3));       # load key
833         &mov    ($Tbl,&wparam(4));      # load ivp

835         # allocate aligned stack frame...
836         &lea    ($idx,&DWP(-64,"esp"));
837         &and    ($idx,-64);

839         # place stack frame just "above mod 1024" the key schedule
840         # this ensures that cache associativity of 2 suffices
841         &lea    ($key,&DWP(-64-63,$s3));
842         &sub    ($key,$idx);
843         &neg    ($key);
844         &and    ($key,0x3C0);   # modulo 1024, but aligned to cache-line
845         &sub    ($idx,$key);

847         &mov    ($key,&wparam(5));      # load enc

849         &exch   ("esp",$idx);
850         &add    ("esp",4);              # reserve for return address!
851         &mov    ($_esp,$idx);           # save %esp

853         &mov    ($_inp,$s0);            # save copy of inp
```



```
854         &mov    ($_out,$s1);            # save copy of out
855         &mov    ($_len,$s2);            # save copy of len
856         &mov    ($_key,$s3);            # save copy of key
857         &mov    ($_ivp,$Tbl);           # save copy of ivp

859         &call   (&label("pic_point"));  # make it PIC!
860         &set_label("pic_point");
861         &blindpop($Tbl);
862         &lea    ($Tbl,&DWP(&label("Camellia_SBOX")."-".&label("pic_point"),$Tbl)

864         &mov    ($idx,32);
865         &set_label("prefetch_sbox",4);
866                 &mov    ($s0,&DWP(0,$Tbl));
867                 &mov    ($s1,&DWP(32,$Tbl));
868                 &mov    ($s2,&DWP(64,$Tbl));
869                 &mov    ($s3,&DWP(96,$Tbl));
870                 &lea    ($Tbl,&DWP(128,$Tbl));
871                 &dec    ($idx);
872         &jnz    (&label("prefetch_sbox"));
873         &mov    ($s0,$_key);
874         &sub    ($Tbl,4096);
875         &mov    ($idx,$_inp);
876         &mov    ($s3,&DWP(272,$s0));            # load grandRounds

878         &cmp    ($key,0);
879         &je     (&label("DECRYPT"));

881         &mov    ($s2,$_len);
882         &mov    ($key,$_ivp);
883         &shl    ($s3,6);
884         &lea    ($s3,&DWP(0,$s0,$s3));
885         &mov    ($_end,$s3);

887         &test   ($s2,0xFFFFFFF0);
888         &jz     (&label("enc_tail"));           # short input...

890         &mov    ($s0,&DWP(0,$key));             # load iv
891         &mov    ($s1,&DWP(4,$key));

893         &set_label("enc_loop",4);
894                 &mov    ($s2,&DWP(8,$key));
895                 &mov    ($s3,&DWP(12,$key));

897                 &xor    ($s0,&DWP(0,$idx));     # xor input data
898                 &xor    ($s1,&DWP(4,$idx));
899                 &xor    ($s2,&DWP(8,$idx));
900                 &bswap  ($s0);
901                 &xor    ($s3,&DWP(12,$idx));
902                 &bswap  ($s1);
903                 &mov    ($key,$_key);           # load key
904                 &bswap  ($s2);
905                 &bswap  ($s3);

907                 &call   ("_x86_Camellia_encrypt");

909                 &mov    ($idx,$_inp);           # load inp
910                 &mov    ($key,$_out);           # load out

912                 &bswap  ($s0);
913                 &bswap  ($s1);
914                 &bswap  ($s2);
915                 &mov    (&DWP(0,$key),$s0);     # save output data
916                 &bswap  ($s3);
917                 &mov    (&DWP(4,$key),$s1);
918                 &mov    (&DWP(8,$key),$s2);
919                 &mov    (&DWP(12,$key),$s3);
```

```
921                 &mov    ($s2,$_len);            # load len

923                 &lea    ($idx,&DWP(16,$idx));
924                 &mov    ($_inp,$idx);           # save inp

926                 &lea    ($s3,&DWP(16,$key));
927                 &mov    ($_out,$s3);            # save out

929                 &sub    ($s2,16);
930                 &test   ($s2,0xFFFFFFF0);
931                 &mov    ($_len,$s2);            # save len
932         &jnz    (&label("enc_loop"));
933         &test   ($s2,15);
934         &jnz    (&label("enc_tail"));
935         &mov    ($idx,$_ivp);           # load ivp
936         &mov    ($s2,&DWP(8,$key));     # restore last dwords
937         &mov    ($s3,&DWP(12,$key));
938         &mov    (&DWP(0,$idx),$s0);     # save ivec
939         &mov    (&DWP(4,$idx),$s1);
940         &mov    (&DWP(8,$idx),$s2);
941         &mov    (&DWP(12,$idx),$s3);

943         &mov    ("esp",$_esp);
944         &popf   ();
945     &set_label("enc_out");
946         &function_end_A();
947         &pushf  ();                     # kludge, never executed

949     &set_label("enc_tail",4);
950         &mov    ($s0,$key eq "edi" ? $key : "");
951         &mov    ($key,$_out);                   # load out
952         &push   ($s0);                          # push ivp
953         &mov    ($s1,16);
954         &sub    ($s1,$s2);
955         &cmp    ($key,$idx);                    # compare with inp
956         &je     (&label("enc_in_place"));
957         &align  (4);
958         &data_word(0xA4F3F689); # rep movsb     # copy input
959         &jmp    (&label("enc_skip_in_place"));
960     &set_label("enc_in_place");
961         &lea    ($key,&DWP(0,$key,$s2));
962     &set_label("enc_skip_in_place");
963         &mov    ($s2,$s1);
964         &xor    ($s0,$s0);
965         &align  (4);
966         &data_word(0xAAF3F689); # rep stosb     # zero tail
967         &pop    ($key);                         # pop ivp

969         &mov    ($idx,$_out);                   # output as input
970         &mov    ($s0,&DWP(0,$key));
971         &mov    ($s1,&DWP(4,$key));
972         &mov    ($_len,16);                     # len=16
973         &jmp    (&label("enc_loop"));           # one more spin...

975 #----------------------------- DECRYPT -----------------------------#
976 &set_label("DECRYPT",16);
977         &shl    ($s3,6);
978         &lea    ($s3,&DWP(0,$s0,$s3));
979         &mov    ($_end,$s0);
980         &mov    ($_key,$s3);

982         &cmp    ($idx,$_out);
983         &je     (&label("dec_in_place"));       # in-place processing...

985         &mov    ($key,$_ivp);                   # load ivp
```



```
986         &mov    ($_tmp,$key);

988         &set_label("dec_loop",4);
989                 &mov    ($s0,&DWP(0,$idx));     # read input
990                 &mov    ($s1,&DWP(4,$idx));
991                 &mov    ($s2,&DWP(8,$idx));
992                 &bswap  ($s0);
993                 &mov    ($s3,&DWP(12,$idx));
994                 &bswap  ($s1);
995                 &mov    ($key,$_key);           # load key
996                 &bswap  ($s2);
997                 &bswap  ($s3);

999                 &call   ("_x86_Camellia_decrypt");

1001                &mov    ($key,$_tmp);           # load ivp
1002                &mov    ($idx,$_len);           # load len

1004                &bswap  ($s0);
1005                &bswap  ($s1);
1006                &bswap  ($s2);
1007                &xor    ($s0,&DWP(0,$key));     # xor iv
1008                &bswap  ($s3);
1009                &xor    ($s1,&DWP(4,$key));
1010                &xor    ($s2,&DWP(8,$key));
1011                &xor    ($s3,&DWP(12,$key));

1013                &sub    ($idx,16);
1014                &jc     (&label("dec_partial"));
1015                &mov    ($_len,$idx);           # save len
1016                &mov    ($idx,$_inp);           # load inp
1017                &mov    ($key,$_out);           # load out

1019                &mov    (&DWP(0,$key),$s0);     # write output
1020                &mov    (&DWP(4,$key),$s1);
1021                &mov    (&DWP(8,$key),$s2);
1022                &mov    (&DWP(12,$key),$s3);

1024                &mov    ($_tmp,$idx);           # save ivp
1025                &lea    ($idx,&DWP(16,$idx));
1026                &mov    ($_inp,$idx);           # save inp

1028                &lea    ($key,&DWP(16,$key));
1029                &mov    ($_out,$key);           # save out

1031        &jnz    (&label("dec_loop"));
1032        &mov    ($key,$_tmp);           # load temp ivp
1033    &set_label("dec_end");
1034        &mov    ($idx,$_ivp);           # load user ivp
1035        &mov    ($s0,&DWP(0,$key));     # load iv
1036        &mov    ($s1,&DWP(4,$key));
1037        &mov    ($s2,&DWP(8,$key));
1038        &mov    ($s3,&DWP(12,$key));
1039        &mov    (&DWP(0,$idx),$s0);     # copy back to user
1040        &mov    (&DWP(4,$idx),$s1);
1041        &mov    (&DWP(8,$idx),$s2);
1042        &mov    (&DWP(12,$idx),$s3);
1043        &jmp    (&label("dec_out"));

1045    &set_label("dec_partial",4);
1046        &lea    ($key,$ivec);
1047        &mov    (&DWP(0,$key),$s0);     # dump output to stack
1048        &mov    (&DWP(4,$key),$s1);
1049        &mov    (&DWP(8,$key),$s2);
1050        &mov    (&DWP(12,$key),$s3);
1051        &lea    ($s2 eq "ecx" ? $s2 : "",&DWP(16,$idx));
```

```
1052         &mov    ($idx eq "esi" ? $idx : "",$key);
1053         &mov    ($key eq "edi" ? $key : "",$_out);      # load out
1054         &data_word(0xA4F3F689); # rep movsb               # copy output
1055         &mov    ($key,$_inp);                           # use inp as temp ivp
1056         &jmp    (&label("dec_end"));

1058     &set_label("dec_in_place",4);
1059         &set_label("dec_in_place_loop");
1060                 &lea    ($key,$ivec);
1061                 &mov    ($s0,&DWP(0,$idx));     # read input
1062                 &mov    ($s1,&DWP(4,$idx));
1063                 &mov    ($s2,&DWP(8,$idx));
1064                 &mov    ($s3,&DWP(12,$idx));

1066                 &mov    (&DWP(0,$key),$s0);     # copy to temp
1067                 &mov    (&DWP(4,$key),$s1);
1068                 &mov    (&DWP(8,$key),$s2);
1069                 &bswap  ($s0);
1070                 &mov    (&DWP(12,$key),$s3);
1071                 &bswap  ($s1);
1072                 &mov    ($key,$_key);           # load key
1073                 &bswap  ($s2);
1074                 &bswap  ($s3);

1076                 &call   ("_x86_Camellia_decrypt");

1078                 &mov    ($key,$_ivp);           # load ivp
1079                 &mov    ($idx,$_out);           # load out

1081                 &bswap  ($s0);
1082                 &bswap  ($s1);
1083                 &bswap  ($s2);
1084                 &xor    ($s0,&DWP(0,$key));     # xor iv
1085                 &bswap  ($s3);
1086                 &xor    ($s1,&DWP(4,$key));
1087                 &xor    ($s2,&DWP(8,$key));
1088                 &xor    ($s3,&DWP(12,$key));

1090                 &mov    (&DWP(0,$idx),$s0);     # write output
1091                 &mov    (&DWP(4,$idx),$s1);
1092                 &mov    (&DWP(8,$idx),$s2);
1093                 &mov    (&DWP(12,$idx),$s3);

1095                 &lea    ($idx,&DWP(16,$idx));
1096                 &mov    ($_out,$idx);           # save out

1098                 &lea    ($idx,$ivec);
1099                 &mov    ($s0,&DWP(0,$idx));     # read temp
1100                 &mov    ($s1,&DWP(4,$idx));
1101                 &mov    ($s2,&DWP(8,$idx));
1102                 &mov    ($s3,&DWP(12,$idx));

1104                 &mov    (&DWP(0,$key),$s0);     # copy iv
1105                 &mov    (&DWP(4,$key),$s1);
1106                 &mov    (&DWP(8,$key),$s2);
1107                 &mov    (&DWP(12,$key),$s3);

1109                 &mov    ($idx,$_inp);           # load inp

1111                 &lea    ($idx,&DWP(16,$idx));
1112                 &mov    ($_inp,$idx);           # save inp

1114                 &mov    ($s2,$_len);            # load len
1115                 &sub    ($s2,16);
1116                 &jc     (&label("dec_in_place_partial"));
1117                 &mov    ($_len,$s2);            # save len
```



```
1118         &jnz    (&label("dec_in_place_loop"));
1119         &jmp    (&label("dec_out"));

1121     &set_label("dec_in_place_partial",4);
1122         # one can argue if this is actually required...
1123         &mov    ($key eq "edi" ? $key : "",$_out);
1124         &lea    ($idx eq "esi" ? $idx : "",$ivec);
1125         &lea    ($key,&DWP(0,$key,$s2));
1126         &lea    ($idx,&DWP(16,$idx,$s2));
1127         &neg    ($s2 eq "ecx" ? $s2 : "");
1128         &data_word(0xA4F3F689); # rep movsb     # restore tail

1130     &set_label("dec_out",4);
1131     &mov        ("esp",$_esp);
1132     &popf       ();
1133 &function_end("Camellia_cbc_encrypt");
1134 }

1136 &asciz("Camellia for x86 by <appro\@openssl.org>");

1138 &asm_finish();
1139 #endif /* ! codereview */
```

```
**********************************************************
   25677 Wed Aug 13 19:53:06 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/cmll-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# ====================================================================
# Copyright (c) 2008 Andy Polyakov <appro@openssl.org>
#
# This module may be used under the terms of either the GNU General
# Public License version 2 or later, the GNU Lesser General Public
# License version 2.1 or later, the Mozilla Public License version
# 1.1 or the BSD License. The exact terms of either license are
# distributed along with this module. For further details see
# http://www.openssl.org/~appro/camellia/.
# ====================================================================

# Performance in cycles per processed byte (less is better) in
# 'openssl speed ...' benchmark:
#
#                       AMD64   Core2   EM64T
# -evp camellia-128-ecb 16.7    21.0    22.7
# + over gcc 3.4.6      +25%    +5%     0%
#
# camellia-128-cbc      15.7    20.4    21.1
#
# 128-bit key setup     128     216     205     cycles/key
# + over gcc 3.4.6      +54%    +39%    +15%
#
# Numbers in "+" rows represent performance improvement over compiler
# generated code. Key setup timings are impressive on AMD and Core2
# thanks to 64-bit operations being covertly deployed. Improvement on
# EM64T, pre-Core2 Intel x86_64 CPU, is not as impressive, because it
# apparently emulates some of 64-bit operations in [32-bit] microcode.

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

sub hi() { my $r=shift; $r =~ s/%[er]([a-d])x/%\1h/;    $r; }
sub lo() { my $r=shift; $r =~ s/%[er]([a-d])x/%\1l/;
                        $r =~ s/%[er]([sd]i)/%\1l/;
                        $r =~ s/%(r[0-9]+)[d]?/%\1b/;   $r; }

$t0="%eax";$t1="%ebx";$t2="%ecx";$t3="%edx";
@S=("%r8d","%r9d","%r10d","%r11d");
$i0="%esi";
$i1="%edi";
$Tbl="%rbp";    # size optimization
$inp="%r12";
$out="%r13";
$key="%r14";
$keyend="%r15";
$arg0d=$win64?"%ecx":"%edi";
```



```
# const unsigned int Camellia_SBOX[4][256];
# Well, sort of... Camellia_SBOX[0][] is interleaved with [1][],
# and [2][] - with [3][]. This is done to minimize code size.
$SBOX1_1110=0;          # Camellia_SBOX[0]
$SBOX4_4404=4;          # Camellia_SBOX[1]
$SBOX2_0222=2048;       # Camellia_SBOX[2]
$SBOX3_3033=2052;       # Camellia_SBOX[3]

sub Camellia_Feistel {
my $i=@_[0];
my $seed=defined(@_[1])?@_[1]:0;
my $scale=$seed<0?-8:8;
my $j=($i&1)*2;
my $s0=@S[($j)%4],$s1=@S[($j+1)%4],$s2=@S[($j+2)%4],$s3=@S[($j+3)%4];

$code.=<<___;
        xor     $s0,$t0                         # t0^=key[0]
        xor     $s1,$t1                         # t1^=key[1]
        movz    `&hi("$t0")`,$i0                # (t0>>8)&0xff
        movz    `&lo("$t1")`,$i1                # (t1>>0)&0xff
        mov     $SBOX3_3033($Tbl,$i0,8),$t3     # t3=SBOX3_3033[0]
        mov     $SBOX1_1110($Tbl,$i1,8),$t2     # t2=SBOX1_1110[1]
        movz    `&lo("$t0")`,$i0                # (t0>>0)&0xff
        shr     \$16,$t0
        movz    `&hi("$t1")`,$i1                # (t1>>8)&0xff
        xor     $SBOX4_4404($Tbl,$i0,8),$t3     # t3^=SBOX4_4404[0]
        shr     \$16,$t1
        xor     $SBOX4_4404($Tbl,$i1,8),$t2     # t2^=SBOX4_4404[1]
        movz    `&hi("$t0")`,$i0                # (t0>>24)&0xff
        movz    `&lo("$t1")`,$i1                # (t1>>16)&0xff
        xor     $SBOX1_1110($Tbl,$i0,8),$t3     # t3^=SBOX1_1110[0]
        xor     $SBOX3_3033($Tbl,$i1,8),$t2     # t2^=SBOX3_3033[1]
        movz    `&lo("$t0")`,$i0                # (t0>>16)&0xff
        movz    `&hi("$t1")`,$i1                # (t1>>24)&0xff
        xor     $SBOX2_0222($Tbl,$i0,8),$t3     # t3^=SBOX2_0222[0]
        xor     $SBOX2_0222($Tbl,$i1,8),$t2     # t2^=SBOX2_0222[1]
        mov     `$seed+($i+1)*$scale`($key),$t1 # prefetch key[i+1]
        mov     `$seed+($i+1)*$scale+4`($key),$t0
        xor     $t3,$t2                         # t2^=t3
        ror     \$8,$t3                         # t3=RightRotate(t3,8)
        xor     $t2,$s2
        xor     $t2,$s3
        xor     $t3,$s3
___
}

# void Camellia_EncryptBlock_Rounds(
#               int grandRounds,
#               const Byte plaintext[],
#               const KEY_TABLE_TYPE keyTable,
#               Byte ciphertext[])
$code=<<___;
.text

# V1.x API
.globl  Camellia_EncryptBlock
.type   Camellia_EncryptBlock,\@abi-omnipotent
.align  16
Camellia_EncryptBlock:
        movl    \$128,%eax
        subl    $arg0d,%eax
        movl    \$3,$arg0d
        adcl    \$0,$arg0d      # keyBitLength==128?3:4
        jmp     .Lenc_rounds
.size   Camellia_EncryptBlock,.-Camellia_EncryptBlock
# V2
```

```
.globl  Camellia_EncryptBlock_Rounds
.type   Camellia_EncryptBlock_Rounds,\@function,4
.align  16
.Lenc_rounds:
Camellia_EncryptBlock_Rounds:
        push    %rbx
        push    %rbp
        push    %r13
        push    %r14
        push    %r15
.Lenc_prologue:

        #mov    %rsi,$inp               # put away arguments
        mov     %rcx,$out
        mov     %rdx,$key

        shl     \$6,%edi                # process grandRounds
        lea     .LCamellia_SBOX(%rip),$Tbl
        lea     ($key,%rdi),$keyend

        mov     0(%rsi),@S[0]           # load plaintext
        mov     4(%rsi),@S[1]
        mov     8(%rsi),@S[2]
        bswap   @S[0]
        mov     12(%rsi),@S[3]
        bswap   @S[1]
        bswap   @S[2]
        bswap   @S[3]

        call    _x86_64_Camellia_encrypt

        bswap   @S[0]
        bswap   @S[1]
        bswap   @S[2]
        mov     @S[0],0($out)
        bswap   @S[3]
        mov     @S[1],4($out)
        mov     @S[2],8($out)
        mov     @S[3],12($out)

        mov     0(%rsp),%r15
        mov     8(%rsp),%r14
        mov     16(%rsp),%r13
        mov     24(%rsp),%rbp
        mov     32(%rsp),%rbx
        lea     40(%rsp),%rsp
.Lenc_epilogue:
        ret
.size   Camellia_EncryptBlock_Rounds,.-Camellia_EncryptBlock_Rounds

.type   _x86_64_Camellia_encrypt,\@abi-omnipotent
.align  16
_x86_64_Camellia_encrypt:
        xor     0($key),@S[1]
        xor     4($key),@S[0]           # ^=key[0-3]
        xor     8($key),@S[3]
        xor     12($key),@S[2]
.align  16
.Leloop:
        mov     16($key),$t1            # prefetch key[4-5]
        mov     20($key),$t0

___
        for ($i=0;$i<6;$i++) { Camellia_Feistel($i,16); }
$code.=<<___;
        lea     16*4($key),$key
```



```
        cmp     $keyend,$key
        mov     8($key),$t3             # prefetch key[2-3]
        mov     12($key),$t2
        je      .Ledone

        and     @S[0],$t0
        or      @S[3],$t3
        rol     \$1,$t0
        xor     $t3,@S[2]               # s2^=s3|key[3];
        xor     $t0,@S[1]               # s1^=LeftRotate(s0&key[0],1);
        and     @S[2],$t2
        or      @S[1],$t1
        rol     \$1,$t2
        xor     $t1,@S[0]               # s0^=s1|key[1];
        xor     $t2,@S[3]               # s3^=LeftRotate(s2&key[2],1);
        jmp     .Leloop

.align  16
.Ledone:
        xor     @S[2],$t0               # SwapHalf
        xor     @S[3],$t1
        xor     @S[0],$t2
        xor     @S[1],$t3

        mov     $t0,@S[0]
        mov     $t1,@S[1]
        mov     $t2,@S[2]
        mov     $t3,@S[3]

        .byte   0xf3,0xc3               # rep ret
.size   _x86_64_Camellia_encrypt,.-_x86_64_Camellia_encrypt

# V1.x API
.globl  Camellia_DecryptBlock
.type   Camellia_DecryptBlock,\@abi-omnipotent
.align  16
Camellia_DecryptBlock:
        movl    \$128,%eax
        subl    $arg0d,%eax
        movl    \$3,$arg0d
        adcl    \$0,$arg0d      # keyBitLength==128?3:4
        jmp     .Ldec_rounds
.size   Camellia_DecryptBlock,.-Camellia_DecryptBlock
# V2
.globl  Camellia_DecryptBlock_Rounds
.type   Camellia_DecryptBlock_Rounds,\@function,4
.align  16
.Ldec_rounds:
Camellia_DecryptBlock_Rounds:
        push    %rbx
        push    %rbp
        push    %r13
        push    %r14
        push    %r15
.Ldec_prologue:

        #mov    %rsi,$inp               # put away arguments
        mov     %rcx,$out
        mov     %rdx,$keyend

        shl     \$6,%edi                # process grandRounds
        lea     .LCamellia_SBOX(%rip),$Tbl
        lea     ($keyend,%rdi),$key

        mov     0(%rsi),@S[0]           # load plaintext
        mov     4(%rsi),@S[1]
```

```
260         mov     8(%rsi),@S[2]
261         bswap   @S[0]
262         mov     12(%rsi),@S[3]
263         bswap   @S[1]
264         bswap   @S[2]
265         bswap   @S[3]

267         call    _x86_64_Camellia_decrypt

269         bswap   @S[0]
270         bswap   @S[1]
271         bswap   @S[2]
272         mov     @S[0],0($out)
273         bswap   @S[3]
274         mov     @S[1],4($out)
275         mov     @S[2],8($out)
276         mov     @S[3],12($out)

278         mov     0(%rsp),%r15
279         mov     8(%rsp),%r14
280         mov     16(%rsp),%r13
281         mov     24(%rsp),%rbp
282         mov     32(%rsp),%rbx
283         lea     40(%rsp),%rsp
284 .Ldec_epilogue:
285         ret
286 .size   Camellia_DecryptBlock_Rounds,.-Camellia_DecryptBlock_Rounds

288 .type   _x86_64_Camellia_decrypt,\@abi-omnipotent
289 .align  16
290 _x86_64_Camellia_decrypt:
291         xor     0($key),@S[1]
292         xor     4($key),@S[0]           # ^=key[0-3]
293         xor     8($key),@S[3]
294         xor     12($key),@S[2]
295 .align  16
296 .Ldloop:
297         mov     -8($key),$t1            # prefetch key[4-5]
298         mov     -4($key),$t0

300 ___
301         for ($i=0;$i<6;$i++) { Camellia_Feistel($i,-8); }
302 $code.=<<___;
303         lea     -16*4($key),$key
304         cmp     $keyend,$key
305         mov     0($key),$t3             # prefetch key[2-3]
306         mov     4($key),$t2
307         je      .Lddone

309         and     @S[0],$t0
310         or      @S[3],$t3
311         rol     \$1,$t0
312         xor     $t3,@S[2]               # s2^=s3|key[3];
313         xor     $t0,@S[1]               # s1^=LeftRotate(s0&key[0],1);
314         and     @S[2],$t2
315         or      @S[1],$t1
316         rol     \$1,$t2
317         xor     $t1,@S[0]               # s0^=s1|key[1];
318         xor     $t2,@S[3]               # s3^=LeftRotate(s2&key[2],1);

320         jmp     .Ldloop

322 .align  16
323 .Lddone:
324         xor     @S[2],$t2
325         xor     @S[3],$t3
```



```
326         xor     @S[0],$t0
327         xor     @S[1],$t1

329         mov     $t2,@S[0]               # SwapHalf
330         mov     $t3,@S[1]
331         mov     $t0,@S[2]
332         mov     $t1,@S[3]

334         .byte   0xf3,0xc3               # rep ret
335 .size   _x86_64_Camellia_decrypt,.-_x86_64_Camellia_decrypt
336 ___

338 sub _saveround {
339 my ($rnd,$key,@T)=@_;
340 my $bias=int(@T[0])?shift(@T):0;

342     if ($#T==3) {
343         $code.=<<___;
344         mov     @T[1],‘$bias+$rnd*8+0‘($key)
345         mov     @T[0],‘$bias+$rnd*8+4‘($key)
346         mov     @T[3],‘$bias+$rnd*8+8‘($key)
347         mov     @T[2],‘$bias+$rnd*8+12‘($key)
348 ___
349     } else {
350         $code.="        mov     @T[0],‘$bias+$rnd*8+0‘($key)\n";
351         $code.="        mov     @T[1],‘$bias+$rnd*8+8‘($key)\n" if ($#T>=1);
352     }
353 }

355 sub _loadround {
356 my ($rnd,$key,@T)=@_;
357 my $bias=int(@T[0])?shift(@T):0;

359 $code.="        mov     ‘$bias+$rnd*8+0‘($key),@T[0]\n";
360 $code.="        mov     ‘$bias+$rnd*8+8‘($key),@T[1]\n" if ($#T>=1);
361 }

363 # shld is very slow on Intel EM64T family. Even on AMD it limits
364 # instruction decode rate [because it’s VectorPath] and consequently
365 # performance...
366 sub __rotl128 {
367 my ($i0,$i1,$rot)=@_;

369     if ($rot) {
370         $code.=<<___;
371         mov     $i0,%r11
372         shld    \$$rot,$i1,$i0
373         shld    \$$rot,%r11,$i1
374 ___
375     }
376 }

378 # ... Implementing 128-bit rotate without shld gives 80% better
379 # performance EM64T, +15% on AMD64 and only ~7% degradation on
380 # Core2. This is therefore preferred.
381 sub _rotl128 {
382 my ($i0,$i1,$rot)=@_;

384     if ($rot) {
385         $code.=<<___;
386         mov     $i0,%r11
387         shl     \$$rot,$i0
388         mov     $i1,%r9
389         shr     \$‘64-$rot‘,%r9
390         shr     \$‘64-$rot‘,%r11
391         or      %r9,$i0
```

```
392         shl     \$$rot,$i1
393         or      %r11,$i1
394 ___
395     }
396 }

398 { my $step=0;

400 $code.=<<___;
401 .globl  Camellia_Ekeygen
402 .type   Camellia_Ekeygen,\@function,3
403 .align  16
404 Camellia_Ekeygen:
405         push    %rbx
406         push    %rbp
407         push    %r13
408         push    %r14
409         push    %r15
410 .Lkey_prologue:

412         mov     %rdi,$keyend            # put away arguments, keyBitLength
413         mov     %rdx,$out               # keyTable

415         mov     0(%rsi),@S[0]           # load 0-127 bits
416         mov     4(%rsi),@S[1]
417         mov     8(%rsi),@S[2]
418         mov     12(%rsi),@S[3]

420         bswap   @S[0]
421         bswap   @S[1]
422         bswap   @S[2]
423         bswap   @S[3]
424 ___
425         &_saveround     (0,$out,@S);    # KL<<<0
426 $code.=<<___;
427         cmp     \$128,$keyend           # check keyBitLength
428         je      .L1st128

430         mov     16(%rsi),@S[0]          # load 128-191 bits
431         mov     20(%rsi),@S[1]
432         cmp     \$192,$keyend
433         je      .L1st192
434         mov     24(%rsi),@S[2]          # load 192-255 bits
435         mov     28(%rsi),@S[3]
436         jmp     .L1st256
437 .L1st192:
438         mov     @S[0],@S[2]
439         mov     @S[1],@S[3]
440         not     @S[2]
441         not     @S[3]
442 .L1st256:
443         bswap   @S[0]
444         bswap   @S[1]
445         bswap   @S[2]
446         bswap   @S[3]
447 ___
448         &_saveround     (4,$out,@S);    # temp storage for KR!
449 $code.=<<___;
450         xor     0($out),@S[1]           # KR^KL
451         xor     4($out),@S[0]
452         xor     8($out),@S[3]
453         xor     12($out),@S[2]

455 .L1st128:
456         lea     .LCamellia_SIGMA(%rip),$key
457         lea     .LCamellia_SBOX(%rip),$Tbl
```



```
459         mov     0($key),$t1
460         mov     4($key),$t0
461 ___
462         &Camellia_Feistel($step++);
463         &Camellia_Feistel($step++);
464 $code.=<<___;
465         xor     0($out),@S[1]           # ^KL
466         xor     4($out),@S[0]
467         xor     8($out),@S[3]
468         xor     12($out),@S[2]
469 ___
470         &Camellia_Feistel($step++);
471         &Camellia_Feistel($step++);
472 $code.=<<___;
473         cmp     \$128,$keyend
474         jne     .L2nd256

476         lea     128($out),$out          # size optimization
477         shl     \$32,%r8                # @S[0]||
478         shl     \$32,%r10               # @S[2]||
479         or      %r9,%r8                 # ||@S[1]
480         or      %r11,%r10               # ||@S[3]
481 ___
482         &_loadround     (0,$out,-128,"%rax","%rbx");    # KL
483         &_saveround     (2,$out,-128,"%r8","%r10");     # KA<<<0
484         &_rotl128       ("%rax","%rbx",15);
485         &_saveround     (4,$out,-128,"%rax","%rbx");    # KL<<<15
486         &_rotl128       ("%r8","%r10",15);
487         &_saveround     (6,$out,-128,"%r8","%r10");     # KA<<<15
488         &_rotl128       ("%r8","%r10",15);              # 15+15=30
489         &_saveround     (8,$out,-128,"%r8","%r10");     # KA<<<30
490         &_rotl128       ("%rax","%rbx",30);             # 15+30=45
491         &_saveround     (10,$out,-128,"%rax","%rbx");   # KL<<<45
492         &_rotl128       ("%r8","%r10",15);              # 30+15=45
493         &_saveround     (12,$out,-128,"%r8");           # KA<<<45
494         &_rotl128       ("%rax","%rbx",15);             # 45+15=60
495         &_saveround     (13,$out,-128,"%rbx");          # KL<<<60
496         &_rotl128       ("%r8","%r10",15);              # 45+15=60
497         &_saveround     (14,$out,-128,"%r8","%r10");    # KA<<<60
498         &_rotl128       ("%rax","%rbx",17);             # 60+17=77
499         &_saveround     (16,$out,-128,"%rax","%rbx");   # KL<<<77
500         &_rotl128       ("%rax","%rbx",17);             # 77+17=94
501         &_saveround     (18,$out,-128,"%rax","%rbx");   # KL<<<94
502         &_rotl128       ("%r8","%r10",34);              # 60+34=94
503         &_saveround     (20,$out,-128,"%r8","%r10");    # KA<<<94
504         &_rotl128       ("%rax","%rbx",17);             # 94+17=111
505         &_saveround     (22,$out,-128,"%rax","%rbx");   # KL<<<111
506         &_rotl128       ("%r8","%r10",17);              # 94+17=111
507         &_saveround     (24,$out,-128,"%r8","%r10");    # KA<<<111
508 $code.=<<___;
509         mov     \$3,%eax
510         jmp     .Ldone
511 .align  16
512 .L2nd256:
513 ___
514         &_saveround     (6,$out,@S);    # temp storage for KA!
515 $code.=<<___;
516         xor     ‘4*8+0‘($out),@S[1]     # KA^KR
517         xor     ‘4*8+4‘($out),@S[0]
518         xor     ‘5*8+0‘($out),@S[3]
519         xor     ‘5*8+4‘($out),@S[2]
520 ___
521         &Camellia_Feistel($step++);
522         &Camellia_Feistel($step++);
```

```
524         &_loadround     (0,$out,"%rax","%rbx"); # KL
525         &_loadround     (4,$out,"%rcx","%rdx"); # KR
526         &_loadround     (6,$out,"%r14","%r15"); # KA
527 $code.=<<___;
528         lea     128($out),$out          # size optimization
529         shl     \$32,%r8                # @S[0]||
530         shl     \$32,%r10               # @S[2]||
531         or      %r9,%r8                 # ||@S[1]
532         or      %r11,%r10               # ||@S[3]
533 ___
534         &_saveround     (2,$out,-128,"%r8","%r10");     # KB<<<0
535         &_rotl128       ("%rcx","%rdx",15);
536         &_saveround     (4,$out,-128,"%rcx","%rdx");    # KR<<<15
537         &_rotl128       ("%r14","%r15",15);
538         &_saveround     (6,$out,-128,"%r14","%r15");    # KA<<<15
539         &_rotl128       ("%rcx","%rdx",15);             # 15+15=30
540         &_saveround     (8,$out,-128,"%rcx","%rdx");    # KR<<<30
541         &_rotl128       ("%r8","%r10",30);
542         &_saveround     (10,$out,-128,"%r8","%r10");    # KB<<<30
543         &_rotl128       ("%rax","%rbx",45);
544         &_saveround     (12,$out,-128,"%rax","%rbx");   # KL<<<45
545         &_rotl128       ("%r14","%r15",30);             # 15+30=45
546         &_saveround     (14,$out,-128,"%r14","%r15");   # KA<<<45
547         &_rotl128       ("%rax","%rbx",15);             # 45+15=60
548         &_saveround     (16,$out,-128,"%rax","%rbx");   # KL<<<60
549         &_rotl128       ("%rcx","%rdx",30);             # 30+30=60
550         &_saveround     (18,$out,-128,"%rcx","%rdx");   # KR<<<60
551         &_rotl128       ("%r8","%r10",30);              # 30+30=60
552         &_saveround     (20,$out,-128,"%r8","%r10");    # KB<<<60
553         &_rotl128       ("%rax","%rbx",17);             # 60+17=77
554         &_saveround     (22,$out,-128,"%rax","%rbx");   # KL<<<77
555         &_rotl128       ("%r14","%r15",32);             # 45+32=77
556         &_saveround     (24,$out,-128,"%r14","%r15");   # KA<<<77
557         &_rotl128       ("%rcx","%rdx",34);             # 60+34=94
558         &_saveround     (26,$out,-128,"%rcx","%rdx");   # KR<<<94
559         &_rotl128       ("%r14","%r15",17);             # 77+17=94
560         &_saveround     (28,$out,-128,"%r14","%r15");   # KA<<<77
561         &_rotl128       ("%rax","%rbx",34);             # 77+34=111
562         &_saveround     (30,$out,-128,"%rax","%rbx");   # KL<<<111
563         &_rotl128       ("%r8","%r10",51);              # 60+51=111
564         &_saveround     (32,$out,-128,"%r8","%r10");    # KB<<<111
565 $code.=<<___;
566         mov     \$4,%eax
567 .Ldone:
568         mov     0(%rsp),%r15
569         mov     8(%rsp),%r14
570         mov     16(%rsp),%r13
571         mov     24(%rsp),%rbp
572         mov     32(%rsp),%rbx
573         lea     40(%rsp),%rsp
574 .Lkey_epilogue:
575         ret
576 .size   Camellia_Ekeygen,.-Camellia_Ekeygen
577 ___
578 }

580 @SBOX=(
581 112,130, 44,236,179, 39,192,229,228,133, 87, 53,234, 12,174, 65,
582  35,239,107,147, 69, 25,165, 33,237, 14, 79, 78, 29,101,146,189,
583 134,184,175,143,124,235, 31,206, 62, 48,220, 95, 94,197, 11, 26,
584 166,225, 57,202,213, 71, 93, 61,217,  1, 90,214, 81, 86,108, 77,
585 139, 13,154,102,251,204,176, 45,116, 18, 43, 32,240,177,132,153,
586 223, 76,203,194, 52,126,118,  5,109,183,169, 49,209, 23,  4,215,
587  20, 88, 58, 97,222, 27, 17, 28, 50, 15,156, 22, 83, 24,242, 34,
588 254, 68,207,178,195,181,122,145, 36,  8,232,168, 96,252,105, 80,
589 170,208,160,125,161,137, 98,151, 84, 91, 30,149,224,255,100,210,
```



```
590  16,196,  0, 72,163,247,117,219,138,  3,230,218,  9, 63,221,148,
591 135, 92,131,  2,205, 74,144, 51,115,103,246,243,157,127,191,226,
592  82,155,216, 38,200, 55,198, 59,129,150,111, 75, 19,190, 99, 46,
593 233,121,167,140,159,110,188,142, 41,245,249,182, 47,253,180, 89,
594 120,152,  6,106,231, 70,113,186,212, 37,171, 66,136,162,141,250,
595 114,  7,185, 85,248,238,172, 10, 54, 73, 42,104, 60, 56,241,164,
596  64, 40,211,123,187,201, 67,193, 21,227,173,244,119,199,128,158);

598 sub S1110 { my $i=shift; $i=@SBOX[$i]; $i=$i<<24|$i<<16|$i<<8; sprintf("0x%08x",
599 sub S4404 { my $i=shift; $i=($i<<1|$i>>7)&0xff; $i=@SBOX[$i]; $i=$i<<24|$i<<16|$
600 sub S0222 { my $i=shift; $i=@SBOX[$i]; $i=($i<<1|$i>>7)&0xff; $i=$i<<16|$i<<8|$i
601 sub S3033 { my $i=shift; $i=@SBOX[$i]; $i=($i>>1|$i<<7)&0xff; $i=$i<<24|$i<<8|$i

603 $code.=<<___;
604 .align  64
605 .LCamellia_SIGMA:
606 .long   0x3bcc908b, 0xa09e667f, 0x4caa73b2, 0xb67ae858
607 .long   0xe94f82be, 0xc6ef372f, 0xf1d36f1c, 0x54ff53a5
608 .long   0xde682d1d, 0x10e527fa, 0xb3e6c1fd, 0xb05688c2
609 .long   0,          0,          0,          0
610 .LCamellia_SBOX:
611 ___
612 # tables are interleaved, remember?
613 sub data_word { $code.=".long\t".join(',',@_)."\n"; }
614 for ($i=0;$i<256;$i++) { &data_word(&S1110($i),&S4404($i)); }
615 for ($i=0;$i<256;$i++) { &data_word(&S0222($i),&S3033($i)); }

617 # void Camellia_cbc_encrypt (const void char *inp, unsigned char *out,
618 #                        size_t length, const CAMELLIA_KEY *key,
619 #                        unsigned char *ivp,const int enc);
620 {
621 $_key="0(%rsp)";
622 $_end="8(%rsp)";        # inp+len&~15
623 $_res="16(%rsp)";       # len&15
624 $ivec="24(%rsp)";
625 $_ivp="40(%rsp)";
626 $_rsp="48(%rsp)";

628 $code.=<<___;
629 .globl  Camellia_cbc_encrypt
630 .type   Camellia_cbc_encrypt,\@function,6
631 .align  16
632 Camellia_cbc_encrypt:
633         cmp     \$0,%rdx
634         je      .Lcbc_abort
635         push    %rbx
636         push    %rbp
637         push    %r12
638         push    %r13
639         push    %r14
640         push    %r15
641 .Lcbc_prologue:

643         mov     %rsp,%rbp
644         sub     \$64,%rsp
645         and     \$-64,%rsp

647         # place stack frame just "above mod 1024" the key schedule,
648         # this ensures that cache associativity suffices
649         lea     -64-63(%rcx),%r10
650         sub     %rsp,%r10
651         neg     %r10
652         and     \$0x3C0,%r10
653         sub     %r10,%rsp
654         #add    \$8,%rsp                # 8 is reserved for callee's ra
```

```
656         mov     %rdi,$inp               # inp argument
657         mov     %rsi,$out               # out argument
658         mov     %r8,%rbx                # ivp argument
659         mov     %rcx,$key               # key argument
660         mov     272(%rcx),${keyend}d    # grandRounds

662         mov     %r8,$_ivp
663         mov     %rbp,$_rsp

665 .Lcbc_body:
666         lea     .LCamellia_SBOX(%rip),$Tbl

668         mov     \$32,%ecx
669 .align  4
670 .Lcbc_prefetch_sbox:
671         mov     0($Tbl),%rax
672         mov     32($Tbl),%rsi
673         mov     64($Tbl),%rdi
674         mov     96($Tbl),%r11
675         lea     128($Tbl),$Tbl
676         loop    .Lcbc_prefetch_sbox
677         sub     \$4096,$Tbl
678         shl     \$6,$keyend
679         mov     %rdx,%rcx               # len argument
680         lea     ($key,$keyend),$keyend

682         cmp     \$0,%r9d                # enc argument
683         je      .LCBC_DECRYPT

685         and     \$-16,%rdx
686         and     \$15,%rcx               # length residue
687         lea     ($inp,%rdx),%rdx
688         mov     $key,$_key
689         mov     %rdx,$_end
690         mov     %rcx,$_res

692         cmp     $inp,%rdx
693         mov     0(%rbx),@S[0]           # load IV
694         mov     4(%rbx),@S[1]
695         mov     8(%rbx),@S[2]
696         mov     12(%rbx),@S[3]
697         je      .Lcbc_enc_tail
698         jmp     .Lcbc_eloop

700 .align  16
701 .Lcbc_eloop:
702         xor     0($inp),@S[0]
703         xor     4($inp),@S[1]
704         xor     8($inp),@S[2]
705         bswap   @S[0]
706         xor     12($inp),@S[3]
707         bswap   @S[1]
708         bswap   @S[2]
709         bswap   @S[3]

711         call    _x86_64_Camellia_encrypt

713         mov     $_key,$key              # "rewind" the key
714         bswap   @S[0]
715         mov     $_end,%rdx
716         bswap   @S[1]
717         mov     $_res,%rcx
718         bswap   @S[2]
719         mov     @S[0],0($out)
720         bswap   @S[3]
721         mov     @S[1],4($out)
```



```
722         mov     @S[2],8($out)
723         lea     16($inp),$inp
724         mov     @S[3],12($out)
725         cmp     %rdx,$inp
726         lea     16($out),$out
727         jne     .Lcbc_eloop

729         cmp     \$0,%rcx
730         jne     .Lcbc_enc_tail

732         mov     $_ivp,$out
733         mov     @S[0],0($out)           # write out IV residue
734         mov     @S[1],4($out)
735         mov     @S[2],8($out)
736         mov     @S[3],12($out)
737         jmp     .Lcbc_done

739 .align  16
740 .Lcbc_enc_tail:
741         xor     %rax,%rax
742         mov     %rax,0+$ivec
743         mov     %rax,8+$ivec
744         mov     %rax,$_res

746 .Lcbc_enc_pushf:
747         pushfq
748         cld
749         mov     $inp,%rsi
750         lea     8+$ivec,%rdi
751         .long   0x9066A4F3              # rep movsb
752         popfq
753 .Lcbc_enc_popf:

755         lea     $ivec,$inp
756         lea     16+$ivec,%rax
757         mov     %rax,$_end
758         jmp     .Lcbc_eloop             # one more time

760 .align  16
761 .LCBC_DECRYPT:
762         xchg    $key,$keyend
763         add     \$15,%rdx
764         and     \$15,%rcx               # length residue
765         and     \$-16,%rdx
766         mov     $key,$_key
767         lea     ($inp,%rdx),%rdx
768         mov     %rdx,$_end
769         mov     %rcx,$_res

771         mov     (%rbx),%rax             # load IV
772         mov     8(%rbx),%rbx
773         jmp     .Lcbc_dloop
774 .align  16
775 .Lcbc_dloop:
776         mov     0($inp),@S[0]
777         mov     4($inp),@S[1]
778         mov     8($inp),@S[2]
779         bswap   @S[0]
780         mov     12($inp),@S[3]
781         bswap   @S[1]
782         mov     %rax,0+$ivec            # save IV to temporary storage
783         bswap   @S[2]
784         mov     %rbx,8+$ivec
785         bswap   @S[3]

787         call    _x86_64_Camellia_decrypt
```

```
789         mov     $_key,$key              # "rewind" the key
790         mov     $_end,%rdx
791         mov     $_res,%rcx

793         bswap   @S[0]
794         mov     ($inp),%rax             # load IV for next iteration
795         bswap   @S[1]
796         mov     8($inp),%rbx
797         bswap   @S[2]
798         xor     0+$ivec,@S[0]
799         bswap   @S[3]
800         xor     4+$ivec,@S[1]
801         xor     8+$ivec,@S[2]
802         lea     16($inp),$inp
803         xor     12+$ivec,@S[3]
804         cmp     %rdx,$inp
805         je      .Lcbc_ddone

807         mov     @S[0],0($out)
808         mov     @S[1],4($out)
809         mov     @S[2],8($out)
810         mov     @S[3],12($out)

812         lea     16($out),$out
813         jmp     .Lcbc_dloop

815 .align  16
816 .Lcbc_ddone:
817         mov     $_ivp,%rdx
818         cmp     \$0,%rcx
819         jne     .Lcbc_dec_tail

821         mov     @S[0],0($out)
822         mov     @S[1],4($out)
823         mov     @S[2],8($out)
824         mov     @S[3],12($out)

826         mov     %rax,(%rdx)             # write out IV residue
827         mov     %rbx,8(%rdx)
828         jmp     .Lcbc_done
829 .align  16
830 .Lcbc_dec_tail:
831         mov     @S[0],0+$ivec
832         mov     @S[1],4+$ivec
833         mov     @S[2],8+$ivec
834         mov     @S[3],12+$ivec

836 .Lcbc_dec_pushf:
837         pushfq
838         cld
839         lea     8+$ivec,%rsi
840         lea     ($out),%rdi
841         .long   0x9066A4F3              # rep movsb
842         popfq
843 .Lcbc_dec_popf:

845         mov     %rax,(%rdx)             # write out IV residue
846         mov     %rbx,8(%rdx)
847         jmp     .Lcbc_done

849 .align  16
850 .Lcbc_done:
851         mov     $_rsp,%rcx
852         mov     0(%rcx),%r15
853         mov     8(%rcx),%r14
```



```
854         mov     16(%rcx),%r13
855         mov     24(%rcx),%r12
856         mov     32(%rcx),%rbp
857         mov     40(%rcx),%rbx
858         lea     48(%rcx),%rsp
859 .Lcbc_abort:
860         ret
861 .size   Camellia_cbc_encrypt,.-Camellia_cbc_encrypt

863 .asciz  "Camellia for x86_64 by <appro\@openssl.org>"
864 ___
865 }

867 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
868 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
869 if ($win64) {
870 $rec="%rcx";
871 $frame="%rdx";
872 $context="%r8";
873 $disp="%r9";

875 $code.=<<___;
876 .extern __imp_RtlVirtualUnwind
877 .type   common_se_handler,\@abi-omnipotent
878 .align  16
879 common_se_handler:
880         push    %rsi
881         push    %rdi
882         push    %rbx
883         push    %rbp
884         push    %r12
885         push    %r13
886         push    %r14
887         push    %r15
888         pushfq
889         lea     -64(%rsp),%rsp

891         mov     120($context),%rax      # pull context->Rax
892         mov     248($context),%rbx      # pull context->Rip

894         mov     8($disp),%rsi           # disp->ImageBase
895         mov     56($disp),%r11          # disp->HandlerData

897         mov     0(%r11),%r10d           # HandlerData[0]
898         lea     (%rsi,%r10),%r10        # prologue label
899         cmp     %r10,%rbx               # context->Rip<prologue label
900         jb      .Lin_prologue

902         mov     152($context),%rax      # pull context->Rsp

904         mov     4(%r11),%r10d           # HandlerData[1]
905         lea     (%rsi,%r10),%r10        # epilogue label
906         cmp     %r10,%rbx               # context->Rip>=epilogue label
907         jae     .Lin_prologue

909         lea     40(%rax),%rax
910         mov     -8(%rax),%rbx
911         mov     -16(%rax),%rbp
912         mov     -24(%rax),%r13
913         mov     -32(%rax),%r14
914         mov     -40(%rax),%r15
915         mov     %rbx,144($context)      # restore context->Rbx
916         mov     %rbp,160($context)      # restore context->Rbp
917         mov     %r13,224($context)      # restore context->R13
918         mov     %r14,232($context)      # restore context->R14
919         mov     %r15,240($context)      # restore context->R15
```

```
.Lin_prologue:
        mov     8(%rax),%rdi
        mov     16(%rax),%rsi
        mov     %rax,152($context)      # restore context->Rsp
        mov     %rsi,168($context)      # restore context->Rsi
        mov     %rdi,176($context)      # restore context->Rdi

        jmp     .Lcommon_seh_exit
.size   common_se_handler,.-common_se_handler

.type   cbc_se_handler,\@abi-omnipotent
.align  16
cbc_se_handler:
        push    %rsi
        push    %rdi
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
        pushfq
        lea     -64(%rsp),%rsp

        mov     120($context),%rax      # pull context->Rax
        mov     248($context),%rbx      # pull context->Rip

        lea     .Lcbc_prologue(%rip),%r10
        cmp     %r10,%rbx               # context->Rip<.Lcbc_prologue
        jb      .Lin_cbc_prologue

        lea     .Lcbc_body(%rip),%r10
        cmp     %r10,%rbx               # context->Rip<.Lcbc_body
        jb      .Lin_cbc_frame_setup

        mov     152($context),%rax      # pull context->Rsp

        lea     .Lcbc_abort(%rip),%r10
        cmp     %r10,%rbx               # context->Rip>=.Lcbc_abort
        jae     .Lin_cbc_prologue

        # handle pushf/popf in Camellia_cbc_encrypt
        lea     .Lcbc_enc_pushf(%rip),%r10
        cmp     %r10,%rbx               # context->Rip<=.Lcbc_enc_pushf
        jbe     .Lin_cbc_no_flag
        lea     8(%rax),%rax
        lea     .Lcbc_enc_popf(%rip),%r10
        cmp     %r10,%rbx               # context->Rip<.Lcbc_enc_popf
        jb      .Lin_cbc_no_flag
        lea     -8(%rax),%rax
        lea     .Lcbc_dec_pushf(%rip),%r10
        cmp     %r10,%rbx               # context->Rip<=.Lcbc_dec_pushf
        jbe     .Lin_cbc_no_flag
        lea     8(%rax),%rax
        lea     .Lcbc_dec_popf(%rip),%r10
        cmp     %r10,%rbx               # context->Rip<.Lcbc_dec_popf
        jb      .Lin_cbc_no_flag
        lea     -8(%rax),%rax

.Lin_cbc_no_flag:
        mov     48(%rax),%rax           # $_rsp
        lea     48(%rax),%rax

.Lin_cbc_frame_setup:
        mov     -8(%rax),%rbx
```



```
        mov     -16(%rax),%rbp
        mov     -24(%rax),%r12
        mov     -32(%rax),%r13
        mov     -40(%rax),%r14
        mov     -48(%rax),%r15
        mov     %rbx,144($context)      # restore context->Rbx
        mov     %rbp,160($context)      # restore context->Rbp
        mov     %r12,216($context)      # restore context->R12
        mov     %r13,224($context)      # restore context->R13
        mov     %r14,232($context)      # restore context->R14
        mov     %r15,240($context)      # restore context->R15

.Lin_cbc_prologue:
        mov     8(%rax),%rdi
        mov     16(%rax),%rsi
        mov     %rax,152($context)      # restore context->Rsp
        mov     %rsi,168($context)      # restore context->Rsi
        mov     %rdi,176($context)      # restore context->Rdi

.align  4
.Lcommon_seh_exit:

        mov     40($disp),%rdi          # disp->ContextRecord
        mov     $context,%rsi           # context
        mov     \$‘1232/8‘,%ecx         # sizeof(CONTEXT)
        .long   0xa548f3fc              # cld; rep movsq

        mov     $disp,%rsi
        xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
        mov     8(%rsi),%rdx            # arg2, disp->ImageBase
        mov     0(%rsi),%r8             # arg3, disp->ControlPc
        mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
        mov     40(%rsi),%r10           # disp->ContextRecord
        lea     56(%rsi),%r11           # &disp->HandlerData
        lea     24(%rsi),%r12           # &disp->EstablisherFrame
        mov     %r10,32(%rsp)           # arg5
        mov     %r11,40(%rsp)           # arg6
        mov     %r12,48(%rsp)           # arg7
        mov     %rcx,56(%rsp)           # arg8, (NULL)
        call    *__imp_RtlVirtualUnwind(%rip)

        mov     \$1,%eax                # ExceptionContinueSearch
        lea     64(%rsp),%rsp
        popfq
        pop     %r15
        pop     %r14
        pop     %r13
        pop     %r12
        pop     %rbp
        pop     %rbx
        pop     %rdi
        pop     %rsi
        ret
.size   cbc_se_handler,.-cbc_se_handler

.section        .pdata
.align  4
        .rva    .LSEH_begin_Camellia_EncryptBlock_Rounds
        .rva    .LSEH_end_Camellia_EncryptBlock_Rounds
        .rva    .LSEH_info_Camellia_EncryptBlock_Rounds

        .rva    .LSEH_begin_Camellia_DecryptBlock_Rounds
        .rva    .LSEH_end_Camellia_DecryptBlock_Rounds
        .rva    .LSEH_info_Camellia_DecryptBlock_Rounds

        .rva    .LSEH_begin_Camellia_Ekeygen
```

```
1052         .rva    .LSEH_end_Camellia_Ekeygen
1053         .rva    .LSEH_info_Camellia_Ekeygen

1055         .rva    .LSEH_begin_Camellia_cbc_encrypt
1056         .rva    .LSEH_end_Camellia_cbc_encrypt
1057         .rva    .LSEH_info_Camellia_cbc_encrypt

1059 .section        .xdata
1060 .align  8
1061 .LSEH_info_Camellia_EncryptBlock_Rounds:
1062         .byte   9,0,0,0
1063         .rva    common_se_handler
1064         .rva    .Lenc_prologue,.Lenc_epilogue   # HandlerData[]
1065 .LSEH_info_Camellia_DecryptBlock_Rounds:
1066         .byte   9,0,0,0
1067         .rva    common_se_handler
1068         .rva    .Ldec_prologue,.Ldec_epilogue   # HandlerData[]
1069 .LSEH_info_Camellia_Ekeygen:
1070         .byte   9,0,0,0
1071         .rva    common_se_handler
1072         .rva    .Lkey_prologue,.Lkey_epilogue   # HandlerData[]
1073 .LSEH_info_Camellia_cbc_encrypt:
1074         .byte   9,0,0,0
1075         .rva    cbc_se_handler
1076 ___
1077 }

1079 $code =~ s/\`([^\`]*)\`/eval $1/gem;
1080 print $code;
1081 close STDOUT;
1082 #endif /* ! codereview */
```

```
**********************************************************
    5634 Wed Aug 13 19:53:06 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/co-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],$0);

&bn_mul_comba("bn_mul_comba8",8);
&bn_mul_comba("bn_mul_comba4",4);
&bn_sqr_comba("bn_sqr_comba8",8);
&bn_sqr_comba("bn_sqr_comba4",4);

&asm_finish();

sub mul_add_c
        {
        local($a,$ai,$b,$bi,$c0,$c1,$c2,$pos,$i,$na,$nb)=@_;

        # pos == -1 if eax and edx are pre-loaded, 0 to load from next
        # words, and 1 if load return value

        &comment("mul a[$ai]*b[$bi]");

        # "eax" and "edx" will always be pre-loaded.
        # &mov("eax",&DWP($ai*4,$a,"",0)) ;
        # &mov("edx",&DWP($bi*4,$b,"",0));

        &mul("edx");
        &add($c0,"eax");
         &mov("eax",&DWP(($na)*4,$a,"",0)) if $pos == 0;        # laod next a
         &mov("eax",&wparam(0)) if $pos > 0;                    # load r[]
         ###
        &adc($c1,"edx");
         &mov("edx",&DWP(($nb)*4,$b,"",0)) if $pos == 0;        # laod next b
         &mov("edx",&DWP(($nb)*4,$b,"",0)) if $pos == 1;        # laod next b
         ###
        &adc($c2,0);
         # is pos > 1, it means it is the last loop
         &mov(&DWP($i*4,"eax","",0),$c0) if $pos > 0;           # save r[];
        &mov("eax",&DWP(($na)*4,$a,"",0)) if $pos == 1;         # laod next a
        }

sub sqr_add_c
        {
        local($r,$a,$ai,$bi,$c0,$c1,$c2,$pos,$i,$na,$nb)=@_;

        # pos == -1 if eax and edx are pre-loaded, 0 to load from next
        # words, and 1 if load return value

        &comment("sqr a[$ai]*a[$bi]");

        # "eax" and "edx" will always be pre-loaded.
        # &mov("eax",&DWP($ai*4,$a,"",0)) ;
        # &mov("edx",&DWP($bi*4,$b,"",0));

        if ($ai == $bi)
                { &mul("eax");}
        else
                { &mul("edx");}
        &add($c0,"eax");
```



```
         &mov("eax",&DWP(($na)*4,$a,"",0)) if $pos == 0;        # load next a
         ###
        &adc($c1,"edx");
         &mov("edx",&DWP(($nb)*4,$a,"",0)) if ($pos == 1) && ($na != $nb);
         ###
        &adc($c2,0);
         # is pos > 1, it means it is the last loop
         &mov(&DWP($i*4,$r,"",0),$c0) if $pos > 0;              # save r[];
        &mov("eax",&DWP(($na)*4,$a,"",0)) if $pos == 1;         # load next b
        }

sub sqr_add_c2
        {
        local($r,$a,$ai,$bi,$c0,$c1,$c2,$pos,$i,$na,$nb)=@_;

        # pos == -1 if eax and edx are pre-loaded, 0 to load from next
        # words, and 1 if load return value

        &comment("sqr a[$ai]*a[$bi]");

        # "eax" and "edx" will always be pre-loaded.
        # &mov("eax",&DWP($ai*4,$a,"",0)) ;
        # &mov("edx",&DWP($bi*4,$a,"",0));

        if ($ai == $bi)
                { &mul("eax");}
        else
                { &mul("edx");}
        &add("eax","eax");
         ###
        &adc("edx","edx");
         ###
        &adc($c2,0);
         &add($c0,"eax");
        &adc($c1,"edx");
         &mov("eax",&DWP(($na)*4,$a,"",0)) if $pos == 0;        # load next a
         &mov("eax",&DWP(($na)*4,$a,"",0)) if $pos == 1;        # load next b
        &adc($c2,0);
        &mov(&DWP($i*4,$r,"",0),$c0) if $pos > 0;               # save r[];
         &mov("edx",&DWP(($nb)*4,$a,"",0)) if ($pos <= 1) && ($na != $nb);
         ###
        }

sub bn_mul_comba
        {
        local($name,$num)=@_;
        local($a,$b,$c0,$c1,$c2);
        local($i,$as,$ae,$bs,$be,$ai,$bi);
        local($tot,$end);

        &function_begin_B($name,"");

        $c0="ebx";
        $c1="ecx";
        $c2="ebp";
        $a="esi";
        $b="edi";

        $as=0;
        $ae=0;
        $bs=0;
        $be=0;
        $tot=$num+$num-1;

        &push("esi");
         &mov($a,&wparam(1));
```

```
128         &push("edi");
129          &mov($b,&wparam(2));
130         &push("ebp");
131          &push("ebx");

133         &xor($c0,$c0);
134          &mov("eax",&DWP(0,$a,"",0));   # load the first word
135         &xor($c1,$c1);
136          &mov("edx",&DWP(0,$b,"",0));   # load the first second

138         for ($i=0; $i<$tot; $i++)
139                 {
140                 $ai=$as;
141                 $bi=$bs;
142                 $end=$be+1;

144                 &comment("################## Calculate word $i");

146                 for ($j=$bs; $j<$end; $j++)
147                         {
148                         &xor($c2,$c2) if ($j == $bs);
149                         if (($j+1) == $end)
150                                 {
151                                 $v=1;
152                                 $v=2 if (($i+1) == $tot);
153                                 }
154                         else
155                                 { $v=0; }
156                         if (($j+1) != $end)
157                                 {
158                                 $na=($ai-1);
159                                 $nb=($bi+1);
160                                 }
161                         else
162                                 {
163                                 $na=$as+($i < ($num-1));
164                                 $nb=$bs+($i >= ($num-1));
165                                 }
166 #printf STDERR "[$ai,$bi] -> [$na,$nb]\n";
167                         &mul_add_c($a,$ai,$b,$bi,$c0,$c1,$c2,$v,$i,$na,$nb);
168                         if ($v)
169                                 {
170                                 &comment("saved r[$i]");
171                                 # &mov("eax",&wparam(0));
172                                 # &mov(&DWP($i*4,"eax","",0),$c0);
173                                 ($c0,$c1,$c2)=($c1,$c2,$c0);
174                                 }
175                         $ai--;
176                         $bi++;
177                         }
178                 $as++ if ($i < ($num-1));
179                 $ae++ if ($i >= ($num-1));

181                 $bs++ if ($i >= ($num-1));
182                 $be++ if ($i < ($num-1));
183                 }
184         &comment("save r[$i]");
185         # &mov("eax",&wparam(0));
186         &mov(&DWP($i*4,"eax","",0),$c0);

188         &pop("ebx");
189         &pop("ebp");
190         &pop("edi");
191         &pop("esi");
192         &ret();
193         &function_end_B($name);
```



```
194         }

196 sub bn_sqr_comba
197         {
198         local($name,$num)=@_;
199         local($r,$a,$c0,$c1,$c2)=@_;
200         local($i,$as,$ae,$bs,$be,$ai,$bi);
201         local($b,$tot,$end,$half);

203         &function_begin_B($name,"");

205         $c0="ebx";
206         $c1="ecx";
207         $c2="ebp";
208         $a="esi";
209         $r="edi";

211         &push("esi");
212          &push("edi");
213         &push("ebp");
214          &push("ebx");
215         &mov($r,&wparam(0));
216          &mov($a,&wparam(1));
217         &xor($c0,$c0);
218          &xor($c1,$c1);
219         &mov("eax",&DWP(0,$a,"",0)); # load the first word

221         $as=0;
222         $ae=0;
223         $bs=0;
224         $be=0;
225         $tot=$num+$num-1;

227         for ($i=0; $i<$tot; $i++)
228                 {
229                 $ai=$as;
230                 $bi=$bs;
231                 $end=$be+1;

233                 &comment("############### Calculate word $i");
234                 for ($j=$bs; $j<$end; $j++)
235                         {
236                         &xor($c2,$c2) if ($j == $bs);
237                         if (($ai-1) < ($bi+1))
238                                 {
239                                 $v=1;
240                                 $v=2 if ($i+1) == $tot;
241                                 }
242                         else
243                                 { $v=0; }
244                         if (!$v)
245                                 {
246                                 $na=$ai-1;
247                                 $nb=$bi+1;
248                                 }
249                         else
250                                 {
251                                 $na=$as+($i < ($num-1));
252                                 $nb=$bs+($i >= ($num-1));
253                                 }
254                         if ($ai == $bi)
255                                 {
256                                 &sqr_add_c($r,$a,$ai,$bi,
257                                         $c0,$c1,$c2,$v,$i,$na,$nb);
258                                 }
259                         else
```

```
260                                 {
261                                 &sqr_add_c2($r,$a,$ai,$bi,
262                                         $c0,$c1,$c2,$v,$i,$na,$nb);
263                                 }
264                         if ($v)
265                                 {
266                                 &comment("saved r[$i]");
267                                 #&mov(&DWP($i*4,$r,"",0),$c0);
268                                 ($c0,$c1,$c2)=($c1,$c2,$c0);
269                                 last;
270                                 }
271                         $ai--;
272                         $bi++;
273                         }
274                 $as++ if ($i < ($num-1));
275                 $ae++ if ($i >= ($num-1));

277                 $bs++ if ($i >= ($num-1));
278                 $be++ if ($i < ($num-1));
279                 }
280         &mov(&DWP($i*4,$r,"",0),$c0);
281         &pop("ebx");
282         &pop("ebp");
283         &pop("edi");
284         &pop("esi");
285         &ret();
286         &function_end_B($name);
287         }
288 #endif /* ! codereview */
```

```
**********************************************************
    4355 Wed Aug 13 19:53:07 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/crypt586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl
#
# The inner loop instruction sequence and the IP/FP modifications are from
# Svend Olaf Mikkelsen <svolaf@inet.uni-c.dk>
# I've added the stuff needed for crypt() but I've not worried about making
# things perfect.
#

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],"crypt586.pl");

$L="edi";
$R="esi";

&external_label("DES_SPtrans");
&fcrypt_body("fcrypt_body");
&asm_finish();

sub fcrypt_body
        {
        local($name,$do_ip)=@_;

        &function_begin($name);

        &comment("");
        &comment("Load the 2 words");
        $trans="ebp";

        &xor(   $L,     $L);
        &xor(   $R,     $R);

        # PIC-ification:-)
        &picmeup("edx","DES_SPtrans");
        #if ($cpp)      { &picmeup("edx","DES_SPtrans");   }
        #else           { &lea("edx",&DWP("DES_SPtrans")); }
        &push("edx");   # becomes &swtmp(1)
        #
        &mov($trans,&wparam(1)); # reloaded with DES_SPtrans in D_ENCRYPT

        &push(&DWC(25)); # add a variable

        &set_label("start");
        for ($i=0; $i<16; $i+=2)
                {
                &comment("");
                &comment("Round $i");
                &D_ENCRYPT($i,$L,$R,$i*2,$trans,"eax","ebx","ecx","edx");

                &comment("");
                &comment("Round ".sprintf("%d",$i+1));
                &D_ENCRYPT($i+1,$R,$L,($i+1)*2,$trans,"eax","ebx","ecx","edx");
                }
         &mov("ebx",    &swtmp(0));
        &mov("eax",     $L);
         &dec("ebx");
        &mov($L,        $R);
         &mov($R,       "eax");
        &mov(&swtmp(0), "ebx");
         &jnz(&label("start"));

        &comment("");
        &comment("FP");
        &mov("edx",&wparam(0));

        &FP_new($R,$L,"eax",3);
        &mov(&DWP(0,"edx","",0),"eax");
        &mov(&DWP(4,"edx","",0),$L);

        &add("esp",8);  # remove variables

        &function_end($name);
        }

sub D_ENCRYPT
        {
        local($r,$L,$R,$S,$trans,$u,$tmp1,$tmp2,$t)=@_;

        &mov(   $u,             &wparam(2));                    # 2
        &mov(   $t,             $R);
        &shr(   $t,             16);                            # 1
        &mov(   $tmp2,          &wparam(3));                    # 2
        &xor(   $t,             $R);                            # 1

        &and(   $u,             $t);                            # 2
        &and(   $t,             $tmp2);                         # 2

        &mov(   $tmp1,          $u);
        &shl(   $tmp1,          16);                            # 1
        &mov(   $tmp2,          $t);
        &shl(   $tmp2,          16);                            # 1
        &xor(   $u,             $tmp1);                         # 2
        &xor(   $t,             $tmp2);                         # 2
        &mov(   $tmp1,          &DWP(&n2a($S*4),$trans,"",0));  # 2
        &xor(   $u,             $tmp1);
        &mov(   $tmp2,          &DWP(&n2a(($S+1)*4),$trans,"",0));      # 2
        &xor(   $u,             $R);
        &xor(   $t,             $R);
        &xor(   $t,             $tmp2);

        &and(   $u,             "0xfcfcfcfc"    );              # 2
        &xor(   $tmp1,          $tmp1);                         # 1
        &and(   $t,             "0xcfcfcfcf"    );              # 2
        &xor(   $tmp2,          $tmp2);
        &movb(  &LB($tmp1),     &LB($u) );
        &movb(  &LB($tmp2),     &HB($u) );
        &rotr(  $t,             4               );
        &mov(   $trans,         &swtmp(1));
        &xor(   $L,             &DWP("     ",$trans,$tmp1,0));
        &movb(  &LB($tmp1),     &LB($t) );
        &xor(   $L,             &DWP("0x200",$trans,$tmp2,0));
        &movb(  &LB($tmp2),     &HB($t) );
        &shr(   $u,             16);
        &xor(   $L,             &DWP("0x100",$trans,$tmp1,0));
        &movb(  &LB($tmp1),     &HB($u) );
        &shr(   $t,             16);
        &xor(   $L,             &DWP("0x300",$trans,$tmp2,0));
        &movb(  &LB($tmp2),     &HB($t) );
        &and(   $u,             "0xff"  );
        &and(   $t,             "0xff"  );
        &mov(   $tmp1,          &DWP("0x600",$trans,$tmp1,0));
        &xor(   $L,             $tmp1);
        &mov(   $tmp1,          &DWP("0x700",$trans,$tmp2,0));
        &xor(   $L,             $tmp1);
        &mov(   $tmp1,          &DWP("0x400",$trans,$u,0));
```

```
128         &xor(   $L,             $tmp1);
129         &mov(   $tmp1,          &DWP("0x500",$trans,$t,0));
130         &xor(   $L,             $tmp1);
131         &mov(   $trans,         &wparam(1));
132         }

134 sub n2a
135         {
136         sprintf("%d",$_[0]);
137         }

139 # now has a side affect of rotating $a by $shift
140 sub R_PERM_OP
141         {
142         local($a,$b,$tt,$shift,$mask,$last)=@_;

144         &rotl(  $a,             $shift          ) if ($shift != 0);
145         &mov(   $tt,            $a              );
146         &xor(   $a,             $b              );
147         &and(   $a,             $mask           );
148         if ($notlast eq $b)
149                 {
150                 &xor(   $b,             $a              );
151                 &xor(   $tt,            $a              );
152                 }
153         else
154                 {
155                 &xor(   $tt,            $a              );
156                 &xor(   $b,             $a              );
157                 }
158         &comment("");
159         }

161 sub IP_new
162         {
163         local($l,$r,$tt,$lr)=@_;

165         &R_PERM_OP($l,$r,$tt, 4,"0xf0f0f0f0",$l);
166         &R_PERM_OP($r,$tt,$l,20,"0xfff0000f",$l);
167         &R_PERM_OP($l,$tt,$r,14,"0x33333333",$r);
168         &R_PERM_OP($tt,$r,$l,22,"0x03fc03fc",$r);
169         &R_PERM_OP($l,$r,$tt, 9,"0xaaaaaaaa",$r);

171         if ($lr != 3)
172                 {
173                 if (($lr-3) < 0)
174                         { &rotr($tt,    3-$lr); }
175                 else    { &rotl($tt,    $lr-3); }
176                 }
177         if ($lr != 2)
178                 {
179                 if (($lr-2) < 0)
180                         { &rotr($r,     2-$lr); }
181                 else    { &rotl($r,     $lr-2); }
182                 }
183         }

185 sub FP_new
186         {
187         local($l,$r,$tt,$lr)=@_;

189         if ($lr != 2)
190                 {
191                 if (($lr-2) < 0)
192                         { &rotl($r,     2-$lr); }
193                 else    { &rotr($r,     $lr-2); }
```



```
194                 }
195         if ($lr != 3)
196                 {
197                 if (($lr-3) < 0)
198                         { &rotl($l,     3-$lr); }
199                 else    { &rotr($l,     $lr-3); }
200                 }

202         &R_PERM_OP($l,$r,$tt, 0,"0xaaaaaaaa",$r);
203         &R_PERM_OP($tt,$r,$l,23,"0x03fc03fc",$r);
204         &R_PERM_OP($l,$r,$tt,10,"0x33333333",$l);
205         &R_PERM_OP($r,$tt,$l,18,"0xfff0000f",$l);
206         &R_PERM_OP($l,$tt,$r,12,"0xf0f0f0f0",$r);
207         &rotr($tt       , 4);
208         }
209 #endif /* ! codereview */
```

```
**********************************************************
   14495 Wed Aug 13 19:53:07 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/des-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl
#
# The inner loop instruction sequence and the IP/FP modifications are from
# Svend Olaf Mikkelsen <svolaf@inet.uni-c.dk>
#

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";
require "cbc.pl";
require "desboth.pl";

# base code is in microsft
# op dest, source
# format.
#

&asm_init($ARGV[0],"des-586.pl");

$L="edi";
$R="esi";
$trans="ebp";
$small_footprint=1 if (grep(/\-DOPENSSL_SMALL_FOOTPRINT/,@ARGV));
# one can discuss setting this variable to 1 unconditionally, as
# the folded loop is only 3% slower than unrolled, but >7 times smaller

&public_label("DES_SPtrans");

&DES_encrypt_internal();
&DES_decrypt_internal();
&DES_encrypt("DES_encrypt1",1);
&DES_encrypt("DES_encrypt2",0);
&DES_encrypt3("DES_encrypt3",1);
&DES_encrypt3("DES_decrypt3",0);
&cbc("DES_ncbc_encrypt","DES_encrypt1","DES_encrypt1",0,4,5,3,5,-1);
&cbc("DES_ede3_cbc_encrypt","DES_encrypt3","DES_decrypt3",0,6,7,3,4,5);
&DES_SPtrans();

&asm_finish();

sub DES_encrypt_internal()
        {
        &function_begin_B("_x86_DES_encrypt");

        if ($small_footprint)
            {
            &lea("edx",&DWP(128,"ecx"));
            &push("edx");
            &push("ecx");
            &set_label("eloop");
                &D_ENCRYPT(0,$L,$R,0,$trans,"eax","ebx","ecx","edx",&swtmp(0));
                &comment("");
                &D_ENCRYPT(1,$R,$L,2,$trans,"eax","ebx","ecx","edx",&swtmp(0));
                &comment("");
                &add("ecx",16);
                &cmp("ecx",&swtmp(1));
                &mov(&swtmp(0),"ecx");
                &jb(&label("eloop"));
            &add("esp",8);
            }
        else
```



```
            {
            &push("ecx");
            for ($i=0; $i<16; $i+=2)
                {
                &comment("Round $i");
                &D_ENCRYPT($i,$L,$R,$i*2,$trans,"eax","ebx","ecx","edx",&swtmp(0
                &comment("Round ".sprintf("%d",$i+1));
                &D_ENCRYPT($i+1,$R,$L,($i+1)*2,$trans,"eax","ebx","ecx","edx",&s
                }
            &add("esp",4);
        }
        &ret();

        &function_end_B("_x86_DES_encrypt");
        }

sub DES_decrypt_internal()
        {
        &function_begin_B("_x86_DES_decrypt");

        if ($small_footprint)
            {
            &push("ecx");
            &lea("ecx",&DWP(128,"ecx"));
            &push("ecx");
            &set_label("dloop");
                &D_ENCRYPT(0,$L,$R,-2,$trans,"eax","ebx","ecx","edx",&swtmp(0));
                &comment("");
                &D_ENCRYPT(1,$R,$L,-4,$trans,"eax","ebx","ecx","edx",&swtmp(0));
                &comment("");
                &sub("ecx",16);
                &cmp("ecx",&swtmp(1));
                &mov(&swtmp(0),"ecx");
                &ja(&label("dloop"));
            &add("esp",8);
            }
        else
            {
            &push("ecx");
            for ($i=15; $i>0; $i-=2)
                {
                &comment("Round $i");
                &D_ENCRYPT(15-$i,$L,$R,$i*2,$trans,"eax","ebx","ecx","edx",&swtm
                &comment("Round ".sprintf("%d",$i-1));
                &D_ENCRYPT(15-$i+1,$R,$L,($i-1)*2,$trans,"eax","ebx","ecx","edx"
                }
            &add("esp",4);
            }
        &ret();

        &function_end_B("_x86_DES_decrypt");
        }

sub DES_encrypt
        {
        local($name,$do_ip)=@_;

        &function_begin_B($name);

        &push("esi");
        &push("edi");

        &comment("");
        &comment("Load the 2 words");

        if ($do_ip)
```

```
128                 {
129                 &mov($R,&wparam(0));
130                  &xor(  "ecx",          "ecx"           );

132                 &push("ebx");
133                 &push("ebp");

135                 &mov("eax",&DWP(0,$R,"",0));
136                  &mov("ebx",&wparam(2));        # get encrypt flag
137                 &mov($L,&DWP(4,$R,"",0));
138                 &comment("");
139                 &comment("IP");
140                 &IP_new("eax",$L,$R,3);
141                 }
142         else
143                 {
144                 &mov("eax",&wparam(0));
145                  &xor(  "ecx",          "ecx"           );

147                 &push("ebx");
148                 &push("ebp");

150                 &mov($R,&DWP(0,"eax","",0));
151                  &mov("ebx",&wparam(2));        # get encrypt flag
152                 &rotl($R,3);
153                 &mov($L,&DWP(4,"eax","",0));
154                 &rotl($L,3);
155                 }

157         # PIC-ification:-)
158         &call   (&label("pic_point"));
159         &set_label("pic_point");
160         &blindpop($trans);
161         &lea    ($trans,&DWP(&label("DES_SPtrans")."-".&label("pic_point"),$tran

163         &mov(   "ecx",  &wparam(1)      );

165         &cmp("ebx","0");
166         &je(&label("decrypt"));
167         &call("_x86_DES_encrypt");
168         &jmp(&label("done"));
169         &set_label("decrypt");
170         &call("_x86_DES_decrypt");
171         &set_label("done");

173         if ($do_ip)
174                 {
175                 &comment("");
176                 &comment("FP");
177                 &mov("edx",&wparam(0));
178                 &FP_new($L,$R,"eax",3);

180                 &mov(&DWP(0,"edx","",0),"eax");
181                 &mov(&DWP(4,"edx","",0),$R);
182                 }
183         else
184                 {
185                 &comment("");
186                 &comment("Fixup");
187                 &rotr($L,3);           # r
188                  &mov("eax",&wparam(0));
189                 &rotr($R,3);           # l
190                  &mov(&DWP(0,"eax","",0),$L);
191                  &mov(&DWP(4,"eax","",0),$R);
192                 }
```



```
194         &pop("ebp");
195         &pop("ebx");
196         &pop("edi");
197         &pop("esi");
198         &ret();

200         &function_end_B($name);
201         }

203 sub D_ENCRYPT
204         {
205         local($r,$L,$R,$S,$trans,$u,$tmp1,$tmp2,$t,$wp1)=@_;

207          &mov(  $u,             &DWP(&n2a($S*4),$tmp2,"",0));
208         &xor(   $tmp1,          $tmp1);
209          &mov(  $t,             &DWP(&n2a(($S+1)*4),$tmp2,"",0));
210         &xor(   $u,             $R);
211         &xor(   $tmp2,          $tmp2);
212          &xor(  $t,             $R);
213         &and(   $u,             "0xfcfcfcfc"    );
214          &and(  $t,             "0xcfcfcfcf"    );
215         &movb(  &LB($tmp1),     &LB($u) );
216          &movb( &LB($tmp2),     &HB($u) );
217         &rotr(  $t,             4               );
218         &xor(   $L,             &DWP("     ",$trans,$tmp1,0));
219          &movb( &LB($tmp1),     &LB($t) );
220          &xor(  $L,             &DWP("0x200",$trans,$tmp2,0));
221          &movb( &LB($tmp2),     &HB($t) );
222         &shr(   $u,             16);
223          &xor(  $L,             &DWP("0x100",$trans,$tmp1,0));
224          &movb( &LB($tmp1),     &HB($u) );
225         &shr(   $t,             16);
226          &xor(  $L,             &DWP("0x300",$trans,$tmp2,0));
227         &movb(  &LB($tmp2),     &HB($t) );
228          &and(  $u,             "0xff"  );
229         &and(   $t,             "0xff"  );
230          &xor(  $L,             &DWP("0x600",$trans,$tmp1,0));
231          &xor(  $L,             &DWP("0x700",$trans,$tmp2,0));
232         &mov(   $tmp2,          $wp1    );
233          &xor(  $L,             &DWP("0x400",$trans,$u,0));
234          &xor(  $L,             &DWP("0x500",$trans,$t,0));
235         }

237 sub n2a
238         {
239         sprintf("%d",$_[0]);
240         }

242 # now has a side affect of rotating $a by $shift
243 sub R_PERM_OP
244         {
245         local($a,$b,$tt,$shift,$mask,$last)=@_;

247         &rotl(  $a,             $shift          ) if ($shift != 0);
248         &mov(   $tt,            $a              );
249         &xor(   $a,             $b              );
250         &and(   $a,             $mask           );
251         # This can never succeed, and besides it is difficult to see what the
252         # idea was - Ben 13 Feb 99
253         if (!$last eq $b)
254                 {
255                 &xor(   $b,             $a              );
256                 &xor(   $tt,            $a              );
257                 }
258         else
259                 {
```

```
260                 &xor(   $tt,            $a              );
261                 &xor(   $b,             $a              );
262                 }
263         &comment("");
264         }

266 sub IP_new
267         {
268         local($l,$r,$tt,$lr)=@_;

270         &R_PERM_OP($l,$r,$tt, 4,"0xf0f0f0f0",$l);
271         &R_PERM_OP($r,$tt,$l,20,"0xfff0000f",$l);
272         &R_PERM_OP($l,$tt,$r,14,"0x33333333",$r);
273         &R_PERM_OP($tt,$r,$l,22,"0x03fc03fc",$r);
274         &R_PERM_OP($l,$r,$tt, 9,"0xaaaaaaaa",$r);

276         if ($lr != 3)
277                 {
278                 if (($lr-3) < 0)
279                         { &rotr($tt,    3-$lr); }
280                 else    { &rotl($tt,    $lr-3); }
281                 }
282         if ($lr != 2)
283                 {
284                 if (($lr-2) < 0)
285                         { &rotr($r,     2-$lr); }
286                 else    { &rotl($r,     $lr-2); }
287                 }
288         }

290 sub FP_new
291         {
292         local($l,$r,$tt,$lr)=@_;

294         if ($lr != 2)
295                 {
296                 if (($lr-2) < 0)
297                         { &rotl($r,     2-$lr); }
298                 else    { &rotr($r,     $lr-2); }
299                 }
300         if ($lr != 3)
301                 {
302                 if (($lr-3) < 0)
303                         { &rotl($l,     3-$lr); }
304                 else    { &rotr($l,     $lr-3); }
305                 }

307         &R_PERM_OP($l,$r,$tt, 0,"0xaaaaaaaa",$r);
308         &R_PERM_OP($tt,$r,$l,23,"0x03fc03fc",$r);
309         &R_PERM_OP($l,$r,$tt,10,"0x33333333",$l);
310         &R_PERM_OP($r,$tt,$l,18,"0xfff0000f",$l);
311         &R_PERM_OP($l,$tt,$r,12,"0xf0f0f0f0",$r);
312         &rotr($tt       , 4);
313         }

315 sub DES_SPtrans
316         {
317         &set_label("DES_SPtrans",64);
318         &data_word(0x02080800, 0x00080000, 0x02000002, 0x02080802);
319         &data_word(0x02000000, 0x00080802, 0x00080002, 0x02000002);
320         &data_word(0x00080802, 0x02080800, 0x02080000, 0x00000802);
321         &data_word(0x02000802, 0x02000000, 0x00000000, 0x00080002);
322         &data_word(0x00080000, 0x00000002, 0x02000800, 0x00080800);
323         &data_word(0x02080802, 0x02080000, 0x00000802, 0x02000800);
324         &data_word(0x00000002, 0x00000800, 0x00080800, 0x02080002);
325         &data_word(0x00000800, 0x02000802, 0x02080002, 0x00000000);
```



```
326         &data_word(0x00000000, 0x02080802, 0x02000800, 0x00080002);
327         &data_word(0x02080800, 0x00080000, 0x00000802, 0x02000800);
328         &data_word(0x02080002, 0x00000800, 0x00080800, 0x02000002);
329         &data_word(0x00080802, 0x00000002, 0x02000002, 0x02080000);
330         &data_word(0x02080802, 0x00080800, 0x02080000, 0x02000802);
331         &data_word(0x02000000, 0x00000802, 0x00080002, 0x00000000);
332         &data_word(0x00080000, 0x02000000, 0x02000802, 0x02080800);
333         &data_word(0x00000002, 0x02080002, 0x00000800, 0x00080802);
334         # nibble 1
335         &data_word(0x40108010, 0x00000000, 0x00108000, 0x40100000);
336         &data_word(0x40000010, 0x00008010, 0x40008000, 0x00108000);
337         &data_word(0x00008000, 0x40100010, 0x00000010, 0x40008000);
338         &data_word(0x00100010, 0x40108000, 0x40100000, 0x00000010);
339         &data_word(0x00100000, 0x40008010, 0x40100010, 0x00008000);
340         &data_word(0x00108010, 0x40000000, 0x00000000, 0x00100010);
341         &data_word(0x40008010, 0x00108010, 0x40108000, 0x40000010);
342         &data_word(0x40000000, 0x00100000, 0x00008010, 0x40108010);
343         &data_word(0x00100010, 0x40108000, 0x40008000, 0x00108010);
344         &data_word(0x40108010, 0x00100010, 0x40000010, 0x00000000);
345         &data_word(0x40000000, 0x00008010, 0x00100000, 0x40100010);
346         &data_word(0x00008000, 0x40000000, 0x00108010, 0x40008010);
347         &data_word(0x40108000, 0x00008000, 0x00000000, 0x40000010);
348         &data_word(0x00000010, 0x40108010, 0x00108000, 0x40100000);
349         &data_word(0x40100010, 0x00100000, 0x00008010, 0x40008000);
350         &data_word(0x40008010, 0x00000010, 0x40100000, 0x00108000);
351         # nibble 2
352         &data_word(0x04000001, 0x04040100, 0x00000100, 0x04000101);
353         &data_word(0x00040001, 0x04000000, 0x04000101, 0x00040100);
354         &data_word(0x04000100, 0x00040000, 0x04040000, 0x00000001);
355         &data_word(0x04040101, 0x00000101, 0x00000001, 0x04040001);
356         &data_word(0x00000000, 0x00040001, 0x04040100, 0x00000100);
357         &data_word(0x00000101, 0x04040101, 0x00040000, 0x04000001);
358         &data_word(0x04040001, 0x04000100, 0x00040101, 0x04040000);
359         &data_word(0x00040100, 0x00000000, 0x04000000, 0x00040101);
360         &data_word(0x04040100, 0x00000100, 0x00000001, 0x00040000);
361         &data_word(0x00000101, 0x00040001, 0x04040000, 0x04000101);
362         &data_word(0x00000000, 0x04040100, 0x00040100, 0x04040001);
363         &data_word(0x00040001, 0x04000000, 0x04040101, 0x00000001);
364         &data_word(0x00040101, 0x04000001, 0x04000000, 0x04040101);
365         &data_word(0x00040000, 0x04000100, 0x04000101, 0x00040100);
366         &data_word(0x04000100, 0x00000000, 0x04040001, 0x00000101);
367         &data_word(0x04000001, 0x00040101, 0x00000100, 0x04040000);
368         # nibble 3
369         &data_word(0x00401008, 0x10001000, 0x00000008, 0x10401008);
370         &data_word(0x00000000, 0x10400000, 0x10001008, 0x00400008);
371         &data_word(0x10401000, 0x10000008, 0x10000000, 0x00001008);
372         &data_word(0x10000008, 0x00401008, 0x00400000, 0x10000000);
373         &data_word(0x10400008, 0x00401000, 0x00001000, 0x00000008);
374         &data_word(0x00401000, 0x10001008, 0x10400000, 0x00001000);
375         &data_word(0x00001008, 0x00000000, 0x00400008, 0x10401000);
376         &data_word(0x10001000, 0x10400008, 0x10401008, 0x00400000);
377         &data_word(0x10400008, 0x00001008, 0x00400000, 0x10000008);
378         &data_word(0x00401000, 0x10001000, 0x00000008, 0x10400000);
379         &data_word(0x10001008, 0x00000000, 0x00001000, 0x00400008);
380         &data_word(0x00000000, 0x10400008, 0x10401000, 0x00001000);
381         &data_word(0x10000000, 0x10401008, 0x00401008, 0x00400000);
382         &data_word(0x10401008, 0x00000008, 0x10001000, 0x00401008);
383         &data_word(0x00400008, 0x00401000, 0x10400000, 0x10001008);
384         &data_word(0x00001008, 0x10000000, 0x10000008, 0x10401000);
385         # nibble 4
386         &data_word(0x08000000, 0x00010000, 0x00000400, 0x08010420);
387         &data_word(0x08010020, 0x08000400, 0x00010420, 0x08010000);
388         &data_word(0x00010000, 0x00000020, 0x08000020, 0x00010400);
389         &data_word(0x08000420, 0x08010020, 0x08010400, 0x00000420);
390         &data_word(0x00010400, 0x08000000, 0x00010020, 0x00000420);
391         &data_word(0x08000400, 0x00010420, 0x00000000, 0x08000020);
```

```
392         &data_word(0x00000020, 0x08000420, 0x08010420, 0x00010020);
393         &data_word(0x08010000, 0x00000400, 0x00000420, 0x08010400);
394         &data_word(0x08010400, 0x08000420, 0x00010020, 0x08010000);
395         &data_word(0x00010000, 0x00000020, 0x08000020, 0x08000400);
396         &data_word(0x08000000, 0x00010400, 0x08010420, 0x00000000);
397         &data_word(0x00010420, 0x08000000, 0x00000400, 0x00010020);
398         &data_word(0x08000420, 0x00000400, 0x00000000, 0x08010420);
399         &data_word(0x08010020, 0x08010400, 0x00000420, 0x00010000);
400         &data_word(0x00010400, 0x08010020, 0x08000400, 0x00000420);
401         &data_word(0x00000020, 0x00010420, 0x08010000, 0x08000020);
402         # nibble 5
403         &data_word(0x80000040, 0x00200040, 0x00000000, 0x80202000);
404         &data_word(0x00200040, 0x00002000, 0x80002040, 0x00200000);
405         &data_word(0x00002040, 0x80202040, 0x00202000, 0x80000000);
406         &data_word(0x80002000, 0x80000040, 0x80200000, 0x00202040);
407         &data_word(0x00200000, 0x80002040, 0x80200040, 0x00000000);
408         &data_word(0x00002000, 0x00000040, 0x80202000, 0x80200040);
409         &data_word(0x80202040, 0x80200000, 0x80000000, 0x00002040);
410         &data_word(0x00000040, 0x00202000, 0x00202040, 0x80002000);
411         &data_word(0x00002040, 0x80000000, 0x80002000, 0x00202040);
412         &data_word(0x80202000, 0x00200040, 0x00000000, 0x80002000);
413         &data_word(0x80000000, 0x00002000, 0x80200040, 0x00200000);
414         &data_word(0x00200040, 0x80202040, 0x00202000, 0x00000040);
415         &data_word(0x80202040, 0x00202000, 0x00200000, 0x80002040);
416         &data_word(0x80000040, 0x80200000, 0x00202040, 0x00000000);
417         &data_word(0x00002000, 0x80000040, 0x80002040, 0x80202000);
418         &data_word(0x80200000, 0x00002040, 0x00000040, 0x80200040);
419         # nibble 6
420         &data_word(0x00004000, 0x00000200, 0x01000200, 0x01000004);
421         &data_word(0x01004204, 0x00004004, 0x00004200, 0x00000000);
422         &data_word(0x01000000, 0x01000204, 0x00000204, 0x01004000);
423         &data_word(0x00000004, 0x01004200, 0x01004000, 0x00000204);
424         &data_word(0x01000204, 0x00004000, 0x00004004, 0x01004204);
425         &data_word(0x00000000, 0x01000200, 0x01000004, 0x00004200);
426         &data_word(0x01004004, 0x00004204, 0x01004200, 0x00000004);
427         &data_word(0x00004204, 0x01004004, 0x00000200, 0x01000000);
428         &data_word(0x00004204, 0x01004000, 0x01004004, 0x00000204);
429         &data_word(0x00004000, 0x00000200, 0x01000000, 0x01004004);
430         &data_word(0x01000204, 0x00004204, 0x00004200, 0x00000000);
431         &data_word(0x00000200, 0x01000004, 0x00000004, 0x01000200);
432         &data_word(0x00000000, 0x01000204, 0x01000200, 0x00004200);
433         &data_word(0x00000204, 0x00004000, 0x01004204, 0x01000000);
434         &data_word(0x01004200, 0x00000004, 0x00004004, 0x01004204);
435         &data_word(0x01000004, 0x01004200, 0x01004000, 0x00004004);
436         # nibble 7
437         &data_word(0x20800080, 0x20820000, 0x00020080, 0x00000000);
438         &data_word(0x20020000, 0x00800080, 0x20800000, 0x20820080);
439         &data_word(0x00000080, 0x20000000, 0x00820000, 0x00020080);
440         &data_word(0x00820080, 0x20020080, 0x20000080, 0x20800000);
441         &data_word(0x00020000, 0x00820080, 0x00800080, 0x20020000);
442         &data_word(0x20820080, 0x20000080, 0x00000000, 0x00820000);
443         &data_word(0x20000000, 0x00800000, 0x20020080, 0x20800080);
444         &data_word(0x00800000, 0x00020000, 0x20820000, 0x00000080);
445         &data_word(0x00800000, 0x00020000, 0x20000080, 0x20820080);
446         &data_word(0x00020080, 0x20000000, 0x00000000, 0x00820000);
447         &data_word(0x20800080, 0x20020080, 0x20020000, 0x00800080);
448         &data_word(0x20820000, 0x00000080, 0x00800080, 0x20020000);
449         &data_word(0x20820080, 0x00800000, 0x20800000, 0x20000080);
450         &data_word(0x00820000, 0x00020080, 0x20020080, 0x20800000);
451         &data_word(0x00000080, 0x20820000, 0x00820080, 0x00000000);
452         &data_word(0x20000000, 0x20800080, 0x00020000, 0x00820080);
453         }
454 #endif /* ! codereview */
```

```
**********************************************************
    1372 Wed Aug 13 19:53:07 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/desboth.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#!/usr/local/bin/perl

$L="edi";
$R="esi";

sub DES_encrypt3
        {
        local($name,$enc)=@_;

        &function_begin_B($name,"");
        &push("ebx");
        &mov("ebx",&wparam(0));

        &push("ebp");
        &push("esi");

        &push("edi");

        &comment("");
        &comment("Load the data words");
        &mov($L,&DWP(0,"ebx","",0));
        &mov($R,&DWP(4,"ebx","",0));
        &stack_push(3);

        &comment("");
        &comment("IP");
        &IP_new($L,$R,"edx",0);

        # put them back

        if ($enc)
                {
                &mov(&DWP(4,"ebx","",0),$R);
                 &mov("eax",&wparam(1));
                &mov(&DWP(0,"ebx","",0),"edx");
                 &mov("edi",&wparam(2));
                 &mov("esi",&wparam(3));
                }
        else
                {
                &mov(&DWP(4,"ebx","",0),$R);
                 &mov("esi",&wparam(1));
                &mov(&DWP(0,"ebx","",0),"edx");
                 &mov("edi",&wparam(2));
                 &mov("eax",&wparam(3));
                }
        &mov(&swtmp(2), (DWC(($enc)?"1":"0")));
        &mov(&swtmp(1), "eax");
        &mov(&swtmp(0), "ebx");
        &call("DES_encrypt2");
        &mov(&swtmp(2), (DWC(($enc)?"0":"1")));
        &mov(&swtmp(1), "edi");
        &mov(&swtmp(0), "ebx");
        &call("DES_encrypt2");
        &mov(&swtmp(2), (DWC(($enc)?"1":"0")));
        &mov(&swtmp(1), "esi");
        &mov(&swtmp(0), "ebx");
        &call("DES_encrypt2");

        &stack_pop(3);
        &mov($L,&DWP(0,"ebx","",0));
```



```
        &mov($R,&DWP(4,"ebx","",0));

        &comment("");
        &comment("FP");
        &FP_new($L,$R,"eax",0);

        &mov(&DWP(0,"ebx","",0),"eax");
        &mov(&DWP(4,"ebx","",0),$R);

        &pop("edi");
        &pop("esi");
        &pop("ebp");
        &pop("ebx");
        &ret();
        &function_end_B($name);
        }
#endif /* ! codereview */
```

```
**********************************************************
   39580 Wed Aug 13 19:53:07 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/ghash-x86.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
  1 #!/usr/bin/env perl
  2 #
  3 # ====================================================================
  4 # Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
  5 # project. The module is, however, dual licensed under OpenSSL and
  6 # CRYPTOGAMS licenses depending on where you obtain it. For further
  7 # details see http://www.openssl.org/~appro/cryptogams/.
  8 # ====================================================================
  9 #
 10 # March, May, June 2010
 11 #
 12 # The module implements "4-bit" GCM GHASH function and underlying
 13 # single multiplication operation in GF(2^128). "4-bit" means that it
 14 # uses 256 bytes per-key table [+64/128 bytes fixed table]. It has two
 15 # code paths: vanilla x86 and vanilla MMX. Former will be executed on
 16 # 486 and Pentium, latter on all others. MMX GHASH features so called
 17 # "528B" variant of "4-bit" method utilizing additional 256+16 bytes
 18 # of per-key storage [+512 bytes shared table]. Performance results
 19 # are for streamed GHASH subroutine and are expressed in cycles per
 20 # processed byte, less is better:
 21 #
 22 #               gcc 2.95.3(*)   MMX assembler   x86 assembler
 23 #
 24 # Pentium       105/111(**)     -               50
 25 # PIII          68 /75          12.2            24
 26 # P4            125/125         17.8            84(***)
 27 # Opteron       66 /70          10.1            30
 28 # Core2         54 /67          8.4             18
 29 #
 30 # (*)   gcc 3.4.x was observed to generate few percent slower code,
 31 #       which is one of reasons why 2.95.3 results were chosen,
 32 #       another reason is lack of 3.4.x results for older CPUs;
 33 #       comparison with MMX results is not completely fair, because C
 34 #       results are for vanilla "256B" implementation, while
 35 #       assembler results are for "528B";-)
 36 # (**)  second number is result for code compiled with -fPIC flag,
 37 #       which is actually more relevant, because assembler code is
 38 #       position-independent;
 39 # (***) see comment in non-MMX routine for further details;
 40 #
 41 # To summarize, it's >2-5 times faster than gcc-generated code. To
 42 # anchor it to something else SHA1 assembler processes one byte in
 43 # 11-13 cycles on contemporary x86 cores. As for choice of MMX in
 44 # particular, see comment at the end of the file...

 46 # May 2010
 47 #
 48 # Add PCLMULQDQ version performing at 2.10 cycles per processed byte.
 49 # The question is how close is it to theoretical limit? The pclmulqdq
 50 # instruction latency appears to be 14 cycles and there can't be more
 51 # than 2 of them executing at any given time. This means that single
 52 # Karatsuba multiplication would take 28 cycles *plus* few cycles for
 53 # pre- and post-processing. Then multiplication has to be followed by
 54 # modulo-reduction. Given that aggregated reduction method [see
 55 # "Carry-less Multiplication and Its Usage for Computing the GCM Mode"
 56 # white paper by Intel] allows you to perform reduction only once in
 57 # a while we can assume that asymptotic performance can be estimated
 58 # as (28+Tmod/Naggr)/16, where Tmod is time to perform reduction
 59 # and Naggr is the aggregation factor.
 60 #
 61 # Before we proceed to this implementation let's have closer look at
```



```
 62 # the best-performing code suggested by Intel in their white paper.
 63 # By tracing inter-register dependencies Tmod is estimated as ~19
 64 # cycles and Naggr chosen by Intel is 4, resulting in 2.05 cycles per
 65 # processed byte. As implied, this is quite optimistic estimate,
 66 # because it does not account for Karatsuba pre- and post-processing,
 67 # which for a single multiplication is ~5 cycles. Unfortunately Intel
 68 # does not provide performance data for GHASH alone. But benchmarking
 69 # AES_GCM_encrypt ripped out of Fig. 15 of the white paper with aadt
 70 # alone resulted in 2.46 cycles per byte of out 16KB buffer. Note that
 71 # the result accounts even for pre-computing of degrees of the hash
 72 # key H, but its portion is negligible at 16KB buffer size.
 73 #
 74 # Moving on to the implementation in question. Tmod is estimated as
 75 # ~13 cycles and Naggr is 2, giving asymptotic performance of ...
 76 # 2.16. How is it possible that measured performance is better than
 77 # optimistic theoretical estimate? There is one thing Intel failed
 78 # to recognize. By serializing GHASH with CTR in same subroutine
 79 # former's performance is really limited to above (Tmul + Tmod/Naggr)
 80 # equation. But if GHASH procedure is detached, the modulo-reduction
 81 # can be interleaved with Naggr-1 multiplications at instruction level
 82 # and under ideal conditions even disappear from the equation. So that
 83 # optimistic theoretical estimate for this implementation is ...
 84 # 28/16=1.75, and not 2.16. Well, it's probably way too optimistic,
 85 # at least for such small Naggr. I'd argue that (28+Tproc/Naggr),
 86 # where Tproc is time required for Karatsuba pre- and post-processing,
 87 # is more realistic estimate. In this case it gives ... 1.91 cycles.
 88 # Or in other words, depending on how well we can interleave reduction
 89 # and one of the two multiplications the performance should be betwen
 90 # 1.91 and 2.16. As already mentioned, this implementation processes
 91 # one byte out of 8KB buffer in 2.10 cycles, while x86_64 counterpart
 92 # - in 2.02. x86_64 performance is better, because larger register
 93 # bank allows to interleave reduction and multiplication better.
 94 #
 95 # Does it make sense to increase Naggr? To start with it's virtually
 96 # impossible in 32-bit mode, because of limited register bank
 97 # capacity. Otherwise improvement has to be weighed agiainst slower
 98 # setup, as well as code size and complexity increase. As even
 99 # optimistic estimate doesn't promise 30% performance improvement,
100 # there are currently no plans to increase Naggr.
101 #
102 # Special thanks to David Woodhouse <dwmw2@infradead.org> for
103 # providing access to a Westmere-based system on behalf of Intel
104 # Open Source Technology Centre.

106 # January 2010
107 #
108 # Tweaked to optimize transitions between integer and FP operations
109 # on same XMM register, PCLMULQDQ subroutine was measured to process
110 # one byte in 2.07 cycles on Sandy Bridge, and in 2.12 - on Westmere.
111 # The minor regression on Westmere is outweighed by ~15% improvement
112 # on Sandy Bridge. Strangely enough attempt to modify 64-bit code in
113 # similar manner resulted in almost 20% degradation on Sandy Bridge,
114 # where original 64-bit code processes one byte in 1.95 cycles.

116 $0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
117 push(@INC,"${dir}","${dir}../../perlasm");
118 require "x86asm.pl";

120 &asm_init($ARGV[0],"ghash-x86.pl",$x86only = $ARGV[$#ARGV] eq "386");

122 $sse2=0;
123 for (@ARGV) { $sse2=1 if (/-DOPENSSL_IA32_SSE2/); }

125 ($Zhh,$Zhl,$Zlh,$Zll) = ("ebp","edx","ecx","ebx");
126 $inp  = "edi";
127 $Htbl = "esi";
```

```
128 $unroll = 0;    # Affects x86 loop. Folded loop performs ~7% worse
129                 # than unrolled, which has to be weighted against
130                 # 2.5x x86-specific code size reduction.

132 sub x86_loop {
133     my $off = shift;
134     my $rem = "eax";

136         &mov    ($Zhh,&DWP(4,$Htbl,$Zll));
137         &mov    ($Zhl,&DWP(0,$Htbl,$Zll));
138         &mov    ($Zlh,&DWP(12,$Htbl,$Zll));
139         &mov    ($Zll,&DWP(8,$Htbl,$Zll));
140         &xor    ($rem,$rem);    # avoid partial register stalls on PIII

142         # shrd practically kills P4, 2.5x deterioration, but P4 has
143         # MMX code-path to execute. shrd runs tad faster [than twice
144         # the shifts, move's and or's] on pre-MMX Pentium (as well as
145         # PIII and Core2), *but* minimizes code size, spares register
146         # and thus allows to fold the loop...
147         if (!$unroll) {
148         my $cnt = $inp;
149         &mov    ($cnt,15);
150         &jmp    (&label("x86_loop"));
151         &set_label("x86_loop",16);
152             for($i=1;$i<=2;$i++) {
153                 &mov    (&LB($rem),&LB($Zll));
154                 &shrd   ($Zll,$Zlh,4);
155                 &and    (&LB($rem),0xf);
156                 &shrd   ($Zlh,$Zhl,4);
157                 &shrd   ($Zhl,$Zhh,4);
158                 &shr    ($Zhh,4);
159                 &xor    ($Zhh,&DWP($off+16,"esp",$rem,4));

161                 &mov    (&LB($rem),&BP($off,"esp",$cnt));
162                 if ($i&1) {
163                     &and    (&LB($rem),0xf0);
164                 } else {
165                     &shl    (&LB($rem),4);
166                 }

168                 &xor    ($Zll,&DWP(8,$Htbl,$rem));
169                 &xor    ($Zlh,&DWP(12,$Htbl,$rem));
170                 &xor    ($Zhl,&DWP(0,$Htbl,$rem));
171                 &xor    ($Zhh,&DWP(4,$Htbl,$rem));

173                 if ($i&1) {
174                     &dec    ($cnt);
175                     &js     (&label("x86_break"));
176                 } else {
177                     &jmp    (&label("x86_loop"));
178                 }
179             }
180         &set_label("x86_break",16);
181         } else {
182             for($i=1;$i<32;$i++) {
183                 &comment($i);
184                 &mov    (&LB($rem),&LB($Zll));
185                 &shrd   ($Zll,$Zlh,4);
186                 &and    (&LB($rem),0xf);
187                 &shrd   ($Zlh,$Zhl,4);
188                 &shrd   ($Zhl,$Zhh,4);
189                 &shr    ($Zhh,4);
190                 &xor    ($Zhh,&DWP($off+16,"esp",$rem,4));

192                 if ($i&1) {
193                     &mov    (&LB($rem),&BP($off+15-($i>>1),"esp"));
```

```
194                         &and    (&LB($rem),0xf0);
195                 } else {
196                         &mov    (&LB($rem),&BP($off+15-($i>>1),"esp"));
197                         &shl    (&LB($rem),4);
198                 }

200                 &xor    ($Zll,&DWP(8,$Htbl,$rem));
201                 &xor    ($Zlh,&DWP(12,$Htbl,$rem));
202                 &xor    ($Zhl,&DWP(0,$Htbl,$rem));
203                 &xor    ($Zhh,&DWP(4,$Htbl,$rem));
204             }
205         }
206         &bswap  ($Zll);
207         &bswap  ($Zlh);
208         &bswap  ($Zhl);
209         if (!$x86only) {
210                 &bswap  ($Zhh);
211         } else {
212                 &mov    ("eax",$Zhh);
213                 &bswap  ("eax");
214                 &mov    ($Zhh,"eax");
215         }
216 }

218 if ($unroll) {
219     &function_begin_B("_x86_gmult_4bit_inner");
220         &x86_loop(4);
221         &ret    ();
222     &function_end_B("_x86_gmult_4bit_inner");
223 }

225 sub deposit_rem_4bit {
226     my $bias = shift;

228         &mov    (&DWP($bias+0, "esp"),0x0000<<16);
229         &mov    (&DWP($bias+4, "esp"),0x1C20<<16);
230         &mov    (&DWP($bias+8, "esp"),0x3840<<16);
231         &mov    (&DWP($bias+12,"esp"),0x2460<<16);
232         &mov    (&DWP($bias+16,"esp"),0x7080<<16);
233         &mov    (&DWP($bias+20,"esp"),0x6CA0<<16);
234         &mov    (&DWP($bias+24,"esp"),0x48C0<<16);
235         &mov    (&DWP($bias+28,"esp"),0x54E0<<16);
236         &mov    (&DWP($bias+32,"esp"),0xE100<<16);
237         &mov    (&DWP($bias+36,"esp"),0xFD20<<16);
238         &mov    (&DWP($bias+40,"esp"),0xD940<<16);
239         &mov    (&DWP($bias+44,"esp"),0xC560<<16);
240         &mov    (&DWP($bias+48,"esp"),0x9180<<16);
241         &mov    (&DWP($bias+52,"esp"),0x8DA0<<16);
242         &mov    (&DWP($bias+56,"esp"),0xA9C0<<16);
243         &mov    (&DWP($bias+60,"esp"),0xB5E0<<16);
244 }
```



```
245 $suffix = $x86only ? "" : "_x86";

247 &function_begin("gcm_gmult_4bit".$suffix);
248         &stack_push(16+4+1);                    # +1 for stack alignment
249         &mov    ($inp,&wparam(0));              # load Xi
250         &mov    ($Htbl,&wparam(1));             # load Htable

252         &mov    ($Zhh,&DWP(0,$inp));            # load Xi[16]
253         &mov    ($Zhl,&DWP(4,$inp));
254         &mov    ($Zlh,&DWP(8,$inp));
255         &mov    ($Zll,&DWP(12,$inp));

257         &deposit_rem_4bit(16);

259         &mov    (&DWP(0,"esp"),$Zhh);           # copy Xi[16] on stack
260         &mov    (&DWP(4,"esp"),$Zhl);
261         &mov    (&DWP(8,"esp"),$Zlh);
262         &mov    (&DWP(12,"esp"),$Zll);
263         &shr    ($Zll,20);
264         &and    ($Zll,0xf0);

266         if ($unroll) {
267                 &call   ("_x86_gmult_4bit_inner");
268         } else {
269                 &x86_loop(0);
270                 &mov    ($inp,&wparam(0));
271         }

273         &mov    (&DWP(12,$inp),$Zll);
274         &mov    (&DWP(8,$inp),$Zlh);
275         &mov    (&DWP(4,$inp),$Zhl);
276         &mov    (&DWP(0,$inp),$Zhh);
277         &stack_pop(16+4+1);
278 &function_end("gcm_gmult_4bit".$suffix);

280 &function_begin("gcm_ghash_4bit".$suffix);
281         &stack_push(16+4+1);                    # +1 for 64-bit alignment
282         &mov    ($Zll,&wparam(0));              # load Xi
283         &mov    ($Htbl,&wparam(1));             # load Htable
284         &mov    ($inp,&wparam(2));              # load in
285         &mov    ("ecx",&wparam(3));             # load len
286         &add    ("ecx",$inp);
287         &mov    (&wparam(3),"ecx");

289         &mov    ($Zhh,&DWP(0,$Zll));            # load Xi[16]
290         &mov    ($Zhl,&DWP(4,$Zll));
291         &mov    ($Zlh,&DWP(8,$Zll));
292         &mov    ($Zll,&DWP(12,$Zll));

294         &deposit_rem_4bit(16);

296     &set_label("x86_outer_loop",16);
297         &xor    ($Zll,&DWP(12,$inp));           # xor with input
298         &xor    ($Zlh,&DWP(8,$inp));
299         &xor    ($Zhl,&DWP(4,$inp));
300         &xor    ($Zhh,&DWP(0,$inp));
301         &mov    (&DWP(12,"esp"),$Zll);          # dump it on stack
302         &mov    (&DWP(8,"esp"),$Zlh);
303         &mov    (&DWP(4,"esp"),$Zhl);
304         &mov    (&DWP(0,"esp"),$Zhh);

306         &shr    ($Zll,20);
307         &and    ($Zll,0xf0);

309         if ($unroll) {
310                 &call   ("_x86_gmult_4bit_inner");
```

```
311         } else {
312                 &x86_loop(0);
313                 &mov    ($inp,&wparam(2));
314         }
315         &lea    ($inp,&DWP(16,$inp));
316         &cmp    ($inp,&wparam(3));
317         &mov    (&wparam(2),$inp)       if (!$unroll);
318         &jb     (&label("x86_outer_loop"));

320         &mov    ($inp,&wparam(0));      # load Xi
321         &mov    (&DWP(12,$inp),$Zll);
322         &mov    (&DWP(8,$inp),$Zlh);
323         &mov    (&DWP(4,$inp),$Zhl);
324         &mov    (&DWP(0,$inp),$Zhh);
325         &stack_pop(16+4+1);
326 &function_end("gcm_ghash_4bit".$suffix);
```



```
327 if (!$x86only) {{{

329 &static_label("rem_4bit");

331 if (!$sse2) {{  # pure-MMX "May" version...

333 $S=12;          # shift factor for rem_4bit

335 &function_begin_B("_mmx_gmult_4bit_inner");
336 # MMX version performs 3.5 times better on P4 (see comment in non-MMX
337 # routine for further details), 100% better on Opteron, ~70% better
338 # on Core2 and PIII... In other words effort is considered to be well
339 # spent... Since initial release the loop was unrolled in order to
340 # "liberate" register previously used as loop counter. Instead it's
341 # used to optimize critical path in 'Z.hi ^= rem_4bit[Z.lo&0xf]'.
342 # The path involves move of Z.lo from MMX to integer register,
343 # effective address calculation and finally merge of value to Z.hi.
344 # Reference to rem_4bit is scheduled so late that I had to >>4
345 # rem_4bit elements. This resulted in 20-45% procent improvement
346 # on contemporary µ-archs.
347 {
348     my $cnt;
349     my $rem_4bit = "eax";
350     my @rem = ($Zhh,$Zll);
351     my $nhi = $Zhl;
352     my $nlo = $Zlh;

354     my ($Zlo,$Zhi) = ("mm0","mm1");
355     my $tmp = "mm2";

357         &xor    ($nlo,$nlo);    # avoid partial register stalls on PIII
358         &mov    ($nhi,$Zll);
359         &mov    (&LB($nlo),&LB($nhi));
360         &shl    (&LB($nlo),4);
361         &and    ($nhi,0xf0);
362         &movq   ($Zlo,&QWP(8,$Htbl,$nlo));
363         &movq   ($Zhi,&QWP(0,$Htbl,$nlo));
364         &movd   ($rem[0],$Zlo);

366         for ($cnt=28;$cnt>=-2;$cnt--) {
367             my $odd = $cnt&1;
368             my $nix = $odd ? $nlo : $nhi;

370                 &shl    (&LB($nlo),4)                   if ($odd);
371                 &psrlq  ($Zlo,4);
372                 &movq   ($tmp,$Zhi);
373                 &psrlq  ($Zhi,4);
374                 &pxor   ($Zlo,&QWP(8,$Htbl,$nix));
375                 &mov    (&LB($nlo),&BP($cnt/2,$inp))    if (!$odd && $cnt>=0);
376                 &psllq  ($tmp,60);
377                 &and    ($nhi,0xf0)                     if ($odd);
378                 &pxor   ($Zhi,&QWP(0,$rem_4bit,$rem[1],8)) if ($cnt<28);
379                 &and    ($rem[0],0xf);
380                 &pxor   ($Zhi,&QWP(0,$Htbl,$nix));
381                 &mov    ($nhi,$nlo)                     if (!$odd && $cnt>=0);
382                 &movd   ($rem[1],$Zlo);
383                 &pxor   ($Zlo,$tmp);

385                 push    (@rem,shift(@rem));             # "rotate" registers
386         }

388         &mov    ($inp,&DWP(4,$rem_4bit,$rem[1],8));     # last rem_4bit[rem]

390         &psrlq  ($Zlo,32);      # lower part of Zlo is already there
391         &movd   ($Zhl,$Zhi);
392         &psrlq  ($Zhi,32);
```

```
393         &movd   ($Zlh,$Zlo);
394         &movd   ($Zhh,$Zhi);
395         &shl    ($inp,4);       # compensate for rem_4bit[i] being >>4

397         &bswap  ($Zll);
398         &bswap  ($Zhl);
399         &bswap  ($Zlh);
400         &xor    ($Zhh,$inp);
401         &bswap  ($Zhh);

403         &ret    ();
404 }
405 &function_end_B("_mmx_gmult_4bit_inner");

407 &function_begin("gcm_gmult_4bit_mmx");
408         &mov    ($inp,&wparam(0));      # load Xi
409         &mov    ($Htbl,&wparam(1));     # load Htable

411         &call   (&label("pic_point"));
412         &set_label("pic_point");
413         &blindpop("eax");
414         &lea    ("eax",&DWP(&label("rem_4bit")."-".&label("pic_point"),"eax"));

416         &movz   ($Zll,&BP(15,$inp));

418         &call   ("_mmx_gmult_4bit_inner");

420         &mov    ($inp,&wparam(0));      # load Xi
421         &emms   ();
422         &mov    (&DWP(12,$inp),$Zll);
423         &mov    (&DWP(4,$inp),$Zhl);
424         &mov    (&DWP(8,$inp),$Zlh);
425         &mov    (&DWP(0,$inp),$Zhh);
426 &function_end("gcm_gmult_4bit_mmx");
```



```
427 # Streamed version performs 20% better on P4, 7% on Opteron,
428 # 10% on Core2 and PIII...
429 &function_begin("gcm_ghash_4bit_mmx");
430         &mov    ($Zhh,&wparam(0));      # load Xi
431         &mov    ($Htbl,&wparam(1));     # load Htable
432         &mov    ($inp,&wparam(2));      # load in
433         &mov    ($Zlh,&wparam(3));      # load len

435         &call   (&label("pic_point"));
436         &set_label("pic_point");
437         &blindpop("eax");
438         &lea    ("eax",&DWP(&label("rem_4bit")."-".&label("pic_point"),"eax"));

440         &add    ($Zlh,$inp);
441         &mov    (&wparam(3),$Zlh);      # len to point at the end of input
442         &stack_push(4+1);               # +1 for stack alignment

444         &mov    ($Zll,&DWP(12,$Zhh));   # load Xi[16]
445         &mov    ($Zhl,&DWP(4,$Zhh));
446         &mov    ($Zlh,&DWP(8,$Zhh));
447         &mov    ($Zhh,&DWP(0,$Zhh));
448         &jmp    (&label("mmx_outer_loop"));

450     &set_label("mmx_outer_loop",16);
451         &xor    ($Zll,&DWP(12,$inp));
452         &xor    ($Zhl,&DWP(4,$inp));
453         &xor    ($Zlh,&DWP(8,$inp));
454         &xor    ($Zhh,&DWP(0,$inp));
455         &mov    (&wparam(2),$inp);
456         &mov    (&DWP(12,"esp"),$Zll);
457         &mov    (&DWP(4,"esp"),$Zhl);
458         &mov    (&DWP(8,"esp"),$Zlh);
459         &mov    (&DWP(0,"esp"),$Zhh);

461         &mov    ($inp,"esp");
462         &shr    ($Zll,24);

464         &call   ("_mmx_gmult_4bit_inner");

466         &mov    ($inp,&wparam(2));
467         &lea    ($inp,&DWP(16,$inp));
468         &cmp    ($inp,&wparam(3));
469         &jb     (&label("mmx_outer_loop"));

471         &mov    ($inp,&wparam(0));      # load Xi
472         &emms   ();
473         &mov    (&DWP(12,$inp),$Zll);
474         &mov    (&DWP(4,$inp),$Zhl);
475         &mov    (&DWP(8,$inp),$Zlh);
476         &mov    (&DWP(0,$inp),$Zhh);

478         &stack_pop(4+1);
479 &function_end("gcm_ghash_4bit_mmx");
```

```
480 }} else {{      # "June" MMX version...
481                 # ... has slower "April" gcm_gmult_4bit_mmx with folded
482                 # loop. This is done to conserve code size...
483 $S=16;          # shift factor for rem_4bit

485 sub mmx_loop() {
486 # MMX version performs 2.8 times better on P4 (see comment in non-MMX
487 # routine for further details), 40% better on Opteron and Core2, 50%
488 # better on PIII... In other words effort is considered to be well
489 # spent...
490     my $inp = shift;
491     my $rem_4bit = shift;
492     my $cnt = $Zhh;
493     my $nhi = $Zhl;
494     my $nlo = $Zlh;
495     my $rem = $Zll;

497     my ($Zlo,$Zhi) = ("mm0","mm1");
498     my $tmp = "mm2";

500         &xor    ($nlo,$nlo);    # avoid partial register stalls on PIII
501         &mov    ($nhi,$Zll);
502         &mov    (&LB($nlo),&LB($nhi));
503         &mov    ($cnt,14);
504         &shl    (&LB($nlo),4);
505         &and    ($nhi,0xf0);
506         &movq   ($Zlo,&QWP(8,$Htbl,$nlo));
507         &movq   ($Zhi,&QWP(0,$Htbl,$nlo));
508         &movd   ($rem,$Zlo);
509         &jmp    (&label("mmx_loop"));

511     &set_label("mmx_loop",16);
512         &psrlq  ($Zlo,4);
513         &and    ($rem,0xf);
514         &movq   ($tmp,$Zhi);
515         &psrlq  ($Zhi,4);
516         &pxor   ($Zlo,&QWP(8,$Htbl,$nhi));
517         &mov    (&LB($nlo),&BP(0,$inp,$cnt));
518         &psllq  ($tmp,60);
519         &pxor   ($Zhi,&QWP(0,$rem_4bit,$rem,8));
520         &dec    ($cnt);
521         &movd   ($rem,$Zlo);
522         &pxor   ($Zhi,&QWP(0,$Htbl,$nhi));
523         &mov    ($nhi,$nlo);
524         &pxor   ($Zlo,$tmp);
525         &js     (&label("mmx_break"));

527         &shl    (&LB($nlo),4);
528         &and    ($rem,0xf);
529         &psrlq  ($Zlo,4);
530         &and    ($nhi,0xf0);
531         &movq   ($tmp,$Zhi);
532         &psrlq  ($Zhi,4);
533         &pxor   ($Zlo,&QWP(8,$Htbl,$nlo));
534         &psllq  ($tmp,60);
535         &pxor   ($Zhi,&QWP(0,$rem_4bit,$rem,8));
536         &movd   ($rem,$Zlo);
537         &pxor   ($Zhi,&QWP(0,$Htbl,$nlo));
538         &pxor   ($Zlo,$tmp);
539         &jmp    (&label("mmx_loop"));

541     &set_label("mmx_break",16);
542         &shl    (&LB($nlo),4);
543         &and    ($rem,0xf);
544         &psrlq  ($Zlo,4);
545         &and    ($nhi,0xf0);
```



```
546         &movq   ($tmp,$Zhi);
547         &psrlq  ($Zhi,4);
548         &pxor   ($Zlo,&QWP(8,$Htbl,$nlo));
549         &psllq  ($tmp,60);
550         &pxor   ($Zhi,&QWP(0,$rem_4bit,$rem,8));
551         &movd   ($rem,$Zlo);
552         &pxor   ($Zhi,&QWP(0,$Htbl,$nlo));
553         &pxor   ($Zlo,$tmp);

555         &psrlq  ($Zlo,4);
556         &and    ($rem,0xf);
557         &movq   ($tmp,$Zhi);
558         &psrlq  ($Zhi,4);
559         &pxor   ($Zlo,&QWP(8,$Htbl,$nhi));
560         &psllq  ($tmp,60);
561         &pxor   ($Zhi,&QWP(0,$rem_4bit,$rem,8));
562         &movd   ($rem,$Zlo);
563         &pxor   ($Zhi,&QWP(0,$Htbl,$nhi));
564         &pxor   ($Zlo,$tmp);

566         &psrlq  ($Zlo,32);      # lower part of Zlo is already there
567         &movd   ($Zhl,$Zhi);
568         &psrlq  ($Zhi,32);
569         &movd   ($Zlh,$Zlo);
570         &movd   ($Zhh,$Zhi);

572         &bswap  ($Zll);
573         &bswap  ($Zhl);
574         &bswap  ($Zlh);
575         &bswap  ($Zhh);
576 }

578 &function_begin("gcm_gmult_4bit_mmx");
579         &mov    ($inp,&wparam(0));      # load Xi
580         &mov    ($Htbl,&wparam(1));     # load Htable

582         &call   (&label("pic_point"));
583         &set_label("pic_point");
584         &blindpop("eax");
585         &lea    ("eax",&DWP(&label("rem_4bit")."-".&label("pic_point"),"eax"));

587         &movz   ($Zll,&BP(15,$inp));

589         &mmx_loop($inp,"eax");

591         &emms   ();
592         &mov    (&DWP(12,$inp),$Zll);
593         &mov    (&DWP(4,$inp),$Zhl);
594         &mov    (&DWP(8,$inp),$Zlh);
595         &mov    (&DWP(0,$inp),$Zhh);
596 &function_end("gcm_gmult_4bit_mmx");
```

```
597 ################################################################################
598 # Below subroutine is "528B" variant of "4-bit" GCM GHASH function
599 # (see gcm128.c for details). It provides further 20-40% performance
600 # improvement over above mentioned "May" version.

602 &static_label("rem_8bit");

604 &function_begin("gcm_ghash_4bit_mmx");
605 { my ($Zlo,$Zhi) = ("mm7","mm6");
606   my $rem_8bit = "esi";
607   my $Htbl = "ebx";

609     # parameter block
610     &mov        ("eax",&wparam(0));             # Xi
611     &mov        ("ebx",&wparam(1));             # Htable
612     &mov        ("ecx",&wparam(2));             # inp
613     &mov        ("edx",&wparam(3));             # len
614     &mov        ("ebp","esp");                  # original %esp
615     &call       (&label("pic_point"));
616     &set_label  ("pic_point");
617     &blindpop   ($rem_8bit);
618     &lea        ($rem_8bit,&DWP(&label("rem_8bit")."-".&label("pic_point"),$rem_

620     &sub        ("esp",512+16+16);              # allocate stack frame...
621     &and        ("esp",-64);                    # ...and align it
622     &sub        ("esp",16);                     # place for (u8)(H[]<<4)

624     &add        ("edx","ecx");                  # pointer to the end of input
625     &mov        (&DWP(528+16+0,"esp"),"eax");   # save Xi
626     &mov        (&DWP(528+16+8,"esp"),"edx");   # save inp+len
627     &mov        (&DWP(528+16+12,"esp"),"ebp");  # save original %esp

629     { my @lo  = ("mm0","mm1","mm2");
630       my @hi  = ("mm3","mm4","mm5");
631       my @tmp = ("mm6","mm7");
632       my ($off1,$off2,$i) = (0,0,);

634       &add      ($Htbl,128);                    # optimize for size
635       &lea      ("edi",&DWP(16+128,"esp"));
636       &lea      ("ebp",&DWP(16+256+128,"esp"));

638       # decompose Htable (low and high parts are kept separately),
639       # generate Htable[]>>4, (u8)(Htable[]<<4), save to stack...
640       for ($i=0;$i<18;$i++) {

642         &mov    ("edx",&DWP(16*$i+8-128,$Htbl))         if ($i<16);
643         &movq   ($lo[0],&QWP(16*$i+8-128,$Htbl))        if ($i<16);
644         &psllq  ($tmp[1],60)                            if ($i>1);
645         &movq   ($hi[0],&QWP(16*$i+0-128,$Htbl))        if ($i<16);
646         &por    ($lo[2],$tmp[1])                        if ($i>1);
647         &movq   (&QWP($off1-128,"edi"),$lo[1])          if ($i>0 && $i<17);
648         &psrlq  ($lo[1],4)                              if ($i>0 && $i<17);
649         &movq   (&QWP($off1,"edi"),$hi[1])              if ($i>0 && $i<17);
650         &movq   ($tmp[0],$hi[1])                        if ($i>0 && $i<17);
651         &movq   (&QWP($off2-128,"ebp"),$lo[2])          if ($i>1);
652         &psrlq  ($hi[1],4)                              if ($i>0 && $i<17);
653         &movq   (&QWP($off2,"ebp"),$hi[2])              if ($i>1);
654         &shl    ("edx",4)                               if ($i<16);
655         &mov    (&BP($i,"esp"),&LB("edx"))              if ($i<16);

657         unshift (@lo,pop(@lo));                 # "rotate" registers
658         unshift (@hi,pop(@hi));
659         unshift (@tmp,pop(@tmp));
660         $off1 += 8      if ($i>0);
661         $off2 += 8      if ($i>1);
662       }
```



```
663     }

665     &movq       ($Zhi,&QWP(0,"eax"));
666     &mov        ("ebx",&DWP(8,"eax"));
667     &mov        ("edx",&DWP(12,"eax"));         # load Xi

669 &set_label("outer",16);
670   { my $nlo = "eax";
671     my $dat = "edx";
672     my @nhi = ("edi","ebp");
673     my @rem = ("ebx","ecx");
674     my @red = ("mm0","mm1","mm2");
675     my $tmp = "mm3";

677     &xor        ($dat,&DWP(12,"ecx"));          # merge input data
678     &xor        ("ebx",&DWP(8,"ecx"));
679     &pxor       ($Zhi,&QWP(0,"ecx"));
680     &lea        ("ecx",&DWP(16,"ecx"));         # inp+=16
681     #&mov       (&DWP(528+12,"esp"),$dat);      # save inp^Xi
682     &mov        (&DWP(528+8,"esp"),"ebx");
683     &movq       (&QWP(528+0,"esp"),$Zhi);
684     &mov        (&DWP(528+16+4,"esp"),"ecx");   # save inp

686     &xor        ($nlo,$nlo);
687     &rol        ($dat,8);
688     &mov        (&LB($nlo),&LB($dat));
689     &mov        ($nhi[1],$nlo);
690     &and        (&LB($nlo),0x0f);
691     &shr        ($nhi[1],4);
692     &pxor       ($red[0],$red[0]);
693     &rol        ($dat,8);                       # next byte
694     &pxor       ($red[1],$red[1]);
695     &pxor       ($red[2],$red[2]);

697     # Just like in "May" verson modulo-schedule for critical path in
698     # 'Z.hi ^= rem_8bit[Z.lo&0xff^((u8)H[nhi]<<4)]<<48'. Final 'pxor'
699     # is scheduled so late that rem_8bit[] has to be shifted *right*
700     # by 16, which is why last argument to pinsrw is 2, which
701     # corresponds to <<32=<<48>>16...
702     for ($j=11,$i=0;$i<15;$i++) {

704       if ($i>0) {
705         &pxor   ($Zlo,&QWP(16,"esp",$nlo,8));           # Z^=H[nlo]
706         &rol    ($dat,8);                               # next byte
707         &pxor   ($Zhi,&QWP(16+128,"esp",$nlo,8));

709         &pxor   ($Zlo,$tmp);
710         &pxor   ($Zhi,&QWP(16+256+128,"esp",$nhi[0],8));
711         &xor    (&LB($rem[1]),&BP(0,"esp",$nhi[0]));    # rem^(H[nhi]<<4)
712       } else {
713         &movq   ($Zlo,&QWP(16,"esp",$nlo,8));
714         &movq   ($Zhi,&QWP(16+128,"esp",$nlo,8));
715       }

717         &mov    (&LB($nlo),&LB($dat));
718         &mov    ($dat,&DWP(528+$j,"esp"))               if (--$j%4==0);

720         &movd   ($rem[0],$Zlo);
721         &movz   ($rem[1],&LB($rem[1]))                  if ($i>0);
722         &psrlq  ($Zlo,8);                               # Z>>=8

724         &movq   ($tmp,$Zhi);
725         &mov    ($nhi[0],$nlo);
726         &psrlq  ($Zhi,8);

728         &pxor   ($Zlo,&QWP(16+256+0,"esp",$nhi[1],8));  # Z^=H[nhi]>>4
```

```
729         &and    (&LB($nlo),0x0f);
730         &psllq  ($tmp,56);

732         &pxor   ($Zhi,$red[1])                          if ($i>1);
733         &shr    ($nhi[0],4);
734         &pinsrw ($red[0],&WP(0,$rem_8bit,$rem[1],2),2)  if ($i>0);

736         unshift (@red,pop(@red));                       # "rotate" registers
737         unshift (@rem,pop(@rem));
738         unshift (@nhi,pop(@nhi));
739     }

741     &pxor       ($Zlo,&QWP(16,"esp",$nlo,8));           # Z^=H[nlo]
742     &pxor       ($Zhi,&QWP(16+128,"esp",$nlo,8));
743     &xor        (&LB($rem[1]),&BP(0,"esp",$nhi[0]));    # rem^(H[nhi]<<4)

745     &pxor       ($Zlo,$tmp);
746     &pxor       ($Zhi,&QWP(16+256+128,"esp",$nhi[0],8));
747     &movz       ($rem[1],&LB($rem[1]));

749     &pxor       ($red[2],$red[2]);                      # clear 2nd word
750     &psllq      ($red[1],4);

752     &movd       ($rem[0],$Zlo);
753     &psrlq      ($Zlo,4);                               # Z>>=4

755     &movq       ($tmp,$Zhi);
756     &psrlq      ($Zhi,4);
757     &shl        ($rem[0],4);                            # rem<<4

759     &pxor       ($Zlo,&QWP(16,"esp",$nhi[1],8));        # Z^=H[nhi]
760     &psllq      ($tmp,60);
761     &movz       ($rem[0],&LB($rem[0]));

763     &pxor       ($Zlo,$tmp);
764     &pxor       ($Zhi,&QWP(16+128,"esp",$nhi[1],8));

766     &pinsrw     ($red[0],&WP(0,$rem_8bit,$rem[1],2),2);
767     &pxor       ($Zhi,$red[1]);

769     &movd       ($dat,$Zlo);
770     &pinsrw     ($red[2],&WP(0,$rem_8bit,$rem[0],2),3); # last is <<48

772     &psllq      ($red[0],12);                           # correct by <<16>>4
773     &pxor       ($Zhi,$red[0]);
774     &psrlq      ($Zlo,32);
775     &pxor       ($Zhi,$red[2]);

777     &mov        ("ecx",&DWP(528+16+4,"esp"));   # restore inp
778     &movd       ("ebx",$Zlo);
779     &movq       ($tmp,$Zhi);                    # 01234567
780     &psllw      ($Zhi,8);                       # 1.3.5.7.
781     &psrlw      ($tmp,8);                       # .0.2.4.6
782     &por        ($Zhi,$tmp);                    # 10325476
783     &bswap      ($dat);
784     &pshufw     ($Zhi,$Zhi,0b00011011);         # 76543210
785     &bswap      ("ebx");

787     &cmp        ("ecx",&DWP(528+16+8,"esp"));   # are we done?
788     &jne        (&label("outer"));
789 }

791     &mov        ("eax",&DWP(528+16+0,"esp"));   # restore Xi
792     &mov        (&DWP(12,"eax"),"edx");
793     &mov        (&DWP(8,"eax"),"ebx");
794     &movq       (&QWP(0,"eax"),$Zhi);
```



```
796     &mov        ("esp",&DWP(528+16+12,"esp"));  # restore original %esp
797     &emms       ();
798 }
799 &function_end("gcm_ghash_4bit_mmx");
800 }}
```

```
if ($sse2) {{
################################################################################
# PCLMULQDQ version.

$Xip="eax";
$Htbl="edx";
$const="ecx";
$inp="esi";
$len="ebx";

($Xi,$Xhi)=("xmm0","xmm1");     $Hkey="xmm2";
($T1,$T2,$T3)=("xmm3","xmm4","xmm5");
($Xn,$Xhn)=("xmm6","xmm7");

&static_label("bswap");

sub clmul64x64_T2 {     # minimal "register" pressure
my ($Xhi,$Xi,$Hkey)=@_;

        &movdqa         ($Xhi,$Xi);             #
        &pshufd         ($T1,$Xi,0b01001110);
        &pshufd         ($T2,$Hkey,0b01001110);
        &pxor           ($T1,$Xi);              #
        &pxor           ($T2,$Hkey);

        &pclmulqdq      ($Xi,$Hkey,0x00);       #######
        &pclmulqdq      ($Xhi,$Hkey,0x11);      #######
        &pclmulqdq      ($T1,$T2,0x00);         #######
        &xorps          ($T1,$Xi);              #
        &xorps          ($T1,$Xhi);             #

        &movdqa         ($T2,$T1);              #
        &psrldq         ($T1,8);
        &pslldq         ($T2,8);                #
        &pxor           ($Xhi,$T1);
        &pxor           ($Xi,$T2);              #
}

sub clmul64x64_T3 {
# Even though this subroutine offers visually better ILP, it
# was empirically found to be a tad slower than above version.
# At least in gcm_ghash_clmul context. But it's just as well,
# because loop modulo-scheduling is possible only thanks to
# minimized "register" pressure...
my ($Xhi,$Xi,$Hkey)=@_;

        &movdqa         ($T1,$Xi);              #
        &movdqa         ($Xhi,$Xi);
        &pclmulqdq      ($Xi,$Hkey,0x00);       #######
        &pclmulqdq      ($Xhi,$Hkey,0x11);      #######
        &pshufd         ($T2,$T1,0b01001110);   #
        &pshufd         ($T3,$Hkey,0b01001110);
        &pxor           ($T2,$T1);              #
        &pxor           ($T3,$Hkey);
        &pclmulqdq      ($T2,$T3,0x00);         #######
        &pxor           ($T2,$Xi);              #
        &pxor           ($T2,$Xhi);             #

        &movdqa         ($T3,$T2);              #
        &psrldq         ($T2,8);
        &pslldq         ($T3,8);                #
        &pxor           ($Xhi,$T2);
        &pxor           ($Xi,$T3);              #
}
```



```
if (1) {                # Algorithm 9 with <<1 twist.
                        # Reduction is shorter and uses only two
                        # temporary registers, which makes it better
                        # candidate for interleaving with 64x64
                        # multiplication. Pre-modulo-scheduled loop
                        # was found to be ~20% faster than Algorithm 5
                        # below. Algorithm 9 was therefore chosen for
                        # further optimization...

sub reduction_alg9 {    # 17/13 times faster than Intel version
my ($Xhi,$Xi) = @_;

        # 1st phase
        &movdqa         ($T1,$Xi);              #
        &psllq          ($Xi,1);
        &pxor           ($Xi,$T1);              #
        &psllq          ($Xi,5);                #
        &pxor           ($Xi,$T1);              #
        &psllq          ($Xi,57);               #
        &movdqa         ($T2,$Xi);              #
        &pslldq         ($Xi,8);
        &psrldq         ($T2,8);                #
        &pxor           ($Xi,$T1);
        &pxor           ($Xhi,$T2);             #

        # 2nd phase
        &movdqa         ($T2,$Xi);
        &psrlq          ($Xi,5);
        &pxor           ($Xi,$T2);              #
        &psrlq          ($Xi,1);                #
        &pxor           ($Xi,$T2);              #
        &pxor           ($T2,$Xhi);
        &psrlq          ($Xi,1);                #
        &pxor           ($Xi,$T2);              #
}

&function_begin_B("gcm_init_clmul");
        &mov            ($Htbl,&wparam(0));
        &mov            ($Xip,&wparam(1));

        &call           (&label("pic"));
&set_label("pic");
        &blindpop       ($const);
        &lea            ($const,&DWP(&label("bswap")."-".&label("pic"),$const));

        &movdqu         ($Hkey,&QWP(0,$Xip));
        &pshufd         ($Hkey,$Hkey,0b01001110);# dword swap

        # <<1 twist
        &pshufd         ($T2,$Hkey,0b11111111); # broadcast uppermost dword
        &movdqa         ($T1,$Hkey);
        &psllq          ($Hkey,1);
        &pxor           ($T3,$T3);              #
        &psrlq          ($T1,63);
        &pcmpgtd        ($T3,$T2);              # broadcast carry bit
        &pslldq         ($T1,8);
        &por            ($Hkey,$T1);            # H<<=1

        # magic reduction
        &pand           ($T3,&QWP(16,$const));  # 0x1c2_polynomial
        &pxor           ($Hkey,$T3);            # if(carry) H^=0x1c2_polynomial

        # calculate H^2
        &movdqa         ($Xi,$Hkey);
        &clmul64x64_T2  ($Xhi,$Xi,$Hkey);
        &reduction_alg9 ($Xhi,$Xi);
```

```
932         &movdqu         (&QWP(0,$Htbl),$Hkey);  # save H
933         &movdqu         (&QWP(16,$Htbl),$Xi);   # save H^2

935         &ret            ();
936 &function_end_B("gcm_init_clmul");

938 &function_begin_B("gcm_gmult_clmul");
939         &mov            ($Xip,&wparam(0));
940         &mov            ($Htbl,&wparam(1));

942         &call           (&label("pic"));
943 &set_label("pic");
944         &blindpop       ($const);
945         &lea            ($const,&DWP(&label("bswap")."-".&label("pic"),$const));

947         &movdqu         ($Xi,&QWP(0,$Xip));
948         &movdqa         ($T3,&QWP(0,$const));
949         &movups         ($Hkey,&QWP(0,$Htbl));
950         &pshufb         ($Xi,$T3);

952         &clmul64x64_T2  ($Xhi,$Xi,$Hkey);
953         &reduction_alg9 ($Xhi,$Xi);

955         &pshufb         ($Xi,$T3);
956         &movdqu         (&QWP(0,$Xip),$Xi);

958         &ret    ();
959 &function_end_B("gcm_gmult_clmul");

961 &function_begin("gcm_ghash_clmul");
962         &mov            ($Xip,&wparam(0));
963         &mov            ($Htbl,&wparam(1));
964         &mov            ($inp,&wparam(2));
965         &mov            ($len,&wparam(3));

967         &call           (&label("pic"));
968 &set_label("pic");
969         &blindpop       ($const);
970         &lea            ($const,&DWP(&label("bswap")."-".&label("pic"),$const));

972         &movdqu         ($Xi,&QWP(0,$Xip));
973         &movdqa         ($T3,&QWP(0,$const));
974         &movdqu         ($Hkey,&QWP(0,$Htbl));
975         &pshufb         ($Xi,$T3);

977         &sub            ($len,0x10);
978         &jz             (&label("odd_tail"));

980         #######
981         # Xi+2 =[H*(Ii+1 + Xi+1)] mod P =
982         #       [(H*Ii+1) + (H*Xi+1)] mod P =
983         #       [(H*Ii+1) + H^2*(Ii+Xi)] mod P
984         #
985         &movdqu         ($T1,&QWP(0,$inp));     # Ii
986         &movdqu         ($Xn,&QWP(16,$inp));    # Ii+1
987         &pshufb         ($T1,$T3);
988         &pshufb         ($Xn,$T3);
989         &pxor           ($Xi,$T1);              # Ii+Xi

991         &clmul64x64_T2  ($Xhn,$Xn,$Hkey);       # H*Ii+1
992         &movups         ($Hkey,&QWP(16,$Htbl)); # load H^2

994         &lea            ($inp,&DWP(32,$inp));   # i+=2
995         &sub            ($len,0x20);
996         &jbe            (&label("even_tail"));
```



```
 998 &set_label("mod_loop");
 999         &clmul64x64_T2  ($Xhi,$Xi,$Hkey);       # H^2*(Ii+Xi)
1000         &movdqu         ($T1,&QWP(0,$inp));     # Ii
1001         &movups         ($Hkey,&QWP(0,$Htbl));  # load H

1003         &pxor           ($Xi,$Xn);              # (H*Ii+1) + H^2*(Ii+Xi)
1004         &pxor           ($Xhi,$Xhn);

1006         &movdqu         ($Xn,&QWP(16,$inp));    # Ii+1
1007         &pshufb         ($T1,$T3);
1008         &pshufb         ($Xn,$T3);

1010         &movdqa         ($T3,$Xn);              #&clmul64x64_TX ($Xhn,$Xn,$Hkey)
1011         &movdqa         ($Xhn,$Xn);
1012          &pxor          ($Xhi,$T1);             # "Ii+Xi", consume early

1014           &movdqa       ($T1,$Xi);              #&reduction_alg9($Xhi,$Xi); 1st
1015           &psllq        ($Xi,1);
1016           &pxor         ($Xi,$T1);              #
1017           &psllq        ($Xi,5);                #
1018           &pxor         ($Xi,$T1);              #
1019         &pclmulqdq      ($Xn,$Hkey,0x00);       #######
1020           &psllq        ($Xi,57);               #
1021           &movdqa       ($T2,$Xi);              #
1022           &pslldq       ($Xi,8);
1023           &psrldq       ($T2,8);                #
1024           &pxor         ($Xi,$T1);
1025         &pshufd         ($T1,$T3,0b01001110);
1026           &pxor         ($Xhi,$T2);             #
1027         &pxor           ($T1,$T3);
1028         &pshufd         ($T3,$Hkey,0b01001110);
1029         &pxor           ($T3,$Hkey);            #

1031         &pclmulqdq      ($Xhn,$Hkey,0x11);      #######
1032           &movdqa       ($T2,$Xi);              # 2nd phase
1033           &psrlq        ($Xi,5);
1034           &pxor         ($Xi,$T2);              #
1035           &psrlq        ($Xi,1);                #
1036           &pxor         ($Xi,$T2);              #
1037           &pxor         ($T2,$Xhi);
1038           &psrlq        ($Xi,1);                #
1039           &pxor         ($Xi,$T2);              #

1041         &pclmulqdq      ($T1,$T3,0x00);         #######
1042         &movups         ($Hkey,&QWP(16,$Htbl)); # load H^2
1043         &xorps          ($T1,$Xn);              #
1044         &xorps          ($T1,$Xhn);             #

1046         &movdqa         ($T3,$T1);              #
1047         &psrldq         ($T1,8);
1048         &pslldq         ($T3,8);                #
1049         &pxor           ($Xhn,$T1);
1050         &pxor           ($Xn,$T3);              #
1051         &movdqa         ($T3,&QWP(0,$const));

1053         &lea            ($inp,&DWP(32,$inp));
1054         &sub            ($len,0x20);
1055         &ja             (&label("mod_loop"));

1057 &set_label("even_tail");
1058         &clmul64x64_T2  ($Xhi,$Xi,$Hkey);       # H^2*(Ii+Xi)

1060         &pxor           ($Xi,$Xn);              # (H*Ii+1) + H^2*(Ii+Xi)
1061         &pxor           ($Xhi,$Xhn);
```

```
1063         &reduction_alg9 ($Xhi,$Xi);

1065         &test           ($len,$len);
1066         &jnz            (&label("done"));

1068         &movups         ($Hkey,&QWP(0,$Htbl));  # load H
1069 &set_label("odd_tail");
1070         &movdqu         ($T1,&QWP(0,$inp));     # Ii
1071         &pshufb         ($T1,$T3);
1072         &pxor           ($Xi,$T1);              # Ii+Xi

1074         &clmul64x64_T2  ($Xhi,$Xi,$Hkey);       # H*(Ii+Xi)
1075         &reduction_alg9 ($Xhi,$Xi);

1077 &set_label("done");
1078         &pshufb         ($Xi,$T3);
1079         &movdqu         (&QWP(0,$Xip),$Xi);
1080 &function_end("gcm_ghash_clmul");
```



```
1081 } else {                # Algorith 5. Kept for reference purposes.

1083 sub reduction_alg5 {    # 19/16 times faster than Intel version
1084 my ($Xhi,$Xi)=@_;

1086         # <<1
1087         &movdqa         ($T1,$Xi);              #
1088         &movdqa         ($T2,$Xhi);
1089         &pslld          ($Xi,1);
1090         &pslld          ($Xhi,1);               #
1091         &psrld          ($T1,31);
1092         &psrld          ($T2,31);               #
1093         &movdqa         ($T3,$T1);
1094         &pslldq         ($T1,4);
1095         &psrldq         ($T3,12);               #
1096         &psslldq        ($T2,4);
1097         &por            ($Xhi,$T3);             #
1098         &por            ($Xi,$T1);
1099         &por            ($Xhi,$T2);             #

1101         # 1st phase
1102         &movdqa         ($T1,$Xi);
1103         &movdqa         ($T2,$Xi);
1104         &movdqa         ($T3,$Xi);              #
1105         &pslld          ($T1,31);
1106         &pslld          ($T2,30);
1107         &pslld          ($Xi,25);               #
1108         &pxor           ($T1,$T2);
1109         &pxor           ($T1,$Xi);              #
1110         &movdqa         ($T2,$T1);              #
1111         &psslldq        ($T1,12);
1112         &psrldq         ($T2,4);                #
1113         &pxor           ($T3,$T1);

1115         # 2nd phase
1116         &pxor           ($Xhi,$T3);             #
1117         &movdqa         ($Xi,$T3);
1118         &movdqa         ($T1,$T3);
1119         &psrld          ($Xi,1);                #
1120         &psrld          ($T1,2);
1121         &psrld          ($T3,7);                #
1122         &pxor           ($Xi,$T1);
1123         &pxor           ($Xhi,$T2);
1124         &pxor           ($Xi,$T3);              #
1125         &pxor           ($Xi,$Xhi);             #
1126 }

1128 &function_begin_B("gcm_init_clmul");
1129         &mov            ($Htbl,&wparam(0));
1130         &mov            ($Xip,&wparam(1));

1132         &call           (&label("pic"));
1133 &set_label("pic");
1134         &blindpop       ($const);
1135         &lea            ($const,&DWP(&label("bswap")."-".&label("pic"),$const));

1137         &movdqu         ($Hkey,&QWP(0,$Xip));
1138         &pshufd         ($Hkey,$Hkey,0b01001110);# dword swap

1140         # calculate H^2
1141         &movdqa         ($Xi,$Hkey);
1142         &clmul64x64_T3  ($Xhi,$Xi,$Hkey);
1143         &reduction_alg5 ($Xhi,$Xi);

1145         &movdqu         (&QWP(0,$Htbl),$Hkey);  # save H
1146         &movdqu         (&QWP(16,$Htbl),$Xi);   # save H^2
```

```
1148         &ret            ();
1149 &function_end_B("gcm_init_clmul");

1151 &function_begin_B("gcm_gmult_clmul");
1152         &mov            ($Xip,&wparam(0));
1153         &mov            ($Htbl,&wparam(1));

1155         &call           (&label("pic"));
1156 &set_label("pic");
1157         &blindpop       ($const);
1158         &lea            ($const,&DWP(&label("bswap")."-".&label("pic"),$const));

1160         &movdqu         ($Xi,&QWP(0,$Xip));
1161         &movdqa         ($Xn,&QWP(0,$const));
1162         &movdqu         ($Hkey,&QWP(0,$Htbl));
1163         &pshufb         ($Xi,$Xn);

1165         &clmul64x64_T3  ($Xhi,$Xi,$Hkey);
1166         &reduction_alg5 ($Xhi,$Xi);

1168         &pshufb         ($Xi,$Xn);
1169         &movdqu         (&QWP(0,$Xip),$Xi);

1171         &ret    ();
1172 &function_end_B("gcm_gmult_clmul");

1174 &function_begin("gcm_ghash_clmul");
1175         &mov            ($Xip,&wparam(0));
1176         &mov            ($Htbl,&wparam(1));
1177         &mov            ($inp,&wparam(2));
1178         &mov            ($len,&wparam(3));

1180         &call           (&label("pic"));
1181 &set_label("pic");
1182         &blindpop       ($const);
1183         &lea            ($const,&DWP(&label("bswap")."-".&label("pic"),$const));

1185         &movdqu         ($Xi,&QWP(0,$Xip));
1186         &movdqa         ($T3,&QWP(0,$const));
1187         &movdqu         ($Hkey,&QWP(0,$Htbl));
1188         &pshufb         ($Xi,$T3);

1190         &sub            ($len,0x10);
1191         &jz             (&label("odd_tail"));

1193         #######
1194         # Xi+2 =[H*(Ii+1 + Xi+1)] mod P =
1195         #       [(H*Ii+1) + (H*Xi+1)] mod P =
1196         #       [(H*Ii+1) + H^2*(Ii+Xi)] mod P
1197         #
1198         &movdqu         ($T1,&QWP(0,$inp));     # Ii
1199         &movdqu         ($Xn,&QWP(16,$inp));    # Ii+1
1200         &pshufb         ($T1,$T3);
1201         &pshufb         ($Xn,$T3);
1202         &pxor           ($Xi,$T1);              # Ii+Xi

1204         &clmul64x64_T3  ($Xhn,$Xn,$Hkey);       # H*Ii+1
1205         &movdqu         ($Hkey,&QWP(16,$Htbl)); # load H^2

1207         &sub            ($len,0x20);
1208         &lea            ($inp,&DWP(32,$inp));   # i+=2
1209         &jbe            (&label("even_tail"));

1211 &set_label("mod_loop");
1212         &clmul64x64_T3  ($Xhi,$Xi,$Hkey);       # H^2*(Ii+Xi)
```



```
1213         &movdqu         ($Hkey,&QWP(0,$Htbl));  # load H

1215         &pxor           ($Xi,$Xn);              # (H*Ii+1) + H^2*(Ii+Xi)
1216         &pxor           ($Xhi,$Xhn);

1218         &reduction_alg5 ($Xhi,$Xi);

1220         #######
1221         &movdqa         ($T3,&QWP(0,$const));
1222         &movdqu         ($T1,&QWP(0,$inp));     # Ii
1223         &movdqu         ($Xn,&QWP(16,$inp));    # Ii+1
1224         &pshufb         ($T1,$T3);
1225         &pshufb         ($Xn,$T3);
1226         &pxor           ($Xi,$T1);              # Ii+Xi

1228         &clmul64x64_T3  ($Xhn,$Xn,$Hkey);       # H*Ii+1
1229         &movdqu         ($Hkey,&QWP(16,$Htbl)); # load H^2

1231         &sub            ($len,0x20);
1232         &lea            ($inp,&DWP(32,$inp));
1233         &ja             (&label("mod_loop"));

1235 &set_label("even_tail");
1236         &clmul64x64_T3  ($Xhi,$Xi,$Hkey);       # H^2*(Ii+Xi)

1238         &pxor           ($Xi,$Xn);              # (H*Ii+1) + H^2*(Ii+Xi)
1239         &pxor           ($Xhi,$Xhn);

1241         &reduction_alg5 ($Xhi,$Xi);

1243         &movdqa         ($T3,&QWP(0,$const));
1244         &test           ($len,$len);
1245         &jnz            (&label("done"));

1247         &movdqu         ($Hkey,&QWP(0,$Htbl));  # load H
1248 &set_label("odd_tail");
1249         &movdqu         ($T1,&QWP(0,$inp));     # Ii
1250         &pshufb         ($T1,$T3);
1251         &pxor           ($Xi,$T1);              # Ii+Xi

1253         &clmul64x64_T3  ($Xhi,$Xi,$Hkey);       # H*(Ii+Xi)
1254         &reduction_alg5 ($Xhi,$Xi);

1256         &movdqa         ($T3,&QWP(0,$const));
1257 &set_label("done");
1258         &pshufb         ($Xi,$T3);
1259         &movdqu         (&QWP(0,$Xip),$Xi);
1260 &function_end("gcm_ghash_clmul");

1262 }
```

```
&set_label("bswap",64);
        &data_byte(15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0);
        &data_byte(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0xc2); # 0x1c2_polynomial
}}      # $sse2

&set_label("rem_4bit",64);
        &data_word(0,0x0000<<$S,0,0x1C20<<$S,0,0x3840<<$S,0,0x2460<<$S);
        &data_word(0,0x7080<<$S,0,0x6CA0<<$S,0,0x48C0<<$S,0,0x54E0<<$S);
        &data_word(0,0xE100<<$S,0,0xFD20<<$S,0,0xD940<<$S,0,0xC560<<$S);
        &data_word(0,0x9180<<$S,0,0x8DA0<<$S,0,0xA9C0<<$S,0,0xB5E0<<$S);
&set_label("rem_8bit",64);
        &data_short(0x0000,0x01C2,0x0384,0x0246,0x0708,0x06CA,0x048C,0x054E);
        &data_short(0x0E10,0x0FD2,0x0D94,0x0C56,0x0918,0x08DA,0x0A9C,0x0B5E);
        &data_short(0x1C20,0x1DE2,0x1FA4,0x1E66,0x1B28,0x1AEA,0x18AC,0x196E);
        &data_short(0x1230,0x13F2,0x11B4,0x1076,0x1538,0x14FA,0x16BC,0x177E);
        &data_short(0x3840,0x3982,0x3BC4,0x3A06,0x3F48,0x3E8A,0x3CCC,0x3D0E);
        &data_short(0x3650,0x3792,0x35D4,0x3416,0x3158,0x309A,0x32DC,0x331E);
        &data_short(0x2460,0x25A2,0x27E4,0x2626,0x2368,0x22AA,0x20EC,0x212E);
        &data_short(0x2A70,0x2BB2,0x29F4,0x2836,0x2D78,0x2CBA,0x2EFC,0x2F3E);
        &data_short(0x7080,0x7142,0x7304,0x72C6,0x7788,0x764A,0x740C,0x75CE);
        &data_short(0x7E90,0x7F52,0x7D14,0x7CD6,0x7998,0x785A,0x7A1C,0x7BDE);
        &data_short(0x6CA0,0x6D62,0x6F24,0x6EE6,0x6BA8,0x6A6A,0x682C,0x69EE);
        &data_short(0x62B0,0x6372,0x6134,0x60F6,0x65B8,0x647A,0x663C,0x67FE);
        &data_short(0x48C0,0x4902,0x4B44,0x4A86,0x4FC8,0x4E0A,0x4C4C,0x4D8E);
        &data_short(0x46D0,0x4712,0x4554,0x4496,0x41D8,0x401A,0x425C,0x439E);
        &data_short(0x54E0,0x5522,0x5764,0x56A6,0x53E8,0x522A,0x506C,0x51AE);
        &data_short(0x5AF0,0x5B32,0x5974,0x58B6,0x5DF8,0x5C3A,0x5E7C,0x5FBE);
        &data_short(0xE100,0xE0C2,0xE284,0xE346,0xE608,0xE7CA,0xE58C,0xE44E);
        &data_short(0xEF10,0xEED2,0xEC94,0xED56,0xE818,0xE9DA,0xEB9C,0xEA5E);
        &data_short(0xFD20,0xFCE2,0xFEA4,0xFF66,0xFA28,0xFBEA,0xF9AC,0xF86E);
        &data_short(0xF330,0xF2F2,0xF0B4,0xF176,0xF438,0xF5FA,0xF7BC,0xF67E);
        &data_short(0xD940,0xD882,0xDAC4,0xDB06,0xDE48,0xDF8A,0xDDCC,0xDC0E);
        &data_short(0xD750,0xD692,0xD4D4,0xD516,0xD058,0xD19A,0xD3DC,0xD21E);
        &data_short(0xC560,0xC4A2,0xC6E4,0xC726,0xC268,0xC3AA,0xC1EC,0xC02E);
        &data_short(0xCB70,0xCAB2,0xC8F4,0xC936,0xCC78,0xCDBA,0xCFFC,0xCE3E);
        &data_short(0x9180,0x9042,0x9204,0x93C6,0x9688,0x974A,0x950C,0x94CE);
        &data_short(0x9F90,0x9E52,0x9C14,0x9DD6,0x9898,0x995A,0x9B1C,0x9ADE);
        &data_short(0x8DA0,0x8C62,0x8E24,0x8FE6,0x8AA8,0x8B6A,0x892C,0x88EE);
        &data_short(0x83B0,0x8272,0x8034,0x81F6,0x84B8,0x857A,0x873C,0x86FE);
        &data_short(0xA9C0,0xA802,0xAA44,0xAB86,0xAEC8,0xAF0A,0xAD4C,0xAC8E);
        &data_short(0xA7D0,0xA612,0xA454,0xA596,0xA0D8,0xA11A,0xA35C,0xA29E);
        &data_short(0xB5E0,0xB422,0xB664,0xB7A6,0xB2E8,0xB32A,0xB16C,0xB0AE);
        &data_short(0xBBF0,0xBA32,0xB874,0xB9B6,0xBCF8,0xBD3A,0xBF7C,0xBEBE);
}}}     # !$x86only

&asciz("GHASH for x86, CRYPTOGAMS by <appro\@openssl.org>");
&asm_finish();

# A question was risen about choice of vanilla MMX. Or rather why wasn't
# SSE2 chosen instead? In addition to the fact that MMX runs on legacy
# CPUs such as PIII, "4-bit" MMX version was observed to provide better
# performance than *corresponding* SSE2 one even on contemporary CPUs.
# SSE2 results were provided by Peter-Michael Hager. He maintains SSE2
# implementation featuring full range of lookup-table sizes, but with
# per-invocation lookup table setup. Latter means that table size is
# chosen depending on how much data is to be hashed in every given call,
# more data - larger table. Best reported result for Core2 is ~4 cycles
# per processed byte out of 64KB block. This number accounts even for
# 64KB table setup overhead. As discussed in gcm128.c we choose to be
# more conservative in respect to lookup table sizes, but how do the
# results compare? Minimalistic "256B" MMX version delivers ~11 cycles
# on same platform. As also discussed in gcm128.c, next in line "8-bit
# Shoup's" or "4KB" method should deliver twice the performance of
# "256B" one, in other words not worse than ~6 cycles per byte. It
# should be also be noted that in SSE2 case improvement can be "super-
# linear," i.e. more than twice, mostly because >>8 maps to single
```



```
# instruction on SSE2 register. This is unlike "4-bit" case when >>4
# maps to same amount of instructions in both MMX and SSE2 cases.
# Bottom line is that switch to SSE2 is considered to be justifiable
# only in case we choose to implement "8-bit" method...
#endif /* ! codereview */
```

```
**********************************************************
   19325 Wed Aug 13 19:53:07 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/ghash-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# March, June 2010
#
# The module implements "4-bit" GCM GHASH function and underlying
# single multiplication operation in GF(2^128). "4-bit" means that
# it uses 256 bytes per-key table [+128 bytes shared table]. GHASH
# function features so called "528B" variant utilizing additional
# 256+16 bytes of per-key storage [+512 bytes shared table].
# Performance results are for this streamed GHASH subroutine and are
# expressed in cycles per processed byte, less is better:
#
#               gcc 3.4.x(*)    assembler
#
# P4            28.6            14.0            +100%
# Opteron       19.3            7.7             +150%
# Core2         17.8            8.1(**)         +120%
#
# (*)   comparison is not completely fair, because C results are
#       for vanilla "256B" implementation, while assembler results
#       are for "528B";-)
# (**)  it's mystery [to me] why Core2 result is not same as for
#       Opteron;

# May 2010
#
# Add PCLMULQDQ version performing at 2.02 cycles per processed byte.
# See ghash-x86.pl for background information and details about coding
# techniques.
#
# Special thanks to David Woodhouse <dwmw2@infradead.org> for
# providing access to a Westmere-based system on behalf of Intel
# Open Source Technology Centre.

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

# common register layout
$nlo="%rax";
$nhi="%rbx";
$Zlo="%r8";
$Zhi="%r9";
$tmp="%r10";
```



```
$rem_4bit = "%r11";

$Xi="%rdi";
$Htbl="%rsi";

# per-function register layout
$cnt="%rcx";
$rem="%rdx";

sub LB() { my $r=shift; $r =~ s/%[er]([a-d])x/%\1l/     or
                        $r =~ s/%[er]([sd]i)/%\1l/      or
                        $r =~ s/%[er](bp)/%\1l/         or
                        $r =~ s/%(r[0-9]+)[d]?/%\1b/;   $r; }

sub AUTOLOAD()          # thunk [simplified] 32-bit style perlasm
{ my $opcode = $AUTOLOAD; $opcode =~ s/.*::/;
  my $arg = pop;
    $arg = "\$$arg" if ($arg*1 eq $arg);
    $code .= "\t$opcode\t".join(',',$arg,reverse @_)."\n";
}
```

```
{ my $N;
  sub loop() {
  my $inp = shift;

        $N++;
$code.=<<___;
        xor     $nlo,$nlo
        xor     $nhi,$nhi
        mov     `&LB("$Zlo")`,`&LB("$nlo")`
        mov     `&LB("$Zlo")`,`&LB("$nhi")`
        shl     \$4,`&LB("$nlo")`
        mov     \$14,$cnt
        mov     8($Htbl,$nlo),$Zlo
        mov     ($Htbl,$nlo),$Zhi
        and     \$0xf0,`&LB("$nhi")`
        mov     $Zlo,$rem
        jmp     .Loop$N

.align  16
.Loop$N:
        shr     \$4,$Zlo
        and     \$0xf,$rem
        mov     $Zhi,$tmp
        mov     ($inp,$cnt),`&LB("$nlo")`
        shr     \$4,$Zhi
        xor     8($Htbl,$nhi),$Zlo
        shl     \$60,$tmp
        xor     ($Htbl,$nhi),$Zhi
        mov     `&LB("$nlo")`,`&LB("$nhi")`
        xor     ($rem_4bit,$rem,8),$Zhi
        mov     $Zlo,$rem
        shl     \$4,`&LB("$nlo")`
        xor     $tmp,$Zlo
        dec     $cnt
        js      .Lbreak$N

        shr     \$4,$Zlo
        and     \$0xf,$rem
        mov     $Zhi,$tmp
        shr     \$4,$Zhi
        xor     8($Htbl,$nlo),$Zlo
        shl     \$60,$tmp
        xor     ($Htbl,$nlo),$Zhi
        and     \$0xf0,`&LB("$nhi")`
        xor     ($rem_4bit,$rem,8),$Zhi
        mov     $Zlo,$rem
        xor     $tmp,$Zlo
        jmp     .Loop$N

.align  16
.Lbreak$N:
        shr     \$4,$Zlo
        and     \$0xf,$rem
        mov     $Zhi,$tmp
        shr     \$4,$Zhi
        xor     8($Htbl,$nlo),$Zlo
        shl     \$60,$tmp
        xor     ($Htbl,$nlo),$Zhi
        and     \$0xf0,`&LB("$nhi")`
        xor     ($rem_4bit,$rem,8),$Zhi
        mov     $Zlo,$rem
        xor     $tmp,$Zlo

        shr     \$4,$Zlo
        and     \$0xf,$rem
        mov     $Zhi,$tmp
```



```
        shr     \$4,$Zhi
        xor     8($Htbl,$nhi),$Zlo
        shl     \$60,$tmp
        xor     ($Htbl,$nhi),$Zhi
        xor     $tmp,$Zlo
        xor     ($rem_4bit,$rem,8),$Zhi

        bswap   $Zlo
        bswap   $Zhi
___
}}

$code=<<___;
.text

.globl  gcm_gmult_4bit
.type   gcm_gmult_4bit,\@function,2
.align  16
gcm_gmult_4bit:
        push    %rbx
        push    %rbp            # %rbp and %r12 are pushed exclusively in
        push    %r12            # order to reuse Win64 exception handler...
.Lgmult_prologue:

        movzb   15($Xi),$Zlo
        lea     .Lrem_4bit(%rip),$rem_4bit
___
        &loop   ($Xi);
$code.=<<___;
        mov     $Zlo,8($Xi)
        mov     $Zhi,($Xi)

        mov     16(%rsp),%rbx
        lea     24(%rsp),%rsp
.Lgmult_epilogue:
        ret
.size   gcm_gmult_4bit,.-gcm_gmult_4bit
___
```

```
# per-function register layout
$inp="%rdx";
$len="%rcx";
$rem_8bit=$rem_4bit;

$code.=<<___;
.globl  gcm_ghash_4bit
.type   gcm_ghash_4bit,\@function,4
.align  16
gcm_ghash_4bit:
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
        sub     \$280,%rsp
.Lghash_prologue:
        mov     $inp,%r14               # reassign couple of args
        mov     $len,%r15
___
{ my $inp="%r14";
  my $dat="%edx";
  my $len="%r15";
  my @nhi=("%ebx","%ecx");
  my @rem=("%r12","%r13");
  my $Hshr4="%rbp";

        &sub    ($Htbl,-128);           # size optimization
        &lea    ($Hshr4,"16+128(%rsp)");
        { my @lo =($nlo,$nhi);
          my @hi =($Zlo,$Zhi);

          &xor  ($dat,$dat);
          for ($i=0,$j=-2;$i<18;$i++,$j++) {
            &mov        ("$j(%rsp)",&LB($dat))          if ($i>1);
            &or         ($lo[0],$tmp)                   if ($i>1);
            &mov        (&LB($dat),&LB($lo[1]))         if ($i>0 && $i<17);
            &shr        ($lo[1],4)                      if ($i>0 && $i<17);
            &mov        ($tmp,$hi[1])                   if ($i>0 && $i<17);
            &shr        ($hi[1],4)                      if ($i>0 && $i<17);
            &mov        ("8*$j($Hshr4)",$hi[0])         if ($i>1);
            &mov        ($hi[0],"16*$i+0-128($Htbl)")   if ($i<16);
            &shl        (&LB($dat),4)                   if ($i>0 && $i<17);
            &mov        ("8*$j-128($Hshr4)",$lo[0])     if ($i>1);
            &mov        ($lo[0],"16*$i+8-128($Htbl)")   if ($i<16);
            &shl        ($tmp,60)                       if ($i>0 && $i<17);

            push        (@lo,shift(@lo));
            push        (@hi,shift(@hi));
          }
        }
        &add    ($Htbl,-128);
        &mov    ($Zlo,"8($Xi)");
        &mov    ($Zhi,"0($Xi)");
        &add    ($len,$inp);            # pointer to the end of data
        &lea    ($rem_8bit,".Lrem_8bit(%rip)");
        &jmp    (".Louter_loop");

$code.=".align  16\n.Louter_loop:\n";
        &xor    ($Zhi,"($inp)");
        &mov    ("%rdx","8($inp)");
        &lea    ($inp,"16($inp)");
        &xor    ("%rdx",$Zlo);
        &mov    (",($Xi)",$Zhi);
        &mov    ("8($Xi)","%rdx");
```



```
        &shr    ("%rdx",32);

        &xor    ($nlo,$nlo);
        &rol    ($dat,8);
        &mov    (&LB($nlo),&LB($dat));
        &movz   ($nhi[0],&LB($dat));
        &shl    (&LB($nlo),4);
        &shr    ($nhi[0],4);

        for ($j=11,$i=0;$i<15;$i++) {
            &rol        ($dat,8);
            &xor        ($Zlo,"8($Htbl,$nlo)")                  if ($i>0);
            &xor        ($Zhi,"($Htbl,$nlo)")                   if ($i>0);
            &mov        ($Zlo,"8($Htbl,$nlo)")                  if ($i==0);
            &mov        ($Zhi,"($Htbl,$nlo)")                   if ($i==0);

            &mov        (&LB($nlo),&LB($dat));
            &xor        ($Zlo,$tmp)                             if ($i>0);
            &movzw      ($rem[1],"($rem_8bit,$rem[1],2)")       if ($i>0);

            &movz       ($nhi[1],&LB($dat));
            &shl        (&LB($nlo),4);
            &movzb      ($rem[0],"(%rsp,$nhi[0])");

            &shr        ($nhi[1],4)                             if ($i<14);
            &and        ($nhi[1],0xf0)                          if ($i==14);
            &shl        ($rem[1],48)                            if ($i>0);
            &xor        ($rem[0],$Zlo);

            &mov        ($tmp,$Zhi);
            &xor        ($Zhi,$rem[1])                          if ($i>0);
            &shr        ($Zlo,8);

            &movz       ($rem[0],&LB($rem[0]));
            &mov        ($dat,"$j($Xi)")                        if (--$j%4==0);
            &shr        ($Zhi,8);

            &xor        ($Zlo,"-128($Hshr4,$nhi[0],8)");
            &shl        ($tmp,56);
            &xor        ($Zhi,"($Hshr4,$nhi[0],8)");

            unshift     (@nhi,pop(@nhi));               # "rotate" registers
            unshift     (@rem,pop(@rem));
        }
        &movzw  ($rem[1],"($rem_8bit,$rem[1],2)");
        &xor    ($Zlo,"8($Htbl,$nlo)");
        &xor    ($Zhi,"($Htbl,$nlo)");

        &shl    ($rem[1],48);
        &xor    ($Zlo,$tmp);

        &xor    ($Zhi,$rem[1]);
        &movz   ($rem[0],&LB($Zlo));
        &shr    ($Zlo,4);

        &mov    ($tmp,$Zhi);
        &shl    (&LB($rem[0]),4);
        &shr    ($Zhi,4);

        &xor    ($Zlo,"8($Htbl,$nhi[0])");
        &movzw  ($rem[0],"($rem_8bit,$rem[0],2)");
        &shl    ($tmp,60);

        &xor    ($Zhi,"($Htbl,$nhi[0])");
        &xor    ($Zlo,$tmp);
        &shl    ($rem[0],48);
```

```
319         &bswap  ($Zlo);
320         &xor    ($Zhi,$rem[0]);

322         &bswap  ($Zhi);
323         &cmp    ($inp,$len);
324         &jb     (".Louter_loop");
325 }
326 $code.=<<___;
327         mov     $Zlo,8($Xi)
328         mov     $Zhi,($Xi)

330         lea     280(%rsp),%rsi
331         mov     0(%rsi),%r15
332         mov     8(%rsi),%r14
333         mov     16(%rsi),%r13
334         mov     24(%rsi),%r12
335         mov     32(%rsi),%rbp
336         mov     40(%rsi),%rbx
337         lea     48(%rsi),%rsp
338 .Lghash_epilogue:
339         ret
340 .size   gcm_ghash_4bit,.-gcm_ghash_4bit
341 ___
```



```
342 ######################################################################
343 # PCLMULQDQ version.

345 @_4args=$win64? ("%rcx","%rdx","%r8", "%r9") :  # Win64 order
346                 ("%rdi","%rsi","%rdx","%rcx");  # Unix order

348 ($Xi,$Xhi)=("%xmm0","%xmm1");   $Hkey="%xmm2";
349 ($T1,$T2,$T3)=("%xmm3","%xmm4","%xmm5");

351 sub clmul64x64_T2 {     # minimal register pressure
352 my ($Xhi,$Xi,$Hkey,$modulo)=@_;

354 $code.=<<___ if (!defined($modulo));
355         movdqa          $Xi,$Xhi                #
356         pshufd          \$0b01001110,$Xi,$T1
357         pshufd          \$0b01001110,$Hkey,$T2
358         pxor            $Xi,$T1                 #
359         pxor            $Hkey,$T2
360 ___
361 $code.=<<___;
362         pclmulqdq       \$0x00,$Hkey,$Xi        #######
363         pclmulqdq       \$0x11,$Hkey,$Xhi       #######
364         pclmulqdq       \$0x00,$T2,$T1          #######
365         pxor            $Xi,$T1                 #
366         pxor            $Xhi,$T1                #

368         movdqa          $T1,$T2                 #
369         psrldq          \$8,$T1
370         pslldq          \$8,$T2                 #
371         pxor            $T1,$Xhi
372         pxor            $T2,$Xi                 #
373 ___
374 }

376 sub reduction_alg9 {    # 17/13 times faster than Intel version
377 my ($Xhi,$Xi) = @_;

379 $code.=<<___;
380         # 1st phase
381         movdqa          $Xi,$T1                 #
382         psllq           \$1,$Xi
383         pxor            $T1,$Xi                 #
384         psllq           \$5,$Xi                 #
385         pxor            $T1,$Xi                 #
386         psllq           \$57,$Xi                #
387         movdqa          $Xi,$T2                 #
388         pslldq          \$8,$Xi
389         psrldq          \$8,$T2                 #
390         pxor            $T1,$Xi
391         pxor            $T2,$Xhi                #

393         # 2nd phase
394         movdqa          $Xi,$T2
395         psrlq           \$5,$Xi
396         pxor            $T2,$Xi                 #
397         psrlq           \$1,$Xi                 #
398         pxor            $T2,$Xi                 #
399         pxor            $Xhi,$T2
400         psrlq           \$1,$Xi                 #
401         pxor            $T2,$Xi                 #
402 ___
403 }
```

```
404 { my ($Htbl,$Xip)=@_4args;
405
406 $code.=<<___;
407 .globl  gcm_init_clmul
408 .type   gcm_init_clmul,\@abi-omnipotent
409 .align  16
410 gcm_init_clmul:
411         movdqu          ($Xip),$Hkey
412         pshufd          \$0b01001110,$Hkey,$Hkey        # dword swap
413
414         # <<1 twist
415         pshufd          \$0b11111111,$Hkey,$T2  # broadcast uppermost dword
416         movdqa          $Hkey,$T1
417         psllq           \$1,$Hkey
418         pxor            $T3,$T3                 #
419         psrlq           \$63,$T1
420         pcmpgtd         $T2,$T3                 # broadcast carry bit
421         pslldq          \$8,$T1
422         por             $T1,$Hkey               # H<<=1
423
424         # magic reduction
425         pand            .L0x1c2_polynomial(%rip),$T3
426         pxor            $T3,$Hkey               # if(carry) H^=0x1c2_polynomial
427
428         # calculate H^2
429         movdqa          $Hkey,$Xi
430 ___
431         &clmul64x64_T2  ($Xhi,$Xi,$Hkey);
432         &reduction_alg9 ($Xhi,$Xi);
433 $code.=<<___;
434         movdqu          $Hkey,($Htbl)           # save H
435         movdqu          $Xi,16($Htbl)           # save H^2
436         ret
437 .size   gcm_init_clmul,.-gcm_init_clmul
438 ___
439 }
440
441 { my ($Xip,$Htbl)=@_4args;
442
443 $code.=<<___;
444 .globl  gcm_gmult_clmul
445 .type   gcm_gmult_clmul,\@abi-omnipotent
446 .align  16
447 gcm_gmult_clmul:
448         movdqu          ($Xip),$Xi
449         movdqa          .Lbswap_mask(%rip),$T3
450         movdqu          ($Htbl),$Hkey
451         pshufb          $T3,$Xi
452 ___
453         &clmul64x64_T2  ($Xhi,$Xi,$Hkey);
454         &reduction_alg9 ($Xhi,$Xi);
455 $code.=<<___;
456         pshufb          $T3,$Xi
457         movdqu          $Xi,($Xip)
458         ret
459 .size   gcm_gmult_clmul,.-gcm_gmult_clmul
460 ___
461 }
```



```
462 { my ($Xip,$Htbl,$inp,$len)=@_4args;
463   my $Xn="%xmm6";
464   my $Xhn="%xmm7";
465   my $Hkey2="%xmm8";
466   my $T1n="%xmm9";
467   my $T2n="%xmm10";
468
469 $code.=<<___;
470 .globl  gcm_ghash_clmul
471 .type   gcm_ghash_clmul,\@abi-omnipotent
472 .align  16
473 gcm_ghash_clmul:
474 ___
475 $code.=<<___ if ($win64);
476 .LSEH_begin_gcm_ghash_clmul:
477         # I can't trust assembler to use specific encoding:-(
478         .byte   0x48,0x83,0xec,0x58             #sub    \$0x58,%rsp
479         .byte   0x0f,0x29,0x34,0x24             #movaps %xmm6,(%rsp)
480         .byte   0x0f,0x29,0x7c,0x24,0x10        #movdqa %xmm7,0x10(%rsp)
481         .byte   0x44,0x0f,0x29,0x44,0x24,0x20   #movaps %xmm8,0x20(%rsp)
482         .byte   0x44,0x0f,0x29,0x4c,0x24,0x30   #movaps %xmm9,0x30(%rsp)
483         .byte   0x44,0x0f,0x29,0x54,0x24,0x40   #movaps %xmm10,0x40(%rsp)
484 ___
485 $code.=<<___;
486         movdqa          .Lbswap_mask(%rip),$T3
487
488         movdqu          ($Xip),$Xi
489         movdqu          ($Htbl),$Hkey
490         pshufb          $T3,$Xi
491
492         sub             \$0x10,$len
493         jz              .Lodd_tail
494
495         movdqu          16($Htbl),$Hkey2
496         #######
497         # Xi+2 =[H*(Ii+1 + Xi+1)] mod P =
498         #       [(H*Ii+1) + (H*Xi+1)] mod P =
499         #       [(H*Ii+1) + H^2*(Ii+Xi)] mod P
500         #
501         movdqu          ($inp),$T1              # Ii
502         movdqu          16($inp),$Xn            # Ii+1
503         pshufb          $T3,$T1
504         pshufb          $T3,$Xn
505         pxor            $T1,$Xi                 # Ii+Xi
506 ___
507         &clmul64x64_T2  ($Xhn,$Xn,$Hkey);       # H*Ii+1
508 $code.=<<___;
509         movdqa          $Xi,$Xhi                #
510         pshufd          \$0b01001110,$Xi,$T1
511         pshufd          \$0b01001110,$Hkey2,$T2
512         pxor            $Xi,$T1                 #
513         pxor            $Hkey2,$T2
514
515         lea             32($inp),$inp           # i+=2
516         sub             \$0x20,$len
517         jbe             .Leven_tail
518
519 .Lmod_loop:
520 ___
521         &clmul64x64_T2  ($Xhi,$Xi,$Hkey2,1);    # H^2*(Ii+Xi)
522 $code.=<<___;
523         movdqu          ($inp),$T1              # Ii
524         pxor            $Xn,$Xi                 # (H*Ii+1) + H^2*(Ii+Xi)
525         pxor            $Xhn,$Xhi
526
527         movdqu          16($inp),$Xn            # Ii+1
```

```
528         pshufb          $T3,$T1
529         pshufb          $T3,$Xn

531         movdqa          $Xn,$Xhn                #
532         pshufd          \$0b01001110,$Xn,$T1n
533         pshufd          \$0b01001110,$HKey,$T2n
534         pxor            $Xn,$T1n                #
535         pxor            $HKey,$T2n
536          pxor           $T1,$Xhi                # "Ii+Xi", consume early

538           movdqa        $Xi,$T1                 # 1st phase
539           psllq         \$1,$Xi
540           pxor          $T1,$Xi                 #
541           psllq         \$5,$Xi                 #
542           pxor          $T1,$Xi                 #
543         pclmulqdq       \$0x00,$HKey,$Xn        #######
544           psllq         \$57,$Xi                #
545           movdqa        $Xi,$T2                 #
546           pslldq        \$8,$Xi
547           psrldq        \$8,$T2                 #
548           pxor          $T1,$Xi
549           pxor          $T2,$Xhi                #

551         pclmulqdq       \$0x11,$HKey,$Xhn       #######
552           movdqa        $Xi,$T2                 # 2nd phase
553           psrlq         \$5,$Xi
554           pxor          $T2,$Xi                 #
555           psrlq         \$1,$Xi                 #
556           pxor          $T2,$Xi                 #
557           pxor          $Xhi,$T2
558           psrlq         \$1,$Xi                 #
559           pxor          $T2,$Xi                 #

561         pclmulqdq       \$0x00,$T2n,$T1n        #######
562          movdqa         $Xi,$Xhi                #
563          pshufd         \$0b01001110,$Xi,$T1
564          pshufd         \$0b01001110,$HKey2,$T2
565          pxor           $Xi,$T1                 #
566          pxor           $HKey2,$T2

568         pxor            $Xn,$T1n                #
569         pxor            $Xhn,$T1n               #
570         movdqa          $T1n,$T2n               #
571         psrldq          \$8,$T1n
572         pslldq          \$8,$T2n                #
573         pxor            $T1n,$Xhn
574         pxor            $T2n,$Xn                #

576         lea             32($inp),$inp
577         sub             \$0x20,$len
578         ja              .Lmod_loop

580 .Leven_tail:
581 ___
582         &clmul64x64_T2  ($Xhi,$Xi,$HKey2,1);    # H^2*(Ii+Xi)
583 $code.=<<___;
584         pxor            $Xn,$Xi                 # (H*Ii+1) + H^2*(Ii+Xi)
585         pxor            $Xhn,$Xhi
586 ___
587         &reduction_alg9 ($Xhi,$Xi);
588 $code.=<<___;
589         test            $len,$len
590         jnz             .Ldone

592 .Lodd_tail:
593         movdqu          ($inp),$T1              # Ii
```



```
594         pshufb          $T3,$T1
595         pxor            $T1,$Xi                 # Ii+Xi
596
597 ___
598         &clmul64x64_T2  ($Xhi,$Xi,$HKey);       # H*(Ii+Xi)
599         &reduction_alg9 ($Xhi,$Xi);
600 $code.=<<___;
601 .Ldone:
602         pshufb          $T3,$Xi
603         movdqu          $Xi,($Xip)
604 ___
605 $code.=<<___ if ($win64);
606         movaps  (%rsp),%xmm6
607         movaps  0x10(%rsp),%xmm7
608         movaps  0x20(%rsp),%xmm8
609         movaps  0x30(%rsp),%xmm9
610         movaps  0x40(%rsp),%xmm10
611         add     \$0x58,%rsp
612 ___
613 $code.=<<___;
614         ret
615 .LSEH_end_gcm_ghash_clmul:
616 .size   gcm_ghash_clmul,.-gcm_ghash_clmul
617 ___
618 }

619 $code.=<<___;
620 .align  64
621 .Lbswap_mask:
622         .byte   15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0
623 .L0x1c2_polynomial:
624         .byte   1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0xc2
625 .align  64
626 .type   .Lrem_4bit,\@object
627 .Lrem_4bit:
628         .long   0,`0x0000<<16`,0,`0x1C20<<16`,0,`0x3840<<16`,0,`0x2460<<16`
629         .long   0,`0x7080<<16`,0,`0x6CA0<<16`,0,`0x48C0<<16`,0,`0x54E0<<16`
630         .long   0,`0xE100<<16`,0,`0xFD20<<16`,0,`0xD940<<16`,0,`0xC560<<16`
631         .long   0,`0x9180<<16`,0,`0x8DA0<<16`,0,`0xA9C0<<16`,0,`0xB5E0<<16`
632 .type   .Lrem_8bit,\@object
633 .Lrem_8bit:
634         .value  0x0000,0x01C2,0x0384,0x0246,0x0708,0x06CA,0x048C,0x054E
635         .value  0x0E10,0x0FD2,0x0D94,0x0C56,0x0918,0x08DA,0x0A9C,0x0B5E
636         .value  0x1C20,0x1DE2,0x1FA4,0x1E66,0x1B28,0x1AEA,0x18AC,0x196E
637         .value  0x1230,0x13F2,0x11B4,0x1076,0x1538,0x14FA,0x16BC,0x177E
638         .value  0x3840,0x3982,0x3BC4,0x3A06,0x3F48,0x3E8A,0x3CCC,0x3D0E
639         .value  0x3650,0x3792,0x35D4,0x3416,0x3158,0x309A,0x32DC,0x331E
640         .value  0x2460,0x25A2,0x27E4,0x2626,0x2368,0x22AA,0x20EC,0x212E
641         .value  0x2A70,0x2BB2,0x29F4,0x2836,0x2D78,0x2CBA,0x2EFC,0x2F3E
642         .value  0x7080,0x7142,0x7304,0x72C6,0x7788,0x764A,0x740C,0x75CE
643         .value  0x7E90,0x7F52,0x7D14,0x7CD6,0x7998,0x785A,0x7A1C,0x7BDE
644         .value  0x6CA0,0x6D62,0x6F24,0x6EE6,0x6BA8,0x6A6A,0x682C,0x69EE
645         .value  0x62B0,0x6372,0x6134,0x60F6,0x65B8,0x647A,0x663C,0x67FE
646         .value  0x48C0,0x4902,0x4B44,0x4A86,0x4FC8,0x4E0A,0x4C4C,0x4D8E
647         .value  0x46D0,0x4712,0x4554,0x4496,0x41D8,0x401A,0x425C,0x439E
648         .value  0x54E0,0x5522,0x5764,0x56A6,0x53E8,0x522A,0x506C,0x51AE
649         .value  0x5AF0,0x5B32,0x5974,0x58B6,0x5DF8,0x5C3A,0x5E7C,0x5FBE
650         .value  0xE100,0xE0C2,0xE284,0xE346,0xE608,0xE7CA,0xE58C,0xE44E
651         .value  0xEF10,0xEED2,0xEC94,0xED56,0xE818,0xE9DA,0xEB9C,0xEA5E
652         .value  0xFD20,0xFCE2,0xFEA4,0xFF66,0xFA28,0xFBEA,0xF9AC,0xF86E
653         .value  0xF330,0xF2F2,0xF0B4,0xF176,0xF438,0xF5FA,0xF7BC,0xF67E
654         .value  0xD940,0xD882,0xDAC4,0xDB06,0xDE48,0xDF8A,0xDDCC,0xDC0E
655         .value  0xD750,0xD692,0xD4D4,0xD516,0xD058,0xD19A,0xD3DC,0xD21E
656         .value  0xC560,0xC4A2,0xC6E4,0xC726,0xC268,0xC3AA,0xC1EC,0xC02E
657         .value  0xCB70,0xCAB2,0xC8F4,0xC936,0xCC78,0xCDBA,0xCFFC,0xCE3E
658         .value  0x9180,0x9042,0x9204,0x93C6,0x9688,0x974A,0x950C,0x94CE
659         .value  0x9F90,0x9E52,0x9C14,0x9DD6,0x9898,0x995A,0x9B1C,0x9ADE
```

```
660         .value  0x8DA0,0x8C62,0x8E24,0x8FE6,0x8AA8,0x8B6A,0x892C,0x88EE
661         .value  0x83B0,0x8272,0x8034,0x81F6,0x84B8,0x857A,0x873C,0x86FE
662         .value  0xA9C0,0xA802,0xAA44,0xAB86,0xAEC8,0xAF0A,0xAD4C,0xAC8E
663         .value  0xA7D0,0xA612,0xA454,0xA596,0xA0D8,0xA11A,0xA35C,0xA29E
664         .value  0xB5E0,0xB422,0xB664,0xB7A6,0xB2E8,0xB32A,0xB16C,0xB0AE
665         .value  0xBBF0,0xBA32,0xB874,0xB9B6,0xBCF8,0xBD3A,0xBF7C,0xBEBE

667 .asciz  "GHASH for x86_64, CRYPTOGAMS by <appro\@openssl.org>"
668 .align  64
669 ___
```



```
670 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
671 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
672 if ($win64) {
673 $rec="%rcx";
674 $frame="%rdx";
675 $context="%r8";
676 $disp="%r9";

678 $code.=<<___;
679 .extern __imp_RtlVirtualUnwind
680 .type   se_handler,\@abi-omnipotent
681 .align  16
682 se_handler:
683         push    %rsi
684         push    %rdi
685         push    %rbx
686         push    %rbp
687         push    %r12
688         push    %r13
689         push    %r14
690         push    %r15
691         pushfq
692         sub     \$64,%rsp

694         mov     120($context),%rax      # pull context->Rax
695         mov     248($context),%rbx      # pull context->Rip

697         mov     8($disp),%rsi           # disp->ImageBase
698         mov     56($disp),%r11          # disp->HandlerData

700         mov     0(%r11),%r10d           # HandlerData[0]
701         lea     (%rsi,%r10),%r10        # prologue label
702         cmp     %r10,%rbx               # context->Rip<prologue label
703         jb      .Lin_prologue

705         mov     152($context),%rax      # pull context->Rsp

707         mov     4(%r11),%r10d           # HandlerData[1]
708         lea     (%rsi,%r10),%r10        # epilogue label
709         cmp     %r10,%rbx               # context->Rip>=epilogue label
710         jae     .Lin_prologue

712         lea     24(%rax),%rax           # adjust "rsp"

714         mov     -8(%rax),%rbx
715         mov     -16(%rax),%rbp
716         mov     -24(%rax),%r12
717         mov     %rbx,144($context)      # restore context->Rbx
718         mov     %rbp,160($context)      # restore context->Rbp
719         mov     %r12,216($context)      # restore context->R12

721 .Lin_prologue:
722         mov     8(%rax),%rdi
723         mov     16(%rax),%rsi
724         mov     %rax,152($context)      # restore context->Rsp
725         mov     %rsi,168($context)      # restore context->Rsi
726         mov     %rdi,176($context)      # restore context->Rdi

728         mov     40($disp),%rdi          # disp->ContextRecord
729         mov     $context,%rsi           # context
730         mov     \$‘1232/8‘,%ecx         # sizeof(CONTEXT)
731         .long   0xa548f3fc              # cld; rep movsq

733         mov     $disp,%rsi
734         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
735         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
```

```
736         mov     0(%rsi),%r8             # arg3, disp->ControlPc
737         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
738         mov     40(%rsi),%r10           # disp->ContextRecord
739         lea     56(%rsi),%r11           # &disp->HandlerData
740         lea     24(%rsi),%r12           # &disp->EstablisherFrame
741         mov     %r10,32(%rsp)           # arg5
742         mov     %r11,40(%rsp)           # arg6
743         mov     %r12,48(%rsp)           # arg7
744         mov     %rcx,56(%rsp)           # arg8, (NULL)
745         call    *__imp_RtlVirtualUnwind(%rip)

747         mov     \$1,%eax                # ExceptionContinueSearch
748         add     \$64,%rsp
749         popfq
750         pop     %r15
751         pop     %r14
752         pop     %r13
753         pop     %r12
754         pop     %rbp
755         pop     %rbx
756         pop     %rdi
757         pop     %rsi
758         ret
759 .size   se_handler,.-se_handler

761 .section        .pdata
762 .align  4
763         .rva    .LSEH_begin_gcm_gmult_4bit
764         .rva    .LSEH_end_gcm_gmult_4bit
765         .rva    .LSEH_info_gcm_gmult_4bit

767         .rva    .LSEH_begin_gcm_ghash_4bit
768         .rva    .LSEH_end_gcm_ghash_4bit
769         .rva    .LSEH_info_gcm_ghash_4bit

771         .rva    .LSEH_begin_gcm_ghash_clmul
772         .rva    .LSEH_end_gcm_ghash_clmul
773         .rva    .LSEH_info_gcm_ghash_clmul

775 .section        .xdata
776 .align  8
777 .LSEH_info_gcm_gmult_4bit:
778         .byte   9,0,0,0
779         .rva    se_handler
780         .rva    .Lgmult_prologue,.Lgmult_epilogue       # HandlerData
781 .LSEH_info_gcm_ghash_4bit:
782         .byte   9,0,0,0
783         .rva    se_handler
784         .rva    .Lghash_prologue,.Lghash_epilogue       # HandlerData
785 .LSEH_info_gcm_ghash_clmul:
786         .byte   0x01,0x1f,0x0b,0x00
787         .byte   0x1f,0xa8,0x04,0x00     #movaps 0x40(rsp),xmm10
788         .byte   0x19,0x98,0x03,0x00     #movaps 0x30(rsp),xmm9
789         .byte   0x13,0x88,0x02,0x00     #movaps 0x20(rsp),xmm8
790         .byte   0x0d,0x78,0x01,0x00     #movaps 0x10(rsp),xmm7
791         .byte   0x08,0x68,0x00,0x00     #movaps (rsp),xmm6
792         .byte   0x04,0xa2,0x00,0x00     #sub    rsp,0x58
793 ___
794 }
```



```
795 $code =~ s/\`([^\`]*)\`/eval($1)/gem;

797 print $code;

799 close STDOUT;
800 #endif /* ! codereview */
```

```
**********************************************************
    6961 Wed Aug 13 19:53:07 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/keysets.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/perl

$NUMBER=0x01;
$UPPER=0x02;
$LOWER=0x04;
$UNDER=0x100;
$PUNCTUATION=0x200;
$WS=0x10;
$ESC=0x20;
$QUOTE=0x40;
$DQUOTE=0x400;
$COMMENT=0x80;
$FCOMMENT=0x800;
$EOF=0x08;
$HIGHBIT=0x1000;

foreach (0 .. 255)
        {
        $v=0;
        $c=sprintf("%c",$_);
        $v|=$NUMBER     if ($c =~ /[0-9]/);
        $v|=$UPPER      if ($c =~ /[A-Z]/);
        $v|=$LOWER      if ($c =~ /[a-z]/);
        $v|=$UNDER      if ($c =~ /_/);
        $v|=$PUNCTUATION if ($c =~ /[!\.%&\*\+,\/;\?\@\^\~\|-]/);
        $v|=$WS         if ($c =~ /[ \t\r\n]/);
        $v|=$ESC        if ($c =~ /\\/);
        $v|=$QUOTE      if ($c =~ /['"]/); # for emacs: "'}/)
        $v|=$COMMENT    if ($c =~ /\#/);
        $v|=$EOF        if ($c =~ /\0/);
        $v|=$HIGHBIT    if ($c =~/[\x80-\xff]/);

        push(@V_def,$v);
        }

foreach (0 .. 255)
        {
        $v=0;
        $c=sprintf("%c",$_);
        $v|=$NUMBER     if ($c =~ /[0-9]/);
        $v|=$UPPER      if ($c =~ /[A-Z]/);
        $v|=$LOWER      if ($c =~ /[a-z]/);
        $v|=$UNDER      if ($c =~ /_/);
        $v|=$PUNCTUATION if ($c =~ /[!\.%&\*\+,\/;\?\@\^\~\|-]/);
        $v|=$WS         if ($c =~ /[ \t\r\n]/);
        $v|=$DQUOTE     if ($c =~ /["]/); # for emacs: "}/)
        $v|=$FCOMMENT   if ($c =~ /;/);
        $v|=$EOF        if ($c =~ /\0/);
        $v|=$HIGHBIT    if ($c =~/[\x80-\xff]/);

        push(@V_w32,$v);
        }

print <<"EOF";
/* crypto/conf/conf_def.h */
/* Copyright (C) 1995-1998 Eric Young (eay\@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay\@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
```



```
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh\@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay\@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh\@cryptsoft.com)
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* THIS FILE WAS AUTOMAGICALLY GENERATED!
   Please modify and use keysets.pl to regenerate it. */

#define CONF_NUMBER             $NUMBER
#define CONF_UPPER              $UPPER
#define CONF_LOWER              $LOWER
#define CONF_UNDER              $UNDER
#define CONF_PUNCTUATION        $PUNCTUATION
#define CONF_WS                 $WS
#define CONF_ESC                $ESC
#define CONF_QUOTE              $QUOTE
#define CONF_DQUOTE             $DQUOTE
#define CONF_COMMENT            $COMMENT
#define CONF_FCOMMENT           $FCOMMENT
#define CONF_EOF                $EOF
```

```
#define CONF_HIGHBIT            $HIGHBIT
#define CONF_ALPHA              (CONF_UPPER|CONF_LOWER)
#define CONF_ALPHA_NUMERIC      (CONF_ALPHA|CONF_NUMBER|CONF_UNDER)
#define CONF_ALPHA_NUMERIC_PUNCT (CONF_ALPHA|CONF_NUMBER|CONF_UNDER| \\
                                  CONF_PUNCTUATION)

#define KEYTYPES(c)             ((unsigned short *)((c)->meth_data))
#ifndef CHARSET_EBCDIC
#define IS_COMMENT(c,a)         (KEYTYPES(c)[(a)&0xff]&CONF_COMMENT)
#define IS_FCOMMENT(c,a)        (KEYTYPES(c)[(a)&0xff]&CONF_FCOMMENT)
#define IS_EOF(c,a)             (KEYTYPES(c)[(a)&0xff]&CONF_EOF)
#define IS_ESC(c,a)             (KEYTYPES(c)[(a)&0xff]&CONF_ESC)
#define IS_NUMBER(c,a)          (KEYTYPES(c)[(a)&0xff]&CONF_NUMBER)
#define IS_WS(c,a)              (KEYTYPES(c)[(a)&0xff]&CONF_WS)
#define IS_ALPHA_NUMERIC(c,a)   (KEYTYPES(c)[(a)&0xff]&CONF_ALPHA_NUMERIC)
#define IS_ALPHA_NUMERIC_PUNCT(c,a) \\
                                (KEYTYPES(c)[(a)&0xff]&CONF_ALPHA_NUMERIC_PUNCT)
#define IS_QUOTE(c,a)           (KEYTYPES(c)[(a)&0xff]&CONF_QUOTE)
#define IS_DQUOTE(c,a)          (KEYTYPES(c)[(a)&0xff]&CONF_DQUOTE)
#define IS_HIGHBIT(c,a)         (KEYTYPES(c)[(a)&0xff]&CONF_HIGHBIT)

#else /*CHARSET_EBCDIC*/

#define IS_COMMENT(c,a)         (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_COMMENT)
#define IS_FCOMMENT(c,a)        (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_FCOMMENT)
#define IS_EOF(c,a)             (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_EOF)
#define IS_ESC(c,a)             (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_ESC)
#define IS_NUMBER(c,a)          (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_NUMBER)
#define IS_WS(c,a)              (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_WS)
#define IS_ALPHA_NUMERIC(c,a)   (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_ALPHA_NUME
#define IS_ALPHA_NUMERIC_PUNCT(c,a) \\
                                (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_ALPHA_NUME
#define IS_QUOTE(c,a)           (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_QUOTE)
#define IS_DQUOTE(c,a)          (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_DQUOTE)
#define IS_HIGHBIT(c,a)         (KEYTYPES(c)[os_toascii[a]&0xff]&CONF_HIGHBIT)
#endif /*CHARSET_EBCDIC*/

EOF

print "static unsigned short CONF_type_default[256]={";

for ($i=0; $i<256; $i++)
        {
        print "\n\t" if ($i % 8) == 0;
        printf "0x%04X,",$V_def[$i];
        }

print "\n\t};\n\n";

print "static unsigned short CONF_type_win32[256]={";

for ($i=0; $i<256; $i++)
        {
        print "\n\t" if ($i % 8) == 0;
        printf "0x%04X,",$V_w32[$i];
        }

print "\n\t};\n\n";
#endif /* ! codereview */
```

```
**********************************************************
    7610 Wed Aug 13 19:53:08 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/md5-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl

# Normal is the
# md5_block_x86(MD5_CTX *c, ULONG *X);
# version, non-normal is the
# md5_block_x86(MD5_CTX *c, ULONG *X,int blocks);

$normal=0;

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],$0);

$A="eax";
$B="ebx";
$C="ecx";
$D="edx";
$tmp1="edi";
$tmp2="ebp";
$X="esi";

# What we need to load into $tmp for the next round
%Ltmp1=("R0",&Np($C), "R1",&Np($C), "R2",&Np($C), "R3",&Np($D));
@xo=(
 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,  # R0
 1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12,  # R1
 5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2,  # R2
 0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9,  # R3
 );

&md5_block("md5_block_asm_data_order");
&asm_finish();

sub Np
        {
        local($p)=@_;
        local(%n)=($A,$D,$B,$A,$C,$B,$D,$C);
        return($n{$p});
        }

sub R0
        {
        local($pos,$a,$b,$c,$d,$K,$ki,$s,$t)=@_;

        &mov($tmp1,$C)  if $pos < 0;
        &mov($tmp2,&DWP($xo[$ki]*4,$K,"",0)) if $pos < 0; # very first one

        # body proper

        &comment("R0 $ki");
        &xor($tmp1,$d); # F function - part 2

        &and($tmp1,$b); # F function - part 3
        &lea($a,&DWP($t,$a,$tmp2,1));

        &xor($tmp1,$d); # F function - part 4

        &add($a,$tmp1);
        &mov($tmp1,&Np($c)) if $pos < 1;        # next tmp1 for R0
```



```
        &mov($tmp1,&Np($c)) if $pos == 1;       # next tmp1 for R1

        &rotl($a,$s);

        &mov($tmp2,&DWP($xo[$ki+1]*4,$K,"",0)) if ($pos != 2);

        &add($a,$b);
        }

sub R1
        {
        local($pos,$a,$b,$c,$d,$K,$ki,$s,$t)=@_;

        &comment("R1 $ki");

        &lea($a,&DWP($t,$a,$tmp2,1));

        &xor($tmp1,$b); # G function - part 2
        &and($tmp1,$d); # G function - part 3

        &mov($tmp2,&DWP($xo[$ki+1]*4,$K,"",0)) if ($pos != 2);
        &xor($tmp1,$c);                 # G function - part 4

        &add($a,$tmp1);
        &mov($tmp1,&Np($c)) if $pos < 1;        # G function - part 1
        &mov($tmp1,&Np($c)) if $pos == 1;       # G function - part 1

        &rotl($a,$s);

        &add($a,$b);
        }

sub R2
        {
        local($n,$pos,$a,$b,$c,$d,$K,$ki,$s,$t)=@_;
        # This one is different, only 3 logical operations

if (($n & 1) == 0)
        {
        &comment("R2 $ki");
        # make sure to do 'D' first, not 'B', else we clash with
        # the last add from the previous round.

        &xor($tmp1,$d); # H function - part 2

        &xor($tmp1,$b); # H function - part 3
        &lea($a,&DWP($t,$a,$tmp2,1));

        &add($a,$tmp1);

        &rotl($a,$s);

        &mov($tmp2,&DWP($xo[$ki+1]*4,$K,"",0));
        &mov($tmp1,&Np($c));
        }
else
        {
        &comment("R2 $ki");
        # make sure to do 'D' first, not 'B', else we clash with
        # the last add from the previous round.

        &lea($a,&DWP($t,$a,$tmp2,1));

        &add($b,$c);                    # MOVED FORWARD
        &xor($tmp1,$d); # H function - part 2
```

```
128         &xor($tmp1,$b); # H function - part 3
129         &mov($tmp2,&DWP($xo[$ki+1]*4,$K,"",0)) if ($pos != 2);

131         &add($a,$tmp1);
132         &mov($tmp1,&Np($c)) if $pos < 1;        # H function - part 1
133         &mov($tmp1,-1) if $pos == 1;            # I function - part 1

135         &rotl($a,$s);

137         &add($a,$b);
138         }
139         }

141 sub R3
142         {
143         local($pos,$a,$b,$c,$d,$K,$ki,$s,$t)=@_;

145         &comment("R3 $ki");

147         # &not($tmp1)
148         &xor($tmp1,$d) if $pos < 0;     # I function - part 2

150         &or($tmp1,$b);                          # I function - part 3
151         &lea($a,&DWP($t,$a,$tmp2,1));

153         &xor($tmp1,$c);                         # I function - part 4
154         &mov($tmp2,&DWP($xo[$ki+1]*4,$K,"",0))  if $pos != 2; # load X/k value
155         &mov($tmp2,&wparam(0)) if $pos == 2;

157         &add($a,$tmp1);
158         &mov($tmp1,-1) if $pos < 1;     # H function - part 1
159         &add($K,64) if $pos >=1 && !$normal;

161         &rotl($a,$s);

163         &xor($tmp1,&Np($d)) if $pos <= 0;       # I function - part = first time
164         &mov($tmp1,&DWP( 0,$tmp2,"",0)) if $pos > 0;
165         &add($a,$b);
166         }


169 sub md5_block
170         {
171         local($name)=@_;

173         &function_begin_B($name,"",3);

175         # parameter 1 is the MD5_CTX structure.
176         # A     0
177         # B     4
178         # C     8
179         # D     12

181         &push("esi");
182          &push("edi");
183         &mov($tmp1,     &wparam(0)); # edi
184          &mov($X,       &wparam(1)); # esi
185         &mov($C,        &wparam(2));
186          &push("ebp");
187         &shl($C,        6);
188         &push("ebx");
189          &add($C,       $X); # offset we end at
190         &sub($C,        64);
191          &mov($A,       &DWP( 0,$tmp1,"",0));
192         &push($C);      # Put on the TOS
193          &mov($B,       &DWP( 4,$tmp1,"",0));
```



```
194         &mov($C,        &DWP( 8,$tmp1,"",0));
195          &mov($D,       &DWP(12,$tmp1,"",0));

197         &set_label("start") unless $normal;
198         &comment("");
199         &comment("R0 section");

201         &R0(-2,$A,$B,$C,$D,$X, 0, 7,0xd76aa478);
202         &R0( 0,$D,$A,$B,$C,$X, 1,12,0xe8c7b756);
203         &R0( 0,$C,$D,$A,$B,$X, 2,17,0x242070db);
204         &R0( 0,$B,$C,$D,$A,$X, 3,22,0xc1bdceee);
205         &R0( 0,$A,$B,$C,$D,$X, 4, 7,0xf57c0faf);
206         &R0( 0,$D,$A,$B,$C,$X, 5,12,0x4787c62a);
207         &R0( 0,$C,$D,$A,$B,$X, 6,17,0xa8304613);
208         &R0( 0,$B,$C,$D,$A,$X, 7,22,0xfd469501);
209         &R0( 0,$A,$B,$C,$D,$X, 8, 7,0x698098d8);
210         &R0( 0,$D,$A,$B,$C,$X, 9,12,0x8b44f7af);
211         &R0( 0,$C,$D,$A,$B,$X,10,17,0xffff5bb1);
212         &R0( 0,$B,$C,$D,$A,$X,11,22,0x895cd7be);
213         &R0( 0,$A,$B,$C,$D,$X,12, 7,0x6b901122);
214         &R0( 0,$D,$A,$B,$C,$X,13,12,0xfd987193);
215         &R0( 0,$C,$D,$A,$B,$X,14,17,0xa679438e);
216         &R0( 1,$B,$C,$D,$A,$X,15,22,0x49b40821);

218         &comment("");
219         &comment("R1 section");
220         &R1(-1,$A,$B,$C,$D,$X,16, 5,0xf61e2562);
221         &R1( 0,$D,$A,$B,$C,$X,17, 9,0xc040b340);
222         &R1( 0,$C,$D,$A,$B,$X,18,14,0x265e5a51);
223         &R1( 0,$B,$C,$D,$A,$X,19,20,0xe9b6c7aa);
224         &R1( 0,$A,$B,$C,$D,$X,20, 5,0xd62f105d);
225         &R1( 0,$D,$A,$B,$C,$X,21, 9,0x02441453);
226         &R1( 0,$C,$D,$A,$B,$X,22,14,0xd8a1e681);
227         &R1( 0,$B,$C,$D,$A,$X,23,20,0xe7d3fbc8);
228         &R1( 0,$A,$B,$C,$D,$X,24, 5,0x21e1cde6);
229         &R1( 0,$D,$A,$B,$C,$X,25, 9,0xc33707d6);
230         &R1( 0,$C,$D,$A,$B,$X,26,14,0xf4d50d87);
231         &R1( 0,$B,$C,$D,$A,$X,27,20,0x455a14ed);
232         &R1( 0,$A,$B,$C,$D,$X,28, 5,0xa9e3e905);
233         &R1( 0,$D,$A,$B,$C,$X,29, 9,0xfcefa3f8);
234         &R1( 0,$C,$D,$A,$B,$X,30,14,0x676f02d9);
235         &R1( 1,$B,$C,$D,$A,$X,31,20,0x8d2a4c8a);

237         &comment("");
238         &comment("R2 section");
239         &R2( 0,-1,$A,$B,$C,$D,$X,32, 4,0xfffa3942);
240         &R2( 1, 0,$D,$A,$B,$C,$X,33,11,0x8771f681);
241         &R2( 2, 0,$C,$D,$A,$B,$X,34,16,0x6d9d6122);
242         &R2( 3, 0,$B,$C,$D,$A,$X,35,23,0xfde5380c);
243         &R2( 4, 0,$A,$B,$C,$D,$X,36, 4,0xa4beea44);
244         &R2( 5, 0,$D,$A,$B,$C,$X,37,11,0x4bdecfa9);
245         &R2( 6, 0,$C,$D,$A,$B,$X,38,16,0xf6bb4b60);
246         &R2( 7, 0,$B,$C,$D,$A,$X,39,23,0xbebfbc70);
247         &R2( 8, 0,$A,$B,$C,$D,$X,40, 4,0x289b7ec6);
248         &R2( 9, 0,$D,$A,$B,$C,$X,41,11,0xeaa127fa);
249         &R2(10, 0,$C,$D,$A,$B,$X,42,16,0xd4ef3085);
250         &R2(11, 0,$B,$C,$D,$A,$X,43,23,0x04881d05);
251         &R2(12, 0,$A,$B,$C,$D,$X,44, 4,0xd9d4d039);
252         &R2(13, 0,$D,$A,$B,$C,$X,45,11,0xe6db99e5);
253         &R2(14, 0,$C,$D,$A,$B,$X,46,16,0x1fa27cf8);
254         &R2(15, 1,$B,$C,$D,$A,$X,47,23,0xc4ac5665);

256         &comment("");
257         &comment("R3 section");
258         &R3(-1,$A,$B,$C,$D,$X,48, 6,0xf4292244);
259         &R3( 0,$D,$A,$B,$C,$X,49,10,0x432aff97);
```

```
260         &R3( 0,$C,$D,$A,$B,$X,50,15,0xab9423a7);
261         &R3( 0,$B,$C,$D,$A,$X,51,21,0xfc93a039);
262         &R3( 0,$A,$B,$C,$D,$X,52, 6,0x655b59c3);
263         &R3( 0,$D,$A,$B,$C,$X,53,10,0x8f0ccc92);
264         &R3( 0,$C,$D,$A,$B,$X,54,15,0xffeff47d);
265         &R3( 0,$B,$C,$D,$A,$X,55,21,0x85845dd1);
266         &R3( 0,$A,$B,$C,$D,$X,56, 6,0x6fa87e4f);
267         &R3( 0,$D,$A,$B,$C,$X,57,10,0xfe2ce6e0);
268         &R3( 0,$C,$D,$A,$B,$X,58,15,0xa3014314);
269         &R3( 0,$B,$C,$D,$A,$X,59,21,0x4e0811a1);
270         &R3( 0,$A,$B,$C,$D,$X,60, 6,0xf7537e82);
271         &R3( 0,$D,$A,$B,$C,$X,61,10,0xbd3af235);
272         &R3( 0,$C,$D,$A,$B,$X,62,15,0x2ad7d2bb);
273         &R3( 2,$B,$C,$D,$A,$X,63,21,0xeb86d391);

275         # &mov($tmp2,&wparam(0));       # done in the last R3
276         # &mov($tmp1,   &DWP( 0,$tmp2,"",0)); # done is the last R3

278         &add($A,$tmp1);
279          &mov($tmp1,    &DWP( 4,$tmp2,"",0));

281         &add($B,$tmp1);
282         &mov($tmp1,     &DWP( 8,$tmp2,"",0));

284         &add($C,$tmp1);
285         &mov($tmp1,     &DWP(12,$tmp2,"",0));

287         &add($D,$tmp1);
288         &mov(&DWP( 0,$tmp2,"",0),$A);

290         &mov(&DWP( 4,$tmp2,"",0),$B);
291         &mov($tmp1,&swtmp(0)) unless $normal;

293         &mov(&DWP( 8,$tmp2,"",0),$C);
294          &mov(&DWP(12,$tmp2,"",0),$D);

296         &cmp($tmp1,$X) unless $normal;                  # check count
297          &jae(&label("start")) unless $normal;

299         &pop("eax"); # pop the temp variable off the stack
300          &pop("ebx");
301         &pop("ebp");
302          &pop("edi");
303         &pop("esi");
304          &ret();
305         &function_end_B($name);
306         }
307 #endif /* ! codereview */
```

```
**********************************************************
   12448 Wed Aug 13 19:53:08 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/md5-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/perl -w
#
# MD5 optimized for AMD64.
#
# Author: Marc Bevand <bevand_m (at) epita.fr>
# Licence: I hereby disclaim the copyright on this code and place it
# in the public domain.
#

use strict;

my $code;

# round1_step() does:
#   dst = x + ((dst + F(x,y,z) + X[k] + T_i) <<< s)
#   %r10d = X[k_next]
#   %r11d = z' (copy of z for the next step)
# Each round1_step() takes about 5.3 clocks (9 instructions, 1.7 IPC)
sub round1_step
{
    my ($pos, $dst, $x, $y, $z, $k_next, $T_i, $s) = @_;
    $code .= " mov      0*4(%rsi),      %r10d           /* (NEXT STEP) X[0] */\n
    $code .= " mov      %edx,           %r11d           /* (NEXT STEP) z' = %edx
    $code .= <<EOF;
        xor     $y,             %r11d           /* y ^ ... */
        lea     $T_i($dst,%r10d),$dst           /* Const + dst + ... */
        and     $x,             %r11d           /* x & ... */
        xor     $z,             %r11d           /* z ^ ... */
        mov     $k_next*4(%rsi),%r10d           /* (NEXT STEP) X[$k_next] */
        add     %r11d,          $dst            /* dst += ... */
        rol     \$$s,           $dst            /* dst <<< s */
        mov     $y,             %r11d           /* (NEXT STEP) z' = $y */
        add     $x,             $dst            /* dst += x */
EOF
}

# round2_step() does:
#   dst = x + ((dst + G(x,y,z) + X[k] + T_i) <<< s)
#   %r10d = X[k_next]
#   %r11d = z' (copy of z for the next step)
#   %r12d = z' (copy of z for the next step)
# Each round2_step() takes about 5.4 clocks (11 instructions, 2.0 IPC)
sub round2_step
{
    my ($pos, $dst, $x, $y, $z, $k_next, $T_i, $s) = @_;
    $code .= " mov      1*4(%rsi),      %r10d           /* (NEXT STEP) X[1] */\n
    $code .= " mov      %edx,           %r11d           /* (NEXT STEP) z' = %edx
    $code .= " mov      %edx,           %r12d           /* (NEXT STEP) z' = %edx
    $code .= <<EOF;
        not     %r11d                           /* not z */
        lea     $T_i($dst,%r10d),$dst           /* Const + dst + ... */
        and     $x,             %r12d           /* x & z */
        and     $y,             %r11d           /* y & (not z) */
        mov     $k_next*4(%rsi),%r10d           /* (NEXT STEP) X[$k_next] */
        or      %r11d,          %r12d           /* (y & (not z)) | (x & z) */
        mov     $y,             %r11d           /* (NEXT STEP) z' = $y */
        add     %r12d,          $dst            /* dst += ... */
        mov     $y,             %r12d           /* (NEXT STEP) z' = $y */
        rol     \$$s,           $dst            /* dst <<< s */
        add     $x,             $dst            /* dst += x */
EOF
```



```
}

# round3_step() does:
#   dst = x + ((dst + H(x,y,z) + X[k] + T_i) <<< s)
#   %r10d = X[k_next]
#   %r11d = y' (copy of y for the next step)
# Each round3_step() takes about 4.2 clocks (8 instructions, 1.9 IPC)
sub round3_step
{
    my ($pos, $dst, $x, $y, $z, $k_next, $T_i, $s) = @_;
    $code .= " mov      5*4(%rsi),      %r10d           /* (NEXT STEP) X[5] */\n
    $code .= " mov      %ecx,           %r11d           /* (NEXT STEP) y' = %ecx
    $code .= <<EOF;
        lea     $T_i($dst,%r10d),$dst           /* Const + dst + ... */
        mov     $k_next*4(%rsi),%r10d           /* (NEXT STEP) X[$k_next] */
        xor     $z,             %r11d           /* z ^ ... */
        xor     $x,             %r11d           /* x ^ ... */
        add     %r11d,          $dst            /* dst += ... */
        rol     \$$s,           $dst            /* dst <<< s */
        mov     $x,             %r11d           /* (NEXT STEP) y' = $x */
        add     $x,             $dst            /* dst += x */
EOF
}

# round4_step() does:
#   dst = x + ((dst + I(x,y,z) + X[k] + T_i) <<< s)
#   %r10d = X[k_next]
#   %r11d = not z' (copy of not z for the next step)
# Each round4_step() takes about 5.2 clocks (9 instructions, 1.7 IPC)
sub round4_step
{
    my ($pos, $dst, $x, $y, $z, $k_next, $T_i, $s) = @_;
    $code .= " mov      0*4(%rsi),      %r10d           /* (NEXT STEP) X[0] */\n
    $code .= " mov      \$0xffffffff,   %r11d\n" if ($pos == -1);
    $code .= " xor      %edx,           %r11d           /* (NEXT STEP) not z' =
    if ($pos == -1);
    $code .= <<EOF;
        lea     $T_i($dst,%r10d),$dst           /* Const + dst + ... */
        or      $x,             %r11d           /* x | ... */
        xor     $y,             %r11d           /* y ^ ... */
        add     %r11d,          $dst            /* dst += ... */
        mov     $k_next*4(%rsi),%r10d           /* (NEXT STEP) X[$k_next] */
        mov     \$0xffffffff,   %r11d
        rol     \$$s,           $dst            /* dst <<< s */
        xor     $y,             %r11d           /* (NEXT STEP) not z' = not $y *
        add     $x,             $dst            /* dst += x */
EOF
}

my $flavour = shift;
my $output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

my $win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; my $dir=$1; my $xlate;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

no warnings qw(uninitialized);
open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

$code .= <<EOF;
.text
```

```
.align 16

.globl md5_block_asm_data_order
.type md5_block_asm_data_order,\@function,3
md5_block_asm_data_order:
        push    %rbp
        push    %rbx
        push    %r12
        push    %r14
        push    %r15
.Lprologue:

        # rdi = arg #1 (ctx, MD5_CTX pointer)
        # rsi = arg #2 (ptr, data pointer)
        # rdx = arg #3 (nbr, number of 16-word blocks to process)
        mov     %rdi,           %rbp    # rbp = ctx
        shl     \$6,            %rdx    # rdx = nbr in bytes
        lea     (%rsi,%rdx),    %rdi    # rdi = end
        mov     0*4(%rbp),      %eax    # eax = ctx->A
        mov     1*4(%rbp),      %ebx    # ebx = ctx->B
        mov     2*4(%rbp),      %ecx    # ecx = ctx->C
        mov     3*4(%rbp),      %edx    # edx = ctx->D
        # end is 'rdi'
        # ptr is 'rsi'
        # A is 'eax'
        # B is 'ebx'
        # C is 'ecx'
        # D is 'edx'

        cmp     %rdi,           %rsi            # cmp end with ptr
        je      .Lend                           # jmp if ptr == end

        # BEGIN of loop over 16-word blocks
.Lloop: # save old values of A, B, C, D
        mov     %eax,           %r8d
        mov     %ebx,           %r9d
        mov     %ecx,           %r14d
        mov     %edx,           %r15d
EOF
round1_step(-1,'%eax','%ebx','%ecx','%edx', '1','0xd76aa478', '7');
round1_step( 0,'%edx','%eax','%ebx','%ecx', '2','0xe8c7b756','12');
round1_step( 0,'%ecx','%edx','%eax','%ebx', '3','0x242070db','17');
round1_step( 0,'%ebx','%ecx','%edx','%eax', '4','0xc1bdceee','22');
round1_step( 0,'%eax','%ebx','%ecx','%edx', '5','0xf57c0faf', '7');
round1_step( 0,'%edx','%eax','%ebx','%ecx', '6','0x4787c62a','12');
round1_step( 0,'%ecx','%edx','%eax','%ebx', '7','0xa8304613','17');
round1_step( 0,'%ebx','%ecx','%edx','%eax', '8','0xfd469501','22');
round1_step( 0,'%eax','%ebx','%ecx','%edx', '9','0x698098d8', '7');
round1_step( 0,'%edx','%eax','%ebx','%ecx','10','0x8b44f7af','12');
round1_step( 0,'%ecx','%edx','%eax','%ebx','11','0xffff5bb1','17');
round1_step( 0,'%ebx','%ecx','%edx','%eax','12','0x895cd7be','22');
round1_step( 0,'%eax','%ebx','%ecx','%edx','13','0x6b901122', '7');
round1_step( 0,'%edx','%eax','%ebx','%ecx','14','0xfd987193','12');
round1_step( 0,'%ecx','%edx','%eax','%ebx','15','0xa679438e','17');
round1_step( 1,'%ebx','%ecx','%edx','%eax', '0','0x49b40821','22');

round2_step(-1,'%eax','%ebx','%ecx','%edx', '6','0xf61e2562', '5');
round2_step( 0,'%edx','%eax','%ebx','%ecx','11','0xc040b340', '9');
round2_step( 0,'%ecx','%edx','%eax','%ebx', '0','0x265e5a51','14');
round2_step( 0,'%ebx','%ecx','%edx','%eax', '5','0xe9b6c7aa','20');
round2_step( 0,'%eax','%ebx','%ecx','%edx','10','0xd62f105d', '5');
round2_step( 0,'%edx','%eax','%ebx','%ecx','15', '0x2441453', '9');
round2_step( 0,'%ecx','%edx','%eax','%ebx', '4','0xd8a1e681','14');
round2_step( 0,'%ebx','%ecx','%edx','%eax', '9','0xe7d3fbc8','20');
round2_step( 0,'%eax','%ebx','%ecx','%edx','14','0x21e1cde6', '5');
round2_step( 0,'%edx','%eax','%ebx','%ecx', '3','0xc33707d6', '9');
```



```
round2_step( 0,'%ecx','%edx','%eax','%ebx', '8','0xf4d50d87','14');
round2_step( 0,'%ebx','%ecx','%edx','%eax','13','0x455a14ed','20');
round2_step( 0,'%eax','%ebx','%ecx','%edx', '2','0xa9e3e905', '5');
round2_step( 0,'%edx','%eax','%ebx','%ecx', '7','0xfcefa3f8', '9');
round2_step( 0,'%ecx','%edx','%eax','%ebx','12','0x676f02d9','14');
round2_step( 1,'%ebx','%ecx','%edx','%eax', '0','0x8d2a4c8a','20');

round3_step(-1,'%eax','%ebx','%ecx','%edx', '8','0xfffa3942', '4');
round3_step( 0,'%edx','%eax','%ebx','%ecx','11','0x8771f681','11');
round3_step( 0,'%ecx','%edx','%eax','%ebx','14','0x6d9d6122','16');
round3_step( 0,'%ebx','%ecx','%edx','%eax', '1','0xfde5380c','23');
round3_step( 0,'%eax','%ebx','%ecx','%edx', '4','0xa4beea44', '4');
round3_step( 0,'%edx','%eax','%ebx','%ecx', '7','0x4bdecfa9','11');
round3_step( 0,'%ecx','%edx','%eax','%ebx','10','0xf6bb4b60','16');
round3_step( 0,'%ebx','%ecx','%edx','%eax','13','0xbebfbc70','23');
round3_step( 0,'%eax','%ebx','%ecx','%edx', '0','0x289b7ec6', '4');
round3_step( 0,'%edx','%eax','%ebx','%ecx', '3','0xeaa127fa','11');
round3_step( 0,'%ecx','%edx','%eax','%ebx', '6','0xd4ef3085','16');
round3_step( 0,'%ebx','%ecx','%edx','%eax', '9', '0x4881d05','23');
round3_step( 0,'%eax','%ebx','%ecx','%edx','12','0xd9d4d039', '4');
round3_step( 0,'%edx','%eax','%ebx','%ecx','15','0xe6db99e5','11');
round3_step( 0,'%ecx','%edx','%eax','%ebx', '2','0x1fa27cf8','16');
round3_step( 1,'%ebx','%ecx','%edx','%eax', '0','0xc4ac5665','23');

round4_step(-1,'%eax','%ebx','%ecx','%edx', '7','0xf4292244', '6');
round4_step( 0,'%edx','%eax','%ebx','%ecx','14','0x432aff97','10');
round4_step( 0,'%ecx','%edx','%eax','%ebx', '5','0xab9423a7','15');
round4_step( 0,'%ebx','%ecx','%edx','%eax','12','0xfc93a039','21');
round4_step( 0,'%eax','%ebx','%ecx','%edx', '3','0x655b59c3', '6');
round4_step( 0,'%edx','%eax','%ebx','%ecx','10','0x8f0ccc92','10');
round4_step( 0,'%ecx','%edx','%eax','%ebx', '1','0xffeff47d','15');
round4_step( 0,'%ebx','%ecx','%edx','%eax', '8','0x85845dd1','21');
round4_step( 0,'%eax','%ebx','%ecx','%edx','15','0x6fa87e4f', '6');
round4_step( 0,'%edx','%eax','%ebx','%ecx', '6','0xfe2ce6e0','10');
round4_step( 0,'%ecx','%edx','%eax','%ebx','13','0xa3014314','15');
round4_step( 0,'%ebx','%ecx','%edx','%eax', '4','0x4e0811a1','21');
round4_step( 0,'%eax','%ebx','%ecx','%edx','11','0xf7537e82', '6');
round4_step( 0,'%edx','%eax','%ebx','%ecx', '2','0xbd3af235','10');
round4_step( 0,'%ecx','%edx','%eax','%ebx', '9','0x2ad7d2bb','15');
round4_step( 1,'%ebx','%ecx','%edx','%eax', '0','0xeb86d391','21');
$code .= <<EOF;
        # add old values of A, B, C, D
        add     %r8d,   %eax
        add     %r9d,   %ebx
        add     %r14d,  %ecx
        add     %r15d,  %edx

        # loop control
        add     \$64,           %rsi            # ptr += 64
        cmp     %rdi,           %rsi            # cmp end with ptr
        jb      .Lloop                          # jmp if ptr < end
        # END of loop over 16-word blocks

.Lend:
        mov     %eax,           0*4(%rbp)       # ctx->A = A
        mov     %ebx,           1*4(%rbp)       # ctx->B = B
        mov     %ecx,           2*4(%rbp)       # ctx->C = C
        mov     %edx,           3*4(%rbp)       # ctx->D = D

        mov     (%rsp),%r15
        mov     8(%rsp),%r14
        mov     16(%rsp),%r12
        mov     24(%rsp),%rbx
        mov     32(%rsp),%rbp
        add     \$40,%rsp
.Lepilogue:
```

```
260         ret
261 .size md5_block_asm_data_order,.-md5_block_asm_data_order
262 EOF

264 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
265 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
266 if ($win64) {
267 my $rec="%rcx";
268 my $frame="%rdx";
269 my $context="%r8";
270 my $disp="%r9";

272 $code.=<<___;
273 .extern __imp_RtlVirtualUnwind
274 .type   se_handler,\@abi-omnipotent
275 .align  16
276 se_handler:
277         push    %rsi
278         push    %rdi
279         push    %rbx
280         push    %rbp
281         push    %r12
282         push    %r13
283         push    %r14
284         push    %r15
285         pushfq
286         sub     \$64,%rsp

288         mov     120($context),%rax      # pull context->Rax
289         mov     248($context),%rbx      # pull context->Rip

291         lea     .Lprologue(%rip),%r10
292         cmp     %r10,%rbx               # context->Rip<.Lprologue
293         jb      .Lin_prologue

295         mov     152($context),%rax      # pull context->Rsp

297         lea     .Lepilogue(%rip),%r10
298         cmp     %r10,%rbx               # context->Rip>=.Lepilogue
299         jae     .Lin_prologue

301         lea     40(%rax),%rax

303         mov     -8(%rax),%rbp
304         mov     -16(%rax),%rbx
305         mov     -24(%rax),%r12
306         mov     -32(%rax),%r14
307         mov     -40(%rax),%r15
308         mov     %rbx,144($context)      # restore context->Rbx
309         mov     %rbp,160($context)      # restore context->Rbp
310         mov     %r12,216($context)      # restore context->R12
311         mov     %r14,232($context)      # restore context->R14
312         mov     %r15,240($context)      # restore context->R15

314 .Lin_prologue:
315         mov     8(%rax),%rdi
316         mov     16(%rax),%rsi
317         mov     %rax,152($context)      # restore context->Rsp
318         mov     %rsi,168($context)      # restore context->Rsi
319         mov     %rdi,176($context)      # restore context->Rdi

321         mov     40($disp),%rdi          # disp->ContextRecord
322         mov     $context,%rsi           # context
323         mov     \$154,%ecx              # sizeof(CONTEXT)
324         .long   0xa548f3fc              # cld; rep movsq
```



```
326         mov     $disp,%rsi
327         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
328         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
329         mov     0(%rsi),%r8             # arg3, disp->ControlPc
330         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
331         mov     40(%rsi),%r10           # disp->ContextRecord
332         lea     56(%rsi),%r11           # &disp->HandlerData
333         lea     24(%rsi),%r12           # &disp->EstablisherFrame
334         mov     %r10,32(%rsp)           # arg5
335         mov     %r11,40(%rsp)           # arg6
336         mov     %r12,48(%rsp)           # arg7
337         mov     %rcx,56(%rsp)           # arg8, (NULL)
338         call    *__imp_RtlVirtualUnwind(%rip)

340         mov     \$1,%eax                # ExceptionContinueSearch
341         add     \$64,%rsp
342         popfq
343         pop     %r15
344         pop     %r14
345         pop     %r13
346         pop     %r12
347         pop     %rbp
348         pop     %rbx
349         pop     %rdi
350         pop     %rsi
351         ret
352 .size   se_handler,.-se_handler

354 .section        .pdata
355 .align  4
356         .rva    .LSEH_begin_md5_block_asm_data_order
357         .rva    .LSEH_end_md5_block_asm_data_order
358         .rva    .LSEH_info_md5_block_asm_data_order

360 .section        .xdata
361 .align  8
362 .LSEH_info_md5_block_asm_data_order:
363         .byte   9,0,0,0
364         .rva    se_handler
365 ___
366 }

368 print $code;

370 close STDOUT;
371 #endif /* ! codereview */
```

```
**********************************************************
   34519 Wed Aug 13 19:53:08 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/modexp512-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# Copyright (c) 2010-2011 Intel Corp.
#   Author: Vinodh.Gopal@intel.com
#           Jim Guilford
#           Erdinc.Ozturk@intel.com
#           Maxim.Perminov@intel.com
#
# More information about algorithm used can be found at:
#   http://www.cse.buffalo.edu/srds2009/escs2009_submission_Gopal.pdf
#
# ====================================================================
# Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
#
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
#
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in
#    the documentation and/or other materials provided with the
#    distribution.
#
# 3. All advertising materials mentioning features or use of this
#    software must display the following acknowledgment:
#    "This product includes software developed by the OpenSSL Project
#    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
#
# 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
#    endorse or promote products derived from this software without
#    prior written permission. For written permission, please contact
#    licensing@OpenSSL.org.
#
# 5. Products derived from this software may not be called "OpenSSL"
#    nor may "OpenSSL" appear in their names without prior written
#    permission of the OpenSSL Project.
#
# 6. Redistributions of any form whatsoever must retain the following
#    acknowledgment:
#    "This product includes software developed by the OpenSSL Project
#    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
#
# THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
# EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
# PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
# ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
# SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
# NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
# STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
# ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
# OF THE POSSIBILITY OF SUCH DAMAGE.
# ====================================================================

$flavour = shift;
$output  = shift;
```



```
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

my $win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

use strict;
my $code=".text\n\n";
my $m=0;

#
# Define x512 macros
#

#MULSTEP_512_ADD        MACRO   x7, x6, x5, x4, x3, x2, x1, x0, dst, src1, src2,
#
# uses rax, rdx, and args
sub MULSTEP_512_ADD
{
 my ($x, $DST, $SRC2, $ASRC, $OP, $TMP)=@_;
 my @X=@$x;     # make a copy
$code.=<<___;
        mov     (+8*0)($SRC2), %rax
        mul     $OP                     # rdx:rax = %OP * [0]
        mov     ($ASRC), $X[0]
        add     %rax, $X[0]
        adc     \$0, %rdx
        mov     $X[0], $DST
___
for(my $i=1;$i<8;$i++) {
$code.=<<___;
        mov     %rdx, $TMP

        mov     (+8*$i)($SRC2), %rax
        mul     $OP                     # rdx:rax = %OP * [$i]
        mov     (+8*$i)($ASRC), $X[$i]
        add     %rax, $X[$i]
        adc     \$0, %rdx
        add     $TMP, $X[$i]
        adc     \$0, %rdx
___
}
$code.=<<___;
        mov     %rdx, $X[0]
___
}

#MULSTEP_512    MACRO   x7, x6, x5, x4, x3, x2, x1, x0, dst, src2, src1_val, tmp
#
# uses rax, rdx, and args
sub MULSTEP_512
{
 my ($x, $DST, $SRC2, $OP, $TMP)=@_;
 my @X=@$x;     # make a copy
$code.=<<___;
        mov     (+8*0)($SRC2), %rax
        mul     $OP                     # rdx:rax = %OP * [0]
        add     %rax, $X[0]
        adc     \$0, %rdx
        mov     $X[0], $DST
```

```
___
for(my $i=1;$i<8;$i++) {
$code.=<<___;
        mov     %rdx, $TMP

        mov     (+8*$i)($SRC2), %rax
        mul     $OP                     # rdx:rax = %OP * [$i]
        add     %rax, $X[$i]
        adc     \$0, %rdx
        add     $TMP, $X[$i]
        adc     \$0, %rdx
___
}
$code.=<<___;
        mov     %rdx, $X[0]
___
}

#
# Swizzle Macros
#

# macro to copy data from flat space to swizzled table
#MACRO swizzle  pDst, pSrc, tmp1, tmp2
# pDst and pSrc are modified
sub swizzle
{
 my ($pDst, $pSrc, $cnt, $d0)=@_;
$code.=<<___;
        mov     \$8, $cnt
loop_$m:
        mov     ($pSrc), $d0
        mov     $d0#w, ($pDst)
        shr     \$16, $d0
        mov     $d0#w, (+64*1)($pDst)
        shr     \$16, $d0
        mov     $d0#w, (+64*2)($pDst)
        shr     \$16, $d0
        mov     $d0#w, (+64*3)($pDst)
        lea     8($pSrc), $pSrc
        lea     64*4($pDst), $pDst
        dec     $cnt
        jnz     loop_$m
___

 $m++;
}

# macro to copy data from swizzled table to  flat space
#MACRO unswizzle        pDst, pSrc, tmp*3
sub unswizzle
{
 my ($pDst, $pSrc, $cnt, $d0, $d1)=@_;
$code.=<<___;
        mov     \$4, $cnt
loop_$m:
        movzxw (+64*3+256*0)($pSrc), $d0
        movzxw (+64*3+256*1)($pSrc), $d1
        shl     \$16, $d0
        shl     \$16, $d1
        mov     (+64*2+256*0)($pSrc), $d0#w
        mov     (+64*2+256*1)($pSrc), $d1#w
        shl     \$16, $d0
        shl     \$16, $d1
        mov     (+64*1+256*0)($pSrc), $d0#w
        mov     (+64*1+256*1)($pSrc), $d1#w
```



```
        shl     \$16, $d0
        shl     \$16, $d1
        mov     (+64*0+256*0)($pSrc), $d0#w
        mov     (+64*0+256*1)($pSrc), $d1#w
        mov     $d0, (+8*0)($pDst)
        mov     $d1, (+8*1)($pDst)
        lea     256*2($pSrc), $pSrc
        lea     8*2($pDst), $pDst
        sub     \$1, $cnt
        jnz     loop_$m
___

 $m++;
}

#
# Data Structures
#

# Reduce Data
#
#
# Offset  Value
# 0C0     Carries
# 0B8     X2[10]
# 0B0     X2[9]
# 0A8     X2[8]
# 0A0     X2[7]
# 098     X2[6]
# 090     X2[5]
# 088     X2[4]
# 080     X2[3]
# 078     X2[2]
# 070     X2[1]
# 068     X2[0]
# 060     X1[12]  P[10]
# 058     X1[11]  P[9]  Z[8]
# 050     X1[10]  P[8]  Z[7]
# 048     X1[9]   P[7]  Z[6]
# 040     X1[8]   P[6]  Z[5]
# 038     X1[7]   P[5]  Z[4]
# 030     X1[6]   P[4]  Z[3]
# 028     X1[5]   P[3]  Z[2]
# 020     X1[4]   P[2]  Z[1]
# 018     X1[3]   P[1]  Z[0]
# 010     X1[2]   P[0]  Y[2]
# 008     X1[1]   Q[1]  Y[1]
# 000     X1[0]   Q[0]  Y[0]

my $X1_offset           =  0;                   # 13 qwords
my $X2_offset           =  $X1_offset + 13*8;                   # 11 qwords
my $Carries_offset      =  $X2_offset + 11*8;                   # 1 qword
my $Q_offset            =  0;                   # 2 qwords
my $P_offset            =  $Q_offset + 2*8;                     # 11 qwords
my $Y_offset            =  0;                   # 3 qwords
my $Z_offset            =  $Y_offset + 3*8;                     # 9 qwords

my $Red_Data_Size       =  $Carries_offset + 1*8;                       # (25 qw

#
# Stack Frame
#
#
# offset        value
# ...           <old stack contents>
# ...
```

```
# 280           Garray

# 278           tmp16[15]
# ...           ...
# 200           tmp16[0]

# 1F8           tmp[7]
# ...           ...
# 1C0           tmp[0]

# 1B8           GT[7]
# ...           ...
# 180           GT[0]

# 178           Reduce Data
# ...           ...
# 0B8           Reduce Data
# 0B0           reserved
# 0A8           reserved
# 0A0           reserved
# 098           reserved
# 090           reserved
# 088           reduce result addr
# 080           exp[8]

# ...
# 048           exp[1]
# 040           exp[0]

# 038           reserved
# 030           loop_idx
# 028           pg
# 020           i
# 018           pData   ; arg 4
# 010           pG      ; arg 2
# 008           pResult ; arg 1
# 000           rsp     ; stack pointer before subtract

my $rsp_offset          =  0;
my $pResult_offset      =  8*1 + $rsp_offset;
my $pG_offset           =  8*1 + $pResult_offset;
my $pData_offset        =  8*1 + $pG_offset;
my $i_offset            =  8*1 + $pData_offset;
my $pg_offset           =  8*1 + $i_offset;
my $loop_idx_offset     =  8*1 + $pg_offset;
my $reserved1_offset    =  8*1 + $loop_idx_offset;
my $exp_offset          =  8*1 + $reserved1_offset;
my $red_result_addr_offset=  8*9 + $exp_offset;
my $reserved2_offset    =  8*1 + $red_result_addr_offset;
my $Reduce_Data_offset  =  8*5 + $reserved2_offset;
my $GT_offset           =  $Red_Data_Size + $Reduce_Data_offset;
my $tmp_offset          =  8*8 + $GT_offset;
my $tmp16_offset        =  8*8 + $tmp_offset;
my $garray_offset       =  8*16 + $tmp16_offset;
my $mem_size            =  8*8*32 + $garray_offset;

#
# Offsets within Reduce Data
#
#
#       struct MODF_2FOLD_MONT_512_C1_DATA {
#       UINT64 t[8][8];
#       UINT64 m[8];
#       UINT64 m1[8]; /* 2^768 % m */
#       UINT64 m2[8]; /* 2^640 % m */
#       UINT64 k1[2]; /* (- 1/m) % 2^128 */
```



```
#       };

my $T                   =  0;
my $M                   =  512;                 # = 8 * 8 * 8
my $M1                  =  576;                 # = 8 * 8 * 9 /* += 8 * 8 */
my $M2                  =  640;                 # = 8 * 8 * 10 /* += 8 * 8 */
my $K1                  =  704;                 # = 8 * 8 * 11 /* += 8 * 8 */

#
#   FUNCTIONS
#

{{{
#
# MULADD_128x512 : Function to multiply 128-bits (2 qwords) by 512-bits (8 qword
#                       and add 512-bits (8 qwords)
#                       to get 640 bits (10 qwords)
# Input: 128-bit mul source: [rdi+8*1], rbp
#        512-bit mul source: [rsi+8*n]
#        512-bit add source: r15, r14, ..., r9, r8
# Output: r9, r8, r15, r14, r13, r12, r11, r10, [rcx+8*1], [rcx+8*0]
# Clobbers all regs except: rcx, rsi, rdi
$code.=<<___;
.type   MULADD_128x512,\@abi-omnipotent
.align  16
MULADD_128x512:
___
        &MULSTEP_512([map("%r$_",(8..15))], "(+8*0)(%rcx)", "%rsi", "%rbp", "%rb
$code.=<<___;
         mov    (+8*1)(%rdi), %rbp
___
        &MULSTEP_512([map("%r$_",(9..15,8))], "(+8*1)(%rcx)", "%rsi", "%rbp", "%
$code.=<<___;
         ret
.size   MULADD_128x512,.-MULADD_128x512
___
}}}

{{{
#MULADD_256x512 MACRO   pDst, pA, pB, OP, TMP, X7, X6, X5, X4, X3, X2, X1, X0
#
# Inputs: pDst: Destination  (768 bits, 12 qwords)
#         pA:   Multiplicand (1024 bits, 16 qwords)
#         pB:   Multiplicand (512 bits, 8 qwords)
# Dst = Ah * B + Al
# where Ah is (in qwords) A[15:12] (256 bits) and Al is A[7:0] (512 bits)
# Results in X3 X2 X1 X0 X7 X6 X5 X4 Dst[3:0]
# Uses registers: arguments, RAX, RDX
sub MULADD_256x512
{
 my ($pDst, $pA, $pB, $OP, $TMP, $X)=@_;
$code.=<<___;
        mov     (+8*12)($pA), $OP
___
        &MULSTEP_512_ADD($X, "(+8*0)($pDst)", $pB, $pA, $OP, $TMP);
        push(@$X,shift(@$X));

$code.=<<___;
         mov    (+8*13)($pA), $OP
___
        &MULSTEP_512($X, "(+8*1)($pDst)", $pB, $OP, $TMP);
        push(@$X,shift(@$X));

$code.=<<___;
         mov    (+8*14)($pA), $OP
___
```

```
392         &MULSTEP_512($X, "(+8*2)($pDst)", $pB, $OP, $TMP);
393         push(@$X,shift(@$X));

395 $code.=<<___;
396          mov    (+8*15)($pA), $OP
397 ___
398         &MULSTEP_512($X, "(+8*3)($pDst)", $pB, $OP, $TMP);
399         push(@$X,shift(@$X));
400 }

402 #
403 # mont_reduce(UINT64 *x,  /* 1024 bits, 16 qwords */
404 #             UINT64 *m,  /*  512 bits,  8 qwords */
405 #             MODF_2FOLD_MONT_512_C1_DATA *data,
406 #            UINT64 *r)  /*  512 bits,  8 qwords */
407 # Input:  x (number to be reduced): tmp16 (Implicit)
408 #         m (modulus):              [pM]  (Implicit)
409 #         data (reduce data):       [pData] (Implicit)
410 # Output: r (result):                Address in [red_res_addr]
411 #         result also in: r9, r8, r15, r14, r13, r12, r11, r10

413 my @X=map("%r$_",(8..15));

415 $code.=<<___;
416 .type   mont_reduce,\@abi-omnipotent
417 .align  16
418 mont_reduce:
419 ___

421 my $STACK_DEPTH         =  8;
422         #
423         # X1 = Xh * M1 + Xl
424 $code.=<<___;
425          lea    (+$Reduce_Data_offset+$X1_offset+$STACK_DEPTH)(%rsp), %rdi
426          mov    (+$pData_offset+$STACK_DEPTH)(%rsp), %rsi
427          add    \$$M1, %rsi
428          lea    (+$tmp16_offset+$STACK_DEPTH)(%rsp), %rcx

430 ___

432         &MULADD_256x512("%rdi", "%rcx", "%rsi", "%rbp", "%rbx", \@X);   # rotate
433         # results in r11, r10, r9, r8, r15, r14, r13, r12, X1[3:0]

435 $code.=<<___;
436          xor    %rax, %rax
437         # X1 += xl
438          add    (+8*8)(%rcx), $X[4]
439          adc    (+8*9)(%rcx), $X[5]
440          adc    (+8*10)(%rcx), $X[6]
441          adc    (+8*11)(%rcx), $X[7]
442          adc    \$0, %rax
443         # X1 is now rax, r11-r8, r15-r12, tmp16[3:0]

445         #
446         # check for carry ;; carry stored in rax
447          mov    $X[4], (+8*8)(%rdi)                     # rdi points to X1
448          mov    $X[5], (+8*9)(%rdi)
449          mov    $X[6], %rbp
450          mov    $X[7], (+8*11)(%rdi)

452          mov    %rax, (+$Reduce_Data_offset+$Carries_offset+$STACK_DEPTH)(%rsp)

454          mov    (+8*0)(%rdi), $X[4]
455          mov    (+8*1)(%rdi), $X[5]
456          mov    (+8*2)(%rdi), $X[6]
457          mov    (+8*3)(%rdi), $X[7]
```



```
459         # X1 is now stored in: X1[11], rbp, X1[9:8], r15-r8
460         # rdi -> X1
461         # rsi -> M1

463         #
464         # X2 = Xh * M2 + Xl
465         # do first part (X2 = Xh * M2)
466          add    \$8*10, %rdi                    # rdi -> pXh ; 128 bits, 2 qword
467                                 #        Xh is actually { [rdi+8*1], rbp }
468          add    \$‘$M2-$M1‘, %rsi                       # rsi -> M2
469          lea    (+$Reduce_Data_offset+$X2_offset+$STACK_DEPTH)(%rsp), %rcx
470 ___
471         unshift(@X,pop(@X));    unshift(@X,pop(@X));
472 $code.=<<___;

474          call   MULADD_128x512                  # args in rcx, rdi / rbp, rsi, r
475         # result in r9, r8, r15, r14, r13, r12, r11, r10, X2[1:0]
476          mov    (+$Reduce_Data_offset+$Carries_offset+$STACK_DEPTH)(%rsp), %rax

478         # X2 += Xl
479          add    (+8*8-8*10)(%rdi), $X[6]                # (-8*10) is to adjust r
480          adc    (+8*9-8*10)(%rdi), $X[7]
481          mov    $X[6], (+8*8)(%rcx)
482          mov    $X[7], (+8*9)(%rcx)

484          adc    %rax, %rax
485          mov    %rax, (+$Reduce_Data_offset+$Carries_offset+$STACK_DEPTH)(%rsp)

487          lea    (+$Reduce_Data_offset+$Q_offset+$STACK_DEPTH)(%rsp), %rdi
488          add    \$‘$K1-$M2‘, %rsi                       # rsi -> pK1 ; 128 bits,

490         # MUL_128x128t128       rdi, rcx, rsi   ; Q = X2 * K1 (bottom half)
491         # B1:B0 = rsi[1:0] = K1[1:0]
492         # A1:A0 = rcx[1:0] = X2[1:0]
493         # Result = rdi[1],rbp = Q[1],rbp
494          mov    (%rsi), %r8                     # B0
495          mov    (+8*1)(%rsi), %rbx                      # B1

497          mov    (%rcx), %rax                    # A0
498          mul    %r8                     # B0
499          mov    %rax, %rbp
500          mov    %rdx, %r9

502          mov    (+8*1)(%rcx), %rax                      # A1
503          mul    %r8                     # B0
504          add    %rax, %r9

506          mov    (%rcx), %rax                    # A0
507          mul    %rbx                    # B1
508          add    %rax, %r9

510          mov    %r9, (+8*1)(%rdi)
511         # end MUL_128x128t128

513          sub    \$‘$K1-$M‘, %rsi

515          mov    (%rcx), $X[6]
516          mov    (+8*1)(%rcx), $X[7]                     # r9:r8 = X2[1:0]

518          call   MULADD_128x512                  # args in rcx, rdi / rbp, rsi, r
519         # result in r9, r8, r15, r14, r13, r12, r11, r10, X2[1:0]

521         # load first half of m to rdx, rdi, rbx, rax
522         # moved this here for efficiency
523          mov    (+8*0)(%rsi), %rax
```

```
524          mov    (+8*1)(%rsi), %rbx
525          mov    (+8*2)(%rsi), %rdi
526          mov    (+8*3)(%rsi), %rdx

528         # continue with reduction
529          mov    (+$Reduce_Data_offset+$Carries_offset+$STACK_DEPTH)(%rsp), %rbp

531          add    (+8*8)(%rcx), $X[6]
532          adc    (+8*9)(%rcx), $X[7]

534         #accumulate the final carry to rbp
535          adc    %rbp, %rbp

537         # Add in overflow corrections: R = (X2>>128) += T[overflow]
538         # R = {r9, r8, r15, r14, ..., r10}
539          shl    \$3, %rbp
540          mov    (+$pData_offset+$STACK_DEPTH)(%rsp), %rcx
541          add    %rcx, %rbp                     # pT ; 512 bits, 8 qwords, sprea

543         # rsi will be used to generate a mask after the addition
544          xor    %rsi, %rsi

546          add    (+8*8*0)(%rbp), $X[0]
547          adc    (+8*8*1)(%rbp), $X[1]
548          adc    (+8*8*2)(%rbp), $X[2]
549          adc    (+8*8*3)(%rbp), $X[3]
550          adc    (+8*8*4)(%rbp), $X[4]
551          adc    (+8*8*5)(%rbp), $X[5]
552          adc    (+8*8*6)(%rbp), $X[6]
553          adc    (+8*8*7)(%rbp), $X[7]

555         # if there is a carry:  rsi = 0xFFFFFFFFFFFFFFFF
556         # if carry is clear:    rsi = 0x0000000000000000
557          sbb    \$0, %rsi

559         # if carry is clear, subtract 0. Otherwise, subtract 256 bits of m
560          and    %rsi, %rax
561          and    %rsi, %rbx
562          and    %rsi, %rdi
563          and    %rsi, %rdx

565          mov    \$1, %rbp
566          sub    %rax, $X[0]
567          sbb    %rbx, $X[1]
568          sbb    %rdi, $X[2]
569          sbb    %rdx, $X[3]

571         # if there is a borrow:         rbp = 0
572         # if there is no borrow:        rbp = 1
573         # this is used to save the borrows in between the first half and the 2nd
574          sbb    \$0, %rbp

576         #load second half of m to rdx, rdi, rbx, rax

578          add    \$\$M, %rcx
579          mov    (+8*4)(%rcx), %rax
580          mov    (+8*5)(%rcx), %rbx
581          mov    (+8*6)(%rcx), %rdi
582          mov    (+8*7)(%rcx), %rdx

584         # use the rsi mask as before
585         # if carry is clear, subtract 0. Otherwise, subtract 256 bits of m
586          and    %rsi, %rax
587          and    %rsi, %rbx
588          and    %rsi, %rdi
589          and    %rsi, %rdx
```



```
591         # if rbp = 0, there was a borrow before, it is moved to the carry flag
592         # if rbp = 1, there was not a borrow before, carry flag is cleared
593          sub    \$1, %rbp

595          sbb    %rax, $X[4]
596          sbb    %rbx, $X[5]
597          sbb    %rdi, $X[6]
598          sbb    %rdx, $X[7]

600         # write R back to memory

602          mov    (+$red_result_addr_offset+$STACK_DEPTH)(%rsp), %rsi
603          mov    $X[0], (+8*0)(%rsi)
604          mov    $X[1], (+8*1)(%rsi)
605          mov    $X[2], (+8*2)(%rsi)
606          mov    $X[3], (+8*3)(%rsi)
607          mov    $X[4], (+8*4)(%rsi)
608          mov    $X[5], (+8*5)(%rsi)
609          mov    $X[6], (+8*6)(%rsi)
610          mov    $X[7], (+8*7)(%rsi)

612          ret
613 .size   mont_reduce,.-mont_reduce
614 ___
615 }}}

617 {{{
618 #MUL_512x512    MACRO   pDst, pA, pB, x7, x6, x5, x4, x3, x2, x1, x0, tmp*2
619 #
620 # Inputs: pDst: Destination  (1024 bits, 16 qwords)
621 #         pA:   Multiplicand (512 bits, 8 qwords)
622 #         pB:   Multiplicand (512 bits, 8 qwords)
623 # Uses registers rax, rdx, args
624 #   B operand in [pB] and also in x7...x0
625 sub MUL_512x512
626 {
627  my ($pDst, $pA, $pB, $x, $OP, $TMP, $pDst_o)=@_;
628  my ($pDst,  $pDst_o) = ($pDst =~ m/([^+]*)\+?(.*)?/);
629  my @X=@$x;     # make a copy

631 $code.=<<___;
632          mov    (+8*0)($pA), $OP

634          mov    $X[0], %rax
635          mul    $OP                    # rdx:rax = %OP * [0]
636          mov    %rax, (+$pDst_o+8*0)($pDst)
637          mov    %rdx, $X[0]
638 ___
639 for(my $i=1;$i<8;$i++) {
640 $code.=<<___;
641          mov    $X[$i], %rax
642          mul    $OP                    # rdx:rax = %OP * [$i]
643          add    %rax, $X[$i-1]
644          adc    \$0, %rdx
645          mov    %rdx, $X[$i]
646 ___
647 }

649 for(my $i=1;$i<8;$i++) {
650 $code.=<<___;
651          mov    (+8*$i)($pA), $OP
652 ___

654         &MULSTEP_512(\@X, "(+$pDst_o+8*$i)($pDst)", $pB, $OP, $TMP);
655         push(@X,shift(@X));
```

```
656 }

658 $code.=<<___;
659          mov    $X[0], (+$pDst_o+8*8)($pDst)
660          mov    $X[1], (+$pDst_o+8*9)($pDst)
661          mov    $X[2], (+$pDst_o+8*10)($pDst)
662          mov    $X[3], (+$pDst_o+8*11)($pDst)
663          mov    $X[4], (+$pDst_o+8*12)($pDst)
664          mov    $X[5], (+$pDst_o+8*13)($pDst)
665          mov    $X[6], (+$pDst_o+8*14)($pDst)
666          mov    $X[7], (+$pDst_o+8*15)($pDst)
667 ___
668 }

670 #
671 # mont_mul_a3b : subroutine to compute (Src1 * Src2) % M (all 512-bits)
672 # Input:  src1: Address of source 1: rdi
673 #         src2: Address of source 2: rsi
674 # Output: dst:  Address of destination: [red_res_addr]
675 #    src2 and result also in: r9, r8, r15, r14, r13, r12, r11, r10
676 # Temp:   Clobbers [tmp16], all registers
677 $code.=<<___;
678 .type   mont_mul_a3b,\@abi-omnipotent
679 .align  16
680 mont_mul_a3b:
681         #
682         # multiply tmp = src1 * src2
683         # For multiply: dst = rcx, src1 = rdi, src2 = rsi
684         # stack depth is extra 8 from call
685 ___
686         &MUL_512x512("%rsp+$tmp16_offset+8", "%rdi", "%rsi", [map("%r$_",(10..15
687 $code.=<<___;
688         #
689         # Dst = tmp % m
690         # Call reduce(tmp, m, data, dst)

692         # tail recursion optimization: jmp to mont_reduce and return from there
693          jmp    mont_reduce
694         # call  mont_reduce
695         # ret
696 .size   mont_mul_a3b,.-mont_mul_a3b
697 ___
698 }}}

700 {{{
701 #SQR_512 MACRO pDest, pA, x7, x6, x5, x4, x3, x2, x1, x0, tmp*4
702 #
703 # Input in memory [pA] and also in x7...x0
704 # Uses all argument registers plus rax and rdx
705 #
706 # This version computes all of the off-diagonal terms into memory,
707 # and then it adds in the diagonal terms

709 sub SQR_512
710 {
711  my ($pDst, $pA, $x, $A, $tmp, $x7, $x6, $pDst_o)=@_;
712  my ($pDst,  $pDst_o) = ($pDst =~ m/([^+]*)\+?(.*)?/);
713  my @X=@$x;     # make a copy
714 $code.=<<___;
715         # ------------------
716         # first pass 01...07
717         # ------------------
718          mov    $X[0], $A

720          mov    $X[1],%rax
721          mul    $A
```



```
722          mov    %rax, (+$pDst_o+8*1)($pDst)
723 ___
724 for(my $i=2;$i<8;$i++) {
725 $code.=<<___;
726          mov    %rdx, $X[$i-2]
727          mov    $X[$i],%rax
728          mul    $A
729          add    %rax, $X[$i-2]
730          adc    \$0, %rdx
731 ___
732 }
733 $code.=<<___;
734          mov    %rdx, $x7

736          mov    $X[0], (+$pDst_o+8*2)($pDst)

738         # ------------------
739         # second pass 12...17
740         # ------------------

742          mov    (+8*1)($pA), $A

744          mov    (+8*2)($pA),%rax
745          mul    $A
746          add    %rax, $X[1]
747          adc    \$0, %rdx
748          mov    $X[1], (+$pDst_o+8*3)($pDst)

750          mov    %rdx, $X[0]
751          mov    (+8*3)($pA),%rax
752          mul    $A
753          add    %rax, $X[2]
754          adc    \$0, %rdx
755          add    $X[0], $X[2]
756          adc    \$0, %rdx
757          mov    $X[2], (+$pDst_o+8*4)($pDst)

759          mov    %rdx, $X[0]
760          mov    (+8*4)($pA),%rax
761          mul    $A
762          add    %rax, $X[3]
763          adc    \$0, %rdx
764          add    $X[0], $X[3]
765          adc    \$0, %rdx

767          mov    %rdx, $X[0]
768          mov    (+8*5)($pA),%rax
769          mul    $A
770          add    %rax, $X[4]
771          adc    \$0, %rdx
772          add    $X[0], $X[4]
773          adc    \$0, %rdx

775          mov    %rdx, $X[0]
776          mov    $X[6],%rax
777          mul    $A
778          add    %rax, $X[5]
779          adc    \$0, %rdx
780          add    $X[0], $X[5]
781          adc    \$0, %rdx

783          mov    %rdx, $X[0]
784          mov    $X[7],%rax
785          mul    $A
786          add    %rax, $x7
787          adc    \$0, %rdx
```

```
788          add    $X[0], $x7
789          adc    \$0, %rdx

791          mov    %rdx, $X[1]

793         # ------------------
794         # third pass 23...27
795         # ------------------
796          mov    (+8*2)($pA), $A

798          mov    (+8*3)($pA),%rax
799          mul    $A
800          add    %rax, $X[3]
801          adc    \$0, %rdx
802          mov    $X[3], (+$pDst_o+8*5)($pDst)

804          mov    %rdx, $X[0]
805          mov    (+8*4)($pA),%rax
806          mul    $A
807          add    %rax, $X[4]
808          adc    \$0, %rdx
809          add    $X[0], $X[4]
810          adc    \$0, %rdx
811          mov    $X[4], (+$pDst_o+8*6)($pDst)

813          mov    %rdx, $X[0]
814          mov    (+8*5)($pA),%rax
815          mul    $A
816          add    %rax, $X[5]
817          adc    \$0, %rdx
818          add    $X[0], $X[5]
819          adc    \$0, %rdx

821          mov    %rdx, $X[0]
822          mov    $X[6],%rax
823          mul    $A
824          add    %rax, $x7
825          adc    \$0, %rdx
826          add    $X[0], $x7
827          adc    \$0, %rdx

829          mov    %rdx, $X[0]
830          mov    $X[7],%rax
831          mul    $A
832          add    %rax, $X[1]
833          adc    \$0, %rdx
834          add    $X[0], $X[1]
835          adc    \$0, %rdx

837          mov    %rdx, $X[2]

839         # ------------------
840         # fourth pass 34...37
841         # ------------------

843          mov    (+8*3)($pA), $A

845          mov    (+8*4)($pA),%rax
846          mul    $A
847          add    %rax, $X[5]
848          adc    \$0, %rdx
849          mov    $X[5], (+$pDst_o+8*7)($pDst)

851          mov    %rdx, $X[0]
852          mov    (+8*5)($pA),%rax
853          mul    $A
```



```
854          add    %rax, $x7
855          adc    \$0, %rdx
856          add    $X[0], $x7
857          adc    \$0, %rdx
858          mov    $x7, (+$pDst_o+8*8)($pDst)

860          mov    %rdx, $X[0]
861          mov    $X[6],%rax
862          mul    $A
863          add    %rax, $X[1]
864          adc    \$0, %rdx
865          add    $X[0], $X[1]
866          adc    \$0, %rdx

868          mov    %rdx, $X[0]
869          mov    $X[7],%rax
870          mul    $A
871          add    %rax, $X[2]
872          adc    \$0, %rdx
873          add    $X[0], $X[2]
874          adc    \$0, %rdx

876          mov    %rdx, $X[5]

878         # ------------------
879         # fifth pass 45...47
880         # ------------------
881          mov    (+8*4)($pA), $A

883          mov    (+8*5)($pA),%rax
884          mul    $A
885          add    %rax, $X[1]
886          adc    \$0, %rdx
887          mov    $X[1], (+$pDst_o+8*9)($pDst)

889          mov    %rdx, $X[0]
890          mov    $X[6],%rax
891          mul    $A
892          add    %rax, $X[2]
893          adc    \$0, %rdx
894          add    $X[0], $X[2]
895          adc    \$0, %rdx
896          mov    $X[2], (+$pDst_o+8*10)($pDst)

898          mov    %rdx, $X[0]
899          mov    $X[7],%rax
900          mul    $A
901          add    %rax, $X[5]
902          adc    \$0, %rdx
903          add    $X[0], $X[5]
904          adc    \$0, %rdx

906          mov    %rdx, $X[1]

908         # ------------------
909         # sixth pass 56...57
910         # ------------------
911          mov    (+8*5)($pA), $A

913          mov    $X[6],%rax
914          mul    $A
915          add    %rax, $X[5]
916          adc    \$0, %rdx
917          mov    $X[5], (+$pDst_o+8*11)($pDst)

919          mov    %rdx, $X[0]
```

```
920          mov    $X[7],%rax
921          mul    $A
922          add    %rax, $X[1]
923          adc    \$0, %rdx
924          add    $X[0], $X[1]
925          adc    \$0, %rdx
926          mov    $X[1], (+$pDst_o+8*12)($pDst)

928          mov    %rdx, $X[2]

930         # ------------------
931         # seventh pass 67
932         # ------------------
933          mov    $X[6], $A

935          mov    $X[7],%rax
936          mul    $A
937          add    %rax, $X[2]
938          adc    \$0, %rdx
939          mov    $X[2], (+$pDst_o+8*13)($pDst)

941          mov    %rdx, (+$pDst_o+8*14)($pDst)

943         # start finalize (add   in squares, and double off-terms)
944          mov    (+$pDst_o+8*1)($pDst), $X[0]
945          mov    (+$pDst_o+8*2)($pDst), $X[1]
946          mov    (+$pDst_o+8*3)($pDst), $X[2]
947          mov    (+$pDst_o+8*4)($pDst), $X[3]
948          mov    (+$pDst_o+8*5)($pDst), $X[4]
949          mov    (+$pDst_o+8*6)($pDst), $X[5]

951          mov    (+8*3)($pA), %rax
952          mul    %rax
953          mov    %rax, $x6
954          mov    %rdx, $X[6]

956          add    $X[0], $X[0]
957          adc    $X[1], $X[1]
958          adc    $X[2], $X[2]
959          adc    $X[3], $X[3]
960          adc    $X[4], $X[4]
961          adc    $X[5], $X[5]
962          adc    \$0, $X[6]

964          mov    (+8*0)($pA), %rax
965          mul    %rax
966          mov    %rax, (+$pDst_o+8*0)($pDst)
967          mov    %rdx, $A

969          mov    (+8*1)($pA), %rax
970          mul    %rax

972          add    $A, $X[0]
973          adc    %rax, $X[1]
974          adc    \$0, %rdx

976          mov    %rdx, $A
977          mov    $X[0], (+$pDst_o+8*1)($pDst)
978          mov    $X[1], (+$pDst_o+8*2)($pDst)

980          mov    (+8*2)($pA), %rax
981          mul    %rax

983          add    $A, $X[2]
984          adc    %rax, $X[3]
985          adc    \$0, %rdx
```



```
987          mov    %rdx, $A

989          mov    $X[2], (+$pDst_o+8*3)($pDst)
990          mov    $X[3], (+$pDst_o+8*4)($pDst)

992          xor    $tmp, $tmp
993          add    $A, $X[4]
994          adc    $x6, $X[5]
995          adc    \$0, $tmp

997          mov    $X[4], (+$pDst_o+8*5)($pDst)
998          mov    $X[5], (+$pDst_o+8*6)($pDst)

1000        # %%tmp has 0/1 in column 7
1001        # %%A6 has a full value in column 7

1003         mov    (+$pDst_o+8*7)($pDst), $X[0]
1004         mov    (+$pDst_o+8*8)($pDst), $X[1]
1005         mov    (+$pDst_o+8*9)($pDst), $X[2]
1006         mov    (+$pDst_o+8*10)($pDst), $X[3]
1007         mov    (+$pDst_o+8*11)($pDst), $X[4]
1008         mov    (+$pDst_o+8*12)($pDst), $X[5]
1009         mov    (+$pDst_o+8*13)($pDst), $x6
1010         mov    (+$pDst_o+8*14)($pDst), $x7

1012         mov    $X[7], %rax
1013         mul    %rax
1014         mov    %rax, $X[7]
1015         mov    %rdx, $A

1017         add    $X[0], $X[0]
1018         adc    $X[1], $X[1]
1019         adc    $X[2], $X[2]
1020         adc    $X[3], $X[3]
1021         adc    $X[4], $X[4]
1022         adc    $X[5], $X[5]
1023         adc    $x6, $x6
1024         adc    $x7, $x7
1025         adc    \$0, $A

1027         add    $tmp, $X[0]

1029         mov    (+8*4)($pA), %rax
1030         mul    %rax

1032         add    $X[6], $X[0]
1033         adc    %rax, $X[1]
1034         adc    \$0, %rdx

1036         mov    %rdx, $tmp

1038         mov    $X[0], (+$pDst_o+8*7)($pDst)
1039         mov    $X[1], (+$pDst_o+8*8)($pDst)

1041         mov    (+8*5)($pA), %rax
1042         mul    %rax

1044         add    $tmp, $X[2]
1045         adc    %rax, $X[3]
1046         adc    \$0, %rdx

1048         mov    %rdx, $tmp

1050         mov    $X[2], (+$pDst_o+8*9)($pDst)
1051         mov    $X[3], (+$pDst_o+8*10)($pDst)
```

```
1053          mov    (+8*6)($pA), %rax
1054          mul    %rax

1056          add    $tmp, $X[4]
1057          adc    %rax, $X[5]
1058          adc    \$0, %rdx

1060          mov    $X[4], (+$pDst_o+8*11)($pDst)
1061          mov    $X[5], (+$pDst_o+8*12)($pDst)

1063          add    %rdx, $x6
1064          adc    $X[7], $x7
1065          adc    \$0, $A

1067          mov    $x6, (+$pDst_o+8*13)($pDst)
1068          mov    $x7, (+$pDst_o+8*14)($pDst)
1069          mov    $A, (+$pDst_o+8*15)($pDst)
1070 ___
1071 }

1073 #
1074 # sqr_reduce: subroutine to compute Result = reduce(Result * Result)
1075 #
1076 # input and result also in: r9, r8, r15, r14, r13, r12, r11, r10
1077 #
1078 $code.=<<___;
1079 .type   sqr_reduce,\@abi-omnipotent
1080 .align  16
1081 sqr_reduce:
1082          mov    (+$pResult_offset+8)(%rsp), %rcx
1083 ___
1084         &SQR_512("%rsp+$tmp16_offset+8", "%rcx", [map("%r$_",(10..15,8..9))], "%
1085 $code.=<<___;
1086         # tail recursion optimization: jmp to mont_reduce and return from there
1087          jmp    mont_reduce
1088         # call  mont_reduce
1089         # ret
1090 .size   sqr_reduce,.-sqr_reduce
1091 ___
1092 }}}

1094 #
1095 # MAIN FUNCTION
1096 #

1098 #mod_exp_512(UINT64 *result, /* 512 bits, 8 qwords */
1099 #           UINT64 *g,   /* 512 bits, 8 qwords */
1100 #           UINT64 *exp, /* 512 bits, 8 qwords */
1101 #           struct mod_ctx_512 *data)

1103 # window size = 5
1104 # table size = 2^5 = 32
1105 #table_entries  equ     32
1106 #table_size     equ     table_entries * 8
1107 $code.=<<___;
1108 .globl  mod_exp_512
1109 .type   mod_exp_512,\@function,4
1110 mod_exp_512:
1111          push   %rbp
1112          push   %rbx
1113          push   %r12
1114          push   %r13
1115          push   %r14
1116          push   %r15
```



```
1118         # adjust stack down and then align it with cache boundary
1119          mov    %rsp, %r8
1120          sub    \$$mem_size, %rsp
1121          and    \$-64, %rsp

1123         # store previous stack pointer and arguments
1124          mov    %r8, (+$rsp_offset)(%rsp)
1125          mov    %rdi, (+$pResult_offset)(%rsp)
1126          mov    %rsi, (+$pG_offset)(%rsp)
1127          mov    %rcx, (+$pData_offset)(%rsp)
1128 .Lbody:
1129         # transform g into montgomery space
1130         # GT = reduce(g * C2) = reduce(g * (2^256))
1131         # reduce expects to have the input in [tmp16]
1132          pxor   %xmm4, %xmm4
1133          movdqu (+16*0)(%rsi), %xmm0
1134          movdqu (+16*1)(%rsi), %xmm1
1135          movdqu (+16*2)(%rsi), %xmm2
1136          movdqu (+16*3)(%rsi), %xmm3
1137          movdqa %xmm4, (+$tmp16_offset+16*0)(%rsp)
1138          movdqa %xmm4, (+$tmp16_offset+16*1)(%rsp)
1139          movdqa %xmm4, (+$tmp16_offset+16*6)(%rsp)
1140          movdqa %xmm4, (+$tmp16_offset+16*7)(%rsp)
1141          movdqa %xmm0, (+$tmp16_offset+16*2)(%rsp)
1142          movdqa %xmm1, (+$tmp16_offset+16*3)(%rsp)
1143          movdqa %xmm2, (+$tmp16_offset+16*4)(%rsp)
1144          movdqa %xmm3, (+$tmp16_offset+16*5)(%rsp)

1146         # load pExp before rdx gets blown away
1147          movdqu (+16*0)(%rdx), %xmm0
1148          movdqu (+16*1)(%rdx), %xmm1
1149          movdqu (+16*2)(%rdx), %xmm2
1150          movdqu (+16*3)(%rdx), %xmm3

1152          lea    (+$GT_offset)(%rsp), %rbx
1153          mov    %rbx, (+$red_result_addr_offset)(%rsp)
1154          call   mont_reduce

1156         # Initialize tmp = C
1157          lea    (+$tmp_offset)(%rsp), %rcx
1158          xor    %rax, %rax
1159          mov    %rax, (+8*0)(%rcx)
1160          mov    %rax, (+8*1)(%rcx)
1161          mov    %rax, (+8*3)(%rcx)
1162          mov    %rax, (+8*4)(%rcx)
1163          mov    %rax, (+8*5)(%rcx)
1164          mov    %rax, (+8*6)(%rcx)
1165          mov    %rax, (+8*7)(%rcx)
1166          mov    %rax, (+$exp_offset+8*8)(%rsp)
1167          movq   \$1, (+8*2)(%rcx)

1169          lea    (+$garray_offset)(%rsp), %rbp
1170          mov    %rcx, %rsi                      # pTmp
1171          mov    %rbp, %rdi                      # Garray[][0]
1172 ___

1174         &swizzle("%rdi", "%rcx", "%rax", "%rbx");

1176         # for (rax = 31; rax != 0; rax--) {
1177         #     tmp = reduce(tmp * G)
1178         #     swizzle(pg, tmp);
1179         #     pg += 2; }
1180 $code.=<<___;
1181          mov    \$31, %rax
1182          mov    %rax, (+$i_offset)(%rsp)
1183          mov    %rbp, (+$pg_offset)(%rsp)
```

```
1184         # rsi -> pTmp
1185          mov    %rsi, (+$red_result_addr_offset)(%rsp)
1186          mov    (+8*0)(%rsi), %r10
1187          mov    (+8*1)(%rsi), %r11
1188          mov    (+8*2)(%rsi), %r12
1189          mov    (+8*3)(%rsi), %r13
1190          mov    (+8*4)(%rsi), %r14
1191          mov    (+8*5)(%rsi), %r15
1192          mov    (+8*6)(%rsi), %r8
1193          mov    (+8*7)(%rsi), %r9
1194 init_loop:
1195          lea    (+$GT_offset)(%rsp), %rdi
1196          call   mont_mul_a3b
1197          lea    (+$tmp_offset)(%rsp), %rsi
1198          mov    (+$pg_offset)(%rsp), %rbp
1199          add    \$2, %rbp
1200          mov    %rbp, (+$pg_offset)(%rsp)
1201          mov    %rsi, %rcx                      # rcx = rsi = addr of tmp
1202 ___

1204         &swizzle("%rbp", "%rcx", "%rax", "%rbx");
1205 $code.=<<___;
1206          mov    (+$i_offset)(%rsp), %rax
1207          sub    \$1, %rax
1208          mov    %rax, (+$i_offset)(%rsp)
1209          jne    init_loop

1211         #
1212         # Copy exponent onto stack
1213          movdqa %xmm0, (+$exp_offset+16*0)(%rsp)
1214          movdqa %xmm1, (+$exp_offset+16*1)(%rsp)
1215          movdqa %xmm2, (+$exp_offset+16*2)(%rsp)
1216          movdqa %xmm3, (+$exp_offset+16*3)(%rsp)


1219         #
1220         # Do exponentiation
1221         # Initialize result to G[exp{511:507}]
1222          mov    (+$exp_offset+62)(%rsp), %eax
1223          mov    %rax, %rdx
1224          shr    \$11, %rax
1225          and    \$0x07FF, %edx
1226          mov    %edx, (+$exp_offset+62)(%rsp)
1227          lea    (+$garray_offset)(%rsp,%rax,2), %rsi
1228          mov    (+$pResult_offset)(%rsp), %rdx
1229 ___

1231         &unswizzle("%rdx", "%rsi", "%rbp", "%rbx", "%rax");

1233         #
1234         # Loop variables
1235         # rcx = [loop_idx] = index: 510-5 to 0 by 5
1236 $code.=<<___;
1237          movq   \$505, (+$loop_idx_offset)(%rsp)

1239          mov    (+$pResult_offset)(%rsp), %rcx
1240          mov    %rcx, (+$red_result_addr_offset)(%rsp)
1241          mov    (+8*0)(%rcx), %r10
1242          mov    (+8*1)(%rcx), %r11
1243          mov    (+8*2)(%rcx), %r12
1244          mov    (+8*3)(%rcx), %r13
1245          mov    (+8*4)(%rcx), %r14
1246          mov    (+8*5)(%rcx), %r15
1247          mov    (+8*6)(%rcx), %r8
1248          mov    (+8*7)(%rcx), %r9
1249          jmp    sqr_2
```



```
1251 main_loop_a3b:
1252          call   sqr_reduce
1253          call   sqr_reduce
1254          call   sqr_reduce
1255 sqr_2:
1256          call   sqr_reduce
1257          call   sqr_reduce

1259         #
1260         # Do multiply, first look up proper value in Garray
1261          mov    (+$loop_idx_offset)(%rsp), %rcx                # bit index
1262          mov    %rcx, %rax
1263          shr    \$4, %rax                       # rax is word pointer
1264          mov    (+$exp_offset)(%rsp,%rax,2), %edx
1265          and    \$15, %rcx
1266          shrq   %cl, %rdx
1267          and    \$0x1F, %rdx

1269          lea    (+$garray_offset)(%rsp,%rdx,2), %rsi
1270          lea    (+$tmp_offset)(%rsp), %rdx
1271          mov    %rdx, %rdi
1272 ___

1274         &unswizzle("%rdx", "%rsi", "%rbp", "%rbx", "%rax");
1275         # rdi = tmp = pG

1277         #
1278         # Call mod_mul_a1(pDst,  pSrc1, pSrc2, pM, pData)
1279         #                 result result pG     M   Data
1280 $code.=<<___;
1281          mov    (+$pResult_offset)(%rsp), %rsi
1282          call   mont_mul_a3b

1284         #
1285         # finish loop
1286          mov    (+$loop_idx_offset)(%rsp), %rcx
1287          sub    \$5, %rcx
1288          mov    %rcx, (+$loop_idx_offset)(%rsp)
1289          jge    main_loop_a3b

1291         #

1293 end_main_loop_a3b:
1294         # transform result out of Montgomery space
1295         # result = reduce(result)
1296          mov    (+$pResult_offset)(%rsp), %rdx
1297          pxor   %xmm4, %xmm4
1298          movdqu (+16*0)(%rdx), %xmm0
1299          movdqu (+16*1)(%rdx), %xmm1
1300          movdqu (+16*2)(%rdx), %xmm2
1301          movdqu (+16*3)(%rdx), %xmm3
1302          movdqa %xmm4, (+$tmp16_offset+16*4)(%rsp)
1303          movdqa %xmm4, (+$tmp16_offset+16*5)(%rsp)
1304          movdqa %xmm4, (+$tmp16_offset+16*6)(%rsp)
1305          movdqa %xmm4, (+$tmp16_offset+16*7)(%rsp)
1306          movdqa %xmm0, (+$tmp16_offset+16*0)(%rsp)
1307          movdqa %xmm1, (+$tmp16_offset+16*1)(%rsp)
1308          movdqa %xmm2, (+$tmp16_offset+16*2)(%rsp)
1309          movdqa %xmm3, (+$tmp16_offset+16*3)(%rsp)
1310          call   mont_reduce

1312         # If result > m, subract m
1313         # load result into r15:r8
1314          mov    (+$pResult_offset)(%rsp), %rax
1315          mov    (+8*0)(%rax), %r8
```

```
1316          mov    (+8*1)(%rax), %r9
1317          mov    (+8*2)(%rax), %r10
1318          mov    (+8*3)(%rax), %r11
1319          mov    (+8*4)(%rax), %r12
1320          mov    (+8*5)(%rax), %r13
1321          mov    (+8*6)(%rax), %r14
1322          mov    (+8*7)(%rax), %r15

1324         # subtract m
1325          mov    (+$pData_offset)(%rsp), %rbx
1326          add    \$$M, %rbx

1328          sub    (+8*0)(%rbx), %r8
1329          sbb    (+8*1)(%rbx), %r9
1330          sbb    (+8*2)(%rbx), %r10
1331          sbb    (+8*3)(%rbx), %r11
1332          sbb    (+8*4)(%rbx), %r12
1333          sbb    (+8*5)(%rbx), %r13
1334          sbb    (+8*6)(%rbx), %r14
1335          sbb    (+8*7)(%rbx), %r15

1337         # if Carry is clear, replace result with difference
1338          mov    (+8*0)(%rax), %rsi
1339          mov    (+8*1)(%rax), %rdi
1340          mov    (+8*2)(%rax), %rcx
1341          mov    (+8*3)(%rax), %rdx
1342          cmovnc %r8, %rsi
1343          cmovnc %r9, %rdi
1344          cmovnc %r10, %rcx
1345          cmovnc %r11, %rdx
1346          mov    %rsi, (+8*0)(%rax)
1347          mov    %rdi, (+8*1)(%rax)
1348          mov    %rcx, (+8*2)(%rax)
1349          mov    %rdx, (+8*3)(%rax)

1351          mov    (+8*4)(%rax), %rsi
1352          mov    (+8*5)(%rax), %rdi
1353          mov    (+8*6)(%rax), %rcx
1354          mov    (+8*7)(%rax), %rdx
1355          cmovnc %r12, %rsi
1356          cmovnc %r13, %rdi
1357          cmovnc %r14, %rcx
1358          cmovnc %r15, %rdx
1359          mov    %rsi, (+8*4)(%rax)
1360          mov    %rdi, (+8*5)(%rax)
1361          mov    %rcx, (+8*6)(%rax)
1362          mov    %rdx, (+8*7)(%rax)

1364          mov    (+$rsp_offset)(%rsp), %rsi
1365          mov    0(%rsi),%r15
1366          mov    8(%rsi),%r14
1367          mov    16(%rsi),%r13
1368          mov    24(%rsi),%r12
1369          mov    32(%rsi),%rbx
1370          mov    40(%rsi),%rbp
1371          lea    48(%rsi),%rsp
1372 .Lepilogue:
1373          ret
1374 .size mod_exp_512, . - mod_exp_512
1375 ___

1377 if ($win64) {
1378 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
1379 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
1380 my $rec="%rcx";
1381 my $frame="%rdx";
```



```
1382 my $context="%r8";
1383 my $disp="%r9";

1385 $code.=<<___;
1386 .extern __imp_RtlVirtualUnwind
1387 .type   mod_exp_512_se_handler,\@abi-omnipotent
1388 .align  16
1389 mod_exp_512_se_handler:
1390          push    %rsi
1391          push    %rdi
1392          push    %rbx
1393          push    %rbp
1394          push    %r12
1395          push    %r13
1396          push    %r14
1397          push    %r15
1398          pushfq
1399          sub     \$64,%rsp

1401          mov     120($context),%rax      # pull context->Rax
1402          mov     248($context),%rbx      # pull context->Rip

1404          lea     .Lbody(%rip),%r10
1405          cmp     %r10,%rbx               # context->Rip<prologue label
1406          jb      .Lin_prologue

1408          mov     152($context),%rax      # pull context->Rsp

1410          lea     .Lepilogue(%rip),%r10
1411          cmp     %r10,%rbx               # context->Rip>=epilogue label
1412          jae     .Lin_prologue

1414          mov     $rsp_offset(%rax),%rax  # pull saved Rsp

1416          mov     32(%rax),%rbx
1417          mov     40(%rax),%rbp
1418          mov     24(%rax),%r12
1419          mov     16(%rax),%r13
1420          mov     8(%rax),%r14
1421          mov     0(%rax),%r15
1422          lea     48(%rax),%rax
1423          mov     %rbx,144($context)      # restore context->Rbx
1424          mov     %rbp,160($context)      # restore context->Rbp
1425          mov     %r12,216($context)      # restore context->R12
1426          mov     %r13,224($context)      # restore context->R13
1427          mov     %r14,232($context)      # restore context->R14
1428          mov     %r15,240($context)      # restore context->R15

1430 .Lin_prologue:
1431          mov     8(%rax),%rdi
1432          mov     16(%rax),%rsi
1433          mov     %rax,152($context)      # restore context->Rsp
1434          mov     %rsi,168($context)      # restore context->Rsi
1435          mov     %rdi,176($context)      # restore context->Rdi

1437          mov     40($disp),%rdi          # disp->ContextRecord
1438          mov     $context,%rsi           # context
1439          mov     \$154,%ecx              # sizeof(CONTEXT)
1440          .long   0xa548f3fc              # cld; rep movsq

1442          mov     $disp,%rsi
1443          xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
1444          mov     8(%rsi),%rdx            # arg2, disp->ImageBase
1445          mov     0(%rsi),%r8             # arg3, disp->ControlPc
1446          mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
1447          mov     40(%rsi),%r10           # disp->ContextRecord
```

```
1448         lea     56(%rsi),%r11           # &disp->HandlerData
1449         lea     24(%rsi),%r12           # &disp->EstablisherFrame
1450         mov     %r10,32(%rsp)           # arg5
1451         mov     %r11,40(%rsp)           # arg6
1452         mov     %r12,48(%rsp)           # arg7
1453         mov     %rcx,56(%rsp)           # arg8, (NULL)
1454         call    *__imp_RtlVirtualUnwind(%rip)

1456         mov     \$1,%eax                # ExceptionContinueSearch
1457         add     \$64,%rsp
1458         popfq
1459         pop     %r15
1460         pop     %r14
1461         pop     %r13
1462         pop     %r12
1463         pop     %rbp
1464         pop     %rbx
1465         pop     %rdi
1466         pop     %rsi
1467         ret
1468 .size   mod_exp_512_se_handler,.-mod_exp_512_se_handler

1470 .section        .pdata
1471 .align  4
1472         .rva    .LSEH_begin_mod_exp_512
1473         .rva    .LSEH_end_mod_exp_512
1474         .rva    .LSEH_info_mod_exp_512

1476 .section        .xdata
1477 .align  8
1478 .LSEH_info_mod_exp_512:
1479         .byte   9,0,0,0
1480         .rva    mod_exp_512_se_handler
1481 ___
1482 }

1484 sub reg_part {
1485 my ($reg,$conv)=@_;
1486     if ($reg =~ /%r[0-9]+/)     { $reg .= $conv; }
1487     elsif ($conv eq "b")        { $reg =~ s/%[er]([^x]+)x?/%$1l/;       }
1488     elsif ($conv eq "w")        { $reg =~ s/%[er](.+)/%$1/;             }
1489     elsif ($conv eq "d")        { $reg =~ s/%[er](.+)/%e$1/;            }
1490     return $reg;
1491 }

1493 $code =~ s/(%[a-z0-9]+)#([bwd])/reg_part($1,$2)/gem;
1494 $code =~ s/\`([^\`]*)\`/eval $1/gem;
1495 $code =~ s/(\(\+[^)]+\))/eval $1/gem;
1496 print $code;
1497 close STDOUT;
1498 #endif /* ! codereview */
```

```
**********************************************************
   11985 Wed Aug 13 19:53:08 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/rc4-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# ====================================================================
# [Re]written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================

# At some point it became apparent that the original SSLeay RC4
# assembler implementation performs suboptimally on latest IA-32
# microarchitectures. After re-tuning performance has changed as
# following:
#
# Pentium       -10%
# Pentium III   +12%
# AMD           +50%(*)
# P4            +250%(**)
#
# (*)   This number is actually a trade-off:-) It's possible to
#       achieve +72%, but at the cost of -48% off PIII performance.
#       In other words code performing further 13% faster on AMD
#       would perform almost 2 times slower on Intel PIII...
#       For reference! This code delivers ~80% of rc4-amd64.pl
#       performance on the same Opteron machine.
# (**)  This number requires compressed key schedule set up by
#       RC4_set_key [see commentary below for further details].
#
#                                       <appro@fy.chalmers.se>

# May 2011
#
# Optimize for Core2 and Westmere [and incidentally Opteron]. Current
# performance in cycles per processed byte (less is better) and
# improvement relative to previous version of this module is:
#
# Pentium       10.2                    # original numbers
# Pentium III   7.8(*)
# Intel P4      7.5
#
# Opteron       6.1/+20%                # new MMX numbers
# Core2         5.3/+67%(**)
# Westmere      5.1/+94%(**)
# Sandy Bridge  5.0/+8%
# Atom          12.6/+6%
#
# (*)   PIII can actually deliver 6.6 cycles per byte with MMX code,
#       but this specific code performs poorly on Core2. And vice
#       versa, below MMX/SSE code delivering 5.8/7.1 on Core2 performs
#       poorly on PIII, at 8.0/14.5:-( As PIII is not a "hot" CPU
#       [anymore], I chose to discard PIII-specific code path and opt
#       for original IALU-only code, which is why MMX/SSE code path
#       is guarded by SSE2 bit (see below), not MMX/SSE.
# (**)  Performance vs. block size on Core2 and Westmere had a maximum
#       at ... 64 bytes block size. And it was quite a maximum, 40-60%
#       in comparison to largest 8KB block size. Above improvement
#       coefficients are for the largest block size.

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";
```



```
&asm_init($ARGV[0],"rc4-586.pl");

$xx="eax";
$yy="ebx";
$tx="ecx";
$ty="edx";
$inp="esi";
$out="ebp";
$dat="edi";

sub RC4_loop {
  my $i=shift;
  my $func = ($i==0)?*mov:*or;

        &add    (&LB($yy),&LB($tx));
        &mov    ($ty,&DWP(0,$dat,$yy,4));
        &mov    (&DWP(0,$dat,$yy,4),$tx);
        &mov    (&DWP(0,$dat,$xx,4),$ty);
        &add    ($ty,$tx);
        &inc    (&LB($xx));
        &and    ($ty,0xff);
        &ror    ($out,8)        if ($i!=0);
        if ($i<3) {
          &mov  ($tx,&DWP(0,$dat,$xx,4));
        } else {
          &mov  ($tx,&wparam(3));       # reload [re-biased] out
        }
        &$func  ($out,&DWP(0,$dat,$ty,4));
}

if ($alt=0) {
  # >20% faster on Atom and Sandy Bridge[!], 8% faster on Opteron,
  # but ~40% slower on Core2 and Westmere... Attempt to add movz
  # brings down Opteron by 25%, Atom and Sandy Bridge by 15%, yet
  # on Core2 with movz it's almost 20% slower than below alternative
  # code... Yes, it's a total mess...
  my @XX=($xx,$out);
  $RC4_loop_mmx = sub {         # SSE actually...
    my $i=shift;
    my $j=$i<=0?0:$i>>1;
    my $mm=$i<=0?"mm0":"mm".($i&1);

        &add    (&LB($yy),&LB($tx));
        &lea    (@XX[1],&DWP(1,@XX[0]));
        &pxor   ("mm2","mm0")                           if ($i==0);
        &psllq  ("mm1",8)                               if ($i==0);
        &and    (@XX[1],0xff);
        &pxor   ("mm0","mm0")                           if ($i<=0);
        &mov    ($ty,&DWP(0,$dat,$yy,4));
        &mov    (&DWP(0,$dat,$yy,4),$tx);
        &pxor   ("mm1","mm2")                           if ($i==0);
        &mov    (&DWP(0,$dat,$XX[0],4),$ty);
        &add    (&LB($ty),&LB($tx));
        &movd   (@XX[0],"mm7")                          if ($i==0);
        &mov    ($tx,&DWP(0,$dat,@XX[1],4));
        &pxor   ("mm1","mm1")                           if ($i==1);
        &movq   ("mm2",&QWP(0,$inp))                    if ($i==1);
        &movq   (&QWP(-8,(@XX[0],$inp)),"mm1")          if ($i==0);
        &pinsrw ($mm,&DWP(0,$dat,$ty,4),$j);

        push    (@XX,shift(@XX))                        if ($i>=0);
  }
} else {
  # Using pinsrw here improves performane on Intel CPUs by 2-3%, but
  # brings down AMD by 7%...
```

```
  $RC4_loop_mmx = sub {
    my $i=shift;

        &add    (&LB($yy),&LB($tx));
        &psllq  ("mm1",8*(($i-1)&7))                    if (abs($i)!=1);
        &mov    ($ty,&DWP(0,$dat,$yy,4));
        &mov    (&DWP(0,$dat,$yy,4),$tx);
        &mov    (&DWP(0,$dat,$xx,4),$ty);
        &inc    ($xx);
        &add    ($ty,$tx);
        &movz   ($xx,&LB($xx));                         # (*)
        &movz   ($ty,&LB($ty));                         # (*)
        &pxor   ("mm2",$i==1?"mm0":"mm1")               if ($i>=0);
        &movq   ("mm0",&QWP(0,$inp))                    if ($i<=0);
        &movq   (&QWP(-8,($out,$inp)),"mm2")            if ($i==0);
        &mov    ($tx,&DWP(0,$dat,$xx,4));
        &movd   ($i>0?"mm1":"mm2",&DWP(0,$dat,$ty,4));

        # (*)   This is the key to Core2 and Westmere performance.
        #       Whithout movz out-of-order execution logic confuses
        #       itself and fails to reorder loads and stores. Problem
        #       appears to be fixed in Sandy Bridge...
  }
}

&external_label("OPENSSL_ia32cap_P");

# void RC4(RC4_KEY *key,size_t len,const unsigned char *inp,unsigned char *out);
&function_begin("RC4");
        &mov    ($dat,&wparam(0));      # load key schedule pointer
        &mov    ($ty, &wparam(1));      # load len
        &mov    ($inp,&wparam(2));      # load inp
        &mov    ($out,&wparam(3));      # load out

        &xor    ($xx,$xx);              # avoid partial register stalls
        &xor    ($yy,$yy);

        &cmp    ($ty,0);                # safety net
        &je     (&label("abort"));

        &mov    (&LB($xx),&BP(0,$dat)); # load key->x
        &mov    (&LB($yy),&BP(4,$dat)); # load key->y
        &add    ($dat,8);

        &lea    ($tx,&DWP(0,$inp,$ty));
        &sub    ($out,$inp);            # re-bias out
        &mov    (&wparam(1),$tx);       # save input+len

        &inc    (&LB($xx));

        # detect compressed key schedule...
        &cmp    (&DWP(256,$dat),-1);
        &je     (&label("RC4_CHAR"));

        &mov    ($tx,&DWP(0,$dat,$xx,4));

        &and    ($ty,-4);               # how many 4-byte chunks?
        &jz     (&label("loop1"));

        &test   ($ty,-8);
        &mov    (&wparam(3),$out);      # $out as accumulator in these loops
        &jz     (&label("go4loop4"));

        &picmeup($out,"OPENSSL_ia32cap_P");
        &bt     (&DWP(0,$out),26);      # check SSE2 bit [could have been MMX]
        &jnc    (&label("go4loop4"));
```



```
        &mov    ($out,&wparam(3))       if (!$alt);
        &movd   ("mm7",&wparam(3))      if ($alt);
        &and    ($ty,-8);
        &lea    ($ty,&DWP(-8,$inp,$ty));
        &mov    (&DWP(-4,$dat),$ty);    # save input+(len/8)*8-8

        &$RC4_loop_mmx(-1);
        &jmp(&label("loop_mmx_enter"));

        &set_label("loop_mmx",16);
                &$RC4_loop_mmx(0);
        &set_label("loop_mmx_enter");
                for     ($i=1;$i<8;$i++) { &$RC4_loop_mmx($i); }
                &mov    ($ty,$yy);
                &xor    ($yy,$yy);              # this is second key to Core2
                &mov    (&LB($yy),&LB($ty));    # and Westmere performance...
                &cmp    ($inp,&DWP(-4,$dat));
                &lea    ($inp,&DWP(8,$inp));
        &jb     (&label("loop_mmx"));

    if ($alt) {
        &movd   ($out,"mm7");
        &pxor   ("mm2","mm0");
        &psllq  ("mm1",8);
        &pxor   ("mm1","mm2");
        &movq   (&QWP(-8,$out,$inp),"mm1");
    } else {
        &psllq  ("mm1",56);
        &pxor   ("mm2","mm1");
        &movq   (&QWP(-8,$out,$inp),"mm2");
    }
        &emms   ();

        &cmp    ($inp,&wparam(1));      # compare to input+len
        &je     (&label("done"));
        &jmp    (&label("loop1"));

&set_label("go4loop4",16);
        &lea    ($ty,&DWP(-4,$inp,$ty));
        &mov    (&wparam(2),$ty);       # save input+(len/4)*4-4

        &set_label("loop4");
                for ($i=0;$i<4;$i++) { RC4_loop($i); }
                &ror    ($out,8);
                &xor    ($out,&DWP(0,$inp));
                &cmp    ($inp,&wparam(2));      # compare to input+(len/4)*4-4
                &mov    (&DWP(0,$tx,$inp),$out);# $tx holds re-biased out here
                &lea    ($inp,&DWP(4,$inp));
                &mov    ($tx,&DWP(0,$dat,$xx,4));
        &jb     (&label("loop4"));

        &cmp    ($inp,&wparam(1));      # compare to input+len
        &je     (&label("done"));
        &mov    ($out,&wparam(3));      # restore $out

        &set_label("loop1",16);
                &add    (&LB($yy),&LB($tx));
                &mov    ($ty,&DWP(0,$dat,$yy,4));
                &mov    (&DWP(0,$dat,$yy,4),$tx);
                &mov    (&DWP(0,$dat,$xx,4),$ty);
                &add    ($ty,$tx);
                &inc    (&LB($xx));
                &and    ($ty,0xff);
                &mov    ($ty,&DWP(0,$dat,$ty,4));
                &xor    (&LB($ty),&BP(0,$inp));
```

```
260                 &lea    ($inp,&DWP(1,$inp));
261                 &mov    ($tx,&DWP(0,$dat,$xx,4));
262                 &cmp    ($inp,&wparam(1));      # compare to input+len
263                 &mov    (&BP(-1,$out,$inp),&LB($ty));
264         &jb     (&label("loop1"));

266         &jmp    (&label("done"));

268 # this is essentially Intel P4 specific codepath...
269 &set_label("RC4_CHAR",16);
270         &movz   ($tx,&BP(0,$dat,$xx));
271         # strangely enough unrolled loop performs over 20% slower...
272         &set_label("cloop1");
273                 &add    (&LB($yy),&LB($tx));
274                 &movz   ($ty,&BP(0,$dat,$yy));
275                 &mov    (&BP(0,$dat,$yy),&LB($tx));
276                 &mov    (&BP(0,$dat,$xx),&LB($ty));
277                 &add    (&LB($ty),&LB($tx));
278                 &movz   ($ty,&BP(0,$dat,$ty));
279                 &add    (&LB($xx),1);
280                 &xor    (&LB($ty),&BP(0,$inp));
281                 &lea    ($inp,&DWP(1,$inp));
282                 &movz   ($tx,&BP(0,$dat,$xx));
283                 &cmp    ($inp,&wparam(1));
284                 &mov    (&BP(-1,$out,$inp),&LB($ty));
285         &jb     (&label("cloop1"));

287 &set_label("done");
288         &dec    (&LB($xx));
289         &mov    (&DWP(-4,$dat),$yy);            # save key->y
290         &mov    (&BP(-8,$dat),&LB($xx));        # save key->x
291 &set_label("abort");
292 &function_end("RC4");

294 ########################################################################

296 $inp="esi";
297 $out="edi";
298 $idi="ebp";
299 $ido="ecx";
300 $idx="edx";

302 # void RC4_set_key(RC4_KEY *key,int len,const unsigned char *data);
303 &function_begin("private_RC4_set_key");
304         &mov    ($out,&wparam(0));              # load key
305         &mov    ($idi,&wparam(1));              # load len
306         &mov    ($inp,&wparam(2));              # load data
307         &picmeup($idx,"OPENSSL_ia32cap_P");

309         &lea    ($out,&DWP(2*4,$out));          # &key->data
310         &lea    ($inp,&DWP(0,$inp,$idi));       # $inp to point at the end
311         &neg    ($idi);
312         &xor    ("eax","eax");
313         &mov    (&DWP(-4,$out),$idi);           # borrow key->y

315         &bt     (&DWP(0,$idx),20);              # check for bit#20
316         &jc     (&label("c1stloop"));

318 &set_label("w1stloop",16);
319         &mov    (&DWP(0,$out,"eax",4),"eax");   # key->data[i]=i;
320         &add    (&LB("eax"),1);                 # i++;
321         &jnc    (&label("w1stloop"));

323         &xor    ($ido,$ido);
324         &xor    ($idx,$idx);
```



```
326 &set_label("w2ndloop",16);
327         &mov    ("eax",&DWP(0,$out,$ido,4));
328         &add    (&LB($idx),&BP(0,$inp,$idi));
329         &add    (&LB($idx),&LB("eax"));
330         &add    ($idi,1);
331         &mov    ("ebx",&DWP(0,$out,$idx,4));
332         &jnz    (&label("wnowrap"));
333           &mov  ($idi,&DWP(-4,$out));
334         &set_label("wnowrap");
335         &mov    (&DWP(0,$out,$idx,4),"eax");
336         &mov    (&DWP(0,$out,$ido,4),"ebx");
337         &add    (&LB($ido),1);
338         &jnc    (&label("w2ndloop"));
339 &jmp    (&label("exit"));

341 # Unlike all other x86 [and x86_64] implementations, Intel P4 core
342 # [including EM64T] was found to perform poorly with above "32-bit" key
343 # schedule, a.k.a. RC4_INT. Performance improvement for IA-32 hand-coded
344 # assembler turned out to be 3.5x if re-coded for compressed 8-bit one,
345 # a.k.a. RC4_CHAR! It's however inappropriate to just switch to 8-bit
346 # schedule for x86[_64], because non-P4 implementations suffer from
347 # significant performance losses then, e.g. PIII exhibits >2x
348 # deterioration, and so does Opteron. In order to assure optimal
349 # all-round performance, we detect P4 at run-time and set up compressed
350 # key schedule, which is recognized by RC4 procedure.

352 &set_label("c1stloop",16);
353         &mov    (&BP(0,$out,"eax"),&LB("eax")); # key->data[i]=i;
354         &add    (&LB("eax"),1);                 # i++;
355         &jnc    (&label("c1stloop"));

357         &xor    ($ido,$ido);
358         &xor    ($idx,$idx);
359         &xor    ("ebx","ebx");

361 &set_label("c2ndloop",16);
362         &mov    (&LB("eax"),&BP(0,$out,$ido));
363         &add    (&LB($idx),&BP(0,$inp,$idi));
364         &add    (&LB($idx),&LB("eax"));
365         &add    ($idi,1);
366         &mov    (&LB("ebx"),&BP(0,$out,$idx));
367         &jnz    (&label("cnowrap"));
368           &mov  ($idi,&DWP(-4,$out));
369         &set_label("cnowrap");
370         &mov    (&BP(0,$out,$idx),&LB("eax"));
371         &mov    (&BP(0,$out,$ido),&LB("ebx"));
372         &add    (&LB($ido),1);
373         &jnc    (&label("c2ndloop"));

375         &mov    (&DWP(256,$out),-1);            # mark schedule as compressed

377 &set_label("exit");
378         &xor    ("eax","eax");
379         &mov    (&DWP(-8,$out),"eax");          # key->x=0;
380         &mov    (&DWP(-4,$out),"eax");          # key->y=0;
381 &function_end("private_RC4_set_key");

383 # const char *RC4_options(void);
384 &function_begin_B("RC4_options");
385         &call   (&label("pic_point"));
386 &set_label("pic_point");
387         &blindpop("eax");
388         &lea    ("eax",&DWP(&label("opts")."-".&label("pic_point"),"eax"));
389         &picmeup("edx","OPENSSL_ia32cap_P");
390         &mov    ("edx",&DWP(0,"edx"));
391         &bt     ("edx",20);
```

```
392         &jc     (&label("1xchar"));
393         &bt     ("edx",26);
394         &jnc    (&label("ret"));
395         &add    ("eax",25);
396         &ret    ();
397 &set_label("1xchar");
398         &add    ("eax",12);
399 &set_label("ret");
400         &ret    ();
401 &set_label("opts",64);
402 &asciz  ("rc4(4x,int)");
403 &asciz  ("rc4(1x,char)");
404 &asciz  ("rc4(8x,mmx)");
405 &asciz  ("RC4 for x86, CRYPTOGAMS by <appro\@openssl.org>");
406 &align  (64);
407 &function_end_B("RC4_options");

409 &asm_finish();
410 #endif /* ! codereview */
```

```
**********************************************************
   15968 Wed Aug 13 19:53:08 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/rc4-md5-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================

# June 2011
#
# This is RC4+MD5 "stitch" implementation. The idea, as spelled in
# http://download.intel.com/design/intarch/papers/323686.pdf, is that
# since both algorithms exhibit instruction-level parallelism, ILP,
# below theoretical maximum, interleaving them would allow to utilize
# processor resources better and achieve better performance. RC4
# instruction sequence is virtually identical to rc4-x86_64.pl, which
# is heavily based on submission by Maxim Perminov, Maxim Locktyukhin
# and Jim Guilford of Intel. MD5 is fresh implementation aiming to
# minimize register usage, which was used as "main thread" with RC4
# weaved into it, one RC4 round per one MD5 round. In addition to the
# stiched subroutine the script can generate standalone replacement
# md5_block_asm_data_order and RC4. Below are performance numbers in
# cycles per processed byte, less is better, for these the standalone
# subroutines, sum of them, and stitched one:
#
#               RC4     MD5     RC4+MD5 stitch  gain
# Opteron       6.5(*)  5.4     11.9    7.0     +70%(*)
# Core2         6.5     5.8     12.3    7.7     +60%
# Westmere      4.3     5.2     9.5     7.0     +36%
# Sandy Bridge  4.2     5.5     9.7     6.8     +43%
# Atom          9.3     6.5     15.8    11.1    +42%
#
# (*)   rc4-x86_64.pl delivers 5.3 on Opteron, so real improvement
#       is +53%...

my ($rc4,$md5)=(1,1);   # what to generate?
my $D="#" if (!$md5);   # if set to "#", MD5 is stitched into RC4(),
                        # but its result is discarded. Idea here is
                        # to be able to use 'openssl speed rc4' for
                        # benchmarking the stitched subroutine...

my $flavour = shift;
my $output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

my $win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; my $dir=$1; my $xlate;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

my ($dat,$in0,$out,$ctx,$inp,$len, $func,$nargs);

if ($rc4 && !$md5) {
  ($dat,$len,$in0,$out) = ("%rdi","%rsi","%rdx","%rcx");
  $func="RC4";                          $nargs=4;
```



```
} elsif ($md5 && !$rc4) {
  ($ctx,$inp,$len) = ("%rdi","%rsi","%rdx");
  $func="md5_block_asm_data_order";     $nargs=3;
} else {
  ($dat,$in0,$out,$ctx,$inp,$len) = ("%rdi","%rsi","%rdx","%rcx","%r8","%r9");
  $func="rc4_md5_enc";                  $nargs=6;
  # void rc4_md5_enc(
  #             RC4_KEY *key,           #
  #             const void *in0,        # RC4 input
  #             void *out,              # RC4 output
  #             MD5_CTX *ctx,           #
  #             const void *inp,        # MD5 input
  #             size_t len);            # number of 64-byte blocks
}

my @K=( 0xd76aa478,0xe8c7b756,0x242070db,0xc1bdceee,
        0xf57c0faf,0x4787c62a,0xa8304613,0xfd469501,
        0x698098d8,0x8b44f7af,0xffff5bb1,0x895cd7be,
        0x6b901122,0xfd987193,0xa679438e,0x49b40821,

        0xf61e2562,0xc040b340,0x265e5a51,0xe9b6c7aa,
        0xd62f105d,0x02441453,0xd8a1e681,0xe7d3fbc8,
        0x21e1cde6,0xc33707d6,0xf4d50d87,0x455a14ed,
        0xa9e3e905,0xfcefa3f8,0x676f02d9,0x8d2a4c8a,

        0xfffa3942,0x8771f681,0x6d9d6122,0xfde5380c,
        0xa4beea44,0x4bdecfa9,0xf6bb4b60,0xbebfbc70,
        0x289b7ec6,0xeaa127fa,0xd4ef3085,0x04881d05,
        0xd9d4d039,0xe6db99e5,0x1fa27cf8,0xc4ac5665,

        0xf4292244,0x432aff97,0xab9423a7,0xfc93a039,
        0x655b59c3,0x8f0ccc92,0xffeff47d,0x85845dd1,
        0x6fa87e4f,0xfe2ce6e0,0xa3014314,0x4e0811a1,
        0xf7537e82,0xbd3af235,0x2ad7d2bb,0xeb86d391     );

my @V=("%r8d","%r9d","%r10d","%r11d");  # MD5 registers
my $tmp="%r12d";

my @XX=("%rbp","%rsi");                 # RC4 registers
my @TX=("%rax","%rbx");
my $YY="%rcx";
my $TY="%rdx";

my $MOD=32;                             # 16, 32 or 64

$code.=<<___;
.text
.align 16

.globl  $func
.type   $func,\@function,$nargs
$func:
        cmp     \$0,$len
        je      .Labort
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
        sub     \$40,%rsp
.Lbody:
___
if ($rc4) {
$code.=<<___;
$D#md5# mov     $ctx,%r11               # reassign arguments
```

```
        mov     $len,%r12
        mov     $in0,%r13
        mov     $out,%r14
$D#md5# mov     $inp,%r15
___
    $ctx="%r11" if ($md5);              # reassign arguments
    $len="%r12";
    $in0="%r13";
    $out="%r14";
    $inp="%r15" if ($md5);
    $inp=$in0   if (!$md5);
$code.=<<___;
        xor     $XX[0],$XX[0]
        xor     $YY,$YY

        lea     8($dat),$dat
        mov     -8($dat),$XX[0]#b
        mov     -4($dat),$YY#b

        inc     $XX[0]#b
        sub     $in0,$out
        movl    ($dat,$XX[0],4),$TX[0]#d
___
$code.=<<___ if (!$md5);
        xor     $TX[1],$TX[1]
        test    \$-128,$len
        jz      .Loop1
        sub     $XX[0],$TX[1]
        and     \$`$MOD-1`,$TX[1]
        jz      .Loop${MOD}_is_hot
        sub     $TX[1],$len
.Loop${MOD}_warmup:
        add     $TX[0]#b,$YY#b
        movl    ($dat,$YY,4),$TY#d
        movl    $TX[0]#d,($dat,$YY,4)
        movl    $TY#d,($dat,$XX[0],4)
        add     $TY#b,$TX[0]#b
        inc     $XX[0]#b
        movl    ($dat,$TX[0],4),$TY#d
        movl    ($dat,$XX[0],4),$TX[0]#d
        xorb    ($in0),$TY#b
        movb    $TY#b,($out,$in0)
        lea     1($in0),$in0
        dec     $TX[1]
        jnz     .Loop${MOD}_warmup

        mov     $YY,$TX[1]
        xor     $YY,$YY
        mov     $TX[1]#b,$YY#b

.Loop${MOD}_is_hot:
        mov     $len,32(%rsp)           # save original $len
        shr     \$6,$len                # number of 64-byte blocks
___
  if ($D && !$md5) {                    # stitch in dummy MD5
    $md5=1;
    $ctx="%r11";
    $inp="%r15";
    $code.=<<___;
        mov     %rsp,$ctx
        mov     $in0,$inp
___
  }
}
$code.=<<___;
#rc4#   add     $TX[0]#b,$YY#b
```



```
#rc4#   lea     ($dat,$XX[0],4),$XX[1]
        shl     \$6,$len
        add     $inp,$len               # pointer to the end of input
        mov     $len,16(%rsp)

#md5#   mov     $ctx,24(%rsp)           # save pointer to MD5_CTX
#md5#   mov     0*4($ctx),$V[0]         # load current hash value from MD5_CTX
#md5#   mov     1*4($ctx),$V[1]
#md5#   mov     2*4($ctx),$V[2]
#md5#   mov     3*4($ctx),$V[3]
        jmp     .Loop

.align  16
.Loop:
#md5#   mov     $V[0],0*4(%rsp)         # put aside current hash value
#md5#   mov     $V[1],1*4(%rsp)
#md5#   mov     $V[2],2*4(%rsp)
#md5#   mov     $V[3],$tmp              # forward reference
#md5#   mov     $V[3],3*4(%rsp)
___

sub R0 {
  my ($i,$a,$b,$c,$d)=@_;
  my @rot0=(7,12,17,22);
  my $j=$i%16;
  my $k=$i%$MOD;
  my $xmm="%xmm".($j&1);
    $code.="    movdqu  ($in0),%xmm2\n"         if ($rc4 && $j==15);
    $code.="    add     \$$MOD,$XX[0]#b\n"      if ($rc4 && $j==15 && $k==$MOD-1
    $code.="    pxor    $xmm,%xmm\n"            if ($rc4 && $j<=1);
    $code.=<<___;
#rc4#   movl    ($dat,$YY,4),$TY#d
#md5#   xor     $c,$tmp
#rc4#   movl    $TX[0]#d,($dat,$YY,4)
#md5#   and     $b,$tmp
#md5#   add     4*`$j`($inp),$a
#rc4#   add     $TY#b,$TX[0]#b
#rc4#   movl    `4*(($k+1)%$MOD)`(`$k==$MOD-1?"$dat,$XX[0],4":"$XX[1]"`),$TX[1]#
#md5#   add     \$$K[$i],$a
#md5#   xor     $d,$tmp
#rc4#   movz    $TX[0]#b,$TX[0]#d
#rc4#   movl    $TY#d,4*$k($XX[1])
#md5#   add     $tmp,$a
#rc4#   add     $TX[1]#b,$YY#b
#md5#   rol     \$$rot0[$j%4],$a
#md5#   mov     `$j==15?"$b":"$c"`,$tmp         # forward reference
#rc4#   pinsrw  \$`($j>>1)&7`,($dat,$TX[0],4),$xmm\n
#md5#   add     $b,$a
___
    $code.=<<___ if ($rc4 && $j==15 && $k==$MOD-1);
        mov     $YY,$XX[1]
        xor     $YY,$YY                         # keyword to partial register
        mov     $XX[1]#b,$YY#b
        lea     ($dat,$XX[0],4),$XX[1]
___
    $code.=<<___ if ($rc4 && $j==15);
        psllq   \$8,%xmm1
        pxor    %xmm0,%xmm2
        pxor    %xmm1,%xmm2
___
}
sub R1 {
  my ($i,$a,$b,$c,$d)=@_;
  my @rot1=(5,9,14,20);
  my $j=$i%16;
  my $k=$i%$MOD;
```

```
  my $xmm="%xmm".($j&1);
    $code.="    movdqu  16($in0),%xmm3\n"       if ($rc4 && $j==15);
    $code.="    add     \$$MOD,$XX[0]#b\n"      if ($rc4 && $j==15 && $k==$MOD-1
    $code.="    pxor    $xmm,$xmm\n"            if ($rc4 && $j<=1);
    $code.=<<___;
#rc4#   movl    ($dat,$YY,4),$TY#d
#md5#   xor     $b,$tmp
#rc4#   movl    $TX[0]#d,($dat,$YY,4)
#md5#   and     $d,$tmp
#md5#   add     4*`((1+5*$j)%16)`($inp),$a
#rc4#   add     $TY#b,$TX[0]#b
#rc4#   movl    `4*(($k+1)%$MOD)`(`$k==$MOD-1?"$dat,$XX[0],4":"$XX[1]"`),$TX[1]#
#md5#   add     \$$K[$i],$a
#md5#   xor     $c,$tmp
#rc4#   movz    $TX[0]#b,$TX[0]#d
#rc4#   movl    $TY#d,4*$k($XX[1])
#md5#   add     $tmp,$a
#rc4#   add     $TX[1]#b,$YY#b
#md5#   rol     \$$rot1[$j%4],$a
#md5#   mov     `$j==15?"$c":"$b"`,$tmp         # forward reference
#rc4#   pinsrw  \$`($j>>1)&7`,($dat,$TX[0],4),$xmm\n
#md5#   add     $b,$a
___
    $code.=<<___ if ($rc4 && $j==15 && $k==$MOD-1);
        mov     $YY,$XX[1]
        xor     $YY,$YY                         # keyword to partial register
        mov     $XX[1]#b,$YY#b
        lea     ($dat,$XX[0],4),$XX[1]
___
    $code.=<<___ if ($rc4 && $j==15);
        psllq   \$8,%xmm1
        pxor    %xmm0,%xmm3
        pxor    %xmm1,%xmm3
___
}
sub R2 {
  my ($i,$a,$b,$c,$d)=@_;
  my @rot2=(4,11,16,23);
  my $j=$i%16;
  my $k=$i%$MOD;
  my $xmm="%xmm".($j&1);
    $code.="    movdqu  32($in0),%xmm4\n"       if ($rc4 && $j==15);
    $code.="    add     \$$MOD,$XX[0]#b\n"      if ($rc4 && $j==15 && $k==$MOD-1
    $code.="    pxor    $xmm,$xmm\n"            if ($rc4 && $j<=1);
    $code.=<<___;
#rc4#   movl    ($dat,$YY,4),$TY#d
#md5#   xor     $c,$tmp
#rc4#   movl    $TX[0]#d,($dat,$YY,4)
#md5#   xor     $b,$tmp
#md5#   add     4*`((5+3*$j)%16)`($inp),$a
#rc4#   add     $TY#b,$TX[0]#b
#rc4#   movl    `4*(($k+1)%$MOD)`(`$k==$MOD-1?"$dat,$XX[0],4":"$XX[1]"`),$TX[1]#
#md5#   add     \$$K[$i],$a
#rc4#   movz    $TX[0]#b,$TX[0]#d
#md5#   add     $tmp,$a
#rc4#   movl    $TY#d,4*$k($XX[1])
#rc4#   add     $TX[1]#b,$YY#b
#md5#   rol     \$$rot2[$j%4],$a
#md5#   mov     `$j==15?"\\\$-1":"$c"`,$tmp     # forward reference
#rc4#   pinsrw  \$`($j>>1)&7`,($dat,$TX[0],4),$xmm\n
#md5#   add     $b,$a
___
    $code.=<<___ if ($rc4 && $j==15 && $k==$MOD-1);
        mov     $YY,$XX[1]
        xor     $YY,$YY                         # keyword to partial register
        mov     $XX[1]#b,$YY#b
```



```
        lea     ($dat,$XX[0],4),$XX[1]
___
    $code.=<<___ if ($rc4 && $j==15);
        psllq   \$8,%xmm1
        pxor    %xmm0,%xmm4
        pxor    %xmm1,%xmm4
___
}
sub R3 {
  my ($i,$a,$b,$c,$d)=@_;
  my @rot3=(6,10,15,21);
  my $j=$i%16;
  my $k=$i%$MOD;
  my $xmm="%xmm".($j&1);
    $code.="    movdqu  48($in0),%xmm5\n"       if ($rc4 && $j==15);
    $code.="    add     \$$MOD,$XX[0]#b\n"      if ($rc4 && $j==15 && $k==$MOD-1
    $code.="    pxor    $xmm,$xmm\n"            if ($rc4 && $j<=1);
    $code.=<<___;
#rc4#   movl    ($dat,$YY,4),$TY#d
#md5#   xor     $d,$tmp
#rc4#   movl    $TX[0]#d,($dat,$YY,4)
#md5#   or      $b,$tmp
#md5#   add     4*`((7*$j)%16)`($inp),$a
#rc4#   add     $TY#b,$TX[0]#b
#rc4#   movl    `4*(($k+1)%$MOD)`(`$k==$MOD-1?"$dat,$XX[0],4":"$XX[1]"`),$TX[1]#
#md5#   add     \$$K[$i],$a
#rc4#   movz    $TX[0]#b,$TX[0]#d
#md5#   xor     $c,$tmp
#rc4#   movl    $TY#d,4*$k($XX[1])
#md5#   add     $tmp,$a
#rc4#   add     $TX[1]#b,$YY#b
#md5#   rol     \$$rot3[$j%4],$a
#md5#   mov     \$-1,$tmp                       # forward reference
#rc4#   pinsrw  \$`($j>>1)&7`,($dat,$TX[0],4),$xmm\n
#md5#   add     $b,$a
___
    $code.=<<___ if ($rc4 && $j==15);
        mov     $XX[0],$XX[1]
        xor     $XX[0],$XX[0]                   # keyword to partial register
        mov     $XX[1]#b,$XX[0]#b
        mov     $YY,$XX[1]
        xor     $YY,$YY                         # keyword to partial register
        mov     $XX[1]#b,$YY#b
        lea     ($dat,$XX[0],4),$XX[1]
        psllq   \$8,%xmm1
        pxor    %xmm0,%xmm5
        pxor    %xmm1,%xmm5
___
}

my $i=0;
for(;$i<16;$i++) { R0($i,@V); unshift(@V,pop(@V)); push(@TX,shift(@TX)); }
for(;$i<32;$i++) { R1($i,@V); unshift(@V,pop(@V)); push(@TX,shift(@TX)); }
for(;$i<48;$i++) { R2($i,@V); unshift(@V,pop(@V)); push(@TX,shift(@TX)); }
for(;$i<64;$i++) { R3($i,@V); unshift(@V,pop(@V)); push(@TX,shift(@TX)); }

$code.=<<___;
#md5#   add     0*4(%rsp),$V[0]         # accumulate hash value
#md5#   add     1*4(%rsp),$V[1]
#md5#   add     2*4(%rsp),$V[2]
#md5#   add     3*4(%rsp),$V[3]

#rc4#   movdqu  %xmm2,($out,$in0)       # write RC4 output
#rc4#   movdqu  %xmm3,16($out,$in0)
#rc4#   movdqu  %xmm4,32($out,$in0)
#rc4#   movdqu  %xmm5,48($out,$in0)
```

```
#md5#   lea     64($inp),$inp
#rc4#   lea     64($in0),$in0
        cmp     16(%rsp),$inp           # are we done?
        jb      .Loop

#md5#   mov     24(%rsp),$len           # restore pointer to MD5_CTX
#rc4#   sub     $TX[0]#b,$YY#b          # correct $YY
#md5#   mov     $V[0],0*4($len)         # write MD5_CTX
#md5#   mov     $V[1],1*4($len)
#md5#   mov     $V[2],2*4($len)
#md5#   mov     $V[3],3*4($len)
___
$code.=<<___ if ($rc4 && (!$md5 || $D));
        mov     32(%rsp),$len           # restore original $len
        and     \$63,$len               # remaining bytes
        jnz     .Loop1
        jmp     .Ldone

.align  16
.Loop1:
        add     $TX[0]#b,$YY#b
        movl    ($dat,$YY,4),$TY#d
        movl    $TX[0]#d,($dat,$YY,4)
        movl    $TY#d,($dat,$XX[0],4)
        add     $TY#b,$TX[0]#b
        inc     $XX[0]#b
        movl    ($dat,$TX[0],4),$TY#d
        movl    ($dat,$XX[0],4),$TX[0]#d
        xorb    ($in0),$TY#b
        movb    $TY#b,($out,$in0)
        lea     1($in0),$in0
        dec     $len
        jnz     .Loop1

.Ldone:
___
$code.=<<___;
#rc4#   sub     \$1,$XX[0]#b
#rc4#   movl    $XX[0]#d,-8($dat)
#rc4#   movl    $YY#d,-4($dat)

        mov     40(%rsp),%r15
        mov     48(%rsp),%r14
        mov     56(%rsp),%r13
        mov     64(%rsp),%r12
        mov     72(%rsp),%rbp
        mov     80(%rsp),%rbx
        lea     88(%rsp),%rsp
.Lepilogue:
.Labort:
        ret
.size $func,.-$func
___

if ($rc4 && $D) {       # sole purpose of this section is to provide
                        # option to use the generated module as drop-in
                        # replacement for rc4-x86_64.pl for debugging
                        # and testing purposes...
my ($idx,$ido)=("%r8","%r9");
my ($dat,$len,$inp)=("%rdi","%rsi","%rdx");

$code.=<<___;
.globl  RC4_set_key
.type   RC4_set_key,\@function,3
.align  16
RC4_set_key:
```



```
        lea     8($dat),$dat
        lea     ($inp,$len),$inp
        neg     $len
        mov     $len,%rcx
        xor     %eax,%eax
        xor     $ido,$ido
        xor     %r10,%r10
        xor     %r11,%r11
        jmp     .Lw1stloop

.align  16
.Lw1stloop:
        mov     %eax,($dat,%rax,4)
        add     \$1,%al
        jnc     .Lw1stloop

        xor     $ido,$ido
        xor     $idx,$idx
.align  16
.Lw2ndloop:
        mov     ($dat,$ido,4),%r10d
        add     ($inp,$len,1),$idx#b
        add     %r10b,$idx#b
        add     \$1,$len
        mov     ($dat,$idx,4),%r11d
        cmovz   %rcx,$len
        mov     %r10d,($dat,$idx,4)
        mov     %r11d,($dat,$ido,4)
        add     \$1,$ido#b
        jnc     .Lw2ndloop

        xor     %eax,%eax
        mov     %eax,-8($dat)
        mov     %eax,-4($dat)
        ret
.size   RC4_set_key,.-RC4_set_key

.globl  RC4_options
.type   RC4_options,\@abi-omnipotent
.align  16
RC4_options:
        lea     .Lopts(%rip),%rax
        ret
.align  64
.Lopts:
.asciz  "rc4(64x,int)"
.align  64
.size   RC4_options,.-RC4_options
___
}
# EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
#               CONTEXT *context,DISPATCHER_CONTEXT *disp)
if ($win64) {
my $rec="%rcx";
my $frame="%rdx";
my $context="%r8";
my $disp="%r9";

$code.=<<___;
.extern __imp_RtlVirtualUnwind
.type   se_handler,\@abi-omnipotent
.align  16
se_handler:
        push    %rsi
        push    %rdi
        push    %rbx
```

```
524         push    %rbp
525         push    %r12
526         push    %r13
527         push    %r14
528         push    %r15
529         pushfq
530         sub     \$64,%rsp

532         mov     120($context),%rax      # pull context->Rax
533         mov     248($context),%rbx      # pull context->Rip

535         lea     .Lbody(%rip),%r10
536         cmp     %r10,%rbx               # context->Rip<.Lbody
537         jb      .Lin_prologue

539         mov     152($context),%rax      # pull context->Rsp

541         lea     .Lepilogue(%rip),%r10
542         cmp     %r10,%rbx               # context->Rip>=.Lepilogue
543         jae     .Lin_prologue

545         mov     40(%rax),%r15
546         mov     48(%rax),%r14
547         mov     56(%rax),%r13
548         mov     64(%rax),%r12
549         mov     72(%rax),%rbp
550         mov     80(%rax),%rbx
551         lea     88(%rax),%rax

553         mov     %rbx,144($context)      # restore context->Rbx
554         mov     %rbp,160($context)      # restore context->Rbp
555         mov     %r12,216($context)      # restore context->R12
556         mov     %r13,224($context)      # restore context->R12
557         mov     %r14,232($context)      # restore context->R14
558         mov     %r15,240($context)      # restore context->R15

560 .Lin_prologue:
561         mov     8(%rax),%rdi
562         mov     16(%rax),%rsi
563         mov     %rax,152($context)      # restore context->Rsp
564         mov     %rsi,168($context)      # restore context->Rsi
565         mov     %rdi,176($context)      # restore context->Rdi

567         mov     40($disp),%rdi          # disp->ContextRecord
568         mov     $context,%rsi           # context
569         mov     \$154,%ecx              # sizeof(CONTEXT)
570         .long   0xa548f3fc              # cld; rep movsq

572         mov     $disp,%rsi
573         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
574         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
575         mov     0(%rsi),%r8             # arg3, disp->ControlPc
576         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
577         mov     40(%rsi),%r10           # disp->ContextRecord
578         lea     56(%rsi),%r11           # &disp->HandlerData
579         lea     24(%rsi),%r12           # &disp->EstablisherFrame
580         mov     %r10,32(%rsp)           # arg5
581         mov     %r11,40(%rsp)           # arg6
582         mov     %r12,48(%rsp)           # arg7
583         mov     %rcx,56(%rsp)           # arg8, (NULL)
584         call    *__imp_RtlVirtualUnwind(%rip)

586         mov     \$1,%eax                # ExceptionContinueSearch
587         add     \$64,%rsp
588         popfq
589         pop     %r15
```



```
590         pop     %r14
591         pop     %r13
592         pop     %r12
593         pop     %rbp
594         pop     %rbx
595         pop     %rdi
596         pop     %rsi
597         ret
598 .size   se_handler,.-se_handler

600 .section        .pdata
601 .align  4
602         .rva    .LSEH_begin_$func
603         .rva    .LSEH_end_$func
604         .rva    .LSEH_info_$func

606 .section        .xdata
607 .align  8
608 .LSEH_info_$func:
609         .byte   9,0,0,0
610         .rva    se_handler
611 ___
612 }

614 sub reg_part {
615 my ($reg,$conv)=@_;
616     if ($reg =~ /%r[0-9]+/)     { $reg .= $conv; }
617     elsif ($conv eq "b")        { $reg =~ s/%[er]([^x]+)x?/%$1l/;       }
618     elsif ($conv eq "w")        { $reg =~ s/%[er](.+)/%$1/;             }
619     elsif ($conv eq "d")        { $reg =~ s/%[er](.+)/%e$1/;            }
620     return $reg;
621 }

623 $code =~ s/(%[a-z0-9]+)#([bwd])/reg_part($1,$2)/gem;
624 $code =~ s/\`([^\`]*)\`/eval $1/gem;
625 $code =~ s/pinsrw\s+\$0,/movd   /gm;

627 $code =~ s/#md5#//gm    if ($md5);
628 $code =~ s/#rc4#//gm    if ($rc4);

630 print $code;

632 close STDOUT;
633 #endif /* ! codereview */
```

```
**********************************************************
   15812 Wed Aug 13 19:53:09 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/rc4-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# July 2004
#
# 2.22x RC4 tune-up:-) It should be noted though that my hand [as in
# "hand-coded assembler"] doesn't stand for the whole improvement
# coefficient. It turned out that eliminating RC4_CHAR from config
# line results in ~40% improvement (yes, even for C implementation).
# Presumably it has everything to do with AMD cache architecture and
# RAW or whatever penalties. Once again! The module *requires* config
# line *without* RC4_CHAR! As for coding "secret," I bet on partial
# register arithmetics. For example instead of 'inc %r8; and $255,%r8'
# I simply 'inc %r8b'. Even though optimization manual discourages
# to operate on partial registers, it turned out to be the best bet.
# At least for AMD... How IA32E would perform remains to be seen...

# November 2004
#
# As was shown by Marc Bevand reordering of couple of load operations
# results in even higher performance gain of 3.3x:-) At least on
# Opteron... For reference, 1x in this case is RC4_CHAR C-code
# compiled with gcc 3.3.2, which performs at ~54MBps per 1GHz clock.
# Latter means that if you want to *estimate* what to expect from
# *your* Opteron, then multiply 54 by 3.3 and clock frequency in GHz.

# November 2004
#
# Intel P4 EM64T core was found to run the AMD64 code really slow...
# The only way to achieve comparable performance on P4 was to keep
# RC4_CHAR. Kind of ironic, huh? As it's apparently impossible to
# compose blended code, which would perform even within 30% marginal
# on either AMD and Intel platforms, I implement both cases. See
# rc4_skey.c for further details...

# April 2005
#
# P4 EM64T core appears to be "allergic" to 64-bit inc/dec. Replacing
# those with add/sub results in 50% performance improvement of folded
# loop...

# May 2005
#
# As was shown by Zou Nanhai loop unrolling can improve Intel EM64T
# performance by >30% [unlike P4 32-bit case that is]. But this is
# provided that loads are reordered even more aggressively! Both code
# pathes, AMD64 and EM64T, reorder loads in essentially same manner
# as my IA-64 implementation. On Opteron this resulted in modest 5%
# improvement [I had to test it], while final Intel P4 performance
# achieves respectful 432MBps on 2.8GHz processor now. For reference.
# If executed on Xeon, current RC4_CHAR code-path is 2.7x faster than
# RC4_INT code-path. While if executed on Opteron, it's only 25%
# slower than the RC4_INT one [meaning that if CPU µ-arch detection
# is not implemented, then this final RC4_CHAR code-path should be
# preferred, as it provides better *all-round* performance].
```



```
# March 2007
#
# Intel Core2 was observed to perform poorly on both code paths:-( It
# apparently suffers from some kind of partial register stall, which
# occurs in 64-bit mode only [as virtually identical 32-bit loop was
# observed to outperform 64-bit one by almost 50%]. Adding two movzb to
# cloop1 boosts its performance by 80%! This loop appears to be optimal
# fit for Core2 and therefore the code was modified to skip cloop8 on
# this CPU.

# May 2010
#
# Intel Westmere was observed to perform suboptimally. Adding yet
# another movzb to cloop1 improved performance by almost 50%! Core2
# performance is improved too, but nominally...

# May 2011
#
# The only code path that was not modified is P4-specific one. Non-P4
# Intel code path optimization is heavily based on submission by Maxim
# Perminov, Maxim Locktyukhin and Jim Guilford of Intel. I've used
# some of the ideas even in attempt to optmize the original RC4_INT
# code path... Current performance in cycles per processed byte (less
# is better) and improvement coefficients relative to previous
# version of this module are:
#
# Opteron       5.3/+0%(*)
# P4            6.5
# Core2         6.2/+15%(**)
# Westmere      4.2/+60%
# Sandy Bridge  4.2/+120%
# Atom          9.3/+80%
#
# (*)   But corresponding loop has less instructions, which should have
#       positive effect on upcoming Bulldozer, which has one less ALU.
#       For reference, Intel code runs at 6.8 cpb rate on Opteron.
# (**)  Note that Core2 result is ~15% lower than corresponding result
#       for 32-bit code, meaning that it's possible to improve it,
#       but more than likely at the cost of the others (see rc4-586.pl
#       to get the idea)...

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

$dat="%rdi";        # arg1
$len="%rsi";        # arg2
$inp="%rdx";        # arg3
$out="%rcx";        # arg4

{
$code=<<___;
.text
.extern OPENSSL_ia32cap_P
```

```
.globl  RC4
.type   RC4,\@function,4
.align  16
RC4:    or      $len,$len
        jne     .Lentry
        ret
.Lentry:
        push    %rbx
        push    %r12
        push    %r13
.Lprologue:
        mov     $len,%r11
        mov     $inp,%r12
        mov     $out,%r13
___
my $len="%r11";         # reassign input arguments
my $inp="%r12";
my $out="%r13";

my @XX=("%r10","%rsi");
my @TX=("%rax","%rbx");
my $YY="%rcx";
my $TY="%rdx";

$code.=<<___;
        xor     $XX[0],$XX[0]
        xor     $YY,$YY

        lea     8($dat),$dat
        mov     -8($dat),$XX[0]#b
        mov     -4($dat),$YY#b
        cmpl    \$-1,256($dat)
        je      .LRC4_CHAR
        mov     OPENSSL_ia32cap_P(%rip),%r8d
        xor     $TX[1],$TX[1]
        inc     $XX[0]#b
        sub     $XX[0],$TX[1]
        sub     $inp,$out
        movl    ($dat,$XX[0],4),$TX[0]#d
        test    \$-16,$len
        jz      .Lloop1
        bt      \$30,%r8d       # Intel CPU?
        jc      .Lintel
        and     \$7,$TX[1]
        lea     1($XX[0]),$XX[1]
        jz      .Loop8
        sub     $TX[1],$len
.Loop8_warmup:
        add     $TX[0]#b,$YY#b
        movl    ($dat,$YY,4),$TY#d
        movl    $TX[0]#d,($dat,$YY,4)
        movl    $TY#d,($dat,$XX[0],4)
        add     $TY#b,$TX[0]#b
        inc     $XX[0]#b
        movl    ($dat,$TX[0],4),$TY#d
        movl    ($dat,$XX[0],4),$TX[0]#d
        xorb    ($inp),$TY#b
        movb    $TY#b,($out,$inp)
        lea     1($inp),$inp
        dec     $TX[1]
        jnz     .Loop8_warmup

        lea     1($XX[0]),$XX[1]
        jmp     .Loop8
.align  16
.Loop8:
```



```
___
for ($i=0;$i<8;$i++) {
$code.=<<___ if ($i==7);
        add     \$8,$XX[1]#b
___
$code.=<<___;
        add     $TX[0]#b,$YY#b
        movl    ($dat,$YY,4),$TY#d
        movl    $TX[0]#d,($dat,$YY,4)
        movl    '4*($i==7?-1:$i)'($dat,$XX[1],4),$TX[1]#d
        ror     \$8,%r8                 # ror is redundant when $i=0
        movl    $TY#d,4*$i($dat,$XX[0],4)
        add     $TX[0]#b,$TY#b
        movb    ($dat,$TY,4),%r8b
___
push(@TX,shift(@TX)); #push(@XX,shift(@XX));    # "rotate" registers
}
$code.=<<___;
        add     \$8,$XX[0]#b
        ror     \$8,%r8
        sub     \$8,$len

        xor     ($inp),%r8
        mov     %r8,($out,$inp)
        lea     8($inp),$inp

        test    \$-8,$len
        jnz     .Loop8
        cmp     \$0,$len
        jne     .Lloop1
        jmp     .Lexit

.align  16
.Lintel:
        test    \$-32,$len
        jz      .Lloop1
        and     \$15,$TX[1]
        jz      .Loop16_is_hot
        sub     $TX[1],$len
.Loop16_warmup:
        add     $TX[0]#b,$YY#b
        movl    ($dat,$YY,4),$TY#d
        movl    $TX[0]#d,($dat,$YY,4)
        movl    $TY#d,($dat,$XX[0],4)
        add     $TY#b,$TX[0]#b
        inc     $XX[0]#b
        movl    ($dat,$TX[0],4),$TY#d
        movl    ($dat,$XX[0],4),$TX[0]#d
        xorb    ($inp),$TY#b
        movb    $TY#b,($out,$inp)
        lea     1($inp),$inp
        dec     $TX[1]
        jnz     .Loop16_warmup

        mov     $YY,$TX[1]
        xor     $YY,$YY
        mov     $TX[1]#b,$YY#b

.Loop16_is_hot:
        lea     ($dat,$XX[0],4),$XX[1]
___
sub RC4_loop {
  my $i=shift;
  my $j=$i<0?0:$i;
  my $xmm="%xmm".($j&1);
```

```
260     $code.="    add     \$16,$XX[0]#b\n"                if ($i==15);
261     $code.="    movdqu  ($inp),%xmm2\n"                 if ($i==15);
262     $code.="    add     $TX[0]#b,$YY#b\n"               if ($i<=0);
263     $code.="    movl    ($dat,$YY,4),$TY#d\n";
264     $code.="    pxor    %xmm0,%xmm2\n"                  if ($i==0);
265     $code.="    psllq   \$8,%xmm1\n"                    if ($i==0);
266     $code.="    pxor    $xmm,$xmm\n"                    if ($i<=1);
267     $code.="    movl    $TX[0]#d,($dat,$YY,4)\n";
268     $code.="    add     $TY#b,$TX[0]#b\n";
269     $code.="    movl    '4*($j+1)'($XX[1]),$TX[1]#d\n"  if ($i<15);
270     $code.="    movz    $TX[0]#b,$TX[0]#d\n";
271     $code.="    movl    $TY#d,4*$j($XX[1])\n";
272     $code.="    pxor    %xmm1,%xmm2\n"                  if ($i==0);
273     $code.="    lea     ($dat,$XX[0],4),$XX[1]\n"       if ($i==15);
274     $code.="    add     $TX[1]#b,$YY#b\n"               if ($i<15);
275     $code.="    pinsrw  \$'($j>>1)&7',($dat,$TX[0],4),$xmm\n";
276     $code.="    movdqu  %xmm2,($out,$inp)\n"            if ($i==0);
277     $code.="    lea     16($inp),$inp\n"                if ($i==0);
278     $code.="    movl    ($XX[1]),$TX[1]#d\n"            if ($i==15);
279 }
280         RC4_loop(-1);
281 $code.=<<___;
282         jmp     .Loop16_enter
283 .align  16
284 .Loop16:
285 ___

287 for ($i=0;$i<16;$i++) {
288     $code.=".Loop16_enter:\n"           if ($i==1);
289         RC4_loop($i);
290         push(@TX,shift(@TX));           # "rotate" registers
291 }
292 $code.=<<___;
293         mov     $YY,$TX[1]
294         xor     $YY,$YY                 # keyword to partial register
295         sub     \$16,$len
296         mov     $TX[1]#b,$YY#b
297         test    \$-16,$len
298         jnz     .Loop16

300         psllq   \$8,%xmm1
301         pxor    %xmm0,%xmm2
302         pxor    %xmm1,%xmm2
303         movdqu  %xmm2,($out,$inp)
304         lea     16($inp),$inp

306         cmp     \$0,$len
307         jne     .Lloop1
308         jmp     .Lexit

310 .align  16
311 .Lloop1:
312         add     $TX[0]#b,$YY#b
313         movl    ($dat,$YY,4),$TY#d
314         movl    $TX[0]#d,($dat,$YY,4)
315         movl    $TY#d,($dat,$XX[0],4)
316         add     $TY#b,$TX[0]#b
317         inc     $XX[0]#b
318         movl    ($dat,$TX[0],4),$TY#d
319         movl    ($dat,$XX[0],4),$TX[0]#d
320         xorb    ($inp),$TY#b
321         movb    $TY#b,($out,$inp)
322         lea     1($inp),$inp
323         dec     $len
324         jnz     .Lloop1
325         jmp     .Lexit
```



```
327 .align  16
328 .LRC4_CHAR:
329         add     \$1,$XX[0]#b
330         movzb   ($dat,$XX[0]),$TX[0]#d
331         test    \$-8,$len
332         jz      .Lcloop1
333         jmp     .Lcloop8
334 .align  16
335 .Lcloop8:
336         mov     ($inp),%r8d
337         mov     4($inp),%r9d
338 ___
339 # unroll 2x4-wise, because 64-bit rotates kill Intel P4...
340 for ($i=0;$i<4;$i++) {
341 $code.=<<___;
342         add     $TX[0]#b,$YY#b
343         lea     1($XX[0]),$XX[1]
344         movzb   ($dat,$YY),$TY#d
345         movzb   $XX[1]#b,$XX[1]#d
346         movzb   ($dat,$XX[1]),$TX[1]#d
347         movb    $TX[0]#b,($dat,$YY)
348         cmp     $XX[1],$YY
349         movb    $TY#b,($dat,$XX[0])
350         jne     .Lcmov$i                        # Intel cmov is sloooow...
351         mov     $TX[0],$TX[1]
352 .Lcmov$i:
353         add     $TX[0]#b,$TY#b
354         xor     ($dat,$TY),%r8b
355         ror     \$8,%r8d
356 ___
357 push(@TX,shift(@TX)); push(@XX,shift(@XX));     # "rotate" registers
358 }
359 for ($i=4;$i<8;$i++) {
360 $code.=<<___;
361         add     $TX[0]#b,$YY#b
362         lea     1($XX[0]),$XX[1]
363         movzb   ($dat,$YY),$TY#d
364         movzb   $XX[1]#b,$XX[1]#d
365         movzb   ($dat,$XX[1]),$TX[1]#d
366         movb    $TX[0]#b,($dat,$YY)
367         cmp     $XX[1],$YY
368         movb    $TY#b,($dat,$XX[0])
369         jne     .Lcmov$i                        # Intel cmov is sloooow...
370         mov     $TX[0],$TX[1]
371 .Lcmov$i:
372         add     $TX[0]#b,$TY#b
373         xor     ($dat,$TY),%r9b
374         ror     \$8,%r9d
375 ___
376 push(@TX,shift(@TX)); push(@XX,shift(@XX));     # "rotate" registers
377 }
378 $code.=<<___;
379         lea     -8($len),$len
380         mov     %r8d,($out)
381         lea     8($inp),$inp
382         mov     %r9d,4($out)
383         lea     8($out),$out

385         test    \$-8,$len
386         jnz     .Lcloop8
387         cmp     \$0,$len
388         jne     .Lcloop1
389         jmp     .Lexit
390 ___
391 $code.=<<___;
```

```
392 .align  16
393 .Lcloop1:
394         add     $TX[0]#b,$YY#b
395         movzb   $YY#b,$YY#d
396         movzb   ($dat,$YY),$TY#d
397         movb    $TX[0]#b,($dat,$YY)
398         movb    $TY#b,($dat,$XX[0])
399         add     $TX[0]#b,$TY#b
400         add     \$1,$XX[0]#b
401         movzb   $TY#b,$TY#d
402         movzb   $XX[0]#b,$XX[0]#d
403         movzb   ($dat,$TY),$TY#d
404         movzb   ($dat,$XX[0]),$TX[0]#d
405         xorb    ($inp),$TY#b
406         lea     1($inp),$inp
407         movb    $TY#b,($out)
408         lea     1($out),$out
409         sub     \$1,$len
410         jnz     .Lcloop1
411         jmp     .Lexit

413 .align  16
414 .Lexit:
415         sub     \$1,$XX[0]#b
416         movl    $XX[0]#d,-8($dat)
417         movl    $YY#d,-4($dat)

419         mov     (%rsp),%r13
420         mov     8(%rsp),%r12
421         mov     16(%rsp),%rbx
422         add     \$24,%rsp
423 .Lepilogue:
424         ret
425 .size   RC4,.-RC4
426 ___
427 }

429 $idx="%r8";
430 $ido="%r9";

432 $code.=<<___;
433 .globl  private_RC4_set_key
434 .type   private_RC4_set_key,\@function,3
435 .align  16
436 private_RC4_set_key:
437         lea     8($dat),$dat
438         lea     ($inp,$len),$inp
439         neg     $len
440         mov     $len,%rcx
441         xor     %eax,%eax
442         xor     $ido,$ido
443         xor     %r10,%r10
444         xor     %r11,%r11

446         mov     OPENSSL_ia32cap_P(%rip),$idx#d
447         bt      \$20,$idx#d     # RC4_CHAR?
448         jc      .Lc1stloop
449         jmp     .Lw1stloop

451 .align  16
452 .Lw1stloop:
453         mov     %eax,($dat,%rax,4)
454         add     \$1,%al
455         jnc     .Lw1stloop

457         xor     $ido,$ido
```



```
458         xor     $idx,$idx
459 .align  16
460 .Lw2ndloop:
461         mov     ($dat,$ido,4),%r10d
462         add     ($inp,$len,1),$idx#b
463         add     %r10b,$idx#b
464         add     \$1,$len
465         mov     ($dat,$idx,4),%r11d
466         cmovz   %rcx,$len
467         mov     %r10d,($dat,$idx,4)
468         mov     %r11d,($dat,$ido,4)
469         add     \$1,$ido#b
470         jnc     .Lw2ndloop
471         jmp     .Lexit_key

473 .align  16
474 .Lc1stloop:
475         mov     %al,($dat,%rax)
476         add     \$1,%al
477         jnc     .Lc1stloop

479         xor     $ido,$ido
480         xor     $idx,$idx
481 .align  16
482 .Lc2ndloop:
483         mov     ($dat,$ido),%r10b
484         add     ($inp,$len),$idx#b
485         add     %r10b,$idx#b
486         add     \$1,$len
487         mov     ($dat,$idx),%r11b
488         jnz     .Lcnowrap
489         mov     %rcx,$len
490 .Lcnowrap:
491         mov     %r10b,($dat,$idx)
492         mov     %r11b,($dat,$ido)
493         add     \$1,$ido#b
494         jnc     .Lc2ndloop
495         movl    \$-1,256($dat)

497 .align  16
498 .Lexit_key:
499         xor     %eax,%eax
500         mov     %eax,-8($dat)
501         mov     %eax,-4($dat)
502         ret
503 .size   private_RC4_set_key,.-private_RC4_set_key

505 .globl  RC4_options
506 .type   RC4_options,\@abi-omnipotent
507 .align  16
508 RC4_options:
509         lea     .Lopts(%rip),%rax
510         mov     OPENSSL_ia32cap_P(%rip),%edx
511         bt      \$20,%edx
512         jc      .L8xchar
513         bt      \$30,%edx
514         jnc     .Ldone
515         add     \$25,%rax
516         ret
517 .L8xchar:
518         add     \$12,%rax
519 .Ldone:
520         ret
521 .align  64
522 .Lopts:
523 .asciz  "rc4(8x,int)"
```

```
.asciz  "rc4(8x,char)"
.asciz  "rc4(16x,int)"
.asciz  "RC4 for x86_64, CRYPTOGAMS by <appro\@openssl.org>"
.align  64
.size   RC4_options,.-RC4_options
___

# EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
#               CONTEXT *context,DISPATCHER_CONTEXT *disp)
if ($win64) {
$rec="%rcx";
$frame="%rdx";
$context="%r8";
$disp="%r9";

$code.=<<___;
.extern __imp_RtlVirtualUnwind
.type   stream_se_handler,\@abi-omnipotent
.align  16
stream_se_handler:
        push    %rsi
        push    %rdi
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
        pushfq
        sub     \$64,%rsp

        mov     120($context),%rax      # pull context->Rax
        mov     248($context),%rbx      # pull context->Rip

        lea     .Lprologue(%rip),%r10
        cmp     %r10,%rbx               # context->Rip<prologue label
        jb      .Lin_prologue

        mov     152($context),%rax      # pull context->Rsp

        lea     .Lepilogue(%rip),%r10
        cmp     %r10,%rbx               # context->Rip>=epilogue label
        jae     .Lin_prologue

        lea     24(%rax),%rax

        mov     -8(%rax),%rbx
        mov     -16(%rax),%r12
        mov     -24(%rax),%r13
        mov     %rbx,144($context)      # restore context->Rbx
        mov     %r12,216($context)      # restore context->R12
        mov     %r13,224($context)      # restore context->R13

.Lin_prologue:
        mov     8(%rax),%rdi
        mov     16(%rax),%rsi
        mov     %rax,152($context)      # restore context->Rsp
        mov     %rsi,168($context)      # restore context->Rsi
        mov     %rdi,176($context)      # restore context->Rdi

        jmp     .Lcommon_seh_exit
.size   stream_se_handler,.-stream_se_handler

.type   key_se_handler,\@abi-omnipotent
.align  16
key_se_handler:
```



```
        push    %rsi
        push    %rdi
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
        pushfq
        sub     \$64,%rsp

        mov     152($context),%rax      # pull context->Rsp
        mov     8(%rax),%rdi
        mov     16(%rax),%rsi
        mov     %rsi,168($context)      # restore context->Rsi
        mov     %rdi,176($context)      # restore context->Rdi

.Lcommon_seh_exit:

        mov     40($disp),%rdi          # disp->ContextRecord
        mov     $context,%rsi           # context
        mov     \$154,%ecx              # sizeof(CONTEXT)
        .long   0xa548f3fc              # cld; rep movsq

        mov     $disp,%rsi
        xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
        mov     8(%rsi),%rdx            # arg2, disp->ImageBase
        mov     0(%rsi),%r8             # arg3, disp->ControlPc
        mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
        mov     40(%rsi),%r10           # disp->ContextRecord
        lea     56(%rsi),%r11           # &disp->HandlerData
        lea     24(%rsi),%r12           # &disp->EstablisherFrame
        mov     %r10,32(%rsp)           # arg5
        mov     %r11,40(%rsp)           # arg6
        mov     %r12,48(%rsp)           # arg7
        mov     %rcx,56(%rsp)           # arg8, (NULL)
        call    *__imp_RtlVirtualUnwind(%rip)

        mov     \$1,%eax                # ExceptionContinueSearch
        add     \$64,%rsp
        popfq
        pop     %r15
        pop     %r14
        pop     %r13
        pop     %r12
        pop     %rbp
        pop     %rbx
        pop     %rdi
        pop     %rsi
        ret
.size   key_se_handler,.-key_se_handler

.section        .pdata
.align  4
        .rva    .LSEH_begin_RC4
        .rva    .LSEH_end_RC4
        .rva    .LSEH_info_RC4

        .rva    .LSEH_begin_private_RC4_set_key
        .rva    .LSEH_end_private_RC4_set_key
        .rva    .LSEH_info_private_RC4_set_key

.section        .xdata
.align  8
.LSEH_info_RC4:
        .byte   9,0,0,0
```

```
656         .rva    stream_se_handler
657 .LSEH_info_private_RC4_set_key:
658         .byte   9,0,0,0
659         .rva    key_se_handler
660 ___
661 }

663 sub reg_part {
664 my ($reg,$conv)=@_;
665     if ($reg =~ /%r[0-9]+/)     { $reg .= $conv; }
666     elsif ($conv eq "b")        { $reg =~ s/%[er]([^x]+)x?/%$1l/;       }
667     elsif ($conv eq "w")        { $reg =~ s/%[er](.+)/%$1/;             }
668     elsif ($conv eq "d")        { $reg =~ s/%[er](.+)/%e$1/;            }
669     return $reg;
670 }

672 $code =~ s/(%[a-z0-9]+)#([bwd])/reg_part($1,$2)/gem;
673 $code =~ s/\`([^\`]*)\`/eval $1/gem;

675 print $code;

677 close STDOUT;
678 #endif /* ! codereview */
```

```
**********************************************************
   16234 Wed Aug 13 19:53:09 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/rmd-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/local/bin/perl

# Normal is the
# ripemd160_block_asm_data_order(RIPEMD160_CTX *c, ULONG *X,int blocks);

$normal=0;

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],$0);

$A="ecx";
$B="esi";
$C="edi";
$D="ebx";
$E="ebp";
$tmp1="eax";
$tmp2="edx";

$KL1=0x5A827999;
$KL2=0x6ED9EBA1;
$KL3=0x8F1BBCDC;
$KL4=0xA953FD4E;
$KR0=0x50A28BE6;
$KR1=0x5C4DD124;
$KR2=0x6D703EF3;
$KR3=0x7A6D76E9;


@wl=(    0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12,13,14,15,
         7, 4,13, 1,10, 6,15, 3,12, 0, 9, 5, 2,14,11, 8,
         3,10,14, 4, 9,15, 8, 1, 2, 7, 0, 6,13,11, 5,12,
         1, 9,11,10, 0, 8,12, 4,13, 3, 7,15,14, 5, 6, 2,
         4, 0, 5, 9, 7,12, 2,10,14, 1, 3, 8,11, 6,15,13,
         );

@wr=(    5,14, 7, 0, 9, 2,11, 4,13, 6,15, 8, 1,10, 3,12,
         6,11, 3, 7, 0,13, 5,10,14,15, 8,12, 4, 9, 1, 2,
        15, 5, 1, 3, 7,14, 6, 9,11, 8,12, 2,10, 0, 4,13,
         8, 6, 4, 1, 3,11,15, 0, 5,12, 2,13, 9, 7,10,14,
        12,15,10, 4, 1, 5, 8, 7, 6, 2,13,14, 0, 3, 9,11,
        );

@sl=(   11,14,15,12, 5, 8, 7, 9,11,13,14,15, 6, 7, 9, 8,
         7, 6, 8,13,11, 9, 7,15, 7,12,15, 9,11, 7,13,12,
        11,13, 6, 7,14, 9,13,15,14, 8,13, 6, 5,12, 7, 5,
        11,12,14,15,14,15, 9, 8, 9,14, 5, 6, 8, 6, 5,12,
         9,15, 5,11, 6, 8,13,12, 5,12,13,14,11, 8, 5, 6,
         );

@sr=(    8, 9, 9,11,13,15,15, 5, 7, 7, 8,11,14,14,12, 6,
         9,13,15, 7,12, 8, 9,11, 7, 7,12, 7, 6,15,13,11,
         9, 7,15,11, 8, 6, 6,14,12,13, 5,14,13,13, 7, 5,
        15, 5, 8,11,14,14, 6,14, 6, 9,12, 9,12, 5,15, 8,
         8, 5,12, 9,12, 5,14, 6, 8,13, 6, 5,15,13,11,11,
        );

&ripemd160_block("ripemd160_block_asm_data_order");
&asm_finish();
```



```
sub Xv
        {
        local($n)=@_;
        return(&swtmp($n));
        # tmp on stack
        }

sub Np
        {
        local($p)=@_;
        local(%n)=($A,$E,$B,$A,$C,$B,$D,$C,$E,$D);
        return($n{$p});
        }

sub RIP1
        {
        local($a,$b,$c,$d,$e,$pos,$s,$o,$pos2)=@_;

        &comment($p++);
        if ($p & 1)
                {
         #&mov($tmp1,   $c) if $o == -1;
        &xor($tmp1,     $d) if $o == -1;
         &mov($tmp2,    &Xv($pos));
        &xor($tmp1,     $b);
         &add($a,       $tmp2);
        &rotl($c,       10);
        &add($a,        $tmp1);
         &mov($tmp1,    &Np($c));       # NEXT
         # XXX
        &rotl($a,       $s);
        &add($a,        $e);
                }
        else
                {
         &xor($tmp1,    $d);
        &mov($tmp2,     &Xv($pos));
         &xor($tmp1,    $b);
        &add($a,        $tmp1);
         &mov($tmp1,    &Np($c)) if $o <= 0;
         &mov($tmp1,    -1) if $o == 1;
         # XXX if $o == 2;
        &rotl($c,       10);
        &add($a,        $tmp2);
         &xor($tmp1,    &Np($d)) if $o <= 0;
         &mov($tmp2,    &Xv($pos2)) if $o == 1;
         &mov($tmp2,    &wparam(0)) if $o == 2;
        &rotl($a,       $s);
        &add($a,        $e);
                }
        }

sub RIP2
        {
        local($a,$b,$c,$d,$e,$pos,$pos2,$s,$K,$o)=@_;

# XXXXXX
        &comment($p++);
        if ($p & 1)
                {
#        &mov($tmp2,    &Xv($pos)) if $o < -1;
#       &mov($tmp1,     -1) if $o < -1;

         &add($a,       $tmp2);
        &mov($tmp2,     $c);
```

```
128          &sub($tmp1,    $b);
129         &and($tmp2,     $b);
130          &and($tmp1,    $d);
131         &or($tmp2,      $tmp1);
132          &mov($tmp1,    &Xv($pos2)) if $o <= 0; # XXXXXXXXXXXXXX
133          # XXX
134         &rotl($c,       10);
135         &lea($a,        &DWP($K,$a,$tmp2,1));
136          &mov($tmp2,    -1) if $o <= 0;
137          # XXX
138         &rotl($a,       $s);
139         &add($a,        $e);
140                 }
141         else
142                 {
143          # XXX
144          &add($a,       $tmp1);
145         &mov($tmp1,     $c);
146          &sub($tmp2,    $b);
147         &and($tmp1,     $b);
148          &and($tmp2,    $d);
149         if ($o != 2)
150                 {
151         &or($tmp1,      $tmp2);
152          &mov($tmp2,    &Xv($pos2)) if $o <= 0;
153          &mov($tmp2,    -1) if $o == 1;
154         &rotl($c,       10);
155         &lea($a,        &DWP($K,$a,$tmp1,1));
156          &mov($tmp1,    -1) if $o <= 0;
157          &sub($tmp2,    &Np($c)) if $o == 1;
158                 } else {
159         &or($tmp2,      $tmp1);
160          &mov($tmp1,    &Np($c));
161         &rotl($c,       10);
162         &lea($a,        &DWP($K,$a,$tmp2,1));
163          &xor($tmp1,    &Np($d));
164                 }
165         &rotl($a,       $s);
166         &add($a,        $e);
167                 }
168         }

170 sub RIP3
171         {
172         local($a,$b,$c,$d,$e,$pos,$s,$K,$o,$pos2)=@_;

174         &comment($p++);
175         if ($p & 1)
176                 {
177 #        &mov($tmp2,    -1) if $o < -1;
178 #       &sub($tmp2,     $c) if $o < -1;
179          &mov($tmp1,    &Xv($pos));
180         &or($tmp2,      $b);
181          &add($a,       $tmp1);
182         &xor($tmp2,     $d);
183          &mov($tmp1,    -1) if $o <= 0;         # NEXT
184          # XXX
185         &rotl($c,       10);
186         &lea($a,        &DWP($K,$a,$tmp2,1));
187          &sub($tmp1,    &Np($c)) if $o <= 0;    # NEXT
188          # XXX
189         &rotl($a,       $s);
190         &add($a,        $e);
191                 }
192         else
193                 {
```



```
194          &mov($tmp2,    &Xv($pos));
195         &or($tmp1,      $b);
196          &add($a,       $tmp2);
197         &xor($tmp1,     $d);
198          &mov($tmp2,    -1) if $o <= 0;         # NEXT
199          &mov($tmp2,    -1) if $o == 1;
200          &mov($tmp2,    &Xv($pos2)) if $o == 2;
201         &rotl($c,       10);
202         &lea($a,        &DWP($K,$a,$tmp1,1));
203          &sub($tmp2,    &Np($c)) if $o <= 0;    # NEXT
204          &mov($tmp1,    &Np($d)) if $o == 1;
205          &mov($tmp1,    -1) if $o == 2;
206         &rotl($a,       $s);
207         &add($a,        $e);
208                 }
209         }

211 sub RIP4
212         {
213         local($a,$b,$c,$d,$e,$pos,$s,$K,$o)=@_;

215         &comment($p++);
216         if ($p & 1)
217                 {
218 #        &mov($tmp2,    -1) if $o == -2;
219 #       &mov($tmp1,     $d) if $o == -2;
220          &sub($tmp2,    $d);
221         &and($tmp1,     $b);
222          &and($tmp2,    $c);
223         &or($tmp2,      $tmp1);
224          &mov($tmp1,    &Xv($pos));
225         &rotl($c,       10);
226         &lea($a,        &DWP($K,$a,$tmp2));
227          &mov($tmp2,    -1) unless $o > 0;      # NEXT
228          # XXX
229         &add($a,        $tmp1);
230          &mov($tmp1,    &Np($d)) unless $o > 0; # NEXT
231          # XXX
232         &rotl($a,       $s);
233         &add($a,        $e);
234                 }
235         else
236                 {
237          &sub($tmp2,    $d);
238         &and($tmp1,     $b);
239          &and($tmp2,    $c);
240         &or($tmp2,      $tmp1);
241          &mov($tmp1,    &Xv($pos));
242         &rotl($c,       10);
243         &lea($a,        &DWP($K,$a,$tmp2));
244          &mov($tmp2,    -1) if $o == 0; # NEXT
245          &mov($tmp2,    -1) if $o == 1;
246          &mov($tmp2,    -1) if $o == 2;
247          # XXX
248         &add($a,        $tmp1);
249          &mov($tmp1,    &Np($d)) if $o == 0;    # NEXT
250          &sub($tmp2,    &Np($d)) if $o == 1;
251          &sub($tmp2,    &Np($c)) if $o == 2;
252          # XXX
253         &rotl($a,       $s);
254         &add($a,        $e);
255                 }
256         }

258 sub RIP5
259         {
```

```
260         local($a,$b,$c,$d,$e,$pos,$s,$K,$o)=@_;
261
262         &comment($p++);
263         if ($p & 1)
264                 {
265          &mov($tmp2,    -1) if $o == -2;
266         &sub($tmp2,     $d) if $o == -2;
267          &mov($tmp1,    &Xv($pos));
268         &or($tmp2,      $c);
269          &add($a,       $tmp1);
270         &xor($tmp2,     $b);
271          &mov($tmp1,    -1) if $o <= 0;
272          # XXX
273         &rotl($c,       10);
274         &lea($a,        &DWP($K,$a,$tmp2,1));
275          &sub($tmp1,    &Np($d)) if $o <= 0;
276          # XXX
277         &rotl($a,       $s);
278         &add($a,        $e);
279                 }
280         else
281                 {
282          &mov($tmp2,    &Xv($pos));
283         &or($tmp1,      $c);
284          &add($a,       $tmp2);
285         &xor($tmp1,     $b);
286          &mov($tmp2,    -1) if $o <= 0;
287          &mov($tmp2,    &wparam(0)) if $o == 1; # Middle code
288          &mov($tmp2,    -1) if $o == 2;
289         &rotl($c,       10);
290         &lea($a,        &DWP($K,$a,$tmp1,1));
291          &sub($tmp2,    &Np($d)) if $o <= 0;
292          &mov(&swtmp(16),       $A) if $o == 1;
293          &mov($tmp1,    &Np($d)) if $o == 2;
294         &rotl($a,       $s);
295         &add($a,        $e);
296                 }
297         }
298
299 sub ripemd160_block
300         {
301         local($name)=@_;
302
303         &function_begin_B($name,"",3);
304
305         # parameter 1 is the RIPEMD160_CTX structure.
306         # A     0
307         # B     4
308         # C     8
309         # D     12
310         # E     16
311
312         &mov($tmp2,     &wparam(0));
313          &mov($tmp1,    &wparam(1));
314         &push("esi");
315          &mov($A,       &DWP( 0,$tmp2,"",0));
316         &push("edi");
317          &mov($B,       &DWP( 4,$tmp2,"",0));
318         &push("ebp");
319          &mov($C,       &DWP( 8,$tmp2,"",0));
320         &push("ebx");
321          &stack_push(16+5+6);
322                           # Special comment about the figure of 6.
323                           # Idea is to pad the current frame so
324                           # that the top of the stack gets fairly
325                           # aligned. Well, as you realize it would
```



```
326                           # always depend on how the frame below is
327                           # aligned. The good news are that gcc-2.95
328                           # and later does keep first argument at
329                           # least double-wise aligned.
330                           #                       <appro@fy.chalmers.se>
331
332         &set_label("start") unless $normal;
333         &comment("");
334
335         # &mov($tmp1,   &wparam(1)); # Done at end of loop
336         # &mov($tmp2,   &wparam(0)); # Done at end of loop
337
338         for ($z=0; $z<16; $z+=2)
339                 {
340                 &mov($D,                &DWP( $z*4,$tmp1,"",0));
341                  &mov($E,               &DWP( ($z+1)*4,$tmp1,"",0));
342                 &mov(&swtmp($z),        $D);
343                  &mov(&swtmp($z+1),     $E);
344                 }
345         &mov($tmp1,     $C);
346          &mov($D,       &DWP(12,$tmp2,"",0));
347         &mov($E,        &DWP(16,$tmp2,"",0));
348
349         &RIP1($A,$B,$C,$D,$E,$wl[ 0],$sl[ 0],-1);
350         &RIP1($E,$A,$B,$C,$D,$wl[ 1],$sl[ 1],0);
351         &RIP1($D,$E,$A,$B,$C,$wl[ 2],$sl[ 2],0);
352         &RIP1($C,$D,$E,$A,$B,$wl[ 3],$sl[ 3],0);
353         &RIP1($B,$C,$D,$E,$A,$wl[ 4],$sl[ 4],0);
354         &RIP1($A,$B,$C,$D,$E,$wl[ 5],$sl[ 5],0);
355         &RIP1($E,$A,$B,$C,$D,$wl[ 6],$sl[ 6],0);
356         &RIP1($D,$E,$A,$B,$C,$wl[ 7],$sl[ 7],0);
357         &RIP1($C,$D,$E,$A,$B,$wl[ 8],$sl[ 8],0);
358         &RIP1($B,$C,$D,$E,$A,$wl[ 9],$sl[ 9],0);
359         &RIP1($A,$B,$C,$D,$E,$wl[10],$sl[10],0);
360         &RIP1($E,$A,$B,$C,$D,$wl[11],$sl[11],0);
361         &RIP1($D,$E,$A,$B,$C,$wl[12],$sl[12],0);
362         &RIP1($C,$D,$E,$A,$B,$wl[13],$sl[13],0);
363         &RIP1($B,$C,$D,$E,$A,$wl[14],$sl[14],0);
364         &RIP1($A,$B,$C,$D,$E,$wl[15],$sl[15],1,$wl[16]);
365
366         &RIP2($E,$A,$B,$C,$D,$wl[16],$wl[17],$sl[16],$KL1,-1);
367         &RIP2($D,$E,$A,$B,$C,$wl[17],$wl[18],$sl[17],$KL1,0);
368         &RIP2($C,$D,$E,$A,$B,$wl[18],$wl[19],$sl[18],$KL1,0);
369         &RIP2($B,$C,$D,$E,$A,$wl[19],$wl[20],$sl[19],$KL1,0);
370         &RIP2($A,$B,$C,$D,$E,$wl[20],$wl[21],$sl[20],$KL1,0);
371         &RIP2($E,$A,$B,$C,$D,$wl[21],$wl[22],$sl[21],$KL1,0);
372         &RIP2($D,$E,$A,$B,$C,$wl[22],$wl[23],$sl[22],$KL1,0);
373         &RIP2($C,$D,$E,$A,$B,$wl[23],$wl[24],$sl[23],$KL1,0);
374         &RIP2($B,$C,$D,$E,$A,$wl[24],$wl[25],$sl[24],$KL1,0);
375         &RIP2($A,$B,$C,$D,$E,$wl[25],$wl[26],$sl[25],$KL1,0);
376         &RIP2($E,$A,$B,$C,$D,$wl[26],$wl[27],$sl[26],$KL1,0);
377         &RIP2($D,$E,$A,$B,$C,$wl[27],$wl[28],$sl[27],$KL1,0);
378         &RIP2($C,$D,$E,$A,$B,$wl[28],$wl[29],$sl[28],$KL1,0);
379         &RIP2($B,$C,$D,$E,$A,$wl[29],$wl[30],$sl[29],$KL1,0);
380         &RIP2($A,$B,$C,$D,$E,$wl[30],$wl[31],$sl[30],$KL1,0);
381         &RIP2($E,$A,$B,$C,$D,$wl[31],$wl[32],$sl[31],$KL1,1);
382
383         &RIP3($D,$E,$A,$B,$C,$wl[32],$sl[32],$KL2,-1);
384         &RIP3($C,$D,$E,$A,$B,$wl[33],$sl[33],$KL2,0);
385         &RIP3($B,$C,$D,$E,$A,$wl[34],$sl[34],$KL2,0);
386         &RIP3($A,$B,$C,$D,$E,$wl[35],$sl[35],$KL2,0);
387         &RIP3($E,$A,$B,$C,$D,$wl[36],$sl[36],$KL2,0);
388         &RIP3($D,$E,$A,$B,$C,$wl[37],$sl[37],$KL2,0);
389         &RIP3($C,$D,$E,$A,$B,$wl[38],$sl[38],$KL2,0);
390         &RIP3($B,$C,$D,$E,$A,$wl[39],$sl[39],$KL2,0);
391         &RIP3($A,$B,$C,$D,$E,$wl[40],$sl[40],$KL2,0);
```

```
392         &RIP3($E,$A,$B,$C,$D,$wl[41],$sl[41],$KL2,0);
393         &RIP3($D,$E,$A,$B,$C,$wl[42],$sl[42],$KL2,0);
394         &RIP3($C,$D,$E,$A,$B,$wl[43],$sl[43],$KL2,0);
395         &RIP3($B,$C,$D,$E,$A,$wl[44],$sl[44],$KL2,0);
396         &RIP3($A,$B,$C,$D,$E,$wl[45],$sl[45],$KL2,0);
397         &RIP3($E,$A,$B,$C,$D,$wl[46],$sl[46],$KL2,0);
398         &RIP3($D,$E,$A,$B,$C,$wl[47],$sl[47],$KL2,1);

400         &RIP4($C,$D,$E,$A,$B,$wl[48],$sl[48],$KL3,-1);
401         &RIP4($B,$C,$D,$E,$A,$wl[49],$sl[49],$KL3,0);
402         &RIP4($A,$B,$C,$D,$E,$wl[50],$sl[50],$KL3,0);
403         &RIP4($E,$A,$B,$C,$D,$wl[51],$sl[51],$KL3,0);
404         &RIP4($D,$E,$A,$B,$C,$wl[52],$sl[52],$KL3,0);
405         &RIP4($C,$D,$E,$A,$B,$wl[53],$sl[53],$KL3,0);
406         &RIP4($B,$C,$D,$E,$A,$wl[54],$sl[54],$KL3,0);
407         &RIP4($A,$B,$C,$D,$E,$wl[55],$sl[55],$KL3,0);
408         &RIP4($E,$A,$B,$C,$D,$wl[56],$sl[56],$KL3,0);
409         &RIP4($D,$E,$A,$B,$C,$wl[57],$sl[57],$KL3,0);
410         &RIP4($C,$D,$E,$A,$B,$wl[58],$sl[58],$KL3,0);
411         &RIP4($B,$C,$D,$E,$A,$wl[59],$sl[59],$KL3,0);
412         &RIP4($A,$B,$C,$D,$E,$wl[60],$sl[60],$KL3,0);
413         &RIP4($E,$A,$B,$C,$D,$wl[61],$sl[61],$KL3,0);
414         &RIP4($D,$E,$A,$B,$C,$wl[62],$sl[62],$KL3,0);
415         &RIP4($C,$D,$E,$A,$B,$wl[63],$sl[63],$KL3,1);

417         &RIP5($B,$C,$D,$E,$A,$wl[64],$sl[64],$KL4,-1);
418         &RIP5($A,$B,$C,$D,$E,$wl[65],$sl[65],$KL4,0);
419         &RIP5($E,$A,$B,$C,$D,$wl[66],$sl[66],$KL4,0);
420         &RIP5($D,$E,$A,$B,$C,$wl[67],$sl[67],$KL4,0);
421         &RIP5($C,$D,$E,$A,$B,$wl[68],$sl[68],$KL4,0);
422         &RIP5($B,$C,$D,$E,$A,$wl[69],$sl[69],$KL4,0);
423         &RIP5($A,$B,$C,$D,$E,$wl[70],$sl[70],$KL4,0);
424         &RIP5($E,$A,$B,$C,$D,$wl[71],$sl[71],$KL4,0);
425         &RIP5($D,$E,$A,$B,$C,$wl[72],$sl[72],$KL4,0);
426         &RIP5($C,$D,$E,$A,$B,$wl[73],$sl[73],$KL4,0);
427         &RIP5($B,$C,$D,$E,$A,$wl[74],$sl[74],$KL4,0);
428         &RIP5($A,$B,$C,$D,$E,$wl[75],$sl[75],$KL4,0);
429         &RIP5($E,$A,$B,$C,$D,$wl[76],$sl[76],$KL4,0);
430         &RIP5($D,$E,$A,$B,$C,$wl[77],$sl[77],$KL4,0);
431         &RIP5($C,$D,$E,$A,$B,$wl[78],$sl[78],$KL4,0);
432         &RIP5($B,$C,$D,$E,$A,$wl[79],$sl[79],$KL4,1);

434         # &mov($tmp2,   &wparam(0)); # moved into last RIP5
435         # &mov(&swtmp(16),      $A);
436          &mov($A,       &DWP( 0,$tmp2,"",0));
437         &mov(&swtmp(16+1),      $B);
438          &mov(&swtmp(16+2),     $C);
439         &mov($B,        &DWP( 4,$tmp2,"",0));
440          &mov(&swtmp(16+3),     $D);
441         &mov($C,        &DWP( 8,$tmp2,"",0));
442          &mov(&swtmp(16+4),     $E);
443         &mov($D,        &DWP(12,$tmp2,"",0));
444          &mov($E,       &DWP(16,$tmp2,"",0));

446         &RIP5($A,$B,$C,$D,$E,$wr[ 0],$sr[ 0],$KR0,-2);
447         &RIP5($E,$A,$B,$C,$D,$wr[ 1],$sr[ 1],$KR0,0);
448         &RIP5($D,$E,$A,$B,$C,$wr[ 2],$sr[ 2],$KR0,0);
449         &RIP5($C,$D,$E,$A,$B,$wr[ 3],$sr[ 3],$KR0,0);
450         &RIP5($B,$C,$D,$E,$A,$wr[ 4],$sr[ 4],$KR0,0);
451         &RIP5($A,$B,$C,$D,$E,$wr[ 5],$sr[ 5],$KR0,0);
452         &RIP5($E,$A,$B,$C,$D,$wr[ 6],$sr[ 6],$KR0,0);
453         &RIP5($D,$E,$A,$B,$C,$wr[ 7],$sr[ 7],$KR0,0);
454         &RIP5($C,$D,$E,$A,$B,$wr[ 8],$sr[ 8],$KR0,0);
455         &RIP5($B,$C,$D,$E,$A,$wr[ 9],$sr[ 9],$KR0,0);
456         &RIP5($A,$B,$C,$D,$E,$wr[10],$sr[10],$KR0,0);
457         &RIP5($E,$A,$B,$C,$D,$wr[11],$sr[11],$KR0,0);
```



```
458         &RIP5($D,$E,$A,$B,$C,$wr[12],$sr[12],$KR0,0);
459         &RIP5($C,$D,$E,$A,$B,$wr[13],$sr[13],$KR0,0);
460         &RIP5($B,$C,$D,$E,$A,$wr[14],$sr[14],$KR0,0);
461         &RIP5($A,$B,$C,$D,$E,$wr[15],$sr[15],$KR0,2);

463         &RIP4($E,$A,$B,$C,$D,$wr[16],$sr[16],$KR1,-2);
464         &RIP4($D,$E,$A,$B,$C,$wr[17],$sr[17],$KR1,0);
465         &RIP4($C,$D,$E,$A,$B,$wr[18],$sr[18],$KR1,0);
466         &RIP4($B,$C,$D,$E,$A,$wr[19],$sr[19],$KR1,0);
467         &RIP4($A,$B,$C,$D,$E,$wr[20],$sr[20],$KR1,0);
468         &RIP4($E,$A,$B,$C,$D,$wr[21],$sr[21],$KR1,0);
469         &RIP4($D,$E,$A,$B,$C,$wr[22],$sr[22],$KR1,0);
470         &RIP4($C,$D,$E,$A,$B,$wr[23],$sr[23],$KR1,0);
471         &RIP4($B,$C,$D,$E,$A,$wr[24],$sr[24],$KR1,0);
472         &RIP4($A,$B,$C,$D,$E,$wr[25],$sr[25],$KR1,0);
473         &RIP4($E,$A,$B,$C,$D,$wr[26],$sr[26],$KR1,0);
474         &RIP4($D,$E,$A,$B,$C,$wr[27],$sr[27],$KR1,0);
475         &RIP4($C,$D,$E,$A,$B,$wr[28],$sr[28],$KR1,0);
476         &RIP4($B,$C,$D,$E,$A,$wr[29],$sr[29],$KR1,0);
477         &RIP4($A,$B,$C,$D,$E,$wr[30],$sr[30],$KR1,0);
478         &RIP4($E,$A,$B,$C,$D,$wr[31],$sr[31],$KR1,2);

480         &RIP3($D,$E,$A,$B,$C,$wr[32],$sr[32],$KR2,-2);
481         &RIP3($C,$D,$E,$A,$B,$wr[33],$sr[33],$KR2,0);
482         &RIP3($B,$C,$D,$E,$A,$wr[34],$sr[34],$KR2,0);
483         &RIP3($A,$B,$C,$D,$E,$wr[35],$sr[35],$KR2,0);
484         &RIP3($E,$A,$B,$C,$D,$wr[36],$sr[36],$KR2,0);
485         &RIP3($D,$E,$A,$B,$C,$wr[37],$sr[37],$KR2,0);
486         &RIP3($C,$D,$E,$A,$B,$wr[38],$sr[38],$KR2,0);
487         &RIP3($B,$C,$D,$E,$A,$wr[39],$sr[39],$KR2,0);
488         &RIP3($A,$B,$C,$D,$E,$wr[40],$sr[40],$KR2,0);
489         &RIP3($E,$A,$B,$C,$D,$wr[41],$sr[41],$KR2,0);
490         &RIP3($D,$E,$A,$B,$C,$wr[42],$sr[42],$KR2,0);
491         &RIP3($C,$D,$E,$A,$B,$wr[43],$sr[43],$KR2,0);
492         &RIP3($B,$C,$D,$E,$A,$wr[44],$sr[44],$KR2,0);
493         &RIP3($A,$B,$C,$D,$E,$wr[45],$sr[45],$KR2,0);
494         &RIP3($E,$A,$B,$C,$D,$wr[46],$sr[46],$KR2,0);
495         &RIP3($D,$E,$A,$B,$C,$wr[47],$sr[47],$KR2,2,$wr[48]);

497         &RIP2($C,$D,$E,$A,$B,$wr[48],$wr[49],$sr[48],$KR3,-2);
498         &RIP2($B,$C,$D,$E,$A,$wr[49],$wr[50],$sr[49],$KR3,0);
499         &RIP2($A,$B,$C,$D,$E,$wr[50],$wr[51],$sr[50],$KR3,0);
500         &RIP2($E,$A,$B,$C,$D,$wr[51],$wr[52],$sr[51],$KR3,0);
501         &RIP2($D,$E,$A,$B,$C,$wr[52],$wr[53],$sr[52],$KR3,0);
502         &RIP2($C,$D,$E,$A,$B,$wr[53],$wr[54],$sr[53],$KR3,0);
503         &RIP2($B,$C,$D,$E,$A,$wr[54],$wr[55],$sr[54],$KR3,0);
504         &RIP2($A,$B,$C,$D,$E,$wr[55],$wr[56],$sr[55],$KR3,0);
505         &RIP2($E,$A,$B,$C,$D,$wr[56],$wr[57],$sr[56],$KR3,0);
506         &RIP2($D,$E,$A,$B,$C,$wr[57],$wr[58],$sr[57],$KR3,0);
507         &RIP2($C,$D,$E,$A,$B,$wr[58],$wr[59],$sr[58],$KR3,0);
508         &RIP2($B,$C,$D,$E,$A,$wr[59],$wr[60],$sr[59],$KR3,0);
509         &RIP2($A,$B,$C,$D,$E,$wr[60],$wr[61],$sr[60],$KR3,0);
510         &RIP2($E,$A,$B,$C,$D,$wr[61],$wr[62],$sr[61],$KR3,0);
511         &RIP2($D,$E,$A,$B,$C,$wr[62],$wr[63],$sr[62],$KR3,0);
512         &RIP2($C,$D,$E,$A,$B,$wr[63],$wr[64],$sr[63],$KR3,2);

514         &RIP1($B,$C,$D,$E,$A,$wr[64],$sr[64],-2);
515         &RIP1($A,$B,$C,$D,$E,$wr[65],$sr[65],0);
516         &RIP1($E,$A,$B,$C,$D,$wr[66],$sr[66],0);
517         &RIP1($D,$E,$A,$B,$C,$wr[67],$sr[67],0);
518         &RIP1($C,$D,$E,$A,$B,$wr[68],$sr[68],0);
519         &RIP1($B,$C,$D,$E,$A,$wr[69],$sr[69],0);
520         &RIP1($A,$B,$C,$D,$E,$wr[70],$sr[70],0);
521         &RIP1($E,$A,$B,$C,$D,$wr[71],$sr[71],0);
522         &RIP1($D,$E,$A,$B,$C,$wr[72],$sr[72],0);
523         &RIP1($C,$D,$E,$A,$B,$wr[73],$sr[73],0);
```

```
524         &RIP1($B,$C,$D,$E,$A,$wr[74],$sr[74],0);
525         &RIP1($A,$B,$C,$D,$E,$wr[75],$sr[75],0);
526         &RIP1($E,$A,$B,$C,$D,$wr[76],$sr[76],0);
527         &RIP1($D,$E,$A,$B,$C,$wr[77],$sr[77],0);
528         &RIP1($C,$D,$E,$A,$B,$wr[78],$sr[78],0);
529         &RIP1($B,$C,$D,$E,$A,$wr[79],$sr[79],2);

531         # &mov($tmp2,   &wparam(0)); # Moved into last round

533          &mov($tmp1,    &DWP( 4,$tmp2,"",0));   # ctx->B
534         &add($D,        $tmp1);
535          &mov($tmp1,    &swtmp(16+2));          # $c
536         &add($D,        $tmp1);

538          &mov($tmp1,    &DWP( 8,$tmp2,"",0));   # ctx->C
539         &add($E,        $tmp1);
540          &mov($tmp1,    &swtmp(16+3));          # $d
541         &add($E,        $tmp1);

543          &mov($tmp1,    &DWP(12,$tmp2,"",0));   # ctx->D
544         &add($A,        $tmp1);
545          &mov($tmp1,    &swtmp(16+4));          # $e
546         &add($A,        $tmp1);


549          &mov($tmp1,    &DWP(16,$tmp2,"",0));   # ctx->E
550         &add($B,        $tmp1);
551          &mov($tmp1,    &swtmp(16+0));          # $a
552         &add($B,        $tmp1);

554          &mov($tmp1,    &DWP( 0,$tmp2,"",0));   # ctx->A
555         &add($C,        $tmp1);
556          &mov($tmp1,    &swtmp(16+1));          # $b
557         &add($C,        $tmp1);

559          &mov($tmp1,    &wparam(2));

561         &mov(&DWP( 0,$tmp2,"",0),       $D);
562          &mov(&DWP( 4,$tmp2,"",0),      $E);
563         &mov(&DWP( 8,$tmp2,"",0),       $A);
564          &sub($tmp1,1);
565         &mov(&DWP(12,$tmp2,"",0),       $B);
566          &mov(&DWP(16,$tmp2,"",0),      $C);

568         &jle(&label("get_out"));

570         &mov(&wparam(2),$tmp1);
571          &mov($C,       $A);
572         &mov($tmp1,     &wparam(1));
573          &mov($A,       $D);
574         &add($tmp1,     64);
575          &mov($B,       $E);
576         &mov(&wparam(1),$tmp1);

578         &jmp(&label("start"));

580         &set_label("get_out");

582         &stack_pop(16+5+6);

584         &pop("ebx");
585         &pop("ebp");
586         &pop("edi");
587         &pop("esi");
588         &ret();
589         &function_end_B($name);
```



```
590         }
591 #endif /* ! codereview */
```

```
**********************************************************
   37088 Wed Aug 13 19:53:09 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/sha1-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# ====================================================================
# [Re]written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================

# "[Re]written" was achieved in two major overhauls. In 2004 BODY_*
# functions were re-implemented to address P4 performance issue [see
# commentary below], and in 2006 the rest was rewritten in order to
# gain freedom to liberate licensing terms.

# January, September 2004.
#
# It was noted that Intel IA-32 C compiler generates code which
# performs ~30% *faster* on P4 CPU than original *hand-coded*
# SHA1 assembler implementation. To address this problem (and
# prove that humans are still better than machines:-), the
# original code was overhauled, which resulted in following
# performance changes:
#
#               compared with original  compared with Intel cc
#               assembler impl.         generated code
# Pentium       -16%                    +48%
# PIII/AMD      +8%                     +16%
# P4            +85%(!)                 +45%
#
# As you can see Pentium came out as looser:-( Yet I reckoned that
# improvement on P4 outweights the loss and incorporate this
# re-tuned code to 0.9.7 and later.
# ----------------------------------------------------------------
#                                       <appro@fy.chalmers.se>

# August 2009.
#
# George Spelvin has tipped that F_40_59(b,c,d) can be rewritten as
# '(c&d) + (b&(c^d))', which allows to accumulate partial results
# and lighten "pressure" on scratch registers. This resulted in
# >12% performance improvement on contemporary AMD cores (with no
# degradation on other CPUs:-). Also, the code was revised to maximize
# "distance" between instructions producing input to 'lea' instruction
# and the 'lea' instruction itself, which is essential for Intel Atom
# core and resulted in ~15% improvement.

# October 2010.
#
# Add SSSE3, Supplemental[!] SSE3, implementation. The idea behind it
# is to offload message schedule denoted by Wt in NIST specification,
# or Xupdate in OpenSSL source, to SIMD unit. The idea is not novel,
# and in SSE2 context was first explored by Dean Gaudet in 2004, see
# http://arctic.org/~dean/crypto/sha1.html. Since then several things
# have changed that made it interesting again:
#
# a) XMM units became faster and wider;
# b) instruction set became more versatile;
# c) an important observation was made by Max Locktykhin, which made
#    it possible to reduce amount of instructions required to perform
#    the operation in question, for further details see
#    http://software.intel.com/en-us/articles/improving-the-performance-of-the-s
```



```
# April 2011.
#
# Add AVX code path, probably most controversial... The thing is that
# switch to AVX alone improves performance by as little as 4% in
# comparison to SSSE3 code path. But below result doesn't look like
# 4% improvement... Trouble is that Sandy Bridge decodes 'ro[rl]' as
# pair of µ-ops, and it's the additional µ-ops, two per round, that
# make it run slower than Core2 and Westmere. But 'sh[rl]d' is decoded
# as single µ-op by Sandy Bridge and it's replacing 'ro[rl]' with
# equivalent 'sh[rl]d' that is responsible for the impressive 5.1
# cycles per processed byte. But 'sh[rl]d' is not something that used
# to be fast, nor does it appear to be fast in upcoming Bulldozer
# [according to its optimization manual]. Which is why AVX code path
# is guarded by *both* AVX and synthetic bit denoting Intel CPUs.
# One can argue that it's unfair to AMD, but without 'sh[rl]d' it
# makes no sense to keep the AVX code path. If somebody feels that
# strongly, it's probably more appropriate to discuss possibility of
# using vector rotate XOP on AMD...

######################################################################
# Current performance is summarized in following table. Numbers are
# CPU clock cycles spent to process single byte (less is better).
#
#               x86             SSSE3           AVX
# Pentium       15.7            -
# PIII          11.5            -
# P4            10.6            -
# AMD K8        7.1             -
# Core2         7.3             6.1/+20%        -
# Atom          12.5            9.5(*)/+32%     -
# Westmere      7.3             5.6/+30%        -
# Sandy Bridge  8.8             6.2/+40%        5.1(**)/+70%
#
# (*)   Loop is 1056 instructions long and expected result is ~8.25.
#       It remains mystery [to me] why ILP is limited to 1.7.
#
# (**)  As per above comment, the result is for AVX *plus* sh[rl]d.

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],"sha1-586.pl",$ARGV[$#ARGV] eq "386");

$xmm=$ymm=0;
for (@ARGV) { $xmm=1 if (/-DOPENSSL_IA32_SSE2/); }

$ymm=1 if ($xmm &&
               `$ENV{CC} -Wa,-v -c -o /dev/null -x assembler /dev/null 2>&1`
                       =~ /GNU assembler version ([2-9]\.[0-9]+)/ &&
               $1>=2.19);      # first version supporting AVX

$ymm=1 if ($xmm && !$ymm && $ARGV[0] eq "win32n" &&
               `nasm -v 2>&1` =~ /NASM version ([2-9]\.[0-9]+)/ &&
               $1>=2.03);      # first version supporting AVX

&external_label("OPENSSL_ia32cap_P") if ($xmm);


$A="eax";
$B="ebx";
$C="ecx";
$D="edx";
$E="edi";
$T="esi";
```

```
128 $tmp1="ebp";
129
130 @V=($A,$B,$C,$D,$E,$T);
131
132 $alt=0; # 1 denotes alternative IALU implementation, which performs
133         # 8% *worse* on P4, same on Westmere and Atom, 2% better on
134         # Sandy Bridge...
135
136 sub BODY_00_15
137         {
138         local($n,$a,$b,$c,$d,$e,$f)=@_;
139
140         &comment("00_15 $n");
141
142         &mov($f,$c);                    # f to hold F_00_19(b,c,d)
143          if ($n==0)  { &mov($tmp1,$a); }
144          else        { &mov($a,$tmp1); }
145         &rotl($tmp1,5);                 # tmp1=ROTATE(a,5)
146          &xor($f,$d);
147         &add($tmp1,$e);                 # tmp1+=e;
148          &mov($e,&swtmp($n%16));        # e becomes volatile and is loaded
149                                         # with xi, also note that e becomes
150                                         # f in next round...
151         &and($f,$b);
152         &rotr($b,2);                    # b=ROTATE(b,30)
153          &xor($f,$d);                   # f holds F_00_19(b,c,d)
154         &lea($tmp1,&DWP(0x5a827999,$tmp1,$e));  # tmp1+=K_00_19+xi
155
156         if ($n==15) { &mov($e,&swtmp(($n+1)%16));# pre-fetch f for next round
157                       &add($f,$tmp1); } # f+=tmp1
158         else        { &add($tmp1,$f); } # f becomes a in next round
159         &mov($tmp1,$a)                  if ($alt && $n==15);
160         }
161
162 sub BODY_16_19
163         {
164         local($n,$a,$b,$c,$d,$e,$f)=@_;
165
166         &comment("16_19 $n");
167
168 if ($alt) {
169         &xor($c,$d);
170          &xor($f,&swtmp(($n+2)%16));    # f to hold Xupdate(xi,xa,xb,xc,xd)
171         &and($tmp1,$c);                 # tmp1 to hold F_00_19(b,c,d), b&=c^d
172          &xor($f,&swtmp(($n+8)%16));
173         &xor($tmp1,$d);                 # tmp1=F_00_19(b,c,d)
174          &xor($f,&swtmp(($n+13)%16));   # f holds xa^xb^xc^xd
175         &rotl($f,1);                    # f=ROTATE(f,1)
176          &add($e,$tmp1);                # e+=F_00_19(b,c,d)
177         &xor($c,$d);                    # restore $c
178          &mov($tmp1,$a);                # b in next round
179         &rotr($b,$n==16?2:7);           # b=ROTATE(b,30)
180          &mov(&swtmp($n%16),$f);        # xi=f
181         &rotl($a,5);                    # ROTATE(a,5)
182          &lea($f,&DWP(0x5a827999,$f,$e));# f+=F_00_19(b,c,d)+e
183         &mov($e,&swtmp(($n+1)%16));     # pre-fetch f for next round
184          &add($f,$a);                   # f+=ROTATE(a,5)
185 } else {
186         &mov($tmp1,$c);                 # tmp1 to hold F_00_19(b,c,d)
187          &xor($f,&swtmp(($n+2)%16));    # f to hold Xupdate(xi,xa,xb,xc,xd)
188         &xor($tmp1,$d);
189          &xor($f,&swtmp(($n+8)%16));
190         &and($tmp1,$b);
191          &xor($f,&swtmp(($n+13)%16));   # f holds xa^xb^xc^xd
192         &rotl($f,1);                    # f=ROTATE(f,1)
193          &xor($tmp1,$d);                # tmp1=F_00_19(b,c,d)
```



```
194         &add($e,$tmp1);                 # e+=F_00_19(b,c,d)
195          &mov($tmp1,$a);
196         &rotr($b,2);                    # b=ROTATE(b,30)
197          &mov(&swtmp($n%16),$f);        # xi=f
198         &rotl($tmp1,5);                 # ROTATE(a,5)
199          &lea($f,&DWP(0x5a827999,$f,$e));# f+=F_00_19(b,c,d)+e
200         &mov($e,&swtmp(($n+1)%16));     # pre-fetch f for next round
201          &add($f,$tmp1);                # f+=ROTATE(a,5)
202 }
203         }
204
205 sub BODY_20_39
206         {
207         local($n,$a,$b,$c,$d,$e,$f)=@_;
208         local $K=($n<40)?0x6ed9eba1:0xca62c1d6;
209
210         &comment("20_39 $n");
211
212 if ($alt) {
213         &xor($tmp1,$c);                 # tmp1 to hold F_20_39(b,c,d), b^=c
214          &xor($f,&swtmp(($n+2)%16));    # f to hold Xupdate(xi,xa,xb,xc,xd)
215         &xor($tmp1,$d);                 # tmp1 holds F_20_39(b,c,d)
216          &xor($f,&swtmp(($n+8)%16));
217         &add($e,$tmp1);                 # e+=F_20_39(b,c,d)
218          &xor($f,&swtmp(($n+13)%16));   # f holds xa^xb^xc^xd
219         &rotl($f,1);                    # f=ROTATE(f,1)
220          &mov($tmp1,$a);                # b in next round
221         &rotr($b,7);                    # b=ROTATE(b,30)
222          &mov(&swtmp($n%16),$f)         if($n<77);# xi=f
223         &rotl($a,5);                    # ROTATE(a,5)
224          &xor($b,$c)                    if($n==39);# warm up for BODY_40_59
225         &and($tmp1,$b)                  if($n==39);
226          &lea($f,&DWP($K,$f,$e));       # f+=e+K_XX_YY
227         &mov($e,&swtmp(($n+1)%16))      if($n<79);# pre-fetch f for next round
228          &add($f,$a);                   # f+=ROTATE(a,5)
229         &rotr($a,5)                     if ($n==79);
230 } else {
231         &mov($tmp1,$b);                 # tmp1 to hold F_20_39(b,c,d)
232          &xor($f,&swtmp(($n+2)%16));    # f to hold Xupdate(xi,xa,xb,xc,xd)
233         &xor($tmp1,$c);
234          &xor($f,&swtmp(($n+8)%16));
235         &xor($tmp1,$d);                 # tmp1 holds F_20_39(b,c,d)
236          &xor($f,&swtmp(($n+13)%16));   # f holds xa^xb^xc^xd
237         &rotl($f,1);                    # f=ROTATE(f,1)
238          &add($e,$tmp1);                # e+=F_20_39(b,c,d)
239         &rotr($b,2);                    # b=ROTATE(b,30)
240          &mov($tmp1,$a);
241         &rotl($tmp1,5);                 # ROTATE(a,5)
242          &mov(&swtmp($n%16),$f) if($n<77);# xi=f
243         &lea($f,&DWP($K,$f,$e));        # f+=e+K_XX_YY
244          &mov($e,&swtmp(($n+1)%16)) if($n<79);# pre-fetch f for next round
245         &add($f,$tmp1);                 # f+=ROTATE(a,5)
246 }
247         }
248
249 sub BODY_40_59
250         {
251         local($n,$a,$b,$c,$d,$e,$f)=@_;
252
253         &comment("40_59 $n");
254
255 if ($alt) {
256         &add($e,$tmp1);                 # e+=b&(c^d)
257          &xor($f,&swtmp(($n+2)%16));    # f to hold Xupdate(xi,xa,xb,xc,xd)
258         &mov($tmp1,$d);
259          &xor($f,&swtmp(($n+8)%16));
```

```
260         &xor($c,$d);                    # restore $c
261          &xor($f,&swtmp(($n+13)%16));   # f holds xa^xb^xc^xd
262         &rotl($f,1);                    # f=ROTATE(f,1)
263          &and($tmp1,$c);
264         &rotr($b,7);                    # b=ROTATE(b,30)
265          &add($e,$tmp1);                # e+=c&d
266         &mov($tmp1,$a);                 # b in next round
267          &mov(&swtmp($n%16),$f);        # xi=f
268         &rotl($a,5);                    # ROTATE(a,5)
269          &xor($b,$c)                    if ($n<59);
270         &and($tmp1,$b)                  if ($n<59);# tmp1 to hold F_40_59(b,c,d)
271          &lea($f,&DWP(0x8f1bbcdc,$f,$e));# f+=K_40_59+e+(b&(c^d))
272         &mov($e,&swtmp(($n+1)%16));     # pre-fetch f for next round
273          &add($f,$a);                   # f+=ROTATE(a,5)
274 } else {
275         &mov($tmp1,$c);                 # tmp1 to hold F_40_59(b,c,d)
276          &xor($f,&swtmp(($n+2)%16));    # f to hold Xupdate(xi,xa,xb,xc,xd)
277         &xor($tmp1,$d);
278          &xor($f,&swtmp(($n+8)%16));
279         &and($tmp1,$b);
280          &xor($f,&swtmp(($n+13)%16));   # f holds xa^xb^xc^xd
281         &rotl($f,1);                    # f=ROTATE(f,1)
282          &add($tmp1,$e);                # b&(c^d)+=e
283         &rotr($b,2);                    # b=ROTATE(b,30)
284          &mov($e,$a);                   # e becomes volatile
285         &rotl($e,5);                    # ROTATE(a,5)
286          &mov(&swtmp($n%16),$f);        # xi=f
287         &lea($f,&DWP(0x8f1bbcdc,$f,$tmp1));# f+=K_40_59+e+(b&(c^d))
288          &mov($tmp1,$c);
289         &add($f,$e);                    # f+=ROTATE(a,5)
290          &and($tmp1,$d);
291         &mov($e,&swtmp(($n+1)%16));     # pre-fetch f for next round
292          &add($f,$tmp1);                # f+=c&d
293 }
294         }

296 &function_begin("sha1_block_data_order");
297 if ($xmm) {
298   &static_label("ssse3_shortcut");
299   &static_label("avx_shortcut")         if ($ymm);
300   &static_label("K_XX_XX");

302         &call   (&label("pic_point"));  # make it PIC!
303   &set_label("pic_point");
304         &blindpop($tmp1);
305         &picmeup($T,"OPENSSL_ia32cap_P",$tmp1,&label("pic_point"));
306         &lea    ($tmp1,&DWP(&label("K_XX_XX")."-".&label("pic_point"),$tmp1));

308         &mov    ($A,&DWP(0,$T));
309         &mov    ($D,&DWP(4,$T));
310         &test   ($D,1<<9);              # check SSSE3 bit
311         &jz     (&label("x86"));
312         &test   ($A,1<<24);             # check FXSR bit
313         &jz     (&label("x86"));
314         if ($ymm) {
315                 &and    ($D,1<<28);             # mask AVX bit
316                 &and    ($A,1<<30);             # mask "Intel CPU" bit
317                 &or     ($A,$D);
318                 &cmp    ($A,1<<28|1<<30);
319                 &je     (&label("avx_shortcut"));
320         }
321         &jmp    (&label("ssse3_shortcut"));
322   &set_label("x86",16);
323 }
324         &mov($tmp1,&wparam(0)); # SHA_CTX *c
325         &mov($T,&wparam(1));    # const void *input
```



```
326         &mov($A,&wparam(2));    # size_t num
327         &stack_push(16+3);      # allocate X[16]
328         &shl($A,6);
329         &add($A,$T);
330         &mov(&wparam(2),$A);    # pointer beyond the end of input
331         &mov($E,&DWP(16,$tmp1));# pre-load E
332         &jmp(&label("loop"));

334 &set_label("loop",16);

336         # copy input chunk to X, but reversing byte order!
337         for ($i=0; $i<16; $i+=4)
338                 {
339                 &mov($A,&DWP(4*($i+0),$T));
340                 &mov($B,&DWP(4*($i+1),$T));
341                 &mov($C,&DWP(4*($i+2),$T));
342                 &mov($D,&DWP(4*($i+3),$T));
343                 &bswap($A);
344                 &bswap($B);
345                 &bswap($C);
346                 &bswap($D);
347                 &mov(&swtmp($i+0),$A);
348                 &mov(&swtmp($i+1),$B);
349                 &mov(&swtmp($i+2),$C);
350                 &mov(&swtmp($i+3),$D);
351                 }
352         &mov(&wparam(1),$T);    # redundant in 1st spin

354         &mov($A,&DWP(0,$tmp1)); # load SHA_CTX
355         &mov($B,&DWP(4,$tmp1));
356         &mov($C,&DWP(8,$tmp1));
357         &mov($D,&DWP(12,$tmp1));
358         # E is pre-loaded

360         for($i=0;$i<16;$i++)    { &BODY_00_15($i,@V); unshift(@V,pop(@V)); }
361         for(;$i<20;$i++)        { &BODY_16_19($i,@V); unshift(@V,pop(@V)); }
362         for(;$i<40;$i++)        { &BODY_20_39($i,@V); unshift(@V,pop(@V)); }
363         for(;$i<60;$i++)        { &BODY_40_59($i,@V); unshift(@V,pop(@V)); }
364         for(;$i<80;$i++)        { &BODY_20_39($i,@V); unshift(@V,pop(@V)); }

366         (($V[5] eq $D) and ($V[0] eq $E)) or die;       # double-check

368         &mov($tmp1,&wparam(0)); # re-load SHA_CTX*
369         &mov($D,&wparam(1));    # D is last "T" and is discarded

371         &add($E,&DWP(0,$tmp1)); # E is last "A"...
372         &add($T,&DWP(4,$tmp1));
373         &add($A,&DWP(8,$tmp1));
374         &add($B,&DWP(12,$tmp1));
375         &add($C,&DWP(16,$tmp1));

377         &mov(&DWP(0,$tmp1),$E); # update SHA_CTX
378          &add($D,64);           # advance input pointer
379         &mov(&DWP(4,$tmp1),$T);
380          &cmp($D,&wparam(2));   # have we reached the end yet?
381         &mov(&DWP(8,$tmp1),$A);
382          &mov($E,$C);           # C is last "E" which needs to be "pre-loaded"
383         &mov(&DWP(12,$tmp1),$B);
384          &mov($T,$D);           # input pointer
385         &mov(&DWP(16,$tmp1),$C);
386         &jb(&label("loop"));

388         &stack_pop(16+3);
389 &function_end("sha1_block_data_order");

391 if ($xmm) {
```

```
392 ######################################################################
393 # The SSSE3 implementation.
394 #
395 # %xmm[0-7] are used as ring @X[] buffer containing quadruples of last
396 # 32 elements of the message schedule or Xupdate outputs. First 4
397 # quadruples are simply byte-swapped input, next 4 are calculated
398 # according to method originally suggested by Dean Gaudet (modulo
399 # being implemented in SSSE3). Once 8 quadruples or 32 elements are
400 # collected, it switches to routine proposed by Max Locktyukhin.
401 #
402 # Calculations inevitably require temporary registers, and there are
403 # no %xmm registers left to spare. For this reason part of the ring
404 # buffer, X[2..4] to be specific, is offloaded to 3 quadriples ring
405 # buffer on the stack. Keep in mind that X[2] is alias X[-6], X[3] -
406 # X[-5], and X[4] - X[-4]...
407 #
408 # Another notable optimization is aggressive stack frame compression
409 # aiming to minimize amount of 9-byte instructions...
410 #
411 # Yet another notable optimization is "jumping" $B variable. It means
412 # that there is no register permanently allocated for $B value. This
413 # allowed to eliminate one instruction from body_20_39...
414 #
415 my $Xi=4;                       # 4xSIMD Xupdate round, start pre-seeded
416 my @X=map("xmm$_",(4..7,0..3)); # pre-seeded for $Xi=4
417 my @V=($A,$B,$C,$D,$E);
418 my $j=0;                        # hash round
419 my @T=($T,$tmp1);
420 my $inp;

422 my $_rol=sub { &rol(@_) };
423 my $_ror=sub { &ror(@_) };

425 &function_begin("_sha1_block_data_order_ssse3");
426         &call   (&label("pic_point"));  # make it PIC!
427         &set_label("pic_point");
428         &blindpop($tmp1);
429         &lea    ($tmp1,&DWP(&label("K_XX_XX")."-".&label("pic_point"),$tmp1));
430 &set_label("ssse3_shortcut");

432         &movdqa (@X[3],&QWP(0,$tmp1));          # K_00_19
433         &movdqa (@X[4],&QWP(16,$tmp1));         # K_20_39
434         &movdqa (@X[5],&QWP(32,$tmp1));         # K_40_59
435         &movdqa (@X[6],&QWP(48,$tmp1));         # K_60_79
436         &movdqa (@X[2],&QWP(64,$tmp1));         # pbswap mask

438         &mov    ($E,&wparam(0));                # load argument block
439         &mov    ($inp=@T[1],&wparam(1));
440         &mov    ($D,&wparam(2));
441         &mov    (@T[0],"esp");

443         # stack frame layout
444         #
445         # +0    X[0]+K  X[1]+K  X[2]+K  X[3]+K  # XMM->IALU xfer area
446         #       X[4]+K  X[5]+K  X[6]+K  X[7]+K
447         #       X[8]+K  X[9]+K  X[10]+K X[11]+K
448         #       X[12]+K X[13]+K X[14]+K X[15]+K
449         #
450         # +64   X[0]    X[1]    X[2]    X[3]    # XMM->XMM backtrace area
451         #       X[4]    X[5]    X[6]    X[7]
452         #       X[8]    X[9]    X[10]   X[11]   # even borrowed for K_00_19
453         #
454         # +112  K_20_39 K_20_39 K_20_39 K_20_39 # constants
455         #       K_40_59 K_40_59 K_40_59 K_40_59
456         #       K_60_79 K_60_79 K_60_79 K_60_79
457         #       K_00_19 K_00_19 K_00_19 K_00_19
```



```
458         #       pbswap mask
459         #
460         # +192  ctx                             # argument block
461         # +196  inp
462         # +200  end
463         # +204  esp
464         &sub    ("esp",208);
465         &and    ("esp",-64);

467         &movdqa (&QWP(112+0,"esp"),@X[4]);      # copy constants
468         &movdqa (&QWP(112+16,"esp"),@X[5]);
469         &movdqa (&QWP(112+32,"esp"),@X[6]);
470         &shl    ($D,6);                         # len*64
471         &movdqa (&QWP(112+48,"esp"),@X[3]);
472         &add    ($D,$inp);                      # end of input
473         &movdqa (&QWP(112+64,"esp"),@X[2]);
474         &add    ($inp,64);
475         &mov    (&DWP(192+0,"esp"),$E);         # save argument block
476         &mov    (&DWP(192+4,"esp"),$inp);
477         &mov    (&DWP(192+8,"esp"),$D);
478         &mov    (&DWP(192+12,"esp"),@T[0]);     # save original %esp

480         &mov    ($A,&DWP(0,$E));                # load context
481         &mov    ($B,&DWP(4,$E));
482         &mov    ($C,&DWP(8,$E));
483         &mov    ($D,&DWP(12,$E));
484         &mov    ($E,&DWP(16,$E));
485         &mov    (@T[0],$B);                     # magic seed

487         &movdqu (@X[-4&7],&QWP(-64,$inp));      # load input to %xmm[0-3]
488         &movdqu (@X[-3&7],&QWP(-48,$inp));
489         &movdqu (@X[-2&7],&QWP(-32,$inp));
490         &movdqu (@X[-1&7],&QWP(-16,$inp));
491         &pshufb (@X[-4&7],@X[2]);               # byte swap
492         &pshufb (@X[-3&7],@X[2]);
493         &pshufb (@X[-2&7],@X[2]);
494         &movdqa (&QWP(112-16,"esp"),@X[3]);     # borrow last backtrace slot
495         &pshufb (@X[-1&7],@X[2]);
496         &paddd  (@X[-4&7],@X[3]);               # add K_00_19
497         &paddd  (@X[-3&7],@X[3]);
498         &paddd  (@X[-2&7],@X[3]);
499         &movdqa (&QWP(0,"esp"),@X[-4&7]);       # X[]+K xfer to IALU
500         &psubd  (@X[-4&7],@X[3]);               # restore X[]
501         &movdqa (&QWP(0+16,"esp"),@X[-3&7]);
502         &psubd  (@X[-3&7],@X[3]);
503         &movdqa (&QWP(0+32,"esp"),@X[-2&7]);
504         &psubd  (@X[-2&7],@X[3]);
505         &movdqa (@X[0],@X[-3&7]);
506         &jmp    (&label("loop"));

508 ######################################################################
509 # SSE instruction sequence is first broken to groups of indepentent
510 # instructions, independent in respect to their inputs and shifter
511 # (not all architectures have more than one). Then IALU instructions
512 # are "knitted in" between the SSE groups. Distance is maintained for
513 # SSE latency of 2 in hope that it fits better upcoming AMD Bulldozer
514 # [which allegedly also implements SSSE3]...
515 #
516 # Temporary registers usage. X[2] is volatile at the entry and at the
517 # end is restored from backtrace ring buffer. X[3] is expected to
518 # contain current K_XX_XX constant and is used to caclulate X[-1]+K
519 # from previous round, it becomes volatile the moment the value is
520 # saved to stack for transfer to IALU. X[4] becomes volatile whenever
521 # X[-4] is accumulated and offloaded to backtrace ring buffer, at the
522 # end it is loaded with next K_XX_XX [which becomes X[3] in next
523 # round]...
```

```
524 #
525 sub Xupdate_ssse3_16_31()               # recall that $Xi starts wtih 4
526 { use integer;
527   my $body = shift;
528   my @insns = (&$body,&$body,&$body,&$body);    # 40 instructions
529   my ($a,$b,$c,$d,$e);

531         eval(shift(@insns));
532         eval(shift(@insns));
533        &palignr(@X[0],@X[-4&7],8);     # compose "X[-14]" in "X[0]"
534        &movdqa (@X[2],@X[-1&7]);
535         eval(shift(@insns));
536         eval(shift(@insns));

538          &paddd        (@X[3],@X[-1&7]);
539          &movdqa       (&QWP(64+16*(($Xi-4)%3),"esp"),@X[-4&7]);# save X[] to b
540         eval(shift(@insns));
541         eval(shift(@insns));
542        &psrldq (@X[2],4);              # "X[-3]", 3 dwords
543         eval(shift(@insns));
544         eval(shift(@insns));
545        &pxor   (@X[0],@X[-4&7]);       # "X[0]"^="X[-16]"
546         eval(shift(@insns));
547         eval(shift(@insns));

549        &pxor   (@X[2],@X[-2&7]);       # "X[-3]"^"X[-8]"
550         eval(shift(@insns));
551         eval(shift(@insns));
552         eval(shift(@insns));
553         eval(shift(@insns));

555        &pxor   (@X[0],@X[2]);          # "X[0]"^="X[-3]"^"X[-8]"
556         eval(shift(@insns));
557         eval(shift(@insns));
558          &movdqa       (&QWP(0+16*(($Xi-1)&3),"esp"),@X[3]);   # X[]+K xfer to
559         eval(shift(@insns));
560         eval(shift(@insns));

562        &movdqa (@X[4],@X[0]);
563        &movdqa (@X[2],@X[0]);
564         eval(shift(@insns));
565         eval(shift(@insns));
566         eval(shift(@insns));
567         eval(shift(@insns));

569        &pslldq (@X[4],12);             # "X[0]"<<96, extract one dword
570        &paddd  (@X[0],@X[0]);
571         eval(shift(@insns));
572         eval(shift(@insns));
573         eval(shift(@insns));
574         eval(shift(@insns));

576        &psrld  (@X[2],31);
577         eval(shift(@insns));
578         eval(shift(@insns));
579        &movdqa (@X[3],@X[4]);
580         eval(shift(@insns));
581         eval(shift(@insns));

583        &psrld  (@X[4],30);
584        &por    (@X[0],@X[2]);          # "X[0]"<<<=1
585         eval(shift(@insns));
586         eval(shift(@insns));
587          &movdqa       (@X[2],&QWP(64+16*(($Xi-6)%3),"esp")) if ($Xi>5);
588         eval(shift(@insns));
589         eval(shift(@insns));
```



```
591         &pslld  (@X[3],2);
592         &pxor   (@X[0],@X[4]);
593          eval(shift(@insns));
594          eval(shift(@insns));
595           &movdqa       (@X[4],&QWP(112-16+16*(($Xi)/5),"esp"));        # K_XX_X
596          eval(shift(@insns));
597          eval(shift(@insns));

599         &pxor   (@X[0],@X[3]);          # "X[0]"^=("X[0]"<<96)<<<2
600           &movdqa       (@X[1],@X[-2&7])        if ($Xi<7);
601          eval(shift(@insns));
602          eval(shift(@insns));

604          foreach (@insns) { eval; }     # remaining instructions [if any]

606   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
607 }

609 sub Xupdate_ssse3_32_79()
610 { use integer;
611   my $body = shift;
612   my @insns = (&$body,&$body,&$body,&$body);    # 32 to 48 instructions
613   my ($a,$b,$c,$d,$e);

615         &movdqa (@X[2],@X[-1&7])        if ($Xi==8);
616          eval(shift(@insns));           # body_20_39
617         &pxor   (@X[0],@X[-4&7]);       # "X[0]"="X[-32]"^"X[-16]"
618         &palignr(@X[2],@X[-2&7],8);     # compose "X[-6]"
619          eval(shift(@insns));
620          eval(shift(@insns));
621          eval(shift(@insns));           # rol

623         &pxor   (@X[0],@X[-7&7]);       # "X[0]"^="X[-28]"
624           &movdqa       (&QWP(64+16*(($Xi-4)%3),"esp"),@X[-4&7]);       # save X
625          eval(shift(@insns));
626          eval(shift(@insns));
627          if ($Xi%5) {
628           &movdqa       (@X[4],@X[3]);  # "perpetuate" K_XX_XX...
629          } else {                       # ... or load next one
630           &movdqa       (@X[4],&QWP(112-16+16*($Xi/5),"esp"));
631          }
632           &paddd        (@X[3],@X[-1&7]);
633          eval(shift(@insns));           # ror
634          eval(shift(@insns));

636         &pxor   (@X[0],@X[2]);          # "X[0]"^="X[-6]"
637          eval(shift(@insns));           # body_20_39
638          eval(shift(@insns));
639          eval(shift(@insns));
640          eval(shift(@insns));           # rol

642         &movdqa (@X[2],@X[0]);
643           &movdqa       (&QWP(0+16*(($Xi-1)&3),"esp"),@X[3]);   # X[]+K xfer to
644          eval(shift(@insns));
645          eval(shift(@insns));
646          eval(shift(@insns));           # ror
647          eval(shift(@insns));

649         &pslld  (@X[0],2);
650          eval(shift(@insns));           # body_20_39
651          eval(shift(@insns));
652         &psrld  (@X[2],30);
653          eval(shift(@insns));
654          eval(shift(@insns));           # rol
655          eval(shift(@insns));
```

```
656         eval(shift(@insns));
657         eval(shift(@insns));          # ror
658         eval(shift(@insns));

660        &por    (@X[0],@X[2]);         # "X[0]"<<<=2
661         eval(shift(@insns));          # body_20_39
662         eval(shift(@insns));
663          &movdqa       (@X[2],&QWP(64+16*(($Xi-6)%3),"esp")) if($Xi<19);
664         eval(shift(@insns));
665         eval(shift(@insns));          # rol
666         eval(shift(@insns));
667         eval(shift(@insns));
668         eval(shift(@insns));          # ror
669          &movdqa       (@X[3],@X[0])   if ($Xi<19);
670         eval(shift(@insns));

672         foreach (@insns) { eval; }     # remaining instructions

674   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
675 }

677 sub Xuplast_ssse3_80()
678 { use integer;
679   my $body = shift;
680   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
681   my ($a,$b,$c,$d,$e);

683         eval(shift(@insns));
684          &paddd        (@X[3],@X[-1&7]);
685         eval(shift(@insns));
686         eval(shift(@insns));
687         eval(shift(@insns));
688         eval(shift(@insns));

690          &movdqa       (&QWP(0+16*(($Xi-1)&3),"esp"),@X[3]);   # X[]+K xfer IAL

692         foreach (@insns) { eval; }             # remaining instructions

694        &mov    ($inp=@T[1],&DWP(192+4,"esp"));
695        &cmp    ($inp,&DWP(192+8,"esp"));
696        &je     (&label("done"));

698        &movdqa (@X[3],&QWP(112+48,"esp"));     # K_00_19
699        &movdqa (@X[2],&QWP(112+64,"esp"));     # pbswap mask
700        &movdqu (@X[-4&7],&QWP(0,$inp));        # load input
701        &movdqu (@X[-3&7],&QWP(16,$inp));
702        &movdqu (@X[-2&7],&QWP(32,$inp));
703        &movdqu (@X[-1&7],&QWP(48,$inp));
704        &add    ($inp,64);
705        &pshufb (@X[-4&7],@X[2]);               # byte swap
706        &mov    (&DWP(192+4,"esp"),$inp);
707        &movdqa (&QWP(112-16,"esp"),@X[3]);     # borrow last backtrace slot

709   $Xi=0;
710 }

712 sub Xloop_ssse3()
713 { use integer;
714   my $body = shift;
715   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
716   my ($a,$b,$c,$d,$e);

718         eval(shift(@insns));
719         eval(shift(@insns));
720        &pshufb (@X[($Xi-3)&7],@X[2]);
721         eval(shift(@insns));
```



```
722         eval(shift(@insns));
723        &paddd  (@X[($Xi-4)&7],@X[3]);
724         eval(shift(@insns));
725         eval(shift(@insns));
726         eval(shift(@insns));
727         eval(shift(@insns));
728        &movdqa (&QWP(0+16*$Xi,"esp"),@X[($Xi-4)&7]);   # X[]+K xfer to IALU
729         eval(shift(@insns));
730         eval(shift(@insns));
731        &psubd  (@X[($Xi-4)&7],@X[3]);

733        foreach (@insns) { eval; }
734   $Xi++;
735 }

737 sub Xtail_ssse3()
738 { use integer;
739   my $body = shift;
740   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
741   my ($a,$b,$c,$d,$e);

743        foreach (@insns) { eval; }
744 }

746 sub body_00_19 () {
747        (
748        '($a,$b,$c,$d,$e)=@V;'.
749        '&add   ($e,&DWP(4*($j&15),"esp"));',   # X[]+K xfer
750        '&xor   ($c,$d);',
751        '&mov   (@T[1],$a);',   # $b in next round
752        '&$_rol ($a,5);',
753        '&and   (@T[0],$c);',   # ($b&($c^$d))
754        '&xor   ($c,$d);',      # restore $c
755        '&xor   (@T[0],$d);',
756        '&add   ($e,$a);',
757        '&$_ror ($b,$j?7:2);',  # $b>>>2
758        '&add   ($e,@T[0]);'    .'$j++; unshift(@V,pop(@V)); unshift(@T,pop(@T))
759        );
760 }

762 sub body_20_39 () {
763        (
764        '($a,$b,$c,$d,$e)=@V;'.
765        '&add   ($e,&DWP(4*($j++&15),"esp"));', # X[]+K xfer
766        '&xor   (@T[0],$d);',   # ($b^$d)
767        '&mov   (@T[1],$a);',   # $b in next round
768        '&$_rol ($a,5);',
769        '&xor   (@T[0],$c);',   # ($b^$d^$c)
770        '&add   ($e,$a);',
771        '&$_ror ($b,7);',       # $b>>>2
772        '&add   ($e,@T[0]);'    .'unshift(@V,pop(@V)); unshift(@T,pop(@T));'
773        );
774 }

776 sub body_40_59 () {
777        (
778        '($a,$b,$c,$d,$e)=@V;'.
779        '&mov   (@T[1],$c);',
780        '&xor   ($c,$d);',
781        '&add   ($e,&DWP(4*($j++&15),"esp"));', # X[]+K xfer
782        '&and   (@T[1],$d);',
783        '&and   (@T[0],$c);',   # ($b&($c^$d))
784        '&$_ror ($b,7);',       # $b>>>2
785        '&add   ($e,@T[1]);',
786        '&mov   (@T[1],$a);',   # $b in next round
787        '&$_rol ($a,5);',
```

```
788         '&add   ($e,@T[0]);',
789         '&xor   ($c,$d);',      # restore $c
790         '&add   ($e,$a);'       .'unshift(@V,pop(@V)); unshift(@T,pop(@T));'
791         );
792 }

793 &set_label("loop",16);
794         &Xupdate_ssse3_16_31(\&body_00_19);
795         &Xupdate_ssse3_16_31(\&body_00_19);
796         &Xupdate_ssse3_16_31(\&body_00_19);
797         &Xupdate_ssse3_16_31(\&body_00_19);
798         &Xupdate_ssse3_32_79(\&body_00_19);
799         &Xupdate_ssse3_32_79(\&body_20_39);
800         &Xupdate_ssse3_32_79(\&body_20_39);
801         &Xupdate_ssse3_32_79(\&body_20_39);
802         &Xupdate_ssse3_32_79(\&body_20_39);
803         &Xupdate_ssse3_32_79(\&body_20_39);
804         &Xupdate_ssse3_32_79(\&body_40_59);
805         &Xupdate_ssse3_32_79(\&body_40_59);
806         &Xupdate_ssse3_32_79(\&body_40_59);
807         &Xupdate_ssse3_32_79(\&body_40_59);
808         &Xupdate_ssse3_32_79(\&body_40_59);
809         &Xupdate_ssse3_32_79(\&body_20_39);
810         &Xuplast_ssse3_80(\&body_20_39);        # can jump to "done"

811                                 $saved_j=$j; @saved_V=@V;

812         &Xloop_ssse3(\&body_20_39);
813         &Xloop_ssse3(\&body_20_39);
814         &Xloop_ssse3(\&body_20_39);

815         &mov    (@T[1],&DWP(192,"esp"));        # update context
816         &add    ($A,&DWP(0,@T[1]));
817         &add    (@T[0],&DWP(4,@T[1]));          # $b
818         &add    ($C,&DWP(8,@T[1]));
819         &mov    (&DWP(0,@T[1]),$A);
820         &add    ($D,&DWP(12,@T[1]));
821         &mov    (&DWP(4,@T[1]),@T[0]);
822         &add    ($E,&DWP(16,@T[1]));
823         &mov    (&DWP(8,@T[1]),$C);
824         &mov    ($B,@T[0]);
825         &mov    (&DWP(12,@T[1]),$D);
826         &mov    (&DWP(16,@T[1]),$E);
827         &movdqa (@X[0],@X[-3&7]);

828         &jmp    (&label("loop"));

829 &set_label("done",16);          $j=$saved_j; @V=@saved_V;

830         &Xtail_ssse3(\&body_20_39);
831         &Xtail_ssse3(\&body_20_39);
832         &Xtail_ssse3(\&body_20_39);

833         &mov    (@T[1],&DWP(192,"esp"));        # update context
834         &add    ($A,&DWP(0,@T[1]));
835         &mov    ("esp",&DWP(192+12,"esp"));     # restore %esp
836         &add    (@T[0],&DWP(4,@T[1]));          # $b
837         &add    ($C,&DWP(8,@T[1]));
838         &mov    (&DWP(0,@T[1]),$A);
839         &add    ($D,&DWP(12,@T[1]));
840         &mov    (&DWP(4,@T[1]),@T[0]);
841         &add    ($E,&DWP(16,@T[1]));
842         &mov    (&DWP(8,@T[1]),$C);
843         &mov    (&DWP(12,@T[1]),$D);
844         &mov    (&DWP(16,@T[1]),$E);
```



```
845 &function_end("_sha1_block_data_order_ssse3");

846 if ($ymm) {
847 my $Xi=4;                       # 4xSIMD Xupdate round, start pre-seeded
848 my @X=map("xmm$_",(4..7,0..3)); # pre-seeded for $Xi=4
849 my @V=($A,$B,$C,$D,$E);
850 my $j=0;                        # hash round
851 my @T=($T,$tmp1);
852 my $inp;

853 my $_rol=sub { &shld(@_[0],@_) };
854 my $_ror=sub { &shrd(@_[0],@_) };

855 &function_begin("_sha1_block_data_order_avx");
856         &call   (&label("pic_point"));  # make it PIC!
857         &set_label("pic_point");
858         &blindpop($tmp1);
859         &lea    ($tmp1,&DWP(&label("K_XX_XX")."-".&label("pic_point"),$tmp1));
860 &set_label("avx_shortcut");
861         &vzeroall();

862         &vmovdqa(@X[3],&QWP(0,$tmp1));          # K_00_19
863         &vmovdqa(@X[4],&QWP(16,$tmp1));         # K_20_39
864         &vmovdqa(@X[5],&QWP(32,$tmp1));         # K_40_59
865         &vmovdqa(@X[6],&QWP(48,$tmp1));         # K_60_79
866         &vmovdqa(@X[2],&QWP(64,$tmp1));         # pbswap mask

867         &mov    ($E,&wparam(0));                # load argument block
868         &mov    ($inp=@T[1],&wparam(1));
869         &mov    ($D,&wparam(2));
870         &mov    (@T[0],"esp");

871         # stack frame layout
872         #
873         # +0    X[0]+K  X[1]+K  X[2]+K  X[3]+K  # XMM->IALU xfer area
874         #       X[4]+K  X[5]+K  X[6]+K  X[7]+K
875         #       X[8]+K  X[9]+K  X[10]+K X[11]+K
876         #       X[12]+K X[13]+K X[14]+K X[15]+K
877         #
878         # +64   X[0]    X[1]    X[2]    X[3]    # XMM->XMM backtrace area
879         #       X[4]    X[5]    X[6]    X[7]
880         #       X[8]    X[9]    X[10]   X[11]   # even borrowed for K_00_19
881         #
882         # +112  K_20_39 K_20_39 K_20_39 K_20_39 # constants
883         #       K_40_59 K_40_59 K_40_59 K_40_59
884         #       K_60_79 K_60_79 K_60_79 K_60_79
885         #       K_00_19 K_00_19 K_00_19 K_00_19
886         #       pbswap mask
887         #
888         # +192  ctx                             # argument block
889         # +196  inp
890         # +200  end
891         # +204  esp
892         &sub    ("esp",208);
893         &and    ("esp",-64);

894         &vmovdqa(&QWP(112+0,"esp"),@X[4]);      # copy constants
895         &vmovdqa(&QWP(112+16,"esp"),@X[5]);
896         &vmovdqa(&QWP(112+32,"esp"),@X[6]);
897         &shl    ($D,6);                         # len*64
898         &vmovdqa(&QWP(112+48,"esp"),@X[3]);
899         &add    ($D,$inp);                      # end of input
900         &vmovdqa(&QWP(112+64,"esp"),@X[2]);
901         &add    ($inp,64);
902         &mov    (&DWP(192+0,"esp"),$E);         # save argument block
903         &mov    (&DWP(192+4,"esp"),$inp);
```

```
920         &mov    (&DWP(192+8,"esp"),$D);
921         &mov    (&DWP(192+12,"esp"),@T[0]);     # save original %esp

923         &mov    ($A,&DWP(0,$E));                # load context
924         &mov    ($B,&DWP(4,$E));
925         &mov    ($C,&DWP(8,$E));
926         &mov    ($D,&DWP(12,$E));
927         &mov    ($E,&DWP(16,$E));
928         &mov    (@T[0],$B);                     # magic seed

930         &vmovdqu(@X[-4&7],&QWP(-64,$inp));      # load input to %xmm[0-3]
931         &vmovdqu(@X[-3&7],&QWP(-48,$inp));
932         &vmovdqu(@X[-2&7],&QWP(-32,$inp));
933         &vmovdqu(@X[-1&7],&QWP(-16,$inp));
934         &vpshufb(@X[-4&7],@X[-4&7],@X[2]);      # byte swap
935         &vpshufb(@X[-3&7],@X[-3&7],@X[2]);
936         &vpshufb(@X[-2&7],@X[-2&7],@X[2]);
937         &vmovdqa(&QWP(112-16,"esp"),@X[3]);     # borrow last backtrace slot
938         &vpshufb(@X[-1&7],@X[-1&7],@X[2]);
939         &vpaddd (@X[0],@X[-4&7],@X[3]);         # add K_00_19
940         &vpaddd (@X[1],@X[-3&7],@X[3]);
941         &vpaddd (@X[2],@X[-2&7],@X[3]);
942         &vmovdqa(&QWP(0,"esp"),@X[0]);          # X[]+K xfer to IALU
943         &vmovdqa(&QWP(0+16,"esp"),@X[1]);
944         &vmovdqa(&QWP(0+32,"esp"),@X[2]);
945         &jmp    (&label("loop"));

947 sub Xupdate_avx_16_31()         # recall that $Xi starts wtih 4
948 { use integer;
949   my $body = shift;
950   my @insns = (&$body,&$body,&$body,&$body);    # 40 instructions
951   my ($a,$b,$c,$d,$e);

953         eval(shift(@insns));
954         eval(shift(@insns));
955        &vpalignr(@X[0],@X[-3&7],@X[-4&7],8);   # compose "X[-14]" in "X[0]"
956         eval(shift(@insns));
957         eval(shift(@insns));

959          &vpaddd       (@X[3],@X[3],@X[-1&7]);
960          &vmovdqa      (&QWP(64+16*(($Xi-4)%3),"esp"),@X[-4&7]);# save X[] to b
961         eval(shift(@insns));
962         eval(shift(@insns));
963        &vpsrldq(@X[2],@X[-1&7],4);             # "X[-3]", 3 dwords
964         eval(shift(@insns));
965         eval(shift(@insns));
966        &vpxor  (@X[0],@X[0],@X[-4&7]);         # "X[0]"^="X[-16]"
967         eval(shift(@insns));
968         eval(shift(@insns));

970        &vpxor  (@X[2],@X[2],@X[-2&7]);         # "X[-3]"^"X[-8]"
971         eval(shift(@insns));
972         eval(shift(@insns));
973          &vmovdqa      (&QWP(0+16*(($Xi-1)&3),"esp"),@X[3]);   # X[]+K xfer to
974         eval(shift(@insns));
975         eval(shift(@insns));

977        &vpxor  (@X[0],@X[0],@X[2]);            # "X[0]"^="X[-3]"^"X[-8]"
978         eval(shift(@insns));
979         eval(shift(@insns));
980         eval(shift(@insns));
981         eval(shift(@insns));

983        &vpsrld (@X[2],@X[0],31);
984         eval(shift(@insns));
985         eval(shift(@insns));
```



```
986         eval(shift(@insns));
987         eval(shift(@insns));

989        &vpslldq(@X[4],@X[0],12);               # "X[0]"<<96, extract one dword
990        &vpaddd (@X[0],@X[0],@X[0]);
991         eval(shift(@insns));
992         eval(shift(@insns));
993         eval(shift(@insns));
994         eval(shift(@insns));

996        &vpsrld (@X[3],@X[4],30);
997        &vpor   (@X[0],@X[0],@X[2]);            # "X[0]"<<<=1
998         eval(shift(@insns));
999         eval(shift(@insns));
1000        eval(shift(@insns));
1001        eval(shift(@insns));

1003       &vpslld (@X[4],@X[4],2);
1004         &vmovdqa      (@X[2],&QWP(64+16*(($Xi-6)%3),"esp")) if ($Xi>5);
1005        eval(shift(@insns));
1006        eval(shift(@insns));
1007       &vpxor  (@X[0],@X[0],@X[3]);
1008        eval(shift(@insns));
1009        eval(shift(@insns));
1010        eval(shift(@insns));
1011        eval(shift(@insns));

1013       &vpxor  (@X[0],@X[0],@X[4]);            # "X[0]"^=("X[0]"<<96)<<<2
1014        eval(shift(@insns));
1015        eval(shift(@insns));
1016         &vmovdqa      (@X[4],&QWP(112-16+16*(($Xi)/5),"esp"));        # K_XX_X
1017        eval(shift(@insns));
1018        eval(shift(@insns));

1020        foreach (@insns) { eval; }     # remaining instructions [if any]

1022  $Xi++;        push(@X,shift(@X));     # "rotate" X[]
1023 }

1025 sub Xupdate_avx_32_79()
1026 { use integer;
1027   my $body = shift;
1028   my @insns = (&$body,&$body,&$body,&$body);    # 32 to 48 instructions
1029   my ($a,$b,$c,$d,$e);

1031        &vpalignr(@X[2],@X[-1&7],@X[-2&7],8);   # compose "X[-6]"
1032        &vpxor  (@X[0],@X[0],@X[-4&7]); # "X[0]"="X[-32]"^"X[-16]"
1033         eval(shift(@insns));           # body_20_39
1034         eval(shift(@insns));
1035         eval(shift(@insns));
1036         eval(shift(@insns));           # rol

1038        &vpxor  (@X[0],@X[0],@X[-7&7]); # "X[0]"^="X[-28]"
1039          &vmovdqa      (&QWP(64+16*(($Xi-4)%3),"esp"),@X[-4&7]);       # save X
1040        eval(shift(@insns));
1041        eval(shift(@insns));
1042        if ($Xi%5) {
1043         &vmovdqa      (@X[4],@X[3]);  # "perpetuate" K_XX_XX...
1044        } else {                        # ... or load next one
1045         &vmovdqa      (@X[4],&QWP(112-16+16*($Xi/5),"esp"));
1046        }
1047         &vpaddd       (@X[3],@X[3],@X[-1&7]);
1048        eval(shift(@insns));           # ror
1049        eval(shift(@insns));

1051       &vpxor  (@X[0],@X[0],@X[2]);            # "X[0]"^="X[-6]"
```

```
1052          eval(shift(@insns));            # body_20_39
1053          eval(shift(@insns));
1054          eval(shift(@insns));
1055          eval(shift(@insns));            # rol

1057         &vpsrld (@X[2],@X[0],30);
1058           &vmovdqa      (&QWP(0+16*(($Xi-1)&3),"esp"),@X[3]);   # X[]+K xfer to
1059          eval(shift(@insns));
1060          eval(shift(@insns));
1061          eval(shift(@insns));            # ror
1062          eval(shift(@insns));

1064         &vpslld (@X[0],@X[0],2);
1065          eval(shift(@insns));            # body_20_39
1066          eval(shift(@insns));
1067          eval(shift(@insns));
1068          eval(shift(@insns));            # rol
1069          eval(shift(@insns));
1070          eval(shift(@insns));
1071          eval(shift(@insns));            # ror
1072          eval(shift(@insns));

1074         &vpor   (@X[0],@X[0],@X[2]);    # "X[0]"<<<=2
1075          eval(shift(@insns));            # body_20_39
1076          eval(shift(@insns));
1077           &vmovdqa      (@X[2],&QWP(64+16*(($Xi-6)%3),"esp")) if($Xi<19);
1078          eval(shift(@insns));
1079          eval(shift(@insns));            # rol
1080          eval(shift(@insns));
1081          eval(shift(@insns));
1082          eval(shift(@insns));            # ror
1083          eval(shift(@insns));

1085          foreach (@insns) { eval; }     # remaining instructions

1087   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
1088 }

1090 sub Xuplast_avx_80()
1091 { use integer;
1092   my $body = shift;
1093   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
1094   my ($a,$b,$c,$d,$e);

1096          eval(shift(@insns));
1097           &vpaddd       (@X[3],@X[3],@X[-1&7]);
1098          eval(shift(@insns));
1099          eval(shift(@insns));
1100          eval(shift(@insns));
1101          eval(shift(@insns));

1103           &vmovdqa      (&QWP(0+16*(($Xi-1)&3),"esp"),@X[3]);   # X[]+K xfer IAL

1105          foreach (@insns) { eval; }             # remaining instructions

1107         &mov    ($inp=@T[1],&DWP(192+4,"esp"));
1108         &cmp    ($inp,&DWP(192+8,"esp"));
1109         &je     (&label("done"));

1111         &vmovdqa(@X[3],&QWP(112+48,"esp"));     # K_00_19
1112         &vmovdqa(@X[2],&QWP(112+64,"esp"));     # pbswap mask
1113         &vmovdqu(@X[-4&7],&QWP(0,$inp));        # load input
1114         &vmovdqu(@X[-3&7],&QWP(16,$inp));
1115         &vmovdqu(@X[-2&7],&QWP(32,$inp));
1116         &vmovdqu(@X[-1&7],&QWP(48,$inp));
1117         &add    ($inp,64);
```



```
1118         &vpshufb(@X[-4&7],@X[-4&7],@X[2]);              # byte swap
1119         &mov    (&DWP(192+4,"esp"),$inp);
1120         &vmovdqa(&QWP(112-16,"esp"),@X[3]);     # borrow last backtrace slot

1122   $Xi=0;
1123 }

1125 sub Xloop_avx()
1126 { use integer;
1127   my $body = shift;
1128   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
1129   my ($a,$b,$c,$d,$e);

1131          eval(shift(@insns));
1132          eval(shift(@insns));
1133         &vpshufb        (@X[($Xi-3)&7],@X[($Xi-3)&7],@X[2]);
1134          eval(shift(@insns));
1135          eval(shift(@insns));
1136         &vpaddd (@X[$Xi&7],@X[($Xi-4)&7],@X[3]);
1137          eval(shift(@insns));
1138          eval(shift(@insns));
1139          eval(shift(@insns));
1140          eval(shift(@insns));
1141         &vmovdqa        (&QWP(0+16*$Xi,"esp"),@X[$Xi&7]);       # X[]+K xfer to
1142          eval(shift(@insns));
1143          eval(shift(@insns));

1145         foreach (@insns) { eval; }
1146   $Xi++;
1147 }

1149 sub Xtail_avx()
1150 { use integer;
1151   my $body = shift;
1152   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
1153   my ($a,$b,$c,$d,$e);

1155         foreach (@insns) { eval; }
1156 }

1158 &set_label("loop",16);
1159         &Xupdate_avx_16_31(\&body_00_19);
1160         &Xupdate_avx_16_31(\&body_00_19);
1161         &Xupdate_avx_16_31(\&body_00_19);
1162         &Xupdate_avx_16_31(\&body_00_19);
1163         &Xupdate_avx_32_79(\&body_00_19);
1164         &Xupdate_avx_32_79(\&body_20_39);
1165         &Xupdate_avx_32_79(\&body_20_39);
1166         &Xupdate_avx_32_79(\&body_20_39);
1167         &Xupdate_avx_32_79(\&body_20_39);
1168         &Xupdate_avx_32_79(\&body_20_39);
1169         &Xupdate_avx_32_79(\&body_40_59);
1170         &Xupdate_avx_32_79(\&body_40_59);
1171         &Xupdate_avx_32_79(\&body_40_59);
1172         &Xupdate_avx_32_79(\&body_40_59);
1173         &Xupdate_avx_32_79(\&body_40_59);
1174         &Xupdate_avx_32_79(\&body_20_39);
1175         &Xuplast_avx_80(\&body_20_39);  # can jump to "done"

1177                                 $saved_j=$j; @saved_V=@V;

1179         &Xloop_avx(\&body_20_39);
1180         &Xloop_avx(\&body_20_39);
1181         &Xloop_avx(\&body_20_39);

1183         &mov    (@T[1],&DWP(192,"esp"));        # update context
```

```
1184         &add    ($A,&DWP(0,@T[1]));
1185         &add    (@T[0],&DWP(4,@T[1]));          # $b
1186         &add    ($C,&DWP(8,@T[1]));
1187         &mov    (&DWP(0,@T[1]),$A);
1188         &add    ($D,&DWP(12,@T[1]));
1189         &mov    (&DWP(4,@T[1]),@T[0]);
1190         &add    ($E,&DWP(16,@T[1]));
1191         &mov    (&DWP(8,@T[1]),$C);
1192         &mov    ($B,@T[0]);
1193         &mov    (&DWP(12,@T[1]),$D);
1194         &mov    (&DWP(16,@T[1]),$E);

1196         &jmp    (&label("loop"));

1198 &set_label("done",16);          $j=$saved_j; @V=@saved_V;

1200         &Xtail_avx(\&body_20_39);
1201         &Xtail_avx(\&body_20_39);
1202         &Xtail_avx(\&body_20_39);

1204         &vzeroall();

1206         &mov    (@T[1],&DWP(192,"esp"));        # update context
1207         &add    ($A,&DWP(0,@T[1]));
1208         &mov    ("esp",&DWP(192+12,"esp"));     # restore %esp
1209         &add    (@T[0],&DWP(4,@T[1]));          # $b
1210         &add    ($C,&DWP(8,@T[1]));
1211         &mov    (&DWP(0,@T[1]),$A);
1212         &add    ($D,&DWP(12,@T[1]));
1213         &mov    (&DWP(4,@T[1]),@T[0]);
1214         &add    ($E,&DWP(16,@T[1]));
1215         &mov    (&DWP(8,@T[1]),$C);
1216         &mov    (&DWP(12,@T[1]),$D);
1217         &mov    (&DWP(16,@T[1]),$E);
1218 &function_end("_sha1_block_data_order_avx");
1219 }
1220 &set_label("K_XX_XX",64);
1221 &data_word(0x5a827999,0x5a827999,0x5a827999,0x5a827999);        # K_00_19
1222 &data_word(0x6ed9eba1,0x6ed9eba1,0x6ed9eba1,0x6ed9eba1);        # K_20_39
1223 &data_word(0x8f1bbcdc,0x8f1bbcdc,0x8f1bbcdc,0x8f1bbcdc);        # K_40_59
1224 &data_word(0xca62c1d6,0xca62c1d6,0xca62c1d6,0xca62c1d6);        # K_60_79
1225 &data_word(0x00010203,0x04050607,0x08090a0b,0x0c0d0e0f);        # pbswap mask
1226 }
1227 &asciz("SHA1 block transform for x86, CRYPTOGAMS by <appro\@openssl.org>");

1229 &asm_finish();
1230 #endif /* ! codereview */
```

****
30275 Wed Aug 13 19:53:09 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/sha1-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# sha1_block procedure for x86_64.
#
# It was brought to my attention that on EM64T compiler-generated code
# was far behind 32-bit assembler implementation. This is unlike on
# Opteron where compiler-generated code was only 15% behind 32-bit
# assembler, which originally made it hard to motivate the effort.
# There was suggestion to mechanically translate 32-bit code, but I
# dismissed it, reasoning that x86_64 offers enough register bank
# capacity to fully utilize SHA-1 parallelism. Therefore this fresh
# implementation:-) However! While 64-bit code does perform better
# on Opteron, I failed to beat 32-bit assembler on EM64T core. Well,
# x86_64 does offer larger *addressable* bank, but out-of-order core
# reaches for even more registers through dynamic aliasing, and EM64T
# core must have managed to run-time optimize even 32-bit code just as
# good as 64-bit one. Performance improvement is summarized in the
# following table:
#
#               gcc 3.4         32-bit asm      cycles/byte
# Opteron       +45%            +20%            6.8
# Xeon P4       +65%            +0%             9.9
# Core2         +60%            +10%            7.0

# August 2009.
#
# The code was revised to minimize code size and to maximize
# "distance" between instructions producing input to 'lea'
# instruction and the 'lea' instruction itself, which is essential
# for Intel Atom core.

# October 2010.
#
# Add SSSE3, Supplemental[!] SSE3, implementation. The idea behind it
# is to offload message schedule denoted by Wt in NIST specification,
# or Xupdate in OpenSSL source, to SIMD unit. See sha1-586.pl module
# for background and implementation details. The only difference from
# 32-bit code is that 64-bit code doesn't have to spill @X[] elements
# to free temporary registers.

# April 2011.
#
# Add AVX code path. See sha1-586.pl for further information.

######################################################################
# Current performance is summarized in following table. Numbers are
# CPU clock cycles spent to process single byte (less is better).
#
#               x86_64          SSSE3           AVX
# P4            9.8             -
# Opteron       6.6             -
# Core2         6.7             6.1/+10%        -
# Atom          11.0            9.7/+13%        -
# Westmere      7.1             5.6/+27%        -
```



```
# Sandy Bridge  7.9             6.3/+25%        5.2/+51%

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

$avx=1 if ('$ENV{CC} -Wa,-v -c -o /dev/null -x assembler /dev/null 2>&1'
                =~ /GNU assembler version ([2-9]\.[0-9]+)/ &&
           $1>=2.19);
$avx=1 if (!$avx && $win64 && ($flavour =~ /nasm/ || $ENV{ASM} =~ /nasm/) &&
           'nasm -v 2>&1' =~ /NASM version ([2-9]\.[0-9]+)/ &&
           $1>=2.09);
$avx=1 if (!$avx && $win64 && ($flavour =~ /masm/ || $ENV{ASM} =~ /ml64/) &&
           'ml64 2>&1' =~ /Version ([0-9]+)\./ &&
           $1>=10);

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

$ctx="%rdi";    # 1st arg
$inp="%rsi";    # 2nd arg
$num="%rdx";    # 3rd arg

# reassign arguments in order to produce more compact code
$ctx="%r8";
$inp="%r9";
$num="%r10";

$t0="%eax";
$t1="%ebx";
$t2="%ecx";
@xi=("%edx","%ebp");
$A="%esi";
$B="%edi";
$C="%r11d";
$D="%r12d";
$E="%r13d";

@V=($A,$B,$C,$D,$E);

sub BODY_00_19 {
my ($i,$a,$b,$c,$d,$e)=@_;
my $j=$i+1;
$code.=<<___ if ($i==0);
        mov     '4*$i'($inp),$xi[0]
        bswap   $xi[0]
        mov     $xi[0],'4*$i'(%rsp)
___
$code.=<<___ if ($i<15);
        mov     $c,$t0
        mov     '4*$j'($inp),$xi[1]
        mov     $a,$t2
        xor     $d,$t0
        bswap   $xi[1]
        rol     \$5,$t2
        lea     0x5a827999($xi[0],$e),$e
        and     $b,$t0
        mov     $xi[1],'4*$j'(%rsp)
        add     $t2,$e
```

```
        xor     $d,$t0
        rol     \$30,$b
        add     $t0,$e
___
$code.=<<___ if ($i>=15);
        mov     `4*($j%16)`(%rsp),$xi[1]
        mov     $c,$t0
        mov     $a,$t2
        xor     `4*(($j+2)%16)`(%rsp),$xi[1]
        xor     $d,$t0
        rol     \$5,$t2
        xor     `4*(($j+8)%16)`(%rsp),$xi[1]
        and     $b,$t0
        lea     0x5a827999($xi[0],$e),$e
        xor     `4*(($j+13)%16)`(%rsp),$xi[1]
        xor     $d,$t0
        rol     \$1,$xi[1]
        add     $t2,$e
        rol     \$30,$b
        mov     $xi[1],`4*($j%16)`(%rsp)
        add     $t0,$e
___
unshift(@xi,pop(@xi));
}

sub BODY_20_39 {
my ($i,$a,$b,$c,$d,$e)=@_;
my $j=$i+1;
my $K=($i<40)?0x6ed9eba1:0xca62c1d6;
$code.=<<___ if ($i<79);
        mov     `4*($j%16)`(%rsp),$xi[1]
        mov     $c,$t0
        mov     $a,$t2
        xor     `4*(($j+2)%16)`(%rsp),$xi[1]
        xor     $b,$t0
        rol     \$5,$t2
        lea     $K($xi[0],$e),$e
        xor     `4*(($j+8)%16)`(%rsp),$xi[1]
        xor     $d,$t0
        add     $t2,$e
        xor     `4*(($j+13)%16)`(%rsp),$xi[1]
        rol     \$30,$b
        add     $t0,$e
        rol     \$1,$xi[1]
___
$code.=<<___ if ($i<76);
        mov     $xi[1],`4*($j%16)`(%rsp)
___
$code.=<<___ if ($i==79);
        mov     $c,$t0
        mov     $a,$t2
        xor     $b,$t0
        lea     $K($xi[0],$e),$e
        rol     \$5,$t2
        xor     $d,$t0
        add     $t2,$e
        rol     \$30,$b
        add     $t0,$e
___
unshift(@xi,pop(@xi));
}

sub BODY_40_59 {
my ($i,$a,$b,$c,$d,$e)=@_;
my $j=$i+1;
$code.=<<___;
```



```
        mov     `4*($j%16)`(%rsp),$xi[1]
        mov     $c,$t0
        mov     $c,$t1
        xor     `4*(($j+2)%16)`(%rsp),$xi[1]
        and     $d,$t0
        mov     $a,$t2
        xor     `4*(($j+8)%16)`(%rsp),$xi[1]
        xor     $d,$t1
        lea     0x8f1bbcdc($xi[0],$e),$e
        rol     \$5,$t2
        xor     `4*(($j+13)%16)`(%rsp),$xi[1]
        add     $t0,$e
        and     $b,$t1
        rol     \$1,$xi[1]
        add     $t1,$e
        rol     \$30,$b
        mov     $xi[1],`4*($j%16)`(%rsp)
        add     $t2,$e
___
unshift(@xi,pop(@xi));
}

$code.=<<___;
.text
.extern OPENSSL_ia32cap_P

.globl  sha1_block_data_order
.type   sha1_block_data_order,\@function,3
.align  16
sha1_block_data_order:
        mov     OPENSSL_ia32cap_P+0(%rip),%r9d
        mov     OPENSSL_ia32cap_P+4(%rip),%r8d
        test    \$`1<<9`,%r8d           # check SSSE3 bit
        jz      .Lialu
___
$code.=<<___ if ($avx);
        and     \$`1<<28`,%r8d          # mask AVX bit
        and     \$`1<<30`,%r9d          # mask "Intel CPU" bit
        or      %r9d,%r8d
        cmp     \$`1<<28|1<<30`,%r8d
        je      _avx_shortcut
___
$code.=<<___;
        jmp     _ssse3_shortcut

.align  16
.Lialu:
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        mov     %rsp,%r11
        mov     %rdi,$ctx       # reassigned argument
        sub     \$`8+16*4`,%rsp
        mov     %rsi,$inp       # reassigned argument
        and     \$-64,%rsp
        mov     %rdx,$num       # reassigned argument
        mov     %r11,`16*4`(%rsp)
.Lprologue:

        mov     0($ctx),$A
        mov     4($ctx),$B
        mov     8($ctx),$C
        mov     12($ctx),$D
        mov     16($ctx),$E
        jmp     .Lloop
```

```
.align  16
.Lloop:
___
for($i=0;$i<20;$i++)    { &BODY_00_19($i,@V); unshift(@V,pop(@V)); }
for(;$i<40;$i++)        { &BODY_20_39($i,@V); unshift(@V,pop(@V)); }
for(;$i<60;$i++)        { &BODY_40_59($i,@V); unshift(@V,pop(@V)); }
for(;$i<80;$i++)        { &BODY_20_39($i,@V); unshift(@V,pop(@V)); }
$code.=<<___;
        add     0($ctx),$A
        add     4($ctx),$B
        add     8($ctx),$C
        add     12($ctx),$D
        add     16($ctx),$E
        mov     $A,0($ctx)
        mov     $B,4($ctx)
        mov     $C,8($ctx)
        mov     $D,12($ctx)
        mov     $E,16($ctx)

        sub     \$1,$num
        lea     '16*4'($inp),$inp
        jnz     .Lloop

        mov     '16*4'(%rsp),%rsi
        mov     (%rsi),%r13
        mov     8(%rsi),%r12
        mov     16(%rsi),%rbp
        mov     24(%rsi),%rbx
        lea     32(%rsi),%rsp
.Lepilogue:
        ret
.size   sha1_block_data_order,.-sha1_block_data_order
___
{{{
my $Xi=4;
my @X=map("%xmm$_",(4..7,0..3));
my @Tx=map("%xmm$_",(8..10));
my @V=($A,$B,$C,$D,$E)=("%eax","%ebx","%ecx","%edx","%ebp");    # size optimizat
my @T=("%esi","%edi");
my $j=0;
my $K_XX_XX="%r11";

my $_rol=sub { &rol(@_) };
my $_ror=sub { &ror(@_) };

$code.=<<___;
.type   sha1_block_data_order_ssse3,\@function,3
.align  16
sha1_block_data_order_ssse3:
_ssse3_shortcut:
        push    %rbx
        push    %rbp
        push    %r12
        lea     '-64-($win64?5*16:0)'(%rsp),%rsp
___
$code.=<<___ if ($win64);
        movaps  %xmm6,64+0(%rsp)
        movaps  %xmm7,64+16(%rsp)
        movaps  %xmm8,64+32(%rsp)
        movaps  %xmm9,64+48(%rsp)
        movaps  %xmm10,64+64(%rsp)
.Lprologue_ssse3:
___
$code.=<<___;
        mov     %rdi,$ctx       # reassigned argument
```



```
        mov     %rsi,$inp       # reassigned argument
        mov     %rdx,$num       # reassigned argument

        shl     \$6,$num
        add     $inp,$num
        lea     K_XX_XX(%rip),$K_XX_XX

        mov     0($ctx),$A              # load context
        mov     4($ctx),$B
        mov     8($ctx),$C
        mov     12($ctx),$D
        mov     $B,@T[0]                # magic seed
        mov     16($ctx),$E

        movdqa  64($K_XX_XX),@X[2]      # pbswap mask
        movdqa  0($K_XX_XX),@Tx[1]      # K_00_19
        movdqu  0($inp),@X[-4&7]        # load input to %xmm[0-3]
        movdqu  16($inp),@X[-3&7]
        movdqu  32($inp),@X[-2&7]
        movdqu  48($inp),@X[-1&7]
        pshufb  @X[2],@X[-4&7]          # byte swap
        add     \$64,$inp
        pshufb  @X[2],@X[-3&7]
        pshufb  @X[2],@X[-2&7]
        pshufb  @X[2],@X[-1&7]
        paddd   @Tx[1],@X[-4&7]         # add K_00_19
        paddd   @Tx[1],@X[-3&7]
        paddd   @Tx[1],@X[-2&7]
        movdqa  @X[-4&7],0(%rsp)        # X[]+K xfer to IALU
        psubd   @Tx[1],@X[-4&7]         # restore X[]
        movdqa  @X[-3&7],16(%rsp)
        psubd   @Tx[1],@X[-3&7]
        movdqa  @X[-2&7],32(%rsp)
        psubd   @Tx[1],@X[-2&7]
        jmp     .Loop_ssse3
___

sub AUTOLOAD()          # thunk [simplified] 32-bit style perlasm
{ my $opcode = $AUTOLOAD; $opcode =~ s/.*::/;
  my $arg = pop;
    $arg = "\$$arg" if ($arg*1 eq $arg);
    $code .= "\t$opcode\t".join(',',$arg,reverse @_)."\n";
}

sub Xupdate_ssse3_16_31()               # recall that $Xi starts wtih 4
{ use integer;
  my $body = shift;
  my @insns = (&$body,&$body,&$body,&$body);    # 40 instructions
  my ($a,$b,$c,$d,$e);

        &movdqa (@X[0],@X[-3&7]);
         eval(shift(@insns));
         eval(shift(@insns));
        &movdqa (@Tx[0],@X[-1&7]);
        &palignr(@X[0],@X[-4&7],8);     # compose "X[-14]" in "X[0]"
         eval(shift(@insns));
         eval(shift(@insns));

          &paddd        (@Tx[1],@X[-1&7]);
         eval(shift(@insns));
         eval(shift(@insns));
        &psrldq (@Tx[0],4);             # "X[-3]", 3 dwords
         eval(shift(@insns));
         eval(shift(@insns));
        &pxor   (@X[0],@X[-4&7]);       # "X[0]"^="X[-16]"
         eval(shift(@insns));
```

```
392          eval(shift(@insns));

394         &pxor   (@Tx[0],@X[-2&7]);      # "X[-3]"^"X[-8]"
395          eval(shift(@insns));
396          eval(shift(@insns));
397          eval(shift(@insns));
398          eval(shift(@insns));

400         &pxor   (@X[0],@Tx[0]);         # "X[0]"^="X[-3]"^"X[-8]"
401          eval(shift(@insns));
402          eval(shift(@insns));
403           &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
404          eval(shift(@insns));
405          eval(shift(@insns));

407         &movdqa (@Tx[2],@X[0]);
408         &movdqa (@Tx[0],@X[0]);
409          eval(shift(@insns));
410          eval(shift(@insns));
411          eval(shift(@insns));
412          eval(shift(@insns));

414         &pslldq (@Tx[2],12);            # "X[0]"<<96, extract one dword
415         &paddd  (@X[0],@X[0]);
416          eval(shift(@insns));
417          eval(shift(@insns));
418          eval(shift(@insns));
419          eval(shift(@insns));

421         &psrld  (@Tx[0],31);
422          eval(shift(@insns));
423          eval(shift(@insns));
424         &movdqa (@Tx[1],@Tx[2]);
425          eval(shift(@insns));
426          eval(shift(@insns));

428         &psrld  (@Tx[2],30);
429         &por    (@X[0],@Tx[0]);         # "X[0]"<<<1
430          eval(shift(@insns));
431          eval(shift(@insns));
432          eval(shift(@insns));
433          eval(shift(@insns));

435         &pslld  (@Tx[1],2);
436         &pxor   (@X[0],@Tx[2]);
437          eval(shift(@insns));
438          eval(shift(@insns));
439           &movdqa       (@Tx[2],eval(16*(($Xi)/5))."($K_XX_XX)");       # K_XX_X
440          eval(shift(@insns));
441          eval(shift(@insns));

443         &pxor   (@X[0],@Tx[1]);         # "X[0]"^=("X[0]">>96)<<<2

445          foreach (@insns) { eval; }     # remaining instructions [if any]

447   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
448                 push(@Tx,shift(@Tx));
449 }

451 sub Xupdate_ssse3_32_79()
452 { use integer;
453   my $body = shift;
454   my @insns = (&$body,&$body,&$body,&$body);    # 32 to 48 instructions
455   my ($a,$b,$c,$d,$e);

457         &movdqa (@Tx[0],@X[-1&7])       if ($Xi==8);
```



```
458          eval(shift(@insns));           # body_20_39
459         &pxor   (@X[0],@X[-4&7]);       # "X[0]"="X[-32]"^"X[-16]"
460         &palignr(@Tx[0],@X[-2&7],8);    # compose "X[-6]"
461          eval(shift(@insns));
462          eval(shift(@insns));
463          eval(shift(@insns));           # rol

465         &pxor   (@X[0],@X[-7&7]);       # "X[0]"^="X[-28]"
466          eval(shift(@insns));
467          eval(shift(@insns))    if (@insns[0] !~ /&ro[rl]/);
468         if ($Xi%5) {
469           &movdqa       (@Tx[2],@Tx[1]);# "perpetuate" K_XX_XX...
470         } else {                        # ... or load next one
471           &movdqa       (@Tx[2],eval(16*($Xi/5))."($K_XX_XX)");
472         }
473           &paddd        (@Tx[1],@X[-1&7]);
474          eval(shift(@insns));           # ror
475          eval(shift(@insns));

477         &pxor   (@X[0],@Tx[0]);         # "X[0]"^="X[-6]"
478          eval(shift(@insns));           # body_20_39
479          eval(shift(@insns));
480          eval(shift(@insns));
481          eval(shift(@insns));           # rol

483         &movdqa (@Tx[0],@X[0]);
484           &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
485          eval(shift(@insns));
486          eval(shift(@insns));
487          eval(shift(@insns));           # ror
488          eval(shift(@insns));

490         &pslld  (@X[0],2);
491          eval(shift(@insns));           # body_20_39
492          eval(shift(@insns));
493         &psrld  (@Tx[0],30);
494          eval(shift(@insns));
495          eval(shift(@insns));           # rol
496          eval(shift(@insns));
497          eval(shift(@insns));
498          eval(shift(@insns));           # ror
499          eval(shift(@insns));

501         &por    (@X[0],@Tx[0]);         # "X[0]"<<<2
502          eval(shift(@insns));           # body_20_39
503          eval(shift(@insns));
504           &movdqa       (@Tx[1],@X[0])  if ($Xi<19);
505          eval(shift(@insns));
506          eval(shift(@insns));           # rol
507          eval(shift(@insns));
508          eval(shift(@insns));
509          eval(shift(@insns));           # rol
510          eval(shift(@insns));

512          foreach (@insns) { eval; }     # remaining instructions

514   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
515                 push(@Tx,shift(@Tx));
516 }

518 sub Xuplast_ssse3_80()
519 { use integer;
520   my $body = shift;
521   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
522   my ($a,$b,$c,$d,$e);
```

```
524          eval(shift(@insns));
525           &paddd        (@Tx[1],@X[-1&7]);
526          eval(shift(@insns));
527          eval(shift(@insns));
528          eval(shift(@insns));
529          eval(shift(@insns));

531           &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer IAL

533          foreach (@insns) { eval; }             # remaining instructions

535         &cmp    ($inp,$num);
536         &je     (".Ldone_ssse3");

538         unshift(@Tx,pop(@Tx));

540         &movdqa (@X[2],"64($K_XX_XX)");         # pbswap mask
541         &movdqa (@Tx[1],"0($K_XX_XX)");         # K_00_19
542         &movdqu (@X[-4&7],"0($inp)");           # load input
543         &movdqu (@X[-3&7],"16($inp)");
544         &movdqu (@X[-2&7],"32($inp)");
545         &movdqu (@X[-1&7],"48($inp)");
546         &pshufb (@X[-4&7],@X[2]);               # byte swap
547         &add    ($inp,64);

549   $Xi=0;
550 }

552 sub Xloop_ssse3()
553 { use integer;
554   my $body = shift;
555   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
556   my ($a,$b,$c,$d,$e);

558          eval(shift(@insns));
559          eval(shift(@insns));
560         &pshufb (@X[($Xi-3)&7],@X[2]);
561          eval(shift(@insns));
562          eval(shift(@insns));
563         &paddd  (@X[($Xi-4)&7],@Tx[1]);
564          eval(shift(@insns));
565          eval(shift(@insns));
566          eval(shift(@insns));
567          eval(shift(@insns));
568         &movdqa (eval(16*$Xi)."(%rsp)",@X[($Xi-4)&7]);  # X[]+K xfer to IALU
569          eval(shift(@insns));
570          eval(shift(@insns));
571         &psubd  (@X[($Xi-4)&7],@Tx[1]);

573         foreach (@insns) { eval; }
574   $Xi++;
575 }

577 sub Xtail_ssse3()
578 { use integer;
579   my $body = shift;
580   my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
581   my ($a,$b,$c,$d,$e);

583         foreach (@insns) { eval; }
584 }

586 sub body_00_19 () {
587         (
588         '($a,$b,$c,$d,$e)=@V;'.
589         '&add   ($e,eval(4*($j&15))."(%rsp)");',        # X[]+K xfer
```



```
590         '&xor   ($c,$d);',
591         '&mov   (@T[1],$a);',   # $b in next round
592         '&$_rol ($a,5);',
593         '&and   (@T[0],$c);',   # ($b&($c^$d))
594         '&xor   ($c,$d);',      # restore $c
595         '&xor   (@T[0],$d);',
596         '&add   ($e,$a);',
597         '&$_ror ($b,$j?7:2);',  # $b>>>2
598         '&add   ($e,@T[0]);'    .'$j++; unshift(@V,pop(@V)); unshift(@T,pop(@T))
599         );
600 }

602 sub body_20_39 () {
603         (
604         '($a,$b,$c,$d,$e)=@V;'.
605         '&add   ($e,eval(4*($j++&15))."(%rsp)");',      # X[]+K xfer
606         '&xor   (@T[0],$d);',   # ($b^$d)
607         '&mov   (@T[1],$a);',   # $b in next round
608         '&$_rol ($a,5);',
609         '&xor   (@T[0],$c);',   # ($b^$d^$c)
610         '&add   ($e,$a);',
611         '&$_ror ($b,7);',       # $b>>>2
612         '&add   ($e,@T[0]);'    .'unshift(@V,pop(@V)); unshift(@T,pop(@T));'
613         );
614 }

616 sub body_40_59 () {
617         (
618         '($a,$b,$c,$d,$e)=@V;'.
619         '&mov   (@T[1],$c);',
620         '&xor   ($c,$d);',
621         '&add   ($e,eval(4*($j++&15))."(%rsp)");',      # X[]+K xfer
622         '&and   (@T[1],$d);',
623         '&and   (@T[0],$c);',   # ($b&($c^$d))
624         '&$_ror ($b,7);',       # $b>>>2
625         '&add   ($e,@T[1]);',
626         '&mov   (@T[1],$a);',   # $b in next round
627         '&$_rol ($a,5);',
628         '&add   ($e,@T[0]);',
629         '&xor   ($c,$d);',      # restore $c
630         '&add   ($e,$a);'       .'unshift(@V,pop(@V)); unshift(@T,pop(@T));'
631         );
632 }
633 $code.=<<___;
634 .align  16
635 .Loop_ssse3:
636 ___
637         &Xupdate_ssse3_16_31(\&body_00_19);
638         &Xupdate_ssse3_16_31(\&body_00_19);
639         &Xupdate_ssse3_16_31(\&body_00_19);
640         &Xupdate_ssse3_16_31(\&body_00_19);
641         &Xupdate_ssse3_32_79(\&body_00_19);
642         &Xupdate_ssse3_32_79(\&body_20_39);
643         &Xupdate_ssse3_32_79(\&body_20_39);
644         &Xupdate_ssse3_32_79(\&body_20_39);
645         &Xupdate_ssse3_32_79(\&body_20_39);
646         &Xupdate_ssse3_32_79(\&body_20_39);
647         &Xupdate_ssse3_32_79(\&body_40_59);
648         &Xupdate_ssse3_32_79(\&body_40_59);
649         &Xupdate_ssse3_32_79(\&body_40_59);
650         &Xupdate_ssse3_32_79(\&body_40_59);
651         &Xupdate_ssse3_32_79(\&body_40_59);
652         &Xupdate_ssse3_32_79(\&body_20_39);
653         &Xuplast_ssse3_80(\&body_20_39);        # can jump to "done"

655                                 $saved_j=$j; @saved_V=@V;
```

```
657         &Xloop_ssse3(\&body_20_39);
658         &Xloop_ssse3(\&body_20_39);
659         &Xloop_ssse3(\&body_20_39);

661 $code.=<<___;
662         add     0($ctx),$A                      # update context
663         add     4($ctx),@T[0]
664         add     8($ctx),$C
665         add     12($ctx),$D
666         mov     $A,0($ctx)
667         add     16($ctx),$E
668         mov     @T[0],4($ctx)
669         mov     @T[0],$B                        # magic seed
670         mov     $C,8($ctx)
671         mov     $D,12($ctx)
672         mov     $E,16($ctx)
673         jmp     .Loop_ssse3

675 .align  16
676 .Ldone_ssse3:
677 ___
678                                 $j=$saved_j; @V=@saved_V;

680         &Xtail_ssse3(\&body_20_39);
681         &Xtail_ssse3(\&body_20_39);
682         &Xtail_ssse3(\&body_20_39);

684 $code.=<<___;
685         add     0($ctx),$A                      # update context
686         add     4($ctx),@T[0]
687         add     8($ctx),$C
688         mov     $A,0($ctx)
689         add     12($ctx),$D
690         mov     @T[0],4($ctx)
691         add     16($ctx),$E
692         mov     $C,8($ctx)
693         mov     $D,12($ctx)
694         mov     $E,16($ctx)
695 ___
696 $code.=<<___ if ($win64);
697         movaps  64+0(%rsp),%xmm6
698         movaps  64+16(%rsp),%xmm7
699         movaps  64+32(%rsp),%xmm8
700         movaps  64+48(%rsp),%xmm9
701         movaps  64+64(%rsp),%xmm10
702 ___
703 $code.=<<___;
704         lea     `64+($win64?5*16:0)`(%rsp),%rsi
705         mov     0(%rsi),%r12
706         mov     8(%rsi),%rbp
707         mov     16(%rsi),%rbx
708         lea     24(%rsi),%rsp
709 .Lepilogue_ssse3:
710         ret
711 .size   sha1_block_data_order_ssse3,.-sha1_block_data_order_ssse3
712 ___

714 if ($avx) {
715 my $Xi=4;
716 my @X=map("%xmm$_",(4..7,0..3));
717 my @Tx=map("%xmm$_",(8..10));
718 my @V=($A,$B,$C,$D,$E)=("%eax","%ebx","%ecx","%edx","%ebp");    # size optimizat
719 my @T=("%esi","%edi");
720 my $j=0;
721 my $K_XX_XX="%r11";
```



```
723 my $_rol=sub { &shld(@_[0],@_) };
724 my $_ror=sub { &shrd(@_[0],@_) };

726 $code.=<<___;
727 .type   sha1_block_data_order_avx,\@function,3
728 .align  16
729 sha1_block_data_order_avx:
730 _avx_shortcut:
731         push    %rbx
732         push    %rbp
733         push    %r12
734         lea     `-64-($win64?5*16:0)`(%rsp),%rsp
735 ___
736 $code.=<<___ if ($win64);
737         movaps  %xmm6,64+0(%rsp)
738         movaps  %xmm7,64+16(%rsp)
739         movaps  %xmm8,64+32(%rsp)
740         movaps  %xmm9,64+48(%rsp)
741         movaps  %xmm10,64+64(%rsp)
742 .Lprologue_avx:
743 ___
744 $code.=<<___;
745         mov     %rdi,$ctx       # reassigned argument
746         mov     %rsi,$inp       # reassigned argument
747         mov     %rdx,$num       # reassigned argument
748         vzeroupper

750         shl     \$6,$num
751         add     $inp,$num
752         lea     K_XX_XX(%rip),$K_XX_XX

754         mov     0($ctx),$A              # load context
755         mov     4($ctx),$B
756         mov     8($ctx),$C
757         mov     12($ctx),$D
758         mov     $B,@T[0]                # magic seed
759         mov     16($ctx),$E

761         vmovdqa 64($K_XX_XX),@X[2]      # pbswap mask
762         vmovdqa 0($K_XX_XX),@Tx[1]      # K_00_19
763         vmovdqu 0($inp),@X[-4&7]        # load input to %xmm[0-3]
764         vmovdqu 16($inp),@X[-3&7]
765         vmovdqu 32($inp),@X[-2&7]
766         vmovdqu 48($inp),@X[-1&7]
767         vpshufb @X[2],@X[-4&7],@X[-4&7] # byte swap
768         add     \$64,$inp
769         vpshufb @X[2],@X[-3&7],@X[-3&7]
770         vpshufb @X[2],@X[-2&7],@X[-2&7]
771         vpshufb @X[2],@X[-1&7],@X[-1&7]
772         vpaddd  @Tx[1],@X[-4&7],@X[0]   # add K_00_19
773         vpaddd  @Tx[1],@X[-3&7],@X[1]
774         vpaddd  @Tx[1],@X[-2&7],@X[2]
775         vmovdqa @X[0],0(%rsp)           # X[]+K xfer to IALU
776         vmovdqa @X[1],16(%rsp)
777         vmovdqa @X[2],32(%rsp)
778         jmp     .Loop_avx
779 ___

781 sub Xupdate_avx_16_31()         # recall that $Xi starts wtih 4
782 { use integer;
783   my $body = shift;
784   my @insns = (&$body,&$body,&$body,&$body);    # 40 instructions
785   my ($a,$b,$c,$d,$e);

787          eval(shift(@insns));
```

```
788          eval(shift(@insns));
789         &vpalignr(@X[0],@X[-3&7],@X[-4&7],8);   # compose "X[-14]" in "X[0]"
790          eval(shift(@insns));
791          eval(shift(@insns));

793           &vpaddd       (@Tx[1],@Tx[1],@X[-1&7]);
794          eval(shift(@insns));
795          eval(shift(@insns));
796         &vpsrldq(@Tx[0],@X[-1&7],4);    # "X[-3]", 3 dwords
797          eval(shift(@insns));
798          eval(shift(@insns));
799         &vpxor  (@X[0],@X[0],@X[-4&7]);         # "X[0]"^="X[-16]"
800          eval(shift(@insns));
801          eval(shift(@insns));

803         &vpxor  (@Tx[0],@Tx[0],@X[-2&7]);       # "X[-3]"^"X[-8]"
804          eval(shift(@insns));
805          eval(shift(@insns));
806          eval(shift(@insns));
807          eval(shift(@insns));

809         &vpxor  (@X[0],@X[0],@Tx[0]);           # "X[0]"^="X[-3]"^"X[-8]"
810          eval(shift(@insns));
811          eval(shift(@insns));
812           &vmovdqa      (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
813          eval(shift(@insns));
814          eval(shift(@insns));

816         &vpsrld (@Tx[0],@X[0],31);
817          eval(shift(@insns));
818          eval(shift(@insns));
819          eval(shift(@insns));
820          eval(shift(@insns));

822         &vpslldq(@Tx[2],@X[0],12);              # "X[0]"<<96, extract one dword
823         &vpaddd (@X[0],@X[0],@X[0]);
824          eval(shift(@insns));
825          eval(shift(@insns));
826          eval(shift(@insns));
827          eval(shift(@insns));

829         &vpsrld (@Tx[1],@Tx[2],30);
830         &vpor   (@X[0],@X[0],@Tx[0]);           # "X[0]"<<<=1
831          eval(shift(@insns));
832          eval(shift(@insns));
833          eval(shift(@insns));
834          eval(shift(@insns));

836         &vpslld (@Tx[2],@Tx[2],2);
837         &vpxor  (@X[0],@X[0],@Tx[1]);
838          eval(shift(@insns));
839          eval(shift(@insns));
840          eval(shift(@insns));
841          eval(shift(@insns));

843         &vpxor  (@X[0],@X[0],@Tx[2]);           # "X[0]"^=("X[0]">>96)<<<2
844          eval(shift(@insns));
845          eval(shift(@insns));
846           &vmovdqa      (@Tx[2],eval(16*(($Xi)/5))."($K_XX_XX)");       # K_XX_X
847          eval(shift(@insns));
848          eval(shift(@insns));


851          foreach (@insns) { eval; }     # remaining instructions [if any]

853   $Xi++;        push(@X,shift(@X));     # "rotate" X[]
```



```
854                 push(@Tx,shift(@Tx));
855   }

857   sub Xupdate_avx_32_79()
858   { use integer;
859     my $body = shift;
860     my @insns = (&$body,&$body,&$body,&$body);    # 32 to 48 instructions
861     my ($a,$b,$c,$d,$e);

863           &vpalignr(@Tx[0],@X[-1&7],@X[-2&7],8);  # compose "X[-6]"
864           &vpxor  (@X[0],@X[0],@X[-4&7]);         # "X[0]"="X[-32]"^"X[-16]"
865            eval(shift(@insns));           # body_20_39
866            eval(shift(@insns));
867            eval(shift(@insns));
868            eval(shift(@insns));           # rol

870           &vpxor  (@X[0],@X[0],@X[-7&7]);         # "X[0]"^="X[-28]"
871            eval(shift(@insns));
872            eval(shift(@insns))    if (@insns[0] !~ /&ro[rl]/);
873           if ($Xi%5) {
874             &vmovdqa      (@Tx[2],@Tx[1]);# "perpetuate" K_XX_XX...
875           } else {                        # ... or load next one
876             &vmovdqa      (@Tx[2],eval(16*($Xi/5))."($K_XX_XX)");
877           }
878             &vpaddd       (@Tx[1],@Tx[1],@X[-1&7]);
879            eval(shift(@insns));           # ror
880            eval(shift(@insns));

882           &vpxor  (@X[0],@X[0],@Tx[0]);           # "X[0]"^="X[-6]"
883            eval(shift(@insns));           # body_20_39
884            eval(shift(@insns));
885            eval(shift(@insns));
886            eval(shift(@insns));           # rol

888           &vpsrld (@Tx[0],@X[0],30);
889             &vmovdqa      (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer to
890            eval(shift(@insns));
891            eval(shift(@insns));
892            eval(shift(@insns));           # ror
893            eval(shift(@insns));

895           &vpslld (@X[0],@X[0],2);
896            eval(shift(@insns));           # body_20_39
897            eval(shift(@insns));
898            eval(shift(@insns));
899            eval(shift(@insns));           # rol
900            eval(shift(@insns));
901            eval(shift(@insns));
902            eval(shift(@insns));           # ror
903            eval(shift(@insns));

905           &vpor   (@X[0],@X[0],@Tx[0]);           # "X[0]"<<<=2
906            eval(shift(@insns));           # body_20_39
907            eval(shift(@insns));
908             &vmovdqa      (@Tx[1],@X[0])  if ($Xi<19);
909            eval(shift(@insns));
910            eval(shift(@insns));           # rol
911            eval(shift(@insns));
912            eval(shift(@insns));
913            eval(shift(@insns));           # rol
914            eval(shift(@insns));

916            foreach (@insns) { eval; }     # remaining instructions

918     $Xi++;        push(@X,shift(@X));     # "rotate" X[]
919                   push(@Tx,shift(@Tx));
```

```
}

sub Xuplast_avx_80()
{ use integer;
  my $body = shift;
  my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
  my ($a,$b,$c,$d,$e);

         eval(shift(@insns));
          &vpaddd       (@Tx[1],@Tx[1],@X[-1&7]);
         eval(shift(@insns));
         eval(shift(@insns));
         eval(shift(@insns));
         eval(shift(@insns));

          &movdqa       (eval(16*(($Xi-1)&3))."(%rsp)",@Tx[1]); # X[]+K xfer IAL

         foreach (@insns) { eval; }             # remaining instructions

        &cmp    ($inp,$num);
        &je     (".Ldone_avx");

        unshift(@Tx,pop(@Tx));

        &vmovdqa(@X[2],"64($K_XX_XX)");         # pbswap mask
        &vmovdqa(@Tx[1],"0($K_XX_XX)");         # K_00_19
        &vmovdqu(@X[-4&7],"0($inp)");           # load input
        &vmovdqu(@X[-3&7],"16($inp)");
        &vmovdqu(@X[-2&7],"32($inp)");
        &vmovdqu(@X[-1&7],"48($inp)");
        &vpshufb(@X[-4&7],@X[-4&7],@X[2]);      # byte swap
        &add    ($inp,64);

  $Xi=0;
}

sub Xloop_avx()
{ use integer;
  my $body = shift;
  my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
  my ($a,$b,$c,$d,$e);

         eval(shift(@insns));
         eval(shift(@insns));
        &vpshufb(@X[($Xi-3)&7],@X[($Xi-3)&7],@X[2]);
         eval(shift(@insns));
         eval(shift(@insns));
        &vpaddd (@X[$Xi&7],@X[($Xi-4)&7],@Tx[1]);
         eval(shift(@insns));
         eval(shift(@insns));
         eval(shift(@insns));
         eval(shift(@insns));
        &vmovdqa(eval(16*$Xi)."(%rsp)",@X[$Xi&7]);      # X[]+K xfer to IALU
         eval(shift(@insns));
         eval(shift(@insns));

        foreach (@insns) { eval; }
  $Xi++;
}

sub Xtail_avx()
{ use integer;
  my $body = shift;
  my @insns = (&$body,&$body,&$body,&$body);    # 32 instructions
  my ($a,$b,$c,$d,$e);
```



```
        foreach (@insns) { eval; }
}

$code.=<<___;
.align  16
.Loop_avx:
___
        &Xupdate_avx_16_31(\&body_00_19);
        &Xupdate_avx_16_31(\&body_00_19);
        &Xupdate_avx_16_31(\&body_00_19);
        &Xupdate_avx_16_31(\&body_00_19);
        &Xupdate_avx_32_79(\&body_00_19);
        &Xupdate_avx_32_79(\&body_20_39);
        &Xupdate_avx_32_79(\&body_20_39);
        &Xupdate_avx_32_79(\&body_20_39);
        &Xupdate_avx_32_79(\&body_20_39);
        &Xupdate_avx_32_79(\&body_20_39);
        &Xupdate_avx_32_79(\&body_40_59);
        &Xupdate_avx_32_79(\&body_40_59);
        &Xupdate_avx_32_79(\&body_40_59);
        &Xupdate_avx_32_79(\&body_40_59);
        &Xupdate_avx_32_79(\&body_40_59);
        &Xupdate_avx_32_79(\&body_20_39);
        &Xuplast_avx_80(\&body_20_39);  # can jump to "done"

                                $saved_j=$j; @saved_V=@V;

        &Xloop_avx(\&body_20_39);
        &Xloop_avx(\&body_20_39);
        &Xloop_avx(\&body_20_39);

$code.=<<___;
        add     0($ctx),$A                      # update context
        add     4($ctx),@T[0]
        add     8($ctx),$C
        add     12($ctx),$D
        mov     $A,0($ctx)
        add     16($ctx),$E
        mov     @T[0],4($ctx)
        mov     @T[0],$B                        # magic seed
        mov     $C,8($ctx)
        mov     $D,12($ctx)
        mov     $E,16($ctx)
        jmp     .Loop_avx

.align  16
.Ldone_avx:
___
                                $j=$saved_j; @V=@saved_V;

        &Xtail_avx(\&body_20_39);
        &Xtail_avx(\&body_20_39);
        &Xtail_avx(\&body_20_39);

$code.=<<___;
        vzeroupper

        add     0($ctx),$A                      # update context
        add     4($ctx),@T[0]
        add     8($ctx),$C
        mov     $A,0($ctx)
        add     12($ctx),$D
        mov     @T[0],4($ctx)
        add     16($ctx),$E
        mov     $C,8($ctx)
        mov     $D,12($ctx)
```

```
1052         mov     $E,16($ctx)
1053 ___
1054 $code.=<<___ if ($win64);
1055         movaps  64+0(%rsp),%xmm6
1056         movaps  64+16(%rsp),%xmm7
1057         movaps  64+32(%rsp),%xmm8
1058         movaps  64+48(%rsp),%xmm9
1059         movaps  64+64(%rsp),%xmm10
1060 ___
1061 $code.=<<___;
1062         lea     `64+($win64?5*16:0)`(%rsp),%rsi
1063         mov     0(%rsi),%r12
1064         mov     8(%rsi),%rbp
1065         mov     16(%rsi),%rbx
1066         lea     24(%rsi),%rsp
1067 .Lepilogue_avx:
1068         ret
1069 .size   sha1_block_data_order_avx,.-sha1_block_data_order_avx
1070 ___
1071 }
1072 $code.=<<___;
1073 .align  64
1074 K_XX_XX:
1075 .long   0x5a827999,0x5a827999,0x5a827999,0x5a827999     # K_00_19
1076 .long   0x6ed9eba1,0x6ed9eba1,0x6ed9eba1,0x6ed9eba1     # K_20_39
1077 .long   0x8f1bbcdc,0x8f1bbcdc,0x8f1bbcdc,0x8f1bbcdc     # K_40_59
1078 .long   0xca62c1d6,0xca62c1d6,0xca62c1d6,0xca62c1d6     # K_60_79
1079 .long   0x00010203,0x04050607,0x08090a0b,0x0c0d0e0f     # pbswap mask
1080 ___
1081 }}}
1082 $code.=<<___;
1083 .asciz  "SHA1 block transform for x86_64, CRYPTOGAMS by <appro\@openssl.org>"
1084 .align  64
1085 ___
1086
1087 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
1088 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
1089 if ($win64) {
1090 $rec="%rcx";
1091 $frame="%rdx";
1092 $context="%r8";
1093 $disp="%r9";
1094
1095 $code.=<<___;
1096 .extern __imp_RtlVirtualUnwind
1097 .type   se_handler,\@abi-omnipotent
1098 .align  16
1099 se_handler:
1100         push    %rsi
1101         push    %rdi
1102         push    %rbx
1103         push    %rbp
1104         push    %r12
1105         push    %r13
1106         push    %r14
1107         push    %r15
1108         pushfq
1109         sub     \$64,%rsp
1110
1111         mov     120($context),%rax      # pull context->Rax
1112         mov     248($context),%rbx      # pull context->Rip
1113
1114         lea     .Lprologue(%rip),%r10
1115         cmp     %r10,%rbx               # context->Rip<.Lprologue
1116         jb      .Lcommon_seh_tail
```



```
1118         mov     152($context),%rax      # pull context->Rsp
1119
1120         lea     .Lepilogue(%rip),%r10
1121         cmp     %r10,%rbx               # context->Rip>=.Lepilogue
1122         jae     .Lcommon_seh_tail
1123
1124         mov     `16*4`(%rax),%rax       # pull saved stack pointer
1125         lea     32(%rax),%rax
1126
1127         mov     -8(%rax),%rbx
1128         mov     -16(%rax),%rbp
1129         mov     -24(%rax),%r12
1130         mov     -32(%rax),%r13
1131         mov     %rbx,144($context)      # restore context->Rbx
1132         mov     %rbp,160($context)      # restore context->Rbp
1133         mov     %r12,216($context)      # restore context->R12
1134         mov     %r13,224($context)      # restore context->R13
1135
1136         jmp     .Lcommon_seh_tail
1137 .size   se_handler,.-se_handler
1138
1139 .type   ssse3_handler,\@abi-omnipotent
1140 .align  16
1141 ssse3_handler:
1142         push    %rsi
1143         push    %rdi
1144         push    %rbx
1145         push    %rbp
1146         push    %r12
1147         push    %r13
1148         push    %r14
1149         push    %r15
1150         pushfq
1151         sub     \$64,%rsp
1152
1153         mov     120($context),%rax      # pull context->Rax
1154         mov     248($context),%rbx      # pull context->Rip
1155
1156         mov     8($disp),%rsi           # disp->ImageBase
1157         mov     56($disp),%r11          # disp->HandlerData
1158
1159         mov     0(%r11),%r10d           # HandlerData[0]
1160         lea     (%rsi,%r10),%r10        # prologue label
1161         cmp     %r10,%rbx               # context->Rip<prologue label
1162         jb      .Lcommon_seh_tail
1163
1164         mov     152($context),%rax      # pull context->Rsp
1165
1166         mov     4(%r11),%r10d           # HandlerData[1]
1167         lea     (%rsi,%r10),%r10        # epilogue label
1168         cmp     %r10,%rbx               # context->Rip>=epilogue label
1169         jae     .Lcommon_seh_tail
1170
1171         lea     64(%rax),%rsi
1172         lea     512($context),%rdi      # &context.Xmm6
1173         mov     \$10,%ecx
1174         .long   0xa548f3fc              # cld; rep movsq
1175         lea     `24+64+5*16`(%rax),%rax # adjust stack pointer
1176
1177         mov     -8(%rax),%rbx
1178         mov     -16(%rax),%rbp
1179         mov     -24(%rax),%r12
1180         mov     %rbx,144($context)      # restore context->Rbx
1181         mov     %rbp,160($context)      # restore context->Rbp
1182         mov     %r12,216($context)      # restore cotnext->R12
```

```
.Lcommon_seh_tail:
        mov     8(%rax),%rdi
        mov     16(%rax),%rsi
        mov     %rax,152($context)      # restore context->Rsp
        mov     %rsi,168($context)      # restore context->Rsi
        mov     %rdi,176($context)      # restore context->Rdi

        mov     40($disp),%rdi          # disp->ContextRecord
        mov     $context,%rsi           # context
        mov     \$154,%ecx              # sizeof(CONTEXT)
        .long   0xa548f3fc              # cld; rep movsq

        mov     $disp,%rsi
        xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
        mov     8(%rsi),%rdx            # arg2, disp->ImageBase
        mov     0(%rsi),%r8             # arg3, disp->ControlPc
        mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
        mov     40(%rsi),%r10           # disp->ContextRecord
        lea     56(%rsi),%r11           # &disp->HandlerData
        lea     24(%rsi),%r12           # &disp->EstablisherFrame
        mov     %r10,32(%rsp)           # arg5
        mov     %r11,40(%rsp)           # arg6
        mov     %r12,48(%rsp)           # arg7
        mov     %rcx,56(%rsp)           # arg8, (NULL)
        call    *__imp_RtlVirtualUnwind(%rip)

        mov     \$1,%eax                # ExceptionContinueSearch
        add     \$64,%rsp
        popfq
        pop     %r15
        pop     %r14
        pop     %r13
        pop     %r12
        pop     %rbp
        pop     %rbx
        pop     %rdi
        pop     %rsi
        ret
.size   ssse3_handler,.-ssse3_handler

.section        .pdata
.align  4
        .rva    .LSEH_begin_sha1_block_data_order
        .rva    .LSEH_end_sha1_block_data_order
        .rva    .LSEH_info_sha1_block_data_order
        .rva    .LSEH_begin_sha1_block_data_order_ssse3
        .rva    .LSEH_end_sha1_block_data_order_ssse3
        .rva    .LSEH_info_sha1_block_data_order_ssse3
___
$code.=<<___ if ($avx);
        .rva    .LSEH_begin_sha1_block_data_order_avx
        .rva    .LSEH_end_sha1_block_data_order_avx
        .rva    .LSEH_info_sha1_block_data_order_avx
___
$code.=<<___;
.section        .xdata
.align  8
.LSEH_info_sha1_block_data_order:
        .byte   9,0,0,0
        .rva    se_handler
.LSEH_info_sha1_block_data_order_ssse3:
        .byte   9,0,0,0
        .rva    ssse3_handler
        .rva    .Lprologue_ssse3,.Lepilogue_ssse3       # HandlerData[]
___
$code.=<<___ if ($avx);
```



```
.LSEH_info_sha1_block_data_order_avx:
        .byte   9,0,0,0
        .rva    ssse3_handler
        .rva    .Lprologue_avx,.Lepilogue_avx           # HandlerData[]
___
}

########################################################################

$code =~ s/\`([^\`]*)\`/eval $1/gem;
print $code;
close STDOUT;
#endif /* ! codereview */
```

```
**********************************************************
    6974 Wed Aug 13 19:53:09 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/sha256-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# SHA256 block transform for x86. September 2007.
#
# Performance in clock cycles per processed byte (less is better):
#
#               Pentium PIII    P4      AMD K8  Core2
# gcc           46      36      41      27      26
# icc           57      33      38      25      23
# x86 asm       40      30      33      20      18
# x86_64 asm(*) -       -       21      16      16
#
# (*) x86_64 assembler performance is presented for reference
#     purposes.
#
# Performance improvement over compiler generated code varies from
# 10% to 40% [see above]. Not very impressive on some µ-archs, but
# it's 5 times smaller and optimizies amount of writes.

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],"sha512-586.pl",$ARGV[$#ARGV] eq "386");

$A="eax";
$E="edx";
$T="ebx";
$Aoff=&DWP(0,"esp");
$Boff=&DWP(4,"esp");
$Coff=&DWP(8,"esp");
$Doff=&DWP(12,"esp");
$Eoff=&DWP(16,"esp");
$Foff=&DWP(20,"esp");
$Goff=&DWP(24,"esp");
$Hoff=&DWP(28,"esp");
$Xoff=&DWP(32,"esp");
$K256="ebp";

sub BODY_00_15() {
    my $in_16_63=shift;

        &mov    ("ecx",$E);
         &add   ($T,"edi")                      if ($in_16_63); # T += sigma1(X[
        &ror    ("ecx",25-11);
         &mov   ("esi",$Foff);
        &xor    ("ecx",$E);
        &ror    ("ecx",11-6);
         &mov   (&DWP(4*(8+15),"esp"),$T)       if ($in_16_63); # save X[0]
        &xor    ("ecx",$E);
        &ror    ("ecx",6);      # Sigma1(e)
         &mov   ("edi",$Goff);
        &add    ($T,"ecx");     # T += Sigma1(e)
        &xor    ("esi","edi");
         &mov   ($Eoff,$E);     # modulo-scheduled
         &mov   ("ecx",$A);
        &and    ("esi",$E);
         &mov   ($E,$Doff);     # e becomes d, which is e in next iteration
        &xor    ("esi","edi");  # Ch(e,f,g)
         &mov   ("edi",$A);
        &add    ($T,"esi");     # T += Ch(e,f,g)

        &ror    ("ecx",22-13);
         &add   ($T,$Hoff);     # T += h
        &xor    ("ecx",$A);
        &ror    ("ecx",13-2);
         &mov   ("esi",$Boff);
        &xor    ("ecx",$A);
        &ror    ("ecx",2);      # Sigma0(a)
         &add   ($E,$T);        # d += T
         &mov   ("edi",$Coff);

        &add    ($T,"ecx");     # T += Sigma0(a)
         &mov   ($Aoff,$A);     # modulo-scheduled

        &mov    ("ecx",$A);
         &sub   ("esp",4);
        &or     ($A,"esi");     # a becomes h, which is a in next iteration
        &and    ("ecx","esi");
        &and    ($A,"edi");
         &mov   ("esi",&DWP(0,$K256));
        &or     ($A,"ecx");     # h=Maj(a,b,c)

        &add    ($K256,4);
        &add    ($A,$T);        # h += T
         &mov   ($T,&DWP(4*(8+15+16-1),"esp"))  if ($in_16_63); # preload T
        &add    ($E,"esi");     # d += K256[i]
        &add    ($A,"esi");     # h += K256[i]
}

&function_begin("sha256_block_data_order");
        &mov    ("esi",wparam(0));      # ctx
        &mov    ("edi",wparam(1));      # inp
        &mov    ("eax",wparam(2));      # num
        &mov    ("ebx","esp");          # saved sp

        &call   (&label("pic_point"));  # make it PIC!
&set_label("pic_point");
        &blindpop($K256);
        &lea    ($K256,&DWP(&label("K256")."-".&label("pic_point"),$K256));

        &sub    ("esp",16);
        &and    ("esp",-64);

        &shl    ("eax",6);
        &add    ("eax","edi");
        &mov    (&DWP(0,"esp"),"esi");  # ctx
        &mov    (&DWP(4,"esp"),"edi");  # inp
        &mov    (&DWP(8,"esp"),"eax");  # inp+num*128
        &mov    (&DWP(12,"esp"),"ebx"); # saved sp

&set_label("loop",16);
    # copy input block to stack reversing byte and dword order
    for($i=0;$i<4;$i++) {
        &mov    ("eax",&DWP($i*16+0,"edi"));
        &mov    ("ebx",&DWP($i*16+4,"edi"));
        &mov    ("ecx",&DWP($i*16+8,"edi"));
        &mov    ("edx",&DWP($i*16+12,"edi"));
        &bswap  ("eax");
```

```
128         &bswap  ("ebx");
129         &bswap  ("ecx");
130         &bswap  ("edx");
131         &push   ("eax");
132         &push   ("ebx");
133         &push   ("ecx");
134         &push   ("edx");
135     }
136         &add    ("edi",64);
137         &sub    ("esp",4*8);            # place for A,B,C,D,E,F,G,H
138         &mov    (&DWP(4*(8+16)+4,"esp"),"edi");

140         # copy ctx->h[0-7] to A,B,C,D,E,F,G,H on stack
141         &mov    ($A,&DWP(0,"esi"));
142         &mov    ("ebx",&DWP(4,"esi"));
143         &mov    ("ecx",&DWP(8,"esi"));
144         &mov    ("edi",&DWP(12,"esi"));
145         # &mov  ($Aoff,$A);
146         &mov    ($Boff,"ebx");
147         &mov    ($Coff,"ecx");
148         &mov    ($Doff,"edi");
149         &mov    ($E,&DWP(16,"esi"));
150         &mov    ("ebx",&DWP(20,"esi"));
151         &mov    ("ecx",&DWP(24,"esi"));
152         &mov    ("edi",&DWP(28,"esi"));
153         # &mov  ($Eoff,$E);
154         &mov    ($Foff,"ebx");
155         &mov    ($Goff,"ecx");
156         &mov    ($Hoff,"edi");

158 &set_label("00_15",16);
159         &mov    ($T,&DWP(4*(8+15),"esp"));

161         &BODY_00_15();

163         &cmp    ("esi",0xc19bf174);
164         &jne    (&label("00_15"));

166         &mov    ($T,&DWP(4*(8+15+16-1),"esp")); # preloaded in BODY_00_15(1)
167 &set_label("16_63",16);
168         &mov    ("esi",$T);
169          &mov   ("ecx",&DWP(4*(8+15+16-14),"esp"));
170         &ror    ("esi",18-7);
171          &mov   ("edi","ecx");
172         &xor    ("esi",$T);
173         &ror    ("esi",7);
174         &shr    ($T,3);

176         &ror    ("edi",19-17);
177          &xor   ($T,"esi");                     # T = sigma0(X[-15])
178         &xor    ("edi","ecx");
179         &ror    ("edi",17);
180         &shr    ("ecx",10);
181          &add   ($T,&DWP(4*(8+15+16),"esp"));   # T += X[-16]
182         &xor    ("edi","ecx");                  # sigma1(X[-2])

184          &add   ($T,&DWP(4*(8+15+16-9),"esp")); # T += X[-7]
185         # &add  ($T,"edi");                     # T += sigma1(X[-2])
186         # &mov  (&DWP(4*(8+15),"esp"),$T);      # save X[0]

188         &BODY_00_15(1);

190         &cmp    ("esi",0xc67178f2);
191         &jne    (&label("16_63"));

193         &mov    ("esi",&DWP(4*(8+16+64)+0,"esp"));#ctx
```



```
194         # &mov  ($A,$Aoff);
195         &mov    ("ebx",$Boff);
196         &mov    ("ecx",$Coff);
197         &mov    ("edi",$Doff);
198         &add    ($A,&DWP(0,"esi"));
199         &add    ("ebx",&DWP(4,"esi"));
200         &add    ("ecx",&DWP(8,"esi"));
201         &add    ("edi",&DWP(12,"esi"));
202         &mov    (&DWP(0,"esi"),$A);
203         &mov    (&DWP(4,"esi"),"ebx");
204         &mov    (&DWP(8,"esi"),"ecx");
205         &mov    (&DWP(12,"esi"),"edi");
206         # &mov  ($E,$Eoff);
207         &mov    ("eax",$Foff);
208         &mov    ("ebx",$Goff);
209         &mov    ("ecx",$Hoff);
210         &mov    ("edi",&DWP(4*(8+16+64)+4,"esp"));#inp
211         &add    ($E,&DWP(16,"esi"));
212         &add    ("eax",&DWP(20,"esi"));
213         &add    ("ebx",&DWP(24,"esi"));
214         &add    ("ecx",&DWP(28,"esi"));
215         &mov    (&DWP(16,"esi"),$E);
216         &mov    (&DWP(20,"esi"),"eax");
217         &mov    (&DWP(24,"esi"),"ebx");
218         &mov    (&DWP(28,"esi"),"ecx");

220         &add    ("esp",4*(8+16+64));            # destroy frame
221         &sub    ($K256,4*64);                   # rewind K

223         &cmp    ("edi",&DWP(8,"esp"));          # are we done yet?
224         &jb     (&label("loop"));

226         &mov    ("esp",&DWP(12,"esp"));         # restore sp
227 &function_end_A();

229 &set_label("K256",64);  # Yes! I keep it in the code segment!
230         &data_word(0x428a2f98,0x71374491,0xb5c0fbcf,0xe9b5dba5);
231         &data_word(0x3956c25b,0x59f111f1,0x923f82a4,0xab1c5ed5);
232         &data_word(0xd807aa98,0x12835b01,0x243185be,0x550c7dc3);
233         &data_word(0x72be5d74,0x80deb1fe,0x9bdc06a7,0xc19bf174);
234         &data_word(0xe49b69c1,0xefbe4786,0x0fc19dc6,0x240ca1cc);
235         &data_word(0x2de92c6f,0x4a7484aa,0x5cb0a9dc,0x76f988da);
236         &data_word(0x983e5152,0xa831c66d,0xb00327c8,0xbf597fc7);
237         &data_word(0xc6e00bf3,0xd5a79147,0x06ca6351,0x14292967);
238         &data_word(0x27b70a85,0x2e1b2138,0x4d2c6dfc,0x53380d13);
239         &data_word(0x650a7354,0x766a0abb,0x81c2c92e,0x92722c85);
240         &data_word(0xa2bfe8a1,0xa81a664b,0xc24b8b70,0xc76c51a3);
241         &data_word(0xd192e819,0xd6990624,0xf40e3585,0x106aa070);
242         &data_word(0x19a4c116,0x1e376c08,0x2748774c,0x34b0bcb5);
243         &data_word(0x391c0cb3,0x4ed8aa4a,0x5b9cca4f,0x682e6ff3);
244         &data_word(0x748f82ee,0x78a5636f,0x84c87814,0x8cc70208);
245         &data_word(0x90befffa,0xa4506ceb,0xbef9a3f7,0xc67178f2);
246 &function_end_B("sha256_block_data_order");
247 &asciz("SHA256 block transform for x86, CRYPTOGAMS by <appro\@openssl.org>");

249 &asm_finish();
250 #endif /* ! codereview */
```

```
**********************************************************
   18165 Wed Aug 13 19:53:09 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/sha512-586.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# SHA512 block transform for x86. September 2007.
#
# Performance in clock cycles per processed byte (less is better):
#
#               Pentium PIII    P4      AMD K8  Core2
# gcc           100     75      116     54      66
# icc           97      77      95      55      57
# x86 asm       61      56      82      36      40
# SSE2 asm      -       -       38      24      20
# x86_64 asm(*) -       -       30      10.0    10.5
#
# (*) x86_64 assembler performance is presented for reference
#     purposes.
#
# IALU code-path is optimized for elder Pentiums. On vanilla Pentium
# performance improvement over compiler generated code reaches ~60%,
# while on PIII - ~35%. On newer µ-archs improvement varies from 15%
# to 50%, but it's less important as they are expected to execute SSE2
# code-path, which is commonly ~2-3x faster [than compiler generated
# code]. SSE2 code-path is as fast as original sha512-sse2.pl, even
# though it does not use 128-bit operations. The latter means that
# SSE2-aware kernel is no longer required to execute the code. Another
# difference is that new code optimizes amount of writes, but at the
# cost of increased data cache "footprint" by 1/2KB.

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],"sha512-586.pl",$ARGV[$#ARGV] eq "386");

$sse2=0;
for (@ARGV) { $sse2=1 if (/-DOPENSSL_IA32_SSE2/); }

&external_label("OPENSSL_ia32cap_P") if ($sse2);

$Tlo=&DWP(0,"esp");     $Thi=&DWP(4,"esp");
$Alo=&DWP(8,"esp");     $Ahi=&DWP(8+4,"esp");
$Blo=&DWP(16,"esp");    $Bhi=&DWP(16+4,"esp");
$Clo=&DWP(24,"esp");    $Chi=&DWP(24+4,"esp");
$Dlo=&DWP(32,"esp");    $Dhi=&DWP(32+4,"esp");
$Elo=&DWP(40,"esp");    $Ehi=&DWP(40+4,"esp");
$Flo=&DWP(48,"esp");    $Fhi=&DWP(48+4,"esp");
$Glo=&DWP(56,"esp");    $Ghi=&DWP(56+4,"esp");
$Hlo=&DWP(64,"esp");    $Hhi=&DWP(64+4,"esp");
$K512="ebp";

$Asse2=&QWP(0,"esp");
$Bsse2=&QWP(8,"esp");
$Csse2=&QWP(16,"esp");
$Dsse2=&QWP(24,"esp");
$Esse2=&QWP(32,"esp");
```



```
$Fsse2=&QWP(40,"esp");
$Gsse2=&QWP(48,"esp");
$Hsse2=&QWP(56,"esp");

$A="mm0";       # B-D and
$E="mm4";       # F-H are commonly loaded to respectively mm1-mm3 and
                # mm5-mm7, but it's done on on-demand basis...

sub BODY_00_15_sse2 {
    my $prefetch=shift;

        &movq   ("mm5",$Fsse2);                 # load f
        &movq   ("mm6",$Gsse2);                 # load g
        &movq   ("mm7",$Hsse2);                 # load h

        &movq   ("mm1",$E);                     # %mm1 is sliding right
        &movq   ("mm2",$E);                     # %mm2 is sliding left
        &psrlq  ("mm1",14);
        &movq   ($Esse2,$E);                    # modulo-scheduled save e
        &psllq  ("mm2",23);
        &movq   ("mm3","mm1");                  # %mm3 is T1
        &psrlq  ("mm1",4);
        &pxor   ("mm3","mm2");
        &psllq  ("mm2",23);
        &pxor   ("mm3","mm1");
        &psrlq  ("mm1",23);
        &pxor   ("mm3","mm2");
        &psllq  ("mm2",4);
        &pxor   ("mm3","mm1");
        &paddq  ("mm7",QWP(0,$K512));           # h+=K512[i]
        &pxor   ("mm3","mm2");                  # T1=Sigma1_512(e)

        &pxor   ("mm5","mm6");                  # f^=g
        &movq   ("mm1",$Bsse2);                 # load b
        &pand   ("mm5",$E);                     # f&=e
        &movq   ("mm2",$Csse2);                 # load c
        &pxor   ("mm5","mm6");                  # f^=g
        &movq   ($E,$Dsse2);                    # e = load d
        &paddq  ("mm3","mm5");                  # T1+=Ch(e,f,g)
        &movq   (&QWP(0,"esp"),$A);             # modulo-scheduled save a
        &paddq  ("mm3","mm7");                  # T1+=h

        &movq   ("mm5",$A);                     # %mm5 is sliding right
        &movq   ("mm6",$A);                     # %mm6 is sliding left
        &paddq  ("mm3",&QWP(8*9,"esp"));        # T1+=X[0]
        &psrlq  ("mm5",28);
        &paddq  ($E,"mm3");                     # e += T1
        &psllq  ("mm6",25);
        &movq   ("mm7","mm5");                  # %mm7 is T2
        &psrlq  ("mm5",6);
        &pxor   ("mm7","mm6");
        &psllq  ("mm6",5);
        &pxor   ("mm7","mm5");
        &psrlq  ("mm5",5);
        &pxor   ("mm7","mm6");
        &psllq  ("mm6",6);
        &pxor   ("mm7","mm5");
        &sub    ("esp",8);
        &pxor   ("mm7","mm6");                  # T2=Sigma0_512(a)

        &movq   ("mm5",$A);                     # %mm5=a
        &por    ($A,"mm2");                     # a=a|c
        &movq   ("mm6",&QWP(8*(9+16-14),"esp")) if ($prefetch);
        &pand   ("mm5","mm2");                  # %mm5=a&c
        &pand   ($A,"mm1");                     # a=(a|c)&b
        &movq   ("mm2",&QWP(8*(9+16-1),"esp"))  if ($prefetch);
```

```
128         &por    ("mm5",$A);                     # %mm5=(a&c)|((a|c)&b)
129         &paddq  ("mm7","mm5");                  # T2+=Maj(a,b,c)
130         &movq   ($A,"mm3");                     # a=T1

132         &mov    (&LB("edx"),&BP(0,$K512));
133         &paddq  ($A,"mm7");                     # a+=T2
134         &add    ($K512,8);
135 }

137 sub BODY_00_15_x86 {
138         #define Sigma1(x)       (ROTR((x),14) ^ ROTR((x),18)  ^ ROTR((x),41))
139         #       LO              lo>>14^hi<<18 ^ lo>>18^hi<<14 ^ hi>>9^lo<<23
140         #       HI              hi>>14^lo<<18 ^ hi>>18^lo<<14 ^ lo>>9^hi<<23
141         &mov    ("ecx",$Elo);
142         &mov    ("edx",$Ehi);
143         &mov    ("esi","ecx");

145         &shr    ("ecx",9);      # lo>>9
146         &mov    ("edi","edx");
147         &shr    ("edx",9);      # hi>>9
148         &mov    ("ebx","ecx");
149         &shl    ("esi",14);     # lo<<14
150         &mov    ("eax","edx");
151         &shl    ("edi",14);     # hi<<14
152         &xor    ("ebx","esi");

154         &shr    ("ecx",14-9);   # lo>>14
155         &xor    ("eax","edi");
156         &shr    ("edx",14-9);   # hi>>14
157         &xor    ("eax","ecx");
158         &shl    ("esi",18-14);  # lo<<18
159         &xor    ("ebx","edx");
160         &shl    ("edi",18-14);  # hi<<18
161         &xor    ("ebx","esi");

163         &shr    ("ecx",18-14);  # lo>>18
164         &xor    ("eax","edi");
165         &shr    ("edx",18-14);  # hi>>18
166         &xor    ("eax","ecx");
167         &shl    ("esi",23-18);  # lo<<23
168         &xor    ("ebx","edx");
169         &shl    ("edi",23-18);  # hi<<23
170         &xor    ("eax","esi");
171         &xor    ("ebx","edi");                  # T1 = Sigma1(e)

173         &mov    ("ecx",$Flo);
174         &mov    ("edx",$Fhi);
175         &mov    ("esi",$Glo);
176         &mov    ("edi",$Ghi);
177          &add   ("eax",$Hlo);
178          &adc   ("ebx",$Hhi);                   # T1 += h
179         &xor    ("ecx","esi");
180         &xor    ("edx","edi");
181         &and    ("ecx",$Elo);
182         &and    ("edx",$Ehi);
183          &add   ("eax",&DWP(8*(9+15)+0,"esp"));
184          &adc   ("ebx",&DWP(8*(9+15)+4,"esp")); # T1 += X[0]
185         &xor    ("ecx","esi");
186         &xor    ("edx","edi");                  # Ch(e,f,g) = (f^g)&e)^g

188         &mov    ("esi",&DWP(0,$K512));
189         &mov    ("edi",&DWP(4,$K512));          # K[i]
190         &add    ("eax","ecx");
191         &adc    ("ebx","edx");                  # T1 += Ch(e,f,g)
192         &mov    ("ecx",$Dlo);
193         &mov    ("edx",$Dhi);
```



```
194         &add    ("eax","esi");
195         &adc    ("ebx","edi");                  # T1 += K[i]
196         &mov    ($Tlo,"eax");
197         &mov    ($Thi,"ebx");                   # put T1 away
198         &add    ("eax","ecx");
199         &adc    ("ebx","edx");                  # d += T1

201         #define Sigma0(x)       (ROTR((x),28) ^ ROTR((x),34) ^ ROTR((x),39))
202         #       LO              lo>>28^hi<<4  ^ hi>>2^lo<<30 ^ hi>>7^lo<<25
203         #       HI              hi>>28^lo<<4  ^ lo>>2^hi<<30 ^ lo>>7^hi<<25
204         &mov    ("ecx",$Alo);
205         &mov    ("edx",$Ahi);
206         &mov    ($Dlo,"eax");
207         &mov    ($Dhi,"ebx");
208         &mov    ("esi","ecx");

210         &shr    ("ecx",2);      # lo>>2
211         &mov    ("edi","edx");
212         &shr    ("edx",2);      # hi>>2
213         &mov    ("ebx","ecx");
214         &shl    ("esi",4);      # lo<<4
215         &mov    ("eax","edx");
216         &shl    ("edi",4);      # hi<<4
217         &xor    ("ebx","esi");

219         &shr    ("ecx",7-2);    # lo>>7
220         &xor    ("eax","edi");
221         &shr    ("edx",7-2);    # hi>>7
222         &xor    ("ebx","ecx");
223         &shl    ("esi",25-4);   # lo<<25
224         &xor    ("eax","edx");
225         &shl    ("edi",25-4);   # hi<<25
226         &xor    ("eax","esi");

228         &shr    ("ecx",28-7);   # lo>>28
229         &xor    ("ebx","edi");
230         &shr    ("edx",28-7);   # hi>>28
231         &xor    ("eax","ecx");
232         &shl    ("esi",30-25);  # lo<<30
233         &xor    ("ebx","edx");
234         &shl    ("edi",30-25);  # hi<<30
235         &xor    ("eax","esi");
236         &xor    ("ebx","edi");                  # Sigma0(a)

238         &mov    ("ecx",$Alo);
239         &mov    ("edx",$Ahi);
240         &mov    ("esi",$Blo);
241         &mov    ("edi",$Bhi);
242         &add    ("eax",$Tlo);
243         &adc    ("ebx",$Thi);                   # T1 = Sigma0(a)+T1
244         &or     ("ecx","esi");
245         &or     ("edx","edi");
246         &and    ("ecx",$Clo);
247         &and    ("edx",$Chi);
248         &and    ("esi",$Alo);
249         &and    ("edi",$Ahi);
250         &or     ("ecx","esi");
251         &or     ("edx","edi");                  # Maj(a,b,c) = ((a|b)&c)|(a&b)

253         &add    ("eax","ecx");
254         &adc    ("ebx","edx");                  # T1 += Maj(a,b,c)
255         &mov    ($Tlo,"eax");
256         &mov    ($Thi,"ebx");

258         &mov    (&LB("edx"),&BP(0,$K512));      # pre-fetch LSB of *K
259         &sub    ("esp",8);
```

```
260         &lea    ($K512,&DWP(8,$K512));          # K++
261 }


264 &function_begin("sha512_block_data_order");
265         &mov    ("esi",wparam(0));      # ctx
266         &mov    ("edi",wparam(1));      # inp
267         &mov    ("eax",wparam(2));      # num
268         &mov    ("ebx","esp");          # saved sp

270         &call   (&label("pic_point"));  # make it PIC!
271 &set_label("pic_point");
272         &blindpop($K512);
273         &lea    ($K512,&DWP(&label("K512")."-".&label("pic_point"),$K512));

275         &sub    ("esp",16);
276         &and    ("esp",-64);

278         &shl    ("eax",7);
279         &add    ("eax","edi");
280         &mov    (&DWP(0,"esp"),"esi");  # ctx
281         &mov    (&DWP(4,"esp"),"edi");  # inp
282         &mov    (&DWP(8,"esp"),"eax");  # inp+num*128
283         &mov    (&DWP(12,"esp"),"ebx"); # saved sp

285 if ($sse2) {
286         &picmeup("edx","OPENSSL_ia32cap_P",$K512,&label("K512"));
287         &bt     (&DWP(0,"edx"),26);
288         &jnc    (&label("loop_x86"));

290         # load ctx->h[0-7]
291         &movq   ($A,&QWP(0,"esi"));
292         &movq   ("mm1",&QWP(8,"esi"));
293         &movq   ("mm2",&QWP(16,"esi"));
294         &movq   ("mm3",&QWP(24,"esi"));
295         &movq   ($E,&QWP(32,"esi"));
296         &movq   ("mm5",&QWP(40,"esi"));
297         &movq   ("mm6",&QWP(48,"esi"));
298         &movq   ("mm7",&QWP(56,"esi"));
299         &sub    ("esp",8*10);

301 &set_label("loop_sse2",16);
302         # &movq ($Asse2,$A);
303         &movq   ($Bsse2,"mm1");
304         &movq   ($Csse2,"mm2");
305         &movq   ($Dsse2,"mm3");
306         # &movq ($Esse2,$E);
307         &movq   ($Fsse2,"mm5");
308         &movq   ($Gsse2,"mm6");
309         &movq   ($Hsse2,"mm7");

311         &mov    ("ecx",&DWP(0,"edi"));
312         &mov    ("edx",&DWP(4,"edi"));
313         &add    ("edi",8);
314         &bswap  ("ecx");
315         &bswap  ("edx");
316         &mov    (&DWP(8*9+4,"esp"),"ecx");
317         &mov    (&DWP(8*9+0,"esp"),"edx");

319 &set_label("00_14_sse2",16);
320         &mov    ("eax",&DWP(0,"edi"));
321         &mov    ("ebx",&DWP(4,"edi"));
322         &add    ("edi",8);
323         &bswap  ("eax");
324         &bswap  ("ebx");
325         &mov    (&DWP(8*8+4,"esp"),"eax");
```



```
326         &mov    (&DWP(8*8+0,"esp"),"ebx");

328         &BODY_00_15_sse2();

330         &cmp    (&LB("edx"),0x35);
331         &jne    (&label("00_14_sse2"));

333         &BODY_00_15_sse2(1);

335 &set_label("16_79_sse2",16);
336         #&movq  ("mm2",&QWP(8*(9+16-1),"esp")); #prefetched in BODY_00_15
337         #&movq  ("mm6",&QWP(8*(9+16-14),"esp"));
338         &movq   ("mm1","mm2");

340         &psrlq  ("mm2",1);
341         &movq   ("mm7","mm6");
342         &psrlq  ("mm6",6);
343         &movq   ("mm3","mm2");

345         &psrlq  ("mm2",7-1);
346         &movq   ("mm5","mm6");
347         &psrlq  ("mm6",19-6);
348         &pxor   ("mm3","mm2");

350         &psrlq  ("mm2",8-7);
351         &pxor   ("mm5","mm6");
352         &psrlq  ("mm6",61-19);
353         &pxor   ("mm3","mm2");

355         &movq   ("mm2",&QWP(8*(9+16),"esp"));

357         &psllq  ("mm1",56);
358         &pxor   ("mm5","mm6");
359         &psllq  ("mm7",3);
360         &pxor   ("mm3","mm1");

362         &paddq  ("mm2",&QWP(8*(9+16-9),"esp"));

364         &psllq  ("mm1",63-56);
365         &pxor   ("mm5","mm7");
366         &psllq  ("mm7",45-3);
367         &pxor   ("mm3","mm1");
368         &pxor   ("mm5","mm7");

370         &paddq  ("mm3","mm5");
371         &paddq  ("mm3","mm2");
372         &movq   (&QWP(8*9,"esp"),"mm3");

374         &BODY_00_15_sse2(1);

376         &cmp    (&LB("edx"),0x17);
377         &jne    (&label("16_79_sse2"));

379         # &movq ($A,$Asse2);
380         &movq   ("mm1",$Bsse2);
381         &movq   ("mm2",$Csse2);
382         &movq   ("mm3",$Dsse2);
383         # &movq ($E,$Esse2);
384         &movq   ("mm5",$Fsse2);
385         &movq   ("mm6",$Gsse2);
386         &movq   ("mm7",$Hsse2);

388         &paddq  ($A,&QWP(0,"esi"));
389         &paddq  ("mm1",&QWP(8,"esi"));
390         &paddq  ("mm2",&QWP(16,"esi"));
391         &paddq  ("mm3",&QWP(24,"esi"));
```

```
392         &paddq  ($E,&QWP(32,"esi"));
393         &paddq  ("mm5",&QWP(40,"esi"));
394         &paddq  ("mm6",&QWP(48,"esi"));
395         &paddq  ("mm7",&QWP(56,"esi"));
396
397         &movq   (&QWP(0,"esi"),$A);
398         &movq   (&QWP(8,"esi"),"mm1");
399         &movq   (&QWP(16,"esi"),"mm2");
400         &movq   (&QWP(24,"esi"),"mm3");
401         &movq   (&QWP(32,"esi"),$E);
402         &movq   (&QWP(40,"esi"),"mm5");
403         &movq   (&QWP(48,"esi"),"mm6");
404         &movq   (&QWP(56,"esi"),"mm7");
405
406         &add    ("esp",8*80);                   # destroy frame
407         &sub    ($K512,8*80);                   # rewind K
408
409         &cmp    ("edi",&DWP(8*10+8,"esp"));     # are we done yet?
410         &jb     (&label("loop_sse2"));
411
412         &emms   ();
413         &mov    ("esp",&DWP(8*10+12,"esp"));    # restore sp
414 &function_end_A();
415 }
416 &set_label("loop_x86",16);
417     # copy input block to stack reversing byte and qword order
418     for ($i=0;$i<8;$i++) {
419         &mov    ("eax",&DWP($i*16+0,"edi"));
420         &mov    ("ebx",&DWP($i*16+4,"edi"));
421         &mov    ("ecx",&DWP($i*16+8,"edi"));
422         &mov    ("edx",&DWP($i*16+12,"edi"));
423         &bswap  ("eax");
424         &bswap  ("ebx");
425         &bswap  ("ecx");
426         &bswap  ("edx");
427         &push   ("eax");
428         &push   ("ebx");
429         &push   ("ecx");
430         &push   ("edx");
431     }
432         &add    ("edi",128);
433         &sub    ("esp",9*8);            # place for T,A,B,C,D,E,F,G,H
434         &mov    (&DWP(8*(9+16)+4,"esp"),"edi");
435
436         # copy ctx->h[0-7] to A,B,C,D,E,F,G,H on stack
437         &lea    ("edi",&DWP(8,"esp"));
438         &mov    ("ecx",16);
439         &data_word(0xA5F3F689);         # rep movsd
440
441 &set_label("00_15_x86",16);
442         &BODY_00_15_x86();
443
444         &cmp    (&LB("edx"),0x94);
445         &jne    (&label("00_15_x86"));
446
447 &set_label("16_79_x86",16);
448         #define sigma0(x)       (ROTR((x),1)  ^ ROTR((x),8)  ^ ((x)>>7))
449         #       LO              lo>>1^hi<<31  ^ lo>>8^hi<<24 ^ lo>>7^hi<<25
450         #       HI              hi>>1^lo<<31  ^ hi>>8^lo<<24 ^ hi>>7
451         &mov    ("ecx",&DWP(8*(9+15+16-1)+0,"esp"));
452         &mov    ("edx",&DWP(8*(9+15+16-1)+4,"esp"));
453         &mov    ("esi","ecx");
454
455         &shr    ("ecx",1);      # lo>>1
456         &mov    ("edi","edx");
457         &shr    ("edx",1);      # hi>>1
```



```
458         &mov    ("eax","ecx");
459         &shl    ("esi",24);     # lo<<24
460         &mov    ("ebx","edx");
461         &shl    ("edi",24);     # hi<<24
462         &xor    ("ebx","esi");
463
464         &shr    ("ecx",7-1);    # lo>>7
465         &xor    ("eax","edi");
466         &shr    ("edx",7-1);    # hi>>7
467         &xor    ("eax","ecx");
468         &shl    ("esi",31-24);  # lo<<31
469         &xor    ("ebx","edx");
470         &shl    ("edi",25-24);  # hi<<25
471         &xor    ("ebx","esi");
472
473         &shr    ("ecx",8-7);    # lo>>8
474         &xor    ("eax","edi");
475         &shr    ("edx",8-7);    # hi>>8
476         &xor    ("eax","ecx");
477         &shl    ("edi",31-25);  # hi<<31
478         &xor    ("ebx","edx");
479         &xor    ("eax","edi");                  # T1 = sigma0(X[-15])
480
481         &mov    (&DWP(0,"esp"),"eax");
482         &mov    (&DWP(4,"esp"),"ebx");          # put T1 away
483
484         #define sigma1(x)       (ROTR((x),19) ^ ROTR((x),61) ^ ((x)>>6))
485         #       LO              lo>>19^hi<<13 ^ hi>>29^lo<<3 ^ lo>>6^hi<<26
486         #       HI              hi>>19^lo<<13 ^ lo>>29^hi<<3 ^ hi>>6
487         &mov    ("ecx",&DWP(8*(9+15+16-14)+0,"esp"));
488         &mov    ("edx",&DWP(8*(9+15+16-14)+4,"esp"));
489         &mov    ("esi","ecx");
490
491         &shr    ("ecx",6);      # lo>>6
492         &mov    ("edi","edx");
493         &shr    ("edx",6);      # hi>>6
494         &mov    ("eax","ecx");
495         &shl    ("esi",3);      # lo<<3
496         &mov    ("ebx","edx");
497         &shl    ("edi",3);      # hi<<3
498         &xor    ("eax","esi");
499
500         &shr    ("ecx",19-6);   # lo>>19
501         &xor    ("ebx","edi");
502         &shr    ("edx",19-6);   # hi>>19
503         &xor    ("eax","ecx");
504         &shl    ("esi",13-3);   # lo<<13
505         &xor    ("ebx","edx");
506         &shl    ("edi",13-3);   # hi<<13
507         &xor    ("ebx","esi");
508
509         &shr    ("ecx",29-19);  # lo>>29
510         &xor    ("eax","edi");
511         &shr    ("edx",29-19);  # hi>>29
512         &xor    ("ebx","ecx");
513         &shl    ("edi",26-13);  # hi<<26
514         &xor    ("eax","edx");
515         &xor    ("eax","edi");                  # sigma1(X[-2])
516
517         &mov    ("ecx",&DWP(8*(9+15+16)+0,"esp"));
518         &mov    ("edx",&DWP(8*(9+15+16)+4,"esp"));
519         &add    ("eax",&DWP(0,"esp"));
520         &adc    ("ebx",&DWP(4,"esp"));          # T1 = sigma1(X[-2])+T1
521         &mov    ("esi",&DWP(8*(9+15+16-9)+0,"esp"));
522         &mov    ("edi",&DWP(8*(9+15+16-9)+4,"esp"));
523         &add    ("eax","ecx");
```

```
524         &adc    ("ebx","edx");                  # T1 += X[-16]
525         &add    ("eax","esi");
526         &adc    ("ebx","edi");                  # T1 += X[-7]
527         &mov    (&DWP(8*(9+15)+0,"esp"),"eax");
528         &mov    (&DWP(8*(9+15)+4,"esp"),"ebx"); # save X[0]

530         &BODY_00_15_x86();

532         &cmp    (&LB("edx"),0x17);
533         &jne    (&label("16_79_x86"));

535         &mov    ("esi",&DWP(8*(9+16+80)+0,"esp"));# ctx
536         &mov    ("edi",&DWP(8*(9+16+80)+4,"esp"));# inp
537     for($i=0;$i<4;$i++) {
538         &mov    ("eax",&DWP($i*16+0,"esi"));
539         &mov    ("ebx",&DWP($i*16+4,"esi"));
540         &mov    ("ecx",&DWP($i*16+8,"esi"));
541         &mov    ("edx",&DWP($i*16+12,"esi"));
542         &add    ("eax",&DWP(8+($i*16)+0,"esp"));
543         &adc    ("ebx",&DWP(8+($i*16)+4,"esp"));
544         &mov    (&DWP($i*16+0,"esi"),"eax");
545         &mov    (&DWP($i*16+4,"esi"),"ebx");
546         &add    ("ecx",&DWP(8+($i*16)+8,"esp"));
547         &adc    ("edx",&DWP(8+($i*16)+12,"esp"));
548         &mov    (&DWP($i*16+8,"esi"),"ecx");
549         &mov    (&DWP($i*16+12,"esi"),"edx");
550     }
551         &add    ("esp",8*(9+16+80));            # destroy frame
552         &sub    ($K512,8*80);                   # rewind K

554         &cmp    ("edi",&DWP(8,"esp"));          # are we done yet?
555         &jb     (&label("loop_x86"));

557         &mov    ("esp",&DWP(12,"esp"));         # restore sp
558 &function_end_A();

560 &set_label("K512",64);  # Yes! I keep it in the code segment!
561         &data_word(0xd728ae22,0x428a2f98);      # u64
562         &data_word(0x23ef65cd,0x71374491);      # u64
563         &data_word(0xec4d3b2f,0xb5c0fbcf);      # u64
564         &data_word(0x8189dbbc,0xe9b5dba5);      # u64
565         &data_word(0xf348b538,0x3956c25b);      # u64
566         &data_word(0xb605d019,0x59f111f1);      # u64
567         &data_word(0xaf194f9b,0x923f82a4);      # u64
568         &data_word(0xda6d8118,0xab1c5ed5);      # u64
569         &data_word(0xa3030242,0xd807aa98);      # u64
570         &data_word(0x45706fbe,0x12835b01);      # u64
571         &data_word(0x4ee4b28c,0x243185be);      # u64
572         &data_word(0xd5ffb4e2,0x550c7dc3);      # u64
573         &data_word(0xf27b896f,0x72be5d74);      # u64
574         &data_word(0x3b1696b1,0x80deb1fe);      # u64
575         &data_word(0x25c71235,0x9bdc06a7);      # u64
576         &data_word(0xcf692694,0xc19bf174);      # u64
577         &data_word(0x9ef14ad2,0xe49b69c1);      # u64
578         &data_word(0x384f25e3,0xefbe4786);      # u64
579         &data_word(0x8b8cd5b5,0x0fc19dc6);      # u64
580         &data_word(0x77ac9c65,0x240ca1cc);      # u64
581         &data_word(0x592b0275,0x2de92c6f);      # u64
582         &data_word(0x6ea6e483,0x4a7484aa);      # u64
583         &data_word(0xbd41fbd4,0x5cb0a9dc);      # u64
584         &data_word(0x831153b5,0x76f988da);      # u64
585         &data_word(0xee66dfab,0x983e5152);      # u64
586         &data_word(0x2db43210,0xa831c66d);      # u64
587         &data_word(0x98fb213f,0xb00327c8);      # u64
588         &data_word(0xbeef0ee4,0xbf597fc7);      # u64
589         &data_word(0x3da88fc2,0xc6e00bf3);      # u64
```



```
590         &data_word(0x930aa725,0xd5a79147);      # u64
591         &data_word(0xe003826f,0x06ca6351);      # u64
592         &data_word(0x0a0e6e70,0x14292967);      # u64
593         &data_word(0x46d22ffc,0x27b70a85);      # u64
594         &data_word(0x5c26c926,0x2e1b2138);      # u64
595         &data_word(0x5ac42aed,0x4d2c6dfc);      # u64
596         &data_word(0x9d95b3df,0x53380d13);      # u64
597         &data_word(0x8baf63de,0x650a7354);      # u64
598         &data_word(0x3c77b2a8,0x766a0abb);      # u64
599         &data_word(0x47edaee6,0x81c2c92e);      # u64
600         &data_word(0x1482353b,0x92722c85);      # u64
601         &data_word(0x4cf10364,0xa2bfe8a1);      # u64
602         &data_word(0xbc423001,0xa81a664b);      # u64
603         &data_word(0xd0f89791,0xc24b8b70);      # u64
604         &data_word(0x0654be30,0xc76c51a3);      # u64
605         &data_word(0xd6ef5218,0xd192e819);      # u64
606         &data_word(0x5565a910,0xd6990624);      # u64
607         &data_word(0x5771202a,0xf40e3585);      # u64
608         &data_word(0x32bbd1b8,0x106aa070);      # u64
609         &data_word(0xb8d2d0c8,0x19a4c116);      # u64
610         &data_word(0x5141ab53,0x1e376c08);      # u64
611         &data_word(0xdf8eeb99,0x2748774c);      # u64
612         &data_word(0xe19b48a8,0x34b0bcb5);      # u64
613         &data_word(0xc5c95a63,0x391c0cb3);      # u64
614         &data_word(0xe3418acb,0x4ed8aa4a);      # u64
615         &data_word(0x7763e373,0x5b9cca4f);      # u64
616         &data_word(0xd6b2b8a3,0x682e6ff3);      # u64
617         &data_word(0x5defb2fc,0x748f82ee);      # u64
618         &data_word(0x43172f60,0x78a5636f);      # u64
619         &data_word(0xa1f0ab72,0x84c87814);      # u64
620         &data_word(0x1a6439ec,0x8cc70208);      # u64
621         &data_word(0x23631e28,0x90befffa);      # u64
622         &data_word(0xde82bde9,0xa4506ceb);      # u64
623         &data_word(0xb2c67915,0xbef9a3f7);      # u64
624         &data_word(0xe372532b,0xc67178f2);      # u64
625         &data_word(0xea26619c,0xca273ece);      # u64
626         &data_word(0x21c0c207,0xd186b8c7);      # u64
627         &data_word(0xcde0eb1e,0xeada7dd6);      # u64
628         &data_word(0xee6ed178,0xf57d4f7f);      # u64
629         &data_word(0x72176fba,0x06f067aa);      # u64
630         &data_word(0xa2c898a6,0x0a637dc5);      # u64
631         &data_word(0xbef90dae,0x113f9804);      # u64
632         &data_word(0x131c471b,0x1b710b35);      # u64
633         &data_word(0x23047d84,0x28db77f5);      # u64
634         &data_word(0x40c72493,0x32caab7b);      # u64
635         &data_word(0x15c9bebc,0x3c9ebe0a);      # u64
636         &data_word(0x9c100d4c,0x431d67c4);      # u64
637         &data_word(0xcb3e42b6,0x4cc5d4be);      # u64
638         &data_word(0xfc657e2a,0x597f299c);      # u64
639         &data_word(0x3ad6faec,0x5fcb6fab);      # u64
640         &data_word(0x4a475817,0x6c44198c);      # u64
641 &function_end_B("sha512_block_data_order");
642 &asciz("SHA512 block transform for x86, CRYPTOGAMS by <appro\@openssl.org>");

644 &asm_finish();
645 #endif /* ! codereview */
```

```
**********************************************************
   11902 Wed Aug 13 19:53:10 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/sha512-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. Rights for redistribution and usage in source and binary
# forms are granted according to the OpenSSL license.
# ====================================================================
#
# sha256/512_block procedure for x86_64.
#
# 40% improvement over compiler-generated code on Opteron. On EM64T
# sha256 was observed to run >80% faster and sha512 - >40%. No magical
# tricks, just straight implementation... I really wonder why gcc
# [being armed with inline assembler] fails to generate as fast code.
# The only thing which is cool about this module is that it's very
# same instruction sequence used for both SHA-256 and SHA-512. In
# former case the instructions operate on 32-bit operands, while in
# latter - on 64-bit ones. All I had to do is to get one flavor right,
# the other one passed the test right away:-)
#
# sha256_block runs in ~1005 cycles on Opteron, which gives you
# asymptotic performance of 64*1000/1005=63.7MBps times CPU clock
# frequency in GHz. sha512_block runs in ~1275 cycles, which results
# in 128*1000/1275=100MBps per GHz. Is there room for improvement?
# Well, if you compare it to IA-64 implementation, which maintains
# X[16] in register bank[!], tends to 4 instructions per CPU clock
# cycle and runs in 1003 cycles, 1275 is very good result for 3-way
# issue Opteron pipeline and X[16] maintained in memory. So that *if*
# there is a way to improve it, *then* the only way would be to try to
# offload X[16] updates to SSE unit, but that would require "deeper"
# loop unroll, which in turn would naturally cause size blow-up, not
# to mention increased complexity! And once again, only *if* it's
# actually possible to noticeably improve overall ILP, instruction
# level parallelism, on a given CPU implementation in this case.
#
# Special note on Intel EM64T. While Opteron CPU exhibits perfect
# perfromance ratio of 1.5 between 64- and 32-bit flavors [see above],
# [currently available] EM64T CPUs apparently are far from it. On the
# contrary, 64-bit version, sha512_block, is ~30% *slower* than 32-bit
# sha256_block:-( This is presumably because 64-bit shifts/rotates
# apparently are not atomic instructions, but implemented in microcode.

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

if ($output =~ /512/) {
        $func="sha512_block_data_order";
        $TABLE="K512";
        $SZ=8;
        @ROT=($A,$B,$C,$D,$E,$F,$G,$H)=("%rax","%rbx","%rcx","%rdx",
```



```
                                        "%r8", "%r9", "%r10","%r11");
        ($T1,$a0,$a1,$a2)=("%r12","%r13","%r14","%r15");
        @Sigma0=(28,34,39);
        @Sigma1=(14,18,41);
        @sigma0=(1,  8, 7);
        @sigma1=(19,61, 6);
        $rounds=80;
} else {
        $func="sha256_block_data_order";
        $TABLE="K256";
        $SZ=4;
        @ROT=($A,$B,$C,$D,$E,$F,$G,$H)=("%eax","%ebx","%ecx","%edx",
                                        "%r8d","%r9d","%r10d","%r11d");
        ($T1,$a0,$a1,$a2)=("%r12d","%r13d","%r14d","%r15d");
        @Sigma0=( 2,13,22);
        @Sigma1=( 6,11,25);
        @sigma0=( 7,18, 3);
        @sigma1=(17,19,10);
        $rounds=64;
}

$ctx="%rdi";    # 1st arg
$round="%rdi";  # zaps $ctx
$inp="%rsi";    # 2nd arg
$Tbl="%rbp";

$_ctx="16*$SZ+0*8(%rsp)";
$_inp="16*$SZ+1*8(%rsp)";
$_end="16*$SZ+2*8(%rsp)";
$_rsp="16*$SZ+3*8(%rsp)";
$framesz="16*$SZ+4*8";


sub ROUND_00_15()
{ my ($i,$a,$b,$c,$d,$e,$f,$g,$h) = @_;

$code.=<<___;
        ror     \$'$Sigma1[2]-$Sigma1[1]',$a0
        mov     $f,$a2
        mov     $T1,'$SZ*($i&0xf)'(%rsp)

        ror     \$'$Sigma0[2]-$Sigma0[1]',$a1
        xor     $e,$a0
        xor     $g,$a2                  # f^g

        ror     \$'$Sigma1[1]-$Sigma1[0]',$a0
        add     $h,$T1                  # T1+=h
        xor     $a,$a1

        add     ($Tbl,$round,$SZ),$T1   # T1+=K[round]
        and     $e,$a2                  # (f^g)&e
        mov     $b,$h

        ror     \$'$Sigma0[1]-$Sigma0[0]',$a1
        xor     $e,$a0
        xor     $g,$a2                  # Ch(e,f,g)=((f^g)&e)^g

        xor     $c,$h                   # b^c
        xor     $a,$a1
        add     $a2,$T1                 # T1+=Ch(e,f,g)
        mov     $b,$a2

        ror     \$\$Sigma1[0],$a0       # Sigma1(e)
        and     $a,$h                   # h=(b^c)&a
        and     $c,$a2                  # b&c
```

```
128         ror     \$$Sigma0[0],$a1        # Sigma0(a)
129         add     $a0,$T1                 # T1+=Sigma1(e)
130         add     $a2,$h                  # h+=b&c (completes +=Maj(a,b,c)

132         add     $T1,$d                  # d+=T1
133         add     $T1,$h                  # h+=T1
134         lea     1($round),$round        # round++
135         add     $a1,$h                  # h+=Sigma0(a)

137 ___
138 }

140 sub ROUND_16_XX()
141 { my ($i,$a,$b,$c,$d,$e,$f,$g,$h) = @_;

143 $code.=<<___;
144         mov     `$SZ*(($i+1)&0xf)`(%rsp),$a0
145         mov     `$SZ*(($i+14)&0xf)`(%rsp),$a1
146         mov     $a0,$T1
147         mov     $a1,$a2

149         ror     \$`$sigma0[1]-$sigma0[0]`,$T1
150         xor     $a0,$T1
151         shr     \$$sigma0[2],$a0

153         ror     \$$sigma0[0],$T1
154         xor     $T1,$a0                 # sigma0(X[(i+1)&0xf])
155         mov     `$SZ*(($i+9)&0xf)`(%rsp),$T1

157         ror     \$`$sigma1[1]-$sigma1[0]`,$a2
158         xor     $a1,$a2
159         shr     \$$sigma1[2],$a1

161         ror     \$$sigma1[0],$a2
162         add     $a0,$T1
163         xor     $a2,$a1                 # sigma1(X[(i+14)&0xf])

165         add     `$SZ*($i&0xf)`(%rsp),$T1
166         mov     $e,$a0
167         add     $a1,$T1
168         mov     $a,$a1
169 ___
170         &ROUND_00_15(@_);
171 }

173 $code=<<___;
174 .text

176 .globl  $func
177 .type   $func,\@function,4
178 .align  16
179 $func:
180         push    %rbx
181         push    %rbp
182         push    %r12
183         push    %r13
184         push    %r14
185         push    %r15
186         mov     %rsp,%r11               # copy %rsp
187         shl     \$4,%rdx                # num*16
188         sub     \$$framesz,%rsp
189         lea     ($inp,%rdx,$SZ),%rdx    # inp+num*16*$SZ
190         and     \$-64,%rsp              # align stack frame
191         mov     $ctx,$_ctx              # save ctx, 1st arg
192         mov     $inp,$_inp              # save inp, 2nd arh
193         mov     %rdx,$_end              # save end pointer, "3rd" arg
```



```
194         mov     %r11,$_rsp              # save copy of %rsp
195 .Lprologue:

197         lea     $TABLE(%rip),$Tbl

199         mov     $SZ*0($ctx),$A
200         mov     $SZ*1($ctx),$B
201         mov     $SZ*2($ctx),$C
202         mov     $SZ*3($ctx),$D
203         mov     $SZ*4($ctx),$E
204         mov     $SZ*5($ctx),$F
205         mov     $SZ*6($ctx),$G
206         mov     $SZ*7($ctx),$H
207         jmp     .Lloop

209 .align  16
210 .Lloop:
211         xor     $round,$round
212 ___
213         for($i=0;$i<16;$i++) {
214                 $code.="        mov     $SZ*$i($inp),$T1\n";
215                 $code.="        mov     @ROT[4],$a0\n";
216                 $code.="        mov     @ROT[0],$a1\n";
217                 $code.="        bswap   $T1\n";
218                 &ROUND_00_15($i,@ROT);
219                 unshift(@ROT,pop(@ROT));
220         }
221 $code.=<<___;
222         jmp     .Lrounds_16_xx
223 .align  16
224 .Lrounds_16_xx:
225 ___
226         for(;$i<32;$i++) {
227                 &ROUND_16_XX($i,@ROT);
228                 unshift(@ROT,pop(@ROT));
229         }

231 $code.=<<___;
232         cmp     \$$rounds,$round
233         jb      .Lrounds_16_xx

235         mov     $_ctx,$ctx
236         lea     16*$SZ($inp),$inp

238         add     $SZ*0($ctx),$A
239         add     $SZ*1($ctx),$B
240         add     $SZ*2($ctx),$C
241         add     $SZ*3($ctx),$D
242         add     $SZ*4($ctx),$E
243         add     $SZ*5($ctx),$F
244         add     $SZ*6($ctx),$G
245         add     $SZ*7($ctx),$H

247         cmp     $_end,$inp

249         mov     $A,$SZ*0($ctx)
250         mov     $B,$SZ*1($ctx)
251         mov     $C,$SZ*2($ctx)
252         mov     $D,$SZ*3($ctx)
253         mov     $E,$SZ*4($ctx)
254         mov     $F,$SZ*5($ctx)
255         mov     $G,$SZ*6($ctx)
256         mov     $H,$SZ*7($ctx)
257         jb      .Lloop

259         mov     $_rsp,%rsi
```

```
260         mov     (%rsi),%r15
261         mov     8(%rsi),%r14
262         mov     16(%rsi),%r13
263         mov     24(%rsi),%r12
264         mov     32(%rsi),%rbp
265         mov     40(%rsi),%rbx
266         lea     48(%rsi),%rsp
267 .Lepilogue:
268         ret
269 .size   $func,.-$func
270 ___

272 if ($SZ==4) {
273 $code.=<<___;
274 .align  64
275 .type   $TABLE,\@object
276 $TABLE:
277         .long   0x428a2f98,0x71374491,0xb5c0fbcf,0xe9b5dba5
278         .long   0x3956c25b,0x59f111f1,0x923f82a4,0xab1c5ed5
279         .long   0xd807aa98,0x12835b01,0x243185be,0x550c7dc3
280         .long   0x72be5d74,0x80deb1fe,0x9bdc06a7,0xc19bf174
281         .long   0xe49b69c1,0xefbe4786,0x0fc19dc6,0x240ca1cc
282         .long   0x2de92c6f,0x4a7484aa,0x5cb0a9dc,0x76f988da
283         .long   0x983e5152,0xa831c66d,0xb00327c8,0xbf597fc7
284         .long   0xc6e00bf3,0xd5a79147,0x06ca6351,0x14292967
285         .long   0x27b70a85,0x2e1b2138,0x4d2c6dfc,0x53380d13
286         .long   0x650a7354,0x766a0abb,0x81c2c92e,0x92722c85
287         .long   0xa2bfe8a1,0xa81a664b,0xc24b8b70,0xc76c51a3
288         .long   0xd192e819,0xd6990624,0xf40e3585,0x106aa070
289         .long   0x19a4c116,0x1e376c08,0x2748774c,0x34b0bcb5
290         .long   0x391c0cb3,0x4ed8aa4a,0x5b9cca4f,0x682e6ff3
291         .long   0x748f82ee,0x78a5636f,0x84c87814,0x8cc70208
292         .long   0x90befffa,0xa4506ceb,0xbef9a3f7,0xc67178f2
293 ___
294 } else {
295 $code.=<<___;
296 .align  64
297 .type   $TABLE,\@object
298 $TABLE:
299         .quad   0x428a2f98d728ae22,0x7137449123ef65cd
300         .quad   0xb5c0fbcfec4d3b2f,0xe9b5dba58189dbbc
301         .quad   0x3956c25bf348b538,0x59f111f1b605d019
302         .quad   0x923f82a4af194f9b,0xab1c5ed5da6d8118
303         .quad   0xd807aa98a3030242,0x12835b0145706fbe
304         .quad   0x243185be4ee4b28c,0x550c7dc3d5ffb4e2
305         .quad   0x72be5d74f27b896f,0x80deb1fe3b1696b1
306         .quad   0x9bdc06a725c71235,0xc19bf174cf692694
307         .quad   0xe49b69c19ef14ad2,0xefbe4786384f25e3
308         .quad   0x0fc19dc68b8cd5b5,0x240ca1cc77ac9c65
309         .quad   0x2de92c6f592b0275,0x4a7484aa6ea6e483
310         .quad   0x5cb0a9dcbd41fbd4,0x76f988da831153b5
311         .quad   0x983e5152ee66dfab,0xa831c66d2db43210
312         .quad   0xb00327c898fb213f,0xbf597fc7beef0ee4
313         .quad   0xc6e00bf33da88fc2,0xd5a79147930aa725
314         .quad   0x06ca6351e003826f,0x142929670a0e6e70
315         .quad   0x27b70a8546d22ffc,0x2e1b21385c26c926
316         .quad   0x4d2c6dfc5ac42aed,0x53380d139d95b3df
317         .quad   0x650a73548baf63de,0x766a0abb3c77b2a8
318         .quad   0x81c2c92e47edaee6,0x92722c851482353b
319         .quad   0xa2bfe8a14cf10364,0xa81a664bbc423001
320         .quad   0xc24b8b70d0f89791,0xc76c51a30654be30
321         .quad   0xd192e819d6ef5218,0xd69906245565a910
322         .quad   0xf40e35855771202a,0x106aa07032bbd1b8
323         .quad   0x19a4c116b8d2d0c8,0x1e376c085141ab53
324         .quad   0x2748774cdf8eeb99,0x34b0bcb5e19b48a8
325         .quad   0x391c0cb3c5c95a63,0x4ed8aa4ae3418acb
```



```
326         .quad   0x5b9cca4f7763e373,0x682e6ff3d6b2b8a3
327         .quad   0x748f82ee5defb2fc,0x78a5636f43172f60
328         .quad   0x84c87814a1f0ab72,0x8cc702081a6439ec
329         .quad   0x90befffa23631e28,0xa4506cebde82bde9
330         .quad   0xbef9a3f7b2c67915,0xc67178f2e372532b
331         .quad   0xca273eceea26619c,0xd186b8c721c0c207
332         .quad   0xeada7dd6cde0eb1e,0xf57d4f7fee6ed178
333         .quad   0x06f067aa72176fba,0x0a637dc5a2c898a6
334         .quad   0x113f9804bef90dae,0x1b710b35131c471b
335         .quad   0x28db77f523047d84,0x32caab7b40c72493
336         .quad   0x3c9ebe0a15c9bebc,0x431d67c49c100d4c
337         .quad   0x4cc5d4becb3e42b6,0x597f299cfc657e2a
338         .quad   0x5fcb6fab3ad6faec,0x6c44198c4a475817
339 ___
340 }

342 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
343 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
344 if ($win64) {
345 $rec="%rcx";
346 $frame="%rdx";
347 $context="%r8";
348 $disp="%r9";

350 $code.=<<___;
351 .extern __imp_RtlVirtualUnwind
352 .type   se_handler,\@abi-omnipotent
353 .align  16
354 se_handler:
355         push    %rsi
356         push    %rdi
357         push    %rbx
358         push    %rbp
359         push    %r12
360         push    %r13
361         push    %r14
362         push    %r15
363         pushfq
364         sub     \$64,%rsp

366         mov     120($context),%rax      # pull context->Rax
367         mov     248($context),%rbx      # pull context->Rip

369         lea     .Lprologue(%rip),%r10
370         cmp     %r10,%rbx               # context->Rip<.Lprologue
371         jb      .Lin_prologue

373         mov     152($context),%rax      # pull context->Rsp

375         lea     .Lepilogue(%rip),%r10
376         cmp     %r10,%rbx               # context->Rip>=.Lepilogue
377         jae     .Lin_prologue

379         mov     16*$SZ+3*8(%rax),%rax   # pull $_rsp
380         lea     48(%rax),%rax

382         mov     -8(%rax),%rbx
383         mov     -16(%rax),%rbp
384         mov     -24(%rax),%r12
385         mov     -32(%rax),%r13
386         mov     -40(%rax),%r14
387         mov     -48(%rax),%r15
388         mov     %rbx,144($context)      # restore context->Rbx
389         mov     %rbp,160($context)      # restore context->Rbp
390         mov     %r12,216($context)      # restore context->R12
391         mov     %r13,224($context)      # restore context->R13
```

```
392         mov     %r14,232($context)      # restore context->R14
393         mov     %r15,240($context)      # restore context->R15

395 .Lin_prologue:
396         mov     8(%rax),%rdi
397         mov     16(%rax),%rsi
398         mov     %rax,152($context)      # restore context->Rsp
399         mov     %rsi,168($context)      # restore context->Rsi
400         mov     %rdi,176($context)      # restore context->Rdi

402         mov     40($disp),%rdi          # disp->ContextRecord
403         mov     $context,%rsi           # context
404         mov     \$154,%ecx              # sizeof(CONTEXT)
405         .long   0xa548f3fc              # cld; rep movsq

407         mov     $disp,%rsi
408         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
409         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
410         mov     0(%rsi),%r8             # arg3, disp->ControlPc
411         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
412         mov     40(%rsi),%r10           # disp->ContextRecord
413         lea     56(%rsi),%r11           # &disp->HandlerData
414         lea     24(%rsi),%r12           # &disp->EstablisherFrame
415         mov     %r10,32(%rsp)           # arg5
416         mov     %r11,40(%rsp)           # arg6
417         mov     %r12,48(%rsp)           # arg7
418         mov     %rcx,56(%rsp)           # arg8, (NULL)
419         call    *__imp_RtlVirtualUnwind(%rip)

421         mov     \$1,%eax                # ExceptionContinueSearch
422         add     \$64,%rsp
423         popfq
424         pop     %r15
425         pop     %r14
426         pop     %r13
427         pop     %r12
428         pop     %rbp
429         pop     %rbx
430         pop     %rdi
431         pop     %rsi
432         ret
433 .size   se_handler,.-se_handler

435 .section        .pdata
436 .align  4
437         .rva    .LSEH_begin_$func
438         .rva    .LSEH_end_$func
439         .rva    .LSEH_info_$func

441 .section        .xdata
442 .align  8
443 .LSEH_info_$func:
444         .byte   9,0,0,0
445         .rva    se_handler
446 ___
447 }

449 $code =~ s/\`([^\`]*)\`/eval $1/gem;
450 print $code;
451 close STDOUT;
452 #endif /* ! codereview */
```

```
**********************************************************
   27663 Wed Aug 13 19:53:10 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/vpaes-x86.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

######################################################################
## Constant-time SSSE3 AES core implementation.
## version 0.1
##
## By Mike Hamburg (Stanford University), 2009
## Public domain.
##
## For details see http://shiftleft.org/papers/vector_aes/ and
## http://crypto.stanford.edu/vpaes/.

######################################################################
# September 2011.
#
# Port vpaes-x86_64.pl as 32-bit "almost" drop-in replacement for
# aes-586.pl. "Almost" refers to the fact that AES_cbc_encrypt
# doesn't handle partial vectors (doesn't have to if called from
# EVP only). "Drop-in" implies that this module doesn't share key
# schedule structure with the original nor does it make assumption
# about its alignment...
#
# Performance summary. aes-586.pl column lists large-block CBC
# encrypt/decrypt/with-hyper-threading-off(*) results in cycles per
# byte processed with 128-bit key, and vpaes-x86.pl column - [also
# large-block CBC] encrypt/decrypt.
#
#               aes-586.pl              vpaes-x86.pl
#
# Core 2(**)    29.1/42.3/18.3          22.0/25.6(***)
# Nehalem       27.9/40.4/18.1          10.3/12.0
# Atom          102./119./60.1          64.5/85.3(***)
#
# (*)   "Hyper-threading" in the context refers rather to cache shared
#       among multiple cores, than to specifically Intel HTT. As vast
#       majority of contemporary cores share cache, slower code path
#       is common place. In other words "with-hyper-threading-off"
#       results are presented mostly for reference purposes.
#
# (**)  "Core 2" refers to initial 65nm design, a.k.a. Conroe.
#
# (***) Less impressive improvement on Core 2 and Atom is due to slow
#       pshufb, yet it's respectable +32%/65%  improvement on Core 2
#       and +58%/40% on Atom (as implied, over "hyper-threading-safe"
#       code path).
#
#                                               <appro@openssl.org>

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],"vpaes-x86.pl",$x86only = $ARGV[$#ARGV] eq "386");

$PREFIX="vpaes";

my  ($round, $base, $magic, $key, $const, $inp, $out)=
    ("eax",  "ebx", "ecx",  "edx","ebp",  "esi","edi");

&static_label("_vpaes_consts");
&static_label("_vpaes_schedule_low_round");
```



```
&set_label("_vpaes_consts",64);
$k_inv=-0x30;           # inv, inva
        &data_word(0x0D080180,0x0E05060F,0x0A0B0C02,0x04070309);
        &data_word(0x0F0B0780,0x01040A06,0x02050809,0x030D0E0C);

$k_s0F=-0x10;           # s0F
        &data_word(0x0F0F0F0F,0x0F0F0F0F,0x0F0F0F0F,0x0F0F0F0F);

$k_ipt=0x00;            # input transform (lo, hi)
        &data_word(0x5A2A7000,0xC2B2E898,0x52227808,0xCABAE090);
        &data_word(0x317C4D00,0x4C01307D,0xB0FDCC81,0xCD80B1FC);

$k_sb1=0x20;            # sb1u, sb1t
        &data_word(0xCB503E00,0xB19BE18F,0x142AF544,0xA5DF7A6E);
        &data_word(0xFAE22300,0x3618D415,0x0D2ED9EF,0x3BF7CCC1);
$k_sb2=0x40;            # sb2u, sb2t
        &data_word(0x0B712400,0xE27A93C6,0xBC982FCD,0x5EB7E955);
        &data_word(0x0AE12900,0x69EB8840,0xAB82234A,0xC2A163C8);
$k_sbo=0x60;            # sbou, sbot
        &data_word(0x6FBDC700,0xD0D26D17,0xC502A878,0x15AABF7A);
        &data_word(0x5FBB6A00,0xCFE474A5,0x412B35FA,0x8E1E90D1);

$k_mc_forward=0x80;     # mc_forward
        &data_word(0x00030201,0x04070605,0x080B0A09,0x0C0F0E0D);
        &data_word(0x04070605,0x080B0A09,0x0C0F0E0D,0x00030201);
        &data_word(0x080B0A09,0x0C0F0E0D,0x00030201,0x04070605);
        &data_word(0x0C0F0E0D,0x00030201,0x04070605,0x080B0A09);

$k_mc_backward=0xc0;    # mc_backward
        &data_word(0x02010003,0x06050407,0x0A09080B,0x0E0D0C0F);
        &data_word(0x0E0D0C0F,0x02010003,0x06050407,0x0A09080B);
        &data_word(0x0A09080B,0x0E0D0C0F,0x02010003,0x06050407);
        &data_word(0x06050407,0x0A09080B,0x0E0D0C0F,0x02010003);

$k_sr=0x100;            # sr
        &data_word(0x03020100,0x07060504,0x0B0A0908,0x0F0E0D0C);
        &data_word(0x0F0A0500,0x030E0904,0x07020D08,0x0B06010C);
        &data_word(0x0B020900,0x0F060D04,0x030A0108,0x070E050C);
        &data_word(0x070A0D00,0x0B0E0104,0x0F020508,0x0306090C);

$k_rcon=0x140;          # rcon
        &data_word(0xAF9DEEB6,0x1F8391B9,0x4D7C7D81,0x702A9808);

$k_s63=0x150;           # s63: all equal to 0x63 transformed
        &data_word(0x5B5B5B5B,0x5B5B5B5B,0x5B5B5B5B,0x5B5B5B5B);

$k_opt=0x160;           # output transform
        &data_word(0xD6B66000,0xFF9F4929,0xDEBE6808,0xF7974121);
        &data_word(0x50BCEC00,0x01EDBD51,0xB05C0CE0,0xE10D5DB1);

$k_deskew=0x180;        # deskew tables: inverts the sbox's "skew"
        &data_word(0x47A4E300,0x07E4A340,0x5DBEF91A,0x1DFEB95A);
        &data_word(0x83EA6900,0x5F36B5DC,0xF49D1E77,0x2841C2AB);
##
##  Decryption stuff
##  Key schedule constants
##
$k_dksd=0x1a0;          # decryption key schedule: invskew x*D
        &data_word(0xA3E44700,0xFEB91A5D,0x5A1DBEF9,0x0740E3A4);
        &data_word(0xB5368300,0x41C277F4,0xAB289D1E,0x5FDC69EA);
$k_dksb=0x1c0;          # decryption key schedule: invskew x*B
        &data_word(0x8550D500,0x9A4FCA1F,0x1CC94C99,0x03D65386);
        &data_word(0xB6FC4A00,0x115BEDA7,0x7E3482C8,0xD993256F);
$k_dkse=0x1e0;          # decryption key schedule: invskew x*E + 0x63
        &data_word(0x1FC9D600,0xD5031CCA,0x994F5086,0x53859A4C);
```

```
128         &data_word(0x4FDC7BE8,0xA2319605,0x20B31487,0xCD5EF96A);
129 $k_dks9=0x200;          # decryption key schedule: invskew x*9
130         &data_word(0x7ED9A700,0xB6116FC8,0x82255BFC,0x4AED9334);
131         &data_word(0x27143300,0x45765162,0xE9DAFDCE,0x8BB89FAC);

133 ##
134 ##  Decryption stuff
135 ##  Round function constants
136 ##
137 $k_dipt=0x220;          # decryption input transform
138         &data_word(0x0B545F00,0x0F505B04,0x114E451A,0x154A411E);
139         &data_word(0x60056500,0x86E383E6,0xF491F194,0x12771772);

141 $k_dsb9=0x240;          # decryption sbox output *9*u, *9*t
142         &data_word(0x9A86D600,0x851C0353,0x4F994CC9,0xCAD51F50);
143         &data_word(0xECD74900,0xC03B1789,0xB2FBA565,0x725E2C9E);
144 $k_dsbd=0x260;          # decryption sbox output *D*u, *D*t
145         &data_word(0xE6B1A200,0x7D57CCDF,0x882A4439,0xF56E9B13);
146         &data_word(0x24C6CB00,0x3CE2FAF7,0x15DEEFD3,0x2931180D);
147 $k_dsbb=0x280;          # decryption sbox output *B*u, *B*t
148         &data_word(0x96B44200,0xD0226492,0xB0F2D404,0x602646F6);
149         &data_word(0xCD596700,0xC19498A6,0x3255AA6B,0xF3FF0C3E);
150 $k_dsbe=0x2a0;          # decryption sbox output *E*u, *E*t
151         &data_word(0x26D4D000,0x46F29296,0x64B4F6B0,0x22426004);
152         &data_word(0xFFAAC100,0x0C55A6CD,0x98593E32,0x9467F36B);
153 $k_dsbo=0x2c0;          # decryption sbox final output
154         &data_word(0x7EF94000,0x1387EA53,0xD4943E2D,0xC7AA6DB9);
155         &data_word(0x93441D00,0x12D7560F,0xD8C58E9C,0xCA4B8159);
156 &asciz  ("Vector Permutation AES for x86/SSSE3, Mike Hamburg (Stanford Universit
157 &align  (64);

159 &function_begin_B("_vpaes_preheat");
160         &add    ($const,&DWP(0,"esp"));
161         &movdqa ("xmm7",&QWP($k_inv,$const));
162         &movdqa ("xmm6",&QWP($k_s0F,$const));
163         &ret    ();
164 &function_end_B("_vpaes_preheat");

166 ##
167 ##  _aes_encrypt_core
168 ##
169 ##  AES-encrypt %xmm0.
170 ##
171 ##  Inputs:
172 ##     %xmm0 = input
173 ##     %xmm6-%xmm7 as in _vpaes_preheat
174 ##    (%edx) = scheduled keys
175 ##
176 ##  Output in %xmm0
177 ##  Clobbers  %xmm1-%xmm5, %eax, %ebx, %ecx, %edx
178 ##
179 ##
180 &function_begin_B("_vpaes_encrypt_core");
181         &mov    ($magic,16);
182         &mov    ($round,&DWP(240,$key));
183         &movdqa ("xmm1","xmm6")
184         &movdqa ("xmm2",&QWP($k_ipt,$const));
185         &pandn  ("xmm1","xmm0");
186         &movdqu ("xmm5",&QWP(0,$key));
187         &psrld  ("xmm1",4);
188         &pand   ("xmm0","xmm6");
189         &pshufb ("xmm2","xmm0");
190         &movdqa ("xmm0",&QWP($k_ipt+16,$const));
191         &pshufb ("xmm0","xmm1");
192         &pxor   ("xmm2","xmm5");
193         &pxor   ("xmm0","xmm2");
```



```
194         &add    ($key,16);
195         &lea    ($base,&DWP($k_mc_backward,$const));
196         &jmp    (&label("enc_entry"));


199 &set_label("enc_loop",16);
200         # middle of middle round
201         &movdqa ("xmm4",&QWP($k_sb1,$const));   # 4 : sb1u
202         &pshufb ("xmm4","xmm2");                # 4 = sb1u
203         &pxor   ("xmm4","xmm5");                # 4 = sb1u + k
204         &movdqa ("xmm0",&QWP($k_sb1+16,$const));# 0 : sb1t
205         &pshufb ("xmm0","xmm3");                # 0 = sb1t
206         &pxor   ("xmm0","xmm4");                # 0 = A
207         &movdqa ("xmm5",&QWP($k_sb2,$const));   # 4 : sb2u
208         &pshufb ("xmm5","xmm2");                # 4 = sb2u
209         &movdqa ("xmm1",&QWP(-0x40,$base,$magic));# .Lk_mc_forward[]
210         &movdqa ("xmm2",&QWP($k_sb2+16,$const));# 2 : sb2t
211         &pshufb ("xmm2","xmm3");                # 2 = sb2t
212         &pxor   ("xmm2","xmm5");                # 2 = 2A
213         &movdqa ("xmm4",&QWP(0,$base,$magic));  # .Lk_mc_backward[]
214         &movdqa ("xmm3","xmm0");                # 3 = A
215         &pshufb ("xmm0","xmm1");                # 0 = B
216         &add    ($key,16);                      # next key
217         &pxor   ("xmm0","xmm2");                # 0 = 2A+B
218         &pshufb ("xmm3","xmm4");                # 3 = D
219         &add    ($magic,16);                    # next mc
220         &pxor   ("xmm3","xmm0");                # 3 = 2A+B+D
221         &pshufb ("xmm0","xmm1");                # 0 = 2B+C
222         &and    ($magic,0x30);                  # ... mod 4
223         &pxor   ("xmm0","xmm3");                # 0 = 2A+3B+C+D
224         &sub    ($round,1);                     # nr--

226 &set_label("enc_entry");
227         # top of round
228         &movdqa ("xmm1","xmm6");                # 1 : i
229         &pandn  ("xmm1","xmm0");                # 1 = i<<4
230         &psrld  ("xmm1",4);                     # 1 = i
231         &pand   ("xmm0","xmm6");                # 0 = k
232         &movdqa ("xmm5",&QWP($k_inv+16,$const));# 2 : a/k
233         &pshufb ("xmm5","xmm0");                # 2 = a/k
234         &pxor   ("xmm0","xmm1");                # 0 = j
235         &movdqa ("xmm3","xmm7");                # 3 : 1/i
236         &pshufb ("xmm3","xmm1");                # 3 = 1/i
237         &pxor   ("xmm3","xmm5");                # 3 = iak = 1/i + a/k
238         &movdqa ("xmm4","xmm7");                # 4 : 1/j
239         &pshufb ("xmm4","xmm0");                # 4 = 1/j
240         &pxor   ("xmm4","xmm5");                # 4 = jak = 1/j + a/k
241         &movdqa ("xmm2","xmm7");                # 2 : 1/iak
242         &pshufb ("xmm2","xmm3");                # 2 = 1/iak
243         &pxor   ("xmm2","xmm0");                # 2 = io
244         &movdqa ("xmm3","xmm7");                # 3 : 1/jak
245         &movdqu ("xmm5",&QWP(0,$key));
246         &pshufb ("xmm3","xmm4");                # 3 = 1/jak
247         &pxor   ("xmm3","xmm1");                # 3 = jo
248         &jnz    (&label("enc_loop"));

250         # middle of last round
251         &movdqa ("xmm4",&QWP($k_sbo,$const));   # 3 : sbou      .Lk_sbo
252         &movdqa ("xmm0",&QWP($k_sbo+16,$const));# 3 : sbot      .Lk_sbo+16
253         &pshufb ("xmm4","xmm2");                # 4 = sbou
254         &pxor   ("xmm4","xmm5");                # 4 = sb1u + k
255         &pshufb ("xmm0","xmm3");                # 0 = sb1t
256         &movdqa ("xmm1",&QWP(0x40,$base,$magic));# .Lk_sr[]
257         &pxor   ("xmm0","xmm4");                # 0 = A
258         &pshufb ("xmm0","xmm1");
259         &ret    ();
```

```
260 &function_end_B("_vpaes_encrypt_core");

262 ##
263 ##  Decryption core
264 ##
265 ##  Same API as encryption core.
266 ##
267 &function_begin_B("_vpaes_decrypt_core");
268         &mov    ($round,&DWP(240,$key));
269         &lea    ($base,&DWP($k_dsbd,$const));
270         &movdqa ("xmm1","xmm6");
271         &movdqa ("xmm2",&QWP($k_dipt-$k_dsbd,$base));
272         &pandn  ("xmm1","xmm0");
273         &mov    ($magic,$round);
274         &psrld  ("xmm1",4)
275         &movdqu ("xmm5",&QWP(0,$key));
276         &shl    ($magic,4);
277         &pand   ("xmm0","xmm6");
278         &pshufb ("xmm2","xmm0");
279         &movdqa ("xmm0",&QWP($k_dipt-$k_dsbd+16,$base));
280         &xor    ($magic,0x30);
281         &pshufb ("xmm0","xmm1");
282         &and    ($magic,0x30);
283         &pxor   ("xmm2","xmm5");
284         &movdqa ("xmm5",&QWP($k_mc_forward+48,$const));
285         &pxor   ("xmm0","xmm2");
286         &add    ($key,16);
287         &lea    ($magic,&DWP($k_sr-$k_dsbd,$base,$magic));
288         &jmp    (&label("dec_entry"));

290 &set_label("dec_loop",16);
291 ##
292 ##  Inverse mix columns
293 ##
294         &movdqa ("xmm4",&QWP(-0x20,$base));      # 4 : sb9u
295         &pshufb ("xmm4","xmm2");                 # 4 = sb9u
296         &pxor   ("xmm4","xmm0");
297         &movdqa ("xmm0",&QWP(-0x10,$base));      # 0 : sb9t
298         &pshufb ("xmm0","xmm3");                 # 0 = sb9t
299         &pxor   ("xmm0","xmm4");                 # 0 = ch
300         &add    ($key,16);                       # next round key

302         &pshufb ("xmm0","xmm5");                 # MC ch
303         &movdqa ("xmm4",&QWP(0,$base));          # 4 : sbdu
304         &pshufb ("xmm4","xmm2");                 # 4 = sbdu
305         &pxor   ("xmm4","xmm0");                 # 4 = ch
306         &movdqa ("xmm0",&QWP(0x10,$base));       # 0 : sbdt
307         &pshufb ("xmm0","xmm3");                 # 0 = sbdt
308         &pxor   ("xmm0","xmm4");                 # 0 = ch
309         &sub    ($round,1);                      # nr--

311         &pshufb ("xmm0","xmm5");                 # MC ch
312         &movdqa ("xmm4",&QWP(0x20,$base));       # 4 : sbbu
313         &pshufb ("xmm4","xmm2");                 # 4 = sbbu
314         &pxor   ("xmm4","xmm0");                 # 4 = ch
315         &movdqa ("xmm0",&QWP(0x30,$base));       # 0 : sbbt
316         &pshufb ("xmm0","xmm3");                 # 0 = sbbt
317         &pxor   ("xmm0","xmm4");                 # 0 = ch

319         &pshufb ("xmm0","xmm5");                 # MC ch
320         &movdqa ("xmm4",&QWP(0x40,$base));       # 4 : sbeu
321         &pshufb ("xmm4","xmm2");                 # 4 = sbeu
322         &pxor   ("xmm4","xmm0");                 # 4 = ch
323         &movdqa ("xmm0",&QWP(0x50,$base));       # 0 : sbet
324         &pshufb ("xmm0","xmm3");                 # 0 = sbet
325         &pxor   ("xmm0","xmm4");                 # 0 = ch
```



```
327         &palignr("xmm5","xmm5",12);

329 &set_label("dec_entry");
330         # top of round
331         &movdqa ("xmm1","xmm6");                 # 1 : i
332         &pandn  ("xmm1","xmm0");                 # 1 = i<<4
333         &psrld  ("xmm1",4);                      # 1 = i
334         &pand   ("xmm0","xmm6");                 # 0 = k
335         &movdqa ("xmm2",&QWP($k_inv+16,$const));# 2 : a/k
336         &pshufb ("xmm2","xmm0");                 # 2 = a/k
337         &pxor   ("xmm0","xmm1");                 # 0 = j
338         &movdqa ("xmm3","xmm7");                 # 3 : 1/i
339         &pshufb ("xmm3","xmm1");                 # 3 = 1/i
340         &pxor   ("xmm3","xmm2");                 # 3 = iak = 1/i + a/k
341         &movdqa ("xmm4","xmm7");                 # 4 : 1/j
342         &pshufb ("xmm4","xmm0");                 # 4 = 1/j
343         &pxor   ("xmm4","xmm2");                 # 4 = jak = 1/j + a/k
344         &movdqa ("xmm2","xmm7");                 # 2 : 1/iak
345         &pshufb ("xmm2","xmm3");                 # 2 = 1/iak
346         &pxor   ("xmm2","xmm0");                 # 2 = io
347         &movdqa ("xmm3","xmm7");                 # 3 : 1/jak
348         &pshufb ("xmm3","xmm4");                 # 3 = 1/jak
349         &pxor   ("xmm3","xmm1");                 # 3 = jo
350         &movdqu ("xmm0",&QWP(0,$key));
351         &jnz    (&label("dec_loop"));

353         # middle of last round
354         &movdqa ("xmm4",&QWP(0x60,$base));       # 3 : sbou
355         &pshufb ("xmm4","xmm2");                 # 4 = sbou
356         &pxor   ("xmm4","xmm0");                 # 4 = sb1u + k
357         &movdqa ("xmm0",&QWP(0x70,$base));       # 0 : sbot
358         &movdqa ("xmm2",&QWP(0,$magic));
359         &pshufb ("xmm0","xmm3");                 # 0 = sb1t
360         &pxor   ("xmm0","xmm4");                 # 0 = A
361         &pshufb ("xmm0","xmm2");
362         &ret    ();
363 &function_end_B("_vpaes_decrypt_core");

365 ########################################################
366 ##                                                    ##
367 ##                  AES key schedule                  ##
368 ##                                                    ##
369 ########################################################
370 &function_begin_B("_vpaes_schedule_core");
371         &add    ($const,&DWP(0,"esp"));
372         &movdqu ("xmm0",&QWP(0,$inp));           # load key (unaligned)
373         &movdqa ("xmm2",&QWP($k_rcon,$const));  # load rcon

375         # input transform
376         &movdqa ("xmm3","xmm0");
377         &lea    ($base,&DWP($k_ipt,$const));
378         &movdqa (&QWP(4,"esp"),"xmm2");          # xmm8
379         &call   ("_vpaes_schedule_transform");
380         &movdqa ("xmm7","xmm0");

382         &test   ($out,$out);
383         &jnz    (&label("schedule_am_decrypting"));

385         # encrypting, output zeroth round key after transform
386         &movdqu (&QWP(0,$key),"xmm0");
387         &jmp    (&label("schedule_go"));

389 &set_label("schedule_am_decrypting");
390         # decrypting, output zeroth round key after shiftrows
391         &movdqa ("xmm1",&QWP($k_sr,$const,$magic));
```

```
392         &pshufb ("xmm3","xmm1");
393         &movdqu (&QWP(0,$key),"xmm3");
394         &xor    ($magic,0x30);

396 &set_label("schedule_go");
397         &cmp    ($round,192);
398         &ja     (&label("schedule_256"));
399         &je     (&label("schedule_192"));
400         # 128: fall though

402 ##
403 ##  .schedule_128
404 ##
405 ##  128-bit specific part of key schedule.
406 ##
407 ##  This schedule is really simple, because all its parts
408 ##  are accomplished by the subroutines.
409 ##
410 &set_label("schedule_128");
411         &mov    ($round,10);

413 &set_label("loop_schedule_128");
414         &call   ("_vpaes_schedule_round");
415         &dec    ($round);
416         &jz     (&label("schedule_mangle_last"));
417         &call   ("_vpaes_schedule_mangle");     # write output
418         &jmp    (&label("loop_schedule_128"));

420 ##
421 ##  .aes_schedule_192
422 ##
423 ##  192-bit specific part of key schedule.
424 ##
425 ##  The main body of this schedule is the same as the 128-bit
426 ##  schedule, but with more smearing.  The long, high side is
427 ##  stored in %xmm7 as before, and the short, low side is in
428 ##  the high bits of %xmm6.
429 ##
430 ##  This schedule is somewhat nastier, however, because each
431 ##  round produces 192 bits of key material, or 1.5 round keys.
432 ##  Therefore, on each cycle we do 2 rounds and produce 3 round
433 ##  keys.
434 ##
435 &set_label("schedule_192",16);
436         &movdqu ("xmm0",&QWP(8,$inp));          # load key part 2 (very unaligne
437         &call   ("_vpaes_schedule_transform");  # input transform
438         &movdqa ("xmm6","xmm0");                # save short part
439         &pxor   ("xmm4","xmm4");                # clear 4
440         &movhlps("xmm6","xmm4");                # clobber low side with zeros
441         &mov    ($round,4);

443 &set_label("loop_schedule_192");
444         &call   ("_vpaes_schedule_round");
445         &palignr("xmm0","xmm6",8);
446         &call   ("_vpaes_schedule_mangle");     # save key n
447         &call   ("_vpaes_schedule_192_smear");
448         &call   ("_vpaes_schedule_mangle");     # save key n+1
449         &call   ("_vpaes_schedule_round");
450         &dec    ($round);
451         &jz     (&label("schedule_mangle_last"));
452         &call   ("_vpaes_schedule_mangle");     # save key n+2
453         &call   ("_vpaes_schedule_192_smear");
454         &jmp    (&label("loop_schedule_192"));

456 ##
457 ##  .aes_schedule_256
```



```
458 ##
459 ##  256-bit specific part of key schedule.
460 ##
461 ##  The structure here is very similar to the 128-bit
462 ##  schedule, but with an additional "low side" in
463 ##  %xmm6.  The low side's rounds are the same as the
464 ##  high side's, except no rcon and no rotation.
465 ##
466 &set_label("schedule_256",16);
467         &movdqu ("xmm0",&QWP(16,$inp));         # load key part 2 (unaligned)
468         &call   ("_vpaes_schedule_transform");  # input transform
469         &mov    ($round,7);

471 &set_label("loop_schedule_256");
472         &call   ("_vpaes_schedule_mangle");     # output low result
473         &movdqa ("xmm6","xmm0");                # save cur_lo in xmm6

475         # high round
476         &call   ("_vpaes_schedule_round");
477         &dec    ($round);
478         &jz     (&label("schedule_mangle_last"));
479         &call   ("_vpaes_schedule_mangle");

481         # low round. swap xmm7 and xmm6
482         &pshufd ("xmm0","xmm0",0xFF);
483         &movdqa (&QWP(20,"esp"),"xmm7");
484         &movdqa ("xmm7","xmm6");
485         &call   ("_vpaes_schedule_low_round");
486         &movdqa ("xmm7",&QWP(20,"esp"));

488         &jmp    (&label("loop_schedule_256"));

490 ##
491 ##  .aes_schedule_mangle_last
492 ##
493 ##  Mangler for last round of key schedule
494 ##  Mangles %xmm0
495 ##    when encrypting, outputs out(%xmm0) ^ 63
496 ##    when decrypting, outputs unskew(%xmm0)
497 ##
498 ##  Always called right before return... jumps to cleanup and exits
499 ##
500 &set_label("schedule_mangle_last",16);
501         # schedule last round key from xmm0
502         &lea    ($base,&DWP($k_deskew,$const));
503         &test   ($out,$out);
504         &jnz    (&label("schedule_mangle_last_dec"));

506         # encrypting
507         &movdqa ("xmm1",&QWP($k_sr,$const,$magic));
508         &pshufb ("xmm0","xmm1");                # output permute
509         &lea    ($base,&DWP($k_opt,$const));    # prepare to output transform
510         &add    ($key,32);

512 &set_label("schedule_mangle_last_dec");
513         &add    ($key,-16);
514         &pxor   ("xmm0",&QWP($k_s63,$const));
515         &call   ("_vpaes_schedule_transform");  # output transform
516         &movdqu (&QWP(0,$key),"xmm0");          # save last key

518         # cleanup
519         &pxor   ("xmm0","xmm0");
520         &pxor   ("xmm1","xmm1");
521         &pxor   ("xmm2","xmm2");
522         &pxor   ("xmm3","xmm3");
523         &pxor   ("xmm4","xmm4");
```

```
524         &pxor   ("xmm5","xmm5");
525         &pxor   ("xmm6","xmm6");
526         &pxor   ("xmm7","xmm7");
527         &ret    ();
528 &function_end_B("_vpaes_schedule_core");

530 ##
531 ##  .aes_schedule_192_smear
532 ##
533 ##  Smear the short, low side in the 192-bit key schedule.
534 ##
535 ##  Inputs:
536 ##    %xmm7: high side, b  a  x  y
537 ##    %xmm6:  low side, d  c  0  0
538 ##    %xmm13: 0
539 ##
540 ##  Outputs:
541 ##    %xmm6: b+c+d  b+c  0  0
542 ##    %xmm0: b+c+d  b+c  b  a
543 ##
544 &function_begin_B("_vpaes_schedule_192_smear");
545         &pshufd ("xmm0","xmm6",0x80);           # d c 0 0 -> c 0 0 0
546         &pxor   ("xmm6","xmm0");                # -> c+d c 0 0
547         &pshufd ("xmm0","xmm7",0xFE);           # b a _ _ -> b b b a
548         &pxor   ("xmm6","xmm0");                # -> b+c+d b+c b a
549         &movdqa ("xmm0","xmm6");
550         &pxor   ("xmm1","xmm1");
551         &movhlps("xmm6","xmm1");                # clobber low side with zeros
552         &ret    ();
553 &function_end_B("_vpaes_schedule_192_smear");

555 ##
556 ##  .aes_schedule_round
557 ##
558 ##  Runs one main round of the key schedule on %xmm0, %xmm7
559 ##
560 ##  Specifically, runs subbytes on the high dword of %xmm0
561 ##  then rotates it by one byte and xors into the low dword of
562 ##  %xmm7.
563 ##
564 ##  Adds rcon from low byte of %xmm8, then rotates %xmm8 for
565 ##  next rcon.
566 ##
567 ##  Smears the dwords of %xmm7 by xoring the low into the
568 ##  second low, result into third, result into highest.
569 ##
570 ##  Returns results in %xmm7 = %xmm0.
571 ##  Clobbers %xmm1-%xmm5.
572 ##
573 &function_begin_B("_vpaes_schedule_round");
574         # extract rcon from xmm8
575         &movdqa ("xmm2",&QWP(8,"esp"));         # xmm8
576         &pxor   ("xmm1","xmm1");
577         &palignr("xmm1","xmm2",15);
578         &palignr("xmm2","xmm2",15);
579         &pxor   ("xmm7","xmm1");

581         # rotate
582         &pshufd ("xmm0","xmm0",0xFF);
583         &palignr("xmm0","xmm0",1);

585         # fall through...
586         &movdqa (&QWP(8,"esp"),"xmm2");         # xmm8

588         # low round: same as high round, but no rotation and no rcon.
589 &set_label("_vpaes_schedule_low_round");
```



```
590         # smear xmm7
591         &movdqa ("xmm1","xmm7");
592         &pslldq ("xmm7",4);
593         &pxor   ("xmm7","xmm1");
594         &movdqa ("xmm1","xmm7");
595         &pslldq ("xmm7",8);
596         &pxor   ("xmm7","xmm1");
597         &pxor   ("xmm7",&QWP($k_s63,$const));

599         # subbyte
600         &movdqa ("xmm4",&QWP($k_s0F,$const));
601         &movdqa ("xmm5",&QWP($k_inv,$const));   # 4 : 1/j
602         &movdqa ("xmm1","xmm4");
603         &pandn  ("xmm1","xmm0");
604         &psrld  ("xmm1",4);                     # 1 = i
605         &pand   ("xmm0","xmm4");                # 0 = k
606         &movdqa ("xmm2",&QWP($k_inv+16,$const));# 2 : a/k
607         &pshufb ("xmm2","xmm0");                # 2 = a/k
608         &pxor   ("xmm0","xmm1");                # 0 = j
609         &movdqa ("xmm3","xmm5");                # 3 : 1/i
610         &pshufb ("xmm3","xmm1");                # 3 = 1/i
611         &pxor   ("xmm3","xmm2");                # 3 = iak = 1/i + a/k
612         &movdqa ("xmm4","xmm5");                # 4 : 1/j
613         &pshufb ("xmm4","xmm0");                # 4 = 1/j
614         &pxor   ("xmm4","xmm2");                # 4 = jak = 1/j + a/k
615         &movdqa ("xmm2","xmm5");                # 2 : 1/iak
616         &pshufb ("xmm2","xmm3");                # 2 = 1/iak
617         &pxor   ("xmm2","xmm0");                # 2 = io
618         &movdqa ("xmm3","xmm5");                # 3 : 1/jak
619         &pshufb ("xmm3","xmm4");                # 3 = 1/jak
620         &pxor   ("xmm3","xmm1");                # 3 = jo
621         &movdqa ("xmm4",&QWP($k_sb1,$const));   # 4 : sbou
622         &pshufb ("xmm4","xmm2");                # 4 = sbou
623         &movdqa ("xmm0",&QWP($k_sb1+16,$const));# 0 : sbot
624         &pshufb ("xmm0","xmm3");                # 0 = sb1t
625         &pxor   ("xmm0","xmm4");                # 0 = sbox output

627         # add in smeared stuff
628         &pxor   ("xmm0","xmm7");
629         &movdqa ("xmm7","xmm0");
630         &ret    ();
631 &function_end_B("_vpaes_schedule_round");

633 ##
634 ##  .aes_schedule_transform
635 ##
636 ##  Linear-transform %xmm0 according to tables at (%ebx)
637 ##
638 ##  Output in %xmm0
639 ##  Clobbers %xmm1, %xmm2
640 ##
641 &function_begin_B("_vpaes_schedule_transform");
642         &movdqa ("xmm2",&QWP($k_s0F,$const));
643         &movdqa ("xmm1","xmm2");
644         &pandn  ("xmm1","xmm0");
645         &psrld  ("xmm1",4);
646         &pand   ("xmm0","xmm2");
647         &movdqa ("xmm2",&QWP(0,$base));
648         &pshufb ("xmm2","xmm0");
649         &movdqa ("xmm0",&QWP(16,$base));
650         &pshufb ("xmm0","xmm1");
651         &pxor   ("xmm0","xmm2");
652         &ret    ();
653 &function_end_B("_vpaes_schedule_transform");

655 ##
```

```
656 ##  .aes_schedule_mangle
657 ##
658 ##  Mangle xmm0 from (basis-transformed) standard version
659 ##  to our version.
660 ##
661 ##  On encrypt,
662 ##    xor with 0x63
663 ##    multiply by circulant 0,1,1,1
664 ##    apply shiftrows transform
665 ##
666 ##  On decrypt,
667 ##    xor with 0x63
668 ##    multiply by "inverse mixcolumns" circulant E,B,D,9
669 ##    deskew
670 ##    apply shiftrows transform
671 ##
672 ##
673 ##  Writes out to (%edx), and increments or decrements it
674 ##  Keeps track of round number mod 4 in %ecx
675 ##  Preserves xmm0
676 ##  Clobbers xmm1-xmm5
677 ##
678 &function_begin_B("_vpaes_schedule_mangle");
679         &movdqa ("xmm4","xmm0");        # save xmm0 for later
680         &movdqa ("xmm5",&QWP($k_mc_forward,$const));
681         &test   ($out,$out);
682         &jnz    (&label("schedule_mangle_dec"));
683
684         # encrypting
685         &add    ($key,16);
686         &pxor   ("xmm4",&QWP($k_s63,$const));
687         &pshufb ("xmm4","xmm5");
688         &movdqa ("xmm3","xmm4");
689         &pshufb ("xmm4","xmm5");
690         &pxor   ("xmm3","xmm4");
691         &pshufb ("xmm4","xmm5");
692         &pxor   ("xmm3","xmm4");
693
694         &jmp    (&label("schedule_mangle_both"));
695
696 &set_label("schedule_mangle_dec",16);
697         # inverse mix columns
698         &movdqa ("xmm2",&QWP($k_s0F,$const));
699         &lea    ($inp,&DWP($k_dksd,$const));
700         &movdqa ("xmm1","xmm2");
701         &pandn  ("xmm1","xmm4");
702         &psrld  ("xmm1",4);                     # 1 = hi
703         &pand   ("xmm4","xmm2");                # 4 = lo
704
705         &movdqa ("xmm2",&QWP(0,$inp));
706         &pshufb ("xmm2","xmm4");
707         &movdqa ("xmm3",&QWP(0x10,$inp));
708         &pshufb ("xmm3","xmm1");
709         &pxor   ("xmm3","xmm2");
710         &pshufb ("xmm3","xmm5");
711
712         &movdqa ("xmm2",&QWP(0x20,$inp));
713         &pshufb ("xmm2","xmm4");
714         &pxor   ("xmm2","xmm3");
715         &movdqa ("xmm3",&QWP(0x30,$inp));
716         &pshufb ("xmm3","xmm1");
717         &pxor   ("xmm3","xmm2");
718         &pshufb ("xmm3","xmm5");
719
720         &movdqa ("xmm2",&QWP(0x40,$inp));
721         &pshufb ("xmm2","xmm4");
```



```
722         &pxor   ("xmm2","xmm3");
723         &movdqa ("xmm3",&QWP(0x50,$inp));
724         &pshufb ("xmm3","xmm1");
725         &pxor   ("xmm3","xmm2");
726         &pshufb ("xmm3","xmm5");
727
728         &movdqa ("xmm2",&QWP(0x60,$inp));
729         &pshufb ("xmm2","xmm4");
730         &pxor   ("xmm2","xmm3");
731         &movdqa ("xmm3",&QWP(0x70,$inp));
732         &pshufb ("xmm3","xmm1");
733         &pxor   ("xmm3","xmm2");
734
735         &add    ($key,-16);
736
737 &set_label("schedule_mangle_both");
738         &movdqa ("xmm1",&QWP($k_sr,$const,$magic));
739         &pshufb ("xmm3","xmm1");
740         &add    ($magic,-16);
741         &and    ($magic,0x30);
742         &movdqu (&QWP(0,$key),"xmm3");
743         &ret    ();
744 &function_end_B("_vpaes_schedule_mangle");
745
746 #
747 # Interface to OpenSSL
748 #
749 &function_begin("${PREFIX}_set_encrypt_key");
750         &mov    ($inp,&wparam(0));              # inp
751         &lea    ($base,&DWP(-56,"esp"));
752         &mov    ($round,&wparam(1));            # bits
753         &and    ($base,-16);
754         &mov    ($key,&wparam(2));              # key
755         &xchg   ($base,"esp");                  # alloca
756         &mov    (&DWP(48,"esp"),$base);
757
758         &mov    ($base,$round);
759         &shr    ($base,5);
760         &add    ($base,5);
761         &mov    (&DWP(240,$key),$base);         # AES_KEY->rounds = nbits/32+5;
762         &mov    ($magic,0x30);
763         &mov    ($out,0);
764
765         &lea    ($const,&DWP(&label("_vpaes_consts")."+0x30-".&label("pic_point"
766         &call   ("_vpaes_schedule_core");
767 &set_label("pic_point");
768
769         &mov    ("esp",&DWP(48,"esp"));
770         &xor    ("eax","eax");
771 &function_end("${PREFIX}_set_encrypt_key");
772
773 &function_begin("${PREFIX}_set_decrypt_key");
774         &mov    ($inp,&wparam(0));              # inp
775         &lea    ($base,&DWP(-56,"esp"));
776         &mov    ($round,&wparam(1));            # bits
777         &and    ($base,-16);
778         &mov    ($key,&wparam(2));              # key
779         &xchg   ($base,"esp");                  # alloca
780         &mov    (&DWP(48,"esp"),$base);
781
782         &mov    ($base,$round);
783         &shr    ($base,5);
784         &add    ($base,5);
785         &mov    (&DWP(240,$key),$base); # AES_KEY->rounds = nbits/32+5;
786         &shl    ($base,4);
787         &lea    ($key,&DWP(16,$key,$base));
```

```
789         &mov    ($out,1);
790         &mov    ($magic,$round);
791         &shr    ($magic,1);
792         &and    ($magic,32);
793         &xor    ($magic,32);                    # nbist==192?0:32;

795         &lea    ($const,&DWP(&label("_vpaes_consts")."+0x30-".&label("pic_point"
796         &call   ("_vpaes_schedule_core");
797 &set_label("pic_point");

799         &mov    ("esp",&DWP(48,"esp"));
800         &xor    ("eax","eax");
801 &function_end("${PREFIX}_set_decrypt_key");

803 &function_begin("${PREFIX}_encrypt");
804         &lea    ($const,&DWP(&label("_vpaes_consts")."+0x30-".&label("pic_point"
805         &call   ("_vpaes_preheat");
806 &set_label("pic_point");
807         &mov    ($inp,&wparam(0));              # inp
808         &lea    ($base,&DWP(-56,"esp"));
809         &mov    ($out,&wparam(1));              # out
810         &and    ($base,-16);
811         &mov    ($key,&wparam(2));              # key
812         &xchg   ($base,"esp");                  # alloca
813         &mov    (&DWP(48,"esp"),$base);

815         &movdqu ("xmm0",&QWP(0,$inp));
816         &call   ("_vpaes_encrypt_core");
817         &movdqu (&QWP(0,$out),"xmm0");

819         &mov    ("esp",&DWP(48,"esp"));
820 &function_end("${PREFIX}_encrypt");

822 &function_begin("${PREFIX}_decrypt");
823         &lea    ($const,&DWP(&label("_vpaes_consts")."+0x30-".&label("pic_point"
824         &call   ("_vpaes_preheat");
825 &set_label("pic_point");
826         &mov    ($inp,&wparam(0));              # inp
827         &lea    ($base,&DWP(-56,"esp"));
828         &mov    ($out,&wparam(1));              # out
829         &and    ($base,-16);
830         &mov    ($key,&wparam(2));              # key
831         &xchg   ($base,"esp");                  # alloca
832         &mov    (&DWP(48,"esp"),$base);

834         &movdqu ("xmm0",&QWP(0,$inp));
835         &call   ("_vpaes_decrypt_core");
836         &movdqu (&QWP(0,$out),"xmm0");

838         &mov    ("esp",&DWP(48,"esp"));
839 &function_end("${PREFIX}_decrypt");

841 &function_begin("${PREFIX}_cbc_encrypt");
842         &mov    ($inp,&wparam(0));              # inp
843         &mov    ($out,&wparam(1));              # out
844         &mov    ($round,&wparam(2));            # len
845         &mov    ($key,&wparam(3));              # key
846         &sub    ($round,16);
847         &jc     (&label("cbc_abort"));
848         &lea    ($base,&DWP(-56,"esp"));
849         &mov    ($const,&wparam(4));            # ivp
850         &and    ($base,-16);
851         &mov    ($magic,&wparam(5));            # enc
852         &xchg   ($base,"esp");                  # alloca
853         &movdqu ("xmm1",&QWP(0,$const));        # load IV
```



```
854         &sub    ($out,$inp);
855         &mov    (&DWP(48,"esp"),$base);

857         &mov    (&DWP(0,"esp"),$out);           # save out
858         &mov    (&DWP(4,"esp"),$key)            # save key
859         &mov    (&DWP(8,"esp"),$const);         # save ivp
860         &mov    ($out,$round);                  # $out works as $len

862         &lea    ($const,&DWP(&label("_vpaes_consts")."+0x30-".&label("pic_point"
863         &call   ("_vpaes_preheat");
864 &set_label("pic_point");
865         &cmp    ($magic,0);
866         &je     (&label("cbc_dec_loop"));
867         &jmp    (&label("cbc_enc_loop"));

869 &set_label("cbc_enc_loop",16);
870         &movdqu ("xmm0",&QWP(0,$inp));          # load input
871         &pxor   ("xmm0","xmm1");                # inp^=iv
872         &call   ("_vpaes_encrypt_core");
873         &mov    ($base,&DWP(0,"esp"));          # restore out
874         &mov    ($key,&DWP(4,"esp"));           # restore key
875         &movdqa ("xmm1","xmm0");
876         &movdqu (&QWP(0,$base,$inp),"xmm0");    # write output
877         &lea    ($inp,&DWP(16,$inp));
878         &sub    ($out,16);
879         &jnc    (&label("cbc_enc_loop"));
880         &jmp    (&label("cbc_done"));

882 &set_label("cbc_dec_loop",16);
883         &movdqu ("xmm0",&QWP(0,$inp));          # load input
884         &movdqa (&QWP(16,"esp"),"xmm1");        # save IV
885         &movdqa (&QWP(32,"esp"),"xmm0");        # save future IV
886         &call   ("_vpaes_decrypt_core");
887         &mov    ($base,&DWP(0,"esp"));          # restore out
888         &mov    ($key,&DWP(4,"esp"));           # restore key
889         &pxor   ("xmm0",&QWP(16,"esp"));        # out^=iv
890         &movdqa ("xmm1",&QWP(32,"esp"));        # load next IV
891         &movdqu (&QWP(0,$base,$inp),"xmm0");    # write output
892         &lea    ($inp,&DWP(16,$inp));
893         &sub    ($out,16);
894         &jnc    (&label("cbc_dec_loop"));

896 &set_label("cbc_done");
897         &mov    ($base,&DWP(8,"esp"));          # restore ivp
898         &mov    ("esp",&DWP(48,"esp"));
899         &movdqu (&QWP(0,$base),"xmm1");         # write IV
900 &set_label("cbc_abort");
901 &function_end("${PREFIX}_cbc_encrypt");

903 &asm_finish();
904 #endif /* ! codereview */
```

```
**********************************************************
   30540 Wed Aug 13 19:53:10 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/vpaes-x86_64.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

######################################################################
## Constant-time SSSE3 AES core implementation.
## version 0.1
##
## By Mike Hamburg (Stanford University), 2009
## Public domain.
##
## For details see http://shiftleft.org/papers/vector_aes/ and
## http://crypto.stanford.edu/vpaes/.

######################################################################
# September 2011.
#
# Interface to OpenSSL as "almost" drop-in replacement for
# aes-x86_64.pl. "Almost" refers to the fact that AES_cbc_encrypt
# doesn't handle partial vectors (doesn't have to if called from
# EVP only). "Drop-in" implies that this module doesn't share key
# schedule structure with the original nor does it make assumption
# about its alignment...
#
# Performance summary. aes-x86_64.pl column lists large-block CBC
# encrypt/decrypt/with-hyper-threading-off(*) results in cycles per
# byte processed with 128-bit key, and vpaes-x86_64.pl column -
# [also large-block CBC] encrypt/decrypt.
#
#               aes-x86_64.pl           vpaes-x86_64.pl
#
# Core 2(**)    30.5/43.7/14.3          21.8/25.7(***)
# Nehalem       30.5/42.2/14.6           9.8/11.8
# Atom          63.9/79.0/32.1          64.0/84.8(***)
#
# (*)   "Hyper-threading" in the context refers rather to cache shared
#       among multiple cores, than to specifically Intel HTT. As vast
#       majority of contemporary cores share cache, slower code path
#       is common place. In other words "with-hyper-threading-off"
#       results are presented mostly for reference purposes.
#
# (**)  "Core 2" refers to initial 65nm design, a.k.a. Conroe.
#
# (***) Less impressive improvement on Core 2 and Atom is due to slow
#       pshufb, yet it's respectable +40%/78% improvement on Core 2
#       (as implied, over "hyper-threading-safe" code path).
#
#                                               <appro@openssl.org>

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;
```



```
$PREFIX="vpaes";

$code.=<<___;
.text

##
##  _aes_encrypt_core
##
##  AES-encrypt %xmm0.
##
##  Inputs:
##     %xmm0 = input
##     %xmm9-%xmm15 as in _vpaes_preheat
##    (%rdx) = scheduled keys
##
##  Output in %xmm0
##  Clobbers  %xmm1-%xmm5, %r9, %r10, %r11, %rax
##  Preserves %xmm6 - %xmm8 so you get some local vectors
##
##
.type   _vpaes_encrypt_core,\@abi-omnipotent
.align 16
_vpaes_encrypt_core:
        mov     %rdx,   %r9
        mov     \$16,   %r11
        mov     240(%rdx),%eax
        movdqa  %xmm9,  %xmm1
        movdqa  .Lk_ipt(%rip), %xmm2    # iptlo
        pandn   %xmm0,  %xmm1
        movdqu  (%r9),  %xmm5           # round0 key
        psrld   \$4,    %xmm1
        pand    %xmm9,  %xmm0
        pshufb  %xmm0,  %xmm2
        movdqa  .Lk_ipt+16(%rip), %xmm0 # ipthi
        pshufb  %xmm1,  %xmm0
        pxor    %xmm5,  %xmm2
        pxor    %xmm2,  %xmm0
        add     \$16,   %r9
        lea     .Lk_mc_backward(%rip),%r10
        jmp     .Lenc_entry

.align 16
.Lenc_loop:
        # middle of middle round
        movdqa  %xmm13, %xmm4   # 4 : sb1u
        pshufb  %xmm2,  %xmm4   # 4 = sb1u
        pxor    %xmm5,  %xmm4   # 4 = sb1u + k
        movdqa  %xmm12, %xmm0   # 0 : sb1t
        pshufb  %xmm3,  %xmm0   # 0 = sb1t
        pxor    %xmm4,  %xmm0   # 0 = A
        movdqa  %xmm15, %xmm5   # 4 : sb2u
        pshufb  %xmm2,  %xmm5   # 4 = sb2u
        movdqa  -0x40(%r11,%r10), %xmm1         # .Lk_mc_forward[]
        movdqa  %xmm14, %xmm2   # 2 : sb2t
        pshufb  %xmm3,  %xmm2   # 2 = sb2t
        pxor    %xmm5,  %xmm2   # 2 = 2A
        movdqa  (%r11,%r10), %xmm4              # .Lk_mc_backward[]
        movdqa  %xmm0,  %xmm3   # 3 = A
        pshufb  %xmm1,  %xmm0   # 0 = B
        add     \$16,   %r9     # next key
        pxor    %xmm2,  %xmm0   # 0 = 2A+B
        pshufb  %xmm4,  %xmm3   # 3 = D
        add     \$16,   %r11    # next mc
        pxor    %xmm0,  %xmm3   # 3 = 2A+B+D
        pshufb  %xmm1,  %xmm0   # 0 = 2B+C
        and     \$0x30, %r11    # ... mod 4
```

```
128         pxor    %xmm3,  %xmm0   # 0 = 2A+3B+C+D
129         sub     \$1,%rax        # nr--

131 .Lenc_entry:
132         # top of round
133         movdqa  %xmm9,  %xmm1   # 1 : i
134         pandn   %xmm0,  %xmm1   # 1 = i<<4
135         psrld   \$4,    %xmm1   # 1 = i
136         pand    %xmm9,  %xmm0   # 0 = k
137         movdqa  %xmm11, %xmm5   # 2 : a/k
138         pshufb  %xmm0,  %xmm5   # 2 = a/k
139         pxor    %xmm1,  %xmm0   # 0 = j
140         movdqa  %xmm10, %xmm3   # 3 : 1/i
141         pshufb  %xmm1,  %xmm3   # 3 = 1/i
142         pxor    %xmm5,  %xmm3   # 3 = iak = 1/i + a/k
143         movdqa  %xmm10, %xmm4   # 4 : 1/j
144         pshufb  %xmm0,  %xmm4   # 4 = 1/j
145         pxor    %xmm5,  %xmm4   # 4 = jak = 1/j + a/k
146         movdqa  %xmm10, %xmm2   # 2 : 1/iak
147         pshufb  %xmm3,  %xmm2   # 2 = 1/iak
148         pxor    %xmm0,  %xmm2   # 2 = io
149         movdqa  %xmm10, %xmm3   # 3 : 1/jak
150         movdqu  (%r9),  %xmm5
151         pshufb  %xmm4,  %xmm3   # 3 = 1/jak
152         pxor    %xmm1,  %xmm3   # 3 = jo
153         jnz     .Lenc_loop

155         # middle of last round
156         movdqa  -0x60(%r10), %xmm4      # 3 : sbou      .Lk_sbo
157         movdqa  -0x50(%r10), %xmm0      # 0 : sbot      .Lk_sbo+16
158         pshufb  %xmm2,  %xmm4   # 4 = sbou
159         pxor    %xmm5,  %xmm4   # 4 = sb1u + k
160         pshufb  %xmm3,  %xmm0   # 0 = sb1t
161         movdqa  0x40(%r11,%r10), %xmm1          # .Lk_sr[]
162         pxor    %xmm4,  %xmm0   # 0 = A
163         pshufb  %xmm1,  %xmm0
164         ret
165 .size   _vpaes_encrypt_core,.-_vpaes_encrypt_core

167 ##
168 ##  Decryption core
169 ##
170 ##  Same API as encryption core.
171 ##
172 .type   _vpaes_decrypt_core,\@abi-omnipotent
173 .align  16
174 _vpaes_decrypt_core:
175         mov     %rdx,   %r9             # load key
176         mov     240(%rdx),%eax
177         movdqa  %xmm9,  %xmm1
178         movdqa  .Lk_dipt(%rip), %xmm2   # iptlo
179         pandn   %xmm0,  %xmm1
180         mov     %rax,   %r11
181         psrld   \$4,    %xmm1
182         movdqu  (%r9),  %xmm5           # round0 key
183         shl     \$4,    %r11
184         pand    %xmm9,  %xmm0
185         pshufb  %xmm0,  %xmm2
186         movdqa  .Lk_dipt+16(%rip), %xmm0 # ipthi
187         xor     \$0x30, %r11
188         lea     .Lk_dsbd(%rip),%r10
189         pshufb  %xmm1,  %xmm0
190         and     \$0x30, %r11
191         pxor    %xmm5,  %xmm2
192         movdqa  .Lk_mc_forward+48(%rip), %xmm5
193         pxor    %xmm2,  %xmm0
```



```
194         add     \$16,   %r9
195         add     %r10,   %r11
196         jmp     .Ldec_entry

198 .align 16
199 .Ldec_loop:
200 ##
201 ##  Inverse mix columns
202 ##
203         movdqa  -0x20(%r10),%xmm4       # 4 : sb9u
204         pshufb  %xmm2,  %xmm4           # 4 = sb9u
205         pxor    %xmm0,  %xmm4
206         movdqa  -0x10(%r10),%xmm0       # 0 : sb9t
207         pshufb  %xmm3,  %xmm0           # 0 = sb9t
208         pxor    %xmm4,  %xmm0           # 0 = ch
209         add     \$16, %r9               # next round key

211         pshufb  %xmm5,  %xmm0           # MC ch
212         movdqa  0x00(%r10),%xmm4        # 4 : sbdu
213         pshufb  %xmm2,  %xmm4           # 4 = sbdu
214         pxor    %xmm0,  %xmm4           # 4 = ch
215         movdqa  0x10(%r10),%xmm0        # 0 : sbdt
216         pshufb  %xmm3,  %xmm0           # 0 = sbdt
217         pxor    %xmm4,  %xmm0           # 0 = ch
218         sub     \$1,%rax                # nr--

220         pshufb  %xmm5,  %xmm0           # MC ch
221         movdqa  0x20(%r10),%xmm4        # 4 : sbbu
222         pshufb  %xmm2,  %xmm4           # 4 = sbbu
223         pxor    %xmm0,  %xmm4           # 4 = ch
224         movdqa  0x30(%r10),%xmm0        # 0 : sbbt
225         pshufb  %xmm3,  %xmm0           # 0 = sbbt
226         pxor    %xmm4,  %xmm0           # 0 = ch

228         pshufb  %xmm5,  %xmm0           # MC ch
229         movdqa  0x40(%r10),%xmm4        # 4 : sbeu
230         pshufb  %xmm2,  %xmm4           # 4 = sbeu
231         pxor    %xmm0,  %xmm4           # 4 = ch
232         movdqa  0x50(%r10),%xmm0        # 0 : sbet
233         pshufb  %xmm3,  %xmm0           # 0 = sbet
234         pxor    %xmm4,  %xmm0           # 0 = ch

236         palignr \$12,   %xmm5,  %xmm5

238 .Ldec_entry:
239         # top of round
240         movdqa  %xmm9,  %xmm1   # 1 : i
241         pandn   %xmm0,  %xmm1   # 1 = i<<4
242         psrld   \$4,    %xmm1   # 1 = i
243         pand    %xmm9,  %xmm0   # 0 = k
244         movdqa  %xmm11, %xmm2   # 2 : a/k
245         pshufb  %xmm0,  %xmm2   # 2 = a/k
246         pxor    %xmm1,  %xmm0   # 0 = j
247         movdqa  %xmm10, %xmm3   # 3 : 1/i
248         pshufb  %xmm1,  %xmm3   # 3 = 1/i
249         pxor    %xmm2,  %xmm3   # 3 = iak = 1/i + a/k
250         movdqa  %xmm10, %xmm4   # 4 : 1/j
251         pshufb  %xmm0,  %xmm4   # 4 = 1/j
252         pxor    %xmm2,  %xmm4   # 4 = jak = 1/j + a/k
253         movdqa  %xmm10, %xmm2   # 2 : 1/iak
254         pshufb  %xmm3,  %xmm2   # 2 = 1/iak
255         pxor    %xmm0,  %xmm2   # 2 = io
256         movdqa  %xmm10, %xmm3   # 3 : 1/jak
257         pshufb  %xmm4,  %xmm3   # 3 = 1/jak
258         pxor    %xmm1,  %xmm3   # 3 = jo
259         movdqu  (%r9),  %xmm0
```

```
260         jnz     .Ldec_loop

262         # middle of last round
263         movdqa  0x60(%r10), %xmm4       # 3 : sbou
264         pshufb  %xmm2,  %xmm4   # 4 = sbou
265         pxor    %xmm0,  %xmm4   # 4 = sb1u + k
266         movdqa  0x70(%r10), %xmm0       # 0 : sbot
267         movdqa  -0x160(%r11), %xmm2     # .Lk_sr-.Lk_dsbd=-0x160
268         pshufb  %xmm3,  %xmm0   # 0 = sb1t
269         pxor    %xmm4,  %xmm0   # 0 = A
270         pshufb  %xmm2,  %xmm0
271         ret
272 .size   _vpaes_decrypt_core,.-_vpaes_decrypt_core

274 ########################################################
275 ##                                                    ##
276 ##                  AES key schedule                  ##
277 ##                                                    ##
278 ########################################################
279 .type   _vpaes_schedule_core,\@abi-omnipotent
280 .align  16
281 _vpaes_schedule_core:
282         # rdi = key
283         # rsi = size in bits
284         # rdx = buffer
285         # rcx = direction.  0=encrypt, 1=decrypt

287         call    _vpaes_preheat          # load the tables
288         movdqa  .Lk_rcon(%rip), %xmm8   # load rcon
289         movdqu  (%rdi), %xmm0           # load key (unaligned)

291         # input transform
292         movdqa  %xmm0,  %xmm3
293         lea     .Lk_ipt(%rip), %r11
294         call    _vpaes_schedule_transform
295         movdqa  %xmm0,  %xmm7

297         lea     .Lk_sr(%rip),%r10
298         test    %rcx,   %rcx
299         jnz     .Lschedule_am_decrypting

301         # encrypting, output zeroth round key after transform
302         movdqu  %xmm0,  (%rdx)
303         jmp     .Lschedule_go

305 .Lschedule_am_decrypting:
306         # decrypting, output zeroth round key after shiftrows
307         movdqa  (%r8,%r10),%xmm1
308         pshufb  %xmm1,  %xmm3
309         movdqu  %xmm3,  (%rdx)
310         xor     \$0x30, %r8

312 .Lschedule_go:
313         cmp     \$192,  %esi
314         ja      .Lschedule_256
315         je      .Lschedule_192
316         # 128: fall though

318 ##
319 ##  .schedule_128
320 ##
321 ##  128-bit specific part of key schedule.
322 ##
323 ##  This schedule is really simple, because all its parts
324 ##  are accomplished by the subroutines.
325 ##
```



```
326 .Lschedule_128:
327         mov     \$10, %esi

329 .Loop_schedule_128:
330         call    _vpaes_schedule_round
331         dec     %rsi
332         jz      .Lschedule_mangle_last
333         call    _vpaes_schedule_mangle  # write output
334         jmp     .Loop_schedule_128

336 ##
337 ##  .aes_schedule_192
338 ##
339 ##  192-bit specific part of key schedule.
340 ##
341 ##  The main body of this schedule is the same as the 128-bit
342 ##  schedule, but with more smearing.  The long, high side is
343 ##  stored in %xmm7 as before, and the short, low side is in
344 ##  the high bits of %xmm6.
345 ##
346 ##  This schedule is somewhat nastier, however, because each
347 ##  round produces 192 bits of key material, or 1.5 round keys.
348 ##  Therefore, on each cycle we do 2 rounds and produce 3 round
349 ##  keys.
350 ##
351 .align  16
352 .Lschedule_192:
353         movdqu  8(%rdi),%xmm0           # load key part 2 (very unaligned)
354         call    _vpaes_schedule_transform       # input transform
355         movdqa  %xmm0,  %xmm6           # save short part
356         pxor    %xmm4,  %xmm4           # clear 4
357         movhlps %xmm4,  %xmm6           # clobber low side with zeros
358         mov     \$4,    %esi

360 .Loop_schedule_192:
361         call    _vpaes_schedule_round
362         palignr \$8,%xmm6,%xmm0
363         call    _vpaes_schedule_mangle  # save key n
364         call    _vpaes_schedule_192_smear
365         call    _vpaes_schedule_mangle  # save key n+1
366         call    _vpaes_schedule_round
367         dec     %rsi
368         jz      .Lschedule_mangle_last
369         call    _vpaes_schedule_mangle  # save key n+2
370         call    _vpaes_schedule_192_smear
371         jmp     .Loop_schedule_192

373 ##
374 ##  .aes_schedule_256
375 ##
376 ##  256-bit specific part of key schedule.
377 ##
378 ##  The structure here is very similar to the 128-bit
379 ##  schedule, but with an additional "low side" in
380 ##  %xmm6.  The low side's rounds are the same as the
381 ##  high side's, except no rcon and no rotation.
382 ##
383 .align  16
384 .Lschedule_256:
385         movdqu  16(%rdi),%xmm0          # load key part 2 (unaligned)
386         call    _vpaes_schedule_transform       # input transform
387         mov     \$7, %esi

389 .Loop_schedule_256:
390         call    _vpaes_schedule_mangle  # output low result
391         movdqa  %xmm0,  %xmm6           # save cur_lo in xmm6
```

```
393         # high round
394         call    _vpaes_schedule_round
395         dec     %rsi
396         jz      .Lschedule_mangle_last
397         call    _vpaes_schedule_mangle

399         # low round. swap xmm7 and xmm6
400         pshufd  \$0xFF, %xmm0,  %xmm0
401         movdqa  %xmm7,  %xmm5
402         movdqa  %xmm6,  %xmm7
403         call    _vpaes_schedule_low_round
404         movdqa  %xmm5,  %xmm7

406         jmp     .Loop_schedule_256


409 ##
410 ##  .aes_schedule_mangle_last
411 ##
412 ##  Mangler for last round of key schedule
413 ##  Mangles %xmm0
414 ##    when encrypting, outputs out(%xmm0) ^ 63
415 ##    when decrypting, outputs unskew(%xmm0)
416 ##
417 ##  Always called right before return... jumps to cleanup and exits
418 ##
419 .align  16
420 .Lschedule_mangle_last:
421         # schedule last round key from xmm0
422         lea     .Lk_deskew(%rip),%r11   # prepare to deskew
423         test    %rcx,   %rcx
424         jnz     .Lschedule_mangle_last_dec

426         # encrypting
427         movdqa  (%r8,%r10),%xmm1
428         pshufb  %xmm1,  %xmm0           # output permute
429         lea     .Lk_opt(%rip),  %r11    # prepare to output transform
430         add     \$32,   %rdx

432 .Lschedule_mangle_last_dec:
433         add     \$-16,  %rdx
434         pxor    .Lk_s63(%rip),  %xmm0
435         call    _vpaes_schedule_transform # output transform
436         movdqu  %xmm0,  (%rdx)          # save last key

438         # cleanup
439         pxor    %xmm0,  %xmm0
440         pxor    %xmm1,  %xmm1
441         pxor    %xmm2,  %xmm2
442         pxor    %xmm3,  %xmm3
443         pxor    %xmm4,  %xmm4
444         pxor    %xmm5,  %xmm5
445         pxor    %xmm6,  %xmm6
446         pxor    %xmm7,  %xmm7
447         ret
448 .size   _vpaes_schedule_core,.-_vpaes_schedule_core

450 ##
451 ##  .aes_schedule_192_smear
452 ##
453 ##  Smear the short, low side in the 192-bit key schedule.
454 ##
455 ##  Inputs:
456 ##    %xmm7: high side, b  a  x  y
457 ##    %xmm6:  low side, d  c  0  0
```



```
458 ##    %xmm13: 0
459 ##
460 ##  Outputs:
461 ##    %xmm6: b+c+d  b+c  0  0
462 ##    %xmm0: b+c+d  b+c  b  a
463 ##
464 .type   _vpaes_schedule_192_smear,\@abi-omnipotent
465 .align  16
466 _vpaes_schedule_192_smear:
467         pshufd  \$0x80, %xmm6,  %xmm0   # d c 0 0 -> c 0 0 0
468         pxor    %xmm0,  %xmm6           # -> c+d c 0 0
469         pshufd  \$0xFE, %xmm7,  %xmm0   # b a _ _ -> b b b a
470         pxor    %xmm0,  %xmm6           # -> b+c+d b+c b a
471         movdqa  %xmm6,  %xmm0
472         pxor    %xmm1,  %xmm1
473         movhlps %xmm1,  %xmm6           # clobber low side with zeros
474         ret
475 .size   _vpaes_schedule_192_smear,.-_vpaes_schedule_192_smear

477 ##
478 ##  .aes_schedule_round
479 ##
480 ##  Runs one main round of the key schedule on %xmm0, %xmm7
481 ##
482 ##  Specifically, runs subbytes on the high dword of %xmm0
483 ##  then rotates it by one byte and xors into the low dword of
484 ##  %xmm7.
485 ##
486 ##  Adds rcon from low byte of %xmm8, then rotates %xmm8 for
487 ##  next rcon.
488 ##
489 ##  Smears the dwords of %xmm7 by xoring the low into the
490 ##  second low, result into third, result into highest.
491 ##
492 ##  Returns results in %xmm7 = %xmm0.
493 ##  Clobbers %xmm1-%xmm4, %r11.
494 ##
495 .type   _vpaes_schedule_round,\@abi-omnipotent
496 .align  16
497 _vpaes_schedule_round:
498         # extract rcon from xmm8
499         pxor    %xmm1,  %xmm1
500         palignr \$15,   %xmm8,  %xmm1
501         palignr \$15,   %xmm8,  %xmm8
502         pxor    %xmm1,  %xmm7

504         # rotate
505         pshufd  \$0xFF, %xmm0,  %xmm0
506         palignr \$1,    %xmm0,  %xmm0

508         # fall through...

510         # low round: same as high round, but no rotation and no rcon.
511 _vpaes_schedule_low_round:
512         # smear xmm7
513         movdqa  %xmm7,  %xmm1
514         pslldq  \$4,    %xmm7
515         pxor    %xmm1,  %xmm7
516         movdqa  %xmm7,  %xmm1
517         pslldq  \$8,    %xmm7
518         pxor    %xmm1,  %xmm7
519         pxor    .Lk_s63(%rip), %xmm7

521         # subbytes
522         movdqa  %xmm9,  %xmm1
523         pandn   %xmm0,  %xmm1
```

```
524         psrld   \$4,    %xmm1           # 1 = i
525         pand    %xmm9,  %xmm0           # 0 = k
526         movdqa  %xmm11, %xmm2           # 2 : a/k
527         pshufb  %xmm0,  %xmm2           # 2 = a/k
528         pxor    %xmm1,  %xmm0           # 0 = j
529         movdqa  %xmm10, %xmm3           # 3 : 1/i
530         pshufb  %xmm1,  %xmm3           # 3 = 1/i
531         pxor    %xmm2,  %xmm3           # 3 = iak = 1/i + a/k
532         movdqa  %xmm10, %xmm4           # 4 : 1/j
533         pshufb  %xmm0,  %xmm4           # 4 = 1/j
534         pxor    %xmm2,  %xmm4           # 4 = jak = 1/j + a/k
535         movdqa  %xmm10, %xmm2           # 2 : 1/iak
536         pshufb  %xmm3,  %xmm2           # 2 = 1/iak
537         pxor    %xmm0,  %xmm2           # 2 = io
538         movdqa  %xmm10, %xmm3           # 3 : 1/jak
539         pshufb  %xmm4,  %xmm3           # 3 = 1/jak
540         pxor    %xmm1,  %xmm3           # 3 = jo
541         movdqa  %xmm13, %xmm4           # 4 : sbou
542         pshufb  %xmm2,  %xmm4           # 4 = sbou
543         movdqa  %xmm12, %xmm0           # 0 : sbot
544         pshufb  %xmm3,  %xmm0           # 0 = sb1t
545         pxor    %xmm4,  %xmm0           # 0 = sbox output

547         # add in smeared stuff
548         pxor    %xmm7,  %xmm0
549         movdqa  %xmm0,  %xmm7
550         ret
551 .size   _vpaes_schedule_round,.-_vpaes_schedule_round

553 ##
554 ##  .aes_schedule_transform
555 ##
556 ##  Linear-transform %xmm0 according to tables at (%r11)
557 ##
558 ##  Requires that %xmm9 = 0x0F0F... as in preheat
559 ##  Output in %xmm0
560 ##  Clobbers %xmm1, %xmm2
561 ##
562 .type   _vpaes_schedule_transform,\@abi-omnipotent
563 .align  16
564 _vpaes_schedule_transform:
565         movdqa  %xmm9,  %xmm1
566         pandn   %xmm0,  %xmm1
567         psrld   \$4,    %xmm1
568         pand    %xmm9,  %xmm0
569         movdqa  (%r11), %xmm2   # lo
570         pshufb  %xmm0,  %xmm2
571         movdqa  16(%r11), %xmm0 # hi
572         pshufb  %xmm1,  %xmm0
573         pxor    %xmm2,  %xmm0
574         ret
575 .size   _vpaes_schedule_transform,.-_vpaes_schedule_transform

577 ##
578 ##  .aes_schedule_mangle
579 ##
580 ##  Mangle xmm0 from (basis-transformed) standard version
581 ##  to our version.
582 ##
583 ##  On encrypt,
584 ##    xor with 0x63
585 ##    multiply by circulant 0,1,1,1
586 ##    apply shiftrows transform
587 ##
588 ##  On decrypt,
589 ##    xor with 0x63
```



```
590 ##    multiply by "inverse mixcolumns" circulant E,B,D,9
591 ##    deskew
592 ##    apply shiftrows transform
593 ##
594 ##
595 ##  Writes out to (%rdx), and increments or decrements it
596 ##  Keeps track of round number mod 4 in %r8
597 ##  Preserves xmm0
598 ##  Clobbers xmm1-xmm5
599 ##
600 .type   _vpaes_schedule_mangle,\@abi-omnipotent
601 .align  16
602 _vpaes_schedule_mangle:
603         movdqa  %xmm0,  %xmm4   # save xmm0 for later
604         movdqa  .Lk_mc_forward(%rip),%xmm5
605         test    %rcx,   %rcx
606         jnz     .Lschedule_mangle_dec

608         # encrypting
609         add     \$16,   %rdx
610         pxor    .Lk_s63(%rip),%xmm4
611         pshufb  %xmm5,  %xmm4
612         movdqa  %xmm4,  %xmm3
613         pshufb  %xmm5,  %xmm4
614         pxor    %xmm4,  %xmm3
615         pshufb  %xmm5,  %xmm4
616         pxor    %xmm4,  %xmm3

618         jmp     .Lschedule_mangle_both
619 .align  16
620 .Lschedule_mangle_dec:
621         # inverse mix columns
622         lea     .Lk_dksd(%rip),%r11
623         movdqa  %xmm9,  %xmm1
624         pandn   %xmm4,  %xmm1
625         psrld   \$4,    %xmm1   # 1 = hi
626         pand    %xmm9,  %xmm4   # 4 = lo

628         movdqa  0x00(%r11), %xmm2
629         pshufb  %xmm4,  %xmm2
630         movdqa  0x10(%r11), %xmm3
631         pshufb  %xmm1,  %xmm3
632         pxor    %xmm2,  %xmm3
633         pshufb  %xmm5,  %xmm3

635         movdqa  0x20(%r11), %xmm2
636         pshufb  %xmm4,  %xmm2
637         pxor    %xmm3,  %xmm2
638         movdqa  0x30(%r11), %xmm3
639         pshufb  %xmm1,  %xmm3
640         pxor    %xmm2,  %xmm3
641         pshufb  %xmm5,  %xmm3

643         movdqa  0x40(%r11), %xmm2
644         pshufb  %xmm4,  %xmm2
645         pxor    %xmm3,  %xmm2
646         movdqa  0x50(%r11), %xmm3
647         pshufb  %xmm1,  %xmm3
648         pxor    %xmm2,  %xmm3
649         pshufb  %xmm5,  %xmm3

651         movdqa  0x60(%r11), %xmm2
652         pshufb  %xmm4,  %xmm2
653         pxor    %xmm3,  %xmm2
654         movdqa  0x70(%r11), %xmm3
655         pshufb  %xmm1,  %xmm3
```

```
656         pxor    %xmm2,  %xmm3

658         add     \$-16,  %rdx

660 .Lschedule_mangle_both:
661         movdqa  (%r8,%r10),%xmm1
662         pshufb  %xmm1,%xmm3
663         add     \$-16,  %r8
664         and     \$0x30, %r8
665         movdqu  %xmm3,  (%rdx)
666         ret
667 .size   _vpaes_schedule_mangle,.-_vpaes_schedule_mangle

669 #
670 # Interface to OpenSSL
671 #
672 .globl  ${PREFIX}_set_encrypt_key
673 .type   ${PREFIX}_set_encrypt_key,\@function,3
674 .align  16
675 ${PREFIX}_set_encrypt_key:
676 ___
677 $code.=<<___ if ($win64);
678         lea     -0xb8(%rsp),%rsp
679         movaps  %xmm6,0x10(%rsp)
680         movaps  %xmm7,0x20(%rsp)
681         movaps  %xmm8,0x30(%rsp)
682         movaps  %xmm9,0x40(%rsp)
683         movaps  %xmm10,0x50(%rsp)
684         movaps  %xmm11,0x60(%rsp)
685         movaps  %xmm12,0x70(%rsp)
686         movaps  %xmm13,0x80(%rsp)
687         movaps  %xmm14,0x90(%rsp)
688         movaps  %xmm15,0xa0(%rsp)
689 .Lenc_key_body:
690 ___
691 $code.=<<___;
692         mov     %esi,%eax
693         shr     \$5,%eax
694         add     \$5,%eax
695         mov     %eax,240(%rdx)  # AES_KEY->rounds = nbits/32+5;

697         mov     \$0,%ecx
698         mov     \$0x30,%r8d
699         call    _vpaes_schedule_core
700 ___
701 $code.=<<___ if ($win64);
702         movaps  0x10(%rsp),%xmm6
703         movaps  0x20(%rsp),%xmm7
704         movaps  0x30(%rsp),%xmm8
705         movaps  0x40(%rsp),%xmm9
706         movaps  0x50(%rsp),%xmm10
707         movaps  0x60(%rsp),%xmm11
708         movaps  0x70(%rsp),%xmm12
709         movaps  0x80(%rsp),%xmm13
710         movaps  0x90(%rsp),%xmm14
711         movaps  0xa0(%rsp),%xmm15
712         lea     0xb8(%rsp),%rsp
713 .Lenc_key_epilogue:
714 ___
715 $code.=<<___;
716         xor     %eax,%eax
717         ret
718 .size   ${PREFIX}_set_encrypt_key,.-${PREFIX}_set_encrypt_key

720 .globl  ${PREFIX}_set_decrypt_key
721 .type   ${PREFIX}_set_decrypt_key,\@function,3
```



```
722 .align  16
723 ${PREFIX}_set_decrypt_key:
724 ___
725 $code.=<<___ if ($win64);
726         lea     -0xb8(%rsp),%rsp
727         movaps  %xmm6,0x10(%rsp)
728         movaps  %xmm7,0x20(%rsp)
729         movaps  %xmm8,0x30(%rsp)
730         movaps  %xmm9,0x40(%rsp)
731         movaps  %xmm10,0x50(%rsp)
732         movaps  %xmm11,0x60(%rsp)
733         movaps  %xmm12,0x70(%rsp)
734         movaps  %xmm13,0x80(%rsp)
735         movaps  %xmm14,0x90(%rsp)
736         movaps  %xmm15,0xa0(%rsp)
737 .Ldec_key_body:
738 ___
739 $code.=<<___;
740         mov     %esi,%eax
741         shr     \$5,%eax
742         add     \$5,%eax
743         mov     %eax,240(%rdx)  # AES_KEY->rounds = nbits/32+5;
744         shl     \$4,%eax
745         lea     16(%rdx,%rax),%rdx

747         mov     \$1,%ecx
748         mov     %esi,%r8d
749         shr     \$1,%r8d
750         and     \$32,%r8d
751         xor     \$32,%r8d       # nbits==192?0:32
752         call    _vpaes_schedule_core
753 ___
754 $code.=<<___ if ($win64);
755         movaps  0x10(%rsp),%xmm6
756         movaps  0x20(%rsp),%xmm7
757         movaps  0x30(%rsp),%xmm8
758         movaps  0x40(%rsp),%xmm9
759         movaps  0x50(%rsp),%xmm10
760         movaps  0x60(%rsp),%xmm11
761         movaps  0x70(%rsp),%xmm12
762         movaps  0x80(%rsp),%xmm13
763         movaps  0x90(%rsp),%xmm14
764         movaps  0xa0(%rsp),%xmm15
765         lea     0xb8(%rsp),%rsp
766 .Ldec_key_epilogue:
767 ___
768 $code.=<<___;
769         xor     %eax,%eax
770         ret
771 .size   ${PREFIX}_set_decrypt_key,.-${PREFIX}_set_decrypt_key

773 .globl  ${PREFIX}_encrypt
774 .type   ${PREFIX}_encrypt,\@function,3
775 .align  16
776 ${PREFIX}_encrypt:
777 ___
778 $code.=<<___ if ($win64);
779         lea     -0xb8(%rsp),%rsp
780         movaps  %xmm6,0x10(%rsp)
781         movaps  %xmm7,0x20(%rsp)
782         movaps  %xmm8,0x30(%rsp)
783         movaps  %xmm9,0x40(%rsp)
784         movaps  %xmm10,0x50(%rsp)
785         movaps  %xmm11,0x60(%rsp)
786         movaps  %xmm12,0x70(%rsp)
787         movaps  %xmm13,0x80(%rsp)
```

```
788         movaps  %xmm14,0x90(%rsp)
789         movaps  %xmm15,0xa0(%rsp)
790 .Lenc_body:
791 ___
792 $code.=<<___;
793         movdqu  (%rdi),%xmm0
794         call    _vpaes_preheat
795         call    _vpaes_encrypt_core
796         movdqu  %xmm0,(%rsi)
797 ___
798 $code.=<<___ if ($win64);
799         movaps  0x10(%rsp),%xmm6
800         movaps  0x20(%rsp),%xmm7
801         movaps  0x30(%rsp),%xmm8
802         movaps  0x40(%rsp),%xmm9
803         movaps  0x50(%rsp),%xmm10
804         movaps  0x60(%rsp),%xmm11
805         movaps  0x70(%rsp),%xmm12
806         movaps  0x80(%rsp),%xmm13
807         movaps  0x90(%rsp),%xmm14
808         movaps  0xa0(%rsp),%xmm15
809         lea     0xb8(%rsp),%rsp
810 .Lenc_epilogue:
811 ___
812 $code.=<<___;
813         ret
814 .size   ${PREFIX}_encrypt,.-${PREFIX}_encrypt

816 .globl  ${PREFIX}_decrypt
817 .type   ${PREFIX}_decrypt,\@function,3
818 .align  16
819 ${PREFIX}_decrypt:
820 ___
821 $code.=<<___ if ($win64);
822         lea     -0xb8(%rsp),%rsp
823         movaps  %xmm6,0x10(%rsp)
824         movaps  %xmm7,0x20(%rsp)
825         movaps  %xmm8,0x30(%rsp)
826         movaps  %xmm9,0x40(%rsp)
827         movaps  %xmm10,0x50(%rsp)
828         movaps  %xmm11,0x60(%rsp)
829         movaps  %xmm12,0x70(%rsp)
830         movaps  %xmm13,0x80(%rsp)
831         movaps  %xmm14,0x90(%rsp)
832         movaps  %xmm15,0xa0(%rsp)
833 .Ldec_body:
834 ___
835 $code.=<<___;
836         movdqu  (%rdi),%xmm0
837         call    _vpaes_preheat
838         call    _vpaes_decrypt_core
839         movdqu  %xmm0,(%rsi)
840 ___
841 $code.=<<___ if ($win64);
842         movaps  0x10(%rsp),%xmm6
843         movaps  0x20(%rsp),%xmm7
844         movaps  0x30(%rsp),%xmm8
845         movaps  0x40(%rsp),%xmm9
846         movaps  0x50(%rsp),%xmm10
847         movaps  0x60(%rsp),%xmm11
848         movaps  0x70(%rsp),%xmm12
849         movaps  0x80(%rsp),%xmm13
850         movaps  0x90(%rsp),%xmm14
851         movaps  0xa0(%rsp),%xmm15
852         lea     0xb8(%rsp),%rsp
853 .Ldec_epilogue:
```



```
854 ___
855 $code.=<<___;
856         ret
857 .size   ${PREFIX}_decrypt,.-${PREFIX}_decrypt
858 ___
859 {
860 my ($inp,$out,$len,$key,$ivp,$enc)=("%rdi","%rsi","%rdx","%rcx","%r8","%r9");
861 # void AES_cbc_encrypt (const void char *inp, unsigned char *out,
862 #                       size_t length, const AES_KEY *key,
863 #                       unsigned char *ivp,const int enc);
864 $code.=<<___;
865 .globl  ${PREFIX}_cbc_encrypt
866 .type   ${PREFIX}_cbc_encrypt,\@function,6
867 .align  16
868 ${PREFIX}_cbc_encrypt:
869         xchg    $key,$len
870 ___
871 ($len,$key)=($key,$len);
872 $code.=<<___;
873         sub     \$16,$len
874         jc      .Lcbc_abort
875 ___
876 $code.=<<___ if ($win64);
877         lea     -0xb8(%rsp),%rsp
878         movaps  %xmm6,0x10(%rsp)
879         movaps  %xmm7,0x20(%rsp)
880         movaps  %xmm8,0x30(%rsp)
881         movaps  %xmm9,0x40(%rsp)
882         movaps  %xmm10,0x50(%rsp)
883         movaps  %xmm11,0x60(%rsp)
884         movaps  %xmm12,0x70(%rsp)
885         movaps  %xmm13,0x80(%rsp)
886         movaps  %xmm14,0x90(%rsp)
887         movaps  %xmm15,0xa0(%rsp)
888 .Lcbc_body:
889 ___
890 $code.=<<___;
891         movdqu  ($ivp),%xmm6            # load IV
892         sub     $inp,$out
893         call    _vpaes_preheat
894         cmp     \$0,${enc}d
895         je      .Lcbc_dec_loop
896         jmp     .Lcbc_enc_loop
897 .align  16
898 .Lcbc_enc_loop:
899         movdqu  ($inp),%xmm0
900         pxor    %xmm6,%xmm0
901         call    _vpaes_encrypt_core
902         movdqa  %xmm0,%xmm6
903         movdqu  %xmm0,($out,$inp)
904         lea     16($inp),$inp
905         sub     \$16,$len
906         jnc     .Lcbc_enc_loop
907         jmp     .Lcbc_done
908 .align  16
909 .Lcbc_dec_loop:
910         movdqu  ($inp),%xmm0
911         movdqa  %xmm0,%xmm7
912         call    _vpaes_decrypt_core
913         pxor    %xmm6,%xmm0
914         movdqa  %xmm7,%xmm6
915         movdqu  %xmm0,($out,$inp)
916         lea     16($inp),$inp
917         sub     \$16,$len
918         jnc     .Lcbc_dec_loop
919 .Lcbc_done:
```

```
920         movdqu  %xmm6,($ivp)            # save IV
921 ___
922 $code.=<<___ if ($win64);
923         movaps  0x10(%rsp),%xmm6
924         movaps  0x20(%rsp),%xmm7
925         movaps  0x30(%rsp),%xmm8
926         movaps  0x40(%rsp),%xmm9
927         movaps  0x50(%rsp),%xmm10
928         movaps  0x60(%rsp),%xmm11
929         movaps  0x70(%rsp),%xmm12
930         movaps  0x80(%rsp),%xmm13
931         movaps  0x90(%rsp),%xmm14
932         movaps  0xa0(%rsp),%xmm15
933         lea     0xb8(%rsp),%rsp
934 .Lcbc_epilogue:
935 ___
936 $code.=<<___;
937 .Lcbc_abort:
938         ret
939 .size   ${PREFIX}_cbc_encrypt,.-${PREFIX}_cbc_encrypt
940 ___
941 }
942 $code.=<<___;
943 ##
944 ##  _aes_preheat
945 ##
946 ##  Fills register %r10 -> .aes_consts (so you can -fPIC)
947 ##  and %xmm9-%xmm15 as specified below.
948 ##
949 .type   _vpaes_preheat,\@abi-omnipotent
950 .align  16
951 _vpaes_preheat:
952         lea     .Lk_s0F(%rip), %r10
953         movdqa  -0x20(%r10), %xmm10     # .Lk_inv
954         movdqa  -0x10(%r10), %xmm11     # .Lk_inv+16
955         movdqa  0x00(%r10), %xmm9       # .Lk_s0F
956         movdqa  0x30(%r10), %xmm13      # .Lk_sb1
957         movdqa  0x40(%r10), %xmm12      # .Lk_sb1+16
958         movdqa  0x50(%r10), %xmm15      # .Lk_sb2
959         movdqa  0x60(%r10), %xmm14      # .Lk_sb2+16
960         ret
961 .size   _vpaes_preheat,.-_vpaes_preheat
962 ########################################################
963 ##                                                    ##
964 ##                     Constants                      ##
965 ##                                                    ##
966 ########################################################
967 .type   _vpaes_consts,\@object
968 .align  64
969 _vpaes_consts:
970 .Lk_inv:        # inv, inva
971         .quad   0x0E05060F0D080180, 0x040703090A0B0C02
972         .quad   0x01040A060F0B0780, 0x030D0E0C02050809

974 .Lk_s0F:        # s0F
975         .quad   0x0F0F0F0F0F0F0F0F, 0x0F0F0F0F0F0F0F0F

977 .Lk_ipt:        # input transform (lo, hi)
978         .quad   0xC2B2E8985A2A7000, 0xCABAE09052227808
979         .quad   0x4C01307D317C4D00, 0xCD80B1FCB0FDCC81

981 .Lk_sb1:        # sb1u, sb1t
982         .quad   0xB19BE18FCB503E00, 0xA5DF7A6E142AF544
983         .quad   0x3618D415FAE22300, 0x3BF7CCC10D2ED9EF
984 .Lk_sb2:        # sb2u, sb2t
985         .quad   0xE27A93C60B712400, 0x5EB7E955BC982FCD
```



```
986         .quad   0x69EB88400AE12900, 0xC2A163C8AB82234A
987 .Lk_sbo:        # sbou, sbot
988         .quad   0xD0D26D176FBDC700, 0x15AABF7AC502A878
989         .quad   0xCFE474A55FBB6A00, 0x8E1E90D1412B35FA

991 .Lk_mc_forward: # mc_forward
992         .quad   0x0407060500030201, 0x0C0F0E0D080B0A09
993         .quad   0x080B0A0904070605, 0x000302010C0F0E0D
994         .quad   0x0C0F0E0D080B0A09, 0x0407060500030201
995         .quad   0x000302010C0F0E0D, 0x080B0A0904070605

997 .Lk_mc_backward:# mc_backward
998         .quad   0x0605040702010003, 0x0E0D0C0F0A09080B
999         .quad   0x020100030E0D0C0F, 0x0A09080B06050407
1000        .quad   0x0E0D0C0F0A09080B, 0x0605040702010003
1001        .quad   0x0A09080B06050407, 0x020100030E0D0C0F

1003 .Lk_sr:         # sr
1004        .quad   0x0706050403020100, 0x0F0E0D0C0B0A0908
1005        .quad   0x030E09040F0A0500, 0x0B06010C07020D08
1006        .quad   0x0F060D040B020900, 0x070E050C030A0108
1007        .quad   0x0B0E0104070A0D00, 0x0306090C0F020508

1009 .Lk_rcon:       # rcon
1010        .quad   0x1F8391B9AF9DEEB6, 0x702A98084D7C7D81

1012 .Lk_s63:        # s63: all equal to 0x63 transformed
1013        .quad   0x5B5B5B5B5B5B5B5B, 0x5B5B5B5B5B5B5B5B

1015 .Lk_opt:        # output transform
1016        .quad   0xFF9F4929D6B66000, 0xF7974121DEBE6808
1017        .quad   0x01EDBD5150BCEC00, 0xE10D5DB1B05C0CE0

1019 .Lk_deskew:     # deskew tables: inverts the sbox's "skew"
1020        .quad   0x07E4A34047A4E300, 0x1DFEB95A5DBEF91A
1021        .quad   0x5F36B5DC83EA6900, 0x2841C2ABF49D1E77

1023 ##
1024 ##  Decryption stuff
1025 ##  Key schedule constants
1026 ##
1027 .Lk_dksd:       # decryption key schedule: invskew x*D
1028        .quad   0xFEB91A5DA3E44700, 0x0740E3A45A1DBEF9
1029        .quad   0x41C277F4B5368300, 0x5FDC69EAAB289D1E
1030 .Lk_dksb:       # decryption key schedule: invskew x*B
1031        .quad   0x9A4FCA1F8550D500, 0x03D653861CC94C99
1032        .quad   0x115BEDA7B6FC4A00, 0xD993256F7E3482C8
1033 .Lk_dkse:       # decryption key schedule: invskew x*E + 0x63
1034        .quad   0xD5031CCA1FC9D600, 0x53859A4C994F5086
1035        .quad   0xA23196054FDC7BE8, 0xCD5EF96A20B31487
1036 .Lk_dks9:       # decryption key schedule: invskew x*9
1037        .quad   0xB6116FC87ED9A700, 0x4AED933482255BFC
1038        .quad   0x4576516227143300, 0x8BB89FACE9DAFDCE

1040 ##
1041 ##  Decryption stuff
1042 ##  Round function constants
1043 ##
1044 .Lk_dipt:       # decryption input transform
1045        .quad   0x0F505B040B545F00, 0x154A411E114E451A
1046        .quad   0x86E383E660056500, 0x12771772F491F194

1048 .Lk_dsb9:       # decryption sbox output *9*u, *9*t
1049        .quad   0x851C03539A86D600, 0xCAD51F504F994CC9
1050        .quad   0xC03B1789ECD74900, 0x725E2C9EB2FBA565
1051 .Lk_dsbd:       # decryption sbox output *D*u, *D*t
```

```
1052         .quad   0x7D57CCDFE6B1A200, 0xF56E9B13882A4439
1053         .quad   0x3CE2FAF724C6CB00, 0x2931180D15DEEFD3
1054 .Lk_dsbb:       # decryption sbox output *B*u, *B*t
1055         .quad   0xD022649296B44200, 0x602646F6B0F2D404
1056         .quad   0xC19498A6CD596700, 0xF3FF0C3E3255AA6B
1057 .Lk_dsbe:       # decryption sbox output *E*u, *E*t
1058         .quad   0x46F2929626D4D000, 0x2242600464B4F6B0
1059         .quad   0x0C55A6CDFFAAC100, 0x9467F36B98593E32
1060 .Lk_dsbo:       # decryption sbox final output
1061         .quad   0x1387EA537EF94000, 0xC7AA6DB9D4943E2D
1062         .quad   0x12D7560F93441D00, 0xCA4B8159D8C58E9C
1063 .asciz  "Vector Permutation AES for x86_64/SSSE3, Mike Hamburg (Stanford Univers
1064 .align  64
1065 .size   _vpaes_consts,.-_vpaes_consts
1066 ___

1068 if ($win64) {
1069 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
1070 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
1071 $rec="%rcx";
1072 $frame="%rdx";
1073 $context="%r8";
1074 $disp="%r9";

1076 $code.=<<___;
1077 .extern __imp_RtlVirtualUnwind
1078 .type   se_handler,\@abi-omnipotent
1079 .align  16
1080 se_handler:
1081         push    %rsi
1082         push    %rdi
1083         push    %rbx
1084         push    %rbp
1085         push    %r12
1086         push    %r13
1087         push    %r14
1088         push    %r15
1089         pushfq
1090         sub     \$64,%rsp

1092         mov     120($context),%rax      # pull context->Rax
1093         mov     248($context),%rbx      # pull context->Rip

1095         mov     8($disp),%rsi           # disp->ImageBase
1096         mov     56($disp),%r11          # disp->HandlerData

1098         mov     0(%r11),%r10d           # HandlerData[0]
1099         lea     (%rsi,%r10),%r10        # prologue label
1100         cmp     %r10,%rbx               # context->Rip<prologue label
1101         jb      .Lin_prologue

1103         mov     152($context),%rax      # pull context->Rsp

1105         mov     4(%r11),%r10d           # HandlerData[1]
1106         lea     (%rsi,%r10),%r10        # epilogue label
1107         cmp     %r10,%rbx               # context->Rip>=epilogue label
1108         jae     .Lin_prologue

1110         lea     16(%rax),%rsi           # %xmm save area
1111         lea     512($context),%rdi      # &context.Xmm6
1112         mov     \$20,%ecx               # 10*sizeof(%xmm0)/sizeof(%rax)
1113         .long   0xa548f3fc              # cld; rep movsq
1114         lea     0xb8(%rax),%rax         # adjust stack pointer

1116 .Lin_prologue:
1117         mov     8(%rax),%rdi
```



```
1118         mov     16(%rax),%rsi
1119         mov     %rax,152($context)      # restore context->Rsp
1120         mov     %rsi,168($context)      # restore context->Rsi
1121         mov     %rdi,176($context)      # restore context->Rdi

1123         mov     40($disp),%rdi          # disp->ContextRecord
1124         mov     $context,%rsi           # context
1125         mov     \$‘1232/8‘,%ecx         # sizeof(CONTEXT)
1126         .long   0xa548f3fc              # cld; rep movsq

1128         mov     $disp,%rsi
1129         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
1130         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
1131         mov     0(%rsi),%r8             # arg3, disp->ControlPc
1132         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
1133         mov     40(%rsi),%r10           # disp->ContextRecord
1134         lea     56(%rsi),%r11           # &disp->HandlerData
1135         lea     24(%rsi),%r12           # &disp->EstablisherFrame
1136         mov     %r10,32(%rsp)           # arg5
1137         mov     %r11,40(%rsp)           # arg6
1138         mov     %r12,48(%rsp)           # arg7
1139         mov     %rcx,56(%rsp)           # arg8, (NULL)
1140         call    *__imp_RtlVirtualUnwind(%rip)

1142         mov     \$1,%eax                # ExceptionContinueSearch
1143         add     \$64,%rsp
1144         popfq
1145         pop     %r15
1146         pop     %r14
1147         pop     %r13
1148         pop     %r12
1149         pop     %rbp
1150         pop     %rbx
1151         pop     %rdi
1152         pop     %rsi
1153         ret
1154 .size   se_handler,.-se_handler

1156 .section        .pdata
1157 .align  4
1158         .rva    .LSEH_begin_${PREFIX}_set_encrypt_key
1159         .rva    .LSEH_end_${PREFIX}_set_encrypt_key
1160         .rva    .LSEH_info_${PREFIX}_set_encrypt_key

1162         .rva    .LSEH_begin_${PREFIX}_set_decrypt_key
1163         .rva    .LSEH_end_${PREFIX}_set_decrypt_key
1164         .rva    .LSEH_info_${PREFIX}_set_decrypt_key

1166         .rva    .LSEH_begin_${PREFIX}_encrypt
1167         .rva    .LSEH_end_${PREFIX}_encrypt
1168         .rva    .LSEH_info_${PREFIX}_encrypt

1170         .rva    .LSEH_begin_${PREFIX}_decrypt
1171         .rva    .LSEH_end_${PREFIX}_decrypt
1172         .rva    .LSEH_info_${PREFIX}_decrypt

1174         .rva    .LSEH_begin_${PREFIX}_cbc_encrypt
1175         .rva    .LSEH_end_${PREFIX}_cbc_encrypt
1176         .rva    .LSEH_info_${PREFIX}_cbc_encrypt

1178 .section        .xdata
1179 .align  8
1180 .LSEH_info_${PREFIX}_set_encrypt_key:
1181         .byte   9,0,0,0
1182         .rva    se_handler
1183         .rva    .Lenc_key_body,.Lenc_key_epilogue       # HandlerData[]
```

```
.LSEH_info_${PREFIX}_set_decrypt_key:
        .byte   9,0,0,0
        .rva    se_handler
        .rva    .Ldec_key_body,.Ldec_key_epilogue       # HandlerData[]
.LSEH_info_${PREFIX}_encrypt:
        .byte   9,0,0,0
        .rva    se_handler
        .rva    .Lenc_body,.Lenc_epilogue               # HandlerData[]
.LSEH_info_${PREFIX}_decrypt:
        .byte   9,0,0,0
        .rva    se_handler
        .rva    .Ldec_body,.Ldec_epilogue               # HandlerData[]
.LSEH_info_${PREFIX}_cbc_encrypt:
        .byte   9,0,0,0
        .rva    se_handler
        .rva    .Lcbc_body,.Lcbc_epilogue               # HandlerData[]
___
}

$code =~ s/\`([^\`]*)\`/eval($1)/gem;

print $code;

close STDOUT;
#endif /* ! codereview */
```

```
**********************************************************
    7713 Wed Aug 13 19:53:10 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86-gf2m.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl
#
# ====================================================================
# Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================
#
# May 2011
#
# The module implements bn_GF2m_mul_2x2 polynomial multiplication used
# in bn_gf2m.c. It's kind of low-hanging mechanical port from C for
# the time being... Except that it has three code paths: pure integer
# code suitable for any x86 CPU, MMX code suitable for PIII and later
# and PCLMULQDQ suitable for Westmere and later. Improvement varies
# from one benchmark and µ-arch to another. Below are interval values
# for 163- and 571-bit ECDH benchmarks relative to compiler-generated
# code:
#
# PIII          16%-30%
# P4            12%-12%
# Opteron       18%-40%
# Core2         19%-44%
# Atom          38%-64%
# Westmere      53%-121%(PCLMULQDQ)/20%-32%(MMX)
# Sandy Bridge  72%-127%(PCLMULQDQ)/27%-23%(MMX)
#
# Note that above improvement coefficients are not coefficients for
# bn_GF2m_mul_2x2 itself. For example 120% ECDH improvement is result
# of bn_GF2m_mul_2x2 being >4x faster. As it gets faster, benchmark
# is more and more dominated by other subroutines, most notably by
# BN_GF2m_mod[_mul]_arr...

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],$0,$x86only = $ARGV[$#ARGV] eq "386");

$sse2=0;
for (@ARGV) { $sse2=1 if (/-DOPENSSL_IA32_SSE2/); }

&external_label("OPENSSL_ia32cap_P") if ($sse2);

$a="eax";
$b="ebx";
($a1,$a2,$a4)=("ecx","edx","ebp");

$R="mm0";
@T=("mm1","mm2");
($A,$B,$B30,$B31)=("mm2","mm3","mm4","mm5");
@i=("esi","edi");

                                        if (!$x86only) {
&function_begin_B("_mul_1x1_mmx");
        &sub    ("esp",32+4);
         &mov   ($a1,$a);
         &lea   ($a2,&DWP(0,$a,$a));
         &and   ($a1,0x3fffffff);
         &lea   ($a4,&DWP(0,$a2,$a2));
```



```
         &mov   (&DWP(0*4,"esp"),0);
         &and   ($a2,0x7fffffff);
        &movd   ($A,$a);
        &movd   ($B,$b);
         &mov   (&DWP(1*4,"esp"),$a1);  # a1
         &xor   ($a1,$a2);              # a1^a2
        &pxor   ($B31,$B31);
        &pxor   ($B30,$B30);
         &mov   (&DWP(2*4,"esp"),$a2);  # a2
         &xor   ($a2,$a4);              # a2^a4
         &mov   (&DWP(3*4,"esp"),$a1);  # a1^a2
        &pcmpgtd($B31,$A);              # broadcast 31st bit
        &paddd  ($A,$A);                # $A<<=1
         &xor   ($a1,$a2);              # a1^a4=a1^a2^a2^a4
         &mov   (&DWP(4*4,"esp"),$a4);  # a4
         &xor   ($a4,$a2);              # a2=a4^a2^a4
        &pand   ($B31,$B);
        &pcmpgtd($B30,$A);              # broadcast 30th bit
         &mov   (&DWP(5*4,"esp"),$a1);  # a1^a4
         &xor   ($a4,$a1);              # a1^a2^a4
        &psllq  ($B31,31);
        &pand   ($B30,$B);
         &mov   (&DWP(6*4,"esp"),$a2);  # a2^a4
        &mov    (@i[0],0x7);
         &mov   (&DWP(7*4,"esp"),$a4);  # a1^a2^a4
         &mov   ($a4,@i[0]);
        &and    (@i[0],$b);
        &shr    ($b,3);
        &mov    (@i[1],$a4);
        &psllq  ($B30,30);
        &and    (@i[1],$b);
        &shr    ($b,3);
        &movd   ($R,&DWP(0,"esp",@i[0],4));
        &mov    (@i[0],$a4);
        &and    (@i[0],$b);
        &shr    ($b,3);
        for($n=1;$n<9;$n++) {
                &movd   (@T[1],&DWP(0,"esp",@i[1],4));
                &mov    (@i[1],$a4);
                &psllq  (@T[1],3*$n);
                &and    (@i[1],$b);
                &shr    ($b,3);
                &pxor   ($R,@T[1]);

                push(@i,shift(@i)); push(@T,shift(@T));
        }
        &movd   (@T[1],&DWP(0,"esp",@i[1],4));
        &pxor   ($R,$B30);
        &psllq  (@T[1],3*$n++);
        &pxor   ($R,@T[1]);

        &movd   (@T[0],&DWP(0,"esp",@i[0],4));
        &pxor   ($R,$B31);
        &psllq  (@T[0],3*$n);
        &add    ("esp",32+4);
        &pxor   ($R,@T[0]);
        &ret    ();
&function_end_B("_mul_1x1_mmx");
                                        }

($lo,$hi)=("eax","edx");
@T=("ecx","ebp");

&function_begin_B("_mul_1x1_ialu");
        &sub    ("esp",32+4);
         &mov   ($a1,$a);
```

```
128          &lea   ($a2,&DWP(0,$a,$a));
129          &lea   ($a4,&DWP(0,"",$a,4));
130          &and   ($a1,0x3fffffff);
131         &lea    (@i[1],&DWP(0,$lo,$lo));
132         &sar    ($lo,31);               # broadcast 31st bit
133          &mov   (&DWP(0*4,"esp"),0);
134          &and   ($a2,0x7fffffff);
135          &mov   (&DWP(1*4,"esp"),$a1);  # a1
136          &xor   ($a1,$a2);              # a1^a2
137          &mov   (&DWP(2*4,"esp"),$a2);  # a2
138          &xor   ($a2,$a4);              # a2^a4
139          &mov   (&DWP(3*4,"esp"),$a1);  # a1^a2
140          &xor   ($a1,$a2);              # a1^a4=a1^a2^a2^a4
141          &mov   (&DWP(4*4,"esp"),$a4);  # a4
142          &xor   ($a4,$a2);              # a2=a4^a2^a4
143          &mov   (&DWP(5*4,"esp"),$a1);  # a1^a4
144          &xor   ($a4,$a1);              # a1^a2^a4
145         &sar    (@i[1],31);             # broardcast 30th bit
146         &and    ($lo,$b);
147          &mov   (&DWP(6*4,"esp"),$a2);  # a2^a4
148         &and    (@i[1],$b);
149          &mov   (&DWP(7*4,"esp"),$a4);  # a1^a2^a4
150         &mov    ($hi,$lo);
151         &shl    ($lo,31);
152         &mov    (@T[0],@i[1]);
153         &shr    ($hi,1);

155          &mov   (@i[0],0x7);
156         &shl    (@i[1],30);
157          &and   (@i[0],$b);
158         &shr    (@T[0],2);
159         &xor    ($lo,@i[1]);

161         &shr    ($b,3);
162         &mov    (@i[1],0x7);            # 5-byte instruction!?
163         &and    (@i[1],$b);
164         &shr    ($b,3);
165          &xor   ($hi,@T[0]);
166         &xor    ($lo,&DWP(0,"esp",@i[0],4));
167         &mov    (@i[0],0x7);
168         &and    (@i[0],$b);
169         &shr    ($b,3);
170         for($n=1;$n<9;$n++) {
171                 &mov    (@T[1],&DWP(0,"esp",@i[1],4));
172                 &mov    (@i[1],0x7);
173                 &mov    (@T[0],@T[1]);
174                 &shl    (@T[1],3*$n);
175                 &and    (@i[1],$b);
176                 &shr    (@T[0],32-3*$n);
177                 &xor    ($lo,@T[1]);
178                 &shr    ($b,3);
179                 &xor    ($hi,@T[0]);

181                 push(@i,shift(@i)); push(@T,shift(@T));
182         }
183         &mov    (@T[1],&DWP(0,"esp",@i[1],4));
184         &mov    (@T[0],@T[1]);
185         &shl    (@T[1],3*$n);
186         &mov    (@i[1],&DWP(0,"esp",@i[0],4));
187         &shr    (@T[0],32-3*$n);        $n++;
188         &mov    (@i[0],@i[1]);
189         &xor    ($lo,@T[1]);
190         &shl    (@i[1],3*$n);
191         &xor    ($hi,@T[0]);
192         &shr    (@i[0],32-3*$n);
193         &xor    ($lo,@i[1]);
```



```
194         &xor    ($hi,@i[0]);

196         &add    ("esp",32+4);
197         &ret    ();
198 &function_end_B("_mul_1x1_ialu");

200 # void bn_GF2m_mul_2x2(BN_ULONG *r, BN_ULONG a1, BN_ULONG a0, BN_ULONG b1, BN_UL
201 &function_begin_B("bn_GF2m_mul_2x2");
202 if (!$x86only) {
203         &picmeup("edx","OPENSSL_ia32cap_P");
204         &mov    ("eax",&DWP(0,"edx"));
205         &mov    ("edx",&DWP(4,"edx"));
206         &test   ("eax",1<<23);          # check MMX bit
207         &jz     (&label("ialu"));
208 if ($sse2) {
209         &test   ("eax",1<<24);          # check FXSR bit
210         &jz     (&label("mmx"));
211         &test   ("edx",1<<1);           # check PCLMULQDQ bit
212         &jz     (&label("mmx"));

214         &movups         ("xmm0",&QWP(8,"esp"));
215         &shufps         ("xmm0","xmm0",0b10110001);
216         &pclmulqdq      ("xmm0","xmm0",1);
217         &mov            ("eax",&DWP(4,"esp"));
218         &movups         (&QWP(0,"eax"),"xmm0");
219         &ret    ();

221 &set_label("mmx",16);
222 }
223         &push   ("ebp");
224         &push   ("ebx");
225         &push   ("esi");
226         &push   ("edi");
227         &mov    ($a,&wparam(1));
228         &mov    ($b,&wparam(3));
229         &call   ("_mul_1x1_mmx");       # a1•b1
230         &movq   ("mm7",$R);

232         &mov    ($a,&wparam(2));
233         &mov    ($b,&wparam(4));
234         &call   ("_mul_1x1_mmx");       # a0•b0
235         &movq   ("mm6",$R);

237         &mov    ($a,&wparam(1));
238         &mov    ($b,&wparam(3));
239         &xor    ($a,&wparam(2));
240         &xor    ($b,&wparam(4));
241         &call   ("_mul_1x1_mmx");       # (a0+a1)•(b0+b1)
242         &pxor   ($R,"mm7");
243         &mov    ($a,&wparam(0));
244         &pxor   ($R,"mm6");             # (a0+a1)•(b0+b1)-a1•b1-a0•b0

246         &movq   ($A,$R);
247         &psllq  ($R,32);
248         &pop    ("edi");
249         &psrlq  ($A,32);
250         &pop    ("esi");
251         &pxor   ($R,"mm6");
252         &pop    ("ebx");
253         &pxor   ($A,"mm7");
254         &movq   (&QWP(0,$a),$R);
255         &pop    ("ebp");
256         &movq   (&QWP(8,$a),$A);
257         &emms   ();
258         &ret    ();
259 &set_label("ialu",16);
```

```
}
        &push   ("ebp");
        &push   ("ebx");
        &push   ("esi");
        &push   ("edi");
        &stack_push(4+1);

        &mov    ($a,&wparam(1));
        &mov    ($b,&wparam(3));
        &call   ("_mul_1x1_ialu");      # a1•b1
        &mov    (&DWP(8,"esp"),$lo);
        &mov    (&DWP(12,"esp"),$hi);

        &mov    ($a,&wparam(2));
        &mov    ($b,&wparam(4));
        &call   ("_mul_1x1_ialu");      # a0•b0
        &mov    (&DWP(0,"esp"),$lo);
        &mov    (&DWP(4,"esp"),$hi);

        &mov    ($a,&wparam(1));
        &mov    ($b,&wparam(3));
        &xor    ($a,&wparam(2));
        &xor    ($b,&wparam(4));
        &call   ("_mul_1x1_ialu");      # (a0+a1)•(b0+b1)

        &mov    ("ebp",&wparam(0));
                 @r=("ebx","ecx","edi","esi");
        &mov    (@r[0],&DWP(0,"esp"));
        &mov    (@r[1],&DWP(4,"esp"));
        &mov    (@r[2],&DWP(8,"esp"));
        &mov    (@r[3],&DWP(12,"esp"));

        &xor    ($lo,$hi);
        &xor    ($hi,@r[1]);
        &xor    ($lo,@r[0]);
        &mov    (&DWP(0,"ebp"),@r[0]);
        &xor    ($hi,@r[2]);
        &mov    (&DWP(12,"ebp"),@r[3]);
        &xor    ($lo,@r[3]);
        &stack_pop(4+1);
        &xor    ($hi,@r[3]);
        &pop    ("edi");
        &xor    ($lo,$hi);
        &pop    ("esi");
        &mov    (&DWP(8,"ebp"),$hi);
        &pop    ("ebx");
        &mov    (&DWP(4,"ebp"),$lo);
        &pop    ("ebp");
        &ret    ();
&function_end_B("bn_GF2m_mul_2x2");

&asciz  ("GF(2^m) Multiplication for x86, CRYPTOGAMS by <appro\@openssl.org>");

&asm_finish();
#endif /* ! codereview */
```

```
**********************************************************
   16468 Wed Aug 13 19:53:10 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86-mont.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# ====================================================================
# Written by Andy Polyakov <appro@fy.chalmers.se> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================

# October 2005
#
# This is a "teaser" code, as it can be improved in several ways...
# First of all non-SSE2 path should be implemented (yes, for now it
# performs Montgomery multiplication/convolution only on SSE2-capable
# CPUs such as P4, others fall down to original code). Then inner loop
# can be unrolled and modulo-scheduled to improve ILP and possibly
# moved to 128-bit XMM register bank (though it would require input
# rearrangement and/or increase bus bandwidth utilization). Dedicated
# squaring procedure should give further performance improvement...
# Yet, for being draft, the code improves rsa512 *sign* benchmark by
# 110%(!), rsa1024 one - by 70% and rsa4096 - by 20%:-)

# December 2006
#
# Modulo-scheduling SSE2 loops results in further 15-20% improvement.
# Integer-only code [being equipped with dedicated squaring procedure]
# gives ~40% on rsa512 sign benchmark...

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC,"${dir}","${dir}../../perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],$0);

$sse2=0;
for (@ARGV) { $sse2=1 if (/-DOPENSSL_IA32_SSE2/); }

&external_label("OPENSSL_ia32cap_P") if ($sse2);

&function_begin("bn_mul_mont");

$i="edx";
$j="ecx";
$ap="esi";      $tp="esi";              # overlapping variables!!!
$rp="edi";      $bp="edi";              # overlapping variables!!!
$np="ebp";
$num="ebx";

$_num=&DWP(4*0,"esp");                  # stack top layout
$_rp=&DWP(4*1,"esp");
$_ap=&DWP(4*2,"esp");
$_bp=&DWP(4*3,"esp");
$_np=&DWP(4*4,"esp");
$_n0=&DWP(4*5,"esp");   $_n0q=&QWP(4*5,"esp");
$_sp=&DWP(4*6,"esp");
$_bpend=&DWP(4*7,"esp");
$frame=32;                              # size of above frame rounded up to 16n

        &xor    ("eax","eax");
        &mov    ("edi",&wparam(5));     # int num
        &cmp    ("edi",4);
```



```
        &jl     (&label("just_leave"));

        &lea    ("esi",&wparam(0));     # put aside pointer to argument block
        &lea    ("edx",&wparam(1));     # load ap
        &mov    ("ebp","esp");          # saved stack pointer!
        &add    ("edi",2);              # extra two words on top of tp
        &neg    ("edi");
        &lea    ("esp",&DWP(-$frame,"esp","edi",4));    # alloca($frame+4*(num+2
        &neg    ("edi");

        # minimize cache contention by arraning 2K window between stack
        # pointer and ap argument [np is also position sensitive vector,
        # but it's assumed to be near ap, as it's allocated at ~same
        # time].
        &mov    ("eax","esp");
        &sub    ("eax","edx");
        &and    ("eax",2047);
        &sub    ("esp","eax");          # this aligns sp and ap modulo 2048

        &xor    ("edx","esp");
        &and    ("edx",2048);
        &xor    ("edx",2048);
        &sub    ("esp","edx");          # this splits them apart modulo 4096

        &and    ("esp",-64);            # align to cache line

        ################################ load argument block...
        &mov    ("eax",&DWP(0*4,"esi"));# BN_ULONG *rp
        &mov    ("ebx",&DWP(1*4,"esi"));# const BN_ULONG *ap
        &mov    ("ecx",&DWP(2*4,"esi"));# const BN_ULONG *bp
        &mov    ("edx",&DWP(3*4,"esi"));# const BN_ULONG *np
        &mov    ("esi",&DWP(4*4,"esi"));# const BN_ULONG *n0
        #&mov   ("edi",&DWP(5*4,"esi"));# int num

        &mov    ("esi",&DWP(0,"esi"));  # pull n0[0]
        &mov    ($_rp,"eax");           # ... save a copy of argument block
        &mov    ($_ap,"ebx");
        &mov    ($_bp,"ecx");
        &mov    ($_np,"edx");
        &mov    ($_n0,"esi");
        &lea    ($num,&DWP(-3,"edi"));  # num=num-1 to assist modulo-scheduling
        #&mov   ($_num,$num);           # redundant as $num is not reused
        &mov    ($_sp,"ebp");           # saved stack pointer!
```

```
105 if($sse2) {
106 $acc0="mm0";    # mmx register bank layout
107 $acc1="mm1";
108 $car0="mm2";
109 $car1="mm3";
110 $mul0="mm4";
111 $mul1="mm5";
112 $temp="mm6";
113 $mask="mm7";

115         &picmeup("eax","OPENSSL_ia32cap_P");
116         &bt     (&DWP(0,"eax"),26);
117         &jnc    (&label("non_sse2"));

119         &mov    ("eax",-1);
120         &movd   ($mask,"eax");          # mask 32 lower bits

122         &mov    ($ap,$_ap);             # load input pointers
123         &mov    ($bp,$_bp);
124         &mov    ($np,$_np);

126         &xor    ($i,$i);                # i=0
127         &xor    ($j,$j);                # j=0

129         &movd   ($mul0,&DWP(0,$bp));            # bp[0]
130         &movd   ($mul1,&DWP(0,$ap));            # ap[0]
131         &movd   ($car1,&DWP(0,$np));            # np[0]

133         &pmuludq($mul1,$mul0);                  # ap[0]*bp[0]
134         &movq   ($car0,$mul1);
135         &movq   ($acc0,$mul1);                  # I wish movd worked for
136         &pand   ($acc0,$mask);                  # inter-register transfers

138         &pmuludq($mul1,$_n0q);                  # *=n0

140         &pmuludq($car1,$mul1);                  # "t[0]"*np[0]*n0
141         &paddq  ($car1,$acc0);

143         &movd   ($acc1,&DWP(4,$np));            # np[1]
144         &movd   ($acc0,&DWP(4,$ap));            # ap[1]

146         &psrlq  ($car0,32);
147         &psrlq  ($car1,32);

149         &inc    ($j);                           # j++
150 &set_label("1st",16);
151         &pmuludq($acc0,$mul0);                  # ap[j]*bp[0]
152         &pmuludq($acc1,$mul1);                  # np[j]*m1
153         &paddq  ($car0,$acc0);                  # +=c0
154         &paddq  ($car1,$acc1);                  # +=c1

156         &movq   ($acc0,$car0);
157         &pand   ($acc0,$mask);
158         &movd   ($acc1,&DWP(4,$np,$j,4));       # np[j+1]
159         &paddq  ($car1,$acc0);                  # +=ap[j]*bp[0];
160         &movd   ($acc0,&DWP(4,$ap,$j,4));       # ap[j+1]
161         &psrlq  ($car0,32);
162         &movd   (&DWP($frame-4,"esp",$j,4),$car1);      # tp[j-1]=
163         &psrlq  ($car1,32);

165         &lea    ($j,&DWP(1,$j));
166         &cmp    ($j,$num);
167         &jl     (&label("1st"));

169         &pmuludq($acc0,$mul0);                  # ap[num-1]*bp[0]
170         &pmuludq($acc1,$mul1);                  # np[num-1]*m1
```



```
171         &paddq  ($car0,$acc0);                  # +=c0
172         &paddq  ($car1,$acc1);                  # +=c1

174         &movq   ($acc0,$car0);
175         &pand   ($acc0,$mask);
176         &paddq  ($car1,$acc0);                  # +=ap[num-1]*bp[0];
177         &movd   (&DWP($frame-4,"esp",$j,4),$car1);      # tp[num-2]=

179         &psrlq  ($car0,32);
180         &psrlq  ($car1,32);

182         &paddq  ($car1,$car0);
183         &movq   (&QWP($frame,"esp",$num,4),$car1);      # tp[num].tp[num-1]
```

```
184         &inc    ($i);                           # i++
185 &set_label("outer");
186         &xor    ($j,$j);                        # j=0

188         &movd   ($mul0,&DWP(0,$bp,$i,4));       # bp[i]
189         &movd   ($mul1,&DWP(0,$ap));            # ap[0]
190         &movd   ($temp,&DWP($frame,"esp"));     # tp[0]
191         &movd   ($car1,&DWP(0,$np));            # np[0]
192         &pmuludq($mul1,$mul0);                  # ap[0]*bp[i]

194         &paddq  ($mul1,$temp);                  # +=tp[0]
195         &movq   ($acc0,$mul1);
196         &movq   ($car0,$mul1);
197         &pand   ($acc0,$mask);

199         &pmuludq($mul1,$_n0q);                  # *=n0

201         &pmuludq($car1,$mul1);
202         &paddq  ($car1,$acc0);

204         &movd   ($temp,&DWP($frame+4,"esp"));   # tp[1]
205         &movd   ($acc1,&DWP(4,$np));            # np[1]
206         &movd   ($acc0,&DWP(4,$ap));            # ap[1]

208         &psrlq  ($car0,32);
209         &psrlq  ($car1,32);
210         &paddq  ($car0,$temp);                  # +=tp[1]

212         &inc    ($j);                           # j++
213         &dec    ($num);
214 &set_label("inner");
215         &pmuludq($acc0,$mul0);                  # ap[j]*bp[i]
216         &pmuludq($acc1,$mul1);                  # np[j]*m1
217         &paddq  ($car0,$acc0);                  # +=c0
218         &paddq  ($car1,$acc1);                  # +=c1

220         &movq   ($acc0,$car0);
221         &movd   ($temp,&DWP($frame+4,"esp",$j,4));# tp[j+1]
222         &pand   ($acc0,$mask);
223         &movd   ($acc1,&DWP(4,$np,$j,4));       # np[j+1]
224         &paddq  ($car1,$acc0);                  # +=ap[j]*bp[i]+tp[j]
225         &movd   ($acc0,&DWP(4,$ap,$j,4));       # ap[j+1]
226         &psrlq  ($car0,32);
227         &movd   (&DWP($frame-4,"esp",$j,4),$car1);# tp[j-1]=
228         &psrlq  ($car1,32);
229         &paddq  ($car0,$temp);                  # +=tp[j+1]

231         &dec    ($num);
232         &lea    ($j,&DWP(1,$j));                # j++
233         &jnz    (&label("inner"));

235         &mov    ($num,$j);
236         &pmuludq($acc0,$mul0);                  # ap[num-1]*bp[i]
237         &pmuludq($acc1,$mul1);                  # np[num-1]*m1
238         &paddq  ($car0,$acc0);                  # +=c0
239         &paddq  ($car1,$acc1);                  # +=c1

241         &movq   ($acc0,$car0);
242         &pand   ($acc0,$mask);
243         &paddq  ($car1,$acc0);                  # +=ap[num-1]*bp[i]+tp[num-1]
244         &movd   (&DWP($frame-4,"esp",$j,4),$car1);      # tp[num-2]=
245         &psrlq  ($car0,32);
246         &psrlq  ($car1,32);

248         &movd   ($temp,&DWP($frame+4,"esp",$num,4));    # += tp[num]
249         &paddq  ($car1,$car0);
```



```
250         &paddq  ($car1,$temp);
251         &movq   (&QWP($frame,"esp",$num,4),$car1);      # tp[num].tp[num-1]

253         &lea    ($i,&DWP(1,$i));                # i++
254         &cmp    ($i,$num);
255         &jle    (&label("outer"));

257         &emms   ();                             # done with mmx bank
258         &jmp    (&label("common_tail"));

260 &set_label("non_sse2",16);
261 }
```

```
262 if (0) {
263         &mov    ("esp",$_sp);
264         &xor    ("eax","eax");  # signal "not fast enough [yet]"
265         &jmp    (&label("just_leave"));
266         # While the below code provides competitive performance for
267         # all key lengthes on modern Intel cores, it's still more
268         # than 10% slower for 4096-bit key elsewhere:-( "Competitive"
269         # means compared to the original integer-only assembler.
270         # 512-bit RSA sign is better by ~40%, but that's about all
271         # one can say about all CPUs...
272 } else {
273 $inp="esi";     # integer path uses these registers differently
274 $word="edi";
275 $carry="ebp";

277         &mov    ($inp,$_ap);
278         &lea    ($carry,&DWP(1,$num));
279         &mov    ($word,$_bp);
280         &xor    ($j,$j);                                # j=0
281         &mov    ("edx",$inp);
282         &and    ($carry,1);                             # see if num is even
283         &sub    ("edx",$word);                          # see if ap==bp
284         &lea    ("eax",&DWP(4,$word,$num,4));           # &bp[num]
285         &or     ($carry,"edx");
286         &mov    ($word,&DWP(0,$word));                  # bp[0]
287         &jz     (&label("bn_sqr_mont"));
288         &mov    ($_bpend,"eax");
289         &mov    ("eax",&DWP(0,$inp));
290         &xor    ("edx","edx");

292 &set_label("mull",16);
293         &mov    ($carry,"edx");
294         &mul    ($word);                                # ap[j]*bp[0]
295         &add    ($carry,"eax");
296         &lea    ($j,&DWP(1,$j));
297         &adc    ("edx",0);
298         &mov    ("eax",&DWP(0,$inp,$j,4));              # ap[j+1]
299         &cmp    ($j,$num);
300         &mov    (&DWP($frame-4,"esp",$j,4),$carry);     # tp[j]=
301         &jl     (&label("mull"));

303         &mov    ($carry,"edx");
304         &mul    ($word);                                # ap[num-1]*bp[0]
305          &mov   ($word,$_n0);
306         &add    ("eax",$carry);
307          &mov   ($inp,$_np);
308         &adc    ("edx",0);
309          &imul  ($word,&DWP($frame,"esp"));             # n0*tp[0]

311         &mov    (&DWP($frame,"esp",$num,4),"eax");      # tp[num-1]=
312         &xor    ($j,$j);
313         &mov    (&DWP($frame+4,"esp",$num,4),"edx");    # tp[num]=
314         &mov    (&DWP($frame+8,"esp",$num,4),$j);       # tp[num+1]=

316         &mov    ("eax",&DWP(0,$inp));                   # np[0]
317         &mul    ($word);                                # np[0]*m
318         &add    ("eax",&DWP($frame,"esp"));             # +=tp[0]
319         &mov    ("eax",&DWP(4,$inp));                   # np[1]
320         &adc    ("edx",0);
321         &inc    ($j);

323         &jmp    (&label("2ndmadd"));
```

```
&set_label("1stmadd",16);
        &mov    ($carry,"edx");
        &mul    ($word);                        # ap[j]*bp[i]
        &add    ($carry,&DWP($frame,"esp",$j,4));       # +=tp[j]
        &lea    ($j,&DWP(1,$j));
        &adc    ("edx",0);
        &add    ($carry,"eax");
        &mov    ("eax",&DWP(0,$inp,$j,4));              # ap[j+1]
        &adc    ("edx",0);
        &cmp    ($j,$num);
        &mov    (&DWP($frame-4,"esp",$j,4),$carry);     # tp[j]=
        &jl     (&label("1stmadd"));

        &mov    ($carry,"edx");
        &mul    ($word);                        # ap[num-1]*bp[i]
        &add    ("eax",&DWP($frame,"esp",$num,4));      # +=tp[num-1]
         &mov   ($word,$_n0);
        &adc    ("edx",0);
         &mov   ($inp,$_np);
        &add    ($carry,"eax");
        &adc    ("edx",0);
         &imul  ($word,&DWP($frame,"esp"));             # n0*tp[0]

        &xor    ($j,$j);
        &add    ("edx",&DWP($frame+4,"esp",$num,4));    # carry+=tp[num]
        &mov    (&DWP($frame,"esp",$num,4),$carry);     # tp[num-1]=
        &adc    ($j,0);
         &mov   ("eax",&DWP(0,$inp));                   # np[0]
        &mov    (&DWP($frame+4,"esp",$num,4),"edx");    # tp[num]=
        &mov    (&DWP($frame+8,"esp",$num,4),$j);       # tp[num+1]=

        &mul    ($word);                        # np[0]*m
        &add    ("eax",&DWP($frame,"esp"));             # +=tp[0]
        &mov    ("eax",&DWP(4,$inp));                   # np[1]
        &adc    ("edx",0);
        &mov    ($j,1);
```



```
&set_label("2ndmadd",16);
        &mov    ($carry,"edx");
        &mul    ($word);                        # np[j]*m
        &add    ($carry,&DWP($frame,"esp",$j,4));       # +=tp[j]
        &lea    ($j,&DWP(1,$j));
        &adc    ("edx",0);
        &add    ($carry,"eax");
        &mov    ("eax",&DWP(0,$inp,$j,4));              # np[j+1]
        &adc    ("edx",0);
        &cmp    ($j,$num);
        &mov    (&DWP($frame-8,"esp",$j,4),$carry);     # tp[j-1]=
        &jl     (&label("2ndmadd"));

        &mov    ($carry,"edx");
        &mul    ($word);                        # np[j]*m
        &add    ($carry,&DWP($frame,"esp",$num,4));     # +=tp[num-1]
        &adc    ("edx",0);
        &add    ($carry,"eax");
        &adc    ("edx",0);
        &mov    (&DWP($frame-4,"esp",$num,4),$carry);   # tp[num-2]=

        &xor    ("eax","eax");
         &mov   ($j,$_bp);                      # &bp[i]
        &add    ("edx",&DWP($frame+4,"esp",$num,4));    # carry+=tp[num]
        &adc    ("eax",&DWP($frame+8,"esp",$num,4));    # +=tp[num+1]
         &lea   ($j,&DWP(4,$j));
        &mov    (&DWP($frame,"esp",$num,4),"edx");      # tp[num-1]=
         &cmp   ($j,$_bpend);
        &mov    (&DWP($frame+4,"esp",$num,4),"eax");    # tp[num]=
        &je     (&label("common_tail"));

        &mov    ($word,&DWP(0,$j));                     # bp[i+1]
        &mov    ($inp,$_ap);
        &mov    ($_bp,$j);                      # &bp[++i]
        &xor    ($j,$j);
        &xor    ("edx","edx");
        &mov    ("eax",&DWP(0,$inp));
        &jmp    (&label("1stmadd"));
```

```
398 &set_label("bn_sqr_mont",16);
399 $sbit=$num;
400         &mov    ($_num,$num);
401         &mov    ($_bp,$j);                              # i=0

403         &mov    ("eax",$word);                          # ap[0]
404         &mul    ($word);                                # ap[0]*ap[0]
405         &mov    (&DWP($frame,"esp"),"eax");             # tp[0]=
406         &mov    ($sbit,"edx");
407         &shr    ("edx",1);
408         &and    ($sbit,1);
409         &inc    ($j);
410 &set_label("sqr",16);
411         &mov    ("eax",&DWP(0,$inp,$j,4));              # ap[j]
412         &mov    ($carry,"edx");
413         &mul    ($word);                                # ap[j]*ap[0]
414         &add    ("eax",$carry);
415         &lea    ($j,&DWP(1,$j));
416         &adc    ("edx",0);
417         &lea    ($carry,&DWP(0,$sbit,"eax",2));
418         &shr    ("eax",31);
419         &cmp    ($j,$_num);
420         &mov    ($sbit,"eax");
421         &mov    (&DWP($frame-4,"esp",$j,4),$carry);     # tp[j]=
422         &jl     (&label("sqr"));

424         &mov    ("eax",&DWP(0,$inp,$j,4));              # ap[num-1]
425         &mov    ($carry,"edx");
426         &mul    ($word);                                # ap[num-1]*ap[0]
427         &add    ("eax",$carry);
428          &mov   ($word,$_n0);
429         &adc    ("edx",0);
430          &mov   ($inp,$_np);
431         &lea    ($carry,&DWP(0,$sbit,"eax",2));
432          &imul  ($word,&DWP($frame,"esp"));             # n0*tp[0]
433         &shr    ("eax",31);
434         &mov    (&DWP($frame,"esp",$j,4),$carry);       # tp[num-1]=

436         &lea    ($carry,&DWP(0,"eax","edx",2));
437          &mov   ("eax",&DWP(0,$inp));                   # np[0]
438         &shr    ("edx",31);
439         &mov    (&DWP($frame+4,"esp",$j,4),$carry);     # tp[num]=
440         &mov    (&DWP($frame+8,"esp",$j,4),"edx");      # tp[num+1]=

442         &mul    ($word);                                # np[0]*m
443         &add    ("eax",&DWP($frame,"esp"));             # +=tp[0]
444         &mov    ($num,$j);
445         &adc    ("edx",0);
446         &mov    ("eax",&DWP(4,$inp));                   # np[1]
447         &mov    ($j,1);
```

```
448 &set_label("3rdmadd",16);
449         &mov    ($carry,"edx");
450         &mul    ($word);                        # np[j]*m
451         &add    ($carry,&DWP($frame,"esp",$j,4));       # +=tp[j]
452         &adc    ("edx",0);
453         &add    ($carry,"eax");
454         &mov    ("eax",&DWP(4,$inp,$j,4));              # np[j+1]
455         &adc    ("edx",0);
456         &mov    (&DWP($frame-4,"esp",$j,4),$carry);     # tp[j-1]=

458         &mov    ($carry,"edx");
459         &mul    ($word);                        # np[j+1]*m
460         &add    ($carry,&DWP($frame+4,"esp",$j,4));     # +=tp[j+1]
461         &lea    ($j,&DWP(2,$j));
462         &adc    ("edx",0);
463         &add    ($carry,"eax");
464         &mov    ("eax",&DWP(0,$inp,$j,4));              # np[j+2]
465         &adc    ("edx",0);
466         &cmp    ($j,$num);
467         &mov    (&DWP($frame-8,"esp",$j,4),$carry);     # tp[j]=
468         &jl     (&label("3rdmadd"));

470         &mov    ($carry,"edx");
471         &mul    ($word);                        # np[j]*m
472         &add    ($carry,&DWP($frame,"esp",$num,4));     # +=tp[num-1]
473         &adc    ("edx",0);
474         &add    ($carry,"eax");
475         &adc    ("edx",0);
476         &mov    (&DWP($frame-4,"esp",$num,4),$carry);   # tp[num-2]=

478         &mov    ($j,$_bp);                      # i
479         &xor    ("eax","eax");
480         &mov    ($inp,$_ap);
481         &add    ("edx",&DWP($frame+4,"esp",$num,4));    # carry+=tp[num]
482         &adc    ("eax",&DWP($frame+8,"esp",$num,4));    # +=tp[num+1]
483         &mov    (&DWP($frame,"esp",$num,4),"edx");      # tp[num-1]=
484         &cmp    ($j,$num);
485         &mov    (&DWP($frame+4,"esp",$num,4),"eax");    # tp[num]=
486         &je     (&label("common_tail"));
```



```
487         &mov    ($word,&DWP(4,$inp,$j,4));              # ap[i]
488         &lea    ($j,&DWP(1,$j));
489         &mov    ("eax",$word);
490         &mov    ($_bp,$j);                      # ++i
491         &mul    ($word);                        # ap[i]*ap[i]
492         &add    ("eax",&DWP($frame,"esp",$j,4));        # +=tp[i]
493         &adc    ("edx",0);
494         &mov    (&DWP($frame,"esp",$j,4),"eax");        # tp[i]=
495         &xor    ($carry,$carry);
496         &cmp    ($j,$num);
497         &lea    ($j,&DWP(1,$j));
498         &je     (&label("sqrlast"));

500         &mov    ($sbit,"edx");                  # zaps $num
501         &shr    ("edx",1);
502         &and    ($sbit,1);
503 &set_label("sqradd",16);
504         &mov    ("eax",&DWP(0,$inp,$j,4));              # ap[j]
505         &mov    ($carry,"edx");
506         &mul    ($word);                        # ap[j]*ap[i]
507         &add    ("eax",$carry);
508         &lea    ($carry,&DWP(0,"eax","eax"));
509         &adc    ("edx",0);
510         &shr    ("eax",31);
511         &add    ($carry,&DWP($frame,"esp",$j,4));       # +=tp[j]
512         &lea    ($j,&DWP(1,$j));
513         &adc    ("eax",0);
514         &add    ($carry,$sbit);
515         &adc    ("eax",0);
516         &cmp    ($j,$_num);
517         &mov    (&DWP($frame-4,"esp",$j,4),$carry);     # tp[j]=
518         &mov    ($sbit,"eax");
519         &jle    (&label("sqradd"));

521         &mov    ($carry,"edx");
522         &add    ("edx","edx");
523         &shr    ($carry,31);
524         &add    ("edx",$sbit);
525         &adc    ($carry,0);
526 &set_label("sqrlast");
527         &mov    ($word,$_n0);
528         &mov    ($inp,$_np);
529         &imul   ($word,&DWP($frame,"esp"));             # n0*tp[0]

531         &add    ("edx",&DWP($frame,"esp",$j,4));        # +=tp[num]
532         &mov    ("eax",&DWP(0,$inp));                   # np[0]
533         &adc    ($carry,0);
534         &mov    (&DWP($frame,"esp",$j,4),"edx");        # tp[num]=
535         &mov    (&DWP($frame+4,"esp",$j,4),$carry);     # tp[num+1]=

537         &mul    ($word);                        # np[0]*m
538         &add    ("eax",&DWP($frame,"esp"));             # +=tp[0]
539         &lea    ($num,&DWP(-1,$j));
540         &adc    ("edx",0);
541         &mov    ($j,1);
542         &mov    ("eax",&DWP(4,$inp));                   # np[1]

544         &jmp    (&label("3rdmadd"));
545 }
```

```
546 &set_label("common_tail",16);
547         &mov    ($np,$_np);                     # load modulus pointer
548         &mov    ($rp,$_rp);                     # load result pointer
549         &lea    ($tp,&DWP($frame,"esp"));       # [$ap and $bp are zapped]

551         &mov    ("eax",&DWP(0,$tp));            # tp[0]
552         &mov    ($j,$num);                      # j=num-1
553         &xor    ($i,$i);                        # i=0 and clear CF!

555 &set_label("sub",16);
556         &sbb    ("eax",&DWP(0,$np,$i,4));
557         &mov    (&DWP(0,$rp,$i,4),"eax");       # rp[i]=tp[i]-np[i]
558         &dec    ($j);                           # doesn't affect CF!
559         &mov    ("eax",&DWP(4,$tp,$i,4));       # tp[i+1]
560         &lea    ($i,&DWP(1,$i));                # i++
561         &jge    (&label("sub"));

563         &sbb    ("eax",0);                      # handle upmost overflow bit
564         &and    ($tp,"eax");
565         &not    ("eax");
566         &mov    ($np,$rp);
567         &and    ($np,"eax");
568         &or     ($tp,$np);                      # tp=carry?tp:rp

570 &set_label("copy",16);                          # copy or in-place refresh
571         &mov    ("eax",&DWP(0,$tp,$num,4));
572         &mov    (&DWP(0,$rp,$num,4),"eax");     # rp[i]=tp[i]
573         &mov    (&DWP($frame,"esp",$num,4),$j); # zap temporary vector
574         &dec    ($num);
575         &jge    (&label("copy"));

577         &mov    ("esp",$_sp);           # pull saved stack pointer
578         &mov    ("eax",1);
579 &set_label("just_leave");
580 &function_end("bn_mul_mont");

582 &asciz("Montgomery Multiplication for x86, CRYPTOGAMS by <appro\@openssl.org>");

584 &asm_finish();
585 #endif /* ! codereview */
```

```
**********************************************************
    8577 Wed Aug 13 19:53:10 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86_64-gf2m.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
  1 #!/usr/bin/env perl
  2 #
  3 # ====================================================================
  4 # Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
  5 # project. The module is, however, dual licensed under OpenSSL and
  6 # CRYPTOGAMS licenses depending on where you obtain it. For further
  7 # details see http://www.openssl.org/~appro/cryptogams/.
  8 # ====================================================================
  9 #
 10 # May 2011
 11 #
 12 # The module implements bn_GF2m_mul_2x2 polynomial multiplication used
 13 # in bn_gf2m.c. It's kind of low-hanging mechanical port from C for
 14 # the time being... Except that it has two code paths: code suitable
 15 # for any x86_64 CPU and PCLMULQDQ one suitable for Westmere and
 16 # later. Improvement varies from one benchmark and µ-arch to another.
 17 # Vanilla code path is at most 20% faster than compiler-generated code
 18 # [not very impressive], while PCLMULQDQ - whole 85%-160% better on
 19 # 163- and 571-bit ECDH benchmarks on Intel CPUs. Keep in mind that
 20 # these coefficients are not ones for bn_GF2m_mul_2x2 itself, as not
 21 # all CPU time is burnt in it...

 23 $flavour = shift;
 24 $output  = shift;
 25 if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

 27 $win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

 29 $0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
 30 ( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
 31 ( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
 32 die "can't locate x86_64-xlate.pl";

 34 open OUT,"| \"$^X\" $xlate $flavour $output";
 35 *STDOUT=*OUT;

 37 ($lo,$hi)=("%rax","%rdx");      $a=$lo;
 38 ($i0,$i1)=("%rsi","%rdi");
 39 ($t0,$t1)=("%rbx","%rcx");
 40 ($b,$mask)=("%rbp","%r8");
 41 ($a1,$a2,$a4,$a8,$a12,$a48)=map("%r$_",(9..15));
 42 ($R,$Tx)=("%xmm0","%xmm1");

 44 $code.=<<___;
 45 .text

 47 .type   _mul_1x1,\@abi-omnipotent
 48 .align  16
 49 _mul_1x1:
 50         sub     \$128+8,%rsp
 51         mov     \$-1,$a1
 52         lea     ($a,$a),$i0
 53         shr     \$3,$a1
 54         lea     (,$a,4),$i1
 55         and     $a,$a1                  # a1=a&0x1fffffffffffffff
 56         lea     (,$a,8),$a8
 57         sar     \$63,$a                 # broadcast 63rd bit
 58         lea     ($a1,$a1),$a2
 59         sar     \$63,$i0                # broadcast 62nd bit
 60         lea     (,$a1,4),$a4
 61         and     $b,$a
```



```
 62         sar     \$63,$i1                # boardcast 61st bit
 63         mov     $a,$hi                  # $a is $lo
 64         shl     \$63,$lo
 65         and     $b,$i0
 66         shr     \$1,$hi
 67         mov     $i0,$t1
 68         shl     \$62,$i0
 69         and     $b,$i1
 70         shr     \$2,$t1
 71         xor     $i0,$lo
 72         mov     $i1,$t0
 73         shl     \$61,$i1
 74         xor     $t1,$hi
 75         shr     \$3,$t0
 76         xor     $i1,$lo
 77         xor     $t0,$hi

 79         mov     $a1,$a12
 80         movq    \$0,0(%rsp)             # tab[0]=0
 81         xor     $a2,$a12                # a1^a2
 82         mov     $a1,8(%rsp)             # tab[1]=a1
 83          mov    $a4,$a48
 84         mov     $a2,16(%rsp)            # tab[2]=a2
 85          xor    $a8,$a48                # a4^a8
 86         mov     $a12,24(%rsp)           # tab[3]=a1^a2

 88         xor     $a4,$a1
 89         mov     $a4,32(%rsp)            # tab[4]=a4
 90         xor     $a4,$a2
 91         mov     $a1,40(%rsp)            # tab[5]=a1^a4
 92         xor     $a4,$a12
 93         mov     $a2,48(%rsp)            # tab[6]=a2^a4
 94          xor    $a48,$a1                # a1^a4^a4^a8=a1^a8
 95         mov     $a12,56(%rsp)           # tab[7]=a1^a2^a4
 96          xor    $a48,$a2                # a2^a4^a4^a8=a1^a8

 98         mov     $a8,64(%rsp)            # tab[8]=a8
 99         xor     $a48,$a12               # a1^a2^a4^a4^a8=a1^a2^a8
100         mov     $a1,72(%rsp)            # tab[9]=a1^a8
101          xor    $a4,$a1                 # a1^a8^a4
102         mov     $a2,80(%rsp)            # tab[10]=a2^a8
103          xor    $a4,$a2                 # a2^a8^a4
104         mov     $a12,88(%rsp)           # tab[11]=a1^a2^a8

106         xor     $a4,$a12                # a1^a2^a8^a4
107         mov     $a48,96(%rsp)           # tab[12]=a4^a8
108          mov    $mask,$i0
109         mov     $a1,104(%rsp)           # tab[13]=a1^a4^a8
110          and    $b,$i0
111         mov     $a2,112(%rsp)           # tab[14]=a2^a4^a8
112          shr    \$4,$b
113         mov     $a12,120(%rsp)          # tab[15]=a1^a2^a4^a8
114          mov    $mask,$i1
115          and    $b,$i1
116          shr    \$4,$b

118         movq    (%rsp,$i0,8),$R         # half of calculations is done in SSE2
119         mov     $mask,$i0
120         and     $b,$i0
121         shr     \$4,$b
122 ___
123     for ($n=1;$n<8;$n++) {
124         $code.=<<___;
125         mov     (%rsp,$i1,8),$t1
126         mov     $mask,$i1
127         mov     $t1,$t0
```

```
128         shl     \$‘8*$n-4‘,$t1
129         and     $b,$i1
130          movq   (%rsp,$i0,8),$Tx
131         shr     \$‘64-(8*$n-4)‘,$t0
132         xor     $t1,$lo
133          pslldq \$$n,$Tx
134          mov    $mask,$i0
135         shr     \$4,$b
136         xor     $t0,$hi
137          and    $b,$i0
138          shr    \$4,$b
139          pxor   $Tx,$R
140 ___
141     }
142 $code.=<<___;
143         mov     (%rsp,$i1,8),$t1
144         mov     $t1,$t0
145         shl     \$‘8*$n-4‘,$t1
146         movq    $R,$i0
147         shr     \$‘64-(8*$n-4)‘,$t0
148         xor     $t1,$lo
149         psrldq  \$8,$R
150         xor     $t0,$hi
151         movq    $R,$i1
152         xor     $i0,$lo
153         xor     $i1,$hi

155         add     \$128+8,%rsp
156         ret
157 .Lend_mul_1x1:
158 .size   _mul_1x1,.-_mul_1x1
159 ___

161 ($rp,$a1,$a0,$b1,$b0) = $win64? ("%rcx","%rdx","%r8", "%r9","%r10") :   # Win64
162                                 ("%rdi","%rsi","%rdx","%rcx","%r8");    # Unix o

164 $code.=<<___;
165 .extern OPENSSL_ia32cap_P
166 .globl  bn_GF2m_mul_2x2
167 .type   bn_GF2m_mul_2x2,\@abi-omnipotent
168 .align  16
169 bn_GF2m_mul_2x2:
170         mov     OPENSSL_ia32cap_P(%rip),%rax
171         bt      \$33,%rax
172         jnc     .Lvanilla_mul_2x2

174         movq            $a1,%xmm0
175         movq            $b1,%xmm1
176         movq            $a0,%xmm2
177 ___
178 $code.=<<___ if ($win64);
179         movq            40(%rsp),%xmm3
180 ___
181 $code.=<<___ if (!$win64);
182         movq            $b0,%xmm3
183 ___
184 $code.=<<___;
185         movdqa          %xmm0,%xmm4
186         movdqa          %xmm1,%xmm5
187         pclmulqdq       \$0,%xmm1,%xmm0 # a1•b1
188         pxor            %xmm2,%xmm4
189         pxor            %xmm3,%xmm5
190         pclmulqdq       \$0,%xmm3,%xmm2 # a0•b0
191         pclmulqdq       \$0,%xmm5,%xmm4 # (a0+a1)•(b0+b1)
192         xorps           %xmm0,%xmm4
193         xorps           %xmm2,%xmm4     # (a0+a1)•(b0+b1)-a0•b0-a1•b1
```



```
194         movdqa          %xmm4,%xmm5
195         pslldq          \$8,%xmm4
196         psrldq          \$8,%xmm5
197         pxor            %xmm4,%xmm2
198         pxor            %xmm5,%xmm0
199         movdqu          %xmm2,0($rp)
200         movdqu          %xmm0,16($rp)
201         ret

203 .align  16
204 .Lvanilla_mul_2x2:
205         lea     -8*17(%rsp),%rsp
206 ___
207 $code.=<<___ if ($win64);
208         mov     ‘8*17+40‘(%rsp),$b0
209         mov     %rdi,8*15(%rsp)
210         mov     %rsi,8*16(%rsp)
211 ___
212 $code.=<<___;
213         mov     %r14,8*10(%rsp)
214         mov     %r13,8*11(%rsp)
215         mov     %r12,8*12(%rsp)
216         mov     %rbp,8*13(%rsp)
217         mov     %rbx,8*14(%rsp)
218 .Lbody_mul_2x2:
219         mov     $rp,32(%rsp)            # save the arguments
220         mov     $a1,40(%rsp)
221         mov     $a0,48(%rsp)
222         mov     $b1,56(%rsp)
223         mov     $b0,64(%rsp)

225         mov     \$0xf,$mask
226         mov     $a1,$a
227         mov     $b1,$b
228         call    _mul_1x1                # a1•b1
229         mov     $lo,16(%rsp)
230         mov     $hi,24(%rsp)

232         mov     48(%rsp),$a
233         mov     64(%rsp),$b
234         call    _mul_1x1                # a0•b0
235         mov     $lo,0(%rsp)
236         mov     $hi,8(%rsp)

238         mov     40(%rsp),$a
239         mov     56(%rsp),$b
240         xor     48(%rsp),$a
241         xor     64(%rsp),$b
242         call    _mul_1x1                # (a0+a1)•(b0+b1)
243 ___
244         @r=("%rbx","%rcx","%rdi","%rsi");
245 $code.=<<___;
246         mov     0(%rsp),@r[0]
247         mov     8(%rsp),@r[1]
248         mov     16(%rsp),@r[2]
249         mov     24(%rsp),@r[3]
250         mov     32(%rsp),%rbp

252         xor     $hi,$lo
253         xor     @r[1],$hi
254         xor     @r[0],$lo
255         mov     @r[0],0(%rbp)
256         xor     @r[2],$hi
257         mov     @r[3],24(%rbp)
258         xor     @r[3],$lo
259         xor     @r[3],$hi
```

```
260         xor     $hi,$lo
261         mov     $hi,16(%rbp)
262         mov     $lo,8(%rbp)

264         mov     8*10(%rsp),%r14
265         mov     8*11(%rsp),%r13
266         mov     8*12(%rsp),%r12
267         mov     8*13(%rsp),%rbp
268         mov     8*14(%rsp),%rbx
269 ___
270 $code.=<<___ if ($win64);
271         mov     8*15(%rsp),%rdi
272         mov     8*16(%rsp),%rsi
273 ___
274 $code.=<<___;
275         lea     8*17(%rsp),%rsp
276         ret
277 .Lend_mul_2x2:
278 .size   bn_GF2m_mul_2x2,.-bn_GF2m_mul_2x2
279 .asciz  "GF(2^m) Multiplication for x86_64, CRYPTOGAMS by <appro\@openssl.org>"
280 .align  16
281 ___

283 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
284 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
285 if ($win64) {
286 $rec="%rcx";
287 $frame="%rdx";
288 $context="%r8";
289 $disp="%r9";

291 $code.=<<___;
292 .extern __imp_RtlVirtualUnwind

294 .type   se_handler,\@abi-omnipotent
295 .align  16
296 se_handler:
297         push    %rsi
298         push    %rdi
299         push    %rbx
300         push    %rbp
301         push    %r12
302         push    %r13
303         push    %r14
304         push    %r15
305         pushfq
306         sub     \$64,%rsp

308         mov     152($context),%rax      # pull context->Rsp
309         mov     248($context),%rbx      # pull context->Rip

311         lea     .Lbody_mul_2x2(%rip),%r10
312         cmp     %r10,%rbx               # context->Rip<"prologue" label
313         jb      .Lin_prologue

315         mov     8*10(%rax),%r14         # mimic epilogue
316         mov     8*11(%rax),%r13
317         mov     8*12(%rax),%r12
318         mov     8*13(%rax),%rbp
319         mov     8*14(%rax),%rbx
320         mov     8*15(%rax),%rdi
321         mov     8*16(%rax),%rsi

323         mov     %rbx,144($context)      # restore context->Rbx
324         mov     %rbp,160($context)      # restore context->Rbp
325         mov     %rsi,168($context)      # restore context->Rsi
```



```
326         mov     %rdi,176($context)      # restore context->Rdi
327         mov     %r12,216($context)      # restore context->R12
328         mov     %r13,224($context)      # restore context->R13
329         mov     %r14,232($context)      # restore context->R14

331 .Lin_prologue:
332         lea     8*17(%rax),%rax
333         mov     %rax,152($context)      # restore context->Rsp

335         mov     40($disp),%rdi          # disp->ContextRecord
336         mov     $context,%rsi           # context
337         mov     \$154,%ecx              # sizeof(CONTEXT)
338         .long   0xa548f3fc              # cld; rep movsq

340         mov     $disp,%rsi
341         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
342         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
343         mov     0(%rsi),%r8             # arg3, disp->ControlPc
344         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
345         mov     40(%rsi),%r10           # disp->ContextRecord
346         lea     56(%rsi),%r11           # &disp->HandlerData
347         lea     24(%rsi),%r12           # &disp->EstablisherFrame
348         mov     %r10,32(%rsp)           # arg5
349         mov     %r11,40(%rsp)           # arg6
350         mov     %r12,48(%rsp)           # arg7
351         mov     %rcx,56(%rsp)           # arg8, (NULL)
352         call    *__imp_RtlVirtualUnwind(%rip)

354         mov     \$1,%eax                # ExceptionContinueSearch
355         add     \$64,%rsp
356         popfq
357         pop     %r15
358         pop     %r14
359         pop     %r13
360         pop     %r12
361         pop     %rbp
362         pop     %rbx
363         pop     %rdi
364         pop     %rsi
365         ret
366 .size   se_handler,.-se_handler

368 .section        .pdata
369 .align  4
370         .rva    _mul_1x1
371         .rva    .Lend_mul_1x1
372         .rva    .LSEH_info_1x1

374         .rva    .Lvanilla_mul_2x2
375         .rva    .Lend_mul_2x2
376         .rva    .LSEH_info_2x2
377 .section        .xdata
378 .align  8
379 .LSEH_info_1x1:
380         .byte   0x01,0x07,0x02,0x00
381         .byte   0x07,0x01,0x11,0x00     # sub rsp,128+8
382 .LSEH_info_2x2:
383         .byte   9,0,0,0
384         .rva    se_handler
385 ___
386 }

388 $code =~ s/\`([^\`]*)\`/eval($1)/gem;
389 print $code;
390 close STDOUT;
391 #endif /* ! codereview */
```

```
**********************************************************
   36986 Wed Aug 13 19:53:11 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86_64-mont.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# ====================================================================
# Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================

# October 2005.
#
# Montgomery multiplication routine for x86_64. While it gives modest
# 9% improvement of rsa4096 sign on Opteron, rsa512 sign runs more
# than twice, >2x, as fast. Most common rsa1024 sign is improved by
# respectful 50%. It remains to be seen if loop unrolling and
# dedicated squaring routine can provide further improvement...

# July 2011.
#
# Add dedicated squaring procedure. Performance improvement varies
# from platform to platform, but in average it's ~5%/15%/25%/33%
# for 512-/1024-/2048-/4096-bit RSA *sign* benchmarks respectively.

# August 2011.
#
# Unroll and modulo-schedule inner loops in such manner that they
# are "fallen through" for input lengths of 8, which is critical for
# 1024-bit RSA *sign*. Average performance improvement in comparison
# to *initial* version of this module from 2005 is ~0%/30%/40%/45%
# for 512-/1024-/2048-/4096-bit RSA *sign* benchmarks respectively.

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

# int bn_mul_mont(
$rp="%rdi";     # BN_ULONG *rp,
$ap="%rsi";     # const BN_ULONG *ap,
$bp="%rdx";     # const BN_ULONG *bp,
$np="%rcx";     # const BN_ULONG *np,
$n0="%r8";      # const BN_ULONG *n0,
$num="%r9";     # int num);
$lo0="%r10";
$hi0="%r11";
$hi1="%r13";
$i="%r14";
$j="%r15";
$m0="%rbx";
$m1="%rbp";

$code=<<___;
```



```
.text

.globl  bn_mul_mont
.type   bn_mul_mont,\@function,6
.align  16
bn_mul_mont:
        test    \$3,${num}d
        jnz     .Lmul_enter
        cmp     \$8,${num}d
        jb      .Lmul_enter
        cmp     $ap,$bp
        jne     .Lmul4x_enter
        jmp     .Lsqr4x_enter

.align  16
.Lmul_enter:
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15

        mov     ${num}d,${num}d
        lea     2($num),%r10
        mov     %rsp,%r11
        neg     %r10
        lea     (%rsp,%r10,8),%rsp      # tp=alloca(8*(num+2))
        and     \$-1024,%rsp            # minimize TLB usage

        mov     %r11,8(%rsp,$num,8)     # tp[num+1]=%rsp
.Lmul_body:
        mov     $bp,%r12                # reassign $bp
___
                $bp="%r12";
$code.=<<___;
        mov     ($n0),$n0               # pull n0[0] value
        mov     ($bp),$m0               # m0=bp[0]
        mov     ($ap),%rax

        xor     $i,$i                   # i=0
        xor     $j,$j                   # j=0

        mov     $n0,$m1
        mulq    $m0                     # ap[0]*bp[0]
        mov     %rax,$lo0
        mov     ($np),%rax

        imulq   $lo0,$m1                # "tp[0]"*n0
        mov     %rdx,$hi0

        mulq    $m1                     # np[0]*m1
        add     %rax,$lo0               # discarded
        mov     8($ap),%rax
        adc     \$0,%rdx
        mov     %rdx,$hi1

        lea     1($j),$j                # j++
        jmp     .L1st_enter

.align  16
.L1st:
        add     %rax,$hi1
        mov     ($ap,$j,8),%rax
        adc     \$0,%rdx
        add     $hi0,$hi1               # np[j]*m1+ap[j]*bp[0]
```

```
128         mov     $lo0,$hi0
129         adc     \$0,%rdx
130         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
131         mov     %rdx,$hi1

133 .L1st_enter:
134         mulq    $m0                     # ap[j]*bp[0]
135         add     %rax,$hi0
136         mov     ($np,$j,8),%rax
137         adc     \$0,%rdx
138         lea     1($j),$j                # j++
139         mov     %rdx,$lo0

141         mulq    $m1                     # np[j]*m1
142         cmp     $num,$j
143         jne     .L1st

145         add     %rax,$hi1
146         mov     ($ap),%rax              # ap[0]
147         adc     \$0,%rdx
148         add     $hi0,$hi1               # np[j]*m1+ap[j]*bp[0]
149         adc     \$0,%rdx
150         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
151         mov     %rdx,$hi1
152         mov     $lo0,$hi0

154         xor     %rdx,%rdx
155         add     $hi0,$hi1
156         adc     \$0,%rdx
157         mov     $hi1,-8(%rsp,$num,8)
158         mov     %rdx,(%rsp,$num,8)      # store upmost overflow bit

160         lea     1($i),$i                # i++
161         jmp     .Louter
162 .align  16
163 .Louter:
164         mov     ($bp,$i,8),$m0          # m0=bp[i]
165         xor     $j,$j                   # j=0
166         mov     $n0,$m1
167         mov     (%rsp),$lo0
168         mulq    $m0                     # ap[0]*bp[i]
169         add     %rax,$lo0               # ap[0]*bp[i]+tp[0]
170         mov     ($np),%rax
171         adc     \$0,%rdx

173         imulq   $lo0,$m1                # tp[0]*n0
174         mov     %rdx,$hi0

176         mulq    $m1                     # np[0]*m1
177         add     %rax,$lo0               # discarded
178         mov     8($ap),%rax
179         adc     \$0,%rdx
180         mov     8(%rsp),$lo0            # tp[1]
181         mov     %rdx,$hi1

183         lea     1($j),$j                # j++
184         jmp     .Linner_enter

186 .align  16
187 .Linner:
188         add     %rax,$hi1
189         mov     ($ap,$j,8),%rax
190         adc     \$0,%rdx
191         add     $lo0,$hi1               # np[j]*m1+ap[j]*bp[i]+tp[j]
192         mov     (%rsp,$j,8),$lo0
193         adc     \$0,%rdx
```



```
194         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
195         mov     %rdx,$hi1

197 .Linner_enter:
198         mulq    $m0                     # ap[j]*bp[i]
199         add     %rax,$hi0
200         mov     ($np,$j,8),%rax
201         adc     \$0,%rdx
202         add     $hi0,$lo0               # ap[j]*bp[i]+tp[j]
203         mov     %rdx,$hi0
204         adc     \$0,$hi0
205         lea     1($j),$j                # j++

207         mulq    $m1                     # np[j]*m1
208         cmp     $num,$j
209         jne     .Linner

211         add     %rax,$hi1
212         mov     ($ap),%rax              # ap[0]
213         adc     \$0,%rdx
214         add     $lo0,$hi1               # np[j]*m1+ap[j]*bp[i]+tp[j]
215         mov     (%rsp,$j,8),$lo0
216         adc     \$0,%rdx
217         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
218         mov     %rdx,$hi1

220         xor     %rdx,%rdx
221         add     $hi0,$hi1
222         adc     \$0,%rdx
223         add     $lo0,$hi1               # pull upmost overflow bit
224         adc     \$0,%rdx
225         mov     $hi1,-8(%rsp,$num,8)
226         mov     %rdx,(%rsp,$num,8)      # store upmost overflow bit

228         lea     1($i),$i                # i++
229         cmp     $num,$i
230         jl      .Louter

232         xor     $i,$i                   # i=0 and clear CF!
233         mov     (%rsp),%rax             # tp[0]
234         lea     (%rsp),$ap              # borrow ap for tp
235         mov     $num,$j                 # j=num
236         jmp     .Lsub
237 .align  16
238 .Lsub:  sbb     ($np,$i,8),%rax
239         mov     %rax,($rp,$i,8)         # rp[i]=tp[i]-np[i]
240         mov     8($ap,$i,8),%rax        # tp[i+1]
241         lea     1($i),$i                # i++
242         dec     $j                      # doesnn't affect CF!
243         jnz     .Lsub

245         sbb     \$0,%rax                # handle upmost overflow bit
246         xor     $i,$i
247         and     %rax,$ap
248         not     %rax
249         mov     $rp,$np
250         and     %rax,$np
251         mov     $num,$j                 # j=num
252         or      $np,$ap                 # ap=borrow?tp:rp
253 .align  16
254 .Lcopy:                                 # copy or in-place refresh
255         mov     ($ap,$i,8),%rax
256         mov     $i,(%rsp,$i,8)          # zap temporary vector
257         mov     %rax,($rp,$i,8)         # rp[i]=tp[i]
258         lea     1($i),$i
259         sub     \$1,$j
```

```
260         jnz     .Lcopy

262         mov     8(%rsp,$num,8),%rsi     # restore %rsp
263         mov     \$1,%rax
264         mov     (%rsi),%r15
265         mov     8(%rsi),%r14
266         mov     16(%rsi),%r13
267         mov     24(%rsi),%r12
268         mov     32(%rsi),%rbp
269         mov     40(%rsi),%rbx
270         lea     48(%rsi),%rsp
271 .Lmul_epilogue:
272         ret
273 .size   bn_mul_mont,.-bn_mul_mont
274 ___
275 {{{
276 my @A=("%r10","%r11");
277 my @N=("%r13","%rdi");
278 $code.=<<___;
279 .type   bn_mul4x_mont,\@function,6
280 .align  16
281 bn_mul4x_mont:
282 .Lmul4x_enter:
283         push    %rbx
284         push    %rbp
285         push    %r12
286         push    %r13
287         push    %r14
288         push    %r15

290         mov     ${num}d,${num}d
291         lea     4($num),%r10
292         mov     %rsp,%r11
293         neg     %r10
294         lea     (%rsp,%r10,8),%rsp      # tp=alloca(8*(num+4))
295         and     \$-1024,%rsp            # minimize TLB usage

297         mov     %r11,8(%rsp,$num,8)     # tp[num+1]=%rsp
298 .Lmul4x_body:
299         mov     $rp,16(%rsp,$num,8)     # tp[num+2]=$rp
300         mov     %rdx,%r12               # reassign $bp
301 ___
302                 $bp="%r12";
303 $code.=<<___;
304         mov     ($n0),$n0               # pull n0[0] value
305         mov     ($bp),$m0               # m0=bp[0]
306         mov     ($ap),%rax

308         xor     $i,$i                   # i=0
309         xor     $j,$j                   # j=0

311         mov     $n0,$m1
312         mulq    $m0                     # ap[0]*bp[0]
313         mov     %rax,$A[0]
314         mov     ($np),%rax

316         imulq   $A[0],$m1               # "tp[0]"*n0
317         mov     %rdx,$A[1]

319         mulq    $m1                     # np[0]*m1
320         add     %rax,$A[0]              # discarded
321         mov     8($ap),%rax
322         adc     \$0,%rdx
323         mov     %rdx,$N[1]

325         mulq    $m0
```



```
326         add     %rax,$A[1]
327         mov     8($np),%rax
328         adc     \$0,%rdx
329         mov     %rdx,$A[0]

331         mulq    $m1
332         add     %rax,$N[1]
333         mov     16($ap),%rax
334         adc     \$0,%rdx
335         add     $A[1],$N[1]
336         lea     4($j),$j                # j++
337         adc     \$0,%rdx
338         mov     $N[1],(%rsp)
339         mov     %rdx,$N[0]
340         jmp     .L1st4x
341 .align  16
342 .L1st4x:
343         mulq    $m0                     # ap[j]*bp[0]
344         add     %rax,$A[0]
345         mov     -16($np,$j,8),%rax
346         adc     \$0,%rdx
347         mov     %rdx,$A[1]

349         mulq    $m1                     # np[j]*m1
350         add     %rax,$N[0]
351         mov     -8($ap,$j,8),%rax
352         adc     \$0,%rdx
353         add     $A[0],$N[0]             # np[j]*m1+ap[j]*bp[0]
354         adc     \$0,%rdx
355         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
356         mov     %rdx,$N[1]

358         mulq    $m0                     # ap[j]*bp[0]
359         add     %rax,$A[1]
360         mov     -8($np,$j,8),%rax
361         adc     \$0,%rdx
362         mov     %rdx,$A[0]

364         mulq    $m1                     # np[j]*m1
365         add     %rax,$N[1]
366         mov     ($ap,$j,8),%rax
367         adc     \$0,%rdx
368         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[0]
369         adc     \$0,%rdx
370         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]
371         mov     %rdx,$N[0]

373         mulq    $m0                     # ap[j]*bp[0]
374         add     %rax,$A[0]
375         mov     ($np,$j,8),%rax
376         adc     \$0,%rdx
377         mov     %rdx,$A[1]

379         mulq    $m1                     # np[j]*m1
380         add     %rax,$N[0]
381         mov     8($ap,$j,8),%rax
382         adc     \$0,%rdx
383         add     $A[0],$N[0]             # np[j]*m1+ap[j]*bp[0]
384         adc     \$0,%rdx
385         mov     $N[0],-8(%rsp,$j,8)     # tp[j-1]
386         mov     %rdx,$N[1]

388         mulq    $m0                     # ap[j]*bp[0]
389         add     %rax,$A[1]
390         mov     8($np,$j,8),%rax
391         adc     \$0,%rdx
```

```
392         lea     4($j),$j                # j++
393         mov     %rdx,$A[0]

395         mulq    $m1                     # np[j]*m1
396         add     %rax,$N[1]
397         mov     -16($ap,$j,8),%rax
398         adc     \$0,%rdx
399         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[0]
400         adc     \$0,%rdx
401         mov     $N[1],-32(%rsp,$j,8)    # tp[j-1]
402         mov     %rdx,$N[0]
403         cmp     $num,$j
404         jl      .L1st4x

406         mulq    $m0                     # ap[j]*bp[0]
407         add     %rax,$A[0]
408         mov     -16($np,$j,8),%rax
409         adc     \$0,%rdx
410         mov     %rdx,$A[1]

412         mulq    $m1                     # np[j]*m1
413         add     %rax,$N[0]
414         mov     -8($ap,$j,8),%rax
415         adc     \$0,%rdx
416         add     $A[0],$N[0]             # np[j]*m1+ap[j]*bp[0]
417         adc     \$0,%rdx
418         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
419         mov     %rdx,$N[1]

421         mulq    $m0                     # ap[j]*bp[0]
422         add     %rax,$A[1]
423         mov     -8($np,$j,8),%rax
424         adc     \$0,%rdx
425         mov     %rdx,$A[0]

427         mulq    $m1                     # np[j]*m1
428         add     %rax,$N[1]
429         mov     ($ap),%rax              # ap[0]
430         adc     \$0,%rdx
431         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[0]
432         adc     \$0,%rdx
433         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]
434         mov     %rdx,$N[0]

436         xor     $N[1],$N[1]
437         add     $A[0],$N[0]
438         adc     \$0,$N[1]
439         mov     $N[0],-8(%rsp,$j,8)
440         mov     $N[1],(%rsp,$j,8)       # store upmost overflow bit

442         lea     1($i),$i                # i++
443 .align  4
444 .Louter4x:
445         mov     ($bp,$i,8),$m0          # m0=bp[i]
446         xor     $j,$j                   # j=0
447         mov     (%rsp),$A[0]
448         mov     $n0,$m1
449         mulq    $m0                     # ap[0]*bp[i]
450         add     %rax,$A[0]              # ap[0]*bp[i]+tp[0]
451         mov     ($np),%rax
452         adc     \$0,%rdx

454         imulq   $A[0],$m1               # tp[0]*n0
455         mov     %rdx,$A[1]

457         mulq    $m1                     # np[0]*m1
```



```
458         add     %rax,$A[0]              # "$N[0]", discarded
459         mov     8($ap),%rax
460         adc     \$0,%rdx
461         mov     %rdx,$N[1]

463         mulq    $m0                     # ap[j]*bp[i]
464         add     %rax,$A[1]
465         mov     8($np),%rax
466         adc     \$0,%rdx
467         add     8(%rsp),$A[1]           # +tp[1]
468         adc     \$0,%rdx
469         mov     %rdx,$A[0]

471         mulq    $m1                     # np[j]*m1
472         add     %rax,$N[1]
473         mov     16($ap),%rax
474         adc     \$0,%rdx
475         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[i]+tp[j]
476         lea     4($j),$j                # j+=2
477         adc     \$0,%rdx
478         mov     $N[1],(%rsp)            # tp[j-1]
479         mov     %rdx,$N[0]
480         jmp     .Linner4x
481 .align  16
482 .Linner4x:
483         mulq    $m0                     # ap[j]*bp[i]
484         add     %rax,$A[0]
485         mov     -16($np,$j,8),%rax
486         adc     \$0,%rdx
487         add     -16(%rsp,$j,8),$A[0]    # ap[j]*bp[i]+tp[j]
488         adc     \$0,%rdx
489         mov     %rdx,$A[1]

491         mulq    $m1                     # np[j]*m1
492         add     %rax,$N[0]
493         mov     -8($ap,$j,8),%rax
494         adc     \$0,%rdx
495         add     $A[0],$N[0]
496         adc     \$0,%rdx
497         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
498         mov     %rdx,$N[1]

500         mulq    $m0                     # ap[j]*bp[i]
501         add     %rax,$A[1]
502         mov     -8($np,$j,8),%rax
503         adc     \$0,%rdx
504         add     -8(%rsp,$j,8),$A[1]
505         adc     \$0,%rdx
506         mov     %rdx,$A[0]

508         mulq    $m1                     # np[j]*m1
509         add     %rax,$N[1]
510         mov     ($ap,$j,8),%rax
511         adc     \$0,%rdx
512         add     $A[1],$N[1]
513         adc     \$0,%rdx
514         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]
515         mov     %rdx,$N[0]

517         mulq    $m0                     # ap[j]*bp[i]
518         add     %rax,$A[0]
519         mov     ($np,$j,8),%rax
520         adc     \$0,%rdx
521         add     (%rsp,$j,8),$A[0]       # ap[j]*bp[i]+tp[j]
522         adc     \$0,%rdx
523         mov     %rdx,$A[1]
```

```
525         mulq    $m1                     # np[j]*m1
526         add     %rax,$N[0]
527         mov     8($ap,$j,8),%rax
528         adc     \$0,%rdx
529         add     $A[0],$N[0]
530         adc     \$0,%rdx
531         mov     $N[0],-8(%rsp,$j,8)     # tp[j-1]
532         mov     %rdx,$N[1]

534         mulq    $m0                     # ap[j]*bp[i]
535         add     %rax,$A[1]
536         mov     8($np,$j,8),%rax
537         adc     \$0,%rdx
538         add     8(%rsp,$j,8),$A[1]
539         adc     \$0,%rdx
540         lea     4($j),$j                # j++
541         mov     %rdx,$A[0]

543         mulq    $m1                     # np[j]*m1
544         add     %rax,$N[1]
545         mov     -16($ap,$j,8),%rax
546         adc     \$0,%rdx
547         add     $A[1],$N[1]
548         adc     \$0,%rdx
549         mov     $N[1],-32(%rsp,$j,8)    # tp[j-1]
550         mov     %rdx,$N[0]
551         cmp     $num,$j
552         jl      .Linner4x

554         mulq    $m0                     # ap[j]*bp[i]
555         add     %rax,$A[0]
556         mov     -16($np,$j,8),%rax
557         adc     \$0,%rdx
558         add     -16(%rsp,$j,8),$A[0]    # ap[j]*bp[i]+tp[j]
559         adc     \$0,%rdx
560         mov     %rdx,$A[1]

562         mulq    $m1                     # np[j]*m1
563         add     %rax,$N[0]
564         mov     -8($ap,$j,8),%rax
565         adc     \$0,%rdx
566         add     $A[0],$N[0]
567         adc     \$0,%rdx
568         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
569         mov     %rdx,$N[1]

571         mulq    $m0                     # ap[j]*bp[i]
572         add     %rax,$A[1]
573         mov     -8($np,$j,8),%rax
574         adc     \$0,%rdx
575         add     -8(%rsp,$j,8),$A[1]
576         adc     \$0,%rdx
577         lea     1($i),$i                # i++
578         mov     %rdx,$A[0]

580         mulq    $m1                     # np[j]*m1
581         add     %rax,$N[1]
582         mov     ($ap),%rax              # ap[0]
583         adc     \$0,%rdx
584         add     $A[1],$N[1]
585         adc     \$0,%rdx
586         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]
587         mov     %rdx,$N[0]

589         xor     $N[1],$N[1]
```



```
590         add     $A[0],$N[0]
591         adc     \$0,$N[1]
592         add     (%rsp,$num,8),$N[0]     # pull upmost overflow bit
593         adc     \$0,$N[1]
594         mov     $N[0],-8(%rsp,$j,8)
595         mov     $N[1],(%rsp,$j,8)       # store upmost overflow bit

597         cmp     $num,$i
598         jl      .Louter4x
599 ___
600 {
601 my @ri=("%rax","%rdx",$m0,$m1);
602 $code.=<<___;
603         mov     16(%rsp,$num,8),$rp     # restore $rp
604         mov     0(%rsp),@ri[0]          # tp[0]
605         pxor    %xmm0,%xmm0
606         mov     8(%rsp),@ri[1]          # tp[1]
607         shr     \$2,$num                # num/=4
608         lea     (%rsp),$ap              # borrow ap for tp
609         xor     $i,$i                   # i=0 and clear CF!

611         sub     0($np),@ri[0]
612         mov     16($ap),@ri[2]          # tp[2]
613         mov     24($ap),@ri[3]          # tp[3]
614         sbb     8($np),@ri[1]
615         lea     -1($num),$j             # j=num/4-1
616         jmp     .Lsub4x
617 .align  16
618 .Lsub4x:
619         mov     @ri[0],0($rp,$i,8)      # rp[i]=tp[i]-np[i]
620         mov     @ri[1],8($rp,$i,8)      # rp[i]=tp[i]-np[i]
621         sbb     16($np,$i,8),@ri[2]
622         mov     32($ap,$i,8),@ri[0]     # tp[i+1]
623         mov     40($ap,$i,8),@ri[1]
624         sbb     24($np,$i,8),@ri[3]
625         mov     @ri[2],16($rp,$i,8)     # rp[i]=tp[i]-np[i]
626         mov     @ri[3],24($rp,$i,8)     # rp[i]=tp[i]-np[i]
627         sbb     32($np,$i,8),@ri[0]
628         mov     48($ap,$i,8),@ri[2]
629         mov     56($ap,$i,8),@ri[3]
630         sbb     40($np,$i,8),@ri[1]
631         lea     4($i),$i                # i++
632         dec     $j                      # doesnn't affect CF!
633         jnz     .Lsub4x

635         mov     @ri[0],0($rp,$i,8)      # rp[i]=tp[i]-np[i]
636         mov     32($ap,$i,8),@ri[0]     # load overflow bit
637         sbb     16($np,$i,8),@ri[2]
638         mov     @ri[1],8($rp,$i,8)      # rp[i]=tp[i]-np[i]
639         sbb     24($np,$i,8),@ri[3]
640         mov     @ri[2],16($rp,$i,8)     # rp[i]=tp[i]-np[i]

642         sbb     \$0,@ri[0]              # handle upmost overflow bit
643         mov     @ri[3],24($rp,$i,8)     # rp[i]=tp[i]-np[i]
644         xor     $i,$i                   # i=0
645         and     @ri[0],$ap
646         not     @ri[0]
647         mov     $rp,$np
648         and     @ri[0],$np
649         lea     -1($num),$j
650         or      $np,$ap                 # ap=borrow?tp:rp

652         movdqu  ($ap),%xmm1
653         movdqa  %xmm0,(%rsp)
654         movdqu  %xmm1,($rp)
655         jmp     .Lcopy4x
```

```
.align  16
.Lcopy4x:                                       # copy or in-place refresh
        movdqu  16($ap,$i),%xmm2
        movdqu  32($ap,$i),%xmm1
        movdqa  %xmm0,16(%rsp,$i)
        movdqu  %xmm2,16($rp,$i)
        movdqa  %xmm0,32(%rsp,$i)
        movdqu  %xmm1,32($rp,$i)
        lea     32($i),$i
        dec     $j
        jnz     .Lcopy4x

        shl     \$2,$num
        movdqu  16($ap,$i),%xmm2
        movdqa  %xmm0,16(%rsp,$i)
        movdqu  %xmm2,16($rp,$i)
___
}
$code.=<<___;
        mov     8(%rsp,$num,8),%rsi     # restore %rsp
        mov     \$1,%rax
        mov     (%rsi),%r15
        mov     8(%rsi),%r14
        mov     16(%rsi),%r13
        mov     24(%rsi),%r12
        mov     32(%rsi),%rbp
        mov     40(%rsi),%rbx
        lea     48(%rsi),%rsp
.Lmul4x_epilogue:
        ret
.size   bn_mul4x_mont,.-bn_mul4x_mont
___
}}}
```



```
{{{
######################################################################
# void bn_sqr4x_mont(
my $rptr="%rdi";        # const BN_ULONG *rptr,
my $aptr="%rsi";        # const BN_ULONG *aptr,
my $bptr="%rdx";        # not used
my $nptr="%rcx";        # const BN_ULONG *nptr,
my $n0  ="%r8";         # const BN_ULONG *n0);
my $num ="%r9";         # int num, has to be divisible by 4 and
                        # not less than 8

my ($i,$j,$tptr)=("%rbp","%rcx",$rptr);
my @A0=("%r10","%r11");
my @A1=("%r12","%r13");
my ($a0,$a1,$ai)=("%r14","%r15","%rbx");

$code.=<<___;
.type   bn_sqr4x_mont,\@function,6
.align  16
bn_sqr4x_mont:
.Lsqr4x_enter:
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15

        shl     \$3,${num}d             # convert $num to bytes
        xor     %r10,%r10
        mov     %rsp,%r11               # put aside %rsp
        sub     $num,%r10               # -$num
        mov     ($n0),$n0               # *n0
        lea     -72(%rsp,%r10,2),%rsp   # alloca(frame+2*$num)
        and     \$-1024,%rsp            # minimize TLB usage
        ##############################################################
        # Stack layout
        #
        # +0    saved $num, used in reduction section
        # +8    &t[2*$num], used in reduction section
        # +32   saved $rptr
        # +40   saved $nptr
        # +48   saved *n0
        # +56   saved %rsp
        # +64   t[2*$num]
        #
        mov     $rptr,32(%rsp)          # save $rptr
        mov     $nptr,40(%rsp)
        mov     $n0,  48(%rsp)
        mov     %r11, 56(%rsp)          # save original %rsp
.Lsqr4x_body:
        ##############################################################
        # Squaring part:
        #
        # a) multiply-n-add everything but a[i]*a[i];
        # b) shift result of a) by 1 to the left and accumulate
        #    a[i]*a[i] products;
        #
        lea     32(%r10),$i             # $i=-($num-32)
        lea     ($aptr,$num),$aptr      # end of a[] buffer, ($aptr,$i)=&ap[2]

        mov     $num,$j                 # $j=$num

                                        # comments apply to $num==8 case
        mov     -32($aptr,$i),$a0       # a[0]
        lea     64(%rsp,$num,2),$tptr   # end of tp[] buffer, &tp[2*$num]
```

```
755         mov     -24($aptr,$i),%rax      # a[1]
756         lea     -32($tptr,$i),$tptr     # end of tp[] window, &tp[2*$num-"$i"]
757         mov     -16($aptr,$i),$ai       # a[2]
758         mov     %rax,$a1

760         mul     $a0                     # a[1]*a[0]
761         mov     %rax,$A0[0]             # a[1]*a[0]
762          mov    $ai,%rax                # a[2]
763         mov     %rdx,$A0[1]
764         mov     $A0[0],-24($tptr,$i)    # t[1]

766         xor     $A0[0],$A0[0]
767         mul     $a0                     # a[2]*a[0]
768         add     %rax,$A0[1]
769          mov    $ai,%rax
770         adc     %rdx,$A0[0]
771         mov     $A0[1],-16($tptr,$i)    # t[2]

773         lea     -16($i),$j              # j=-16


776          mov    8($aptr,$j),$ai         # a[3]
777         mul     $a1                     # a[2]*a[1]
778         mov     %rax,$A1[0]             # a[2]*a[1]+t[3]
779          mov    $ai,%rax
780         mov     %rdx,$A1[1]

782         xor     $A0[1],$A0[1]
783         add     $A1[0],$A0[0]
784          lea    16($j),$j
785         adc     \$0,$A0[1]
786         mul     $a0                     # a[3]*a[0]
787         add     %rax,$A0[0]             # a[3]*a[0]+a[2]*a[1]+t[3]
788          mov    $ai,%rax
789         adc     %rdx,$A0[1]
790         mov     $A0[0],-8($tptr,$j)     # t[3]
791         jmp     .Lsqr4x_1st

793 .align  16
794 .Lsqr4x_1st:
795          mov    ($aptr,$j),$ai          # a[4]
796         xor     $A1[0],$A1[0]
797         mul     $a1                     # a[3]*a[1]
798         add     %rax,$A1[1]             # a[3]*a[1]+t[4]
799          mov    $ai,%rax
800         adc     %rdx,$A1[0]

802         xor     $A0[0],$A0[0]
803         add     $A1[1],$A0[1]
804         adc     \$0,$A0[0]
805         mul     $a0                     # a[4]*a[0]
806         add     %rax,$A0[1]             # a[4]*a[0]+a[3]*a[1]+t[4]
807          mov    $ai,%rax                # a[3]
808         adc     %rdx,$A0[0]
809         mov     $A0[1],($tptr,$j)       # t[4]


812          mov    8($aptr,$j),$ai         # a[5]
813         xor     $A1[1],$A1[1]
814         mul     $a1                     # a[4]*a[3]
815         add     %rax,$A1[0]             # a[4]*a[3]+t[5]
816          mov    $ai,%rax
817         adc     %rdx,$A1[1]

819         xor     $A0[1],$A0[1]
820         add     $A1[0],$A0[0]
```



```
821         adc     \$0,$A0[1]
822         mul     $a0                     # a[5]*a[2]
823         add     %rax,$A0[0]             # a[5]*a[2]+a[4]*a[3]+t[5]
824          mov    $ai,%rax
825         adc     %rdx,$A0[1]
826         mov     $A0[0],8($tptr,$j)      # t[5]

828          mov    16($aptr,$j),$ai        # a[6]
829         xor     $A1[0],$A1[0]
830         mul     $a1                     # a[5]*a[3]
831         add     %rax,$A1[1]             # a[5]*a[3]+t[6]
832          mov    $ai,%rax
833         adc     %rdx,$A1[0]

835         xor     $A0[0],$A0[0]
836         add     $A1[1],$A0[1]
837         adc     \$0,$A0[0]
838         mul     $a0                     # a[6]*a[2]
839         add     %rax,$A0[1]             # a[6]*a[2]+a[5]*a[3]+t[6]
840          mov    $ai,%rax                # a[3]
841         adc     %rdx,$A0[0]
842         mov     $A0[1],16($tptr,$j)     # t[6]


845          mov    24($aptr,$j),$ai        # a[7]
846         xor     $A1[1],$A1[1]
847         mul     $a1                     # a[6]*a[5]
848         add     %rax,$A1[0]             # a[6]*a[5]+t[7]
849          mov    $ai,%rax
850         adc     %rdx,$A1[1]

852         xor     $A0[1],$A0[1]
853         add     $A1[0],$A0[0]
854          lea    32($j),$j
855         adc     \$0,$A0[1]
856         mul     $a0                     # a[7]*a[4]
857         add     %rax,$A0[0]             # a[7]*a[4]+a[6]*a[5]+t[6]
858          mov    $ai,%rax
859         adc     %rdx,$A0[1]
860         mov     $A0[0],-8($tptr,$j)     # t[7]

862         cmp     \$0,$j
863         jne     .Lsqr4x_1st

865         xor     $A1[0],$A1[0]
866         add     $A0[1],$A1[1]
867         adc     \$0,$A1[0]
868         mul     $a1                     # a[7]*a[5]
869         add     %rax,$A1[1]
870         adc     %rdx,$A1[0]

872         mov     $A1[1],($tptr)          # t[8]
873         lea     16($i),$i
874         mov     $A1[0],8($tptr)         # t[9]
875         jmp     .Lsqr4x_outer

877 .align  16
878 .Lsqr4x_outer:                          # comments apply to $num==6 case
879         mov     -32($aptr,$i),$a0       # a[0]
880         lea     64(%rsp,$num,2),$tptr   # end of tp[] buffer, &tp[2*$num]
881         mov     -24($aptr,$i),%rax      # a[1]
882         lea     -32($tptr,$i),$tptr     # end of tp[] window, &tp[2*$num-"$i"]
883         mov     -16($aptr,$i),$ai       # a[2]
884         mov     %rax,$a1

886         mov     -24($tptr,$i),$A0[0]    # t[1]
```

```
887         xor     $A0[1],$A0[1]
888         mul     $a0                     # a[1]*a[0]
889         add     %rax,$A0[0]             # a[1]*a[0]+t[1]
890          mov    $ai,%rax                # a[2]
891         adc     %rdx,$A0[1]
892         mov     $A0[0],-24($tptr,$i)    # t[1]

894         xor     $A0[0],$A0[0]
895         add     -16($tptr,$i),$A0[1]    # a[2]*a[0]+t[2]
896         adc     \$0,$A0[0]
897         mul     $a0                     # a[2]*a[0]
898         add     %rax,$A0[1]
899          mov    $ai,%rax
900         adc     %rdx,$A0[0]
901         mov     $A0[1],-16($tptr,$i)    # t[2]

903         lea     -16($i),$j              # j=-16
904         xor     $A1[0],$A1[0]


907          mov    8($aptr,$j),$ai         # a[3]
908         xor     $A1[1],$A1[1]
909         add     8($tptr,$j),$A1[0]
910         adc     \$0,$A1[1]
911         mul     $a1                     # a[2]*a[1]
912         add     %rax,$A1[0]             # a[2]*a[1]+t[3]
913          mov    $ai,%rax
914         adc     %rdx,$A1[1]

916         xor     $A1[1],$A1[1]
917         add     $A1[0],$A0[0]
918         adc     \$0,$A1[1]
919         mul     $a0                     # a[3]*a[0]
920         add     %rax,$A0[0]             # a[3]*a[0]+a[2]*a[1]+t[3]
921          mov    $ai,%rax
922         adc     %rdx,$A0[1]
923         mov     $A0[0],8($tptr,$j)      # t[3]

925         lea     16($j),$j
926         jmp     .Lsqr4x_inner

928 .align  16
929 .Lsqr4x_inner:
930          mov    ($aptr,$j),$ai          # a[4]
931         xor     $A1[0],$A1[0]
932         add     ($tptr,$j),$A1[1]
933         adc     \$0,$A1[0]
934         mul     $a1                     # a[3]*a[1]
935         add     %rax,$A1[1]             # a[3]*a[1]+t[4]
936          mov    $ai,%rax
937         adc     %rdx,$A1[0]

939         xor     $A0[0],$A0[0]
940         add     $A1[1],$A0[1]
941         adc     \$0,$A0[0]
942         mul     $a0                     # a[4]*a[0]
943         add     %rax,$A0[1]             # a[4]*a[0]+a[3]*a[1]+t[4]
944          mov    $ai,%rax                # a[3]
945         adc     %rdx,$A0[0]
946         mov     $A0[1],($tptr,$j)       # t[4]

948          mov    8($aptr,$j),$ai         # a[5]
949         xor     $A1[1],$A1[1]
950         add     8($tptr,$j),$A1[0]
951         adc     \$0,$A1[1]
952         mul     $a1                     # a[4]*a[3]
```



```
953         add     %rax,$A1[0]             # a[4]*a[3]+t[5]
954          mov    $ai,%rax
955         adc     %rdx,$A1[1]

957         xor     $A0[1],$A0[1]
958         add     $A1[0],$A0[0]
959         lea     16($j),$j               # j++
960         adc     \$0,$A0[1]
961         mul     $a0                     # a[5]*a[2]
962         add     %rax,$A0[0]             # a[5]*a[2]+a[4]*a[3]+t[5]
963          mov    $ai,%rax
964         adc     %rdx,$A0[1]
965         mov     $A0[0],-8($tptr,$j)     # t[5], "preloaded t[1]" below

967         cmp     \$0,$j
968         jne     .Lsqr4x_inner

970         xor     $A1[0],$A1[0]
971         add     $A0[1],$A1[1]
972         adc     \$0,$A1[0]
973         mul     $a1                     # a[5]*a[3]
974         add     %rax,$A1[1]
975         adc     %rdx,$A1[0]

977         mov     $A1[1],($tptr)          # t[6], "preloaded t[2]" below
978         mov     $A1[0],8($tptr)         # t[7], "preloaded t[3]" below

980         add     \$16,$i
981         jnz     .Lsqr4x_outer

983                                         # comments apply to $num==4 case
984         mov     -32($aptr),$a0          # a[0]
985         lea     64(%rsp,$num,2),$tptr   # end of tp[] buffer, &tp[2*$num]
986         mov     -24($aptr),%rax         # a[1]
987         lea     -32($tptr,$i),$tptr     # end of tp[] window, &tp[2*$num-"$i"]
988         mov     -16($aptr),$ai          # a[2]
989         mov     %rax,$a1

991         xor     $A0[1],$A0[1]
992         mul     $a0                     # a[1]*a[0]
993         add     %rax,$A0[0]             # a[1]*a[0]+t[1], preloaded t[1]
994          mov    $ai,%rax                # a[2]
995         adc     %rdx,$A0[1]
996         mov     $A0[0],-24($tptr)       # t[1]

998         xor     $A0[0],$A0[0]
999         add     $A1[1],$A0[1]           # a[2]*a[0]+t[2], preloaded t[2]
1000        adc     \$0,$A0[0]
1001        mul     $a0                     # a[2]*a[0]
1002        add     %rax,$A0[1]
1003         mov    $ai,%rax
1004        adc     %rdx,$A0[0]
1005        mov     $A0[1],-16($tptr)       # t[2]

1007         mov    -8($aptr),$ai           # a[3]
1008        mul     $a1                     # a[2]*a[1]
1009        add     %rax,$A1[0]             # a[2]*a[1]+t[3], preloaded t[3]
1010         mov    $ai,%rax
1011        adc     \$0,%rdx

1013        xor     $A0[1],$A0[1]
1014        add     $A1[0],$A0[0]
1015         mov    %rdx,$A1[1]
1016        adc     \$0,$A0[1]
1017        mul     $a0                     # a[3]*a[0]
1018        add     %rax,$A0[0]             # a[3]*a[0]+a[2]*a[1]+t[3]
```

```
1019          mov     $ai,%rax
1020         adc     %rdx,$A0[1]
1021         mov     $A0[0],-8($tptr)        # t[3]

1023         xor     $A1[0],$A1[0]
1024         add     $A0[1],$A1[1]
1025         adc     \$0,$A1[0]
1026         mul     $a1                     # a[3]*a[1]
1027         add     %rax,$A1[1]
1028          mov    -16($aptr),%rax         # a[2]
1029         adc     %rdx,$A1[0]

1031         mov     $A1[1],($tptr)          # t[4]
1032         mov     $A1[0],8($tptr)         # t[5]

1034         mul     $ai                     # a[2]*a[3]
1035 ___
1036 {
1037 my ($shift,$carry)=($a0,$a1);
1038 my @S=(@A1,$ai,$n0);
1039 $code.=<<___;
1040          add    \$16,$i
1041          xor    $shift,$shift
1042          sub    $num,$i                 # $i=16-$num
1043          xor    $carry,$carry

1045         add     $A1[0],%rax             # t[5]
1046         adc     \$0,%rdx
1047         mov     %rax,8($tptr)           # t[5]
1048         mov     %rdx,16($tptr)          # t[6]
1049         mov     $carry,24($tptr)        # t[7]

1051          mov    -16($aptr,$i),%rax      # a[0]
1052         lea     64(%rsp,$num,2),$tptr
1053          xor    $A0[0],$A0[0]           # t[0]
1054          mov    -24($tptr,$i,2),$A0[1]  # t[1]

1056         lea     ($shift,$A0[0],2),$S[0] # t[2*i]<<1 | shift
1057         shr     \$63,$A0[0]
1058         lea     ($j,$A0[1],2),$S[1]     # t[2*i+1]<<1 |
1059         shr     \$63,$A0[1]
1060         or      $A0[0],$S[1]            # | t[2*i]>>63
1061          mov    -16($tptr,$i,2),$A0[0]  # t[2*i+2]      # prefetch
1062         mov     $A0[1],$shift           # shift=t[2*i+1]>>63
1063         mul     %rax                    # a[i]*a[i]
1064         neg     $carry                  # mov $carry,cf
1065          mov    -8($tptr,$i,2),$A0[1]   # t[2*i+2+1]    # prefetch
1066         adc     %rax,$S[0]
1067          mov    -8($aptr,$i),%rax       # a[i+1]        # prefetch
1068         mov     $S[0],-32($tptr,$i,2)
1069         adc     %rdx,$S[1]

1071         lea     ($shift,$A0[0],2),$S[2] # t[2*i]<<1 | shift
1072          mov    $S[1],-24($tptr,$i,2)
1073          sbb    $carry,$carry           # mov cf,$carry
1074         shr     \$63,$A0[0]
1075         lea     ($j,$A0[1],2),$S[3]     # t[2*i+1]<<1 |
1076         shr     \$63,$A0[1]
1077         or      $A0[0],$S[3]            # | t[2*i]>>63
1078          mov    0($tptr,$i,2),$A0[0]    # t[2*i+2]      # prefetch
1079         mov     $A0[1],$shift           # shift=t[2*i+1]>>63
1080         mul     %rax                    # a[i]*a[i]
1081         neg     $carry                  # mov $carry,cf
1082          mov    8($tptr,$i,2),$A0[1]    # t[2*i+2+1]    # prefetch
1083         adc     %rax,$S[2]
1084          mov    0($aptr,$i),%rax        # a[i+1]        # prefetch
```



```
1085         mov     $S[2],-16($tptr,$i,2)
1086         adc     %rdx,$S[3]
1087         lea     16($i),$i
1088         mov     $S[3],-40($tptr,$i,2)
1089         sbb     $carry,$carry           # mov cf,$carry
1090         jmp     .Lsqr4x_shift_n_add

1092 .align  16
1093 .Lsqr4x_shift_n_add:
1094         lea     ($shift,$A0[0],2),$S[0] # t[2*i]<<1 | shift
1095         shr     \$63,$A0[0]
1096         lea     ($j,$A0[1],2),$S[1]     # t[2*i+1]<<1 |
1097         shr     \$63,$A0[1]
1098         or      $A0[0],$S[1]            # | t[2*i]>>63
1099          mov    -16($tptr,$i,2),$A0[0]  # t[2*i+2]      # prefetch
1100         mov     $A0[1],$shift           # shift=t[2*i+1]>>63
1101         mul     %rax                    # a[i]*a[i]
1102         neg     $carry                  # mov $carry,cf
1103          mov    -8($tptr,$i,2),$A0[1]   # t[2*i+2+1]    # prefetch
1104         adc     %rax,$S[0]
1105          mov    -8($aptr,$i),%rax       # a[i+1]        # prefetch
1106         mov     $S[0],-32($tptr,$i,2)
1107         adc     %rdx,$S[1]

1109         lea     ($shift,$A0[0],2),$S[2] # t[2*i]<<1 | shift
1110          mov    $S[1],-24($tptr,$i,2)
1111          sbb    $carry,$carry           # mov cf,$carry
1112         shr     \$63,$A0[0]
1113         lea     ($j,$A0[1],2),$S[3]     # t[2*i+1]<<1 |
1114         shr     \$63,$A0[1]
1115         or      $A0[0],$S[3]            # | t[2*i]>>63
1116          mov    0($tptr,$i,2),$A0[0]    # t[2*i+2]      # prefetch
1117         mov     $A0[1],$shift           # shift=t[2*i+1]>>63
1118         mul     %rax                    # a[i]*a[i]
1119         neg     $carry                  # mov $carry,cf
1120          mov    8($tptr,$i,2),$A0[1]    # t[2*i+2+1]    # prefetch
1121         adc     %rax,$S[2]
1122          mov    0($aptr,$i),%rax        # a[i+1]        # prefetch
1123         mov     $S[2],-16($tptr,$i,2)
1124         adc     %rdx,$S[3]

1126         lea     ($shift,$A0[0],2),$S[0] # t[2*i]<<1 | shift
1127          mov    $S[3],-8($tptr,$i,2)
1128          sbb    $carry,$carry           # mov cf,$carry
1129         shr     \$63,$A0[0]
1130         lea     ($j,$A0[1],2),$S[1]     # t[2*i+1]<<1 |
1131         shr     \$63,$A0[1]
1132         or      $A0[0],$S[1]            # | t[2*i]>>63
1133          mov    16($tptr,$i,2),$A0[0]   # t[2*i+2]      # prefetch
1134         mov     $A0[1],$shift           # shift=t[2*i+1]>>63
1135         mul     %rax                    # a[i]*a[i]
1136         neg     $carry                  # mov $carry,cf
1137          mov    24($tptr,$i,2),$A0[1]   # t[2*i+2+1]    # prefetch
1138         adc     %rax,$S[0]
1139          mov    8($aptr,$i),%rax        # a[i+1]        # prefetch
1140         mov     $S[0],0($tptr,$i,2)
1141         adc     %rdx,$S[1]

1143         lea     ($shift,$A0[0],2),$S[2] # t[2*i]<<1 | shift
1144          mov    $S[1],8($tptr,$i,2)
1145          sbb    $carry,$carry           # mov cf,$carry
1146         shr     \$63,$A0[0]
1147         lea     ($j,$A0[1],2),$S[3]     # t[2*i+1]<<1 |
1148         shr     \$63,$A0[1]
1149         or      $A0[0],$S[3]            # | t[2*i]>>63
1150          mov    32($tptr,$i,2),$A0[0]   # t[2*i+2]      # prefetch
```

```
1151         mov     $A0[1],$shift           # shift=t[2*i+1]>>63
1152         mul     %rax                    # a[i]*a[i]
1153         neg     $carry                  # mov $carry,cf
1154          mov    40($tptr,$i,2),$A0[1]   # t[2*i+2+1]    # prefetch
1155         adc     %rax,$S[2]
1156          mov    16($aptr,$i),%rax       # a[i+1]        # prefetch
1157         mov     $S[2],16($tptr,$i,2)
1158         adc     %rdx,$S[3]
1159         mov     $S[3],24($tptr,$i,2)
1160         sbb     $carry,$carry           # mov cf,$carry
1161         add     \$32,$i
1162         jnz     .Lsqr4x_shift_n_add

1164         lea     ($shift,$A0[0],2),$S[0] # t[2*i]<<1 | shift
1165         shr     \$63,$A0[0]
1166         lea     ($j,$A0[1],2),$S[1]     # t[2*i+1]<<1 |
1167         shr     \$63,$A0[1]
1168         or      $A0[0],$S[1]            # | t[2*i]>>63
1169          mov    -16($tptr),$A0[0]       # t[2*i+2]      # prefetch
1170         mov     $A0[1],$shift           # shift=t[2*i+1]>>63
1171         mul     %rax                    # a[i]*a[i]
1172         neg     $carry                  # mov $carry,cf
1173          mov    -8($tptr),$A0[1]        # t[2*i+2+1]    # prefetch
1174         adc     %rax,$S[0]
1175          mov    -8($aptr),%rax          # a[i+1]        # prefetch
1176         mov     $S[0],-32($tptr)
1177         adc     %rdx,$S[1]

1179         lea     ($shift,$A0[0],2),$S[2] # t[2*i]<<1|shift
1180          mov    $S[1],-24($tptr)
1181          sbb    $carry,$carry           # mov cf,$carry
1182         shr     \$63,$A0[0]
1183         lea     ($j,$A0[1],2),$S[3]     # t[2*i+1]<<1 |
1184         shr     \$63,$A0[1]
1185         or      $A0[0],$S[3]            # | t[2*i]>>63
1186         mul     %rax                    # a[i]*a[i]
1187         neg     $carry                  # mov $carry,cf
1188         adc     %rax,$S[2]
1189         adc     %rdx,$S[3]
1190         mov     $S[2],-16($tptr)
1191         mov     $S[3],-8($tptr)
1192 ___
1193 }
```



```
1194 ########################################################################
1195 # Montgomery reduction part, "word-by-word" algorithm.
1196 #
1197 {
1198 my ($topbit,$nptr)=("%rbp",$aptr);
1199 my ($m0,$m1)=($a0,$a1);
1200 my @Ni=("%rbx","%r9");
1201 $code.=<<___;
1202         mov     40(%rsp),$nptr          # restore $nptr
1203         mov     48(%rsp),$n0            # restore *n0
1204         xor     $j,$j
1205         mov     $num,0(%rsp)            # save $num
1206         sub     $num,$j                 # $j=-$num
1207          mov    64(%rsp),$A0[0]         # t[0]          # modsched #
1208          mov    $n0,$m0                 #               # modsched #
1209         lea     64(%rsp,$num,2),%rax    # end of t[] buffer
1210         lea     64(%rsp,$num),$tptr     # end of t[] window
1211         mov     %rax,8(%rsp)            # save end of t[] buffer
1212         lea     ($nptr,$num),$nptr      # end of n[] buffer
1213         xor     $topbit,$topbit         # $topbit=0

1215         mov     0($nptr,$j),%rax        # n[0]          # modsched #
1216         mov     8($nptr,$j),$Ni[1]      # n[1]          # modsched #
1217          imulq  $A0[0],$m0              # m0=t[0]*n0    # modsched #
1218          mov    %rax,$Ni[0]             #               # modsched #
1219         jmp     .Lsqr4x_mont_outer

1221 .align  16
1222 .Lsqr4x_mont_outer:
1223         xor     $A0[1],$A0[1]
1224         mul     $m0                     # n[0]*m0
1225         add     %rax,$A0[0]             # n[0]*m0+t[0]
1226          mov    $Ni[1],%rax
1227         adc     %rdx,$A0[1]
1228         mov     $n0,$m1

1230         xor     $A0[0],$A0[0]
1231         add     8($tptr,$j),$A0[1]
1232         adc     \$0,$A0[0]
1233         mul     $m0                     # n[1]*m0
1234         add     %rax,$A0[1]             # n[1]*m0+t[1]
1235          mov    $Ni[0],%rax
1236         adc     %rdx,$A0[0]

1238         imulq   $A0[1],$m1

1240         mov     16($nptr,$j),$Ni[0]     # n[2]
1241         xor     $A1[1],$A1[1]
1242         add     $A0[1],$A1[0]
1243         adc     \$0,$A1[1]
1244         mul     $m1                     # n[0]*m1
1245         add     %rax,$A1[0]             # n[0]*m1+"t[1]"
1246          mov    $Ni[0],%rax
1247         adc     %rdx,$A1[1]
1248         mov     $A1[0],8($tptr,$j)      # "t[1]"

1250         xor     $A0[1],$A0[1]
1251         add     16($tptr,$j),$A0[0]
1252         adc     \$0,$A0[1]
1253         mul     $m0                     # n[2]*m0
1254         add     %rax,$A0[0]             # n[2]*m0+t[2]
1255          mov    $Ni[1],%rax
1256         adc     %rdx,$A0[1]

1258         mov     24($nptr,$j),$Ni[1]     # n[3]
1259         xor     $A1[0],$A1[0]
```

```
1260         add     $A0[0],$A1[1]
1261         adc     \$0,$A1[0]
1262         mul     $m1                     # n[1]*m1
1263         add     %rax,$A1[1]             # n[1]*m1+"t[2]"
1264          mov    $Ni[1],%rax
1265         adc     %rdx,$A1[0]
1266         mov     $A1[1],16($tptr,$j)     # "t[2]"

1268         xor     $A0[0],$A0[0]
1269         add     24($tptr,$j),$A0[1]
1270         lea     32($j),$j
1271         adc     \$0,$A0[0]
1272         mul     $m0                     # n[3]*m0
1273         add     %rax,$A0[1]             # n[3]*m0+t[3]
1274          mov    $Ni[0],%rax
1275         adc     %rdx,$A0[0]
1276         jmp     .Lsqr4x_mont_inner

1278 .align  16
1279 .Lsqr4x_mont_inner:
1280         mov     ($nptr,$j),$Ni[0]       # n[4]
1281         xor     $A1[1],$A1[1]
1282         add     $A0[1],$A1[0]
1283         adc     \$0,$A1[1]
1284         mul     $m1                     # n[2]*m1
1285         add     %rax,$A1[0]             # n[2]*m1+"t[3]"
1286          mov    $Ni[0],%rax
1287         adc     %rdx,$A1[1]
1288         mov     $A1[0],-8($tptr,$j)     # "t[3]"

1290         xor     $A0[1],$A0[1]
1291         add     ($tptr,$j),$A0[0]
1292         adc     \$0,$A0[1]
1293         mul     $m0                     # n[4]*m0
1294         add     %rax,$A0[0]             # n[4]*m0+t[4]
1295          mov    $Ni[1],%rax
1296         adc     %rdx,$A0[1]

1298         mov     8($nptr,$j),$Ni[1]      # n[5]
1299         xor     $A1[0],$A1[0]
1300         add     $A0[0],$A1[1]
1301         adc     \$0,$A1[0]
1302         mul     $m1                     # n[3]*m1
1303         add     %rax,$A1[1]             # n[3]*m1+"t[4]"
1304          mov    $Ni[1],%rax
1305         adc     %rdx,$A1[0]
1306         mov     $A1[1],($tptr,$j)       # "t[4]"

1308         xor     $A0[0],$A0[0]
1309         add     8($tptr,$j),$A0[1]
1310         adc     \$0,$A0[0]
1311         mul     $m0                     # n[5]*m0
1312         add     %rax,$A0[1]             # n[5]*m0+t[5]
1313          mov    $Ni[0],%rax
1314         adc     %rdx,$A0[0]


1317         mov     16($nptr,$j),$Ni[0]     # n[6]
1318         xor     $A1[1],$A1[1]
1319         add     $A0[1],$A1[0]
1320         adc     \$0,$A1[1]
1321         mul     $m1                     # n[4]*m1
1322         add     %rax,$A1[0]             # n[4]*m1+"t[5]"
1323          mov    $Ni[0],%rax
1324         adc     %rdx,$A1[1]
1325         mov     $A1[0],8($tptr,$j)      # "t[5]"
```



```
1327         xor     $A0[1],$A0[1]
1328         add     16($tptr,$j),$A0[0]
1329         adc     \$0,$A0[1]
1330         mul     $m0                     # n[6]*m0
1331         add     %rax,$A0[0]             # n[6]*m0+t[6]
1332          mov    $Ni[1],%rax
1333         adc     %rdx,$A0[1]

1335         mov     24($nptr,$j),$Ni[1]     # n[7]
1336         xor     $A1[0],$A1[0]
1337         add     $A0[0],$A1[1]
1338         adc     \$0,$A1[0]
1339         mul     $m1                     # n[5]*m1
1340         add     %rax,$A1[1]             # n[5]*m1+"t[6]"
1341          mov    $Ni[1],%rax
1342         adc     %rdx,$A1[0]
1343         mov     $A1[1],16($tptr,$j)     # "t[6]"

1345         xor     $A0[0],$A0[0]
1346         add     24($tptr,$j),$A0[1]
1347         lea     32($j),$j
1348         adc     \$0,$A0[0]
1349         mul     $m0                     # n[7]*m0
1350         add     %rax,$A0[1]             # n[7]*m0+t[7]
1351          mov    $Ni[0],%rax
1352         adc     %rdx,$A0[0]
1353         cmp     \$0,$j
1354         jne     .Lsqr4x_mont_inner

1356          sub    0(%rsp),$j              # $j=-$num      # modsched #
1357          mov    $n0,$m0                 #               # modsched #

1359         xor     $A1[1],$A1[1]
1360         add     $A0[1],$A1[0]
1361         adc     \$0,$A1[1]
1362         mul     $m1                     # n[6]*m1
1363         add     %rax,$A1[0]             # n[6]*m1+"t[7]"
1364         mov     $Ni[1],%rax
1365         adc     %rdx,$A1[1]
1366         mov     $A1[0],-8($tptr)        # "t[7]"

1368         xor     $A0[1],$A0[1]
1369         add     ($tptr),$A0[0]          # +t[8]
1370         adc     \$0,$A0[1]
1371          mov    0($nptr,$j),$Ni[0]      # n[0]          # modsched #
1372         add     $topbit,$A0[0]
1373         adc     \$0,$A0[1]

1375          imulq  16($tptr,$j),$m0        # m0=t[0]*n0    # modsched #
1376         xor     $A1[0],$A1[0]
1377          mov    8($nptr,$j),$Ni[1]      # n[1]          # modsched #
1378         add     $A0[0],$A1[1]
1379          mov    16($tptr,$j),$A0[0]     # t[0]          # modsched #
1380         adc     \$0,$A1[0]
1381         mul     $m1                     # n[7]*m1
1382         add     %rax,$A1[1]             # n[7]*m1+"t[8]"
1383          mov    $Ni[0],%rax             #               # modsched #
1384         adc     %rdx,$A1[0]
1385         mov     $A1[1],($tptr)          # "t[8]"

1387         xor     $topbit,$topbit
1388         add     8($tptr),$A1[0]         # +t[9]
1389         adc     $topbit,$topbit
1390         add     $A0[1],$A1[0]
1391         lea     16($tptr),$tptr         # "t[$num]>>128"
```

```
1392         adc     \$0,$topbit
1393         mov     $A1[0],-8($tptr)        # "t[9]"
1394         cmp     8(%rsp),$tptr           # are we done?
1395         jb      .Lsqr4x_mont_outer

1397         mov     0(%rsp),$num            # restore $num
1398         mov     $topbit,($tptr)         # save $topbit
1399 ___
1400 }
```



```
1401 ##############################################################
1402 # Post-condition, 4x unrolled copy from bn_mul_mont
1403 #
1404 {
1405 my ($tptr,$nptr)=("%rbx",$aptr);
1406 my @ri=("%rax","%rdx","%r10","%r11");
1407 $code.=<<___;
1408         mov     64(%rsp,$num),@ri[0]    # tp[0]
1409         lea     64(%rsp,$num),$tptr     # upper half of t[2*$num] holds result
1410         mov     40(%rsp),$nptr          # restore $nptr
1411         shr     \$5,$num                # num/4
1412         mov     8($tptr),@ri[1]         # t[1]
1413         xor     $i,$i                   # i=0 and clear CF!

1415         mov     32(%rsp),$rptr          # restore $rptr
1416         sub     0($nptr),@ri[0]
1417         mov     16($tptr),@ri[2]        # t[2]
1418         mov     24($tptr),@ri[3]        # t[3]
1419         sbb     8($nptr),@ri[1]
1420         lea     -1($num),$j             # j=num/4-1
1421         jmp     .Lsqr4x_sub
1422 .align  16
1423 .Lsqr4x_sub:
1424         mov     @ri[0],0($rptr,$i,8)    # rp[i]=tp[i]-np[i]
1425         mov     @ri[1],8($rptr,$i,8)    # rp[i]=tp[i]-np[i]
1426         sbb     16($nptr,$i,8),@ri[2]
1427         mov     32($tptr,$i,8),@ri[0]   # tp[i+1]
1428         mov     40($tptr,$i,8),@ri[1]
1429         sbb     24($nptr,$i,8),@ri[3]
1430         mov     @ri[2],16($rptr,$i,8)   # rp[i]=tp[i]-np[i]
1431         mov     @ri[3],24($rptr,$i,8)   # rp[i]=tp[i]-np[i]
1432         sbb     32($nptr,$i,8),@ri[0]
1433         mov     48($tptr,$i,8),@ri[2]
1434         mov     56($tptr,$i,8),@ri[3]
1435         sbb     40($nptr,$i,8),@ri[1]
1436         lea     4($i),$i                # i++
1437         dec     $j                      # doesn't affect CF!
1438         jnz     .Lsqr4x_sub

1440         mov     @ri[0],0($rptr,$i,8)    # rp[i]=tp[i]-np[i]
1441         mov     32($tptr,$i,8),@ri[0]   # load overflow bit
1442         sbb     16($nptr,$i,8),@ri[2]
1443         mov     @ri[1],8($rptr,$i,8)    # rp[i]=tp[i]-np[i]
1444         sbb     24($nptr,$i,8),@ri[3]
1445         mov     @ri[2],16($rptr,$i,8)   # rp[i]=tp[i]-np[i]

1447         sbb     \$0,@ri[0]              # handle upmost overflow bit
1448         mov     @ri[3],24($rptr,$i,8)   # rp[i]=tp[i]-np[i]
1449         xor     $i,$i                   # i=0
1450         and     @ri[0],$tptr
1451         not     @ri[0]
1452         mov     $rptr,$nptr
1453         and     @ri[0],$nptr
1454         lea     -1($num),$j
1455         or      $nptr,$tptr             # tp=borrow?tp:rp

1457         pxor    %xmm0,%xmm0
1458         lea     64(%rsp,$num,8),$nptr
1459         movdqu  ($tptr),%xmm1
1460         lea     ($nptr,$num,8),$nptr
1461         movdqa  %xmm0,64(%rsp)          # zap lower half of temporary vector
1462         movdqa  %xmm0,($nptr)           # zap upper half of temporary vector
1463         movdqu  %xmm1,($rptr)
1464         jmp     .Lsqr4x_copy
1465 .align  16
1466 .Lsqr4x_copy:                           # copy or in-place refresh
```

```
1467         movdqu  16($tptr,$i),%xmm2
1468         movdqu  32($tptr,$i),%xmm1
1469         movdqa  %xmm0,80(%rsp,$i)       # zap lower half of temporary vector
1470         movdqa  %xmm0,96(%rsp,$i)       # zap lower half of temporary vector
1471         movdqa  %xmm0,16($nptr,$i)      # zap upper half of temporary vector
1472         movdqa  %xmm0,32($nptr,$i)      # zap upper half of temporary vector
1473         movdqu  %xmm2,16($rptr,$i)
1474         movdqu  %xmm1,32($rptr,$i)
1475         lea     32($i),$i
1476         dec     $j
1477         jnz     .Lsqr4x_copy

1479         movdqu  16($tptr,$i),%xmm2
1480         movdqa  %xmm0,80(%rsp,$i)       # zap lower half of temporary vector
1481         movdqa  %xmm0,16($nptr,$i)      # zap upper half of temporary vector
1482         movdqu  %xmm2,16($rptr,$i)
1483 ___
1484 }
1485 $code.=<<___;
1486         mov     56(%rsp),%rsi           # restore %rsp
1487         mov     \$1,%rax
1488         mov     0(%rsi),%r15
1489         mov     8(%rsi),%r14
1490         mov     16(%rsi),%r13
1491         mov     24(%rsi),%r12
1492         mov     32(%rsi),%rbp
1493         mov     40(%rsi),%rbx
1494         lea     48(%rsi),%rsp
1495 .Lsqr4x_epilogue:
1496         ret
1497 .size   bn_sqr4x_mont,.-bn_sqr4x_mont
1498 ___
1499 }}}
1500 $code.=<<___;
1501 .asciz  "Montgomery Multiplication for x86_64, CRYPTOGAMS by <appro\@openssl.org
1502 .align  16
1503 ___

1505 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
1506 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
1507 if ($win64) {
1508 $rec="%rcx";
1509 $frame="%rdx";
1510 $context="%r8";
1511 $disp="%r9";

1513 $code.=<<___;
1514 .extern __imp_RtlVirtualUnwind
1515 .type   mul_handler,\@abi-omnipotent
1516 .align  16
1517 mul_handler:
1518         push    %rsi
1519         push    %rdi
1520         push    %rbx
1521         push    %rbp
1522         push    %r12
1523         push    %r13
1524         push    %r14
1525         push    %r15
1526         pushfq
1527         sub     \$64,%rsp

1529         mov     120($context),%rax      # pull context->Rax
1530         mov     248($context),%rbx      # pull context->Rip

1532         mov     8($disp),%rsi           # disp->ImageBase
```



```
1533         mov     56($disp),%r11          # disp->HandlerData

1535         mov     0(%r11),%r10d           # HandlerData[0]
1536         lea     (%rsi,%r10),%r10        # end of prologue label
1537         cmp     %r10,%rbx               # context->Rip<end of prologue label
1538         jb      .Lcommon_seh_tail

1540         mov     152($context),%rax      # pull context->Rsp

1542         mov     4(%r11),%r10d           # HandlerData[1]
1543         lea     (%rsi,%r10),%r10        # epilogue label
1544         cmp     %r10,%rbx               # context->Rip>=epilogue label
1545         jae     .Lcommon_seh_tail

1547         mov     192($context),%r10      # pull $num
1548         mov     8(%rax,%r10,8),%rax     # pull saved stack pointer
1549         lea     48(%rax),%rax

1551         mov     -8(%rax),%rbx
1552         mov     -16(%rax),%rbp
1553         mov     -24(%rax),%r12
1554         mov     -32(%rax),%r13
1555         mov     -40(%rax),%r14
1556         mov     -48(%rax),%r15
1557         mov     %rbx,144($context)      # restore context->Rbx
1558         mov     %rbp,160($context)      # restore context->Rbp
1559         mov     %r12,216($context)      # restore context->R12
1560         mov     %r13,224($context)      # restore context->R13
1561         mov     %r14,232($context)      # restore context->R14
1562         mov     %r15,240($context)      # restore context->R15

1564         jmp     .Lcommon_seh_tail
1565 .size   mul_handler,.-mul_handler

1567 .type   sqr_handler,\@abi-omnipotent
1568 .align  16
1569 sqr_handler:
1570         push    %rsi
1571         push    %rdi
1572         push    %rbx
1573         push    %rbp
1574         push    %r12
1575         push    %r13
1576         push    %r14
1577         push    %r15
1578         pushfq
1579         sub     \$64,%rsp

1581         mov     120($context),%rax      # pull context->Rax
1582         mov     248($context),%rbx      # pull context->Rip

1584         lea     .Lsqr4x_body(%rip),%r10
1585         cmp     %r10,%rbx               # context->Rip<.Lsqr_body
1586         jb      .Lcommon_seh_tail

1588         mov     152($context),%rax      # pull context->Rsp

1590         lea     .Lsqr4x_epilogue(%rip),%r10
1591         cmp     %r10,%rbx               # context->Rip>=.Lsqr_epilogue
1592         jae     .Lcommon_seh_tail

1594         mov     56(%rax),%rax           # pull saved stack pointer
1595         lea     48(%rax),%rax

1597         mov     -8(%rax),%rbx
1598         mov     -16(%rax),%rbp
```

```
1599         mov     -24(%rax),%r12
1600         mov     -32(%rax),%r13
1601         mov     -40(%rax),%r14
1602         mov     -48(%rax),%r15
1603         mov     %rbx,144($context)      # restore context->Rbx
1604         mov     %rbp,160($context)      # restore context->Rbp
1605         mov     %r12,216($context)      # restore context->R12
1606         mov     %r13,224($context)      # restore context->R13
1607         mov     %r14,232($context)      # restore context->R14
1608         mov     %r15,240($context)      # restore context->R15

1610 .Lcommon_seh_tail:
1611         mov     8(%rax),%rdi
1612         mov     16(%rax),%rsi
1613         mov     %rax,152($context)      # restore context->Rsp
1614         mov     %rsi,168($context)      # restore context->Rsi
1615         mov     %rdi,176($context)      # restore context->Rdi

1617         mov     40($disp),%rdi          # disp->ContextRecord
1618         mov     $context,%rsi           # context
1619         mov     \$154,%ecx              # sizeof(CONTEXT)
1620         .long   0xa548f3fc              # cld; rep movsq

1622         mov     $disp,%rsi
1623         xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
1624         mov     8(%rsi),%rdx            # arg2, disp->ImageBase
1625         mov     0(%rsi),%r8             # arg3, disp->ControlPc
1626         mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
1627         mov     40(%rsi),%r10           # disp->ContextRecord
1628         lea     56(%rsi),%r11           # &disp->HandlerData
1629         lea     24(%rsi),%r12           # &disp->EstablisherFrame
1630         mov     %r10,32(%rsp)           # arg5
1631         mov     %r11,40(%rsp)           # arg6
1632         mov     %r12,48(%rsp)           # arg7
1633         mov     %rcx,56(%rsp)           # arg8, (NULL)
1634         call    *__imp_RtlVirtualUnwind(%rip)

1636         mov     \$1,%eax                # ExceptionContinueSearch
1637         add     \$64,%rsp
1638         popfq
1639         pop     %r15
1640         pop     %r14
1641         pop     %r13
1642         pop     %r12
1643         pop     %rbp
1644         pop     %rbx
1645         pop     %rdi
1646         pop     %rsi
1647         ret
1648 .size   sqr_handler,.-sqr_handler

1650 .section        .pdata
1651 .align  4
1652         .rva    .LSEH_begin_bn_mul_mont
1653         .rva    .LSEH_end_bn_mul_mont
1654         .rva    .LSEH_info_bn_mul_mont

1656         .rva    .LSEH_begin_bn_mul4x_mont
1657         .rva    .LSEH_end_bn_mul4x_mont
1658         .rva    .LSEH_info_bn_mul4x_mont

1660         .rva    .LSEH_begin_bn_sqr4x_mont
1661         .rva    .LSEH_end_bn_sqr4x_mont
1662         .rva    .LSEH_info_bn_sqr4x_mont

1664 .section        .xdata
```



```
1665 .align  8
1666 .LSEH_info_bn_mul_mont:
1667         .byte   9,0,0,0
1668         .rva    mul_handler
1669         .rva    .Lmul_body,.Lmul_epilogue       # HandlerData[]
1670 .LSEH_info_bn_mul4x_mont:
1671         .byte   9,0,0,0
1672         .rva    mul_handler
1673         .rva    .Lmul4x_body,.Lmul4x_epilogue   # HandlerData[]
1674 .LSEH_info_bn_sqr4x_mont:
1675         .byte   9,0,0,0
1676         .rva    sqr_handler
1677 ___
1678 }

1680 print $code;
1681 close STDOUT;
1682 #endif /* ! codereview */
```

```
**********************************************************
   22488 Wed Aug 13 19:53:11 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86_64-mont5.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# ====================================================================
# Written by Andy Polyakov <appro@openssl.org> for the OpenSSL
# project. The module is, however, dual licensed under OpenSSL and
# CRYPTOGAMS licenses depending on where you obtain it. For further
# details see http://www.openssl.org/~appro/cryptogams/.
# ====================================================================

# August 2011.
#
# Companion to x86_64-mont.pl that optimizes cache-timing attack
# countermeasures. The subroutines are produced by replacing bp[i]
# references in their x86_64-mont.pl counterparts with cache-neutral
# references to powers table computed in BN_mod_exp_mont_consttime.
# In addition subroutine that scatters elements of the powers table
# is implemented, so that scatter-/gathering can be tuned without
# bn_exp.c modifications.

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}../../perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

# int bn_mul_mont_gather5(
$rp="%rdi";     # BN_ULONG *rp,
$ap="%rsi";     # const BN_ULONG *ap,
$bp="%rdx";     # const BN_ULONG *bp,
$np="%rcx";     # const BN_ULONG *np,
$n0="%r8";      # const BN_ULONG *n0,
$num="%r9";     # int num,
                # int idx);     # 0 to 2^5-1, "index" in $bp holding
                                # pre-computed powers of a', interlaced
                                # in such manner that b[0] is $bp[idx],
                                # b[1] is [2^5+idx], etc.
$lo0="%r10";
$hi0="%r11";
$hi1="%r13";
$i="%r14";
$j="%r15";
$m0="%rbx";
$m1="%rbp";

$code=<<___;
.text

.globl  bn_mul_mont_gather5
.type   bn_mul_mont_gather5,\@function,6
.align  64
bn_mul_mont_gather5:
        test    \$3,${num}d
        jnz     .Lmul_enter
```



```
        cmp     \$8,${num}d
        jb      .Lmul_enter
        jmp     .Lmul4x_enter

.align  16
.Lmul_enter:
        mov     ${num}d,${num}d
        mov     '($win64?56:8)'(%rsp),%r10d     # load 7th argument
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
___
$code.=<<___ if ($win64);
        lea     -0x28(%rsp),%rsp
        movaps  %xmm6,(%rsp)
        movaps  %xmm7,0x10(%rsp)
.Lmul_alloca:
___
$code.=<<___;
        mov     %rsp,%rax
        lea     2($num),%r11
        neg     %r11
        lea     (%rsp,%r11,8),%rsp      # tp=alloca(8*(num+2))
        and     \$-1024,%rsp            # minimize TLB usage

        mov     %rax,8(%rsp,$num,8)     # tp[num+1]=%rsp
.Lmul_body:
        mov     $bp,%r12                # reassign $bp
___
                $bp="%r12";
                $STRIDE=2**5*8;         # 5 is "window size"
                $N=$STRIDE/4;           # should match cache line size
$code.=<<___;
        mov     %r10,%r11
        shr     \$'log($N/8)/log(2)',%r10
        and     \$'$N/8-1',%r11
        not     %r10
        lea     .Lmagic_masks(%rip),%rax
        and     \$'2**5/($N/8)-1',%r10  # 5 is "window size"
        lea     96($bp,%r11,8),$bp      # pointer within 1st cache line
        movq    0(%rax,%r10,8),%xmm4    # set of masks denoting which
        movq    8(%rax,%r10,8),%xmm5    # cache line contains element
        movq    16(%rax,%r10,8),%xmm6   # denoted by 7th argument
        movq    24(%rax,%r10,8),%xmm7

        movq    '0*$STRIDE/4-96'($bp),%xmm0
        movq    '1*$STRIDE/4-96'($bp),%xmm1
        pand    %xmm4,%xmm0
        movq    '2*$STRIDE/4-96'($bp),%xmm2
        pand    %xmm5,%xmm1
        movq    '3*$STRIDE/4-96'($bp),%xmm3
        pand    %xmm6,%xmm2
        por     %xmm1,%xmm0
        pand    %xmm7,%xmm3
        por     %xmm2,%xmm0
        lea     $STRIDE($bp),$bp
        por     %xmm3,%xmm0

        movq    %xmm0,$m0               # m0=bp[0]

        mov     ($n0),$n0               # pull n0[0] value
        mov     ($ap),%rax
```

```
128         xor     $i,$i                   # i=0
129         xor     $j,$j                   # j=0

131         movq    '0*$STRIDE/4-96'($bp),%xmm0
132         movq    '1*$STRIDE/4-96'($bp),%xmm1
133         pand    %xmm4,%xmm0
134         movq    '2*$STRIDE/4-96'($bp),%xmm2
135         pand    %xmm5,%xmm1

137         mov     $n0,$m1
138         mulq    $m0                     # ap[0]*bp[0]
139         mov     %rax,$lo0
140         mov     ($np),%rax

142         movq    '3*$STRIDE/4-96'($bp),%xmm3
143         pand    %xmm6,%xmm2
144         por     %xmm1,%xmm0
145         pand    %xmm7,%xmm3

147         imulq   $lo0,$m1                # "tp[0]"*n0
148         mov     %rdx,$hi0

150         por     %xmm2,%xmm0
151         lea     $STRIDE($bp),$bp
152         por     %xmm3,%xmm0

154         mulq    $m1                     # np[0]*m1
155         add     %rax,$lo0               # discarded
156         mov     8($ap),%rax
157         adc     \$0,%rdx
158         mov     %rdx,$hi1

160         lea     1($j),$j                # j++
161         jmp     .L1st_enter

163 .align  16
164 .L1st:
165         add     %rax,$hi1
166         mov     ($ap,$j,8),%rax
167         adc     \$0,%rdx
168         add     $hi0,$hi1               # np[j]*m1+ap[j]*bp[0]
169         mov     $lo0,$hi0
170         adc     \$0,%rdx
171         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
172         mov     %rdx,$hi1

174 .L1st_enter:
175         mulq    $m0                     # ap[j]*bp[0]
176         add     %rax,$hi0
177         mov     ($np,$j,8),%rax
178         adc     \$0,%rdx
179         lea     1($j),$j                # j++
180         mov     %rdx,$lo0

182         mulq    $m1                     # np[j]*m1
183         cmp     $num,$j
184         jne     .L1st

186         movq    %xmm0,$m0               # bp[1]

188         add     %rax,$hi1
189         mov     ($ap),%rax              # ap[0]
190         adc     \$0,%rdx
191         add     $hi0,$hi1               # np[j]*m1+ap[j]*bp[0]
192         adc     \$0,%rdx
193         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
```



```
194         mov     %rdx,$hi1
195         mov     $lo0,$hi0

197         xor     %rdx,%rdx
198         add     $hi0,$hi1
199         adc     \$0,%rdx
200         mov     $hi1,-8(%rsp,$num,8)
201         mov     %rdx,(%rsp,$num,8)      # store upmost overflow bit

203         lea     1($i),$i                # i++
204         jmp     .Louter
205 .align  16
206 .Louter:
207         xor     $j,$j                   # j=0
208         mov     $n0,$m1
209         mov     (%rsp),$lo0

211         movq    '0*$STRIDE/4-96'($bp),%xmm0
212         movq    '1*$STRIDE/4-96'($bp),%xmm1
213         pand    %xmm4,%xmm0
214         movq    '2*$STRIDE/4-96'($bp),%xmm2
215         pand    %xmm5,%xmm1

217         mulq    $m0                     # ap[0]*bp[i]
218         add     %rax,$lo0               # ap[0]*bp[i]+tp[0]
219         mov     ($np),%rax
220         adc     \$0,%rdx

222         movq    '3*$STRIDE/4-96'($bp),%xmm3
223         pand    %xmm6,%xmm2
224         por     %xmm1,%xmm0
225         pand    %xmm7,%xmm3

227         imulq   $lo0,$m1                # tp[0]*n0
228         mov     %rdx,$hi0

230         por     %xmm2,%xmm0
231         lea     $STRIDE($bp),$bp
232         por     %xmm3,%xmm0

234         mulq    $m1                     # np[0]*m1
235         add     %rax,$lo0               # discarded
236         mov     8($ap),%rax
237         adc     \$0,%rdx
238         mov     8(%rsp),$lo0            # tp[1]
239         mov     %rdx,$hi1

241         lea     1($j),$j                # j++
242         jmp     .Linner_enter

244 .align  16
245 .Linner:
246         add     %rax,$hi1
247         mov     ($ap,$j,8),%rax
248         adc     \$0,%rdx
249         add     $lo0,$hi1               # np[j]*m1+ap[j]*bp[i]+tp[j]
250         mov     (%rsp,$j,8),$lo0
251         adc     \$0,%rdx
252         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
253         mov     %rdx,$hi1

255 .Linner_enter:
256         mulq    $m0                     # ap[j]*bp[i]
257         add     %rax,$hi0
258         mov     ($np,$j,8),%rax
259         adc     \$0,%rdx
```

```
260         add     $hi0,$lo0               # ap[j]*bp[i]+tp[j]
261         mov     %rdx,$hi0
262         adc     \$0,$hi0
263         lea     1($j),$j                # j++

265         mulq    $m1                     # np[j]*m1
266         cmp     $num,$j
267         jne     .Linner

269         movq    %xmm0,$m0               # bp[i+1]

271         add     %rax,$hi1
272         mov     ($ap),%rax              # ap[0]
273         adc     \$0,%rdx
274         add     $lo0,$hi1               # np[j]*m1+ap[j]*bp[i]+tp[j]
275         mov     (%rsp,$j,8),$lo0
276         adc     \$0,%rdx
277         mov     $hi1,-16(%rsp,$j,8)     # tp[j-1]
278         mov     %rdx,$hi1

280         xor     %rdx,%rdx
281         add     $hi0,$hi1
282         adc     \$0,%rdx
283         add     $lo0,$hi1               # pull upmost overflow bit
284         adc     \$0,%rdx
285         mov     $hi1,-8(%rsp,$num,8)
286         mov     %rdx,(%rsp,$num,8)      # store upmost overflow bit

288         lea     1($i),$i                # i++
289         cmp     $num,$i
290         jl      .Louter

292         xor     $i,$i                   # i=0 and clear CF!
293         mov     (%rsp),%rax             # tp[0]
294         lea     (%rsp),$ap              # borrow ap for tp
295         mov     $num,$j                 # j=num
296         jmp     .Lsub
297 .align  16
298 .Lsub:  sbb     ($np,$i,8),%rax
299         mov     %rax,($rp,$i,8)         # rp[i]=tp[i]-np[i]
300         mov     8($ap,$i,8),%rax        # tp[i+1]
301         lea     1($i),$i                # i++
302         dec     $j                      # doesnn't affect CF!
303         jnz     .Lsub

305         sbb     \$0,%rax                # handle upmost overflow bit
306         xor     $i,$i
307         and     %rax,$ap
308         not     %rax
309         mov     $rp,$np
310         and     %rax,$np
311         mov     $num,$j                 # j=num
312         or      $np,$ap                 # ap=borrow?tp:rp
313 .align  16
314 .Lcopy:                                 # copy or in-place refresh
315         mov     ($ap,$i,8),%rax
316         mov     $i,(%rsp,$i,8)          # zap temporary vector
317         mov     %rax,($rp,$i,8)         # rp[i]=tp[i]
318         lea     1($i),$i
319         sub     \$1,$j
320         jnz     .Lcopy

322         mov     8(%rsp,$num,8),%rsi     # restore %rsp
323         mov     \$1,%rax
324 ___
325 $code.=<<___ if ($win64);
```



```
326         movaps  (%rsi),%xmm6
327         movaps  0x10(%rsi),%xmm7
328         lea     0x28(%rsi),%rsi
329 ___
330 $code.=<<___;
331         mov     (%rsi),%r15
332         mov     8(%rsi),%r14
333         mov     16(%rsi),%r13
334         mov     24(%rsi),%r12
335         mov     32(%rsi),%rbp
336         mov     40(%rsi),%rbx
337         lea     48(%rsi),%rsp
338 .Lmul_epilogue:
339         ret
340 .size   bn_mul_mont_gather5,.-bn_mul_mont_gather5
341 ___
342 {{{
343 my @A=("%r10","%r11");
344 my @N=("%r13","%rdi");
345 $code.=<<___;
346 .type   bn_mul4x_mont_gather5,\@function,6
347 .align  16
348 bn_mul4x_mont_gather5:
349 .Lmul4x_enter:
350         mov     ${num}d,${num}d
351         mov     `($win64?56:8)`(%rsp),%r10d     # load 7th argument
352         push    %rbx
353         push    %rbp
354         push    %r12
355         push    %r13
356         push    %r14
357         push    %r15
358 ___
359 $code.=<<___ if ($win64);
360         lea     -0x28(%rsp),%rsp
361         movaps  %xmm6,(%rsp)
362         movaps  %xmm7,0x10(%rsp)
363 .Lmul4x_alloca:
364 ___
365 $code.=<<___;
366         mov     %rsp,%rax
367         lea     4($num),%r11
368         neg     %r11
369         lea     (%rsp,%r11,8),%rsp      # tp=alloca(8*(num+4))
370         and     \$-1024,%rsp            # minimize TLB usage

372         mov     %rax,8(%rsp,$num,8)     # tp[num+1]=%rsp
373 .Lmul4x_body:
374         mov     $rp,16(%rsp,$num,8)     # tp[num+2]=$rp
375         mov     %rdx,%r12               # reassign $bp
376 ___
377                 $bp="%r12";
378                 $STRIDE=2**5*8;         # 5 is "window size"
379                 $N=$STRIDE/4;           # should match cache line size
380 $code.=<<___;
381         mov     %r10,%r11
382         shr     \$`log($N/8)/log(2)`,%r10
383         and     \$`$N/8-1`,%r11
384         not     %r10
385         lea     .Lmagic_masks(%rip),%rax
386         and     \$`2**5/($N/8)-1`,%r10  # 5 is "window size"
387         lea     96($bp,%r11,8),$bp      # pointer within 1st cache line
388         movq    0(%rax,%r10,8),%xmm4    # set of masks denoting which
389         movq    8(%rax,%r10,8),%xmm5    # cache line contains element
390         movq    16(%rax,%r10,8),%xmm6   # denoted by 7th argument
391         movq    24(%rax,%r10,8),%xmm7
```

```
393         movq    `0*$STRIDE/4-96`($bp),%xmm0
394         movq    `1*$STRIDE/4-96`($bp),%xmm1
395         pand    %xmm4,%xmm0
396         movq    `2*$STRIDE/4-96`($bp),%xmm2
397         pand    %xmm5,%xmm1
398         movq    `3*$STRIDE/4-96`($bp),%xmm3
399         pand    %xmm6,%xmm2
400         por     %xmm1,%xmm0
401         pand    %xmm7,%xmm3
402         por     %xmm2,%xmm0
403         lea     $STRIDE($bp),$bp
404         por     %xmm3,%xmm0

406         movq    %xmm0,$m0               # m0=bp[0]
407         mov     ($n0),$n0               # pull n0[0] value
408         mov     ($ap),%rax

410         xor     $i,$i                   # i=0
411         xor     $j,$j                   # j=0

413         movq    `0*$STRIDE/4-96`($bp),%xmm0
414         movq    `1*$STRIDE/4-96`($bp),%xmm1
415         pand    %xmm4,%xmm0
416         movq    `2*$STRIDE/4-96`($bp),%xmm2
417         pand    %xmm5,%xmm1

419         mov     $n0,$m1
420         mulq    $m0                     # ap[0]*bp[0]
421         mov     %rax,$A[0]
422         mov     ($np),%rax

424         movq    `3*$STRIDE/4-96`($bp),%xmm3
425         pand    %xmm6,%xmm2
426         por     %xmm1,%xmm0
427         pand    %xmm7,%xmm3

429         imulq   $A[0],$m1               # "tp[0]"*n0
430         mov     %rdx,$A[1]

432         por     %xmm2,%xmm0
433         lea     $STRIDE($bp),$bp
434         por     %xmm3,%xmm0

436         mulq    $m1                     # np[0]*m1
437         add     %rax,$A[0]              # discarded
438         mov     8($ap),%rax
439         adc     \$0,%rdx
440         mov     %rdx,$N[1]

442         mulq    $m0
443         add     %rax,$A[1]
444         mov     8($np),%rax
445         adc     \$0,%rdx
446         mov     %rdx,$A[0]

448         mulq    $m1
449         add     %rax,$N[1]
450         mov     16($ap),%rax
451         adc     \$0,%rdx
452         add     $A[1],$N[1]
453         lea     4($j),$j                # j++
454         adc     \$0,%rdx
455         mov     $N[1],(%rsp)
456         mov     %rdx,$N[0]
457         jmp     .L1st4x
```



```
458 .align  16
459 .L1st4x:
460         mulq    $m0                     # ap[j]*bp[0]
461         add     %rax,$A[0]
462         mov     -16($np,$j,8),%rax
463         adc     \$0,%rdx
464         mov     %rdx,$A[1]

466         mulq    $m1                     # np[j]*m1
467         add     %rax,$N[0]
468         mov     -8($ap,$j,8),%rax
469         adc     \$0,%rdx
470         add     $A[0],$N[0]             # np[j]*m1+ap[j]*bp[0]
471         adc     \$0,%rdx
472         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
473         mov     %rdx,$N[1]

475         mulq    $m0                     # ap[j]*bp[0]
476         add     %rax,$A[1]
477         mov     -8($np,$j,8),%rax
478         adc     \$0,%rdx
479         mov     %rdx,$A[0]

481         mulq    $m1                     # np[j]*m1
482         add     %rax,$N[1]
483         mov     ($ap,$j,8),%rax
484         adc     \$0,%rdx
485         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[0]
486         adc     \$0,%rdx
487         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]
488         mov     %rdx,$N[0]

490         mulq    $m0                     # ap[j]*bp[0]
491         add     %rax,$A[0]
492         mov     ($np,$j,8),%rax
493         adc     \$0,%rdx
494         mov     %rdx,$A[1]

496         mulq    $m1                     # np[j]*m1
497         add     %rax,$N[0]
498         mov     8($ap,$j,8),%rax
499         adc     \$0,%rdx
500         add     $A[0],$N[0]             # np[j]*m1+ap[j]*bp[0]
501         adc     \$0,%rdx
502         mov     $N[0],-8(%rsp,$j,8)     # tp[j-1]
503         mov     %rdx,$N[1]

505         mulq    $m0                     # ap[j]*bp[0]
506         add     %rax,$A[1]
507         mov     8($np,$j,8),%rax
508         adc     \$0,%rdx
509         lea     4($j),$j                # j++
510         mov     %rdx,$A[0]

512         mulq    $m1                     # np[j]*m1
513         add     %rax,$N[1]
514         mov     -16($ap,$j,8),%rax
515         adc     \$0,%rdx
516         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[0]
517         adc     \$0,%rdx
518         mov     $N[1],-32(%rsp,$j,8)    # tp[j-1]
519         mov     %rdx,$N[0]
520         cmp     $num,$j
521         jl      .L1st4x

523         mulq    $m0                     # ap[j]*bp[0]
```

```
524         add     %rax,$A[0]
525         mov     -16($np,$j,8),%rax
526         adc     \$0,%rdx
527         mov     %rdx,$A[1]

529         mulq    $m1                     # np[j]*m1
530         add     %rax,$N[0]
531         mov     -8($ap,$j,8),%rax
532         adc     \$0,%rdx
533         add     $A[0],$N[0]             # np[j]*m1+ap[j]*bp[0]
534         adc     \$0,%rdx
535         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
536         mov     %rdx,$N[1]

538         mulq    $m0                     # ap[j]*bp[0]
539         add     %rax,$A[1]
540         mov     -8($np,$j,8),%rax
541         adc     \$0,%rdx
542         mov     %rdx,$A[0]

544         mulq    $m1                     # np[j]*m1
545         add     %rax,$N[1]
546         mov     ($ap),%rax              # ap[0]
547         adc     \$0,%rdx
548         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[0]
549         adc     \$0,%rdx
550         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]
551         mov     %rdx,$N[0]

553         movq    %xmm0,$m0               # bp[1]

555         xor     $N[1],$N[1]
556         add     $A[0],$N[0]
557         adc     \$0,$N[1]
558         mov     $N[0],-8(%rsp,$j,8)
559         mov     $N[1],(%rsp,$j,8)       # store upmost overflow bit

561         lea     1($i),$i                # i++
562 .align  4
563 .Louter4x:
564         xor     $j,$j                   # j=0
565         movq    '0*$STRIDE/4-96'($bp),%xmm0
566         movq    '1*$STRIDE/4-96'($bp),%xmm1
567         pand    %xmm4,%xmm0
568         movq    '2*$STRIDE/4-96'($bp),%xmm2
569         pand    %xmm5,%xmm1

571         mov     (%rsp),$A[0]
572         mov     $n0,$m1
573         mulq    $m0                     # ap[0]*bp[i]
574         add     %rax,$A[0]              # ap[0]*bp[i]+tp[0]
575         mov     ($np),%rax
576         adc     \$0,%rdx

578         movq    '3*$STRIDE/4-96'($bp),%xmm3
579         pand    %xmm6,%xmm2
580         por     %xmm1,%xmm0
581         pand    %xmm7,%xmm3

583         imulq   $A[0],$m1               # tp[0]*n0
584         mov     %rdx,$A[1]

586         por     %xmm2,%xmm0
587         lea     $STRIDE($bp),$bp
588         por     %xmm3,%xmm0
```



```
590         mulq    $m1                     # np[0]*m1
591         add     %rax,$A[0]              # "$N[0]", discarded
592         mov     8($ap),%rax
593         adc     \$0,%rdx
594         mov     %rdx,$N[1]

596         mulq    $m0                     # ap[j]*bp[i]
597         add     %rax,$A[1]
598         mov     8($np),%rax
599         adc     \$0,%rdx
600         add     8(%rsp),$A[1]           # +tp[1]
601         adc     \$0,%rdx
602         mov     %rdx,$A[0]

604         mulq    $m1                     # np[j]*m1
605         add     %rax,$N[1]
606         mov     16($ap),%rax
607         adc     \$0,%rdx
608         add     $A[1],$N[1]             # np[j]*m1+ap[j]*bp[i]+tp[j]
609         lea     4($j),$j                # j+=2
610         adc     \$0,%rdx
611         mov     %rdx,$N[0]
612         jmp     .Linner4x
613 .align  16
614 .Linner4x:
615         mulq    $m0                     # ap[j]*bp[i]
616         add     %rax,$A[0]
617         mov     -16($np,$j,8),%rax
618         adc     \$0,%rdx
619         add     -16(%rsp,$j,8),$A[0]    # ap[j]*bp[i]+tp[j]
620         adc     \$0,%rdx
621         mov     %rdx,$A[1]

623         mulq    $m1                     # np[j]*m1
624         add     %rax,$N[0]
625         mov     -8($ap,$j,8),%rax
626         adc     \$0,%rdx
627         add     $A[0],$N[0]
628         adc     \$0,%rdx
629         mov     $N[1],-32(%rsp,$j,8)    # tp[j-1]
630         mov     %rdx,$N[1]

632         mulq    $m0                     # ap[j]*bp[i]
633         add     %rax,$A[1]
634         mov     -8($np,$j,8),%rax
635         adc     \$0,%rdx
636         add     -8(%rsp,$j,8),$A[1]
637         adc     \$0,%rdx
638         mov     %rdx,$A[0]

640         mulq    $m1                     # np[j]*m1
641         add     %rax,$N[1]
642         mov     ($ap,$j,8),%rax
643         adc     \$0,%rdx
644         add     $A[1],$N[1]
645         adc     \$0,%rdx
646         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
647         mov     %rdx,$N[0]

649         mulq    $m0                     # ap[j]*bp[i]
650         add     %rax,$A[0]
651         mov     ($np,$j,8),%rax
652         adc     \$0,%rdx
653         add     (%rsp,$j,8),$A[0]       # ap[j]*bp[i]+tp[j]
654         adc     \$0,%rdx
655         mov     %rdx,$A[1]
```

```
657         mulq    $m1                     # np[j]*m1
658         add     %rax,$N[0]
659         mov     8($ap,$j,8),%rax
660         adc     \$0,%rdx
661         add     $A[0],$N[0]
662         adc     \$0,%rdx
663         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]
664         mov     %rdx,$N[1]

666         mulq    $m0                     # ap[j]*bp[i]
667         add     %rax,$A[1]
668         mov     8($np,$j,8),%rax
669         adc     \$0,%rdx
670         add     8(%rsp,$j,8),$A[1]
671         adc     \$0,%rdx
672         lea     4($j),$j                # j++
673         mov     %rdx,$A[0]

675         mulq    $m1                     # np[j]*m1
676         add     %rax,$N[1]
677         mov     -16($ap,$j,8),%rax
678         adc     \$0,%rdx
679         add     $A[1],$N[1]
680         adc     \$0,%rdx
681         mov     $N[0],-40(%rsp,$j,8)    # tp[j-1]
682         mov     %rdx,$N[0]
683         cmp     $num,$j
684         jl      .Linner4x

686         mulq    $m0                     # ap[j]*bp[i]
687         add     %rax,$A[0]
688         mov     -16($np,$j,8),%rax
689         adc     \$0,%rdx
690         add     -16(%rsp,$j,8),$A[0]    # ap[j]*bp[i]+tp[j]
691         adc     \$0,%rdx
692         mov     %rdx,$A[1]

694         mulq    $m1                     # np[j]*m1
695         add     %rax,$N[0]
696         mov     -8($ap,$j,8),%rax
697         adc     \$0,%rdx
698         add     $A[0],$N[0]
699         adc     \$0,%rdx
700         mov     $N[1],-32(%rsp,$j,8)    # tp[j-1]
701         mov     %rdx,$N[1]

703         mulq    $m0                     # ap[j]*bp[i]
704         add     %rax,$A[1]
705         mov     -8($np,$j,8),%rax
706         adc     \$0,%rdx
707         add     -8(%rsp,$j,8),$A[1]
708         adc     \$0,%rdx
709         lea     1($i),$i                # i++
710         mov     %rdx,$A[0]

712         mulq    $m1                     # np[j]*m1
713         add     %rax,$N[1]
714         mov     ($ap),%rax              # ap[0]
715         adc     \$0,%rdx
716         add     $A[1],$N[1]
717         adc     \$0,%rdx
718         mov     $N[0],-24(%rsp,$j,8)    # tp[j-1]
719         mov     %rdx,$N[0]

721         movq    %xmm0,$m0               # bp[i+1]
```



```
722         mov     $N[1],-16(%rsp,$j,8)    # tp[j-1]

724         xor     $N[1],$N[1]
725         add     $A[0],$N[0]
726         adc     \$0,$N[1]
727         add     (%rsp,$num,8),$N[0]     # pull upmost overflow bit
728         adc     \$0,$N[1]
729         mov     $N[0],-8(%rsp,$j,8)
730         mov     $N[1],(%rsp,$j,8)       # store upmost overflow bit

732         cmp     $num,$i
733         jl      .Louter4x
734 ___
735 {
736 my @ri=("%rax","%rdx",$m0,$m1);
737 $code.=<<___;
738         mov     16(%rsp,$num,8),$rp     # restore $rp
739         mov     0(%rsp),@ri[0]          # tp[0]
740         pxor    %xmm0,%xmm0
741         mov     8(%rsp),@ri[1]          # tp[1]
742         shr     \$2,$num                # num/=4
743         lea     (%rsp),$ap              # borrow ap for tp
744         xor     $i,$i                   # i=0 and clear CF!

746         sub     0($np),@ri[0]
747         mov     16($ap),@ri[2]          # tp[2]
748         mov     24($ap),@ri[3]          # tp[3]
749         sbb     8($np),@ri[1]
750         lea     -1($num),$j             # j=num/4-1
751         jmp     .Lsub4x
752 .align  16
753 .Lsub4x:
754         mov     @ri[0],0($rp,$i,8)      # rp[i]=tp[i]-np[i]
755         mov     @ri[1],8($rp,$i,8)      # rp[i]=tp[i]-np[i]
756         sbb     16($np,$i,8),@ri[2]
757         mov     32($ap,$i,8),@ri[0]     # tp[i+1]
758         mov     40($ap,$i,8),@ri[1]
759         sbb     24($np,$i,8),@ri[3]
760         mov     @ri[2],16($rp,$i,8)     # rp[i]=tp[i]-np[i]
761         mov     @ri[3],24($rp,$i,8)     # rp[i]=tp[i]-np[i]
762         sbb     32($np,$i,8),@ri[0]
763         mov     48($ap,$i,8),@ri[2]
764         mov     56($ap,$i,8),@ri[3]
765         sbb     40($np,$i,8),@ri[1]
766         lea     4($i),$i                # i++
767         dec     $j                      # doesnn't affect CF!
768         jnz     .Lsub4x

770         mov     @ri[0],0($rp,$i,8)      # rp[i]=tp[i]-np[i]
771         mov     32($ap,$i,8),@ri[0]     # load overflow bit
772         sbb     16($np,$i,8),@ri[2]
773         mov     @ri[1],8($rp,$i,8)      # rp[i]=tp[i]-np[i]
774         sbb     24($np,$i,8),@ri[3]
775         mov     @ri[2],16($rp,$i,8)     # rp[i]=tp[i]-np[i]

777         sbb     \$0,@ri[0]              # handle upmost overflow bit
778         mov     @ri[3],24($rp,$i,8)     # rp[i]=tp[i]-np[i]
779         xor     $i,$i                   # i=0
780         and     @ri[0],$ap
781         not     @ri[0]
782         mov     $rp,$np
783         and     @ri[0],$np
784         lea     -1($num),$j
785         or      $np,$ap                 # ap=borrow?tp:rp

787         movdqu  ($ap),%xmm1
```

```
788         movdqa  %xmm0,(%rsp)
789         movdqu  %xmm1,($rp)
790         jmp     .Lcopy4x
791 .align  16
792 .Lcopy4x:                                       # copy or in-place refresh
793         movdqu  16($ap,$i),%xmm2
794         movdqu  32($ap,$i),%xmm1
795         movdqa  %xmm0,16(%rsp,$i)
796         movdqu  %xmm2,16($rp,$i)
797         movdqa  %xmm0,32(%rsp,$i)
798         movdqu  %xmm1,32($rp,$i)
799         lea     32($i),$i
800         dec     $j
801         jnz     .Lcopy4x

803         shl     \$2,$num
804         movdqu  16($ap,$i),%xmm2
805         movdqa  %xmm0,16(%rsp,$i)
806         movdqu  %xmm2,16($rp,$i)
807 ___
808 }
809 $code.=<<___;
810         mov     8(%rsp,$num,8),%rsi     # restore %rsp
811         mov     \$1,%rax
812 ___
813 $code.=<<___ if ($win64);
814         movaps  (%rsi),%xmm6
815         movaps  0x10(%rsi),%xmm7
816         lea     0x28(%rsi),%rsi
817 ___
818 $code.=<<___;
819         mov     (%rsi),%r15
820         mov     8(%rsi),%r14
821         mov     16(%rsi),%r13
822         mov     24(%rsi),%r12
823         mov     32(%rsi),%rbp
824         mov     40(%rsi),%rbx
825         lea     48(%rsi),%rsp
826 .Lmul4x_epilogue:
827         ret
828 .size   bn_mul4x_mont_gather5,.-bn_mul4x_mont_gather5
829 ___
830 }}}

832 {
833 my ($inp,$num,$tbl,$idx)=$win64?("%rcx","%rdx","%r8",  "%r9") : # Win64 order
834                              ("%rdi","%rsi","%rdx","%rcx"); # Unix order
835 my $out=$inp;
836 my $STRIDE=2**5*8;
837 my $N=$STRIDE/4;

839 $code.=<<___;
840 .globl  bn_scatter5
841 .type   bn_scatter5,\@abi-omnipotent
842 .align  16
843 bn_scatter5:
844         cmp     \$0, $num
845         jz      .Lscatter_epilogue
846         lea     ($tbl,$idx,8),$tbl
847 .Lscatter:
848         mov     ($inp),%rax
849         lea     8($inp),$inp
850         mov     %rax,($tbl)
851         lea     32*8($tbl),$tbl
852         sub     \$1,$num
853         jnz     .Lscatter
```



```
854 .Lscatter_epilogue:
855         ret
856 .size   bn_scatter5,.-bn_scatter5

858 .globl  bn_gather5
859 .type   bn_gather5,\@abi-omnipotent
860 .align  16
861 bn_gather5:
862 ___
863 $code.=<<___ if ($win64);
864 .LSEH_begin_bn_gather5:
865         # I can’t trust assembler to use specific encoding:-(
866         .byte   0x48,0x83,0xec,0x28             #sub    \$0x28,%rsp
867         .byte   0x0f,0x29,0x34,0x24             #movaps %xmm6,(%rsp)
868         .byte   0x0f,0x29,0x7c,0x24,0x10        #movdqa %xmm7,0x10(%rsp)
869 ___
870 $code.=<<___;
871         mov     $idx,%r11
872         shr     \$‘log($N/8)/log(2)‘,$idx
873         and     \$‘$N/8-1‘,%r11
874         not     $idx
875         lea     .Lmagic_masks(%rip),%rax
876         and     \$‘2**5/($N/8)-1‘,$idx  # 5 is "window size"
877         lea     96($tbl,%r11,8),$tbl    # pointer within 1st cache line
878         movq    0(%rax,$idx,8),%xmm4    # set of masks denoting which
879         movq    8(%rax,$idx,8),%xmm5    # cache line contains element
880         movq    16(%rax,$idx,8),%xmm6   # denoted by 7th argument
881         movq    24(%rax,$idx,8),%xmm7
882         jmp     .Lgather
883 .align  16
884 .Lgather:
885         movq    ‘0*$STRIDE/4-96‘($tbl),%xmm0
886         movq    ‘1*$STRIDE/4-96‘($tbl),%xmm1
887         pand    %xmm4,%xmm0
888         movq    ‘2*$STRIDE/4-96‘($tbl),%xmm2
889         pand    %xmm5,%xmm1
890         movq    ‘3*$STRIDE/4-96‘($tbl),%xmm3
891         pand    %xmm6,%xmm2
892         por     %xmm1,%xmm0
893         pand    %xmm7,%xmm3
894         por     %xmm2,%xmm0
895         lea     $STRIDE($tbl),$tbl
896         por     %xmm3,%xmm0

898         movq    %xmm0,($out)            # m0=bp[0]
899         lea     8($out),$out
900         sub     \$1,$num
901         jnz     .Lgather
902 ___
903 $code.=<<___ if ($win64);
904         movaps  (%rsp),%xmm6
905         movaps  0x10(%rsp),%xmm7
906         lea     0x28(%rsp),%rsp
907 ___
908 $code.=<<___;
909         ret
910 .LSEH_end_bn_gather5:
911 .size   bn_gather5,.-bn_gather5
912 ___
913 }
914 $code.=<<___;
915 .align  64
916 .Lmagic_masks:
917         .long   0,0, 0,0, 0,0, -1,-1
918         .long   0,0, 0,0, 0,0,  0,0
919 .asciz  "Montgomery Multiplication with scatter/gather for x86_64, CRYPTOGAMS by
```

```
___

# EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
#               CONTEXT *context,DISPATCHER_CONTEXT *disp)
if ($win64) {
$rec="%rcx";
$frame="%rdx";
$context="%r8";
$disp="%r9";

$code.=<<___;
.extern __imp_RtlVirtualUnwind
.type   mul_handler,\@abi-omnipotent
.align  16
mul_handler:
        push    %rsi
        push    %rdi
        push    %rbx
        push    %rbp
        push    %r12
        push    %r13
        push    %r14
        push    %r15
        pushfq
        sub     \$64,%rsp

        mov     120($context),%rax      # pull context->Rax
        mov     248($context),%rbx      # pull context->Rip

        mov     8($disp),%rsi           # disp->ImageBase
        mov     56($disp),%r11          # disp->HandlerData

        mov     0(%r11),%r10d           # HandlerData[0]
        lea     (%rsi,%r10),%r10        # end of prologue label
        cmp     %r10,%rbx               # context->Rip<end of prologue label
        jb      .Lcommon_seh_tail

        lea     ‘40+48‘(%rax),%rax

        mov     4(%r11),%r10d           # HandlerData[1]
        lea     (%rsi,%r10),%r10        # end of alloca label
        cmp     %r10,%rbx               # context->Rip<end of alloca label
        jb      .Lcommon_seh_tail

        mov     152($context),%rax      # pull context->Rsp

        mov     8(%r11),%r10d           # HandlerData[2]
        lea     (%rsi,%r10),%r10        # epilogue label
        cmp     %r10,%rbx               # context->Rip>=epilogue label
        jae     .Lcommon_seh_tail

        mov     192($context),%r10      # pull $num
        mov     8(%rax,%r10,8),%rax     # pull saved stack pointer

        movaps  (%rax),%xmm0
        movaps  16(%rax),%xmm1
        lea     ‘40+48‘(%rax),%rax

        mov     -8(%rax),%rbx
        mov     -16(%rax),%rbp
        mov     -24(%rax),%r12
        mov     -32(%rax),%r13
        mov     -40(%rax),%r14
        mov     -48(%rax),%r15
        mov     %rbx,144($context)      # restore context->Rbx
        mov     %rbp,160($context)      # restore context->Rbp
```



```
        mov     %r12,216($context)      # restore context->R12
        mov     %r13,224($context)      # restore context->R13
        mov     %r14,232($context)      # restore context->R14
        mov     %r15,240($context)      # restore context->R15
        movups  %xmm0,512($context)     # restore context->Xmm6
        movups  %xmm1,528($context)     # restore context->Xmm7

.Lcommon_seh_tail:
        mov     8(%rax),%rdi
        mov     16(%rax),%rsi
        mov     %rax,152($context)      # restore context->Rsp
        mov     %rsi,168($context)      # restore context->Rsi
        mov     %rdi,176($context)      # restore context->Rdi

        mov     40($disp),%rdi          # disp->ContextRecord
        mov     $context,%rsi           # context
        mov     \$154,%ecx              # sizeof(CONTEXT)
        .long   0xa548f3fc              # cld; rep movsq

        mov     $disp,%rsi
        xor     %rcx,%rcx               # arg1, UNW_FLAG_NHANDLER
        mov     8(%rsi),%rdx            # arg2, disp->ImageBase
        mov     0(%rsi),%r8             # arg3, disp->ControlPc
        mov     16(%rsi),%r9            # arg4, disp->FunctionEntry
        mov     40(%rsi),%r10           # disp->ContextRecord
        lea     56(%rsi),%r11           # &disp->HandlerData
        lea     24(%rsi),%r12           # &disp->EstablisherFrame
        mov     %r10,32(%rsp)           # arg5
        mov     %r11,40(%rsp)           # arg6
        mov     %r12,48(%rsp)           # arg7
        mov     %rcx,56(%rsp)           # arg8, (NULL)
        call    *__imp_RtlVirtualUnwind(%rip)

        mov     \$1,%eax                # ExceptionContinueSearch
        add     \$64,%rsp
        popfq
        pop     %r15
        pop     %r14
        pop     %r13
        pop     %r12
        pop     %rbp
        pop     %rbx
        pop     %rdi
        pop     %rsi
        ret
.size   mul_handler,.-mul_handler

.section        .pdata
.align  4
        .rva    .LSEH_begin_bn_mul_mont_gather5
        .rva    .LSEH_end_bn_mul_mont_gather5
        .rva    .LSEH_info_bn_mul_mont_gather5

        .rva    .LSEH_begin_bn_mul4x_mont_gather5
        .rva    .LSEH_end_bn_mul4x_mont_gather5
        .rva    .LSEH_info_bn_mul4x_mont_gather5

        .rva    .LSEH_begin_bn_gather5
        .rva    .LSEH_end_bn_gather5
        .rva    .LSEH_info_bn_gather5

.section        .xdata
.align  8
.LSEH_info_bn_mul_mont_gather5:
        .byte   9,0,0,0
        .rva    mul_handler
```

```
1052         .rva    .Lmul_alloca,.Lmul_body,.Lmul_epilogue          # HandlerData[]
1053 .align  8
1054 .LSEH_info_bn_mul4x_mont_gather5:
1055         .byte   9,0,0,0
1056         .rva    mul_handler
1057         .rva    .Lmul4x_alloca,.Lmul4x_body,.Lmul4x_epilogue    # HandlerData[]
1058 .align  8
1059 .LSEH_info_bn_gather5:
1060         .byte   0x01,0x0d,0x05,0x00
1061         .byte   0x0d,0x78,0x01,0x00     #movaps 0x10(rsp),xmm7
1062         .byte   0x08,0x68,0x00,0x00     #movaps (rsp),xmm6
1063         .byte   0x04,0x42,0x00,0x00     #sub    rsp,0x28
1064 .align  8
1065 ___
1066 }

1068 $code =~ s/\`([^\`]*)\`/eval($1)/gem;

1070 print $code;
1071 close STDOUT;
1072 #endif /* ! codereview */
```

```
**********************************************************
   34196 Wed Aug 13 19:53:11 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86_64-xlate.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# Ascetic x86_64 AT&T to MASM/NASM assembler translator by <appro>.
#
# Why AT&T to MASM and not vice versa? Several reasons. Because AT&T
# format is way easier to parse. Because it's simpler to "gear" from
# Unix ABI to Windows one [see cross-reference "card" at the end of
# file]. Because Linux targets were available first...
#
# In addition the script also "distills" code suitable for GNU
# assembler, so that it can be compiled with more rigid assemblers,
# such as Solaris /usr/ccs/bin/as.
#
# This translator is not designed to convert *arbitrary* assembler
# code from AT&T format to MASM one. It's designed to convert just
# enough to provide for dual-ABI OpenSSL modules development...
# There *are* limitations and you might have to modify your assembler
# code or this script to achieve the desired result...
#
# Currently recognized limitations:
#
# - can't use multiple ops per line;
#
# Dual-ABI styling rules.
#
# 1. Adhere to Unix register and stack layout [see cross-reference
#    ABI "card" at the end for explanation].
# 2. Forget about "red zone," stick to more traditional blended
#    stack frame allocation. If volatile storage is actually required
#    that is. If not, just leave the stack as is.
# 3. Functions tagged with ".type name,@function" get crafted with
#    unified Win64 prologue and epilogue automatically. If you want
#    to take care of ABI differences yourself, tag functions as
#    ".type name,@abi-omnipotent" instead.
# 4. To optimize the Win64 prologue you can specify number of input
#    arguments as ".type name,@function,N." Keep in mind that if N is
#    larger than 6, then you *have to* write "abi-omnipotent" code,
#    because >6 cases can't be addressed with unified prologue.
# 5. Name local labels as .L*, do *not* use dynamic labels such as 1:
#    (sorry about latter).
# 6. Don't use [or hand-code with .byte] "rep ret." "ret" mnemonic is
#    required to identify the spots, where to inject Win64 epilogue!
#    But on the pros, it's then prefixed with rep automatically:-)
# 7. Stick to explicit ip-relative addressing. If you have to use
#    GOTPCREL addressing, stick to mov symbol@GOTPCREL(%rip),%r??.
#    Both are recognized and translated to proper Win64 addressing
#    modes. To support legacy code a synthetic directive, .picmeup,
#    is implemented. It puts address of the *next* instruction into
#    target register, e.g.:
#
#               .picmeup        %rax
#               lea             .Label-.(%rax),%rax
#
# 8. In order to provide for structured exception handling unified
#    Win64 prologue copies %rsp value to %rax. For further details
#    see SEH paragraph at the end.
# 9. .init segment is allowed to contain calls to functions only.
# a. If function accepts more than 4 arguments *and* >4th argument
#    is declared as non 64-bit value, do clear its upper part.
```



```
my $flavour = shift;
my $output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

open STDOUT,">$output" || die "can't open $output: $!"
        if (defined($output));

my $gas=1;      $gas=0 if ($output =~ /\.asm$/);
my $self=1;     $self=0 if (!$gas);
my $win64=0;
my $prefix="sunw_";
my $decor=".L";

my $masmref=8 + 50727*2**-32;   # 8.00.50727 shipped with VS2005
my $masm=0;
my $PTR=" PTR";

my $nasmref=2.03;
my $nasm=0;

if    ($flavour eq "mingw64")   { $gas=1; $self=0; $win64=1;
                                  $prefix=`echo __USER_LABEL_PREFIX__ | $ENV{CC}
                                  chomp($prefix);
                                }
elsif ($flavour eq "macosx")    { $gas=1; $self=0; $prefix="_"; $decor="L\$"; }
elsif ($flavour eq "masm")      { $gas=0; $self=0; $masm=$masmref; $win64=1; $dec
elsif ($flavour eq "nasm")      { $gas=0; $self=0; $nasm=$nasmref; $win64=1; $dec
elsif (!$gas)
{   if ($ENV{ASM} =~ m/nasm/ && `nasm -v` =~ m/version ([0-9]+)\.([0-9]+)/i)
    {   $nasm = $1 + $2*0.01; $PTR="";  }
    elsif (`ml64 2>&1` =~ m/Version ([0-9]+)\.([0-9]+)(\.([0-9]+))?/)
    {   $masm = $1 + $2*2**-16 + $4*2**-32;   }
    die "no assembler found on %PATH" if (!($nasm || $masm));
    $win64=1;
    $self=0;
    $decor="\$L\$";
}

my $current_segment;
my $current_function;
my %globals;

{ package opcode;       # pick up opcodes
    sub re {
        my      $self = shift;  # single instance in enough...
        local   *line = shift;
        undef   $ret;

        if ($line =~ /^([a-z][a-z0-9]*)/i) {
            $self->{op} = $1;
            $ret = $self;
            $line = substr($line,@+[0]); $line =~ s/^\s+//;

            undef $self->{sz};
            if ($self->{op} =~ /^(movz)x?([bw]).*/) {   # movz is pain...
                $self->{op} = $1;
                $self->{sz} = $2;
            } elsif ($self->{op} =~ /call|jmp/) {
                $self->{sz} = "";
            } elsif ($self->{op} =~ /^p/ && $' !~ /^(ush|op|insrw)/) { # SSEn
                $self->{sz} = "";
            } elsif ($self->{op} =~ /^v/) { # VEX
                $self->{sz} = "";
            } elsif ($self->{op} =~ /movq/ && $line =~ /%xmm/) {
                $self->{sz} = "";
            } elsif ($self->{op} =~ /([a-z]{3,})([qlwb])$/) {
```

```
126                 $self->{op} = $1;
127                 $self->{sz} = $2;
128             }
129         }
130         $ret;
131     }
132     sub size {
133         my $self = shift;
134         my $sz   = shift;
135         $self->{sz} = $sz if (defined($sz) && !defined($self->{sz}));
136         $self->{sz};
137     }
138     sub out {
139         my $self = shift;
140         if ($gas) {
141             if ($self->{op} eq "movz") {        # movz is pain...
142                 sprintf "%s%s%s",$self->{op},$self->{sz},shift;
143             } elsif ($self->{op} =~ /^set/) {
144                 "$self->{op}";
145             } elsif ($self->{op} eq "ret") {
146                 my $epilogue = "";
147                 if ($win64 && $current_function->{abi} eq "svr4") {
148                     $epilogue = "movq   8(%rsp),%rdi\n\t" .
149                                 "movq   16(%rsp),%rsi\n\t";
150                 }
151                 $epilogue . ".byte      0xf3,0xc3";
152             } elsif ($self->{op} eq "call" && !$elf && $current_segment eq ".ini
153                 ".p2align\t3\n\t.quad";
154             } else {
155                 "$self->{op}$self->{sz}";
156             }
157         } else {
158             $self->{op} =~ s/^movz/movzx/;
159             if ($self->{op} eq "ret") {
160                 $self->{op} = "";
161                 if ($win64 && $current_function->{abi} eq "svr4") {
162                     $self->{op} = "mov  rdi,QWORD${PTR}[8+rsp]\t;WIN64 epilogue\
163                                   "mov  rsi,QWORD${PTR}[16+rsp]\n\t";
164                 }
165                 $self->{op} .= "DB\t0F3h,0C3h\t\t;repret";
166             } elsif ($self->{op} =~ /^(pop|push)f/) {
167                 $self->{op} .= $self->{sz};
168             } elsif ($self->{op} eq "call" && $current_segment eq ".CRT\$XCU") {
169                 $self->{op} = "\tDQ";
170             }
171             $self->{op};
172         }
173     }
174     sub mnemonic {
175         my $self=shift;
176         my $op=shift;
177         $self->{op}=$op if (defined($op));
178         $self->{op};
179     }
180 }
181 { package const;        # pick up constants, which start with $
182     sub re {
183         my      $self = shift;  # single instance in enough...
184         local   *line = shift;
185         undef   $ret;

187         if ($line =~ /^\$([^,]+)/) {
188             $self->{value} = $1;
189             $ret = $self;
190             $line = substr($line,@+[0]); $line =~ s/^\s+//;
191         }
```



```
192         $ret;
193     }
194     sub out {
195         my $self = shift;

197         if ($gas) {
198             # Solaris /usr/ccs/bin/as can't handle multiplications
199             # in $self->{value}
200             $self->{value} =~ s/(?<![\w\$\.])(0x?[0-9a-f]+)/oct($1)/egi;
201             $self->{value} =~ s/([0-9]+\s*[\*\/\%]\s*[0-9]+)/eval($1)/eg;
202             sprintf "\$%s",$self->{value};
203         } else {
204             $self->{value} =~ s/(0b[0-1]+)/oct($1)/eig;
205             $self->{value} =~ s/0x([0-9a-f]+)/0$1h/ig if ($masm);
206             sprintf "%s",$self->{value};
207         }
208     }
209 }
210 { package ea;           # pick up effective addresses: expr(%reg,%reg,scale)
211     sub re {
212         my      $self = shift;  # single instance in enough...
213         local   *line = shift;
214         undef   $ret;

216         # optional * ---vvv--- appears in indirect jmp/call
217         if ($line =~ /^(\*?)([^\(,]*)\(([%\w,]+)\)/) {
218             $self->{asterisk} = $1;
219             $self->{label} = $2;
220             ($self->{base},$self->{index},$self->{scale})=split(/,/,$3);
221             $self->{scale} = 1 if (!defined($self->{scale}));
222             $ret = $self;
223             $line = substr($line,@+[0]); $line =~ s/^\s+//;

225             if ($win64 && $self->{label} =~ s/\@GOTPCREL//) {
226                 die if (opcode->mnemonic() ne "mov");
227                 opcode->mnemonic("lea");
228             }
229             $self->{base}  =~ s/^%//;
230             $self->{index} =~ s/^%// if (defined($self->{index}));
231         }
232         $ret;
233     }
234     sub size {}
235     sub out {
236         my $self = shift;
237         my $sz = shift;

239         $self->{label} =~ s/([_a-z][_a-z0-9]*)/$globals{$1} or $1/gei;
240         $self->{label} =~ s/\.L/$decor/g;

242         # Silently convert all EAs to 64-bit. This is required for
243         # elder GNU assembler and results in more compact code,
244         # *but* most importantly AES module depends on this feature!
245         $self->{index} =~ s/^[er](.?[0-9xpi])[d]?$/r\1/;
246         $self->{base}  =~ s/^[er](.?[0-9xpi])[d]?$/r\1/;

248         # Solaris /usr/ccs/bin/as can't handle multiplications
249         # in $self->{label}, new gas requires sign extension...
250         use integer;
251         $self->{label} =~ s/(?<![\w\$\.])(0x?[0-9a-f]+)/oct($1)/egi;
252         $self->{label} =~ s/([0-9]+\s*[\*\/\%]\s*[0-9]+)/eval($1)/eg;
253         $self->{label} =~ s/([0-9]+)/$1<<32>>32/eg;

255         if ($gas) {
256             $self->{label} =~ s/^___imp_/__imp__/   if ($flavour eq "mingw64");
```

```
258             if (defined($self->{index})) {
259                 sprintf "%s%s(%s,%%%s,%d)",$self->{asterisk},
260                                            $self->{label},
261                                            $self->{base}?"%$self->{base}":"",
262                                            $self->{index},$self->{scale};
263             } else {
264                 sprintf "%s%s(%%%s)",   $self->{asterisk},$self->{label},$self->
265             }
266         } else {
267             %szmap = (  b=>"BYTE$PTR", w=>"WORD$PTR", l=>"DWORD$PTR",
268                         q=>"QWORD$PTR",o=>"OWORD$PTR",x=>"XMMWORD$PTR" );
269
270             $self->{label} =~ s/\./\$/g;
271             $self->{label} =~ s/(?<![\w\$\.])0x([0-9a-f]+)/0$1h/ig;
272             $self->{label} = "($self->{label})" if ($self->{label} =~ /[\*\+\-\/
273             $sz="q" if ($self->{asterisk} || opcode->mnemonic() eq "movq");
274             $sz="l" if (opcode->mnemonic() eq "movd");
275
276             if (defined($self->{index})) {
277                 sprintf "%s[%s%s*%d%s]",$szmap{$sz},
278                                            $self->{label}?"$self->{label}+":"",
279                                            $self->{index},$self->{scale},
280                                            $self->{base}?"+$self->{base}":"";
281             } elsif ($self->{base} eq "rip") {
282                 sprintf "%s[%s]",$szmap{$sz},$self->{label};
283             } else {
284                 sprintf "%s[%s%s]",$szmap{$sz},
285                                            $self->{label}?"$self->{label}+":"",
286                                            $self->{base};
287             }
288         }
289     }
290 }
291 { package register;     # pick up registers, which start with %.
292     sub re {
293         my      $class = shift; # muliple instances...
294         my      $self = {};
295         local   *line = shift;
296         undef   $ret;
297
298         # optional * ---vvv--- appears in indirect jmp/call
299         if ($line =~ /^(\*?)%(\w+)/) {
300             bless $self,$class;
301             $self->{asterisk} = $1;
302             $self->{value} = $2;
303             $ret = $self;
304             $line = substr($line,@+[0]); $line =~ s/^\s+//;
305         }
306         $ret;
307     }
308     sub size {
309         my      $self = shift;
310         undef   $ret;
311
312         if    ($self->{value} =~ /^r[\d]+b$/i)  { $ret="b"; }
313         elsif ($self->{value} =~ /^r[\d]+w$/i)  { $ret="w"; }
314         elsif ($self->{value} =~ /^r[\d]+d$/i)  { $ret="l"; }
315         elsif ($self->{value} =~ /^r[\w]+$/i)   { $ret="q"; }
316         elsif ($self->{value} =~ /^[a-d][hl]$/i){ $ret="b"; }
317         elsif ($self->{value} =~ /^[\w]{2}l$/i) { $ret="b"; }
318         elsif ($self->{value} =~ /^[\w]{2}$/i)  { $ret="w"; }
319         elsif ($self->{value} =~ /^e[a-z]{2}$/i){ $ret="l"; }
320
321         $ret;
322     }
323     sub out {
```



```
324         my $self = shift;
325         if ($gas)       { sprintf "%s%%%s",$self->{asterisk},$self->{value}; }
326         else            { $self->{value}; }
327     }
328 }
329 { package label;        # pick up labels, which end with :
330     sub re {
331         my      $self = shift;  # single instance is enough...
332         local   *line = shift;
333         undef   $ret;
334
335         if ($line =~ /(^[\.\w]+)\:/) {
336             $self->{value} = $1;
337             $ret = $self;
338             $line = substr($line,@+[0]); $line =~ s/^\s+//;
339
340             $self->{value} =~ s/^\.L/$decor/;
341         }
342         $ret;
343     }
344     sub out {
345         my $self = shift;
346
347         if ($gas) {
348             my $func = ($globals{$self->{value}} or $self->{value}) . ":";
349             if ($win64  &&
350                         $current_function->{name} eq $self->{value} &&
351                         $current_function->{abi} eq "svr4") {
352                 $func .= "\n";
353                 $func .= "      movq    %rdi,8(%rsp)\n";
354                 $func .= "      movq    %rsi,16(%rsp)\n";
355                 $func .= "      movq    %rsp,%rax\n";
356                 $func .= "${decor}SEH_begin_$current_function->{name}:\n";
357                 my $narg = $current_function->{narg};
358                 $narg=6 if (!defined($narg));
359                 $func .= "      movq    %rcx,%rdi\n" if ($narg>0);
360                 $func .= "      movq    %rdx,%rsi\n" if ($narg>1);
361                 $func .= "      movq    %r8,%rdx\n"  if ($narg>2);
362                 $func .= "      movq    %r9,%rcx\n"  if ($narg>3);
363                 $func .= "      movq    40(%rsp),%r8\n" if ($narg>4);
364                 $func .= "      movq    48(%rsp),%r9\n" if ($narg>5);
365             }
366             $func;
367         } elsif ($self->{value} ne "$current_function->{name}") {
368             $self->{value} .= ":" if ($masm && $ret!~m/^\$/);
369             $self->{value} . ":";
370         } elsif ($win64 && $current_function->{abi} eq "svr4") {
371             my $func =  "$current_function->{name}" .
372                         ($nasm ? ":" : "\tPROC $current_function->{scope}") .
373                         "\n";
374             $func .= "  mov     QWORD${PTR}[8+rsp],rdi\t;WIN64 prologue\n";
375             $func .= "  mov     QWORD${PTR}[16+rsp],rsi\n";
376             $func .= "  mov     rax,rsp\n";
377             $func .= "${decor}SEH_begin_$current_function->{name}:";
378             $func .= ":" if ($masm);
379             $func .= "\n";
380             my $narg = $current_function->{narg};
381             $narg=6 if (!defined($narg));
382             $func .= "  mov     rdi,rcx\n" if ($narg>0);
383             $func .= "  mov     rsi,rdx\n" if ($narg>1);
384             $func .= "  mov     rdx,r8\n"  if ($narg>2);
385             $func .= "  mov     rcx,r9\n"  if ($narg>3);
386             $func .= "  mov     r8,QWORD${PTR}[40+rsp]\n" if ($narg>4);
387             $func .= "  mov     r9,QWORD${PTR}[48+rsp]\n" if ($narg>5);
388             $func .= "\n";
389         } else {
```

```
390             "$current_function->{name}".
391                         ($nasm ? ":" : "\tPROC $current_function->{scope}");
392         }
393     }
394 }
395 { package expr;         # pick up expressioins
396     sub re {
397         my      $self = shift;  # single instance is enough...
398         local   *line = shift;
399         undef   $ret;
400
401         if ($line =~ /(^[^,]+)/) {
402             $self->{value} = $1;
403             $ret = $self;
404             $line = substr($line,@+[0]); $line =~ s/^\s+//;
405
406             $self->{value} =~ s/\@PLT// if (!$self);
407             $self->{value} =~ s/([_a-z][_a-z0-9]*)/$globals{$1} or $1/gei;
408             $self->{value} =~ s/\.L/$decor/g;
409         }
410         $ret;
411     }
412     sub out {
413         my $self = shift;
414         if ($nasm && opcode->mnemonic()=~m/^j/) {
415             "NEAR ".$self->{value};
416         } else {
417             $self->{value};
418         }
419     }
420 }
421 { package directive;    # pick up directives, which start with .
422     sub re {
423         my      $self = shift;  # single instance is enough...
424         local   *line = shift;
425         undef   $ret;
426         my      $dir;
427         my      %opcode =       # lea 2f-1f(%rip),%dst; 1: nop; 2:
428                 (       "%rax"=>0x01058d48,     "%rcx"=>0x010d8d48,
429                         "%rdx"=>0x01158d48,     "%rbx"=>0x011d8d48,
430                         "%rsp"=>0x01258d48,     "%rbp"=>0x012d8d48,
431                         "%rsi"=>0x01358d48,     "%rdi"=>0x013d8d48,
432                         "%r8" =>0x01058d4c,     "%r9" =>0x010d8d4c,
433                         "%r10"=>0x01158d4c,     "%r11"=>0x011d8d4c,
434                         "%r12"=>0x01258d4c,     "%r13"=>0x012d8d4c,
435                         "%r14"=>0x01358d4c,     "%r15"=>0x013d8d4c      );
436
437         if ($line =~ /^\s*(\.\w+)/) {
438             $dir = $1;
439             $ret = $self;
440             undef $self->{value};
441             $line = substr($line,@+[0]); $line =~ s/^\s+//;
442
443             SWITCH: for ($dir) {
444                 /\.picmeup/ && do { if ($line =~ /(%r[\w]+)/i) {
445                                         $dir="\t.long";
446                                         $line=sprintf "0x%x,0x90000000",$opcode{
447                                     }
448                                     last;
449                                   };
450                 /\.global|\.globl|\.extern/
451                             && do { $globals{$line} = $prefix . $line;
452                                     $line = $globals{$line} if ($prefix);
453                                     last;
454                                   };
455                 /\.type/    && do { ($sym,$type,$narg) = split(',',$line);
```



```
456                                     if ($type eq "\@function") {
457                                         undef $current_function;
458                                         $current_function->{name} = $sym;
459                                         $current_function->{abi}  = "svr4";
460                                         $current_function->{narg} = $narg;
461                                         $current_function->{scope} = defined($gl
462                                     } elsif ($type eq "\@abi-omnipotent") {
463                                         undef $current_function;
464                                         $current_function->{name} = $sym;
465                                         $current_function->{scope} = defined($gl
466                                     }
467                                     $line =~ s/\@abi\-omnipotent/\@function/;
468                                     $line =~ s/\@function.*/\@function/;
469                                     $line =~ s/$sym/$globals{$sym} or $sym/e;
470                                     last;
471                                   };
472                 /\.asciz/   && do { if ($line =~ /^"(.*)"$/) {
473                                         $dir  = ".byte";
474                                         $line = join(",",unpack("C*",$1),0);
475                                     }
476                                     last;
477                                   };
478                 /\.rva|\.long|\.quad/
479                             && do { $line =~ s/([_a-z][_a-z0-9]*)/$globals{$1} o
480                                     $line =~ s/\.L/$decor/g;
481                                     last;
482                                   };
483                 /\.size/    && do { $line =~ s/([_a-z][_a-z0-9]*)/$globals{$1} o
484                                     last;
485                                   };
486             }
487
488             if ($gas) {
489                 $self->{value} = $dir . "\t" . $line;
490
491                 if ($dir =~ /\.extern/) {
492                     $self->{value} = ""; # swallow extern
493                 } elsif (!$elf && $dir =~ /\.type/) {
494                     $self->{value} = "";
495                     $self->{value} = ".def\t" . ($globals{$1} or $1) . ";\t" .
496                                      (defined($globals{$1})?".scl 2;":".scl 3;") .
497                                      "\t.type 32;\t.endef"
498                                      if ($win64 && $line =~ /([^,]+),\@function/);
499                 } elsif (!$elf && $dir =~ /\.size/) {
500                     $self->{value} = "";
501                     if (defined($current_function)) {
502                         $self->{value} .= "${decor}SEH_end_$current_function->{n
503                                           if ($win64 && $current_function->{abi} eq "svr4"
504                         undef $current_function;
505                     }
506                 } elsif (!$elf && $dir =~ /\.align/) {
507                     $self->{value} = ".p2align\t" . (log($line)/log(2));
508                 } elsif ($dir eq ".section") {
509                     $current_segment=$line;
510                     if (!$elf && $current_segment eq ".init") {
511                         if      ($flavour eq "macosx")  { $self->{value} = ".mod
512                         elsif   ($flavour eq "mingw64") { $self->{value} = ".sec
513                     }
514                 } elsif ($dir =~ /\.(text|data)/) {
515                     $current_segment=".$1";
516                 } elsif ($dir =~ /\.hidden/) {
517                     if    ($flavour eq "macosx")  { $self->{value} = ".private_e
518                     elsif ($flavour eq "mingw64") { $self->{value} = ""; }
519                     else { $self->{value} = ".hidden\t$prefix$line"; }
520                 } elsif ($dir =~ /\.comm/) {
521                     $self->{value} = "$dir\t$prefix$line";
```

```
522                     $self->{value} =~ s|,([0-9]+),([0-9]+)$|",$1,".log($2)/log(2
523                 }
524                 $line = "";
525                 return $self;
526             }
527
528             # non-gas case or nasm/masm
529             SWITCH: for ($dir) {
530                 /\.text/    && do { my $v=undef;
531                                     if ($nasm) {
532                                         $v="section     .text code align=64\n";
533                                     } else {
534                                         $v="$current_segment\tENDS\n" if ($curre
535                                         $current_segment = ".text\$";
536                                         $v.="$current_segment\tSEGMENT ";
537                                         $v.=$masm>=$masmref ? "ALIGN(64)" : "PAG
538                                         $v.=" 'CODE'";
539                                     }
540                                     $self->{value} = $v;
541                                     last;
542                                   };
543                 /\.data/    && do { my $v=undef;
544                                     if ($nasm) {
545                                         $v="section     .data data align=8\n";
546                                     } else {
547                                         $v="$current_segment\tENDS\n" if ($curre
548                                         $current_segment = "_DATA";
549                                         $v.="$current_segment\tSEGMENT";
550                                     }
551                                     $self->{value} = $v;
552                                     last;
553                                   };
554                 /\.section/ && do { my $v=undef;
555                                     $line =~ s/([^,]*).*/$1/;
556                                     $line = ".CRT\$XCU" if ($line eq ".init");
557                                     if ($nasm) {
558                                         $v="section     $line";
559                                         if ($line=~/\.([px])data/) {
560                                             $v.=" rdata align=";
561                                             $v.=$1 eq "p"? 4 : 8;
562                                         } elsif ($line=~/\.CRT\$/i) {
563                                             $v.=" rdata align=8";
564                                         }
565                                     } else {
566                                         $v="$current_segment\tENDS\n" if ($curre
567                                         $v.="$line\tSEGMENT";
568                                         if ($line=~/\.([px])data/) {
569                                             $v.=" READONLY";
570                                             $v.=" ALIGN(".($1 eq "p" ? 4 : 8).")
571                                         } elsif ($line=~/\.CRT\$/i) {
572                                             $v.=" READONLY ";
573                                             $v.=$masm>=$masmref ? "ALIGN(8)" : "
574                                         }
575                                     }
576                                     $current_segment = $line;
577                                     $self->{value} = $v;
578                                     last;
579                                   };
580                 /\.extern/  && do { $self->{value}  = "EXTERN\t".$line;
581                                     $self->{value} .= ":NEAR" if ($masm);
582                                     last;
583                                   };
584                 /\.globl|.global/
585                             && do { $self->{value}  = $masm?"PUBLIC":"global";
586                                     $self->{value} .= "\t".$line;
587                                     last;
```



```
588                                   };
589                 /\.size/    && do { if (defined($current_function)) {
590                                         undef $self->{value};
591                                         if ($current_function->{abi} eq "svr4")
592                                             $self->{value}="${decor}SEH_end_$cur
593                                             $self->{value}.=":\n" if($masm);
594                                         }
595                                         $self->{value}.="$current_function->{nam
596                                         undef $current_function;
597                                     }
598                                     last;
599                                   };
600                 /\.align/   && do { $self->{value} = "ALIGN\t".$line; last; };
601                 /\.(value|long|rva|quad)/
602                             && do { my $sz  = substr($1,0,1);
603                                     my @arr = split(/,\s*/,$line);
604                                     my $last = pop(@arr);
605                                     my $conv = sub  {   my $var=shift;
606                                                         $var=~s/^(0b[0-1]+)/oct(
607                                                         $var=~s/^0x([0-9a-f]+)/0
608                                                         if ($sz eq "D" && ($curr
609                                                         { $var=~s/([_a-z\$\@][_a
610                                                         $var;
611                                                     };
612
613                                     $sz =~ tr/bvlrq/BWDDQ/;
614                                     $self->{value} = "\tD$sz\t";
615                                     for (@arr) { $self->{value} .= &$conv($_).",
616                                     $self->{value} .= &$conv($last);
617                                     last;
618                                   };
619                 /\.byte/    && do { my @str=split(/,\s*/,$line);
620                                     map(s/(0b[0-1]+)/oct($1)/eig,@str);
621                                     map(s/0x([0-9a-f]+)/0$1h/ig,@str) if ($masm)
622                                     while ($#str>15) {
623                                         $self->{value}.="DB\t"
624                                                 .join(",",@str[0..15])."\n";
625                                         foreach (0..15) { shift @str; }
626                                     }
627                                     $self->{value}.="DB\t"
628                                                 .join(",",@str) if (@str);
629                                     last;
630                                   };
631                 /\.comm/    && do { my @str=split(/,\s*/,$line);
632                                     my $v=undef;
633                                     if ($nasm) {
634                                         $v.="common     $prefix@str[0] @str[1]";
635                                     } else {
636                                         $v="$current_segment\tENDS\n" if ($curre
637                                         $current_segment = "_DATA";
638                                         $v.="$current_segment\tSEGMENT\n";
639                                         $v.="COMM       @str[0]:DWORD:".@str[1]/
640                                     }
641                                     $self->{value} = $v;
642                                     last;
643                                   };
644             }
645             $line = "";
646         }
647
648         $ret;
649     }
650     sub out {
651         my $self = shift;
652         $self->{value};
653     }
```

```
654 }
656 sub rex {
657  local *opcode=shift;
658  my ($dst,$src,$rex)=@_;
660    $rex|=0x04 if($dst>=8);
661    $rex|=0x01 if($src>=8);
662    push @opcode,($rex|0x40) if ($rex);
663 }
665 # older gas and ml64 don't handle SSE>2 instructions
666 my %regrm = (   "%eax"=>0, "%ecx"=>1, "%edx"=>2, "%ebx"=>3,
667               "%esp"=>4, "%ebp"=>5, "%esi"=>6, "%edi"=>7      );
669 my $movq = sub {        # elderly gas can't handle inter-register movq
670   my $arg = shift;
671   my @opcode=(0x66);
672     if ($arg =~ /%xmm([0-9]+),\s*%r(\w+)/) {
673         my ($src,$dst)=($1,$2);
674         if ($dst !~ /[0-9]+/)   { $dst = $regrm{"%e$dst"}; }
675         rex(\@opcode,$src,$dst,0x8);
676         push @opcode,0x0f,0x7e;
677         push @opcode,0xc0|(($src&7)<<3)|($dst&7);       # ModR/M
678         @opcode;
679     } elsif ($arg =~ /%r(\w+),\s*%xmm([0-9]+)/) {
680         my ($src,$dst)=($2,$1);
681         if ($dst !~ /[0-9]+/)   { $dst = $regrm{"%e$dst"}; }
682         rex(\@opcode,$src,$dst,0x8);
683         push @opcode,0x0f,0x6e;
684         push @opcode,0xc0|(($src&7)<<3)|($dst&7);       # ModR/M
685         @opcode;
686     } else {
687         ();
688     }
689 };
691 my $pextrd = sub {
692     if (shift =~ /\$([0-9]+),\s*%xmm([0-9]+),\s*(%\w+)/) {
693       my @opcode=(0x66);
694         $imm=$1;
695         $src=$2;
696         $dst=$3;
697         if ($dst =~ /%r([0-9]+)d/)      { $dst = $1; }
698         elsif ($dst =~ /%e/)            { $dst = $regrm{$dst}; }
699         rex(\@opcode,$src,$dst);
700         push @opcode,0x0f,0x3a,0x16;
701         push @opcode,0xc0|(($src&7)<<3)|($dst&7);       # ModR/M
702         push @opcode,$imm;
703         @opcode;
704     } else {
705         ();
706     }
707 };
709 my $pinsrd = sub {
710     if (shift =~ /\$([0-9]+),\s*(%\w+),\s*%xmm([0-9]+)/) {
711       my @opcode=(0x66);
712         $imm=$1;
713         $src=$2;
714         $dst=$3;
715         if ($src =~ /%r([0-9]+)/)       { $src = $1; }
716         elsif ($src =~ /%e/)            { $src = $regrm{$src}; }
717         rex(\@opcode,$dst,$src);
718         push @opcode,0x0f,0x3a,0x22;
719         push @opcode,0xc0|(($dst&7)<<3)|($src&7);       # ModR/M
```



```
720         push @opcode,$imm;
721         @opcode;
722     } else {
723         ();
724     }
725 };
727 my $pshufb = sub {
728     if (shift =~ /%xmm([0-9]+),\s*%xmm([0-9]+)/) {
729       my @opcode=(0x66);
730         rex(\@opcode,$2,$1);
731         push @opcode,0x0f,0x38,0x00;
732         push @opcode,0xc0|($1&7)|(($2&7)<<3);           # ModR/M
733         @opcode;
734     } else {
735         ();
736     }
737 };
739 my $palignr = sub {
740     if (shift =~ /\$([0-9]+),\s*%xmm([0-9]+),\s*%xmm([0-9]+)/) {
741       my @opcode=(0x66);
742         rex(\@opcode,$3,$2);
743         push @opcode,0x0f,0x3a,0x0f;
744         push @opcode,0xc0|($2&7)|(($3&7)<<3);           # ModR/M
745         push @opcode,$1;
746         @opcode;
747     } else {
748         ();
749     }
750 };
752 my $pclmulqdq = sub {
753     if (shift =~ /\$([x0-9a-f]+),\s*%xmm([0-9]+),\s*%xmm([0-9]+)/) {
754       my @opcode=(0x66);
755         rex(\@opcode,$3,$2);
756         push @opcode,0x0f,0x3a,0x44;
757         push @opcode,0xc0|($2&7)|(($3&7)<<3);           # ModR/M
758         my $c=$1;
759         push @opcode,$c=~/^0/?oct($c):$c;
760         @opcode;
761     } else {
762         ();
763     }
764 };
766 my $rdrand = sub {
767     if (shift =~ /%[er](\w+)/) {
768       my @opcode=();
769       my $dst=$1;
770         if ($dst !~ /[0-9]+/) { $dst = $regrm{"%e$dst"}; }
771         rex(\@opcode,0,$1,8);
772         push @opcode,0x0f,0xc7,0xf0|($dst&7);
773         @opcode;
774     } else {
775         ();
776     }
777 };
779 if ($nasm) {
780     print <<___;
781 default rel
782 %define XMMWORD
783 ___
784 } elsif ($masm) {
785     print <<___;
```

```
786 OPTION  DOTNAME
787 ___
788 }
789 while($line=<>) {

791     chomp($line);

793     $line =~ s|[#!].*$||;       # get rid of asm-style comments...
794     $line =~ s|/\*.*\*/||;      # ... and C-style comments...
795     $line =~ s|^\s+||;          # ... and skip white spaces in beginning

797     undef $label;
798     undef $opcode;
799     undef @args;

801     if ($label=label->re(\$line))       { print $label->out(); }

803     if (directive->re(\$line)) {
804         printf "%s",directive->out();
805     } elsif ($opcode=opcode->re(\$line)) {
806         my $asm = eval("\$".$opcode->mnemonic());
807         undef @bytes;

809         if ((ref($asm) eq 'CODE') && scalar(@bytes=&$asm($line))) {
810             print $gas?".byte\t":"DB\t",join(',',@bytes),"\n";
811             next;
812         }

814         ARGUMENT: while (1) {
815         my $arg;

817         if ($arg=register->re(\$line))  { opcode->size($arg->size()); }
818         elsif ($arg=const->re(\$line))  { }
819         elsif ($arg=ea->re(\$line))     { }
820         elsif ($arg=expr->re(\$line))   { }
821         else                            { last ARGUMENT; }

823         push @args,$arg;

825         last ARGUMENT if ($line !~ /^,/);

827         $line =~ s/^,\s*//;
828         } # ARGUMENT:

830         if ($#args>=0) {
831             my $insn;
832             my $sz=opcode->size();

834             if ($gas) {
835                 $insn = $opcode->out($#args>=1?$args[$#args]->size():$sz);
836                 @args = map($_->out($sz),@args);
837                 printf "\t%s\t%s",$insn,join(",",@args);
838             } else {
839                 $insn = $opcode->out();
840                 foreach (@args) {
841                     my $arg = $_->out();
842                     # $insn.=$sz compensates for movq, pinsrw, ...
843                     if ($arg =~ /^xmm[0-9]+$/) { $insn.=$sz; $sz="x" if(!$sz); l
844                     if ($arg =~ /^mm[0-9]+$/)  { $insn.=$sz; $sz="q" if(!$sz); l
845                 }
846                 @args = reverse(@args);
847                 undef $sz if ($nasm && $opcode->mnemonic() eq "lea");
848                 printf "\t%s\t%s",$insn,join(",",map($_->out($sz),@args));
849             }
850         } else {
851             printf "\t%s",$opcode->out();
```



```
852         }
853     }

855     print $line,"\n";
856 }

858 print "\n$current_segment\tENDS\n"      if ($current_segment && $masm);
859 print "END\n"                           if ($masm);

861 close STDOUT;
```

```
863 ################################################
864 # Cross-reference x86_64 ABI "card"
865 #
866 #               Unix            Win64
867 # %rax          *               *
868 # %rbx          -               -
869 # %rcx          #4              #1
870 # %rdx          #3              #2
871 # %rsi          #2              -
872 # %rdi          #1              -
873 # %rbp          -               -
874 # %rsp          -               -
875 # %r8           #5              #3
876 # %r9           #6              #4
877 # %r10          *               *
878 # %r11          *               *
879 # %r12          -               -
880 # %r13          -               -
881 # %r14          -               -
882 # %r15          -               -
883 #
884 # (*)   volatile register
885 # (-)   preserved by callee
886 # (#)   Nth argument, volatile
887 #
888 # In Unix terms top of stack is argument transfer area for arguments
889 # which could not be accomodated in registers. Or in other words 7th
890 # [integer] argument resides at 8(%rsp) upon function entry point.
891 # 128 bytes above %rsp constitute a "red zone" which is not touched
892 # by signal handlers and can be used as temporal storage without
893 # allocating a frame.
894 #
895 # In Win64 terms N*8 bytes on top of stack is argument transfer area,
896 # which belongs to/can be overwritten by callee. N is the number of
897 # arguments passed to callee, *but* not less than 4! This means that
898 # upon function entry point 5th argument resides at 40(%rsp), as well
899 # as that 32 bytes from 8(%rsp) can always be used as temporal
900 # storage [without allocating a frame]. One can actually argue that
901 # one can assume a "red zone" above stack pointer under Win64 as well.
902 # Point is that at apparently no occasion Windows kernel would alter
903 # the area above user stack pointer in true asynchronous manner...
904 #
905 # All the above means that if assembler programmer adheres to Unix
906 # register and stack layout, but disregards the "red zone" existense,
907 # it's possible to use following prologue and epilogue to "gear" from
908 # Unix to Win64 ABI in leaf functions with not more than 6 arguments.
909 #
910 # omnipotent_function:
911 # ifdef WIN64
912 #       movq    %rdi,8(%rsp)
913 #       movq    %rsi,16(%rsp)
914 #       movq    %rcx,%rdi       ; if 1st argument is actually present
915 #       movq    %rdx,%rsi       ; if 2nd argument is actually ...
916 #       movq    %r8,%rdx        ; if 3rd argument is ...
917 #       movq    %r9,%rcx        ; if 4th argument ...
918 #       movq    40(%rsp),%r8    ; if 5th ...
919 #       movq    48(%rsp),%r9    ; if 6th ...
920 # endif
921 #       ...
922 # ifdef WIN64
923 #       movq    8(%rsp),%rdi
924 #       movq    16(%rsp),%rsi
925 # endif
926 #       ret
927 #
```



```
928 ################################################
929 # Win64 SEH, Structured Exception Handling.
930 #
931 # Unlike on Unix systems(*) lack of Win64 stack unwinding information
932 # has undesired side-effect at run-time: if an exception is raised in
933 # assembler subroutine such as those in question (basically we're
934 # referring to segmentation violations caused by malformed input
935 # parameters), the application is briskly terminated without invoking
936 # any exception handlers, most notably without generating memory dump
937 # or any user notification whatsoever. This poses a problem. It's
938 # possible to address it by registering custom language-specific
939 # handler that would restore processor context to the state at
940 # subroutine entry point and return "exception is not handled, keep
941 # unwinding" code. Writing such handler can be a challenge... But it's
942 # doable, though requires certain coding convention. Consider following
943 # snippet:
944 #
945 # .type function,@function
946 # function:
947 #       movq    %rsp,%rax       # copy rsp to volatile register
948 #       pushq   %r15            # save non-volatile registers
949 #       pushq   %rbx
950 #       pushq   %rbp
951 #       movq    %rsp,%r11
952 #       subq    %rdi,%r11       # prepare [variable] stack frame
953 #       andq    $-64,%r11
954 #       movq    %rax,0(%r11)    # check for exceptions
955 #       movq    %r11,%rsp       # allocate [variable] stack frame
956 #       movq    %rax,0(%rsp)    # save original rsp value
957 # magic_point:
958 #       ...
959 #       movq    0(%rsp),%rcx    # pull original rsp value
960 #       movq    -24(%rcx),%rbp  # restore non-volatile registers
961 #       movq    -16(%rcx),%rbx
962 #       movq    -8(%rcx),%r15
963 #       movq    %rcx,%rsp       # restore original rsp
964 #       ret
965 # .size function,.-function
966 #
967 # The key is that up to magic_point copy of original rsp value remains
968 # in chosen volatile register and no non-volatile register, except for
969 # rsp, is modified. While past magic_point rsp remains constant till
970 # the very end of the function. In this case custom language-specific
971 # exception handler would look like this:
972 #
973 # EXCEPTION_DISPOSITION handler (EXCEPTION_RECORD *rec,ULONG64 frame,
974 #               CONTEXT *context,DISPATCHER_CONTEXT *disp)
975 # {     ULONG64 *rsp = (ULONG64 *)context->Rax;
976 #       if (context->Rip >= magic_point)
977 #       {   rsp = ((ULONG64 **)context->Rsp)[0];
978 #           context->Rbp = rsp[-3];
979 #           context->Rbx = rsp[-2];
980 #           context->R15 = rsp[-1];
981 #       }
982 #       context->Rsp = (ULONG64)rsp;
983 #       context->Rdi = rsp[1];
984 #       context->Rsi = rsp[2];
985 #
986 #       memcpy (disp->ContextRecord,context,sizeof(CONTEXT));
987 #       RtlVirtualUnwind(UNW_FLAG_NHANDLER,disp->ImageBase,
988 #               dips->ControlPc,disp->FunctionEntry,disp->ContextRecord,
989 #               &disp->HandlerData,&disp->EstablisherFrame,NULL);
990 #       return ExceptionContinueSearch;
991 # }
992 #
993 # It's appropriate to implement this handler in assembler, directly in
```

```
 994 # function's module. In order to do that one has to know members'
 995 # offsets in CONTEXT and DISPATCHER_CONTEXT structures and some constant
 996 # values. Here they are:
 997 #
 998 #       CONTEXT.Rax                             120
 999 #       CONTEXT.Rcx                             128
1000 #       CONTEXT.Rdx                             136
1001 #       CONTEXT.Rbx                             144
1002 #       CONTEXT.Rsp                             152
1003 #       CONTEXT.Rbp                             160
1004 #       CONTEXT.Rsi                             168
1005 #       CONTEXT.Rdi                             176
1006 #       CONTEXT.R8                              184
1007 #       CONTEXT.R9                              192
1008 #       CONTEXT.R10                             200
1009 #       CONTEXT.R11                             208
1010 #       CONTEXT.R12                             216
1011 #       CONTEXT.R13                             224
1012 #       CONTEXT.R14                             232
1013 #       CONTEXT.R15                             240
1014 #       CONTEXT.Rip                             248
1015 #       CONTEXT.Xmm6                            512
1016 #       sizeof(CONTEXT)                         1232
1017 #       DISPATCHER_CONTEXT.ControlPc            0
1018 #       DISPATCHER_CONTEXT.ImageBase            8
1019 #       DISPATCHER_CONTEXT.FunctionEntry        16
1020 #       DISPATCHER_CONTEXT.EstablisherFrame     24
1021 #       DISPATCHER_CONTEXT.TargetIp             32
1022 #       DISPATCHER_CONTEXT.ContextRecord        40
1023 #       DISPATCHER_CONTEXT.LanguageHandler      48
1024 #       DISPATCHER_CONTEXT.HandlerData          56
1025 #       UNW_FLAG_NHANDLER                       0
1026 #       ExceptionContinueSearch                 1
1027 #
1028 # In order to tie the handler to the function one has to compose
1029 # couple of structures: one for .xdata segment and one for .pdata.
1030 #
1031 # UNWIND_INFO structure for .xdata segment would be
1032 #
1033 # function_unwind_info:
1034 #       .byte   9,0,0,0
1035 #       .rva    handler
1036 #
1037 # This structure designates exception handler for a function with
1038 # zero-length prologue, no stack frame or frame register.
1039 #
1040 # To facilitate composing of .pdata structures, auto-generated "gear"
1041 # prologue copies rsp value to rax and denotes next instruction with
1042 # .LSEH_begin_{function_name} label. This essentially defines the SEH
1043 # styling rule mentioned in the beginning. Position of this label is
1044 # chosen in such manner that possible exceptions raised in the "gear"
1045 # prologue would be accounted to caller and unwound from latter's frame.
1046 # End of function is marked with respective .LSEH_end_{function_name}
1047 # label. To summarize, .pdata segment would contain
1048 #
1049 #       .rva    .LSEH_begin_function
1050 #       .rva    .LSEH_end_function
1051 #       .rva    function_unwind_info
1052 #
1053 # Reference to functon_unwind_info from .xdata segment is the anchor.
1054 # In case you wonder why references are 32-bit .rvas and not 64-bit
1055 # .quads. References put into these two segments are required to be
1056 # *relative* to the base address of the current binary module, a.k.a.
1057 # image base. No Win64 module, be it .exe or .dll, can be larger than
1058 # 2GB and thus such relative references can be and are accommodated in
1059 # 32 bits.
```



```
1060 #
1061 # Having reviewed the example function code, one can argue that "movq
1062 # %rsp,%rax" above is redundant. It is not! Keep in mind that on Unix
1063 # rax would contain an undefined value. If this "offends" you, use
1064 # another register and refrain from modifying rax till magic_point is
1065 # reached, i.e. as if it was a non-volatile register. If more registers
1066 # are required prior [variable] frame setup is completed, note that
1067 # nobody says that you can have only one "magic point." You can
1068 # "liberate" non-volatile registers by denoting last stack off-load
1069 # instruction and reflecting it in finer grade unwind logic in handler.
1070 # After all, isn't it why it's called *language-specific* handler...
1071 #
1072 # Attentive reader can notice that exceptions would be mishandled in
1073 # auto-generated "gear" epilogue. Well, exception effectively can't
1074 # occur there, because if memory area used by it was subject to
1075 # segmentation violation, then it would be raised upon call to the
1076 # function (and as already mentioned be accounted to caller, which is
1077 # not a problem). If you're still not comfortable, then define tail
1078 # "magic point" just prior ret instruction and have handler treat it...
1079 #
1080 # (*)   Note that we're talking about run-time, not debug-time. Lack of
1081 #       unwind information makes debugging hard on both Windows and
1082 #       Unix. "Unlike" referes to the fact that on Unix signal handler
1083 #       will always be invoked, core dumped and appropriate exit code
1084 #       returned to parent (for user notification).
1085 #endif /* ! codereview */
```

```
**********************************************************
    5680 Wed Aug 13 19:53:11 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86_64cpuid.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

$flavour = shift;
$output  = shift;
if ($flavour =~ /\./) { $output = $flavour; undef $flavour; }

$win64=0; $win64=1 if ($flavour =~ /[nm]asm|mingw64/ || $output =~ /\.asm$/);

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
( $xlate="${dir}x86_64-xlate.pl" and -f $xlate ) or
( $xlate="${dir}perlasm/x86_64-xlate.pl" and -f $xlate) or
die "can't locate x86_64-xlate.pl";

open OUT,"| \"$^X\" $xlate $flavour $output";
*STDOUT=*OUT;

($arg1,$arg2,$arg3,$arg4)=$win64?("%rcx","%rdx","%r8", "%r9") : # Win64 order
                               ("%rdi","%rsi","%rdx","%rcx"); # Unix order

print<<___;
.extern         OPENSSL_cpuid_setup
.hidden         OPENSSL_cpuid_setup
.extern         illumos_locking_setup
.hidden         illumos_locking_setup
.section        .init
        call    illumos_locking_setup
        call    OPENSSL_cpuid_setup

.hidden OPENSSL_ia32cap_P
.comm   OPENSSL_ia32cap_P,8,4

.text

.globl  OPENSSL_atomic_add
.type   OPENSSL_atomic_add,\@abi-omnipotent
.align  16
OPENSSL_atomic_add:
        movl    ($arg1),%eax
.Lspin: leaq    ($arg2,%rax),%r8
        .byte   0xf0            # lock
        cmpxchgl        %r8d,($arg1)
        jne     .Lspin
        movl    %r8d,%eax
        .byte   0x48,0x98       # cltq/cdqe
        ret
.size   OPENSSL_atomic_add,.-OPENSSL_atomic_add

.globl  OPENSSL_rdtsc
.type   OPENSSL_rdtsc,\@abi-omnipotent
.align  16
OPENSSL_rdtsc:
        rdtsc
        shl     \$32,%rdx
        or      %rdx,%rax
        ret
.size   OPENSSL_rdtsc,.-OPENSSL_rdtsc

.globl  OPENSSL_ia32_cpuid
.type   OPENSSL_ia32_cpuid,\@abi-omnipotent
.align  16
OPENSSL_ia32_cpuid:
```



```
        mov     %rbx,%r8                # save %rbx

        xor     %eax,%eax
        cpuid
        mov     %eax,%r11d              # max value for standard query level

        xor     %eax,%eax
        cmp     \$0x756e6547,%ebx       # "Genu"
        setne   %al
        mov     %eax,%r9d
        cmp     \$0x49656e69,%edx       # "ineI"
        setne   %al
        or      %eax,%r9d
        cmp     \$0x6c65746e,%ecx       # "ntel"
        setne   %al
        or      %eax,%r9d               # 0 indicates Intel CPU
        jz      .Lintel

        cmp     \$0x68747541,%ebx       # "Auth"
        setne   %al
        mov     %eax,%r10d
        cmp     \$0x69746E65,%edx       # "enti"
        setne   %al
        or      %eax,%r10d
        cmp     \$0x444D4163,%ecx       # "cAMD"
        setne   %al
        or      %eax,%r10d              # 0 indicates AMD CPU
        jnz     .Lintel

        # AMD specific
        mov     \$0x80000000,%eax
        cpuid
        cmp     \$0x80000001,%eax
        jb      .Lintel
        mov     %eax,%r10d
        mov     \$0x80000001,%eax
        cpuid
        or      %ecx,%r9d
        and     \$0x00000801,%r9d       # isolate AMD XOP bit, 1<<11

        cmp     \$0x80000008,%r10d
        jb      .Lintel

        mov     \$0x80000008,%eax
        cpuid
        movzb   %cl,%r10                # number of cores - 1
        inc     %r10                    # number of cores

        mov     \$1,%eax
        cpuid
        bt      \$28,%edx               # test hyper-threading bit
        jnc     .Lgeneric
        shr     \$16,%ebx               # number of logical processors
        cmp     %r10b,%bl
        ja      .Lgeneric
        and     \$0xefffffff,%edx       # ~(1<<28)
        jmp     .Lgeneric

.Lintel:
        cmp     \$4,%r11d
        mov     \$-1,%r10d
        jb      .Lnocacheinfo

        mov     \$4,%eax
        mov     \$0,%ecx                # query L1D
        cpuid
```

```
128         mov     %eax,%r10d
129         shr     \$14,%r10d
130         and     \$0xfff,%r10d           # number of cores -1 per L1D

132 .Lnocacheinfo:
133         mov     \$1,%eax
134         cpuid
135         and     \$0xbfefffff,%edx       # force reserved bits to 0
136         cmp     \$0,%r9d
137         jne     .Lnotintel
138         or      \$0x40000000,%edx       # set reserved bit#30 on Intel CPUs
139         and     \$15,%ah
140         cmp     \$15,%ah                # examine Family ID
141         jne     .Lnotintel
142         or      \$0x00100000,%edx       # set reserved bit#20 to engage RC4_CHAR
143 .Lnotintel:
144         bt      \$28,%edx               # test hyper-threading bit
145         jnc     .Lgeneric
146         and     \$0xefffffff,%edx       # ~(1<<28)
147         cmp     \$0,%r10d
148         je      .Lgeneric

150         or      \$0x10000000,%edx       # 1<<28
151         shr     \$16,%ebx
152         cmp     \$1,%bl                 # see if cache is shared
153         ja      .Lgeneric
154         and     \$0xefffffff,%edx       # ~(1<<28)
155 .Lgeneric:
156         and     \$0x00000800,%r9d       # isolate AMD XOP flag
157         and     \$0xfffff7ff,%ecx
158         or      %ecx,%r9d               # merge AMD XOP flag

160         mov     %edx,%r10d              # %r9d:%r10d is copy of %ecx:%edx
161         bt      \$27,%r9d               # check OSXSAVE bit
162         jnc     .Lclear_avx
163         xor     %ecx,%ecx               # XCR0
164         .byte   0x0f,0x01,0xd0          # xgetbv
165         and     \$6,%eax                # isolate XMM and YMM state support
166         cmp     \$6,%eax
167         je      .Ldone
168 .Lclear_avx:
169         mov     \$0xefffe7ff,%eax       # ~(1<<28|1<<12|1<<11)
170         and     %eax,%r9d               # clear AVX, FMA and AMD XOP bits
171 .Ldone:
172         shl     \$32,%r9
173         mov     %r10d,%eax
174         mov     %r8,%rbx                # restore %rbx
175         or      %r9,%rax
176         ret
177 .size   OPENSSL_ia32_cpuid,.-OPENSSL_ia32_cpuid

179 .globl  OPENSSL_cleanse
180 .type   OPENSSL_cleanse,\@abi-omnipotent
181 .align  16
182 OPENSSL_cleanse:
183         xor     %rax,%rax
184         cmp     \$15,$arg2
185         jae     .Lot
186         cmp     \$0,$arg2
187         je      .Lret
188 .Little:
189         mov     %al,($arg1)
190         sub     \$1,$arg2
191         lea     1($arg1),$arg1
192         jnz     .Little
193 .Lret:
```



```
194         ret
195 .align  16
196 .Lot:
197         test    \$7,$arg1
198         jz      .Laligned
199         mov     %al,($arg1)
200         lea     -1($arg2),$arg2
201         lea     1($arg1),$arg1
202         jmp     .Lot
203 .Laligned:
204         mov     %rax,($arg1)
205         lea     -8($arg2),$arg2
206         test    \$-8,$arg2
207         lea     8($arg1),$arg1
208         jnz     .Laligned
209         cmp     \$0,$arg2
210         jne     .Little
211         ret
212 .size   OPENSSL_cleanse,.-OPENSSL_cleanse
213 ___

215 print<<___ if (!$win64);
216 .globl  OPENSSL_wipe_cpu
217 .type   OPENSSL_wipe_cpu,\@abi-omnipotent
218 .align  16
219 OPENSSL_wipe_cpu:
220         pxor    %xmm0,%xmm0
221         pxor    %xmm1,%xmm1
222         pxor    %xmm2,%xmm2
223         pxor    %xmm3,%xmm3
224         pxor    %xmm4,%xmm4
225         pxor    %xmm5,%xmm5
226         pxor    %xmm6,%xmm6
227         pxor    %xmm7,%xmm7
228         pxor    %xmm8,%xmm8
229         pxor    %xmm9,%xmm9
230         pxor    %xmm10,%xmm10
231         pxor    %xmm11,%xmm11
232         pxor    %xmm12,%xmm12
233         pxor    %xmm13,%xmm13
234         pxor    %xmm14,%xmm14
235         pxor    %xmm15,%xmm15
236         xorq    %rcx,%rcx
237         xorq    %rdx,%rdx
238         xorq    %rsi,%rsi
239         xorq    %rdi,%rdi
240         xorq    %r8,%r8
241         xorq    %r9,%r9
242         xorq    %r10,%r10
243         xorq    %r11,%r11
244         leaq    8(%rsp),%rax
245         ret
246 .size   OPENSSL_wipe_cpu,.-OPENSSL_wipe_cpu
247 ___
248 print<<___ if ($win64);
249 .globl  OPENSSL_wipe_cpu
250 .type   OPENSSL_wipe_cpu,\@abi-omnipotent
251 .align  16
252 OPENSSL_wipe_cpu:
253         pxor    %xmm0,%xmm0
254         pxor    %xmm1,%xmm1
255         pxor    %xmm2,%xmm2
256         pxor    %xmm3,%xmm3
257         pxor    %xmm4,%xmm4
258         pxor    %xmm5,%xmm5
259         xorq    %rcx,%rcx
```

```
260         xorq    %rdx,%rdx
261         xorq    %r8,%r8
262         xorq    %r9,%r9
263         xorq    %r10,%r10
264         xorq    %r11,%r11
265         leaq    8(%rsp),%rax
266         ret
267 .size   OPENSSL_wipe_cpu,.-OPENSSL_wipe_cpu
268 ___

270 print<<___;
271 .globl  OPENSSL_ia32_rdrand
272 .type   OPENSSL_ia32_rdrand,\@abi-omnipotent
273 .align  16
274 OPENSSL_ia32_rdrand:
275         mov     \$8,%ecx
276 .Loop_rdrand:
277         rdrand  %rax
278         jc      .Lbreak_rdrand
279         loop    .Loop_rdrand
280 .Lbreak_rdrand:
281         cmp     \$0,%rax
282         cmove   %rcx,%rax
283         ret
284 .size   OPENSSL_ia32_rdrand,.-OPENSSL_ia32_rdrand
285 ___

287 close STDOUT;   # flush
288 #endif /* ! codereview */
```

```
**********************************************************
    6344 Wed Aug 13 19:53:11 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86asm.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

# require 'x86asm.pl';
# &asm_init(<flavor>,"des-586.pl"[,$i386only]);
# &function_begin("foo");
# ...
# &function_end("foo");
# &asm_finish

$out=();
$i386=0;

# AUTOLOAD is this context has quite unpleasant side effect, namely
# that typos in function calls effectively go to assembler output,
# but on the pros side we don't have to implement one subroutine per
# each opcode...
sub ::AUTOLOAD
{ my $opcode = $AUTOLOAD;

    die "more than 4 arguments passed to $opcode" if ($#_>3);

    $opcode =~ s/.*::///;
    if    ($opcode =~ /^push/) { $stack+=4; }
    elsif ($opcode =~ /^pop/)  { $stack-=4; }

    &generic($opcode,@_) or die "undefined subroutine \&$AUTOLOAD";
}

sub ::emit
{ my $opcode=shift;

    if ($#_==-1)    { push(@out,"\t$opcode\n");                          }
    else            { push(@out,"\t$opcode\t".join(',',@_)."\n");       }
}

sub ::LB
{   $_[0] =~ m/^e?([a-d])x$/o or die "$_[0] does not have a 'low byte'";
  $1."l";
}
sub ::HB
{   $_[0] =~ m/^e?([a-d])x$/o or die "$_[0] does not have a 'high byte'";
  $1."h";
}
sub ::stack_push{ my $num=$_[0]*4; $stack+=$num; &sub("esp",$num);      }
sub ::stack_pop { my $num=$_[0]*4; $stack-=$num; &add("esp",$num);      }
sub ::blindpop  { &pop($_[0]); $stack+=4;                               }
sub ::wparam    { &DWP($stack+4*$_[0],"esp");                           }
sub ::swtmp     { &DWP(4*$_[0],"esp");                                  }

sub ::bswap
{   if ($i386)  # emulate bswap for i386
    {   &comment("bswap @_");
        &xchg(&HB(@_),&LB(@_));
        &ror (@_,16);
        &xchg(&HB(@_),&LB(@_));
    }
    else
    {   &generic("bswap",@_);   }
}
# These are made-up opcodes introduced over the years essentially
# by ignorance, just alias them to real ones...
```



```
sub ::movb      { &mov(@_);     }
sub ::xorb      { &xor(@_);     }
sub ::rotl      { &rol(@_);     }
sub ::rotr      { &ror(@_);     }
sub ::exch      { &xchg(@_);    }
sub ::halt      { &hlt;         }
sub ::movz      { &movzx(@_);   }
sub ::pushf     { &pushfd;      }
sub ::popf      { &popfd;       }

# 3 argument instructions
sub ::movq
{ my($p1,$p2,$optimize)=@_;

    if ($optimize && $p1=~/^mm[0-7]$/ && $p2=~/^mm[0-7]$/)
    # movq between mmx registers can sink Intel CPUs
    {   &::pshufw($p1,$p2,0xe4);                }
    else
    {   &::generic("movq",@_);                  }
}

# SSE>2 instructions
my %regrm = (   "eax"=>0, "ecx"=>1, "edx"=>2, "ebx"=>3,
                "esp"=>4, "ebp"=>5, "esi"=>6, "edi"=>7  );
sub ::pextrd
{ my($dst,$src,$imm)=@_;
    if ("$dst:$src" =~ /(e[a-dsd][ixp]):xmm([0-7])/)
    {   &::data_byte(0x66,0x0f,0x3a,0x16,0xc0|($2<<3)|$regrm{$1},$imm); }
    else
    {   &::generic("pextrd",@_);                }
}

sub ::pinsrd
{ my($dst,$src,$imm)=@_;
    if ("$dst:$src" =~ /xmm([0-7]):(e[a-dsd][ixp])/)
    {   &::data_byte(0x66,0x0f,0x3a,0x22,0xc0|($1<<3)|$regrm{$2},$imm); }
    else
    {   &::generic("pinsrd",@_);                }
}

sub ::pshufb
{ my($dst,$src)=@_;
    if ("$dst:$src" =~ /xmm([0-7]):xmm([0-7])/)
    {   &data_byte(0x66,0x0f,0x38,0x00,0xc0|($1<<3)|$2);        }
    else
    {   &::generic("pshufb",@_);                }
}

sub ::palignr
{ my($dst,$src,$imm)=@_;
    if ("$dst:$src" =~ /xmm([0-7]):xmm([0-7])/)
    {   &::data_byte(0x66,0x0f,0x3a,0x0f,0xc0|($1<<3)|$2,$imm); }
    else
    {   &::generic("palignr",@_);               }
}

sub ::pclmulqdq
{ my($dst,$src,$imm)=@_;
    if ("$dst:$src" =~ /xmm([0-7]):xmm([0-7])/)
    {   &::data_byte(0x66,0x0f,0x3a,0x44,0xc0|($1<<3)|$2,$imm); }
    else
    {   &::generic("pclmulqdq",@_);             }
}

sub ::rdrand
{ my ($dst)=@_;
```

```
128     if ($dst =~ /(e[a-dsd][ixp])/)
129     {   &::data_byte(0x0f,0xc7,0xf0|$regrm{$dst});      }
130     else
131     {   &::generic("rdrand",@_);        }
132 }

133 # label management
134 $lbdecor="L";           # local label decoration, set by package
135 $label="000";

136 sub ::islabel           # see is argument is a known label
137 { my $i;
138     if ($_[0] eq "_GLOBAL_OFFSET_TABLE_") { return $_[0]; }
139     foreach $i (values %label) { return $i if ($i eq $_[0]); }
140   $label{$_[0]};        # can be undef
141 }

142 sub ::label             # instantiate a function-scope label
143 {   if (!defined($label{$_[0]}))
144     {   $label{$_[0]}="${lbdecor}${label}${_[0]}"; $label++;   }
145   $label{$_[0]};
146 }

147 sub ::LABEL             # instantiate a file-scope label
148 {   $label{$_[0]}=$_[1] if (!defined($label{$_[0]}));
149   $label{$_[0]};
150 }

151 sub ::static_label      { &::LABEL($_[0],$lbdecor.$_[0]); }

152 sub ::set_label_B       { push(@out,"@_:\n"); }
153 sub ::set_label
154 { my $label=&::label($_[0]);
155     &::align($_[1]) if ($_[1]>1);
156     &::set_label_B($label);
157   $label;
158 }

159 sub ::wipe_labels       # wipes function-scope labels
160 {   foreach $i (keys %label)
161     {   delete $label{$i} if ($label{$i} =~ /^\Q${lbdecor}\E[0-9]{3}/); }
162 }

163 # subroutine management
164 sub ::function_begin
165 {   &function_begin_B(@_);
166     $stack=4;
167     &push("ebp");
168     &push("ebx");
169     &push("esi");
170     &push("edi");
171 }

172 sub ::function_end
173 {   &pop("edi");
174     &pop("esi");
175     &pop("ebx");
176     &pop("ebp");
177     &ret();
178     &function_end_B(@_);
179     $stack=0;
180     &wipe_labels();
181 }

182 sub ::function_end_A
183 {   &pop("edi");
```



```
184     &pop("esi");
185     &pop("ebx");
186     &pop("ebp");
187     &ret();
188     $stack+=16; # readjust esp as if we didn't pop anything
189 }

190 sub ::asciz
191 { my @str=unpack("C*",shift);
192     push @str,0;
193     while ($#str>15) {
194         &data_byte(@str[0..15]);
195         foreach (0..15) { shift @str; }
196     }
197     &data_byte(@str) if (@str);
198 }

199 sub ::asm_finish
200 {   &file_end();
201     print @out;
202 }

203 sub ::asm_init
204 { my ($type,$fn,$cpu)=@_;

205     $filename=$fn;
206     $i386=$cpu;

207     $self=$cpp=$coff=$aout=$macosx=$win32=$netware=$mwerks=$android=0;
208     if    (($type eq "elf"))
209     {   $self=1;                    require "x86gas.pl";    }
210     elsif (($type eq "a\.out"))
211     {   $aout=1;                    require "x86gas.pl";    }
212     elsif (($type eq "coff" or $type eq "gaswin"))
213     {   $coff=1;                    require "x86gas.pl";    }
214     elsif (($type eq "win32n"))
215     {   $win32=1;                   require "x86nasm.pl";   }
216     elsif (($type eq "nw-nasm"))
217     {   $netware=1;                 require "x86nasm.pl";   }
218     #elsif (($type eq "nw-mwasm"))
219     #{  $netware=1; $mwerks=1;      require "x86nasm.pl";   }
220     elsif (($type eq "win32"))
221     {   $win32=1;                   require "x86masm.pl";   }
222     elsif (($type eq "macosx"))
223     {   $aout=1; $macosx=1;         require "x86gas.pl";    }
224     elsif (($type eq "android"))
225     {   $self=1; $android=1;        require "x86gas.pl";    }
226     else
227     {   print STDERR <<"EOF";
228 Pick one target type from
229         elf     - Linux, FreeBSD, Solaris x86, etc.
230         a.out   - DJGPP, elder OpenBSD, etc.
231         coff    - GAS/COFF such as Win32 targets
232         win32n  - Windows 95/Windows NT NASM format
233         nw-nasm - NetWare NASM format
234         macosx  - Mac OS X
235 EOF
236         exit(1);
237     }

238     $pic=0;
239     for (@ARGV) { $pic=1 if (/\-[fK]PIC/i); }

240     $filename =~ s/\.pl$//;
241     &file($filename);
242 }
```

```
261 sub ::hidden {}

263 1;
264 #endif /* ! codereview */
```

```
**********************************************************
    9112 Wed Aug 13 19:53:11 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86cpuid.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

$0 =~ m/(.*[\/\\])[^\/\\]+$/; $dir=$1;
push(@INC, "${dir}", "perlasm");
require "x86asm.pl";

&asm_init($ARGV[0],"x86cpuid");

for (@ARGV) { $sse2=1 if (/-DOPENSSL_IA32_SSE2/); }

&function_begin("OPENSSL_ia32_cpuid");
        &xor    ("edx","edx");
        &pushf  ();
        &pop    ("eax");
        &mov    ("ecx","eax");
        &xor    ("eax",1<<21);
        &push   ("eax");
        &popf   ();
        &pushf  ();
        &pop    ("eax");
        &xor    ("ecx","eax");
        &xor    ("eax","eax");
        &bt     ("ecx",21);
        &jnc    (&label("nocpuid"));
        &cpuid  ();
        &mov    ("edi","eax");          # max value for standard query level

        &xor    ("eax","eax");
        &cmp    ("ebx",0x756e6547);     # "Genu"
        &setne  (&LB("eax"));
        &mov    ("ebp","eax");
        &cmp    ("edx",0x49656e69);     # "ineI"
        &setne  (&LB("eax"));
        &or     ("ebp","eax");
        &cmp    ("ecx",0x6c65746e);     # "ntel"
        &setne  (&LB("eax"));
        &or     ("ebp","eax");          # 0 indicates Intel CPU
        &jz     (&label("intel"));

        &cmp    ("ebx",0x68747541);     # "Auth"
        &setne  (&LB("eax"));
        &mov    ("esi","eax");
        &cmp    ("edx",0x69746E65);     # "enti"
        &setne  (&LB("eax"));
        &or     ("esi","eax");
        &cmp    ("ecx",0x444D4163);     # "cAMD"
        &setne  (&LB("eax"));
        &or     ("esi","eax");          # 0 indicates AMD CPU
        &jnz    (&label("intel"));

        # AMD specific
        &mov    ("eax",0x80000000);
        &cpuid  ();
        &cmp    ("eax",0x80000001);
        &jb     (&label("intel"));
        &mov    ("esi","eax");
        &mov    ("eax",0x80000001);
        &cpuid  ();
        &or     ("ebp","ecx");
        &and    ("ebp",1<<11|1);        # isolate XOP bit
        &cmp    ("esi",0x80000008);
```



```
        &jb     (&label("intel"));

        &mov    ("eax",0x80000008);
        &cpuid  ();
        &movz   ("esi",&LB("ecx"));     # number of cores - 1
        &inc    ("esi");                # number of cores

        &mov    ("eax",1);
        &xor    ("ecx","ecx");
        &cpuid  ();
        &bt     ("edx",28);
        &jnc    (&label("generic"));
        &shr    ("ebx",16);
        &and    ("ebx",0xff);
        &cmp    ("ebx","esi");
        &ja     (&label("generic"));
        &and    ("edx",0xefffffff);     # clear hyper-threading bit
        &jmp    (&label("generic"));

&set_label("intel");
        &cmp    ("edi",4);
        &mov    ("edi",-1);
        &jb     (&label("nocacheinfo"));

        &mov    ("eax",4);
        &mov    ("ecx",0);              # query L1D
        &cpuid  ();
        &mov    ("edi","eax");
        &shr    ("edi",14);
        &and    ("edi",0xfff);          # number of cores -1 per L1D

&set_label("nocacheinfo");
        &mov    ("eax",1);
        &xor    ("ecx","ecx");
        &cpuid  ();
        &and    ("edx",0xbfefffff);     # force reserved bits #20, #30 to 0
        &cmp    ("ebp",0);
        &jne    (&label("notintel"));
        &or     ("edx",1<<30);          # set reserved bit#30 on Intel CPUs
        &and    (&HB("eax"),15);        # familiy ID
        &cmp    (&HB("eax"),15);        # P4?
        &jne    (&label("notintel"));
        &or     ("edx",1<<20);          # set reserved bit#20 to engage RC4_CHAR
&set_label("notintel");
        &bt     ("edx",28);             # test hyper-threading bit
        &jnc    (&label("generic"));
        &and    ("edx",0xefffffff);
        &cmp    ("edi",0);
        &je     (&label("generic"));

        &or     ("edx",0x10000000);
        &shr    ("ebx",16);
        &cmp    (&LB("ebx"),1);
        &ja     (&label("generic"));
        &and    ("edx",0xefffffff);     # clear hyper-threading bit if not

&set_label("generic");
        &and    ("ebp",1<<11);          # isolate AMD XOP flag
        &and    ("ecx",0xfffff7ff);     # force 11th bit to 0
        &mov    ("esi","edx");
        &or     ("ebp","ecx");          # merge AMD XOP flag

        &bt     ("ecx",27);             # check OSXSAVE bit
        &jnc    (&label("clear_avx"));
        &xor    ("ecx","ecx");
        &data_byte(0x0f,0x01,0xd0);     # xgetbv
```

```
128         &and    ("eax",6);
129         &cmp    ("eax",6);
130         &je     (&label("done"));
131         &cmp    ("eax",2);
132         &je     (&label("clear_avx"));
133 &set_label("clear_xmm");
134         &and    ("ebp",0xfdffffff);     # clear AESNI and PCLMULQDQ bits
135         &and    ("esi",0xfeffffff);     # clear FXSR
136 &set_label("clear_avx");
137         &and    ("ebp",0xefffe7ff);     # clear AVX, FMA and AMD XOP bits
138 &set_label("done");
139         &mov    ("eax","esi");
140         &mov    ("edx","ebp");
141 &set_label("nocpuid");
142 &function_end("OPENSSL_ia32_cpuid");

144 &external_label("OPENSSL_ia32cap_P");

146 &function_begin_B("OPENSSL_rdtsc","EXTRN\t_OPENSSL_ia32cap_P:DWORD");
147         &xor    ("eax","eax");
148         &xor    ("edx","edx");
149         &picmeup("ecx","OPENSSL_ia32cap_P");
150         &bt     (&DWP(0,"ecx"),4);
151         &jnc    (&label("notsc"));
152         &rdtsc  ();
153 &set_label("notsc");
154         &ret    ();
155 &function_end_B("OPENSSL_rdtsc");

157 # This works in Ring 0 only [read DJGPP+MS-DOS+privileged DPMI host],
158 # but it's safe to call it on any [supported] 32-bit platform...
159 # Just check for [non-]zero return value...
160 &function_begin_B("OPENSSL_instrument_halt","EXTRN\t_OPENSSL_ia32cap_P:DWORD");
161         &picmeup("ecx","OPENSSL_ia32cap_P");
162         &bt     (&DWP(0,"ecx"),4);
163         &jnc    (&label("nohalt"));     # no TSC

165         &data_word(0x9058900e);         # push %cs; pop %eax
166         &and    ("eax",3);
167         &jnz    (&label("nohalt"));     # not enough privileges

169         &pushf  ();
170         &pop    ("eax");
171         &bt     ("eax",9);
172         &jnc    (&label("nohalt"));     # interrupts are disabled

174         &rdtsc  ();
175         &push   ("edx");
176         &push   ("eax");
177         &halt   ();
178         &rdtsc  ();

180         &sub    ("eax",&DWP(0,"esp"));
181         &sbb    ("edx",&DWP(4,"esp"));
182         &add    ("esp",8);
183         &ret    ();

185 &set_label("nohalt");
186         &xor    ("eax","eax");
187         &xor    ("edx","edx");
188         &ret    ();
189 &function_end_B("OPENSSL_instrument_halt");

191 # Essentially there is only one use for this function. Under DJGPP:
192 #
193 #       #include <go32.h>
```



```
194 #       ...
195 #       i=OPENSSL_far_spin(_dos_ds,0x46c);
196 #       ...
197 # to obtain the number of spins till closest timer interrupt.

199 &function_begin_B("OPENSSL_far_spin");
200         &pushf  ();
201         &pop    ("eax")
202         &bt     ("eax",9);
203         &jnc    (&label("nospin"));     # interrupts are disabled

205         &mov    ("eax",&DWP(4,"esp"));
206         &mov    ("ecx",&DWP(8,"esp"));
207         &data_word (0x90d88e1e);        # push %ds, mov %eax,%ds
208         &xor    ("eax","eax");
209         &mov    ("edx",&DWP(0,"ecx"));
210         &jmp    (&label("spin"));

212         &align  (16);
213 &set_label("spin");
214         &inc    ("eax");
215         &cmp    ("edx",&DWP(0,"ecx"));
216         &je     (&label("spin"));

218         &data_word (0x1f909090);        # pop   %ds
219         &ret    ();

221 &set_label("nospin");
222         &xor    ("eax","eax");
223         &xor    ("edx","edx");
224         &ret    ();
225 &function_end_B("OPENSSL_far_spin");

227 &function_begin_B("OPENSSL_wipe_cpu","EXTRN\t_OPENSSL_ia32cap_P:DWORD");
228         &xor    ("eax","eax");
229         &xor    ("edx","edx");
230         &picmeup("ecx","OPENSSL_ia32cap_P");
231         &mov    ("ecx",&DWP(0,"ecx"));
232         &bt     (&DWP(0,"ecx"),1);
233         &jnc    (&label("no_x87"));
234         if ($sse2) {
235                 &and    ("ecx",1<<26|1<<24);    # check SSE2 and FXSR bits
236                 &cmp    ("ecx",1<<26|1<<24);
237                 &jne    (&label("no_sse2"));
238                 &pxor   ("xmm0","xmm0");
239                 &pxor   ("xmm1","xmm1");
240                 &pxor   ("xmm2","xmm2");
241                 &pxor   ("xmm3","xmm3");
242                 &pxor   ("xmm4","xmm4");
243                 &pxor   ("xmm5","xmm5");
244                 &pxor   ("xmm6","xmm6");
245                 &pxor   ("xmm7","xmm7");
246         &set_label("no_sse2");
247         }
248         # just a bunch of fldz to zap the fp/mm bank followed by finit...
249         &data_word(0xeed9eed9,0xeed9eed9,0xeed9eed9,0xeed9eed9,0x90e3db9b);
250 &set_label("no_x87");
251         &lea    ("eax",&DWP(4,"esp"));
252         &ret    ();
253 &function_end_B("OPENSSL_wipe_cpu");

255 &function_begin_B("OPENSSL_atomic_add");
256         &mov    ("edx",&DWP(4,"esp"));  # fetch the pointer, 1st arg
257         &mov    ("ecx",&DWP(8,"esp"));  # fetch the increment, 2nd arg
258         &push   ("ebx");
259         &nop    ();
```

```
260         &mov    ("eax",&DWP(0,"edx"));
261 &set_label("spin");
262         &lea    ("ebx",&DWP(0,"eax","ecx"));
263         &nop    ();
264         &data_word(0x1ab10ff0); # lock; cmpxchg %ebx,(%edx)     # %eax is envolv
265         &jne    (&label("spin"));
266         &mov    ("eax","ebx");  # OpenSSL expects the new value
267         &pop    ("ebx");
268         &ret    ();
269 &function_end_B("OPENSSL_atomic_add");

271 # This function can become handy under Win32 in situations when
272 # we don't know which calling convention, __stdcall or __cdecl(*),
273 # indirect callee is using. In C it can be deployed as
274 #
275 #ifdef OPENSSL_CPUID_OBJ
276 #       type OPENSSL_indirect_call(void *f,...);
277 #       ...
278 #       OPENSSL_indirect_call(func,[up to $max arguments]);
279 #endif
280 #
281 # (*)   it's designed to work even for __fastcall if number of
282 #       arguments is 1 or 2!
283 &function_begin_B("OPENSSL_indirect_call");
284         {
285         my ($max,$i)=(7,);      # $max has to be chosen as 4*n-1
286                                 # in order to preserve eventual
287                                 # stack alignment
288         &push   ("ebp");
289         &mov    ("ebp","esp");
290         &sub    ("esp",$max*4);
291         &mov    ("ecx",&DWP(12,"ebp"));
292         &mov    (&DWP(0,"esp"),"ecx");
293         &mov    ("edx",&DWP(16,"ebp"));
294         &mov    (&DWP(4,"esp"),"edx");
295         for($i=2;$i<$max;$i++)
296                 {
297                 # Some copies will be redundant/bogus...
298                 &mov    ("eax",&DWP(12+$i*4,"ebp"));
299                 &mov    (&DWP(0+$i*4,"esp"),"eax");
300                 }
301         &call_ptr       (&DWP(8,"ebp"));# make the call...
302         &mov    ("esp","ebp");  # ... and just restore the stack pointer
303                                 # without paying attention to what we called,
304                                 # (__cdecl *func) or (__stdcall *one).
305         &pop    ("ebp");
306         &ret    ();
307         }
308 &function_end_B("OPENSSL_indirect_call");

310 &function_begin_B("OPENSSL_cleanse");
311         &mov    ("edx",&wparam(0));
312         &mov    ("ecx",&wparam(1));
313         &xor    ("eax","eax");
314         &cmp    ("ecx",7);
315         &jae    (&label("lot"));
316         &cmp    ("ecx",0);
317         &je     (&label("ret"));
318 &set_label("little");
319         &mov    (&BP(0,"edx"),"al");
320         &sub    ("ecx",1);
321         &lea    ("edx",&DWP(1,"edx"));
322         &jnz    (&label("little"));
323 &set_label("ret");
324         &ret    ();
```



```
326 &set_label("lot",16);
327         &test   ("edx",3);
328         &jz     (&label("aligned"));
329         &mov    (&BP(0,"edx"),"al");
330         &lea    ("ecx",&DWP(-1,"ecx"));
331         &lea    ("edx",&DWP(1,"edx"));
332         &jmp    (&label("lot"));
333 &set_label("aligned");
334         &mov    (&DWP(0,"edx"),"eax");
335         &lea    ("ecx",&DWP(-4,"ecx"));
336         &test   ("ecx",-4);
337         &lea    ("edx",&DWP(4,"edx"));
338         &jnz    (&label("aligned"));
339         &cmp    ("ecx",0);
340         &jne    (&label("little"));
341         &ret    ();
342 &function_end_B("OPENSSL_cleanse");

344 &function_begin_B("OPENSSL_ia32_rdrand");
345         &mov    ("ecx",8);
346 &set_label("loop");
347         &rdrand ("eax");
348         &jc     (&label("break"));
349         &loop   (&label("loop"));
350 &set_label("break");
351         &cmp    ("eax",0);
352         &cmove  ("eax","ecx");
353         &ret    ();
354 &function_end_B("OPENSSL_ia32_rdrand");

356 &initseg("illumos_locking_setup");
357 &initseg("OPENSSL_cpuid_setup");

359 &hidden("illumos_locking_setup");
360 &hidden("OPENSSL_cpuid_setup");
361 &hidden("OPENSSL_ia32cap_P");

363 &asm_finish();
364 #endif /* ! codereview */
```

```
**********************************************************
    5986 Wed Aug 13 19:53:12 2014
new/usr/src/lib/openssl/libsunw_crypto/pl/x86gas.pl
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
#!/usr/bin/env perl

package x86gas;

*out=\@::out;

$::lbdecor=$::aout?"L":".L";            # local label decoration
$nmdecor=($::aout or $::coff)?"_":"sunw_";      # external name decoration

$initseg="";

$align=16;
$align=log($align)/log(2) if ($::aout);
$com_start="#" if ($::aout or $::coff);

sub opsize()
{ my $reg=shift;
    if    ($reg =~ m/^%e/o)             { "l"; }
    elsif ($reg =~ m/^%[a-d][hl]$/o)    { "b"; }
    elsif ($reg =~ m/^%[xm]/o)          { undef; }
    else                                { "w"; }
}

# swap arguments;
# expand opcode with size suffix;
# prefix numeric constants with $;
sub ::generic
{ my($opcode,@arg)=@_;
  my($suffix,$dst,$src);

    @arg=reverse(@arg);

    for (@arg)
    {   s/^(\*?)(e?[a-dsixphl]{2})$/$1%$2/o;    # gp registers
        s/^([xy]?mm[0-7])$/%$1/o;               # xmm/mmx registers
        s/^(\-?[0-9]+)$/\$$1/o;                 # constants
        s/^(\-?0x[0-9a-f]+)$/\$$1/o;            # constants
    }

    $dst = $arg[$#arg]          if ($#arg>=0);
    $src = $arg[$#arg-1]        if ($#arg>=1);
    if    ($dst =~ m/^%/o)      { $suffix=&opsize($dst); }
    elsif ($src =~ m/^%/o)      { $suffix=&opsize($src); }
    else                        { $suffix="l";           }
    undef $suffix if ($dst =~ m/^%[xm]/o || $src =~ m/^%[xm]/o);

    if ($#_==0)                         { &::emit($opcode);             }
    elsif ($#_==1 && $opcode =~ m/^(call|clflush|j|loop|set)/o)
                                        { &::emit($opcode,@arg);        }
    else                                { &::emit($opcode.$suffix,@arg);}

  1;
}
#
# opcodes not covered by ::generic above, mostly inconsistent namings...
#
sub ::movzx     { &::movzb(@_);                 }
sub ::pushfd    { &::pushfl;                    }
sub ::popfd     { &::popfl;                     }
sub ::cpuid     { &::emit(".byte\t0x0f,0xa2");  }
sub ::rdtsc     { &::emit(".byte\t0x0f,0x31");  }
```



```
sub ::call      { &::emit("call",(&::islabel($_[0]) or "$nmdecor$_[0]")); }
sub ::call_ptr  { &::generic("call","*$_[0]");  }
sub ::jmp_ptr   { &::generic("jmp","*$_[0]");   }

*::bswap = sub  { &::emit("bswap","%$_[0]");    } if (!$::i386);

sub ::DWP
{ my($addr,$reg1,$reg2,$idx)=@_;
  my $ret="";

    $addr =~ s/^\s+//;
    # prepend global references with optional underscore
    $addr =~ s/^([^\+\-0-9][^\+\-]*)/&::islabel($1) or "$nmdecor$1"/ige;

    $reg1 = "%$reg1" if ($reg1);
    $reg2 = "%$reg2" if ($reg2);

    $ret .= $addr if (($addr ne "") && ($addr ne 0));

    if ($reg2)
    {   $idx!= 0 or $idx=1;
        $ret .= "($reg1,$reg2,$idx)";
    }
    elsif ($reg1)
    {   $ret .= "($reg1)";      }

  $ret;
}
sub ::QWP       { &::DWP(@_);   }
sub ::BP        { &::DWP(@_);   }
sub ::WP        { &::DWP(@_);   }
sub ::BC        { @_;           }
sub ::DWC       { @_;           }

sub ::file
{   push(@out,".file\t\"$_[0].s\"\n.text\n");   }

sub ::function_begin_B
{ my $func=shift;
  my $global=($func !~ /^_/);
  my $begin="${::lbdecor}_${func}_begin";

    &::LABEL($func,$global?"$begin":"$nmdecor$func");
    $func=$nmdecor.$func;

    push(@out,".globl\t$func\n")        if ($global);
    if ($::coff)
    {   push(@out,".def\t$func;\t.scl\t".(3-$global).";\t.type\t32;\t.endef\n");
    elsif (($::aout and !$::pic) or $::macosx)
    { }
    else
    {   push(@out,".type        $func,\@function\n"); }
    push(@out,".align\t$align\n");
    push(@out,"$func:\n");
    push(@out,"$begin:\n")              if ($global);
    $::stack=4;
}

sub ::function_end_B
{ my $func=shift;
    push(@out,".size\t$nmdecor$func,.-".&::LABEL($func)."\n") if ($::elf);
    $::stack=0;
    &::wipe_labels();
}
```

```
sub ::comment
        {
        if (!defined($com_start) or $::elf)
                {       # Regarding $::elf above...
                # GNU and SVR4 as'es use different comment delimiters,
                push(@out,"\n");        # so we just skip ELF comments...
                return;
                }
        foreach (@_)
                {
                if (/^\s*$/)
                        { push(@out,"\n"); }
                else
                        { push(@out,"\t$com_start $_ $com_end\n"); }
                }
        }

sub ::external_label
{   foreach(@_) { &::LABEL($_,$nmdecor.$_); }   }

sub ::public_label
{   push(@out,".globl\t".&::LABEL($_[0],$nmdecor.$_[0])."\n");   }

sub ::file_end
{   if ($::macosx)
    {   if (%non_lazy_ptr)
        {   push(@out,".section __IMPORT,__pointers,non_lazy_symbol_pointers\n")
            foreach $i (keys %non_lazy_ptr)
            {   push(@out,"$non_lazy_ptr{$i}:\n.indirect_symbol\t$i\n.long\t0\n"
        }
    }
    if (grep {/\b${nmdecor}OPENSSL_ia32cap_P\b/i} @out) {
        my $tmp=".comm\t${nmdecor}OPENSSL_ia32cap_P,8";
        if ($::macosx)  { push (@out,"$tmp,2\n"); }
        elsif ($::elf)  { push (@out,"$tmp,4\n"); }
        else            { push (@out,"$tmp\n"); }
    }
    push(@out,$initseg) if ($initseg);
}

sub ::data_byte {   push(@out,".byte\t".join(',',@_)."\n");   }
sub ::data_short{   push(@out,".value\t".join(',',@_)."\n");  }
sub ::data_word {   push(@out,".long\t".join(',',@_)."\n");   }

sub ::align
{ my $val=$_[0],$p2,$i;
    if ($::aout)
    {   for ($p2=0;$val!=0;$val>>=1) { $p2++; }
        $val=$p2-1;
        $val.=",0x90";
    }
    push(@out,".align\t$val\n");
}

sub ::picmeup
{ my($dst,$sym,$base,$reflabel)=@_;

    if (($::pic && ($::elf || $::aout)) || $::macosx)
    {   if (!defined($base))
        {   &::call(&::label("PIC_me_up"));
            &::set_label("PIC_me_up");
            &::blindpop($dst);
            $base=$dst;
            $reflabel=&::label("PIC_me_up");
        }
        if ($::macosx)
```



```
        {   my $indirect=&::static_label("$nmdecor$sym\$non_lazy_ptr");
            &::mov($dst,&::DWP("$indirect-$reflabel",$base));
            $non_lazy_ptr{"$nmdecor$sym"}=$indirect;
        }
        else
        {   &::lea($dst,&::DWP("_GLOBAL_OFFSET_TABLE_+[.-$reflabel]",
                               $base));
            &::mov($dst,&::DWP("$sym\@GOT",$dst));
        }
    }
    else
    {   &::lea($dst,&::DWP($sym));      }
}

sub ::initseg
{ my $f=$nmdecor.shift;

    if ($::android)
    {   $initseg.=<<___;
.section        .init_array
.align  4
.long   $f
___
    }
    elsif ($::elf)
    {   $initseg.=<<___;
.section        .init
        call    $f
___
    }
    elsif ($::coff)
    {   $initseg.=<<___;        # applies to both Cygwin and Mingw
.section        .ctors
.long   $f
___
    }
    elsif ($::macosx)
    {   $initseg.=<<___;
.mod_init_func
.align 2
.long   $f
___
    }
    elsif ($::aout)
    {   my $ctor="${nmdecor}_GLOBAL_\$I\$\$f";
        $initseg.=".text\n";
        $initseg.=".type        $ctor,\@function\n" if ($::pic);
        $initseg.=<<___;        # OpenBSD way...
.globl  $ctor
.align  2
$ctor:
        jmp     $f
___
    }
}

sub ::dataseg
{   push(@out,".data\n");   }

*::hidden = sub { push(@out,".hidden\t$nmdecor$_[0]\n"); } if ($::elf);

1;
#endif /* ! codereview */
```

```
**********************************************************
    5679 Wed Aug 13 19:53:12 2014
new/usr/src/lib/openssl/libsunw_crypto/pqueue/pqueue.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/pqueue/pqueue.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/pqueue.h>

typedef struct _pqueue
        {
        pitem *items;
        int count;
        } pqueue_s;

pitem *
pitem_new(unsigned char *prio64be, void *data)
        {
        pitem *item = (pitem *) OPENSSL_malloc(sizeof(pitem));
        if (item == NULL) return NULL;

        memcpy(item->priority,prio64be,sizeof(item->priority));

        item->data = data;
        item->next = NULL;

        return item;
        }

void
pitem_free(pitem *item)
        {
        if (item == NULL) return;

        OPENSSL_free(item);
        }

pqueue_s *
pqueue_new()
        {
        pqueue_s *pq = (pqueue_s *) OPENSSL_malloc(sizeof(pqueue_s));
        if (pq == NULL) return NULL;

        memset(pq, 0x00, sizeof(pqueue_s));
        return pq;
        }

void
pqueue_free(pqueue_s *pq)
        {
        if (pq == NULL) return;

        OPENSSL_free(pq);
        }

pitem *
pqueue_insert(pqueue_s *pq, pitem *item)
        {
        pitem *curr, *next;

        if (pq->items == NULL)
                {
                pq->items = item;
                return item;
                }

        for(curr = NULL, next = pq->items;
                next != NULL;
                curr = next, next = next->next)
                {
                /* we can compare 64-bit value in big-endian encoding
                 * with memcmp:-) */
                int cmp = memcmp(next->priority, item->priority,8);
```

```
128                 if (cmp > 0)            /* next > item */
129                         {
130                         item->next = next;
131
132                         if (curr == NULL)
133                                 pq->items = item;
134                         else
135                                 curr->next = item;
136
137                         return item;
138                         }
139
140                 else if (cmp == 0)      /* duplicates not allowed */
141                         return NULL;
142                 }
143
144         item->next = NULL;
145         curr->next = item;
146
147         return item;
148         }
149
150 pitem *
151 pqueue_peek(pqueue_s *pq)
152         {
153         return pq->items;
154         }
155
156 pitem *
157 pqueue_pop(pqueue_s *pq)
158         {
159         pitem *item = pq->items;
160
161         if (pq->items != NULL)
162                 pq->items = pq->items->next;
163
164         return item;
165         }
166
167 pitem *
168 pqueue_find(pqueue_s *pq, unsigned char *prio64be)
169         {
170         pitem *next;
171         pitem *found = NULL;
172
173         if ( pq->items == NULL)
174                 return NULL;
175
176         for ( next = pq->items; next->next != NULL; next = next->next)
177                 {
178                 if ( memcmp(next->priority, prio64be,8) == 0)
179                         {
180                         found = next;
181                         break;
182                         }
183                 }
184
185         /* check the one last node */
186         if ( memcmp(next->priority, prio64be,8) ==0)
187                 found = next;
188
189         if ( ! found)
190                 return NULL;
191
192 #if 0 /* find works in peek mode */
193         if ( prev == NULL)
```



```
194                 pq->items = next->next;
195         else
196                 prev->next = next->next;
197 #endif
198
199         return found;
200         }
201
202 void
203 pqueue_print(pqueue_s *pq)
204         {
205         pitem *item = pq->items;
206
207         while(item != NULL)
208                 {
209                 printf("item\t%02x%02x%02x%02x%02x%02x%02x%02x\n",
210                         item->priority[0],item->priority[1],
211                         item->priority[2],item->priority[3],
212                         item->priority[4],item->priority[5],
213                         item->priority[6],item->priority[7]);
214                 item = item->next;
215                 }
216         }
217
218 pitem *
219 pqueue_iterator(pqueue_s *pq)
220         {
221         return pqueue_peek(pq);
222         }
223
224 pitem *
225 pqueue_next(pitem **item)
226         {
227         pitem *ret;
228
229         if ( item == NULL || *item == NULL)
230                 return NULL;
231
232
233         /* *item != NULL */
234         ret = *item;
235         *item = (*item)->next;
236
237         return ret;
238         }
239
240 int
241 pqueue_size(pqueue_s *pq)
242 {
243         pitem *item = pq->items;
244         int count = 0;
245
246         while(item != NULL)
247         {
248                 count++;
249                 item = item->next;
250         }
251         return count;
252 }
253 #endif /* ! codereview */
```

```
**********************************************************
   18662 Wed Aug 13 19:53:12 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/md_rand.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/rand/md_rand.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#define OPENSSL_FIPSEVP

#ifdef MD_RAND_DEBUG
# ifndef NDEBUG
#   define NDEBUG
# endif
#endif

#include <assert.h>
#include <stdio.h>
#include <string.h>

#include "e_os.h"

#include <openssl/crypto.h>
#include <openssl/rand.h>
```

```
128 #include "rand_lcl.h"

130 #include <openssl/err.h>

132 #ifdef BN_DEBUG
133 # define PREDICT
134 #endif

136 /* #define PREDICT      1 */

138 #define STATE_SIZE      1023
139 static int state_num=0,state_index=0;
140 static unsigned char state[STATE_SIZE+MD_DIGEST_LENGTH];
141 static unsigned char md[MD_DIGEST_LENGTH];
142 static long md_count[2]={0,0};
143 static double entropy=0;
144 static int initialized=0;

146 static unsigned int crypto_lock_rand = 0; /* may be set only when a thread
147                                            * holds CRYPTO_LOCK_RAND
148                                            * (to prevent double locking) */
149 /* access to lockin_thread is synchronized by CRYPTO_LOCK_RAND2 */
150 static CRYPTO_THREADID locking_threadid; /* valid iff crypto_lock_rand is set */


153 #ifdef PREDICT
154 int rand_predictable=0;
155 #endif

157 const char RAND_version[]="RAND" OPENSSL_VERSION_PTEXT;

159 static void ssleay_rand_cleanup(void);
160 static void ssleay_rand_seed(const void *buf, int num);
161 static void ssleay_rand_add(const void *buf, int num, double add_entropy);
162 static int ssleay_rand_nopseudo_bytes(unsigned char *buf, int num);
163 static int ssleay_rand_pseudo_bytes(unsigned char *buf, int num);
164 static int ssleay_rand_status(void);

166 RAND_METHOD rand_ssleay_meth={
167         ssleay_rand_seed,
168         ssleay_rand_nopseudo_bytes,
169         ssleay_rand_cleanup,
170         ssleay_rand_add,
171         ssleay_rand_pseudo_bytes,
172         ssleay_rand_status
173         };

175 RAND_METHOD *RAND_SSLeay(void)
176         {
177         return(&rand_ssleay_meth);
178         }

180 static void ssleay_rand_cleanup(void)
181         {
182         OPENSSL_cleanse(state,sizeof(state));
183         state_num=0;
184         state_index=0;
185         OPENSSL_cleanse(md,MD_DIGEST_LENGTH);
186         md_count[0]=0;
187         md_count[1]=0;
188         entropy=0;
189         initialized=0;
190         }

192 static void ssleay_rand_add(const void *buf, int num, double add)
193         {
```



```
194         int i,j,k,st_idx;
195         long md_c[2];
196         unsigned char local_md[MD_DIGEST_LENGTH];
197         EVP_MD_CTX m;
198         int do_not_lock;

200         if (!num)
201                 return;

203         /*
204          * (Based on the rand(3) manpage)
205          *
206          * The input is chopped up into units of 20 bytes (or less for
207          * the last block).  Each of these blocks is run through the hash
208          * function as follows:  The data passed to the hash function
209          * is the current 'md', the same number of bytes from the 'state'
210          * (the location determined by in incremented looping index) as
211          * the current 'block', the new key data 'block', and 'count'
212          * (which is incremented after each use).
213          * The result of this is kept in 'md' and also xored into the
214          * 'state' at the same locations that were used as input into the
215          * hash function.
216          */

218         /* check if we already have the lock */
219         if (crypto_lock_rand)
220                 {
221                 CRYPTO_THREADID cur;
222                 CRYPTO_THREADID_current(&cur);
223                 CRYPTO_r_lock(CRYPTO_LOCK_RAND2);
224                 do_not_lock = !CRYPTO_THREADID_cmp(&locking_threadid, &cur);
225                 CRYPTO_r_unlock(CRYPTO_LOCK_RAND2);
226                 }
227         else
228                 do_not_lock = 0;

230         if (!do_not_lock) CRYPTO_w_lock(CRYPTO_LOCK_RAND);
231         st_idx=state_index;

233         /* use our own copies of the counters so that even
234          * if a concurrent thread seeds with exactly the
235          * same data and uses the same subarray there's _some_
236          * difference */
237         md_c[0] = md_count[0];
238         md_c[1] = md_count[1];

240         memcpy(local_md, md, sizeof md);

242         /* state_index <= state_num <= STATE_SIZE */
243         state_index += num;
244         if (state_index >= STATE_SIZE)
245                 {
246                 state_index%=STATE_SIZE;
247                 state_num=STATE_SIZE;
248                 }
249         else if (state_num < STATE_SIZE)
250                 {
251                 if (state_index > state_num)
252                         state_num=state_index;
253                 }
254         /* state_index <= state_num <= STATE_SIZE */

256         /* state[st_idx], ..., state[(st_idx + num - 1) % STATE_SIZE]
257          * are what we will use now, but other threads may use them
258          * as well */
```

```
260         md_count[1] += (num / MD_DIGEST_LENGTH) + (num % MD_DIGEST_LENGTH > 0);

262         if (!do_not_lock) CRYPTO_w_unlock(CRYPTO_LOCK_RAND);

264         EVP_MD_CTX_init(&m);
265         for (i=0; i<num; i+=MD_DIGEST_LENGTH)
266                 {
267                 j=(num-i);
268                 j=(j > MD_DIGEST_LENGTH)?MD_DIGEST_LENGTH:j;

270                 MD_Init(&m);
271                 MD_Update(&m,local_md,MD_DIGEST_LENGTH);
272                 k=(st_idx+j)-STATE_SIZE;
273                 if (k > 0)
274                         {
275                         MD_Update(&m,&(state[st_idx]),j-k);
276                         MD_Update(&m,&(state[0]),k);
277                         }
278                 else
279                         MD_Update(&m,&(state[st_idx]),j);

281                 /* DO NOT REMOVE THE FOLLOWING CALL TO MD_Update()! */
282                 MD_Update(&m,buf,j);
283                 /* We know that line may cause programs such as
284                    purify and valgrind to complain about use of
285                    uninitialized data.  The problem is not, it's
286                    with the caller.  Removing that line will make
287                    sure you get really bad randomness and thereby
288                    other problems such as very insecure keys. */

290                 MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
291                 MD_Final(&m,local_md);
292                 md_c[1]++;

294                 buf=(const char *)buf + j;

296                 for (k=0; k<j; k++)
297                         {
298                         /* Parallel threads may interfere with this,
299                          * but always each byte of the new state is
300                          * the XOR of some previous value of its
301                          * and local_md (itermediate values may be lost).
302                          * Alway using locking could hurt performance more
303                          * than necessary given that conflicts occur only
304                          * when the total seeding is longer than the random
305                          * state. */
306                         state[st_idx++]^=local_md[k];
307                         if (st_idx >= STATE_SIZE)
308                                 st_idx=0;
309                         }
310                 }
311         EVP_MD_CTX_cleanup(&m);

313         if (!do_not_lock) CRYPTO_w_lock(CRYPTO_LOCK_RAND);
314         /* Don't just copy back local_md into md -- this could mean that
315          * other thread's seeding remains without effect (except for
316          * the incremented counter).  By XORing it we keep at least as
317          * much entropy as fits into md. */
318         for (k = 0; k < (int)sizeof(md); k++)
319                 {
320                 md[k] ^= local_md[k];
321                 }
322         if (entropy < ENTROPY_NEEDED) /* stop counting when we have enough */
323             entropy += add;
324         if (!do_not_lock) CRYPTO_w_unlock(CRYPTO_LOCK_RAND);
```



```
326 #if !defined(OPENSSL_THREADS) && !defined(OPENSSL_SYS_WIN32)
327         assert(md_c[1] == md_count[1]);
328 #endif
329         }

331 static void ssleay_rand_seed(const void *buf, int num)
332         {
333         ssleay_rand_add(buf, num, (double)num);
334         }

336 int ssleay_rand_bytes(unsigned char *buf, int num, int pseudo, int lock)
337         {
338         static volatile int stirred_pool = 0;
339         int i,j,k,st_num,st_idx;
340         int num_ceil;
341         int ok;
342         long md_c[2];
343         unsigned char local_md[MD_DIGEST_LENGTH];
344         EVP_MD_CTX m;
345 #ifndef GETPID_IS_MEANINGLESS
346         pid_t curr_pid = getpid();
347 #endif
348         int do_stir_pool = 0;

350 #ifdef PREDICT
351         if (rand_predictable)
352                 {
353                 static unsigned char val=0;

355                 for (i=0; i<num; i++)
356                         buf[i]=val++;
357                 return(1);
358                 }
359 #endif

361         if (num <= 0)
362                 return 1;

364         EVP_MD_CTX_init(&m);
365         /* round upwards to multiple of MD_DIGEST_LENGTH/2 */
366         num_ceil = (1 + (num-1)/(MD_DIGEST_LENGTH/2)) * (MD_DIGEST_LENGTH/2);

368         /*
369          * (Based on the rand(3) manpage:)
370          *
371          * For each group of 10 bytes (or less), we do the following:
372          *
373          * Input into the hash function the local 'md' (which is initialized fro
374          * the global 'md' before any bytes are generated), the bytes that are t
375          * be overwritten by the random bytes, and bytes from the 'state'
376          * (incrementing looping index). From this digest output (which is kept
377          * in 'md'), the top (up to) 10 bytes are returned to the caller and the
378          * bottom 10 bytes are xored into the 'state'.
379          *
380          * Finally, after we have finished 'num' random bytes for the
381          * caller, 'count' (which is incremented) and the local and global 'md'
382          * are fed into the hash function and the results are kept in the
383          * global 'md'.
384          */
385         if (lock)
386                 CRYPTO_w_lock(CRYPTO_LOCK_RAND);

388         /* prevent ssleay_rand_bytes() from trying to obtain the lock again */
389         CRYPTO_w_lock(CRYPTO_LOCK_RAND2);
390         CRYPTO_THREADID_current(&locking_threadid);
391         CRYPTO_w_unlock(CRYPTO_LOCK_RAND2);
```

```
392         crypto_lock_rand = 1;

394         if (!initialized)
395                 {
396                 RAND_poll();
397                 initialized = 1;
398                 }

400         if (!stirred_pool)
401                 do_stir_pool = 1;

403         ok = (entropy >= ENTROPY_NEEDED);
404         if (!ok)
405                 {
406                 /* If the PRNG state is not yet unpredictable, then seeing
407                  * the PRNG output may help attackers to determine the new
408                  * state; thus we have to decrease the entropy estimate.
409                  * Once we've had enough initial seeding we don't bother to
410                  * adjust the entropy count, though, because we're not ambitious
411                  * to provide *information-theoretic* randomness.
412                  *
413                  * NOTE: This approach fails if the program forks before
414                  * we have enough entropy. Entropy should be collected
415                  * in a separate input pool and be transferred to the
416                  * output pool only when the entropy limit has been reached.
417                  */
418                 entropy -= num;
419                 if (entropy < 0)
420                         entropy = 0;
421                 }

423         if (do_stir_pool)
424                 {
425                 /* In the output function only half of 'md' remains secret,
426                  * so we better make sure that the required entropy gets
427                  * 'evenly distributed' through 'state', our randomness pool.
428                  * The input function (ssleay_rand_add) chains all of 'md',
429                  * which makes it more suitable for this purpose.
430                  */

432                 int n = STATE_SIZE; /* so that the complete pool gets accessed *
433                 while (n > 0)
434                         {
435 #if MD_DIGEST_LENGTH > 20
436 # error "Please adjust DUMMY_SEED."
437 #endif
438 #define DUMMY_SEED "...................." /* at least MD_DIGEST_LENGTH */
439                         /* Note that the seed does not matter, it's just that
440                          * ssleay_rand_add expects to have something to hash. */
441                         ssleay_rand_add(DUMMY_SEED, MD_DIGEST_LENGTH, 0.0);
442                         n -= MD_DIGEST_LENGTH;
443                         }
444                 if (ok)
445                         stirred_pool = 1;
446                 }

448         st_idx=state_index;
449         st_num=state_num;
450         md_c[0] = md_count[0];
451         md_c[1] = md_count[1];
452         memcpy(local_md, md, sizeof md);

454         state_index+=num_ceil;
455         if (state_index > state_num)
456                 state_index %= state_num;
```



```
458         /* state[st_idx], ..., state[(st_idx + num_ceil - 1) % st_num]
459          * are now ours (but other threads may use them too) */

461         md_count[0] += 1;

463         /* before unlocking, we must clear 'crypto_lock_rand' */
464         crypto_lock_rand = 0;
465         if (lock)
466                 CRYPTO_w_unlock(CRYPTO_LOCK_RAND);

468         while (num > 0)
469                 {
470                 /* num_ceil -= MD_DIGEST_LENGTH/2 */
471                 j=(num >= MD_DIGEST_LENGTH/2)?MD_DIGEST_LENGTH/2:num;
472                 num-=j;
473                 MD_Init(&m);
474 #ifndef GETPID_IS_MEANINGLESS
475                 if (curr_pid) /* just in the first iteration to save time */
476                         {
477                         MD_Update(&m,(unsigned char*)&curr_pid,sizeof curr_pid);
478                         curr_pid = 0;
479                         }
480 #endif
481                 MD_Update(&m,local_md,MD_DIGEST_LENGTH);
482                 MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));

484 #ifndef PURIFY /* purify complains */
485                 /* The following line uses the supplied buffer as a small
486                  * source of entropy: since this buffer is often uninitialised
487                  * it may cause programs such as purify or valgrind to
488                  * complain. So for those builds it is not used: the removal
489                  * of such a small source of entropy has negligible impact on
490                  * security.
491                  */
492                 MD_Update(&m,buf,j);
493 #endif

495                 k=(st_idx+MD_DIGEST_LENGTH/2)-st_num;
496                 if (k > 0)
497                         {
498                         MD_Update(&m,&(state[st_idx]),MD_DIGEST_LENGTH/2-k);
499                         MD_Update(&m,&(state[0]),k);
500                         }
501                 else
502                         MD_Update(&m,&(state[st_idx]),MD_DIGEST_LENGTH/2);
503                 MD_Final(&m,local_md);

505                 for (i=0; i<MD_DIGEST_LENGTH/2; i++)
506                         {
507                         state[st_idx++]^=local_md[i]; /* may compete with other
508                         if (st_idx >= st_num)
509                                 st_idx=0;
510                         if (i < j)
511                                 *(buf++)=local_md[i+MD_DIGEST_LENGTH/2];
512                         }
513                 }

515         MD_Init(&m);
516         MD_Update(&m,(unsigned char *)&(md_c[0]),sizeof(md_c));
517         MD_Update(&m,local_md,MD_DIGEST_LENGTH);
518         if (lock)
519                 CRYPTO_w_lock(CRYPTO_LOCK_RAND);
520         MD_Update(&m,md,MD_DIGEST_LENGTH);
521         MD_Final(&m,md);
522         if (lock)
523                 CRYPTO_w_unlock(CRYPTO_LOCK_RAND);
```

```
525         EVP_MD_CTX_cleanup(&m);
526         if (ok)
527                 return(1);
528         else if (pseudo)
529                 return 0;
530         else
531                 {
532                 RANDerr(RAND_F_SSLEAY_RAND_BYTES,RAND_R_PRNG_NOT_SEEDED);
533                 ERR_add_error_data(1, "You need to read the OpenSSL FAQ, "
534                         "http://www.openssl.org/support/faq.html");
535                 return(0);
536                 }
537         }

539 static int ssleay_rand_nopseudo_bytes(unsigned char *buf, int num)
540         {
541         return ssleay_rand_bytes(buf, num, 0, 1);
542         }

544 /* pseudo-random bytes that are guaranteed to be unique but not
545    unpredictable */
546 static int ssleay_rand_pseudo_bytes(unsigned char *buf, int num)
547         {
548         return ssleay_rand_bytes(buf, num, 1, 1);
549         }

551 static int ssleay_rand_status(void)
552         {
553         CRYPTO_THREADID cur;
554         int ret;
555         int do_not_lock;

557         CRYPTO_THREADID_current(&cur);
558         /* check if we already have the lock
559          * (could happen if a RAND_poll() implementation calls RAND_status()) */
560         if (crypto_lock_rand)
561                 {
562                 CRYPTO_r_lock(CRYPTO_LOCK_RAND2);
563                 do_not_lock = !CRYPTO_THREADID_cmp(&locking_threadid, &cur);
564                 CRYPTO_r_unlock(CRYPTO_LOCK_RAND2);
565                 }
566         else
567                 do_not_lock = 0;

569         if (!do_not_lock)
570                 {
571                 CRYPTO_w_lock(CRYPTO_LOCK_RAND);

573                 /* prevent ssleay_rand_bytes() from trying to obtain the lock ag
574                 CRYPTO_w_lock(CRYPTO_LOCK_RAND2);
575                 CRYPTO_THREADID_cpy(&locking_threadid, &cur);
576                 CRYPTO_w_unlock(CRYPTO_LOCK_RAND2);
577                 crypto_lock_rand = 1;
578                 }

580         if (!initialized)
581                 {
582                 RAND_poll();
583                 initialized = 1;
584                 }

586         ret = entropy >= ENTROPY_NEEDED;

588         if (!do_not_lock)
589                 {
```



```
590                 /* before unlocking, we must clear 'crypto_lock_rand' */
591                 crypto_lock_rand = 0;

593                 CRYPTO_w_unlock(CRYPTO_LOCK_RAND);
594                 }

596         return ret;
597         }
598 #endif /* ! codereview */
```

```
**********************************************************
    8572 Wed Aug 13 19:53:12 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/rand_egd.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/rand_egd.c */
/* Written by Ulf Moeller and Lutz Jaenicke for the OpenSSL project. */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/e_os2.h>
#include <openssl/rand.h>
#include <openssl/buffer.h>

/*
```



```
 * Query the EGD <URL: http://www.lothar.com/tech/crypto/>.
 *
 * This module supplies three routines:
 *
 * RAND_query_egd_bytes(path, buf, bytes)
 *   will actually query "bytes" bytes of entropy form the egd-socket located
 *   at path and will write them to buf (if supplied) or will directly feed
 *   it to RAND_seed() if buf==NULL.
 *   The number of bytes is not limited by the maximum chunk size of EGD,
 *   which is 255 bytes. If more than 255 bytes are wanted, several chunks
 *   of entropy bytes are requested. The connection is left open until the
 *   query is competed.
 *   RAND_query_egd_bytes() returns with
 *     -1  if an error occured during connection or communication.
 *     num the number of bytes read from the EGD socket. This number is either
 *         the number of bytes requested or smaller, if the EGD pool is
 *         drained and the daemon signals that the pool is empty.
 *   This routine does not touch any RAND_status(). This is necessary, since
 *   PRNG functions may call it during initialization.
 *
 * RAND_egd_bytes(path, bytes) will query "bytes" bytes and have them
 *   used to seed the PRNG.
 *   RAND_egd_bytes() is a wrapper for RAND_query_egd_bytes() with buf=NULL.
 *   Unlike RAND_query_egd_bytes(), RAND_status() is used to test the
 *   seed status so that the return value can reflect the seed state:
 *     -1  if an error occured during connection or communication _or_
 *         if the PRNG has still not received the required seeding.
 *     num the number of bytes read from the EGD socket. This number is either
 *         the number of bytes requested or smaller, if the EGD pool is
 *         drained and the daemon signals that the pool is empty.
 *
 * RAND_egd(path) will query 255 bytes and use the bytes retreived to seed
 *   the PRNG.
 *   RAND_egd() is a wrapper for RAND_egd_bytes() with numbytes=255.
 */

#if defined(OPENSSL_SYS_WIN32) || defined(OPENSSL_SYS_VMS) || defined(OPENSSL_SY
int RAND_query_egd_bytes(const char *path, unsigned char *buf, int bytes)
        {
        return(-1);
        }
int RAND_egd(const char *path)
        {
        return(-1);
        }

int RAND_egd_bytes(const char *path,int bytes)
        {
        return(-1);
        }
#else
#include <openssl/opensslconf.h>
#include OPENSSL_UNISTD
#include <sys/types.h>
#include <sys/socket.h>
#ifndef NO_SYS_UN_H
# ifdef OPENSSL_SYS_VXWORKS
#   include <streams/un.h>
# else
#   include <sys/un.h>
# endif
#else
struct  sockaddr_un {
        short   sun_family;             /* AF_UNIX */
        char    sun_path[108];          /* path name (gag) */
};
```

```
#endif /* NO_SYS_UN_H */
#include <string.h>
#include <errno.h>

#ifndef offsetof
#  define offsetof(TYPE, MEMBER) ((size_t) &((TYPE *)0)->MEMBER)
#endif

int RAND_query_egd_bytes(const char *path, unsigned char *buf, int bytes)
        {
        int ret = 0;
        struct sockaddr_un addr;
        int len, num, numbytes;
        int fd = -1;
        int success;
        unsigned char egdbuf[2], tempbuf[255], *retrievebuf;

        memset(&addr, 0, sizeof(addr));
        addr.sun_family = AF_UNIX;
        if (strlen(path) >= sizeof(addr.sun_path))
                return (-1);
        BUF_strlcpy(addr.sun_path,path,sizeof addr.sun_path);
        len = offsetof(struct sockaddr_un, sun_path) + strlen(path);
        fd = socket(AF_UNIX, SOCK_STREAM, 0);
        if (fd == -1) return (-1);
        success = 0;
        while (!success)
            {
            if (connect(fd, (struct sockaddr *)&addr, len) == 0)
               success = 1;
            else
                {
                switch (errno)
                    {
#ifdef EINTR
                    case EINTR:
#endif
#ifdef EAGAIN
                    case EAGAIN:
#endif
#ifdef EINPROGRESS
                    case EINPROGRESS:
#endif
#ifdef EALREADY
                    case EALREADY:
#endif
                        /* No error, try again */
                        break;
#ifdef EISCONN
                    case EISCONN:
                        success = 1;
                        break;
#endif
                    default:
                        goto err;       /* failure */
                    }
                }
            }

        while(bytes > 0)
            {
            egdbuf[0] = 1;
            egdbuf[1] = bytes < 255 ? bytes : 255;
            numbytes = 0;
            while (numbytes != 2)
                {
```



```
                num = write(fd, egdbuf + numbytes, 2 - numbytes);
                if (num >= 0)
                    numbytes += num;
                else
                    {
                    switch (errno)
                        {
#ifdef EINTR
                        case EINTR:
#endif
#ifdef EAGAIN
                        case EAGAIN:
#endif
                            /* No error, try again */
                            break;
                        default:
                            ret = -1;
                            goto err;   /* failure */
                        }
                    }
                }
            numbytes = 0;
            while (numbytes != 1)
                {
                num = read(fd, egdbuf, 1);
                if (num == 0)
                        goto err;       /* descriptor closed */
                else if (num > 0)
                    numbytes += num;
                else
                    {
                    switch (errno)
                        {
#ifdef EINTR
                        case EINTR:
#endif
#ifdef EAGAIN
                        case EAGAIN:
#endif
                            /* No error, try again */
                            break;
                        default:
                            ret = -1;
                            goto err;   /* failure */
                        }
                    }
                }
            if(egdbuf[0] == 0)
                goto err;
            if (buf)
                retrievebuf = buf + ret;
            else
                retrievebuf = tempbuf;
            numbytes = 0;
            while (numbytes != egdbuf[0])
                {
                num = read(fd, retrievebuf + numbytes, egdbuf[0] - numbytes);
                if (num == 0)
                        goto err;       /* descriptor closed */
                else if (num > 0)
                    numbytes += num;
                else
                    {
                    switch (errno)
                        {
#ifdef EINTR
```

```
260                         case EINTR:
261 #endif
262 #ifdef EAGAIN
263                         case EAGAIN:
264 #endif
265                             /* No error, try again */
266                             break;
267                         default:
268                             ret = -1;
269                             goto err;   /* failure */
270                         }
271                     }
272                 }
273             ret += egdbuf[0];
274             bytes -= egdbuf[0];
275             if (!buf)
276                 RAND_seed(tempbuf, egdbuf[0]);
277             }
278  err:
279         if (fd != -1) close(fd);
280         return(ret);
281         }


284 int RAND_egd_bytes(const char *path, int bytes)
285         {
286         int num, ret = 0;

288         num = RAND_query_egd_bytes(path, NULL, bytes);
289         if (num < 1) goto err;
290         if (RAND_status() == 1)
291             ret = num;
292  err:
293         return(ret);
294         }


297 int RAND_egd(const char *path)
298         {
299         return (RAND_egd_bytes(path, 255));
300         }


303 #endif
304 #endif /* ! codereview */
```

```
**********************************************************
    3965 Wed Aug 13 19:53:12 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/rand_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/rand_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/rand.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_RAND,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_RAND,0,reason)

static ERR_STRING_DATA RAND_str_functs[]=
        {
{ERR_FUNC(RAND_F_RAND_GET_RAND_METHOD), "RAND_get_rand_method"},
{ERR_FUNC(RAND_F_RAND_INIT_FIPS),       "RAND_init_fips"},
{ERR_FUNC(RAND_F_SSLEAY_RAND_BYTES),    "SSLEAY_RAND_BYTES"},
{0,NULL}
        };

static ERR_STRING_DATA RAND_str_reasons[]=
        {
{ERR_REASON(RAND_R_DUAL_EC_DRBG_DISABLED),"dual ec drbg disabled"},
{ERR_REASON(RAND_R_ERROR_INITIALISING_DRBG),"error initialising drbg"},
{ERR_REASON(RAND_R_ERROR_INSTANTIATING_DRBG),"error instantiating drbg"},
{ERR_REASON(RAND_R_NO_FIPS_RANDOM_METHOD_SET),"no fips random method set"},
{ERR_REASON(RAND_R_PRNG_NOT_SEEDED)      ,"PRNG not seeded"},
{0,NULL}
        };

#endif

void ERR_load_RAND_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(RAND_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,RAND_str_functs);
                ERR_load_strings(0,RAND_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    8660 Wed Aug 13 19:53:12 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/rand_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/rand_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
```



```
#include <openssl/rand.h>

#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#include <openssl/fips_rand.h>
#include "rand_lcl.h"
#endif

#ifndef OPENSSL_NO_ENGINE
/* non-NULL if default_RAND_meth is ENGINE-provided */
static ENGINE *funct_ref =NULL;
#endif
static const RAND_METHOD *default_RAND_meth = NULL;

int RAND_set_rand_method(const RAND_METHOD *meth)
        {
#ifndef OPENSSL_NO_ENGINE
        if(funct_ref)
                {
                ENGINE_finish(funct_ref);
                funct_ref = NULL;
                }
#endif
        default_RAND_meth = meth;
        return 1;
        }

const RAND_METHOD *RAND_get_rand_method(void)
        {
        if (!default_RAND_meth)
                {
#ifndef OPENSSL_NO_ENGINE
                ENGINE *e = ENGINE_get_default_RAND();
                if(e)
                        {
                        default_RAND_meth = ENGINE_get_RAND(e);
                        if(!default_RAND_meth)
                                {
                                ENGINE_finish(e);
                                e = NULL;
                                }
                        }
                if(e)
                        funct_ref = e;
                else
#endif
                        default_RAND_meth = RAND_SSLeay();
                }
        return default_RAND_meth;
        }

#ifndef OPENSSL_NO_ENGINE
int RAND_set_rand_engine(ENGINE *engine)
        {
        const RAND_METHOD *tmp_meth = NULL;
        if(engine)
                {
                if(!ENGINE_init(engine))
                        return 0;
                tmp_meth = ENGINE_get_RAND(engine);
                if(!tmp_meth)
                        {
```

```
128                         ENGINE_finish(engine);
129                         return 0;
130                         }
131                 }
132         /* This function releases any prior ENGINE so call it first */
133         RAND_set_rand_method(tmp_meth);
134         funct_ref = engine;
135         return 1;
136         }
137 #endif

139 void RAND_cleanup(void)
140         {
141         const RAND_METHOD *meth = RAND_get_rand_method();
142         if (meth && meth->cleanup)
143                 meth->cleanup();
144         RAND_set_rand_method(NULL);
145         }

147 void RAND_seed(const void *buf, int num)
148         {
149         const RAND_METHOD *meth = RAND_get_rand_method();
150         if (meth && meth->seed)
151                 meth->seed(buf,num);
152         }

154 void RAND_add(const void *buf, int num, double entropy)
155         {
156         const RAND_METHOD *meth = RAND_get_rand_method();
157         if (meth && meth->add)
158                 meth->add(buf,num,entropy);
159         }

161 int RAND_bytes(unsigned char *buf, int num)
162         {
163         const RAND_METHOD *meth = RAND_get_rand_method();
164         if (meth && meth->bytes)
165                 return meth->bytes(buf,num);
166         return(-1);
167         }

169 int RAND_pseudo_bytes(unsigned char *buf, int num)
170         {
171         const RAND_METHOD *meth = RAND_get_rand_method();
172         if (meth && meth->pseudorand)
173                 return meth->pseudorand(buf,num);
174         return(-1);
175         }

177 int RAND_status(void)
178         {
179         const RAND_METHOD *meth = RAND_get_rand_method();
180         if (meth && meth->status)
181                 return meth->status();
182         return 0;
183         }

185 #ifdef OPENSSL_FIPS

187 /* FIPS DRBG initialisation code. This sets up the DRBG for use by the
188  * rest of OpenSSL.
189  */

191 /* Entropy gatherer: use standard OpenSSL PRNG to seed (this will gather
192  * entropy internally through RAND_poll().
193  */
```



```
195 static size_t drbg_get_entropy(DRBG_CTX *ctx, unsigned char **pout,
196                                int entropy, size_t min_len, size_t max_len)
197         {
198         /* Round up request to multiple of block size */
199         min_len = ((min_len + 19) / 20) * 20;
200         *pout = OPENSSL_malloc(min_len);
201         if (!*pout)
202                 return 0;
203         if (ssleay_rand_bytes(*pout, min_len, 0, 0) <= 0)
204                 {
205                 OPENSSL_free(*pout);
206                 *pout = NULL;
207                 return 0;
208                 }
209         return min_len;
210         }

212 static void drbg_free_entropy(DRBG_CTX *ctx, unsigned char *out, size_t olen)
213         {
214         if (out)
215                 {
216                 OPENSSL_cleanse(out, olen);
217                 OPENSSL_free(out);
218                 }
219         }

221 /* Set "additional input" when generating random data. This uses the
222  * current PID, a time value and a counter.
223  */

225 static size_t drbg_get_adin(DRBG_CTX *ctx, unsigned char **pout)
226         {
227         /* Use of static variables is OK as this happens under a lock */
228         static unsigned char buf[16];
229         static unsigned long counter;
230         FIPS_get_timevec(buf, &counter);
231         *pout = buf;
232         return sizeof(buf);
233         }

235 /* RAND_add() and RAND_seed() pass through to OpenSSL PRNG so it is
236  * correctly seeded by RAND_poll().
237  */

239 static int drbg_rand_add(DRBG_CTX *ctx, const void *in, int inlen,
240                                 double entropy)
241         {
242         RAND_SSLeay()->add(in, inlen, entropy);
243         return 1;
244         }

246 static int drbg_rand_seed(DRBG_CTX *ctx, const void *in, int inlen)
247         {
248         RAND_SSLeay()->seed(in, inlen);
249         return 1;
250         }

252 #ifndef OPENSSL_DRBG_DEFAULT_TYPE
253 #define OPENSSL_DRBG_DEFAULT_TYPE       NID_aes_256_ctr
254 #endif
255 #ifndef OPENSSL_DRBG_DEFAULT_FLAGS
256 #define OPENSSL_DRBG_DEFAULT_FLAGS      DRBG_FLAG_CTR_USE_DF
257 #endif

259 static int fips_drbg_type = OPENSSL_DRBG_DEFAULT_TYPE;
```

```
260 static int fips_drbg_flags = OPENSSL_DRBG_DEFAULT_FLAGS;

262 void RAND_set_fips_drbg_type(int type, int flags)
263         {
264         fips_drbg_type = type;
265         fips_drbg_flags = flags;
266         }

268 int RAND_init_fips(void)
269         {
270         DRBG_CTX *dctx;
271         size_t plen;
272         unsigned char pers[32], *p;
273 #ifndef OPENSSL_ALLOW_DUAL_EC_DRBG
274         if (fips_drbg_type >> 16)
275                 {
276                 RANDerr(RAND_F_RAND_INIT_FIPS, RAND_R_DUAL_EC_DRBG_DISABLED);
277                 return 0;
278                 }
279 #endif

281         dctx = FIPS_get_default_drbg();
282         if (FIPS_drbg_init(dctx, fips_drbg_type, fips_drbg_flags) <= 0)
283                 {
284                 RANDerr(RAND_F_RAND_INIT_FIPS, RAND_R_ERROR_INITIALISING_DRBG);
285                 return 0;
286                 }

288         FIPS_drbg_set_callbacks(dctx,
289                                 drbg_get_entropy, drbg_free_entropy, 20,
290                                 drbg_get_entropy, drbg_free_entropy);
291         FIPS_drbg_set_rand_callbacks(dctx, drbg_get_adin, 0,
292                                      drbg_rand_seed, drbg_rand_add);
293         /* Personalisation string: a string followed by date time vector */
294         strcpy((char *)pers, "OpenSSL DRBG2.0");
295         plen = drbg_get_adin(dctx, &p);
296         memcpy(pers + 16, p, plen);

298         if (FIPS_drbg_instantiate(dctx, pers, sizeof(pers)) <= 0)
299                 {
300                 RANDerr(RAND_F_RAND_INIT_FIPS, RAND_R_ERROR_INSTANTIATING_DRBG);
301                 return 0;
302                 }
303         FIPS_rand_set_method(FIPS_drbg_method());
304         return 1;
305         }

307 #endif
308 #endif /* ! codereview */
```

```
**********************************************************
    7700 Wed Aug 13 19:53:13 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/rand_nw.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/rand/rand_nw.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/rand.h>
#include "rand_lcl.h"

#if defined (OPENSSL_SYS_NETWARE)

#if defined(NETWARE_LIBC)
#include <nks/thread.h>
#else
#include <nwthread.h>
#endif

extern int GetProcessSwitchCount(void);
#if !defined(NETWARE_LIBC) || (CURRENT_NDK_THRESHOLD < 509220000)
extern void *RunningProcess; /* declare here same as found in newer NDKs */
extern unsigned long GetSuperHighResolutionTimer(void);
```

```
#endif

   /* the FAQ indicates we need to provide at least 20 bytes (160 bits) of seed
   */
int RAND_poll(void)
{
   unsigned long l;
   unsigned long tsc;
   int i;

      /* There are several options to gather miscellaneous data
       * but for now we will loop checking the time stamp counter (rdtsc) and
       * the SuperHighResolutionTimer.  Each iteration will collect 8 bytes
       * of data but it is treated as only 1 byte of entropy.  The call to
       * ThreadSwitchWithDelay() will introduce additional variability into
       * the data returned by rdtsc.
       *
       * Applications can agument the seed material by adding additional
       * stuff with RAND_add() and should probably do so.
      */
   l = GetProcessSwitchCount();
   RAND_add(&l,sizeof(l),1);

   /* need to cast the void* to unsigned long here */
   l = (unsigned long)RunningProcess;
   RAND_add(&l,sizeof(l),1);

   for( i=2; i<ENTROPY_NEEDED; i++)
   {
#ifdef __MWERKS__
      asm
      {
         rdtsc
         mov tsc, eax
      }
#elif defined(__GNUC__) && __GNUC__>=2 && !defined(OPENSSL_NO_ASM) && !defined(O
      asm volatile("rdtsc":"=a"(tsc)::"edx");
#endif

      RAND_add(&tsc, sizeof(tsc), 1);

      l = GetSuperHighResolutionTimer();
      RAND_add(&l, sizeof(l), 0);

# if defined(NETWARE_LIBC)
      NXThreadYield();
# else /* NETWARE_CLIB */
      ThreadSwitchWithDelay();
# endif
   }

   return 1;
}

#endif
#endif /* ! codereview */
```

```
**********************************************************
    5686 Wed Aug 13 19:53:13 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/rand_os2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/rand_os2.c */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/rand.h>
#include "rand_lcl.h"

#ifdef OPENSSL_SYS_OS2
```



```
#define INCL_DOSPROCESS
#define INCL_DOSPROFILE
#define INCL_DOSMISC
#define INCL_DOSMODULEMGR
#include <os2.h>

#define   CMD_KI_RDCNT    (0x63)

typedef struct _CPUUTIL {
    ULONG ulTimeLow;            /* Low 32 bits of time stamp      */
    ULONG ulTimeHigh;           /* High 32 bits of time stamp     */
    ULONG ulIdleLow;            /* Low 32 bits of idle time       */
    ULONG ulIdleHigh;           /* High 32 bits of idle time      */
    ULONG ulBusyLow;            /* Low 32 bits of busy time       */
    ULONG ulBusyHigh;           /* High 32 bits of busy time      */
    ULONG ulIntrLow;            /* Low 32 bits of interrupt time  */
    ULONG ulIntrHigh;           /* High 32 bits of interrupt time */
} CPUUTIL;

#ifndef __KLIBC__
APIRET APIENTRY(*DosPerfSysCall) (ULONG ulCommand, ULONG ulParm1, ULONG ulParm2,
APIRET APIENTRY(*DosQuerySysState) (ULONG func, ULONG arg1, ULONG pid, ULONG _re
#endif
HMODULE hDoscalls = 0;

int RAND_poll(void)
{
    char failed_module[20];
    QWORD qwTime;
    ULONG SysVars[QSV_FOREGROUND_PROCESS];

    if (hDoscalls == 0) {
        ULONG rc = DosLoadModule(failed_module, sizeof(failed_module), "DOSCALLS

#ifndef __KLIBC__
        if (rc == 0) {
            rc = DosQueryProcAddr(hDoscalls, 976, NULL, (PFN *)&DosPerfSysCall);

            if (rc)
                DosPerfSysCall = NULL;

            rc = DosQueryProcAddr(hDoscalls, 368, NULL, (PFN *)&DosQuerySysState

            if (rc)
                DosQuerySysState = NULL;
        }
#endif
    }

    /* Sample the hi-res timer, runs at around 1.1 MHz */
    DosTmrQueryTime(&qwTime);
    RAND_add(&qwTime, sizeof(qwTime), 2);

    /* Sample a bunch of system variables, includes various process & memory sta
    DosQuerySysInfo(1, QSV_FOREGROUND_PROCESS, SysVars, sizeof(SysVars));
    RAND_add(SysVars, sizeof(SysVars), 4);

    /* If available, sample CPU registers that count at CPU MHz
     * Only fairly new CPUs (PPro & K6 onwards) & OS/2 versions support this
     */
    if (DosPerfSysCall) {
        CPUUTIL util;

        if (DosPerfSysCall(CMD_KI_RDCNT, (ULONG)&util, 0, 0) == 0) {
            RAND_add(&util, sizeof(util), 10);
        }
```

```
128         else {
129 #ifndef __KLIBC__
130             DosPerfSysCall = NULL;
131 #endif
132         }
133     }

135     /* DosQuerySysState() gives us a huge quantity of process, thread, memory &
136     if (DosQuerySysState) {
137         char *buffer = OPENSSL_malloc(256 * 1024);

139         if (DosQuerySysState(0x1F, 0, 0, 0, buffer, 256 * 1024) == 0) {
140             /* First 4 bytes in buffer is a pointer to the thread count
141              * there should be at least 1 byte of entropy per thread
142              */
143             RAND_add(buffer, 256 * 1024, **(ULONG **)buffer);
144         }

146         OPENSSL_free(buffer);
147         return 1;
148     }

150     return 0;
151 }

153 #endif /* OPENSSL_SYS_OS2 */
154 #endif /* ! codereview */
```

```
**********************************************************
   13482 Wed Aug 13 19:53:13 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/rand_unix.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/rand_unix.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#include <stdio.h>

#define USE_SOCKETS
#include "e_os.h"
#include "cryptlib.h"
#include <openssl/rand.h>
#include "rand_lcl.h"

#if !(defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_WIN32) || defined(OPEN

#include <sys/types.h>
#include <sys/time.h>
#include <sys/times.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
```

```
#if defined(OPENSSL_SYS_LINUX) /* should actually be available virtually everywh
# include <poll.h>
#endif
#include <limits.h>
#ifndef FD_SETSIZE
# define FD_SETSIZE (8*sizeof(fd_set))
#endif

#if defined(OPENSSL_SYS_VOS)

/* The following algorithm repeatedly samples the real-time clock
   (RTC) to generate a sequence of unpredictable data.  The algorithm
   relies upon the uneven execution speed of the code (due to factors
   such as cache misses, interrupts, bus activity, and scheduling) and
   upon the rather large relative difference between the speed of the
   clock and the rate at which it can be read.

   If this code is ported to an environment where execution speed is
   more constant or where the RTC ticks at a much slower rate, or the
   clock can be read with fewer instructions, it is likely that the
   results would be far more predictable.

   As a precaution, we generate 4 times the minimum required amount of
   seed data.  */

int RAND_poll(void)
{
        short int code;
        gid_t curr_gid;
        pid_t curr_pid;
        uid_t curr_uid;
        int i, k;
        struct timespec ts;
        unsigned char v;

#ifdef OPENSSL_SYS_VOS_HPPA
        long duration;
        extern void s$sleep (long *_duration, short int *_code);
#else
#ifdef OPENSSL_SYS_VOS_IA32
        long long duration;
        extern void s$sleep2 (long long *_duration, short int *_code);
#else
#error "Unsupported Platform."
#endif /* OPENSSL_SYS_VOS_IA32 */
#endif /* OPENSSL_SYS_VOS_HPPA */

        /* Seed with the gid, pid, and uid, to ensure *some*
           variation between different processes.  */

        curr_gid = getgid();
        RAND_add (&curr_gid, sizeof curr_gid, 1);
        curr_gid = 0;

        curr_pid = getpid();
        RAND_add (&curr_pid, sizeof curr_pid, 1);
        curr_pid = 0;

        curr_uid = getuid();
        RAND_add (&curr_uid, sizeof curr_uid, 1);
        curr_uid = 0;

        for (i=0; i<(ENTROPY_NEEDED*4); i++)
        {
                /* burn some cpu; hope for interrupts, cache
                   collisions, bus interference, etc.  */
```



```
                for (k=0; k<99; k++)
                        ts.tv_nsec = random ();

#ifdef OPENSSL_SYS_VOS_HPPA
                /* sleep for 1/1024 of a second (976 us).  */
                duration = 1;
                s$sleep (&duration, &code);
#else
#ifdef OPENSSL_SYS_VOS_IA32
                /* sleep for 1/65536 of a second (15 us).  */
                duration = 1;
                s$sleep2 (&duration, &code);
#endif /* OPENSSL_SYS_VOS_IA32 */
#endif /* OPENSSL_SYS_VOS_HPPA */

                /* get wall clock time.  */
                clock_gettime (CLOCK_REALTIME, &ts);

                /* take 8 bits */
                v = (unsigned char) (ts.tv_nsec % 256);
                RAND_add (&v, sizeof v, 1);
                v = 0;
        }
        return 1;
}
#elif defined __OpenBSD__
int RAND_poll(void)
{
        u_int32_t rnd = 0, i;
        unsigned char buf[ENTROPY_NEEDED];

        for (i = 0; i < sizeof(buf); i++) {
                if (i % 4 == 0)
                        rnd = arc4random();
                buf[i] = rnd;
                rnd >>= 8;
        }
        RAND_add(buf, sizeof(buf), ENTROPY_NEEDED);
        memset(buf, 0, sizeof(buf));

        return 1;
}
#else /* !defined(__OpenBSD__) */
int RAND_poll(void)
{
        unsigned long l;
        pid_t curr_pid = getpid();
#if defined(DEVRANDOM) || defined(DEVRANDOM_EGD)
        unsigned char tmpbuf[ENTROPY_NEEDED];
        int n = 0;
#endif
#ifdef DEVRANDOM
        static const char *randomfiles[] = { DEVRANDOM };
        struct stat randomstats[sizeof(randomfiles)/sizeof(randomfiles[0])];
        int fd;
        unsigned int i;
#endif
#ifdef DEVRANDOM_EGD
        static const char *egdsockets[] = { DEVRANDOM_EGD, NULL };
        const char **egdsocket = NULL;
#endif

#ifdef DEVRANDOM
        memset(randomstats,0,sizeof(randomstats));
        /* Use a random entropy pool device. Linux, FreeBSD and OpenBSD
         * have this. Use /dev/urandom if you can as /dev/random may block
```

```
260          * if it runs out of random entries.  */

262         for (i = 0; (i < sizeof(randomfiles)/sizeof(randomfiles[0])) &&
263                         (n < ENTROPY_NEEDED); i++)
264                 {
265                 if ((fd = open(randomfiles[i], O_RDONLY
266 #ifdef O_NONBLOCK
267                         |O_NONBLOCK
268 #endif
269 #ifdef O_BINARY
270                         |O_BINARY
271 #endif
272 #ifdef O_NOCTTY /* If it happens to be a TTY (god forbid), do not make it
273                    our controlling tty */
274                         |O_NOCTTY
275 #endif
276                         )) >= 0)
277                         {
278                         int usec = 10*1000; /* spend 10ms on each file */
279                         int r;
280                         unsigned int j;
281                         struct stat *st=&randomstats[i];

283                         /* Avoid using same input... Used to be O_NOFOLLOW
284                          * above, but it's not universally appropriate... */
285                         if (fstat(fd,st) != 0)  { close(fd); continue; }
286                         for (j=0;j<i;j++)
287                                 {
288                                 if (randomstats[j].st_ino==st->st_ino &&
289                                     randomstats[j].st_dev==st->st_dev)
290                                         break;
291                                 }
292                         if (j<i)                { close(fd); continue; }

294                         do
295                                 {
296                                 int try_read = 0;

298 #if defined(OPENSSL_SYS_BEOS_R5)
299                                 /* select() is broken in BeOS R5, so we simply
300                                  *  try to read something and snooze if we could
301                                 try_read = 1;

303 #elif defined(OPENSSL_SYS_LINUX)
304                                 /* use poll() */
305                                 struct pollfd pset;

307                                 pset.fd = fd;
308                                 pset.events = POLLIN;
309                                 pset.revents = 0;

311                                 if (poll(&pset, 1, usec / 1000) < 0)
312                                         usec = 0;
313                                 else
314                                         try_read = (pset.revents & POLLIN) != 0;

316 #else
317                                 /* use select() */
318                                 fd_set fset;
319                                 struct timeval t;

321                                 t.tv_sec = 0;
322                                 t.tv_usec = usec;

324                                 if (FD_SETSIZE > 0 && (unsigned)fd >= FD_SETSIZE
325                                         {
```



```
326                                         /* can't use select, so just try to read
327                                         try_read = 1;
328                                         }
329                                 else
330                                         {
331                                         FD_ZERO(&fset);
332                                         FD_SET(fd, &fset);

334                                         if (select(fd+1,&fset,NULL,NULL,&t) >= 0
335                                                 {
336                                                 usec = t.tv_usec;
337                                                 if (FD_ISSET(fd, &fset))
338                                                         try_read = 1;
339                                                 }
340                                         else
341                                                 usec = 0;
342                                         }
343 #endif

345                                 if (try_read)
346                                         {
347                                         r = read(fd,(unsigned char *)tmpbuf+n, E
348                                         if (r > 0)
349                                                 n += r;
350 #if defined(OPENSSL_SYS_BEOS_R5)
351                                         if (r == 0)
352                                                 snooze(t.tv_usec);
353 #endif
354                                         }
355                                 else
356                                         r = -1;

358                                 /* Some Unixen will update t in select(), some
359                                    won't.  For those who won't, or if we
360                                    didn't use select() in the first place,
361                                    give up here, otherwise, we will do
362                                    this once again for the remaining
363                                    time. */
364                                 if (usec == 10*1000)
365                                         usec = 0;
366                                 }
367                         while ((r > 0 ||
368                                 (errno == EINTR || errno == EAGAIN)) && usec != 0

370                         close(fd);
371                         }
372                 }
373 #endif /* defined(DEVRANDOM) */

375 #ifdef DEVRANDOM_EGD
376         /* Use an EGD socket to read entropy from an EGD or PRNGD entropy
377          * collecting daemon. */

379         for (egdsocket = egdsockets; *egdsocket && n < ENTROPY_NEEDED; egdsocket
380                 {
381                 int r;

383                 r = RAND_query_egd_bytes(*egdsocket, (unsigned char *)tmpbuf+n,
384                                          ENTROPY_NEEDED-n);
385                 if (r > 0)
386                         n += r;
387                 }
388 #endif /* defined(DEVRANDOM_EGD) */

390 #if defined(DEVRANDOM) || defined(DEVRANDOM_EGD)
391         if (n > 0)
```

```
392                 {
393                 RAND_add(tmpbuf,sizeof tmpbuf,(double)n);
394                 OPENSSL_cleanse(tmpbuf,n);
395                 }
396 #endif

398         /* put in some default random data, we need more than just this */
399         l=curr_pid;
400         RAND_add(&l,sizeof(l),0.0);
401         l=getuid();
402         RAND_add(&l,sizeof(l),0.0);

404         l=time(NULL);
405         RAND_add(&l,sizeof(l),0.0);

407 #if defined(OPENSSL_SYS_BEOS)
408         {
409         system_info sysInfo;
410         get_system_info(&sysInfo);
411         RAND_add(&sysInfo,sizeof(sysInfo),0);
412         }
413 #endif

415 #if defined(DEVRANDOM) || defined(DEVRANDOM_EGD)
416         return 1;
417 #else
418         return 0;
419 #endif
420 }

422 #endif /* defined(__OpenBSD__) */
423 #endif /* !(defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_WIN32) || define


426 #if defined(OPENSSL_SYS_VXWORKS)
427 int RAND_poll(void)
428         {
429         return 0;
430         }
431 #endif
432 #endif /* ! codereview */
```

```
**********************************************************
   26769 Wed Aug 13 19:53:13 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/rand_win.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/rand_win.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/rand.h>
#include "rand_lcl.h"

#if defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_WIN32)
#include <windows.h>
#ifndef _WIN32_WINNT
# define _WIN32_WINNT 0x0400
#endif
#include <wincrypt.h>
#include <tlhelp32.h>

/* Limit the time spent walking through the heap, processes, threads and modules
   a maximum of 1000 miliseconds each, unless CryptoGenRandom failed */
#define MAXDELAY 1000
```

```
/* Intel hardware RNG CSP -- available from
 * http://developer.intel.com/design/security/rng/redist_license.htm
 */
#define PROV_INTEL_SEC 22
#define INTEL_DEF_PROV L"Intel Hardware Cryptographic Service Provider"

static void readtimer(void);
static void readscreen(void);

/* It appears like CURSORINFO, PCURSORINFO and LPCURSORINFO are only defined
   when WINVER is 0x0500 and up, which currently only happens on Win2000.
   Unfortunately, those are typedefs, so they're a little bit difficult to
   detect properly.  On the other hand, the macro CURSOR_SHOWING is defined
   within the same conditional, so it can be use to detect the absence of said
   typedefs. */

#ifndef CURSOR_SHOWING
/*
 * Information about the global cursor.
 */
typedef struct tagCURSORINFO
{
    DWORD   cbSize;
    DWORD   flags;
    HCURSOR hCursor;
    POINT   ptScreenPos;
} CURSORINFO, *PCURSORINFO, *LPCURSORINFO;

#define CURSOR_SHOWING     0x00000001
#endif /* CURSOR_SHOWING */

#if !defined(OPENSSL_SYS_WINCE)
typedef BOOL (WINAPI *CRYPTACQUIRECONTEXTW)(HCRYPTPROV *, LPCWSTR, LPCWSTR,
                                            DWORD, DWORD);
typedef BOOL (WINAPI *CRYPTGENRANDOM)(HCRYPTPROV, DWORD, BYTE *);
typedef BOOL (WINAPI *CRYPTRELEASECONTEXT)(HCRYPTPROV, DWORD);

typedef HWND (WINAPI *GETFOREGROUNDWINDOW)(VOID);
typedef BOOL (WINAPI *GETCURSORINFO)(PCURSORINFO);
typedef DWORD (WINAPI *GETQUEUESTATUS)(UINT);

typedef HANDLE (WINAPI *CREATETOOLHELP32SNAPSHOT)(DWORD, DWORD);
typedef BOOL (WINAPI *CLOSETOOLHELP32SNAPSHOT)(HANDLE);
typedef BOOL (WINAPI *HEAP32FIRST)(LPHEAPENTRY32, DWORD, size_t);
typedef BOOL (WINAPI *HEAP32NEXT)(LPHEAPENTRY32);
typedef BOOL (WINAPI *HEAP32LIST)(HANDLE, LPHEAPLIST32);
typedef BOOL (WINAPI *PROCESS32)(HANDLE, LPPROCESSENTRY32);
typedef BOOL (WINAPI *THREAD32)(HANDLE, LPTHREADENTRY32);
typedef BOOL (WINAPI *MODULE32)(HANDLE, LPMODULEENTRY32);

#include <lmcons.h>
#include <lmstats.h>
#if 1 /* The NET API is Unicode only.  It requires the use of the UNICODE
       * macro.  When UNICODE is defined LPTSTR becomes LPWSTR.  LMSTR was
       * was added to the Platform SDK to allow the NET API to be used in
       * non-Unicode applications provided that Unicode strings were still
       * used for input.  LMSTR is defined as LPWSTR.
       */
typedef NET_API_STATUS (NET_API_FUNCTION * NETSTATGET)
        (LPWSTR, LPWSTR, DWORD, DWORD, LPBYTE*);
typedef NET_API_STATUS (NET_API_FUNCTION * NETFREE)(LPBYTE);
#endif /* 1 */
#endif /* !OPENSSL_SYS_WINCE */

int RAND_poll(void)
{
```



```
        MEMORYSTATUS m;
        HCRYPTPROV hProvider = 0;
        DWORD w;
        int good = 0;

        /* Determine the OS version we are on so we can turn off things
         * that do not work properly.
         */
        OSVERSIONINFO osverinfo ;
        osverinfo.dwOSVersionInfoSize = sizeof(OSVERSIONINFO) ;
        GetVersionEx( &osverinfo ) ;

#if defined(OPENSSL_SYS_WINCE)
# if defined(_WIN32_WCE) && _WIN32_WCE>=300
/* Even though MSDN says _WIN32_WCE>=210, it doesn't seem to be available
 * in commonly available implementations prior 300... */
        {
        BYTE buf[64];
        /* poll the CryptoAPI PRNG */
        /* The CryptoAPI returns sizeof(buf) bytes of randomness */
        if (CryptAcquireContextW(&hProvider, NULL, NULL, PROV_RSA_FULL,
                                 CRYPT_VERIFYCONTEXT))
                {
                if (CryptGenRandom(hProvider, sizeof(buf), buf))
                        RAND_add(buf, sizeof(buf), sizeof(buf));
                CryptReleaseContext(hProvider, 0);
                }
        }
# endif
#else   /* OPENSSL_SYS_WINCE */
        /*
         * None of below libraries are present on Windows CE, which is
         * why we #ifndef the whole section. This also excuses us from
         * handling the GetProcAddress issue. The trouble is that in
         * real Win32 API GetProcAddress is available in ANSI flavor
         * only. In WinCE on the other hand GetProcAddress is a macro
         * most commonly defined as GetProcAddressW, which accepts
         * Unicode argument. If we were to call GetProcAddress under
         * WinCE, I'd recommend to either redefine GetProcAddress as
         * GetProcAddressA (there seem to be one in common CE spec) or
         * implement own shim routine, which would accept ANSI argument
         * and expand it to Unicode.
         */
        {
        /* load functions dynamically - not available on all systems */
        HMODULE advapi = LoadLibrary(TEXT("ADVAPI32.DLL"));
        HMODULE kernel = LoadLibrary(TEXT("KERNEL32.DLL"));
        HMODULE user = NULL;
        HMODULE netapi = LoadLibrary(TEXT("NETAPI32.DLL"));
        CRYPTACQUIRECONTEXTW acquire = NULL;
        CRYPTGENRANDOM gen = NULL;
        CRYPTRELEASECONTEXT release = NULL;
        NETSTATGET netstatget = NULL;
        NETFREE netfree = NULL;
        BYTE buf[64];

        if (netapi)
                {
                netstatget = (NETSTATGET) GetProcAddress(netapi,"NetStatisticsGe
                netfree = (NETFREE) GetProcAddress(netapi,"NetApiBufferFree");
                }

        if (netstatget && netfree)
                {
                LPBYTE outbuf;
                /* NetStatisticsGet() is a Unicode only function
```

```
260                  * STAT_WORKSTATION_0 contains 45 fields and STAT_SERVER_0
261                  * contains 17 fields.  We treat each field as a source of
262                  * one byte of entropy.
263                  */
264
265                 if (netstatget(NULL, L"LanmanWorkstation", 0, 0, &outbuf) == 0)
266                         {
267                         RAND_add(outbuf, sizeof(STAT_WORKSTATION_0), 45);
268                         netfree(outbuf);
269                         }
270                 if (netstatget(NULL, L"LanmanServer", 0, 0, &outbuf) == 0)
271                         {
272                         RAND_add(outbuf, sizeof(STAT_SERVER_0), 17);
273                         netfree(outbuf);
274                         }
275                 }
276
277         if (netapi)
278                 FreeLibrary(netapi);
279
280         /* It appears like this can cause an exception deep within ADVAPI32.DLL
281          * at random times on Windows 2000.  Reported by Jeffrey Altman.
282          * Only use it on NT.
283          */
284         /* Wolfgang Marczy <WMarczy@topcall.co.at> reports that
285          * the RegQueryValueEx call below can hang on NT4.0 (SP6).
286          * So we don't use this at all for now. */
287 #if 0
288         if ( osverinfo.dwPlatformId == VER_PLATFORM_WIN32_NT &&
289                 osverinfo.dwMajorVersion < 5)
290                 {
291                 /* Read Performance Statistics from NT/2000 registry
292                  * The size of the performance data can vary from call
293                  * to call so we must guess the size of the buffer to use
294                  * and increase its size if we get an ERROR_MORE_DATA
295                  * return instead of ERROR_SUCCESS.
296                  */
297                 LONG   rc=ERROR_MORE_DATA;
298                 char * buf=NULL;
299                 DWORD bufsz=0;
300                 DWORD length;
301
302                 while (rc == ERROR_MORE_DATA)
303                         {
304                         buf = realloc(buf,bufsz+8192);
305                         if (!buf)
306                                 break;
307                         bufsz += 8192;
308
309                         length = bufsz;
310                         rc = RegQueryValueEx(HKEY_PERFORMANCE_DATA, TEXT("Global
311                                 NULL, NULL, buf, &length);
312                         }
313                 if (rc == ERROR_SUCCESS)
314                         {
315                         /* For entropy count assume only least significant
316                          * byte of each DWORD is random.
317                          */
318                         RAND_add(&length, sizeof(length), 0);
319                         RAND_add(buf, length, length / 4.0);
320
321                         /* Close the Registry Key to allow Windows to cleanup/cl
322                          * the open handle
323                          * Note: The 'HKEY_PERFORMANCE_DATA' key is implicitly o
324                          *       when the RegQueryValueEx above is done.  Howeve
325                          *       it is not explicitly closed, it can cause disk
```



```
326                          *       partition manipulation problems.
327                          */
328                         RegCloseKey(HKEY_PERFORMANCE_DATA);
329                         }
330                 if (buf)
331                         free(buf);
332                 }
333 #endif
334
335         if (advapi)
336                 {
337                 /*
338                  * If it's available, then it's available in both ANSI
339                  * and UNICODE flavors even in Win9x, documentation says.
340                  * We favor Unicode...
341                  */
342                 acquire = (CRYPTACQUIRECONTEXTW) GetProcAddress(advapi,
343                         "CryptAcquireContextW");
344                 gen = (CRYPTGENRANDOM) GetProcAddress(advapi,
345                         "CryptGenRandom");
346                 release = (CRYPTRELEASECONTEXT) GetProcAddress(advapi,
347                         "CryptReleaseContext");
348                 }
349
350         if (acquire && gen && release)
351                 {
352                 /* poll the CryptoAPI PRNG */
353                 /* The CryptoAPI returns sizeof(buf) bytes of randomness */
354                 if (acquire(&hProvider, NULL, NULL, PROV_RSA_FULL,
355                         CRYPT_VERIFYCONTEXT))
356                         {
357                         if (gen(hProvider, sizeof(buf), buf) != 0)
358                                 {
359                                 RAND_add(buf, sizeof(buf), 0);
360                                 good = 1;
361 #if 0
362                                 printf("randomness from PROV_RSA_FULL\n");
363 #endif
364                                 }
365                         release(hProvider, 0);
366                         }
367
368                 /* poll the Pentium PRG with CryptoAPI */
369                 if (acquire(&hProvider, 0, INTEL_DEF_PROV, PROV_INTEL_SEC, 0))
370                         {
371                         if (gen(hProvider, sizeof(buf), buf) != 0)
372                                 {
373                                 RAND_add(buf, sizeof(buf), sizeof(buf));
374                                 good = 1;
375 #if 0
376                                 printf("randomness from PROV_INTEL_SEC\n");
377 #endif
378                                 }
379                         release(hProvider, 0);
380                         }
381                 }
382
383         if (advapi)
384                 FreeLibrary(advapi);
385
386         if ((osverinfo.dwPlatformId != VER_PLATFORM_WIN32_NT ||
387              !OPENSSL_isservice()) &&
388             (user = LoadLibrary(TEXT("USER32.DLL"))))
389                 {
390                 GETCURSORINFO cursor;
391                 GETFOREGROUNDWINDOW win;
```

```
392                 GETQUEUESTATUS queue;
393
394                 win = (GETFOREGROUNDWINDOW) GetProcAddress(user, "GetForegroundW
395                 cursor = (GETCURSORINFO) GetProcAddress(user, "GetCursorInfo");
396                 queue = (GETQUEUESTATUS) GetProcAddress(user, "GetQueueStatus");
397
398                 if (win)
399                         {
400                         /* window handle */
401                         HWND h = win();
402                         RAND_add(&h, sizeof(h), 0);
403                         }
404                 if (cursor)
405                         {
406                         /* unfortunately, its not safe to call GetCursorInfo()
407                          * on NT4 even though it exists in SP3 (or SP6) and
408                          * higher.
409                          */
410                         if ( osverinfo.dwPlatformId == VER_PLATFORM_WIN32_NT &&
411                                 osverinfo.dwMajorVersion < 5)
412                                 cursor = 0;
413                         }
414                 if (cursor)
415                         {
416                         /* cursor position */
417                         /* assume 2 bytes of entropy */
418                         CURSORINFO ci;
419                         ci.cbSize = sizeof(CURSORINFO);
420                         if (cursor(&ci))
421                                 RAND_add(&ci, ci.cbSize, 2);
422                         }
423
424                 if (queue)
425                         {
426                         /* message queue status */
427                         /* assume 1 byte of entropy */
428                         w = queue(QS_ALLEVENTS);
429                         RAND_add(&w, sizeof(w), 1);
430                         }
431
432                 FreeLibrary(user);
433                 }
434
435         /* Toolhelp32 snapshot: enumerate processes, threads, modules and heap
436          * http://msdn.microsoft.com/library/psdk/winbase/toolhelp_5pfd.htm
437          * (Win 9x and 2000 only, not available on NT)
438          *
439          * This seeding method was proposed in Peter Gutmann, Software
440          * Generation of Practically Strong Random Numbers,
441          * http://www.usenix.org/publications/library/proceedings/sec98/gutmann.
442          * revised version at http://www.cryptoengines.com/~peter/06_random.pdf
443          * (The assignment of entropy estimates below is arbitrary, but based
444          * on Peter's analysis the full poll appears to be safe. Additional
445          * interactive seeding is encouraged.)
446          */
447
448         if (kernel)
449                 {
450                 CREATETOOLHELP32SNAPSHOT snap;
451                 CLOSETOOLHELP32SNAPSHOT close_snap;
452                 HANDLE handle;
453
454                 HEAP32FIRST heap_first;
455                 HEAP32NEXT heap_next;
456                 HEAP32LIST heaplist_first, heaplist_next;
457                 PROCESS32 process_first, process_next;
```



```
458                 THREAD32 thread_first, thread_next;
459                 MODULE32 module_first, module_next;
460
461                 HEAPLIST32 hlist;
462                 HEAPENTRY32 hentry;
463                 PROCESSENTRY32 p;
464                 THREADENTRY32 t;
465                 MODULEENTRY32 m;
466                 DWORD starttime = 0;
467
468                 snap = (CREATETOOLHELP32SNAPSHOT)
469                         GetProcAddress(kernel, "CreateToolhelp32Snapshot");
470                 close_snap = (CLOSETOOLHELP32SNAPSHOT)
471                         GetProcAddress(kernel, "CloseToolhelp32Snapshot");
472                 heap_first = (HEAP32FIRST) GetProcAddress(kernel, "Heap32First")
473                 heap_next = (HEAP32NEXT) GetProcAddress(kernel, "Heap32Next");
474                 heaplist_first = (HEAP32LIST) GetProcAddress(kernel, "Heap32List
475                 heaplist_next = (HEAP32LIST) GetProcAddress(kernel, "Heap32ListN
476                 process_first = (PROCESS32) GetProcAddress(kernel, "Process32Fir
477                 process_next = (PROCESS32) GetProcAddress(kernel, "Process32Next
478                 thread_first = (THREAD32) GetProcAddress(kernel, "Thread32First"
479                 thread_next = (THREAD32) GetProcAddress(kernel, "Thread32Next");
480                 module_first = (MODULE32) GetProcAddress(kernel, "Module32First"
481                 module_next = (MODULE32) GetProcAddress(kernel, "Module32Next");
482
483                 if (snap && heap_first && heap_next && heaplist_first &&
484                         heaplist_next && process_first && process_next &&
485                         thread_first && thread_next && module_first &&
486                         module_next && (handle = snap(TH32CS_SNAPALL,0))
487                         != INVALID_HANDLE_VALUE)
488                         {
489                         /* heap list and heap walking */
490                         /* HEAPLIST32 contains 3 fields that will change with
491                          * each entry.  Consider each field a source of 1 byte
492                          * of entropy.
493                          * HEAPENTRY32 contains 5 fields that will change with
494                          * each entry.  Consider each field a source of 1 byte
495                          * of entropy.
496                          */
497                         ZeroMemory(&hlist, sizeof(HEAPLIST32));
498                         hlist.dwSize = sizeof(HEAPLIST32);
499                         if (good) starttime = GetTickCount();
500 #ifdef _MSC_VER
501                         if (heaplist_first(handle, &hlist))
502                                 {
503                                 /*
504                                    following discussion on dev ML, exception on
505                                    platform) is theoretically of unknown origin;
506                                    loop here when this theoretical case occurs;
507                                    the expected (MSDN documented) exception-thro
508                                    Heap32Next() on WinCE.
509
510                                    based on patch in original message by Tanguy
511                                    Subject: RAND_poll() and CreateToolhelp32Snap
512                                 */
513                                 int ex_cnt_limit = 42;
514                                 do
515                                         {
516                                         RAND_add(&hlist, hlist.dwSize, 3);
517                                         __try
518                                                 {
519                                                 ZeroMemory(&hentry, sizeof(HEAPE
520                                         hentry.dwSize = sizeof(HEAPENTRY32);
521                                         if (heap_first(&hentry,
522                                                 hlist.th32ProcessID,
523                                                 hlist.th32HeapID))
```

```
524                                             {
525                                             int entrycnt = 80;
526                                             do
527                                                     RAND_add(&hentry,
528                                                             hentry.dwSize, 5
529                                             while (heap_next(&hentry)
530                                             && (!good || (GetTickCount()-sta
531                                                     && --entrycnt > 0);
532                                             }
533                                             }
534                                     __except (EXCEPTION_EXECUTE_HANDLER)
535                                             {
536                                                     /* ignore access violati
537                                                     ex_cnt_limit--;
538                                             }
539                                     } while (heaplist_next(handle, &hlist)
540                                             && (!good || (GetTickCount()-sta
541                                             && ex_cnt_limit > 0);
542                             }

544 #else
545                         if (heaplist_first(handle, &hlist))
546                                 {
547                                 do
548                                         {
549                                         RAND_add(&hlist, hlist.dwSize, 3);
550                                         hentry.dwSize = sizeof(HEAPENTRY32);
551                                         if (heap_first(&hentry,
552                                                 hlist.th32ProcessID,
553                                                 hlist.th32HeapID))
554                                                 {
555                                                 int entrycnt = 80;
556                                                 do
557                                                         RAND_add(&hentry,
558                                                                 hentry.dwSize, 5
559                                                 while (heap_next(&hentry)
560                                                         && --entrycnt > 0);
561                                                 }
562                                         } while (heaplist_next(handle, &hlist)
563                                                 && (!good || (GetTickCount()-sta
564                                 }
565 #endif

567                         /* process walking */
568                         /* PROCESSENTRY32 contains 9 fields that will change
569                          * with each entry.  Consider each field a source of
570                          * 1 byte of entropy.
571                          */
572                         p.dwSize = sizeof(PROCESSENTRY32);

574                         if (good) starttime = GetTickCount();
575                         if (process_first(handle, &p))
576                                 do
577                                         RAND_add(&p, p.dwSize, 9);
578                                 while (process_next(handle, &p) && (!good || (Ge

580                         /* thread walking */
581                         /* THREADENTRY32 contains 6 fields that will change
582                          * with each entry.  Consider each field a source of
583                          * 1 byte of entropy.
584                          */
585                         t.dwSize = sizeof(THREADENTRY32);
586                         if (good) starttime = GetTickCount();
587                         if (thread_first(handle, &t))
588                                 do
589                                         RAND_add(&t, t.dwSize, 6);
```



```
590                                         while (thread_next(handle, &t) && (!good || (Get

592                         /* module walking */
593                         /* MODULEENTRY32 contains 9 fields that will change
594                          * with each entry.  Consider each field a source of
595                          * 1 byte of entropy.
596                          */
597                         m.dwSize = sizeof(MODULEENTRY32);
598                         if (good) starttime = GetTickCount();
599                         if (module_first(handle, &m))
600                                 do
601                                         RAND_add(&m, m.dwSize, 9);
602                                 while (module_next(handle, &m)
603                                         && (!good || (GetTickCount()-sta
604                         if (close_snap)
605                                 close_snap(handle);
606                         else
607                                 CloseHandle(handle);

609                         }

611                 FreeLibrary(kernel);
612                 }
613         }
614 #endif /* !OPENSSL_SYS_WINCE */

616         /* timer data */
617         readtimer();

619         /* memory usage statistics */
620         GlobalMemoryStatus(&m);
621         RAND_add(&m, sizeof(m), 1);

623         /* process ID */
624         w = GetCurrentProcessId();
625         RAND_add(&w, sizeof(w), 1);

627 #if 0
628         printf("Exiting RAND_poll\n");
629 #endif

631         return(1);
632 }

634 int RAND_event(UINT iMsg, WPARAM wParam, LPARAM lParam)
635         {
636         double add_entropy=0;

638         switch (iMsg)
639                 {
640         case WM_KEYDOWN:
641                         {
642                         static WPARAM key;
643                         if (key != wParam)
644                                 add_entropy = 0.05;
645                         key = wParam;
646                         }
647                         break;
648         case WM_MOUSEMOVE:
649                         {
650                         static int lastx,lasty,lastdx,lastdy;
651                         int x,y,dx,dy;

653                         x=LOWORD(lParam);
654                         y=HIWORD(lParam);
655                         dx=lastx-x;
```

```
656                         dy=lasty-y;
657                         if (dx != 0 && dy != 0 && dx-lastdx != 0 && dy-lastdy !=
658                                 add_entropy=.2;
659                         lastx=x, lasty=y;
660                         lastdx=dx, lastdy=dy;
661                         }
662                 break;
663                 }

665         readtimer();
666         RAND_add(&iMsg, sizeof(iMsg), add_entropy);
667         RAND_add(&wParam, sizeof(wParam), 0);
668         RAND_add(&lParam, sizeof(lParam), 0);

670         return (RAND_status());
671         }


674 void RAND_screen(void) /* function available for backward compatibility */
675 {
676         RAND_poll();
677         readscreen();
678 }


681 /* feed timing information to the PRNG */
682 static void readtimer(void)
683 {
684         DWORD w;
685         LARGE_INTEGER l;
686         static int have_perfc = 1;
687 #if defined(_MSC_VER) && defined(_M_X86)
688         static int have_tsc = 1;
689         DWORD cyclecount;

691         if (have_tsc) {
692           __try {
693             __asm {
694               _emit 0x0f
695               _emit 0x31
696               mov cyclecount, eax
697               }
698             RAND_add(&cyclecount, sizeof(cyclecount), 1);
699           } __except(EXCEPTION_EXECUTE_HANDLER) {
700             have_tsc = 0;
701           }
702         }
703 #else
704 # define have_tsc 0
705 #endif

707         if (have_perfc) {
708           if (QueryPerformanceCounter(&l) == 0)
709             have_perfc = 0;
710           else
711             RAND_add(&l, sizeof(l), 0);
712         }

714         if (!have_tsc && !have_perfc) {
715           w = GetTickCount();
716           RAND_add(&w, sizeof(w), 0);
717         }
718 }

720 /* feed screen contents to PRNG */
721 /*****************************************************************************
```



```
722  *
723  * Created 960901 by Gertjan van Oosten, gertjan@West.NL, West Consulting B.V.
724  *
725  * Code adapted from
726  * <URL:http://support.microsoft.com/default.aspx?scid=kb;[LN];97193>;
727  * the original copyright message is:
728  *
729  *   (C) Copyright Microsoft Corp. 1993.  All rights reserved.
730  *
731  *   You have a royalty-free right to use, modify, reproduce and
732  *   distribute the Sample Files (and/or any modified version) in
733  *   any way you find useful, provided that you agree that
734  *   Microsoft has no warranty obligations or liability for any
735  *   Sample Application Files which are modified.
736  */

738 static void readscreen(void)
739 {
740 #if !defined(OPENSSL_SYS_WINCE) && !defined(OPENSSL_SYS_WIN32_CYGWIN)
741   HDC           hScrDC;         /* screen DC */
742   HDC           hMemDC;         /* memory DC */
743   HBITMAP       hBitmap;        /* handle for our bitmap */
744   HBITMAP       hOldBitmap;     /* handle for previous bitmap */
745   BITMAP        bm;             /* bitmap properties */
746   unsigned int  size;           /* size of bitmap */
747   char          *bmbits;        /* contents of bitmap */
748   int           w;              /* screen width */
749   int           h;              /* screen height */
750   int           y;              /* y-coordinate of screen lines to grab */
751   int           n = 16;         /* number of screen lines to grab at a time */

753   if (check_winnt() && OPENSSL_isservice()>0)
754     return;

756   /* Create a screen DC and a memory DC compatible to screen DC */
757   hScrDC = CreateDC(TEXT("DISPLAY"), NULL, NULL, NULL);
758   hMemDC = CreateCompatibleDC(hScrDC);

760   /* Get screen resolution */
761   w = GetDeviceCaps(hScrDC, HORZRES);
762   h = GetDeviceCaps(hScrDC, VERTRES);

764   /* Create a bitmap compatible with the screen DC */
765   hBitmap = CreateCompatibleBitmap(hScrDC, w, n);

767   /* Select new bitmap into memory DC */
768   hOldBitmap = SelectObject(hMemDC, hBitmap);

770   /* Get bitmap properties */
771   GetObject(hBitmap, sizeof(BITMAP), (LPSTR)&bm);
772   size = (unsigned int)bm.bmWidthBytes * bm.bmHeight * bm.bmPlanes;

774   bmbits = OPENSSL_malloc(size);
775   if (bmbits) {
776     /* Now go through the whole screen, repeatedly grabbing n lines */
777     for (y = 0; y < h-n; y += n)
778         {
779         unsigned char md[MD_DIGEST_LENGTH];

781         /* Bitblt screen DC to memory DC */
782         BitBlt(hMemDC, 0, 0, w, n, hScrDC, 0, y, SRCCOPY);

784         /* Copy bitmap bits from memory DC to bmbits */
785         GetBitmapBits(hBitmap, size, bmbits);

787         /* Get the hash of the bitmap */
```

```
788         MD(bmbits,size,md);

790         /* Seed the random generator with the hash value */
791         RAND_add(md, MD_DIGEST_LENGTH, 0);
792         }

794     OPENSSL_free(bmbits);
795   }

797   /* Select old bitmap back into memory DC */
798   hBitmap = SelectObject(hMemDC, hOldBitmap);

800   /* Clean up */
801   DeleteObject(hBitmap);
802   DeleteDC(hMemDC);
803   DeleteDC(hScrDC);
804 #endif /* !OPENSSL_SYS_WINCE */
805 }

807 #endif
808 #endif /* ! codereview */
```

```
**********************************************************
    9940 Wed Aug 13 19:53:13 2014
new/usr/src/lib/openssl/libsunw_crypto/rand/randfile.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rand/randfile.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* We need to define this to get macros like S_IFBLK and S_IFCHR */
#if !defined(OPENSSL_SYS_VXWORKS)
#define _XOPEN_SOURCE 500
```



```
#endif

#include <errno.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "e_os.h"
#include <openssl/crypto.h>
#include <openssl/rand.h>
#include <openssl/buffer.h>

#ifdef OPENSSL_SYS_VMS
#include <unixio.h>
#endif
#ifndef NO_SYS_TYPES_H
# include <sys/types.h>
#endif
#ifndef OPENSSL_NO_POSIX_IO
# include <sys/stat.h>
# include <fcntl.h>
#endif

#ifdef _WIN32
#define stat    _stat
#define chmod   _chmod
#define open    _open
#define fdopen  _fdopen
#endif

#undef BUFSIZE
#define BUFSIZE 1024
#define RAND_DATA 1024

#ifdef OPENSSL_SYS_VMS
/* This declaration is a nasty hack to get around vms' extension to fopen
 * for passing in sharing options being disabled by our /STANDARD=ANSI89 */
static FILE *(*const vms_fopen)(const char *, const char *, ...) =
    (FILE *(*)(const char *, const char *, ...))fopen;
#define VMS_OPEN_ATTRS "shr=get,put,upd,del","ctx=bin,stm","rfm=stm","rat=none",
#endif

/* #define RFILE ".rnd" - defined in ../../e_os.h */

/* Note that these functions are intended for seed files only.
 * Entropy devices and EGD sockets are handled in rand_unix.c */

int RAND_load_file(const char *file, long bytes)
        {
        /* If bytes >= 0, read up to 'bytes' bytes.
         * if bytes == -1, read complete file. */

        MS_STATIC unsigned char buf[BUFSIZE];
#ifndef OPENSSL_NO_POSIX_IO
        struct stat sb;
#endif
        int i,ret=0,n;
        FILE *in;

        if (file == NULL) return(0);

#ifndef OPENSSL_NO_POSIX_IO
#ifdef PURIFY
        /* struct stat can have padding and unused fields that may not be
         * initialized in the call to stat(). We need to clear the entire
         * structure before calling RAND_add() to avoid complaints from
```

```
128          * applications such as Valgrind.
129          */
130         memset(&sb, 0, sizeof(sb));
131 #endif
132         if (stat(file,&sb) < 0) return(0);
133         RAND_add(&sb,sizeof(sb),0.0);
134 #endif
135         if (bytes == 0) return(ret);

137 #ifdef OPENSSL_SYS_VMS
138         in=vms_fopen(file,"rb",VMS_OPEN_ATTRS);
139 #else
140         in=fopen(file,"rb");
141 #endif
142         if (in == NULL) goto err;
143 #if defined(S_IFBLK) && defined(S_IFCHR) && !defined(OPENSSL_NO_POSIX_IO)
144         if (sb.st_mode & (S_IFBLK | S_IFCHR)) {
145           /* this file is a device. we don't want read an infinite number
146            * of bytes from a random device, nor do we want to use buffered
147            * I/O because we will waste system entropy.
148            */
149           bytes = (bytes == -1) ? 2048 : bytes; /* ok, is 2048 enough? */
150 #ifndef OPENSSL_NO_SETVBUF_IONBF
151           setvbuf(in, NULL, _IONBF, 0); /* don't do buffered reads */
152 #endif /* ndef OPENSSL_NO_SETVBUF_IONBF */
153         }
154 #endif
155         for (;;)
156                 {
157                 if (bytes > 0)
158                         n = (bytes < BUFSIZE)?(int)bytes:BUFSIZE;
159                 else
160                         n = BUFSIZE;
161                 i=fread(buf,1,n,in);
162                 if (i <= 0) break;
163 #ifdef PURIFY
164                 RAND_add(buf,i,(double)i);
165 #else
166                 /* even if n != i, use the full array */
167                 RAND_add(buf,n,(double)i);
168 #endif
169                 ret+=i;
170                 if (bytes > 0)
171                         {
172                         bytes-=n;
173                         if (bytes <= 0) break;
174                         }
175                 }
176         fclose(in);
177         OPENSSL_cleanse(buf,BUFSIZE);
178 err:
179         return(ret);
180         }

182 int RAND_write_file(const char *file)
183         {
184         unsigned char buf[BUFSIZE];
185         int i,ret=0,rand_err=0;
186         FILE *out = NULL;
187         int n;
188 #ifndef OPENSSL_NO_POSIX_IO
189         struct stat sb;

191         i=stat(file,&sb);
192         if (i != -1) {
193 #if defined(S_ISBLK) && defined(S_ISCHR)
```



```
194           if (S_ISBLK(sb.st_mode) || S_ISCHR(sb.st_mode)) {
195             /* this file is a device. we don't write back to it.
196              * we "succeed" on the assumption this is some sort
197              * of random device. Otherwise attempting to write to
198              * and chmod the device causes problems.
199              */
200             return(1);
201           }
202 #endif
203         }
204 #endif

206 #if defined(O_CREAT) && !defined(OPENSSL_NO_POSIX_IO) && !defined(OPENSSL_SYS_VM
207         {
208 #ifndef O_BINARY
209 #define O_BINARY 0
210 #endif
211         /* chmod(..., 0600) is too late to protect the file,
212          * permissions should be restrictive from the start */
213         int fd = open(file, O_WRONLY|O_CREAT|O_BINARY, 0600);
214         if (fd != -1)
215                 out = fdopen(fd, "wb");
216         }
217 #endif

219 #ifdef OPENSSL_SYS_VMS
220         /* VMS NOTE: Prior versions of this routine created a _new_
221          * version of the rand file for each call into this routine, then
222          * deleted all existing versions named ;-1, and finally renamed
223          * the current version as ';1'. Under concurrent usage, this
224          * resulted in an RMS race condition in rename() which could
225          * orphan files (see vms message help for RMS$_REENT). With the
226          * fopen() calls below, openssl/VMS now shares the top-level
227          * version of the rand file. Note that there may still be
228          * conditions where the top-level rand file is locked. If so, this
229          * code will then create a new version of the rand file. Without
230          * the delete and rename code, this can result in ascending file
231          * versions that stop at version 32767, and this routine will then
232          * return an error. The remedy for this is to recode the calling
233          * application to avoid concurrent use of the rand file, or
234          * synchronize usage at the application level. Also consider
235          * whether or not you NEED a persistent rand file in a concurrent
236          * use situation.
237          */

239         out = vms_fopen(file,"rb+",VMS_OPEN_ATTRS);
240         if (out == NULL)
241                 out = vms_fopen(file,"wb",VMS_OPEN_ATTRS);
242 #else
243         if (out == NULL)
244                 out = fopen(file,"wb");
245 #endif
246         if (out == NULL) goto err;

248 #ifndef NO_CHMOD
249         chmod(file,0600);
250 #endif
251         n=RAND_DATA;
252         for (;;)
253                 {
254                 i=(n > BUFSIZE)?BUFSIZE:n;
255                 n-=BUFSIZE;
256                 if (RAND_bytes(buf,i) <= 0)
257                         rand_err=1;
258                 i=fwrite(buf,1,i,out);
259                 if (i <= 0)
```

```
260                         {
261                         ret=0;
262                         break;
263                         }
264                 ret+=i;
265                 if (n <= 0) break;
266                 }

268         fclose(out);
269         OPENSSL_cleanse(buf,BUFSIZE);
270 err:
271         return (rand_err ? -1 : ret);
272         }

274 const char *RAND_file_name(char *buf, size_t size)
275         {
276         char *s=NULL;
277 #ifdef __OpenBSD__
278         struct stat sb;
279 #endif

281         if (OPENSSL_issetugid() == 0)
282                 s=getenv("RANDFILE");
283         if (s != NULL && *s && strlen(s) + 1 < size)
284                 {
285                 if (BUF_strlcpy(buf,s,size) >= size)
286                         return NULL;
287                 }
288         else
289                 {
290                 if (OPENSSL_issetugid() == 0)
291                         s=getenv("HOME");
292 #ifdef DEFAULT_HOME
293                 if (s == NULL)
294                         {
295                         s = DEFAULT_HOME;
296                         }
297 #endif
298                 if (s && *s && strlen(s)+strlen(RFILE)+2 < size)
299                         {
300                         BUF_strlcpy(buf,s,size);
301 #ifndef OPENSSL_SYS_VMS
302                         BUF_strlcat(buf,"/",size);
303 #endif
304                         BUF_strlcat(buf,RFILE,size);
305                         }
306                 else
307                         buf[0] = '\0'; /* no file name */
308                 }

310 #ifdef __OpenBSD__
311         /* given that all random loads just fail if the file can't be
312          * seen on a stat, we stat the file we're returning, if it
313          * fails, use /dev/arandom instead. this allows the user to
314          * use their own source for good random data, but defaults
315          * to something hopefully decent if that isn't available.
316          */

318         if (!buf[0])
319                 if (BUF_strlcpy(buf,"/dev/arandom",size) >= size) {
320                         return(NULL);
321                 }
322         if (stat(buf,&sb) == -1)
323                 if (BUF_strlcpy(buf,"/dev/arandom",size) >= size) {
324                         return(NULL);
325                 }
```



```
327 #endif
328         return(buf);
329         }
330 #endif /* ! codereview */
```

```
**********************************************************
    6439 Wed Aug 13 19:53:14 2014
new/usr/src/lib/openssl/libsunw_crypto/rc2/rc2_cbc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc2/rc2_cbc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/rc2.h>
#include "rc2_locl.h"
```



```
void RC2_cbc_encrypt(const unsigned char *in, unsigned char *out, long length,
             RC2_KEY *ks, unsigned char *iv, int encrypt)
        {
        register unsigned long tin0,tin1;
        register unsigned long tout0,tout1,xor0,xor1;
        register long l=length;
        unsigned long tin[2];

        if (encrypt)
                {
                c2l(iv,tout0);
                c2l(iv,tout1);
                iv-=8;
                for (l-=8; l>=0; l-=8)
                        {
                        c2l(in,tin0);
                        c2l(in,tin1);
                        tin0^=tout0;
                        tin1^=tout1;
                        tin[0]=tin0;
                        tin[1]=tin1;
                        RC2_encrypt(tin,ks);
                        tout0=tin[0]; l2c(tout0,out);
                        tout1=tin[1]; l2c(tout1,out);
                        }
                if (l != -8)
                        {
                        c2ln(in,tin0,tin1,l+8);
                        tin0^=tout0;
                        tin1^=tout1;
                        tin[0]=tin0;
                        tin[1]=tin1;
                        RC2_encrypt(tin,ks);
                        tout0=tin[0]; l2c(tout0,out);
                        tout1=tin[1]; l2c(tout1,out);
                        }
                l2c(tout0,iv);
                l2c(tout1,iv);
                }
        else
                {
                c2l(iv,xor0);
                c2l(iv,xor1);
                iv-=8;
                for (l-=8; l>=0; l-=8)
                        {
                        c2l(in,tin0); tin[0]=tin0;
                        c2l(in,tin1); tin[1]=tin1;
                        RC2_decrypt(tin,ks);
                        tout0=tin[0]^xor0;
                        tout1=tin[1]^xor1;
                        l2c(tout0,out);
                        l2c(tout1,out);
                        xor0=tin0;
                        xor1=tin1;
                        }
                if (l != -8)
                        {
                        c2l(in,tin0); tin[0]=tin0;
                        c2l(in,tin1); tin[1]=tin1;
                        RC2_decrypt(tin,ks);
                        tout0=tin[0]^xor0;
                        tout1=tin[1]^xor1;
                        l2cn(tout0,tout1,out,l+8);
                        xor0=tin0;
                        xor1=tin1;
```

```
128                     }
129                 l2c(xor0,iv);
130                 l2c(xor1,iv);
131                 }
132         tin0=tin1=tout0=tout1=xor0=xor1=0;
133         tin[0]=tin[1]=0;
134         }

136 void RC2_encrypt(unsigned long *d, RC2_KEY *key)
137         {
138         int i,n;
139         register RC2_INT *p0,*p1;
140         register RC2_INT x0,x1,x2,x3,t;
141         unsigned long l;

143         l=d[0];
144         x0=(RC2_INT)l&0xffff;
145         x1=(RC2_INT)(l>>16L);
146         l=d[1];
147         x2=(RC2_INT)l&0xffff;
148         x3=(RC2_INT)(l>>16L);

150         n=3;
151         i=5;

153         p0=p1= &(key->data[0]);
154         for (;;)
155                 {
156                 t=(x0+(x1& ~x3)+(x2&x3)+ *(p0++))&0xffff;
157                 x0=(t<<1)|(t>>15);
158                 t=(x1+(x2& ~x0)+(x3&x0)+ *(p0++))&0xffff;
159                 x1=(t<<2)|(t>>14);
160                 t=(x2+(x3& ~x1)+(x0&x1)+ *(p0++))&0xffff;
161                 x2=(t<<3)|(t>>13);
162                 t=(x3+(x0& ~x2)+(x1&x2)+ *(p0++))&0xffff;
163                 x3=(t<<5)|(t>>11);

165                 if (--i == 0)
166                         {
167                         if (--n == 0) break;
168                         i=(n == 2)?6:5;

170                         x0+=p1[x3&0x3f];
171                         x1+=p1[x0&0x3f];
172                         x2+=p1[x1&0x3f];
173                         x3+=p1[x2&0x3f];
174                         }
175                 }

177         d[0]=(unsigned long)(x0&0xffff)|((unsigned long)(x1&0xffff)<<16L);
178         d[1]=(unsigned long)(x2&0xffff)|((unsigned long)(x3&0xffff)<<16L);
179         }

181 void RC2_decrypt(unsigned long *d, RC2_KEY *key)
182         {
183         int i,n;
184         register RC2_INT *p0,*p1;
185         register RC2_INT x0,x1,x2,x3,t;
186         unsigned long l;

188         l=d[0];
189         x0=(RC2_INT)l&0xffff;
190         x1=(RC2_INT)(l>>16L);
191         l=d[1];
192         x2=(RC2_INT)l&0xffff;
193         x3=(RC2_INT)(l>>16L);
```



```
195         n=3;
196         i=5;

198         p0= &(key->data[63]);
199         p1= &(key->data[0]);
200         for (;;)
201                 {
202                 t=((x3<<11)|(x3>>5))&0xffff;
203                 x3=(t-(x0& ~x2)-(x1&x2)- *(p0--))&0xffff;
204                 t=((x2<<13)|(x2>>3))&0xffff;
205                 x2=(t-(x3& ~x1)-(x0&x1)- *(p0--))&0xffff;
206                 t=((x1<<14)|(x1>>2))&0xffff;
207                 x1=(t-(x2& ~x0)-(x3&x0)- *(p0--))&0xffff;
208                 t=((x0<<15)|(x0>>1))&0xffff;
209                 x0=(t-(x1& ~x3)-(x2&x3)- *(p0--))&0xffff;

211                 if (--i == 0)
212                         {
213                         if (--n == 0) break;
214                         i=(n == 2)?6:5;

216                         x3=(x3-p1[x2&0x3f])&0xffff;
217                         x2=(x2-p1[x1&0x3f])&0xffff;
218                         x1=(x1-p1[x0&0x3f])&0xffff;
219                         x0=(x0-p1[x3&0x3f])&0xffff;
220                         }
221                 }

223         d[0]=(unsigned long)(x0&0xffff)|((unsigned long)(x1&0xffff)<<16L);
224         d[1]=(unsigned long)(x2&0xffff)|((unsigned long)(x3&0xffff)<<16L);
225         }
226 #endif /* ! codereview */
```

```
**********************************************************
    3871 Wed Aug 13 19:53:14 2014
new/usr/src/lib/openssl/libsunw_crypto/rc2/rc2_ecb.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc2/rc2_ecb.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/rc2.h>
#include "rc2_locl.h"
#include <openssl/opensslv.h>
```



```
const char RC2_version[]="RC2" OPENSSL_VERSION_PTEXT;

/* RC2 as implemented frm a posting from
 * Newsgroups: sci.crypt
 * Sender: pgut01@cs.auckland.ac.nz (Peter Gutmann)
 * Subject: Specification for Ron Rivests Cipher No.2
 * Message-ID: <4fk39f$f70@net.auckland.ac.nz>
 * Date: 11 Feb 1996 06:45:03 GMT
 */

void RC2_ecb_encrypt(const unsigned char *in, unsigned char *out, RC2_KEY *ks,
                     int encrypt)
        {
        unsigned long l,d[2];

        c2l(in,l); d[0]=l;
        c2l(in,l); d[1]=l;
        if (encrypt)
                RC2_encrypt(d,ks);
        else
                RC2_decrypt(d,ks);
        l=d[0]; l2c(l,out);
        l=d[1]; l2c(l,out);
        l=d[0]=d[1]=0;
        }
#endif /* ! codereview */
```

```
**********************************************************
    6106 Wed Aug 13 19:53:14 2014
new/usr/src/lib/openssl/libsunw_crypto/rc2/rc2_skey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc2/rc2_skey.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/crypto.h>
#include <openssl/rc2.h>
#include "rc2_locl.h"
```



```
static const unsigned char key_table[256]={
        0xd9,0x78,0xf9,0xc4,0x19,0xdd,0xb5,0xed,0x28,0xe9,0xfd,0x79,
        0x4a,0xa0,0xd8,0x9d,0xc6,0x7e,0x37,0x83,0x2b,0x76,0x53,0x8e,
        0x62,0x4c,0x64,0x88,0x44,0x8b,0xfb,0xa2,0x17,0x9a,0x59,0xf5,
        0x87,0xb3,0x4f,0x13,0x61,0x45,0x6d,0x8d,0x09,0x81,0x7d,0x32,
        0xbd,0x8f,0x40,0xeb,0x86,0xb7,0x7b,0x0b,0xf0,0x95,0x21,0x22,
        0x5c,0x6b,0x4e,0x82,0x54,0xd6,0x65,0x93,0xce,0x60,0xb2,0x1c,
        0x73,0x56,0xc0,0x14,0xa7,0x8c,0xf1,0xdc,0x12,0x75,0xca,0x1f,
        0x3b,0xbe,0xe4,0xd1,0x42,0x3d,0xd4,0x30,0xa3,0x3c,0xb6,0x26,
        0x6f,0xbf,0x0e,0xda,0x46,0x69,0x07,0x57,0x27,0xf2,0x1d,0x9b,
        0xbc,0x94,0x43,0x03,0xf8,0x11,0xc7,0xf6,0x90,0xef,0x3e,0xe7,
        0x06,0xc3,0xd5,0x2f,0xc8,0x66,0x1e,0xd7,0x08,0xe8,0xea,0xde,
        0x80,0x52,0xee,0xf7,0x84,0xaa,0x72,0xac,0x35,0x4d,0x6a,0x2a,
        0x96,0x1a,0xd2,0x71,0x5a,0x15,0x49,0x74,0x4b,0x9f,0xd0,0x5e,
        0x04,0x18,0xa4,0xec,0xc2,0xe0,0x41,0x6e,0x0f,0x51,0xcb,0xcc,
        0x24,0x91,0xaf,0x50,0xa1,0xf4,0x70,0x39,0x99,0x7c,0x3a,0x85,
        0x23,0xb8,0xb4,0x7a,0xfc,0x02,0x36,0x5b,0x25,0x55,0x97,0x31,
        0x2d,0x5d,0xfa,0x98,0xe3,0x8a,0x92,0xae,0x05,0xdf,0x29,0x10,
        0x67,0x6c,0xba,0xc9,0xd3,0x00,0xe6,0xcf,0xe1,0x9e,0xa8,0x2c,
        0x63,0x16,0x01,0x3f,0x58,0xe2,0x89,0xa9,0x0d,0x38,0x34,0x1b,
        0xab,0x33,0xff,0xb0,0xbb,0x48,0x0c,0x5f,0xb9,0xb1,0xcd,0x2e,
        0xc5,0xf3,0xdb,0x47,0xe5,0xa5,0x9c,0x77,0x0a,0xa6,0x20,0x68,
        0xfe,0x7f,0xc1,0xad,
        };

#if defined(_MSC_VER) && defined(_ARM_)
#pragma optimize("g",off)
#endif

/* It has come to my attention that there are 2 versions of the RC2
 * key schedule.  One which is normal, and anther which has a hook to
 * use a reduced key length.
 * BSAFE uses the 'retarded' version.  What I previously shipped is
 * the same as specifying 1024 for the 'bits' parameter.  Bsafe uses
 * a version where the bits parameter is the same as len*8 */
void RC2_set_key(RC2_KEY *key, int len, const unsigned char *data, int bits)
#ifdef OPENSSL_FIPS
        {
        fips_cipher_abort(RC2);
        private_RC2_set_key(key, len, data, bits);
        }
void private_RC2_set_key(RC2_KEY *key, int len, const unsigned char *data, int b
#endif
        {
        int i,j;
        unsigned char *k;
        RC2_INT *ki;
        unsigned int c,d;

        k= (unsigned char *)&(key->data[0]);
        *k=0; /* for if there is a zero length key */

        if (len > 128) len=128;
        if (bits <= 0) bits=1024;
        if (bits > 1024) bits=1024;

        for (i=0; i<len; i++)
                k[i]=data[i];

        /* expand table */
        d=k[len-1];
        j=0;
        for (i=len; i < 128; i++,j++)
                {
                d=key_table[(k[j]+d)&0xff];
```

```
128                 k[i]=d;
129                 }

131         /* hmm.... key reduction to 'bits' bits */

133         j=(bits+7)>>3;
134         i=128-j;
135         c= (0xff>>(-bits & 0x07));

137         d=key_table[k[i]&c];
138         k[i]=d;
139         while (i--)
140                 {
141                 d=key_table[k[i+j]^d];
142                 k[i]=d;
143                 }

145         /* copy from bytes into RC2_INT's */
146         ki= &(key->data[63]);
147         for (i=127; i>=0; i-=2)
148                 *(ki--)=((k[i]<<8)|k[i-1])&0xffff;
149         }

151 #if defined(_MSC_VER)
152 #pragma optimize("",on)
153 #endif
154 #endif /* ! codereview */
```

```
**********************************************************
    4482 Wed Aug 13 19:53:14 2014
new/usr/src/lib/openssl/libsunw_crypto/rc2/rc2cfb64.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc2/rc2cfb64.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/rc2.h>
#include "rc2_locl.h"
```



```
/* The input and output encrypted as though 64bit cfb mode is being
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */

void RC2_cfb64_encrypt(const unsigned char *in, unsigned char *out,
                       long length, RC2_KEY *schedule, unsigned char *ivec,
                       int *num, int encrypt)
        {
        register unsigned long v0,v1,t;
        register int n= *num;
        register long l=length;
        unsigned long ti[2];
        unsigned char *iv,c,cc;

        iv=(unsigned char *)ivec;
        if (encrypt)
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                c2l(iv,v0); ti[0]=v0;
                                c2l(iv,v1); ti[1]=v1;
                                RC2_encrypt((unsigned long *)ti,schedule);
                                iv=(unsigned char *)ivec;
                                t=ti[0]; l2c(t,iv);
                                t=ti[1]; l2c(t,iv);
                                iv=(unsigned char *)ivec;
                                }
                        c= *(in++)^iv[n];
                        *(out++)=c;
                        iv[n]=c;
                        n=(n+1)&0x07;
                        }
                }
        else
                {
                while (l--)
                        {
                        if (n == 0)
                                {
                                c2l(iv,v0); ti[0]=v0;
                                c2l(iv,v1); ti[1]=v1;
                                RC2_encrypt((unsigned long *)ti,schedule);
                                iv=(unsigned char *)ivec;
                                t=ti[0]; l2c(t,iv);
                                t=ti[1]; l2c(t,iv);
                                iv=(unsigned char *)ivec;
                                }
                        cc= *(in++);
                        c=iv[n];
                        iv[n]=cc;
                        *(out++)=c^cc;
                        n=(n+1)&0x07;
                        }
                }
        v0=v1=ti[0]=ti[1]=t=c=cc=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4214 Wed Aug 13 19:53:14 2014
new/usr/src/lib/openssl/libsunw_crypto/rc2/rc2ofb64.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc2/rc2ofb64.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/rc2.h>
#include "rc2_locl.h"
```



```
/* The input and output encrypted as though 64bit ofb mode is being
 * used.  The extra state information to record how much of the
 * 64bit block we have used is contained in *num;
 */
void RC2_ofb64_encrypt(const unsigned char *in, unsigned char *out,
                       long length, RC2_KEY *schedule, unsigned char *ivec,
                       int *num)
        {
        register unsigned long v0,v1,t;
        register int n= *num;
        register long l=length;
        unsigned char d[8];
        register char *dp;
        unsigned long ti[2];
        unsigned char *iv;
        int save=0;

        iv=(unsigned char *)ivec;
        c2l(iv,v0);
        c2l(iv,v1);
        ti[0]=v0;
        ti[1]=v1;
        dp=(char *)d;
        l2c(v0,dp);
        l2c(v1,dp);
        while (l--)
                {
                if (n == 0)
                        {
                        RC2_encrypt((unsigned long *)ti,schedule);
                        dp=(char *)d;
                        t=ti[0]; l2c(t,dp);
                        t=ti[1]; l2c(t,dp);
                        save++;
                        }
                *(out++)= *(in++)^d[n];
                n=(n+1)&0x07;
                }
        if (save)
                {
                v0=ti[0];
                v1=ti[1];
                iv=(unsigned char *)ivec;
                l2c(v0,iv);
                l2c(v1,iv);
                }
        t=v0=v1=ti[0]=ti[1]=0;
        *num=n;
        }
#endif /* ! codereview */
```

```
**********************************************************
    2764 Wed Aug 13 19:53:14 2014
new/usr/src/lib/openssl/libsunw_crypto/rc4/rc4_utl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rc4/rc4_utl.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 */

#include <openssl/opensslv.h>
#include <openssl/crypto.h>
#include <openssl/rc4.h>

void RC4_set_key(RC4_KEY *key, int len, const unsigned char *data)
        {
#ifdef OPENSSL_FIPS
        fips_cipher_abort(RC4);
#endif
        private_RC4_set_key(key, len, data);
```



```
        }
#endif /* ! codereview */
```

```
**********************************************************
   10096 Wed Aug 13 19:53:15 2014
new/usr/src/lib/openssl/libsunw_crypto/ripemd/rmd_dgst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ripemd/rmd_dgst.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "rmd_locl.h"
#include <openssl/opensslv.h>
```



```
#include <openssl/crypto.h>

const char RMD160_version[]="RIPE-MD160" OPENSSL_VERSION_PTEXT;

#  ifdef RMD160_ASM
#    ifdef OPENSSL_IA64_SSE2
     void ripemd160_block_x86(RIPEMD160_CTX *c, unsigned long *p,size_t num);
#    define ripemd160_block ripemd160_block_x86
#  else
     void ripemd160_block(RIPEMD160_CTX *c, unsigned long *p,size_t num);
#  endif

fips_md_init(RIPEMD160)
        {
        memset (c,0,sizeof(*c));
        c->A=RIPEMD160_A;
        c->B=RIPEMD160_B;
        c->C=RIPEMD160_C;
        c->D=RIPEMD160_D;
        c->E=RIPEMD160_E;
        return 1;
        }

#ifndef ripemd160_block_data_order
#ifdef X
#undef X
#endif
void ripemd160_block_data_order (RIPEMD160_CTX *ctx, const void *p, size_t num)
        {
        const unsigned char *data=p;
        register unsigned MD32_REG_T A,B,C,D,E;
        unsigned MD32_REG_T a,b,c,d,e,l;
#ifndef MD32_XARRAY
        /* See comment in crypto/sha/sha_locl.h for details. */
        unsigned MD32_REG_T     XX0, XX1, XX2, XX3, XX4, XX5, XX6, XX7,
                                XX8, XX9,XX10,XX11,XX12,XX13,XX14,XX15;
# define X(i)   XX##i
#else
        RIPEMD160_LONG  XX[16];
# define X(i)   XX[i]
#endif

        for (;num--;)
                {

        A=ctx->A; B=ctx->B; C=ctx->C; D=ctx->D; E=ctx->E;

        (void)HOST_c2l(data,l); X( 0)=l;(void)HOST_c2l(data,l); X( 1)=l;
        RIP1(A,B,C,D,E,WL00,SL00);      (void)HOST_c2l(data,l); X( 2)=l;
        RIP1(E,A,B,C,D,WL01,SL01);      (void)HOST_c2l(data,l); X( 3)=l;
        RIP1(D,E,A,B,C,WL02,SL02);      (void)HOST_c2l(data,l); X( 4)=l;
        RIP1(C,D,E,A,B,WL03,SL03);      (void)HOST_c2l(data,l); X( 5)=l;
        RIP1(B,C,D,E,A,WL04,SL04);      (void)HOST_c2l(data,l); X( 6)=l;
        RIP1(A,B,C,D,E,WL05,SL05);      (void)HOST_c2l(data,l); X( 7)=l;
        RIP1(E,A,B,C,D,WL06,SL06);      (void)HOST_c2l(data,l); X( 8)=l;
        RIP1(D,E,A,B,C,WL07,SL07);      (void)HOST_c2l(data,l); X( 9)=l;
        RIP1(C,D,E,A,B,WL08,SL08);      (void)HOST_c2l(data,l); X(10)=l;
        RIP1(B,C,D,E,A,WL09,SL09);      (void)HOST_c2l(data,l); X(11)=l;
        RIP1(A,B,C,D,E,WL10,SL10);      (void)HOST_c2l(data,l); X(12)=l;
        RIP1(E,A,B,C,D,WL11,SL11);      (void)HOST_c2l(data,l); X(13)=l;
        RIP1(D,E,A,B,C,WL12,SL12);      (void)HOST_c2l(data,l); X(14)=l;
        RIP1(C,D,E,A,B,WL13,SL13);      (void)HOST_c2l(data,l); X(15)=l;
        RIP1(B,C,D,E,A,WL14,SL14);
        RIP1(A,B,C,D,E,WL15,SL15);

        RIP2(E,A,B,C,D,WL16,SL16,KL1);
        RIP2(D,E,A,B,C,WL17,SL17,KL1);
```

```
128         RIP2(C,D,E,A,B,WL18,SL18,KL1);
129         RIP2(B,C,D,E,A,WL19,SL19,KL1);
130         RIP2(A,B,C,D,E,WL20,SL20,KL1);
131         RIP2(E,A,B,C,D,WL21,SL21,KL1);
132         RIP2(D,E,A,B,C,WL22,SL22,KL1);
133         RIP2(C,D,E,A,B,WL23,SL23,KL1);
134         RIP2(B,C,D,E,A,WL24,SL24,KL1);
135         RIP2(A,B,C,D,E,WL25,SL25,KL1);
136         RIP2(E,A,B,C,D,WL26,SL26,KL1);
137         RIP2(D,E,A,B,C,WL27,SL27,KL1);
138         RIP2(C,D,E,A,B,WL28,SL28,KL1);
139         RIP2(B,C,D,E,A,WL29,SL29,KL1);
140         RIP2(A,B,C,D,E,WL30,SL30,KL1);
141         RIP2(E,A,B,C,D,WL31,SL31,KL1);

143         RIP3(D,E,A,B,C,WL32,SL32,KL2);
144         RIP3(C,D,E,A,B,WL33,SL33,KL2);
145         RIP3(B,C,D,E,A,WL34,SL34,KL2);
146         RIP3(A,B,C,D,E,WL35,SL35,KL2);
147         RIP3(E,A,B,C,D,WL36,SL36,KL2);
148         RIP3(D,E,A,B,C,WL37,SL37,KL2);
149         RIP3(C,D,E,A,B,WL38,SL38,KL2);
150         RIP3(B,C,D,E,A,WL39,SL39,KL2);
151         RIP3(A,B,C,D,E,WL40,SL40,KL2);
152         RIP3(E,A,B,C,D,WL41,SL41,KL2);
153         RIP3(D,E,A,B,C,WL42,SL42,KL2);
154         RIP3(C,D,E,A,B,WL43,SL43,KL2);
155         RIP3(B,C,D,E,A,WL44,SL44,KL2);
156         RIP3(A,B,C,D,E,WL45,SL45,KL2);
157         RIP3(E,A,B,C,D,WL46,SL46,KL2);
158         RIP3(D,E,A,B,C,WL47,SL47,KL2);

160         RIP4(C,D,E,A,B,WL48,SL48,KL3);
161         RIP4(B,C,D,E,A,WL49,SL49,KL3);
162         RIP4(A,B,C,D,E,WL50,SL50,KL3);
163         RIP4(E,A,B,C,D,WL51,SL51,KL3);
164         RIP4(D,E,A,B,C,WL52,SL52,KL3);
165         RIP4(C,D,E,A,B,WL53,SL53,KL3);
166         RIP4(B,C,D,E,A,WL54,SL54,KL3);
167         RIP4(A,B,C,D,E,WL55,SL55,KL3);
168         RIP4(E,A,B,C,D,WL56,SL56,KL3);
169         RIP4(D,E,A,B,C,WL57,SL57,KL3);
170         RIP4(C,D,E,A,B,WL58,SL58,KL3);
171         RIP4(B,C,D,E,A,WL59,SL59,KL3);
172         RIP4(A,B,C,D,E,WL60,SL60,KL3);
173         RIP4(E,A,B,C,D,WL61,SL61,KL3);
174         RIP4(D,E,A,B,C,WL62,SL62,KL3);
175         RIP4(C,D,E,A,B,WL63,SL63,KL3);

177         RIP5(B,C,D,E,A,WL64,SL64,KL4);
178         RIP5(A,B,C,D,E,WL65,SL65,KL4);
179         RIP5(E,A,B,C,D,WL66,SL66,KL4);
180         RIP5(D,E,A,B,C,WL67,SL67,KL4);
181         RIP5(C,D,E,A,B,WL68,SL68,KL4);
182         RIP5(B,C,D,E,A,WL69,SL69,KL4);
183         RIP5(A,B,C,D,E,WL70,SL70,KL4);
184         RIP5(E,A,B,C,D,WL71,SL71,KL4);
185         RIP5(D,E,A,B,C,WL72,SL72,KL4);
186         RIP5(C,D,E,A,B,WL73,SL73,KL4);
187         RIP5(B,C,D,E,A,WL74,SL74,KL4);
188         RIP5(A,B,C,D,E,WL75,SL75,KL4);
189         RIP5(E,A,B,C,D,WL76,SL76,KL4);
190         RIP5(D,E,A,B,C,WL77,SL77,KL4);
191         RIP5(C,D,E,A,B,WL78,SL78,KL4);
192         RIP5(B,C,D,E,A,WL79,SL79,KL4);
```



```
194         a=A; b=B; c=C; d=D; e=E;
195         /* Do other half */
196         A=ctx->A; B=ctx->B; C=ctx->C; D=ctx->D; E=ctx->E;

198         RIP5(A,B,C,D,E,WR00,SR00,KR0);
199         RIP5(E,A,B,C,D,WR01,SR01,KR0);
200         RIP5(D,E,A,B,C,WR02,SR02,KR0);
201         RIP5(C,D,E,A,B,WR03,SR03,KR0);
202         RIP5(B,C,D,E,A,WR04,SR04,KR0);
203         RIP5(A,B,C,D,E,WR05,SR05,KR0);
204         RIP5(E,A,B,C,D,WR06,SR06,KR0);
205         RIP5(D,E,A,B,C,WR07,SR07,KR0);
206         RIP5(C,D,E,A,B,WR08,SR08,KR0);
207         RIP5(B,C,D,E,A,WR09,SR09,KR0);
208         RIP5(A,B,C,D,E,WR10,SR10,KR0);
209         RIP5(E,A,B,C,D,WR11,SR11,KR0);
210         RIP5(D,E,A,B,C,WR12,SR12,KR0);
211         RIP5(C,D,E,A,B,WR13,SR13,KR0);
212         RIP5(B,C,D,E,A,WR14,SR14,KR0);
213         RIP5(A,B,C,D,E,WR15,SR15,KR0);

215         RIP4(E,A,B,C,D,WR16,SR16,KR1);
216         RIP4(D,E,A,B,C,WR17,SR17,KR1);
217         RIP4(C,D,E,A,B,WR18,SR18,KR1);
218         RIP4(B,C,D,E,A,WR19,SR19,KR1);
219         RIP4(A,B,C,D,E,WR20,SR20,KR1);
220         RIP4(E,A,B,C,D,WR21,SR21,KR1);
221         RIP4(D,E,A,B,C,WR22,SR22,KR1);
222         RIP4(C,D,E,A,B,WR23,SR23,KR1);
223         RIP4(B,C,D,E,A,WR24,SR24,KR1);
224         RIP4(A,B,C,D,E,WR25,SR25,KR1);
225         RIP4(E,A,B,C,D,WR26,SR26,KR1);
226         RIP4(D,E,A,B,C,WR27,SR27,KR1);
227         RIP4(C,D,E,A,B,WR28,SR28,KR1);
228         RIP4(B,C,D,E,A,WR29,SR29,KR1);
229         RIP4(A,B,C,D,E,WR30,SR30,KR1);
230         RIP4(E,A,B,C,D,WR31,SR31,KR1);

232         RIP3(D,E,A,B,C,WR32,SR32,KR2);
233         RIP3(C,D,E,A,B,WR33,SR33,KR2);
234         RIP3(B,C,D,E,A,WR34,SR34,KR2);
235         RIP3(A,B,C,D,E,WR35,SR35,KR2);
236         RIP3(E,A,B,C,D,WR36,SR36,KR2);
237         RIP3(D,E,A,B,C,WR37,SR37,KR2);
238         RIP3(C,D,E,A,B,WR38,SR38,KR2);
239         RIP3(B,C,D,E,A,WR39,SR39,KR2);
240         RIP3(A,B,C,D,E,WR40,SR40,KR2);
241         RIP3(E,A,B,C,D,WR41,SR41,KR2);
242         RIP3(D,E,A,B,C,WR42,SR42,KR2);
243         RIP3(C,D,E,A,B,WR43,SR43,KR2);
244         RIP3(B,C,D,E,A,WR44,SR44,KR2);
245         RIP3(A,B,C,D,E,WR45,SR45,KR2);
246         RIP3(E,A,B,C,D,WR46,SR46,KR2);
247         RIP3(D,E,A,B,C,WR47,SR47,KR2);

249         RIP2(C,D,E,A,B,WR48,SR48,KR3);
250         RIP2(B,C,D,E,A,WR49,SR49,KR3);
251         RIP2(A,B,C,D,E,WR50,SR50,KR3);
252         RIP2(E,A,B,C,D,WR51,SR51,KR3);
253         RIP2(D,E,A,B,C,WR52,SR52,KR3);
254         RIP2(C,D,E,A,B,WR53,SR53,KR3);
255         RIP2(B,C,D,E,A,WR54,SR54,KR3);
256         RIP2(A,B,C,D,E,WR55,SR55,KR3);
257         RIP2(E,A,B,C,D,WR56,SR56,KR3);
258         RIP2(D,E,A,B,C,WR57,SR57,KR3);
259         RIP2(C,D,E,A,B,WR58,SR58,KR3);
```

```
260         RIP2(B,C,D,E,A,WR59,SR59,KR3);
261         RIP2(A,B,C,D,E,WR60,SR60,KR3);
262         RIP2(E,A,B,C,D,WR61,SR61,KR3);
263         RIP2(D,E,A,B,C,WR62,SR62,KR3);
264         RIP2(C,D,E,A,B,WR63,SR63,KR3);

266         RIP1(B,C,D,E,A,WR64,SR64);
267         RIP1(A,B,C,D,E,WR65,SR65);
268         RIP1(E,A,B,C,D,WR66,SR66);
269         RIP1(D,E,A,B,C,WR67,SR67);
270         RIP1(C,D,E,A,B,WR68,SR68);
271         RIP1(B,C,D,E,A,WR69,SR69);
272         RIP1(A,B,C,D,E,WR70,SR70);
273         RIP1(E,A,B,C,D,WR71,SR71);
274         RIP1(D,E,A,B,C,WR72,SR72);
275         RIP1(C,D,E,A,B,WR73,SR73);
276         RIP1(B,C,D,E,A,WR74,SR74);
277         RIP1(A,B,C,D,E,WR75,SR75);
278         RIP1(E,A,B,C,D,WR76,SR76);
279         RIP1(D,E,A,B,C,WR77,SR77);
280         RIP1(C,D,E,A,B,WR78,SR78);
281         RIP1(B,C,D,E,A,WR79,SR79);

283         D     =ctx->B+c+D;
284         ctx->B=ctx->C+d+E;
285         ctx->C=ctx->D+e+A;
286         ctx->D=ctx->E+a+B;
287         ctx->E=ctx->A+b+C;
288         ctx->A=D;

290                 }
291         }
292 #endif
293 #endif /* ! codereview */
```

```
**********************************************************
    3639 Wed Aug 13 19:53:15 2014
new/usr/src/lib/openssl/libsunw_crypto/ripemd/rmd_one.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ripemd/rmd_one.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <openssl/ripemd.h>
```



```
#include <openssl/crypto.h>

unsigned char *RIPEMD160(const unsigned char *d, size_t n,
             unsigned char *md)
        {
        RIPEMD160_CTX c;
        static unsigned char m[RIPEMD160_DIGEST_LENGTH];

        if (md == NULL) md=m;
        if (!RIPEMD160_Init(&c))
                return NULL;
        RIPEMD160_Update(&c,d,n);
        RIPEMD160_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c)); /* security consideration */
        return(md);
        }
#endif /* ! codereview */
```

```
**********************************************************
   16558 Wed Aug 13 19:53:15 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_ameth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_ameth.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>
#include <openssl/rsa.h>
#include <openssl/bn.h>
#ifndef OPENSSL_NO_CMS
#include <openssl/cms.h>
#endif
#include "asn1_locl.h"

static int rsa_pub_encode(X509_PUBKEY *pk, const EVP_PKEY *pkey)
        {
        unsigned char *penc = NULL;
        int penclen;
        penclen = i2d_RSAPublicKey(pkey->pkey.rsa, &penc);
        if (penclen <= 0)
                return 0;
        if (X509_PUBKEY_set0_param(pk, OBJ_nid2obj(EVP_PKEY_RSA),
                                   V_ASN1_NULL, NULL, penc, penclen))
                return 1;

        OPENSSL_free(penc);
        return 0;
        }

static int rsa_pub_decode(EVP_PKEY *pkey, X509_PUBKEY *pubkey)
        {
        const unsigned char *p;
        int pklen;
        RSA *rsa = NULL;
        if (!X509_PUBKEY_get0_param(NULL, &p, &pklen, NULL, pubkey))
                return 0;
        if (!(rsa = d2i_RSAPublicKey(NULL, &p, pklen)))
                {
                RSAerr(RSA_F_RSA_PUB_DECODE, ERR_R_RSA_LIB);
                return 0;
                }
        EVP_PKEY_assign_RSA (pkey, rsa);
        return 1;
        }

static int rsa_pub_cmp(const EVP_PKEY *a, const EVP_PKEY *b)
        {
        if (BN_cmp(b->pkey.rsa->n,a->pkey.rsa->n) != 0
                || BN_cmp(b->pkey.rsa->e,a->pkey.rsa->e) != 0)
                        return 0;
        return 1;
        }

static int old_rsa_priv_decode(EVP_PKEY *pkey,
                                        const unsigned char **pder, int derlen)
        {
        RSA *rsa;
        if (!(rsa = d2i_RSAPrivateKey (NULL, pder, derlen)))
                {
                RSAerr(RSA_F_OLD_RSA_PRIV_DECODE, ERR_R_RSA_LIB);
                return 0;
                }
        EVP_PKEY_assign_RSA(pkey, rsa);
        return 1;
        }

static int old_rsa_priv_encode(const EVP_PKEY *pkey, unsigned char **pder)
        {
        return i2d_RSAPrivateKey(pkey->pkey.rsa, pder);
        }

static int rsa_priv_encode(PKCS8_PRIV_KEY_INFO *p8, const EVP_PKEY *pkey)
```

```
128         {
129         unsigned char *rk = NULL;
130         int rklen;
131         rklen = i2d_RSAPrivateKey(pkey->pkey.rsa, &rk);
132
133         if (rklen <= 0)
134                 {
135                 RSAerr(RSA_F_RSA_PRIV_ENCODE,ERR_R_MALLOC_FAILURE);
136                 return 0;
137                 }
138
139         if (!PKCS8_pkey_set0(p8, OBJ_nid2obj(NID_rsaEncryption), 0,
140                              V_ASN1_NULL, NULL, rk, rklen))
141                 {
142                 RSAerr(RSA_F_RSA_PRIV_ENCODE,ERR_R_MALLOC_FAILURE);
143                 return 0;
144                 }
145
146         return 1;
147         }
148
149 static int rsa_priv_decode(EVP_PKEY *pkey, PKCS8_PRIV_KEY_INFO *p8)
150         {
151         const unsigned char *p;
152         int pklen;
153         if (!PKCS8_pkey_get0(NULL, &p, &pklen, NULL, p8))
154                 return 0;
155         return old_rsa_priv_decode(pkey, &p, pklen);
156         }
157
158 static int int_rsa_size(const EVP_PKEY *pkey)
159         {
160         return RSA_size(pkey->pkey.rsa);
161         }
162
163 static int rsa_bits(const EVP_PKEY *pkey)
164         {
165         return BN_num_bits(pkey->pkey.rsa->n);
166         }
167
168 static void int_rsa_free(EVP_PKEY *pkey)
169         {
170         RSA_free(pkey->pkey.rsa);
171         }
172
173
174 static void update_buflen(const BIGNUM *b, size_t *pbuflen)
175         {
176         size_t i;
177         if (!b)
178                 return;
179         if (*pbuflen < (i = (size_t)BN_num_bytes(b)))
180                         *pbuflen = i;
181         }
182
183 static int do_rsa_print(BIO *bp, const RSA *x, int off, int priv)
184         {
185         char *str;
186         const char *s;
187         unsigned char *m=NULL;
188         int ret=0, mod_len = 0;
189         size_t buf_len=0;
190
191         update_buflen(x->n, &buf_len);
192         update_buflen(x->e, &buf_len);
```



```
194         if (priv)
195                 {
196                 update_buflen(x->d, &buf_len);
197                 update_buflen(x->p, &buf_len);
198                 update_buflen(x->q, &buf_len);
199                 update_buflen(x->dmp1, &buf_len);
200                 update_buflen(x->dmq1, &buf_len);
201                 update_buflen(x->iqmp, &buf_len);
202                 }
203
204         m=(unsigned char *)OPENSSL_malloc(buf_len+10);
205         if (m == NULL)
206                 {
207                 RSAerr(RSA_F_DO_RSA_PRINT,ERR_R_MALLOC_FAILURE);
208                 goto err;
209                 }
210
211         if (x->n != NULL)
212                 mod_len = BN_num_bits(x->n);
213
214         if(!BIO_indent(bp,off,128))
215                 goto err;
216
217         if (priv && x->d)
218                 {
219                 if (BIO_printf(bp,"Private-Key: (%d bit)\n", mod_len)
220                         <= 0) goto err;
221                 str = "modulus:";
222                 s = "publicExponent:";
223                 }
224         else
225                 {
226                 if (BIO_printf(bp,"Public-Key: (%d bit)\n", mod_len)
227                         <= 0) goto err;
228                 str = "Modulus:";
229                 s= "Exponent:";
230                 }
231         if (!ASN1_bn_print(bp,str,x->n,m,off)) goto err;
232         if (!ASN1_bn_print(bp,s,x->e,m,off))
233                 goto err;
234         if (priv)
235                 {
236                 if (!ASN1_bn_print(bp,"privateExponent:",x->d,m,off))
237                         goto err;
238                 if (!ASN1_bn_print(bp,"prime1:",x->p,m,off))
239                         goto err;
240                 if (!ASN1_bn_print(bp,"prime2:",x->q,m,off))
241                         goto err;
242                 if (!ASN1_bn_print(bp,"exponent1:",x->dmp1,m,off))
243                         goto err;
244                 if (!ASN1_bn_print(bp,"exponent2:",x->dmq1,m,off))
245                         goto err;
246                 if (!ASN1_bn_print(bp,"coefficient:",x->iqmp,m,off))
247                         goto err;
248                 }
249         ret=1;
250 err:
251         if (m != NULL) OPENSSL_free(m);
252         return(ret);
253         }
254
255 static int rsa_pub_print(BIO *bp, const EVP_PKEY *pkey, int indent,
256                                                         ASN1_PCTX *ctx)
257         {
258         return do_rsa_print(bp, pkey->pkey.rsa, indent, 0);
259         }
```

```
262 static int rsa_priv_print(BIO *bp, const EVP_PKEY *pkey, int indent,
263                                                         ASN1_PCTX *ctx)
264         {
265         return do_rsa_print(bp, pkey->pkey.rsa, indent, 1);
266         }

268 static RSA_PSS_PARAMS *rsa_pss_decode(const X509_ALGOR *alg,
269                                       X509_ALGOR **pmaskHash)
270         {
271         const unsigned char *p;
272         int plen;
273         RSA_PSS_PARAMS *pss;

275         *pmaskHash = NULL;

277         if (!alg->parameter || alg->parameter->type != V_ASN1_SEQUENCE)
278                 return NULL;
279         p = alg->parameter->value.sequence->data;
280         plen = alg->parameter->value.sequence->length;
281         pss = d2i_RSA_PSS_PARAMS(NULL, &p, plen);

283         if (!pss)
284                 return NULL;

286         if (pss->maskGenAlgorithm)
287                 {
288                 ASN1_TYPE *param = pss->maskGenAlgorithm->parameter;
289                 if (OBJ_obj2nid(pss->maskGenAlgorithm->algorithm) == NID_mgf1
290                         && param->type == V_ASN1_SEQUENCE)
291                         {
292                         p = param->value.sequence->data;
293                         plen = param->value.sequence->length;
294                         *pmaskHash = d2i_X509_ALGOR(NULL, &p, plen);
295                         }
296                 }

298         return pss;
299         }

301 static int rsa_pss_param_print(BIO *bp, RSA_PSS_PARAMS *pss,
302                                X509_ALGOR *maskHash, int indent)
303         {
304         int rv = 0;
305         if (!pss)
306                 {
307                 if (BIO_puts(bp, " (INVALID PSS PARAMETERS)\n") <= 0)
308                         return 0;
309                 return 1;
310                 }
311         if (BIO_puts(bp, "\n") <= 0)
312                 goto err;
313         if (!BIO_indent(bp, indent, 128))
314                 goto err;
315         if (BIO_puts(bp, "Hash Algorithm: ") <= 0)
316                 goto err;

318         if (pss->hashAlgorithm)
319                 {
320                 if (i2a_ASN1_OBJECT(bp, pss->hashAlgorithm->algorithm) <= 0)
321                         goto err;
322                 }
323         else if (BIO_puts(bp, "sha1 (default)") <= 0)
324                 goto err;
```



```
326         if (BIO_puts(bp, "\n") <= 0)
327                 goto err;

329         if (!BIO_indent(bp, indent, 128))
330                 goto err;

332         if (BIO_puts(bp, "Mask Algorithm: ") <= 0)
333                         goto err;
334         if (pss->maskGenAlgorithm)
335                 {
336                 if (i2a_ASN1_OBJECT(bp, pss->maskGenAlgorithm->algorithm) <= 0)
337                         goto err;
338                 if (BIO_puts(bp, " with ") <= 0)
339                         goto err;
340                 if (maskHash)
341                         {
342                         if (i2a_ASN1_OBJECT(bp, maskHash->algorithm) <= 0)
343                         goto err;
344                         }
345                 else if (BIO_puts(bp, "INVALID") <= 0)
346                         goto err;
347                 }
348         else if (BIO_puts(bp, "mgf1 with sha1 (default)") <= 0)
349                 goto err;
350         BIO_puts(bp, "\n");

352         if (!BIO_indent(bp, indent, 128))
353                 goto err;
354         if (BIO_puts(bp, "Salt Length: 0x") <= 0)
355                         goto err;
356         if (pss->saltLength)
357                 {
358                 if (i2a_ASN1_INTEGER(bp, pss->saltLength) <= 0)
359                         goto err;
360                 }
361         else if (BIO_puts(bp, "14 (default)") <= 0)
362                 goto err;
363         BIO_puts(bp, "\n");

365         if (!BIO_indent(bp, indent, 128))
366                 goto err;
367         if (BIO_puts(bp, "Trailer Field: 0x") <= 0)
368                         goto err;
369         if (pss->trailerField)
370                 {
371                 if (i2a_ASN1_INTEGER(bp, pss->trailerField) <= 0)
372                         goto err;
373                 }
374         else if (BIO_puts(bp, "BC (default)") <= 0)
375                 goto err;
376         BIO_puts(bp, "\n");

378         rv = 1;

380         err:
381         return rv;

383         }

385 static int rsa_sig_print(BIO *bp, const X509_ALGOR *sigalg,
386                                         const ASN1_STRING *sig,
387                                         int indent, ASN1_PCTX *pctx)
388         {
389         if (OBJ_obj2nid(sigalg->algorithm) == NID_rsassaPss)
390                 {
391                 int rv;
```

```
392                 RSA_PSS_PARAMS *pss;
393                 X509_ALGOR *maskHash;
394                 pss = rsa_pss_decode(sigalg, &maskHash);
395                 rv = rsa_pss_param_print(bp, pss, maskHash, indent);
396                 if (pss)
397                         RSA_PSS_PARAMS_free(pss);
398                 if (maskHash)
399                         X509_ALGOR_free(maskHash);
400                 if (!rv)
401                         return 0;
402                 }
403         else if (!sig && BIO_puts(bp, "\n") <= 0)
404                 return 0;
405         if (sig)
406                 return X509_signature_dump(bp, sig, indent);
407         return 1;
408         }

410 static int rsa_pkey_ctrl(EVP_PKEY *pkey, int op, long arg1, void *arg2)
411         {
412         X509_ALGOR *alg = NULL;
413         switch (op)
414                 {

416                 case ASN1_PKEY_CTRL_PKCS7_SIGN:
417                 if (arg1 == 0)
418                         PKCS7_SIGNER_INFO_get0_algs(arg2, NULL, NULL, &alg);
419                 break;

421                 case ASN1_PKEY_CTRL_PKCS7_ENCRYPT:
422                 if (arg1 == 0)
423                         PKCS7_RECIP_INFO_get0_alg(arg2, &alg);
424                 break;
425 #ifndef OPENSSL_NO_CMS
426                 case ASN1_PKEY_CTRL_CMS_SIGN:
427                 if (arg1 == 0)
428                         CMS_SignerInfo_get0_algs(arg2, NULL, NULL, NULL, &alg);
429                 break;

431                 case ASN1_PKEY_CTRL_CMS_ENVELOPE:
432                 if (arg1 == 0)
433                         CMS_RecipientInfo_ktri_get0_algs(arg2, NULL, NULL, &alg)
434                 break;
435 #endif

437                 case ASN1_PKEY_CTRL_DEFAULT_MD_NID:
438                 *(int *)arg2 = NID_sha1;
439                 return 1;

441                 default:
442                 return -2;

444                 }

446         if (alg)
447                 X509_ALGOR_set0(alg, OBJ_nid2obj(NID_rsaEncryption),
448                                                         V_ASN1_NULL, 0);

450         return 1;

452         }

454 /* Customised RSA item verification routine. This is called
455  * when a signature is encountered requiring special handling. We
456  * currently only handle PSS.
457  */
```



```
460 static int rsa_item_verify(EVP_MD_CTX *ctx, const ASN1_ITEM *it, void *asn,
461                         X509_ALGOR *sigalg, ASN1_BIT_STRING *sig,
462                         EVP_PKEY *pkey)
463         {
464         int rv = -1;
465         int saltlen;
466         const EVP_MD *mgf1md = NULL, *md = NULL;
467         RSA_PSS_PARAMS *pss;
468         X509_ALGOR *maskHash;
469         EVP_PKEY_CTX *pkctx;
470         /* Sanity check: make sure it is PSS */
471         if (OBJ_obj2nid(sigalg->algorithm) != NID_rsassaPss)
472                 {
473                 RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_UNSUPPORTED_SIGNATURE_TYPE);
474                 return -1;
475                 }
476         /* Decode PSS parameters */
477         pss = rsa_pss_decode(sigalg, &maskHash);

479         if (pss == NULL)
480                 {
481                 RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_INVALID_PSS_PARAMETERS);
482                 goto err;
483                 }
484         /* Check mask and lookup mask hash algorithm */
485         if (pss->maskGenAlgorithm)
486                 {
487                 if (OBJ_obj2nid(pss->maskGenAlgorithm->algorithm) != NID_mgf1)
488                         {
489                         RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_UNSUPPORTED_MASK_ALG
490                         goto err;
491                         }
492                 if (!maskHash)
493                         {
494                         RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_UNSUPPORTED_MASK_PAR
495                         goto err;
496                         }
497                 mgf1md = EVP_get_digestbyobj(maskHash->algorithm);
498                 if (mgf1md == NULL)
499                         {
500                         RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_UNKNOWN_MASK_DIGEST)
501                         goto err;
502                         }
503                 }
504         else
505                 mgf1md = EVP_sha1();

507         if (pss->hashAlgorithm)
508                 {
509                 md = EVP_get_digestbyobj(pss->hashAlgorithm->algorithm);
510                 if (md == NULL)
511                         {
512                         RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_UNKNOWN_PSS_DIGEST);
513                         goto err;
514                         }
515                 }
516         else
517                 md = EVP_sha1();

519         if (pss->saltLength)
520                 {
521                 saltlen = ASN1_INTEGER_get(pss->saltLength);

523                 /* Could perform more salt length sanity checks but the main
```

```
524                  * RSA routines will trap other invalid values anyway.
525                  */
526                 if (saltlen < 0)
527                         {
528                         RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_INVALID_SALT_LENGTH)
529                         goto err;
530                         }
531                 }
532         else
533                 saltlen = 20;
534
535         /* low-level routines support only trailer field 0xbc (value 1)
536          * and PKCS#1 says we should reject any other value anyway.
537          */
538         if (pss->trailerField && ASN1_INTEGER_get(pss->trailerField) != 1)
539                 {
540                 RSAerr(RSA_F_RSA_ITEM_VERIFY, RSA_R_INVALID_TRAILER);
541                 goto err;
542                 }
543
544         /* We have all parameters now set up context */
545
546         if (!EVP_DigestVerifyInit(ctx, &pkctx, md, NULL, pkey))
547                 goto err;
548
549         if (EVP_PKEY_CTX_set_rsa_padding(pkctx, RSA_PKCS1_PSS_PADDING) <= 0)
550                 goto err;
551
552         if (EVP_PKEY_CTX_set_rsa_pss_saltlen(pkctx, saltlen) <= 0)
553                 goto err;
554
555         if (EVP_PKEY_CTX_set_rsa_mgf1_md(pkctx, mgf1md) <= 0)
556                 goto err;
557         /* Carry on */
558         rv = 2;
559
560         err:
561         RSA_PSS_PARAMS_free(pss);
562         if (maskHash)
563                 X509_ALGOR_free(maskHash);
564         return rv;
565         }
566
567 static int rsa_item_sign(EVP_MD_CTX *ctx, const ASN1_ITEM *it, void *asn,
568                                 X509_ALGOR *alg1, X509_ALGOR *alg2,
569                                 ASN1_BIT_STRING *sig)
570         {
571         int pad_mode;
572         EVP_PKEY_CTX *pkctx = ctx->pctx;
573         if (EVP_PKEY_CTX_get_rsa_padding(pkctx, &pad_mode) <= 0)
574                 return 0;
575         if (pad_mode == RSA_PKCS1_PADDING)
576                 return 2;
577         if (pad_mode == RSA_PKCS1_PSS_PADDING)
578                 {
579                 const EVP_MD *sigmd, *mgf1md;
580                 RSA_PSS_PARAMS *pss = NULL;
581                 X509_ALGOR *mgf1alg = NULL;
582                 ASN1_STRING *os1 = NULL, *os2 = NULL;
583                 EVP_PKEY *pk = EVP_PKEY_CTX_get0_pkey(pkctx);
584                 int saltlen, rv = 0;
585                 sigmd = EVP_MD_CTX_md(ctx);
586                 if (EVP_PKEY_CTX_get_rsa_mgf1_md(pkctx, &mgf1md) <= 0)
587                         goto err;
588                 if (!EVP_PKEY_CTX_get_rsa_pss_saltlen(pkctx, &saltlen))
589                         goto err;
```



```
590                 if (saltlen == -1)
591                         saltlen = EVP_MD_size(sigmd);
592                 else if (saltlen == -2)
593                         {
594                         saltlen = EVP_PKEY_size(pk) - EVP_MD_size(sigmd) - 2;
595                         if (((EVP_PKEY_bits(pk) - 1) & 0x7) == 0)
596                                 saltlen--;
597                         }
598                 pss = RSA_PSS_PARAMS_new();
599                 if (!pss)
600                         goto err;
601                 if (saltlen != 20)
602                         {
603                         pss->saltLength = ASN1_INTEGER_new();
604                         if (!pss->saltLength)
605                                 goto err;
606                         if (!ASN1_INTEGER_set(pss->saltLength, saltlen))
607                                 goto err;
608                         }
609                 if (EVP_MD_type(sigmd) != NID_sha1)
610                         {
611                         pss->hashAlgorithm = X509_ALGOR_new();
612                         if (!pss->hashAlgorithm)
613                                 goto err;
614                         X509_ALGOR_set_md(pss->hashAlgorithm, sigmd);
615                         }
616                 if (EVP_MD_type(mgf1md) != NID_sha1)
617                         {
618                         ASN1_STRING *stmp = NULL;
619                         /* need to embed algorithm ID inside another */
620                         mgf1alg = X509_ALGOR_new();
621                         X509_ALGOR_set_md(mgf1alg, mgf1md);
622                         if (!ASN1_item_pack(mgf1alg, ASN1_ITEM_rptr(X509_ALGOR),
623                                                                 &stmp))
624                                         goto err;
625                         pss->maskGenAlgorithm = X509_ALGOR_new();
626                         if (!pss->maskGenAlgorithm)
627                                 goto err;
628                         X509_ALGOR_set0(pss->maskGenAlgorithm,
629                                         OBJ_nid2obj(NID_mgf1),
630                                         V_ASN1_SEQUENCE, stmp);
631                         }
632                 /* Finally create string with pss parameter encoding. */
633                 if (!ASN1_item_pack(pss, ASN1_ITEM_rptr(RSA_PSS_PARAMS), &os1))
634                         goto err;
635                 if (alg2)
636                         {
637                         os2 = ASN1_STRING_dup(os1);
638                         if (!os2)
639                                 goto err;
640                         X509_ALGOR_set0(alg2, OBJ_nid2obj(NID_rsassaPss),
641                                                 V_ASN1_SEQUENCE, os2);
642                         }
643                 X509_ALGOR_set0(alg1, OBJ_nid2obj(NID_rsassaPss),
644                                         V_ASN1_SEQUENCE, os1);
645                 os1 = os2 = NULL;
646                 rv = 3;
647                 err:
648                 if (mgf1alg)
649                         X509_ALGOR_free(mgf1alg);
650                 if (pss)
651                         RSA_PSS_PARAMS_free(pss);
652                 if (os1)
653                         ASN1_STRING_free(os1);
654                 return rv;
```

```
656                 }
657         return 2;
658         }

660 const EVP_PKEY_ASN1_METHOD rsa_asn1_meths[] =
661         {
662                 {
663                 EVP_PKEY_RSA,
664                 EVP_PKEY_RSA,
665                 ASN1_PKEY_SIGPARAM_NULL,

667                 "RSA",
668                 "OpenSSL RSA method",

670                 rsa_pub_decode,
671                 rsa_pub_encode,
672                 rsa_pub_cmp,
673                 rsa_pub_print,

675                 rsa_priv_decode,
676                 rsa_priv_encode,
677                 rsa_priv_print,

679                 int_rsa_size,
680                 rsa_bits,

682                 0,0,0,0,0,0,

684                 rsa_sig_print,
685                 int_rsa_free,
686                 rsa_pkey_ctrl,
687                 old_rsa_priv_decode,
688                 old_rsa_priv_encode,
689                 rsa_item_verify,
690                 rsa_item_sign
691                 },

693                 {
694                 EVP_PKEY_RSA2,
695                 EVP_PKEY_RSA,
696                 ASN1_PKEY_ALIAS
697                 }
698         };
699 #endif /* ! codereview */
```

```
**********************************************************
    4472 Wed Aug 13 19:53:15 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* rsa_asn1.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/x509.h>
#include <openssl/asn1t.h>

/* Override the default free and new methods */
static int rsa_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
                                                                void *exarg)
{
        if(operation == ASN1_OP_NEW_PRE) {
                *pval = (ASN1_VALUE *)RSA_new();
                if(*pval) return 2;
                return 0;
        } else if(operation == ASN1_OP_FREE_PRE) {
                RSA_free((RSA *)*pval);
                *pval = NULL;
                return 2;
        }
        return 1;
}

ASN1_SEQUENCE_cb(RSAPrivateKey, rsa_cb) = {
        ASN1_SIMPLE(RSA, version, LONG),
        ASN1_SIMPLE(RSA, n, BIGNUM),
        ASN1_SIMPLE(RSA, e, BIGNUM),
        ASN1_SIMPLE(RSA, d, BIGNUM),
        ASN1_SIMPLE(RSA, p, BIGNUM),
        ASN1_SIMPLE(RSA, q, BIGNUM),
        ASN1_SIMPLE(RSA, dmp1, BIGNUM),
        ASN1_SIMPLE(RSA, dmq1, BIGNUM),
        ASN1_SIMPLE(RSA, iqmp, BIGNUM)
} ASN1_SEQUENCE_END_cb(RSA, RSAPrivateKey)


ASN1_SEQUENCE_cb(RSAPublicKey, rsa_cb) = {
        ASN1_SIMPLE(RSA, n, BIGNUM),
        ASN1_SIMPLE(RSA, e, BIGNUM),
} ASN1_SEQUENCE_END_cb(RSA, RSAPublicKey)

ASN1_SEQUENCE(RSA_PSS_PARAMS) = {
        ASN1_EXP_OPT(RSA_PSS_PARAMS, hashAlgorithm, X509_ALGOR,0),
        ASN1_EXP_OPT(RSA_PSS_PARAMS, maskGenAlgorithm, X509_ALGOR,1),
        ASN1_EXP_OPT(RSA_PSS_PARAMS, saltLength, ASN1_INTEGER,2),
        ASN1_EXP_OPT(RSA_PSS_PARAMS, trailerField, ASN1_INTEGER,3)
} ASN1_SEQUENCE_END(RSA_PSS_PARAMS)

IMPLEMENT_ASN1_FUNCTIONS(RSA_PSS_PARAMS)

IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(RSA, RSAPrivateKey, RSAPrivateKey)

IMPLEMENT_ASN1_ENCODE_FUNCTIONS_const_fname(RSA, RSAPublicKey, RSAPublicKey)

RSA *RSAPublicKey_dup(RSA *rsa)
        {
        return ASN1_item_dup(ASN1_ITEM_rptr(RSAPublicKey), rsa);
        }

RSA *RSAPrivateKey_dup(RSA *rsa)
        {
        return ASN1_item_dup(ASN1_ITEM_rptr(RSAPrivateKey), rsa);
        }
#endif /* ! codereview */
```

```
**********************************************************
    5349 Wed Aug 13 19:53:15 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_chk.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_chk.c  -*- Mode: C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 */

#include <openssl/bn.h>
#include <openssl/err.h>
#include <openssl/rsa.h>


int RSA_check_key(const RSA *key)
        {
        BIGNUM *i, *j, *k, *l, *m;
        BN_CTX *ctx;
        int r;
        int ret=1;
```



```
        if (!key->p || !key->q || !key->n || !key->e || !key->d)
                {
                RSAerr(RSA_F_RSA_CHECK_KEY, RSA_R_VALUE_MISSING);
                return 0;
                }

        i = BN_new();
        j = BN_new();
        k = BN_new();
        l = BN_new();
        m = BN_new();
        ctx = BN_CTX_new();
        if (i == NULL || j == NULL || k == NULL || l == NULL ||
                m == NULL || ctx == NULL)
                {
                ret = -1;
                RSAerr(RSA_F_RSA_CHECK_KEY, ERR_R_MALLOC_FAILURE);
                goto err;
                }

        /* p prime? */
        r = BN_is_prime_ex(key->p, BN_prime_checks, NULL, NULL);
        if (r != 1)
                {
                ret = r;
                if (r != 0)
                        goto err;
                RSAerr(RSA_F_RSA_CHECK_KEY, RSA_R_P_NOT_PRIME);
                }

        /* q prime? */
        r = BN_is_prime_ex(key->q, BN_prime_checks, NULL, NULL);
        if (r != 1)
                {
                ret = r;
                if (r != 0)
                        goto err;
                RSAerr(RSA_F_RSA_CHECK_KEY, RSA_R_Q_NOT_PRIME);
                }

        /* n = p*q? */
        r = BN_mul(i, key->p, key->q, ctx);
        if (!r) { ret = -1; goto err; }

        if (BN_cmp(i, key->n) != 0)
                {
                ret = 0;
                RSAerr(RSA_F_RSA_CHECK_KEY, RSA_R_N_DOES_NOT_EQUAL_P_Q);
                }

        /* d*e = 1  mod lcm(p-1,q-1)? */

        r = BN_sub(i, key->p, BN_value_one());
        if (!r) { ret = -1; goto err; }
        r = BN_sub(j, key->q, BN_value_one());
        if (!r) { ret = -1; goto err; }

        /* now compute k = lcm(i,j) */
        r = BN_mul(l, i, j, ctx);
        if (!r) { ret = -1; goto err; }
        r = BN_gcd(m, i, j, ctx);
        if (!r) { ret = -1; goto err; }
        r = BN_div(k, NULL, l, m, ctx); /* remainder is 0 */
        if (!r) { ret = -1; goto err; }
```

```
128         r = BN_mod_mul(i, key->d, key->e, k, ctx);
129         if (!r) { ret = -1; goto err; }

131         if (!BN_is_one(i))
132                 {
133                 ret = 0;
134                 RSAerr(RSA_F_RSA_CHECK_KEY, RSA_R_D_E_NOT_CONGRUENT_TO_1);
135                 }

137         if (key->dmp1 != NULL && key->dmq1 != NULL && key->iqmp != NULL)
138                 {
139                 /* dmp1 = d mod (p-1)? */
140                 r = BN_sub(i, key->p, BN_value_one());
141                 if (!r) { ret = -1; goto err; }

143                 r = BN_mod(j, key->d, i, ctx);
144                 if (!r) { ret = -1; goto err; }

146                 if (BN_cmp(j, key->dmp1) != 0)
147                         {
148                         ret = 0;
149                         RSAerr(RSA_F_RSA_CHECK_KEY,
150                                 RSA_R_DMP1_NOT_CONGRUENT_TO_D);
151                         }

153                 /* dmq1 = d mod (q-1)? */
154                 r = BN_sub(i, key->q, BN_value_one());
155                 if (!r) { ret = -1; goto err; }

157                 r = BN_mod(j, key->d, i, ctx);
158                 if (!r) { ret = -1; goto err; }

160                 if (BN_cmp(j, key->dmq1) != 0)
161                         {
162                         ret = 0;
163                         RSAerr(RSA_F_RSA_CHECK_KEY,
164                                 RSA_R_DMQ1_NOT_CONGRUENT_TO_D);
165                         }

167                 /* iqmp = q^-1 mod p? */
168                 if(!BN_mod_inverse(i, key->q, key->p, ctx))
169                         {
170                         ret = -1;
171                         goto err;
172                         }

174                 if (BN_cmp(i, key->iqmp) != 0)
175                         {
176                         ret = 0;
177                         RSAerr(RSA_F_RSA_CHECK_KEY,
178                                 RSA_R_IQMP_NOT_INVERSE_OF_Q);
179                         }
180                 }

182  err:
183         if (i != NULL) BN_free(i);
184         if (j != NULL) BN_free(j);
185         if (k != NULL) BN_free(k);
186         if (l != NULL) BN_free(l);
187         if (m != NULL) BN_free(m);
188         if (ctx != NULL) BN_CTX_free(ctx);
189         return (ret);
190         }
191 #endif /* ! codereview */
```

```
**********************************************************
    7533 Wed Aug 13 19:53:15 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_crpt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/crypto.h>
#include "cryptlib.h"
```



```
#include <openssl/lhash.h>
#include <openssl/bn.h>
#include <openssl/rsa.h>
#include <openssl/rand.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

int RSA_size(const RSA *r)
        {
        return(BN_num_bytes(r->n));
        }

int RSA_public_encrypt(int flen, const unsigned char *from, unsigned char *to,
             RSA *rsa, int padding)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(rsa->meth->flags & RSA_FLAG_FIPS_METHOD)
                        && !(rsa->flags & RSA_FLAG_NON_FIPS_ALLOW))
                {
                RSAerr(RSA_F_RSA_PUBLIC_ENCRYPT, RSA_R_NON_FIPS_RSA_METHOD);
                return -1;
                }
#endif
        return(rsa->meth->rsa_pub_enc(flen, from, to, rsa, padding));
        }

int RSA_private_encrypt(int flen, const unsigned char *from, unsigned char *to,
             RSA *rsa, int padding)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(rsa->meth->flags & RSA_FLAG_FIPS_METHOD)
                        && !(rsa->flags & RSA_FLAG_NON_FIPS_ALLOW))
                {
                RSAerr(RSA_F_RSA_PRIVATE_ENCRYPT, RSA_R_NON_FIPS_RSA_METHOD);
                return -1;
                }
#endif
        return(rsa->meth->rsa_priv_enc(flen, from, to, rsa, padding));
        }

int RSA_private_decrypt(int flen, const unsigned char *from, unsigned char *to,
             RSA *rsa, int padding)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(rsa->meth->flags & RSA_FLAG_FIPS_METHOD)
                        && !(rsa->flags & RSA_FLAG_NON_FIPS_ALLOW))
                {
                RSAerr(RSA_F_RSA_PRIVATE_DECRYPT, RSA_R_NON_FIPS_RSA_METHOD);
                return -1;
                }
#endif
        return(rsa->meth->rsa_priv_dec(flen, from, to, rsa, padding));
        }

int RSA_public_decrypt(int flen, const unsigned char *from, unsigned char *to,
             RSA *rsa, int padding)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(rsa->meth->flags & RSA_FLAG_FIPS_METHOD)
                        && !(rsa->flags & RSA_FLAG_NON_FIPS_ALLOW))
                {
                RSAerr(RSA_F_RSA_PUBLIC_DECRYPT, RSA_R_NON_FIPS_RSA_METHOD);
                return -1;
                }
#endif
```

```
128         return(rsa->meth->rsa_pub_dec(flen, from, to, rsa, padding));
129         }

131 int RSA_flags(const RSA *r)
132         {
133         return((r == NULL)?0:r->meth->flags);
134         }

136 void RSA_blinding_off(RSA *rsa)
137         {
138         if (rsa->blinding != NULL)
139                 {
140                 BN_BLINDING_free(rsa->blinding);
141                 rsa->blinding=NULL;
142                 }
143         rsa->flags &= ~RSA_FLAG_BLINDING;
144         rsa->flags |= RSA_FLAG_NO_BLINDING;
145         }

147 int RSA_blinding_on(RSA *rsa, BN_CTX *ctx)
148         {
149         int ret=0;

151         if (rsa->blinding != NULL)
152                 RSA_blinding_off(rsa);

154         rsa->blinding = RSA_setup_blinding(rsa, ctx);
155         if (rsa->blinding == NULL)
156                 goto err;

158         rsa->flags |= RSA_FLAG_BLINDING;
159         rsa->flags &= ~RSA_FLAG_NO_BLINDING;
160         ret=1;
161 err:
162         return(ret);
163         }

165 static BIGNUM *rsa_get_public_exp(const BIGNUM *d, const BIGNUM *p,
166         const BIGNUM *q, BN_CTX *ctx)
167 {
168         BIGNUM *ret = NULL, *r0, *r1, *r2;

170         if (d == NULL || p == NULL || q == NULL)
171                 return NULL;

173         BN_CTX_start(ctx);
174         r0 = BN_CTX_get(ctx);
175         r1 = BN_CTX_get(ctx);
176         r2 = BN_CTX_get(ctx);
177         if (r2 == NULL)
178                 goto err;

180         if (!BN_sub(r1, p, BN_value_one())) goto err;
181         if (!BN_sub(r2, q, BN_value_one())) goto err;
182         if (!BN_mul(r0, r1, r2, ctx)) goto err;

184         ret = BN_mod_inverse(NULL, d, r0, ctx);
185 err:
186         BN_CTX_end(ctx);
187         return ret;
188 }

190 BN_BLINDING *RSA_setup_blinding(RSA *rsa, BN_CTX *in_ctx)
191 {
192         BIGNUM local_n;
193         BIGNUM *e,*n;
```



```
194         BN_CTX *ctx;
195         BN_BLINDING *ret = NULL;

197         if (in_ctx == NULL)
198                 {
199                 if ((ctx = BN_CTX_new()) == NULL) return 0;
200                 }
201         else
202                 ctx = in_ctx;

204         BN_CTX_start(ctx);
205         e  = BN_CTX_get(ctx);
206         if (e == NULL)
207                 {
208                 RSAerr(RSA_F_RSA_SETUP_BLINDING, ERR_R_MALLOC_FAILURE);
209                 goto err;
210                 }

212         if (rsa->e == NULL)
213                 {
214                 e = rsa_get_public_exp(rsa->d, rsa->p, rsa->q, ctx);
215                 if (e == NULL)
216                         {
217                         RSAerr(RSA_F_RSA_SETUP_BLINDING, RSA_R_NO_PUBLIC_EXPONEN
218                         goto err;
219                         }
220                 }
221         else
222                 e = rsa->e;


225         if ((RAND_status() == 0) && rsa->d != NULL && rsa->d->d != NULL)
226                 {
227                 /* if PRNG is not properly seeded, resort to secret
228                  * exponent as unpredictable seed */
229                 RAND_add(rsa->d->d, rsa->d->dmax * sizeof rsa->d->d[0], 0.0);
230                 }

232         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
233                 {
234                 /* Set BN_FLG_CONSTTIME flag */
235                 n = &local_n;
236                 BN_with_flags(n, rsa->n, BN_FLG_CONSTTIME);
237                 }
238         else
239                 n = rsa->n;

241         ret = BN_BLINDING_create_param(NULL, e, n, ctx,
242                         rsa->meth->bn_mod_exp, rsa->_method_mod_n);
243         if (ret == NULL)
244                 {
245                 RSAerr(RSA_F_RSA_SETUP_BLINDING, ERR_R_BN_LIB);
246                 goto err;
247                 }
248         CRYPTO_THREADID_current(BN_BLINDING_thread_id(ret));
249 err:
250         BN_CTX_end(ctx);
251         if (in_ctx == NULL)
252                 BN_CTX_free(ctx);
253         if(rsa->e == NULL)
254                 BN_free(e);

256         return ret;
257 }
258 #endif /* ! codereview */
```

```
**********************************************************
    3557 Wed Aug 13 19:53:16 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_depr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_depr.c */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NB: This file contains deprecated functions (compatibility wrappers to the
 * "new" versions). */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
```



```
#include <openssl/bn.h>
#include <openssl/rsa.h>

#ifdef OPENSSL_NO_DEPRECATED

static void *dummy=&dummy;

#else

RSA *RSA_generate_key(int bits, unsigned long e_value,
             void (*callback)(int,int,void *), void *cb_arg)
        {
        BN_GENCB cb;
        int i;
        RSA *rsa = RSA_new();
        BIGNUM *e = BN_new();

        if(!rsa || !e) goto err;

        /* The problem is when building with 8, 16, or 32 BN_ULONG,
         * unsigned long can be larger */
        for (i=0; i<(int)sizeof(unsigned long)*8; i++)
                {
                if (e_value & (1UL<<i))
                        if (BN_set_bit(e,i) == 0)
                                goto err;
                }

        BN_GENCB_set_old(&cb, callback, cb_arg);

        if(RSA_generate_key_ex(rsa, bits, e, &cb)) {
                BN_free(e);
                return rsa;
        }
err:
        if(e) BN_free(e);
        if(rsa) RSA_free(rsa);
        return 0;
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
   25161 Wed Aug 13 19:53:16 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_eay.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_eay.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
#include <openssl/rsa.h>
#include <openssl/rand.h>

#ifndef RSA_NULL

static int RSA_eay_public_encrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
static int RSA_eay_private_encrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
static int RSA_eay_public_decrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
static int RSA_eay_private_decrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
```

```
128 static int RSA_eay_mod_exp(BIGNUM *r0, const BIGNUM *i, RSA *rsa, BN_CTX *ctx);
129 static int RSA_eay_init(RSA *rsa);
130 static int RSA_eay_finish(RSA *rsa);
131 static RSA_METHOD rsa_pkcs1_eay_meth={
132         "Eric Young's PKCS#1 RSA",
133         RSA_eay_public_encrypt,
134         RSA_eay_public_decrypt, /* signature verification */
135         RSA_eay_private_encrypt, /* signing */
136         RSA_eay_private_decrypt,
137         RSA_eay_mod_exp,
138         BN_mod_exp_mont, /* XXX probably we should not use Montgomery if  e == 3
139         RSA_eay_init,
140         RSA_eay_finish,
141         0, /* flags */
142         NULL,
143         0, /* rsa_sign */
144         0, /* rsa_verify */
145         NULL /* rsa_keygen */
146         };

148 const RSA_METHOD *RSA_PKCS1_SSLeay(void)
149         {
150         return(&rsa_pkcs1_eay_meth);
151         }

153 static int RSA_eay_public_encrypt(int flen, const unsigned char *from,
154             unsigned char *to, RSA *rsa, int padding)
155         {
156         BIGNUM *f,*ret;
157         int i,j,k,num=0,r= -1;
158         unsigned char *buf=NULL;
159         BN_CTX *ctx=NULL;

161         if (BN_num_bits(rsa->n) > OPENSSL_RSA_MAX_MODULUS_BITS)
162                 {
163                 RSAerr(RSA_F_RSA_EAY_PUBLIC_ENCRYPT, RSA_R_MODULUS_TOO_LARGE);
164                 return -1;
165                 }

167         if (BN_ucmp(rsa->n, rsa->e) <= 0)
168                 {
169                 RSAerr(RSA_F_RSA_EAY_PUBLIC_ENCRYPT, RSA_R_BAD_E_VALUE);
170                 return -1;
171                 }

173         /* for large moduli, enforce exponent limit */
174         if (BN_num_bits(rsa->n) > OPENSSL_RSA_SMALL_MODULUS_BITS)
175                 {
176                 if (BN_num_bits(rsa->e) > OPENSSL_RSA_MAX_PUBEXP_BITS)
177                         {
178                         RSAerr(RSA_F_RSA_EAY_PUBLIC_ENCRYPT, RSA_R_BAD_E_VALUE);
179                         return -1;
180                         }
181                 }

183         if ((ctx=BN_CTX_new()) == NULL) goto err;
184         BN_CTX_start(ctx);
185         f = BN_CTX_get(ctx);
186         ret = BN_CTX_get(ctx);
187         num=BN_num_bytes(rsa->n);
188         buf = OPENSSL_malloc(num);
189         if (!f || !ret || !buf)
190                 {
191                 RSAerr(RSA_F_RSA_EAY_PUBLIC_ENCRYPT,ERR_R_MALLOC_FAILURE);
192                 goto err;
193                 }
```



```
195         switch (padding)
196                 {
197         case RSA_PKCS1_PADDING:
198                 i=RSA_padding_add_PKCS1_type_2(buf,num,from,flen);
199                 break;
200 #ifndef OPENSSL_NO_SHA
201         case RSA_PKCS1_OAEP_PADDING:
202                 i=RSA_padding_add_PKCS1_OAEP(buf,num,from,flen,NULL,0);
203                 break;
204 #endif
205         case RSA_SSLV23_PADDING:
206                 i=RSA_padding_add_SSLv23(buf,num,from,flen);
207                 break;
208         case RSA_NO_PADDING:
209                 i=RSA_padding_add_none(buf,num,from,flen);
210                 break;
211         default:
212                 RSAerr(RSA_F_RSA_EAY_PUBLIC_ENCRYPT,RSA_R_UNKNOWN_PADDING_TYPE);
213                 goto err;
214                 }
215         if (i <= 0) goto err;

217         if (BN_bin2bn(buf,num,f) == NULL) goto err;

219         if (BN_ucmp(f, rsa->n) >= 0)
220                 {
221                 /* usually the padding functions would catch this */
222                 RSAerr(RSA_F_RSA_EAY_PUBLIC_ENCRYPT,RSA_R_DATA_TOO_LARGE_FOR_MOD
223                 goto err;
224                 }

226         if (rsa->flags & RSA_FLAG_CACHE_PUBLIC)
227                 if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_n, CRYPTO_LOCK_RSA
228                         goto err;

230         if (!rsa->meth->bn_mod_exp(ret,f,rsa->e,rsa->n,ctx,
231                 rsa->_method_mod_n)) goto err;

233         /* put in leading 0 bytes if the number is less than the
234          * length of the modulus */
235         j=BN_num_bytes(ret);
236         i=BN_bn2bin(ret,&(to[num-j]));
237         for (k=0; k<(num-i); k++)
238                 to[k]=0;

240         r=num;
241 err:
242         if (ctx != NULL)
243                 {
244                 BN_CTX_end(ctx);
245                 BN_CTX_free(ctx);
246                 }
247         if (buf != NULL)
248                 {
249                 OPENSSL_cleanse(buf,num);
250                 OPENSSL_free(buf);
251                 }
252         return(r);
253         }

255 static BN_BLINDING *rsa_get_blinding(RSA *rsa, int *local, BN_CTX *ctx)
256 {
257         BN_BLINDING *ret;
258         int got_write_lock = 0;
259         CRYPTO_THREADID cur;
```

```
261         CRYPTO_r_lock(CRYPTO_LOCK_RSA);

263         if (rsa->blinding == NULL)
264                 {
265                 CRYPTO_r_unlock(CRYPTO_LOCK_RSA);
266                 CRYPTO_w_lock(CRYPTO_LOCK_RSA);
267                 got_write_lock = 1;

269                 if (rsa->blinding == NULL)
270                         rsa->blinding = RSA_setup_blinding(rsa, ctx);
271                 }

273         ret = rsa->blinding;
274         if (ret == NULL)
275                 goto err;

277         CRYPTO_THREADID_current(&cur);
278         if (!CRYPTO_THREADID_cmp(&cur, BN_BLINDING_thread_id(ret)))
279                 {
280                 /* rsa->blinding is ours! */

282                 *local = 1;
283                 }
284         else
285                 {
286                 /* resort to rsa->mt_blinding instead */

288                 *local = 0; /* instructs rsa_blinding_convert(), rsa_blinding_in
289                              * that the BN_BLINDING is shared, meaning that acce
290                              * require locks, and that the blinding factor must
291                              * stored outside the BN_BLINDING
292                              */

294                 if (rsa->mt_blinding == NULL)
295                         {
296                         if (!got_write_lock)
297                                 {
298                                 CRYPTO_r_unlock(CRYPTO_LOCK_RSA);
299                                 CRYPTO_w_lock(CRYPTO_LOCK_RSA);
300                                 got_write_lock = 1;
301                                 }

303                         if (rsa->mt_blinding == NULL)
304                                 rsa->mt_blinding = RSA_setup_blinding(rsa, ctx);
305                         }
306                 ret = rsa->mt_blinding;
307                 }

309  err:
310         if (got_write_lock)
311                 CRYPTO_w_unlock(CRYPTO_LOCK_RSA);
312         else
313                 CRYPTO_r_unlock(CRYPTO_LOCK_RSA);
314         return ret;
315 }

317 static int rsa_blinding_convert(BN_BLINDING *b, BIGNUM *f, BIGNUM *unblind,
318         BN_CTX *ctx)
319         {
320         if (unblind == NULL)
321                 /* Local blinding: store the unblinding factor
322                  * in BN_BLINDING. */
323                 return BN_BLINDING_convert_ex(f, NULL, b, ctx);
324         else
325                 {
```



```
326                 /* Shared blinding: store the unblinding factor
327                  * outside BN_BLINDING. */
328                 int ret;
329                 CRYPTO_w_lock(CRYPTO_LOCK_RSA_BLINDING);
330                 ret = BN_BLINDING_convert_ex(f, unblind, b, ctx);
331                 CRYPTO_w_unlock(CRYPTO_LOCK_RSA_BLINDING);
332                 return ret;
333                 }
334         }

336 static int rsa_blinding_invert(BN_BLINDING *b, BIGNUM *f, BIGNUM *unblind,
337         BN_CTX *ctx)
338         {
339         /* For local blinding, unblind is set to NULL, and BN_BLINDING_invert_ex
340          * will use the unblinding factor stored in BN_BLINDING.
341          * If BN_BLINDING is shared between threads, unblind must be non-null:
342          * BN_BLINDING_invert_ex will then use the local unblinding factor,
343          * and will only read the modulus from BN_BLINDING.
344          * In both cases it's safe to access the blinding without a lock.
345          */
346         return BN_BLINDING_invert_ex(f, unblind, b, ctx);
347         }

349 /* signing */
350 static int RSA_eay_private_encrypt(int flen, const unsigned char *from,
351               unsigned char *to, RSA *rsa, int padding)
352         {
353         BIGNUM *f, *ret, *res;
354         int i,j,k,num=0,r= -1;
355         unsigned char *buf=NULL;
356         BN_CTX *ctx=NULL;
357         int local_blinding = 0;
358         /* Used only if the blinding structure is shared. A non-NULL unblind
359          * instructs rsa_blinding_convert() and rsa_blinding_invert() to store
360          * the unblinding factor outside the blinding structure. */
361         BIGNUM *unblind = NULL;
362         BN_BLINDING *blinding = NULL;

364         if ((ctx=BN_CTX_new()) == NULL) goto err;
365         BN_CTX_start(ctx);
366         f   = BN_CTX_get(ctx);
367         ret = BN_CTX_get(ctx);
368         num = BN_num_bytes(rsa->n);
369         buf = OPENSSL_malloc(num);
370         if(!f || !ret || !buf)
371                 {
372                 RSAerr(RSA_F_RSA_EAY_PRIVATE_ENCRYPT,ERR_R_MALLOC_FAILURE);
373                 goto err;
374                 }

376         switch (padding)
377                 {
378         case RSA_PKCS1_PADDING:
379                 i=RSA_padding_add_PKCS1_type_1(buf,num,from,flen);
380                 break;
381         case RSA_X931_PADDING:
382                 i=RSA_padding_add_X931(buf,num,from,flen);
383                 break;
384         case RSA_NO_PADDING:
385                 i=RSA_padding_add_none(buf,num,from,flen);
386                 break;
387         case RSA_SSLV23_PADDING:
388         default:
389                 RSAerr(RSA_F_RSA_EAY_PRIVATE_ENCRYPT,RSA_R_UNKNOWN_PADDING_TYPE)
390                 goto err;
391                 }
```

```
392         if (i <= 0) goto err;

394         if (BN_bin2bn(buf,num,f) == NULL) goto err;

396         if (BN_ucmp(f, rsa->n) >= 0)
397                 {
398                 /* usually the padding functions would catch this */
399                 RSAerr(RSA_F_RSA_EAY_PRIVATE_ENCRYPT,RSA_R_DATA_TOO_LARGE_FOR_MO
400                 goto err;
401                 }

403         if (!(rsa->flags & RSA_FLAG_NO_BLINDING))
404                 {
405                 blinding = rsa_get_blinding(rsa, &local_blinding, ctx);
406                 if (blinding == NULL)
407                         {
408                         RSAerr(RSA_F_RSA_EAY_PRIVATE_ENCRYPT, ERR_R_INTERNAL_ERR
409                         goto err;
410                         }
411                 }

413         if (blinding != NULL)
414                 {
415                 if (!local_blinding && ((unblind = BN_CTX_get(ctx)) == NULL))
416                         {
417                         RSAerr(RSA_F_RSA_EAY_PRIVATE_ENCRYPT,ERR_R_MALLOC_FAILUR
418                         goto err;
419                         }
420                 if (!rsa_blinding_convert(blinding, f, unblind, ctx))
421                         goto err;
422                 }

424         if ( (rsa->flags & RSA_FLAG_EXT_PKEY) ||
425                 ((rsa->p != NULL) &&
426                 (rsa->q != NULL) &&
427                 (rsa->dmp1 != NULL) &&
428                 (rsa->dmq1 != NULL) &&
429                 (rsa->iqmp != NULL)) )
430                 {
431                 if (!rsa->meth->rsa_mod_exp(ret, f, rsa, ctx)) goto err;
432                 }
433         else
434                 {
435                 BIGNUM local_d;
436                 BIGNUM *d = NULL;

438                 if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
439                         {
440                         BN_init(&local_d);
441                         d = &local_d;
442                         BN_with_flags(d, rsa->d, BN_FLG_CONSTTIME);
443                         }
444                 else
445                         d= rsa->d;

447                 if (rsa->flags & RSA_FLAG_CACHE_PUBLIC)
448                         if(!BN_MONT_CTX_set_locked(&rsa->_method_mod_n, CRYPTO_L
449                                 goto err;

451                 if (!rsa->meth->bn_mod_exp(ret,f,d,rsa->n,ctx,
452                                 rsa->_method_mod_n)) goto err;
453                 }

455         if (blinding)
456                 if (!rsa_blinding_invert(blinding, ret, unblind, ctx))
457                         goto err;
```



```
459         if (padding == RSA_X931_PADDING)
460                 {
461                 BN_sub(f, rsa->n, ret);
462                 if (BN_cmp(ret, f) > 0)
463                         res = f;
464                 else
465                         res = ret;
466                 }
467         else
468                 res = ret;

470         /* put in leading 0 bytes if the number is less than the
471          * length of the modulus */
472         j=BN_num_bytes(res);
473         i=BN_bn2bin(res,&(to[num-j]));
474         for (k=0; k<(num-i); k++)
475                 to[k]=0;

477         r=num;
478 err:
479         if (ctx != NULL)
480                 {
481                 BN_CTX_end(ctx);
482                 BN_CTX_free(ctx);
483                 }
484         if (buf != NULL)
485                 {
486                 OPENSSL_cleanse(buf,num);
487                 OPENSSL_free(buf);
488                 }
489         return(r);
490         }

492 static int RSA_eay_private_decrypt(int flen, const unsigned char *from,
493                 unsigned char *to, RSA *rsa, int padding)
494         {
495         BIGNUM *f, *ret;
496         int j,num=0,r= -1;
497         unsigned char *p;
498         unsigned char *buf=NULL;
499         BN_CTX *ctx=NULL;
500         int local_blinding = 0;
501         /* Used only if the blinding structure is shared. A non-NULL unblind
502          * instructs rsa_blinding_convert() and rsa_blinding_invert() to store
503          * the unblinding factor outside the blinding structure. */
504         BIGNUM *unblind = NULL;
505         BN_BLINDING *blinding = NULL;

507         if((ctx = BN_CTX_new()) == NULL) goto err;
508         BN_CTX_start(ctx);
509         f   = BN_CTX_get(ctx);
510         ret = BN_CTX_get(ctx);
511         num = BN_num_bytes(rsa->n);
512         buf = OPENSSL_malloc(num);
513         if(!f || !ret || !buf)
514                 {
515                 RSAerr(RSA_F_RSA_EAY_PRIVATE_DECRYPT,ERR_R_MALLOC_FAILURE);
516                 goto err;
517                 }

519         /* This check was for equality but PGP does evil things
520          * and chops off the top '0' bytes */
521         if (flen > num)
522                 {
523                 RSAerr(RSA_F_RSA_EAY_PRIVATE_DECRYPT,RSA_R_DATA_GREATER_THAN_MOD
```

```
524                 goto err;
525                 }

527         /* make data into a big number */
528         if (BN_bin2bn(from,(int)flen,f) == NULL) goto err;

530         if (BN_ucmp(f, rsa->n) >= 0)
531                 {
532                 RSAerr(RSA_F_RSA_EAY_PRIVATE_DECRYPT,RSA_R_DATA_TOO_LARGE_FOR_MO
533                 goto err;
534                 }

536         if (!(rsa->flags & RSA_FLAG_NO_BLINDING))
537                 {
538                 blinding = rsa_get_blinding(rsa, &local_blinding, ctx);
539                 if (blinding == NULL)
540                         {
541                         RSAerr(RSA_F_RSA_EAY_PRIVATE_DECRYPT, ERR_R_INTERNAL_ERR
542                         goto err;
543                         }
544                 }

546         if (blinding != NULL)
547                 {
548                 if (!local_blinding && ((unblind = BN_CTX_get(ctx)) == NULL))
549                         {
550                         RSAerr(RSA_F_RSA_EAY_PRIVATE_DECRYPT,ERR_R_MALLOC_FAILUR
551                         goto err;
552                         }
553                 if (!rsa_blinding_convert(blinding, f, unblind, ctx))
554                         goto err;
555                 }

557         /* do the decrypt */
558         if ( (rsa->flags & RSA_FLAG_EXT_PKEY) ||
559                 ((rsa->p != NULL) &&
560                 (rsa->q != NULL) &&
561                 (rsa->dmp1 != NULL) &&
562                 (rsa->dmq1 != NULL) &&
563                 (rsa->iqmp != NULL)) )
564                 {
565                 if (!rsa->meth->rsa_mod_exp(ret, f, rsa, ctx)) goto err;
566                 }
567         else
568                 {
569                 BIGNUM local_d;
570                 BIGNUM *d = NULL;

572                 if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
573                         {
574                         d = &local_d;
575                         BN_with_flags(d, rsa->d, BN_FLG_CONSTTIME);
576                         }
577                 else
578                         d = rsa->d;

580                 if (rsa->flags & RSA_FLAG_CACHE_PUBLIC)
581                         if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_n, CRYPTO_
582                                 goto err;
583                 if (!rsa->meth->bn_mod_exp(ret,f,d,rsa->n,ctx,
584                                 rsa->_method_mod_n))
585                   goto err;
586                 }

588         if (blinding)
589                 if (!rsa_blinding_invert(blinding, ret, unblind, ctx))
```



```
590                         goto err;

592         p=buf;
593         j=BN_bn2bin(ret,p); /* j is only used with no-padding mode */

595         switch (padding)
596                 {
597         case RSA_PKCS1_PADDING:
598                 r=RSA_padding_check_PKCS1_type_2(to,num,buf,j,num);
599                 break;
600 #ifndef OPENSSL_NO_SHA
601         case RSA_PKCS1_OAEP_PADDING:
602                 r=RSA_padding_check_PKCS1_OAEP(to,num,buf,j,num,NULL,0);
603                 break;
604 #endif
605         case RSA_SSLV23_PADDING:
606                 r=RSA_padding_check_SSLv23(to,num,buf,j,num);
607                 break;
608         case RSA_NO_PADDING:
609                 r=RSA_padding_check_none(to,num,buf,j,num);
610                 break;
611         default:
612                 RSAerr(RSA_F_RSA_EAY_PRIVATE_DECRYPT,RSA_R_UNKNOWN_PADDING_TYPE)
613                 goto err;
614                 }
615         if (r < 0)
616                 RSAerr(RSA_F_RSA_EAY_PRIVATE_DECRYPT,RSA_R_PADDING_CHECK_FAILED)

618 err:
619         if (ctx != NULL)
620                 {
621                 BN_CTX_end(ctx);
622                 BN_CTX_free(ctx);
623                 }
624         if (buf != NULL)
625                 {
626                 OPENSSL_cleanse(buf,num);
627                 OPENSSL_free(buf);
628                 }
629         return(r);
630         }

632 /* signature verification */
633 static int RSA_eay_public_decrypt(int flen, const unsigned char *from,
634                 unsigned char *to, RSA *rsa, int padding)
635         {
636         BIGNUM *f,*ret;
637         int i,num=0,r= -1;
638         unsigned char *p;
639         unsigned char *buf=NULL;
640         BN_CTX *ctx=NULL;

642         if (BN_num_bits(rsa->n) > OPENSSL_RSA_MAX_MODULUS_BITS)
643                 {
644                 RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT, RSA_R_MODULUS_TOO_LARGE);
645                 return -1;
646                 }

648         if (BN_ucmp(rsa->n, rsa->e) <= 0)
649                 {
650                 RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT, RSA_R_BAD_E_VALUE);
651                 return -1;
652                 }

654         /* for large moduli, enforce exponent limit */
655         if (BN_num_bits(rsa->n) > OPENSSL_RSA_SMALL_MODULUS_BITS)
```

```
656                 {
657                 if (BN_num_bits(rsa->e) > OPENSSL_RSA_MAX_PUBEXP_BITS)
658                         {
659                         RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT, RSA_R_BAD_E_VALUE);
660                         return -1;
661                         }
662                 }

664         if((ctx = BN_CTX_new()) == NULL) goto err;
665         BN_CTX_start(ctx);
666         f = BN_CTX_get(ctx);
667         ret = BN_CTX_get(ctx);
668         num=BN_num_bytes(rsa->n);
669         buf = OPENSSL_malloc(num);
670         if(!f || !ret || !buf)
671                 {
672                 RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT,ERR_R_MALLOC_FAILURE);
673                 goto err;
674                 }

676         /* This check was for equality but PGP does evil things
677          * and chops off the top '0' bytes */
678         if (flen > num)
679                 {
680                 RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT,RSA_R_DATA_GREATER_THAN_MOD_
681                 goto err;
682                 }

684         if (BN_bin2bn(from,flen,f) == NULL) goto err;

686         if (BN_ucmp(f, rsa->n) >= 0)
687                 {
688                 RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT,RSA_R_DATA_TOO_LARGE_FOR_MOD
689                 goto err;
690                 }

692         if (rsa->flags & RSA_FLAG_CACHE_PUBLIC)
693                 if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_n, CRYPTO_LOCK_RSA
694                         goto err;

696         if (!rsa->meth->bn_mod_exp(ret,f,rsa->e,rsa->n,ctx,
697                 rsa->_method_mod_n)) goto err;

699         if ((padding == RSA_X931_PADDING) && ((ret->d[0] & 0xf) != 12))
700                 if (!BN_sub(ret, rsa->n, ret)) goto err;

702         p=buf;
703         i=BN_bn2bin(ret,p);

705         switch (padding)
706                 {
707         case RSA_PKCS1_PADDING:
708                 r=RSA_padding_check_PKCS1_type_1(to,num,buf,i,num);
709                 break;
710         case RSA_X931_PADDING:
711                 r=RSA_padding_check_X931(to,num,buf,i,num);
712                 break;
713         case RSA_NO_PADDING:
714                 r=RSA_padding_check_none(to,num,buf,i,num);
715                 break;
716         default:
717                 RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT,RSA_R_UNKNOWN_PADDING_TYPE);
718                 goto err;
719                 }
720         if (r < 0)
721                 RSAerr(RSA_F_RSA_EAY_PUBLIC_DECRYPT,RSA_R_PADDING_CHECK_FAILED);
```



```
723 err:
724         if (ctx != NULL)
725                 {
726                 BN_CTX_end(ctx);
727                 BN_CTX_free(ctx);
728                 }
729         if (buf != NULL)
730                 {
731                 OPENSSL_cleanse(buf,num);
732                 OPENSSL_free(buf);
733                 }
734         return(r);
735         }

737 static int RSA_eay_mod_exp(BIGNUM *r0, const BIGNUM *I, RSA *rsa, BN_CTX *ctx)
738         {
739         BIGNUM *r1,*m1,*vrfy;
740         BIGNUM local_dmp1,local_dmq1,local_c,local_r1;
741         BIGNUM *dmp1,*dmq1,*c,*pr1;
742         int ret=0;

744         BN_CTX_start(ctx);
745         r1 = BN_CTX_get(ctx);
746         m1 = BN_CTX_get(ctx);
747         vrfy = BN_CTX_get(ctx);

749         {
750                 BIGNUM local_p, local_q;
751                 BIGNUM *p = NULL, *q = NULL;

753                 /* Make sure BN_mod_inverse in Montgomery intialization uses the
754                  * BN_FLG_CONSTTIME flag (unless RSA_FLAG_NO_CONSTTIME is set)
755                  */
756                 if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
757                         {
758                         BN_init(&local_p);
759                         p = &local_p;
760                         BN_with_flags(p, rsa->p, BN_FLG_CONSTTIME);

762                         BN_init(&local_q);
763                         q = &local_q;
764                         BN_with_flags(q, rsa->q, BN_FLG_CONSTTIME);
765                         }
766                 else
767                         {
768                         p = rsa->p;
769                         q = rsa->q;
770                         }

772                 if (rsa->flags & RSA_FLAG_CACHE_PRIVATE)
773                         {
774                         if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_p, CRYPTO_
775                                 goto err;
776                         if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_q, CRYPTO_
777                                 goto err;
778                         }
779         }

781         if (rsa->flags & RSA_FLAG_CACHE_PUBLIC)
782                 if (!BN_MONT_CTX_set_locked(&rsa->_method_mod_n, CRYPTO_LOCK_RSA
783                         goto err;

785         /* compute I mod q */
786         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
787                 {
```

```
788                 c = &local_c;
789                 BN_with_flags(c, I, BN_FLG_CONSTTIME);
790                 if (!BN_mod(r1,c,rsa->q,ctx)) goto err;
791                 }
792         else
793                 {
794                 if (!BN_mod(r1,I,rsa->q,ctx)) goto err;
795                 }

797         /* compute r1^dmq1 mod q */
798         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
799                 {
800                 dmq1 = &local_dmq1;
801                 BN_with_flags(dmq1, rsa->dmq1, BN_FLG_CONSTTIME);
802                 }
803         else
804                 dmq1 = rsa->dmq1;
805         if (!rsa->meth->bn_mod_exp(m1,r1,dmq1,rsa->q,ctx,
806                 rsa->_method_mod_q)) goto err;

808         /* compute I mod p */
809         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
810                 {
811                 c = &local_c;
812                 BN_with_flags(c, I, BN_FLG_CONSTTIME);
813                 if (!BN_mod(r1,c,rsa->p,ctx)) goto err;
814                 }
815         else
816                 {
817                 if (!BN_mod(r1,I,rsa->p,ctx)) goto err;
818                 }

820         /* compute r1^dmp1 mod p */
821         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
822                 {
823                 dmp1 = &local_dmp1;
824                 BN_with_flags(dmp1, rsa->dmp1, BN_FLG_CONSTTIME);
825                 }
826         else
827                 dmp1 = rsa->dmp1;
828         if (!rsa->meth->bn_mod_exp(r0,r1,dmp1,rsa->p,ctx,
829                 rsa->_method_mod_p)) goto err;

831         if (!BN_sub(r0,r0,m1)) goto err;
832         /* This will help stop the size of r0 increasing, which does
833          * affect the multiply if it optimised for a power of 2 size */
834         if (BN_is_negative(r0))
835                 if (!BN_add(r0,r0,rsa->p)) goto err;

837         if (!BN_mul(r1,r0,rsa->iqmp,ctx)) goto err;

839         /* Turn BN_FLG_CONSTTIME flag on before division operation */
840         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
841                 {
842                 pr1 = &local_r1;
843                 BN_with_flags(pr1, r1, BN_FLG_CONSTTIME);
844                 }
845         else
846                 pr1 = r1;
847         if (!BN_mod(r0,pr1,rsa->p,ctx)) goto err;

849         /* If p < q it is occasionally possible for the correction of
850          * adding 'p' if r0 is negative above to leave the result still
851          * negative. This can break the private key operations: the following
852          * second correction should *always* correct this rare occurrence.
853          * This will *never* happen with OpenSSL generated keys because
```



```
854          * they ensure p > q [steve]
855          */
856         if (BN_is_negative(r0))
857                 if (!BN_add(r0,r0,rsa->p)) goto err;
858         if (!BN_mul(r1,r0,rsa->q,ctx)) goto err;
859         if (!BN_add(r0,r1,m1)) goto err;

861         if (rsa->e && rsa->n)
862                 {
863                 if (!rsa->meth->bn_mod_exp(vrfy,r0,rsa->e,rsa->n,ctx,rsa->_metho
864                 /* If 'I' was greater than (or equal to) rsa->n, the operation
865                  * will be equivalent to using 'I mod n'. However, the result of
866                  * the verify will *always* be less than 'n' so we don't check
867                  * for absolute equality, just congruency. */
868                 if (!BN_sub(vrfy, vrfy, I)) goto err;
869                 if (!BN_mod(vrfy, vrfy, rsa->n, ctx)) goto err;
870                 if (BN_is_negative(vrfy))
871                         if (!BN_add(vrfy, vrfy, rsa->n)) goto err;
872                 if (!BN_is_zero(vrfy))
873                         {
874                         /* 'I' and 'vrfy' aren't congruent mod n. Don't leak
875                          * miscalculated CRT output, just do a raw (slower)
876                          * mod_exp and return that instead. */

878                         BIGNUM local_d;
879                         BIGNUM *d = NULL;

881                         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
882                                 {
883                                 d = &local_d;
884                                 BN_with_flags(d, rsa->d, BN_FLG_CONSTTIME);
885                                 }
886                         else
887                                 d = rsa->d;
888                         if (!rsa->meth->bn_mod_exp(r0,I,d,rsa->n,ctx,
889                                                    rsa->_method_mod_n)) goto err
890                         }
891                 }
892         ret=1;
893 err:
894         BN_CTX_end(ctx);
895         return(ret);
896         }

898 static int RSA_eay_init(RSA *rsa)
899         {
900         rsa->flags|=RSA_FLAG_CACHE_PUBLIC|RSA_FLAG_CACHE_PRIVATE;
901         return(1);
902         }

904 static int RSA_eay_finish(RSA *rsa)
905         {
906         if (rsa->_method_mod_n != NULL)
907                 BN_MONT_CTX_free(rsa->_method_mod_n);
908         if (rsa->_method_mod_p != NULL)
909                 BN_MONT_CTX_free(rsa->_method_mod_p);
910         if (rsa->_method_mod_q != NULL)
911                 BN_MONT_CTX_free(rsa->_method_mod_q);
912         return(1);
913         }

915 #endif
916 #endif /* ! codereview */
```

```
**********************************************************
   11137 Wed Aug 13 19:53:16 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/rsa.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_RSA,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_RSA,0,reason)

static ERR_STRING_DATA RSA_str_functs[]=
        {
{ERR_FUNC(RSA_F_CHECK_PADDING_MD),      "CHECK_PADDING_MD"},
{ERR_FUNC(RSA_F_DO_RSA_PRINT),  "DO_RSA_PRINT"},
{ERR_FUNC(RSA_F_INT_RSA_VERIFY),        "INT_RSA_VERIFY"},
{ERR_FUNC(RSA_F_MEMORY_LOCK),   "MEMORY_LOCK"},
{ERR_FUNC(RSA_F_OLD_RSA_PRIV_DECODE),   "OLD_RSA_PRIV_DECODE"},
{ERR_FUNC(RSA_F_PKEY_RSA_CTRL), "PKEY_RSA_CTRL"},
{ERR_FUNC(RSA_F_PKEY_RSA_CTRL_STR),     "PKEY_RSA_CTRL_STR"},
{ERR_FUNC(RSA_F_PKEY_RSA_SIGN), "PKEY_RSA_SIGN"},
{ERR_FUNC(RSA_F_PKEY_RSA_VERIFY),       "PKEY_RSA_VERIFY"},
{ERR_FUNC(RSA_F_PKEY_RSA_VERIFYRECOVER),        "PKEY_RSA_VERIFYRECOVER"},
{ERR_FUNC(RSA_F_RSA_BUILTIN_KEYGEN),    "RSA_BUILTIN_KEYGEN"},
{ERR_FUNC(RSA_F_RSA_CHECK_KEY), "RSA_check_key"},
{ERR_FUNC(RSA_F_RSA_EAY_PRIVATE_DECRYPT),       "RSA_EAY_PRIVATE_DECRYPT"},
{ERR_FUNC(RSA_F_RSA_EAY_PRIVATE_ENCRYPT),       "RSA_EAY_PRIVATE_ENCRYPT"},
{ERR_FUNC(RSA_F_RSA_EAY_PUBLIC_DECRYPT),        "RSA_EAY_PUBLIC_DECRYPT"},
{ERR_FUNC(RSA_F_RSA_EAY_PUBLIC_ENCRYPT),        "RSA_EAY_PUBLIC_ENCRYPT"},
{ERR_FUNC(RSA_F_RSA_GENERATE_KEY),      "RSA_generate_key"},
{ERR_FUNC(RSA_F_RSA_GENERATE_KEY_EX),   "RSA_generate_key_ex"},
{ERR_FUNC(RSA_F_RSA_ITEM_VERIFY),       "RSA_ITEM_VERIFY"},
{ERR_FUNC(RSA_F_RSA_MEMORY_LOCK),       "RSA_memory_lock"},
{ERR_FUNC(RSA_F_RSA_NEW_METHOD),        "RSA_new_method"},
{ERR_FUNC(RSA_F_RSA_NULL),      "RSA_NULL"},
{ERR_FUNC(RSA_F_RSA_NULL_MOD_EXP),      "RSA_NULL_MOD_EXP"},
{ERR_FUNC(RSA_F_RSA_NULL_PRIVATE_DECRYPT),      "RSA_NULL_PRIVATE_DECRYPT"},
{ERR_FUNC(RSA_F_RSA_NULL_PRIVATE_ENCRYPT),      "RSA_NULL_PRIVATE_ENCRYPT"},
{ERR_FUNC(RSA_F_RSA_NULL_PUBLIC_DECRYPT),       "RSA_NULL_PUBLIC_DECRYPT"},
{ERR_FUNC(RSA_F_RSA_NULL_PUBLIC_ENCRYPT),       "RSA_NULL_PUBLIC_ENCRYPT"},
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_NONE),  "RSA_padding_add_none"},
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_PKCS1_OAEP),    "RSA_padding_add_PKCS1_OAEP"},
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_PKCS1_PSS),     "RSA_padding_add_PKCS1_PSS"},
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_PKCS1_PSS_MGF1),        "RSA_padding_add_PKCS1_P
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_PKCS1_TYPE_1),  "RSA_padding_add_PKCS1_type_1"},
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_PKCS1_TYPE_2),  "RSA_padding_add_PKCS1_type_2"},
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_SSLV23),        "RSA_padding_add_SSLv23"},
{ERR_FUNC(RSA_F_RSA_PADDING_ADD_X931),  "RSA_padding_add_X931"},
{ERR_FUNC(RSA_F_RSA_PADDING_CHECK_NONE),        "RSA_padding_check_none"},
{ERR_FUNC(RSA_F_RSA_PADDING_CHECK_PKCS1_OAEP),  "RSA_padding_check_PKCS1_OAEP"},
{ERR_FUNC(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_1),        "RSA_padding_check_PKCS1
{ERR_FUNC(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2),        "RSA_padding_check_PKCS1
{ERR_FUNC(RSA_F_RSA_PADDING_CHECK_SSLV23),      "RSA_padding_check_SSLv23"},
{ERR_FUNC(RSA_F_RSA_PADDING_CHECK_X931),        "RSA_padding_check_X931"},
{ERR_FUNC(RSA_F_RSA_PRINT),     "RSA_print"},
{ERR_FUNC(RSA_F_RSA_PRINT_FP),  "RSA_print_fp"},
{ERR_FUNC(RSA_F_RSA_PRIVATE_DECRYPT),   "RSA_private_decrypt"},
{ERR_FUNC(RSA_F_RSA_PRIVATE_ENCRYPT),   "RSA_private_encrypt"},
{ERR_FUNC(RSA_F_RSA_PRIV_DECODE),       "RSA_PRIV_DECODE"},
{ERR_FUNC(RSA_F_RSA_PRIV_ENCODE),       "RSA_PRIV_ENCODE"},
{ERR_FUNC(RSA_F_RSA_PUBLIC_DECRYPT),    "RSA_public_decrypt"},
{ERR_FUNC(RSA_F_RSA_PUBLIC_ENCRYPT),    "RSA_public_encrypt"},
{ERR_FUNC(RSA_F_RSA_PUB_DECODE),        "RSA_PUB_DECODE"},
{ERR_FUNC(RSA_F_RSA_SETUP_BLINDING),    "RSA_setup_blinding"},
{ERR_FUNC(RSA_F_RSA_SIGN),      "RSA_sign"},
{ERR_FUNC(RSA_F_RSA_SIGN_ASN1_OCTET_STRING),    "RSA_sign_ASN1_OCTET_STRING"},
{ERR_FUNC(RSA_F_RSA_VERIFY),    "RSA_verify"},
{ERR_FUNC(RSA_F_RSA_VERIFY_ASN1_OCTET_STRING),  "RSA_verify_ASN1_OCTET_STRING"},
```

```
{ERR_FUNC(RSA_F_RSA_VERIFY_PKCS1_PSS),  "RSA_verify_PKCS1_PSS"},
{ERR_FUNC(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1),     "RSA_verify_PKCS1_PSS_mgf1"},
{0,NULL}
        };

static ERR_STRING_DATA RSA_str_reasons[]=
        {
{ERR_REASON(RSA_R_ALGORITHM_MISMATCH)    ,"algorithm mismatch"},
{ERR_REASON(RSA_R_BAD_E_VALUE)           ,"bad e value"},
{ERR_REASON(RSA_R_BAD_FIXED_HEADER_DECRYPT),"bad fixed header decrypt"},
{ERR_REASON(RSA_R_BAD_PAD_BYTE_COUNT)    ,"bad pad byte count"},
{ERR_REASON(RSA_R_BAD_SIGNATURE)         ,"bad signature"},
{ERR_REASON(RSA_R_BLOCK_TYPE_IS_NOT_01)  ,"block type is not 01"},
{ERR_REASON(RSA_R_BLOCK_TYPE_IS_NOT_02)  ,"block type is not 02"},
{ERR_REASON(RSA_R_DATA_GREATER_THAN_MOD_LEN),"data greater than mod len"},
{ERR_REASON(RSA_R_DATA_TOO_LARGE)        ,"data too large"},
{ERR_REASON(RSA_R_DATA_TOO_LARGE_FOR_KEY_SIZE),"data too large for key size"},
{ERR_REASON(RSA_R_DATA_TOO_LARGE_FOR_MODULUS),"data too large for modulus"},
{ERR_REASON(RSA_R_DATA_TOO_SMALL)        ,"data too small"},
{ERR_REASON(RSA_R_DATA_TOO_SMALL_FOR_KEY_SIZE),"data too small for key size"},
{ERR_REASON(RSA_R_DIGEST_TOO_BIG_FOR_RSA_KEY),"digest too big for rsa key"},
{ERR_REASON(RSA_R_DMP1_NOT_CONGRUENT_TO_D),"dmp1 not congruent to d"},
{ERR_REASON(RSA_R_DMQ1_NOT_CONGRUENT_TO_D),"dmq1 not congruent to d"},
{ERR_REASON(RSA_R_D_E_NOT_CONGRUENT_TO_1),"d e not congruent to 1"},
{ERR_REASON(RSA_R_FIRST_OCTET_INVALID)   ,"first octet invalid"},
{ERR_REASON(RSA_R_ILLEGAL_OR_UNSUPPORTED_PADDING_MODE),"illegal or unsupported p
{ERR_REASON(RSA_R_INVALID_DIGEST_LENGTH) ,"invalid digest length"},
{ERR_REASON(RSA_R_INVALID_HEADER)        ,"invalid header"},
{ERR_REASON(RSA_R_INVALID_KEYBITS)       ,"invalid keybits"},
{ERR_REASON(RSA_R_INVALID_MESSAGE_LENGTH),"invalid message length"},
{ERR_REASON(RSA_R_INVALID_MGF1_MD)       ,"invalid mgf1 md"},
{ERR_REASON(RSA_R_INVALID_PADDING)       ,"invalid padding"},
{ERR_REASON(RSA_R_INVALID_PADDING_MODE)  ,"invalid padding mode"},
{ERR_REASON(RSA_R_INVALID_PSS_PARAMETERS),"invalid pss parameters"},
{ERR_REASON(RSA_R_INVALID_PSS_SALTLEN)   ,"invalid pss saltlen"},
{ERR_REASON(RSA_R_INVALID_SALT_LENGTH)   ,"invalid salt length"},
{ERR_REASON(RSA_R_INVALID_TRAILER)       ,"invalid trailer"},
{ERR_REASON(RSA_R_INVALID_X931_DIGEST)   ,"invalid x931 digest"},
{ERR_REASON(RSA_R_IQMP_NOT_INVERSE_OF_Q) ,"iqmp not inverse of q"},
{ERR_REASON(RSA_R_KEY_SIZE_TOO_SMALL)    ,"key size too small"},
{ERR_REASON(RSA_R_LAST_OCTET_INVALID)    ,"last octet invalid"},
{ERR_REASON(RSA_R_MODULUS_TOO_LARGE)     ,"modulus too large"},
{ERR_REASON(RSA_R_NON_FIPS_RSA_METHOD)   ,"non fips rsa method"},
{ERR_REASON(RSA_R_NO_PUBLIC_EXPONENT)    ,"no public exponent"},
{ERR_REASON(RSA_R_NULL_BEFORE_BLOCK_MISSING),"null before block missing"},
{ERR_REASON(RSA_R_N_DOES_NOT_EQUAL_P_Q)  ,"n does not equal p q"},
{ERR_REASON(RSA_R_OAEP_DECODING_ERROR)   ,"oaep decoding error"},
{ERR_REASON(RSA_R_OPERATION_NOT_ALLOWED_IN_FIPS_MODE),"operation not allowed in
{ERR_REASON(RSA_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE),"operation not suppo
{ERR_REASON(RSA_R_PADDING_CHECK_FAILED)  ,"padding check failed"},
{ERR_REASON(RSA_R_P_NOT_PRIME)           ,"p not prime"},
{ERR_REASON(RSA_R_Q_NOT_PRIME)           ,"q not prime"},
{ERR_REASON(RSA_R_RSA_OPERATIONS_NOT_SUPPORTED),"rsa operations not supported"},
{ERR_REASON(RSA_R_SLEN_CHECK_FAILED)     ,"salt length check failed"},
{ERR_REASON(RSA_R_SLEN_RECOVERY_FAILED)  ,"salt length recovery failed"},
{ERR_REASON(RSA_R_SSLV3_ROLLBACK_ATTACK) ,"sslv3 rollback attack"},
{ERR_REASON(RSA_R_THE_ASN1_OBJECT_IDENTIFIER_IS_NOT_KNOWN_FOR_THIS_MD),"the asn1
{ERR_REASON(RSA_R_UNKNOWN_ALGORITHM_TYPE),"unknown algorithm type"},
{ERR_REASON(RSA_R_UNKNOWN_MASK_DIGEST)   ,"unknown mask digest"},
{ERR_REASON(RSA_R_UNKNOWN_PADDING_TYPE)  ,"unknown padding type"},
{ERR_REASON(RSA_R_UNKNOWN_PSS_DIGEST)    ,"unknown pss digest"},
{ERR_REASON(RSA_R_UNSUPPORTED_MASK_ALGORITHM),"unsupported mask algorithm"},
{ERR_REASON(RSA_R_UNSUPPORTED_MASK_PARAMETER),"unsupported mask parameter"},
{ERR_REASON(RSA_R_UNSUPPORTED_SIGNATURE_TYPE),"unsupported signature type"},
{ERR_REASON(RSA_R_VALUE_MISSING)         ,"value missing"},
{ERR_REASON(RSA_R_WRONG_SIGNATURE_LENGTH),"wrong signature length"},
```



```
{0,NULL}
        };

#endif

void ERR_load_RSA_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(RSA_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,RSA_str_functs);
                ERR_load_strings(0,RSA_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    7837 Wed Aug 13 19:53:16 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_gen.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_gen.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */


/* NB: these functions have been "upgraded", the deprecated versions (which are
 * compatibility wrappers using these functions) are in rsa_depr.c.
```



```
 * - Geoff
 */

#include <stdio.h>
#include <time.h>
#include "cryptlib.h"
#include <openssl/bn.h>
#include <openssl/rsa.h>
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

static int rsa_builtin_keygen(RSA *rsa, int bits, BIGNUM *e_value, BN_GENCB *cb)

/* NB: this wrapper would normally be placed in rsa_lib.c and the static
 * implementation would probably be in rsa_eay.c. Nonetheless, is kept here so
 * that we don't introduce a new linker dependency. Eg. any application that
 * wasn't previously linking object code related to key-generation won't have to
 * now just because key-generation is part of RSA_METHOD. */
int RSA_generate_key_ex(RSA *rsa, int bits, BIGNUM *e_value, BN_GENCB *cb)
        {
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(rsa->meth->flags & RSA_FLAG_FIPS_METHOD)
                        && !(rsa->flags & RSA_FLAG_NON_FIPS_ALLOW))
                {
                RSAerr(RSA_F_RSA_GENERATE_KEY_EX, RSA_R_NON_FIPS_RSA_METHOD);
                return 0;
                }
#endif
        if(rsa->meth->rsa_keygen)
                return rsa->meth->rsa_keygen(rsa, bits, e_value, cb);
#ifdef OPENSSL_FIPS
        if (FIPS_mode())
                return FIPS_rsa_generate_key_ex(rsa, bits, e_value, cb);
#endif
        return rsa_builtin_keygen(rsa, bits, e_value, cb);
        }

static int rsa_builtin_keygen(RSA *rsa, int bits, BIGNUM *e_value, BN_GENCB *cb)
        {
        BIGNUM *r0=NULL,*r1=NULL,*r2=NULL,*r3=NULL,*tmp;
        BIGNUM local_r0,local_d,local_p;
        BIGNUM *pr0,*d,*p;
        int bitsp,bitsq,ok= -1,n=0;
        BN_CTX *ctx=NULL;

        ctx=BN_CTX_new();
        if (ctx == NULL) goto err;
        BN_CTX_start(ctx);
        r0 = BN_CTX_get(ctx);
        r1 = BN_CTX_get(ctx);
        r2 = BN_CTX_get(ctx);
        r3 = BN_CTX_get(ctx);
        if (r3 == NULL) goto err;

        bitsp=(bits+1)/2;
        bitsq=bits-bitsp;

        /* We need the RSA components non-NULL */
        if(!rsa->n && ((rsa->n=BN_new()) == NULL)) goto err;
        if(!rsa->d && ((rsa->d=BN_new()) == NULL)) goto err;
        if(!rsa->e && ((rsa->e=BN_new()) == NULL)) goto err;
        if(!rsa->p && ((rsa->p=BN_new()) == NULL)) goto err;
        if(!rsa->q && ((rsa->q=BN_new()) == NULL)) goto err;
        if(!rsa->dmp1 && ((rsa->dmp1=BN_new()) == NULL)) goto err;
        if(!rsa->dmq1 && ((rsa->dmq1=BN_new()) == NULL)) goto err;
```

```
128         if(!rsa->iqmp && ((rsa->iqmp=BN_new()) == NULL)) goto err;

130         BN_copy(rsa->e, e_value);

132         /* generate p and q */
133         for (;;)
134                 {
135                 if(!BN_generate_prime_ex(rsa->p, bitsp, 0, NULL, NULL, cb))
136                         goto err;
137                 if (!BN_sub(r2,rsa->p,BN_value_one())) goto err;
138                 if (!BN_gcd(r1,r2,rsa->e,ctx)) goto err;
139                 if (BN_is_one(r1)) break;
140                 if(!BN_GENCB_call(cb, 2, n++))
141                         goto err;
142                 }
143         if(!BN_GENCB_call(cb, 3, 0))
144                 goto err;
145         for (;;)
146                 {
147                 /* When generating ridiculously small keys, we can get stuck
148                  * continually regenerating the same prime values. Check for
149                  * this and bail if it happens 3 times. */
150                 unsigned int degenerate = 0;
151                 do
152                         {
153                         if(!BN_generate_prime_ex(rsa->q, bitsq, 0, NULL, NULL, c
154                                 goto err;
155                         } while((BN_cmp(rsa->p, rsa->q) == 0) && (++degenerate <
156                 if(degenerate == 3)
157                         {
158                         ok = 0; /* we set our own err */
159                         RSAerr(RSA_F_RSA_BUILTIN_KEYGEN,RSA_R_KEY_SIZE_TOO_SMALL
160                         goto err;
161                         }
162                 if (!BN_sub(r2,rsa->q,BN_value_one())) goto err;
163                 if (!BN_gcd(r1,r2,rsa->e,ctx)) goto err;
164                 if (BN_is_one(r1))
165                         break;
166                 if(!BN_GENCB_call(cb, 2, n++))
167                         goto err;
168                 }
169         if(!BN_GENCB_call(cb, 3, 1))
170                 goto err;
171         if (BN_cmp(rsa->p,rsa->q) < 0)
172                 {
173                 tmp=rsa->p;
174                 rsa->p=rsa->q;
175                 rsa->q=tmp;
176                 }

178         /* calculate n */
179         if (!BN_mul(rsa->n,rsa->p,rsa->q,ctx)) goto err;

181         /* calculate d */
182         if (!BN_sub(r1,rsa->p,BN_value_one())) goto err;        /* p-1 */
183         if (!BN_sub(r2,rsa->q,BN_value_one())) goto err;        /* q-1 */
184         if (!BN_mul(r0,r1,r2,ctx)) goto err;    /* (p-1)(q-1) */
185         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
186                 {
187                   pr0 = &local_r0;
188                   BN_with_flags(pr0, r0, BN_FLG_CONSTTIME);
189                 }
190         else
191           pr0 = r0;
192         if (!BN_mod_inverse(rsa->d,rsa->e,pr0,ctx)) goto err;   /* d */
```



```
194         /* set up d for correct BN_FLG_CONSTTIME flag */
195         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
196                 {
197                 d = &local_d;
198                 BN_with_flags(d, rsa->d, BN_FLG_CONSTTIME);
199                 }
200         else
201                 d = rsa->d;

203         /* calculate d mod (p-1) */
204         if (!BN_mod(rsa->dmp1,d,r1,ctx)) goto err;

206         /* calculate d mod (q-1) */
207         if (!BN_mod(rsa->dmq1,d,r2,ctx)) goto err;

209         /* calculate inverse of q mod p */
210         if (!(rsa->flags & RSA_FLAG_NO_CONSTTIME))
211                 {
212                 p = &local_p;
213                 BN_with_flags(p, rsa->p, BN_FLG_CONSTTIME);
214                 }
215         else
216                 p = rsa->p;
217         if (!BN_mod_inverse(rsa->iqmp,rsa->q,p,ctx)) goto err;

219         ok=1;
220 err:
221         if (ok == -1)
222                 {
223                 RSAerr(RSA_F_RSA_BUILTIN_KEYGEN,ERR_LIB_BN);
224                 ok=0;
225                 }
226         if (ctx != NULL)
227                 {
228                 BN_CTX_end(ctx);
229                 BN_CTX_free(ctx);
230                 }

232         return ok;
233         }
234 #endif /* ! codereview */
```

```
**********************************************************
    8762 Wed Aug 13 19:53:16 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/crypto.h>
#include "cryptlib.h"
```



```
#include <openssl/lhash.h>
#include <openssl/bn.h>
#include <openssl/rsa.h>
#include <openssl/rand.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

const char RSA_version[]="RSA" OPENSSL_VERSION_PTEXT;

static const RSA_METHOD *default_RSA_meth=NULL;

RSA *RSA_new(void)
        {
        RSA *r=RSA_new_method(NULL);

        return r;
        }

void RSA_set_default_method(const RSA_METHOD *meth)
        {
        default_RSA_meth = meth;
        }

const RSA_METHOD *RSA_get_default_method(void)
        {
        if (default_RSA_meth == NULL)
                {
#ifdef OPENSSL_FIPS
                if (FIPS_mode())
                        return FIPS_rsa_pkcs1_ssleay();
                else
                        return RSA_PKCS1_SSLeay();
#else
#ifdef RSA_NULL
                default_RSA_meth=RSA_null_method();
#else
                default_RSA_meth=RSA_PKCS1_SSLeay();
#endif
#endif
                }

        return default_RSA_meth;
        }

const RSA_METHOD *RSA_get_method(const RSA *rsa)
        {
        return rsa->meth;
        }

int RSA_set_method(RSA *rsa, const RSA_METHOD *meth)
        {
        /* NB: The caller is specifically setting a method, so it's not up to us
         * to deal with which ENGINE it comes from. */
        const RSA_METHOD *mtmp;
        mtmp = rsa->meth;
        if (mtmp->finish) mtmp->finish(rsa);
#ifndef OPENSSL_NO_ENGINE
        if (rsa->engine)
                {
                ENGINE_finish(rsa->engine);
                rsa->engine = NULL;
```

```
128                 }
129 #endif
130         rsa->meth = meth;
131         if (meth->init) meth->init(rsa);
132         return 1;
133         }
134
135 RSA *RSA_new_method(ENGINE *engine)
136         {
137         RSA *ret;
138
139         ret=(RSA *)OPENSSL_malloc(sizeof(RSA));
140         if (ret == NULL)
141                 {
142                 RSAerr(RSA_F_RSA_NEW_METHOD,ERR_R_MALLOC_FAILURE);
143                 return NULL;
144                 }
145
146         ret->meth = RSA_get_default_method();
147 #ifndef OPENSSL_NO_ENGINE
148         if (engine)
149                 {
150                 if (!ENGINE_init(engine))
151                         {
152                         RSAerr(RSA_F_RSA_NEW_METHOD, ERR_R_ENGINE_LIB);
153                         OPENSSL_free(ret);
154                         return NULL;
155                         }
156                 ret->engine = engine;
157                 }
158         else
159                 ret->engine = ENGINE_get_default_RSA();
160         if(ret->engine)
161                 {
162                 ret->meth = ENGINE_get_RSA(ret->engine);
163                 if(!ret->meth)
164                         {
165                         RSAerr(RSA_F_RSA_NEW_METHOD,
166                                 ERR_R_ENGINE_LIB);
167                         ENGINE_finish(ret->engine);
168                         OPENSSL_free(ret);
169                         return NULL;
170                         }
171                 }
172 #endif
173
174         ret->pad=0;
175         ret->version=0;
176         ret->n=NULL;
177         ret->e=NULL;
178         ret->d=NULL;
179         ret->p=NULL;
180         ret->q=NULL;
181         ret->dmp1=NULL;
182         ret->dmq1=NULL;
183         ret->iqmp=NULL;
184         ret->references=1;
185         ret->_method_mod_n=NULL;
186         ret->_method_mod_p=NULL;
187         ret->_method_mod_q=NULL;
188         ret->blinding=NULL;
189         ret->mt_blinding=NULL;
190         ret->bignum_data=NULL;
191         ret->flags=ret->meth->flags & ~RSA_FLAG_NON_FIPS_ALLOW;
192         if (!CRYPTO_new_ex_data(CRYPTO_EX_INDEX_RSA, ret, &ret->ex_data))
193                 {
```



```
194 #ifndef OPENSSL_NO_ENGINE
195         if (ret->engine)
196                 ENGINE_finish(ret->engine);
197 #endif
198                 OPENSSL_free(ret);
199                 return(NULL);
200                 }
201
202         if ((ret->meth->init != NULL) && !ret->meth->init(ret))
203                 {
204 #ifndef OPENSSL_NO_ENGINE
205                 if (ret->engine)
206                         ENGINE_finish(ret->engine);
207 #endif
208                 CRYPTO_free_ex_data(CRYPTO_EX_INDEX_RSA, ret, &ret->ex_data);
209                 OPENSSL_free(ret);
210                 ret=NULL;
211                 }
212         return(ret);
213         }
214
215 void RSA_free(RSA *r)
216         {
217         int i;
218
219         if (r == NULL) return;
220
221         i=CRYPTO_add(&r->references,-1,CRYPTO_LOCK_RSA);
222 #ifdef REF_PRINT
223         REF_PRINT("RSA",r);
224 #endif
225         if (i > 0) return;
226 #ifdef REF_CHECK
227         if (i < 0)
228                 {
229                 fprintf(stderr,"RSA_free, bad reference count\n");
230                 abort();
231                 }
232 #endif
233
234         if (r->meth->finish)
235                 r->meth->finish(r);
236 #ifndef OPENSSL_NO_ENGINE
237         if (r->engine)
238                 ENGINE_finish(r->engine);
239 #endif
240
241         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_RSA, r, &r->ex_data);
242
243         if (r->n != NULL) BN_clear_free(r->n);
244         if (r->e != NULL) BN_clear_free(r->e);
245         if (r->d != NULL) BN_clear_free(r->d);
246         if (r->p != NULL) BN_clear_free(r->p);
247         if (r->q != NULL) BN_clear_free(r->q);
248         if (r->dmp1 != NULL) BN_clear_free(r->dmp1);
249         if (r->dmq1 != NULL) BN_clear_free(r->dmq1);
250         if (r->iqmp != NULL) BN_clear_free(r->iqmp);
251         if (r->blinding != NULL) BN_BLINDING_free(r->blinding);
252         if (r->mt_blinding != NULL) BN_BLINDING_free(r->mt_blinding);
253         if (r->bignum_data != NULL) OPENSSL_free_locked(r->bignum_data);
254         OPENSSL_free(r);
255         }
256
257 int RSA_up_ref(RSA *r)
258         {
259         int i = CRYPTO_add(&r->references, 1, CRYPTO_LOCK_RSA);
```

```
#ifdef REF_PRINT
        REF_PRINT("RSA",r);
#endif
#ifdef REF_CHECK
        if (i < 2)
                {
                fprintf(stderr, "RSA_up_ref, bad reference count\n");
                abort();
                }
#endif
        return ((i > 1) ? 1 : 0);
        }

int RSA_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
             CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
        {
        return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_RSA, argl, argp,
                                new_func, dup_func, free_func);
        }

int RSA_set_ex_data(RSA *r, int idx, void *arg)
        {
        return(CRYPTO_set_ex_data(&r->ex_data,idx,arg));
        }

void *RSA_get_ex_data(const RSA *r, int idx)
        {
        return(CRYPTO_get_ex_data(&r->ex_data,idx));
        }

int RSA_memory_lock(RSA *r)
        {
        int i,j,k,off;
        char *p;
        BIGNUM *bn,**t[6],*b;
        BN_ULONG *ul;

        if (r->d == NULL) return(1);
        t[0]= &r->d;
        t[1]= &r->p;
        t[2]= &r->q;
        t[3]= &r->dmp1;
        t[4]= &r->dmq1;
        t[5]= &r->iqmp;
        k=sizeof(BIGNUM)*6;
        off=k/sizeof(BN_ULONG)+1;
        j=1;
        for (i=0; i<6; i++)
                j+= (*t[i])->top;
        if ((p=OPENSSL_malloc_locked((off+j)*sizeof(BN_ULONG))) == NULL)
                {
                RSAerr(RSA_F_RSA_MEMORY_LOCK,ERR_R_MALLOC_FAILURE);
                return(0);
                }
        bn=(BIGNUM *)p;
        ul=(BN_ULONG *)&(p[off]);
        for (i=0; i<6; i++)
                {
                b= *(t[i]);
                *(t[i])= &(bn[i]);
                memcpy((char *)&(bn[i]),(char *)b,sizeof(BIGNUM));
                bn[i].flags=BN_FLG_STATIC_DATA;
                bn[i].d=ul;
                memcpy((char *)ul,b->d,sizeof(BN_ULONG)*b->top);
                ul+=b->top;
                BN_clear_free(b);
```



```
                }

        /* I should fix this so it can still be done */
        r->flags&= ~(RSA_FLAG_CACHE_PRIVATE|RSA_FLAG_CACHE_PUBLIC);

        r->bignum_data=p;
        return(1);
        }
#endif /* ! codereview */
```

```
**********************************************************
    3966 Wed Aug 13 19:53:16 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_none.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_none.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/rand.h>

int RSA_padding_add_none(unsigned char *to, int tlen,
        const unsigned char *from, int flen)
        {
        if (flen > tlen)
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_NONE,RSA_R_DATA_TOO_LARGE_FOR_KEY_S
                return(0);
                }

        if (flen < tlen)
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_NONE,RSA_R_DATA_TOO_SMALL_FOR_KEY_S
                return(0);
                }

        memcpy(to,from,(unsigned int)flen);
        return(1);
        }

int RSA_padding_check_none(unsigned char *to, int tlen,
        const unsigned char *from, int flen, int num)
        {

        if (flen > tlen)
                {
                RSAerr(RSA_F_RSA_PADDING_CHECK_NONE,RSA_R_DATA_TOO_LARGE);
                return(-1);
                }

        memset(to,0,tlen-flen);
        memcpy(to+tlen-flen,from,flen);
        return(tlen);
        }
#endif /* ! codereview */
```

```
**********************************************************
    5332 Wed Aug 13 19:53:17 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_null.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* rsa_null.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/rand.h>

/* This is a dummy RSA implementation that just returns errors when called.
 * It is designed to allow some RSA functions to work while stopping those
 * covered by the RSA patent. That is RSA, encryption, decryption, signing
 * and verify is not allowed but RSA key generation, key checking and other
 * operations (like storing RSA keys) are permitted.
 */

static int RSA_null_public_encrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
static int RSA_null_private_encrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
static int RSA_null_public_decrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
static int RSA_null_private_decrypt(int flen, const unsigned char *from,
                unsigned char *to, RSA *rsa,int padding);
#if 0 /* not currently used */
static int RSA_null_mod_exp(const BIGNUM *r0, const BIGNUM *i, RSA *rsa);
#endif
static int RSA_null_init(RSA *rsa);
static int RSA_null_finish(RSA *rsa);
static RSA_METHOD rsa_null_meth={
        "Null RSA",
        RSA_null_public_encrypt,
        RSA_null_public_decrypt,
        RSA_null_private_encrypt,
        RSA_null_private_decrypt,
        NULL,
        NULL,
        RSA_null_init,
        RSA_null_finish,
        0,
        NULL,
        NULL,
        NULL,
        NULL
        };

const RSA_METHOD *RSA_null_method(void)
        {
        return(&rsa_null_meth);
        }

static int RSA_null_public_encrypt(int flen, const unsigned char *from,
             unsigned char *to, RSA *rsa, int padding)
        {
        RSAerr(RSA_F_RSA_NULL_PUBLIC_ENCRYPT, RSA_R_RSA_OPERATIONS_NOT_SUPPORTED
        return -1;
        }

static int RSA_null_private_encrypt(int flen, const unsigned char *from,
             unsigned char *to, RSA *rsa, int padding)
        {
        RSAerr(RSA_F_RSA_NULL_PRIVATE_ENCRYPT, RSA_R_RSA_OPERATIONS_NOT_SUPPORTE
        return -1;
        }

static int RSA_null_private_decrypt(int flen, const unsigned char *from,
             unsigned char *to, RSA *rsa, int padding)
        {
        RSAerr(RSA_F_RSA_NULL_PRIVATE_DECRYPT, RSA_R_RSA_OPERATIONS_NOT_SUPPORTE
        return -1;
        }
```

```
128 static int RSA_null_public_decrypt(int flen, const unsigned char *from,
129              unsigned char *to, RSA *rsa, int padding)
130         {
131         RSAerr(RSA_F_RSA_NULL_PUBLIC_DECRYPT, RSA_R_RSA_OPERATIONS_NOT_SUPPORTED
132         return -1;
133         }

135 #if 0 /* not currently used */
136 static int RSA_null_mod_exp(BIGNUM *r0, BIGNUM *I, RSA *rsa)
137         {
138         ...err(RSA_F_RSA_NULL_MOD_EXP, RSA_R_RSA_OPERATIONS_NOT_SUPPORTED);
139         return -1;
140         }
141 #endif

143 static int RSA_null_init(RSA *rsa)
144         {
145         return(1);
146         }

148 static int RSA_null_finish(RSA *rsa)
149         {
150         return(1);
151         }
152 #endif /* ! codereview */
```

```
**********************************************************
    6285 Wed Aug 13 19:53:17 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_oaep.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_oaep.c */
/* Written by Ulf Moeller. This software is distributed on an "AS IS"
   basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. */

/* EME-OAEP as defined in RFC 2437 (PKCS #1 v2.0) */

/* See Victor Shoup, "OAEP reconsidered," Nov. 2000,
 * <URL: http://www.shoup.net/papers/oaep.ps.Z>
 * for problems with the security proof for the
 * original OAEP scheme, which EME-OAEP is based on.
 *
 * A new proof can be found in E. Fujisaki, T. Okamoto,
 * D. Pointcheval, J. Stern, "RSA-OEAP is Still Alive!",
 * Dec. 2000, <URL: http://eprint.iacr.org/2000/061/>.
 * The new proof has stronger requirements for the
 * underlying permutation: "partial-one-wayness" instead
 * of one-wayness.  For the RSA function, this is
 * an equivalent notion.
 */


#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA1)
#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
#include <openssl/rsa.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
#include <openssl/sha.h>

static int MGF1(unsigned char *mask, long len,
        const unsigned char *seed, long seedlen);

int RSA_padding_add_PKCS1_OAEP(unsigned char *to, int tlen,
        const unsigned char *from, int flen,
        const unsigned char *param, int plen)
        {
        int i, emlen = tlen - 1;
        unsigned char *db, *seed;
        unsigned char *dbmask, seedmask[SHA_DIGEST_LENGTH];

        if (flen > emlen - 2 * SHA_DIGEST_LENGTH - 1)
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_OAEP,
                   RSA_R_DATA_TOO_LARGE_FOR_KEY_SIZE);
                return 0;
                }

        if (emlen < 2 * SHA_DIGEST_LENGTH + 1)
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_OAEP, RSA_R_KEY_SIZE_TOO_SMAL
                return 0;
                }

        to[0] = 0;
        seed = to + 1;
        db = to + SHA_DIGEST_LENGTH + 1;

        if (!EVP_Digest((void *)param, plen, db, NULL, EVP_sha1(), NULL))
                return 0;
        memset(db + SHA_DIGEST_LENGTH, 0,
```



```
                emlen - flen - 2 * SHA_DIGEST_LENGTH - 1);
        db[emlen - flen - SHA_DIGEST_LENGTH - 1] = 0x01;
        memcpy(db + emlen - flen - SHA_DIGEST_LENGTH, from, (unsigned int) flen)
        if (RAND_bytes(seed, SHA_DIGEST_LENGTH) <= 0)
                return 0;
#ifdef PKCS_TESTVECT
        memcpy(seed,
           "\xaa\xfd\x12\xf6\x59\xca\xe6\x34\x89\xb4\x79\xe5\x07\x6d\xde\xc2\xf0
           20);
#endif

        dbmask = OPENSSL_malloc(emlen - SHA_DIGEST_LENGTH);
        if (dbmask == NULL)
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_OAEP, ERR_R_MALLOC_FAILURE);
                return 0;
                }

        if (MGF1(dbmask, emlen - SHA_DIGEST_LENGTH, seed, SHA_DIGEST_LENGTH) < 0
                return 0;
        for (i = 0; i < emlen - SHA_DIGEST_LENGTH; i++)
                db[i] ^= dbmask[i];

        if (MGF1(seedmask, SHA_DIGEST_LENGTH, db, emlen - SHA_DIGEST_LENGTH) < 0
                return 0;
        for (i = 0; i < SHA_DIGEST_LENGTH; i++)
                seed[i] ^= seedmask[i];

        OPENSSL_free(dbmask);
        return 1;
        }

int RSA_padding_check_PKCS1_OAEP(unsigned char *to, int tlen,
        const unsigned char *from, int flen, int num,
        const unsigned char *param, int plen)
        {
        int i, dblen, mlen = -1;
        const unsigned char *maskeddb;
        int lzero;
        unsigned char *db = NULL, seed[SHA_DIGEST_LENGTH], phash[SHA_DIGEST_LENG
        unsigned char *padded_from;
        int bad = 0;

        if (--num < 2 * SHA_DIGEST_LENGTH + 1)
                /* 'num' is the length of the modulus, i.e. does not depend on t
                 * particular ciphertext. */
                goto decoding_err;

        lzero = num - flen;
        if (lzero < 0)
                {
                /* signalling this error immediately after detection might allow
                 * for side-channel attacks (e.g. timing if 'plen' is huge
                 * -- cf. James H. Manger, "A Chosen Ciphertext Attack on RSA Op
                 * Asymmetric Encryption Padding (OAEP) [...]", CRYPTO 2001),
                 * so we use a 'bad' flag */
                bad = 1;
                lzero = 0;
                flen = num; /* don't overflow the memcpy to padded_from */
                }

        dblen = num - SHA_DIGEST_LENGTH;
        db = OPENSSL_malloc(dblen + num);
        if (db == NULL)
                {
                RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_OAEP, ERR_R_MALLOC_FAILURE)
```

```
128                 return -1;
129                 }

131         /* Always do this zero-padding copy (even when lzero == 0)
132          * to avoid leaking timing info about the value of lzero. */
133         padded_from = db + dblen;
134         memset(padded_from, 0, lzero);
135         memcpy(padded_from + lzero, from, flen);

137         maskeddb = padded_from + SHA_DIGEST_LENGTH;

139         if (MGF1(seed, SHA_DIGEST_LENGTH, maskeddb, dblen))
140                 return -1;
141         for (i = 0; i < SHA_DIGEST_LENGTH; i++)
142                 seed[i] ^= padded_from[i];

144         if (MGF1(db, dblen, seed, SHA_DIGEST_LENGTH))
145                 return -1;
146         for (i = 0; i < dblen; i++)
147                 db[i] ^= maskeddb[i];

149         if (!EVP_Digest((void *)param, plen, phash, NULL, EVP_sha1(), NULL))
150                 return -1;

152         if (CRYPTO_memcmp(db, phash, SHA_DIGEST_LENGTH) != 0 || bad)
153                 goto decoding_err;
154         else
155                 {
156                 for (i = SHA_DIGEST_LENGTH; i < dblen; i++)
157                         if (db[i] != 0x00)
158                                 break;
159                 if (i == dblen || db[i] != 0x01)
160                         goto decoding_err;
161                 else
162                         {
163                         /* everything looks OK */

165                         mlen = dblen - ++i;
166                         if (tlen < mlen)
167                                 {
168                                 RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_OAEP, RSA_R
169                                 mlen = -1;
170                                 }
171                         else
172                                 memcpy(to, db + i, mlen);
173                         }
174                 }
175         OPENSSL_free(db);
176         return mlen;

178 decoding_err:
179         /* to avoid chosen ciphertext attacks, the error message should not reve
180          * which kind of decoding error happened */
181         RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_OAEP, RSA_R_OAEP_DECODING_ERROR);
182         if (db != NULL) OPENSSL_free(db);
183         return -1;
184         }

186 int PKCS1_MGF1(unsigned char *mask, long len,
187         const unsigned char *seed, long seedlen, const EVP_MD *dgst)
188         {
189         long i, outlen = 0;
190         unsigned char cnt[4];
191         EVP_MD_CTX c;
192         unsigned char md[EVP_MAX_MD_SIZE];
193         int mdlen;
```



```
194         int rv = -1;

196         EVP_MD_CTX_init(&c);
197         mdlen = EVP_MD_size(dgst);
198         if (mdlen < 0)
199                 goto err;
200         for (i = 0; outlen < len; i++)
201                 {
202                 cnt[0] = (unsigned char)((i >> 24) & 255);
203                 cnt[1] = (unsigned char)((i >> 16) & 255);
204                 cnt[2] = (unsigned char)((i >> 8)) & 255;
205                 cnt[3] = (unsigned char)(i & 255);
206                 if (!EVP_DigestInit_ex(&c,dgst, NULL)
207                         || !EVP_DigestUpdate(&c, seed, seedlen)
208                         || !EVP_DigestUpdate(&c, cnt, 4))
209                         goto err;
210                 if (outlen + mdlen <= len)
211                         {
212                         if (!EVP_DigestFinal_ex(&c, mask + outlen, NULL))
213                                 goto err;
214                         outlen += mdlen;
215                         }
216                 else
217                         {
218                         if (!EVP_DigestFinal_ex(&c, md, NULL))
219                                 goto err;
220                         memcpy(mask + outlen, md, len - outlen);
221                         outlen = len;
222                         }
223                 }
224         rv = 0;
225         err:
226         EVP_MD_CTX_cleanup(&c);
227         return rv;
228         }

230 static int MGF1(unsigned char *mask, long len, const unsigned char *seed,
231                  long seedlen)
232         {
233         return PKCS1_MGF1(mask, len, seed, seedlen, EVP_sha1());
234         }
235 #endif
236 #endif /* ! codereview */
```

```
**********************************************************
    6346 Wed Aug 13 19:53:17 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_pk1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_pk1.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/rand.h>

int RSA_padding_add_PKCS1_type_1(unsigned char *to, int tlen,
             const unsigned char *from, int flen)
        {
        int j;
        unsigned char *p;

        if (flen > (tlen-RSA_PKCS1_PADDING_SIZE))
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_TYPE_1,RSA_R_DATA_TOO_LARGE_F
                return(0);
                }

        p=(unsigned char *)to;

        *(p++)=0;
        *(p++)=1; /* Private Key BT (Block Type) */

        /* pad out with 0xff data */
        j=tlen-3-flen;
        memset(p,0xff,j);
        p+=j;
        *(p++)='\0';
        memcpy(p,from,(unsigned int)flen);
        return(1);
        }

int RSA_padding_check_PKCS1_type_1(unsigned char *to, int tlen,
             const unsigned char *from, int flen, int num)
        {
        int i,j;
        const unsigned char *p;

        p=from;
        if ((num != (flen+1)) || (*(p++) != 01))
                {
                RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_1,RSA_R_BLOCK_TYPE_IS_
                return(-1);
                }

        /* scan over padding data */
        j=flen-1; /* one for type. */
        for (i=0; i<j; i++)
                {
                if (*p != 0xff) /* should decrypt to 0xff */
                        {
                        if (*p == 0)
                                { p++; break; }
                        else    {
                                RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_1,RSA_
                                return(-1);
                                }
                        }
                p++;
                }

        if (i == j)
                {
                RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_1,RSA_R_NULL_BEFORE_BL
                return(-1);
                }

        if (i < 8)
                {
```

```
128                 RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_1,RSA_R_BAD_PAD_BYTE_C
129                 return(-1);
130                 }
131         i++; /* Skip over the '\0' */
132         j-=i;
133         if (j > tlen)
134                 {
135                 RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_1,RSA_R_DATA_TOO_LARGE
136                 return(-1);
137                 }
138         memcpy(to,p,(unsigned int)j);
139
140         return(j);
141         }
142
143 int RSA_padding_add_PKCS1_type_2(unsigned char *to, int tlen,
144              const unsigned char *from, int flen)
145         {
146         int i,j;
147         unsigned char *p;
148
149         if (flen > (tlen-11))
150                 {
151                 RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_TYPE_2,RSA_R_DATA_TOO_LARGE_F
152                 return(0);
153                 }
154
155         p=(unsigned char *)to;
156
157         *(p++)=0;
158         *(p++)=2; /* Public Key BT (Block Type) */
159
160         /* pad out with non-zero random data */
161         j=tlen-3-flen;
162
163         if (RAND_bytes(p,j) <= 0)
164                 return(0);
165         for (i=0; i<j; i++)
166                 {
167                 if (*p == '\0')
168                         do      {
169                                 if (RAND_bytes(p,1) <= 0)
170                                         return(0);
171                                 } while (*p == '\0');
172                 p++;
173                 }
174
175         *(p++)='\0';
176
177         memcpy(p,from,(unsigned int)flen);
178         return(1);
179         }
180
181 int RSA_padding_check_PKCS1_type_2(unsigned char *to, int tlen,
182              const unsigned char *from, int flen, int num)
183         {
184         int i,j;
185         const unsigned char *p;
186
187         p=from;
188         if ((num != (flen+1)) || (*(p++) != 02))
189                 {
190                 RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2,RSA_R_BLOCK_TYPE_IS_
191                 return(-1);
192                 }
193 #ifdef PKCS1_CHECK
```



```
194         return(num-11);
195 #endif
196
197         /* scan over padding data */
198         j=flen-1; /* one for type. */
199         for (i=0; i<j; i++)
200                 if (*(p++) == 0) break;
201
202         if (i == j)
203                 {
204                 RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2,RSA_R_NULL_BEFORE_BL
205                 return(-1);
206                 }
207
208         if (i < 8)
209                 {
210                 RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2,RSA_R_BAD_PAD_BYTE_C
211                 return(-1);
212                 }
213         i++; /* Skip over the '\0' */
214         j-=i;
215         if (j > tlen)
216                 {
217                 RSAerr(RSA_F_RSA_PADDING_CHECK_PKCS1_TYPE_2,RSA_R_DATA_TOO_LARGE
218                 return(-1);
219                 }
220         memcpy(to,p,(unsigned int)j);
221
222         return(j);
223         }
224 #endif /* ! codereview */
```

```
**********************************************************
   16306 Wed Aug 13 19:53:17 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_pmeth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_pmeth.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1t.h>
```



```
#include <openssl/x509.h>
#include <openssl/rsa.h>
#include <openssl/bn.h>
#include <openssl/evp.h>
#ifndef OPENSSL_NO_CMS
#include <openssl/cms.h>
#endif
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif
#include "evp_locl.h"
#include "rsa_locl.h"

/* RSA pkey context structure */

typedef struct
        {
        /* Key gen parameters */
        int nbits;
        BIGNUM *pub_exp;
        /* Keygen callback info */
        int gentmp[2];
        /* RSA padding mode */
        int pad_mode;
        /* message digest */
        const EVP_MD *md;
        /* message digest for MGF1 */
        const EVP_MD *mgf1md;
        /* PSS/OAEP salt length */
        int saltlen;
        /* Temp buffer */
        unsigned char *tbuf;
        } RSA_PKEY_CTX;

static int pkey_rsa_init(EVP_PKEY_CTX *ctx)
        {
        RSA_PKEY_CTX *rctx;
        rctx = OPENSSL_malloc(sizeof(RSA_PKEY_CTX));
        if (!rctx)
                return 0;
        rctx->nbits = 1024;
        rctx->pub_exp = NULL;
        rctx->pad_mode = RSA_PKCS1_PADDING;
        rctx->md = NULL;
        rctx->mgf1md = NULL;
        rctx->tbuf = NULL;

        rctx->saltlen = -2;

        ctx->data = rctx;
        ctx->keygen_info = rctx->gentmp;
        ctx->keygen_info_count = 2;

        return 1;
        }

static int pkey_rsa_copy(EVP_PKEY_CTX *dst, EVP_PKEY_CTX *src)
        {
        RSA_PKEY_CTX *dctx, *sctx;
        if (!pkey_rsa_init(dst))
                return 0;
        sctx = src->data;
        dctx = dst->data;
        dctx->nbits = sctx->nbits;
        if (sctx->pub_exp)
                {
```

```
128                 dctx->pub_exp = BN_dup(sctx->pub_exp);
129                 if (!dctx->pub_exp)
130                         return 0;
131                 }
132         dctx->pad_mode = sctx->pad_mode;
133         dctx->md = sctx->md;
134         return 1;
135         }

137 static int setup_tbuf(RSA_PKEY_CTX *ctx, EVP_PKEY_CTX *pk)
138         {
139         if (ctx->tbuf)
140                 return 1;
141         ctx->tbuf = OPENSSL_malloc(EVP_PKEY_size(pk->pkey));
142         if (!ctx->tbuf)
143                 return 0;
144         return 1;
145         }

147 static void pkey_rsa_cleanup(EVP_PKEY_CTX *ctx)
148         {
149         RSA_PKEY_CTX *rctx = ctx->data;
150         if (rctx)
151                 {
152                 if (rctx->pub_exp)
153                         BN_free(rctx->pub_exp);
154                 if (rctx->tbuf)
155                         OPENSSL_free(rctx->tbuf);
156                 OPENSSL_free(rctx);
157                 }
158         }
159 #ifdef OPENSSL_FIPS
160 /* FIP checker. Return value indicates status of context parameters:
161  * 1  : redirect to FIPS.
162  * 0  : don't redirect to FIPS.
163  * -1 : illegal operation in FIPS mode.
164  */

166 static int pkey_fips_check_ctx(EVP_PKEY_CTX *ctx)
167         {
168         RSA_PKEY_CTX *rctx = ctx->data;
169         RSA *rsa = ctx->pkey->pkey.rsa;
170         int rv = -1;
171         if (!FIPS_mode())
172                 return 0;
173         if (rsa->flags & RSA_FLAG_NON_FIPS_ALLOW)
174                 rv = 0;
175         if (!(rsa->meth->flags & RSA_FLAG_FIPS_METHOD) && rv)
176                 return -1;
177         if (rctx->md && !(rctx->md->flags & EVP_MD_FLAG_FIPS))
178                 return rv;
179         if (rctx->mgf1md && !(rctx->mgf1md->flags & EVP_MD_FLAG_FIPS))
180                 return rv;
181         return 1;
182         }
183 #endif

185 static int pkey_rsa_sign(EVP_PKEY_CTX *ctx, unsigned char *sig, size_t *siglen,
186                                         const unsigned char *tbs, size_t tbslen)
187         {
188         int ret;
189         RSA_PKEY_CTX *rctx = ctx->data;
190         RSA *rsa = ctx->pkey->pkey.rsa;

192 #ifdef OPENSSL_FIPS
193         ret = pkey_fips_check_ctx(ctx);
```



```
194         if (ret < 0)
195                 {
196                 RSAerr(RSA_F_PKEY_RSA_SIGN, RSA_R_OPERATION_NOT_ALLOWED_IN_FIPS_
197                 return -1;
198                 }
199 #endif

201         if (rctx->md)
202                 {
203                 if (tbslen != (size_t)EVP_MD_size(rctx->md))
204                         {
205                         RSAerr(RSA_F_PKEY_RSA_SIGN,
206                                         RSA_R_INVALID_DIGEST_LENGTH);
207                         return -1;
208                         }
209 #ifdef OPENSSL_FIPS
210                 if (ret > 0)
211                         {
212                         unsigned int slen;
213                         ret = FIPS_rsa_sign_digest(rsa, tbs, tbslen, rctx->md,
214                                                    rctx->pad_mode,
215                                                    rctx->saltlen,
216                                                    rctx->mgf1md,
217                                                    sig, &slen);
218                         if (ret > 0)
219                                 *siglen = slen;
220                         else
221                                 *siglen = 0;
222                         return ret;
223                         }
224 #endif

226                 if (EVP_MD_type(rctx->md) == NID_mdc2)
227                         {
228                         unsigned int sltmp;
229                         if (rctx->pad_mode != RSA_PKCS1_PADDING)
230                                 return -1;
231                         ret = RSA_sign_ASN1_OCTET_STRING(NID_mdc2,
232                                                 tbs, tbslen, sig, &sltmp, rsa);

234                         if (ret <= 0)
235                                 return ret;
236                         ret = sltmp;
237                         }
238                 else if (rctx->pad_mode == RSA_X931_PADDING)
239                         {
240                         if (!setup_tbuf(rctx, ctx))
241                                 return -1;
242                         memcpy(rctx->tbuf, tbs, tbslen);
243                         rctx->tbuf[tbslen] =
244                                 RSA_X931_hash_id(EVP_MD_type(rctx->md));
245                         ret = RSA_private_encrypt(tbslen + 1, rctx->tbuf,
246                                                 sig, rsa, RSA_X931_PADDING);
247                         }
248                 else if (rctx->pad_mode == RSA_PKCS1_PADDING)
249                         {
250                         unsigned int sltmp;
251                         ret = RSA_sign(EVP_MD_type(rctx->md),
252                                                 tbs, tbslen, sig, &sltmp, rsa);
253                         if (ret <= 0)
254                                 return ret;
255                         ret = sltmp;
256                         }
257                 else if (rctx->pad_mode == RSA_PKCS1_PSS_PADDING)
258                         {
259                         if (!setup_tbuf(rctx, ctx))
```

```
260                             return -1;
261                     if (!RSA_padding_add_PKCS1_PSS_mgf1(rsa,
262                                             rctx->tbuf, tbs,
263                                             rctx->md, rctx->mgf1md,
264                                             rctx->saltlen))
265                             return -1;
266                     ret = RSA_private_encrypt(RSA_size(rsa), rctx->tbuf,
267                                             sig, rsa, RSA_NO_PADDING);
268                     }
269             else
270                     return -1;
271             }
272         else
273                 ret = RSA_private_encrypt(tbslen, tbs, sig, ctx->pkey->pkey.rsa,
274                                                   rctx->pad_mode);
275         if (ret < 0)
276                 return ret;
277         *siglen = ret;
278         return 1;
279         }


282 static int pkey_rsa_verifyrecover(EVP_PKEY_CTX *ctx,
283                                   unsigned char *rout, size_t *routlen,
284                                   const unsigned char *sig, size_t siglen)
285         {
286         int ret;
287         RSA_PKEY_CTX *rctx = ctx->data;

289         if (rctx->md)
290                 {
291                 if (rctx->pad_mode == RSA_X931_PADDING)
292                         {
293                         if (!setup_tbuf(rctx, ctx))
294                                 return -1;
295                         ret = RSA_public_decrypt(siglen, sig,
296                                                 rctx->tbuf, ctx->pkey->pkey.rsa,
297                                                 RSA_X931_PADDING);
298                         if (ret < 1)
299                                 return 0;
300                         ret--;
301                         if (rctx->tbuf[ret] !=
302                                 RSA_X931_hash_id(EVP_MD_type(rctx->md)))
303                                 {
304                                 RSAerr(RSA_F_PKEY_RSA_VERIFYRECOVER,
305                                                 RSA_R_ALGORITHM_MISMATCH);
306                                 return 0;
307                                 }
308                         if (ret != EVP_MD_size(rctx->md))
309                                 {
310                                 RSAerr(RSA_F_PKEY_RSA_VERIFYRECOVER,
311                                         RSA_R_INVALID_DIGEST_LENGTH);
312                                 return 0;
313                                 }
314                         if (rout)
315                                 memcpy(rout, rctx->tbuf, ret);
316                         }
317                 else if (rctx->pad_mode == RSA_PKCS1_PADDING)
318                         {
319                         size_t sltmp;
320                         ret = int_rsa_verify(EVP_MD_type(rctx->md),
321                                                 NULL, 0, rout, &sltmp,
322                                         sig, siglen, ctx->pkey->pkey.rsa);
323                         if (ret <= 0)
324                                 return 0;
325                         ret = sltmp;
```



```
326                         }
327                 else
328                         return -1;
329                 }
330         else
331                 ret = RSA_public_decrypt(siglen, sig, rout, ctx->pkey->pkey.rsa,
332                                                         rctx->pad_mode);
333         if (ret < 0)
334                 return ret;
335         *routlen = ret;
336         return 1;
337         }

339 static int pkey_rsa_verify(EVP_PKEY_CTX *ctx,
340                                         const unsigned char *sig, size_t siglen,
341                                         const unsigned char *tbs, size_t tbslen)
342         {
343         RSA_PKEY_CTX *rctx = ctx->data;
344         RSA *rsa = ctx->pkey->pkey.rsa;
345         size_t rslen;
346 #ifdef OPENSSL_FIPS
347         int rv;
348         rv = pkey_fips_check_ctx(ctx);
349         if (rv < 0)
350                 {
351                 RSAerr(RSA_F_PKEY_RSA_VERIFY, RSA_R_OPERATION_NOT_ALLOWED_IN_FIP
352                 return -1;
353                 }
354 #endif
355         if (rctx->md)
356                 {
357 #ifdef OPENSSL_FIPS
358                 if (rv > 0)
359                         {
360                         return FIPS_rsa_verify_digest(rsa,
361                                                         tbs, tbslen,
362                                                         rctx->md,
363                                                         rctx->pad_mode,
364                                                         rctx->saltlen,
365                                                         rctx->mgf1md,
366                                                         sig, siglen);

368                         }
369 #endif
370                 if (rctx->pad_mode == RSA_PKCS1_PADDING)
371                         return RSA_verify(EVP_MD_type(rctx->md), tbs, tbslen,
372                                         sig, siglen, rsa);
373                 if (rctx->pad_mode == RSA_X931_PADDING)
374                         {
375                         if (pkey_rsa_verifyrecover(ctx, NULL, &rslen,
376                                         sig, siglen) <= 0)
377                                 return 0;
378                         }
379                 else if (rctx->pad_mode == RSA_PKCS1_PSS_PADDING)
380                         {
381                         int ret;
382                         if (!setup_tbuf(rctx, ctx))
383                                 return -1;
384                         ret = RSA_public_decrypt(siglen, sig, rctx->tbuf,
385                                                         rsa, RSA_NO_PADDING);
386                         if (ret <= 0)
387                                 return 0;
388                         ret = RSA_verify_PKCS1_PSS_mgf1(rsa, tbs,
389                                                 rctx->md, rctx->mgf1md,
390                                                 rctx->tbuf, rctx->saltlen);
391                         if (ret <= 0)
```

```
392                                 return 0;
393                         return 1;
394                         }
395                 else
396                         return -1;
397                 }
398         else
399                 {
400                 if (!setup_tbuf(rctx, ctx))
401                         return -1;
402                 rslen = RSA_public_decrypt(siglen, sig, rctx->tbuf,
403                                                 rsa, rctx->pad_mode);
404                 if (rslen == 0)
405                         return 0;
406                 }

408         if ((rslen != tbslen) || memcmp(tbs, rctx->tbuf, rslen))
409                 return 0;

411         return 1;

413         }


416 static int pkey_rsa_encrypt(EVP_PKEY_CTX *ctx,
417                                         unsigned char *out, size_t *outlen,
418                                         const unsigned char *in, size_t inlen)
419         {
420         int ret;
421         RSA_PKEY_CTX *rctx = ctx->data;
422         ret = RSA_public_encrypt(inlen, in, out, ctx->pkey->pkey.rsa,
423                                                         rctx->pad_mode);
424         if (ret < 0)
425                 return ret;
426         *outlen = ret;
427         return 1;
428         }

430 static int pkey_rsa_decrypt(EVP_PKEY_CTX *ctx,
431                                         unsigned char *out, size_t *outlen,
432                                         const unsigned char *in, size_t inlen)
433         {
434         int ret;
435         RSA_PKEY_CTX *rctx = ctx->data;
436         ret = RSA_private_decrypt(inlen, in, out, ctx->pkey->pkey.rsa,
437                                                         rctx->pad_mode);
438         if (ret < 0)
439                 return ret;
440         *outlen = ret;
441         return 1;
442         }

444 static int check_padding_md(const EVP_MD *md, int padding)
445         {
446         if (!md)
447                 return 1;

449         if (padding == RSA_NO_PADDING)
450                 {
451                 RSAerr(RSA_F_CHECK_PADDING_MD, RSA_R_INVALID_PADDING_MODE);
452                 return 0;
453                 }

455         if (padding == RSA_X931_PADDING)
456                 {
457                 if (RSA_X931_hash_id(EVP_MD_type(md)) == -1)
```



```
458                         {
459                         RSAerr(RSA_F_CHECK_PADDING_MD,
460                                                 RSA_R_INVALID_X931_DIGEST);
461                         return 0;
462                         }
463                 return 1;
464                 }

466         return 1;
467         }


470 static int pkey_rsa_ctrl(EVP_PKEY_CTX *ctx, int type, int p1, void *p2)
471         {
472         RSA_PKEY_CTX *rctx = ctx->data;
473         switch (type)
474                 {
475                 case EVP_PKEY_CTRL_RSA_PADDING:
476                 if ((p1 >= RSA_PKCS1_PADDING) && (p1 <= RSA_PKCS1_PSS_PADDING))
477                         {
478                         if (!check_padding_md(rctx->md, p1))
479                                 return 0;
480                         if (p1 == RSA_PKCS1_PSS_PADDING)
481                                 {
482                                 if (!(ctx->operation &
483                                     (EVP_PKEY_OP_SIGN | EVP_PKEY_OP_VERIFY)))
484                                         goto bad_pad;
485                                 if (!rctx->md)
486                                         rctx->md = EVP_sha1();
487                                 }
488                         if (p1 == RSA_PKCS1_OAEP_PADDING)
489                                 {
490                                 if (!(ctx->operation & EVP_PKEY_OP_TYPE_CRYPT))
491                                         goto bad_pad;
492                                 if (!rctx->md)
493                                         rctx->md = EVP_sha1();
494                                 }
495                         rctx->pad_mode = p1;
496                         return 1;
497                         }
498                 bad_pad:
499                 RSAerr(RSA_F_PKEY_RSA_CTRL,
500                                 RSA_R_ILLEGAL_OR_UNSUPPORTED_PADDING_MODE);
501                 return -2;

503                 case EVP_PKEY_CTRL_GET_RSA_PADDING:
504                 *(int *)p2 = rctx->pad_mode;
505                 return 1;

507                 case EVP_PKEY_CTRL_RSA_PSS_SALTLEN:
508                 case EVP_PKEY_CTRL_GET_RSA_PSS_SALTLEN:
509                 if (rctx->pad_mode != RSA_PKCS1_PSS_PADDING)
510                         {
511                         RSAerr(RSA_F_PKEY_RSA_CTRL, RSA_R_INVALID_PSS_SALTLEN);
512                         return -2;
513                         }
514                 if (type == EVP_PKEY_CTRL_GET_RSA_PSS_SALTLEN)
515                         *(int *)p2 = rctx->saltlen;
516                 else
517                         {
518                         if (p1 < -2)
519                                 return -2;
520                         rctx->saltlen = p1;
521                         }
522                 return 1;
```

```
524                 case EVP_PKEY_CTRL_RSA_KEYGEN_BITS:
525                 if (p1 < 256)
526                         {
527                         RSAerr(RSA_F_PKEY_RSA_CTRL, RSA_R_INVALID_KEYBITS);
528                         return -2;
529                         }
530                 rctx->nbits = p1;
531                 return 1;
532
533                 case EVP_PKEY_CTRL_RSA_KEYGEN_PUBEXP:
534                 if (!p2)
535                         return -2;
536                 rctx->pub_exp = p2;
537                 return 1;
538
539                 case EVP_PKEY_CTRL_MD:
540                 if (!check_padding_md(p2, rctx->pad_mode))
541                         return 0;
542                 rctx->md = p2;
543                 return 1;
544
545                 case EVP_PKEY_CTRL_RSA_MGF1_MD:
546                 case EVP_PKEY_CTRL_GET_RSA_MGF1_MD:
547                 if (rctx->pad_mode != RSA_PKCS1_PSS_PADDING)
548                         {
549                         RSAerr(RSA_F_PKEY_RSA_CTRL, RSA_R_INVALID_MGF1_MD);
550                         return -2;
551                         }
552                 if (type == EVP_PKEY_CTRL_GET_RSA_MGF1_MD)
553                         {
554                         if (rctx->mgf1md)
555                                 *(const EVP_MD **)p2 = rctx->mgf1md;
556                         else
557                                 *(const EVP_MD **)p2 = rctx->md;
558                         }
559                 else
560                         rctx->mgf1md = p2;
561                 return 1;
562
563                 case EVP_PKEY_CTRL_DIGESTINIT:
564                 case EVP_PKEY_CTRL_PKCS7_ENCRYPT:
565                 case EVP_PKEY_CTRL_PKCS7_DECRYPT:
566                 case EVP_PKEY_CTRL_PKCS7_SIGN:
567                 return 1;
568 #ifndef OPENSSL_NO_CMS
569                 case EVP_PKEY_CTRL_CMS_DECRYPT:
570                 {
571                 X509_ALGOR *alg = NULL;
572                 ASN1_OBJECT *encalg = NULL;
573                 if (p2)
574                         CMS_RecipientInfo_ktri_get0_algs(p2, NULL, NULL, &alg);
575                 if (alg)
576                         X509_ALGOR_get0(&encalg, NULL, NULL, alg);
577                 if (encalg && OBJ_obj2nid(encalg) == NID_rsaesOaep)
578                         rctx->pad_mode = RSA_PKCS1_OAEP_PADDING;
579                 }
580                 case EVP_PKEY_CTRL_CMS_ENCRYPT:
581                 case EVP_PKEY_CTRL_CMS_SIGN:
582                 return 1;
583 #endif
584                 case EVP_PKEY_CTRL_PEER_KEY:
585                         RSAerr(RSA_F_PKEY_RSA_CTRL,
586                         RSA_R_OPERATION_NOT_SUPPORTED_FOR_THIS_KEYTYPE);
587                         return -2;
588
589                 default:
```



```
590                 return -2;
591
592                 }
593         }
594
595 static int pkey_rsa_ctrl_str(EVP_PKEY_CTX *ctx,
596                         const char *type, const char *value)
597         {
598         if (!value)
599                 {
600                 RSAerr(RSA_F_PKEY_RSA_CTRL_STR, RSA_R_VALUE_MISSING);
601                 return 0;
602                 }
603         if (!strcmp(type, "rsa_padding_mode"))
604                 {
605                 int pm;
606                 if (!strcmp(value, "pkcs1"))
607                         pm = RSA_PKCS1_PADDING;
608                 else if (!strcmp(value, "sslv23"))
609                         pm = RSA_SSLV23_PADDING;
610                 else if (!strcmp(value, "none"))
611                         pm = RSA_NO_PADDING;
612                 else if (!strcmp(value, "oeap"))
613                         pm = RSA_PKCS1_OAEP_PADDING;
614                 else if (!strcmp(value, "oaep"))
615                         pm = RSA_PKCS1_OAEP_PADDING;
616                 else if (!strcmp(value, "x931"))
617                         pm = RSA_X931_PADDING;
618                 else if (!strcmp(value, "pss"))
619                         pm = RSA_PKCS1_PSS_PADDING;
620                 else
621                         {
622                         RSAerr(RSA_F_PKEY_RSA_CTRL_STR,
623                                                 RSA_R_UNKNOWN_PADDING_TYPE);
624                         return -2;
625                         }
626                 return EVP_PKEY_CTX_set_rsa_padding(ctx, pm);
627                 }
628
629         if (!strcmp(type, "rsa_pss_saltlen"))
630                 {
631                 int saltlen;
632                 saltlen = atoi(value);
633                 return EVP_PKEY_CTX_set_rsa_pss_saltlen(ctx, saltlen);
634                 }
635
636         if (!strcmp(type, "rsa_keygen_bits"))
637                 {
638                 int nbits;
639                 nbits = atoi(value);
640                 return EVP_PKEY_CTX_set_rsa_keygen_bits(ctx, nbits);
641                 }
642
643         if (!strcmp(type, "rsa_keygen_pubexp"))
644                 {
645                 int ret;
646                 BIGNUM *pubexp = NULL;
647                 if (!BN_asc2bn(&pubexp, value))
648                         return 0;
649                 ret = EVP_PKEY_CTX_set_rsa_keygen_pubexp(ctx, pubexp);
650                 if (ret <= 0)
651                         BN_free(pubexp);
652                 return ret;
653                 }
654
655         return -2;
```

```
656         }

658 static int pkey_rsa_keygen(EVP_PKEY_CTX *ctx, EVP_PKEY *pkey)
659         {
660         RSA *rsa = NULL;
661         RSA_PKEY_CTX *rctx = ctx->data;
662         BN_GENCB *pcb, cb;
663         int ret;
664         if (!rctx->pub_exp)
665                 {
666                 rctx->pub_exp = BN_new();
667                 if (!rctx->pub_exp || !BN_set_word(rctx->pub_exp, RSA_F4))
668                         return 0;
669                 }
670         rsa = RSA_new();
671         if (!rsa)
672                 return 0;
673         if (ctx->pkey_gencb)
674                 {
675                 pcb = &cb;
676                 evp_pkey_set_cb_translate(pcb, ctx);
677                 }
678         else
679                 pcb = NULL;
680         ret = RSA_generate_key_ex(rsa, rctx->nbits, rctx->pub_exp, pcb);
681         if (ret > 0)
682                 EVP_PKEY_assign_RSA(pkey, rsa);
683         else
684                 RSA_free(rsa);
685         return ret;
686         }

688 const EVP_PKEY_METHOD rsa_pkey_meth =
689         {
690         EVP_PKEY_RSA,
691         EVP_PKEY_FLAG_AUTOARGLEN,
692         pkey_rsa_init,
693         pkey_rsa_copy,
694         pkey_rsa_cleanup,

696         0,0,

698         0,
699         pkey_rsa_keygen,

701         0,
702         pkey_rsa_sign,

704         0,
705         pkey_rsa_verify,

707         0,
708         pkey_rsa_verifyrecover,


711         0,0,0,0,

713         0,
714         pkey_rsa_encrypt,

716         0,
717         pkey_rsa_decrypt,

719         0,0,

721         pkey_rsa_ctrl,
```



```
722         pkey_rsa_ctrl_str


725         };
726 #endif /* ! codereview */
```

```
**********************************************************
    3375 Wed Aug 13 19:53:17 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_prn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_prn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2006.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/rsa.h>
```



```
#include <openssl/evp.h>

#ifndef OPENSSL_NO_FP_API
int RSA_print_fp(FILE *fp, const RSA *x, int off)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file())) == NULL)
                {
                RSAerr(RSA_F_RSA_PRINT_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=RSA_print(b,x,off);
        BIO_free(b);
        return(ret);
        }
#endif

int RSA_print(BIO *bp, const RSA *x, int off)
        {
        EVP_PKEY *pk;
        int ret;
        pk = EVP_PKEY_new();
        if (!pk || !EVP_PKEY_set1_RSA(pk, (RSA *)x))
                return 0;
        ret = EVP_PKEY_print_private(bp, pk, off, NULL);
        EVP_PKEY_free(pk);
        return ret;
        }
#endif /* ! codereview */
```

```
**********************************************************
    8062 Wed Aug 13 19:53:17 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_pss.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* rsa_pss.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2005.
 */
/* ====================================================================
 * Copyright (c) 2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/evp.h>
#include <openssl/rand.h>
#include <openssl/sha.h>

static const unsigned char zeroes[] = {0,0,0,0,0,0,0,0};

#if defined(_MSC_VER) && defined(_ARM_)
#pragma optimize("g", off)
#endif

int RSA_verify_PKCS1_PSS(RSA *rsa, const unsigned char *mHash,
                         const EVP_MD *Hash, const unsigned char *EM, int sLen)
        {
        return RSA_verify_PKCS1_PSS_mgf1(rsa, mHash, Hash, NULL, EM, sLen);
        }

int RSA_verify_PKCS1_PSS_mgf1(RSA *rsa, const unsigned char *mHash,
                         const EVP_MD *Hash, const EVP_MD *mgf1Hash,
                         const unsigned char *EM, int sLen)
        {
        int i;
        int ret = 0;
        int hLen, maskedDBLen, MSBits, emLen;
        const unsigned char *H;
        unsigned char *DB = NULL;
        EVP_MD_CTX ctx;
        unsigned char H_[EVP_MAX_MD_SIZE];
        EVP_MD_CTX_init(&ctx);

        if (mgf1Hash == NULL)
                mgf1Hash = Hash;

        hLen = EVP_MD_size(Hash);
        if (hLen < 0)
                goto err;
        /*
         * Negative sLen has special meanings:
         *      -1      sLen == hLen
         *      -2      salt length is autorecovered from signature
         *      -N      reserved
         */
        if      (sLen == -1)    sLen = hLen;
        else if (sLen == -2)    sLen = -2;
        else if (sLen < -2)
                {
                RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, RSA_R_SLEN_CHECK_FAILED)
                goto err;
                }

        MSBits = (BN_num_bits(rsa->n) - 1) & 0x7;
        emLen = RSA_size(rsa);
        if (EM[0] & (0xFF << MSBits))
                {
                RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, RSA_R_FIRST_OCTET_INVALI
                goto err;
                }
        if (MSBits == 0)
                {
                EM++;
                emLen--;
                }
        if (emLen < (hLen + sLen + 2)) /* sLen can be small negative */
                {
                RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, RSA_R_DATA_TOO_LARGE);
                goto err;
```

```
128                 }
129         if (EM[emLen - 1] != 0xbc)
130                 {
131                 RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, RSA_R_LAST_OCTET_INVALID
132                 goto err;
133                 }
134         maskedDBLen = emLen - hLen - 1;
135         H = EM + maskedDBLen;
136         DB = OPENSSL_malloc(maskedDBLen);
137         if (!DB)
138                 {
139                 RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, ERR_R_MALLOC_FAILURE);
140                 goto err;
141                 }
142         if (PKCS1_MGF1(DB, maskedDBLen, H, hLen, mgf1Hash) < 0)
143                 goto err;
144         for (i = 0; i < maskedDBLen; i++)
145                 DB[i] ^= EM[i];
146         if (MSBits)
147                 DB[0] &= 0xFF >> (8 - MSBits);
148         for (i = 0; DB[i] == 0 && i < (maskedDBLen-1); i++) ;
149         if (DB[i++] != 0x1)
150                 {
151                 RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, RSA_R_SLEN_RECOVERY_FAIL
152                 goto err;
153                 }
154         if (sLen >= 0 && (maskedDBLen - i) != sLen)
155                 {
156                 RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, RSA_R_SLEN_CHECK_FAILED)
157                 goto err;
158                 }
159         if (!EVP_DigestInit_ex(&ctx, Hash, NULL)
160                 || !EVP_DigestUpdate(&ctx, zeroes, sizeof zeroes)
161                 || !EVP_DigestUpdate(&ctx, mHash, hLen))
162                 goto err;
163         if (maskedDBLen - i)
164                 {
165                 if (!EVP_DigestUpdate(&ctx, DB + i, maskedDBLen - i))
166                         goto err;
167                 }
168         if (!EVP_DigestFinal_ex(&ctx, H_, NULL))
169                 goto err;
170         if (memcmp(H_, H, hLen))
171                 {
172                 RSAerr(RSA_F_RSA_VERIFY_PKCS1_PSS_MGF1, RSA_R_BAD_SIGNATURE);
173                 ret = 0;
174                 }
175         else
176                 ret = 1;

178         err:
179         if (DB)
180                 OPENSSL_free(DB);
181         EVP_MD_CTX_cleanup(&ctx);

183         return ret;

185         }

187 int RSA_padding_add_PKCS1_PSS(RSA *rsa, unsigned char *EM,
188                         const unsigned char *mHash,
189                         const EVP_MD *Hash, int sLen)
190         {
191         return RSA_padding_add_PKCS1_PSS_mgf1(rsa, EM, mHash, Hash, NULL, sLen);
192         }
```



```
194 int RSA_padding_add_PKCS1_PSS_mgf1(RSA *rsa, unsigned char *EM,
195                         const unsigned char *mHash,
196                         const EVP_MD *Hash, const EVP_MD *mgf1Hash, int sLen)
197         {
198         int i;
199         int ret = 0;
200         int hLen, maskedDBLen, MSBits, emLen;
201         unsigned char *H, *salt = NULL, *p;
202         EVP_MD_CTX ctx;

204         if (mgf1Hash == NULL)
205                 mgf1Hash = Hash;

207         hLen = EVP_MD_size(Hash);
208         if (hLen < 0)
209                 goto err;
210         /*
211          * Negative sLen has special meanings:
212          *      -1      sLen == hLen
213          *      -2      salt length is maximized
214          *      -N      reserved
215          */
216         if      (sLen == -1)    sLen = hLen;
217         else if (sLen == -2)    sLen = -2;
218         else if (sLen < -2)
219                 {
220                 RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_PSS_MGF1, RSA_R_SLEN_CHECK_FA
221                 goto err;
222                 }

224         MSBits = (BN_num_bits(rsa->n) - 1) & 0x7;
225         emLen = RSA_size(rsa);
226         if (MSBits == 0)
227                 {
228                 *EM++ = 0;
229                 emLen--;
230                 }
231         if (sLen == -2)
232                 {
233                 sLen = emLen - hLen - 2;
234                 }
235         else if (emLen < (hLen + sLen + 2))
236                 {
237                 RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_PSS_MGF1,RSA_R_DATA_TOO_LARGE
238                 goto err;
239                 }
240         if (sLen > 0)
241                 {
242                 salt = OPENSSL_malloc(sLen);
243                 if (!salt)
244                         {
245                         RSAerr(RSA_F_RSA_PADDING_ADD_PKCS1_PSS_MGF1,ERR_R_MALLOC
246                         goto err;
247                         }
248                 if (RAND_bytes(salt, sLen) <= 0)
249                         goto err;
250                 }
251         maskedDBLen = emLen - hLen - 1;
252         H = EM + maskedDBLen;
253         EVP_MD_CTX_init(&ctx);
254         if (!EVP_DigestInit_ex(&ctx, Hash, NULL)
255                 || !EVP_DigestUpdate(&ctx, zeroes, sizeof zeroes)
256                 || !EVP_DigestUpdate(&ctx, mHash, hLen))
257                 goto err;
258         if (sLen && !EVP_DigestUpdate(&ctx, salt, sLen))
259                 goto err;
```

```
260         if (!EVP_DigestFinal_ex(&ctx, H, NULL))
261                 goto err;
262         EVP_MD_CTX_cleanup(&ctx);

264         /* Generate dbMask in place then perform XOR on it */
265         if (PKCS1_MGF1(EM, maskedDBLen, H, hLen, mgf1Hash))
266                 goto err;

268         p = EM;

270         /* Initial PS XORs with all zeroes which is a NOP so just update
271          * pointer. Note from a test above this value is guaranteed to
272          * be non-negative.
273          */
274         p += emLen - sLen - hLen - 2;
275         *p++ ^= 0x1;
276         if (sLen > 0)
277                 {
278                 for (i = 0; i < sLen; i++)
279                         *p++ ^= salt[i];
280                 }
281         if (MSBits)
282                 EM[0] &= 0xFF >> (8 - MSBits);

284         /* H is already in place so just set final 0xbc */

286         EM[emLen - 1] = 0xbc;

288         ret = 1;

290         err:
291         if (salt)
292                 OPENSSL_free(salt);

294         return ret;

296         }

298 #if defined(_MSC_VER)
299 #pragma optimize("",on)
300 #endif
301 #endif /* ! codereview */
```

```
**********************************************************
    5227 Wed Aug 13 19:53:18 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_saos.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_saos.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/objects.h>
#include <openssl/x509.h>

int RSA_sign_ASN1_OCTET_STRING(int type,
        const unsigned char *m, unsigned int m_len,
        unsigned char *sigret, unsigned int *siglen, RSA *rsa)
        {
        ASN1_OCTET_STRING sig;
        int i,j,ret=1;
        unsigned char *p,*s;

        sig.type=V_ASN1_OCTET_STRING;
        sig.length=m_len;
        sig.data=(unsigned char *)m;

        i=i2d_ASN1_OCTET_STRING(&sig,NULL);
        j=RSA_size(rsa);
        if (i > (j-RSA_PKCS1_PADDING_SIZE))
                {
                RSAerr(RSA_F_RSA_SIGN_ASN1_OCTET_STRING,RSA_R_DIGEST_TOO_BIG_FOR
                return(0);
                }
        s=(unsigned char *)OPENSSL_malloc((unsigned int)j+1);
        if (s == NULL)
                {
                RSAerr(RSA_F_RSA_SIGN_ASN1_OCTET_STRING,ERR_R_MALLOC_FAILURE);
                return(0);
                }
        p=s;
        i2d_ASN1_OCTET_STRING(&sig,&p);
        i=RSA_private_encrypt(i,s,sigret,rsa,RSA_PKCS1_PADDING);
        if (i <= 0)
                ret=0;
        else
                *siglen=i;

        OPENSSL_cleanse(s,(unsigned int)j+1);
        OPENSSL_free(s);
        return(ret);
        }

int RSA_verify_ASN1_OCTET_STRING(int dtype,
        const unsigned char *m,
        unsigned int m_len, unsigned char *sigbuf, unsigned int siglen,
        RSA *rsa)
        {
        int i,ret=0;
        unsigned char *s;
        const unsigned char *p;
        ASN1_OCTET_STRING *sig=NULL;

        if (siglen != (unsigned int)RSA_size(rsa))
                {
                RSAerr(RSA_F_RSA_VERIFY_ASN1_OCTET_STRING,RSA_R_WRONG_SIGNATURE_
                return(0);
                }

        s=(unsigned char *)OPENSSL_malloc((unsigned int)siglen);
        if (s == NULL)
                {
                RSAerr(RSA_F_RSA_VERIFY_ASN1_OCTET_STRING,ERR_R_MALLOC_FAILURE);
                goto err;
                }
        i=RSA_public_decrypt((int)siglen,sigbuf,s,rsa,RSA_PKCS1_PADDING);
```

```
128         if (i <= 0) goto err;

130         p=s;
131         sig=d2i_ASN1_OCTET_STRING(NULL,&p,(long)i);
132         if (sig == NULL) goto err;

134         if (    ((unsigned int)sig->length != m_len) ||
135                 (memcmp(m,sig->data,m_len) != 0))
136                 {
137                 RSAerr(RSA_F_RSA_VERIFY_ASN1_OCTET_STRING,RSA_R_BAD_SIGNATURE);
138                 }
139         else
140                 ret=1;
141 err:
142         if (sig != NULL) M_ASN1_OCTET_STRING_free(sig);
143         if (s != NULL)
144                 {
145                 OPENSSL_cleanse(s,(unsigned int)siglen);
146                 OPENSSL_free(s);
147                 }
148         return(ret);
149         }
150 #endif /* ! codereview */
```

```
**********************************************************
    9361 Wed Aug 13 19:53:18 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_sign.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_sign.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include "rsa_locl.h"

/* Size of an SSL signature: MD5+SHA1 */
#define SSL_SIG_LENGTH  36

int RSA_sign(int type, const unsigned char *m, unsigned int m_len,
             unsigned char *sigret, unsigned int *siglen, RSA *rsa)
        {
        X509_SIG sig;
        ASN1_TYPE parameter;
        int i,j,ret=1;
        unsigned char *p, *tmps = NULL;
        const unsigned char *s = NULL;
        X509_ALGOR algor;
        ASN1_OCTET_STRING digest;
#ifdef OPENSSL_FIPS
        if (FIPS_mode() && !(rsa->meth->flags & RSA_FLAG_FIPS_METHOD)
                        && !(rsa->flags & RSA_FLAG_NON_FIPS_ALLOW))
                {
                RSAerr(RSA_F_RSA_SIGN, RSA_R_NON_FIPS_RSA_METHOD);
                return 0;
                }
#endif
        if((rsa->flags & RSA_FLAG_SIGN_VER) && rsa->meth->rsa_sign)
                {
                return rsa->meth->rsa_sign(type, m, m_len,
                        sigret, siglen, rsa);
                }
        /* Special case: SSL signature, just check the length */
        if(type == NID_md5_sha1) {
                if(m_len != SSL_SIG_LENGTH) {
                        RSAerr(RSA_F_RSA_SIGN,RSA_R_INVALID_MESSAGE_LENGTH);
                        return(0);
                }
                i = SSL_SIG_LENGTH;
                s = m;
        } else {
                sig.algor= &algor;
                sig.algor->algorithm=OBJ_nid2obj(type);
                if (sig.algor->algorithm == NULL)
                        {
                        RSAerr(RSA_F_RSA_SIGN,RSA_R_UNKNOWN_ALGORITHM_TYPE);
                        return(0);
                        }
                if (sig.algor->algorithm->length == 0)
                        {
                        RSAerr(RSA_F_RSA_SIGN,RSA_R_THE_ASN1_OBJECT_IDENTIFIER_I
                        return(0);
                        }
                parameter.type=V_ASN1_NULL;
                parameter.value.ptr=NULL;
                sig.algor->parameter= &parameter;

                sig.digest= &digest;
                sig.digest->data=(unsigned char *)m; /* TMP UGLY CAST */
                sig.digest->length=m_len;

                i=i2d_X509_SIG(&sig,NULL);
        }
        j=RSA_size(rsa);
        if (i > (j-RSA_PKCS1_PADDING_SIZE))
                {
                RSAerr(RSA_F_RSA_SIGN,RSA_R_DIGEST_TOO_BIG_FOR_RSA_KEY);
```

```
128                 return(0);
129                 }
130         if(type != NID_md5_sha1) {
131                 tmps=(unsigned char *)OPENSSL_malloc((unsigned int)j+1);
132                 if (tmps == NULL)
133                         {
134                         RSAerr(RSA_F_RSA_SIGN,ERR_R_MALLOC_FAILURE);
135                         return(0);
136                         }
137                 p=tmps;
138                 i2d_X509_SIG(&sig,&p);
139                 s=tmps;
140         }
141         i=RSA_private_encrypt(i,s,sigret,rsa,RSA_PKCS1_PADDING);
142         if (i <= 0)
143                 ret=0;
144         else
145                 *siglen=i;

147         if(type != NID_md5_sha1) {
148                 OPENSSL_cleanse(tmps,(unsigned int)j+1);
149                 OPENSSL_free(tmps);
150         }
151         return(ret);
152         }

154 int int_rsa_verify(int dtype, const unsigned char *m,
155                            unsigned int m_len,
156                            unsigned char *rm, size_t *prm_len,
157                            const unsigned char *sigbuf, size_t siglen,
158                            RSA *rsa)
159         {
160         int i,ret=0,sigtype;
161         unsigned char *s;
162         X509_SIG *sig=NULL;

164 #ifdef OPENSSL_FIPS
165         if (FIPS_mode() && !(rsa->meth->flags & RSA_FLAG_FIPS_METHOD)
166                         && !(rsa->flags & RSA_FLAG_NON_FIPS_ALLOW))
167                 {
168                 RSAerr(RSA_F_INT_RSA_VERIFY, RSA_R_NON_FIPS_RSA_METHOD);
169                 return 0;
170                 }
171 #endif

173         if (siglen != (unsigned int)RSA_size(rsa))
174                 {
175                 RSAerr(RSA_F_INT_RSA_VERIFY,RSA_R_WRONG_SIGNATURE_LENGTH);
176                 return(0);
177                 }

179         if((dtype == NID_md5_sha1) && rm)
180                 {
181                 i = RSA_public_decrypt((int)siglen,
182                                        sigbuf,rm,rsa,RSA_PKCS1_PADDING);
183                 if (i <= 0)
184                         return 0;
185                 *prm_len = i;
186                 return 1;
187                 }

189         s=(unsigned char *)OPENSSL_malloc((unsigned int)siglen);
190         if (s == NULL)
191                 {
192                 RSAerr(RSA_F_INT_RSA_VERIFY,ERR_R_MALLOC_FAILURE);
193                 goto err;
```



```
194                 }
195         if((dtype == NID_md5_sha1) && (m_len != SSL_SIG_LENGTH) ) {
196                         RSAerr(RSA_F_INT_RSA_VERIFY,RSA_R_INVALID_MESSAGE_LENGTH
197                         goto err;
198         }
199         i=RSA_public_decrypt((int)siglen,sigbuf,s,rsa,RSA_PKCS1_PADDING);

201         if (i <= 0) goto err;
202         /* Oddball MDC2 case: signature can be OCTET STRING.
203          * check for correct tag and length octets.
204          */
205         if (dtype == NID_mdc2 && i == 18 && s[0] == 0x04 && s[1] == 0x10)
206                 {
207                 if (rm)
208                         {
209                         memcpy(rm, s + 2, 16);
210                         *prm_len = 16;
211                         ret = 1;
212                         }
213                 else if(memcmp(m, s + 2, 16))
214                         RSAerr(RSA_F_INT_RSA_VERIFY,RSA_R_BAD_SIGNATURE);
215                 else
216                         ret = 1;
217                 }

219         /* Special case: SSL signature */
220         if(dtype == NID_md5_sha1) {
221                 if((i != SSL_SIG_LENGTH) || memcmp(s, m, SSL_SIG_LENGTH))
222                                 RSAerr(RSA_F_INT_RSA_VERIFY,RSA_R_BAD_SIGNATURE)
223                 else ret = 1;
224         } else {
225                 const unsigned char *p=s;
226                 sig=d2i_X509_SIG(NULL,&p,(long)i);

228                 if (sig == NULL) goto err;

230                 /* Excess data can be used to create forgeries */
231                 if(p != s+i)
232                         {
233                         RSAerr(RSA_F_INT_RSA_VERIFY,RSA_R_BAD_SIGNATURE);
234                         goto err;
235                         }

237                 /* Parameters to the signature algorithm can also be used to
238                    create forgeries */
239                 if(sig->algor->parameter
240                    && ASN1_TYPE_get(sig->algor->parameter) != V_ASN1_NULL)
241                         {
242                         RSAerr(RSA_F_INT_RSA_VERIFY,RSA_R_BAD_SIGNATURE);
243                         goto err;
244                         }

246                 sigtype=OBJ_obj2nid(sig->algor->algorithm);


249         #ifdef RSA_DEBUG
250                 /* put a backward compatibility flag in EAY */
251                 fprintf(stderr,"in(%s) expect(%s)\n",OBJ_nid2ln(sigtype),
252                         OBJ_nid2ln(dtype));
253         #endif
254                 if (sigtype != dtype)
255                         {
256                         if (((dtype == NID_md5) &&
257                                 (sigtype == NID_md5WithRSAEncryption)) ||
258                                 ((dtype == NID_md2) &&
259                                 (sigtype == NID_md2WithRSAEncryption)))
```

```
260                                 {
261                                 /* ok, we will let it through */
262 #if !defined(OPENSSL_NO_STDIO) && !defined(OPENSSL_SYS_WIN16)
263                                 fprintf(stderr,"signature has problems, re-make
264 #endif
265                                 }
266                         else
267                                 {
268                                 RSAerr(RSA_F_INT_RSA_VERIFY,
269                                                 RSA_R_ALGORITHM_MISMATCH);
270                                 goto err;
271                                 }
272                         }
273                 if (rm)
274                         {
275                         const EVP_MD *md;
276                         md = EVP_get_digestbynid(dtype);
277                         if (md && (EVP_MD_size(md) != sig->digest->length))
278                                 RSAerr(RSA_F_INT_RSA_VERIFY,
279                                                 RSA_R_INVALID_DIGEST_LENGTH);
280                         else
281                                 {
282                                 memcpy(rm, sig->digest->data,
283                                                         sig->digest->length);
284                                 *prm_len = sig->digest->length;
285                                 ret = 1;
286                                 }
287                         }
288                 else if (((unsigned int)sig->digest->length != m_len) ||
289                         (memcmp(m,sig->digest->data,m_len) != 0))
290                         {
291                         RSAerr(RSA_F_INT_RSA_VERIFY,RSA_R_BAD_SIGNATURE);
292                         }
293                 else
294                         ret=1;
295         }
296 err:
297         if (sig != NULL) X509_SIG_free(sig);
298         if (s != NULL)
299                 {
300                 OPENSSL_cleanse(s,(unsigned int)siglen);
301                 OPENSSL_free(s);
302                 }
303         return(ret);
304         }

306 int RSA_verify(int dtype, const unsigned char *m, unsigned int m_len,
307                 const unsigned char *sigbuf, unsigned int siglen,
308                 RSA *rsa)
309         {

311         if((rsa->flags & RSA_FLAG_SIGN_VER) && rsa->meth->rsa_verify)
312                 {
313                 return rsa->meth->rsa_verify(dtype, m, m_len,
314                         sigbuf, siglen, rsa);
315                 }

317         return int_rsa_verify(dtype, m, m_len, NULL, NULL, sigbuf, siglen, rsa);
318         }
319 #endif /* ! codereview */
```

```
**********************************************************
    4933 Wed Aug 13 19:53:18 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_ssl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/rsa/rsa_ssl.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/rand.h>

int RSA_padding_add_SSLv23(unsigned char *to, int tlen,
        const unsigned char *from, int flen)
        {
        int i,j;
        unsigned char *p;

        if (flen > (tlen-11))
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_SSLV23,RSA_R_DATA_TOO_LARGE_FOR_KEY
                return(0);
                }

        p=(unsigned char *)to;

        *(p++)=0;
        *(p++)=2; /* Public Key BT (Block Type) */

        /* pad out with non-zero random data */
        j=tlen-3-8-flen;

        if (RAND_bytes(p,j) <= 0)
                return(0);
        for (i=0; i<j; i++)
                {
                if (*p == '\0')
                        do      {
                                if (RAND_bytes(p,1) <= 0)
                                        return(0);
                                } while (*p == '\0');
                p++;
                }

        memset(p,3,8);
        p+=8;
        *(p++)='\0';

        memcpy(p,from,(unsigned int)flen);
        return(1);
        }

int RSA_padding_check_SSLv23(unsigned char *to, int tlen,
        const unsigned char *from, int flen, int num)
        {
        int i,j,k;
        const unsigned char *p;

        p=from;
        if (flen < 10)
                {
                RSAerr(RSA_F_RSA_PADDING_CHECK_SSLV23,RSA_R_DATA_TOO_SMALL);
                return(-1);
                }
        if ((num != (flen+1)) || (*(p++) != 02))
                {
                RSAerr(RSA_F_RSA_PADDING_CHECK_SSLV23,RSA_R_BLOCK_TYPE_IS_NOT_02
                return(-1);
                }

        /* scan over padding data */
        j=flen-1; /* one for type */
        for (i=0; i<j; i++)
                if (*(p++) == 0) break;
```

```
128         if ((i == j) || (i < 8))
129                 {
130                 RSAerr(RSA_F_RSA_PADDING_CHECK_SSLV23,RSA_R_NULL_BEFORE_BLOCK_MI
131                 return(-1);
132                 }
133         for (k = -9; k<-1; k++)
134                 {
135                 if (p[k] !=  0x03) break;
136                 }
137         if (k == -1)
138                 {
139                 RSAerr(RSA_F_RSA_PADDING_CHECK_SSLV23,RSA_R_SSLV3_ROLLBACK_ATTAC
140                 return(-1);
141                 }

143         i++; /* Skip over the '\0' */
144         j-=i;
145         if (j > tlen)
146                 {
147                 RSAerr(RSA_F_RSA_PADDING_CHECK_SSLV23,RSA_R_DATA_TOO_LARGE);
148                 return(-1);
149                 }
150         memcpy(to,p,(unsigned int)j);

152         return(j);
153         }
154 #endif /* ! codereview */
```

```
**********************************************************
    4664 Wed Aug 13 19:53:18 2014
new/usr/src/lib/openssl/libsunw_crypto/rsa/rsa_x931.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* rsa_x931.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2005.
 */
/* ====================================================================
 * Copyright (c) 2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/rsa.h>
#include <openssl/rand.h>
#include <openssl/objects.h>

int RSA_padding_add_X931(unsigned char *to, int tlen,
             const unsigned char *from, int flen)
        {
        int j;
        unsigned char *p;

        /* Absolute minimum amount of padding is 1 header nibble, 1 padding
         * nibble and 2 trailer bytes: but 1 hash if is already in 'from'.
         */

        j = tlen - flen - 2;

        if (j < 0)
                {
                RSAerr(RSA_F_RSA_PADDING_ADD_X931,RSA_R_DATA_TOO_LARGE_FOR_KEY_S
                return -1;
                }

        p=(unsigned char *)to;

        /* If no padding start and end nibbles are in one byte */
        if (j == 0)
                *p++ = 0x6A;
        else
                {
                *p++ = 0x6B;
                if (j > 1)
                        {
                        memset(p, 0xBB, j - 1);
                        p += j - 1;
                        }
                *p++ = 0xBA;
                }
        memcpy(p,from,(unsigned int)flen);
        p += flen;
        *p = 0xCC;
        return(1);
        }

int RSA_padding_check_X931(unsigned char *to, int tlen,
             const unsigned char *from, int flen, int num)
        {
        int i = 0,j;
        const unsigned char *p;

        p=from;
        if ((num != flen) || ((*p != 0x6A) && (*p != 0x6B)))
                {
                RSAerr(RSA_F_RSA_PADDING_CHECK_X931,RSA_R_INVALID_HEADER);
                return -1;
                }

        if (*p++ == 0x6B)
                {
                j=flen-3;
                for (i = 0; i < j; i++)
                        {
                        unsigned char c = *p++;
                        if (c == 0xBA)
                                break;
                        if (c != 0xBB)
                                {
```

```
128                                 RSAerr(RSA_F_RSA_PADDING_CHECK_X931,
129                                         RSA_R_INVALID_PADDING);
130                                 return -1;
131                                 }
132                         }

134                 j -= i;

136                 if (i == 0)
137                         {
138                         RSAerr(RSA_F_RSA_PADDING_CHECK_X931, RSA_R_INVALID_PADDI
139                         return -1;
140                         }

142                 }
143         else j = flen - 2;

145         if (p[j] != 0xCC)
146                 {
147                 RSAerr(RSA_F_RSA_PADDING_CHECK_X931, RSA_R_INVALID_TRAILER);
148                 return -1;
149                 }

151         memcpy(to,p,(unsigned int)j);

153         return(j);
154         }

156 /* Translate between X931 hash ids and NIDs */

158 int RSA_X931_hash_id(int nid)
159         {
160         switch (nid)
161                 {
162                 case NID_sha1:
163                 return 0x33;

165                 case NID_sha256:
166                 return 0x34;

168                 case NID_sha384:
169                 return 0x36;

171                 case NID_sha512:
172                 return 0x35;

174                 }
175         return -1;
176         }
177 #endif /* ! codereview */
```

```
**********************************************************
    3598 Wed Aug 13 19:53:18 2014
new/usr/src/lib/openssl/libsunw_crypto/sha/sha1_one.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha1_one.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <openssl/crypto.h>
```



```
#include <openssl/sha.h>

#ifndef OPENSSL_NO_SHA1
unsigned char *SHA1(const unsigned char *d, size_t n, unsigned char *md)
        {
        SHA_CTX c;
        static unsigned char m[SHA_DIGEST_LENGTH];

        if (md == NULL) md=m;
        if (!SHA1_Init(&c))
                return NULL;
        SHA1_Update(&c,d,n);
        SHA1_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c));
        return(md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    3508 Wed Aug 13 19:53:18 2014
new/usr/src/lib/openssl/libsunw_crypto/sha/sha1dgst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha1dgst.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/crypto.h>
#include <openssl/opensslconf.h>
#if !defined(OPENSSL_NO_SHA1) && !defined(OPENSSL_NO_SHA)
```



```
#undef  SHA_0
#define SHA_1

#include <openssl/opensslv.h>

const char SHA1_version[]="SHA1" OPENSSL_VERSION_PTEXT;

/* The implementation is in ../md32_common.h */

#include "sha_locl.h"

#endif
#endif /* ! codereview */
```

```
**********************************************************
    9306 Wed Aug 13 19:53:19 2014
new/usr/src/lib/openssl/libsunw_crypto/sha/sha256.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha256.c */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved
 * according to the OpenSSL license [found in ../../LICENSE].
 * ====================================================================
 */
#include <openssl/opensslconf.h>
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA256)

#include <stdlib.h>
#include <string.h>

#include <openssl/crypto.h>
#include <openssl/sha.h>
#include <openssl/opensslv.h>

const char SHA256_version[]="SHA-256" OPENSSL_VERSION_PTEXT;

fips_md_init_ctx(SHA224, SHA256)
        {
        memset (c,0,sizeof(*c));
        c->h[0]=0xc1059ed8UL;   c->h[1]=0x367cd507UL;
        c->h[2]=0x3070dd17UL;   c->h[3]=0xf70e5939UL;
        c->h[4]=0xffc00b31UL;   c->h[5]=0x68581511UL;
        c->h[6]=0x64f98fa7UL;   c->h[7]=0xbefa4fa4UL;
        c->md_len=SHA224_DIGEST_LENGTH;
        return 1;
        }

fips_md_init(SHA256)
        {
        memset (c,0,sizeof(*c));
        c->h[0]=0x6a09e667UL;   c->h[1]=0xbb67ae85UL;
        c->h[2]=0x3c6ef372UL;   c->h[3]=0xa54ff53aUL;
        c->h[4]=0x510e527fUL;   c->h[5]=0x9b05688cUL;
        c->h[6]=0x1f83d9abUL;   c->h[7]=0x5be0cd19UL;
        c->md_len=SHA256_DIGEST_LENGTH;
        return 1;
        }

unsigned char *SHA224(const unsigned char *d, size_t n, unsigned char *md)
        {
        SHA256_CTX c;
        static unsigned char m[SHA224_DIGEST_LENGTH];

        if (md == NULL) md=m;
        SHA224_Init(&c);
        SHA256_Update(&c,d,n);
        SHA256_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c));
        return(md);
        }

unsigned char *SHA256(const unsigned char *d, size_t n, unsigned char *md)
        {
        SHA256_CTX c;
        static unsigned char m[SHA256_DIGEST_LENGTH];

        if (md == NULL) md=m;
        SHA256_Init(&c);
        SHA256_Update(&c,d,n);
```



```
        SHA256_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c));
        return(md);
        }

int SHA224_Update(SHA256_CTX *c, const void *data, size_t len)
{   return SHA256_Update (c,data,len);   }
int SHA224_Final (unsigned char *md, SHA256_CTX *c)
{   return SHA256_Final (md,c);   }

#define DATA_ORDER_IS_BIG_ENDIAN

#define HASH_LONG               SHA_LONG
#define HASH_CTX                SHA256_CTX
#define HASH_CBLOCK             SHA_CBLOCK
/*
 * Note that FIPS180-2 discusses "Truncation of the Hash Function Output."
 * default: case below covers for it. It's not clear however if it's
 * permitted to truncate to amount of bytes not divisible by 4. I bet not,
 * but if it is, then default: case shall be extended. For reference.
 * Idea behind separate cases for pre-defined lenghts is to let the
 * compiler decide if it's appropriate to unroll small loops.
 */
#define HASH_MAKE_STRING(c,s)   do {    \
        unsigned long ll;               \
        unsigned int  nn;               \
        switch ((c)->md_len)            \
        {   case SHA224_DIGEST_LENGTH:  \
                for (nn=0;nn<SHA224_DIGEST_LENGTH/4;nn++)       \
                {   ll=(c)->h[nn]; (void)HOST_l2c(ll,(s));   }  \
                break;                  \
            case SHA256_DIGEST_LENGTH:  \
                for (nn=0;nn<SHA256_DIGEST_LENGTH/4;nn++)       \
                {   ll=(c)->h[nn]; (void)HOST_l2c(ll,(s));   }  \
                break;                  \
            default:                    \
                if ((c)->md_len > SHA256_DIGEST_LENGTH) \
                    return 0;                           \
                for (nn=0;nn<(c)->md_len/4;nn++)        \
                {   ll=(c)->h[nn]; (void)HOST_l2c(ll,(s));   }  \
                break;                  \
        }                               \
        } while (0)

#define HASH_UPDATE             SHA256_Update
#define HASH_TRANSFORM          SHA256_Transform
#define HASH_FINAL              SHA256_Final
#define HASH_BLOCK_DATA_ORDER   sha256_block_data_order
#ifndef SHA256_ASM
static
#endif
void sha256_block_data_order (SHA256_CTX *ctx, const void *in, size_t num);

#include "md32_common.h"

#ifndef SHA256_ASM
static const SHA_LONG K256[64] = {
        0x428a2f98UL,0x71374491UL,0xb5c0fbcfUL,0xe9b5dba5UL,
        0x3956c25bUL,0x59f111f1UL,0x923f82a4UL,0xab1c5ed5UL,
        0xd807aa98UL,0x12835b01UL,0x243185beUL,0x550c7dc3UL,
        0x72be5d74UL,0x80deb1feUL,0x9bdc06a7UL,0xc19bf174UL,
        0xe49b69c1UL,0xefbe4786UL,0x0fc19dc6UL,0x240ca1ccUL,
        0x2de92c6fUL,0x4a7484aaUL,0x5cb0a9dcUL,0x76f988daUL,
        0x983e5152UL,0xa831c66dUL,0xb00327c8UL,0xbf597fc7UL,
        0xc6e00bf3UL,0xd5a79147UL,0x06ca6351UL,0x14292967UL,
        0x27b70a85UL,0x2e1b2138UL,0x4d2c6dfcUL,0x53380d13UL,
```

```
128         0x650a7354UL,0x766a0abbUL,0x81c2c92eUL,0x92722c85UL,
129         0xa2bfe8a1UL,0xa81a664bUL,0xc24b8b70UL,0xc76c51a3UL,
130         0xd192e819UL,0xd6990624UL,0xf40e3585UL,0x106aa070UL,
131         0x19a4c116UL,0x1e376c08UL,0x2748774cUL,0x34b0bcb5UL,
132         0x391c0cb3UL,0x4ed8aa4aUL,0x5b9cca4fUL,0x682e6ff3UL,
133         0x748f82eeUL,0x78a5636fUL,0x84c87814UL,0x8cc70208UL,
134         0x90befffaUL,0xa4506cebUL,0xbef9a3f7UL,0xc67178f2UL };

136 /*
137  * FIPS specification refers to right rotations, while our ROTATE macro
138  * is left one. This is why you might notice that rotation coefficients
139  * differ from those observed in FIPS document by 32-N...
140  */
141 #define Sigma0(x)       (ROTATE((x),30) ^ ROTATE((x),19) ^ ROTATE((x),10))
142 #define Sigma1(x)       (ROTATE((x),26) ^ ROTATE((x),21) ^ ROTATE((x),7))
143 #define sigma0(x)       (ROTATE((x),25) ^ ROTATE((x),14) ^ ((x)>>3))
144 #define sigma1(x)       (ROTATE((x),15) ^ ROTATE((x),13) ^ ((x)>>10))

146 #define Ch(x,y,z)       (((x) & (y)) ^ ((~(x)) & (z)))
147 #define Maj(x,y,z)      (((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z)))

149 #ifdef OPENSSL_SMALL_FOOTPRINT

151 static void sha256_block_data_order (SHA256_CTX *ctx, const void *in, size_t num
152         {
153         unsigned MD32_REG_T a,b,c,d,e,f,g,h,s0,s1,T1,T2;
154         SHA_LONG        X[16],l;
155         int i;
156         const unsigned char *data=in;

158                         while (num--) {

160         a = ctx->h[0];  b = ctx->h[1];  c = ctx->h[2];  d = ctx->h[3];
161         e = ctx->h[4];  f = ctx->h[5];  g = ctx->h[6];  h = ctx->h[7];

163         for (i=0;i<16;i++)
164                 {
165                 HOST_c2l(data,l); T1 = X[i] = l;
166                 T1 += h + Sigma1(e) + Ch(e,f,g) + K256[i];
167                 T2 = Sigma0(a) + Maj(a,b,c);
168                 h = g;  g = f;  f = e;  e = d + T1;
169                 d = c;  c = b;  b = a;  a = T1 + T2;
170                 }

172         for (;i<64;i++)
173                 {
174                 s0 = X[(i+1)&0x0f];     s0 = sigma0(s0);
175                 s1 = X[(i+14)&0x0f];    s1 = sigma1(s1);

177                 T1 = X[i&0xf] += s0 + s1 + X[(i+9)&0xf];
178                 T1 += h + Sigma1(e) + Ch(e,f,g) + K256[i];
179                 T2 = Sigma0(a) + Maj(a,b,c);
180                 h = g;  g = f;  f = e;  e = d + T1;
181                 d = c;  c = b;  b = a;  a = T1 + T2;
182                 }

184         ctx->h[0] += a; ctx->h[1] += b; ctx->h[2] += c; ctx->h[3] += d;
185         ctx->h[4] += e; ctx->h[5] += f; ctx->h[6] += g; ctx->h[7] += h;

187                         }
188 }

190 #else

192 #define ROUND_00_15(i,a,b,c,d,e,f,g,h)          do {    \
193         T1 += h + Sigma1(e) + Ch(e,f,g) + K256[i];      \
```



```
194         h = Sigma0(a) + Maj(a,b,c);                     \
195         d += T1;        h += T1;                } while (0)

197 #define ROUND_16_63(i,a,b,c,d,e,f,g,h,X)        do {    \
198         s0 = X[(i+1)&0x0f];     s0 = sigma0(s0);        \
199         s1 = X[(i+14)&0x0f];    s1 = sigma1(s1);        \
200         T1 = X[(i)&0x0f] += s0 + s1 + X[(i+9)&0x0f];    \
201         ROUND_00_15(i,a,b,c,d,e,f,g,h);         } while (0)

203 static void sha256_block_data_order (SHA256_CTX *ctx, const void *in, size_t num
204         {
205         unsigned MD32_REG_T a,b,c,d,e,f,g,h,s0,s1,T1;
206         SHA_LONG        X[16];
207         int i;
208         const unsigned char *data=in;
209         const union { long one; char little; } is_endian = {1};

211                         while (num--) {

213         a = ctx->h[0];  b = ctx->h[1];  c = ctx->h[2];  d = ctx->h[3];
214         e = ctx->h[4];  f = ctx->h[5];  g = ctx->h[6];  h = ctx->h[7];

216         if (!is_endian.little && sizeof(SHA_LONG)==4 && ((size_t)in%4)==0)
217                 {
218                 const SHA_LONG *W=(const SHA_LONG *)data;

220                 T1 = X[0] = W[0];       ROUND_00_15(0,a,b,c,d,e,f,g,h);
221                 T1 = X[1] = W[1];       ROUND_00_15(1,h,a,b,c,d,e,f,g);
222                 T1 = X[2] = W[2];       ROUND_00_15(2,g,h,a,b,c,d,e,f);
223                 T1 = X[3] = W[3];       ROUND_00_15(3,f,g,h,a,b,c,d,e);
224                 T1 = X[4] = W[4];       ROUND_00_15(4,e,f,g,h,a,b,c,d);
225                 T1 = X[5] = W[5];       ROUND_00_15(5,d,e,f,g,h,a,b,c);
226                 T1 = X[6] = W[6];       ROUND_00_15(6,c,d,e,f,g,h,a,b);
227                 T1 = X[7] = W[7];       ROUND_00_15(7,b,c,d,e,f,g,h,a);
228                 T1 = X[8] = W[8];       ROUND_00_15(8,a,b,c,d,e,f,g,h);
229                 T1 = X[9] = W[9];       ROUND_00_15(9,h,a,b,c,d,e,f,g);
230                 T1 = X[10] = W[10];     ROUND_00_15(10,g,h,a,b,c,d,e,f);
231                 T1 = X[11] = W[11];     ROUND_00_15(11,f,g,h,a,b,c,d,e);
232                 T1 = X[12] = W[12];     ROUND_00_15(12,e,f,g,h,a,b,c,d);
233                 T1 = X[13] = W[13];     ROUND_00_15(13,d,e,f,g,h,a,b,c);
234                 T1 = X[14] = W[14];     ROUND_00_15(14,c,d,e,f,g,h,a,b);
235                 T1 = X[15] = W[15];     ROUND_00_15(15,b,c,d,e,f,g,h,a);

237                 data += SHA256_CBLOCK;
238                 }
239         else
240                 {
241                 SHA_LONG l;

243                 HOST_c2l(data,l); T1 = X[0] = l;  ROUND_00_15(0,a,b,c,d,e,f,g,h)
244                 HOST_c2l(data,l); T1 = X[1] = l;  ROUND_00_15(1,h,a,b,c,d,e,f,g)
245                 HOST_c2l(data,l); T1 = X[2] = l;  ROUND_00_15(2,g,h,a,b,c,d,e,f)
246                 HOST_c2l(data,l); T1 = X[3] = l;  ROUND_00_15(3,f,g,h,a,b,c,d,e)
247                 HOST_c2l(data,l); T1 = X[4] = l;  ROUND_00_15(4,e,f,g,h,a,b,c,d)
248                 HOST_c2l(data,l); T1 = X[5] = l;  ROUND_00_15(5,d,e,f,g,h,a,b,c)
249                 HOST_c2l(data,l); T1 = X[6] = l;  ROUND_00_15(6,c,d,e,f,g,h,a,b)
250                 HOST_c2l(data,l); T1 = X[7] = l;  ROUND_00_15(7,b,c,d,e,f,g,h,a)
251                 HOST_c2l(data,l); T1 = X[8] = l;  ROUND_00_15(8,a,b,c,d,e,f,g,h)
252                 HOST_c2l(data,l); T1 = X[9] = l;  ROUND_00_15(9,h,a,b,c,d,e,f,g)
253                 HOST_c2l(data,l); T1 = X[10] = l; ROUND_00_15(10,g,h,a,b,c,d,e,f
254                 HOST_c2l(data,l); T1 = X[11] = l; ROUND_00_15(11,f,g,h,a,b,c,d,e
255                 HOST_c2l(data,l); T1 = X[12] = l; ROUND_00_15(12,e,f,g,h,a,b,c,d
256                 HOST_c2l(data,l); T1 = X[13] = l; ROUND_00_15(13,d,e,f,g,h,a,b,c
257                 HOST_c2l(data,l); T1 = X[14] = l; ROUND_00_15(14,c,d,e,f,g,h,a,b
258                 HOST_c2l(data,l); T1 = X[15] = l; ROUND_00_15(15,b,c,d,e,f,g,h,a
259                 }
```

```
261         for (i=16;i<64;i+=8)
262                 {
263                 ROUND_16_63(i+0,a,b,c,d,e,f,g,h,X);
264                 ROUND_16_63(i+1,h,a,b,c,d,e,f,g,X);
265                 ROUND_16_63(i+2,g,h,a,b,c,d,e,f,X);
266                 ROUND_16_63(i+3,f,g,h,a,b,c,d,e,X);
267                 ROUND_16_63(i+4,e,f,g,h,a,b,c,d,X);
268                 ROUND_16_63(i+5,d,e,f,g,h,a,b,c,X);
269                 ROUND_16_63(i+6,c,d,e,f,g,h,a,b,X);
270                 ROUND_16_63(i+7,b,c,d,e,f,g,h,a,X);
271                 }

273         ctx->h[0] += a; ctx->h[1] += b; ctx->h[2] += c; ctx->h[3] += d;
274         ctx->h[4] += e; ctx->h[5] += f; ctx->h[6] += g; ctx->h[7] += h;

276                         }
277         }

279 #endif
280 #endif /* SHA256_ASM */

282 #endif /* OPENSSL_NO_SHA256 */
283 #endif /* ! codereview */
```

```
**********************************************************
   18710 Wed Aug 13 19:53:19 2014
new/usr/src/lib/openssl/libsunw_crypto/sha/sha512.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha512.c */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved
 * according to the OpenSSL license [found in ../../LICENSE].
 * ====================================================================
 */
#include <openssl/opensslconf.h>
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA512)
/*
 * IMPLEMENTATION NOTES.
 *
 * As you might have noticed 32-bit hash algorithms:
 *
 * - permit SHA_LONG to be wider than 32-bit (case on CRAY);
 * - optimized versions implement two transform functions: one operating
 *   on [aligned] data in host byte order and one - on data in input
 *   stream byte order;
 * - share common byte-order neutral collector and padding function
 *   implementations, ../md32_common.h;
 *
 * Neither of the above applies to this SHA-512 implementations. Reasons
 * [in reverse order] are:
 *
 * - it's the only 64-bit hash algorithm for the moment of this writing,
 *   there is no need for common collector/padding implementation [yet];
 * - by supporting only one transform function [which operates on
 *   *aligned* data in input stream byte order, big-endian in this case]
 *   we minimize burden of maintenance in two ways: a) collector/padding
 *   function is simpler; b) only one transform function to stare at;
 * - SHA_LONG64 is required to be exactly 64-bit in order to be able to
 *   apply a number of optimizations to mitigate potential performance
 *   penalties caused by previous design decision;
 *
 * Caveat lector.
 *
 * Implementation relies on the fact that "long long" is 64-bit on
 * both 32- and 64-bit platforms. If some compiler vendor comes up
 * with 128-bit long long, adjustment to sha.h would be required.
 * As this implementation relies on 64-bit integer type, it's totally
 * inappropriate for platforms which don't support it, most notably
 * 16-bit platforms.
 *                                      <appro@fy.chalmers.se>
 */
#include <stdlib.h>
#include <string.h>

#include <openssl/crypto.h>
#include <openssl/sha.h>
#include <openssl/opensslv.h>

#include "cryptlib.h"

const char SHA512_version[]="SHA-512" OPENSSL_VERSION_PTEXT;

#if defined(__i386) || defined(__i386__) || defined(_M_IX86) || \
    defined(__x86_64) || defined(_M_AMD64) || defined(_M_X64) || \
    defined(__s390__) || defined(__s390x__) || \
    defined(SHA512_ASM)
#define SHA512_BLOCK_CAN_MANAGE_UNALIGNED_DATA
#endif
```



```
fips_md_init_ctx(SHA384, SHA512)
        {
        c->h[0]=U64(0xcbbb9d5dc1059ed8);
        c->h[1]=U64(0x629a292a367cd507);
        c->h[2]=U64(0x9159015a3070dd17);
        c->h[3]=U64(0x152fecd8f70e5939);
        c->h[4]=U64(0x67332667ffc00b31);
        c->h[5]=U64(0x8eb44a8768581511);
        c->h[6]=U64(0xdb0c2e0d64f98fa7);
        c->h[7]=U64(0x47b5481dbefa4fa4);

        c->Nl=0;        c->Nh=0;
        c->num=0;       c->md_len=SHA384_DIGEST_LENGTH;
        return 1;
        }

fips_md_init(SHA512)
        {
        c->h[0]=U64(0x6a09e667f3bcc908);
        c->h[1]=U64(0xbb67ae8584caa73b);
        c->h[2]=U64(0x3c6ef372fe94f82b);
        c->h[3]=U64(0xa54ff53a5f1d36f1);
        c->h[4]=U64(0x510e527fade682d1);
        c->h[5]=U64(0x9b05688c2b3e6c1f);
        c->h[6]=U64(0x1f83d9abfb41bd6b);
        c->h[7]=U64(0x5be0cd19137e2179);

        c->Nl=0;        c->Nh=0;
        c->num=0;       c->md_len=SHA512_DIGEST_LENGTH;
        return 1;
        }

#ifndef SHA512_ASM
static
#endif
void sha512_block_data_order (SHA512_CTX *ctx, const void *in, size_t num);

int SHA512_Final (unsigned char *md, SHA512_CTX *c)
        {
        unsigned char *p=(unsigned char *)c->u.p;
        size_t n=c->num;

        p[n]=0x80;      /* There always is a room for one */
        n++;
        if (n > (sizeof(c->u)-16))
                memset (p+n,0,sizeof(c->u)-n), n=0,
                sha512_block_data_order (c,p,1);

        memset (p+n,0,sizeof(c->u)-16-n);
#ifdef  B_ENDIAN
        c->u.d[SHA_LBLOCK-2] = c->Nh;
        c->u.d[SHA_LBLOCK-1] = c->Nl;
#else
        p[sizeof(c->u)-1]  = (unsigned char)(c->Nl);
        p[sizeof(c->u)-2]  = (unsigned char)(c->Nl>>8);
        p[sizeof(c->u)-3]  = (unsigned char)(c->Nl>>16);
        p[sizeof(c->u)-4]  = (unsigned char)(c->Nl>>24);
        p[sizeof(c->u)-5]  = (unsigned char)(c->Nl>>32);
        p[sizeof(c->u)-6]  = (unsigned char)(c->Nl>>40);
        p[sizeof(c->u)-7]  = (unsigned char)(c->Nl>>48);
        p[sizeof(c->u)-8]  = (unsigned char)(c->Nl>>56);
        p[sizeof(c->u)-9]  = (unsigned char)(c->Nh);
        p[sizeof(c->u)-10] = (unsigned char)(c->Nh>>8);
        p[sizeof(c->u)-11] = (unsigned char)(c->Nh>>16);
        p[sizeof(c->u)-12] = (unsigned char)(c->Nh>>24);
        p[sizeof(c->u)-13] = (unsigned char)(c->Nh>>32);
```

```
128         p[sizeof(c->u)-14] = (unsigned char)(c->Nh>>40);
129         p[sizeof(c->u)-15] = (unsigned char)(c->Nh>>48);
130         p[sizeof(c->u)-16] = (unsigned char)(c->Nh>>56);
131 #endif

133         sha512_block_data_order (c,p,1);

135         if (md==0) return 0;

137         switch (c->md_len)
138                 {
139                 /* Let compiler decide if it's appropriate to unroll... */
140                 case SHA384_DIGEST_LENGTH:
141                         for (n=0;n<SHA384_DIGEST_LENGTH/8;n++)
142                                 {
143                                 SHA_LONG64 t = c->h[n];

145                                 *(md++) = (unsigned char)(t>>56);
146                                 *(md++) = (unsigned char)(t>>48);
147                                 *(md++) = (unsigned char)(t>>40);
148                                 *(md++) = (unsigned char)(t>>32);
149                                 *(md++) = (unsigned char)(t>>24);
150                                 *(md++) = (unsigned char)(t>>16);
151                                 *(md++) = (unsigned char)(t>>8);
152                                 *(md++) = (unsigned char)(t);
153                                 }
154                         break;
155                 case SHA512_DIGEST_LENGTH:
156                         for (n=0;n<SHA512_DIGEST_LENGTH/8;n++)
157                                 {
158                                 SHA_LONG64 t = c->h[n];

160                                 *(md++) = (unsigned char)(t>>56);
161                                 *(md++) = (unsigned char)(t>>48);
162                                 *(md++) = (unsigned char)(t>>40);
163                                 *(md++) = (unsigned char)(t>>32);
164                                 *(md++) = (unsigned char)(t>>24);
165                                 *(md++) = (unsigned char)(t>>16);
166                                 *(md++) = (unsigned char)(t>>8);
167                                 *(md++) = (unsigned char)(t);
168                                 }
169                         break;
170                 /* ... as well as make sure md_len is not abused. */
171                 default:        return 0;
172                 }

174         return 1;
175         }

177 int SHA384_Final (unsigned char *md,SHA512_CTX *c)
178 {   return SHA512_Final (md,c);   }

180 int SHA512_Update (SHA512_CTX *c, const void *_data, size_t len)
181         {
182         SHA_LONG64      l;
183         unsigned char  *p=c->u.p;
184         const unsigned char *data=(const unsigned char *)_data;

186         if (len==0) return  1;

188         l = (c->Nl+(((SHA_LONG64)len)<<3))&U64(0xffffffffffffffff);
189         if (l < c->Nl)          c->Nh++;
190         if (sizeof(len)>=8)     c->Nh+=(((SHA_LONG64)len)>>61);
191         c->Nl=l;

193         if (c->num != 0)
```



```
194                 {
195                 size_t n = sizeof(c->u) - c->num;

197                 if (len < n)
198                         {
199                         memcpy (p+c->num,data,len), c->num += (unsigned int)len;
200                         return 1;
201                         }
202                 else    {
203                         memcpy (p+c->num,data,n), c->num = 0;
204                         len-=n, data+=n;
205                         sha512_block_data_order (c,p,1);
206                         }
207                 }

209         if (len >= sizeof(c->u))
210                 {
211 #ifndef SHA512_BLOCK_CAN_MANAGE_UNALIGNED_DATA
212                 if ((size_t)data%sizeof(c->u.d[0]) != 0)
213                         while (len >= sizeof(c->u))
214                                 memcpy (p,data,sizeof(c->u)),
215                                 sha512_block_data_order (c,p,1),
216                                 len  -= sizeof(c->u),
217                                 data += sizeof(c->u);
218                 else
219 #endif
220                         sha512_block_data_order (c,data,len/sizeof(c->u)),
221                         data += len,
222                         len  %= sizeof(c->u),
223                         data -= len;
224                 }

226         if (len != 0)   memcpy (p,data,len), c->num = (int)len;

228         return 1;
229         }

231 int SHA384_Update (SHA512_CTX *c, const void *data, size_t len)
232 {   return SHA512_Update (c,data,len);   }

234 void SHA512_Transform (SHA512_CTX *c, const unsigned char *data)
235         {
236 #ifndef SHA512_BLOCK_CAN_MANAGE_UNALIGNED_DATA
237         if ((size_t)data%sizeof(c->u.d[0]) != 0)
238                 memcpy(c->u.p,data,sizeof(c->u.p)),
239                 data = c->u.p;
240 #endif
241         sha512_block_data_order (c,data,1);
242         }

244 unsigned char *SHA384(const unsigned char *d, size_t n, unsigned char *md)
245         {
246         SHA512_CTX c;
247         static unsigned char m[SHA384_DIGEST_LENGTH];

249         if (md == NULL) md=m;
250         SHA384_Init(&c);
251         SHA512_Update(&c,d,n);
252         SHA512_Final(md,&c);
253         OPENSSL_cleanse(&c,sizeof(c));
254         return(md);
255         }

257 unsigned char *SHA512(const unsigned char *d, size_t n, unsigned char *md)
258         {
259         SHA512_CTX c;
```

```
260         static unsigned char m[SHA512_DIGEST_LENGTH];

262         if (md == NULL) md=m;
263         SHA512_Init(&c);
264         SHA512_Update(&c,d,n);
265         SHA512_Final(md,&c);
266         OPENSSL_cleanse(&c,sizeof(c));
267         return(md);
268         }

270 #ifndef SHA512_ASM
271 static const SHA_LONG64 K512[80] = {
272         U64(0x428a2f98d728ae22),U64(0x7137449123ef65cd),
273         U64(0xb5c0fbcfec4d3b2f),U64(0xe9b5dba58189dbbc),
274         U64(0x3956c25bf348b538),U64(0x59f111f1b605d019),
275         U64(0x923f82a4af194f9b),U64(0xab1c5ed5da6d8118),
276         U64(0xd807aa98a3030242),U64(0x12835b0145706fbe),
277         U64(0x243185be4ee4b28c),U64(0x550c7dc3d5ffb4e2),
278         U64(0x72be5d74f27b896f),U64(0x80deb1fe3b1696b1),
279         U64(0x9bdc06a725c71235),U64(0xc19bf174cf692694),
280         U64(0xe49b69c19ef14ad2),U64(0xefbe4786384f25e3),
281         U64(0x0fc19dc68b8cd5b5),U64(0x240ca1cc77ac9c65),
282         U64(0x2de92c6f592b0275),U64(0x4a7484aa6ea6e483),
283         U64(0x5cb0a9dcbd41fbd4),U64(0x76f988da831153b5),
284         U64(0x983e5152ee66dfab),U64(0xa831c66d2db43210),
285         U64(0xb00327c898fb213f),U64(0xbf597fc7beef0ee4),
286         U64(0xc6e00bf33da88fc2),U64(0xd5a79147930aa725),
287         U64(0x06ca6351e003826f),U64(0x142929670a0e6e70),
288         U64(0x27b70a8546d22ffc),U64(0x2e1b21385c26c926),
289         U64(0x4d2c6dfc5ac42aed),U64(0x53380d139d95b3df),
290         U64(0x650a73548baf63de),U64(0x766a0abb3c77b2a8),
291         U64(0x81c2c92e47edaee6),U64(0x92722c851482353b),
292         U64(0xa2bfe8a14cf10364),U64(0xa81a664bbc423001),
293         U64(0xc24b8b70d0f89791),U64(0xc76c51a30654be30),
294         U64(0xd192e819d6ef5218),U64(0xd69906245565a910),
295         U64(0xf40e35855771202a),U64(0x106aa07032bbd1b8),
296         U64(0x19a4c116b8d2d0c8),U64(0x1e376c085141ab53),
297         U64(0x2748774cdf8eeb99),U64(0x34b0bcb5e19b48a8),
298         U64(0x391c0cb3c5c95a63),U64(0x4ed8aa4ae3418acb),
299         U64(0x5b9cca4f7763e373),U64(0x682e6ff3d6b2b8a3),
300         U64(0x748f82ee5defb2fc),U64(0x78a5636f43172f60),
301         U64(0x84c87814a1f0ab72),U64(0x8cc702081a6439ec),
302         U64(0x90befffa23631e28),U64(0xa4506cebde82bde9),
303         U64(0xbef9a3f7b2c67915),U64(0xc67178f2e372532b),
304         U64(0xca273eceea26619c),U64(0xd186b8c721c0c207),
305         U64(0xeada7dd6cde0eb1e),U64(0xf57d4f7fee6ed178),
306         U64(0x06f067aa72176fba),U64(0x0a637dc5a2c898a6),
307         U64(0x113f9804bef90dae),U64(0x1b710b35131c471b),
308         U64(0x28db77f523047d84),U64(0x32caab7b40c72493),
309         U64(0x3c9ebe0a15c9bebc),U64(0x431d67c49c100d4c),
310         U64(0x4cc5d4becb3e42b6),U64(0x597f299cfc657e2a),
311         U64(0x5fcb6fab3ad6faec),U64(0x6c44198c4a475817) };

313 #ifndef PEDANTIC
314 # if defined(__GNUC__) && __GNUC__>=2 && !defined(OPENSSL_NO_ASM) && !defined(OP
315 #  if defined(__x86_64) || defined(__x86_64__)
316 #   define ROTR(a,n)    ({ SHA_LONG64 ret;              \
317                                 asm ("rorq %1,%0"       \
318                                 : "=r"(ret)             \
319                                 : "J"(n),"0"(a)         \
320                                 : "cc"); ret;           })
321 #   if !defined(B_ENDIAN)
322 #    define PULL64(x) ({ SHA_LONG64 ret=*((const SHA_LONG64 *)(&(x)));  \
323                                 asm ("bswapq    %0"             \
324                                 : "=r"(ret)                     \
325                                 : "0"(ret)); ret;               })
```



```
326 #   endif
327 #  elif (defined(__i386) || defined(__i386__)) && !defined(B_ENDIAN)
328 #   if defined(I386_ONLY)
329 #    define PULL64(x) ({ const unsigned int *p=(const unsigned int *)(&(x));\
330                                 unsigned int hi=p[0],lo=p[1];           \
331                                         asm("xchgb %%ah,%%al;xchgb %%dh,%%dl;"\
332                                             "roll $16,%%eax; roll $16,%%edx; "\
333                                             "xchgb %%ah,%%al;xchgb %%dh,%%dl;" \
334                                         : "=a"(lo),"=d"(hi)             \
335                                         : "0"(lo),"1"(hi) : "cc");      \
336                                         ((SHA_LONG64)hi)<<32|lo;        })
337 #   else
338 #    define PULL64(x) ({ const unsigned int *p=(const unsigned int *)(&(x));\
339                                 unsigned int hi=p[0],lo=p[1];           \
340                                         asm ("bswapl %0; bswapl %1;"    \
341                                         : "=r"(lo),"=r"(hi)             \
342                                         : "0"(lo),"1"(hi));             \
343                                         ((SHA_LONG64)hi)<<32|lo;        })
344 #   endif
345 #  elif (defined(_ARCH_PPC) && defined(__64BIT__)) || defined(_ARCH_PPC64)
346 #   define ROTR(a,n)    ({ SHA_LONG64 ret;                      \
347                                         asm ("rotrdi %0,%1,%2"  \
348                                         : "=r"(ret)             \
349                                         : "r"(a),"K"(n)); ret;  })
350 #  endif
351 # elif defined(_MSC_VER)
352 #  if defined(_WIN64)   /* applies to both IA-64 and AMD64 */
353 #   pragma intrinsic(_rotr64)
354 #   define ROTR(a,n)    _rotr64((a),n)
355 #  endif
356 #  if defined(_M_IX86) && !defined(OPENSSL_NO_ASM) && !defined(OPENSSL_NO_INLINE
357 #   if defined(I386_ONLY)
358     static SHA_LONG64 __fastcall __pull64be(const void *x)
359     {   _asm    mov     edx, [ecx + 0]
360         _asm    mov     eax, [ecx + 4]
361         _asm    xchg    dh,dl
362         _asm    xchg    ah,al
363         _asm    rol     edx,16
364         _asm    rol     eax,16
365         _asm    xchg    dh,dl
366         _asm    xchg    ah,al
367     }
368 #   else
369     static SHA_LONG64 __fastcall __pull64be(const void *x)
370     {   _asm    mov     edx, [ecx + 0]
371         _asm    mov     eax, [ecx + 4]
372         _asm    bswap   edx
373         _asm    bswap   eax
374     }
375 #   endif
376 #   define PULL64(x) __pull64be(&(x))
377 #   if _MSC_VER<=1200
378 #    pragma inline_depth(0)
379 #   endif
380 #  endif
381 # endif
382 #endif

384 #ifndef PULL64
385 #define B(x,j)    (((SHA_LONG64)(*(((const unsigned char *)(&x))+j)))<<((7-j)*8)
386 #define PULL64(x) (B(x,0)|B(x,1)|B(x,2)|B(x,3)|B(x,4)|B(x,5)|B(x,6)|B(x,7))
387 #endif

389 #ifndef ROTR
390 #define ROTR(x,s)       (((x)>>s) | (x)<<(64-s))
391 #endif
```

```
393 #define Sigma0(x)       (ROTR((x),28) ^ ROTR((x),34) ^ ROTR((x),39))
394 #define Sigma1(x)       (ROTR((x),14) ^ ROTR((x),18) ^ ROTR((x),41))
395 #define sigma0(x)       (ROTR((x),1)  ^ ROTR((x),8)  ^ ((x)>>7))
396 #define sigma1(x)       (ROTR((x),19) ^ ROTR((x),61) ^ ((x)>>6))

398 #define Ch(x,y,z)       (((x) & (y)) ^ ((~(x)) & (z)))
399 #define Maj(x,y,z)      (((x) & (y)) ^ ((x) & (z)) ^ ((y) & (z)))


402 #if defined(__i386) || defined(__i386__) || defined(_M_IX86)
403 /*
404  * This code should give better results on 32-bit CPU with less than
405  * ~24 registers, both size and performance wise...
406  */
407 static void sha512_block_data_order (SHA512_CTX *ctx, const void *in, size_t num
408         {
409         const SHA_LONG64 *W=in;
410         SHA_LONG64      A,E,T;
411         SHA_LONG64      X[9+80],*F;
412         int i;

414                         while (num--) {

416         F    = X+80;
417         A    = ctx->h[0];       F[1] = ctx->h[1];
418         F[2] = ctx->h[2];       F[3] = ctx->h[3];
419         E    = ctx->h[4];       F[5] = ctx->h[5];
420         F[6] = ctx->h[6];       F[7] = ctx->h[7];

422         for (i=0;i<16;i++,F--)
423                 {
424 #ifdef B_ENDIAN
425                 T = W[i];
426 #else
427                 T = PULL64(W[i]);
428 #endif
429                 F[0] = A;
430                 F[4] = E;
431                 F[8] = T;
432                 T   += F[7] + Sigma1(E) + Ch(E,F[5],F[6]) + K512[i];
433                 E    = F[3] + T;
434                 A    = T + Sigma0(A) + Maj(A,F[1],F[2]);
435                 }

437         for (;i<80;i++,F--)
438                 {
439                 T    = sigma0(F[8+16-1]);
440                 T   += sigma1(F[8+16-14]);
441                 T   += F[8+16] + F[8+16-9];

443                 F[0] = A;
444                 F[4] = E;
445                 F[8] = T;
446                 T   += F[7] + Sigma1(E) + Ch(E,F[5],F[6]) + K512[i];
447                 E    = F[3] + T;
448                 A    = T + Sigma0(A) + Maj(A,F[1],F[2]);
449                 }

451         ctx->h[0] += A;         ctx->h[1] += F[1];
452         ctx->h[2] += F[2];      ctx->h[3] += F[3];
453         ctx->h[4] += E;         ctx->h[5] += F[5];
454         ctx->h[6] += F[6];      ctx->h[7] += F[7];

456                         W+=SHA_LBLOCK;
457                         }
```



```
458         }

460 #elif defined(OPENSSL_SMALL_FOOTPRINT)

462 static void sha512_block_data_order (SHA512_CTX *ctx, const void *in, size_t num
463         {
464         const SHA_LONG64 *W=in;
465         SHA_LONG64      a,b,c,d,e,f,g,h,s0,s1,T1,T2;
466         SHA_LONG64      X[16];
467         int i;

469                         while (num--) {

471         a = ctx->h[0];  b = ctx->h[1];  c = ctx->h[2];  d = ctx->h[3];
472         e = ctx->h[4];  f = ctx->h[5];  g = ctx->h[6];  h = ctx->h[7];

474         for (i=0;i<16;i++)
475                 {
476 #ifdef B_ENDIAN
477                 T1 = X[i] = W[i];
478 #else
479                 T1 = X[i] = PULL64(W[i]);
480 #endif
481                 T1 += h + Sigma1(e) + Ch(e,f,g) + K512[i];
482                 T2 = Sigma0(a) + Maj(a,b,c);
483                 h = g;  g = f;  f = e;  e = d + T1;
484                 d = c;  c = b;  b = a;  a = T1 + T2;
485                 }

487         for (;i<80;i++)
488                 {
489                 s0 = X[(i+1)&0x0f];     s0 = sigma0(s0);
490                 s1 = X[(i+14)&0x0f];    s1 = sigma1(s1);

492                 T1 = X[i&0xf] += s0 + s1 + X[(i+9)&0xf];
493                 T1 += h + Sigma1(e) + Ch(e,f,g) + K512[i];
494                 T2 = Sigma0(a) + Maj(a,b,c);
495                 h = g;  g = f;  f = e;  e = d + T1;
496                 d = c;  c = b;  b = a;  a = T1 + T2;
497                 }

499         ctx->h[0] += a; ctx->h[1] += b; ctx->h[2] += c; ctx->h[3] += d;
500         ctx->h[4] += e; ctx->h[5] += f; ctx->h[6] += g; ctx->h[7] += h;

502                         W+=SHA_LBLOCK;
503                         }
504         }

506 #else

508 #define ROUND_00_15(i,a,b,c,d,e,f,g,h)          do {    \
509         T1 += h + Sigma1(e) + Ch(e,f,g) + K512[i];      \
510         h = Sigma0(a) + Maj(a,b,c);                     \
511         d += T1;        h += T1;                } while (0)

513 #define ROUND_16_80(i,j,a,b,c,d,e,f,g,h,X)      do {    \
514         s0 = X[(j+1)&0x0f];     s0 = sigma0(s0);        \
515         s1 = X[(j+14)&0x0f];    s1 = sigma1(s1);        \
516         T1 = X[(j)&0x0f] += s0 + s1 + X[(j+9)&0x0f];    \
517         ROUND_00_15(i+j,a,b,c,d,e,f,g,h);               } while (0)

519 static void sha512_block_data_order (SHA512_CTX *ctx, const void *in, size_t num
520         {
521         const SHA_LONG64 *W=in;
522         SHA_LONG64      a,b,c,d,e,f,g,h,s0,s1,T1;
523         SHA_LONG64      X[16];
```

```
524         int i;

526                         while (num--) {

528         a = ctx->h[0];  b = ctx->h[1];  c = ctx->h[2];  d = ctx->h[3];
529         e = ctx->h[4];  f = ctx->h[5];  g = ctx->h[6];  h = ctx->h[7];

531 #ifdef B_ENDIAN
532         T1 = X[0] = W[0];       ROUND_00_15(0,a,b,c,d,e,f,g,h);
533         T1 = X[1] = W[1];       ROUND_00_15(1,h,a,b,c,d,e,f,g);
534         T1 = X[2] = W[2];       ROUND_00_15(2,g,h,a,b,c,d,e,f);
535         T1 = X[3] = W[3];       ROUND_00_15(3,f,g,h,a,b,c,d,e);
536         T1 = X[4] = W[4];       ROUND_00_15(4,e,f,g,h,a,b,c,d);
537         T1 = X[5] = W[5];       ROUND_00_15(5,d,e,f,g,h,a,b,c);
538         T1 = X[6] = W[6];       ROUND_00_15(6,c,d,e,f,g,h,a,b);
539         T1 = X[7] = W[7];       ROUND_00_15(7,b,c,d,e,f,g,h,a);
540         T1 = X[8] = W[8];       ROUND_00_15(8,a,b,c,d,e,f,g,h);
541         T1 = X[9] = W[9];       ROUND_00_15(9,h,a,b,c,d,e,f,g);
542         T1 = X[10] = W[10];     ROUND_00_15(10,g,h,a,b,c,d,e,f);
543         T1 = X[11] = W[11];     ROUND_00_15(11,f,g,h,a,b,c,d,e);
544         T1 = X[12] = W[12];     ROUND_00_15(12,e,f,g,h,a,b,c,d);
545         T1 = X[13] = W[13];     ROUND_00_15(13,d,e,f,g,h,a,b,c);
546         T1 = X[14] = W[14];     ROUND_00_15(14,c,d,e,f,g,h,a,b);
547         T1 = X[15] = W[15];     ROUND_00_15(15,b,c,d,e,f,g,h,a);
548 #else
549         T1 = X[0]  = PULL64(W[0]);      ROUND_00_15(0,a,b,c,d,e,f,g,h);
550         T1 = X[1]  = PULL64(W[1]);      ROUND_00_15(1,h,a,b,c,d,e,f,g);
551         T1 = X[2]  = PULL64(W[2]);      ROUND_00_15(2,g,h,a,b,c,d,e,f);
552         T1 = X[3]  = PULL64(W[3]);      ROUND_00_15(3,f,g,h,a,b,c,d,e);
553         T1 = X[4]  = PULL64(W[4]);      ROUND_00_15(4,e,f,g,h,a,b,c,d);
554         T1 = X[5]  = PULL64(W[5]);      ROUND_00_15(5,d,e,f,g,h,a,b,c);
555         T1 = X[6]  = PULL64(W[6]);      ROUND_00_15(6,c,d,e,f,g,h,a,b);
556         T1 = X[7]  = PULL64(W[7]);      ROUND_00_15(7,b,c,d,e,f,g,h,a);
557         T1 = X[8]  = PULL64(W[8]);      ROUND_00_15(8,a,b,c,d,e,f,g,h);
558         T1 = X[9]  = PULL64(W[9]);      ROUND_00_15(9,h,a,b,c,d,e,f,g);
559         T1 = X[10] = PULL64(W[10]);     ROUND_00_15(10,g,h,a,b,c,d,e,f);
560         T1 = X[11] = PULL64(W[11]);     ROUND_00_15(11,f,g,h,a,b,c,d,e);
561         T1 = X[12] = PULL64(W[12]);     ROUND_00_15(12,e,f,g,h,a,b,c,d);
562         T1 = X[13] = PULL64(W[13]);     ROUND_00_15(13,d,e,f,g,h,a,b,c);
563         T1 = X[14] = PULL64(W[14]);     ROUND_00_15(14,c,d,e,f,g,h,a,b);
564         T1 = X[15] = PULL64(W[15]);     ROUND_00_15(15,b,c,d,e,f,g,h,a);
565 #endif

567         for (i=16;i<80;i+=16)
568                 {
569                 ROUND_16_80(i, 0,a,b,c,d,e,f,g,h,X);
570                 ROUND_16_80(i, 1,h,a,b,c,d,e,f,g,X);
571                 ROUND_16_80(i, 2,g,h,a,b,c,d,e,f,X);
572                 ROUND_16_80(i, 3,f,g,h,a,b,c,d,e,X);
573                 ROUND_16_80(i, 4,e,f,g,h,a,b,c,d,X);
574                 ROUND_16_80(i, 5,d,e,f,g,h,a,b,c,X);
575                 ROUND_16_80(i, 6,c,d,e,f,g,h,a,b,X);
576                 ROUND_16_80(i, 7,b,c,d,e,f,g,h,a,X);
577                 ROUND_16_80(i, 8,a,b,c,d,e,f,g,h,X);
578                 ROUND_16_80(i, 9,h,a,b,c,d,e,f,g,X);
579                 ROUND_16_80(i,10,g,h,a,b,c,d,e,f,X);
580                 ROUND_16_80(i,11,f,g,h,a,b,c,d,e,X);
581                 ROUND_16_80(i,12,e,f,g,h,a,b,c,d,X);
582                 ROUND_16_80(i,13,d,e,f,g,h,a,b,c,X);
583                 ROUND_16_80(i,14,c,d,e,f,g,h,a,b,X);
584                 ROUND_16_80(i,15,b,c,d,e,f,g,h,a,X);
585                 }

587         ctx->h[0] += a; ctx->h[1] += b; ctx->h[2] += c; ctx->h[3] += d;
588         ctx->h[4] += e; ctx->h[5] += f; ctx->h[6] += g; ctx->h[7] += h;
```



```
590                         W+=SHA_LBLOCK;
591                         }
592         }

594 #endif

596 #endif /* SHA512_ASM */

598 #else /* !OPENSSL_NO_SHA512 */

600 #if defined(PEDANTIC) || defined(__DECC) || defined(OPENSSL_SYS_MACOSX)
601 static void *dummy=&dummy;
602 #endif

604 #endif /* !OPENSSL_NO_SHA512 */
605 #endif /* ! codereview */
```

```
**********************************************************
    3506 Wed Aug 13 19:53:19 2014
new/usr/src/lib/openssl/libsunw_crypto/sha/sha_dgst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha1dgst.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <openssl/crypto.h>
#include <openssl/opensslconf.h>
#if !defined(OPENSSL_NO_SHA0) && !defined(OPENSSL_NO_SHA)
```



```
#undef  SHA_1
#define SHA_0

#include <openssl/opensslv.h>

const char SHA_version[]="SHA" OPENSSL_VERSION_PTEXT;

/* The implementation is in ../md32_common.h */

#include "sha_locl.h"

#endif
#endif /* ! codereview */
```

```
**********************************************************
    3593 Wed Aug 13 19:53:19 2014
new/usr/src/lib/openssl/libsunw_crypto/sha/sha_one.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/sha/sha_one.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <string.h>
#include <openssl/sha.h>
```



```
#include <openssl/crypto.h>

#ifndef OPENSSL_NO_SHA0
unsigned char *SHA(const unsigned char *d, size_t n, unsigned char *md)
        {
        SHA_CTX c;
        static unsigned char m[SHA_DIGEST_LENGTH];

        if (md == NULL) md=m;
        if (!SHA_Init(&c))
                return NULL;
        SHA_Update(&c,d,n);
        SHA_Final(md,&c);
        OPENSSL_cleanse(&c,sizeof(c));
        return(md);
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
    9565 Wed Aug 13 19:53:19 2014
new/usr/src/lib/openssl/libsunw_crypto/srp/srp_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/srp/srp_lib.c */
/* Written by Christophe Renou (christophe.renou@edelweb.fr) with
 * the precious help of Peter Sylvester (peter.sylvester@edelweb.fr)
 * for the EdelKey project and contributed to the OpenSSL project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#ifndef OPENSSL_NO_SRP
#include "cryptlib.h"
#include "srp_lcl.h"
```



```
#include <openssl/srp.h>
#include <openssl/evp.h>

#if (BN_BYTES == 8)
# if (defined(_WIN32) || defined(_WIN64)) && !defined(__MINGW32__)
#  define bn_pack4(a1,a2,a3,a4) ((a1##UI64<<48)|(a2##UI64<<32)|(a3##UI64<<16)|a4
# elif defined(__arch64__)
#  define bn_pack4(a1,a2,a3,a4) ((a1##UL<<48)|(a2##UL<<32)|(a3##UL<<16)|a4##UL)
# else
#  define bn_pack4(a1,a2,a3,a4) ((a1##ULL<<48)|(a2##ULL<<32)|(a3##ULL<<16)|a4##U
# endif
#elif (BN_BYTES == 4)
# define bn_pack4(a1,a2,a3,a4)  ((a3##UL<<16)|a4##UL), ((a1##UL<<16)|a2##UL)
#else
# error "unsupported BN_BYTES"
#endif


#include "srp_grps.h"

static BIGNUM *srp_Calc_k(BIGNUM *N, BIGNUM *g)
        {
        /* k = SHA1(N | PAD(g)) -- tls-srp draft 8 */

        unsigned char digest[SHA_DIGEST_LENGTH];
        unsigned char *tmp;
        EVP_MD_CTX ctxt;
        int longg ;
        int longN = BN_num_bytes(N);

        if (BN_ucmp(g, N) >= 0)
                return NULL;

        if ((tmp = OPENSSL_malloc(longN)) == NULL)
                return NULL;
        BN_bn2bin(N,tmp) ;

        EVP_MD_CTX_init(&ctxt);
        EVP_DigestInit_ex(&ctxt, EVP_sha1(), NULL);
        EVP_DigestUpdate(&ctxt, tmp, longN);

        memset(tmp, 0, longN);
        longg = BN_bn2bin(g,tmp) ;
        /* use the zeros behind to pad on left */
        EVP_DigestUpdate(&ctxt, tmp + longg, longN-longg);
        EVP_DigestUpdate(&ctxt, tmp, longg);
        OPENSSL_free(tmp);

        EVP_DigestFinal_ex(&ctxt, digest, NULL);
        EVP_MD_CTX_cleanup(&ctxt);
        return BN_bin2bn(digest, sizeof(digest), NULL);
        }

BIGNUM *SRP_Calc_u(BIGNUM *A, BIGNUM *B, BIGNUM *N)
        {
        /* k = SHA1(PAD(A) || PAD(B) ) -- tls-srp draft 8 */

        BIGNUM *u;
        unsigned char cu[SHA_DIGEST_LENGTH];
        unsigned char *cAB;
        EVP_MD_CTX ctxt;
        int longN;
        if ((A == NULL) ||(B == NULL) || (N == NULL))
                return NULL;

        if (BN_ucmp(A, N) >= 0 || BN_ucmp(B, N) >= 0)
```

```
128                 return NULL;

130         longN= BN_num_bytes(N);

132         if ((cAB = OPENSSL_malloc(2*longN)) == NULL)
133                 return NULL;

135         memset(cAB, 0, longN);

137         EVP_MD_CTX_init(&ctxt);
138         EVP_DigestInit_ex(&ctxt, EVP_sha1(), NULL);
139         EVP_DigestUpdate(&ctxt, cAB + BN_bn2bin(A,cAB+longN), longN);
140         EVP_DigestUpdate(&ctxt, cAB + BN_bn2bin(B,cAB+longN), longN);
141         OPENSSL_free(cAB);
142         EVP_DigestFinal_ex(&ctxt, cu, NULL);
143         EVP_MD_CTX_cleanup(&ctxt);

145         if (!(u = BN_bin2bn(cu, sizeof(cu), NULL)))
146                 return NULL;
147         if (!BN_is_zero(u))
148                 return u;
149         BN_free(u);
150         return NULL;
151 }

153 BIGNUM *SRP_Calc_server_key(BIGNUM *A, BIGNUM *v, BIGNUM *u, BIGNUM *b, BIGNUM *
154         {
155         BIGNUM *tmp = NULL, *S = NULL;
156         BN_CTX *bn_ctx;

158         if (u == NULL || A == NULL || v == NULL || b == NULL || N == NULL)
159                 return NULL;

161         if ((bn_ctx = BN_CTX_new()) == NULL ||
162                 (tmp = BN_new()) == NULL ||
163                 (S = BN_new()) == NULL )
164                 goto err;

166         /* S = (A*v**u) ** b */

168         if (!BN_mod_exp(tmp,v,u,N,bn_ctx))
169                 goto err;
170         if (!BN_mod_mul(tmp,A,tmp,N,bn_ctx))
171                 goto err;
172         if (!BN_mod_exp(S,tmp,b,N,bn_ctx))
173                 goto err;
174 err:
175         BN_CTX_free(bn_ctx);
176         BN_clear_free(tmp);
177         return S;
178         }

180 BIGNUM *SRP_Calc_B(BIGNUM *b, BIGNUM *N, BIGNUM *g, BIGNUM *v)
181         {
182         BIGNUM  *kv = NULL, *gb = NULL;
183         BIGNUM *B = NULL, *k = NULL;
184         BN_CTX *bn_ctx;

186         if (b == NULL || N == NULL || g == NULL || v == NULL ||
187                 (bn_ctx = BN_CTX_new()) == NULL)
188                 return NULL;

190         if ( (kv = BN_new()) == NULL ||
191                 (gb = BN_new()) == NULL ||
192                 (B = BN_new())== NULL)
193                 goto err;
```



```
195         /* B = g**b + k*v */

197         if (!BN_mod_exp(gb,g,b,N,bn_ctx) ||
198            !(k = srp_Calc_k(N,g)) ||
199            !BN_mod_mul(kv,v,k,N,bn_ctx) ||
200            !BN_mod_add(B,gb,kv,N,bn_ctx))
201                 {
202                 BN_free(B);
203                 B = NULL;
204                 }
205 err:
206         BN_CTX_free(bn_ctx);
207         BN_clear_free(kv);
208         BN_clear_free(gb);
209         BN_free(k);
210         return B;
211         }

213 BIGNUM *SRP_Calc_x(BIGNUM *s, const char *user, const char *pass)
214         {
215         unsigned char dig[SHA_DIGEST_LENGTH];
216         EVP_MD_CTX ctxt;
217         unsigned char *cs;

219         if ((s == NULL) ||
220                 (user == NULL) ||
221                 (pass == NULL))
222                 return NULL;

224         if ((cs = OPENSSL_malloc(BN_num_bytes(s))) == NULL)
225                 return NULL;

227         EVP_MD_CTX_init(&ctxt);
228         EVP_DigestInit_ex(&ctxt, EVP_sha1(), NULL);
229         EVP_DigestUpdate(&ctxt, user, strlen(user));
230         EVP_DigestUpdate(&ctxt, ":", 1);
231         EVP_DigestUpdate(&ctxt, pass, strlen(pass));
232         EVP_DigestFinal_ex(&ctxt, dig, NULL);

234         EVP_DigestInit_ex(&ctxt, EVP_sha1(), NULL);
235         BN_bn2bin(s,cs);
236         EVP_DigestUpdate(&ctxt, cs, BN_num_bytes(s));
237         OPENSSL_free(cs);
238         EVP_DigestUpdate(&ctxt, dig, sizeof(dig));
239         EVP_DigestFinal_ex(&ctxt, dig, NULL);
240         EVP_MD_CTX_cleanup(&ctxt);

242         return BN_bin2bn(dig, sizeof(dig), NULL);
243         }

245 BIGNUM *SRP_Calc_A(BIGNUM *a, BIGNUM *N, BIGNUM *g)
246         {
247         BN_CTX *bn_ctx;
248         BIGNUM * A = NULL;

250         if (a == NULL || N == NULL || g == NULL ||
251                 (bn_ctx = BN_CTX_new()) == NULL)
252                 return NULL;

254         if ((A = BN_new()) != NULL &&
255            !BN_mod_exp(A,g,a,N,bn_ctx))
256                 {
257                 BN_free(A);
258                 A = NULL;
259                 }
```

```
260         BN_CTX_free(bn_ctx);
261         return A;
262         }


265 BIGNUM *SRP_Calc_client_key(BIGNUM *N, BIGNUM *B, BIGNUM *g, BIGNUM *x, BIGNUM *
266         {
267         BIGNUM *tmp = NULL, *tmp2 = NULL, *tmp3 = NULL , *k = NULL, *K = NULL;
268         BN_CTX *bn_ctx;

270         if (u == NULL || B == NULL || N == NULL || g == NULL || x == NULL || a =
271                 (bn_ctx = BN_CTX_new()) == NULL)
272                 return NULL;

274         if ((tmp = BN_new()) == NULL ||
275                 (tmp2 = BN_new())== NULL ||
276                 (tmp3 = BN_new())== NULL ||
277                 (K = BN_new()) == NULL)
278                 goto err;

280         if (!BN_mod_exp(tmp,g,x,N,bn_ctx))
281                 goto err;
282         if (!(k = srp_Calc_k(N,g)))
283                 goto err;
284         if (!BN_mod_mul(tmp2,tmp,k,N,bn_ctx))
285                 goto err;
286         if (!BN_mod_sub(tmp,B,tmp2,N,bn_ctx))
287                 goto err;

289         if (!BN_mod_mul(tmp3,u,x,N,bn_ctx))
290                 goto err;
291         if (!BN_mod_add(tmp2,a,tmp3,N,bn_ctx))
292                 goto err;
293         if (!BN_mod_exp(K,tmp,tmp2,N,bn_ctx))
294                 goto err;

296 err :
297         BN_CTX_free(bn_ctx);
298         BN_clear_free(tmp);
299         BN_clear_free(tmp2);
300         BN_clear_free(tmp3);
301         BN_free(k);
302         return K;
303         }

305 int SRP_Verify_B_mod_N(BIGNUM *B, BIGNUM *N)
306         {
307         BIGNUM *r;
308         BN_CTX *bn_ctx;
309         int ret = 0;

311         if (B == NULL || N == NULL ||
312                 (bn_ctx = BN_CTX_new()) == NULL)
313                 return 0;

315         if ((r = BN_new()) == NULL)
316                 goto err;
317         /* Checks if B % N == 0 */
318         if (!BN_nnmod(r,B,N,bn_ctx))
319                 goto err;
320         ret = !BN_is_zero(r);
321 err:
322         BN_CTX_free(bn_ctx);
323         BN_free(r);
324         return ret;
325         }
```



```
327 int SRP_Verify_A_mod_N(BIGNUM *A, BIGNUM *N)
328         {
329         /* Checks if A % N == 0 */
330         return SRP_Verify_B_mod_N(A,N) ;
331         }


334 /* Check if G and N are kwown parameters.
335    The values have been generated from the ietf-tls-srp draft version 8
336 */
337 char *SRP_check_known_gN_param(BIGNUM *g, BIGNUM *N)
338         {
339         size_t i;
340         if ((g == NULL) || (N == NULL))
341                 return 0;

343         srp_bn_print(g);
344         srp_bn_print(N);

346         for(i = 0; i < KNOWN_GN_NUMBER; i++)
347                 {
348                 if (BN_cmp(knowngN[i].g, g) == 0 && BN_cmp(knowngN[i].N, N) == 0
349                         return knowngN[i].id;
350                 }
351         return NULL;
352         }

354 SRP_gN *SRP_get_default_gN(const char *id)
355         {
356         size_t i;

358         if (id == NULL)
359                 return knowngN;
360         for(i = 0; i < KNOWN_GN_NUMBER; i++)
361                 {
362                 if (strcmp(knowngN[i].id, id)==0)
363                         return knowngN + i;
364                 }
365         return NULL;
366         }
367 #endif
368 #endif /* ! codereview */
```

```
**********************************************************
   15243 Wed Aug 13 19:53:19 2014
new/usr/src/lib/openssl/libsunw_crypto/srp/srp_vfy.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/srp/srp_vfy.c */
/* Written by Christophe Renou (christophe.renou@edelweb.fr) with
 * the precious help of Peter Sylvester (peter.sylvester@edelweb.fr)
 * for the EdelKey project and contributed to the OpenSSL project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#ifndef OPENSSL_NO_SRP
#include "cryptlib.h"
#include "srp_lcl.h"
```



```
#include <openssl/srp.h>
#include <openssl/evp.h>
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/txt_db.h>

#define SRP_RANDOM_SALT_LEN 20
#define MAX_LEN 2500

static char b64table[] =
  "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz./";

/* the following two conversion routines have been inspired by code from Stanfor

/*
 * Convert a base64 string into raw byte array representation.
 */
static int t_fromb64(unsigned char *a, const char *src)
        {
        char *loc;
        int i, j;
        int size;

        while(*src && (*src == ' ' || *src == '\t' || *src == '\n'))
                ++src;
        size = strlen(src);
        i = 0;
        while(i < size)
                {
                loc = strchr(b64table, src[i]);
                if(loc == (char *) 0) break;
                else a[i] = loc - b64table;
                ++i;
                }
        /* if nothing valid to process we have a zero length response */
        if (i == 0)
                return 0;
        size = i;
        i = size - 1;
        j = size;
        while(1)
                {
                a[j] = a[i];
                if(--i < 0) break;
                a[j] |= (a[i] & 3) << 6;
                --j;
                a[j] = (unsigned char) ((a[i] & 0x3c) >> 2);
                if(--i < 0) break;
                a[j] |= (a[i] & 0xf) << 4;
                --j;
                a[j] = (unsigned char) ((a[i] & 0x30) >> 4);
                if(--i < 0) break;
                a[j] |= (a[i] << 2);

                a[--j] = 0;
                if(--i < 0) break;
                }
        while(a[j] == 0 && j <= size) ++j;
        i = 0;
        while (j <= size) a[i++] = a[j++];
        return i;
        }


/*
 * Convert a raw byte string into a null-terminated base64 ASCII string.
```

```
 */
static char *t_tob64(char *dst, const unsigned char *src, int size)
        {
        int c, pos = size % 3;
        unsigned char b0 = 0, b1 = 0, b2 = 0, notleading = 0;
        char *olddst = dst;

        switch(pos)
                {
        case 1:
                b2 = src[0];
                break;
        case 2:
                b1 = src[0];
                b2 = src[1];
                break;
                }

        while(1)
                {
                c = (b0 & 0xfc) >> 2;
                if(notleading || c != 0)
                        {
                        *dst++ = b64table[c];
                        notleading = 1;
                        }
                c = ((b0 & 3) << 4) | ((b1 & 0xf0) >> 4);
                if(notleading || c != 0)
                        {
                        *dst++ = b64table[c];
                        notleading = 1;
                        }
                c = ((b1 & 0xf) << 2) | ((b2 & 0xc0) >> 6);
                if(notleading || c != 0)
                        {
                        *dst++ = b64table[c];
                        notleading = 1;
                        }
                c = b2 & 0x3f;
                if(notleading || c != 0)
                        {
                        *dst++ = b64table[c];
                        notleading = 1;
                        }
                if(pos >= size) break;
                else
                        {
                        b0 = src[pos++];
                        b1 = src[pos++];
                        b2 = src[pos++];
                        }
                }

        *dst++ = '\0';
        return olddst;
        }

static void SRP_user_pwd_free(SRP_user_pwd *user_pwd)
        {
        if (user_pwd == NULL)
                return;
        BN_free(user_pwd->s);
        BN_clear_free(user_pwd->v);
        OPENSSL_free(user_pwd->id);
        OPENSSL_free(user_pwd->info);
        OPENSSL_free(user_pwd);
```



```
        }

static SRP_user_pwd *SRP_user_pwd_new()
        {
        SRP_user_pwd *ret = OPENSSL_malloc(sizeof(SRP_user_pwd));
        if (ret == NULL)
                return NULL;
        ret->N = NULL;
        ret->g = NULL;
        ret->s = NULL;
        ret->v = NULL;
        ret->id = NULL ;
        ret->info = NULL;
        return ret;
        }

static void SRP_user_pwd_set_gN(SRP_user_pwd *vinfo, const BIGNUM *g,
                                const BIGNUM *N)
        {
        vinfo->N = N;
        vinfo->g = g;
        }

static int SRP_user_pwd_set_ids(SRP_user_pwd *vinfo, const char *id,
                                const char *info)
        {
        if (id != NULL && NULL == (vinfo->id = BUF_strdup(id)))
                return 0;
        return (info == NULL || NULL != (vinfo->info = BUF_strdup(info))) ;
        }

static int SRP_user_pwd_set_sv(SRP_user_pwd *vinfo, const char *s,
                               const char *v)
        {
        unsigned char tmp[MAX_LEN];
        int len;

        if (strlen(s) > MAX_LEN || strlen(v) > MAX_LEN)
                return 0;
        len = t_fromb64(tmp, v);
        if (NULL == (vinfo->v = BN_bin2bn(tmp, len, NULL)) )
                return 0;
        len = t_fromb64(tmp, s);
        return ((vinfo->s = BN_bin2bn(tmp, len, NULL)) != NULL) ;
        }

static int SRP_user_pwd_set_sv_BN(SRP_user_pwd *vinfo, BIGNUM *s, BIGNUM *v)
        {
        vinfo->v = v;
        vinfo->s = s;
        return (vinfo->s != NULL && vinfo->v != NULL) ;
        }

SRP_VBASE *SRP_VBASE_new(char *seed_key)
        {
        SRP_VBASE *vb = (SRP_VBASE *) OPENSSL_malloc(sizeof(SRP_VBASE));

        if (vb == NULL)
                return NULL;
        if (!(vb->users_pwd = sk_SRP_user_pwd_new_null()) ||
                !(vb->gN_cache = sk_SRP_gN_cache_new_null()))
                {
                OPENSSL_free(vb);
                return NULL;
                }
        vb->default_g = NULL;
```

```
260         vb->default_N = NULL;
261         vb->seed_key = NULL;
262         if ((seed_key != NULL) &&
263                 (vb->seed_key = BUF_strdup(seed_key)) == NULL)
264                 {
265                 sk_SRP_user_pwd_free(vb->users_pwd);
266                 sk_SRP_gN_cache_free(vb->gN_cache);
267                 OPENSSL_free(vb);
268                 return NULL;
269                 }
270         return vb;
271         }


274 int SRP_VBASE_free(SRP_VBASE *vb)
275         {
276         sk_SRP_user_pwd_pop_free(vb->users_pwd,SRP_user_pwd_free);
277         sk_SRP_gN_cache_free(vb->gN_cache);
278         OPENSSL_free(vb->seed_key);
279         OPENSSL_free(vb);
280         return 0;
281         }


284 static SRP_gN_cache *SRP_gN_new_init(const char *ch)
285         {
286         unsigned char tmp[MAX_LEN];
287         int len;

289         SRP_gN_cache *newgN = (SRP_gN_cache *)OPENSSL_malloc(sizeof(SRP_gN_cache
290         if (newgN == NULL)
291                 return NULL;

293         if ((newgN->b64_bn = BUF_strdup(ch)) == NULL)
294                 goto err;

296         len = t_fromb64(tmp, ch);
297         if ((newgN->bn = BN_bin2bn(tmp, len, NULL)))
298                 return newgN;

300         OPENSSL_free(newgN->b64_bn);
301 err:
302         OPENSSL_free(newgN);
303         return NULL;
304         }


307 static void SRP_gN_free(SRP_gN_cache *gN_cache)
308         {
309         if (gN_cache == NULL)
310                 return;
311         OPENSSL_free(gN_cache->b64_bn);
312         BN_free(gN_cache->bn);
313         OPENSSL_free(gN_cache);
314         }

316 static SRP_gN *SRP_get_gN_by_id(const char *id, STACK_OF(SRP_gN) *gN_tab)
317         {
318         int i;

320         SRP_gN *gN;
321         if (gN_tab != NULL)
322         for(i = 0; i < sk_SRP_gN_num(gN_tab); i++)
323                 {
324                 gN = sk_SRP_gN_value(gN_tab, i);
325                 if (gN && (id == NULL || strcmp(gN->id,id)==0))
```



```
326                         return gN;
327                 }

329         return SRP_get_default_gN(id);
330         }

332 static BIGNUM *SRP_gN_place_bn(STACK_OF(SRP_gN_cache) *gN_cache, char *ch)
333         {
334         int i;
335         if (gN_cache == NULL)
336                 return NULL;

338         /* search if we have already one... */
339         for(i = 0; i < sk_SRP_gN_cache_num(gN_cache); i++)
340                 {
341                 SRP_gN_cache *cache = sk_SRP_gN_cache_value(gN_cache, i);
342                 if (strcmp(cache->b64_bn,ch)==0)
343                         return cache->bn;
344                 }
345                 {               /* it is the first time that we find it */
346                 SRP_gN_cache *newgN = SRP_gN_new_init(ch);
347                 if (newgN)
348                         {
349                         if (sk_SRP_gN_cache_insert(gN_cache,newgN,0)>0)
350                                 return newgN->bn;
351                         SRP_gN_free(newgN);
352                         }
353                 }
354         return NULL;
355         }

357 /* this function parses verifier file. Format is:
358  * string(index):base64(N):base64(g):0
359  * string(username):base64(v):base64(salt):int(index)
360  */


363 int SRP_VBASE_init(SRP_VBASE *vb, char *verifier_file)
364         {
365         int error_code ;
366         STACK_OF(SRP_gN) *SRP_gN_tab = sk_SRP_gN_new_null();
367         char *last_index = NULL;
368         int i;
369         char **pp;

371         SRP_gN *gN = NULL;
372         SRP_user_pwd *user_pwd = NULL ;

374         TXT_DB *tmpdb = NULL;
375         BIO *in = BIO_new(BIO_s_file());

377         error_code = SRP_ERR_OPEN_FILE;

379         if (in == NULL || BIO_read_filename(in,verifier_file) <= 0)
380                 goto err;

382         error_code = SRP_ERR_VBASE_INCOMPLETE_FILE;

384         if ((tmpdb =TXT_DB_read(in,DB_NUMBER)) == NULL)
385                 goto err;

387         error_code = SRP_ERR_MEMORY;


390         if (vb->seed_key)
391                 {
```

```
392                 last_index = SRP_get_default_gN(NULL)->id;
393                 }
394         for (i = 0; i < sk_OPENSSL_PSTRING_num(tmpdb->data); i++)
395                 {
396                 pp = sk_OPENSSL_PSTRING_value(tmpdb->data,i);
397                 if (pp[DB_srptype][0] == DB_SRP_INDEX)
398                         {
399                         /*we add this couple in the internal Stack */

401                         if ((gN = (SRP_gN *)OPENSSL_malloc(sizeof(SRP_gN))) == N
402                                 goto err;

404                         if  (!(gN->id = BUF_strdup(pp[DB_srpid]))
405                         ||  !(gN->N = SRP_gN_place_bn(vb->gN_cache,pp[DB_srpveri
406                         ||  !(gN->g = SRP_gN_place_bn(vb->gN_cache,pp[DB_srpsalt
407                         ||  sk_SRP_gN_insert(SRP_gN_tab,gN,0) == 0)
408                                 goto err;

410                         gN = NULL;

412                         if (vb->seed_key != NULL)
413                                 {
414                                 last_index = pp[DB_srpid];
415                                 }
416                         }
417                 else if (pp[DB_srptype][0] == DB_SRP_VALID)
418                         {
419                         /* it is a user .... */
420                         SRP_gN *lgN;
421                         if ((lgN = SRP_get_gN_by_id(pp[DB_srpgN],SRP_gN_tab))!=N
422                                 {
423                                 error_code = SRP_ERR_MEMORY;
424                                 if ((user_pwd = SRP_user_pwd_new()) == NULL)
425                                         goto err;

427                                 SRP_user_pwd_set_gN(user_pwd,lgN->g,lgN->N);
428                                 if (!SRP_user_pwd_set_ids(user_pwd, pp[DB_srpid]
429                                         goto err;

431                                 error_code = SRP_ERR_VBASE_BN_LIB;
432                                 if (!SRP_user_pwd_set_sv(user_pwd, pp[DB_srpsalt
433                                         goto err;

435                                 if (sk_SRP_user_pwd_insert(vb->users_pwd, user_p
436                                         goto err;
437                                 user_pwd = NULL; /* abandon responsability */
438                                 }
439                         }
440                 }

442         if (last_index != NULL)
443                 {
444                 /* this means that we want to simulate a default user */

446                 if (((gN = SRP_get_gN_by_id(last_index,SRP_gN_tab))==NULL))
447                         {
448                         error_code = SRP_ERR_VBASE_BN_LIB;
449                         goto err;
450                         }
451                 vb->default_g = gN->g ;
452                 vb->default_N = gN->N ;
453                 gN = NULL ;
454                 }
455         error_code = SRP_NO_ERROR;

457  err:
```



```
458         /* there may be still some leaks to fix, if this fails, the application

460         if (gN != NULL)
461                 {
462                 OPENSSL_free(gN->id);
463                 OPENSSL_free(gN);
464                 }

466         SRP_user_pwd_free(user_pwd);

468         if (tmpdb) TXT_DB_free(tmpdb);
469         if (in) BIO_free_all(in);

471         sk_SRP_gN_free(SRP_gN_tab);

473         return error_code;

475         }


478 SRP_user_pwd *SRP_VBASE_get_by_user(SRP_VBASE *vb, char *username)
479         {
480         int i;
481         SRP_user_pwd *user;
482         unsigned char digv[SHA_DIGEST_LENGTH];
483         unsigned char digs[SHA_DIGEST_LENGTH];
484         EVP_MD_CTX ctxt;

486         if (vb == NULL)
487                 return NULL;
488         for(i = 0; i < sk_SRP_user_pwd_num(vb->users_pwd); i++)
489                 {
490                 user = sk_SRP_user_pwd_value(vb->users_pwd, i);
491                 if (strcmp(user->id,username)==0)
492                         return user;
493                 }
494         if ((vb->seed_key == NULL) ||
495                 (vb->default_g == NULL) ||
496                 (vb->default_N == NULL))
497                 return NULL;

499 /* if the user is unknown we set parameters as well if we have a seed_key */

501         if ((user = SRP_user_pwd_new()) == NULL)
502                 return NULL;

504         SRP_user_pwd_set_gN(user,vb->default_g,vb->default_N);

506         if (!SRP_user_pwd_set_ids(user,username,NULL))
507                 goto err;

509         RAND_pseudo_bytes(digv, SHA_DIGEST_LENGTH);
510         EVP_MD_CTX_init(&ctxt);
511         EVP_DigestInit_ex(&ctxt, EVP_sha1(), NULL);
512         EVP_DigestUpdate(&ctxt, vb->seed_key, strlen(vb->seed_key));
513         EVP_DigestUpdate(&ctxt, username, strlen(username));
514         EVP_DigestFinal_ex(&ctxt, digs, NULL);
515         EVP_MD_CTX_cleanup(&ctxt);
516         if (SRP_user_pwd_set_sv_BN(user, BN_bin2bn(digs,SHA_DIGEST_LENGTH,NULL),
517                 return user;

519 err:    SRP_user_pwd_free(user);
520         return NULL;
521         }
```

```
/*
   create a verifier (*salt,*verifier,g and N are in base64)
*/
char *SRP_create_verifier(const char *user, const char *pass, char **salt,
                          char **verifier, const char *N, const char *g)
        {
        int len;
        char * result=NULL;
        char *vf;
        BIGNUM *N_bn = NULL, *g_bn = NULL, *s = NULL, *v = NULL;
        unsigned char tmp[MAX_LEN];
        unsigned char tmp2[MAX_LEN];
        char * defgNid = NULL;

        if ((user == NULL)||
                (pass == NULL)||
                (salt == NULL)||
                (verifier == NULL))
                goto err;

        if (N)
                {
                if (!(len = t_fromb64(tmp, N))) goto err;
                N_bn = BN_bin2bn(tmp, len, NULL);
                if (!(len = t_fromb64(tmp, g))) goto err;
                g_bn = BN_bin2bn(tmp, len, NULL);
                defgNid = "*";
                }
        else
                {
                SRP_gN * gN = SRP_get_gN_by_id(g, NULL) ;
                if (gN == NULL)
                        goto err;
                N_bn = gN->N;
                g_bn = gN->g;
                defgNid = gN->id;
                }

        if (*salt == NULL)
                {
                RAND_pseudo_bytes(tmp2, SRP_RANDOM_SALT_LEN);

                s = BN_bin2bn(tmp2, SRP_RANDOM_SALT_LEN, NULL);
                }
        else
                {
                if (!(len = t_fromb64(tmp2, *salt)))
                        goto err;
                s = BN_bin2bn(tmp2, len, NULL);
                }


        if(!SRP_create_verifier_BN(user, pass, &s, &v, N_bn, g_bn)) goto err;

        BN_bn2bin(v,tmp);
        if (((vf = OPENSSL_malloc(BN_num_bytes(v)*2)) == NULL))
                goto err;
        t_tob64(vf, tmp, BN_num_bytes(v));

        *verifier = vf;
        if (*salt == NULL)
                {
                char *tmp_salt;

                if ((tmp_salt = OPENSSL_malloc(SRP_RANDOM_SALT_LEN * 2)) == NULL
                        {
```



```
                        OPENSSL_free(vf);
                        goto err;
                        }
                t_tob64(tmp_salt, tmp2, SRP_RANDOM_SALT_LEN);
                *salt = tmp_salt;
                }

        result=defgNid;

err:
        if(N)
                {
                BN_free(N_bn);
                BN_free(g_bn);
                }
        return result;
        }

/*
   create a verifier (*salt,*verifier,g and N are BIGNUMs)
*/
int SRP_create_verifier_BN(const char *user, const char *pass, BIGNUM **salt, BI
        {
        int result=0;
        BIGNUM *x = NULL;
        BN_CTX *bn_ctx = BN_CTX_new();
        unsigned char tmp2[MAX_LEN];

        if ((user == NULL)||
                (pass == NULL)||
                (salt == NULL)||
                (verifier == NULL)||
                (N == NULL)||
                (g == NULL)||
                (bn_ctx == NULL))
                goto err;

        srp_bn_print(N);
        srp_bn_print(g);

        if (*salt == NULL)
                {
                RAND_pseudo_bytes(tmp2, SRP_RANDOM_SALT_LEN);

                *salt = BN_bin2bn(tmp2,SRP_RANDOM_SALT_LEN,NULL);
                }

        x = SRP_Calc_x(*salt,user,pass);

        *verifier = BN_new();
        if(*verifier == NULL) goto err;

        if (!BN_mod_exp(*verifier,g,x,N,bn_ctx))
                {
                BN_clear_free(*verifier);
                goto err;
                }

        srp_bn_print(*verifier);

        result=1;

err:

        BN_clear_free(x);
        BN_CTX_free(bn_ctx);
```

```
656         return result;
657         }



661 #endif
662 #endif /* ! codereview */
```

```
**********************************************************
    8553 Wed Aug 13 19:53:20 2014
new/usr/src/lib/openssl/libsunw_crypto/stack/stack.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/stack/stack.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

/* Code for stacks
 * Author - Eric Young v 1.0
 * 1.2 eay 12-Mar-97 -  Modified sk_find so that it _DOES_ return the
```



```
 *                      lowest index for the searched item.
 *
 * 1.1 eay - Take from netdb and added to SSLeay
 *
 * 1.0 eay - First version 29/07/92
 */
#include <stdio.h>
#include "cryptlib.h"
#include <openssl/stack.h>
#include <openssl/objects.h>

#undef MIN_NODES
#define MIN_NODES       4

const char STACK_version[]="Stack" OPENSSL_VERSION_PTEXT;

#include <errno.h>

int (*sk_set_cmp_func(_STACK *sk, int (*c)(const void *, const void *)))
                (const void *, const void *)
        {
        int (*old)(const void *,const void *)=sk->comp;

        if (sk->comp != c)
                sk->sorted=0;
        sk->comp=c;

        return old;
        }

_STACK *sk_dup(_STACK *sk)
        {
        _STACK *ret;
        char **s;

        if ((ret=sk_new(sk->comp)) == NULL) goto err;
        s=(char **)OPENSSL_realloc((char *)ret->data,
                (unsigned int)sizeof(char *)*sk->num_alloc);
        if (s == NULL) goto err;
        ret->data=s;

        ret->num=sk->num;
        memcpy(ret->data,sk->data,sizeof(char *)*sk->num);
        ret->sorted=sk->sorted;
        ret->num_alloc=sk->num_alloc;
        ret->comp=sk->comp;
        return(ret);
err:
        if(ret)
                sk_free(ret);
        return(NULL);
        }

_STACK *sk_new_null(void)
        {
        return sk_new((int (*)(const void *, const void *))0);
        }

_STACK *sk_new(int (*c)(const void *, const void *))
        {
        _STACK *ret;
        int i;

        if ((ret=OPENSSL_malloc(sizeof(_STACK))) == NULL)
                goto err;
        if ((ret->data=OPENSSL_malloc(sizeof(char *)*MIN_NODES)) == NULL)
```

```
128                 goto err;
129         for (i=0; i<MIN_NODES; i++)
130                 ret->data[i]=NULL;
131         ret->comp=c;
132         ret->num_alloc=MIN_NODES;
133         ret->num=0;
134         ret->sorted=0;
135         return(ret);
136 err:
137         if(ret)
138                 OPENSSL_free(ret);
139         return(NULL);
140         }

142 int sk_insert(_STACK *st, void *data, int loc)
143         {
144         char **s;

146         if(st == NULL) return 0;
147         if (st->num_alloc <= st->num+1)
148                 {
149                 s=OPENSSL_realloc((char *)st->data,
150                         (unsigned int)sizeof(char *)*st->num_alloc*2);
151                 if (s == NULL)
152                         return(0);
153                 st->data=s;
154                 st->num_alloc*=2;
155                 }
156         if ((loc >= (int)st->num) || (loc < 0))
157                 st->data[st->num]=data;
158         else
159                 {
160                 int i;
161                 char **f,**t;

163                 f=st->data;
164                 t=&(st->data[1]);
165                 for (i=st->num; i>=loc; i--)
166                         t[i]=f[i];

168 #ifdef undef /* no memmove on sunos :-( */
169                 memmove(&(st->data[loc+1]),
170                         &(st->data[loc]),
171                         sizeof(char *)*(st->num-loc));
172 #endif
173                 st->data[loc]=data;
174                 }
175         st->num++;
176         st->sorted=0;
177         return(st->num);
178         }

180 void *sk_delete_ptr(_STACK *st, void *p)
181         {
182         int i;

184         for (i=0; i<st->num; i++)
185                 if (st->data[i] == p)
186                         return(sk_delete(st,i));
187         return(NULL);
188         }

190 void *sk_delete(_STACK *st, int loc)
191         {
192         char *ret;
193         int i,j;
```



```
195         if(!st || (loc < 0) || (loc >= st->num)) return NULL;

197         ret=st->data[loc];
198         if (loc != st->num-1)
199                 {
200                 j=st->num-1;
201                 for (i=loc; i<j; i++)
202                         st->data[i]=st->data[i+1];
203                 /* In theory memcpy is not safe for this
204                  * memcpy( &(st->data[loc]),
205                  *      &(st->data[loc+1]),
206                  *      sizeof(char *)*(st->num-loc-1));
207                  */
208                 }
209         st->num--;
210         return(ret);
211         }

213 static int internal_find(_STACK *st, void *data, int ret_val_options)
214         {
215         const void * const *r;
216         int i;

218         if(st == NULL) return -1;

220         if (st->comp == NULL)
221                 {
222                 for (i=0; i<st->num; i++)
223                         if (st->data[i] == data)
224                                 return(i);
225                 return(-1);
226                 }
227         sk_sort(st);
228         if (data == NULL) return(-1);
229         r=OBJ_bsearch_ex_(&data,st->data,st->num,sizeof(void *),st->comp,
230                           ret_val_options);
231         if (r == NULL) return(-1);
232         return (int)((char **)r-st->data);
233         }

235 int sk_find(_STACK *st, void *data)
236         {
237         return internal_find(st, data, OBJ_BSEARCH_FIRST_VALUE_ON_MATCH);
238         }
239 int sk_find_ex(_STACK *st, void *data)
240         {
241         return internal_find(st, data, OBJ_BSEARCH_VALUE_ON_NOMATCH);
242         }

244 int sk_push(_STACK *st, void *data)
245         {
246         return(sk_insert(st,data,st->num));
247         }

249 int sk_unshift(_STACK *st, void *data)
250         {
251         return(sk_insert(st,data,0));
252         }

254 void *sk_shift(_STACK *st)
255         {
256         if (st == NULL) return(NULL);
257         if (st->num <= 0) return(NULL);
258         return(sk_delete(st,0));
259         }
```

```
261 void *sk_pop(_STACK *st)
262         {
263         if (st == NULL) return(NULL);
264         if (st->num <= 0) return(NULL);
265         return(sk_delete(st,st->num-1));
266         }

268 void sk_zero(_STACK *st)
269         {
270         if (st == NULL) return;
271         if (st->num <= 0) return;
272         memset((char *)st->data,0,sizeof(st->data)*st->num);
273         st->num=0;
274         }

276 void sk_pop_free(_STACK *st, void (*func)(void *))
277         {
278         int i;

280         if (st == NULL) return;
281         for (i=0; i<st->num; i++)
282                 if (st->data[i] != NULL)
283                         func(st->data[i]);
284         sk_free(st);
285         }

287 void sk_free(_STACK *st)
288         {
289         if (st == NULL) return;
290         if (st->data != NULL) OPENSSL_free(st->data);
291         OPENSSL_free(st);
292         }

294 int sk_num(const _STACK *st)
295 {
296         if(st == NULL) return -1;
297         return st->num;
298 }

300 void *sk_value(const _STACK *st, int i)
301 {
302         if(!st || (i < 0) || (i >= st->num)) return NULL;
303         return st->data[i];
304 }

306 void *sk_set(_STACK *st, int i, void *value)
307 {
308         if(!st || (i < 0) || (i >= st->num)) return NULL;
309         return (st->data[i] = value);
310 }

312 void sk_sort(_STACK *st)
313         {
314         if (st && !st->sorted)
315                 {
316                 int (*comp_func)(const void *,const void *);

318                 /* same comment as in sk_find ... previously st->comp was declar
319                  * as a (void*,void*) callback type, but this made the populatio
320                  * of the callback pointer illogical - our callbacks compare
321                  * type** with type**, so we leave the casting until absolutely
322                  * necessary (ie. "now"). */
323                 comp_func=(int (*)(const void *,const void *))(st->comp);
324                 qsort(st->data,st->num,sizeof(char *), comp_func);
325                 st->sorted=1;
```



```
326                 }
327         }

329 int sk_is_sorted(const _STACK *st)
330         {
331         if (!st)
332                 return 1;
333         return st->sorted;
334         }
335 #endif /* ! codereview */
```

```
**********************************************************
   10367 Wed Aug 13 19:53:20 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_asn1.c */
/* Written by Nils Larsch for the OpenSSL project 2004.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/ts.h>
#include <openssl/err.h>
#include <openssl/asn1t.h>
```



```
ASN1_SEQUENCE(TS_MSG_IMPRINT) = {
        ASN1_SIMPLE(TS_MSG_IMPRINT, hash_algo, X509_ALGOR),
        ASN1_SIMPLE(TS_MSG_IMPRINT, hashed_msg, ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END(TS_MSG_IMPRINT)

IMPLEMENT_ASN1_FUNCTIONS_const(TS_MSG_IMPRINT)
IMPLEMENT_ASN1_DUP_FUNCTION(TS_MSG_IMPRINT)
#ifndef OPENSSL_NO_BIO
TS_MSG_IMPRINT *d2i_TS_MSG_IMPRINT_bio(BIO *bp, TS_MSG_IMPRINT **a)
        {
        return ASN1_d2i_bio_of(TS_MSG_IMPRINT, TS_MSG_IMPRINT_new, d2i_TS_MSG_IM
        }

int i2d_TS_MSG_IMPRINT_bio(BIO *bp, TS_MSG_IMPRINT *a)
{
        return ASN1_i2d_bio_of_const(TS_MSG_IMPRINT, i2d_TS_MSG_IMPRINT, bp, a);
}
#endif
#ifndef OPENSSL_NO_FP_API
TS_MSG_IMPRINT *d2i_TS_MSG_IMPRINT_fp(FILE *fp, TS_MSG_IMPRINT **a)
        {
        return ASN1_d2i_fp_of(TS_MSG_IMPRINT, TS_MSG_IMPRINT_new, d2i_TS_MSG_IMP
        }

int i2d_TS_MSG_IMPRINT_fp(FILE *fp, TS_MSG_IMPRINT *a)
        {
        return ASN1_i2d_fp_of_const(TS_MSG_IMPRINT, i2d_TS_MSG_IMPRINT, fp, a);
        }
#endif

ASN1_SEQUENCE(TS_REQ) = {
        ASN1_SIMPLE(TS_REQ, version, ASN1_INTEGER),
        ASN1_SIMPLE(TS_REQ, msg_imprint, TS_MSG_IMPRINT),
        ASN1_OPT(TS_REQ, policy_id, ASN1_OBJECT),
        ASN1_OPT(TS_REQ, nonce, ASN1_INTEGER),
        ASN1_OPT(TS_REQ, cert_req, ASN1_FBOOLEAN),
        ASN1_IMP_SEQUENCE_OF_OPT(TS_REQ, extensions, X509_EXTENSION, 0)
} ASN1_SEQUENCE_END(TS_REQ)

IMPLEMENT_ASN1_FUNCTIONS_const(TS_REQ)
IMPLEMENT_ASN1_DUP_FUNCTION(TS_REQ)
#ifndef OPENSSL_NO_BIO
TS_REQ *d2i_TS_REQ_bio(BIO *bp, TS_REQ **a)
        {
        return ASN1_d2i_bio_of(TS_REQ, TS_REQ_new, d2i_TS_REQ, bp, a);
        }

int i2d_TS_REQ_bio(BIO *bp, TS_REQ *a)
        {
        return ASN1_i2d_bio_of_const(TS_REQ, i2d_TS_REQ, bp, a);
        }
#endif
#ifndef OPENSSL_NO_FP_API
TS_REQ *d2i_TS_REQ_fp(FILE *fp, TS_REQ **a)
        {
        return ASN1_d2i_fp_of(TS_REQ, TS_REQ_new, d2i_TS_REQ, fp, a);
        }

int i2d_TS_REQ_fp(FILE *fp, TS_REQ *a)
        {
        return ASN1_i2d_fp_of_const(TS_REQ, i2d_TS_REQ, fp, a);
        }
#endif

ASN1_SEQUENCE(TS_ACCURACY) = {
        ASN1_OPT(TS_ACCURACY, seconds, ASN1_INTEGER),
```

```
128         ASN1_IMP_OPT(TS_ACCURACY, millis, ASN1_INTEGER, 0),
129         ASN1_IMP_OPT(TS_ACCURACY, micros, ASN1_INTEGER, 1)
130 } ASN1_SEQUENCE_END(TS_ACCURACY)

132 IMPLEMENT_ASN1_FUNCTIONS_const(TS_ACCURACY)
133 IMPLEMENT_ASN1_DUP_FUNCTION(TS_ACCURACY)

135 ASN1_SEQUENCE(TS_TST_INFO) = {
136         ASN1_SIMPLE(TS_TST_INFO, version, ASN1_INTEGER),
137         ASN1_SIMPLE(TS_TST_INFO, policy_id, ASN1_OBJECT),
138         ASN1_SIMPLE(TS_TST_INFO, msg_imprint, TS_MSG_IMPRINT),
139         ASN1_SIMPLE(TS_TST_INFO, serial, ASN1_INTEGER),
140         ASN1_SIMPLE(TS_TST_INFO, time, ASN1_GENERALIZEDTIME),
141         ASN1_OPT(TS_TST_INFO, accuracy, TS_ACCURACY),
142         ASN1_OPT(TS_TST_INFO, ordering, ASN1_FBOOLEAN),
143         ASN1_OPT(TS_TST_INFO, nonce, ASN1_INTEGER),
144         ASN1_EXP_OPT(TS_TST_INFO, tsa, GENERAL_NAME, 0),
145         ASN1_IMP_SEQUENCE_OF_OPT(TS_TST_INFO, extensions, X509_EXTENSION, 1)
146 } ASN1_SEQUENCE_END(TS_TST_INFO)

148 IMPLEMENT_ASN1_FUNCTIONS_const(TS_TST_INFO)
149 IMPLEMENT_ASN1_DUP_FUNCTION(TS_TST_INFO)
150 #ifndef OPENSSL_NO_BIO
151 TS_TST_INFO *d2i_TS_TST_INFO_bio(BIO *bp, TS_TST_INFO **a)
152         {
153         return ASN1_d2i_bio_of(TS_TST_INFO, TS_TST_INFO_new, d2i_TS_TST_INFO, bp
154         }

156 int i2d_TS_TST_INFO_bio(BIO *bp, TS_TST_INFO *a)
157         {
158         return ASN1_i2d_bio_of_const(TS_TST_INFO, i2d_TS_TST_INFO, bp, a);
159         }
160 #endif
161 #ifndef OPENSSL_NO_FP_API
162 TS_TST_INFO *d2i_TS_TST_INFO_fp(FILE *fp, TS_TST_INFO **a)
163         {
164         return ASN1_d2i_fp_of(TS_TST_INFO, TS_TST_INFO_new, d2i_TS_TST_INFO, fp,
165         }

167 int i2d_TS_TST_INFO_fp(FILE *fp, TS_TST_INFO *a)
168         {
169         return ASN1_i2d_fp_of_const(TS_TST_INFO, i2d_TS_TST_INFO, fp, a);
170         }
171 #endif

173 ASN1_SEQUENCE(TS_STATUS_INFO) = {
174         ASN1_SIMPLE(TS_STATUS_INFO, status, ASN1_INTEGER),
175         ASN1_SEQUENCE_OF_OPT(TS_STATUS_INFO, text, ASN1_UTF8STRING),
176         ASN1_OPT(TS_STATUS_INFO, failure_info, ASN1_BIT_STRING)
177 } ASN1_SEQUENCE_END(TS_STATUS_INFO)

179 IMPLEMENT_ASN1_FUNCTIONS_const(TS_STATUS_INFO)
180 IMPLEMENT_ASN1_DUP_FUNCTION(TS_STATUS_INFO)

182 static int ts_resp_set_tst_info(TS_RESP *a)
183 {
184         long    status;

186         status = ASN1_INTEGER_get(a->status_info->status);

188         if (a->token) {
189                 if (status != 0 && status != 1) {
190                         TSerr(TS_F_TS_RESP_SET_TST_INFO, TS_R_TOKEN_PRESENT);
191                         return 0;
192                 }
193                 if (a->tst_info != NULL)
```



```
194                         TS_TST_INFO_free(a->tst_info);
195                 a->tst_info = PKCS7_to_TS_TST_INFO(a->token);
196                 if (!a->tst_info) {
197                         TSerr(TS_F_TS_RESP_SET_TST_INFO, TS_R_PKCS7_TO_TS_TST_IN
198                         return 0;
199                 }
200         } else if (status == 0 || status == 1) {
201                 TSerr(TS_F_TS_RESP_SET_TST_INFO, TS_R_TOKEN_NOT_PRESENT);
202                 return 0;
203         }

205         return 1;
206 }

208 static int ts_resp_cb(int op, ASN1_VALUE **pval, const ASN1_ITEM *it,
209         void *exarg)
210 {
211         TS_RESP *ts_resp = (TS_RESP *)*pval;
212         if (op == ASN1_OP_NEW_POST) {
213                 ts_resp->tst_info = NULL;
214         } else if (op == ASN1_OP_FREE_POST) {
215                 if (ts_resp->tst_info != NULL)
216                         TS_TST_INFO_free(ts_resp->tst_info);
217         } else if (op == ASN1_OP_D2I_POST) {
218                 if (ts_resp_set_tst_info(ts_resp) == 0)
219                         return 0;
220         }
221         return 1;
222 }

224 ASN1_SEQUENCE_cb(TS_RESP, ts_resp_cb) = {
225         ASN1_SIMPLE(TS_RESP, status_info, TS_STATUS_INFO),
226         ASN1_OPT(TS_RESP, token, PKCS7),
227 } ASN1_SEQUENCE_END_cb(TS_RESP, TS_RESP)

229 IMPLEMENT_ASN1_FUNCTIONS_const(TS_RESP)
230 IMPLEMENT_ASN1_DUP_FUNCTION(TS_RESP)
231 #ifndef OPENSSL_NO_BIO
232 TS_RESP *d2i_TS_RESP_bio(BIO *bp, TS_RESP **a)
233         {
234         return ASN1_d2i_bio_of(TS_RESP, TS_RESP_new, d2i_TS_RESP, bp, a);
235         }

237 int i2d_TS_RESP_bio(BIO *bp, TS_RESP *a)
238         {
239         return ASN1_i2d_bio_of_const(TS_RESP, i2d_TS_RESP, bp, a);
240         }
241 #endif
242 #ifndef OPENSSL_NO_FP_API
243 TS_RESP *d2i_TS_RESP_fp(FILE *fp, TS_RESP **a)
244         {
245         return ASN1_d2i_fp_of(TS_RESP, TS_RESP_new, d2i_TS_RESP, fp, a);
246         }

248 int i2d_TS_RESP_fp(FILE *fp, TS_RESP *a)
249         {
250         return ASN1_i2d_fp_of_const(TS_RESP, i2d_TS_RESP, fp, a);
251         }
252 #endif

254 ASN1_SEQUENCE(ESS_ISSUER_SERIAL) = {
255         ASN1_SEQUENCE_OF(ESS_ISSUER_SERIAL, issuer, GENERAL_NAME),
256         ASN1_SIMPLE(ESS_ISSUER_SERIAL, serial, ASN1_INTEGER)
257 } ASN1_SEQUENCE_END(ESS_ISSUER_SERIAL)

259 IMPLEMENT_ASN1_FUNCTIONS_const(ESS_ISSUER_SERIAL)
```

```
260 IMPLEMENT_ASN1_DUP_FUNCTION(ESS_ISSUER_SERIAL)

262 ASN1_SEQUENCE(ESS_CERT_ID) = {
263         ASN1_SIMPLE(ESS_CERT_ID, hash, ASN1_OCTET_STRING),
264         ASN1_OPT(ESS_CERT_ID, issuer_serial, ESS_ISSUER_SERIAL)
265 } ASN1_SEQUENCE_END(ESS_CERT_ID)

267 IMPLEMENT_ASN1_FUNCTIONS_const(ESS_CERT_ID)
268 IMPLEMENT_ASN1_DUP_FUNCTION(ESS_CERT_ID)

270 ASN1_SEQUENCE(ESS_SIGNING_CERT) = {
271         ASN1_SEQUENCE_OF(ESS_SIGNING_CERT, cert_ids, ESS_CERT_ID),
272         ASN1_SEQUENCE_OF_OPT(ESS_SIGNING_CERT, policy_info, POLICYINFO)
273 } ASN1_SEQUENCE_END(ESS_SIGNING_CERT)

275 IMPLEMENT_ASN1_FUNCTIONS_const(ESS_SIGNING_CERT)
276 IMPLEMENT_ASN1_DUP_FUNCTION(ESS_SIGNING_CERT)

278 /* Getting encapsulated TS_TST_INFO object from PKCS7. */
279 TS_TST_INFO *PKCS7_to_TS_TST_INFO(PKCS7 *token)
280 {
281         PKCS7_SIGNED *pkcs7_signed;
282         PKCS7 *enveloped;
283         ASN1_TYPE *tst_info_wrapper;
284         ASN1_OCTET_STRING *tst_info_der;
285         const unsigned char *p;

287         if (!PKCS7_type_is_signed(token))
288                 {
289                 TSerr(TS_F_PKCS7_TO_TS_TST_INFO, TS_R_BAD_PKCS7_TYPE);
290                 return NULL;
291                 }

293         /* Content must be present. */
294         if (PKCS7_get_detached(token))
295                 {
296                 TSerr(TS_F_PKCS7_TO_TS_TST_INFO, TS_R_DETACHED_CONTENT);
297                 return NULL;
298                 }

300         /* We have a signed data with content. */
301         pkcs7_signed = token->d.sign;
302         enveloped = pkcs7_signed->contents;
303         if (OBJ_obj2nid(enveloped->type) != NID_id_smime_ct_TSTInfo)
304                 {
305                 TSerr(TS_F_PKCS7_TO_TS_TST_INFO, TS_R_BAD_PKCS7_TYPE);
306                 return NULL;
307                 }

309         /* We have a DER encoded TST_INFO as the signed data. */
310         tst_info_wrapper = enveloped->d.other;
311         if (tst_info_wrapper->type != V_ASN1_OCTET_STRING)
312                 {
313                 TSerr(TS_F_PKCS7_TO_TS_TST_INFO, TS_R_BAD_TYPE);
314                 return NULL;
315                 }

317         /* We have the correct ASN1_OCTET_STRING type. */
318         tst_info_der = tst_info_wrapper->value.octet_string;
319         /* At last, decode the TST_INFO. */
320         p = tst_info_der->data;
321         return d2i_TS_TST_INFO(NULL, &p, tst_info_der->length);
322 }
323 #endif /* ! codereview */
```

```
**********************************************************
   13413 Wed Aug 13 19:53:20 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_conf.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_conf.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <string.h>

#include <openssl/crypto.h>
```



```
#include "cryptlib.h"
#include <openssl/pem.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include <openssl/ts.h>

/* Macro definitions for the configuration file. */

#define BASE_SECTION                    "tsa"
#define ENV_DEFAULT_TSA                 "default_tsa"
#define ENV_SERIAL                      "serial"
#define ENV_CRYPTO_DEVICE               "crypto_device"
#define ENV_SIGNER_CERT                 "signer_cert"
#define ENV_CERTS                       "certs"
#define ENV_SIGNER_KEY                  "signer_key"
#define ENV_DEFAULT_POLICY              "default_policy"
#define ENV_OTHER_POLICIES              "other_policies"
#define ENV_DIGESTS                     "digests"
#define ENV_ACCURACY                    "accuracy"
#define ENV_ORDERING                    "ordering"
#define ENV_TSA_NAME                    "tsa_name"
#define ENV_ESS_CERT_ID_CHAIN           "ess_cert_id_chain"
#define ENV_VALUE_SECS                  "secs"
#define ENV_VALUE_MILLISECS             "millisecs"
#define ENV_VALUE_MICROSECS             "microsecs"
#define ENV_CLOCK_PRECISION_DIGITS      "clock_precision_digits"
#define ENV_VALUE_YES                   "yes"
#define ENV_VALUE_NO                    "no"

/* Function definitions for certificate and key loading. */

X509 *TS_CONF_load_cert(const char *file)
        {
        BIO *cert = NULL;
        X509 *x = NULL;

        if ((cert = BIO_new_file(file, "r")) == NULL) goto end;
        x = PEM_read_bio_X509_AUX(cert, NULL, NULL, NULL);
end:
        if (x == NULL)
                fprintf(stderr, "unable to load certificate: %s\n", file);
        BIO_free(cert);
        return x;
        }

STACK_OF(X509) *TS_CONF_load_certs(const char *file)
        {
        BIO *certs = NULL;
        STACK_OF(X509) *othercerts = NULL;
        STACK_OF(X509_INFO) *allcerts = NULL;
        int i;

        if (!(certs = BIO_new_file(file, "r"))) goto end;

        if (!(othercerts = sk_X509_new_null())) goto end;
        allcerts = PEM_X509_INFO_read_bio(certs, NULL, NULL, NULL);
        for(i = 0; i < sk_X509_INFO_num(allcerts); i++)
                {
                X509_INFO *xi = sk_X509_INFO_value(allcerts, i);
                if (xi->x509)
                        {
                        sk_X509_push(othercerts, xi->x509);
                        xi->x509 = NULL;
                        }
                }
```

```
end:
        if (othercerts == NULL)
                fprintf(stderr, "unable to load certificates: %s\n", file);
        sk_X509_INFO_pop_free(allcerts, X509_INFO_free);
        BIO_free(certs);
        return othercerts;
        }

EVP_PKEY *TS_CONF_load_key(const char *file, const char *pass)
        {
        BIO *key = NULL;
        EVP_PKEY *pkey = NULL;

        if (!(key = BIO_new_file(file, "r"))) goto end;
        pkey = PEM_read_bio_PrivateKey(key, NULL, NULL, (char *) pass);
 end:
        if (pkey == NULL)
                fprintf(stderr, "unable to load private key: %s\n", file);
        BIO_free(key);
        return pkey;
        }

/* Function definitions for handling configuration options. */

static void TS_CONF_lookup_fail(const char *name, const char *tag)
        {
        fprintf(stderr, "variable lookup failed for %s::%s\n", name, tag);
        }

static void TS_CONF_invalid(const char *name, const char *tag)
        {
        fprintf(stderr, "invalid variable value for %s::%s\n", name, tag);
        }

const char *TS_CONF_get_tsa_section(CONF *conf, const char *section)
        {
        if (!section)
                {
                section = NCONF_get_string(conf, BASE_SECTION, ENV_DEFAULT_TSA);
                if (!section)
                        TS_CONF_lookup_fail(BASE_SECTION, ENV_DEFAULT_TSA);
                }
        return section;
        }

int TS_CONF_set_serial(CONF *conf, const char *section, TS_serial_cb cb,
                       TS_RESP_CTX *ctx)
        {
        int ret = 0;
        char *serial = NCONF_get_string(conf, section, ENV_SERIAL);
        if (!serial)
                {
                TS_CONF_lookup_fail(section, ENV_SERIAL);
                goto err;
                }
        TS_RESP_CTX_set_serial_cb(ctx, cb, serial);

        ret = 1;
 err:
        return ret;
        }

#ifndef OPENSSL_NO_ENGINE

int TS_CONF_set_crypto_device(CONF *conf, const char *section,
                              const char *device)
```



```
        {
        int ret = 0;

        if (!device)
                device = NCONF_get_string(conf, section,
                                          ENV_CRYPTO_DEVICE);

        if (device && !TS_CONF_set_default_engine(device))
                {
                TS_CONF_invalid(section, ENV_CRYPTO_DEVICE);
                goto err;
                }
        ret = 1;
 err:
        return ret;
        }

int TS_CONF_set_default_engine(const char *name)
        {
        ENGINE *e = NULL;
        int ret = 0;

        /* Leave the default if builtin specified. */
        if (strcmp(name, "builtin") == 0) return 1;

        if (!(e = ENGINE_by_id(name))) goto err;
        /* Enable the use of the NCipher HSM for forked children. */
        if (strcmp(name, "chil") == 0)
                ENGINE_ctrl(e, ENGINE_CTRL_CHIL_SET_FORKCHECK, 1, 0, 0);
        /* All the operations are going to be carried out by the engine. */
        if (!ENGINE_set_default(e, ENGINE_METHOD_ALL)) goto err;
        ret = 1;
 err:
        if (!ret)
                {
                TSerr(TS_F_TS_CONF_SET_DEFAULT_ENGINE,
                      TS_R_COULD_NOT_SET_ENGINE);
                ERR_add_error_data(2, "engine:", name);
                }
        if (e) ENGINE_free(e);
        return ret;
        }

#endif

int TS_CONF_set_signer_cert(CONF *conf, const char *section,
                            const char *cert, TS_RESP_CTX *ctx)
        {
        int ret = 0;
        X509 *cert_obj = NULL;
        if (!cert)
                cert = NCONF_get_string(conf, section, ENV_SIGNER_CERT);
        if (!cert)
                {
                TS_CONF_lookup_fail(section, ENV_SIGNER_CERT);
                goto err;
                }
        if (!(cert_obj = TS_CONF_load_cert(cert)))
                goto err;
        if (!TS_RESP_CTX_set_signer_cert(ctx, cert_obj))
                goto err;

        ret = 1;
 err:
        X509_free(cert_obj);
        return ret;
```

```
260         }

262 int TS_CONF_set_certs(CONF *conf, const char *section, const char *certs,
263                         TS_RESP_CTX *ctx)
264         {
265         int ret = 0;
266         STACK_OF(X509) *certs_obj = NULL;
267         if (!certs)
268                 certs = NCONF_get_string(conf, section, ENV_CERTS);
269         /* Certificate chain is optional. */
270         if (!certs) goto end;
271         if (!(certs_obj = TS_CONF_load_certs(certs))) goto err;
272         if (!TS_RESP_CTX_set_certs(ctx, certs_obj)) goto err;
273  end:
274         ret = 1;
275  err:
276         sk_X509_pop_free(certs_obj, X509_free);
277         return ret;
278         }

280 int TS_CONF_set_signer_key(CONF *conf, const char *section,
281                            const char *key, const char *pass,
282                            TS_RESP_CTX *ctx)
283         {
284         int ret = 0;
285         EVP_PKEY *key_obj = NULL;
286         if (!key)
287                 key = NCONF_get_string(conf, section, ENV_SIGNER_KEY);
288         if (!key)
289                 {
290                 TS_CONF_lookup_fail(section, ENV_SIGNER_KEY);
291                 goto err;
292                 }
293         if (!(key_obj = TS_CONF_load_key(key, pass))) goto err;
294         if (!TS_RESP_CTX_set_signer_key(ctx, key_obj)) goto err;

296         ret = 1;
297  err:
298         EVP_PKEY_free(key_obj);
299         return ret;
300         }

302 int TS_CONF_set_def_policy(CONF *conf, const char *section,
303                            const char *policy, TS_RESP_CTX *ctx)
304         {
305         int ret = 0;
306         ASN1_OBJECT *policy_obj = NULL;
307         if (!policy)
308                 policy = NCONF_get_string(conf, section,
309                                           ENV_DEFAULT_POLICY);
310         if (!policy)
311                 {
312                 TS_CONF_lookup_fail(section, ENV_DEFAULT_POLICY);
313                 goto err;
314                 }
315         if (!(policy_obj = OBJ_txt2obj(policy, 0)))
316                 {
317                 TS_CONF_invalid(section, ENV_DEFAULT_POLICY);
318                 goto err;
319                 }
320         if (!TS_RESP_CTX_set_def_policy(ctx, policy_obj))
321                 goto err;

323         ret = 1;
324  err:
325         ASN1_OBJECT_free(policy_obj);
```



```
326         return ret;
327         }

329 int TS_CONF_set_policies(CONF *conf, const char *section,
330                          TS_RESP_CTX *ctx)
331         {
332         int ret = 0;
333         int i;
334         STACK_OF(CONF_VALUE) *list = NULL;
335         char *policies = NCONF_get_string(conf, section,
336                                           ENV_OTHER_POLICIES);
337         /* If no other policy is specified, that's fine. */
338         if (policies && !(list = X509V3_parse_list(policies)))
339                 {
340                 TS_CONF_invalid(section, ENV_OTHER_POLICIES);
341                 goto err;
342                 }
343         for (i = 0; i < sk_CONF_VALUE_num(list); ++i)
344                 {
345                 CONF_VALUE *val = sk_CONF_VALUE_value(list, i);
346                 const char *extval = val->value ? val->value : val->name;
347                 ASN1_OBJECT *objtmp;
348                 if (!(objtmp = OBJ_txt2obj(extval, 0)))
349                         {
350                         TS_CONF_invalid(section, ENV_OTHER_POLICIES);
351                         goto err;
352                         }
353                 if (!TS_RESP_CTX_add_policy(ctx, objtmp))
354                         goto err;
355                 ASN1_OBJECT_free(objtmp);
356                 }

358         ret = 1;
359  err:
360         sk_CONF_VALUE_pop_free(list, X509V3_conf_free);
361         return ret;
362         }

364 int TS_CONF_set_digests(CONF *conf, const char *section,
365                         TS_RESP_CTX *ctx)
366         {
367         int ret = 0;
368         int i;
369         STACK_OF(CONF_VALUE) *list = NULL;
370         char *digests = NCONF_get_string(conf, section, ENV_DIGESTS);
371         if (!digests)
372                 {
373                 TS_CONF_lookup_fail(section, ENV_DIGESTS);
374                 goto err;
375                 }
376         if (!(list = X509V3_parse_list(digests)))
377                 {
378                 TS_CONF_invalid(section, ENV_DIGESTS);
379                 goto err;
380                 }
381         if (sk_CONF_VALUE_num(list) == 0)
382                 {
383                 TS_CONF_invalid(section, ENV_DIGESTS);
384                 goto err;
385                 }
386         for (i = 0; i < sk_CONF_VALUE_num(list); ++i)
387                 {
388                 CONF_VALUE *val = sk_CONF_VALUE_value(list, i);
389                 const char *extval = val->value ? val->value : val->name;
390                 const EVP_MD *md;
391                 if (!(md = EVP_get_digestbyname(extval)))
```

```
392                         {
393                         TS_CONF_invalid(section, ENV_DIGESTS);
394                         goto err;
395                         }
396                 if (!TS_RESP_CTX_add_md(ctx, md))
397                         goto err;
398                 }

400         ret = 1;
401  err:
402         sk_CONF_VALUE_pop_free(list, X509V3_conf_free);
403         return ret;
404         }

406 int TS_CONF_set_accuracy(CONF *conf, const char *section, TS_RESP_CTX *ctx)
407         {
408         int ret = 0;
409         int i;
410         int secs = 0, millis = 0, micros = 0;
411         STACK_OF(CONF_VALUE) *list = NULL;
412         char *accuracy = NCONF_get_string(conf, section, ENV_ACCURACY);

414         if (accuracy && !(list = X509V3_parse_list(accuracy)))
415                 {
416                 TS_CONF_invalid(section, ENV_ACCURACY);
417                 goto err;
418                 }
419         for (i = 0; i < sk_CONF_VALUE_num(list); ++i)
420                 {
421                 CONF_VALUE *val = sk_CONF_VALUE_value(list, i);
422                 if (strcmp(val->name, ENV_VALUE_SECS) == 0)
423                         {
424                         if (val->value) secs = atoi(val->value);
425                         }
426                 else if (strcmp(val->name, ENV_VALUE_MILLISECS) == 0)
427                         {
428                         if (val->value) millis = atoi(val->value);
429                         }
430                 else if (strcmp(val->name, ENV_VALUE_MICROSECS) == 0)
431                         {
432                         if (val->value) micros = atoi(val->value);
433                         }
434                 else
435                         {
436                         TS_CONF_invalid(section, ENV_ACCURACY);
437                         goto err;
438                         }
439                 }
440         if (!TS_RESP_CTX_set_accuracy(ctx, secs, millis, micros))
441                 goto err;

443         ret = 1;
444  err:
445         sk_CONF_VALUE_pop_free(list, X509V3_conf_free);
446         return ret;
447         }

449 int TS_CONF_set_clock_precision_digits(CONF *conf, const char *section,
450                                        TS_RESP_CTX *ctx)
451         {
452         int ret = 0;
453         long digits = 0;

455         /* If not specified, set the default value to 0, i.e. sec  precision */
456         if (!NCONF_get_number_e(conf, section, ENV_CLOCK_PRECISION_DIGITS,
457                                 &digits))
```



```
458                 digits = 0;
459         if (digits < 0 || digits > TS_MAX_CLOCK_PRECISION_DIGITS)
460                 {
461                 TS_CONF_invalid(section, ENV_CLOCK_PRECISION_DIGITS);
462                 goto err;
463                 }

465         if (!TS_RESP_CTX_set_clock_precision_digits(ctx, digits))
466                 goto err;

468         return 1;
469  err:
470         return ret;
471         }

473 static int TS_CONF_add_flag(CONF *conf, const char *section, const char *field,
474                             int flag, TS_RESP_CTX *ctx)
475         {
476         /* Default is false. */
477         const char *value = NCONF_get_string(conf, section, field);
478         if (value)
479                 {
480                 if (strcmp(value, ENV_VALUE_YES) == 0)
481                         TS_RESP_CTX_add_flags(ctx, flag);
482                 else if (strcmp(value, ENV_VALUE_NO) != 0)
483                         {
484                         TS_CONF_invalid(section, field);
485                         return 0;
486                         }
487                 }

489         return 1;
490         }

492 int TS_CONF_set_ordering(CONF *conf, const char *section, TS_RESP_CTX *ctx)
493         {
494         return TS_CONF_add_flag(conf, section, ENV_ORDERING, TS_ORDERING, ctx);
495         }

497 int TS_CONF_set_tsa_name(CONF *conf, const char *section, TS_RESP_CTX *ctx)
498         {
499         return TS_CONF_add_flag(conf, section, ENV_TSA_NAME, TS_TSA_NAME, ctx);
500         }

502 int TS_CONF_set_ess_cert_id_chain(CONF *conf, const char *section,
503                                   TS_RESP_CTX *ctx)
504         {
505         return TS_CONF_add_flag(conf, section, ENV_ESS_CERT_ID_CHAIN,
506                                 TS_ESS_CERT_ID_CHAIN, ctx);
507         }
508 #endif /* ! codereview */
```

```
**********************************************************
    9071 Wed Aug 13 19:53:20 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_err.c */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/ts.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_TS,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_TS,0,reason)

static ERR_STRING_DATA TS_str_functs[]=
        {
{ERR_FUNC(TS_F_D2I_TS_RESP),     "d2i_TS_RESP"},
{ERR_FUNC(TS_F_DEF_SERIAL_CB),   "DEF_SERIAL_CB"},
{ERR_FUNC(TS_F_DEF_TIME_CB),     "DEF_TIME_CB"},
{ERR_FUNC(TS_F_ESS_ADD_SIGNING_CERT),    "ESS_ADD_SIGNING_CERT"},
{ERR_FUNC(TS_F_ESS_CERT_ID_NEW_INIT),    "ESS_CERT_ID_NEW_INIT"},
{ERR_FUNC(TS_F_ESS_SIGNING_CERT_NEW_INIT),       "ESS_SIGNING_CERT_NEW_INIT"},
{ERR_FUNC(TS_F_INT_TS_RESP_VERIFY_TOKEN),        "INT_TS_RESP_VERIFY_TOKEN"},
{ERR_FUNC(TS_F_PKCS7_TO_TS_TST_INFO),    "PKCS7_to_TS_TST_INFO"},
{ERR_FUNC(TS_F_TS_ACCURACY_SET_MICROS),  "TS_ACCURACY_set_micros"},
{ERR_FUNC(TS_F_TS_ACCURACY_SET_MILLIS),  "TS_ACCURACY_set_millis"},
{ERR_FUNC(TS_F_TS_ACCURACY_SET_SECONDS),         "TS_ACCURACY_set_seconds"},
{ERR_FUNC(TS_F_TS_CHECK_IMPRINTS),       "TS_CHECK_IMPRINTS"},
{ERR_FUNC(TS_F_TS_CHECK_NONCES),         "TS_CHECK_NONCES"},
{ERR_FUNC(TS_F_TS_CHECK_POLICY),         "TS_CHECK_POLICY"},
{ERR_FUNC(TS_F_TS_CHECK_SIGNING_CERTS),  "TS_CHECK_SIGNING_CERTS"},
{ERR_FUNC(TS_F_TS_CHECK_STATUS_INFO),    "TS_CHECK_STATUS_INFO"},
{ERR_FUNC(TS_F_TS_COMPUTE_IMPRINT),      "TS_COMPUTE_IMPRINT"},
{ERR_FUNC(TS_F_TS_CONF_SET_DEFAULT_ENGINE),      "TS_CONF_set_default_engine"},
{ERR_FUNC(TS_F_TS_GET_STATUS_TEXT),      "TS_GET_STATUS_TEXT"},
{ERR_FUNC(TS_F_TS_MSG_IMPRINT_SET_ALGO),         "TS_MSG_IMPRINT_set_algo"},
{ERR_FUNC(TS_F_TS_REQ_SET_MSG_IMPRINT),  "TS_REQ_set_msg_imprint"},
{ERR_FUNC(TS_F_TS_REQ_SET_NONCE),        "TS_REQ_set_nonce"},
{ERR_FUNC(TS_F_TS_REQ_SET_POLICY_ID),    "TS_REQ_set_policy_id"},
{ERR_FUNC(TS_F_TS_RESP_CREATE_RESPONSE),         "TS_RESP_create_response"},
{ERR_FUNC(TS_F_TS_RESP_CREATE_TST_INFO),         "TS_RESP_CREATE_TST_INFO"},
{ERR_FUNC(TS_F_TS_RESP_CTX_ADD_FAILURE_INFO),    "TS_RESP_CTX_add_failure_info"},
{ERR_FUNC(TS_F_TS_RESP_CTX_ADD_MD),      "TS_RESP_CTX_add_md"},
{ERR_FUNC(TS_F_TS_RESP_CTX_ADD_POLICY),  "TS_RESP_CTX_add_policy"},
{ERR_FUNC(TS_F_TS_RESP_CTX_NEW),         "TS_RESP_CTX_new"},
{ERR_FUNC(TS_F_TS_RESP_CTX_SET_ACCURACY),        "TS_RESP_CTX_set_accuracy"},
{ERR_FUNC(TS_F_TS_RESP_CTX_SET_CERTS),   "TS_RESP_CTX_set_certs"},
{ERR_FUNC(TS_F_TS_RESP_CTX_SET_DEF_POLICY),      "TS_RESP_CTX_set_def_policy"},
{ERR_FUNC(TS_F_TS_RESP_CTX_SET_SIGNER_CERT),     "TS_RESP_CTX_set_signer_cert"},
{ERR_FUNC(TS_F_TS_RESP_CTX_SET_STATUS_INFO),     "TS_RESP_CTX_set_status_info"},
{ERR_FUNC(TS_F_TS_RESP_GET_POLICY),      "TS_RESP_GET_POLICY"},
{ERR_FUNC(TS_F_TS_RESP_SET_GENTIME_WITH_PRECISION),      "TS_RESP_SET_GENTIME_WIT
{ERR_FUNC(TS_F_TS_RESP_SET_STATUS_INFO),         "TS_RESP_set_status_info"},
{ERR_FUNC(TS_F_TS_RESP_SET_TST_INFO),    "TS_RESP_set_tst_info"},
{ERR_FUNC(TS_F_TS_RESP_SIGN),    "TS_RESP_SIGN"},
{ERR_FUNC(TS_F_TS_RESP_VERIFY_SIGNATURE),        "TS_RESP_verify_signature"},
{ERR_FUNC(TS_F_TS_RESP_VERIFY_TOKEN),    "TS_RESP_verify_token"},
{ERR_FUNC(TS_F_TS_TST_INFO_SET_ACCURACY),        "TS_TST_INFO_set_accuracy"},
{ERR_FUNC(TS_F_TS_TST_INFO_SET_MSG_IMPRINT),     "TS_TST_INFO_set_msg_imprint"},
{ERR_FUNC(TS_F_TS_TST_INFO_SET_NONCE),   "TS_TST_INFO_set_nonce"},
{ERR_FUNC(TS_F_TS_TST_INFO_SET_POLICY_ID),       "TS_TST_INFO_set_policy_id"},
{ERR_FUNC(TS_F_TS_TST_INFO_SET_SERIAL),  "TS_TST_INFO_set_serial"},
{ERR_FUNC(TS_F_TS_TST_INFO_SET_TIME),    "TS_TST_INFO_set_time"},
{ERR_FUNC(TS_F_TS_TST_INFO_SET_TSA),     "TS_TST_INFO_set_tsa"},
{ERR_FUNC(TS_F_TS_VERIFY),       "TS_VERIFY"},
{ERR_FUNC(TS_F_TS_VERIFY_CERT),  "TS_VERIFY_CERT"},
{ERR_FUNC(TS_F_TS_VERIFY_CTX_NEW),       "TS_VERIFY_CTX_new"},
{0,NULL}
        };

static ERR_STRING_DATA TS_str_reasons[]=
```

```
128         {
129 {ERR_REASON(TS_R_BAD_PKCS7_TYPE)         ,"bad pkcs7 type"},
130 {ERR_REASON(TS_R_BAD_TYPE)               ,"bad type"},
131 {ERR_REASON(TS_R_CERTIFICATE_VERIFY_ERROR),"certificate verify error"},
132 {ERR_REASON(TS_R_COULD_NOT_SET_ENGINE)   ,"could not set engine"},
133 {ERR_REASON(TS_R_COULD_NOT_SET_TIME)     ,"could not set time"},
134 {ERR_REASON(TS_R_D2I_TS_RESP_INT_FAILED) ,"d2i ts resp int failed"},
135 {ERR_REASON(TS_R_DETACHED_CONTENT)       ,"detached content"},
136 {ERR_REASON(TS_R_ESS_ADD_SIGNING_CERT_ERROR),"ess add signing cert error"},
137 {ERR_REASON(TS_R_ESS_SIGNING_CERTIFICATE_ERROR),"ess signing certificate error"}
138 {ERR_REASON(TS_R_INVALID_NULL_POINTER)   ,"invalid null pointer"},
139 {ERR_REASON(TS_R_INVALID_SIGNER_CERTIFICATE_PURPOSE),"invalid signer certificate
140 {ERR_REASON(TS_R_MESSAGE_IMPRINT_MISMATCH),"message imprint mismatch"},
141 {ERR_REASON(TS_R_NONCE_MISMATCH)         ,"nonce mismatch"},
142 {ERR_REASON(TS_R_NONCE_NOT_RETURNED)     ,"nonce not returned"},
143 {ERR_REASON(TS_R_NO_CONTENT)             ,"no content"},
144 {ERR_REASON(TS_R_NO_TIME_STAMP_TOKEN)    ,"no time stamp token"},
145 {ERR_REASON(TS_R_PKCS7_ADD_SIGNATURE_ERROR),"pkcs7 add signature error"},
146 {ERR_REASON(TS_R_PKCS7_ADD_SIGNED_ATTR_ERROR),"pkcs7 add signed attr error"},
147 {ERR_REASON(TS_R_PKCS7_TO_TS_TST_INFO_FAILED),"pkcs7 to ts tst info failed"},
148 {ERR_REASON(TS_R_POLICY_MISMATCH)        ,"policy mismatch"},
149 {ERR_REASON(TS_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE),"private key does not m
150 {ERR_REASON(TS_R_RESPONSE_SETUP_ERROR)   ,"response setup error"},
151 {ERR_REASON(TS_R_SIGNATURE_FAILURE)      ,"signature failure"},
152 {ERR_REASON(TS_R_THERE_MUST_BE_ONE_SIGNER),"there must be one signer"},
153 {ERR_REASON(TS_R_TIME_SYSCALL_ERROR)     ,"time syscall error"},
154 {ERR_REASON(TS_R_TOKEN_NOT_PRESENT)      ,"token not present"},
155 {ERR_REASON(TS_R_TOKEN_PRESENT)          ,"token present"},
156 {ERR_REASON(TS_R_TSA_NAME_MISMATCH)      ,"tsa name mismatch"},
157 {ERR_REASON(TS_R_TSA_UNTRUSTED)          ,"tsa untrusted"},
158 {ERR_REASON(TS_R_TST_INFO_SETUP_ERROR)   ,"tst info setup error"},
159 {ERR_REASON(TS_R_TS_DATASIGN)            ,"ts datasign"},
160 {ERR_REASON(TS_R_UNACCEPTABLE_POLICY)    ,"unacceptable policy"},
161 {ERR_REASON(TS_R_UNSUPPORTED_MD_ALGORITHM),"unsupported md algorithm"},
162 {ERR_REASON(TS_R_UNSUPPORTED_VERSION)    ,"unsupported version"},
163 {ERR_REASON(TS_R_WRONG_CONTENT_TYPE)     ,"wrong content type"},
164 {0,NULL}
165         };

167 #endif

169 void ERR_load_TS_strings(void)
170         {
171 #ifndef OPENSSL_NO_ERR

173         if (ERR_func_error_string(TS_str_functs[0].error) == NULL)
174                 {
175                 ERR_load_strings(0,TS_str_functs);
176                 ERR_load_strings(0,TS_str_reasons);
177                 }
178 #endif
179         }
180 #endif /* ! codereview */
```

```
**********************************************************
    4691 Wed Aug 13 19:53:20 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_lib.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/bn.h>
#include <openssl/x509v3.h>
#include <openssl/ts.h>

/* Local function declarations. */

/* Function definitions. */

int TS_ASN1_INTEGER_print_bio(BIO *bio, const ASN1_INTEGER *num)
        {
        BIGNUM num_bn;
        int result = 0;
        char *hex;

        BN_init(&num_bn);
        ASN1_INTEGER_to_BN(num, &num_bn);
        if ((hex = BN_bn2hex(&num_bn)))
                {
                result = BIO_write(bio, "0x", 2) > 0;
                result = result && BIO_write(bio, hex, strlen(hex)) > 0;
                OPENSSL_free(hex);
                }
        BN_free(&num_bn);

        return result;
        }

int TS_OBJ_print_bio(BIO *bio, const ASN1_OBJECT *obj)
        {
        char obj_txt[128];

        int len = OBJ_obj2txt(obj_txt, sizeof(obj_txt), obj, 0);
        BIO_write(bio, obj_txt, len);
        BIO_write(bio, "\n", 1);

        return 1;
        }

int TS_ext_print_bio(BIO *bio, const STACK_OF(X509_EXTENSION) *extensions)
        {
        int i, critical, n;
        X509_EXTENSION *ex;
        ASN1_OBJECT *obj;

        BIO_printf(bio, "Extensions:\n");
        n = X509v3_get_ext_count(extensions);
        for (i = 0; i < n; i++)
                {
                ex = X509v3_get_ext(extensions, i);
                obj = X509_EXTENSION_get_object(ex);
                i2a_ASN1_OBJECT(bio, obj);
                critical = X509_EXTENSION_get_critical(ex);
                BIO_printf(bio, ": %s\n", critical ? "critical" : "");
                if (!X509V3_EXT_print(bio, ex, 0, 4))
                        {
                        BIO_printf(bio, "%4s", "");
                        M_ASN1_OCTET_STRING_print(bio, ex->value);
                        }
                BIO_write(bio, "\n", 1);
                }

        return 1;
        }

int TS_X509_ALGOR_print_bio(BIO *bio, const X509_ALGOR *alg)
        {
```

```
128         int i = OBJ_obj2nid(alg->algorithm);
129         return BIO_printf(bio, "Hash Algorithm: %s\n",
130                 (i == NID_undef) ? "UNKNOWN" : OBJ_nid2ln(i));
131         }

133 int TS_MSG_IMPRINT_print_bio(BIO *bio, TS_MSG_IMPRINT *a)
134         {
135         const ASN1_OCTET_STRING *msg;

137         TS_X509_ALGOR_print_bio(bio, TS_MSG_IMPRINT_get_algo(a));

139         BIO_printf(bio, "Message data:\n");
140         msg = TS_MSG_IMPRINT_get_msg(a);
141         BIO_dump_indent(bio, (const char *)M_ASN1_STRING_data(msg),
142                         M_ASN1_STRING_length(msg), 4);

144         return 1;
145         }
146 #endif /* ! codereview */
```

```
**********************************************************
    3694 Wed Aug 13 19:53:20 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_req_print.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_req_print.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/bn.h>
#include <openssl/x509v3.h>
#include <openssl/ts.h>

/* Function definitions. */

int TS_REQ_print_bio(BIO *bio, TS_REQ *a)
        {
        int v;
        ASN1_OBJECT *policy_id;
        const ASN1_INTEGER *nonce;

        if (a == NULL) return 0;

        v = TS_REQ_get_version(a);
        BIO_printf(bio, "Version: %d\n", v);

        TS_MSG_IMPRINT_print_bio(bio, TS_REQ_get_msg_imprint(a));

        BIO_printf(bio, "Policy OID: ");
        policy_id = TS_REQ_get_policy_id(a);
        if (policy_id == NULL)
                BIO_printf(bio, "unspecified\n");
        else
                TS_OBJ_print_bio(bio, policy_id);

        BIO_printf(bio, "Nonce: ");
        nonce = TS_REQ_get_nonce(a);
        if (nonce == NULL)
                BIO_printf(bio, "unspecified");
        else
                TS_ASN1_INTEGER_print_bio(bio, nonce);
        BIO_write(bio, "\n", 1);

        BIO_printf(bio, "Certificate required: %s\n",
                   TS_REQ_get_cert_req(a) ? "yes" : "no");

        TS_ext_print_bio(bio, TS_REQ_get_exts(a));

        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    6355 Wed Aug 13 19:53:21 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_req_utils.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_req_utils.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/x509v3.h>
#include <openssl/ts.h>

int TS_REQ_set_version(TS_REQ *a, long version)
        {
        return ASN1_INTEGER_set(a->version, version);
        }

long TS_REQ_get_version(const TS_REQ *a)
        {
        return ASN1_INTEGER_get(a->version);
        }

int TS_REQ_set_msg_imprint(TS_REQ *a, TS_MSG_IMPRINT *msg_imprint)
        {
        TS_MSG_IMPRINT *new_msg_imprint;

        if (a->msg_imprint == msg_imprint)
                return 1;
        new_msg_imprint = TS_MSG_IMPRINT_dup(msg_imprint);
        if (new_msg_imprint == NULL)
                {
                TSerr(TS_F_TS_REQ_SET_MSG_IMPRINT, ERR_R_MALLOC_FAILURE);
                return 0;
                }
        TS_MSG_IMPRINT_free(a->msg_imprint);
        a->msg_imprint = new_msg_imprint;
        return 1;
        }

TS_MSG_IMPRINT *TS_REQ_get_msg_imprint(TS_REQ *a)
        {
        return a->msg_imprint;
        }

int TS_MSG_IMPRINT_set_algo(TS_MSG_IMPRINT *a, X509_ALGOR *alg)
        {
        X509_ALGOR *new_alg;

        if (a->hash_algo == alg)
                return 1;
        new_alg = X509_ALGOR_dup(alg);
        if (new_alg == NULL)
                {
                TSerr(TS_F_TS_MSG_IMPRINT_SET_ALGO, ERR_R_MALLOC_FAILURE);
                return 0;
                }
        X509_ALGOR_free(a->hash_algo);
        a->hash_algo = new_alg;
        return 1;
        }

X509_ALGOR *TS_MSG_IMPRINT_get_algo(TS_MSG_IMPRINT *a)
        {
        return a->hash_algo;
        }

int TS_MSG_IMPRINT_set_msg(TS_MSG_IMPRINT *a, unsigned char *d, int len)
        {
        return ASN1_OCTET_STRING_set(a->hashed_msg, d, len);
        }

ASN1_OCTET_STRING *TS_MSG_IMPRINT_get_msg(TS_MSG_IMPRINT *a)
        {
        return a->hashed_msg;
        }
```

```
129 int TS_REQ_set_policy_id(TS_REQ *a, ASN1_OBJECT *policy)
130         {
131         ASN1_OBJECT *new_policy;
132
133         if (a->policy_id == policy)
134                 return 1;
135         new_policy = OBJ_dup(policy);
136         if (new_policy == NULL)
137                 {
138                 TSerr(TS_F_TS_REQ_SET_POLICY_ID, ERR_R_MALLOC_FAILURE);
139                 return 0;
140                 }
141         ASN1_OBJECT_free(a->policy_id);
142         a->policy_id = new_policy;
143         return 1;
144         }
145
146 ASN1_OBJECT *TS_REQ_get_policy_id(TS_REQ *a)
147         {
148         return a->policy_id;
149         }
150
151 int TS_REQ_set_nonce(TS_REQ *a, const ASN1_INTEGER *nonce)
152         {
153         ASN1_INTEGER *new_nonce;
154
155         if (a->nonce == nonce)
156                 return 1;
157         new_nonce = ASN1_INTEGER_dup(nonce);
158         if (new_nonce == NULL)
159                 {
160                 TSerr(TS_F_TS_REQ_SET_NONCE, ERR_R_MALLOC_FAILURE);
161                 return 0;
162                 }
163         ASN1_INTEGER_free(a->nonce);
164         a->nonce = new_nonce;
165         return 1;
166         }
167
168 const ASN1_INTEGER *TS_REQ_get_nonce(const TS_REQ *a)
169         {
170         return a->nonce;
171         }
172
173 int TS_REQ_set_cert_req(TS_REQ *a, int cert_req)
174         {
175         a->cert_req = cert_req ? 0xFF : 0x00;
176         return 1;
177         }
178
179 int TS_REQ_get_cert_req(const TS_REQ *a)
180         {
181         return a->cert_req ? 1 : 0;
182         }
183
184 STACK_OF(X509_EXTENSION) *TS_REQ_get_exts(TS_REQ *a)
185         {
186         return a->extensions;
187         }
188
189 void TS_REQ_ext_free(TS_REQ *a)
190         {
191         if (!a) return;
192         sk_X509_EXTENSION_pop_free(a->extensions, X509_EXTENSION_free);
193         a->extensions = NULL;
```



```
194         }
195
196 int TS_REQ_get_ext_count(TS_REQ *a)
197         {
198         return X509v3_get_ext_count(a->extensions);
199         }
200
201 int TS_REQ_get_ext_by_NID(TS_REQ *a, int nid, int lastpos)
202         {
203         return X509v3_get_ext_by_NID(a->extensions, nid, lastpos);
204         }
205
206 int TS_REQ_get_ext_by_OBJ(TS_REQ *a, ASN1_OBJECT *obj, int lastpos)
207         {
208         return X509v3_get_ext_by_OBJ(a->extensions, obj, lastpos);
209         }
210
211 int TS_REQ_get_ext_by_critical(TS_REQ *a, int crit, int lastpos)
212         {
213         return X509v3_get_ext_by_critical(a->extensions, crit, lastpos);
214         }
215
216 X509_EXTENSION *TS_REQ_get_ext(TS_REQ *a, int loc)
217         {
218         return X509v3_get_ext(a->extensions,loc);
219         }
220
221 X509_EXTENSION *TS_REQ_delete_ext(TS_REQ *a, int loc)
222         {
223         return X509v3_delete_ext(a->extensions,loc);
224         }
225
226 int TS_REQ_add_ext(TS_REQ *a, X509_EXTENSION *ex, int loc)
227         {
228         return X509v3_add_ext(&a->extensions,ex,loc) != NULL;
229         }
230
231 void *TS_REQ_get_ext_d2i(TS_REQ *a, int nid, int *crit, int *idx)
232         {
233         return X509V3_get_d2i(a->extensions, nid, crit, idx);
234         }
235 #endif /* ! codereview */
```

```
**********************************************************
    8421 Wed Aug 13 19:53:21 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_rsp_print.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_resp_print.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/bn.h>
#include <openssl/x509v3.h>
#include <openssl/ts.h>

struct status_map_st
        {
        int bit;
        const char *text;
        };

/* Local function declarations. */

static int TS_status_map_print(BIO *bio, struct status_map_st *a,
                               ASN1_BIT_STRING *v);
static int TS_ACCURACY_print_bio(BIO *bio, const TS_ACCURACY *accuracy);

/* Function definitions. */

int TS_RESP_print_bio(BIO *bio, TS_RESP *a)
        {
        TS_TST_INFO *tst_info;

        BIO_printf(bio, "Status info:\n");
        TS_STATUS_INFO_print_bio(bio, TS_RESP_get_status_info(a));

        BIO_printf(bio, "\nTST info:\n");
        tst_info = TS_RESP_get_tst_info(a);
        if (tst_info != NULL)
                TS_TST_INFO_print_bio(bio, TS_RESP_get_tst_info(a));
        else
                BIO_printf(bio, "Not included.\n");

        return 1;
        }

int TS_STATUS_INFO_print_bio(BIO *bio, TS_STATUS_INFO *a)
        {
        static const char *status_map[] =
                {
                "Granted.",
                "Granted with modifications.",
                "Rejected.",
                "Waiting.",
                "Revocation warning.",
                "Revoked."
                };
        static struct status_map_st failure_map[] =
                {
                { TS_INFO_BAD_ALG,
                "unrecognized or unsupported algorithm identifier" },
                { TS_INFO_BAD_REQUEST,
                "transaction not permitted or supported" },
                { TS_INFO_BAD_DATA_FORMAT,
                "the data submitted has the wrong format" },
                { TS_INFO_TIME_NOT_AVAILABLE,
                "the TSA's time source is not available" },
                { TS_INFO_UNACCEPTED_POLICY,
                "the requested TSA policy is not supported by the TSA" },
                { TS_INFO_UNACCEPTED_EXTENSION,
                "the requested extension is not supported by the TSA" },
                { TS_INFO_ADD_INFO_NOT_AVAILABLE,
                "the additional information requested could not be understood "
                "or is not available" },
                { TS_INFO_SYSTEM_FAILURE,
                "the request cannot be handled due to system failure" },
                { -1, NULL }
```

```
128                 };
129         long status;
130         int i, lines = 0;

132         /* Printing status code. */
133         BIO_printf(bio, "Status: ");
134         status = ASN1_INTEGER_get(a->status);
135         if (0 <= status && status < (long)(sizeof(status_map)/sizeof(status_map[
136                 BIO_printf(bio, "%s\n", status_map[status]);
137         else
138                 BIO_printf(bio, "out of bounds\n");

140         /* Printing status description. */
141         BIO_printf(bio, "Status description: ");
142         for (i = 0; i < sk_ASN1_UTF8STRING_num(a->text); ++i)
143                 {
144                 if (i > 0)
145                         BIO_puts(bio, "\t");
146                 ASN1_STRING_print_ex(bio, sk_ASN1_UTF8STRING_value(a->text, i),
147                                      0);
148                 BIO_puts(bio, "\n");
149                 }
150         if (i == 0)
151                 BIO_printf(bio, "unspecified\n");

153         /* Printing failure information. */
154         BIO_printf(bio, "Failure info: ");
155         if (a->failure_info != NULL)
156                 lines = TS_status_map_print(bio, failure_map,
157                                             a->failure_info);
158         if (lines == 0)
159                 BIO_printf(bio, "unspecified");
160         BIO_printf(bio, "\n");

162         return 1;
163         }

165 static int TS_status_map_print(BIO *bio, struct status_map_st *a,
166                                ASN1_BIT_STRING *v)
167         {
168         int lines = 0;

170         for (; a->bit >= 0; ++a)
171                 {
172                 if (ASN1_BIT_STRING_get_bit(v, a->bit))
173                         {
174                         if (++lines > 1)
175                                 BIO_printf(bio, ", ");
176                         BIO_printf(bio, "%s", a->text);
177                         }
178                 }

180         return lines;
181         }

183 int TS_TST_INFO_print_bio(BIO *bio, TS_TST_INFO *a)
184         {
185         int v;
186         ASN1_OBJECT *policy_id;
187         const ASN1_INTEGER *serial;
188         const ASN1_GENERALIZEDTIME *gtime;
189         TS_ACCURACY *accuracy;
190         const ASN1_INTEGER *nonce;
191         GENERAL_NAME *tsa_name;

193         if (a == NULL) return 0;
```



```
195         /* Print version. */
196         v = TS_TST_INFO_get_version(a);
197         BIO_printf(bio, "Version: %d\n", v);

199         /* Print policy id. */
200         BIO_printf(bio, "Policy OID: ");
201         policy_id = TS_TST_INFO_get_policy_id(a);
202         TS_OBJ_print_bio(bio, policy_id);

204         /* Print message imprint. */
205         TS_MSG_IMPRINT_print_bio(bio, TS_TST_INFO_get_msg_imprint(a));

207         /* Print serial number. */
208         BIO_printf(bio, "Serial number: ");
209         serial = TS_TST_INFO_get_serial(a);
210         if (serial == NULL)
211                 BIO_printf(bio, "unspecified");
212         else
213                 TS_ASN1_INTEGER_print_bio(bio, serial);
214         BIO_write(bio, "\n", 1);

216         /* Print time stamp. */
217         BIO_printf(bio, "Time stamp: ");
218         gtime = TS_TST_INFO_get_time(a);
219         ASN1_GENERALIZEDTIME_print(bio, gtime);
220         BIO_write(bio, "\n", 1);

222         /* Print accuracy. */
223         BIO_printf(bio, "Accuracy: ");
224         accuracy = TS_TST_INFO_get_accuracy(a);
225         if (accuracy == NULL)
226                 BIO_printf(bio, "unspecified");
227         else
228                 TS_ACCURACY_print_bio(bio, accuracy);
229         BIO_write(bio, "\n", 1);

231         /* Print ordering. */
232         BIO_printf(bio, "Ordering: %s\n",
233                    TS_TST_INFO_get_ordering(a) ? "yes" : "no");

235         /* Print nonce. */
236         BIO_printf(bio, "Nonce: ");
237         nonce = TS_TST_INFO_get_nonce(a);
238         if (nonce == NULL)
239                 BIO_printf(bio, "unspecified");
240         else
241                 TS_ASN1_INTEGER_print_bio(bio, nonce);
242         BIO_write(bio, "\n", 1);

244         /* Print TSA name. */
245         BIO_printf(bio, "TSA: ");
246         tsa_name = TS_TST_INFO_get_tsa(a);
247         if (tsa_name == NULL)
248                 BIO_printf(bio, "unspecified");
249         else
250                 {
251                 STACK_OF(CONF_VALUE) *nval;
252                 if ((nval = i2v_GENERAL_NAME(NULL, tsa_name, NULL)))
253                         X509V3_EXT_val_prn(bio, nval, 0, 0);
254                 sk_CONF_VALUE_pop_free(nval, X509V3_conf_free);
255                 }
256         BIO_write(bio, "\n", 1);

258         /* Print extensions. */
259         TS_ext_print_bio(bio, TS_TST_INFO_get_exts(a));
```

```
261         return 1;
262         }

264 static int TS_ACCURACY_print_bio(BIO *bio, const TS_ACCURACY *accuracy)
265         {
266         const ASN1_INTEGER *seconds = TS_ACCURACY_get_seconds(accuracy);
267         const ASN1_INTEGER *millis = TS_ACCURACY_get_millis(accuracy);
268         const ASN1_INTEGER *micros = TS_ACCURACY_get_micros(accuracy);

270         if (seconds != NULL)
271                 TS_ASN1_INTEGER_print_bio(bio, seconds);
272         else
273                 BIO_printf(bio, "unspecified");
274         BIO_printf(bio, " seconds, ");
275         if (millis != NULL)
276                 TS_ASN1_INTEGER_print_bio(bio, millis);
277         else
278                 BIO_printf(bio, "unspecified");
279         BIO_printf(bio, " millis, ");
280         if (micros != NULL)
281                 TS_ASN1_INTEGER_print_bio(bio, micros);
282         else
283                 BIO_printf(bio, "unspecified");
284         BIO_printf(bio, " micros");

286         return 1;
287         }
288 #endif /* ! codereview */
```

```
**********************************************************
   29331 Wed Aug 13 19:53:21 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_rsp_sign.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 /* crypto/ts/ts_resp_sign.c */
   2 /* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
   3  * project 2002.
   4  */
   5 /* ====================================================================
   6  * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
   7  *
   8  * Redistribution and use in source and binary forms, with or without
   9  * modification, are permitted provided that the following conditions
  10  * are met:
  11  *
  12  * 1. Redistributions of source code must retain the above copyright
  13  *    notice, this list of conditions and the following disclaimer.
  14  *
  15  * 2. Redistributions in binary form must reproduce the above copyright
  16  *    notice, this list of conditions and the following disclaimer in
  17  *    the documentation and/or other materials provided with the
  18  *    distribution.
  19  *
  20  * 3. All advertising materials mentioning features or use of this
  21  *    software must display the following acknowledgment:
  22  *    "This product includes software developed by the OpenSSL Project
  23  *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
  24  *
  25  * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
  26  *    endorse or promote products derived from this software without
  27  *    prior written permission. For written permission, please contact
  28  *    licensing@OpenSSL.org.
  29  *
  30  * 5. Products derived from this software may not be called "OpenSSL"
  31  *    nor may "OpenSSL" appear in their names without prior written
  32  *    permission of the OpenSSL Project.
  33  *
  34  * 6. Redistributions of any form whatsoever must retain the following
  35  *    acknowledgment:
  36  *    "This product includes software developed by the OpenSSL Project
  37  *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
  38  *
  39  * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
  40  * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  41  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
  42  * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
  43  * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
  44  * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
  45  * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
  46  * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
  47  * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
  48  * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
  49  * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
  50  * OF THE POSSIBILITY OF SUCH DAMAGE.
  51  * ====================================================================
  52  *
  53  * This product includes cryptographic software written by Eric Young
  54  * (eay@cryptsoft.com).  This product includes software written by Tim
  55  * Hudson (tjh@cryptsoft.com).
  56  *
  57  */

  59 #include "cryptlib.h"

  61 #if defined(OPENSSL_SYS_UNIX)
```



```
  62 #include <sys/time.h>
  63 #endif

  65 #include <openssl/objects.h>
  66 #include <openssl/ts.h>
  67 #include <openssl/pkcs7.h>

  69 /* Private function declarations. */

  71 static ASN1_INTEGER *def_serial_cb(struct TS_resp_ctx *, void *);
  72 static int def_time_cb(struct TS_resp_ctx *, void *, long *sec, long *usec);
  73 static int def_extension_cb(struct TS_resp_ctx *, X509_EXTENSION *, void *);

  75 static void TS_RESP_CTX_init(TS_RESP_CTX *ctx);
  76 static void TS_RESP_CTX_cleanup(TS_RESP_CTX *ctx);
  77 static int TS_RESP_check_request(TS_RESP_CTX *ctx);
  78 static ASN1_OBJECT *TS_RESP_get_policy(TS_RESP_CTX *ctx);
  79 static TS_TST_INFO *TS_RESP_create_tst_info(TS_RESP_CTX *ctx,
  80                                             ASN1_OBJECT *policy);
  81 static int TS_RESP_process_extensions(TS_RESP_CTX *ctx);
  82 static int TS_RESP_sign(TS_RESP_CTX *ctx);

  84 static ESS_SIGNING_CERT *ESS_SIGNING_CERT_new_init(X509 *signcert,
  85                                                    STACK_OF(X509) *certs);
  86 static ESS_CERT_ID *ESS_CERT_ID_new_init(X509 *cert, int issuer_needed);
  87 static int TS_TST_INFO_content_new(PKCS7 *p7);
  88 static int ESS_add_signing_cert(PKCS7_SIGNER_INFO *si, ESS_SIGNING_CERT *sc);

  90 static ASN1_GENERALIZEDTIME *TS_RESP_set_genTime_with_precision(
  91         ASN1_GENERALIZEDTIME *, long, long, unsigned);

  93 /* Default callbacks for response generation. */

  95 static ASN1_INTEGER *def_serial_cb(struct TS_resp_ctx *ctx, void *data)
  96         {
  97         ASN1_INTEGER *serial = ASN1_INTEGER_new();
  98         if (!serial) goto err;
  99         if (!ASN1_INTEGER_set(serial, 1)) goto err;
 100         return serial;
 101  err:
 102         TSerr(TS_F_DEF_SERIAL_CB, ERR_R_MALLOC_FAILURE);
 103         TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
 104                                     "Error during serial number generation.");
 105         return NULL;
 106         }

 108 #if defined(OPENSSL_SYS_UNIX)

 110 /* Use the gettimeofday function call. */
 111 static int def_time_cb(struct TS_resp_ctx *ctx, void *data,
 112                        long *sec, long *usec)
 113         {
 114         struct timeval tv;
 115         if (gettimeofday(&tv, NULL) != 0)
 116                 {
 117                 TSerr(TS_F_DEF_TIME_CB, TS_R_TIME_SYSCALL_ERROR);
 118                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
 119                                             "Time is not available.");
 120                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_TIME_NOT_AVAILABLE);
 121                 return 0;
 122                 }
 123         /* Return time to caller. */
 124         *sec = tv.tv_sec;
 125         *usec = tv.tv_usec;

 127         return 1;
```

```
128         }

130 #else

132 /* Use the time function call that provides only seconds precision. */
133 static int def_time_cb(struct TS_resp_ctx *ctx, void *data,
134                        long *sec, long *usec)
135         {
136         time_t t;
137         if (time(&t) == (time_t) -1)
138                 {
139                 TSerr(TS_F_DEF_TIME_CB, TS_R_TIME_SYSCALL_ERROR);
140                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
141                                             "Time is not available.");
142                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_TIME_NOT_AVAILABLE);
143                 return 0;
144                 }
145         /* Return time to caller, only second precision. */
146         *sec = (long) t;
147         *usec = 0;

149         return 1;
150         }

152 #endif

154 static int def_extension_cb(struct TS_resp_ctx *ctx, X509_EXTENSION *ext,
155                             void *data)
156         {
157         /* No extensions are processed here. */
158         TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
159                                     "Unsupported extension.");
160         TS_RESP_CTX_add_failure_info(ctx, TS_INFO_UNACCEPTED_EXTENSION);
161         return 0;
162         }

164 /* TS_RESP_CTX management functions. */

166 TS_RESP_CTX *TS_RESP_CTX_new()
167         {
168         TS_RESP_CTX *ctx;

170         if (!(ctx = (TS_RESP_CTX *) OPENSSL_malloc(sizeof(TS_RESP_CTX))))
171                 {
172                 TSerr(TS_F_TS_RESP_CTX_NEW, ERR_R_MALLOC_FAILURE);
173                 return NULL;
174                 }
175         memset(ctx, 0, sizeof(TS_RESP_CTX));

177         /* Setting default callbacks. */
178         ctx->serial_cb = def_serial_cb;
179         ctx->time_cb = def_time_cb;
180         ctx->extension_cb = def_extension_cb;

182         return ctx;
183         }

185 void TS_RESP_CTX_free(TS_RESP_CTX *ctx)
186         {
187         if (!ctx) return;

189         X509_free(ctx->signer_cert);
190         EVP_PKEY_free(ctx->signer_key);
191         sk_X509_pop_free(ctx->certs, X509_free);
192         sk_ASN1_OBJECT_pop_free(ctx->policies, ASN1_OBJECT_free);
193         ASN1_OBJECT_free(ctx->default_policy);
```



```
194         sk_EVP_MD_free(ctx->mds);       /* No EVP_MD_free method exists. */
195         ASN1_INTEGER_free(ctx->seconds);
196         ASN1_INTEGER_free(ctx->millis);
197         ASN1_INTEGER_free(ctx->micros);
198         OPENSSL_free(ctx);
199         }

201 int TS_RESP_CTX_set_signer_cert(TS_RESP_CTX *ctx, X509 *signer)
202         {
203         if (X509_check_purpose(signer, X509_PURPOSE_TIMESTAMP_SIGN, 0) != 1)
204                 {
205                 TSerr(TS_F_TS_RESP_CTX_SET_SIGNER_CERT,
206                       TS_R_INVALID_SIGNER_CERTIFICATE_PURPOSE);
207                 return 0;
208                 }
209         if (ctx->signer_cert) X509_free(ctx->signer_cert);
210         ctx->signer_cert = signer;
211         CRYPTO_add(&ctx->signer_cert->references, +1, CRYPTO_LOCK_X509);
212         return 1;
213         }

215 int TS_RESP_CTX_set_signer_key(TS_RESP_CTX *ctx, EVP_PKEY *key)
216         {
217         if (ctx->signer_key) EVP_PKEY_free(ctx->signer_key);
218         ctx->signer_key = key;
219         CRYPTO_add(&ctx->signer_key->references, +1, CRYPTO_LOCK_EVP_PKEY);

221         return 1;
222         }

224 int TS_RESP_CTX_set_def_policy(TS_RESP_CTX *ctx, ASN1_OBJECT *def_policy)
225         {
226         if (ctx->default_policy) ASN1_OBJECT_free(ctx->default_policy);
227         if (!(ctx->default_policy = OBJ_dup(def_policy))) goto err;
228         return 1;
229  err:
230         TSerr(TS_F_TS_RESP_CTX_SET_DEF_POLICY, ERR_R_MALLOC_FAILURE);
231         return 0;
232         }

234 int TS_RESP_CTX_set_certs(TS_RESP_CTX *ctx, STACK_OF(X509) *certs)
235         {
236         int i;

238         if (ctx->certs)
239                 {
240                 sk_X509_pop_free(ctx->certs, X509_free);
241                 ctx->certs = NULL;
242                 }
243         if (!certs) return 1;
244         if (!(ctx->certs = sk_X509_dup(certs)))
245                 {
246                 TSerr(TS_F_TS_RESP_CTX_SET_CERTS, ERR_R_MALLOC_FAILURE);
247                 return 0;
248                 }
249         for (i = 0; i < sk_X509_num(ctx->certs); ++i)
250                 {
251                 X509 *cert = sk_X509_value(ctx->certs, i);
252                 CRYPTO_add(&cert->references, +1, CRYPTO_LOCK_X509);
253                 }

255         return 1;
256         }

258 int TS_RESP_CTX_add_policy(TS_RESP_CTX *ctx, ASN1_OBJECT *policy)
259         {
```

```
260         ASN1_OBJECT *copy = NULL;

262         /* Create new policy stack if necessary. */
263         if (!ctx->policies && !(ctx->policies = sk_ASN1_OBJECT_new_null()))
264                 goto err;
265         if (!(copy = OBJ_dup(policy))) goto err;
266         if (!sk_ASN1_OBJECT_push(ctx->policies, copy)) goto err;

268         return 1;
269  err:
270         TSerr(TS_F_TS_RESP_CTX_ADD_POLICY, ERR_R_MALLOC_FAILURE);
271         ASN1_OBJECT_free(copy);
272         return 0;
273         }

275 int TS_RESP_CTX_add_md(TS_RESP_CTX *ctx, const EVP_MD *md)
276         {
277         /* Create new md stack if necessary. */
278         if (!ctx->mds && !(ctx->mds = sk_EVP_MD_new_null()))
279                 goto err;
280         /* Add the shared md, no copy needed. */
281         if (!sk_EVP_MD_push(ctx->mds, (EVP_MD *)md)) goto err;

283         return 1;
284  err:
285         TSerr(TS_F_TS_RESP_CTX_ADD_MD, ERR_R_MALLOC_FAILURE);
286         return 0;
287         }

289 #define TS_RESP_CTX_accuracy_free(ctx)          \
290         ASN1_INTEGER_free(ctx->seconds);        \
291         ctx->seconds = NULL;                    \
292         ASN1_INTEGER_free(ctx->millis);         \
293         ctx->millis = NULL;                     \
294         ASN1_INTEGER_free(ctx->micros);         \
295         ctx->micros = NULL;

297 int TS_RESP_CTX_set_accuracy(TS_RESP_CTX *ctx,
298                              int secs, int millis, int micros)
299         {

301         TS_RESP_CTX_accuracy_free(ctx);
302         if (secs && (!(ctx->seconds = ASN1_INTEGER_new())
303                      || !ASN1_INTEGER_set(ctx->seconds, secs)))
304                 goto err;
305         if (millis && (!(ctx->millis = ASN1_INTEGER_new())
306                        || !ASN1_INTEGER_set(ctx->millis, millis)))
307                 goto err;
308         if (micros && (!(ctx->micros = ASN1_INTEGER_new())
309                        || !ASN1_INTEGER_set(ctx->micros, micros)))
310                 goto err;

312         return 1;
313  err:
314         TS_RESP_CTX_accuracy_free(ctx);
315         TSerr(TS_F_TS_RESP_CTX_SET_ACCURACY, ERR_R_MALLOC_FAILURE);
316         return 0;
317         }

319 void TS_RESP_CTX_add_flags(TS_RESP_CTX *ctx, int flags)
320         {
321         ctx->flags |= flags;
322         }

324 void TS_RESP_CTX_set_serial_cb(TS_RESP_CTX *ctx, TS_serial_cb cb, void *data)
325         {
```



```
326         ctx->serial_cb = cb;
327         ctx->serial_cb_data = data;
328         }

330 void TS_RESP_CTX_set_time_cb(TS_RESP_CTX *ctx, TS_time_cb cb, void *data)
331         {
332         ctx->time_cb = cb;
333         ctx->time_cb_data = data;
334         }

336 void TS_RESP_CTX_set_extension_cb(TS_RESP_CTX *ctx,
337                                   TS_extension_cb cb, void *data)
338         {
339         ctx->extension_cb = cb;
340         ctx->extension_cb_data = data;
341         }

343 int TS_RESP_CTX_set_status_info(TS_RESP_CTX *ctx,
344                                 int status, const char *text)
345         {
346         TS_STATUS_INFO *si = NULL;
347         ASN1_UTF8STRING *utf8_text = NULL;
348         int ret = 0;

350         if (!(si = TS_STATUS_INFO_new())) goto err;
351         if (!ASN1_INTEGER_set(si->status, status)) goto err;
352         if (text)
353                 {
354                 if (!(utf8_text = ASN1_UTF8STRING_new())
355                     || !ASN1_STRING_set(utf8_text, text, strlen(text)))
356                         goto err;
357                 if (!si->text && !(si->text = sk_ASN1_UTF8STRING_new_null()))
358                         goto err;
359                 if (!sk_ASN1_UTF8STRING_push(si->text, utf8_text)) goto err;
360                 utf8_text = NULL;       /* Ownership is lost. */
361                 }
362         if (!TS_RESP_set_status_info(ctx->response, si)) goto err;
363         ret = 1;
364  err:
365         if (!ret)
366                 TSerr(TS_F_TS_RESP_CTX_SET_STATUS_INFO, ERR_R_MALLOC_FAILURE);
367         TS_STATUS_INFO_free(si);
368         ASN1_UTF8STRING_free(utf8_text);
369         return ret;
370         }

372 int TS_RESP_CTX_set_status_info_cond(TS_RESP_CTX *ctx,
373                                      int status, const char *text)
374         {
375         int ret = 1;
376         TS_STATUS_INFO *si = TS_RESP_get_status_info(ctx->response);

378         if (ASN1_INTEGER_get(si->status) == TS_STATUS_GRANTED)
379                 {
380                 /* Status has not been set, set it now. */
381                 ret = TS_RESP_CTX_set_status_info(ctx, status, text);
382                 }
383         return ret;
384         }

386 int TS_RESP_CTX_add_failure_info(TS_RESP_CTX *ctx, int failure)
387         {
388         TS_STATUS_INFO *si = TS_RESP_get_status_info(ctx->response);
389         if (!si->failure_info && !(si->failure_info = ASN1_BIT_STRING_new()))
390                 goto err;
391         if (!ASN1_BIT_STRING_set_bit(si->failure_info, failure, 1))
```

```
392                 goto err;
393         return 1;
394  err:
395         TSerr(TS_F_TS_RESP_CTX_ADD_FAILURE_INFO, ERR_R_MALLOC_FAILURE);
396         return 0;
397         }

399 TS_REQ *TS_RESP_CTX_get_request(TS_RESP_CTX *ctx)
400         {
401         return ctx->request;
402         }

404 TS_TST_INFO *TS_RESP_CTX_get_tst_info(TS_RESP_CTX *ctx)
405         {
406         return ctx->tst_info;
407         }

409 int TS_RESP_CTX_set_clock_precision_digits(TS_RESP_CTX *ctx, unsigned precision)
410        {
411        if (precision > TS_MAX_CLOCK_PRECISION_DIGITS)
412                return 0;
413        ctx->clock_precision_digits = precision;
414        return 1;
415        }

417 /* Main entry method of the response generation. */
418 TS_RESP *TS_RESP_create_response(TS_RESP_CTX *ctx, BIO *req_bio)
419         {
420         ASN1_OBJECT *policy;
421         TS_RESP *response;
422         int result = 0;

424         TS_RESP_CTX_init(ctx);

426         /* Creating the response object. */
427         if (!(ctx->response = TS_RESP_new()))
428                 {
429                 TSerr(TS_F_TS_RESP_CREATE_RESPONSE, ERR_R_MALLOC_FAILURE);
430                 goto end;
431                 }

433         /* Parsing DER request. */
434         if (!(ctx->request = d2i_TS_REQ_bio(req_bio, NULL)))
435                 {
436                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
437                                             "Bad request format or "
438                                             "system error.");
439                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_BAD_DATA_FORMAT);
440                 goto end;
441                 }

443         /* Setting default status info. */
444         if (!TS_RESP_CTX_set_status_info(ctx, TS_STATUS_GRANTED, NULL))
445                 goto end;

447         /* Checking the request format. */
448         if (!TS_RESP_check_request(ctx)) goto end;

450         /* Checking acceptable policies. */
451         if (!(policy = TS_RESP_get_policy(ctx))) goto end;

453         /* Creating the TS_TST_INFO object. */
454         if (!(ctx->tst_info = TS_RESP_create_tst_info(ctx, policy)))
455                 goto end;

457         /* Processing extensions. */
```



```
458         if (!TS_RESP_process_extensions(ctx)) goto end;

460         /* Generating the signature. */
461         if (!TS_RESP_sign(ctx)) goto end;

463         /* Everything was successful. */
464         result = 1;
465  end:
466         if (!result)
467                 {
468                 TSerr(TS_F_TS_RESP_CREATE_RESPONSE, TS_R_RESPONSE_SETUP_ERROR);
469                 if (ctx->response != NULL)
470                         {
471                         if (TS_RESP_CTX_set_status_info_cond(ctx,
472                                 TS_STATUS_REJECTION, "Error during response "
473                                 "generation.") == 0)
474                                 {
475                                 TS_RESP_free(ctx->response);
476                                 ctx->response = NULL;
477                                 }
478                         }
479                 }
480         response = ctx->response;
481         ctx->response = NULL;   /* Ownership will be returned to caller. */
482         TS_RESP_CTX_cleanup(ctx);
483         return response;
484         }

486 /* Initializes the variable part of the context. */
487 static void TS_RESP_CTX_init(TS_RESP_CTX *ctx)
488         {
489         ctx->request = NULL;
490         ctx->response = NULL;
491         ctx->tst_info = NULL;
492         }

494 /* Cleans up the variable part of the context. */
495 static void TS_RESP_CTX_cleanup(TS_RESP_CTX *ctx)
496         {
497         TS_REQ_free(ctx->request);
498         ctx->request = NULL;
499         TS_RESP_free(ctx->response);
500         ctx->response = NULL;
501         TS_TST_INFO_free(ctx->tst_info);
502         ctx->tst_info = NULL;
503         }

505 /* Checks the format and content of the request. */
506 static int TS_RESP_check_request(TS_RESP_CTX *ctx)
507         {
508         TS_REQ *request = ctx->request;
509         TS_MSG_IMPRINT *msg_imprint;
510         X509_ALGOR *md_alg;
511         int md_alg_id;
512         const ASN1_OCTET_STRING *digest;
513         EVP_MD *md = NULL;
514         int i;

516         /* Checking request version. */
517         if (TS_REQ_get_version(request) != 1)
518                 {
519                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
520                                             "Bad request version.");
521                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_BAD_REQUEST);
522                 return 0;
523                 }
```

```
525         /* Checking message digest algorithm. */
526         msg_imprint = TS_REQ_get_msg_imprint(request);
527         md_alg = TS_MSG_IMPRINT_get_algo(msg_imprint);
528         md_alg_id = OBJ_obj2nid(md_alg->algorithm);
529         for (i = 0; !md && i < sk_EVP_MD_num(ctx->mds); ++i)
530                 {
531                 EVP_MD *current_md = sk_EVP_MD_value(ctx->mds, i);
532                 if (md_alg_id == EVP_MD_type(current_md))
533                         md = current_md;
534                 }
535         if (!md)
536                 {
537                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
538                                             "Message digest algorithm is "
539                                             "not supported.");
540                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_BAD_ALG);
541                 return 0;
542                 }

544         /* No message digest takes parameter. */
545         if (md_alg->parameter
546             && ASN1_TYPE_get(md_alg->parameter) != V_ASN1_NULL)
547                 {
548                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
549                                             "Superfluous message digest "
550                                             "parameter.");
551                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_BAD_ALG);
552                 return 0;
553                 }
554         /* Checking message digest size. */
555         digest = TS_MSG_IMPRINT_get_msg(msg_imprint);
556         if (digest->length != EVP_MD_size(md))
557                 {
558                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
559                                             "Bad message digest.");
560                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_BAD_DATA_FORMAT);
561                 return 0;
562                 }

564         return 1;
565         }

567 /* Returns the TSA policy based on the requested and acceptable policies. */
568 static ASN1_OBJECT *TS_RESP_get_policy(TS_RESP_CTX *ctx)
569         {
570         ASN1_OBJECT *requested = TS_REQ_get_policy_id(ctx->request);
571         ASN1_OBJECT *policy = NULL;
572         int i;

574         if (ctx->default_policy == NULL)
575                 {
576                 TSerr(TS_F_TS_RESP_GET_POLICY, TS_R_INVALID_NULL_POINTER);
577                 return NULL;
578                 }
579         /* Return the default policy if none is requested or the default is
580            requested. */
581         if (!requested || !OBJ_cmp(requested, ctx->default_policy))
582                 policy = ctx->default_policy;

584         /* Check if the policy is acceptable. */
585         for (i = 0; !policy && i < sk_ASN1_OBJECT_num(ctx->policies); ++i)
586                 {
587                 ASN1_OBJECT *current = sk_ASN1_OBJECT_value(ctx->policies, i);
588                 if (!OBJ_cmp(requested, current))
589                         policy = current;
```



```
590                 }
591         if (!policy)
592                 {
593                 TSerr(TS_F_TS_RESP_GET_POLICY, TS_R_UNACCEPTABLE_POLICY);
594                 TS_RESP_CTX_set_status_info(ctx, TS_STATUS_REJECTION,
595                                             "Requested policy is not "
596                                             "supported.");
597                 TS_RESP_CTX_add_failure_info(ctx, TS_INFO_UNACCEPTED_POLICY);
598                 }
599         return policy;
600         }

602 /* Creates the TS_TST_INFO object based on the settings of the context. */
603 static TS_TST_INFO *TS_RESP_create_tst_info(TS_RESP_CTX *ctx,
604                                             ASN1_OBJECT *policy)
605         {
606         int result = 0;
607         TS_TST_INFO *tst_info = NULL;
608         ASN1_INTEGER *serial = NULL;
609         ASN1_GENERALIZEDTIME *asn1_time = NULL;
610         long sec, usec;
611         TS_ACCURACY *accuracy = NULL;
612         const ASN1_INTEGER *nonce;
613         GENERAL_NAME *tsa_name = NULL;

615         if (!(tst_info = TS_TST_INFO_new())) goto end;
616         if (!TS_TST_INFO_set_version(tst_info, 1)) goto end;
617         if (!TS_TST_INFO_set_policy_id(tst_info, policy)) goto end;
618         if (!TS_TST_INFO_set_msg_imprint(tst_info, ctx->request->msg_imprint))
619                 goto end;
620         if (!(serial = (*ctx->serial_cb)(ctx, ctx->serial_cb_data))
621             || !TS_TST_INFO_set_serial(tst_info, serial))
622                 goto end;
623         if (!(*ctx->time_cb)(ctx, ctx->time_cb_data, &sec, &usec)
624             || !(asn1_time = TS_RESP_set_genTime_with_precision(NULL,
625                                         sec, usec,
626                                         ctx->clock_precision_digits))
627             || !TS_TST_INFO_set_time(tst_info, asn1_time))
628                 goto end;

630         /* Setting accuracy if needed. */
631         if ((ctx->seconds || ctx->millis || ctx->micros)
632             && !(accuracy = TS_ACCURACY_new()))
633                 goto end;

635         if (ctx->seconds && !TS_ACCURACY_set_seconds(accuracy, ctx->seconds))
636                 goto end;
637         if (ctx->millis && !TS_ACCURACY_set_millis(accuracy, ctx->millis))
638                 goto end;
639         if (ctx->micros && !TS_ACCURACY_set_micros(accuracy, ctx->micros))
640                 goto end;
641         if (accuracy && !TS_TST_INFO_set_accuracy(tst_info, accuracy))
642                 goto end;

644         /* Setting ordering. */
645         if ((ctx->flags & TS_ORDERING)
646             && !TS_TST_INFO_set_ordering(tst_info, 1))
647                 goto end;

649         /* Setting nonce if needed. */
650         if ((nonce = TS_REQ_get_nonce(ctx->request)) != NULL
651             && !TS_TST_INFO_set_nonce(tst_info, nonce))
652                 goto end;

654         /* Setting TSA name to subject of signer certificate. */
655         if (ctx->flags & TS_TSA_NAME)
```

```
656                 {
657                 if (!(tsa_name = GENERAL_NAME_new())) goto end;
658                 tsa_name->type = GEN_DIRNAME;
659                 tsa_name->d.dirn =
660                         X509_NAME_dup(ctx->signer_cert->cert_info->subject);
661                 if (!tsa_name->d.dirn) goto end;
662                 if (!TS_TST_INFO_set_tsa(tst_info, tsa_name)) goto end;
663                 }

665         result = 1;
666  end:
667         if (!result)
668                 {
669                 TS_TST_INFO_free(tst_info);
670                 tst_info = NULL;
671                 TSerr(TS_F_TS_RESP_CREATE_TST_INFO, TS_R_TST_INFO_SETUP_ERROR);
672                 TS_RESP_CTX_set_status_info_cond(ctx, TS_STATUS_REJECTION,
673                                                  "Error during TSTInfo "
674                                                  "generation.");
675                 }
676         GENERAL_NAME_free(tsa_name);
677         TS_ACCURACY_free(accuracy);
678         ASN1_GENERALIZEDTIME_free(asn1_time);
679         ASN1_INTEGER_free(serial);

681         return tst_info;
682         }

684 /* Processing the extensions of the request. */
685 static int TS_RESP_process_extensions(TS_RESP_CTX *ctx)
686         {
687         STACK_OF(X509_EXTENSION) *exts = TS_REQ_get_exts(ctx->request);
688         int i;
689         int ok = 1;

691         for (i = 0; ok && i < sk_X509_EXTENSION_num(exts); ++i)
692                 {
693                 X509_EXTENSION *ext = sk_X509_EXTENSION_value(exts, i);
694                 /* XXXXX The last argument was previously
695                    (void *)ctx->extension_cb, but ISO C doesn't permit
696                    converting a function pointer to void *.  For lack of
697                    better information, I'm placing a NULL there instead.
698                    The callback can pick its own address out from the ctx
699                    anyway...
700                 */
701                 ok = (*ctx->extension_cb)(ctx, ext, NULL);
702                 }

704         return ok;
705         }

707 /* Functions for signing the TS_TST_INFO structure of the context. */
708 static int TS_RESP_sign(TS_RESP_CTX *ctx)
709         {
710         int ret = 0;
711         PKCS7 *p7 = NULL;
712         PKCS7_SIGNER_INFO *si;
713         STACK_OF(X509) *certs;  /* Certificates to include in sc. */
714         ESS_SIGNING_CERT *sc = NULL;
715         ASN1_OBJECT *oid;
716         BIO *p7bio = NULL;
717         int i;

719         /* Check if signcert and pkey match. */
720         if (!X509_check_private_key(ctx->signer_cert, ctx->signer_key)) {
721                 TSerr(TS_F_TS_RESP_SIGN,
```



```
722                       TS_R_PRIVATE_KEY_DOES_NOT_MATCH_CERTIFICATE);
723                 goto err;
724         }

726         /* Create a new PKCS7 signed object. */
727         if (!(p7 = PKCS7_new())) {
728                 TSerr(TS_F_TS_RESP_SIGN, ERR_R_MALLOC_FAILURE);
729                 goto err;
730         }
731         if (!PKCS7_set_type(p7, NID_pkcs7_signed)) goto err;

733         /* Force SignedData version to be 3 instead of the default 1. */
734         if (!ASN1_INTEGER_set(p7->d.sign->version, 3)) goto err;

736         /* Add signer certificate and optional certificate chain. */
737         if (TS_REQ_get_cert_req(ctx->request))
738                 {
739                 PKCS7_add_certificate(p7, ctx->signer_cert);
740                 if (ctx->certs)
741                         {
742                         for(i = 0; i < sk_X509_num(ctx->certs); ++i)
743                                 {
744                                 X509 *cert = sk_X509_value(ctx->certs, i);
745                                 PKCS7_add_certificate(p7, cert);
746                                 }
747                         }
748                 }

750         /* Add a new signer info. */
751         if (!(si = PKCS7_add_signature(p7, ctx->signer_cert,
752                                        ctx->signer_key, EVP_sha1())))
753                 {
754                 TSerr(TS_F_TS_RESP_SIGN, TS_R_PKCS7_ADD_SIGNATURE_ERROR);
755                 goto err;
756                 }

758         /* Add content type signed attribute to the signer info. */
759         oid = OBJ_nid2obj(NID_id_smime_ct_TSTInfo);
760         if (!PKCS7_add_signed_attribute(si, NID_pkcs9_contentType,
761                                         V_ASN1_OBJECT, oid))
762                 {
763                 TSerr(TS_F_TS_RESP_SIGN, TS_R_PKCS7_ADD_SIGNED_ATTR_ERROR);
764                 goto err;
765                 }

767         /* Create the ESS SigningCertificate attribute which contains
768            the signer certificate id and optionally the certificate chain. */
769         certs = ctx->flags & TS_ESS_CERT_ID_CHAIN ? ctx->certs : NULL;
770         if (!(sc = ESS_SIGNING_CERT_new_init(ctx->signer_cert, certs)))
771                 goto err;

773         /* Add SigningCertificate signed attribute to the signer info. */
774         if (!ESS_add_signing_cert(si, sc))
775                 {
776                 TSerr(TS_F_TS_RESP_SIGN, TS_R_ESS_ADD_SIGNING_CERT_ERROR);
777                 goto err;
778                 }

780         /* Add a new empty NID_id_smime_ct_TSTInfo encapsulated content. */
781         if (!TS_TST_INFO_content_new(p7)) goto err;

783         /* Add the DER encoded tst_info to the PKCS7 structure. */
784         if (!(p7bio = PKCS7_dataInit(p7, NULL))) {
785                 TSerr(TS_F_TS_RESP_SIGN, ERR_R_MALLOC_FAILURE);
786                 goto err;
787         }
```

```
789         /* Convert tst_info to DER. */
790         if (!i2d_TS_TST_INFO_bio(p7bio, ctx->tst_info))
791                 {
792                 TSerr(TS_F_TS_RESP_SIGN, TS_R_TS_DATASIGN);
793                 goto err;
794                 }
795
796         /* Create the signature and add it to the signer info. */
797         if (!PKCS7_dataFinal(p7, p7bio))
798                 {
799                 TSerr(TS_F_TS_RESP_SIGN, TS_R_TS_DATASIGN);
800                 goto err;
801                 }
802
803         /* Set new PKCS7 and TST_INFO objects. */
804         TS_RESP_set_tst_info(ctx->response, p7, ctx->tst_info);
805         p7 = NULL;              /* Ownership is lost. */
806         ctx->tst_info = NULL;   /* Ownership is lost. */
807
808         ret = 1;
809  err:
810         if (!ret)
811                 TS_RESP_CTX_set_status_info_cond(ctx, TS_STATUS_REJECTION,
812                                                  "Error during signature "
813                                                  "generation.");
814         BIO_free_all(p7bio);
815         ESS_SIGNING_CERT_free(sc);
816         PKCS7_free(p7);
817         return ret;
818         }
819
820 static ESS_SIGNING_CERT *ESS_SIGNING_CERT_new_init(X509 *signcert,
821                                                    STACK_OF(X509) *certs)
822         {
823         ESS_CERT_ID *cid;
824         ESS_SIGNING_CERT *sc = NULL;
825         int i;
826
827         /* Creating the ESS_CERT_ID stack. */
828         if (!(sc = ESS_SIGNING_CERT_new())) goto err;
829         if (!sc->cert_ids && !(sc->cert_ids = sk_ESS_CERT_ID_new_null()))
830                 goto err;
831
832         /* Adding the signing certificate id. */
833         if (!(cid = ESS_CERT_ID_new_init(signcert, 0))
834             || !sk_ESS_CERT_ID_push(sc->cert_ids, cid))
835                 goto err;
836         /* Adding the certificate chain ids. */
837         for (i = 0; i < sk_X509_num(certs); ++i)
838                 {
839                 X509 *cert = sk_X509_value(certs, i);
840                 if (!(cid = ESS_CERT_ID_new_init(cert, 1))
841                     || !sk_ESS_CERT_ID_push(sc->cert_ids, cid))
842                         goto err;
843                 }
844
845         return sc;
846 err:
847         ESS_SIGNING_CERT_free(sc);
848         TSerr(TS_F_ESS_SIGNING_CERT_NEW_INIT, ERR_R_MALLOC_FAILURE);
849         return NULL;
850         }
851
852 static ESS_CERT_ID *ESS_CERT_ID_new_init(X509 *cert, int issuer_needed)
853         {
```



```
854         ESS_CERT_ID *cid = NULL;
855         GENERAL_NAME *name = NULL;
856
857         /* Recompute SHA1 hash of certificate if necessary (side effect). */
858         X509_check_purpose(cert, -1, 0);
859
860         if (!(cid = ESS_CERT_ID_new())) goto err;
861         if (!ASN1_OCTET_STRING_set(cid->hash, cert->sha1_hash,
862                                    sizeof(cert->sha1_hash)))
863                 goto err;
864
865         /* Setting the issuer/serial if requested. */
866         if (issuer_needed)
867                 {
868                 /* Creating issuer/serial structure. */
869                 if (!cid->issuer_serial
870                     && !(cid->issuer_serial = ESS_ISSUER_SERIAL_new()))
871                         goto err;
872                 /* Creating general name from the certificate issuer. */
873                 if (!(name = GENERAL_NAME_new())) goto err;
874                 name->type = GEN_DIRNAME;
875                 if (!(name->d.dirn = X509_NAME_dup(cert->cert_info->issuer)))
876                         goto err;
877                 if (!sk_GENERAL_NAME_push(cid->issuer_serial->issuer, name))
878                         goto err;
879                 name = NULL;    /* Ownership is lost. */
880                 /* Setting the serial number. */
881                 ASN1_INTEGER_free(cid->issuer_serial->serial);
882                 if (!(cid->issuer_serial->serial =
883                       ASN1_INTEGER_dup(cert->cert_info->serialNumber)))
884                         goto err;
885                 }
886
887         return cid;
888 err:
889         GENERAL_NAME_free(name);
890         ESS_CERT_ID_free(cid);
891         TSerr(TS_F_ESS_CERT_ID_NEW_INIT, ERR_R_MALLOC_FAILURE);
892         return NULL;
893         }
894
895 static int TS_TST_INFO_content_new(PKCS7 *p7)
896         {
897         PKCS7 *ret = NULL;
898         ASN1_OCTET_STRING *octet_string = NULL;
899
900         /* Create new encapsulated NID_id_smime_ct_TSTInfo content. */
901         if (!(ret = PKCS7_new())) goto err;
902         if (!(ret->d.other = ASN1_TYPE_new())) goto err;
903         ret->type = OBJ_nid2obj(NID_id_smime_ct_TSTInfo);
904         if (!(octet_string = ASN1_OCTET_STRING_new())) goto err;
905         ASN1_TYPE_set(ret->d.other, V_ASN1_OCTET_STRING, octet_string);
906         octet_string = NULL;
907
908         /* Add encapsulated content to signed PKCS7 structure. */
909         if (!PKCS7_set_content(p7, ret)) goto err;
910
911         return 1;
912  err:
913         ASN1_OCTET_STRING_free(octet_string);
914         PKCS7_free(ret);
915         return 0;
916         }
917
918 static int ESS_add_signing_cert(PKCS7_SIGNER_INFO *si, ESS_SIGNING_CERT *sc)
919         {
```

```
920         ASN1_STRING *seq = NULL;
921         unsigned char *p, *pp = NULL;
922         int len;

924         len = i2d_ESS_SIGNING_CERT(sc, NULL);
925         if (!(pp = (unsigned char *) OPENSSL_malloc(len)))
926                 {
927                 TSerr(TS_F_ESS_ADD_SIGNING_CERT, ERR_R_MALLOC_FAILURE);
928                 goto err;
929                 }
930         p = pp;
931         i2d_ESS_SIGNING_CERT(sc, &p);
932         if (!(seq = ASN1_STRING_new()) || !ASN1_STRING_set(seq, pp, len))
933                 {
934                 TSerr(TS_F_ESS_ADD_SIGNING_CERT, ERR_R_MALLOC_FAILURE);
935                 goto err;
936                 }
937         OPENSSL_free(pp); pp = NULL;
938         return PKCS7_add_signed_attribute(si,
939                                           NID_id_smime_aa_signingCertificate,
940                                           V_ASN1_SEQUENCE, seq);
941  err:
942         ASN1_STRING_free(seq);
943         OPENSSL_free(pp);

945         return 0;
946         }


949 static ASN1_GENERALIZEDTIME *
950 TS_RESP_set_genTime_with_precision(ASN1_GENERALIZEDTIME *asn1_time,
951                                    long sec, long usec, unsigned precision)
952         {
953         time_t time_sec = (time_t) sec;
954         struct tm *tm = NULL;
955         char genTime_str[17 + TS_MAX_CLOCK_PRECISION_DIGITS];
956         char *p = genTime_str;
957         char *p_end = genTime_str + sizeof(genTime_str);

959         if (precision > TS_MAX_CLOCK_PRECISION_DIGITS)
960                 goto err;


963         if (!(tm = gmtime(&time_sec)))
964                 goto err;

966         /*
967          * Put "genTime_str" in GeneralizedTime format.  We work around the
968          * restrictions imposed by rfc3280 (i.e. "GeneralizedTime values MUST
969          * NOT include fractional seconds") and OpenSSL related functions to
970          * meet the rfc3161 requirement: "GeneralizedTime syntax can include
971          * fraction-of-second details".
972          */
973         p += BIO_snprintf(p, p_end - p,
974                           "%04d%02d%02d%02d%02d%02d",
975                           tm->tm_year + 1900, tm->tm_mon + 1, tm->tm_mday,
976                           tm->tm_hour, tm->tm_min, tm->tm_sec);
977         if (precision > 0)
978         {
979                 /* Add fraction of seconds (leave space for dot and null). */
980                 BIO_snprintf(p, 2 + precision, ".%ld", usec);
981                 /* We cannot use the snprintf return value,
982                    because it might have been truncated. */
983                 p += strlen(p);

985                 /* To make things a bit harder, X.690 | ISO/IEC 8825-1 provides
```



```
986                    the following restrictions for a DER-encoding, which OpenSSL
987                    (specifically ASN1_GENERALIZEDTIME_check() function) doesn't
988                    support:
989                    "The encoding MUST terminate with a "Z" (which means "Zulu"
990                    time). The decimal point element, if present, MUST be the
991                    point option ".". The fractional-seconds elements,
992                    if present, MUST omit all trailing 0's;
993                    if the elements correspond to 0, they MUST be wholly
994                    omitted, and the decimal point element also MUST be
995                    omitted." */
996                 /* Remove trailing zeros. The dot guarantees the exit
997                    condition of this loop even if all the digits are zero. */
998                 while (*--p == '0')
999                         /* empty */;
1000                /* p points to either the dot or the last non-zero digit. */
1001                if (*p != '.') ++p;
1002                }
1003        /* Add the trailing Z and the terminating null. */
1004        *p++ = 'Z';
1005        *p++ = '\0';

1007        /* Now call OpenSSL to check and set our genTime value */
1008        if (!asn1_time && !(asn1_time = M_ASN1_GENERALIZEDTIME_new()))
1009                goto err;
1010        if (!ASN1_GENERALIZEDTIME_set_string(asn1_time, genTime_str))
1011                {
1012                ASN1_GENERALIZEDTIME_free(asn1_time);
1013                goto err;
1014                }

1016        return asn1_time;
1017 err:
1018        TSerr(TS_F_TS_RESP_SET_GENTIME_WITH_PRECISION, TS_R_COULD_NOT_SET_TIME);
1019        return NULL;
1020        }
1021 #endif /* ! codereview */
```

```
**********************************************************
   10069 Wed Aug 13 19:53:21 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_rsp_utils.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_resp_utils.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/ts.h>
#include <openssl/pkcs7.h>

/* Function definitions. */

int TS_RESP_set_status_info(TS_RESP *a, TS_STATUS_INFO *status_info)
        {
        TS_STATUS_INFO *new_status_info;

        if (a->status_info == status_info)
                return 1;
        new_status_info = TS_STATUS_INFO_dup(status_info);
        if (new_status_info == NULL)
                {
                TSerr(TS_F_TS_RESP_SET_STATUS_INFO, ERR_R_MALLOC_FAILURE);
                return 0;
                }
        TS_STATUS_INFO_free(a->status_info);
        a->status_info = new_status_info;

        return 1;
        }

TS_STATUS_INFO *TS_RESP_get_status_info(TS_RESP *a)
        {
        return a->status_info;
        }

/* Caller loses ownership of PKCS7 and TS_TST_INFO objects. */
void TS_RESP_set_tst_info(TS_RESP *a, PKCS7 *p7, TS_TST_INFO *tst_info)
        {
        /* Set new PKCS7 and TST_INFO objects. */
        PKCS7_free(a->token);
        a->token = p7;
        TS_TST_INFO_free(a->tst_info);
        a->tst_info = tst_info;
        }

PKCS7 *TS_RESP_get_token(TS_RESP *a)
        {
        return a->token;
        }

TS_TST_INFO *TS_RESP_get_tst_info(TS_RESP *a)
        {
        return a->tst_info;
        }

int TS_TST_INFO_set_version(TS_TST_INFO *a, long version)
        {
        return ASN1_INTEGER_set(a->version, version);
        }

long TS_TST_INFO_get_version(const TS_TST_INFO *a)
        {
        return ASN1_INTEGER_get(a->version);
        }

int TS_TST_INFO_set_policy_id(TS_TST_INFO *a, ASN1_OBJECT *policy)
        {
        ASN1_OBJECT *new_policy;

        if (a->policy_id == policy)
                return 1;
        new_policy = OBJ_dup(policy);
        if (new_policy == NULL)
```

```
128                 {
129                 TSerr(TS_F_TS_TST_INFO_SET_POLICY_ID, ERR_R_MALLOC_FAILURE);
130                 return 0;
131                 }
132         ASN1_OBJECT_free(a->policy_id);
133         a->policy_id = new_policy;
134         return 1;
135         }

137 ASN1_OBJECT *TS_TST_INFO_get_policy_id(TS_TST_INFO *a)
138         {
139         return a->policy_id;
140         }

142 int TS_TST_INFO_set_msg_imprint(TS_TST_INFO *a, TS_MSG_IMPRINT *msg_imprint)
143         {
144         TS_MSG_IMPRINT *new_msg_imprint;

146         if (a->msg_imprint == msg_imprint)
147                 return 1;
148         new_msg_imprint = TS_MSG_IMPRINT_dup(msg_imprint);
149         if (new_msg_imprint == NULL)
150                 {
151                 TSerr(TS_F_TS_TST_INFO_SET_MSG_IMPRINT, ERR_R_MALLOC_FAILURE);
152                 return 0;
153                 }
154         TS_MSG_IMPRINT_free(a->msg_imprint);
155         a->msg_imprint = new_msg_imprint;
156         return 1;
157         }

159 TS_MSG_IMPRINT *TS_TST_INFO_get_msg_imprint(TS_TST_INFO *a)
160         {
161         return a->msg_imprint;
162         }

164 int TS_TST_INFO_set_serial(TS_TST_INFO *a, const ASN1_INTEGER *serial)
165         {
166         ASN1_INTEGER *new_serial;

168         if (a->serial == serial)
169                 return 1;
170         new_serial = ASN1_INTEGER_dup(serial);
171         if (new_serial == NULL)
172                 {
173                 TSerr(TS_F_TS_TST_INFO_SET_SERIAL, ERR_R_MALLOC_FAILURE);
174                 return 0;
175                 }
176         ASN1_INTEGER_free(a->serial);
177         a->serial = new_serial;
178         return 1;
179         }

181 const ASN1_INTEGER *TS_TST_INFO_get_serial(const TS_TST_INFO *a)
182         {
183         return a->serial;
184         }

186 int TS_TST_INFO_set_time(TS_TST_INFO *a, const ASN1_GENERALIZEDTIME *gtime)
187         {
188         ASN1_GENERALIZEDTIME *new_time;

190         if (a->time == gtime)
191                 return 1;
192         new_time = M_ASN1_GENERALIZEDTIME_dup(gtime);
193         if (new_time == NULL)
```



```
194                 {
195                 TSerr(TS_F_TS_TST_INFO_SET_TIME, ERR_R_MALLOC_FAILURE);
196                 return 0;
197                 }
198         ASN1_GENERALIZEDTIME_free(a->time);
199         a->time = new_time;
200         return 1;
201         }

203 const ASN1_GENERALIZEDTIME *TS_TST_INFO_get_time(const TS_TST_INFO *a)
204         {
205         return a->time;
206         }

208 int TS_TST_INFO_set_accuracy(TS_TST_INFO *a, TS_ACCURACY *accuracy)
209         {
210         TS_ACCURACY *new_accuracy;

212         if (a->accuracy == accuracy)
213                 return 1;
214         new_accuracy = TS_ACCURACY_dup(accuracy);
215         if (new_accuracy == NULL)
216                 {
217                 TSerr(TS_F_TS_TST_INFO_SET_ACCURACY, ERR_R_MALLOC_FAILURE);
218                 return 0;
219                 }
220         TS_ACCURACY_free(a->accuracy);
221         a->accuracy = new_accuracy;
222         return 1;
223         }

225 TS_ACCURACY *TS_TST_INFO_get_accuracy(TS_TST_INFO *a)
226         {
227         return a->accuracy;
228         }

230 int TS_ACCURACY_set_seconds(TS_ACCURACY *a, const ASN1_INTEGER *seconds)
231         {
232         ASN1_INTEGER *new_seconds;

234         if (a->seconds == seconds)
235                 return 1;
236         new_seconds = ASN1_INTEGER_dup(seconds);
237         if (new_seconds == NULL)
238                 {
239                 TSerr(TS_F_TS_ACCURACY_SET_SECONDS, ERR_R_MALLOC_FAILURE);
240                 return 0;
241                 }
242         ASN1_INTEGER_free(a->seconds);
243         a->seconds = new_seconds;
244         return 1;
245         }

247 const ASN1_INTEGER *TS_ACCURACY_get_seconds(const TS_ACCURACY *a)
248         {
249         return a->seconds;
250         }

252 int TS_ACCURACY_set_millis(TS_ACCURACY *a, const ASN1_INTEGER *millis)
253         {
254         ASN1_INTEGER *new_millis = NULL;

256         if (a->millis == millis)
257                 return 1;
258         if (millis != NULL)
259                 {
```

```
260                 new_millis = ASN1_INTEGER_dup(millis);
261                 if (new_millis == NULL)
262                         {
263                         TSerr(TS_F_TS_ACCURACY_SET_MILLIS,
264                               ERR_R_MALLOC_FAILURE);
265                         return 0;
266                         }
267                 }
268         ASN1_INTEGER_free(a->millis);
269         a->millis = new_millis;
270         return 1;
271         }

273 const ASN1_INTEGER *TS_ACCURACY_get_millis(const TS_ACCURACY *a)
274         {
275         return a->millis;
276         }

278 int TS_ACCURACY_set_micros(TS_ACCURACY *a, const ASN1_INTEGER *micros)
279         {
280         ASN1_INTEGER *new_micros = NULL;

282         if (a->micros == micros)
283                 return 1;
284         if (micros != NULL)
285                 {
286                 new_micros = ASN1_INTEGER_dup(micros);
287                 if (new_micros == NULL)
288                         {
289                         TSerr(TS_F_TS_ACCURACY_SET_MICROS,
290                               ERR_R_MALLOC_FAILURE);
291                         return 0;
292                         }
293                 }
294         ASN1_INTEGER_free(a->micros);
295         a->micros = new_micros;
296         return 1;
297         }

299 const ASN1_INTEGER *TS_ACCURACY_get_micros(const TS_ACCURACY *a)
300         {
301         return a->micros;
302         }

304 int TS_TST_INFO_set_ordering(TS_TST_INFO *a, int ordering)
305         {
306         a->ordering = ordering ? 0xFF : 0x00;
307         return 1;
308         }

310 int TS_TST_INFO_get_ordering(const TS_TST_INFO *a)
311         {
312         return a->ordering ? 1 : 0;
313         }

315 int TS_TST_INFO_set_nonce(TS_TST_INFO *a, const ASN1_INTEGER *nonce)
316         {
317         ASN1_INTEGER *new_nonce;

319         if (a->nonce == nonce)
320                 return 1;
321         new_nonce = ASN1_INTEGER_dup(nonce);
322         if (new_nonce == NULL)
323                 {
324                 TSerr(TS_F_TS_TST_INFO_SET_NONCE, ERR_R_MALLOC_FAILURE);
325                 return 0;
```



```
326                 }
327         ASN1_INTEGER_free(a->nonce);
328         a->nonce = new_nonce;
329         return 1;
330         }

332 const ASN1_INTEGER *TS_TST_INFO_get_nonce(const TS_TST_INFO *a)
333         {
334         return a->nonce;
335         }

337 int TS_TST_INFO_set_tsa(TS_TST_INFO *a, GENERAL_NAME *tsa)
338         {
339         GENERAL_NAME *new_tsa;

341         if (a->tsa == tsa)
342                 return 1;
343         new_tsa = GENERAL_NAME_dup(tsa);
344         if (new_tsa == NULL)
345                 {
346                 TSerr(TS_F_TS_TST_INFO_SET_TSA, ERR_R_MALLOC_FAILURE);
347                 return 0;
348                 }
349         GENERAL_NAME_free(a->tsa);
350         a->tsa = new_tsa;
351         return 1;
352         }

354 GENERAL_NAME *TS_TST_INFO_get_tsa(TS_TST_INFO *a)
355         {
356         return a->tsa;
357         }

359 STACK_OF(X509_EXTENSION) *TS_TST_INFO_get_exts(TS_TST_INFO *a)
360         {
361         return a->extensions;
362         }

364 void TS_TST_INFO_ext_free(TS_TST_INFO *a)
365         {
366         if (!a) return;
367         sk_X509_EXTENSION_pop_free(a->extensions, X509_EXTENSION_free);
368         a->extensions = NULL;
369         }

371 int TS_TST_INFO_get_ext_count(TS_TST_INFO *a)
372         {
373         return X509v3_get_ext_count(a->extensions);
374         }

376 int TS_TST_INFO_get_ext_by_NID(TS_TST_INFO *a, int nid, int lastpos)
377         {
378         return X509v3_get_ext_by_NID(a->extensions, nid, lastpos);
379         }

381 int TS_TST_INFO_get_ext_by_OBJ(TS_TST_INFO *a, ASN1_OBJECT *obj, int lastpos)
382         {
383         return X509v3_get_ext_by_OBJ(a->extensions, obj, lastpos);
384         }

386 int TS_TST_INFO_get_ext_by_critical(TS_TST_INFO *a, int crit, int lastpos)
387         {
388         return X509v3_get_ext_by_critical(a->extensions, crit, lastpos);
389         }

391 X509_EXTENSION *TS_TST_INFO_get_ext(TS_TST_INFO *a, int loc)
```

```
392         {
393         return X509v3_get_ext(a->extensions,loc);
394         }

396 X509_EXTENSION *TS_TST_INFO_delete_ext(TS_TST_INFO *a, int loc)
397         {
398         return X509v3_delete_ext(a->extensions,loc);
399         }

401 int TS_TST_INFO_add_ext(TS_TST_INFO *a, X509_EXTENSION *ex, int loc)
402         {
403         return X509v3_add_ext(&a->extensions,ex,loc) != NULL;
404         }

406 void *TS_TST_INFO_get_ext_d2i(TS_TST_INFO *a, int nid, int *crit, int *idx)
407         {
408         return X509V3_get_d2i(a->extensions, nid, crit, idx);
409         }
410 #endif /* ! codereview */
```

```
**********************************************************
   21694 Wed Aug 13 19:53:21 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_rsp_verify.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_resp_verify.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2002.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/objects.h>
```



```
#include <openssl/ts.h>
#include <openssl/pkcs7.h>

/* Private function declarations. */

static int TS_verify_cert(X509_STORE *store, STACK_OF(X509) *untrusted,
                          X509 *signer, STACK_OF(X509) **chain);
static int TS_check_signing_certs(PKCS7_SIGNER_INFO *si, STACK_OF(X509) *chain);
static ESS_SIGNING_CERT *ESS_get_signing_cert(PKCS7_SIGNER_INFO *si);
static int TS_find_cert(STACK_OF(ESS_CERT_ID) *cert_ids, X509 *cert);
static int TS_issuer_serial_cmp(ESS_ISSUER_SERIAL *is, X509_CINF *cinfo);
static int int_TS_RESP_verify_token(TS_VERIFY_CTX *ctx,
                                    PKCS7 *token, TS_TST_INFO *tst_info);
static int TS_check_status_info(TS_RESP *response);
static char *TS_get_status_text(STACK_OF(ASN1_UTF8STRING) *text);
static int TS_check_policy(ASN1_OBJECT *req_oid, TS_TST_INFO *tst_info);
static int TS_compute_imprint(BIO *data, TS_TST_INFO *tst_info,
                              X509_ALGOR **md_alg,
                              unsigned char **imprint, unsigned *imprint_len);
static int TS_check_imprints(X509_ALGOR *algor_a,
                             unsigned char *imprint_a, unsigned len_a,
                             TS_TST_INFO *tst_info);
static int TS_check_nonces(const ASN1_INTEGER *a, TS_TST_INFO *tst_info);
static int TS_check_signer_name(GENERAL_NAME *tsa_name, X509 *signer);
static int TS_find_name(STACK_OF(GENERAL_NAME) *gen_names, GENERAL_NAME *name);

/*
 * Local mapping between response codes and descriptions.
 * Don't forget to change TS_STATUS_BUF_SIZE when modifying
 * the elements of this array.
 */
static const char *TS_status_text[] =
        { "granted",
          "grantedWithMods",
          "rejection",
          "waiting",
          "revocationWarning",
          "revocationNotification" };

#define TS_STATUS_TEXT_SIZE     (sizeof(TS_status_text)/sizeof(*TS_status_text))

/*
 * This must be greater or equal to the sum of the strings in TS_status_text
 * plus the number of its elements.
 */
#define TS_STATUS_BUF_SIZE      256

static struct
        {
        int code;
        const char *text;
        } TS_failure_info[] =
                { { TS_INFO_BAD_ALG, "badAlg" },
                  { TS_INFO_BAD_REQUEST, "badRequest" },
                  { TS_INFO_BAD_DATA_FORMAT, "badDataFormat" },
                  { TS_INFO_TIME_NOT_AVAILABLE, "timeNotAvailable" },
                  { TS_INFO_UNACCEPTED_POLICY, "unacceptedPolicy" },
                  { TS_INFO_UNACCEPTED_EXTENSION, "unacceptedExtension" },
                  { TS_INFO_ADD_INFO_NOT_AVAILABLE, "addInfoNotAvailable" },
                  { TS_INFO_SYSTEM_FAILURE, "systemFailure" } };

#define TS_FAILURE_INFO_SIZE    (sizeof(TS_failure_info) / \
                                 sizeof(*TS_failure_info))

/* Functions for verifying a signed TS_TST_INFO structure. */
```

```
/*
 * This function carries out the following tasks:
 *      - Checks if there is one and only one signer.
 *      - Search for the signing certificate in 'certs' and in the response.
 *      - Check the extended key usage and key usage fields of the signer
 *      certificate (done by the path validation).
 *      - Build and validate the certificate path.
 *      - Check if the certificate path meets the requirements of the
 *      SigningCertificate ESS signed attribute.
 *      - Verify the signature value.
 *      - Returns the signer certificate in 'signer', if 'signer' is not NULL.
 */
int TS_RESP_verify_signature(PKCS7 *token, STACK_OF(X509) *certs,
                             X509_STORE *store, X509 **signer_out)
        {
        STACK_OF(PKCS7_SIGNER_INFO) *sinfos = NULL;
        PKCS7_SIGNER_INFO *si;
        STACK_OF(X509) *signers = NULL;
        X509    *signer;
        STACK_OF(X509) *chain = NULL;
        char    buf[4096];
        int     i, j = 0, ret = 0;
        BIO     *p7bio = NULL;

        /* Some sanity checks first. */
        if (!token)
                {
                TSerr(TS_F_TS_RESP_VERIFY_SIGNATURE, TS_R_INVALID_NULL_POINTER);
                goto err;
                }

        /* Check for the correct content type */
        if(!PKCS7_type_is_signed(token))
                {
                TSerr(TS_F_TS_RESP_VERIFY_SIGNATURE, TS_R_WRONG_CONTENT_TYPE);
                goto err;
                }

        /* Check if there is one and only one signer. */
        sinfos = PKCS7_get_signer_info(token);
        if (!sinfos || sk_PKCS7_SIGNER_INFO_num(sinfos) != 1)
                {
                TSerr(TS_F_TS_RESP_VERIFY_SIGNATURE,
                      TS_R_THERE_MUST_BE_ONE_SIGNER);
                goto err;
                }
        si = sk_PKCS7_SIGNER_INFO_value(sinfos, 0);

        /* Check for no content: no data to verify signature. */
        if (PKCS7_get_detached(token))
                {
                TSerr(TS_F_TS_RESP_VERIFY_SIGNATURE, TS_R_NO_CONTENT);
                goto err;
                }

        /* Get hold of the signer certificate, search only internal
           certificates if it was requested. */
        signers = PKCS7_get0_signers(token, certs, 0);
        if (!signers || sk_X509_num(signers) != 1) goto err;
        signer = sk_X509_value(signers, 0);

        /* Now verify the certificate. */
        if (!TS_verify_cert(store, certs, signer, &chain)) goto err;

        /* Check if the signer certificate is consistent with the
           ESS extension. */
```



```
        if (!TS_check_signing_certs(si, chain)) goto err;

        /* Creating the message digest. */
        p7bio = PKCS7_dataInit(token, NULL);

        /* We now have to 'read' from p7bio to calculate digests etc. */
        while ((i = BIO_read(p7bio,buf,sizeof(buf))) > 0);

        /* Verifying the signature. */
        j = PKCS7_signatureVerify(p7bio, token, si, signer);
        if (j <= 0)
                {
                TSerr(TS_F_TS_RESP_VERIFY_SIGNATURE, TS_R_SIGNATURE_FAILURE);
                goto err;
                }

        /* Return the signer certificate if needed. */
        if (signer_out)
                {
                *signer_out = signer;
                CRYPTO_add(&signer->references, 1, CRYPTO_LOCK_X509);
                }

        ret = 1;

 err:
        BIO_free_all(p7bio);
        sk_X509_pop_free(chain, X509_free);
        sk_X509_free(signers);

        return ret;
        }

/*
 * The certificate chain is returned in chain. Caller is responsible for
 * freeing the vector.
 */
static int TS_verify_cert(X509_STORE *store, STACK_OF(X509) *untrusted,
                          X509 *signer, STACK_OF(X509) **chain)
        {
        X509_STORE_CTX  cert_ctx;
        int i;
        int ret = 1;

        /* chain is an out argument. */
        *chain = NULL;
        X509_STORE_CTX_init(&cert_ctx, store, signer, untrusted);
        X509_STORE_CTX_set_purpose(&cert_ctx, X509_PURPOSE_TIMESTAMP_SIGN);
        i = X509_verify_cert(&cert_ctx);
        if (i <= 0)
                {
                int j = X509_STORE_CTX_get_error(&cert_ctx);
                TSerr(TS_F_TS_VERIFY_CERT, TS_R_CERTIFICATE_VERIFY_ERROR);
                ERR_add_error_data(2, "Verify error:",
                                   X509_verify_cert_error_string(j));
                ret = 0;
                }
        else
                {
                /* Get a copy of the certificate chain. */
                *chain = X509_STORE_CTX_get1_chain(&cert_ctx);
                }

        X509_STORE_CTX_cleanup(&cert_ctx);

        return ret;
```

```
260         }

262 static int TS_check_signing_certs(PKCS7_SIGNER_INFO *si, STACK_OF(X509) *chain)
263         {
264         ESS_SIGNING_CERT *ss = ESS_get_signing_cert(si);
265         STACK_OF(ESS_CERT_ID) *cert_ids = NULL;
266         X509 *cert;
267         int i = 0;
268         int ret = 0;

270         if (!ss) goto err;
271         cert_ids = ss->cert_ids;
272         /* The signer certificate must be the first in cert_ids. */
273         cert = sk_X509_value(chain, 0);
274         if (TS_find_cert(cert_ids, cert) != 0) goto err;

276         /* Check the other certificates of the chain if there are more
277            than one certificate ids in cert_ids. */
278         if (sk_ESS_CERT_ID_num(cert_ids) > 1)
279                 {
280                 /* All the certificates of the chain must be in cert_ids. */
281                 for (i = 1; i < sk_X509_num(chain); ++i)
282                         {
283                         cert = sk_X509_value(chain, i);
284                         if (TS_find_cert(cert_ids, cert) < 0) goto err;
285                         }
286                 }
287         ret = 1;
288  err:
289         if (!ret)
290                 TSerr(TS_F_TS_CHECK_SIGNING_CERTS,
291                       TS_R_ESS_SIGNING_CERTIFICATE_ERROR);
292         ESS_SIGNING_CERT_free(ss);
293         return ret;
294         }

296 static ESS_SIGNING_CERT *ESS_get_signing_cert(PKCS7_SIGNER_INFO *si)
297         {
298         ASN1_TYPE *attr;
299         const unsigned char *p;
300         attr = PKCS7_get_signed_attribute(si,
301                                           NID_id_smime_aa_signingCertificate);
302         if (!attr) return NULL;
303         p = attr->value.sequence->data;
304         return d2i_ESS_SIGNING_CERT(NULL, &p, attr->value.sequence->length);
305         }

307 /* Returns < 0 if certificate is not found, certificate index otherwise. */
308 static int TS_find_cert(STACK_OF(ESS_CERT_ID) *cert_ids, X509 *cert)
309         {
310         int i;

312         if (!cert_ids || !cert) return -1;

314         /* Recompute SHA1 hash of certificate if necessary (side effect). */
315         X509_check_purpose(cert, -1, 0);

317         /* Look for cert in the cert_ids vector. */
318         for (i = 0; i < sk_ESS_CERT_ID_num(cert_ids); ++i)
319                 {
320                 ESS_CERT_ID *cid = sk_ESS_CERT_ID_value(cert_ids, i);

322                 /* Check the SHA-1 hash first. */
323                 if (cid->hash->length == sizeof(cert->sha1_hash)
324                     && !memcmp(cid->hash->data, cert->sha1_hash,
325                                sizeof(cert->sha1_hash)))
```



```
326                         {
327                         /* Check the issuer/serial as well if specified. */
328                         ESS_ISSUER_SERIAL *is = cid->issuer_serial;
329                         if (!is || !TS_issuer_serial_cmp(is, cert->cert_info))
330                                 return i;
331                         }
332                 }

334         return -1;
335         }

337 static int TS_issuer_serial_cmp(ESS_ISSUER_SERIAL *is, X509_CINF *cinfo)
338         {
339         GENERAL_NAME *issuer;

341         if (!is || !cinfo || sk_GENERAL_NAME_num(is->issuer) != 1) return -1;

343         /* Check the issuer first. It must be a directory name. */
344         issuer = sk_GENERAL_NAME_value(is->issuer, 0);
345         if (issuer->type != GEN_DIRNAME
346             || X509_NAME_cmp(issuer->d.dirn, cinfo->issuer))
347                 return -1;

349         /* Check the serial number, too. */
350         if (ASN1_INTEGER_cmp(is->serial, cinfo->serialNumber))
351                 return -1;

353         return 0;
354         }

356 /*
357  * Verifies whether 'response' contains a valid response with regards
358  * to the settings of the context:
359  *      - Gives an error message if the TS_TST_INFO is not present.
360  *      - Calls _TS_RESP_verify_token to verify the token content.
361  */
362 int TS_RESP_verify_response(TS_VERIFY_CTX *ctx, TS_RESP *response)
363         {
364         PKCS7 *token = TS_RESP_get_token(response);
365         TS_TST_INFO *tst_info = TS_RESP_get_tst_info(response);
366         int ret = 0;

368         /* Check if we have a successful TS_TST_INFO object in place. */
369         if (!TS_check_status_info(response)) goto err;

371         /* Check the contents of the time stamp token. */
372         if (!int_TS_RESP_verify_token(ctx, token, tst_info))
373                 goto err;

375         ret = 1;
376  err:
377         return ret;
378         }

380 /*
381  * Tries to extract a TS_TST_INFO structure from the PKCS7 token and
382  * calls the internal int_TS_RESP_verify_token function for verifying it.
383  */
384 int TS_RESP_verify_token(TS_VERIFY_CTX *ctx, PKCS7 *token)
385         {
386         TS_TST_INFO *tst_info = PKCS7_to_TS_TST_INFO(token);
387         int ret = 0;
388         if (tst_info)
389                 {
390                 ret = int_TS_RESP_verify_token(ctx, token, tst_info);
391                 TS_TST_INFO_free(tst_info);
```

```
392                 }
393         return ret;
394         }

396 /*
397  * Verifies whether the 'token' contains a valid time stamp token
398  * with regards to the settings of the context. Only those checks are
399  * carried out that are specified in the context:
400  *      - Verifies the signature of the TS_TST_INFO.
401  *      - Checks the version number of the response.
402  *      - Check if the requested and returned policies math.
403  *      - Check if the message imprints are the same.
404  *      - Check if the nonces are the same.
405  *      - Check if the TSA name matches the signer.
406  *      - Check if the TSA name is the expected TSA.
407  */
408 static int int_TS_RESP_verify_token(TS_VERIFY_CTX *ctx,
409                                     PKCS7 *token, TS_TST_INFO *tst_info)
410         {
411         X509 *signer = NULL;
412         GENERAL_NAME *tsa_name = TS_TST_INFO_get_tsa(tst_info);
413         X509_ALGOR *md_alg = NULL;
414         unsigned char *imprint = NULL;
415         unsigned imprint_len = 0;
416         int ret = 0;

418         /* Verify the signature. */
419         if ((ctx->flags & TS_VFY_SIGNATURE)
420             && !TS_RESP_verify_signature(token, ctx->certs, ctx->store,
421                                          &signer))
422                 goto err;

424         /* Check version number of response. */
425         if ((ctx->flags & TS_VFY_VERSION)
426             && TS_TST_INFO_get_version(tst_info) != 1)
427                 {
428                 TSerr(TS_F_INT_TS_RESP_VERIFY_TOKEN, TS_R_UNSUPPORTED_VERSION);
429                 goto err;
430                 }

432         /* Check policies. */
433         if ((ctx->flags & TS_VFY_POLICY)
434             && !TS_check_policy(ctx->policy, tst_info))
435                 goto err;

437         /* Check message imprints. */
438         if ((ctx->flags & TS_VFY_IMPRINT)
439             && !TS_check_imprints(ctx->md_alg, ctx->imprint, ctx->imprint_len,
440                                   tst_info))
441                 goto err;

443         /* Compute and check message imprints. */
444         if ((ctx->flags & TS_VFY_DATA)
445             && (!TS_compute_imprint(ctx->data, tst_info,
446                                     &md_alg, &imprint, &imprint_len)
447             || !TS_check_imprints(md_alg, imprint, imprint_len, tst_info)))
448                 goto err;

450         /* Check nonces. */
451         if ((ctx->flags & TS_VFY_NONCE)
452             && !TS_check_nonces(ctx->nonce, tst_info))
453                 goto err;

455         /* Check whether TSA name and signer certificate match. */
456         if ((ctx->flags & TS_VFY_SIGNER)
457             && tsa_name && !TS_check_signer_name(tsa_name, signer))
```



```
458                 {
459                 TSerr(TS_F_INT_TS_RESP_VERIFY_TOKEN, TS_R_TSA_NAME_MISMATCH);
460                 goto err;
461                 }

463         /* Check whether the TSA is the expected one. */
464         if ((ctx->flags & TS_VFY_TSA_NAME)
465             && !TS_check_signer_name(ctx->tsa_name, signer))
466                 {
467                 TSerr(TS_F_INT_TS_RESP_VERIFY_TOKEN, TS_R_TSA_UNTRUSTED);
468                 goto err;
469                 }

471         ret = 1;
472  err:
473         X509_free(signer);
474         X509_ALGOR_free(md_alg);
475         OPENSSL_free(imprint);
476         return ret;
477         }

479 static int TS_check_status_info(TS_RESP *response)
480         {
481         TS_STATUS_INFO *info = TS_RESP_get_status_info(response);
482         long status = ASN1_INTEGER_get(info->status);
483         const char *status_text = NULL;
484         char *embedded_status_text = NULL;
485         char failure_text[TS_STATUS_BUF_SIZE] = "";

487         /* Check if everything went fine. */
488         if (status == 0 || status == 1) return 1;

490         /* There was an error, get the description in status_text. */
491         if (0 <= status && status < (long)TS_STATUS_TEXT_SIZE)
492                 status_text = TS_status_text[status];
493         else
494                 status_text = "unknown code";

496         /* Set the embedded_status_text to the returned description. */
497         if (sk_ASN1_UTF8STRING_num(info->text) > 0
498             && !(embedded_status_text = TS_get_status_text(info->text)))
499                 return 0;

501         /* Filling in failure_text with the failure information. */
502         if (info->failure_info)
503                 {
504                 int i;
505                 int first = 1;
506                 for (i = 0; i < (int)TS_FAILURE_INFO_SIZE; ++i)
507                         {
508                         if (ASN1_BIT_STRING_get_bit(info->failure_info,
509                                                     TS_failure_info[i].code))
510                                 {
511                                 if (!first)
512                                         strcpy(failure_text, ",");
513                                 else
514                                         first = 0;
515                                 strcat(failure_text, TS_failure_info[i].text);
516                                 }
517                         }
518                 }
519         if (failure_text[0] == '\0')
520                 strcpy(failure_text, "unspecified");

522         /* Making up the error string. */
523         TSerr(TS_F_TS_CHECK_STATUS_INFO, TS_R_NO_TIME_STAMP_TOKEN);
```

```
524         ERR_add_error_data(6,
525                            "status code: ", status_text,
526                            ", status text: ", embedded_status_text ?
527                            embedded_status_text : "unspecified",
528                            ", failure codes: ", failure_text);
529         OPENSSL_free(embedded_status_text);
530
531         return 0;
532         }
533
534 static char *TS_get_status_text(STACK_OF(ASN1_UTF8STRING) *text)
535         {
536         int i;
537         unsigned int length = 0;
538         char *result = NULL;
539         char *p;
540
541         /* Determine length first. */
542         for (i = 0; i < sk_ASN1_UTF8STRING_num(text); ++i)
543                 {
544                 ASN1_UTF8STRING *current = sk_ASN1_UTF8STRING_value(text, i);
545                 length += ASN1_STRING_length(current);
546                 length += 1;    /* separator character */
547                 }
548         /* Allocate memory (closing '\0' included). */
549         if (!(result = OPENSSL_malloc(length)))
550                 {
551                 TSerr(TS_F_TS_GET_STATUS_TEXT, ERR_R_MALLOC_FAILURE);
552                 return NULL;
553                 }
554         /* Concatenate the descriptions. */
555         for (i = 0, p = result; i < sk_ASN1_UTF8STRING_num(text); ++i)
556                 {
557                 ASN1_UTF8STRING *current = sk_ASN1_UTF8STRING_value(text, i);
558                 length = ASN1_STRING_length(current);
559                 if (i > 0) *p++ = '/';
560                 strncpy(p, (const char *)ASN1_STRING_data(current), length);
561                 p += length;
562                 }
563         /* We do have space for this, too. */
564         *p = '\0';
565
566         return result;
567         }
568
569 static int TS_check_policy(ASN1_OBJECT *req_oid, TS_TST_INFO *tst_info)
570         {
571         ASN1_OBJECT *resp_oid = TS_TST_INFO_get_policy_id(tst_info);
572
573         if (OBJ_cmp(req_oid, resp_oid) != 0)
574                 {
575                 TSerr(TS_F_TS_CHECK_POLICY, TS_R_POLICY_MISMATCH);
576                 return 0;
577                 }
578
579         return 1;
580         }
581
582 static int TS_compute_imprint(BIO *data, TS_TST_INFO *tst_info,
583                               X509_ALGOR **md_alg,
584                               unsigned char **imprint, unsigned *imprint_len)
585         {
586         TS_MSG_IMPRINT *msg_imprint = TS_TST_INFO_get_msg_imprint(tst_info);
587         X509_ALGOR *md_alg_resp = TS_MSG_IMPRINT_get_algo(msg_imprint);
588         const EVP_MD *md;
589         EVP_MD_CTX md_ctx;
```



```
590         unsigned char buffer[4096];
591         int length;
592
593         *md_alg = NULL;
594         *imprint = NULL;
595
596         /* Return the MD algorithm of the response. */
597         if (!(*md_alg = X509_ALGOR_dup(md_alg_resp))) goto err;
598
599         /* Getting the MD object. */
600         if (!(md = EVP_get_digestbyobj((*md_alg)->algorithm)))
601                 {
602                 TSerr(TS_F_TS_COMPUTE_IMPRINT, TS_R_UNSUPPORTED_MD_ALGORITHM);
603                 goto err;
604                 }
605
606         /* Compute message digest. */
607         length = EVP_MD_size(md);
608         if (length < 0)
609             goto err;
610         *imprint_len = length;
611         if (!(*imprint = OPENSSL_malloc(*imprint_len)))
612                 {
613                 TSerr(TS_F_TS_COMPUTE_IMPRINT, ERR_R_MALLOC_FAILURE);
614                 goto err;
615                 }
616
617         if (!EVP_DigestInit(&md_ctx, md))
618                 goto err;
619         while ((length = BIO_read(data, buffer, sizeof(buffer))) > 0)
620                 {
621                 if (!EVP_DigestUpdate(&md_ctx, buffer, length))
622                         goto err;
623                 }
624         if (!EVP_DigestFinal(&md_ctx, *imprint, NULL))
625                 goto err;
626
627         return 1;
628  err:
629         X509_ALGOR_free(*md_alg);
630         OPENSSL_free(*imprint);
631         *imprint_len = 0;
632         *imprint = NULL;
633         return 0;
634         }
635
636 static int TS_check_imprints(X509_ALGOR *algor_a,
637                              unsigned char *imprint_a, unsigned len_a,
638                              TS_TST_INFO *tst_info)
639         {
640         TS_MSG_IMPRINT *b = TS_TST_INFO_get_msg_imprint(tst_info);
641         X509_ALGOR *algor_b = TS_MSG_IMPRINT_get_algo(b);
642         int ret = 0;
643
644         /* algor_a is optional. */
645         if (algor_a)
646                 {
647                 /* Compare algorithm OIDs. */
648                 if (OBJ_cmp(algor_a->algorithm, algor_b->algorithm)) goto err;
649
650                 /* The parameter must be NULL in both. */
651                 if ((algor_a->parameter
652                      && ASN1_TYPE_get(algor_a->parameter) != V_ASN1_NULL)
653                     || (algor_b->parameter
654                         && ASN1_TYPE_get(algor_b->parameter) != V_ASN1_NULL))
655                         goto err;
```

```
656                 }

658         /* Compare octet strings. */
659         ret = len_a == (unsigned) ASN1_STRING_length(b->hashed_msg) &&
660                 memcmp(imprint_a, ASN1_STRING_data(b->hashed_msg), len_a) == 0;
661  err:
662         if (!ret)
663                 TSerr(TS_F_TS_CHECK_IMPRINTS, TS_R_MESSAGE_IMPRINT_MISMATCH);
664         return ret;
665         }

667 static int TS_check_nonces(const ASN1_INTEGER *a, TS_TST_INFO *tst_info)
668         {
669         const ASN1_INTEGER *b = TS_TST_INFO_get_nonce(tst_info);

671         /* Error if nonce is missing. */
672         if (!b)
673                 {
674                 TSerr(TS_F_TS_CHECK_NONCES, TS_R_NONCE_NOT_RETURNED);
675                 return 0;
676                 }

678         /* No error if a nonce is returned without being requested. */
679         if (ASN1_INTEGER_cmp(a, b) != 0)
680                 {
681                 TSerr(TS_F_TS_CHECK_NONCES, TS_R_NONCE_MISMATCH);
682                 return 0;
683                 }

685         return 1;
686         }

688 /* Check if the specified TSA name matches either the subject
689    or one of the subject alternative names of the TSA certificate. */
690 static int TS_check_signer_name(GENERAL_NAME *tsa_name, X509 *signer)
691         {
692         STACK_OF(GENERAL_NAME) *gen_names = NULL;
693         int idx = -1;
694         int found = 0;

696         /* Check the subject name first. */
697         if (tsa_name->type == GEN_DIRNAME
698             && X509_name_cmp(tsa_name->d.dirn, signer->cert_info->subject) == 0)
699                 return 1;

701         /* Check all the alternative names. */
702         gen_names = X509_get_ext_d2i(signer, NID_subject_alt_name,
703                                      NULL, &idx);
704         while (gen_names != NULL
705                && !(found = TS_find_name(gen_names, tsa_name) >= 0))
706                 {
707                 /* Get the next subject alternative name,
708                    although there should be no more than one. */
709                 GENERAL_NAMES_free(gen_names);
710                 gen_names = X509_get_ext_d2i(signer, NID_subject_alt_name,
711                                              NULL, &idx);
712                 }
713         if (gen_names) GENERAL_NAMES_free(gen_names);

715         return found;
716         }

718 /* Returns 1 if name is in gen_names, 0 otherwise. */
719 static int TS_find_name(STACK_OF(GENERAL_NAME) *gen_names, GENERAL_NAME *name)
720         {
721         int i, found;
```



```
722         for (i = 0, found = 0; !found && i < sk_GENERAL_NAME_num(gen_names);
723              ++i)
724                 {
725                 GENERAL_NAME *current = sk_GENERAL_NAME_value(gen_names, i);
726                 found = GENERAL_NAME_cmp(current, name) == 0;
727                 }
728         return found ? i - 1 : -1;
729         }
730 #endif /* ! codereview */
```

```
**********************************************************
    4980 Wed Aug 13 19:53:21 2014
new/usr/src/lib/openssl/libsunw_crypto/ts/ts_verify_ctx.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ts/ts_verify_ctx.c */
/* Written by Zoltan Glozik (zglozik@stones.com) for the OpenSSL
 * project 2003.
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/objects.h>
#include <openssl/ts.h>
```



```
TS_VERIFY_CTX *TS_VERIFY_CTX_new(void)
        {
        TS_VERIFY_CTX *ctx =
                (TS_VERIFY_CTX *) OPENSSL_malloc(sizeof(TS_VERIFY_CTX));
        if (ctx)
                memset(ctx, 0, sizeof(TS_VERIFY_CTX));
        else
                TSerr(TS_F_TS_VERIFY_CTX_NEW, ERR_R_MALLOC_FAILURE);
        return ctx;
        }

void TS_VERIFY_CTX_init(TS_VERIFY_CTX *ctx)
        {
        OPENSSL_assert(ctx != NULL);
        memset(ctx, 0, sizeof(TS_VERIFY_CTX));
        }

void TS_VERIFY_CTX_free(TS_VERIFY_CTX *ctx)
        {
        if (!ctx) return;

        TS_VERIFY_CTX_cleanup(ctx);
        OPENSSL_free(ctx);
        }

void TS_VERIFY_CTX_cleanup(TS_VERIFY_CTX *ctx)
        {
        if (!ctx) return;

        X509_STORE_free(ctx->store);
        sk_X509_pop_free(ctx->certs, X509_free);

        ASN1_OBJECT_free(ctx->policy);

        X509_ALGOR_free(ctx->md_alg);
        OPENSSL_free(ctx->imprint);

        BIO_free_all(ctx->data);

        ASN1_INTEGER_free(ctx->nonce);

        GENERAL_NAME_free(ctx->tsa_name);

        TS_VERIFY_CTX_init(ctx);
        }

TS_VERIFY_CTX *TS_REQ_to_TS_VERIFY_CTX(TS_REQ *req, TS_VERIFY_CTX *ctx)
        {
        TS_VERIFY_CTX *ret = ctx;
        ASN1_OBJECT *policy;
        TS_MSG_IMPRINT *imprint;
        X509_ALGOR *md_alg;
        ASN1_OCTET_STRING *msg;
        const ASN1_INTEGER *nonce;

        OPENSSL_assert(req != NULL);
        if (ret)
                TS_VERIFY_CTX_cleanup(ret);
        else
                if (!(ret = TS_VERIFY_CTX_new())) return NULL;

        /* Setting flags. */
        ret->flags = TS_VFY_ALL_IMPRINT & ~(TS_VFY_TSA_NAME | TS_VFY_SIGNATURE);

        /* Setting policy. */
```

```
128         if ((policy = TS_REQ_get_policy_id(req)) != NULL)
129                 {
130                 if (!(ret->policy = OBJ_dup(policy))) goto err;
131                 }
132         else
133                 ret->flags &= ~TS_VFY_POLICY;

135         /* Setting md_alg, imprint and imprint_len. */
136         imprint = TS_REQ_get_msg_imprint(req);
137         md_alg = TS_MSG_IMPRINT_get_algo(imprint);
138         if (!(ret->md_alg = X509_ALGOR_dup(md_alg))) goto err;
139         msg = TS_MSG_IMPRINT_get_msg(imprint);
140         ret->imprint_len = ASN1_STRING_length(msg);
141         if (!(ret->imprint = OPENSSL_malloc(ret->imprint_len))) goto err;
142         memcpy(ret->imprint, ASN1_STRING_data(msg), ret->imprint_len);

144         /* Setting nonce. */
145         if ((nonce = TS_REQ_get_nonce(req)) != NULL)
146                 {
147                 if (!(ret->nonce = ASN1_INTEGER_dup(nonce))) goto err;
148                 }
149         else
150                 ret->flags &= ~TS_VFY_NONCE;

152         return ret;
153  err:
154         if (ctx)
155                 TS_VERIFY_CTX_cleanup(ctx);
156         else
157                 TS_VERIFY_CTX_free(ret);
158         return NULL;
159         }
160 #endif /* ! codereview */
```

```
**********************************************************
    9959 Wed Aug 13 19:53:22 2014
new/usr/src/lib/openssl/libsunw_crypto/txt_db/txt_db.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/txt_db/txt_db.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```



```
#include "cryptlib.h"
#include <openssl/buffer.h>
#include <openssl/txt_db.h>

#undef BUFSIZE
#define BUFSIZE 512

const char TXT_DB_version[]="TXT_DB" OPENSSL_VERSION_PTEXT;

TXT_DB *TXT_DB_read(BIO *in, int num)
        {
        TXT_DB *ret=NULL;
        int er=1;
        int esc=0;
        long ln=0;
        int i,add,n;
        int size=BUFSIZE;
        int offset=0;
        char *p,*f;
        OPENSSL_STRING *pp;
        BUF_MEM *buf=NULL;

        if ((buf=BUF_MEM_new()) == NULL) goto err;
        if (!BUF_MEM_grow(buf,size)) goto err;

        if ((ret=OPENSSL_malloc(sizeof(TXT_DB))) == NULL)
                goto err;
        ret->num_fields=num;
        ret->index=NULL;
        ret->qual=NULL;
        if ((ret->data=sk_OPENSSL_PSTRING_new_null()) == NULL)
                goto err;
        if ((ret->index=OPENSSL_malloc(sizeof(*ret->index)*num)) == NULL)
                goto err;
        if ((ret->qual=OPENSSL_malloc(sizeof(*(ret->qual))*num)) == NULL)
                goto err;
        for (i=0; i<num; i++)
                {
                ret->index[i]=NULL;
                ret->qual[i]=NULL;
                }

        add=(num+1)*sizeof(char *);
        buf->data[size-1]='\0';
        offset=0;
        for (;;)
                {
                if (offset != 0)
                        {
                        size+=BUFSIZE;
                        if (!BUF_MEM_grow_clean(buf,size)) goto err;
                        }
                buf->data[offset]='\0';
                BIO_gets(in,&(buf->data[offset]),size-offset);
                ln++;
                if (buf->data[offset] == '\0') break;
                if ((offset == 0) && (buf->data[0] == '#')) continue;
                i=strlen(&(buf->data[offset]));
                offset+=i;
                if (buf->data[offset-1] != '\n')
                        continue;
                else
                        {
                        buf->data[offset-1]='\0'; /* blat the '\n' */
                        if (!(p=OPENSSL_malloc(add+offset))) goto err;
                        offset=0;
```

```
128                     }
129                 pp=(char **)p;
130                 p+=add;
131                 n=0;
132                 pp[n++]=p;
133                 i=0;
134                 f=buf->data;
135
136                 esc=0;
137                 for (;;)
138                         {
139                         if (*f == '\0') break;
140                         if (*f == '\t')
141                                 {
142                                 if (esc)
143                                         p--;
144                                 else
145                                         {
146                                         *(p++)='\0';
147                                         f++;
148                                         if (n >=  num) break;
149                                         pp[n++]=p;
150                                         continue;
151                                         }
152                                 }
153                         esc=(*f == '\\');
154                         *(p++)= *(f++);
155                         }
156                 *(p++)='\0';
157                 if ((n != num) || (*f != '\0'))
158                         {
159 #if !defined(OPENSSL_NO_STDIO) && !defined(OPENSSL_SYS_WIN16)   /* temporary fix
160                         fprintf(stderr,"wrong number of fields on line %ld (look
161 #endif
162                         er=2;
163                         goto err;
164                         }
165                 pp[n]=p;
166                 if (!sk_OPENSSL_PSTRING_push(ret->data,pp))
167                         {
168 #if !defined(OPENSSL_NO_STDIO) && !defined(OPENSSL_SYS_WIN16)   /* temporary fix
169                         fprintf(stderr,"failure in sk_push\n");
170 #endif
171                         er=2;
172                         goto err;
173                         }
174                 }
175         er=0;
176 err:
177         BUF_MEM_free(buf);
178         if (er)
179                 {
180 #if !defined(OPENSSL_NO_STDIO) && !defined(OPENSSL_SYS_WIN16)
181                 if (er == 1) fprintf(stderr,"OPENSSL_malloc failure\n");
182 #endif
183                 if (ret != NULL)
184                         {
185                         if (ret->data != NULL) sk_OPENSSL_PSTRING_free(ret->data
186                         if (ret->index != NULL) OPENSSL_free(ret->index);
187                         if (ret->qual != NULL) OPENSSL_free(ret->qual);
188                         if (ret != NULL) OPENSSL_free(ret);
189                         }
190                 return(NULL);
191                 }
192         else
193                 return(ret);
```



```
194         }
195
196 OPENSSL_STRING *TXT_DB_get_by_index(TXT_DB *db, int idx, OPENSSL_STRING *value)
197         {
198         OPENSSL_STRING *ret;
199         LHASH_OF(OPENSSL_STRING) *lh;
200
201         if (idx >= db->num_fields)
202                 {
203                 db->error=DB_ERROR_INDEX_OUT_OF_RANGE;
204                 return(NULL);
205                 }
206         lh=db->index[idx];
207         if (lh == NULL)
208                 {
209                 db->error=DB_ERROR_NO_INDEX;
210                 return(NULL);
211                 }
212         ret=lh_OPENSSL_STRING_retrieve(lh,value);
213         db->error=DB_ERROR_OK;
214         return(ret);
215         }
216
217 int TXT_DB_create_index(TXT_DB *db, int field, int (*qual)(OPENSSL_STRING *),
218                         LHASH_HASH_FN_TYPE hash, LHASH_COMP_FN_TYPE cmp)
219         {
220         LHASH_OF(OPENSSL_STRING) *idx;
221         OPENSSL_STRING *r;
222         int i,n;
223
224         if (field >= db->num_fields)
225                 {
226                 db->error=DB_ERROR_INDEX_OUT_OF_RANGE;
227                 return(0);
228                 }
229         /* FIXME: we lose type checking at this point */
230         if ((idx=(LHASH_OF(OPENSSL_STRING) *)lh_new(hash,cmp)) == NULL)
231                 {
232                 db->error=DB_ERROR_MALLOC;
233                 return(0);
234                 }
235         n=sk_OPENSSL_PSTRING_num(db->data);
236         for (i=0; i<n; i++)
237                 {
238                 r=sk_OPENSSL_PSTRING_value(db->data,i);
239                 if ((qual != NULL) && (qual(r) == 0)) continue;
240                 if ((r=lh_OPENSSL_STRING_insert(idx,r)) != NULL)
241                         {
242                         db->error=DB_ERROR_INDEX_CLASH;
243                         db->arg1=sk_OPENSSL_PSTRING_find(db->data,r);
244                         db->arg2=i;
245                         lh_OPENSSL_STRING_free(idx);
246                         return(0);
247                         }
248                 }
249         if (db->index[field] != NULL) lh_OPENSSL_STRING_free(db->index[field]);
250         db->index[field]=idx;
251         db->qual[field]=qual;
252         return(1);
253         }
254
255 long TXT_DB_write(BIO *out, TXT_DB *db)
256         {
257         long i,j,n,nn,l,tot=0;
258         char *p,**pp,*f;
259         BUF_MEM *buf=NULL;
```

```
260         long ret= -1;

262         if ((buf=BUF_MEM_new()) == NULL)
263                 goto err;
264         n=sk_OPENSSL_PSTRING_num(db->data);
265         nn=db->num_fields;
266         for (i=0; i<n; i++)
267                 {
268                 pp=sk_OPENSSL_PSTRING_value(db->data,i);

270                 l=0;
271                 for (j=0; j<nn; j++)
272                         {
273                         if (pp[j] != NULL)
274                                 l+=strlen(pp[j]);
275                         }
276                 if (!BUF_MEM_grow_clean(buf,(int)(l*2+nn))) goto err;

278                 p=buf->data;
279                 for (j=0; j<nn; j++)
280                         {
281                         f=pp[j];
282                         if (f != NULL)
283                                 for (;;)
284                                         {
285                                         if (*f == '\0') break;
286                                         if (*f == '\t') *(p++)='\\';
287                                         *(p++)= *(f++);
288                                         }
289                         *(p++)='\t';
290                         }
291                 p[-1]='\n';
292                 j=p-buf->data;
293                 if (BIO_write(out,buf->data,(int)j) != j)
294                         goto err;
295                 tot+=j;
296                 }
297         ret=tot;
298 err:
299         if (buf != NULL) BUF_MEM_free(buf);
300         return(ret);
301         }

303 int TXT_DB_insert(TXT_DB *db, OPENSSL_STRING *row)
304         {
305         int i;
306         OPENSSL_STRING *r;

308         for (i=0; i<db->num_fields; i++)
309                 {
310                 if (db->index[i] != NULL)
311                         {
312                         if ((db->qual[i] != NULL) &&
313                                 (db->qual[i](row) == 0)) continue;
314                         r=lh_OPENSSL_STRING_retrieve(db->index[i],row);
315                         if (r != NULL)
316                                 {
317                                 db->error=DB_ERROR_INDEX_CLASH;
318                                 db->arg1=i;
319                                 db->arg_row=r;
320                                 goto err;
321                                 }
322                         }
323                 }
324         /* We have passed the index checks, now just append and insert */
325         if (!sk_OPENSSL_PSTRING_push(db->data,row))
```



```
326                 {
327                 db->error=DB_ERROR_MALLOC;
328                 goto err;
329                 }

331         for (i=0; i<db->num_fields; i++)
332                 {
333                 if (db->index[i] != NULL)
334                         {
335                         if ((db->qual[i] != NULL) &&
336                                 (db->qual[i](row) == 0)) continue;
337                         (void)lh_OPENSSL_STRING_insert(db->index[i],row);
338                         }
339                 }
340         return(1);
341 err:
342         return(0);
343         }

345 void TXT_DB_free(TXT_DB *db)
346         {
347         int i,n;
348         char **p,*max;

350         if(db == NULL)
351             return;

353         if (db->index != NULL)
354                 {
355                 for (i=db->num_fields-1; i>=0; i--)
356                         if (db->index[i] != NULL) lh_OPENSSL_STRING_free(db->ind
357                 OPENSSL_free(db->index);
358                 }
359         if (db->qual != NULL)
360                 OPENSSL_free(db->qual);
361         if (db->data != NULL)
362                 {
363                 for (i=sk_OPENSSL_PSTRING_num(db->data)-1; i>=0; i--)
364                         {
365                         /* check if any 'fields' have been allocated
366                          * from outside of the initial block */
367                         p=sk_OPENSSL_PSTRING_value(db->data,i);
368                         max=p[db->num_fields]; /* last address */
369                         if (max == NULL) /* new row */
370                                 {
371                                 for (n=0; n<db->num_fields; n++)
372                                         if (p[n] != NULL) OPENSSL_free(p[n]);
373                                 }
374                         else
375                                 {
376                                 for (n=0; n<db->num_fields; n++)
377                                         {
378                                         if (((p[n] < (char *)p) || (p[n] > max))
379                                                 && (p[n] != NULL))
380                                                 OPENSSL_free(p[n]);
381                                         }
382                                 }
383                         OPENSSL_free(sk_OPENSSL_PSTRING_value(db->data,i));
384                         }
385                 sk_OPENSSL_PSTRING_free(db->data);
386                 }
387         OPENSSL_free(db);
388         }
389 #endif /* ! codereview */
```

```
**********************************************************
    3054 Wed Aug 13 19:53:22 2014
new/usr/src/lib/openssl/libsunw_crypto/ui/ui_compat.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* crypto/ui/ui_compat.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2001-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <string.h>
#include <openssl/ui_compat.h>

int _ossl_old_des_read_pw_string(char *buf,int length,const char *prompt,int ver
        {
        return UI_UTIL_read_pw_string(buf, length, prompt, verify);
```



```
        }

int _ossl_old_des_read_pw(char *buf,char *buff,int size,const char *prompt,int v
        {
        return UI_UTIL_read_pw(buf, buff, size, prompt, verify);
        }
#endif /* ! codereview */
```

```
**********************************************************
    4585 Wed Aug 13 19:53:22 2014
new/usr/src/lib/openssl/libsunw_crypto/ui/ui_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ui/ui_err.c */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/ui.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_UI,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_UI,0,reason)

static ERR_STRING_DATA UI_str_functs[]=
        {
{ERR_FUNC(UI_F_GENERAL_ALLOCATE_BOOLEAN),       "GENERAL_ALLOCATE_BOOLEAN"},
{ERR_FUNC(UI_F_GENERAL_ALLOCATE_PROMPT),        "GENERAL_ALLOCATE_PROMPT"},
{ERR_FUNC(UI_F_GENERAL_ALLOCATE_STRING),        "GENERAL_ALLOCATE_STRING"},
{ERR_FUNC(UI_F_UI_CTRL),        "UI_ctrl"},
{ERR_FUNC(UI_F_UI_DUP_ERROR_STRING),    "UI_dup_error_string"},
{ERR_FUNC(UI_F_UI_DUP_INFO_STRING),     "UI_dup_info_string"},
{ERR_FUNC(UI_F_UI_DUP_INPUT_BOOLEAN),   "UI_dup_input_boolean"},
{ERR_FUNC(UI_F_UI_DUP_INPUT_STRING),    "UI_dup_input_string"},
{ERR_FUNC(UI_F_UI_DUP_VERIFY_STRING),   "UI_dup_verify_string"},
{ERR_FUNC(UI_F_UI_GET0_RESULT), "UI_get0_result"},
{ERR_FUNC(UI_F_UI_NEW_METHOD),  "UI_new_method"},
{ERR_FUNC(UI_F_UI_SET_RESULT),  "UI_set_result"},
{0,NULL}
        };

static ERR_STRING_DATA UI_str_reasons[]=
        {
{ERR_REASON(UI_R_COMMON_OK_AND_CANCEL_CHARACTERS),"common ok and cancel characte
{ERR_REASON(UI_R_INDEX_TOO_LARGE)        ,"index too large"},
{ERR_REASON(UI_R_INDEX_TOO_SMALL)        ,"index too small"},
{ERR_REASON(UI_R_NO_RESULT_BUFFER)       ,"no result buffer"},
{ERR_REASON(UI_R_RESULT_TOO_LARGE)       ,"result too large"},
{ERR_REASON(UI_R_RESULT_TOO_SMALL)       ,"result too small"},
{ERR_REASON(UI_R_UNKNOWN_CONTROL_COMMAND),"unknown control command"},
{0,NULL}
        };

#endif

void ERR_load_UI_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(UI_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,UI_str_functs);
                ERR_load_strings(0,UI_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
   20580 Wed Aug 13 19:53:22 2014
new/usr/src/lib/openssl/libsunw_crypto/ui/ui_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ui/ui_lib.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <string.h>
#include "cryptlib.h"
#include <openssl/e_os2.h>
```



```
#include <openssl/buffer.h>
#include <openssl/ui.h>
#include <openssl/err.h>
#include "ui_locl.h"

IMPLEMENT_STACK_OF(UI_STRING_ST)

static const UI_METHOD *default_UI_meth=NULL;

UI *UI_new(void)
        {
        return(UI_new_method(NULL));
        }

UI *UI_new_method(const UI_METHOD *method)
        {
        UI *ret;

        ret=(UI *)OPENSSL_malloc(sizeof(UI));
        if (ret == NULL)
                {
                UIerr(UI_F_UI_NEW_METHOD,ERR_R_MALLOC_FAILURE);
                return NULL;
                }
        if (method == NULL)
                ret->meth=UI_get_default_method();
        else
                ret->meth=method;

        ret->strings=NULL;
        ret->user_data=NULL;
        ret->flags=0;
        CRYPTO_new_ex_data(CRYPTO_EX_INDEX_UI, ret, &ret->ex_data);
        return ret;
        }

static void free_string(UI_STRING *uis)
        {
        if (uis->flags & OUT_STRING_FREEABLE)
                {
                OPENSSL_free((char *)uis->out_string);
                switch(uis->type)
                        {
                case UIT_BOOLEAN:
                        OPENSSL_free((char *)uis->_.boolean_data.action_desc);
                        OPENSSL_free((char *)uis->_.boolean_data.ok_chars);
                        OPENSSL_free((char *)uis->_.boolean_data.cancel_chars);
                        break;
                default:
                        break;
                        }
                }
        OPENSSL_free(uis);
        }

void UI_free(UI *ui)
        {
        if (ui == NULL)
                return;
        sk_UI_STRING_pop_free(ui->strings,free_string);
        CRYPTO_free_ex_data(CRYPTO_EX_INDEX_UI, ui, &ui->ex_data);
        OPENSSL_free(ui);
        }

static int allocate_string_stack(UI *ui)
        {
```

```
128         if (ui->strings == NULL)
129                 {
130                 ui->strings=sk_UI_STRING_new_null();
131                 if (ui->strings == NULL)
132                         {
133                         return -1;
134                         }
135                 }
136         return 0;
137         }

139 static UI_STRING *general_allocate_prompt(UI *ui, const char *prompt,
140         int prompt_freeable, enum UI_string_types type, int input_flags,
141         char *result_buf)
142         {
143         UI_STRING *ret = NULL;

145         if (prompt == NULL)
146                 {
147                 UIerr(UI_F_GENERAL_ALLOCATE_PROMPT,ERR_R_PASSED_NULL_PARAMETER);
148                 }
149         else if ((type == UIT_PROMPT || type == UIT_VERIFY
150                         || type == UIT_BOOLEAN) && result_buf == NULL)
151                 {
152                 UIerr(UI_F_GENERAL_ALLOCATE_PROMPT,UI_R_NO_RESULT_BUFFER);
153                 }
154         else if ((ret = (UI_STRING *)OPENSSL_malloc(sizeof(UI_STRING))))
155                 {
156                 ret->out_string=prompt;
157                 ret->flags=prompt_freeable ? OUT_STRING_FREEABLE : 0;
158                 ret->input_flags=input_flags;
159                 ret->type=type;
160                 ret->result_buf=result_buf;
161                 }
162         return ret;
163         }

165 static int general_allocate_string(UI *ui, const char *prompt,
166         int prompt_freeable, enum UI_string_types type, int input_flags,
167         char *result_buf, int minsize, int maxsize, const char *test_buf)
168         {
169         int ret = -1;
170         UI_STRING *s = general_allocate_prompt(ui, prompt, prompt_freeable,
171                 type, input_flags, result_buf);

173         if (s)
174                 {
175                 if (allocate_string_stack(ui) >= 0)
176                         {
177                         s->_.string_data.result_minsize=minsize;
178                         s->_.string_data.result_maxsize=maxsize;
179                         s->_.string_data.test_buf=test_buf;
180                         ret=sk_UI_STRING_push(ui->strings, s);
181                         /* sk_push() returns 0 on error.  Let's addapt that */
182                         if (ret <= 0) ret--;
183                         }
184                 else
185                         free_string(s);
186                 }
187         return ret;
188         }

190 static int general_allocate_boolean(UI *ui,
191         const char *prompt, const char *action_desc,
192         const char *ok_chars, const char *cancel_chars,
193         int prompt_freeable, enum UI_string_types type, int input_flags,
```



```
194         char *result_buf)
195         {
196         int ret = -1;
197         UI_STRING *s;
198         const char *p;

200         if (ok_chars == NULL)
201                 {
202                 UIerr(UI_F_GENERAL_ALLOCATE_BOOLEAN,ERR_R_PASSED_NULL_PARAMETER)
203                 }
204         else if (cancel_chars == NULL)
205                 {
206                 UIerr(UI_F_GENERAL_ALLOCATE_BOOLEAN,ERR_R_PASSED_NULL_PARAMETER)
207                 }
208         else
209                 {
210                 for(p = ok_chars; *p; p++)
211                         {
212                         if (strchr(cancel_chars, *p))
213                                 {
214                                 UIerr(UI_F_GENERAL_ALLOCATE_BOOLEAN,
215                                         UI_R_COMMON_OK_AND_CANCEL_CHARACTERS);
216                                 }
217                         }

219                 s = general_allocate_prompt(ui, prompt, prompt_freeable,
220                         type, input_flags, result_buf);

222                 if (s)
223                         {
224                         if (allocate_string_stack(ui) >= 0)
225                                 {
226                                 s->_.boolean_data.action_desc = action_desc;
227                                 s->_.boolean_data.ok_chars = ok_chars;
228                                 s->_.boolean_data.cancel_chars = cancel_chars;
229                                 ret=sk_UI_STRING_push(ui->strings, s);
230                                 /* sk_push() returns 0 on error.
231                                    Let's addapt that */
232                                 if (ret <= 0) ret--;
233                                 }
234                         else
235                                 free_string(s);
236                         }
237                 }
238         return ret;
239         }

241 /* Returns the index to the place in the stack or -1 for error.  Uses a
242    direct reference to the prompt.  */
243 int UI_add_input_string(UI *ui, const char *prompt, int flags,
244         char *result_buf, int minsize, int maxsize)
245         {
246         return general_allocate_string(ui, prompt, 0,
247                 UIT_PROMPT, flags, result_buf, minsize, maxsize, NULL);
248         }

250 /* Same as UI_add_input_string(), excepts it takes a copy of the prompt */
251 int UI_dup_input_string(UI *ui, const char *prompt, int flags,
252         char *result_buf, int minsize, int maxsize)
253         {
254         char *prompt_copy=NULL;

256         if (prompt)
257                 {
258                 prompt_copy=BUF_strdup(prompt);
259                 if (prompt_copy == NULL)
```

```
260                         {
261                         UIerr(UI_F_UI_DUP_INPUT_STRING,ERR_R_MALLOC_FAILURE);
262                         return 0;
263                         }
264                 }

266         return general_allocate_string(ui, prompt_copy, 1,
267                 UIT_PROMPT, flags, result_buf, minsize, maxsize, NULL);
268         }

270 int UI_add_verify_string(UI *ui, const char *prompt, int flags,
271         char *result_buf, int minsize, int maxsize, const char *test_buf)
272         {
273         return general_allocate_string(ui, prompt, 0,
274                 UIT_VERIFY, flags, result_buf, minsize, maxsize, test_buf);
275         }

277 int UI_dup_verify_string(UI *ui, const char *prompt, int flags,
278         char *result_buf, int minsize, int maxsize, const char *test_buf)
279         {
280         char *prompt_copy=NULL;

282         if (prompt)
283                 {
284                 prompt_copy=BUF_strdup(prompt);
285                 if (prompt_copy == NULL)
286                         {
287                         UIerr(UI_F_UI_DUP_VERIFY_STRING,ERR_R_MALLOC_FAILURE);
288                         return -1;
289                         }
290                 }

292         return general_allocate_string(ui, prompt_copy, 1,
293                 UIT_VERIFY, flags, result_buf, minsize, maxsize, test_buf);
294         }

296 int UI_add_input_boolean(UI *ui, const char *prompt, const char *action_desc,
297         const char *ok_chars, const char *cancel_chars,
298         int flags, char *result_buf)
299         {
300         return general_allocate_boolean(ui, prompt, action_desc,
301                 ok_chars, cancel_chars, 0, UIT_BOOLEAN, flags, result_buf);
302         }

304 int UI_dup_input_boolean(UI *ui, const char *prompt, const char *action_desc,
305         const char *ok_chars, const char *cancel_chars,
306         int flags, char *result_buf)
307         {
308         char *prompt_copy = NULL;
309         char *action_desc_copy = NULL;
310         char *ok_chars_copy = NULL;
311         char *cancel_chars_copy = NULL;

313         if (prompt)
314                 {
315                 prompt_copy=BUF_strdup(prompt);
316                 if (prompt_copy == NULL)
317                         {
318                         UIerr(UI_F_UI_DUP_INPUT_BOOLEAN,ERR_R_MALLOC_FAILURE);
319                         goto err;
320                         }
321                 }

323         if (action_desc)
324                 {
325                 action_desc_copy=BUF_strdup(action_desc);
```



```
326                 if (action_desc_copy == NULL)
327                         {
328                         UIerr(UI_F_UI_DUP_INPUT_BOOLEAN,ERR_R_MALLOC_FAILURE);
329                         goto err;
330                         }
331                 }

333         if (ok_chars)
334                 {
335                 ok_chars_copy=BUF_strdup(ok_chars);
336                 if (ok_chars_copy == NULL)
337                         {
338                         UIerr(UI_F_UI_DUP_INPUT_BOOLEAN,ERR_R_MALLOC_FAILURE);
339                         goto err;
340                         }
341                 }

343         if (cancel_chars)
344                 {
345                 cancel_chars_copy=BUF_strdup(cancel_chars);
346                 if (cancel_chars_copy == NULL)
347                         {
348                         UIerr(UI_F_UI_DUP_INPUT_BOOLEAN,ERR_R_MALLOC_FAILURE);
349                         goto err;
350                         }
351                 }

353         return general_allocate_boolean(ui, prompt_copy, action_desc_copy,
354                 ok_chars_copy, cancel_chars_copy, 1, UIT_BOOLEAN, flags,
355                 result_buf);
356  err:
357         if (prompt_copy) OPENSSL_free(prompt_copy);
358         if (action_desc_copy) OPENSSL_free(action_desc_copy);
359         if (ok_chars_copy) OPENSSL_free(ok_chars_copy);
360         if (cancel_chars_copy) OPENSSL_free(cancel_chars_copy);
361         return -1;
362         }

364 int UI_add_info_string(UI *ui, const char *text)
365         {
366         return general_allocate_string(ui, text, 0, UIT_INFO, 0, NULL, 0, 0,
367                 NULL);
368         }

370 int UI_dup_info_string(UI *ui, const char *text)
371         {
372         char *text_copy=NULL;

374         if (text)
375                 {
376                 text_copy=BUF_strdup(text);
377                 if (text_copy == NULL)
378                         {
379                         UIerr(UI_F_UI_DUP_INFO_STRING,ERR_R_MALLOC_FAILURE);
380                         return -1;
381                         }
382                 }

384         return general_allocate_string(ui, text_copy, 1, UIT_INFO, 0, NULL,
385                 0, 0, NULL);
386         }

388 int UI_add_error_string(UI *ui, const char *text)
389         {
390         return general_allocate_string(ui, text, 0, UIT_ERROR, 0, NULL, 0, 0,
391                 NULL);
```

```
392         }

394 int UI_dup_error_string(UI *ui, const char *text)
395         {
396         char *text_copy=NULL;

398         if (text)
399                 {
400                 text_copy=BUF_strdup(text);
401                 if (text_copy == NULL)
402                         {
403                         UIerr(UI_F_UI_DUP_ERROR_STRING,ERR_R_MALLOC_FAILURE);
404                         return -1;
405                         }
406                 }
407         return general_allocate_string(ui, text_copy, 1, UIT_ERROR, 0, NULL,
408                 0, 0, NULL);
409         }

411 char *UI_construct_prompt(UI *ui, const char *object_desc,
412         const char *object_name)
413         {
414         char *prompt = NULL;

416         if (ui->meth->ui_construct_prompt)
417                 prompt = ui->meth->ui_construct_prompt(ui,
418                         object_desc, object_name);
419         else
420                 {
421                 char prompt1[] = "Enter ";
422                 char prompt2[] = " for ";
423                 char prompt3[] = ":";
424                 int len = 0;

426                 if (object_desc == NULL)
427                         return NULL;
428                 len = sizeof(prompt1) - 1 + strlen(object_desc);
429                 if (object_name)
430                         len += sizeof(prompt2) - 1 + strlen(object_name);
431                 len += sizeof(prompt3) - 1;

433                 prompt = (char *)OPENSSL_malloc(len + 1);
434                 BUF_strlcpy(prompt, prompt1, len + 1);
435                 BUF_strlcat(prompt, object_desc, len + 1);
436                 if (object_name)
437                         {
438                         BUF_strlcat(prompt, prompt2, len + 1);
439                         BUF_strlcat(prompt, object_name, len + 1);
440                         }
441                 BUF_strlcat(prompt, prompt3, len + 1);
442                 }
443         return prompt;
444         }

446 void *UI_add_user_data(UI *ui, void *user_data)
447         {
448         void *old_data = ui->user_data;
449         ui->user_data = user_data;
450         return old_data;
451         }

453 void *UI_get0_user_data(UI *ui)
454         {
455         return ui->user_data;
456         }
```



```
458 const char *UI_get0_result(UI *ui, int i)
459         {
460         if (i < 0)
461                 {
462                 UIerr(UI_F_UI_GET0_RESULT,UI_R_INDEX_TOO_SMALL);
463                 return NULL;
464                 }
465         if (i >= sk_UI_STRING_num(ui->strings))
466                 {
467                 UIerr(UI_F_UI_GET0_RESULT,UI_R_INDEX_TOO_LARGE);
468                 return NULL;
469                 }
470         return UI_get0_result_string(sk_UI_STRING_value(ui->strings, i));
471         }

473 static int print_error(const char *str, size_t len, UI *ui)
474         {
475         UI_STRING uis;

477         memset(&uis, 0, sizeof(uis));
478         uis.type = UIT_ERROR;
479         uis.out_string = str;

481         if (ui->meth->ui_write_string
482                 && !ui->meth->ui_write_string(ui, &uis))
483                 return -1;
484         return 0;
485         }

487 int UI_process(UI *ui)
488         {
489         int i, ok=0;

491         if (ui->meth->ui_open_session && !ui->meth->ui_open_session(ui))
492                 return -1;

494         if (ui->flags & UI_FLAG_PRINT_ERRORS)
495                 ERR_print_errors_cb(
496                         (int (*)(const char *, size_t, void *))print_error,
497                         (void *)ui);

499         for(i=0; i<sk_UI_STRING_num(ui->strings); i++)
500                 {
501                 if (ui->meth->ui_write_string
502                         && !ui->meth->ui_write_string(ui,
503                                 sk_UI_STRING_value(ui->strings, i)))
504                         {
505                         ok=-1;
506                         goto err;
507                         }
508                 }

510         if (ui->meth->ui_flush)
511                 switch(ui->meth->ui_flush(ui))
512                         {
513                 case -1: /* Interrupt/Cancel/something... */
514                         ok = -2;
515                         goto err;
516                 case 0: /* Errors */
517                         ok = -1;
518                         goto err;
519                 default: /* Success */
520                         ok = 0;
521                         break;
522                         }
```

```
524         for(i=0; i<sk_UI_STRING_num(ui->strings); i++)
525                 {
526                 if (ui->meth->ui_read_string)
527                         {
528                         switch(ui->meth->ui_read_string(ui,
529                                 sk_UI_STRING_value(ui->strings, i)))
530                                 {
531                         case -1: /* Interrupt/Cancel/something... */
532                                 ok = -2;
533                                 goto err;
534                         case 0: /* Errors */
535                                 ok = -1;
536                                 goto err;
537                         default: /* Success */
538                                 ok = 0;
539                                 break;
540                                 }
541                         }
542                 }
543  err:
544         if (ui->meth->ui_close_session && !ui->meth->ui_close_session(ui))
545                 return -1;
546         return ok;
547         }

549 int UI_ctrl(UI *ui, int cmd, long i, void *p, void (*f)(void))
550         {
551         if (ui == NULL)
552                 {
553                 UIerr(UI_F_UI_CTRL,ERR_R_PASSED_NULL_PARAMETER);
554                 return -1;
555                 }
556         switch(cmd)
557                 {
558         case UI_CTRL_PRINT_ERRORS:
559                 {
560                 int save_flag = !!(ui->flags & UI_FLAG_PRINT_ERRORS);
561                 if (i)
562                         ui->flags |= UI_FLAG_PRINT_ERRORS;
563                 else
564                         ui->flags &= ~UI_FLAG_PRINT_ERRORS;
565                 return save_flag;
566                 }
567         case UI_CTRL_IS_REDOABLE:
568                 return !!(ui->flags & UI_FLAG_REDOABLE);
569         default:
570                 break;
571                 }
572         UIerr(UI_F_UI_CTRL,UI_R_UNKNOWN_CONTROL_COMMAND);
573         return -1;
574         }

576 int UI_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
577              CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
578         {
579         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_UI, argl, argp,
580                                 new_func, dup_func, free_func);
581         }

583 int UI_set_ex_data(UI *r, int idx, void *arg)
584         {
585         return(CRYPTO_set_ex_data(&r->ex_data,idx,arg));
586         }

588 void *UI_get_ex_data(UI *r, int idx)
589         {
```



```
590         return(CRYPTO_get_ex_data(&r->ex_data,idx));
591         }

593 void UI_set_default_method(const UI_METHOD *meth)
594         {
595         default_UI_meth=meth;
596         }

598 const UI_METHOD *UI_get_default_method(void)
599         {
600         if (default_UI_meth == NULL)
601                 {
602                 default_UI_meth=UI_OpenSSL();
603                 }
604         return default_UI_meth;
605         }

607 const UI_METHOD *UI_get_method(UI *ui)
608         {
609         return ui->meth;
610         }

612 const UI_METHOD *UI_set_method(UI *ui, const UI_METHOD *meth)
613         {
614         ui->meth=meth;
615         return ui->meth;
616         }


619 UI_METHOD *UI_create_method(char *name)
620         {
621         UI_METHOD *ui_method = (UI_METHOD *)OPENSSL_malloc(sizeof(UI_METHOD));

623         if (ui_method)
624                 {
625                 memset(ui_method, 0, sizeof(*ui_method));
626                 ui_method->name = BUF_strdup(name);
627                 }
628         return ui_method;
629         }

631 /* BIG FSCKING WARNING!!!!  If you use this on a statically allocated method
632    (that is, it hasn't been allocated using UI_create_method(), you deserve
633    anything Murphy can throw at you and more!  You have been warned. */
634 void UI_destroy_method(UI_METHOD *ui_method)
635         {
636         OPENSSL_free(ui_method->name);
637         ui_method->name = NULL;
638         OPENSSL_free(ui_method);
639         }

641 int UI_method_set_opener(UI_METHOD *method, int (*opener)(UI *ui))
642         {
643         if (method)
644                 {
645                 method->ui_open_session = opener;
646                 return 0;
647                 }
648         else
649                 return -1;
650         }

652 int UI_method_set_writer(UI_METHOD *method, int (*writer)(UI *ui, UI_STRING *uis
653         {
654         if (method)
655                 {
```

```
656                 method->ui_write_string = writer;
657                 return 0;
658                 }
659         else
660                 return -1;
661         }

663 int UI_method_set_flusher(UI_METHOD *method, int (*flusher)(UI *ui))
664         {
665         if (method)
666                 {
667                 method->ui_flush = flusher;
668                 return 0;
669                 }
670         else
671                 return -1;
672         }

674 int UI_method_set_reader(UI_METHOD *method, int (*reader)(UI *ui, UI_STRING *uis
675         {
676         if (method)
677                 {
678                 method->ui_read_string = reader;
679                 return 0;
680                 }
681         else
682                 return -1;
683         }

685 int UI_method_set_closer(UI_METHOD *method, int (*closer)(UI *ui))
686         {
687         if (method)
688                 {
689                 method->ui_close_session = closer;
690                 return 0;
691                 }
692         else
693                 return -1;
694         }

696 int UI_method_set_prompt_constructor(UI_METHOD *method, char *(*prompt_construct
697         {
698         if (method)
699                 {
700                 method->ui_construct_prompt = prompt_constructor;
701                 return 0;
702                 }
703         else
704                 return -1;
705         }

707 int (*UI_method_get_opener(UI_METHOD *method))(UI*)
708         {
709         if (method)
710                 return method->ui_open_session;
711         else
712                 return NULL;
713         }

715 int (*UI_method_get_writer(UI_METHOD *method))(UI*,UI_STRING*)
716         {
717         if (method)
718                 return method->ui_write_string;
719         else
720                 return NULL;
721         }
```



```
723 int (*UI_method_get_flusher(UI_METHOD *method))(UI*)
724         {
725         if (method)
726                 return method->ui_flush;
727         else
728                 return NULL;
729         }

731 int (*UI_method_get_reader(UI_METHOD *method))(UI*,UI_STRING*)
732         {
733         if (method)
734                 return method->ui_read_string;
735         else
736                 return NULL;
737         }

739 int (*UI_method_get_closer(UI_METHOD *method))(UI*)
740         {
741         if (method)
742                 return method->ui_close_session;
743         else
744                 return NULL;
745         }

747 char* (*UI_method_get_prompt_constructor(UI_METHOD *method))(UI*, const char*, c
748         {
749         if (method)
750                 return method->ui_construct_prompt;
751         else
752                 return NULL;
753         }

755 enum UI_string_types UI_get_string_type(UI_STRING *uis)
756         {
757         if (!uis)
758                 return UIT_NONE;
759         return uis->type;
760         }

762 int UI_get_input_flags(UI_STRING *uis)
763         {
764         if (!uis)
765                 return 0;
766         return uis->input_flags;
767         }

769 const char *UI_get0_output_string(UI_STRING *uis)
770         {
771         if (!uis)
772                 return NULL;
773         return uis->out_string;
774         }

776 const char *UI_get0_action_string(UI_STRING *uis)
777         {
778         if (!uis)
779                 return NULL;
780         switch(uis->type)
781                 {
782         case UIT_PROMPT:
783         case UIT_BOOLEAN:
784                 return uis->_.boolean_data.action_desc;
785         default:
786                 return NULL;
787                 }
```

```
788         }

790 const char *UI_get0_result_string(UI_STRING *uis)
791         {
792         if (!uis)
793                 return NULL;
794         switch(uis->type)
795                 {
796         case UIT_PROMPT:
797         case UIT_VERIFY:
798                 return uis->result_buf;
799         default:
800                 return NULL;
801                 }
802         }

804 const char *UI_get0_test_string(UI_STRING *uis)
805         {
806         if (!uis)
807                 return NULL;
808         switch(uis->type)
809                 {
810         case UIT_VERIFY:
811                 return uis->_.string_data.test_buf;
812         default:
813                 return NULL;
814                 }
815         }

817 int UI_get_result_minsize(UI_STRING *uis)
818         {
819         if (!uis)
820                 return -1;
821         switch(uis->type)
822                 {
823         case UIT_PROMPT:
824         case UIT_VERIFY:
825                 return uis->_.string_data.result_minsize;
826         default:
827                 return -1;
828                 }
829         }

831 int UI_get_result_maxsize(UI_STRING *uis)
832         {
833         if (!uis)
834                 return -1;
835         switch(uis->type)
836                 {
837         case UIT_PROMPT:
838         case UIT_VERIFY:
839                 return uis->_.string_data.result_maxsize;
840         default:
841                 return -1;
842                 }
843         }

845 int UI_set_result(UI *ui, UI_STRING *uis, const char *result)
846         {
847         int l = strlen(result);

849         ui->flags &= ~UI_FLAG_REDOABLE;

851         if (!uis)
852                 return -1;
853         switch (uis->type)
```



```
854                 {
855         case UIT_PROMPT:
856         case UIT_VERIFY:
857                 {
858                 char number1[DECIMAL_SIZE(uis->_.string_data.result_minsize)+1];
859                 char number2[DECIMAL_SIZE(uis->_.string_data.result_maxsize)+1];

861                 BIO_snprintf(number1, sizeof(number1), "%d",
862                         uis->_.string_data.result_minsize);
863                 BIO_snprintf(number2, sizeof(number2), "%d",
864                         uis->_.string_data.result_maxsize);

866                 if (l < uis->_.string_data.result_minsize)
867                         {
868                         ui->flags |= UI_FLAG_REDOABLE;
869                         UIerr(UI_F_UI_SET_RESULT,UI_R_RESULT_TOO_SMALL);
870                         ERR_add_error_data(5,"You must type in ",
871                                 number1," to ",number2," characters");
872                         return -1;
873                         }
874                 if (l > uis->_.string_data.result_maxsize)
875                         {
876                         ui->flags |= UI_FLAG_REDOABLE;
877                         UIerr(UI_F_UI_SET_RESULT,UI_R_RESULT_TOO_LARGE);
878                         ERR_add_error_data(5,"You must type in ",
879                                 number1," to ",number2," characters");
880                         return -1;
881                         }
882                 }

884                 if (!uis->result_buf)
885                         {
886                         UIerr(UI_F_UI_SET_RESULT,UI_R_NO_RESULT_BUFFER);
887                         return -1;
888                         }

890                 BUF_strlcpy(uis->result_buf, result,
891                         uis->_.string_data.result_maxsize + 1);
892                 break;
893         case UIT_BOOLEAN:
894                 {
895                 const char *p;

897                 if (!uis->result_buf)
898                         {
899                         UIerr(UI_F_UI_SET_RESULT,UI_R_NO_RESULT_BUFFER);
900                         return -1;
901                         }

903                 uis->result_buf[0] = '\0';
904                 for(p = result; *p; p++)
905                         {
906                         if (strchr(uis->_.boolean_data.ok_chars, *p))
907                                 {
908                                 uis->result_buf[0] =
909                                         uis->_.boolean_data.ok_chars[0];
910                                 break;
911                                 }
912                         if (strchr(uis->_.boolean_data.cancel_chars, *p))
913                                 {
914                                 uis->result_buf[0] =
915                                         uis->_.boolean_data.cancel_chars[0];
916                                 break;
917                                 }
918                         }
919                 }
```

```
920         default:
921                 break;
922                 }
923         return 0;
924         }
925 #endif /* ! codereview */
```

```
**********************************************************
   18950 Wed Aug 13 19:53:22 2014
new/usr/src/lib/openssl/libsunw_crypto/ui/ui_openssl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ui/ui_openssl.c -*- mode:C; c-file-style: "eay" -*- */
/* Written by Richard Levitte (richard@levitte.org) and others
 * for the OpenSSL project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* The lowest level part of this file was previously in crypto/des/read_pwd.c,
 * Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
```



```
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */


#include <openssl/e_os2.h>

/* need for #define _POSIX_C_SOURCE arises whenever you pass -ansi to gcc
 * [maybe others?], because it masks interfaces not discussed in standard,
 * sigaction and fileno included. -pedantic would be more appropriate for
 * the intended purposes, but we can't prevent users from adding -ansi.
 */
#if defined(OPENSSL_SYSNAME_VXWORKS)
#include <sys/types.h>
#endif
```

```
#if !defined(_POSIX_C_SOURCE) && defined(OPENSSL_SYS_VMS)
#ifndef _POSIX_C_SOURCE
#define _POSIX_C_SOURCE 2
#endif
#endif
#include <signal.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>

#if !defined(OPENSSL_SYS_MSDOS) && !defined(OPENSSL_SYS_VMS)
# ifdef OPENSSL_UNISTD
#  include OPENSSL_UNISTD
# else
#  include <unistd.h>
# endif
/* If unistd.h defines _POSIX_VERSION, we conclude that we
 * are on a POSIX system and have sigaction and termios. */
# if defined(_POSIX_VERSION)

#  define SIGACTION
#  if !defined(TERMIOS) && !defined(TERMIO) && !defined(SGTTY)
#   define TERMIOS
#  endif

# endif
#endif

#ifdef WIN16TTY
# undef OPENSSL_SYS_WIN16
# undef WIN16
# undef _WINDOWS
# include <graph.h>
#endif

/* 06-Apr-92 Luke Brennan    Support for VMS */
#include "ui_locl.h"
#include "cryptlib.h"

#ifdef OPENSSL_SYS_VMS          /* prototypes for sys$whatever */
# include <starlet.h>
# ifdef __DECC
#  pragma message disable DOLLARID
# endif
#endif

#ifdef WIN_CONSOLE_BUG
# include <windows.h>
#ifndef OPENSSL_SYS_WINCE
# include <wincon.h>
#endif
#endif


/* There are 5 types of terminal interface supported,
 * TERMIO, TERMIOS, VMS, MSDOS and SGTTY
 */

#if defined(__sgi) && !defined(TERMIOS)
# define TERMIOS
# undef  TERMIO
# undef  SGTTY
#endif

#if defined(linux) && !defined(TERMIO)
```



```
# undef  TERMIOS
# define TERMIO
# undef  SGTTY
#endif

#ifdef _LIBC
# undef  TERMIOS
# define TERMIO
# undef  SGTTY
#endif

#if !defined(TERMIO) && !defined(TERMIOS) && !defined(OPENSSL_SYS_VMS) && !defin
# undef  TERMIOS
# undef  TERMIO
# define SGTTY
#endif

#if defined(OPENSSL_SYS_VXWORKS)
#undef TERMIOS
#undef TERMIO
#undef SGTTY
#endif

#if defined(OPENSSL_SYS_NETWARE)
#undef TERMIOS
#undef TERMIO
#undef SGTTY
#endif

#ifdef TERMIOS
# include <termios.h>
# define TTY_STRUCT             struct termios
# define TTY_FLAGS              c_lflag
# define TTY_get(tty,data)      tcgetattr(tty,data)
# define TTY_set(tty,data)      tcsetattr(tty,TCSANOW,data)
#endif

#ifdef TERMIO
# include <termio.h>
# define TTY_STRUCT             struct termio
# define TTY_FLAGS              c_lflag
# define TTY_get(tty,data)      ioctl(tty,TCGETA,data)
# define TTY_set(tty,data)      ioctl(tty,TCSETA,data)
#endif

#ifdef SGTTY
# include <sgtty.h>
# define TTY_STRUCT             struct sgttyb
# define TTY_FLAGS              sg_flags
# define TTY_get(tty,data)      ioctl(tty,TIOCGETP,data)
# define TTY_set(tty,data)      ioctl(tty,TIOCSETP,data)
#endif

#if !defined(_LIBC) && !defined(OPENSSL_SYS_MSDOS) && !defined(OPENSSL_SYS_VMS)
# include <sys/ioctl.h>
#endif

#ifdef OPENSSL_SYS_MSDOS
# include <conio.h>
#endif

#ifdef OPENSSL_SYS_VMS
# include <ssdef.h>
# include <iodef.h>
# include <ttdef.h>
# include <descrip.h>
```

```
260 struct IOSB {
261         short iosb$w_value;
262         short iosb$w_count;
263         long  iosb$l_info;
264         };
265 #endif

267 #ifdef OPENSSL_SYS_SUNOS
268         typedef int sig_atomic_t;
269 #endif

271 #if defined(OPENSSL_SYS_MACINTOSH_CLASSIC) || defined(MAC_OS_GUSI_SOURCE) || def
272 /*
273  * This one needs work. As a matter of fact the code is unoperational
274  * and this is only a trick to get it compiled.
275  *                                      <appro@fy.chalmers.se>
276  */
277 # define TTY_STRUCT int
278 #endif

280 #ifndef NX509_SIG
281 # define NX509_SIG 32
282 #endif


285 /* Define globals.  They are protected by a lock */
286 #ifdef SIGACTION
287 static struct sigaction savsig[NX509_SIG];
288 #else
289 static void (*savsig[NX509_SIG])(int );
290 #endif

292 #ifdef OPENSSL_SYS_VMS
293 static struct IOSB iosb;
294 static $DESCRIPTOR(terminal,"TT");
295 static long tty_orig[3], tty_new[3]; /* XXX   Is there any guarantee that this w
296 static long status;
297 static unsigned short channel = 0;
298 #else
299 #if !defined(OPENSSL_SYS_MSDOS) || defined(__DJGPP__)
300 static TTY_STRUCT tty_orig,tty_new;
301 #endif
302 #endif
303 static FILE *tty_in, *tty_out;
304 static int is_a_tty;

306 /* Declare static functions */
307 #if !defined(OPENSSL_SYS_WIN16) && !defined(OPENSSL_SYS_WINCE)
308 static int read_till_nl(FILE *);
309 static void recsig(int);
310 static void pushsig(void);
311 static void popsig(void);
312 #endif
313 #if defined(OPENSSL_SYS_MSDOS) && !defined(OPENSSL_SYS_WIN16)
314 static int noecho_fgets(char *buf, int size, FILE *tty);
315 #endif
316 static int read_string_inner(UI *ui, UI_STRING *uis, int echo, int strip_nl);

318 static int read_string(UI *ui, UI_STRING *uis);
319 static int write_string(UI *ui, UI_STRING *uis);

321 static int open_console(UI *ui);
322 static int echo_console(UI *ui);
323 static int noecho_console(UI *ui);
324 static int close_console(UI *ui);
```



```
326 static UI_METHOD ui_openssl =
327         {
328         "OpenSSL default user interface",
329         open_console,
330         write_string,
331         NULL,                   /* No flusher is needed for command lines */
332         read_string,
333         close_console,
334         NULL
335         };

337 /* The method with all the built-in thingies */
338 UI_METHOD *UI_OpenSSL(void)
339         {
340         return &ui_openssl;
341         }

343 /* The following function makes sure that info and error strings are printed
344    before any prompt. */
345 static int write_string(UI *ui, UI_STRING *uis)
346         {
347         switch (UI_get_string_type(uis))
348                 {
349         case UIT_ERROR:
350         case UIT_INFO:
351                 fputs(UI_get0_output_string(uis), tty_out);
352                 fflush(tty_out);
353                 break;
354         default:
355                 break;
356                 }
357         return 1;
358         }

360 static int read_string(UI *ui, UI_STRING *uis)
361         {
362         int ok = 0;

364         switch (UI_get_string_type(uis))
365                 {
366         case UIT_BOOLEAN:
367                 fputs(UI_get0_output_string(uis), tty_out);
368                 fputs(UI_get0_action_string(uis), tty_out);
369                 fflush(tty_out);
370                 return read_string_inner(ui, uis,
371                         UI_get_input_flags(uis) & UI_INPUT_FLAG_ECHO, 0);
372         case UIT_PROMPT:
373                 fputs(UI_get0_output_string(uis), tty_out);
374                 fflush(tty_out);
375                 return read_string_inner(ui, uis,
376                         UI_get_input_flags(uis) & UI_INPUT_FLAG_ECHO, 1);
377         case UIT_VERIFY:
378                 fprintf(tty_out,"Verifying - %s",
379                         UI_get0_output_string(uis));
380                 fflush(tty_out);
381                 if ((ok = read_string_inner(ui, uis,
382                         UI_get_input_flags(uis) & UI_INPUT_FLAG_ECHO, 1)) <= 0)
383                         return ok;
384                 if (strcmp(UI_get0_result_string(uis),
385                         UI_get0_test_string(uis)) != 0)
386                         {
387                         fprintf(tty_out,"Verify failure\n");
388                         fflush(tty_out);
389                         return 0;
390                         }
391                 break;
```

```
392         default:
393                 break;
394                 }
395         return 1;
396         }


399 #if !defined(OPENSSL_SYS_WIN16) && !defined(OPENSSL_SYS_WINCE)
400 /* Internal functions to read a string without echoing */
401 static int read_till_nl(FILE *in)
402         {
403 #define SIZE 4
404         char buf[SIZE+1];

406         do      {
407                 if (!fgets(buf,SIZE,in))
408                         return 0;
409                 } while (strchr(buf,'\n') == NULL);
410         return 1;
411         }

413 static volatile sig_atomic_t intr_signal;
414 #endif

416 static int read_string_inner(UI *ui, UI_STRING *uis, int echo, int strip_nl)
417         {
418         static int ps;
419         int ok;
420         char result[BUFSIZ];
421         int maxsize = BUFSIZ-1;
422 #if !defined(OPENSSL_SYS_WIN16) && !defined(OPENSSL_SYS_WINCE)
423         char *p;

425         intr_signal=0;
426         ok=0;
427         ps=0;

429         pushsig();
430         ps=1;

432         if (!echo && !noecho_console(ui))
433                 goto error;
434         ps=2;

436         result[0]='\0';
437 #ifdef OPENSSL_SYS_MSDOS
438         if (!echo)
439                 {
440                 noecho_fgets(result,maxsize,tty_in);
441                 p=result; /* FIXME: noecho_fgets doesn't return errors */
442                 }
443         else
444                 p=fgets(result,maxsize,tty_in);
445 #else
446         p=fgets(result,maxsize,tty_in);
447 #endif
448         if(!p)
449                 goto error;
450         if (feof(tty_in)) goto error;
451         if (ferror(tty_in)) goto error;
452         if ((p=(char *)strchr(result,'\n')) != NULL)
453                 {
454                 if (strip_nl)
455                         *p='\0';
456                 }
457         else
```



```
458                 if (!read_till_nl(tty_in))
459                         goto error;
460         if (UI_set_result(ui, uis, result) >= 0)
461                 ok=1;

463 error:
464         if (intr_signal == SIGINT)
465                 ok=-1;
466         if (!echo) fprintf(tty_out,"\n");
467         if (ps >= 2 && !echo && !echo_console(ui))
468                 ok=0;

470         if (ps >= 1)
471                 popsig();
472 #else
473         ok=1;
474 #endif

476         OPENSSL_cleanse(result,BUFSIZ);
477         return ok;
478         }


481 /* Internal functions to open, handle and close a channel to the console.  */
482 static int open_console(UI *ui)
483         {
484         CRYPTO_w_lock(CRYPTO_LOCK_UI);
485         is_a_tty = 1;

487 #if defined(OPENSSL_SYS_MACINTOSH_CLASSIC) || defined(OPENSSL_SYS_VXWORKS) || de
488         tty_in=stdin;
489         tty_out=stderr;
490 #else
491 #  ifdef OPENSSL_SYS_MSDOS
492 #    define DEV_TTY "con"
493 #  else
494 #    define DEV_TTY "/dev/tty"
495 #  endif
496         if ((tty_in=fopen(DEV_TTY,"r")) == NULL)
497                 tty_in=stdin;
498         if ((tty_out=fopen(DEV_TTY,"w")) == NULL)
499                 tty_out=stderr;
500 #endif

502 #if defined(TTY_get) && !defined(OPENSSL_SYS_VMS)
503         if (TTY_get(fileno(tty_in),&tty_orig) == -1)
504                 {
505 #ifdef ENOTTY
506                 if (errno == ENOTTY)
507                         is_a_tty=0;
508                 else
509 #endif
510 #ifdef EINVAL
511                 /* Ariel Glenn ariel@columbia.edu reports that solaris
512                  * can return EINVAL instead.  This should be ok */
513                 if (errno == EINVAL)
514                         is_a_tty=0;
515                 else
516 #endif
517                         return 0;
518                 }
519 #endif
520 #ifdef OPENSSL_SYS_VMS
521         status = sys$assign(&terminal,&channel,0,0);
522         if (status != SS$_NORMAL)
523                 return 0;
```

```
524         status=sys$qiow(0,channel,IO$_SENSEMODE,&iosb,0,0,tty_orig,12,0,0,0,0);
525         if ((status != SS$_NORMAL) || (iosb.iosb$w_value != SS$_NORMAL))
526                 return 0;
527 #endif
528         return 1;
529         }

531 static int noecho_console(UI *ui)
532         {
533 #ifdef TTY_FLAGS
534         memcpy(&(tty_new),&(tty_orig),sizeof(tty_orig));
535         tty_new.TTY_FLAGS &= ~ECHO;
536 #endif

538 #if defined(TTY_set) && !defined(OPENSSL_SYS_VMS)
539         if (is_a_tty && (TTY_set(fileno(tty_in),&tty_new) == -1))
540                 return 0;
541 #endif
542 #ifdef OPENSSL_SYS_VMS
543         tty_new[0] = tty_orig[0];
544         tty_new[1] = tty_orig[1] | TT$M_NOECHO;
545         tty_new[2] = tty_orig[2];
546         status = sys$qiow(0,channel,IO$_SETMODE,&iosb,0,0,tty_new,12,0,0,0,0);
547         if ((status != SS$_NORMAL) || (iosb.iosb$w_value != SS$_NORMAL))
548                 return 0;
549 #endif
550         return 1;
551         }

553 static int echo_console(UI *ui)
554         {
555 #if defined(TTY_set) && !defined(OPENSSL_SYS_VMS)
556         memcpy(&(tty_new),&(tty_orig),sizeof(tty_orig));
557         tty_new.TTY_FLAGS |= ECHO;
558 #endif

560 #if defined(TTY_set) && !defined(OPENSSL_SYS_VMS)
561         if (is_a_tty && (TTY_set(fileno(tty_in),&tty_new) == -1))
562                 return 0;
563 #endif
564 #ifdef OPENSSL_SYS_VMS
565         tty_new[0] = tty_orig[0];
566         tty_new[1] = tty_orig[1] & ~TT$M_NOECHO;
567         tty_new[2] = tty_orig[2];
568         status = sys$qiow(0,channel,IO$_SETMODE,&iosb,0,0,tty_new,12,0,0,0,0);
569         if ((status != SS$_NORMAL) || (iosb.iosb$w_value != SS$_NORMAL))
570                 return 0;
571 #endif
572         return 1;
573         }

575 static int close_console(UI *ui)
576         {
577         if (tty_in != stdin) fclose(tty_in);
578         if (tty_out != stderr) fclose(tty_out);
579 #ifdef OPENSSL_SYS_VMS
580         status = sys$dassgn(channel);
581 #endif
582         CRYPTO_w_unlock(CRYPTO_LOCK_UI);

584         return 1;
585         }


588 #if !defined(OPENSSL_SYS_WIN16) && !defined(OPENSSL_SYS_WINCE)
589 /* Internal functions to handle signals and act on them */
```



```
590 static void pushsig(void)
591         {
592 #ifndef OPENSSL_SYS_WIN32
593         int i;
594 #endif
595 #ifdef SIGACTION
596         struct sigaction sa;

598         memset(&sa,0,sizeof sa);
599         sa.sa_handler=recsig;
600 #endif

602 #ifdef OPENSSL_SYS_WIN32
603         savsig[SIGABRT]=signal(SIGABRT,recsig);
604         savsig[SIGFPE]=signal(SIGFPE,recsig);
605         savsig[SIGILL]=signal(SIGILL,recsig);
606         savsig[SIGINT]=signal(SIGINT,recsig);
607         savsig[SIGSEGV]=signal(SIGSEGV,recsig);
608         savsig[SIGTERM]=signal(SIGTERM,recsig);
609 #else
610         for (i=1; i<NX509_SIG; i++)
611                 {
612 #ifdef SIGUSR1
613                 if (i == SIGUSR1)
614                         continue;
615 #endif
616 #ifdef SIGUSR2
617                 if (i == SIGUSR2)
618                         continue;
619 #endif
620 #ifdef SIGKILL
621                 if (i == SIGKILL) /* We can't make any action on that. */
622                         continue;
623 #endif
624 #ifdef SIGACTION
625                 sigaction(i,&sa,&savsig[i]);
626 #else
627                 savsig[i]=signal(i,recsig);
628 #endif
629                 }
630 #endif

632 #ifdef SIGWINCH
633         signal(SIGWINCH,SIG_DFL);
634 #endif
635         }

637 static void popsig(void)
638         {
639 #ifdef OPENSSL_SYS_WIN32
640         signal(SIGABRT,savsig[SIGABRT]);
641         signal(SIGFPE,savsig[SIGFPE]);
642         signal(SIGILL,savsig[SIGILL]);
643         signal(SIGINT,savsig[SIGINT]);
644         signal(SIGSEGV,savsig[SIGSEGV]);
645         signal(SIGTERM,savsig[SIGTERM]);
646 #else
647         int i;
648         for (i=1; i<NX509_SIG; i++)
649                 {
650 #ifdef SIGUSR1
651                 if (i == SIGUSR1)
652                         continue;
653 #endif
654 #ifdef SIGUSR2
655                 if (i == SIGUSR2)
```

```
656                         continue;
657 #endif
658 #ifdef SIGACTION
659                 sigaction(i,&savsig[i],NULL);
660 #else
661                 signal(i,savsig[i]);
662 #endif
663                 }
664 #endif
665         }

667 static void recsig(int i)
668         {
669         intr_signal=i;
670         }
671 #endif

673 /* Internal functions specific for Windows */
674 #if defined(OPENSSL_SYS_MSDOS) && !defined(OPENSSL_SYS_WIN16) && !defined(OPENSS
675 static int noecho_fgets(char *buf, int size, FILE *tty)
676         {
677         int i;
678         char *p;

680         p=buf;
681         for (;;)
682                 {
683                 if (size == 0)
684                         {
685                         *p='\0';
686                         break;
687                         }
688                 size--;
689 #ifdef WIN16TTY
690                 i=_inchar();
691 #elif defined(_WIN32)
692                 i=_getch();
693 #else
694                 i=getch();
695 #endif
696                 if (i == '\r') i='\n';
697                 *(p++)=i;
698                 if (i == '\n')
699                         {
700                         *p='\0';
701                         break;
702                         }
703                 }
704 #ifdef WIN_CONSOLE_BUG
705 /* Win95 has several evil console bugs: one of these is that the
706  * last character read using getch() is passed to the next read: this is
707  * usually a CR so this can be trouble. No STDIO fix seems to work but
708  * flushing the console appears to do the trick.
709  */
710                 {
711                         HANDLE inh;
712                         inh = GetStdHandle(STD_INPUT_HANDLE);
713                         FlushConsoleInputBuffer(inh);
714                 }
715 #endif
716         return(strlen(buf));
717         }
718 #endif
719 #endif /* ! codereview */
```

```
**********************************************************
    3381 Wed Aug 13 19:53:23 2014
new/usr/src/lib/openssl/libsunw_crypto/ui/ui_util.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/ui/ui_util.c -*- mode:C; c-file-style: "eay" -*- */
/* ====================================================================
 * Copyright (c) 2001-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <string.h>
#include "ui_locl.h"

int UI_UTIL_read_pw_string(char *buf,int length,const char *prompt,int verify)
        {
        char buff[BUFSIZ];
```



```
        int ret;

        ret=UI_UTIL_read_pw(buf,buff,(length>BUFSIZ)?BUFSIZ:length,prompt,verify
        OPENSSL_cleanse(buff,BUFSIZ);
        return(ret);
        }

int UI_UTIL_read_pw(char *buf,char *buff,int size,const char *prompt,int verify)
        {
        int ok = 0;
        UI *ui;

        if (size < 1)
                return -1;

        ui = UI_new();
        if (ui)
                {
                ok = UI_add_input_string(ui,prompt,0,buf,0,size-1);
                if (ok >= 0 && verify)
                        ok = UI_add_verify_string(ui,prompt,0,buff,0,size-1,
                                buf);
                if (ok >= 0)
                        ok=UI_process(ui);
                UI_free(ui);
                }
        if (ok > 0)
                ok = 0;
        return(ok);
        }
#endif /* ! codereview */
```

```
**********************************************************
    3186 Wed Aug 13 19:53:23 2014
new/usr/src/lib/openssl/libsunw_crypto/uid.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/uid.c */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/crypto.h>
#include <openssl/opensslconf.h>

#if defined(__OpenBSD__) || (defined(__FreeBSD__) && __FreeBSD__ > 2)

#include OPENSSL_UNISTD
```



```
int OPENSSL_issetugid(void)
        {
        return issetugid();
        }

#elif defined(OPENSSL_SYS_WIN32) || defined(OPENSSL_SYS_VXWORKS) || defined(OPEN

int OPENSSL_issetugid(void)
        {
        return 0;
        }

#else

#include OPENSSL_UNISTD
#include <sys/types.h>

int OPENSSL_issetugid(void)
        {
        if (getuid() != geteuid()) return 1;
        if (getgid() != getegid()) return 1;
        return 0;
        }
#endif
#endif /* ! codereview */
```

```
**********************************************************
   11902 Wed Aug 13 19:53:23 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/by_dir.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/by_dir.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
#include <errno.h>
```



```
#include "cryptlib.h"

#ifndef NO_SYS_TYPES_H
# include <sys/types.h>
#endif
#ifndef OPENSSL_NO_POSIX_IO
# include <sys/stat.h>
#endif

#include <openssl/lhash.h>
#include <openssl/x509.h>


typedef struct lookup_dir_hashes_st
        {
        unsigned long hash;
        int suffix;
        } BY_DIR_HASH;

typedef struct lookup_dir_entry_st
        {
        char *dir;
        int dir_type;
        STACK_OF(BY_DIR_HASH) *hashes;
        } BY_DIR_ENTRY;

typedef struct lookup_dir_st
        {
        BUF_MEM *buffer;
        STACK_OF(BY_DIR_ENTRY) *dirs;
        } BY_DIR;

DECLARE_STACK_OF(BY_DIR_HASH)
DECLARE_STACK_OF(BY_DIR_ENTRY)

static int dir_ctrl(X509_LOOKUP *ctx, int cmd, const char *argp, long argl,
        char **ret);
static int new_dir(X509_LOOKUP *lu);
static void free_dir(X509_LOOKUP *lu);
static int add_cert_dir(BY_DIR *ctx,const char *dir,int type);
static int get_cert_by_subject(X509_LOOKUP *xl,int type,X509_NAME *name,
        X509_OBJECT *ret);
X509_LOOKUP_METHOD x509_dir_lookup=
        {
        "Load certs from files in a directory",
        new_dir,                /* new */
        free_dir,               /* free */
        NULL,                   /* init */
        NULL,                   /* shutdown */
        dir_ctrl,               /* ctrl */
        get_cert_by_subject,    /* get_by_subject */
        NULL,                   /* get_by_issuer_serial */
        NULL,                   /* get_by_fingerprint */
        NULL,                   /* get_by_alias */
        };

X509_LOOKUP_METHOD *X509_LOOKUP_hash_dir(void)
        {
        return(&x509_dir_lookup);
        }

static int dir_ctrl(X509_LOOKUP *ctx, int cmd, const char *argp, long argl,
             char **retp)
        {
        int ret=0;
```

```
128         BY_DIR *ld;
129         char *dir = NULL;

131         ld=(BY_DIR *)ctx->method_data;

133         switch (cmd)
134                 {
135         case X509_L_ADD_DIR:
136                 if (argl == X509_FILETYPE_DEFAULT)
137                         {
138                         dir=(char *)getenv(X509_get_default_cert_dir_env());
139                         if (dir)
140                                 ret=add_cert_dir(ld,dir,X509_FILETYPE_PEM);
141                         else
142                                 ret=add_cert_dir(ld,X509_get_default_cert_dir(),
143                                         X509_FILETYPE_PEM);
144                         if (!ret)
145                                 {
146                                 X509err(X509_F_DIR_CTRL,X509_R_LOADING_CERT_DIR)
147                                 }
148                         }
149                 else
150                         ret=add_cert_dir(ld,argp,(int)argl);
151                 break;
152                 }
153         return(ret);
154         }

156 static int new_dir(X509_LOOKUP *lu)
157         {
158         BY_DIR *a;

160         if ((a=(BY_DIR *)OPENSSL_malloc(sizeof(BY_DIR))) == NULL)
161                 return(0);
162         if ((a->buffer=BUF_MEM_new()) == NULL)
163                 {
164                 OPENSSL_free(a);
165                 return(0);
166                 }
167         a->dirs=NULL;
168         lu->method_data=(char *)a;
169         return(1);
170         }

172 static void by_dir_hash_free(BY_DIR_HASH *hash)
173         {
174         OPENSSL_free(hash);
175         }

177 static int by_dir_hash_cmp(const BY_DIR_HASH * const *a,
178                         const BY_DIR_HASH * const *b)
179         {
180         if ((*a)->hash > (*b)->hash)
181                 return 1;
182         if ((*a)->hash < (*b)->hash)
183                 return -1;
184         return 0;
185         }

187 static void by_dir_entry_free(BY_DIR_ENTRY *ent)
188         {
189         if (ent->dir)
190                 OPENSSL_free(ent->dir);
191         if (ent->hashes)
192                 sk_BY_DIR_HASH_pop_free(ent->hashes, by_dir_hash_free);
193         OPENSSL_free(ent);
```



```
194         }

196 static void free_dir(X509_LOOKUP *lu)
197         {
198         BY_DIR *a;

200         a=(BY_DIR *)lu->method_data;
201         if (a->dirs != NULL)
202                 sk_BY_DIR_ENTRY_pop_free(a->dirs, by_dir_entry_free);
203         if (a->buffer != NULL)
204                 BUF_MEM_free(a->buffer);
205         OPENSSL_free(a);
206         }

208 static int add_cert_dir(BY_DIR *ctx, const char *dir, int type)
209         {
210         int j,len;
211         const char *s,*ss,*p;

213         if (dir == NULL || !*dir)
214             {
215             X509err(X509_F_ADD_CERT_DIR,X509_R_INVALID_DIRECTORY);
216             return 0;
217             }

219         s=dir;
220         p=s;
221         do
222                 {
223                 if ((*p == LIST_SEPARATOR_CHAR) || (*p == '\0'))
224                         {
225                         BY_DIR_ENTRY *ent;
226                         ss=s;
227                         s=p+1;
228                         len=(int)(p-ss);
229                         if (len == 0) continue;
230                         for (j=0; j < sk_BY_DIR_ENTRY_num(ctx->dirs); j++)
231                                 {
232                                 ent = sk_BY_DIR_ENTRY_value(ctx->dirs, j);
233                                 if (strlen(ent->dir) == (size_t)len &&
234                                     strncmp(ent->dir,ss,(unsigned int)len) == 0)
235                                         break;
236                                 }
237                         if (j < sk_BY_DIR_ENTRY_num(ctx->dirs))
238                                 continue;
239                         if (ctx->dirs == NULL)
240                                 {
241                                 ctx->dirs = sk_BY_DIR_ENTRY_new_null();
242                                 if (!ctx->dirs)
243                                         {
244                                         X509err(X509_F_ADD_CERT_DIR,ERR_R_MALLOC
245                                         return 0;
246                                         }
247                                 }
248                         ent = OPENSSL_malloc(sizeof(BY_DIR_ENTRY));
249                         if (!ent)
250                                 return 0;
251                         ent->dir_type = type;
252                         ent->hashes = sk_BY_DIR_HASH_new(by_dir_hash_cmp);
253                         ent->dir = OPENSSL_malloc((unsigned int)len+1);
254                         if (!ent->dir || !ent->hashes)
255                                 {
256                                 by_dir_entry_free(ent);
257                                 return 0;
258                                 }
259                         strncpy(ent->dir,ss,(unsigned int)len);
```

```
260                         ent->dir[len] = '\0';
261                         if (!sk_BY_DIR_ENTRY_push(ctx->dirs, ent))
262                                 {
263                                 by_dir_entry_free(ent);
264                                 return 0;
265                                 }
266                         }
267                 } while (*p++ != '\0');
268         return 1;
269         }

271 static int get_cert_by_subject(X509_LOOKUP *xl, int type, X509_NAME *name,
272              X509_OBJECT *ret)
273         {
274         BY_DIR *ctx;
275         union   {
276                 struct  {
277                         X509 st_x509;
278                         X509_CINF st_x509_cinf;
279                         } x509;
280                 struct  {
281                         X509_CRL st_crl;
282                         X509_CRL_INFO st_crl_info;
283                         } crl;
284                 } data;
285         int ok=0;
286         int i,j,k;
287         unsigned long h;
288         BUF_MEM *b=NULL;
289         X509_OBJECT stmp,*tmp;
290         const char *postfix="";

292         if (name == NULL) return(0);

294         stmp.type=type;
295         if (type == X509_LU_X509)
296                 {
297                 data.x509.st_x509.cert_info= &data.x509.st_x509_cinf;
298                 data.x509.st_x509_cinf.subject=name;
299                 stmp.data.x509= &data.x509.st_x509;
300                 postfix="";
301                 }
302         else if (type == X509_LU_CRL)
303                 {
304                 data.crl.st_crl.crl= &data.crl.st_crl_info;
305                 data.crl.st_crl_info.issuer=name;
306                 stmp.data.crl= &data.crl.st_crl;
307                 postfix="r";
308                 }
309         else
310                 {
311                 X509err(X509_F_GET_CERT_BY_SUBJECT,X509_R_WRONG_LOOKUP_TYPE);
312                 goto finish;
313                 }

315         if ((b=BUF_MEM_new()) == NULL)
316                 {
317                 X509err(X509_F_GET_CERT_BY_SUBJECT,ERR_R_BUF_LIB);
318                 goto finish;
319                 }

321         ctx=(BY_DIR *)xl->method_data;

323         h=X509_NAME_hash(name);
324         for (i=0; i < sk_BY_DIR_ENTRY_num(ctx->dirs); i++)
325                 {
```



```
326                 BY_DIR_ENTRY *ent;
327                 int idx;
328                 BY_DIR_HASH htmp, *hent;
329                 ent = sk_BY_DIR_ENTRY_value(ctx->dirs, i);
330                 j=strlen(ent->dir)+1+8+6+1+1;
331                 if (!BUF_MEM_grow(b,j))
332                         {
333                         X509err(X509_F_GET_CERT_BY_SUBJECT,ERR_R_MALLOC_FAILURE)
334                         goto finish;
335                         }
336                 if (type == X509_LU_CRL && ent->hashes)
337                         {
338                         htmp.hash = h;
339                         CRYPTO_r_lock(CRYPTO_LOCK_X509_STORE);
340                         idx = sk_BY_DIR_HASH_find(ent->hashes, &htmp);
341                         if (idx >= 0)
342                                 {
343                                 hent = sk_BY_DIR_HASH_value(ent->hashes, idx);
344                                 k = hent->suffix;
345                                 }
346                         else
347                                 {
348                                 hent = NULL;
349                                 k=0;
350                                 }
351                         CRYPTO_r_unlock(CRYPTO_LOCK_X509_STORE);
352                         }
353                 else
354                         {
355                         k = 0;
356                         hent = NULL;
357                         }
358                 for (;;)
359                         {
360                         char c = '/';
361 #ifdef OPENSSL_SYS_VMS
362                         c = ent->dir[strlen(ent->dir)-1];
363                         if (c != ':' && c != '>' && c != ']')
364                                 {
365                                 /* If no separator is present, we assume the
366                                    directory specifier is a logical name, and
367                                    add a colon.  We really should use better
368                                    VMS routines for merging things like this,
369                                    but this will do for now...
370                                    -- Richard Levitte */
371                                 c = ':';
372                                 }
373                         else
374                                 {
375                                 c = '\0';
376                                 }
377 #endif
378                         if (c == '\0')
379                                 {
380                                 /* This is special.  When c == '\0', no
381                                    directory separator should be added. */
382                                 BIO_snprintf(b->data,b->max,
383                                         "%s%08lx.%s%d",ent->dir,h,
384                                         postfix,k);
385                                 }
386                         else
387                                 {
388                                 BIO_snprintf(b->data,b->max,
389                                         "%s%c%08lx.%s%d",ent->dir,c,h,
390                                         postfix,k);
391                                 }
```

```
#ifndef OPENSSL_NO_POSIX_IO
#ifdef _WIN32
#define stat _stat
#endif
                        {
                        struct stat st;
                        if (stat(b->data,&st) < 0)
                                break;
                        }
#endif
                        /* found one. */
                        if (type == X509_LU_X509)
                                {
                                if ((X509_load_cert_file(xl,b->data,
                                        ent->dir_type)) == 0)
                                        break;
                                }
                        else if (type == X509_LU_CRL)
                                {
                                if ((X509_load_crl_file(xl,b->data,
                                        ent->dir_type)) == 0)
                                        break;
                                }
                        /* else case will caught higher up */
                        k++;
                        }

                /* we have added it to the cache so now pull
                 * it out again */
                CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
                j = sk_X509_OBJECT_find(xl->store_ctx->objs,&stmp);
                if(j != -1) tmp=sk_X509_OBJECT_value(xl->store_ctx->objs,j);
                else tmp = NULL;
                CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);


                /* If a CRL, update the last file suffix added for this */

                if (type == X509_LU_CRL)
                        {
                        CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
                        /* Look for entry again in case another thread added
                         * an entry first.
                         */
                        if (!hent)
                                {
                                htmp.hash = h;
                                idx = sk_BY_DIR_HASH_find(ent->hashes, &htmp);
                                if (idx >= 0)
                                        hent =
                                         sk_BY_DIR_HASH_value(ent->hashes, idx);
                                }
                        if (!hent)
                                {
                                hent = OPENSSL_malloc(sizeof(BY_DIR_HASH));
                                hent->hash = h;
                                hent->suffix = k;
                                if (!sk_BY_DIR_HASH_push(ent->hashes, hent))
                                        {
                                        CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
                                        OPENSSL_free(hent);
                                        ok = 0;
                                        goto finish;
                                        }
                                }
                        else if (hent->suffix < k)
```



```
                                hent->suffix = k;

                        CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);

                        }

                if (tmp != NULL)
                        {
                        ok=1;
                        ret->type=tmp->type;
                        memcpy(&ret->data,&tmp->data,sizeof(ret->data));
                        /* If we were going to up the reference count,
                         * we would need to do it on a perl 'type'
                         * basis */
        /*              CRYPTO_add(&tmp->data.x509->references,1,
                                CRYPTO_LOCK_X509);*/
                        goto finish;
                        }
                }
finish:
        if (b != NULL) BUF_MEM_free(b);
        return(ok);
        }
#endif /* ! codereview */
```

```
**********************************************************
    8000 Wed Aug 13 19:53:23 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/by_file.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/by_file.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
#include <errno.h>
```



```
#include "cryptlib.h"
#include <openssl/lhash.h>
#include <openssl/buffer.h>
#include <openssl/x509.h>
#include <openssl/pem.h>

#ifndef OPENSSL_NO_STDIO

static int by_file_ctrl(X509_LOOKUP *ctx, int cmd, const char *argc,
        long argl, char **ret);
X509_LOOKUP_METHOD x509_file_lookup=
        {
        "Load file into cache",
        NULL,           /* new */
        NULL,           /* free */
        NULL,           /* init */
        NULL,           /* shutdown */
        by_file_ctrl,   /* ctrl */
        NULL,           /* get_by_subject */
        NULL,           /* get_by_issuer_serial */
        NULL,           /* get_by_fingerprint */
        NULL,           /* get_by_alias */
        };

X509_LOOKUP_METHOD *X509_LOOKUP_file(void)
        {
        return(&x509_file_lookup);
        }

static int by_file_ctrl(X509_LOOKUP *ctx, int cmd, const char *argp, long argl,
             char **ret)
        {
        int ok=0;
        char *file;

        switch (cmd)
                {
        case X509_L_FILE_LOAD:
                if (argl == X509_FILETYPE_DEFAULT)
                        {
                        file = (char *)getenv(X509_get_default_cert_file_env());
                        if (file)
                                ok = (X509_load_cert_crl_file(ctx,file,
                                                X509_FILETYPE_PEM) != 0);

                        else
                                ok = (X509_load_cert_crl_file(ctx,X509_get_defau
                                                X509_FILETYPE_PEM) != 0);

                        if (!ok)
                                {
                                X509err(X509_F_BY_FILE_CTRL,X509_R_LOADING_DEFAU
                                }
                        }
                else
                        {
                        if(argl == X509_FILETYPE_PEM)
                                ok = (X509_load_cert_crl_file(ctx,argp,
                                        X509_FILETYPE_PEM) != 0);
                        else
                                ok = (X509_load_cert_file(ctx,argp,(int)argl) !=
                        }
                break;
                }
        return(ok);
```

```
128         }

130 int X509_load_cert_file(X509_LOOKUP *ctx, const char *file, int type)
131         {
132         int ret=0;
133         BIO *in=NULL;
134         int i,count=0;
135         X509 *x=NULL;

137         if (file == NULL) return(1);
138         in=BIO_new(BIO_s_file_internal());

140         if ((in == NULL) || (BIO_read_filename(in,file) <= 0))
141                 {
142                 X509err(X509_F_X509_LOAD_CERT_FILE,ERR_R_SYS_LIB);
143                 goto err;
144                 }

146         if (type == X509_FILETYPE_PEM)
147                 {
148                 for (;;)
149                         {
150                         x=PEM_read_bio_X509_AUX(in,NULL,NULL,NULL);
151                         if (x == NULL)
152                                 {
153                                 if ((ERR_GET_REASON(ERR_peek_last_error()) ==
154                                         PEM_R_NO_START_LINE) && (count > 0))
155                                         {
156                                         ERR_clear_error();
157                                         break;
158                                         }
159                                 else
160                                         {
161                                         X509err(X509_F_X509_LOAD_CERT_FILE,
162                                                 ERR_R_PEM_LIB);
163                                         goto err;
164                                         }
165                                 }
166                         i=X509_STORE_add_cert(ctx->store_ctx,x);
167                         if (!i) goto err;
168                         count++;
169                         X509_free(x);
170                         x=NULL;
171                         }
172                 ret=count;
173                 }
174         else if (type == X509_FILETYPE_ASN1)
175                 {
176                 x=d2i_X509_bio(in,NULL);
177                 if (x == NULL)
178                         {
179                         X509err(X509_F_X509_LOAD_CERT_FILE,ERR_R_ASN1_LIB);
180                         goto err;
181                         }
182                 i=X509_STORE_add_cert(ctx->store_ctx,x);
183                 if (!i) goto err;
184                 ret=i;
185                 }
186         else
187                 {
188                 X509err(X509_F_X509_LOAD_CERT_FILE,X509_R_BAD_X509_FILETYPE);
189                 goto err;
190                 }
191 err:
192         if (x != NULL) X509_free(x);
193         if (in != NULL) BIO_free(in);
```



```
194         return(ret);
195         }

197 int X509_load_crl_file(X509_LOOKUP *ctx, const char *file, int type)
198         {
199         int ret=0;
200         BIO *in=NULL;
201         int i,count=0;
202         X509_CRL *x=NULL;

204         if (file == NULL) return(1);
205         in=BIO_new(BIO_s_file_internal());

207         if ((in == NULL) || (BIO_read_filename(in,file) <= 0))
208                 {
209                 X509err(X509_F_X509_LOAD_CRL_FILE,ERR_R_SYS_LIB);
210                 goto err;
211                 }

213         if (type == X509_FILETYPE_PEM)
214                 {
215                 for (;;)
216                         {
217                         x=PEM_read_bio_X509_CRL(in,NULL,NULL,NULL);
218                         if (x == NULL)
219                                 {
220                                 if ((ERR_GET_REASON(ERR_peek_last_error()) ==
221                                         PEM_R_NO_START_LINE) && (count > 0))
222                                         {
223                                         ERR_clear_error();
224                                         break;
225                                         }
226                                 else
227                                         {
228                                         X509err(X509_F_X509_LOAD_CRL_FILE,
229                                                 ERR_R_PEM_LIB);
230                                         goto err;
231                                         }
232                                 }
233                         i=X509_STORE_add_crl(ctx->store_ctx,x);
234                         if (!i) goto err;
235                         count++;
236                         X509_CRL_free(x);
237                         x=NULL;
238                         }
239                 ret=count;
240                 }
241         else if (type == X509_FILETYPE_ASN1)
242                 {
243                 x=d2i_X509_CRL_bio(in,NULL);
244                 if (x == NULL)
245                         {
246                         X509err(X509_F_X509_LOAD_CRL_FILE,ERR_R_ASN1_LIB);
247                         goto err;
248                         }
249                 i=X509_STORE_add_crl(ctx->store_ctx,x);
250                 if (!i) goto err;
251                 ret=i;
252                 }
253         else
254                 {
255                 X509err(X509_F_X509_LOAD_CRL_FILE,X509_R_BAD_X509_FILETYPE);
256                 goto err;
257                 }
258 err:
259         if (x != NULL) X509_CRL_free(x);
```

```
260         if (in != NULL) BIO_free(in);
261         return(ret);
262         }

264 int X509_load_cert_crl_file(X509_LOOKUP *ctx, const char *file, int type)
265 {
266         STACK_OF(X509_INFO) *inf;
267         X509_INFO *itmp;
268         BIO *in;
269         int i, count = 0;
270         if(type != X509_FILETYPE_PEM)
271                 return X509_load_cert_file(ctx, file, type);
272         in = BIO_new_file(file, "r");
273         if(!in) {
274                 X509err(X509_F_X509_LOAD_CERT_CRL_FILE,ERR_R_SYS_LIB);
275                 return 0;
276         }
277         inf = PEM_X509_INFO_read_bio(in, NULL, NULL, NULL);
278         BIO_free(in);
279         if(!inf) {
280                 X509err(X509_F_X509_LOAD_CERT_CRL_FILE,ERR_R_PEM_LIB);
281                 return 0;
282         }
283         for(i = 0; i < sk_X509_INFO_num(inf); i++) {
284                 itmp = sk_X509_INFO_value(inf, i);
285                 if(itmp->x509) {
286                         X509_STORE_add_cert(ctx->store_ctx, itmp->x509);
287                         count++;
288                 }
289                 if(itmp->crl) {
290                         X509_STORE_add_crl(ctx->store_ctx, itmp->crl);
291                         count++;
292                 }
293         }
294         sk_X509_INFO_pop_free(inf, X509_INFO_free);
295         return count;
296 }


299 #endif /* OPENSSL_NO_STDIO */
300 #endif /* ! codereview */
```

```
**********************************************************
   10644 Wed Aug 13 19:53:23 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_att.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_att.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/stack.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

int X509at_get_attr_count(const STACK_OF(X509_ATTRIBUTE) *x)
{
        return sk_X509_ATTRIBUTE_num(x);
}

int X509at_get_attr_by_NID(const STACK_OF(X509_ATTRIBUTE) *x, int nid,
                           int lastpos)
{
        ASN1_OBJECT *obj;

        obj=OBJ_nid2obj(nid);
        if (obj == NULL) return(-2);
        return(X509at_get_attr_by_OBJ(x,obj,lastpos));
}

int X509at_get_attr_by_OBJ(const STACK_OF(X509_ATTRIBUTE) *sk, ASN1_OBJECT *obj,
                           int lastpos)
{
        int n;
        X509_ATTRIBUTE *ex;

        if (sk == NULL) return(-1);
        lastpos++;
        if (lastpos < 0)
                lastpos=0;
        n=sk_X509_ATTRIBUTE_num(sk);
        for ( ; lastpos < n; lastpos++)
                {
                ex=sk_X509_ATTRIBUTE_value(sk,lastpos);
                if (OBJ_cmp(ex->object,obj) == 0)
                        return(lastpos);
                }
        return(-1);
}

X509_ATTRIBUTE *X509at_get_attr(const STACK_OF(X509_ATTRIBUTE) *x, int loc)
{
        if (x == NULL || sk_X509_ATTRIBUTE_num(x) <= loc || loc < 0)
                return NULL;
        else
                return sk_X509_ATTRIBUTE_value(x,loc);
}

X509_ATTRIBUTE *X509at_delete_attr(STACK_OF(X509_ATTRIBUTE) *x, int loc)
{
        X509_ATTRIBUTE *ret;

        if (x == NULL || sk_X509_ATTRIBUTE_num(x) <= loc || loc < 0)
                return(NULL);
        ret=sk_X509_ATTRIBUTE_delete(x,loc);
        return(ret);
}

STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr(STACK_OF(X509_ATTRIBUTE) **x,
                                           X509_ATTRIBUTE *attr)
{
        X509_ATTRIBUTE *new_attr=NULL;
        STACK_OF(X509_ATTRIBUTE) *sk=NULL;

        if (x == NULL)
```

```
128                 {
129                 X509err(X509_F_X509AT_ADD1_ATTR, ERR_R_PASSED_NULL_PARAMETER);
130                 goto err2;
131                 }

133         if (*x == NULL)
134                 {
135                 if ((sk=sk_X509_ATTRIBUTE_new_null()) == NULL)
136                         goto err;
137                 }
138         else
139                 sk= *x;

141         if ((new_attr=X509_ATTRIBUTE_dup(attr)) == NULL)
142                 goto err2;
143         if (!sk_X509_ATTRIBUTE_push(sk,new_attr))
144                 goto err;
145         if (*x == NULL)
146                 *x=sk;
147         return(sk);
148 err:
149         X509err(X509_F_X509AT_ADD1_ATTR,ERR_R_MALLOC_FAILURE);
150 err2:
151         if (new_attr != NULL) X509_ATTRIBUTE_free(new_attr);
152         if (sk != NULL) sk_X509_ATTRIBUTE_free(sk);
153         return(NULL);
154 }

156 STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr_by_OBJ(STACK_OF(X509_ATTRIBUTE) **x,
157                         const ASN1_OBJECT *obj, int type,
158                         const unsigned char *bytes, int len)
159 {
160         X509_ATTRIBUTE *attr;
161         STACK_OF(X509_ATTRIBUTE) *ret;
162         attr = X509_ATTRIBUTE_create_by_OBJ(NULL, obj, type, bytes, len);
163         if(!attr) return 0;
164         ret = X509at_add1_attr(x, attr);
165         X509_ATTRIBUTE_free(attr);
166         return ret;
167 }

169 STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr_by_NID(STACK_OF(X509_ATTRIBUTE) **x,
170                         int nid, int type,
171                         const unsigned char *bytes, int len)
172 {
173         X509_ATTRIBUTE *attr;
174         STACK_OF(X509_ATTRIBUTE) *ret;
175         attr = X509_ATTRIBUTE_create_by_NID(NULL, nid, type, bytes, len);
176         if(!attr) return 0;
177         ret = X509at_add1_attr(x, attr);
178         X509_ATTRIBUTE_free(attr);
179         return ret;
180 }

182 STACK_OF(X509_ATTRIBUTE) *X509at_add1_attr_by_txt(STACK_OF(X509_ATTRIBUTE) **x,
183                         const char *attrname, int type,
184                         const unsigned char *bytes, int len)
185 {
186         X509_ATTRIBUTE *attr;
187         STACK_OF(X509_ATTRIBUTE) *ret;
188         attr = X509_ATTRIBUTE_create_by_txt(NULL, attrname, type, bytes, len);
189         if(!attr) return 0;
190         ret = X509at_add1_attr(x, attr);
191         X509_ATTRIBUTE_free(attr);
192         return ret;
193 }
```



```
195 void *X509at_get0_data_by_OBJ(STACK_OF(X509_ATTRIBUTE) *x,
196                                 ASN1_OBJECT *obj, int lastpos, int type)
197 {
198         int i;
199         X509_ATTRIBUTE *at;
200         i = X509at_get_attr_by_OBJ(x, obj, lastpos);
201         if (i == -1)
202                 return NULL;
203         if ((lastpos <= -2) && (X509at_get_attr_by_OBJ(x, obj, i) != -1))
204                 return NULL;
205         at = X509at_get_attr(x, i);
206         if (lastpos <= -3 && (X509_ATTRIBUTE_count(at) != 1))
207                 return NULL;
208         return X509_ATTRIBUTE_get0_data(at, 0, type, NULL);
209 }

211 X509_ATTRIBUTE *X509_ATTRIBUTE_create_by_NID(X509_ATTRIBUTE **attr, int nid,
212              int atrtype, const void *data, int len)
213 {
214         ASN1_OBJECT *obj;
215         X509_ATTRIBUTE *ret;

217         obj=OBJ_nid2obj(nid);
218         if (obj == NULL)
219                 {
220                 X509err(X509_F_X509_ATTRIBUTE_CREATE_BY_NID,X509_R_UNKNOWN_NID);
221                 return(NULL);
222                 }
223         ret=X509_ATTRIBUTE_create_by_OBJ(attr,obj,atrtype,data,len);
224         if (ret == NULL) ASN1_OBJECT_free(obj);
225         return(ret);
226 }

228 X509_ATTRIBUTE *X509_ATTRIBUTE_create_by_OBJ(X509_ATTRIBUTE **attr,
229              const ASN1_OBJECT *obj, int atrtype, const void *data, int len)
230 {
231         X509_ATTRIBUTE *ret;

233         if ((attr == NULL) || (*attr == NULL))
234                 {
235                 if ((ret=X509_ATTRIBUTE_new()) == NULL)
236                         {
237                         X509err(X509_F_X509_ATTRIBUTE_CREATE_BY_OBJ,ERR_R_MALLOC
238                         return(NULL);
239                         }
240                 }
241         else
242                 ret= *attr;

244         if (!X509_ATTRIBUTE_set1_object(ret,obj))
245                 goto err;
246         if (!X509_ATTRIBUTE_set1_data(ret,atrtype,data,len))
247                 goto err;

249         if ((attr != NULL) && (*attr == NULL)) *attr=ret;
250         return(ret);
251 err:
252         if ((attr == NULL) || (ret != *attr))
253                 X509_ATTRIBUTE_free(ret);
254         return(NULL);
255 }

257 X509_ATTRIBUTE *X509_ATTRIBUTE_create_by_txt(X509_ATTRIBUTE **attr,
258                 const char *atrname, int type, const unsigned char *bytes, int l
259         {
```

```
260         ASN1_OBJECT *obj;
261         X509_ATTRIBUTE *nattr;

263         obj=OBJ_txt2obj(atrname, 0);
264         if (obj == NULL)
265                 {
266                 X509err(X509_F_X509_ATTRIBUTE_CREATE_BY_TXT,
267                                                 X509_R_INVALID_FIELD_NAME);
268                 ERR_add_error_data(2, "name=", atrname);
269                 return(NULL);
270                 }
271         nattr = X509_ATTRIBUTE_create_by_OBJ(attr,obj,type,bytes,len);
272         ASN1_OBJECT_free(obj);
273         return nattr;
274         }

276 int X509_ATTRIBUTE_set1_object(X509_ATTRIBUTE *attr, const ASN1_OBJECT *obj)
277 {
278         if ((attr == NULL) || (obj == NULL))
279                 return(0);
280         ASN1_OBJECT_free(attr->object);
281         attr->object=OBJ_dup(obj);
282         return(1);
283 }

285 int X509_ATTRIBUTE_set1_data(X509_ATTRIBUTE *attr, int attrtype, const void *dat
286 {
287         ASN1_TYPE *ttmp;
288         ASN1_STRING *stmp = NULL;
289         int atype = 0;
290         if (!attr) return 0;
291         if(attrtype & MBSTRING_FLAG) {
292                 stmp = ASN1_STRING_set_by_NID(NULL, data, len, attrtype,
293                                                 OBJ_obj2nid(attr->object));
294                 if(!stmp) {
295                         X509err(X509_F_X509_ATTRIBUTE_SET1_DATA, ERR_R_ASN1_LIB)
296                         return 0;
297                 }
298                 atype = stmp->type;
299         } else if (len != -1){
300                 if(!(stmp = ASN1_STRING_type_new(attrtype))) goto err;
301                 if(!ASN1_STRING_set(stmp, data, len)) goto err;
302                 atype = attrtype;
303         }
304         if(!(attr->value.set = sk_ASN1_TYPE_new_null())) goto err;
305         attr->single = 0;
306         /* This is a bit naughty because the attribute should really have
307          * at least one value but some types use and zero length SET and
308          * require this.
309          */
310         if (attrtype == 0)
311                 return 1;
312         if(!(ttmp = ASN1_TYPE_new())) goto err;
313         if ((len == -1) && !(attrtype & MBSTRING_FLAG))
314                 {
315                 if (!ASN1_TYPE_set1(ttmp, attrtype, data))
316                         goto err;
317                 }
318         else
319                 ASN1_TYPE_set(ttmp, atype, stmp);
320         if(!sk_ASN1_TYPE_push(attr->value.set, ttmp)) goto err;
321         return 1;
322         err:
323         X509err(X509_F_X509_ATTRIBUTE_SET1_DATA, ERR_R_MALLOC_FAILURE);
324         return 0;
325 }
```



```
327 int X509_ATTRIBUTE_count(X509_ATTRIBUTE *attr)
328 {
329         if(!attr->single) return sk_ASN1_TYPE_num(attr->value.set);
330         if(attr->value.single) return 1;
331         return 0;
332 }

334 ASN1_OBJECT *X509_ATTRIBUTE_get0_object(X509_ATTRIBUTE *attr)
335 {
336         if (attr == NULL) return(NULL);
337         return(attr->object);
338 }

340 void *X509_ATTRIBUTE_get0_data(X509_ATTRIBUTE *attr, int idx,
341                                         int atrtype, void *data)
342 {
343         ASN1_TYPE *ttmp;
344         ttmp = X509_ATTRIBUTE_get0_type(attr, idx);
345         if(!ttmp) return NULL;
346         if(atrtype != ASN1_TYPE_get(ttmp)){
347                 X509err(X509_F_X509_ATTRIBUTE_GET0_DATA, X509_R_WRONG_TYPE);
348                 return NULL;
349         }
350         return ttmp->value.ptr;
351 }

353 ASN1_TYPE *X509_ATTRIBUTE_get0_type(X509_ATTRIBUTE *attr, int idx)
354 {
355         if (attr == NULL) return(NULL);
356         if(idx >= X509_ATTRIBUTE_count(attr)) return NULL;
357         if(!attr->single) return sk_ASN1_TYPE_value(attr->value.set, idx);
358         else return attr->value.single;
359 }
360 #endif /* ! codereview */
```

```
**********************************************************
    9559 Wed Aug 13 19:53:23 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_cmp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_cmp.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

int X509_issuer_and_serial_cmp(const X509 *a, const X509 *b)
        {
        int i;
        X509_CINF *ai,*bi;

        ai=a->cert_info;
        bi=b->cert_info;
        i=M_ASN1_INTEGER_cmp(ai->serialNumber,bi->serialNumber);
        if (i) return(i);
        return(X509_NAME_cmp(ai->issuer,bi->issuer));
        }

#ifndef OPENSSL_NO_MD5
unsigned long X509_issuer_and_serial_hash(X509 *a)
        {
        unsigned long ret=0;
        EVP_MD_CTX ctx;
        unsigned char md[16];
        char *f;

        EVP_MD_CTX_init(&ctx);
        f=X509_NAME_oneline(a->cert_info->issuer,NULL,0);
        if (!EVP_DigestInit_ex(&ctx, EVP_md5(), NULL))
                goto err;
        if (!EVP_DigestUpdate(&ctx,(unsigned char *)f,strlen(f)))
                goto err;
        OPENSSL_free(f);
        if(!EVP_DigestUpdate(&ctx,(unsigned char *)a->cert_info->serialNumber->d
                (unsigned long)a->cert_info->serialNumber->length))
                goto err;
        if (!EVP_DigestFinal_ex(&ctx,&(md[0]),NULL))
                goto err;
        ret=(   ((unsigned long)md[0]     )|((unsigned long)md[1]<<8L)|
                ((unsigned long)md[2]<<16L)|((unsigned long)md[3]<<24L)
                )&0xffffffffL;
        err:
        EVP_MD_CTX_cleanup(&ctx);
        return(ret);
        }
#endif

int X509_issuer_name_cmp(const X509 *a, const X509 *b)
        {
        return(X509_NAME_cmp(a->cert_info->issuer,b->cert_info->issuer));
        }

int X509_subject_name_cmp(const X509 *a, const X509 *b)
        {
        return(X509_NAME_cmp(a->cert_info->subject,b->cert_info->subject));
        }

int X509_CRL_cmp(const X509_CRL *a, const X509_CRL *b)
        {
        return(X509_NAME_cmp(a->crl->issuer,b->crl->issuer));
        }

#ifndef OPENSSL_NO_SHA
int X509_CRL_match(const X509_CRL *a, const X509_CRL *b)
        {
        return memcmp(a->sha1_hash, b->sha1_hash, 20);
        }
```

```
#endif

X509_NAME *X509_get_issuer_name(X509 *a)
        {
        return(a->cert_info->issuer);
        }

unsigned long X509_issuer_name_hash(X509 *x)
        {
        return(X509_NAME_hash(x->cert_info->issuer));
        }

#ifndef OPENSSL_NO_MD5
unsigned long X509_issuer_name_hash_old(X509 *x)
        {
        return(X509_NAME_hash_old(x->cert_info->issuer));
        }
#endif

X509_NAME *X509_get_subject_name(X509 *a)
        {
        return(a->cert_info->subject);
        }

ASN1_INTEGER *X509_get_serialNumber(X509 *a)
        {
        return(a->cert_info->serialNumber);
        }

unsigned long X509_subject_name_hash(X509 *x)
        {
        return(X509_NAME_hash(x->cert_info->subject));
        }

#ifndef OPENSSL_NO_MD5
unsigned long X509_subject_name_hash_old(X509 *x)
        {
        return(X509_NAME_hash_old(x->cert_info->subject));
        }
#endif

#ifndef OPENSSL_NO_SHA
/* Compare two certificates: they must be identical for
 * this to work. NB: Although "cmp" operations are generally
 * prototyped to take "const" arguments (eg. for use in
 * STACKs), the way X509 handling is - these operations may
 * involve ensuring the hashes are up-to-date and ensuring
 * certain cert information is cached. So this is the point
 * where the "depth-first" constification tree has to halt
 * with an evil cast.
 */
int X509_cmp(const X509 *a, const X509 *b)
{
        /* ensure hash is valid */
        X509_check_purpose((X509 *)a, -1, 0);
        X509_check_purpose((X509 *)b, -1, 0);

        return memcmp(a->sha1_hash, b->sha1_hash, SHA_DIGEST_LENGTH);
}
#endif


int X509_NAME_cmp(const X509_NAME *a, const X509_NAME *b)
        {
        int ret;
```



```
        /* Ensure canonical encoding is present and up to date */

        if (!a->canon_enc || a->modified)
                {
                ret = i2d_X509_NAME((X509_NAME *)a, NULL);
                if (ret < 0)
                        return -2;
                }

        if (!b->canon_enc || b->modified)
                {
                ret = i2d_X509_NAME((X509_NAME *)b, NULL);
                if (ret < 0)
                        return -2;
                }

        ret = a->canon_enclen - b->canon_enclen;

        if (ret)
                return ret;

        return memcmp(a->canon_enc, b->canon_enc, a->canon_enclen);

        }

unsigned long X509_NAME_hash(X509_NAME *x)
        {
        unsigned long ret=0;
        unsigned char md[SHA_DIGEST_LENGTH];

        /* Make sure X509_NAME structure contains valid cached encoding */
        i2d_X509_NAME(x,NULL);
        if (!EVP_Digest(x->canon_enc, x->canon_enclen, md, NULL, EVP_sha1(),
                NULL))
                return 0;

        ret=(   ((unsigned long)md[0]     )|((unsigned long)md[1]<<8L)|
                ((unsigned long)md[2]<<16L)|((unsigned long)md[3]<<24L)
                )&0xffffffffL;
        return(ret);
        }


#ifndef OPENSSL_NO_MD5
/* I now DER encode the name and hash it.  Since I cache the DER encoding,
 * this is reasonably efficient. */

unsigned long X509_NAME_hash_old(X509_NAME *x)
        {
        EVP_MD_CTX md_ctx;
        unsigned long ret=0;
        unsigned char md[16];

        /* Make sure X509_NAME structure contains valid cached encoding */
        i2d_X509_NAME(x,NULL);
        EVP_MD_CTX_init(&md_ctx);
        EVP_MD_CTX_set_flags(&md_ctx, EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
        if (EVP_DigestInit_ex(&md_ctx, EVP_md5(), NULL)
            && EVP_DigestUpdate(&md_ctx, x->bytes->data, x->bytes->length)
            && EVP_DigestFinal_ex(&md_ctx,md,NULL))
                ret=(((unsigned long)md[0]     )|((unsigned long)md[1]<<8L)|
                     ((unsigned long)md[2]<<16L)|((unsigned long)md[3]<<24L)
                     )&0xffffffffL;
        EVP_MD_CTX_cleanup(&md_ctx);

        return(ret);
```

```
260         }
261 #endif

263 /* Search a stack of X509 for a match */
264 X509 *X509_find_by_issuer_and_serial(STACK_OF(X509) *sk, X509_NAME *name,
265                 ASN1_INTEGER *serial)
266         {
267         int i;
268         X509_CINF cinf;
269         X509 x,*x509=NULL;

271         if(!sk) return NULL;

273         x.cert_info= &cinf;
274         cinf.serialNumber=serial;
275         cinf.issuer=name;

277         for (i=0; i<sk_X509_num(sk); i++)
278                 {
279                 x509=sk_X509_value(sk,i);
280                 if (X509_issuer_and_serial_cmp(x509,&x) == 0)
281                         return(x509);
282                 }
283         return(NULL);
284         }

286 X509 *X509_find_by_subject(STACK_OF(X509) *sk, X509_NAME *name)
287         {
288         X509 *x509;
289         int i;

291         for (i=0; i<sk_X509_num(sk); i++)
292                 {
293                 x509=sk_X509_value(sk,i);
294                 if (X509_NAME_cmp(X509_get_subject_name(x509),name) == 0)
295                         return(x509);
296                 }
297         return(NULL);
298         }

300 EVP_PKEY *X509_get_pubkey(X509 *x)
301         {
302         if ((x == NULL) || (x->cert_info == NULL))
303                 return(NULL);
304         return(X509_PUBKEY_get(x->cert_info->key));
305         }

307 ASN1_BIT_STRING *X509_get0_pubkey_bitstr(const X509 *x)
308         {
309         if(!x) return NULL;
310         return x->cert_info->key->public_key;
311         }

313 int X509_check_private_key(X509 *x, EVP_PKEY *k)
314         {
315         EVP_PKEY *xk;
316         int ret;

318         xk=X509_get_pubkey(x);

320         if (xk)
321                 ret = EVP_PKEY_cmp(xk, k);
322         else
323                 ret = -2;

325         switch (ret)
```



```
326                 {
327         case 1:
328                 break;
329         case 0:
330                 X509err(X509_F_X509_CHECK_PRIVATE_KEY,X509_R_KEY_VALUES_MISMATCH
331                 break;
332         case -1:
333                 X509err(X509_F_X509_CHECK_PRIVATE_KEY,X509_R_KEY_TYPE_MISMATCH);
334                 break;
335         case -2:
336                 X509err(X509_F_X509_CHECK_PRIVATE_KEY,X509_R_UNKNOWN_KEY_TYPE);
337                 }
338         if (xk)
339                 EVP_PKEY_free(xk);
340         if (ret > 0)
341                 return 1;
342         return 0;
343         }
344 #endif /* ! codereview */
```

```
**********************************************************
    4330 Wed Aug 13 19:53:24 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_d2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_d2.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/crypto.h>
```



```
#include <openssl/x509.h>

#ifndef OPENSSL_NO_STDIO
int X509_STORE_set_default_paths(X509_STORE *ctx)
        {
        X509_LOOKUP *lookup;

        lookup=X509_STORE_add_lookup(ctx,X509_LOOKUP_file());
        if (lookup == NULL) return(0);
        X509_LOOKUP_load_file(lookup,NULL,X509_FILETYPE_DEFAULT);

        lookup=X509_STORE_add_lookup(ctx,X509_LOOKUP_hash_dir());
        if (lookup == NULL) return(0);
        X509_LOOKUP_add_dir(lookup,NULL,X509_FILETYPE_DEFAULT);

        /* clear any errors */
        ERR_clear_error();

        return(1);
        }

int X509_STORE_load_locations(X509_STORE *ctx, const char *file,
                const char *path)
        {
        X509_LOOKUP *lookup;

        if (file != NULL)
                {
                lookup=X509_STORE_add_lookup(ctx,X509_LOOKUP_file());
                if (lookup == NULL) return(0);
                if (X509_LOOKUP_load_file(lookup,file,X509_FILETYPE_PEM) != 1)
                    return(0);
                }
        if (path != NULL)
                {
                lookup=X509_STORE_add_lookup(ctx,X509_LOOKUP_hash_dir());
                if (lookup == NULL) return(0);
                if (X509_LOOKUP_add_dir(lookup,path,X509_FILETYPE_PEM) != 1)
                    return(0);
                }
        if ((path == NULL) && (file == NULL))
                return(0);
        return(1);
        }

#endif
#endif /* ! codereview */
```

```
**********************************************************
    3754 Wed Aug 13 19:53:24 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_def.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_def.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/crypto.h>
```



```
#include <openssl/x509.h>

const char *X509_get_default_private_dir(void)
        { return(X509_PRIVATE_DIR); }

const char *X509_get_default_cert_area(void)
        { return(X509_CERT_AREA); }

const char *X509_get_default_cert_dir(void)
        { return(X509_CERT_DIR); }

const char *X509_get_default_cert_file(void)
        { return(X509_CERT_FILE); }

const char *X509_get_default_cert_dir_env(void)
        { return(X509_CERT_DIR_EVP); }

const char *X509_get_default_cert_file_env(void)
        { return(X509_CERT_FILE_EVP); }
#endif /* ! codereview */
```

```
**********************************************************
    8109 Wed Aug 13 19:53:24 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_err.c */
/* ====================================================================
 * Copyright (c) 1999-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/x509.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_X509,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_X509,0,reason)

static ERR_STRING_DATA X509_str_functs[]=
        {
{ERR_FUNC(X509_F_ADD_CERT_DIR), "ADD_CERT_DIR"},
{ERR_FUNC(X509_F_BY_FILE_CTRL), "BY_FILE_CTRL"},
{ERR_FUNC(X509_F_CHECK_POLICY), "CHECK_POLICY"},
{ERR_FUNC(X509_F_DIR_CTRL),     "DIR_CTRL"},
{ERR_FUNC(X509_F_GET_CERT_BY_SUBJECT),  "GET_CERT_BY_SUBJECT"},
{ERR_FUNC(X509_F_NETSCAPE_SPKI_B64_DECODE),     "NETSCAPE_SPKI_b64_decode"},
{ERR_FUNC(X509_F_NETSCAPE_SPKI_B64_ENCODE),     "NETSCAPE_SPKI_b64_encode"},
{ERR_FUNC(X509_F_X509AT_ADD1_ATTR),     "X509at_add1_attr"},
{ERR_FUNC(X509_F_X509V3_ADD_EXT),       "X509v3_add_ext"},
{ERR_FUNC(X509_F_X509_ATTRIBUTE_CREATE_BY_NID), "X509_ATTRIBUTE_create_by_NID"},
{ERR_FUNC(X509_F_X509_ATTRIBUTE_CREATE_BY_OBJ), "X509_ATTRIBUTE_create_by_OBJ"},
{ERR_FUNC(X509_F_X509_ATTRIBUTE_CREATE_BY_TXT), "X509_ATTRIBUTE_create_by_txt"},
{ERR_FUNC(X509_F_X509_ATTRIBUTE_GET0_DATA),     "X509_ATTRIBUTE_get0_data"},
{ERR_FUNC(X509_F_X509_ATTRIBUTE_SET1_DATA),     "X509_ATTRIBUTE_set1_data"},
{ERR_FUNC(X509_F_X509_CHECK_PRIVATE_KEY),       "X509_check_private_key"},
{ERR_FUNC(X509_F_X509_CRL_PRINT_FP),    "X509_CRL_print_fp"},
{ERR_FUNC(X509_F_X509_EXTENSION_CREATE_BY_NID), "X509_EXTENSION_create_by_NID"},
{ERR_FUNC(X509_F_X509_EXTENSION_CREATE_BY_OBJ), "X509_EXTENSION_create_by_OBJ"},
{ERR_FUNC(X509_F_X509_GET_PUBKEY_PARAMETERS),   "X509_get_pubkey_parameters"},
{ERR_FUNC(X509_F_X509_LOAD_CERT_CRL_FILE),      "X509_load_cert_crl_file"},
{ERR_FUNC(X509_F_X509_LOAD_CERT_FILE),  "X509_load_cert_file"},
{ERR_FUNC(X509_F_X509_LOAD_CRL_FILE),   "X509_load_crl_file"},
{ERR_FUNC(X509_F_X509_NAME_ADD_ENTRY),  "X509_NAME_add_entry"},
{ERR_FUNC(X509_F_X509_NAME_ENTRY_CREATE_BY_NID),        "X509_NAME_ENTRY_create_
{ERR_FUNC(X509_F_X509_NAME_ENTRY_CREATE_BY_TXT),        "X509_NAME_ENTRY_create_
{ERR_FUNC(X509_F_X509_NAME_ENTRY_SET_OBJECT),   "X509_NAME_ENTRY_set_object"},
{ERR_FUNC(X509_F_X509_NAME_ONELINE),    "X509_NAME_oneline"},
{ERR_FUNC(X509_F_X509_NAME_PRINT),      "X509_NAME_print"},
{ERR_FUNC(X509_F_X509_PRINT_EX_FP),     "X509_print_ex_fp"},
{ERR_FUNC(X509_F_X509_PUBKEY_GET),      "X509_PUBKEY_get"},
{ERR_FUNC(X509_F_X509_PUBKEY_SET),      "X509_PUBKEY_set"},
{ERR_FUNC(X509_F_X509_REQ_CHECK_PRIVATE_KEY),   "X509_REQ_check_private_key"},
{ERR_FUNC(X509_F_X509_REQ_PRINT_EX),    "X509_REQ_print_ex"},
{ERR_FUNC(X509_F_X509_REQ_PRINT_FP),    "X509_REQ_print_fp"},
{ERR_FUNC(X509_F_X509_REQ_TO_X509),     "X509_REQ_to_X509"},
{ERR_FUNC(X509_F_X509_STORE_ADD_CERT),  "X509_STORE_add_cert"},
{ERR_FUNC(X509_F_X509_STORE_ADD_CRL),   "X509_STORE_add_crl"},
{ERR_FUNC(X509_F_X509_STORE_CTX_GET1_ISSUER),   "X509_STORE_CTX_get1_issuer"},
{ERR_FUNC(X509_F_X509_STORE_CTX_INIT),  "X509_STORE_CTX_init"},
{ERR_FUNC(X509_F_X509_STORE_CTX_NEW),   "X509_STORE_CTX_new"},
{ERR_FUNC(X509_F_X509_STORE_CTX_PURPOSE_INHERIT),       "X509_STORE_CTX_purpose_
{ERR_FUNC(X509_F_X509_TO_X509_REQ),     "X509_to_X509_REQ"},
{ERR_FUNC(X509_F_X509_TRUST_ADD),       "X509_TRUST_add"},
{ERR_FUNC(X509_F_X509_TRUST_SET),       "X509_TRUST_set"},
{ERR_FUNC(X509_F_X509_VERIFY_CERT),     "X509_verify_cert"},
{0,NULL}
        };

static ERR_STRING_DATA X509_str_reasons[]=
        {
{ERR_REASON(X509_R_BAD_X509_FILETYPE)    ,"bad x509 filetype"},
{ERR_REASON(X509_R_BASE64_DECODE_ERROR)  ,"base64 decode error"},
{ERR_REASON(X509_R_CANT_CHECK_DH_KEY)    ,"cant check dh key"},
{ERR_REASON(X509_R_CERT_ALREADY_IN_HASH_TABLE),"cert already in hash table"},
{ERR_REASON(X509_R_ERR_ASN1_LIB)         ,"err asn1 lib"},
```

```
{ERR_REASON(X509_R_INVALID_DIRECTORY)    ,"invalid directory"},
{ERR_REASON(X509_R_INVALID_FIELD_NAME)   ,"invalid field name"},
{ERR_REASON(X509_R_INVALID_TRUST)        ,"invalid trust"},
{ERR_REASON(X509_R_KEY_TYPE_MISMATCH)    ,"key type mismatch"},
{ERR_REASON(X509_R_KEY_VALUES_MISMATCH)  ,"key values mismatch"},
{ERR_REASON(X509_R_LOADING_CERT_DIR)     ,"loading cert dir"},
{ERR_REASON(X509_R_LOADING_DEFAULTS)     ,"loading defaults"},
{ERR_REASON(X509_R_METHOD_NOT_SUPPORTED) ,"method not supported"},
{ERR_REASON(X509_R_NO_CERT_SET_FOR_US_TO_VERIFY),"no cert set for us to verify"}
{ERR_REASON(X509_R_PUBLIC_KEY_DECODE_ERROR),"public key decode error"},
{ERR_REASON(X509_R_PUBLIC_KEY_ENCODE_ERROR),"public key encode error"},
{ERR_REASON(X509_R_SHOULD_RETRY)         ,"should retry"},
{ERR_REASON(X509_R_UNABLE_TO_FIND_PARAMETERS_IN_CHAIN),"unable to find parameter
{ERR_REASON(X509_R_UNABLE_TO_GET_CERTS_PUBLIC_KEY),"unable to get certs public k
{ERR_REASON(X509_R_UNKNOWN_KEY_TYPE)     ,"unknown key type"},
{ERR_REASON(X509_R_UNKNOWN_NID)          ,"unknown nid"},
{ERR_REASON(X509_R_UNKNOWN_PURPOSE_ID)   ,"unknown purpose id"},
{ERR_REASON(X509_R_UNKNOWN_TRUST_ID)     ,"unknown trust id"},
{ERR_REASON(X509_R_UNSUPPORTED_ALGORITHM),"unsupported algorithm"},
{ERR_REASON(X509_R_WRONG_LOOKUP_TYPE)    ,"wrong lookup type"},
{ERR_REASON(X509_R_WRONG_TYPE)           ,"wrong type"},
{0,NULL}
        };

#endif

void ERR_load_X509_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(X509_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,X509_str_functs);
                ERR_load_strings(0,X509_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    7068 Wed Aug 13 19:53:24 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_ext.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_ext.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/stack.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>


int X509_CRL_get_ext_count(X509_CRL *x)
        {
        return(X509v3_get_ext_count(x->crl->extensions));
        }

int X509_CRL_get_ext_by_NID(X509_CRL *x, int nid, int lastpos)
        {
        return(X509v3_get_ext_by_NID(x->crl->extensions,nid,lastpos));
        }

int X509_CRL_get_ext_by_OBJ(X509_CRL *x, ASN1_OBJECT *obj, int lastpos)
        {
        return(X509v3_get_ext_by_OBJ(x->crl->extensions,obj,lastpos));
        }

int X509_CRL_get_ext_by_critical(X509_CRL *x, int crit, int lastpos)
        {
        return(X509v3_get_ext_by_critical(x->crl->extensions,crit,lastpos));
        }

X509_EXTENSION *X509_CRL_get_ext(X509_CRL *x, int loc)
        {
        return(X509v3_get_ext(x->crl->extensions,loc));
        }

X509_EXTENSION *X509_CRL_delete_ext(X509_CRL *x, int loc)
        {
        return(X509v3_delete_ext(x->crl->extensions,loc));
        }

void *X509_CRL_get_ext_d2i(X509_CRL *x, int nid, int *crit, int *idx)
{
        return X509V3_get_d2i(x->crl->extensions, nid, crit, idx);
}

int X509_CRL_add1_ext_i2d(X509_CRL *x, int nid, void *value, int crit,
                                                        unsigned long flags)
{
        return X509V3_add1_i2d(&x->crl->extensions, nid, value, crit, flags);
}

int X509_CRL_add_ext(X509_CRL *x, X509_EXTENSION *ex, int loc)
        {
        return(X509v3_add_ext(&(x->crl->extensions),ex,loc) != NULL);
        }

int X509_get_ext_count(X509 *x)
        {
        return(X509v3_get_ext_count(x->cert_info->extensions));
        }

int X509_get_ext_by_NID(X509 *x, int nid, int lastpos)
        {
        return(X509v3_get_ext_by_NID(x->cert_info->extensions,nid,lastpos));
        }

int X509_get_ext_by_OBJ(X509 *x, ASN1_OBJECT *obj, int lastpos)
        {
        return(X509v3_get_ext_by_OBJ(x->cert_info->extensions,obj,lastpos));
```

```
128         }

130 int X509_get_ext_by_critical(X509 *x, int crit, int lastpos)
131         {
132         return(X509v3_get_ext_by_critical(x->cert_info->extensions,crit,lastpos)
133         }

135 X509_EXTENSION *X509_get_ext(X509 *x, int loc)
136         {
137         return(X509v3_get_ext(x->cert_info->extensions,loc));
138         }

140 X509_EXTENSION *X509_delete_ext(X509 *x, int loc)
141         {
142         return(X509v3_delete_ext(x->cert_info->extensions,loc));
143         }

145 int X509_add_ext(X509 *x, X509_EXTENSION *ex, int loc)
146         {
147         return(X509v3_add_ext(&(x->cert_info->extensions),ex,loc) != NULL);
148         }

150 void *X509_get_ext_d2i(X509 *x, int nid, int *crit, int *idx)
151 {
152         return X509V3_get_d2i(x->cert_info->extensions, nid, crit, idx);
153 }

155 int X509_add1_ext_i2d(X509 *x, int nid, void *value, int crit,
156                                                 unsigned long flags)
157 {
158         return X509V3_add1_i2d(&x->cert_info->extensions, nid, value, crit,
159                                                 flags);
160 }

162 int X509_REVOKED_get_ext_count(X509_REVOKED *x)
163         {
164         return(X509v3_get_ext_count(x->extensions));
165         }

167 int X509_REVOKED_get_ext_by_NID(X509_REVOKED *x, int nid, int lastpos)
168         {
169         return(X509v3_get_ext_by_NID(x->extensions,nid,lastpos));
170         }

172 int X509_REVOKED_get_ext_by_OBJ(X509_REVOKED *x, ASN1_OBJECT *obj,
173                  int lastpos)
174         {
175         return(X509v3_get_ext_by_OBJ(x->extensions,obj,lastpos));
176         }

178 int X509_REVOKED_get_ext_by_critical(X509_REVOKED *x, int crit, int lastpos)
179         {
180         return(X509v3_get_ext_by_critical(x->extensions,crit,lastpos));
181         }

183 X509_EXTENSION *X509_REVOKED_get_ext(X509_REVOKED *x, int loc)
184         {
185         return(X509v3_get_ext(x->extensions,loc));
186         }

188 X509_EXTENSION *X509_REVOKED_delete_ext(X509_REVOKED *x, int loc)
189         {
190         return(X509v3_delete_ext(x->extensions,loc));
191         }

193 int X509_REVOKED_add_ext(X509_REVOKED *x, X509_EXTENSION *ex, int loc)
```



```
194         {
195         return(X509v3_add_ext(&(x->extensions),ex,loc) != NULL);
196         }

198 void *X509_REVOKED_get_ext_d2i(X509_REVOKED *x, int nid, int *crit, int *idx)
199 {
200         return X509V3_get_d2i(x->extensions, nid, crit, idx);
201 }

203 int X509_REVOKED_add1_ext_i2d(X509_REVOKED *x, int nid, void *value, int crit,
204                                                 unsigned long flags)
205 {
206         return X509V3_add1_i2d(&x->extensions, nid, value, crit, flags);
207 }

209 IMPLEMENT_STACK_OF(X509_EXTENSION)
210 IMPLEMENT_ASN1_SET_OF(X509_EXTENSION)
211 #endif /* ! codereview */
```

```
**********************************************************
   17776 Wed Aug 13 19:53:24 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_lu.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_lu.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/lhash.h>
```



```
#include <openssl/x509.h>
#include <openssl/x509v3.h>

X509_LOOKUP *X509_LOOKUP_new(X509_LOOKUP_METHOD *method)
        {
        X509_LOOKUP *ret;

        ret=(X509_LOOKUP *)OPENSSL_malloc(sizeof(X509_LOOKUP));
        if (ret == NULL) return NULL;

        ret->init=0;
        ret->skip=0;
        ret->method=method;
        ret->method_data=NULL;
        ret->store_ctx=NULL;
        if ((method->new_item != NULL) && !method->new_item(ret))
                {
                OPENSSL_free(ret);
                return NULL;
                }
        return ret;
        }

void X509_LOOKUP_free(X509_LOOKUP *ctx)
        {
        if (ctx == NULL) return;
        if (    (ctx->method != NULL) &&
                (ctx->method->free != NULL))
                (*ctx->method->free)(ctx);
        OPENSSL_free(ctx);
        }

int X509_LOOKUP_init(X509_LOOKUP *ctx)
        {
        if (ctx->method == NULL) return 0;
        if (ctx->method->init != NULL)
                return ctx->method->init(ctx);
        else
                return 1;
        }

int X509_LOOKUP_shutdown(X509_LOOKUP *ctx)
        {
        if (ctx->method == NULL) return 0;
        if (ctx->method->shutdown != NULL)
                return ctx->method->shutdown(ctx);
        else
                return 1;
        }

int X509_LOOKUP_ctrl(X509_LOOKUP *ctx, int cmd, const char *argc, long argl,
             char **ret)
        {
        if (ctx->method == NULL) return -1;
        if (ctx->method->ctrl != NULL)
                return ctx->method->ctrl(ctx,cmd,argc,argl,ret);
        else
                return 1;
        }

int X509_LOOKUP_by_subject(X509_LOOKUP *ctx, int type, X509_NAME *name,
             X509_OBJECT *ret)
        {
        if ((ctx->method == NULL) || (ctx->method->get_by_subject == NULL))
                return X509_LU_FAIL;
        if (ctx->skip) return 0;
```

```
128         return ctx->method->get_by_subject(ctx,type,name,ret);
129         }

131 int X509_LOOKUP_by_issuer_serial(X509_LOOKUP *ctx, int type, X509_NAME *name,
132              ASN1_INTEGER *serial, X509_OBJECT *ret)
133         {
134         if ((ctx->method == NULL) ||
135                 (ctx->method->get_by_issuer_serial == NULL))
136                 return X509_LU_FAIL;
137         return ctx->method->get_by_issuer_serial(ctx,type,name,serial,ret);
138         }

140 int X509_LOOKUP_by_fingerprint(X509_LOOKUP *ctx, int type,
141              unsigned char *bytes, int len, X509_OBJECT *ret)
142         {
143         if ((ctx->method == NULL) || (ctx->method->get_by_fingerprint == NULL))
144                 return X509_LU_FAIL;
145         return ctx->method->get_by_fingerprint(ctx,type,bytes,len,ret);
146         }

148 int X509_LOOKUP_by_alias(X509_LOOKUP *ctx, int type, char *str, int len,
149              X509_OBJECT *ret)
150         {
151         if ((ctx->method == NULL) || (ctx->method->get_by_alias == NULL))
152                 return X509_LU_FAIL;
153         return ctx->method->get_by_alias(ctx,type,str,len,ret);
154         }


157 static int x509_object_cmp(const X509_OBJECT * const *a, const X509_OBJECT * con
158         {
159         int ret;

161         ret=((*a)->type - (*b)->type);
162         if (ret) return ret;
163         switch ((*a)->type)
164                 {
165         case X509_LU_X509:
166                 ret=X509_subject_name_cmp((*a)->data.x509,(*b)->data.x509);
167                 break;
168         case X509_LU_CRL:
169                 ret=X509_CRL_cmp((*a)->data.crl,(*b)->data.crl);
170                 break;
171         default:
172                 /* abort(); */
173                 return 0;
174                 }
175         return ret;
176         }

178 X509_STORE *X509_STORE_new(void)
179         {
180         X509_STORE *ret;

182         if ((ret=(X509_STORE *)OPENSSL_malloc(sizeof(X509_STORE))) == NULL)
183                 return NULL;
184         ret->objs = sk_X509_OBJECT_new(x509_object_cmp);
185         ret->cache=1;
186         ret->get_cert_methods=sk_X509_LOOKUP_new_null();
187         ret->verify=0;
188         ret->verify_cb=0;

190         if ((ret->param = X509_VERIFY_PARAM_new()) == NULL)
191                 return NULL;

193         ret->get_issuer = 0;
```



```
194         ret->check_issued = 0;
195         ret->check_revocation = 0;
196         ret->get_crl = 0;
197         ret->check_crl = 0;
198         ret->cert_crl = 0;
199         ret->lookup_certs = 0;
200         ret->lookup_crls = 0;
201         ret->cleanup = 0;

203         if (!CRYPTO_new_ex_data(CRYPTO_EX_INDEX_X509_STORE, ret, &ret->ex_data))
204                 {
205                 sk_X509_OBJECT_free(ret->objs);
206                 OPENSSL_free(ret);
207                 return NULL;
208                 }

210         ret->references=1;
211         return ret;
212         }

214 static void cleanup(X509_OBJECT *a)
215         {
216         if (a->type == X509_LU_X509)
217                 {
218                 X509_free(a->data.x509);
219                 }
220         else if (a->type == X509_LU_CRL)
221                 {
222                 X509_CRL_free(a->data.crl);
223                 }
224         else
225                 {
226                 /* abort(); */
227                 }

229         OPENSSL_free(a);
230         }

232 void X509_STORE_free(X509_STORE *vfy)
233         {
234         int i;
235         STACK_OF(X509_LOOKUP) *sk;
236         X509_LOOKUP *lu;

238         if (vfy == NULL)
239             return;

241         sk=vfy->get_cert_methods;
242         for (i=0; i<sk_X509_LOOKUP_num(sk); i++)
243                 {
244                 lu=sk_X509_LOOKUP_value(sk,i);
245                 X509_LOOKUP_shutdown(lu);
246                 X509_LOOKUP_free(lu);
247                 }
248         sk_X509_LOOKUP_free(sk);
249         sk_X509_OBJECT_pop_free(vfy->objs, cleanup);

251         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_X509_STORE, vfy, &vfy->ex_data);
252         if (vfy->param)
253                 X509_VERIFY_PARAM_free(vfy->param);
254         OPENSSL_free(vfy);
255         }

257 X509_LOOKUP *X509_STORE_add_lookup(X509_STORE *v, X509_LOOKUP_METHOD *m)
258         {
259         int i;
```

```
260         STACK_OF(X509_LOOKUP) *sk;
261         X509_LOOKUP *lu;

263         sk=v->get_cert_methods;
264         for (i=0; i<sk_X509_LOOKUP_num(sk); i++)
265                 {
266                 lu=sk_X509_LOOKUP_value(sk,i);
267                 if (m == lu->method)
268                         {
269                         return lu;
270                         }
271                 }
272         /* a new one */
273         lu=X509_LOOKUP_new(m);
274         if (lu == NULL)
275                 return NULL;
276         else
277                 {
278                 lu->store_ctx=v;
279                 if (sk_X509_LOOKUP_push(v->get_cert_methods,lu))
280                         return lu;
281                 else
282                         {
283                         X509_LOOKUP_free(lu);
284                         return NULL;
285                         }
286                 }
287         }

289 int X509_STORE_get_by_subject(X509_STORE_CTX *vs, int type, X509_NAME *name,
290             X509_OBJECT *ret)
291         {
292         X509_STORE *ctx=vs->ctx;
293         X509_LOOKUP *lu;
294         X509_OBJECT stmp,*tmp;
295         int i,j;

297         CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
298         tmp=X509_OBJECT_retrieve_by_subject(ctx->objs,type,name);
299         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);

301         if (tmp == NULL || type == X509_LU_CRL)
302                 {
303                 for (i=vs->current_method; i<sk_X509_LOOKUP_num(ctx->get_cert_me
304                         {
305                         lu=sk_X509_LOOKUP_value(ctx->get_cert_methods,i);
306                         j=X509_LOOKUP_by_subject(lu,type,name,&stmp);
307                         if (j < 0)
308                                 {
309                                 vs->current_method=j;
310                                 return j;
311                                 }
312                         else if (j)
313                                 {
314                                 tmp= &stmp;
315                                 break;
316                                 }
317                         }
318                 vs->current_method=0;
319                 if (tmp == NULL)
320                         return 0;
321                 }

323 /*      if (ret->data.ptr != NULL)
324                 X509_OBJECT_free_contents(ret); */
```



```
326         ret->type=tmp->type;
327         ret->data.ptr=tmp->data.ptr;

329         X509_OBJECT_up_ref_count(ret);

331         return 1;
332         }

334 int X509_STORE_add_cert(X509_STORE *ctx, X509 *x)
335         {
336         X509_OBJECT *obj;
337         int ret=1;

339         if (x == NULL) return 0;
340         obj=(X509_OBJECT *)OPENSSL_malloc(sizeof(X509_OBJECT));
341         if (obj == NULL)
342                 {
343                 X509err(X509_F_X509_STORE_ADD_CERT,ERR_R_MALLOC_FAILURE);
344                 return 0;
345                 }
346         obj->type=X509_LU_X509;
347         obj->data.x509=x;

349         CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);

351         X509_OBJECT_up_ref_count(obj);

353         if (X509_OBJECT_retrieve_match(ctx->objs, obj))
354                 {
355                 X509_OBJECT_free_contents(obj);
356                 OPENSSL_free(obj);
357                 X509err(X509_F_X509_STORE_ADD_CERT,X509_R_CERT_ALREADY_IN_HASH_T
358                 ret=0;
359                 }
360         else sk_X509_OBJECT_push(ctx->objs, obj);

362         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);

364         return ret;
365         }

367 int X509_STORE_add_crl(X509_STORE *ctx, X509_CRL *x)
368         {
369         X509_OBJECT *obj;
370         int ret=1;

372         if (x == NULL) return 0;
373         obj=(X509_OBJECT *)OPENSSL_malloc(sizeof(X509_OBJECT));
374         if (obj == NULL)
375                 {
376                 X509err(X509_F_X509_STORE_ADD_CRL,ERR_R_MALLOC_FAILURE);
377                 return 0;
378                 }
379         obj->type=X509_LU_CRL;
380         obj->data.crl=x;

382         CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);

384         X509_OBJECT_up_ref_count(obj);

386         if (X509_OBJECT_retrieve_match(ctx->objs, obj))
387                 {
388                 X509_OBJECT_free_contents(obj);
389                 OPENSSL_free(obj);
390                 X509err(X509_F_X509_STORE_ADD_CRL,X509_R_CERT_ALREADY_IN_HASH_TA
391                 ret=0;
```

```
392                 }
393         else sk_X509_OBJECT_push(ctx->objs, obj);

395         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);

397         return ret;
398         }

400 void X509_OBJECT_up_ref_count(X509_OBJECT *a)
401         {
402         switch (a->type)
403                 {
404         case X509_LU_X509:
405                 CRYPTO_add(&a->data.x509->references,1,CRYPTO_LOCK_X509);
406                 break;
407         case X509_LU_CRL:
408                 CRYPTO_add(&a->data.crl->references,1,CRYPTO_LOCK_X509_CRL);
409                 break;
410                 }
411         }

413 void X509_OBJECT_free_contents(X509_OBJECT *a)
414         {
415         switch (a->type)
416                 {
417         case X509_LU_X509:
418                 X509_free(a->data.x509);
419                 break;
420         case X509_LU_CRL:
421                 X509_CRL_free(a->data.crl);
422                 break;
423                 }
424         }

426 static int x509_object_idx_cnt(STACK_OF(X509_OBJECT) *h, int type,
427              X509_NAME *name, int *pnmatch)
428         {
429         X509_OBJECT stmp;
430         X509 x509_s;
431         X509_CINF cinf_s;
432         X509_CRL crl_s;
433         X509_CRL_INFO crl_info_s;
434         int idx;

436         stmp.type=type;
437         switch (type)
438                 {
439         case X509_LU_X509:
440                 stmp.data.x509= &x509_s;
441                 x509_s.cert_info= &cinf_s;
442                 cinf_s.subject=name;
443                 break;
444         case X509_LU_CRL:
445                 stmp.data.crl= &crl_s;
446                 crl_s.crl= &crl_info_s;
447                 crl_info_s.issuer=name;
448                 break;
449         default:
450                 /* abort(); */
451                 return -1;
452                 }

454         idx = sk_X509_OBJECT_find(h,&stmp);
455         if (idx >= 0 && pnmatch)
456                 {
457                 int tidx;
```



```
458                 const X509_OBJECT *tobj, *pstmp;
459                 *pnmatch = 1;
460                 pstmp = &stmp;
461                 for (tidx = idx + 1; tidx < sk_X509_OBJECT_num(h); tidx++)
462                         {
463                         tobj = sk_X509_OBJECT_value(h, tidx);
464                         if (x509_object_cmp(&tobj, &pstmp))
465                                 break;
466                         (*pnmatch)++;
467                         }
468                 }
469         return idx;
470         }


473 int X509_OBJECT_idx_by_subject(STACK_OF(X509_OBJECT) *h, int type,
474              X509_NAME *name)
475         {
476         return x509_object_idx_cnt(h, type, name, NULL);
477         }

479 X509_OBJECT *X509_OBJECT_retrieve_by_subject(STACK_OF(X509_OBJECT) *h, int type,
480              X509_NAME *name)
481         {
482         int idx;
483         idx = X509_OBJECT_idx_by_subject(h, type, name);
484         if (idx==-1) return NULL;
485         return sk_X509_OBJECT_value(h, idx);
486         }

488 STACK_OF(X509)* X509_STORE_get1_certs(X509_STORE_CTX *ctx, X509_NAME *nm)
489         {
490         int i, idx, cnt;
491         STACK_OF(X509) *sk;
492         X509 *x;
493         X509_OBJECT *obj;
494         sk = sk_X509_new_null();
495         CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
496         idx = x509_object_idx_cnt(ctx->ctx->objs, X509_LU_X509, nm, &cnt);
497         if (idx < 0)
498                 {
499                 /* Nothing found in cache: do lookup to possibly add new
500                  * objects to cache
501                  */
502                 X509_OBJECT xobj;
503                 CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
504                 if (!X509_STORE_get_by_subject(ctx, X509_LU_X509, nm, &xobj))
505                         {
506                         sk_X509_free(sk);
507                         return NULL;
508                         }
509                 X509_OBJECT_free_contents(&xobj);
510                 CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
511                 idx = x509_object_idx_cnt(ctx->ctx->objs,X509_LU_X509,nm, &cnt);
512                 if (idx < 0)
513                         {
514                         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
515                         sk_X509_free(sk);
516                         return NULL;
517                         }
518                 }
519         for (i = 0; i < cnt; i++, idx++)
520                 {
521                 obj = sk_X509_OBJECT_value(ctx->ctx->objs, idx);
522                 x = obj->data.x509;
523                 CRYPTO_add(&x->references, 1, CRYPTO_LOCK_X509);
```

```
524                 if (!sk_X509_push(sk, x))
525                         {
526                         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
527                         X509_free(x);
528                         sk_X509_pop_free(sk, X509_free);
529                         return NULL;
530                         }
531                 }
532         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
533         return sk;

535         }

537 STACK_OF(X509_CRL)* X509_STORE_get1_crls(X509_STORE_CTX *ctx, X509_NAME *nm)
538         {
539         int i, idx, cnt;
540         STACK_OF(X509_CRL) *sk;
541         X509_CRL *x;
542         X509_OBJECT *obj, xobj;
543         sk = sk_X509_CRL_new_null();
544         CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
545         /* Check cache first */
546         idx = x509_object_idx_cnt(ctx->ctx->objs, X509_LU_CRL, nm, &cnt);

548         /* Always do lookup to possibly add new CRLs to cache
549          */
550         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
551         if (!X509_STORE_get_by_subject(ctx, X509_LU_CRL, nm, &xobj))
552                 {
553                 sk_X509_CRL_free(sk);
554                 return NULL;
555                 }
556         X509_OBJECT_free_contents(&xobj);
557         CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
558         idx = x509_object_idx_cnt(ctx->ctx->objs,X509_LU_CRL, nm, &cnt);
559         if (idx < 0)
560                 {
561                 CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
562                 sk_X509_CRL_free(sk);
563                 return NULL;
564                 }

566         for (i = 0; i < cnt; i++, idx++)
567                 {
568                 obj = sk_X509_OBJECT_value(ctx->ctx->objs, idx);
569                 x = obj->data.crl;
570                 CRYPTO_add(&x->references, 1, CRYPTO_LOCK_X509_CRL);
571                 if (!sk_X509_CRL_push(sk, x))
572                         {
573                         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
574                         X509_CRL_free(x);
575                         sk_X509_CRL_pop_free(sk, X509_CRL_free);
576                         return NULL;
577                         }
578                 }
579         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
580         return sk;
581         }

583 X509_OBJECT *X509_OBJECT_retrieve_match(STACK_OF(X509_OBJECT) *h, X509_OBJECT *x
584         {
585         int idx, i;
586         X509_OBJECT *obj;
587         idx = sk_X509_OBJECT_find(h, x);
588         if (idx == -1) return NULL;
589         if ((x->type != X509_LU_X509) && (x->type != X509_LU_CRL))
```



```
590                 return sk_X509_OBJECT_value(h, idx);
591         for (i = idx; i < sk_X509_OBJECT_num(h); i++)
592                 {
593                 obj = sk_X509_OBJECT_value(h, i);
594                 if (x509_object_cmp((const X509_OBJECT **)&obj, (const X509_OBJE
595                         return NULL;
596                 if (x->type == X509_LU_X509)
597                         {
598                         if (!X509_cmp(obj->data.x509, x->data.x509))
599                                 return obj;
600                         }
601                 else if (x->type == X509_LU_CRL)
602                         {
603                         if (!X509_CRL_match(obj->data.crl, x->data.crl))
604                                 return obj;
605                         }
606                 else
607                         return obj;
608                 }
609         return NULL;
610         }


613 /* Try to get issuer certificate from store. Due to limitations
614  * of the API this can only retrieve a single certificate matching
615  * a given subject name. However it will fill the cache with all
616  * matching certificates, so we can examine the cache for all
617  * matches.
618  *
619  * Return values are:
620  *  1 lookup successful.
621  *  0 certificate not found.
622  * -1 some other error.
623  */
624 int X509_STORE_CTX_get1_issuer(X509 **issuer, X509_STORE_CTX *ctx, X509 *x)
625         {
626         X509_NAME *xn;
627         X509_OBJECT obj, *pobj;
628         int i, ok, idx, ret;
629         xn=X509_get_issuer_name(x);
630         ok=X509_STORE_get_by_subject(ctx,X509_LU_X509,xn,&obj);
631         if (ok != X509_LU_X509)
632                 {
633                 if (ok == X509_LU_RETRY)
634                         {
635                         X509_OBJECT_free_contents(&obj);
636                         X509err(X509_F_X509_STORE_CTX_GET1_ISSUER,X509_R_SHOULD_
637                         return -1;
638                         }
639                 else if (ok != X509_LU_FAIL)
640                         {
641                         X509_OBJECT_free_contents(&obj);
642                         /* not good :-(, break anyway */
643                         return -1;
644                         }
645                 return 0;
646                 }
647         /* If certificate matches all OK */
648         if (ctx->check_issued(ctx, x, obj.data.x509))
649                 {
650                 *issuer = obj.data.x509;
651                 return 1;
652                 }
653         X509_OBJECT_free_contents(&obj);

655         /* Else find index of first cert accepted by 'check_issued' */
```

```
656         ret = 0;
657         CRYPTO_w_lock(CRYPTO_LOCK_X509_STORE);
658         idx = X509_OBJECT_idx_by_subject(ctx->ctx->objs, X509_LU_X509, xn);
659         if (idx != -1) /* should be true as we've had at least one match */
660                 {
661                 /* Look through all matching certs for suitable issuer */
662                 for (i = idx; i < sk_X509_OBJECT_num(ctx->ctx->objs); i++)
663                         {
664                         pobj = sk_X509_OBJECT_value(ctx->ctx->objs, i);
665                         /* See if we've run past the matches */
666                         if (pobj->type != X509_LU_X509)
667                                 break;
668                         if (X509_NAME_cmp(xn, X509_get_subject_name(pobj->data.x
669                                 break;
670                         if (ctx->check_issued(ctx, x, pobj->data.x509))
671                                 {
672                                 *issuer = pobj->data.x509;
673                                 X509_OBJECT_up_ref_count(pobj);
674                                 ret = 1;
675                                 break;
676                                 }
677                         }
678                 }
679         CRYPTO_w_unlock(CRYPTO_LOCK_X509_STORE);
680         return ret;
681         }

683 int X509_STORE_set_flags(X509_STORE *ctx, unsigned long flags)
684         {
685         return X509_VERIFY_PARAM_set_flags(ctx->param, flags);
686         }

688 int X509_STORE_set_depth(X509_STORE *ctx, int depth)
689         {
690         X509_VERIFY_PARAM_set_depth(ctx->param, depth);
691         return 1;
692         }

694 int X509_STORE_set_purpose(X509_STORE *ctx, int purpose)
695         {
696         return X509_VERIFY_PARAM_set_purpose(ctx->param, purpose);
697         }

699 int X509_STORE_set_trust(X509_STORE *ctx, int trust)
700         {
701         return X509_VERIFY_PARAM_set_trust(ctx->param, trust);
702         }

704 int X509_STORE_set1_param(X509_STORE *ctx, X509_VERIFY_PARAM *param)
705         {
706         return X509_VERIFY_PARAM_set1(ctx->param, param);
707         }

709 void X509_STORE_set_verify_cb(X509_STORE *ctx,
710                               int (*verify_cb)(int, X509_STORE_CTX *))
711         {
712         ctx->verify_cb = verify_cb;
713         }

715 IMPLEMENT_STACK_OF(X509_LOOKUP)
716 IMPLEMENT_STACK_OF(X509_OBJECT)
717 #endif /* ! codereview */
```

```
**********************************************************
    6501 Wed Aug 13 19:53:24 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_obj.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_obj.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/lhash.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>
#include <openssl/buffer.h>

char *X509_NAME_oneline(X509_NAME *a, char *buf, int len)
        {
        X509_NAME_ENTRY *ne;
int i;
        int n,lold,l,l1,l2,num,j,type;
        const char *s;
        char *p;
        unsigned char *q;
        BUF_MEM *b=NULL;
        static const char hex[17]="0123456789ABCDEF";
        int gs_doit[4];
        char tmp_buf[80];
#ifdef CHARSET_EBCDIC
        char ebcdic_buf[1024];
#endif

        if (buf == NULL)
                {
                if ((b=BUF_MEM_new()) == NULL) goto err;
                if (!BUF_MEM_grow(b,200)) goto err;
                b->data[0]='\0';
                len=200;
                }
        if (a == NULL)
            {
            if(b)
                {
                buf=b->data;
                OPENSSL_free(b);
                }
            strncpy(buf,"NO X509_NAME",len);
            buf[len-1]='\0';
            return buf;
            }

        len--; /* space for '\0' */
        l=0;
        for (i=0; i<sk_X509_NAME_ENTRY_num(a->entries); i++)
                {
                ne=sk_X509_NAME_ENTRY_value(a->entries,i);
                n=OBJ_obj2nid(ne->object);
                if ((n == NID_undef) || ((s=OBJ_nid2sn(n)) == NULL))
                        {
                        i2t_ASN1_OBJECT(tmp_buf,sizeof(tmp_buf),ne->object);
                        s=tmp_buf;
                        }
                l1=strlen(s);

                type=ne->value->type;
                num=ne->value->length;
                q=ne->value->data;
#ifdef CHARSET_EBCDIC
                if (type == V_ASN1_GENERALSTRING ||
                    type == V_ASN1_VISIBLESTRING ||
                    type == V_ASN1_PRINTABLESTRING ||
                    type == V_ASN1_TELETEXSTRING ||
                    type == V_ASN1_VISIBLESTRING ||
                    type == V_ASN1_IA5STRING) {
                    ascii2ebcdic(ebcdic_buf, q,
                                 (num > sizeof ebcdic_buf)
                                 ? sizeof ebcdic_buf : num);
                    q=ebcdic_buf;
```

```
128                 }
129 #endif

131                 if ((type == V_ASN1_GENERALSTRING) && ((num%4) == 0))
132                         {
133                         gs_doit[0]=gs_doit[1]=gs_doit[2]=gs_doit[3]=0;
134                         for (j=0; j<num; j++)
135                                 if (q[j] != 0) gs_doit[j&3]=1;

137                         if (gs_doit[0]|gs_doit[1]|gs_doit[2])
138                                 gs_doit[0]=gs_doit[1]=gs_doit[2]=gs_doit[3]=1;
139                         else
140                                 {
141                                 gs_doit[0]=gs_doit[1]=gs_doit[2]=0;
142                                 gs_doit[3]=1;
143                                 }
144                         }
145                 else
146                         gs_doit[0]=gs_doit[1]=gs_doit[2]=gs_doit[3]=1;

148                 for (l2=j=0; j<num; j++)
149                         {
150                         if (!gs_doit[j&3]) continue;
151                         l2++;
152 #ifndef CHARSET_EBCDIC
153                         if ((q[j] < ' ') || (q[j] > '~')) l2+=3;
154 #else
155                         if ((os_toascii[q[j]] < os_toascii[' ']) ||
156                             (os_toascii[q[j]] > os_toascii['~'])) l2+=3;
157 #endif
158                         }

160                 lold=l;
161                 l+=1+l1+1+l2;
162                 if (b != NULL)
163                         {
164                         if (!BUF_MEM_grow(b,l+1)) goto err;
165                         p= &(b->data[lold]);
166                         }
167                 else if (l > len)
168                         {
169                         break;
170                         }
171                 else
172                         p= &(buf[lold]);
173                 *(p++)='/';
174                 memcpy(p,s,(unsigned int)l1); p+=l1;
175                 *(p++)='=';

177 #ifndef CHARSET_EBCDIC /* q was assigned above already. */
178                 q=ne->value->data;
179 #endif

181                 for (j=0; j<num; j++)
182                         {
183                         if (!gs_doit[j&3]) continue;
184 #ifndef CHARSET_EBCDIC
185                         n=q[j];
186                         if ((n < ' ') || (n > '~'))
187                                 {
188                                 *(p++)='\\';
189                                 *(p++)='x';
190                                 *(p++)=hex[(n>>4)&0x0f];
191                                 *(p++)=hex[n&0x0f];
192                                 }
193                         else
```



```
194                                 *(p++)=n;
195 #else
196                         n=os_toascii[q[j]];
197                         if ((n < os_toascii[' ']) ||
198                             (n > os_toascii['~']))
199                                 {
200                                 *(p++)='\\';
201                                 *(p++)='x';
202                                 *(p++)=hex[(n>>4)&0x0f];
203                                 *(p++)=hex[n&0x0f];
204                                 }
205                         else
206                                 *(p++)=q[j];
207 #endif
208                         }
209                 *p='\0';
210                 }
211         if (b != NULL)
212                 {
213                 p=b->data;
214                 OPENSSL_free(b);
215                 }
216         else
217                 p=buf;
218         if (i == 0)
219                 *p = '\0';
220         return(p);
221 err:
222         X509err(X509_F_X509_NAME_ONELINE,ERR_R_MALLOC_FAILURE);
223         if (b != NULL) BUF_MEM_free(b);
224         return(NULL);
225         }
226 #endif /* ! codereview */
```

```
**********************************************************
    4432 Wed Aug 13 19:53:25 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_r2x.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_r2x.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/evp.h>
#include <openssl/asn1.h>
#include <openssl/x509.h>
#include <openssl/objects.h>
#include <openssl/buffer.h>

X509 *X509_REQ_to_X509(X509_REQ *r, int days, EVP_PKEY *pkey)
        {
        X509 *ret=NULL;
        X509_CINF *xi=NULL;
        X509_NAME *xn;

        if ((ret=X509_new()) == NULL)
                {
                X509err(X509_F_X509_REQ_TO_X509,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        /* duplicate the request */
        xi=ret->cert_info;

        if (sk_X509_ATTRIBUTE_num(r->req_info->attributes) != 0)
                {
                if ((xi->version=M_ASN1_INTEGER_new()) == NULL) goto err;
                if (!ASN1_INTEGER_set(xi->version,2)) goto err;
/*              xi->extensions=ri->attributes; <- bad, should not ever be done
                ri->attributes=NULL; */
                }

        xn=X509_REQ_get_subject_name(r);
        if (X509_set_subject_name(ret,X509_NAME_dup(xn)) == 0)
                goto err;
        if (X509_set_issuer_name(ret,X509_NAME_dup(xn)) == 0)
                goto err;

        if (X509_gmtime_adj(xi->validity->notBefore,0) == NULL)
                goto err;
        if (X509_gmtime_adj(xi->validity->notAfter,(long)60*60*24*days) == NULL)
                goto err;

        X509_set_pubkey(ret,X509_REQ_get_pubkey(r));

        if (!X509_sign(ret,pkey,EVP_md5()))
                goto err;
        if (0)
                {
err:
                X509_free(ret);
                ret=NULL;
                }
        return(ret);
        }
#endif /* ! codereview */
```

```
**********************************************************
    9382 Wed Aug 13 19:53:25 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_req.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_req.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/bn.h>
```



```
#include <openssl/evp.h>
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509.h>
#include <openssl/objects.h>
#include <openssl/buffer.h>
#include <openssl/pem.h>

X509_REQ *X509_to_X509_REQ(X509 *x, EVP_PKEY *pkey, const EVP_MD *md)
        {
        X509_REQ *ret;
        X509_REQ_INFO *ri;
        int i;
        EVP_PKEY *pktmp;

        ret=X509_REQ_new();
        if (ret == NULL)
                {
                X509err(X509_F_X509_TO_X509_REQ,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        ri=ret->req_info;

        ri->version->length=1;
        ri->version->data=(unsigned char *)OPENSSL_malloc(1);
        if (ri->version->data == NULL) goto err;
        ri->version->data[0]=0; /* version == 0 */

        if (!X509_REQ_set_subject_name(ret,X509_get_subject_name(x)))
                goto err;

        pktmp = X509_get_pubkey(x);
        i=X509_REQ_set_pubkey(ret,pktmp);
        EVP_PKEY_free(pktmp);
        if (!i) goto err;

        if (pkey != NULL)
                {
                if (!X509_REQ_sign(ret,pkey,md))
                        goto err;
                }
        return(ret);
err:
        X509_REQ_free(ret);
        return(NULL);
        }

EVP_PKEY *X509_REQ_get_pubkey(X509_REQ *req)
        {
        if ((req == NULL) || (req->req_info == NULL))
                return(NULL);
        return(X509_PUBKEY_get(req->req_info->pubkey));
        }

int X509_REQ_check_private_key(X509_REQ *x, EVP_PKEY *k)
        {
        EVP_PKEY *xk=NULL;
        int ok=0;

        xk=X509_REQ_get_pubkey(x);
        switch (EVP_PKEY_cmp(xk, k))
                {
        case 1:
                ok=1;
                break;
```

```
128         case 0:
129                 X509err(X509_F_X509_REQ_CHECK_PRIVATE_KEY,X509_R_KEY_VALUES_MISM
130                 break;
131         case -1:
132                 X509err(X509_F_X509_REQ_CHECK_PRIVATE_KEY,X509_R_KEY_TYPE_MISMAT
133                 break;
134         case -2:
135 #ifndef OPENSSL_NO_EC
136                 if (k->type == EVP_PKEY_EC)
137                         {
138                         X509err(X509_F_X509_REQ_CHECK_PRIVATE_KEY, ERR_R_EC_LIB)
139                         break;
140                         }
141 #endif
142 #ifndef OPENSSL_NO_DH
143                 if (k->type == EVP_PKEY_DH)
144                         {
145                         /* No idea */
146                         X509err(X509_F_X509_REQ_CHECK_PRIVATE_KEY,X509_R_CANT_CH
147                         break;
148                         }
149 #endif
150                 X509err(X509_F_X509_REQ_CHECK_PRIVATE_KEY,X509_R_UNKNOWN_KEY_TYP
151                 }

153         EVP_PKEY_free(xk);
154         return(ok);
155         }

157 /* It seems several organisations had the same idea of including a list of
158  * extensions in a certificate request. There are at least two OIDs that are
159  * used and there may be more: so the list is configurable.
160  */

162 static int ext_nid_list[] = { NID_ext_req, NID_ms_ext_req, NID_undef};

164 static int *ext_nids = ext_nid_list;

166 int X509_REQ_extension_nid(int req_nid)
167 {
168         int i, nid;
169         for(i = 0; ; i++) {
170                 nid = ext_nids[i];
171                 if(nid == NID_undef) return 0;
172                 else if (req_nid == nid) return 1;
173         }
174 }

176 int *X509_REQ_get_extension_nids(void)
177 {
178         return ext_nids;
179 }

181 void X509_REQ_set_extension_nids(int *nids)
182 {
183         ext_nids = nids;
184 }

186 STACK_OF(X509_EXTENSION) *X509_REQ_get_extensions(X509_REQ *req)
187         {
188         X509_ATTRIBUTE *attr;
189         ASN1_TYPE *ext = NULL;
190         int idx, *pnid;
191         const unsigned char *p;

193         if ((req == NULL) || (req->req_info == NULL) || !ext_nids)
```



```
194                 return(NULL);
195         for (pnid = ext_nids; *pnid != NID_undef; pnid++)
196                 {
197                 idx = X509_REQ_get_attr_by_NID(req, *pnid, -1);
198                 if (idx == -1)
199                         continue;
200                 attr = X509_REQ_get_attr(req, idx);
201                 if(attr->single) ext = attr->value.single;
202                 else if(sk_ASN1_TYPE_num(attr->value.set))
203                         ext = sk_ASN1_TYPE_value(attr->value.set, 0);
204                 break;
205                 }
206         if(!ext || (ext->type != V_ASN1_SEQUENCE))
207                 return NULL;
208         p = ext->value.sequence->data;
209         return (STACK_OF(X509_EXTENSION) *)
210                 ASN1_item_d2i(NULL, &p, ext->value.sequence->length,
211                                 ASN1_ITEM_rptr(X509_EXTENSIONS));
212 }

214 /* Add a STACK_OF extensions to a certificate request: allow alternative OIDs
215  * in case we want to create a non standard one.
216  */

218 int X509_REQ_add_extensions_nid(X509_REQ *req, STACK_OF(X509_EXTENSION) *exts,
219                                 int nid)
220 {
221         ASN1_TYPE *at = NULL;
222         X509_ATTRIBUTE *attr = NULL;
223         if(!(at = ASN1_TYPE_new()) ||
224                 !(at->value.sequence = ASN1_STRING_new())) goto err;

226         at->type = V_ASN1_SEQUENCE;
227         /* Generate encoding of extensions */
228         at->value.sequence->length =
229                         ASN1_item_i2d((ASN1_VALUE *)exts,
230                                 &at->value.sequence->data,
231                                 ASN1_ITEM_rptr(X509_EXTENSIONS));
232         if(!(attr = X509_ATTRIBUTE_new())) goto err;
233         if(!(attr->value.set = sk_ASN1_TYPE_new_null())) goto err;
234         if(!sk_ASN1_TYPE_push(attr->value.set, at)) goto err;
235         at = NULL;
236         attr->single = 0;
237         attr->object = OBJ_nid2obj(nid);
238         if (!req->req_info->attributes)
239                 {
240                 if (!(req->req_info->attributes = sk_X509_ATTRIBUTE_new_null()))
241                         goto err;
242                 }
243         if(!sk_X509_ATTRIBUTE_push(req->req_info->attributes, attr)) goto err;
244         return 1;
245         err:
246         X509_ATTRIBUTE_free(attr);
247         ASN1_TYPE_free(at);
248         return 0;
249 }
250 /* This is the normal usage: use the "official" OID */
251 int X509_REQ_add_extensions(X509_REQ *req, STACK_OF(X509_EXTENSION) *exts)
252 {
253         return X509_REQ_add_extensions_nid(req, exts, NID_ext_req);
254 }

256 /* Request attribute functions */

258 int X509_REQ_get_attr_count(const X509_REQ *req)
259 {
```

```
260         return X509at_get_attr_count(req->req_info->attributes);
261 }

263 int X509_REQ_get_attr_by_NID(const X509_REQ *req, int nid,
264                              int lastpos)
265 {
266         return X509at_get_attr_by_NID(req->req_info->attributes, nid, lastpos);
267 }

269 int X509_REQ_get_attr_by_OBJ(const X509_REQ *req, ASN1_OBJECT *obj,
270                              int lastpos)
271 {
272         return X509at_get_attr_by_OBJ(req->req_info->attributes, obj, lastpos);
273 }

275 X509_ATTRIBUTE *X509_REQ_get_attr(const X509_REQ *req, int loc)
276 {
277         return X509at_get_attr(req->req_info->attributes, loc);
278 }

280 X509_ATTRIBUTE *X509_REQ_delete_attr(X509_REQ *req, int loc)
281 {
282         return X509at_delete_attr(req->req_info->attributes, loc);
283 }

285 int X509_REQ_add1_attr(X509_REQ *req, X509_ATTRIBUTE *attr)
286 {
287         if(X509at_add1_attr(&req->req_info->attributes, attr)) return 1;
288         return 0;
289 }

291 int X509_REQ_add1_attr_by_OBJ(X509_REQ *req,
292                               const ASN1_OBJECT *obj, int type,
293                               const unsigned char *bytes, int len)
294 {
295         if(X509at_add1_attr_by_OBJ(&req->req_info->attributes, obj,
296                                    type, bytes, len)) return 1;
297         return 0;
298 }

300 int X509_REQ_add1_attr_by_NID(X509_REQ *req,
301                               int nid, int type,
302                               const unsigned char *bytes, int len)
303 {
304         if(X509at_add1_attr_by_NID(&req->req_info->attributes, nid,
305                                    type, bytes, len)) return 1;
306         return 0;
307 }

309 int X509_REQ_add1_attr_by_txt(X509_REQ *req,
310                               const char *attrname, int type,
311                               const unsigned char *bytes, int len)
312 {
313         if(X509at_add1_attr_by_txt(&req->req_info->attributes, attrname,
314                                    type, bytes, len)) return 1;
315         return 0;
316 }
317 #endif /* ! codereview */
```

```
**********************************************************
    5172 Wed Aug 13 19:53:25 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_set.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_set.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>

int X509_set_version(X509 *x, long version)
        {
        if (x == NULL) return(0);
        if (x->cert_info->version == NULL)
                {
                if ((x->cert_info->version=M_ASN1_INTEGER_new()) == NULL)
                        return(0);
                }
        return(ASN1_INTEGER_set(x->cert_info->version,version));
        }

int X509_set_serialNumber(X509 *x, ASN1_INTEGER *serial)
        {
        ASN1_INTEGER *in;

        if (x == NULL) return(0);
        in=x->cert_info->serialNumber;
        if (in != serial)
                {
                in=M_ASN1_INTEGER_dup(serial);
                if (in != NULL)
                        {
                        M_ASN1_INTEGER_free(x->cert_info->serialNumber);
                        x->cert_info->serialNumber=in;
                        }
                }
        return(in != NULL);
        }

int X509_set_issuer_name(X509 *x, X509_NAME *name)
        {
        if ((x == NULL) || (x->cert_info == NULL)) return(0);
        return(X509_NAME_set(&x->cert_info->issuer,name));
        }

int X509_set_subject_name(X509 *x, X509_NAME *name)
        {
        if ((x == NULL) || (x->cert_info == NULL)) return(0);
        return(X509_NAME_set(&x->cert_info->subject,name));
        }

int X509_set_notBefore(X509 *x, const ASN1_TIME *tm)
        {
        ASN1_TIME *in;

        if ((x == NULL) || (x->cert_info->validity == NULL)) return(0);
        in=x->cert_info->validity->notBefore;
        if (in != tm)
                {
                in=M_ASN1_TIME_dup(tm);
                if (in != NULL)
                        {
                        M_ASN1_TIME_free(x->cert_info->validity->notBefore);
                        x->cert_info->validity->notBefore=in;
                        }
                }
        return(in != NULL);
        }

int X509_set_notAfter(X509 *x, const ASN1_TIME *tm)
        {
        ASN1_TIME *in;
```

```
129         if ((x == NULL) || (x->cert_info->validity == NULL)) return(0);
130         in=x->cert_info->validity->notAfter;
131         if (in != tm)
132                 {
133                 in=M_ASN1_TIME_dup(tm);
134                 if (in != NULL)
135                         {
136                         M_ASN1_TIME_free(x->cert_info->validity->notAfter);
137                         x->cert_info->validity->notAfter=in;
138                         }
139                 }
140         return(in != NULL);
141         }

143 int X509_set_pubkey(X509 *x, EVP_PKEY *pkey)
144         {
145         if ((x == NULL) || (x->cert_info == NULL)) return(0);
146         return(X509_PUBKEY_set(&(x->cert_info->key),pkey));
147         }
148 #endif /* ! codereview */
```

```
**********************************************************
    8809 Wed Aug 13 19:53:25 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_trs.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x509_trs.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509v3.h>
```



```
static int tr_cmp(const X509_TRUST * const *a,
                  const X509_TRUST * const *b);
static void trtable_free(X509_TRUST *p);

static int trust_1oidany(X509_TRUST *trust, X509 *x, int flags);
static int trust_1oid(X509_TRUST *trust, X509 *x, int flags);
static int trust_compat(X509_TRUST *trust, X509 *x, int flags);

static int obj_trust(int id, X509 *x, int flags);
static int (*default_trust)(int id, X509 *x, int flags) = obj_trust;

/* WARNING: the following table should be kept in order of trust
 * and without any gaps so we can just subtract the minimum trust
 * value to get an index into the table
 */

static X509_TRUST trstandard[] = {
{X509_TRUST_COMPAT, 0, trust_compat, "compatible", 0, NULL},
{X509_TRUST_SSL_CLIENT, 0, trust_1oidany, "SSL Client", NID_client_auth, NULL},
{X509_TRUST_SSL_SERVER, 0, trust_1oidany, "SSL Server", NID_server_auth, NULL},
{X509_TRUST_EMAIL, 0, trust_1oidany, "S/MIME email", NID_email_protect, NULL},
{X509_TRUST_OBJECT_SIGN, 0, trust_1oidany, "Object Signer", NID_code_sign, NULL},
{X509_TRUST_OCSP_SIGN, 0, trust_1oid, "OCSP responder", NID_OCSP_sign, NULL},
{X509_TRUST_OCSP_REQUEST, 0, trust_1oid, "OCSP request", NID_ad_OCSP, NULL},
{X509_TRUST_TSA, 0, trust_1oidany, "TSA server", NID_time_stamp, NULL}
};

#define X509_TRUST_COUNT        (sizeof(trstandard)/sizeof(X509_TRUST))

IMPLEMENT_STACK_OF(X509_TRUST)

static STACK_OF(X509_TRUST) *trtable = NULL;

static int tr_cmp(const X509_TRUST * const *a,
                  const X509_TRUST * const *b)
{
        return (*a)->trust - (*b)->trust;
}

int (*X509_TRUST_set_default(int (*trust)(int , X509 *, int)))(int, X509 *, int)
{
        int (*oldtrust)(int , X509 *, int);
        oldtrust = default_trust;
        default_trust = trust;
        return oldtrust;
}


int X509_check_trust(X509 *x, int id, int flags)
{
        X509_TRUST *pt;
        int idx;
        if(id == -1) return 1;
        idx = X509_TRUST_get_by_id(id);
        if(idx == -1) return default_trust(id, x, flags);
        pt = X509_TRUST_get0(idx);
        return pt->check_trust(pt, x, flags);
}

int X509_TRUST_get_count(void)
{
        if(!trtable) return X509_TRUST_COUNT;
        return sk_X509_TRUST_num(trtable) + X509_TRUST_COUNT;
}
```

```
129 X509_TRUST * X509_TRUST_get0(int idx)
130 {
131         if(idx < 0) return NULL;
132         if(idx < (int)X509_TRUST_COUNT) return trstandard + idx;
133         return sk_X509_TRUST_value(trtable, idx - X509_TRUST_COUNT);
134 }

136 int X509_TRUST_get_by_id(int id)
137 {
138         X509_TRUST tmp;
139         int idx;
140         if((id >= X509_TRUST_MIN) && (id <= X509_TRUST_MAX))
141                                 return id - X509_TRUST_MIN;
142         tmp.trust = id;
143         if(!trtable) return -1;
144         idx = sk_X509_TRUST_find(trtable, &tmp);
145         if(idx == -1) return -1;
146         return idx + X509_TRUST_COUNT;
147 }

149 int X509_TRUST_set(int *t, int trust)
150 {
151         if(X509_TRUST_get_by_id(trust) == -1) {
152                 X509err(X509_F_X509_TRUST_SET, X509_R_INVALID_TRUST);
153                 return 0;
154         }
155         *t = trust;
156         return 1;
157 }

159 int X509_TRUST_add(int id, int flags, int (*ck)(X509_TRUST *, X509 *, int),
160                                         char *name, int arg1, void *arg2)
161 {
162         int idx;
163         X509_TRUST *trtmp;
164         /* This is set according to what we change: application can't set it */
165         flags &= ~X509_TRUST_DYNAMIC;
166         /* This will always be set for application modified trust entries */
167         flags |= X509_TRUST_DYNAMIC_NAME;
168         /* Get existing entry if any */
169         idx = X509_TRUST_get_by_id(id);
170         /* Need a new entry */
171         if(idx == -1) {
172                 if(!(trtmp = OPENSSL_malloc(sizeof(X509_TRUST)))) {
173                         X509err(X509_F_X509_TRUST_ADD,ERR_R_MALLOC_FAILURE);
174                         return 0;
175                 }
176                 trtmp->flags = X509_TRUST_DYNAMIC;
177         } else trtmp = X509_TRUST_get0(idx);

179         /* OPENSSL_free existing name if dynamic */
180         if(trtmp->flags & X509_TRUST_DYNAMIC_NAME) OPENSSL_free(trtmp->name);
181         /* dup supplied name */
182         if(!(trtmp->name = BUF_strdup(name))) {
183                 X509err(X509_F_X509_TRUST_ADD,ERR_R_MALLOC_FAILURE);
184                 return 0;
185         }
186         /* Keep the dynamic flag of existing entry */
187         trtmp->flags &= X509_TRUST_DYNAMIC;
188         /* Set all other flags */
189         trtmp->flags |= flags;

191         trtmp->trust = id;
192         trtmp->check_trust = ck;
193         trtmp->arg1 = arg1;
```



```
194         trtmp->arg2 = arg2;

196         /* If its a new entry manage the dynamic table */
197         if(idx == -1) {
198                 if(!trtable && !(trtable = sk_X509_TRUST_new(tr_cmp))) {
199                         X509err(X509_F_X509_TRUST_ADD,ERR_R_MALLOC_FAILURE);
200                         return 0;
201                 }
202                 if (!sk_X509_TRUST_push(trtable, trtmp)) {
203                         X509err(X509_F_X509_TRUST_ADD,ERR_R_MALLOC_FAILURE);
204                         return 0;
205                 }
206         }
207         return 1;
208 }

210 static void trtable_free(X509_TRUST *p)
211         {
212         if(!p) return;
213         if (p->flags & X509_TRUST_DYNAMIC)
214                 {
215                 if (p->flags & X509_TRUST_DYNAMIC_NAME)
216                         OPENSSL_free(p->name);
217                 OPENSSL_free(p);
218                 }
219         }

221 void X509_TRUST_cleanup(void)
222 {
223         unsigned int i;
224         for(i = 0; i < X509_TRUST_COUNT; i++) trtable_free(trstandard + i);
225         sk_X509_TRUST_pop_free(trtable, trtable_free);
226         trtable = NULL;
227 }

229 int X509_TRUST_get_flags(X509_TRUST *xp)
230 {
231         return xp->flags;
232 }

234 char *X509_TRUST_get0_name(X509_TRUST *xp)
235 {
236         return xp->name;
237 }

239 int X509_TRUST_get_trust(X509_TRUST *xp)
240 {
241         return xp->trust;
242 }

244 static int trust_1oidany(X509_TRUST *trust, X509 *x, int flags)
245 {
246         if(x->aux && (x->aux->trust || x->aux->reject))
247                 return obj_trust(trust->arg1, x, flags);
248         /* we don't have any trust settings: for compatibility
249          * we return trusted if it is self signed
250          */
251         return trust_compat(trust, x, flags);
252 }

254 static int trust_1oid(X509_TRUST *trust, X509 *x, int flags)
255 {
256         if(x->aux) return obj_trust(trust->arg1, x, flags);
257         return X509_TRUST_UNTRUSTED;
258 }
```

```
260 static int trust_compat(X509_TRUST *trust, X509 *x, int flags)
261 {
262         X509_check_purpose(x, -1, 0);
263         if(x->ex_flags & EXFLAG_SS) return X509_TRUST_TRUSTED;
264         else return X509_TRUST_UNTRUSTED;
265 }

267 static int obj_trust(int id, X509 *x, int flags)
268 {
269         ASN1_OBJECT *obj;
270         int i;
271         X509_CERT_AUX *ax;
272         ax = x->aux;
273         if(!ax) return X509_TRUST_UNTRUSTED;
274         if(ax->reject) {
275                 for(i = 0; i < sk_ASN1_OBJECT_num(ax->reject); i++) {
276                         obj = sk_ASN1_OBJECT_value(ax->reject, i);
277                         if(OBJ_obj2nid(obj) == id) return X509_TRUST_REJECTED;
278                 }
279         }
280         if(ax->trust) {
281                 for(i = 0; i < sk_ASN1_OBJECT_num(ax->trust); i++) {
282                         obj = sk_ASN1_OBJECT_value(ax->trust, i);
283                         if(OBJ_obj2nid(obj) == id) return X509_TRUST_TRUSTED;
284                 }
285         }
286         return X509_TRUST_UNTRUSTED;
287 }
288 #endif /* ! codereview */
```

```
**********************************************************
    8356 Wed Aug 13 19:53:25 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_txt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_txt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
#include <errno.h>
```



```
#include "cryptlib.h"
#include <openssl/lhash.h>
#include <openssl/buffer.h>
#include <openssl/evp.h>
#include <openssl/asn1.h>
#include <openssl/x509.h>
#include <openssl/objects.h>

const char *X509_verify_cert_error_string(long n)
        {
        static char buf[100];

        switch ((int)n)
                {
        case X509_V_OK:
                return("ok");
        case X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT:
                return("unable to get issuer certificate");
        case X509_V_ERR_UNABLE_TO_GET_CRL:
                return("unable to get certificate CRL");
        case X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE:
                return("unable to decrypt certificate's signature");
        case X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE:
                return("unable to decrypt CRL's signature");
        case X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY:
                return("unable to decode issuer public key");
        case X509_V_ERR_CERT_SIGNATURE_FAILURE:
                return("certificate signature failure");
        case X509_V_ERR_CRL_SIGNATURE_FAILURE:
                return("CRL signature failure");
        case X509_V_ERR_CERT_NOT_YET_VALID:
                return("certificate is not yet valid");
        case X509_V_ERR_CRL_NOT_YET_VALID:
                return("CRL is not yet valid");
        case X509_V_ERR_CERT_HAS_EXPIRED:
                return("certificate has expired");
        case X509_V_ERR_CRL_HAS_EXPIRED:
                return("CRL has expired");
        case X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD:
                return("format error in certificate's notBefore field");
        case X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD:
                return("format error in certificate's notAfter field");
        case X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD:
                return("format error in CRL's lastUpdate field");
        case X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD:
                return("format error in CRL's nextUpdate field");
        case X509_V_ERR_OUT_OF_MEM:
                return("out of memory");
        case X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT:
                return("self signed certificate");
        case X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN:
                return("self signed certificate in certificate chain");
        case X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY:
                return("unable to get local issuer certificate");
        case X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE:
                return("unable to verify the first certificate");
        case X509_V_ERR_CERT_CHAIN_TOO_LONG:
                return("certificate chain too long");
        case X509_V_ERR_CERT_REVOKED:
                return("certificate revoked");
        case X509_V_ERR_INVALID_CA:
                return ("invalid CA certificate");
        case X509_V_ERR_INVALID_NON_CA:
                return ("invalid non-CA certificate (has CA markings)");
        case X509_V_ERR_PATH_LENGTH_EXCEEDED:
```

```
128                 return ("path length constraint exceeded");
129         case X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED:
130                 return("proxy path length constraint exceeded");
131         case X509_V_ERR_PROXY_CERTIFICATES_NOT_ALLOWED:
132                 return("proxy certificates not allowed, please set the appropria
133         case X509_V_ERR_INVALID_PURPOSE:
134                 return ("unsupported certificate purpose");
135         case X509_V_ERR_CERT_UNTRUSTED:
136                 return ("certificate not trusted");
137         case X509_V_ERR_CERT_REJECTED:
138                 return ("certificate rejected");
139         case X509_V_ERR_APPLICATION_VERIFICATION:
140                 return("application verification failure");
141         case X509_V_ERR_SUBJECT_ISSUER_MISMATCH:
142                 return("subject issuer mismatch");
143         case X509_V_ERR_AKID_SKID_MISMATCH:
144                 return("authority and subject key identifier mismatch");
145         case X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH:
146                 return("authority and issuer serial number mismatch");
147         case X509_V_ERR_KEYUSAGE_NO_CERTSIGN:
148                 return("key usage does not include certificate signing");
149         case X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER:
150                 return("unable to get CRL issuer certificate");
151         case X509_V_ERR_UNHANDLED_CRITICAL_EXTENSION:
152                 return("unhandled critical extension");
153         case X509_V_ERR_KEYUSAGE_NO_CRL_SIGN:
154                 return("key usage does not include CRL signing");
155         case X509_V_ERR_KEYUSAGE_NO_DIGITAL_SIGNATURE:
156                 return("key usage does not include digital signature");
157         case X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION:
158                 return("unhandled critical CRL extension");
159         case X509_V_ERR_INVALID_EXTENSION:
160                 return("invalid or inconsistent certificate extension");
161         case X509_V_ERR_INVALID_POLICY_EXTENSION:
162                 return("invalid or inconsistent certificate policy extension");
163         case X509_V_ERR_NO_EXPLICIT_POLICY:
164                 return("no explicit policy");
165         case X509_V_ERR_DIFFERENT_CRL_SCOPE:
166         return("Different CRL scope");
167         case X509_V_ERR_UNSUPPORTED_EXTENSION_FEATURE:
168         return("Unsupported extension feature");
169         case X509_V_ERR_UNNESTED_RESOURCE:
170                 return("RFC 3779 resource not subset of parent's resources");

172         case X509_V_ERR_PERMITTED_VIOLATION:
173                 return("permitted subtree violation");
174         case X509_V_ERR_EXCLUDED_VIOLATION:
175                 return("excluded subtree violation");
176         case X509_V_ERR_SUBTREE_MINMAX:
177                 return("name constraints minimum and maximum not supported");
178         case X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE:
179                 return("unsupported name constraint type");
180         case X509_V_ERR_UNSUPPORTED_CONSTRAINT_SYNTAX:
181                 return("unsupported or invalid name constraint syntax");
182         case X509_V_ERR_UNSUPPORTED_NAME_SYNTAX:
183                 return("unsupported or invalid name syntax");
184         case X509_V_ERR_CRL_PATH_VALIDATION_ERROR:
185                 return("CRL path validation error");

187         default:
188                 BIO_snprintf(buf,sizeof buf,"error number %ld",n);
189                 return(buf);
190                 }
191         }
192 #endif /* ! codereview */
```

```
**********************************************************
    7778 Wed Aug 13 19:53:25 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_v3.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_v3.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/stack.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

int X509v3_get_ext_count(const STACK_OF(X509_EXTENSION) *x)
        {
        if (x == NULL) return(0);
        return(sk_X509_EXTENSION_num(x));
        }

int X509v3_get_ext_by_NID(const STACK_OF(X509_EXTENSION) *x, int nid,
                          int lastpos)
        {
        ASN1_OBJECT *obj;

        obj=OBJ_nid2obj(nid);
        if (obj == NULL) return(-2);
        return(X509v3_get_ext_by_OBJ(x,obj,lastpos));
        }

int X509v3_get_ext_by_OBJ(const STACK_OF(X509_EXTENSION) *sk, ASN1_OBJECT *obj,
                          int lastpos)
        {
        int n;
        X509_EXTENSION *ex;

        if (sk == NULL) return(-1);
        lastpos++;
        if (lastpos < 0)
                lastpos=0;
        n=sk_X509_EXTENSION_num(sk);
        for ( ; lastpos < n; lastpos++)
                {
                ex=sk_X509_EXTENSION_value(sk,lastpos);
                if (OBJ_cmp(ex->object,obj) == 0)
                        return(lastpos);
                }
        return(-1);
        }

int X509v3_get_ext_by_critical(const STACK_OF(X509_EXTENSION) *sk, int crit,
                               int lastpos)
        {
        int n;
        X509_EXTENSION *ex;

        if (sk == NULL) return(-1);
        lastpos++;
        if (lastpos < 0)
                lastpos=0;
        n=sk_X509_EXTENSION_num(sk);
        for ( ; lastpos < n; lastpos++)
                {
                ex=sk_X509_EXTENSION_value(sk,lastpos);
                if (    ((ex->critical > 0) && crit) ||
                        ((ex->critical <= 0) && !crit))
                        return(lastpos);
                }
        return(-1);
        }

X509_EXTENSION *X509v3_get_ext(const STACK_OF(X509_EXTENSION) *x, int loc)
        {
        if (x == NULL || sk_X509_EXTENSION_num(x) <= loc || loc < 0)
```

```
128                 return NULL;
129         else
130                 return sk_X509_EXTENSION_value(x,loc);
131         }

133 X509_EXTENSION *X509v3_delete_ext(STACK_OF(X509_EXTENSION) *x, int loc)
134         {
135         X509_EXTENSION *ret;

137         if (x == NULL || sk_X509_EXTENSION_num(x) <= loc || loc < 0)
138                 return(NULL);
139         ret=sk_X509_EXTENSION_delete(x,loc);
140         return(ret);
141         }

143 STACK_OF(X509_EXTENSION) *X509v3_add_ext(STACK_OF(X509_EXTENSION) **x,
144                                           X509_EXTENSION *ex, int loc)
145         {
146         X509_EXTENSION *new_ex=NULL;
147         int n;
148         STACK_OF(X509_EXTENSION) *sk=NULL;

150         if (x == NULL)
151                 {
152                 X509err(X509_F_X509V3_ADD_EXT,ERR_R_PASSED_NULL_PARAMETER);
153                 goto err2;
154                 }

156         if (*x == NULL)
157                 {
158                 if ((sk=sk_X509_EXTENSION_new_null()) == NULL)
159                         goto err;
160                 }
161         else
162                 sk= *x;

164         n=sk_X509_EXTENSION_num(sk);
165         if (loc > n) loc=n;
166         else if (loc < 0) loc=n;

168         if ((new_ex=X509_EXTENSION_dup(ex)) == NULL)
169                 goto err2;
170         if (!sk_X509_EXTENSION_insert(sk,new_ex,loc))
171                 goto err;
172         if (*x == NULL)
173                 *x=sk;
174         return(sk);
175 err:
176         X509err(X509_F_X509V3_ADD_EXT,ERR_R_MALLOC_FAILURE);
177 err2:
178         if (new_ex != NULL) X509_EXTENSION_free(new_ex);
179         if (sk != NULL) sk_X509_EXTENSION_free(sk);
180         return(NULL);
181         }

183 X509_EXTENSION *X509_EXTENSION_create_by_NID(X509_EXTENSION **ex, int nid,
184             int crit, ASN1_OCTET_STRING *data)
185         {
186         ASN1_OBJECT *obj;
187         X509_EXTENSION *ret;

189         obj=OBJ_nid2obj(nid);
190         if (obj == NULL)
191                 {
192                 X509err(X509_F_X509_EXTENSION_CREATE_BY_NID,X509_R_UNKNOWN_NID);
193                 return(NULL);
```



```
194                 }
195         ret=X509_EXTENSION_create_by_OBJ(ex,obj,crit,data);
196         if (ret == NULL) ASN1_OBJECT_free(obj);
197         return(ret);
198         }

200 X509_EXTENSION *X509_EXTENSION_create_by_OBJ(X509_EXTENSION **ex,
201              ASN1_OBJECT *obj, int crit, ASN1_OCTET_STRING *data)
202         {
203         X509_EXTENSION *ret;

205         if ((ex == NULL) || (*ex == NULL))
206                 {
207                 if ((ret=X509_EXTENSION_new()) == NULL)
208                         {
209                         X509err(X509_F_X509_EXTENSION_CREATE_BY_OBJ,ERR_R_MALLOC
210                         return(NULL);
211                         }
212                 }
213         else
214                 ret= *ex;

216         if (!X509_EXTENSION_set_object(ret,obj))
217                 goto err;
218         if (!X509_EXTENSION_set_critical(ret,crit))
219                 goto err;
220         if (!X509_EXTENSION_set_data(ret,data))
221                 goto err;

223         if ((ex != NULL) && (*ex == NULL)) *ex=ret;
224         return(ret);
225 err:
226         if ((ex == NULL) || (ret != *ex))
227                 X509_EXTENSION_free(ret);
228         return(NULL);
229         }

231 int X509_EXTENSION_set_object(X509_EXTENSION *ex, ASN1_OBJECT *obj)
232         {
233         if ((ex == NULL) || (obj == NULL))
234                 return(0);
235         ASN1_OBJECT_free(ex->object);
236         ex->object=OBJ_dup(obj);
237         return(1);
238         }

240 int X509_EXTENSION_set_critical(X509_EXTENSION *ex, int crit)
241         {
242         if (ex == NULL) return(0);
243         ex->critical=(crit)?0xFF:-1;
244         return(1);
245         }

247 int X509_EXTENSION_set_data(X509_EXTENSION *ex, ASN1_OCTET_STRING *data)
248         {
249         int i;

251         if (ex == NULL) return(0);
252         i=M_ASN1_OCTET_STRING_set(ex->value,data->data,data->length);
253         if (!i) return(0);
254         return(1);
255         }

257 ASN1_OBJECT *X509_EXTENSION_get_object(X509_EXTENSION *ex)
258         {
259         if (ex == NULL) return(NULL);
```

```
260         return(ex->object);
261         }

263 ASN1_OCTET_STRING *X509_EXTENSION_get_data(X509_EXTENSION *ex)
264         {
265         if (ex == NULL) return(NULL);
266         return(ex->value);
267         }

269 int X509_EXTENSION_get_critical(X509_EXTENSION *ex)
270         {
271         if (ex == NULL) return(0);
272         if(ex->critical > 0) return 1;
273         return 0;
274         }
275 #endif /* ! codereview */
```

```
**********************************************************
   53170 Wed Aug 13 19:53:26 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_vfy.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509_vfy.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <time.h>
#include <errno.h>
```



```
#include "cryptlib.h"
#include <openssl/crypto.h>
#include <openssl/lhash.h>
#include <openssl/buffer.h>
#include <openssl/evp.h>
#include <openssl/asn1.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
#include <openssl/objects.h>

/* CRL score values */

/* No unhandled critical extensions */

#define CRL_SCORE_NOCRITICAL    0x100

/* certificate is within CRL scope */

#define CRL_SCORE_SCOPE         0x080

/* CRL times valid */

#define CRL_SCORE_TIME          0x040

/* Issuer name matches certificate */

#define CRL_SCORE_ISSUER_NAME   0x020

/* If this score or above CRL is probably valid */

#define CRL_SCORE_VALID (CRL_SCORE_NOCRITICAL|CRL_SCORE_TIME|CRL_SCORE_SCOPE)

/* CRL issuer is certificate issuer */

#define CRL_SCORE_ISSUER_CERT   0x018

/* CRL issuer is on certificate path */

#define CRL_SCORE_SAME_PATH     0x008

/* CRL issuer matches CRL AKID */

#define CRL_SCORE_AKID          0x004

/* Have a delta CRL with valid times */

#define CRL_SCORE_TIME_DELTA    0x002

static int null_callback(int ok,X509_STORE_CTX *e);
static int check_issued(X509_STORE_CTX *ctx, X509 *x, X509 *issuer);
static X509 *find_issuer(X509_STORE_CTX *ctx, STACK_OF(X509) *sk, X509 *x);
static int check_chain_extensions(X509_STORE_CTX *ctx);
static int check_name_constraints(X509_STORE_CTX *ctx);
static int check_trust(X509_STORE_CTX *ctx);
static int check_revocation(X509_STORE_CTX *ctx);
static int check_cert(X509_STORE_CTX *ctx);
static int check_policy(X509_STORE_CTX *ctx);

static int get_crl_score(X509_STORE_CTX *ctx, X509 **pissuer,
                         unsigned int *preasons,
                         X509_CRL *crl, X509 *x);
static int get_crl_delta(X509_STORE_CTX *ctx,
                                X509_CRL **pcrl, X509_CRL **pdcrl, X509 *x);
static void get_delta_sk(X509_STORE_CTX *ctx, X509_CRL **dcrl, int *pcrl_score,
                         X509_CRL *base, STACK_OF(X509_CRL) *crls);
```

```
128 static void crl_akid_check(X509_STORE_CTX *ctx, X509_CRL *crl,
129                              X509 **pissuer, int *pcrl_score);
130 static int crl_crldp_check(X509 *x, X509_CRL *crl, int crl_score,
131                            unsigned int *preasons);
132 static int check_crl_path(X509_STORE_CTX *ctx, X509 *x);
133 static int check_crl_chain(X509_STORE_CTX *ctx,
134                            STACK_OF(X509) *cert_path,
135                            STACK_OF(X509) *crl_path);

137 static int internal_verify(X509_STORE_CTX *ctx);
138 const char X509_version[]="X.509" OPENSSL_VERSION_PTEXT;


141 static int null_callback(int ok, X509_STORE_CTX *e)
142         {
143         return ok;
144         }

146 #if 0
147 static int x509_subject_cmp(X509 **a, X509 **b)
148         {
149         return X509_subject_name_cmp(*a,*b);
150         }
151 #endif

153 int X509_verify_cert(X509_STORE_CTX *ctx)
154         {
155         X509 *x,*xtmp,*chain_ss=NULL;
156         int bad_chain = 0;
157         X509_VERIFY_PARAM *param = ctx->param;
158         int depth,i,ok=0;
159         int num;
160         int (*cb)(int xok,X509_STORE_CTX *xctx);
161         STACK_OF(X509) *sktmp=NULL;
162         if (ctx->cert == NULL)
163                 {
164                 X509err(X509_F_X509_VERIFY_CERT,X509_R_NO_CERT_SET_FOR_US_TO_VER
165                 return -1;
166                 }

168         cb=ctx->verify_cb;

170         /* first we make sure the chain we are going to build is
171          * present and that the first entry is in place */
172         if (ctx->chain == NULL)
173                 {
174                 if (    ((ctx->chain=sk_X509_new_null()) == NULL) ||
175                         (!sk_X509_push(ctx->chain,ctx->cert)))
176                         {
177                         X509err(X509_F_X509_VERIFY_CERT,ERR_R_MALLOC_FAILURE);
178                         goto end;
179                         }
180                 CRYPTO_add(&ctx->cert->references,1,CRYPTO_LOCK_X509);
181                 ctx->last_untrusted=1;
182                 }

184         /* We use a temporary STACK so we can chop and hack at it */
185         if (ctx->untrusted != NULL
186             && (sktmp=sk_X509_dup(ctx->untrusted)) == NULL)
187                 {
188                 X509err(X509_F_X509_VERIFY_CERT,ERR_R_MALLOC_FAILURE);
189                 goto end;
190                 }

192         num=sk_X509_num(ctx->chain);
193         x=sk_X509_value(ctx->chain,num-1);
```



```
194         depth=param->depth;


197         for (;;)
198                 {
199                 /* If we have enough, we break */
200                 if (depth < num) break; /* FIXME: If this happens, we should tak
201                                          * note of it and, if appropriate, use t
202                                          * X509_V_ERR_CERT_CHAIN_TOO_LONG error
203                                          * code later.
204                                          */

206                 /* If we are self signed, we break */
207                 if (ctx->check_issued(ctx, x,x)) break;

209                 /* If we were passed a cert chain, use it first */
210                 if (ctx->untrusted != NULL)
211                         {
212                         xtmp=find_issuer(ctx, sktmp,x);
213                         if (xtmp != NULL)
214                                 {
215                                 if (!sk_X509_push(ctx->chain,xtmp))
216                                         {
217                                         X509err(X509_F_X509_VERIFY_CERT,ERR_R_MA
218                                         goto end;
219                                         }
220                                 CRYPTO_add(&xtmp->references,1,CRYPTO_LOCK_X509)
221                                 (void)sk_X509_delete_ptr(sktmp,xtmp);
222                                 ctx->last_untrusted++;
223                                 x=xtmp;
224                                 num++;
225                                 /* reparse the full chain for
226                                  * the next one */
227                                 continue;
228                                 }
229                         }
230                 break;
231                 }

233         /* at this point, chain should contain a list of untrusted
234          * certificates.  We now need to add at least one trusted one,
235          * if possible, otherwise we complain. */

237         /* Examine last certificate in chain and see if it
238          * is self signed.
239          */

241         i=sk_X509_num(ctx->chain);
242         x=sk_X509_value(ctx->chain,i-1);
243         if (ctx->check_issued(ctx, x, x))
244                 {
245                 /* we have a self signed certificate */
246                 if (sk_X509_num(ctx->chain) == 1)
247                         {
248                         /* We have a single self signed certificate: see if
249                          * we can find it in the store. We must have an exact
250                          * match to avoid possible impersonation.
251                          */
252                         ok = ctx->get_issuer(&xtmp, ctx, x);
253                         if ((ok <= 0) || X509_cmp(x, xtmp))
254                                 {
255                                 ctx->error=X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CER
256                                 ctx->current_cert=x;
257                                 ctx->error_depth=i-1;
258                                 if (ok == 1) X509_free(xtmp);
259                                 bad_chain = 1;
```

```
260                                 ok=cb(0,ctx);
261                                 if (!ok) goto end;
262                                 }
263                         else
264                                 {
265                                 /* We have a match: replace certificate with sto
266                                  * so we get any trust settings.
267                                  */
268                                 X509_free(x);
269                                 x = xtmp;
270                                 (void)sk_X509_set(ctx->chain, i - 1, x);
271                                 ctx->last_untrusted=0;
272                                 }
273                         }
274                 else
275                         {
276                         /* extract and save self signed certificate for later us
277                         chain_ss=sk_X509_pop(ctx->chain);
278                         ctx->last_untrusted--;
279                         num--;
280                         x=sk_X509_value(ctx->chain,num-1);
281                         }
282                 }

284         /* We now lookup certs from the certificate store */
285         for (;;)
286                 {
287                 /* If we have enough, we break */
288                 if (depth < num) break;

290                 /* If we are self signed, we break */
291                 if (ctx->check_issued(ctx,x,x)) break;

293                 ok = ctx->get_issuer(&xtmp, ctx, x);

295                 if (ok < 0) return ok;
296                 if (ok == 0) break;

298                 x = xtmp;
299                 if (!sk_X509_push(ctx->chain,x))
300                         {
301                         X509_free(xtmp);
302                         X509err(X509_F_X509_VERIFY_CERT,ERR_R_MALLOC_FAILURE);
303                         return 0;
304                         }
305                 num++;
306                 }

308         /* we now have our chain, lets check it... */

310         /* Is last certificate looked up self signed? */
311         if (!ctx->check_issued(ctx,x,x))
312                 {
313                 if ((chain_ss == NULL) || !ctx->check_issued(ctx, x, chain_ss))
314                         {
315                         if (ctx->last_untrusted >= num)
316                                 ctx->error=X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_
317                         else
318                                 ctx->error=X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT;
319                         ctx->current_cert=x;
320                         }
321                 else
322                         {

324                         sk_X509_push(ctx->chain,chain_ss);
325                         num++;
```



```
326                         ctx->last_untrusted=num;
327                         ctx->current_cert=chain_ss;
328                         ctx->error=X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN;
329                         chain_ss=NULL;
330                         }

332                 ctx->error_depth=num-1;
333                 bad_chain = 1;
334                 ok=cb(0,ctx);
335                 if (!ok) goto end;
336                 }

338         /* We have the chain complete: now we need to check its purpose */
339         ok = check_chain_extensions(ctx);

341         if (!ok) goto end;

343         /* Check name constraints */

345         ok = check_name_constraints(ctx);

347         if (!ok) goto end;

349         /* The chain extensions are OK: check trust */

351         if (param->trust > 0) ok = check_trust(ctx);

353         if (!ok) goto end;

355         /* We may as well copy down any DSA parameters that are required */
356         X509_get_pubkey_parameters(NULL,ctx->chain);

358         /* Check revocation status: we do this after copying parameters
359          * because they may be needed for CRL signature verification.
360          */

362         ok = ctx->check_revocation(ctx);
363         if(!ok) goto end;

365         /* At this point, we have a chain and need to verify it */
366         if (ctx->verify != NULL)
367                 ok=ctx->verify(ctx);
368         else
369                 ok=internal_verify(ctx);
370         if(!ok) goto end;

372 #ifndef OPENSSL_NO_RFC3779
373         /* RFC 3779 path validation, now that CRL check has been done */
374         ok = v3_asid_validate_path(ctx);
375         if (!ok) goto end;
376         ok = v3_addr_validate_path(ctx);
377         if (!ok) goto end;
378 #endif

380         /* If we get this far evaluate policies */
381         if (!bad_chain && (ctx->param->flags & X509_V_FLAG_POLICY_CHECK))
382                 ok = ctx->check_policy(ctx);
383         if(!ok) goto end;
384         if (0)
385                 {
386 end:
387                 X509_get_pubkey_parameters(NULL,ctx->chain);
388                 }
389         if (sktmp != NULL) sk_X509_free(sktmp);
390         if (chain_ss != NULL) X509_free(chain_ss);
391         return ok;
```

```
392         }


395 /* Given a STACK_OF(X509) find the issuer of cert (if any)
396  */

398 static X509 *find_issuer(X509_STORE_CTX *ctx, STACK_OF(X509) *sk, X509 *x)
399 {
400         int i;
401         X509 *issuer;
402         for (i = 0; i < sk_X509_num(sk); i++)
403                 {
404                 issuer = sk_X509_value(sk, i);
405                 if (ctx->check_issued(ctx, x, issuer))
406                         return issuer;
407                 }
408         return NULL;
409 }

411 /* Given a possible certificate and issuer check them */

413 static int check_issued(X509_STORE_CTX *ctx, X509 *x, X509 *issuer)
414 {
415         int ret;
416         ret = X509_check_issued(issuer, x);
417         if (ret == X509_V_OK)
418                 return 1;
419         /* If we haven't asked for issuer errors don't set ctx */
420         if (!(ctx->param->flags & X509_V_FLAG_CB_ISSUER_CHECK))
421                 return 0;

423         ctx->error = ret;
424         ctx->current_cert = x;
425         ctx->current_issuer = issuer;
426         return ctx->verify_cb(0, ctx);
427         return 0;
428 }

430 /* Alternative lookup method: look from a STACK stored in other_ctx */

432 static int get_issuer_sk(X509 **issuer, X509_STORE_CTX *ctx, X509 *x)
433 {
434         *issuer = find_issuer(ctx, ctx->other_ctx, x);
435         if (*issuer)
436                 {
437                 CRYPTO_add(&(*issuer)->references,1,CRYPTO_LOCK_X509);
438                 return 1;
439                 }
440         else
441                 return 0;
442 }


445 /* Check a certificate chains extensions for consistency
446  * with the supplied purpose
447  */

449 static int check_chain_extensions(X509_STORE_CTX *ctx)
450 {
451 #ifdef OPENSSL_NO_CHAIN_VERIFY
452         return 1;
453 #else
454         int i, ok=0, must_be_ca, plen = 0;
455         X509 *x;
456         int (*cb)(int xok,X509_STORE_CTX *xctx);
457         int proxy_path_length = 0;
```



```
458         int purpose;
459         int allow_proxy_certs;
460         cb=ctx->verify_cb;

462         /* must_be_ca can have 1 of 3 values:
463            -1: we accept both CA and non-CA certificates, to allow direct
464                use of self-signed certificates (which are marked as CA).
465            0:  we only accept non-CA certificates.  This is currently not
466                used, but the possibility is present for future extensions.
467            1:  we only accept CA certificates.  This is currently used for
468                all certificates in the chain except the leaf certificate.
469         */
470         must_be_ca = -1;

472         /* CRL path validation */
473         if (ctx->parent)
474                 {
475                 allow_proxy_certs = 0;
476                 purpose = X509_PURPOSE_CRL_SIGN;
477                 }
478         else
479                 {
480                 allow_proxy_certs =
481                         !!(ctx->param->flags & X509_V_FLAG_ALLOW_PROXY_CERTS);
482                 /* A hack to keep people who don't want to modify their
483                    software happy */
484                 if (getenv("OPENSSL_ALLOW_PROXY_CERTS"))
485                         allow_proxy_certs = 1;
486                 purpose = ctx->param->purpose;
487                 }

489         /* Check all untrusted certificates */
490         for (i = 0; i < ctx->last_untrusted; i++)
491                 {
492                 int ret;
493                 x = sk_X509_value(ctx->chain, i);
494                 if (!(ctx->param->flags & X509_V_FLAG_IGNORE_CRITICAL)
495                         && (x->ex_flags & EXFLAG_CRITICAL))
496                         {
497                         ctx->error = X509_V_ERR_UNHANDLED_CRITICAL_EXTENSION;
498                         ctx->error_depth = i;
499                         ctx->current_cert = x;
500                         ok=cb(0,ctx);
501                         if (!ok) goto end;
502                         }
503                 if (!allow_proxy_certs && (x->ex_flags & EXFLAG_PROXY))
504                         {
505                         ctx->error = X509_V_ERR_PROXY_CERTIFICATES_NOT_ALLOWED;
506                         ctx->error_depth = i;
507                         ctx->current_cert = x;
508                         ok=cb(0,ctx);
509                         if (!ok) goto end;
510                         }
511                 ret = X509_check_ca(x);
512                 switch(must_be_ca)
513                         {
514                 case -1:
515                         if ((ctx->param->flags & X509_V_FLAG_X509_STRICT)
516                                 && (ret != 1) && (ret != 0))
517                                 {
518                                 ret = 0;
519                                 ctx->error = X509_V_ERR_INVALID_CA;
520                                 }
521                         else
522                                 ret = 1;
523                         break;
```

```
524                 case 0:
525                         if (ret != 0)
526                                 {
527                                 ret = 0;
528                                 ctx->error = X509_V_ERR_INVALID_NON_CA;
529                                 }
530                         else
531                                 ret = 1;
532                         break;
533                 default:
534                         if ((ret == 0)
535                                 || ((ctx->param->flags & X509_V_FLAG_X509_STRICT
536                                         && (ret != 1)))
537                                 {
538                                 ret = 0;
539                                 ctx->error = X509_V_ERR_INVALID_CA;
540                                 }
541                         else
542                                 ret = 1;
543                         break;
544                         }
545                 if (ret == 0)
546                         {
547                         ctx->error_depth = i;
548                         ctx->current_cert = x;
549                         ok=cb(0,ctx);
550                         if (!ok) goto end;
551                         }
552                 if (ctx->param->purpose > 0)
553                         {
554                         ret = X509_check_purpose(x, purpose, must_be_ca > 0);
555                         if ((ret == 0)
556                                 || ((ctx->param->flags & X509_V_FLAG_X509_STRICT
557                                         && (ret != 1)))
558                                 {
559                                 ctx->error = X509_V_ERR_INVALID_PURPOSE;
560                                 ctx->error_depth = i;
561                                 ctx->current_cert = x;
562                                 ok=cb(0,ctx);
563                                 if (!ok) goto end;
564                                 }
565                         }
566                 /* Check pathlen if not self issued */
567                 if ((i > 1) && !(x->ex_flags & EXFLAG_SI)
568                           && (x->ex_pathlen != -1)
569                           && (plen > (x->ex_pathlen + proxy_path_length + 1)))
570                         {
571                         ctx->error = X509_V_ERR_PATH_LENGTH_EXCEEDED;
572                         ctx->error_depth = i;
573                         ctx->current_cert = x;
574                         ok=cb(0,ctx);
575                         if (!ok) goto end;
576                         }
577                 /* Increment path length if not self issued */
578                 if (!(x->ex_flags & EXFLAG_SI))
579                         plen++;
580                 /* If this certificate is a proxy certificate, the next
581                    certificate must be another proxy certificate or a EE
582                    certificate.  If not, the next certificate must be a
583                    CA certificate.  */
584                 if (x->ex_flags & EXFLAG_PROXY)
585                         {
586                         if (x->ex_pcpathlen != -1 && i > x->ex_pcpathlen)
587                                 {
588                                 ctx->error =
589                                         X509_V_ERR_PROXY_PATH_LENGTH_EXCEEDED;
```



```
590                                 ctx->error_depth = i;
591                                 ctx->current_cert = x;
592                                 ok=cb(0,ctx);
593                                 if (!ok) goto end;
594                                 }
595                         proxy_path_length++;
596                         must_be_ca = 0;
597                         }
598                 else
599                         must_be_ca = 1;
600                 }
601         ok = 1;
602  end:
603         return ok;
604 #endif
605 }

607 static int check_name_constraints(X509_STORE_CTX *ctx)
608         {
609         X509 *x;
610         int i, j, rv;
611         /* Check name constraints for all certificates */
612         for (i = sk_X509_num(ctx->chain) - 1; i >= 0; i--)
613                 {
614                 x = sk_X509_value(ctx->chain, i);
615                 /* Ignore self issued certs unless last in chain */
616                 if (i && (x->ex_flags & EXFLAG_SI))
617                         continue;
618                 /* Check against constraints for all certificates higher in
619                  * chain including trust anchor. Trust anchor not strictly
620                  * speaking needed but if it includes constraints it is to be
621                  * assumed it expects them to be obeyed.
622                  */
623                 for (j = sk_X509_num(ctx->chain) - 1; j > i; j--)
624                         {
625                         NAME_CONSTRAINTS *nc = sk_X509_value(ctx->chain, j)->nc;
626                         if (nc)
627                                 {
628                                 rv = NAME_CONSTRAINTS_check(x, nc);
629                                 if (rv != X509_V_OK)
630                                         {
631                                         ctx->error = rv;
632                                         ctx->error_depth = i;
633                                         ctx->current_cert = x;
634                                         if (!ctx->verify_cb(0,ctx))
635                                                 return 0;
636                                         }
637                                 }
638                         }
639                 }
640         return 1;
641         }

643 static int check_trust(X509_STORE_CTX *ctx)
644 {
645 #ifdef OPENSSL_NO_CHAIN_VERIFY
646         return 1;
647 #else
648         int i, ok;
649         X509 *x;
650         int (*cb)(int xok,X509_STORE_CTX *xctx);
651         cb=ctx->verify_cb;
652 /* For now just check the last certificate in the chain */
653         i = sk_X509_num(ctx->chain) - 1;
654         x = sk_X509_value(ctx->chain, i);
655         ok = X509_check_trust(x, ctx->param->trust, 0);
```

```
656         if (ok == X509_TRUST_TRUSTED)
657                 return 1;
658         ctx->error_depth = i;
659         ctx->current_cert = x;
660         if (ok == X509_TRUST_REJECTED)
661                 ctx->error = X509_V_ERR_CERT_REJECTED;
662         else
663                 ctx->error = X509_V_ERR_CERT_UNTRUSTED;
664         ok = cb(0, ctx);
665         return ok;
666 #endif
667 }

669 static int check_revocation(X509_STORE_CTX *ctx)
670         {
671         int i, last, ok;
672         if (!(ctx->param->flags & X509_V_FLAG_CRL_CHECK))
673                 return 1;
674         if (ctx->param->flags & X509_V_FLAG_CRL_CHECK_ALL)
675                 last = sk_X509_num(ctx->chain) - 1;
676         else
677                 {
678                 /* If checking CRL paths this isn't the EE certificate */
679                 if (ctx->parent)
680                         return 1;
681                 last = 0;
682                 }
683         for(i = 0; i <= last; i++)
684                 {
685                 ctx->error_depth = i;
686                 ok = check_cert(ctx);
687                 if (!ok) return ok;
688                 }
689         return 1;
690         }

692 static int check_cert(X509_STORE_CTX *ctx)
693         {
694         X509_CRL *crl = NULL, *dcrl = NULL;
695         X509 *x;
696         int ok, cnum;
697         unsigned int last_reasons;
698         cnum = ctx->error_depth;
699         x = sk_X509_value(ctx->chain, cnum);
700         ctx->current_cert = x;
701         ctx->current_issuer = NULL;
702         ctx->current_crl_score = 0;
703         ctx->current_reasons = 0;
704         while (ctx->current_reasons != CRLDP_ALL_REASONS)
705                 {
706                 last_reasons = ctx->current_reasons;
707                 /* Try to retrieve relevant CRL */
708                 if (ctx->get_crl)
709                         ok = ctx->get_crl(ctx, &crl, x);
710                 else
711                         ok = get_crl_delta(ctx, &crl, &dcrl, x);
712                 /* If error looking up CRL, nothing we can do except
713                  * notify callback
714                  */
715                 if(!ok)
716                         {
717                         ctx->error = X509_V_ERR_UNABLE_TO_GET_CRL;
718                         ok = ctx->verify_cb(0, ctx);
719                         goto err;
720                         }
721                 ctx->current_crl = crl;
```



```
722                 ok = ctx->check_crl(ctx, crl);
723                 if (!ok)
724                         goto err;

726                 if (dcrl)
727                         {
728                         ok = ctx->check_crl(ctx, dcrl);
729                         if (!ok)
730                                 goto err;
731                         ok = ctx->cert_crl(ctx, dcrl, x);
732                         if (!ok)
733                                 goto err;
734                         }
735                 else
736                         ok = 1;

738                 /* Don't look in full CRL if delta reason is removefromCRL */
739                 if (ok != 2)
740                         {
741                         ok = ctx->cert_crl(ctx, crl, x);
742                         if (!ok)
743                                 goto err;
744                         }

746                 X509_CRL_free(crl);
747                 X509_CRL_free(dcrl);
748                 crl = NULL;
749                 dcrl = NULL;
750                 /* If reasons not updated we wont get anywhere by
751                  * another iteration, so exit loop.
752                  */
753                 if (last_reasons == ctx->current_reasons)
754                         {
755                         ctx->error = X509_V_ERR_UNABLE_TO_GET_CRL;
756                         ok = ctx->verify_cb(0, ctx);
757                         goto err;
758                         }
759                 }
760         err:
761         X509_CRL_free(crl);
762         X509_CRL_free(dcrl);

764         ctx->current_crl = NULL;
765         return ok;

767         }

769 /* Check CRL times against values in X509_STORE_CTX */

771 static int check_crl_time(X509_STORE_CTX *ctx, X509_CRL *crl, int notify)
772         {
773         time_t *ptime;
774         int i;
775         if (notify)
776                 ctx->current_crl = crl;
777         if (ctx->param->flags & X509_V_FLAG_USE_CHECK_TIME)
778                 ptime = &ctx->param->check_time;
779         else
780                 ptime = NULL;

782         i=X509_cmp_time(X509_CRL_get_lastUpdate(crl), ptime);
783         if (i == 0)
784                 {
785                 if (!notify)
786                         return 0;
787                 ctx->error=X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD;
```

```
788                 if (!ctx->verify_cb(0, ctx))
789                         return 0;
790                 }

792         if (i > 0)
793                 {
794                 if (!notify)
795                         return 0;
796                 ctx->error=X509_V_ERR_CRL_NOT_YET_VALID;
797                 if (!ctx->verify_cb(0, ctx))
798                         return 0;
799                 }

801         if(X509_CRL_get_nextUpdate(crl))
802                 {
803                 i=X509_cmp_time(X509_CRL_get_nextUpdate(crl), ptime);

805                 if (i == 0)
806                         {
807                         if (!notify)
808                                 return 0;
809                         ctx->error=X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD;
810                         if (!ctx->verify_cb(0, ctx))
811                                 return 0;
812                         }
813                 /* Ignore expiry of base CRL is delta is valid */
814                 if ((i < 0) && !(ctx->current_crl_score & CRL_SCORE_TIME_DELTA))
815                         {
816                         if (!notify)
817                                 return 0;
818                         ctx->error=X509_V_ERR_CRL_HAS_EXPIRED;
819                         if (!ctx->verify_cb(0, ctx))
820                                 return 0;
821                         }
822                 }

824         if (notify)
825                 ctx->current_crl = NULL;

827         return 1;
828         }

830 static int get_crl_sk(X509_STORE_CTX *ctx, X509_CRL **pcrl, X509_CRL **pdcrl,
831                       X509 **pissuer, int *pscore, unsigned int *preasons,
832                       STACK_OF(X509_CRL) *crls)
833         {
834         int i, crl_score, best_score = *pscore;
835         unsigned int reasons, best_reasons = 0;
836         X509 *x = ctx->current_cert;
837         X509_CRL *crl, *best_crl = NULL;
838         X509 *crl_issuer = NULL, *best_crl_issuer = NULL;

840         for (i = 0; i < sk_X509_CRL_num(crls); i++)
841                 {
842                 crl = sk_X509_CRL_value(crls, i);
843                 reasons = *preasons;
844                 crl_score = get_crl_score(ctx, &crl_issuer, &reasons, crl, x);

846                 if (crl_score > best_score)
847                         {
848                         best_crl = crl;
849                         best_crl_issuer = crl_issuer;
850                         best_score = crl_score;
851                         best_reasons = reasons;
852                         }
853                 }
```



```
855         if (best_crl)
856                 {
857                 if (*pcrl)
858                         X509_CRL_free(*pcrl);
859                 *pcrl = best_crl;
860                 *pissuer = best_crl_issuer;
861                 *pscore = best_score;
862                 *preasons = best_reasons;
863                 CRYPTO_add(&best_crl->references, 1, CRYPTO_LOCK_X509_CRL);
864                 if (*pdcrl)
865                         {
866                         X509_CRL_free(*pdcrl);
867                         *pdcrl = NULL;
868                         }
869                 get_delta_sk(ctx, pdcrl, pscore, best_crl, crls);
870                 }

872         if (best_score >= CRL_SCORE_VALID)
873                 return 1;

875         return 0;
876         }

878 /* Compare two CRL extensions for delta checking purposes. They should be
879  * both present or both absent. If both present all fields must be identical.
880  */

882 static int crl_extension_match(X509_CRL *a, X509_CRL *b, int nid)
883         {
884         ASN1_OCTET_STRING *exta, *extb;
885         int i;
886         i = X509_CRL_get_ext_by_NID(a, nid, -1);
887         if (i >= 0)
888                 {
889                 /* Can't have multiple occurrences */
890                 if (X509_CRL_get_ext_by_NID(a, nid, i) != -1)
891                         return 0;
892                 exta = X509_EXTENSION_get_data(X509_CRL_get_ext(a, i));
893                 }
894         else
895                 exta = NULL;

897         i = X509_CRL_get_ext_by_NID(b, nid, -1);

899         if (i >= 0)
900                 {

902                 if (X509_CRL_get_ext_by_NID(b, nid, i) != -1)
903                         return 0;
904                 extb = X509_EXTENSION_get_data(X509_CRL_get_ext(b, i));
905                 }
906         else
907                 extb = NULL;

909         if (!exta && !extb)
910                 return 1;

912         if (!exta || !extb)
913                 return 0;


916         if (ASN1_OCTET_STRING_cmp(exta, extb))
917                 return 0;

919         return 1;
```

```
920         }

922 /* See if a base and delta are compatible */

924 static int check_delta_base(X509_CRL *delta, X509_CRL *base)
925         {
926         /* Delta CRL must be a delta */
927         if (!delta->base_crl_number)
928                         return 0;
929         /* Base must have a CRL number */
930         if (!base->crl_number)
931                         return 0;
932         /* Issuer names must match */
933         if (X509_NAME_cmp(X509_CRL_get_issuer(base),
934                                 X509_CRL_get_issuer(delta)))
935                 return 0;
936         /* AKID and IDP must match */
937         if (!crl_extension_match(delta, base, NID_authority_key_identifier))
938                         return 0;
939         if (!crl_extension_match(delta, base, NID_issuing_distribution_point))
940                         return 0;
941         /* Delta CRL base number must not exceed Full CRL number. */
942         if (ASN1_INTEGER_cmp(delta->base_crl_number, base->crl_number) > 0)
943                         return 0;
944         /* Delta CRL number must exceed full CRL number */
945         if (ASN1_INTEGER_cmp(delta->crl_number, base->crl_number) > 0)
946                         return 1;
947         return 0;
948         }

950 /* For a given base CRL find a delta... maybe extend to delta scoring
951  * or retrieve a chain of deltas...
952  */

954 static void get_delta_sk(X509_STORE_CTX *ctx, X509_CRL **dcrl, int *pscore,
955                         X509_CRL *base, STACK_OF(X509_CRL) *crls)
956         {
957         X509_CRL *delta;
958         int i;
959         if (!(ctx->param->flags & X509_V_FLAG_USE_DELTAS))
960                 return;
961         if (!((ctx->current_cert->ex_flags | base->flags) & EXFLAG_FRESHEST))
962                 return;
963         for (i = 0; i < sk_X509_CRL_num(crls); i++)
964                 {
965                 delta = sk_X509_CRL_value(crls, i);
966                 if (check_delta_base(delta, base))
967                         {
968                         if (check_crl_time(ctx, delta, 0))
969                                 *pscore |= CRL_SCORE_TIME_DELTA;
970                         CRYPTO_add(&delta->references, 1, CRYPTO_LOCK_X509_CRL);
971                         *dcrl = delta;
972                         return;
973                         }
974                 }
975         *dcrl = NULL;
976         }

978 /* For a given CRL return how suitable it is for the supplied certificate 'x'.
979  * The return value is a mask of several criteria.
980  * If the issuer is not the certificate issuer this is returned in *pissuer.
981  * The reasons mask is also used to determine if the CRL is suitable: if
982  * no new reasons the CRL is rejected, otherwise reasons is updated.
983  */

985 static int get_crl_score(X509_STORE_CTX *ctx, X509 **pissuer,
```



```
986                         unsigned int *preasons,
987                         X509_CRL *crl, X509 *x)
988         {

990         int crl_score = 0;
991         unsigned int tmp_reasons = *preasons, crl_reasons;

993         /* First see if we can reject CRL straight away */

995         /* Invalid IDP cannot be processed */
996         if (crl->idp_flags & IDP_INVALID)
997                 return 0;
998         /* Reason codes or indirect CRLs need extended CRL support */
999         if (!(ctx->param->flags & X509_V_FLAG_EXTENDED_CRL_SUPPORT))
1000                {
1001                if (crl->idp_flags & (IDP_INDIRECT | IDP_REASONS))
1002                        return 0;
1003                }
1004        else if (crl->idp_flags & IDP_REASONS)
1005                {
1006                /* If no new reasons reject */
1007                if (!(crl->idp_reasons & ~tmp_reasons))
1008                        return 0;
1009                }
1010        /* Don't process deltas at this stage */
1011        else if (crl->base_crl_number)
1012                return 0;
1013        /* If issuer name doesn't match certificate need indirect CRL */
1014        if (X509_NAME_cmp(X509_get_issuer_name(x), X509_CRL_get_issuer(crl)))
1015                {
1016                if (!(crl->idp_flags & IDP_INDIRECT))
1017                        return 0;
1018                }
1019        else
1020                crl_score |= CRL_SCORE_ISSUER_NAME;

1022        if (!(crl->flags & EXFLAG_CRITICAL))
1023                crl_score |= CRL_SCORE_NOCRITICAL;

1025        /* Check expiry */
1026        if (check_crl_time(ctx, crl, 0))
1027                crl_score |= CRL_SCORE_TIME;

1029        /* Check authority key ID and locate certificate issuer */
1030        crl_akid_check(ctx, crl, pissuer, &crl_score);

1032        /* If we can't locate certificate issuer at this point forget it */

1034        if (!(crl_score & CRL_SCORE_AKID))
1035                return 0;

1037        /* Check cert for matching CRL distribution points */

1039        if (crl_crldp_check(x, crl, crl_score, &crl_reasons))
1040                {
1041                /* If no new reasons reject */
1042                if (!(crl_reasons & ~tmp_reasons))
1043                        return 0;
1044                tmp_reasons |= crl_reasons;
1045                crl_score |= CRL_SCORE_SCOPE;
1046                }

1048        *preasons = tmp_reasons;

1050        return crl_score;
```

```
1052         }

1054 static void crl_akid_check(X509_STORE_CTX *ctx, X509_CRL *crl,
1055                             X509 **pissuer, int *pcrl_score)
1056         {
1057         X509 *crl_issuer = NULL;
1058         X509_NAME *cnm = X509_CRL_get_issuer(crl);
1059         int cidx = ctx->error_depth;
1060         int i;

1062         if (cidx != sk_X509_num(ctx->chain) - 1)
1063                 cidx++;

1065         crl_issuer = sk_X509_value(ctx->chain, cidx);

1067         if (X509_check_akid(crl_issuer, crl->akid) == X509_V_OK)
1068                 {
1069                 if (*pcrl_score & CRL_SCORE_ISSUER_NAME)
1070                         {
1071                         *pcrl_score |= CRL_SCORE_AKID|CRL_SCORE_ISSUER_CERT;
1072                         *pissuer = crl_issuer;
1073                         return;
1074                         }
1075                 }

1077         for (cidx++; cidx < sk_X509_num(ctx->chain); cidx++)
1078                 {
1079                 crl_issuer = sk_X509_value(ctx->chain, cidx);
1080                 if (X509_NAME_cmp(X509_get_subject_name(crl_issuer), cnm))
1081                         continue;
1082                 if (X509_check_akid(crl_issuer, crl->akid) == X509_V_OK)
1083                         {
1084                         *pcrl_score |= CRL_SCORE_AKID|CRL_SCORE_SAME_PATH;
1085                         *pissuer = crl_issuer;
1086                         return;
1087                         }
1088                 }

1090         /* Anything else needs extended CRL support */

1092         if (!(ctx->param->flags & X509_V_FLAG_EXTENDED_CRL_SUPPORT))
1093                 return;

1095         /* Otherwise the CRL issuer is not on the path. Look for it in the
1096          * set of untrusted certificates.
1097          */
1098         for (i = 0; i < sk_X509_num(ctx->untrusted); i++)
1099                 {
1100                 crl_issuer = sk_X509_value(ctx->untrusted, i);
1101                 if (X509_NAME_cmp(X509_get_subject_name(crl_issuer), cnm))
1102                         continue;
1103                 if (X509_check_akid(crl_issuer, crl->akid) == X509_V_OK)
1104                         {
1105                         *pissuer = crl_issuer;
1106                         *pcrl_score |= CRL_SCORE_AKID;
1107                         return;
1108                         }
1109                 }
1110         }

1112 /* Check the path of a CRL issuer certificate. This creates a new
1113  * X509_STORE_CTX and populates it with most of the parameters from the
1114  * parent. This could be optimised somewhat since a lot of path checking
1115  * will be duplicated by the parent, but this will rarely be used in
1116  * practice.
1117  */
```



```
1119 static int check_crl_path(X509_STORE_CTX *ctx, X509 *x)
1120         {
1121         X509_STORE_CTX crl_ctx;
1122         int ret;
1123         /* Don't allow recursive CRL path validation */
1124         if (ctx->parent)
1125                 return 0;
1126         if (!X509_STORE_CTX_init(&crl_ctx, ctx->ctx, x, ctx->untrusted))
1127                 return -1;

1129         crl_ctx.crls = ctx->crls;
1130         /* Copy verify params across */
1131         X509_STORE_CTX_set0_param(&crl_ctx, ctx->param);

1133         crl_ctx.parent = ctx;
1134         crl_ctx.verify_cb = ctx->verify_cb;

1136         /* Verify CRL issuer */
1137         ret = X509_verify_cert(&crl_ctx);

1139         if (ret <= 0)
1140                 goto err;

1142         /* Check chain is acceptable */

1144         ret = check_crl_chain(ctx, ctx->chain, crl_ctx.chain);
1145         err:
1146         X509_STORE_CTX_cleanup(&crl_ctx);
1147         return ret;
1148         }

1150 /* RFC3280 says nothing about the relationship between CRL path
1151  * and certificate path, which could lead to situations where a
1152  * certificate could be revoked or validated by a CA not authorised
1153  * to do so. RFC5280 is more strict and states that the two paths must
1154  * end in the same trust anchor, though some discussions remain...
1155  * until this is resolved we use the RFC5280 version
1156  */

1158 static int check_crl_chain(X509_STORE_CTX *ctx,
1159                            STACK_OF(X509) *cert_path,
1160                            STACK_OF(X509) *crl_path)
1161         {
1162         X509 *cert_ta, *crl_ta;
1163         cert_ta = sk_X509_value(cert_path, sk_X509_num(cert_path) - 1);
1164         crl_ta = sk_X509_value(crl_path, sk_X509_num(crl_path) - 1);
1165         if (!X509_cmp(cert_ta, crl_ta))
1166                 return 1;
1167         return 0;
1168         }

1170 /* Check for match between two dist point names: three separate cases.
1171  * 1. Both are relative names and compare X509_NAME types.
1172  * 2. One full, one relative. Compare X509_NAME to GENERAL_NAMES.
1173  * 3. Both are full names and compare two GENERAL_NAMES.
1174  * 4. One is NULL: automatic match.
1175  */


1178 static int idp_check_dp(DIST_POINT_NAME *a, DIST_POINT_NAME *b)
1179         {
1180         X509_NAME *nm = NULL;
1181         GENERAL_NAMES *gens = NULL;
1182         GENERAL_NAME *gena, *genb;
1183         int i, j;
```

```
1184         if (!a || !b)
1185                 return 1;
1186         if (a->type == 1)
1187                 {
1188                 if (!a->dpname)
1189                         return 0;
1190                 /* Case 1: two X509_NAME */
1191                 if (b->type == 1)
1192                         {
1193                         if (!b->dpname)
1194                                 return 0;
1195                         if (!X509_NAME_cmp(a->dpname, b->dpname))
1196                                 return 1;
1197                         else
1198                                 return 0;
1199                         }
1200                 /* Case 2: set name and GENERAL_NAMES appropriately */
1201                 nm = a->dpname;
1202                 gens = b->name.fullname;
1203                 }
1204         else if (b->type == 1)
1205                 {
1206                 if (!b->dpname)
1207                         return 0;
1208                 /* Case 2: set name and GENERAL_NAMES appropriately */
1209                 gens = a->name.fullname;
1210                 nm = b->dpname;
1211                 }

1213         /* Handle case 2 with one GENERAL_NAMES and one X509_NAME */
1214         if (nm)
1215                 {
1216                 for (i = 0; i < sk_GENERAL_NAME_num(gens); i++)
1217                         {
1218                         gena = sk_GENERAL_NAME_value(gens, i);
1219                         if (gena->type != GEN_DIRNAME)
1220                                 continue;
1221                         if (!X509_NAME_cmp(nm, gena->d.directoryName))
1222                                 return 1;
1223                         }
1224                 return 0;
1225                 }

1227         /* Else case 3: two GENERAL_NAMES */

1229         for (i = 0; i < sk_GENERAL_NAME_num(a->name.fullname); i++)
1230                 {
1231                 gena = sk_GENERAL_NAME_value(a->name.fullname, i);
1232                 for (j = 0; j < sk_GENERAL_NAME_num(b->name.fullname); j++)
1233                         {
1234                         genb = sk_GENERAL_NAME_value(b->name.fullname, j);
1235                         if (!GENERAL_NAME_cmp(gena, genb))
1236                                 return 1;
1237                         }
1238                 }

1240         return 0;

1242         }

1244 static int crldp_check_crlissuer(DIST_POINT *dp, X509_CRL *crl, int crl_score)
1245         {
1246         int i;
1247         X509_NAME *nm = X509_CRL_get_issuer(crl);
1248         /* If no CRLissuer return is successful iff don't need a match */
1249         if (!dp->CRLissuer)
```



```
1250                 return !!(crl_score & CRL_SCORE_ISSUER_NAME);
1251         for (i = 0; i < sk_GENERAL_NAME_num(dp->CRLissuer); i++)
1252                 {
1253                 GENERAL_NAME *gen = sk_GENERAL_NAME_value(dp->CRLissuer, i);
1254                 if (gen->type != GEN_DIRNAME)
1255                         continue;
1256                 if (!X509_NAME_cmp(gen->d.directoryName, nm))
1257                         return 1;
1258                 }
1259         return 0;
1260         }

1262 /* Check CRLDP and IDP */

1264 static int crl_crldp_check(X509 *x, X509_CRL *crl, int crl_score,
1265                                 unsigned int *preasons)
1266         {
1267         int i;
1268         if (crl->idp_flags & IDP_ONLYATTR)
1269                 return 0;
1270         if (x->ex_flags & EXFLAG_CA)
1271                 {
1272                 if (crl->idp_flags & IDP_ONLYUSER)
1273                         return 0;
1274                 }
1275         else
1276                 {
1277                 if (crl->idp_flags & IDP_ONLYCA)
1278                         return 0;
1279                 }
1280         *preasons = crl->idp_reasons;
1281         for (i = 0; i < sk_DIST_POINT_num(x->crldp); i++)
1282                 {
1283                 DIST_POINT *dp = sk_DIST_POINT_value(x->crldp, i);
1284                 if (crldp_check_crlissuer(dp, crl, crl_score))
1285                         {
1286                         if (!crl->idp ||
1287                             idp_check_dp(dp->distpoint, crl->idp->distpoint))
1288                                 {
1289                                 *preasons &= dp->dp_reasons;
1290                                 return 1;
1291                                 }
1292                         }
1293                 }
1294         if ((!crl->idp || !crl->idp->distpoint) && (crl_score & CRL_SCORE_ISSUER
1295                 return 1;
1296         return 0;
1297         }

1299 /* Retrieve CRL corresponding to current certificate.
1300  * If deltas enabled try to find a delta CRL too
1301  */

1303 static int get_crl_delta(X509_STORE_CTX *ctx,
1304                                 X509_CRL **pcrl, X509_CRL **pdcrl, X509 *x)
1305         {
1306         int ok;
1307         X509 *issuer = NULL;
1308         int crl_score = 0;
1309         unsigned int reasons;
1310         X509_CRL *crl = NULL, *dcrl = NULL;
1311         STACK_OF(X509_CRL) *skcrl;
1312         X509_NAME *nm = X509_get_issuer_name(x);
1313         reasons = ctx->current_reasons;
1314         ok = get_crl_sk(ctx, &crl, &dcrl,
1315                                 &issuer, &crl_score, &reasons, ctx->crls);
```

```
1317         if (ok)
1318                 goto done;

1320         /* Lookup CRLs from store */

1322         skcrl = ctx->lookup_crls(ctx, nm);

1324         /* If no CRLs found and a near match from get_crl_sk use that */
1325         if (!skcrl && crl)
1326                 goto done;

1328         get_crl_sk(ctx, &crl, &dcrl, &issuer, &crl_score, &reasons, skcrl);

1330         sk_X509_CRL_pop_free(skcrl, X509_CRL_free);

1332         done:

1334         /* If we got any kind of CRL use it and return success */
1335         if (crl)
1336                 {
1337                 ctx->current_issuer = issuer;
1338                 ctx->current_crl_score = crl_score;
1339                 ctx->current_reasons = reasons;
1340                 *pcrl = crl;
1341                 *pdcrl = dcrl;
1342                 return 1;
1343                 }

1345         return 0;
1346         }

1348 /* Check CRL validity */
1349 static int check_crl(X509_STORE_CTX *ctx, X509_CRL *crl)
1350         {
1351         X509 *issuer = NULL;
1352         EVP_PKEY *ikey = NULL;
1353         int ok = 0, chnum, cnum;
1354         cnum = ctx->error_depth;
1355         chnum = sk_X509_num(ctx->chain) - 1;
1356         /* if we have an alternative CRL issuer cert use that */
1357         if (ctx->current_issuer)
1358                 issuer = ctx->current_issuer;

1360         /* Else find CRL issuer: if not last certificate then issuer
1361          * is next certificate in chain.
1362          */
1363         else if (cnum < chnum)
1364                 issuer = sk_X509_value(ctx->chain, cnum + 1);
1365         else
1366                 {
1367                 issuer = sk_X509_value(ctx->chain, chnum);
1368                 /* If not self signed, can't check signature */
1369                 if(!ctx->check_issued(ctx, issuer, issuer))
1370                         {
1371                         ctx->error = X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER;
1372                         ok = ctx->verify_cb(0, ctx);
1373                         if(!ok) goto err;
1374                         }
1375                 }

1377         if(issuer)
1378                 {
1379                 /* Skip most tests for deltas because they have already
1380                  * been done
1381                  */
```



```
1382                 if (!crl->base_crl_number)
1383                         {
1384                         /* Check for cRLSign bit if keyUsage present */
1385                         if ((issuer->ex_flags & EXFLAG_KUSAGE) &&
1386                                 !(issuer->ex_kusage & KU_CRL_SIGN))
1387                                 {
1388                                 ctx->error = X509_V_ERR_KEYUSAGE_NO_CRL_SIGN;
1389                                 ok = ctx->verify_cb(0, ctx);
1390                                 if(!ok) goto err;
1391                                 }

1393                         if (!(ctx->current_crl_score & CRL_SCORE_SCOPE))
1394                                 {
1395                                 ctx->error = X509_V_ERR_DIFFERENT_CRL_SCOPE;
1396                                 ok = ctx->verify_cb(0, ctx);
1397                                 if(!ok) goto err;
1398                                 }

1400                         if (!(ctx->current_crl_score & CRL_SCORE_SAME_PATH))
1401                                 {
1402                                 if (check_crl_path(ctx, ctx->current_issuer) <=
1403                                         {
1404                                         ctx->error = X509_V_ERR_CRL_PATH_VALIDAT
1405                                         ok = ctx->verify_cb(0, ctx);
1406                                         if(!ok) goto err;
1407                                         }
1408                                 }

1410                         if (crl->idp_flags & IDP_INVALID)
1411                                 {
1412                                 ctx->error = X509_V_ERR_INVALID_EXTENSION;
1413                                 ok = ctx->verify_cb(0, ctx);
1414                                 if(!ok) goto err;
1415                                 }


1418                         }

1420                 if (!(ctx->current_crl_score & CRL_SCORE_TIME))
1421                         {
1422                         ok = check_crl_time(ctx, crl, 1);
1423                         if (!ok)
1424                                 goto err;
1425                         }

1427                 /* Attempt to get issuer certificate public key */
1428                 ikey = X509_get_pubkey(issuer);

1430                 if(!ikey)
1431                         {
1432                         ctx->error=X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY
1433                         ok = ctx->verify_cb(0, ctx);
1434                         if (!ok) goto err;
1435                         }
1436                 else
1437                         {
1438                         /* Verify CRL signature */
1439                         if(X509_CRL_verify(crl, ikey) <= 0)
1440                                 {
1441                                 ctx->error=X509_V_ERR_CRL_SIGNATURE_FAILURE;
1442                                 ok = ctx->verify_cb(0, ctx);
1443                                 if (!ok) goto err;
1444                                 }
1445                         }
1446                 }
```

```
1448         ok = 1;

1450         err:
1451         EVP_PKEY_free(ikey);
1452         return ok;
1453         }

1455 /* Check certificate against CRL */
1456 static int cert_crl(X509_STORE_CTX *ctx, X509_CRL *crl, X509 *x)
1457         {
1458         int ok;
1459         X509_REVOKED *rev;
1460         /* The rules changed for this... previously if a CRL contained
1461          * unhandled critical extensions it could still be used to indicate
1462          * a certificate was revoked. This has since been changed since
1463          * critical extension can change the meaning of CRL entries.
1464          */
1465         if (!(ctx->param->flags & X509_V_FLAG_IGNORE_CRITICAL)
1466                 && (crl->flags & EXFLAG_CRITICAL))
1467                 {
1468                 ctx->error = X509_V_ERR_UNHANDLED_CRITICAL_CRL_EXTENSION;
1469                 ok = ctx->verify_cb(0, ctx);
1470                 if(!ok)
1471                         return 0;
1472                 }
1473         /* Look for serial number of certificate in CRL
1474          * If found make sure reason is not removeFromCRL.
1475          */
1476         if (X509_CRL_get0_by_cert(crl, &rev, x))
1477                 {
1478                 if (rev->reason == CRL_REASON_REMOVE_FROM_CRL)
1479                         return 2;
1480                 ctx->error = X509_V_ERR_CERT_REVOKED;
1481                 ok = ctx->verify_cb(0, ctx);
1482                 if (!ok)
1483                         return 0;
1484                 }

1486         return 1;
1487         }

1489 static int check_policy(X509_STORE_CTX *ctx)
1490         {
1491         int ret;
1492         if (ctx->parent)
1493                 return 1;
1494         ret = X509_policy_check(&ctx->tree, &ctx->explicit_policy, ctx->chain,
1495                                 ctx->param->policies, ctx->param->flags);
1496         if (ret == 0)
1497                 {
1498                 X509err(X509_F_CHECK_POLICY,ERR_R_MALLOC_FAILURE);
1499                 return 0;
1500                 }
1501         /* Invalid or inconsistent extensions */
1502         if (ret == -1)
1503                 {
1504                 /* Locate certificates with bad extensions and notify
1505                  * callback.
1506                  */
1507                 X509 *x;
1508                 int i;
1509                 for (i = 1; i < sk_X509_num(ctx->chain); i++)
1510                         {
1511                         x = sk_X509_value(ctx->chain, i);
1512                         if (!(x->ex_flags & EXFLAG_INVALID_POLICY))
1513                                 continue;
```



```
1514                         ctx->current_cert = x;
1515                         ctx->error = X509_V_ERR_INVALID_POLICY_EXTENSION;
1516                         if(!ctx->verify_cb(0, ctx))
1517                                 return 0;
1518                         }
1519                 return 1;
1520                 }
1521         if (ret == -2)
1522                 {
1523                 ctx->current_cert = NULL;
1524                 ctx->error = X509_V_ERR_NO_EXPLICIT_POLICY;
1525                 return ctx->verify_cb(0, ctx);
1526                 }

1528         if (ctx->param->flags & X509_V_FLAG_NOTIFY_POLICY)
1529                 {
1530                 ctx->current_cert = NULL;
1531                 ctx->error = X509_V_OK;
1532                 if (!ctx->verify_cb(2, ctx))
1533                         return 0;
1534                 }

1536         return 1;
1537         }

1539 static int check_cert_time(X509_STORE_CTX *ctx, X509 *x)
1540         {
1541         time_t *ptime;
1542         int i;

1544         if (ctx->param->flags & X509_V_FLAG_USE_CHECK_TIME)
1545                 ptime = &ctx->param->check_time;
1546         else
1547                 ptime = NULL;

1549         i=X509_cmp_time(X509_get_notBefore(x), ptime);
1550         if (i == 0)
1551                 {
1552                 ctx->error=X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD;
1553                 ctx->current_cert=x;
1554                 if (!ctx->verify_cb(0, ctx))
1555                         return 0;
1556                 }

1558         if (i > 0)
1559                 {
1560                 ctx->error=X509_V_ERR_CERT_NOT_YET_VALID;
1561                 ctx->current_cert=x;
1562                 if (!ctx->verify_cb(0, ctx))
1563                         return 0;
1564                 }

1566         i=X509_cmp_time(X509_get_notAfter(x), ptime);
1567         if (i == 0)
1568                 {
1569                 ctx->error=X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD;
1570                 ctx->current_cert=x;
1571                 if (!ctx->verify_cb(0, ctx))
1572                         return 0;
1573                 }

1575         if (i < 0)
1576                 {
1577                 ctx->error=X509_V_ERR_CERT_HAS_EXPIRED;
1578                 ctx->current_cert=x;
1579                 if (!ctx->verify_cb(0, ctx))
```

```
1580                            return 0;
1581                    }

1583            return 1;
1584            }

1586    static int internal_verify(X509_STORE_CTX *ctx)
1587            {
1588            int ok=0,n;
1589            X509 *xs,*xi;
1590            EVP_PKEY *pkey=NULL;
1591            int (*cb)(int xok,X509_STORE_CTX *xctx);

1593            cb=ctx->verify_cb;

1595            n=sk_X509_num(ctx->chain);
1596            ctx->error_depth=n-1;
1597            n--;
1598            xi=sk_X509_value(ctx->chain,n);

1600            if (ctx->check_issued(ctx, xi, xi))
1601                    xs=xi;
1602            else
1603                    {
1604                    if (n <= 0)
1605                            {
1606                            ctx->error=X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE;
1607                            ctx->current_cert=xi;
1608                            ok=cb(0,ctx);
1609                            goto end;
1610                            }
1611                    else
1612                            {
1613                            n--;
1614                            ctx->error_depth=n;
1615                            xs=sk_X509_value(ctx->chain,n);
1616                            }
1617                    }

1619    /*      ctx->error=0;  not needed */
1620            while (n >= 0)
1621                    {
1622                    ctx->error_depth=n;

1624                    /* Skip signature check for self signed certificates unless
1625                     * explicitly asked for. It doesn't add any security and
1626                     * just wastes time.
1627                     */
1628                    if (!xs->valid && (xs != xi || (ctx->param->flags & X509_V_FLAG_
1629                            {
1630                            if ((pkey=X509_get_pubkey(xi)) == NULL)
1631                                    {
1632                                    ctx->error=X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PU
1633                                    ctx->current_cert=xi;
1634                                    ok=(*cb)(0,ctx);
1635                                    if (!ok) goto end;
1636                                    }
1637                            else if (X509_verify(xs,pkey) <= 0)
1638                                    {
1639                                    ctx->error=X509_V_ERR_CERT_SIGNATURE_FAILURE;
1640                                    ctx->current_cert=xs;
1641                                    ok=(*cb)(0,ctx);
1642                                    if (!ok)
1643                                            {
1644                                            EVP_PKEY_free(pkey);
1645                                            goto end;
```



```
1646                                            }
1647                                    }
1648                            EVP_PKEY_free(pkey);
1649                            pkey=NULL;
1650                            }

1652                    xs->valid = 1;

1654                    ok = check_cert_time(ctx, xs);
1655                    if (!ok)
1656                            goto end;

1658                    /* The last error (if any) is still in the error value */
1659                    ctx->current_issuer=xi;
1660                    ctx->current_cert=xs;
1661                    ok=(*cb)(1,ctx);
1662                    if (!ok) goto end;

1664                    n--;
1665                    if (n >= 0)
1666                            {
1667                            xi=xs;
1668                            xs=sk_X509_value(ctx->chain,n);
1669                            }
1670                    }
1671            ok=1;
1672    end:
1673            return ok;
1674            }

1676    int X509_cmp_current_time(const ASN1_TIME *ctm)
1677    {
1678            return X509_cmp_time(ctm, NULL);
1679    }

1681    int X509_cmp_time(const ASN1_TIME *ctm, time_t *cmp_time)
1682            {
1683            char *str;
1684            ASN1_TIME atm;
1685            long offset;
1686            char buff1[24],buff2[24],*p;
1687            int i,j;

1689            p=buff1;
1690            i=ctm->length;
1691            str=(char *)ctm->data;
1692            if (ctm->type == V_ASN1_UTCTIME)
1693                    {
1694                    if ((i < 11) || (i > 17)) return 0;
1695                    memcpy(p,str,10);
1696                    p+=10;
1697                    str+=10;
1698                    }
1699            else
1700                    {
1701                    if (i < 13) return 0;
1702                    memcpy(p,str,12);
1703                    p+=12;
1704                    str+=12;
1705                    }

1707            if ((*str == 'Z') || (*str == '-') || (*str == '+'))
1708                    { *(p++)='0'; *(p++)='0'; }
1709            else
1710                    {
1711                    *(p++)= *(str++);
```

```
1712                 *(p++)= *(str++);
1713                 /* Skip any fractional seconds... */
1714                 if (*str == '.')
1715                         {
1716                         str++;
1717                         while ((*str >= '0') && (*str <= '9')) str++;
1718                         }

1720                 }
1721         *(p++)='Z';
1722         *(p++)='\0';

1724         if (*str == 'Z')
1725                 offset=0;
1726         else
1727                 {
1728                 if ((*str != '+') && (*str != '-'))
1729                         return 0;
1730                 offset=((str[1]-'0')*10+(str[2]-'0'))*60;
1731                 offset+=(str[3]-'0')*10+(str[4]-'0');
1732                 if (*str == '-')
1733                         offset= -offset;
1734                 }
1735         atm.type=ctm->type;
1736         atm.flags = 0;
1737         atm.length=sizeof(buff2);
1738         atm.data=(unsigned char *)buff2;

1740         if (X509_time_adj(&atm, offset*60, cmp_time) == NULL)
1741                 return 0;

1743         if (ctm->type == V_ASN1_UTCTIME)
1744                 {
1745                 i=(buff1[0]-'0')*10+(buff1[1]-'0');
1746                 if (i < 50) i+=100; /* cf. RFC 2459 */
1747                 j=(buff2[0]-'0')*10+(buff2[1]-'0');
1748                 if (j < 50) j+=100;

1750                 if (i < j) return -1;
1751                 if (i > j) return 1;
1752                 }
1753         i=strcmp(buff1,buff2);
1754         if (i == 0) /* wait a second then return younger :-) */
1755                 return -1;
1756         else
1757                 return i;
1758         }

1760 ASN1_TIME *X509_gmtime_adj(ASN1_TIME *s, long adj)
1761 {
1762         return X509_time_adj(s, adj, NULL);
1763 }

1765 ASN1_TIME *X509_time_adj(ASN1_TIME *s, long offset_sec, time_t *in_tm)
1766         {
1767         return X509_time_adj_ex(s, 0, offset_sec, in_tm);
1768         }

1770 ASN1_TIME *X509_time_adj_ex(ASN1_TIME *s,
1771                             int offset_day, long offset_sec, time_t *in_tm)
1772         {
1773         time_t t;

1775         if (in_tm) t = *in_tm;
1776         else time(&t);
```



```
1778         if (s && !(s->flags & ASN1_STRING_FLAG_MSTRING))
1779                 {
1780                 if (s->type == V_ASN1_UTCTIME)
1781                         return ASN1_UTCTIME_adj(s,t, offset_day, offset_sec);
1782                 if (s->type == V_ASN1_GENERALIZEDTIME)
1783                         return ASN1_GENERALIZEDTIME_adj(s, t, offset_day,
1784                                                         offset_sec);
1785                 }
1786         return ASN1_TIME_adj(s, t, offset_day, offset_sec);
1787         }

1789 int X509_get_pubkey_parameters(EVP_PKEY *pkey, STACK_OF(X509) *chain)
1790         {
1791         EVP_PKEY *ktmp=NULL,*ktmp2;
1792         int i,j;

1794         if ((pkey != NULL) && !EVP_PKEY_missing_parameters(pkey)) return 1;

1796         for (i=0; i<sk_X509_num(chain); i++)
1797                 {
1798                 ktmp=X509_get_pubkey(sk_X509_value(chain,i));
1799                 if (ktmp == NULL)
1800                         {
1801                         X509err(X509_F_X509_GET_PUBKEY_PARAMETERS,X509_R_UNABLE_
1802                         return 0;
1803                         }
1804                 if (!EVP_PKEY_missing_parameters(ktmp))
1805                         break;
1806                 else
1807                         {
1808                         EVP_PKEY_free(ktmp);
1809                         ktmp=NULL;
1810                         }
1811                 }
1812         if (ktmp == NULL)
1813                 {
1814                 X509err(X509_F_X509_GET_PUBKEY_PARAMETERS,X509_R_UNABLE_TO_FIND_
1815                 return 0;
1816                 }

1818         /* first, populate the other certs */
1819         for (j=i-1; j >= 0; j--)
1820                 {
1821                 ktmp2=X509_get_pubkey(sk_X509_value(chain,j));
1822                 EVP_PKEY_copy_parameters(ktmp2,ktmp);
1823                 EVP_PKEY_free(ktmp2);
1824                 }

1826         if (pkey != NULL) EVP_PKEY_copy_parameters(pkey,ktmp);
1827         EVP_PKEY_free(ktmp);
1828         return 1;
1829         }

1831 int X509_STORE_CTX_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_fu
1832              CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
1833         {
1834         /* This function is (usually) called only once, by
1835          * SSL_get_ex_data_X509_STORE_CTX_idx (ssl/ssl_cert.c). */
1836         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_X509_STORE_CTX, argl, arg
1837                         new_func, dup_func, free_func);
1838         }

1840 int X509_STORE_CTX_set_ex_data(X509_STORE_CTX *ctx, int idx, void *data)
1841         {
1842         return CRYPTO_set_ex_data(&ctx->ex_data,idx,data);
1843         }
```

```
1845 void *X509_STORE_CTX_get_ex_data(X509_STORE_CTX *ctx, int idx)
1846         {
1847         return CRYPTO_get_ex_data(&ctx->ex_data,idx);
1848         }

1850 int X509_STORE_CTX_get_error(X509_STORE_CTX *ctx)
1851         {
1852         return ctx->error;
1853         }

1855 void X509_STORE_CTX_set_error(X509_STORE_CTX *ctx, int err)
1856         {
1857         ctx->error=err;
1858         }

1860 int X509_STORE_CTX_get_error_depth(X509_STORE_CTX *ctx)
1861         {
1862         return ctx->error_depth;
1863         }

1865 X509 *X509_STORE_CTX_get_current_cert(X509_STORE_CTX *ctx)
1866         {
1867         return ctx->current_cert;
1868         }

1870 STACK_OF(X509) *X509_STORE_CTX_get_chain(X509_STORE_CTX *ctx)
1871         {
1872         return ctx->chain;
1873         }

1875 STACK_OF(X509) *X509_STORE_CTX_get1_chain(X509_STORE_CTX *ctx)
1876         {
1877         int i;
1878         X509 *x;
1879         STACK_OF(X509) *chain;
1880         if (!ctx->chain || !(chain = sk_X509_dup(ctx->chain))) return NULL;
1881         for (i = 0; i < sk_X509_num(chain); i++)
1882                 {
1883                 x = sk_X509_value(chain, i);
1884                 CRYPTO_add(&x->references, 1, CRYPTO_LOCK_X509);
1885                 }
1886         return chain;
1887         }

1889 X509 *X509_STORE_CTX_get0_current_issuer(X509_STORE_CTX *ctx)
1890         {
1891         return ctx->current_issuer;
1892         }

1894 X509_CRL *X509_STORE_CTX_get0_current_crl(X509_STORE_CTX *ctx)
1895         {
1896         return ctx->current_crl;
1897         }

1899 X509_STORE_CTX *X509_STORE_CTX_get0_parent_ctx(X509_STORE_CTX *ctx)
1900         {
1901         return ctx->parent;
1902         }

1904 void X509_STORE_CTX_set_cert(X509_STORE_CTX *ctx, X509 *x)
1905         {
1906         ctx->cert=x;
1907         }

1909 void X509_STORE_CTX_set_chain(X509_STORE_CTX *ctx, STACK_OF(X509) *sk)
```



```
1910         {
1911         ctx->untrusted=sk;
1912         }

1914 void X509_STORE_CTX_set0_crls(X509_STORE_CTX *ctx, STACK_OF(X509_CRL) *sk)
1915         {
1916         ctx->crls=sk;
1917         }

1919 int X509_STORE_CTX_set_purpose(X509_STORE_CTX *ctx, int purpose)
1920         {
1921         return X509_STORE_CTX_purpose_inherit(ctx, 0, purpose, 0);
1922         }

1924 int X509_STORE_CTX_set_trust(X509_STORE_CTX *ctx, int trust)
1925         {
1926         return X509_STORE_CTX_purpose_inherit(ctx, 0, 0, trust);
1927         }

1929 /* This function is used to set the X509_STORE_CTX purpose and trust
1930  * values. This is intended to be used when another structure has its
1931  * own trust and purpose values which (if set) will be inherited by
1932  * the ctx. If they aren't set then we will usually have a default
1933  * purpose in mind which should then be used to set the trust value.
1934  * An example of this is SSL use: an SSL structure will have its own
1935  * purpose and trust settings which the application can set: if they
1936  * aren't set then we use the default of SSL client/server.
1937  */

1939 int X509_STORE_CTX_purpose_inherit(X509_STORE_CTX *ctx, int def_purpose,
1940                                    int purpose, int trust)
1941 {
1942         int idx;
1943         /* If purpose not set use default */
1944         if (!purpose) purpose = def_purpose;
1945         /* If we have a purpose then check it is valid */
1946         if (purpose)
1947                 {
1948                 X509_PURPOSE *ptmp;
1949                 idx = X509_PURPOSE_get_by_id(purpose);
1950                 if (idx == -1)
1951                         {
1952                         X509err(X509_F_X509_STORE_CTX_PURPOSE_INHERIT,
1953                                                 X509_R_UNKNOWN_PURPOSE_ID);
1954                         return 0;
1955                         }
1956                 ptmp = X509_PURPOSE_get0(idx);
1957                 if (ptmp->trust == X509_TRUST_DEFAULT)
1958                         {
1959                         idx = X509_PURPOSE_get_by_id(def_purpose);
1960                         if (idx == -1)
1961                                 {
1962                                 X509err(X509_F_X509_STORE_CTX_PURPOSE_INHERIT,
1963                                                         X509_R_UNKNOWN_PURPOSE_ID);
1964                                 return 0;
1965                                 }
1966                         ptmp = X509_PURPOSE_get0(idx);
1967                         }
1968                 /* If trust not set then get from purpose default */
1969                 if (!trust) trust = ptmp->trust;
1970                 }
1971         if (trust)
1972                 {
1973                 idx = X509_TRUST_get_by_id(trust);
1974                 if (idx == -1)
1975                         {
```

```
1976                        X509err(X509_F_X509_STORE_CTX_PURPOSE_INHERIT,
1977                                                X509_R_UNKNOWN_TRUST_ID);
1978                        return 0;
1979                        }
1980                }

1982        if (purpose && !ctx->param->purpose) ctx->param->purpose = purpose;
1983        if (trust && !ctx->param->trust) ctx->param->trust = trust;
1984        return 1;
1985 }

1987 X509_STORE_CTX *X509_STORE_CTX_new(void)
1988 {
1989        X509_STORE_CTX *ctx;
1990        ctx = (X509_STORE_CTX *)OPENSSL_malloc(sizeof(X509_STORE_CTX));
1991        if (!ctx)
1992                {
1993                X509err(X509_F_X509_STORE_CTX_NEW,ERR_R_MALLOC_FAILURE);
1994                return NULL;
1995                }
1996        memset(ctx, 0, sizeof(X509_STORE_CTX));
1997        return ctx;
1998 }

2000 void X509_STORE_CTX_free(X509_STORE_CTX *ctx)
2001 {
2002        X509_STORE_CTX_cleanup(ctx);
2003        OPENSSL_free(ctx);
2004 }

2006 int X509_STORE_CTX_init(X509_STORE_CTX *ctx, X509_STORE *store, X509 *x509,
2007             STACK_OF(X509) *chain)
2008        {
2009        int ret = 1;
2010        ctx->ctx=store;
2011        ctx->current_method=0;
2012        ctx->cert=x509;
2013        ctx->untrusted=chain;
2014        ctx->crls = NULL;
2015        ctx->last_untrusted=0;
2016        ctx->other_ctx=NULL;
2017        ctx->valid=0;
2018        ctx->chain=NULL;
2019        ctx->error=0;
2020        ctx->explicit_policy=0;
2021        ctx->error_depth=0;
2022        ctx->current_cert=NULL;
2023        ctx->current_issuer=NULL;
2024        ctx->current_crl=NULL;
2025        ctx->current_crl_score=0;
2026        ctx->current_reasons=0;
2027        ctx->tree = NULL;
2028        ctx->parent = NULL;

2030        ctx->param = X509_VERIFY_PARAM_new();

2032        if (!ctx->param)
2033                {
2034                X509err(X509_F_X509_STORE_CTX_INIT,ERR_R_MALLOC_FAILURE);
2035                return 0;
2036                }

2038        /* Inherit callbacks and flags from X509_STORE if not set
2039         * use defaults.
2040         */
```



```
2043        if (store)
2044                ret = X509_VERIFY_PARAM_inherit(ctx->param, store->param);
2045        else
2046                ctx->param->inh_flags |= X509_VP_FLAG_DEFAULT|X509_VP_FLAG_ONCE;

2048        if (store)
2049                {
2050                ctx->verify_cb = store->verify_cb;
2051                ctx->cleanup = store->cleanup;
2052                }
2053        else
2054                ctx->cleanup = 0;

2056        if (ret)
2057                ret = X509_VERIFY_PARAM_inherit(ctx->param,
2058                                        X509_VERIFY_PARAM_lookup("default"));

2060        if (ret == 0)
2061                {
2062                X509err(X509_F_X509_STORE_CTX_INIT,ERR_R_MALLOC_FAILURE);
2063                return 0;
2064                }

2066        if (store && store->check_issued)
2067                ctx->check_issued = store->check_issued;
2068        else
2069                ctx->check_issued = check_issued;

2071        if (store && store->get_issuer)
2072                ctx->get_issuer = store->get_issuer;
2073        else
2074                ctx->get_issuer = X509_STORE_CTX_get1_issuer;

2076        if (store && store->verify_cb)
2077                ctx->verify_cb = store->verify_cb;
2078        else
2079                ctx->verify_cb = null_callback;

2081        if (store && store->verify)
2082                ctx->verify = store->verify;
2083        else
2084                ctx->verify = internal_verify;

2086        if (store && store->check_revocation)
2087                ctx->check_revocation = store->check_revocation;
2088        else
2089                ctx->check_revocation = check_revocation;

2091        if (store && store->get_crl)
2092                ctx->get_crl = store->get_crl;
2093        else
2094                ctx->get_crl = NULL;

2096        if (store && store->check_crl)
2097                ctx->check_crl = store->check_crl;
2098        else
2099                ctx->check_crl = check_crl;

2101        if (store && store->cert_crl)
2102                ctx->cert_crl = store->cert_crl;
2103        else
2104                ctx->cert_crl = cert_crl;

2106        if (store && store->lookup_certs)
2107                ctx->lookup_certs = store->lookup_certs;
```

```
2108         else
2109                 ctx->lookup_certs = X509_STORE_get1_certs;

2111         if (store && store->lookup_crls)
2112                 ctx->lookup_crls = store->lookup_crls;
2113         else
2114                 ctx->lookup_crls = X509_STORE_get1_crls;

2116         ctx->check_policy = check_policy;


2119         /* This memset() can't make any sense anyway, so it's removed. As
2120          * X509_STORE_CTX_cleanup does a proper "free" on the ex_data, we put a
2121          * corresponding "new" here and remove this bogus initialisation. */
2122         /* memset(&(ctx->ex_data),0,sizeof(CRYPTO_EX_DATA)); */
2123         if(!CRYPTO_new_ex_data(CRYPTO_EX_INDEX_X509_STORE_CTX, ctx,
2124                                &(ctx->ex_data)))
2125                 {
2126                 OPENSSL_free(ctx);
2127                 X509err(X509_F_X509_STORE_CTX_INIT,ERR_R_MALLOC_FAILURE);
2128                 return 0;
2129                 }
2130         return 1;
2131         }

2133 /* Set alternative lookup method: just a STACK of trusted certificates.
2134  * This avoids X509_STORE nastiness where it isn't needed.
2135  */

2137 void X509_STORE_CTX_trusted_stack(X509_STORE_CTX *ctx, STACK_OF(X509) *sk)
2138 {
2139         ctx->other_ctx = sk;
2140         ctx->get_issuer = get_issuer_sk;
2141 }

2143 void X509_STORE_CTX_cleanup(X509_STORE_CTX *ctx)
2144         {
2145         if (ctx->cleanup) ctx->cleanup(ctx);
2146         if (ctx->param != NULL)
2147                 {
2148                 if (ctx->parent == NULL)
2149                         X509_VERIFY_PARAM_free(ctx->param);
2150                 ctx->param=NULL;
2151                 }
2152         if (ctx->tree != NULL)
2153                 {
2154                 X509_policy_tree_free(ctx->tree);
2155                 ctx->tree=NULL;
2156                 }
2157         if (ctx->chain != NULL)
2158                 {
2159                 sk_X509_pop_free(ctx->chain,X509_free);
2160                 ctx->chain=NULL;
2161                 }
2162         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_X509_STORE_CTX, ctx, &(ctx->ex_data)
2163         memset(&ctx->ex_data,0,sizeof(CRYPTO_EX_DATA));
2164         }

2166 void X509_STORE_CTX_set_depth(X509_STORE_CTX *ctx, int depth)
2167         {
2168         X509_VERIFY_PARAM_set_depth(ctx->param, depth);
2169         }

2171 void X509_STORE_CTX_set_flags(X509_STORE_CTX *ctx, unsigned long flags)
2172         {
2173         X509_VERIFY_PARAM_set_flags(ctx->param, flags);
```



```
2174         }

2176 void X509_STORE_CTX_set_time(X509_STORE_CTX *ctx, unsigned long flags, time_t t)
2177         {
2178         X509_VERIFY_PARAM_set_time(ctx->param, t);
2179         }

2181 void X509_STORE_CTX_set_verify_cb(X509_STORE_CTX *ctx,
2182                                   int (*verify_cb)(int, X509_STORE_CTX *))
2183         {
2184         ctx->verify_cb=verify_cb;
2185         }

2187 X509_POLICY_TREE *X509_STORE_CTX_get0_policy_tree(X509_STORE_CTX *ctx)
2188         {
2189         return ctx->tree;
2190         }

2192 int X509_STORE_CTX_get_explicit_policy(X509_STORE_CTX *ctx)
2193         {
2194         return ctx->explicit_policy;
2195         }

2197 int X509_STORE_CTX_set_default(X509_STORE_CTX *ctx, const char *name)
2198         {
2199         const X509_VERIFY_PARAM *param;
2200         param = X509_VERIFY_PARAM_lookup(name);
2201         if (!param)
2202                 return 0;
2203         return X509_VERIFY_PARAM_inherit(ctx->param, param);
2204         }

2206 X509_VERIFY_PARAM *X509_STORE_CTX_get0_param(X509_STORE_CTX *ctx)
2207         {
2208         return ctx->param;
2209         }

2211 void X509_STORE_CTX_set0_param(X509_STORE_CTX *ctx, X509_VERIFY_PARAM *param)
2212         {
2213         if (ctx->param)
2214                 X509_VERIFY_PARAM_free(ctx->param);
2215         ctx->param = param;
2216         }

2218 IMPLEMENT_STACK_OF(X509)
2219 IMPLEMENT_ASN1_SET_OF(X509)

2221 IMPLEMENT_STACK_OF(X509_NAME)

2223 IMPLEMENT_STACK_OF(X509_ATTRIBUTE)
2224 IMPLEMENT_ASN1_SET_OF(X509_ATTRIBUTE)
2225 #endif /* ! codereview */
```

```
**********************************************************
   11997 Wed Aug 13 19:53:26 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509_vpm.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x509_vpm.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>

#include "cryptlib.h"
```



```
#include <openssl/crypto.h>
#include <openssl/lhash.h>
#include <openssl/buffer.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

/* X509_VERIFY_PARAM functions */

static void x509_verify_param_zero(X509_VERIFY_PARAM *param)
        {
        if (!param)
                return;
        param->name = NULL;
        param->purpose = 0;
        param->trust = 0;
        /*param->inh_flags = X509_VP_FLAG_DEFAULT;*/
        param->inh_flags = 0;
        param->flags = 0;
        param->depth = -1;
        if (param->policies)
                {
                sk_ASN1_OBJECT_pop_free(param->policies, ASN1_OBJECT_free);
                param->policies = NULL;
                }
        }

X509_VERIFY_PARAM *X509_VERIFY_PARAM_new(void)
        {
        X509_VERIFY_PARAM *param;
        param = OPENSSL_malloc(sizeof(X509_VERIFY_PARAM));
        memset(param, 0, sizeof(X509_VERIFY_PARAM));
        x509_verify_param_zero(param);
        return param;
        }

void X509_VERIFY_PARAM_free(X509_VERIFY_PARAM *param)
        {
        x509_verify_param_zero(param);
        OPENSSL_free(param);
        }

/* This function determines how parameters are "inherited" from one structure
 * to another. There are several different ways this can happen.
 *
 * 1. If a child structure needs to have its values initialized from a parent
 *    they are simply copied across. For example SSL_CTX copied to SSL.
 * 2. If the structure should take on values only if they are currently unset.
 *    For example the values in an SSL structure will take appropriate value
 *    for SSL servers or clients but only if the application has not set new
 *    ones.
 *
 * The "inh_flags" field determines how this function behaves.
 *
 * Normally any values which are set in the default are not copied from the
 * destination and verify flags are ORed together.
 *
 * If X509_VP_FLAG_DEFAULT is set then anything set in the source is copied
 * to the destination. Effectively the values in "to" become default values
 * which will be used only if nothing new is set in "from".
 *
 * If X509_VP_FLAG_OVERWRITE is set then all value are copied across whether
 * they are set or not. Flags is still Ored though.
 *
 * If X509_VP_FLAG_RESET_FLAGS is set then the flags value is copied instead
 * of ORed.
 *
```

```
 * If X509_VP_FLAG_LOCKED is set then no values are copied.
 *
 * If X509_VP_FLAG_ONCE is set then the current inh_flags setting is zeroed
 * after the next call.
 */

/* Macro to test if a field should be copied from src to dest */

#define test_x509_verify_param_copy(field, def) \
        (to_overwrite || \
                ((src->field != def) && (to_default || (dest->field == def))))

/* Macro to test and copy a field if necessary */

#define x509_verify_param_copy(field, def) \
        if (test_x509_verify_param_copy(field, def)) \
                dest->field = src->field


int X509_VERIFY_PARAM_inherit(X509_VERIFY_PARAM *dest,
                                        const X509_VERIFY_PARAM *src)
        {
        unsigned long inh_flags;
        int to_default, to_overwrite;
        if (!src)
                return 1;
        inh_flags = dest->inh_flags | src->inh_flags;

        if (inh_flags & X509_VP_FLAG_ONCE)
                dest->inh_flags = 0;

        if (inh_flags & X509_VP_FLAG_LOCKED)
                return 1;

        if (inh_flags & X509_VP_FLAG_DEFAULT)
                to_default = 1;
        else
                to_default = 0;

        if (inh_flags & X509_VP_FLAG_OVERWRITE)
                to_overwrite = 1;
        else
                to_overwrite = 0;

        x509_verify_param_copy(purpose, 0);
        x509_verify_param_copy(trust, 0);
        x509_verify_param_copy(depth, -1);

        /* If overwrite or check time not set, copy across */

        if (to_overwrite || !(dest->flags & X509_V_FLAG_USE_CHECK_TIME))
                {
                dest->check_time = src->check_time;
                dest->flags &= ~X509_V_FLAG_USE_CHECK_TIME;
                /* Don't need to copy flag: that is done below */
                }

        if (inh_flags & X509_VP_FLAG_RESET_FLAGS)
                dest->flags = 0;

        dest->flags |= src->flags;

        if (test_x509_verify_param_copy(policies, NULL))
                {
                if (!X509_VERIFY_PARAM_set1_policies(dest, src->policies))
                        return 0;
```



```
                }

        return 1;
        }

int X509_VERIFY_PARAM_set1(X509_VERIFY_PARAM *to,
                                        const X509_VERIFY_PARAM *from)
        {
        unsigned long save_flags = to->inh_flags;
        int ret;
        to->inh_flags |= X509_VP_FLAG_DEFAULT;
        ret = X509_VERIFY_PARAM_inherit(to, from);
        to->inh_flags = save_flags;
        return ret;
        }

int X509_VERIFY_PARAM_set1_name(X509_VERIFY_PARAM *param, const char *name)
        {
        if (param->name)
                OPENSSL_free(param->name);
        param->name = BUF_strdup(name);
        if (param->name)
                return 1;
        return 0;
        }

int X509_VERIFY_PARAM_set_flags(X509_VERIFY_PARAM *param, unsigned long flags)
        {
        param->flags |= flags;
        if (flags & X509_V_FLAG_POLICY_MASK)
                param->flags |= X509_V_FLAG_POLICY_CHECK;
        return 1;
        }

int X509_VERIFY_PARAM_clear_flags(X509_VERIFY_PARAM *param, unsigned long flags)
        {
        param->flags &= ~flags;
        return 1;
        }

unsigned long X509_VERIFY_PARAM_get_flags(X509_VERIFY_PARAM *param)
        {
        return param->flags;
        }

int X509_VERIFY_PARAM_set_purpose(X509_VERIFY_PARAM *param, int purpose)
        {
        return X509_PURPOSE_set(&param->purpose, purpose);
        }

int X509_VERIFY_PARAM_set_trust(X509_VERIFY_PARAM *param, int trust)
        {
        return X509_TRUST_set(&param->trust, trust);
        }

void X509_VERIFY_PARAM_set_depth(X509_VERIFY_PARAM *param, int depth)
        {
        param->depth = depth;
        }

void X509_VERIFY_PARAM_set_time(X509_VERIFY_PARAM *param, time_t t)
        {
        param->check_time = t;
        param->flags |= X509_V_FLAG_USE_CHECK_TIME;
        }
```

```
260 int X509_VERIFY_PARAM_add0_policy(X509_VERIFY_PARAM *param, ASN1_OBJECT *policy)
261         {
262         if (!param->policies)
263                 {
264                 param->policies = sk_ASN1_OBJECT_new_null();
265                 if (!param->policies)
266                         return 0;
267                 }
268         if (!sk_ASN1_OBJECT_push(param->policies, policy))
269                 return 0;
270         return 1;
271         }

273 int X509_VERIFY_PARAM_set1_policies(X509_VERIFY_PARAM *param,
274                                     STACK_OF(ASN1_OBJECT) *policies)
275         {
276         int i;
277         ASN1_OBJECT *oid, *doid;
278         if (!param)
279                 return 0;
280         if (param->policies)
281                 sk_ASN1_OBJECT_pop_free(param->policies, ASN1_OBJECT_free);

283         if (!policies)
284                 {
285                 param->policies = NULL;
286                 return 1;
287                 }

289         param->policies = sk_ASN1_OBJECT_new_null();
290         if (!param->policies)
291                 return 0;

293         for (i = 0; i < sk_ASN1_OBJECT_num(policies); i++)
294                 {
295                 oid = sk_ASN1_OBJECT_value(policies, i);
296                 doid = OBJ_dup(oid);
297                 if (!doid)
298                         return 0;
299                 if (!sk_ASN1_OBJECT_push(param->policies, doid))
300                         {
301                         ASN1_OBJECT_free(doid);
302                         return 0;
303                         }
304                 }
305         param->flags |= X509_V_FLAG_POLICY_CHECK;
306         return 1;
307         }

309 int X509_VERIFY_PARAM_get_depth(const X509_VERIFY_PARAM *param)
310         {
311         return param->depth;
312         }

314 /* Default verify parameters: these are used for various
315  * applications and can be overridden by the user specified table.
316  * NB: the 'name' field *must* be in alphabetical order because it
317  * will be searched using OBJ_search.
318  */

320 static const X509_VERIFY_PARAM default_table[] = {
321         {
322         "default",      /* X509 default parameters */
323         0,              /* Check time */
324         0,              /* internal flags */
325         0,              /* flags */
```



```
326         0,              /* purpose */
327         0,              /* trust */
328         100,            /* depth */
329         NULL            /* policies */
330         },
331         {
332         "pkcs7",                        /* S/MIME sign parameters */
333         0,                              /* Check time */
334         0,                              /* internal flags */
335         0,                              /* flags */
336         X509_PURPOSE_SMIME_SIGN,        /* purpose */
337         X509_TRUST_EMAIL,               /* trust */
338         -1,                             /* depth */
339         NULL                            /* policies */
340         },
341         {
342         "smime_sign",                   /* S/MIME sign parameters */
343         0,                              /* Check time */
344         0,                              /* internal flags */
345         0,                              /* flags */
346         X509_PURPOSE_SMIME_SIGN,        /* purpose */
347         X509_TRUST_EMAIL,               /* trust */
348         -1,                             /* depth */
349         NULL                            /* policies */
350         },
351         {
352         "ssl_client",                   /* SSL/TLS client parameters */
353         0,                              /* Check time */
354         0,                              /* internal flags */
355         0,                              /* flags */
356         X509_PURPOSE_SSL_CLIENT,        /* purpose */
357         X509_TRUST_SSL_CLIENT,          /* trust */
358         -1,                             /* depth */
359         NULL                            /* policies */
360         },
361         {
362         "ssl_server",                   /* SSL/TLS server parameters */
363         0,                              /* Check time */
364         0,                              /* internal flags */
365         0,                              /* flags */
366         X509_PURPOSE_SSL_SERVER,        /* purpose */
367         X509_TRUST_SSL_SERVER,          /* trust */
368         -1,                             /* depth */
369         NULL                            /* policies */
370         }};

372 static STACK_OF(X509_VERIFY_PARAM) *param_table = NULL;

374 static int table_cmp(const X509_VERIFY_PARAM *a, const X509_VERIFY_PARAM *b)

376         {
377         return strcmp(a->name, b->name);
378         }

380 DECLARE_OBJ_BSEARCH_CMP_FN(X509_VERIFY_PARAM, X509_VERIFY_PARAM,
381                            table);
382 IMPLEMENT_OBJ_BSEARCH_CMP_FN(X509_VERIFY_PARAM, X509_VERIFY_PARAM,
383                              table);

385 static int param_cmp(const X509_VERIFY_PARAM * const *a,
386                      const X509_VERIFY_PARAM * const *b)
387         {
388         return strcmp((*a)->name, (*b)->name);
389         }

391 int X509_VERIFY_PARAM_add0_table(X509_VERIFY_PARAM *param)
```

```
392         {
393         int idx;
394         X509_VERIFY_PARAM *ptmp;
395         if (!param_table)
396                 {
397                 param_table = sk_X509_VERIFY_PARAM_new(param_cmp);
398                 if (!param_table)
399                         return 0;
400                 }
401         else
402                 {
403                 idx = sk_X509_VERIFY_PARAM_find(param_table, param);
404                 if (idx != -1)
405                         {
406                         ptmp = sk_X509_VERIFY_PARAM_value(param_table, idx);
407                         X509_VERIFY_PARAM_free(ptmp);
408                         (void)sk_X509_VERIFY_PARAM_delete(param_table, idx);
409                         }
410                 }
411         if (!sk_X509_VERIFY_PARAM_push(param_table, param))
412                 return 0;
413         return 1;
414         }

416 const X509_VERIFY_PARAM *X509_VERIFY_PARAM_lookup(const char *name)
417         {
418         int idx;
419         X509_VERIFY_PARAM pm;

421         pm.name = (char *)name;
422         if (param_table)
423                 {
424                 idx = sk_X509_VERIFY_PARAM_find(param_table, &pm);
425                 if (idx != -1)
426                         return sk_X509_VERIFY_PARAM_value(param_table, idx);
427                 }
428         return OBJ_bsearch_table(&pm, default_table,
429                                  sizeof(default_table)/sizeof(X509_VERIFY_PARAM));
430         }

432 void X509_VERIFY_PARAM_table_cleanup(void)
433         {
434         if (param_table)
435                 sk_X509_VERIFY_PARAM_pop_free(param_table,
436                                               X509_VERIFY_PARAM_free);
437         param_table = NULL;
438         }
439 #endif /* ! codereview */
```

```
**********************************************************
    4868 Wed Aug 13 19:53:26 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509cset.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509cset.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>

int X509_CRL_set_version(X509_CRL *x, long version)
        {
        if (x == NULL) return(0);
        if (x->crl->version == NULL)
                {
                if ((x->crl->version=M_ASN1_INTEGER_new()) == NULL)
                        return(0);
                }
        return(ASN1_INTEGER_set(x->crl->version,version));
        }

int X509_CRL_set_issuer_name(X509_CRL *x, X509_NAME *name)
        {
        if ((x == NULL) || (x->crl == NULL)) return(0);
        return(X509_NAME_set(&x->crl->issuer,name));
        }


int X509_CRL_set_lastUpdate(X509_CRL *x, const ASN1_TIME *tm)
        {
        ASN1_TIME *in;

        if (x == NULL) return(0);
        in=x->crl->lastUpdate;
        if (in != tm)
                {
                in=M_ASN1_TIME_dup(tm);
                if (in != NULL)
                        {
                        M_ASN1_TIME_free(x->crl->lastUpdate);
                        x->crl->lastUpdate=in;
                        }
                }
        return(in != NULL);
        }

int X509_CRL_set_nextUpdate(X509_CRL *x, const ASN1_TIME *tm)
        {
        ASN1_TIME *in;

        if (x == NULL) return(0);
        in=x->crl->nextUpdate;
        if (in != tm)
                {
                in=M_ASN1_TIME_dup(tm);
                if (in != NULL)
                        {
                        M_ASN1_TIME_free(x->crl->nextUpdate);
                        x->crl->nextUpdate=in;
                        }
                }
        return(in != NULL);
        }

int X509_CRL_sort(X509_CRL *c)
        {
        int i;
        X509_REVOKED *r;
        /* sort the data so it will be written in serial
         * number order */
        sk_X509_REVOKED_sort(c->crl->revoked);
        for (i=0; i<sk_X509_REVOKED_num(c->crl->revoked); i++)
```

```
128                 {
129                 r=sk_X509_REVOKED_value(c->crl->revoked,i);
130                 r->sequence=i;
131                 }
132         c->crl->enc.modified = 1;
133         return 1;
134         }

136 int X509_REVOKED_set_revocationDate(X509_REVOKED *x, ASN1_TIME *tm)
137         {
138         ASN1_TIME *in;

140         if (x == NULL) return(0);
141         in=x->revocationDate;
142         if (in != tm)
143                 {
144                 in=M_ASN1_TIME_dup(tm);
145                 if (in != NULL)
146                         {
147                         M_ASN1_TIME_free(x->revocationDate);
148                         x->revocationDate=in;
149                         }
150                 }
151         return(in != NULL);
152         }

154 int X509_REVOKED_set_serialNumber(X509_REVOKED *x, ASN1_INTEGER *serial)
155         {
156         ASN1_INTEGER *in;

158         if (x == NULL) return(0);
159         in=x->serialNumber;
160         if (in != serial)
161                 {
162                 in=M_ASN1_INTEGER_dup(serial);
163                 if (in != NULL)
164                         {
165                         M_ASN1_INTEGER_free(x->serialNumber);
166                         x->serialNumber=in;
167                         }
168                 }
169         return(in != NULL);
170         }
171 #endif /* ! codereview */
```

```
**********************************************************
   10795 Wed Aug 13 19:53:26 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509name.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509name.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/stack.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>

int X509_NAME_get_text_by_NID(X509_NAME *name, int nid, char *buf, int len)
        {
        ASN1_OBJECT *obj;

        obj=OBJ_nid2obj(nid);
        if (obj == NULL) return(-1);
        return(X509_NAME_get_text_by_OBJ(name,obj,buf,len));
        }

int X509_NAME_get_text_by_OBJ(X509_NAME *name, ASN1_OBJECT *obj, char *buf,
             int len)
        {
        int i;
        ASN1_STRING *data;

        i=X509_NAME_get_index_by_OBJ(name,obj,-1);
        if (i < 0) return(-1);
        data=X509_NAME_ENTRY_get_data(X509_NAME_get_entry(name,i));
        i=(data->length > (len-1))?(len-1):data->length;
        if (buf == NULL) return(data->length);
        memcpy(buf,data->data,i);
        buf[i]='\0';
        return(i);
        }

int X509_NAME_entry_count(X509_NAME *name)
        {
        if (name == NULL) return(0);
        return(sk_X509_NAME_ENTRY_num(name->entries));
        }

int X509_NAME_get_index_by_NID(X509_NAME *name, int nid, int lastpos)
        {
        ASN1_OBJECT *obj;

        obj=OBJ_nid2obj(nid);
        if (obj == NULL) return(-2);
        return(X509_NAME_get_index_by_OBJ(name,obj,lastpos));
        }

/* NOTE: you should be passsing -1, not 0 as lastpos */
int X509_NAME_get_index_by_OBJ(X509_NAME *name, ASN1_OBJECT *obj,
             int lastpos)
        {
        int n;
        X509_NAME_ENTRY *ne;
        STACK_OF(X509_NAME_ENTRY) *sk;

        if (name == NULL) return(-1);
        if (lastpos < 0)
                lastpos= -1;
        sk=name->entries;
        n=sk_X509_NAME_ENTRY_num(sk);
        for (lastpos++; lastpos < n; lastpos++)
                {
                ne=sk_X509_NAME_ENTRY_value(sk,lastpos);
                if (OBJ_cmp(ne->object,obj) == 0)
                        return(lastpos);
                }
        return(-1);
        }
```

```
X509_NAME_ENTRY *X509_NAME_get_entry(X509_NAME *name, int loc)
        {
        if(name == NULL || sk_X509_NAME_ENTRY_num(name->entries) <= loc
           || loc < 0)
                return(NULL);
        else
                return(sk_X509_NAME_ENTRY_value(name->entries,loc));
        }

X509_NAME_ENTRY *X509_NAME_delete_entry(X509_NAME *name, int loc)
        {
        X509_NAME_ENTRY *ret;
        int i,n,set_prev,set_next;
        STACK_OF(X509_NAME_ENTRY) *sk;

        if (name == NULL || sk_X509_NAME_ENTRY_num(name->entries) <= loc
            || loc < 0)
                return(NULL);
        sk=name->entries;
        ret=sk_X509_NAME_ENTRY_delete(sk,loc);
        n=sk_X509_NAME_ENTRY_num(sk);
        name->modified=1;
        if (loc == n) return(ret);

        /* else we need to fixup the set field */
        if (loc != 0)
                set_prev=(sk_X509_NAME_ENTRY_value(sk,loc-1))->set;
        else
                set_prev=ret->set-1;
        set_next=sk_X509_NAME_ENTRY_value(sk,loc)->set;

        /* set_prev is the previous set
         * set is the current set
         * set_next is the following
         * prev  1 1    1 1     1 1     1 1
         * set   1      1       2       2
         * next  1 1    2 2     2 2     3 2
         * so basically only if prev and next differ by 2, then
         * re-number down by 1 */
        if (set_prev+1 < set_next)
                for (i=loc; i<n; i++)
                        sk_X509_NAME_ENTRY_value(sk,i)->set--;
        return(ret);
        }

int X509_NAME_add_entry_by_OBJ(X509_NAME *name, ASN1_OBJECT *obj, int type,
                        unsigned char *bytes, int len, int loc, int set)
{
        X509_NAME_ENTRY *ne;
        int ret;
        ne = X509_NAME_ENTRY_create_by_OBJ(NULL, obj, type, bytes, len);
        if(!ne) return 0;
        ret = X509_NAME_add_entry(name, ne, loc, set);
        X509_NAME_ENTRY_free(ne);
        return ret;
}

int X509_NAME_add_entry_by_NID(X509_NAME *name, int nid, int type,
                        unsigned char *bytes, int len, int loc, int set)
{
        X509_NAME_ENTRY *ne;
        int ret;
        ne = X509_NAME_ENTRY_create_by_NID(NULL, nid, type, bytes, len);
        if(!ne) return 0;
        ret = X509_NAME_add_entry(name, ne, loc, set);
```



```
        X509_NAME_ENTRY_free(ne);
        return ret;
}

int X509_NAME_add_entry_by_txt(X509_NAME *name, const char *field, int type,
                        const unsigned char *bytes, int len, int loc, int set)
{
        X509_NAME_ENTRY *ne;
        int ret;
        ne = X509_NAME_ENTRY_create_by_txt(NULL, field, type, bytes, len);
        if(!ne) return 0;
        ret = X509_NAME_add_entry(name, ne, loc, set);
        X509_NAME_ENTRY_free(ne);
        return ret;
}

/* if set is -1, append to previous set, 0 'a new one', and 1,
 * prepend to the guy we are about to stomp on. */
int X509_NAME_add_entry(X509_NAME *name, X509_NAME_ENTRY *ne, int loc,
             int set)
        {
        X509_NAME_ENTRY *new_name=NULL;
        int n,i,inc;
        STACK_OF(X509_NAME_ENTRY) *sk;

        if (name == NULL) return(0);
        sk=name->entries;
        n=sk_X509_NAME_ENTRY_num(sk);
        if (loc > n) loc=n;
        else if (loc < 0) loc=n;

        name->modified=1;

        if (set == -1)
                {
                if (loc == 0)
                        {
                        set=0;
                        inc=1;
                        }
                else
                        {
                        set=sk_X509_NAME_ENTRY_value(sk,loc-1)->set;
                        inc=0;
                        }
                }
        else /* if (set >= 0) */
                {
                if (loc >= n)
                        {
                        if (loc != 0)
                                set=sk_X509_NAME_ENTRY_value(sk,loc-1)->set+1;
                        else
                                set=0;
                        }
                else
                        set=sk_X509_NAME_ENTRY_value(sk,loc)->set;
                inc=(set == 0)?1:0;
                }

        if ((new_name=X509_NAME_ENTRY_dup(ne)) == NULL)
                goto err;
        new_name->set=set;
        if (!sk_X509_NAME_ENTRY_insert(sk,new_name,loc))
                {
                X509err(X509_F_X509_NAME_ADD_ENTRY,ERR_R_MALLOC_FAILURE);
```

```
260                 goto err;
261                 }
262         if (inc)
263                 {
264                 n=sk_X509_NAME_ENTRY_num(sk);
265                 for (i=loc+1; i<n; i++)
266                         sk_X509_NAME_ENTRY_value(sk,i-1)->set+=1;
267                 }
268         return(1);
269 err:
270         if (new_name != NULL)
271                 X509_NAME_ENTRY_free(new_name);
272         return(0);
273         }

275 X509_NAME_ENTRY *X509_NAME_ENTRY_create_by_txt(X509_NAME_ENTRY **ne,
276                 const char *field, int type, const unsigned char *bytes, int len
277         {
278         ASN1_OBJECT *obj;
279         X509_NAME_ENTRY *nentry;

281         obj=OBJ_txt2obj(field, 0);
282         if (obj == NULL)
283                 {
284                 X509err(X509_F_X509_NAME_ENTRY_CREATE_BY_TXT,
285                                                 X509_R_INVALID_FIELD_NAME);
286                 ERR_add_error_data(2, "name=", field);
287                 return(NULL);
288                 }
289         nentry = X509_NAME_ENTRY_create_by_OBJ(ne,obj,type,bytes,len);
290         ASN1_OBJECT_free(obj);
291         return nentry;
292         }

294 X509_NAME_ENTRY *X509_NAME_ENTRY_create_by_NID(X509_NAME_ENTRY **ne, int nid,
295              int type, unsigned char *bytes, int len)
296         {
297         ASN1_OBJECT *obj;
298         X509_NAME_ENTRY *nentry;

300         obj=OBJ_nid2obj(nid);
301         if (obj == NULL)
302                 {
303                 X509err(X509_F_X509_NAME_ENTRY_CREATE_BY_NID,X509_R_UNKNOWN_NID)
304                 return(NULL);
305                 }
306         nentry = X509_NAME_ENTRY_create_by_OBJ(ne,obj,type,bytes,len);
307         ASN1_OBJECT_free(obj);
308         return nentry;
309         }

311 X509_NAME_ENTRY *X509_NAME_ENTRY_create_by_OBJ(X509_NAME_ENTRY **ne,
312              ASN1_OBJECT *obj, int type, const unsigned char *bytes, int len)
313         {
314         X509_NAME_ENTRY *ret;

316         if ((ne == NULL) || (*ne == NULL))
317                 {
318                 if ((ret=X509_NAME_ENTRY_new()) == NULL)
319                         return(NULL);
320                 }
321         else
322                 ret= *ne;

324         if (!X509_NAME_ENTRY_set_object(ret,obj))
325                 goto err;
```



```
326         if (!X509_NAME_ENTRY_set_data(ret,type,bytes,len))
327                 goto err;

329         if ((ne != NULL) && (*ne == NULL)) *ne=ret;
330         return(ret);
331 err:
332         if ((ne == NULL) || (ret != *ne))
333                 X509_NAME_ENTRY_free(ret);
334         return(NULL);
335         }

337 int X509_NAME_ENTRY_set_object(X509_NAME_ENTRY *ne, ASN1_OBJECT *obj)
338         {
339         if ((ne == NULL) || (obj == NULL))
340                 {
341                 X509err(X509_F_X509_NAME_ENTRY_SET_OBJECT,ERR_R_PASSED_NULL_PARA
342                 return(0);
343                 }
344         ASN1_OBJECT_free(ne->object);
345         ne->object=OBJ_dup(obj);
346         return((ne->object == NULL)?0:1);
347         }

349 int X509_NAME_ENTRY_set_data(X509_NAME_ENTRY *ne, int type,
350              const unsigned char *bytes, int len)
351         {
352         int i;

354         if ((ne == NULL) || ((bytes == NULL) && (len != 0))) return(0);
355         if((type > 0) && (type & MBSTRING_FLAG))
356                 return ASN1_STRING_set_by_NID(&ne->value, bytes,
357                                                 len, type,
358                                         OBJ_obj2nid(ne->object)) ? 1 : 0;
359         if (len < 0) len=strlen((const char *)bytes);
360         i=ASN1_STRING_set(ne->value,bytes,len);
361         if (!i) return(0);
362         if (type != V_ASN1_UNDEF)
363                 {
364                 if (type == V_ASN1_APP_CHOOSE)
365                         ne->value->type=ASN1_PRINTABLE_type(bytes,len);
366                 else
367                         ne->value->type=type;
368                 }
369         return(1);
370         }

372 ASN1_OBJECT *X509_NAME_ENTRY_get_object(X509_NAME_ENTRY *ne)
373         {
374         if (ne == NULL) return(NULL);
375         return(ne->object);
376         }

378 ASN1_STRING *X509_NAME_ENTRY_get_data(X509_NAME_ENTRY *ne)
379         {
380         if (ne == NULL) return(NULL);
381         return(ne->value);
382         }
383 #endif /* ! codereview */
```

```
**********************************************************
    3823 Wed Aug 13 19:53:26 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509rset.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x509rset.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>

int X509_REQ_set_version(X509_REQ *x, long version)
        {
        if (x == NULL) return(0);
        return(ASN1_INTEGER_set(x->req_info->version,version));
        }

int X509_REQ_set_subject_name(X509_REQ *x, X509_NAME *name)
        {
        if ((x == NULL) || (x->req_info == NULL)) return(0);
        return(X509_NAME_set(&x->req_info->subject,name));
        }

int X509_REQ_set_pubkey(X509_REQ *x, EVP_PKEY *pkey)
        {
        if ((x == NULL) || (x->req_info == NULL)) return(0);
        return(X509_PUBKEY_set(&x->req_info->pubkey,pkey));
        }
#endif /* ! codereview */
```

```
**********************************************************
    4378 Wed Aug 13 19:53:26 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509spki.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* x509spki.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509.h>
```



```
int NETSCAPE_SPKI_set_pubkey(NETSCAPE_SPKI *x, EVP_PKEY *pkey)
{
        if ((x == NULL) || (x->spkac == NULL)) return(0);
        return(X509_PUBKEY_set(&(x->spkac->pubkey),pkey));
}

EVP_PKEY *NETSCAPE_SPKI_get_pubkey(NETSCAPE_SPKI *x)
{
        if ((x == NULL) || (x->spkac == NULL))
                return(NULL);
        return(X509_PUBKEY_get(x->spkac->pubkey));
}

/* Load a Netscape SPKI from a base64 encoded string */

NETSCAPE_SPKI * NETSCAPE_SPKI_b64_decode(const char *str, int len)
{
        unsigned char *spki_der;
        const unsigned char *p;
        int spki_len;
        NETSCAPE_SPKI *spki;
        if(len <= 0) len = strlen(str);
        if (!(spki_der = OPENSSL_malloc(len + 1))) {
                X509err(X509_F_NETSCAPE_SPKI_B64_DECODE, ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        spki_len = EVP_DecodeBlock(spki_der, (const unsigned char *)str, len);
        if(spki_len < 0) {
                X509err(X509_F_NETSCAPE_SPKI_B64_DECODE,
                                                X509_R_BASE64_DECODE_ERROR);
                OPENSSL_free(spki_der);
                return NULL;
        }
        p = spki_der;
        spki = d2i_NETSCAPE_SPKI(NULL, &p, spki_len);
        OPENSSL_free(spki_der);
        return spki;
}

/* Generate a base64 encoded string from an SPKI */

char * NETSCAPE_SPKI_b64_encode(NETSCAPE_SPKI *spki)
{
        unsigned char *der_spki, *p;
        char *b64_str;
        int der_len;
        der_len = i2d_NETSCAPE_SPKI(spki, NULL);
        der_spki = OPENSSL_malloc(der_len);
        b64_str = OPENSSL_malloc(der_len * 2);
        if(!der_spki || !b64_str) {
                X509err(X509_F_NETSCAPE_SPKI_B64_ENCODE, ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        p = der_spki;
        i2d_NETSCAPE_SPKI(spki, &p);
        EVP_EncodeBlock((unsigned char *)b64_str, der_spki, der_len);
        OPENSSL_free(der_spki);
        return b64_str;
}
#endif /* ! codereview */
```

****
4471 Wed Aug 13 19:53:27 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x509type.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* crypto/x509/x509type.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/evp.h>
```



```
#include <openssl/objects.h>
#include <openssl/x509.h>

int X509_certificate_type(X509 *x, EVP_PKEY *pkey)
        {
        EVP_PKEY *pk;
        int ret=0,i;

        if (x == NULL) return(0);

        if (pkey == NULL)
                pk=X509_get_pubkey(x);
        else
                pk=pkey;

        if (pk == NULL) return(0);

        switch (pk->type)
                {
        case EVP_PKEY_RSA:
                ret=EVP_PK_RSA|EVP_PKT_SIGN;
/*              if (!sign only extension) */
                        ret|=EVP_PKT_ENC;
        break;
        case EVP_PKEY_DSA:
                ret=EVP_PK_DSA|EVP_PKT_SIGN;
                break;
        case EVP_PKEY_EC:
                ret=EVP_PK_EC|EVP_PKT_SIGN|EVP_PKT_EXCH;
                break;
        case EVP_PKEY_DH:
                ret=EVP_PK_DH|EVP_PKT_EXCH;
                break;
        case NID_id_GostR3410_94:
        case NID_id_GostR3410_2001:
                ret=EVP_PKT_EXCH|EVP_PKT_SIGN;
                break;
        default:
                break;
                }

        i=OBJ_obj2nid(x->sig_alg->algorithm);
        if (i && OBJ_find_sigid_algs(i, NULL, &i))
                {

                switch (i)
                        {
                case NID_rsaEncryption:
                case NID_rsa:
                        ret|=EVP_PKS_RSA;
                        break;
                case NID_dsa:
                case NID_dsa_2:
                        ret|=EVP_PKS_DSA;
                        break;
                case NID_X9_62_id_ecPublicKey:
                        ret|=EVP_PKS_EC;
                        break;
                default:
                        break;
                        }
                }

        if (EVP_PKEY_size(pk) <= 1024/8)/* /8 because it's 1024 bits we look
                                           for, not bytes */
                ret|=EVP_PKT_EXP;
```

```
128         if(pkey==NULL) EVP_PKEY_free(pk);
129         return(ret);
130         }
131 #endif /* ! codereview */
```

```
**********************************************************
   14626 Wed Aug 13 19:53:27 2014
new/usr/src/lib/openssl/libsunw_crypto/x509/x_all.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509/x_all.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/stack.h>
#include "cryptlib.h"
```



```
#include <openssl/buffer.h>
#include <openssl/asn1.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#ifndef OPENSSL_NO_RSA
#include <openssl/rsa.h>
#endif
#ifndef OPENSSL_NO_DSA
#include <openssl/dsa.h>
#endif

int X509_verify(X509 *a, EVP_PKEY *r)
        {
        return(ASN1_item_verify(ASN1_ITEM_rptr(X509_CINF),a->sig_alg,
                a->signature,a->cert_info,r));
        }

int X509_REQ_verify(X509_REQ *a, EVP_PKEY *r)
        {
        return( ASN1_item_verify(ASN1_ITEM_rptr(X509_REQ_INFO),
                a->sig_alg,a->signature,a->req_info,r));
        }

int NETSCAPE_SPKI_verify(NETSCAPE_SPKI *a, EVP_PKEY *r)
        {
        return(ASN1_item_verify(ASN1_ITEM_rptr(NETSCAPE_SPKAC),
                a->sig_algor,a->signature,a->spkac,r));
        }

int X509_sign(X509 *x, EVP_PKEY *pkey, const EVP_MD *md)
        {
        x->cert_info->enc.modified = 1;
        return(ASN1_item_sign(ASN1_ITEM_rptr(X509_CINF), x->cert_info->signature
                x->sig_alg, x->signature, x->cert_info,pkey,md));
        }

int X509_sign_ctx(X509 *x, EVP_MD_CTX *ctx)
        {
        x->cert_info->enc.modified = 1;
        return ASN1_item_sign_ctx(ASN1_ITEM_rptr(X509_CINF),
                x->cert_info->signature,
                x->sig_alg, x->signature, x->cert_info, ctx);
        }

int X509_REQ_sign(X509_REQ *x, EVP_PKEY *pkey, const EVP_MD *md)
        {
        return(ASN1_item_sign(ASN1_ITEM_rptr(X509_REQ_INFO),x->sig_alg, NULL,
                x->signature, x->req_info,pkey,md));
        }

int X509_REQ_sign_ctx(X509_REQ *x, EVP_MD_CTX *ctx)
        {
        return ASN1_item_sign_ctx(ASN1_ITEM_rptr(X509_REQ_INFO),
                x->sig_alg, NULL, x->signature, x->req_info, ctx);
        }

int X509_CRL_sign(X509_CRL *x, EVP_PKEY *pkey, const EVP_MD *md)
        {
        x->crl->enc.modified = 1;
        return(ASN1_item_sign(ASN1_ITEM_rptr(X509_CRL_INFO),x->crl->sig_alg,
                x->sig_alg, x->signature, x->crl,pkey,md));
        }

int X509_CRL_sign_ctx(X509_CRL *x, EVP_MD_CTX *ctx)
        {
        x->crl->enc.modified = 1;
```

```
128         return ASN1_item_sign_ctx(ASN1_ITEM_rptr(X509_CRL_INFO),
129                 x->crl->sig_alg, x->sig_alg, x->signature, x->crl, ctx);
130         }

132 int NETSCAPE_SPKI_sign(NETSCAPE_SPKI *x, EVP_PKEY *pkey, const EVP_MD *md)
133         {
134         return(ASN1_item_sign(ASN1_ITEM_rptr(NETSCAPE_SPKAC), x->sig_algor,NULL,
135                 x->signature, x->spkac,pkey,md));
136         }

138 #ifndef OPENSSL_NO_FP_API
139 X509 *d2i_X509_fp(FILE *fp, X509 **x509)
140         {
141         return ASN1_item_d2i_fp(ASN1_ITEM_rptr(X509), fp, x509);
142         }

144 int i2d_X509_fp(FILE *fp, X509 *x509)
145         {
146         return ASN1_item_i2d_fp(ASN1_ITEM_rptr(X509), fp, x509);
147         }
148 #endif

150 X509 *d2i_X509_bio(BIO *bp, X509 **x509)
151         {
152         return ASN1_item_d2i_bio(ASN1_ITEM_rptr(X509), bp, x509);
153         }

155 int i2d_X509_bio(BIO *bp, X509 *x509)
156         {
157         return ASN1_item_i2d_bio(ASN1_ITEM_rptr(X509), bp, x509);
158         }

160 #ifndef OPENSSL_NO_FP_API
161 X509_CRL *d2i_X509_CRL_fp(FILE *fp, X509_CRL **crl)
162         {
163         return ASN1_item_d2i_fp(ASN1_ITEM_rptr(X509_CRL), fp, crl);
164         }

166 int i2d_X509_CRL_fp(FILE *fp, X509_CRL *crl)
167         {
168         return ASN1_item_i2d_fp(ASN1_ITEM_rptr(X509_CRL), fp, crl);
169         }
170 #endif

172 X509_CRL *d2i_X509_CRL_bio(BIO *bp, X509_CRL **crl)
173         {
174         return ASN1_item_d2i_bio(ASN1_ITEM_rptr(X509_CRL), bp, crl);
175         }

177 int i2d_X509_CRL_bio(BIO *bp, X509_CRL *crl)
178         {
179         return ASN1_item_i2d_bio(ASN1_ITEM_rptr(X509_CRL), bp, crl);
180         }

182 #ifndef OPENSSL_NO_FP_API
183 PKCS7 *d2i_PKCS7_fp(FILE *fp, PKCS7 **p7)
184         {
185         return ASN1_item_d2i_fp(ASN1_ITEM_rptr(PKCS7), fp, p7);
186         }

188 int i2d_PKCS7_fp(FILE *fp, PKCS7 *p7)
189         {
190         return ASN1_item_i2d_fp(ASN1_ITEM_rptr(PKCS7), fp, p7);
191         }
192 #endif
```



```
194 PKCS7 *d2i_PKCS7_bio(BIO *bp, PKCS7 **p7)
195         {
196         return ASN1_item_d2i_bio(ASN1_ITEM_rptr(PKCS7), bp, p7);
197         }

199 int i2d_PKCS7_bio(BIO *bp, PKCS7 *p7)
200         {
201         return ASN1_item_i2d_bio(ASN1_ITEM_rptr(PKCS7), bp, p7);
202         }

204 #ifndef OPENSSL_NO_FP_API
205 X509_REQ *d2i_X509_REQ_fp(FILE *fp, X509_REQ **req)
206         {
207         return ASN1_item_d2i_fp(ASN1_ITEM_rptr(X509_REQ), fp, req);
208         }

210 int i2d_X509_REQ_fp(FILE *fp, X509_REQ *req)
211         {
212         return ASN1_item_i2d_fp(ASN1_ITEM_rptr(X509_REQ), fp, req);
213         }
214 #endif

216 X509_REQ *d2i_X509_REQ_bio(BIO *bp, X509_REQ **req)
217         {
218         return ASN1_item_d2i_bio(ASN1_ITEM_rptr(X509_REQ), bp, req);
219         }

221 int i2d_X509_REQ_bio(BIO *bp, X509_REQ *req)
222         {
223         return ASN1_item_i2d_bio(ASN1_ITEM_rptr(X509_REQ), bp, req);
224         }

226 #ifndef OPENSSL_NO_RSA

228 #ifndef OPENSSL_NO_FP_API
229 RSA *d2i_RSAPrivateKey_fp(FILE *fp, RSA **rsa)
230         {
231         return ASN1_item_d2i_fp(ASN1_ITEM_rptr(RSAPrivateKey), fp, rsa);
232         }

234 int i2d_RSAPrivateKey_fp(FILE *fp, RSA *rsa)
235         {
236         return ASN1_item_i2d_fp(ASN1_ITEM_rptr(RSAPrivateKey), fp, rsa);
237         }

239 RSA *d2i_RSAPublicKey_fp(FILE *fp, RSA **rsa)
240         {
241         return ASN1_item_d2i_fp(ASN1_ITEM_rptr(RSAPublicKey), fp, rsa);
242         }


245 RSA *d2i_RSA_PUBKEY_fp(FILE *fp, RSA **rsa)
246         {
247         return ASN1_d2i_fp((void *(*)(void))
248                            RSA_new,(D2I_OF(void))d2i_RSA_PUBKEY, fp,
249                            (void **)rsa);
250         }

252 int i2d_RSAPublicKey_fp(FILE *fp, RSA *rsa)
253         {
254         return ASN1_item_i2d_fp(ASN1_ITEM_rptr(RSAPublicKey), fp, rsa);
255         }

257 int i2d_RSA_PUBKEY_fp(FILE *fp, RSA *rsa)
258         {
259         return ASN1_i2d_fp((I2D_OF(void))i2d_RSA_PUBKEY,fp,rsa);
```

```
260         }
261 #endif

263 RSA *d2i_RSAPrivateKey_bio(BIO *bp, RSA **rsa)
264         {
265         return ASN1_item_d2i_bio(ASN1_ITEM_rptr(RSAPrivateKey), bp, rsa);
266         }

268 int i2d_RSAPrivateKey_bio(BIO *bp, RSA *rsa)
269         {
270         return ASN1_item_i2d_bio(ASN1_ITEM_rptr(RSAPrivateKey), bp, rsa);
271         }

273 RSA *d2i_RSAPublicKey_bio(BIO *bp, RSA **rsa)
274         {
275         return ASN1_item_d2i_bio(ASN1_ITEM_rptr(RSAPublicKey), bp, rsa);
276         }


279 RSA *d2i_RSA_PUBKEY_bio(BIO *bp, RSA **rsa)
280         {
281         return ASN1_d2i_bio_of(RSA,RSA_new,d2i_RSA_PUBKEY,bp,rsa);
282         }

284 int i2d_RSAPublicKey_bio(BIO *bp, RSA *rsa)
285         {
286         return ASN1_item_i2d_bio(ASN1_ITEM_rptr(RSAPublicKey), bp, rsa);
287         }

289 int i2d_RSA_PUBKEY_bio(BIO *bp, RSA *rsa)
290         {
291         return ASN1_i2d_bio_of(RSA,i2d_RSA_PUBKEY,bp,rsa);
292         }
293 #endif

295 #ifndef OPENSSL_NO_DSA
296 #ifndef OPENSSL_NO_FP_API
297 DSA *d2i_DSAPrivateKey_fp(FILE *fp, DSA **dsa)
298         {
299         return ASN1_d2i_fp_of(DSA,DSA_new,d2i_DSAPrivateKey,fp,dsa);
300         }

302 int i2d_DSAPrivateKey_fp(FILE *fp, DSA *dsa)
303         {
304         return ASN1_i2d_fp_of_const(DSA,i2d_DSAPrivateKey,fp,dsa);
305         }

307 DSA *d2i_DSA_PUBKEY_fp(FILE *fp, DSA **dsa)
308         {
309         return ASN1_d2i_fp_of(DSA,DSA_new,d2i_DSA_PUBKEY,fp,dsa);
310         }

312 int i2d_DSA_PUBKEY_fp(FILE *fp, DSA *dsa)
313         {
314         return ASN1_i2d_fp_of(DSA,i2d_DSA_PUBKEY,fp,dsa);
315         }
316 #endif

318 DSA *d2i_DSAPrivateKey_bio(BIO *bp, DSA **dsa)
319         {
320         return ASN1_d2i_bio_of(DSA,DSA_new,d2i_DSAPrivateKey,bp,dsa
321 );
322         }

324 int i2d_DSAPrivateKey_bio(BIO *bp, DSA *dsa)
325         {
```



```
326         return ASN1_i2d_bio_of_const(DSA,i2d_DSAPrivateKey,bp,dsa);
327         }

329 DSA *d2i_DSA_PUBKEY_bio(BIO *bp, DSA **dsa)
330         {
331         return ASN1_d2i_bio_of(DSA,DSA_new,d2i_DSA_PUBKEY,bp,dsa);
332         }

334 int i2d_DSA_PUBKEY_bio(BIO *bp, DSA *dsa)
335         {
336         return ASN1_i2d_bio_of(DSA,i2d_DSA_PUBKEY,bp,dsa);
337         }

339 #endif

341 #ifndef OPENSSL_NO_EC
342 #ifndef OPENSSL_NO_FP_API
343 EC_KEY *d2i_EC_PUBKEY_fp(FILE *fp, EC_KEY **eckey)
344         {
345         return ASN1_d2i_fp_of(EC_KEY,EC_KEY_new,d2i_EC_PUBKEY,fp,eckey);
346         }

348 int i2d_EC_PUBKEY_fp(FILE *fp, EC_KEY *eckey)
349         {
350         return ASN1_i2d_fp_of(EC_KEY,i2d_EC_PUBKEY,fp,eckey);
351         }

353 EC_KEY *d2i_ECPrivateKey_fp(FILE *fp, EC_KEY **eckey)
354         {
355         return ASN1_d2i_fp_of(EC_KEY,EC_KEY_new,d2i_ECPrivateKey,fp,eckey);
356         }

358 int i2d_ECPrivateKey_fp(FILE *fp, EC_KEY *eckey)
359         {
360         return ASN1_i2d_fp_of(EC_KEY,i2d_ECPrivateKey,fp,eckey);
361         }
362 #endif
363 EC_KEY *d2i_EC_PUBKEY_bio(BIO *bp, EC_KEY **eckey)
364         {
365         return ASN1_d2i_bio_of(EC_KEY,EC_KEY_new,d2i_EC_PUBKEY,bp,eckey);
366         }

368 int i2d_EC_PUBKEY_bio(BIO *bp, EC_KEY *ecdsa)
369         {
370         return ASN1_i2d_bio_of(EC_KEY,i2d_EC_PUBKEY,bp,ecdsa);
371         }

373 EC_KEY *d2i_ECPrivateKey_bio(BIO *bp, EC_KEY **eckey)
374         {
375         return ASN1_d2i_bio_of(EC_KEY,EC_KEY_new,d2i_ECPrivateKey,bp,eckey);
376         }

378 int i2d_ECPrivateKey_bio(BIO *bp, EC_KEY *eckey)
379         {
380         return ASN1_i2d_bio_of(EC_KEY,i2d_ECPrivateKey,bp,eckey);
381         }
382 #endif


385 int X509_pubkey_digest(const X509 *data, const EVP_MD *type, unsigned char *md,
386             unsigned int *len)
387         {
388         ASN1_BIT_STRING *key;
389         key = X509_get0_pubkey_bitstr(data);
390         if(!key) return 0;
391         return EVP_Digest(key->data, key->length, md, len, type, NULL);
```

```
392         }

394 int X509_digest(const X509 *data, const EVP_MD *type, unsigned char *md,
395             unsigned int *len)
396         {
397         return(ASN1_item_digest(ASN1_ITEM_rptr(X509),type,(char *)data,md,len));
398         }

400 int X509_CRL_digest(const X509_CRL *data, const EVP_MD *type, unsigned char *md,
401             unsigned int *len)
402         {
403         return(ASN1_item_digest(ASN1_ITEM_rptr(X509_CRL),type,(char *)data,md,le
404         }

406 int X509_REQ_digest(const X509_REQ *data, const EVP_MD *type, unsigned char *md,
407             unsigned int *len)
408         {
409         return(ASN1_item_digest(ASN1_ITEM_rptr(X509_REQ),type,(char *)data,md,le
410         }

412 int X509_NAME_digest(const X509_NAME *data, const EVP_MD *type, unsigned char *m
413             unsigned int *len)
414         {
415         return(ASN1_item_digest(ASN1_ITEM_rptr(X509_NAME),type,(char *)data,md,l
416         }

418 int PKCS7_ISSUER_AND_SERIAL_digest(PKCS7_ISSUER_AND_SERIAL *data, const EVP_MD *
419             unsigned char *md, unsigned int *len)
420         {
421         return(ASN1_item_digest(ASN1_ITEM_rptr(PKCS7_ISSUER_AND_SERIAL),type,
422                 (char *)data,md,len));
423         }


426 #ifndef OPENSSL_NO_FP_API
427 X509_SIG *d2i_PKCS8_fp(FILE *fp, X509_SIG **p8)
428         {
429         return ASN1_d2i_fp_of(X509_SIG,X509_SIG_new,d2i_X509_SIG,fp,p8);
430         }

432 int i2d_PKCS8_fp(FILE *fp, X509_SIG *p8)
433         {
434         return ASN1_i2d_fp_of(X509_SIG,i2d_X509_SIG,fp,p8);
435         }
436 #endif

438 X509_SIG *d2i_PKCS8_bio(BIO *bp, X509_SIG **p8)
439         {
440         return ASN1_d2i_bio_of(X509_SIG,X509_SIG_new,d2i_X509_SIG,bp,p8);
441         }

443 int i2d_PKCS8_bio(BIO *bp, X509_SIG *p8)
444         {
445         return ASN1_i2d_bio_of(X509_SIG,i2d_X509_SIG,bp,p8);
446         }

448 #ifndef OPENSSL_NO_FP_API
449 PKCS8_PRIV_KEY_INFO *d2i_PKCS8_PRIV_KEY_INFO_fp(FILE *fp,
450                                                 PKCS8_PRIV_KEY_INFO **p8inf)
451         {
452         return ASN1_d2i_fp_of(PKCS8_PRIV_KEY_INFO,PKCS8_PRIV_KEY_INFO_new,
453                               d2i_PKCS8_PRIV_KEY_INFO,fp,p8inf);
454         }

456 int i2d_PKCS8_PRIV_KEY_INFO_fp(FILE *fp, PKCS8_PRIV_KEY_INFO *p8inf)
457         {
```



```
458         return ASN1_i2d_fp_of(PKCS8_PRIV_KEY_INFO,i2d_PKCS8_PRIV_KEY_INFO,fp,
459                               p8inf);
460         }

462 int i2d_PKCS8PrivateKeyInfo_fp(FILE *fp, EVP_PKEY *key)
463         {
464         PKCS8_PRIV_KEY_INFO *p8inf;
465         int ret;
466         p8inf = EVP_PKEY2PKCS8(key);
467         if(!p8inf) return 0;
468         ret = i2d_PKCS8_PRIV_KEY_INFO_fp(fp, p8inf);
469         PKCS8_PRIV_KEY_INFO_free(p8inf);
470         return ret;
471         }

473 int i2d_PrivateKey_fp(FILE *fp, EVP_PKEY *pkey)
474         {
475         return ASN1_i2d_fp_of(EVP_PKEY,i2d_PrivateKey,fp,pkey);
476         }

478 EVP_PKEY *d2i_PrivateKey_fp(FILE *fp, EVP_PKEY **a)
479 {
480         return ASN1_d2i_fp_of(EVP_PKEY,EVP_PKEY_new,d2i_AutoPrivateKey,fp,a);
481 }

483 int i2d_PUBKEY_fp(FILE *fp, EVP_PKEY *pkey)
484         {
485         return ASN1_i2d_fp_of(EVP_PKEY,i2d_PUBKEY,fp,pkey);
486         }

488 EVP_PKEY *d2i_PUBKEY_fp(FILE *fp, EVP_PKEY **a)
489 {
490         return ASN1_d2i_fp_of(EVP_PKEY,EVP_PKEY_new,d2i_PUBKEY,fp,a);
491 }

493 #endif

495 PKCS8_PRIV_KEY_INFO *d2i_PKCS8_PRIV_KEY_INFO_bio(BIO *bp,
496                                                  PKCS8_PRIV_KEY_INFO **p8inf)
497         {
498         return ASN1_d2i_bio_of(PKCS8_PRIV_KEY_INFO,PKCS8_PRIV_KEY_INFO_new,
499                                d2i_PKCS8_PRIV_KEY_INFO,bp,p8inf);
500         }

502 int i2d_PKCS8_PRIV_KEY_INFO_bio(BIO *bp, PKCS8_PRIV_KEY_INFO *p8inf)
503         {
504         return ASN1_i2d_bio_of(PKCS8_PRIV_KEY_INFO,i2d_PKCS8_PRIV_KEY_INFO,bp,
505                                p8inf);
506         }

508 int i2d_PKCS8PrivateKeyInfo_bio(BIO *bp, EVP_PKEY *key)
509         {
510         PKCS8_PRIV_KEY_INFO *p8inf;
511         int ret;
512         p8inf = EVP_PKEY2PKCS8(key);
513         if(!p8inf) return 0;
514         ret = i2d_PKCS8_PRIV_KEY_INFO_bio(bp, p8inf);
515         PKCS8_PRIV_KEY_INFO_free(p8inf);
516         return ret;
517         }

519 int i2d_PrivateKey_bio(BIO *bp, EVP_PKEY *pkey)
520         {
521         return ASN1_i2d_bio_of(EVP_PKEY,i2d_PrivateKey,bp,pkey);
522         }
```

```
524 EVP_PKEY *d2i_PrivateKey_bio(BIO *bp, EVP_PKEY **a)
525         {
526         return ASN1_d2i_bio_of(EVP_PKEY,EVP_PKEY_new,d2i_AutoPrivateKey,bp,a);
527         }

529 int i2d_PUBKEY_bio(BIO *bp, EVP_PKEY *pkey)
530         {
531         return ASN1_i2d_bio_of(EVP_PKEY,i2d_PUBKEY,bp,pkey);
532         }

534 EVP_PKEY *d2i_PUBKEY_bio(BIO *bp, EVP_PKEY **a)
535         {
536         return ASN1_d2i_bio_of(EVP_PKEY,EVP_PKEY_new,d2i_PUBKEY,bp,a);
537         }
538 #endif /* ! codereview */
```

```
**********************************************************
    7723 Wed Aug 13 19:53:27 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/pcy_cache.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pcy_cache.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/x509v3.h>
```



```
#include "pcy_int.h"

static int policy_data_cmp(const X509_POLICY_DATA * const *a,
                                const X509_POLICY_DATA * const *b);
static int policy_cache_set_int(long *out, ASN1_INTEGER *value);

/* Set cache entry according to CertificatePolicies extension.
 * Note: this destroys the passed CERTIFICATEPOLICIES structure.
 */

static int policy_cache_create(X509 *x,
                        CERTIFICATEPOLICIES *policies, int crit)
        {
        int i;
        int ret = 0;
        X509_POLICY_CACHE *cache = x->policy_cache;
        X509_POLICY_DATA *data = NULL;
        POLICYINFO *policy;
        if (sk_POLICYINFO_num(policies) == 0)
                goto bad_policy;
        cache->data = sk_X509_POLICY_DATA_new(policy_data_cmp);
        if (!cache->data)
                goto bad_policy;
        for (i = 0; i < sk_POLICYINFO_num(policies); i++)
                {
                policy = sk_POLICYINFO_value(policies, i);
                data = policy_data_new(policy, NULL, crit);
                if (!data)
                        goto bad_policy;
                /* Duplicate policy OIDs are illegal: reject if matches
                 * found.
                 */
                if (OBJ_obj2nid(data->valid_policy) == NID_any_policy)
                        {
                        if (cache->anyPolicy)
                                {
                                ret = -1;
                                goto bad_policy;
                                }
                        cache->anyPolicy = data;
                        }
                else if (sk_X509_POLICY_DATA_find(cache->data, data) != -1)
                        {
                        ret = -1;
                        goto bad_policy;
                        }
                else if (!sk_X509_POLICY_DATA_push(cache->data, data))
                        goto bad_policy;
                data = NULL;
                }
        ret = 1;
        bad_policy:
        if (ret == -1)
                x->ex_flags |= EXFLAG_INVALID_POLICY;
        if (data)
                policy_data_free(data);
        sk_POLICYINFO_pop_free(policies, POLICYINFO_free);
        if (ret <= 0)
                {
                sk_X509_POLICY_DATA_pop_free(cache->data, policy_data_free);
                cache->data = NULL;
                }
        return ret;
        }
```

```
129 static int policy_cache_new(X509 *x)
130         {
131         X509_POLICY_CACHE *cache;
132         ASN1_INTEGER *ext_any = NULL;
133         POLICY_CONSTRAINTS *ext_pcons = NULL;
134         CERTIFICATEPOLICIES *ext_cpols = NULL;
135         POLICY_MAPPINGS *ext_pmaps = NULL;
136         int i;
137         cache = OPENSSL_malloc(sizeof(X509_POLICY_CACHE));
138         if (!cache)
139                 return 0;
140         cache->anyPolicy = NULL;
141         cache->data = NULL;
142         cache->any_skip = -1;
143         cache->explicit_skip = -1;
144         cache->map_skip = -1;

146         x->policy_cache = cache;

148         /* Handle requireExplicitPolicy *first*. Need to process this
149          * even if we don't have any policies.
150          */
151         ext_pcons = X509_get_ext_d2i(x, NID_policy_constraints, &i, NULL);

153         if (!ext_pcons)
154                 {
155                 if (i != -1)
156                         goto bad_cache;
157                 }
158         else
159                 {
160                 if (!ext_pcons->requireExplicitPolicy
161                         && !ext_pcons->inhibitPolicyMapping)
162                         goto bad_cache;
163                 if (!policy_cache_set_int(&cache->explicit_skip,
164                         ext_pcons->requireExplicitPolicy))
165                         goto bad_cache;
166                 if (!policy_cache_set_int(&cache->map_skip,
167                         ext_pcons->inhibitPolicyMapping))
168                         goto bad_cache;
169                 }

171         /* Process CertificatePolicies */

173         ext_cpols = X509_get_ext_d2i(x, NID_certificate_policies, &i, NULL);
174         /* If no CertificatePolicies extension or problem decoding then
175          * there is no point continuing because the valid policies will be
176          * NULL.
177          */
178         if (!ext_cpols)
179                 {
180                 /* If not absent some problem with extension */
181                 if (i != -1)
182                         goto bad_cache;
183                 return 1;
184                 }

186         i = policy_cache_create(x, ext_cpols, i);

188         /* NB: ext_cpols freed by policy_cache_set_policies */

190         if (i <= 0)
191                 return i;

193         ext_pmaps = X509_get_ext_d2i(x, NID_policy_mappings, &i, NULL);
```



```
195         if (!ext_pmaps)
196                 {
197                 /* If not absent some problem with extension */
198                 if (i != -1)
199                         goto bad_cache;
200                 }
201         else
202                 {
203                 i = policy_cache_set_mapping(x, ext_pmaps);
204                 if (i <= 0)
205                         goto bad_cache;
206                 }

208         ext_any = X509_get_ext_d2i(x, NID_inhibit_any_policy, &i, NULL);

210         if (!ext_any)
211                 {
212                 if (i != -1)
213                         goto bad_cache;
214                 }
215         else if (!policy_cache_set_int(&cache->any_skip, ext_any))
216                         goto bad_cache;

218         if (0)
219                 {
220                 bad_cache:
221                 x->ex_flags |= EXFLAG_INVALID_POLICY;
222                 }

224         if(ext_pcons)
225                 POLICY_CONSTRAINTS_free(ext_pcons);

227         if (ext_any)
228                 ASN1_INTEGER_free(ext_any);

230         return 1;


233 }

235 void policy_cache_free(X509_POLICY_CACHE *cache)
236         {
237         if (!cache)
238                 return;
239         if (cache->anyPolicy)
240                 policy_data_free(cache->anyPolicy);
241         if (cache->data)
242                 sk_X509_POLICY_DATA_pop_free(cache->data, policy_data_free);
243         OPENSSL_free(cache);
244         }

246 const X509_POLICY_CACHE *policy_cache_set(X509 *x)
247         {

249         if (x->policy_cache == NULL)
250                 {
251                 CRYPTO_w_lock(CRYPTO_LOCK_X509);
252                         policy_cache_new(x);
253                 CRYPTO_w_unlock(CRYPTO_LOCK_X509);
254                 }

256         return x->policy_cache;

258         }
```

```
X509_POLICY_DATA *policy_cache_find_data(const X509_POLICY_CACHE *cache,
                                         const ASN1_OBJECT *id)
        {
        int idx;
        X509_POLICY_DATA tmp;
        tmp.valid_policy = (ASN1_OBJECT *)id;
        idx = sk_X509_POLICY_DATA_find(cache->data, &tmp);
        if (idx == -1)
                return NULL;
        return sk_X509_POLICY_DATA_value(cache->data, idx);
        }

static int policy_data_cmp(const X509_POLICY_DATA * const *a,
                           const X509_POLICY_DATA * const *b)
        {
        return OBJ_cmp((*a)->valid_policy, (*b)->valid_policy);
        }

static int policy_cache_set_int(long *out, ASN1_INTEGER *value)
        {
        if (value == NULL)
                return 1;
        if (value->type == V_ASN1_NEG_INTEGER)
                return 0;
        *out = ASN1_INTEGER_get(value);
        return 1;
        }
#endif /* ! codereview */
```

```
**********************************************************
    4375 Wed Aug 13 19:53:27 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/pcy_data.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pcy_data.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/x509v3.h>
```



```
#include "pcy_int.h"

/* Policy Node routines */

void policy_data_free(X509_POLICY_DATA *data)
        {
        ASN1_OBJECT_free(data->valid_policy);
        /* Don't free qualifiers if shared */
        if (!(data->flags & POLICY_DATA_FLAG_SHARED_QUALIFIERS))
                sk_POLICYQUALINFO_pop_free(data->qualifier_set,
                                           POLICYQUALINFO_free);
        sk_ASN1_OBJECT_pop_free(data->expected_policy_set, ASN1_OBJECT_free);
        OPENSSL_free(data);
        }

/* Create a data based on an existing policy. If 'id' is NULL use the
 * oid in the policy, otherwise use 'id'. This behaviour covers the two
 * types of data in RFC3280: data with from a CertificatePolcies extension
 * and additional data with just the qualifiers of anyPolicy and ID from
 * another source.
 */

X509_POLICY_DATA *policy_data_new(POLICYINFO *policy,
                                  const ASN1_OBJECT *cid, int crit)
        {
        X509_POLICY_DATA *ret;
        ASN1_OBJECT *id;
        if (!policy && !cid)
                return NULL;
        if (cid)
                {
                id = OBJ_dup(cid);
                if (!id)
                        return NULL;
                }
        else
                id = NULL;
        ret = OPENSSL_malloc(sizeof(X509_POLICY_DATA));
        if (!ret)
                return NULL;
        ret->expected_policy_set = sk_ASN1_OBJECT_new_null();
        if (!ret->expected_policy_set)
                {
                OPENSSL_free(ret);
                if (id)
                        ASN1_OBJECT_free(id);
                return NULL;
                }

        if (crit)
                ret->flags = POLICY_DATA_FLAG_CRITICAL;
        else
                ret->flags = 0;

        if (id)
                ret->valid_policy = id;
        else
                {
                ret->valid_policy = policy->policyid;
                policy->policyid = NULL;
                }

        if (policy)
                {
                ret->qualifier_set = policy->qualifiers;
```

```
128                 policy->qualifiers = NULL;
129                 }
130         else
131                 ret->qualifier_set = NULL;

133         return ret;
134         }
135 #endif /* ! codereview */
```

```
**********************************************************
    4689 Wed Aug 13 19:53:27 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/pcy_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pcy_lib.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include "cryptlib.h"
#include <openssl/x509.h>
```



```
#include <openssl/x509v3.h>

#include "pcy_int.h"

/* accessor functions */

/* X509_POLICY_TREE stuff */

int X509_policy_tree_level_count(const X509_POLICY_TREE *tree)
        {
        if (!tree)
                return 0;
        return tree->nlevel;
        }

X509_POLICY_LEVEL *
        X509_policy_tree_get0_level(const X509_POLICY_TREE *tree, int i)
        {
        if (!tree || (i < 0) || (i >= tree->nlevel))
                return NULL;
        return tree->levels + i;
        }

STACK_OF(X509_POLICY_NODE) *
                X509_policy_tree_get0_policies(const X509_POLICY_TREE *tree)
        {
        if (!tree)
                return NULL;
        return tree->auth_policies;
        }

STACK_OF(X509_POLICY_NODE) *
        X509_policy_tree_get0_user_policies(const X509_POLICY_TREE *tree)
        {
        if (!tree)
                return NULL;
        if (tree->flags & POLICY_FLAG_ANY_POLICY)
                return tree->auth_policies;
        else
                return tree->user_policies;
        }

/* X509_POLICY_LEVEL stuff */

int X509_policy_level_node_count(X509_POLICY_LEVEL *level)
        {
        int n;
        if (!level)
                return 0;
        if (level->anyPolicy)
                n = 1;
        else
                n = 0;
        if (level->nodes)
                n += sk_X509_POLICY_NODE_num(level->nodes);
        return n;
        }

X509_POLICY_NODE *X509_policy_level_get0_node(X509_POLICY_LEVEL *level, int i)
        {
        if (!level)
                return NULL;
        if (level->anyPolicy)
                {
                if (i == 0)
                        return level->anyPolicy;
```

```
128                 i--;
129                 }
130         return sk_X509_POLICY_NODE_value(level->nodes, i);
131         }

133 /* X509_POLICY_NODE stuff */

135 const ASN1_OBJECT *X509_policy_node_get0_policy(const X509_POLICY_NODE *node)
136         {
137         if (!node)
138                 return NULL;
139         return node->data->valid_policy;
140         }

142 #if 0
143 int X509_policy_node_get_critical(const X509_POLICY_NODE *node)
144         {
145         if (node_critical(node))
146                 return 1;
147         return 0;
148         }
149 #endif

151 STACK_OF(POLICYQUALINFO) *
152                 X509_policy_node_get0_qualifiers(const X509_POLICY_NODE *node)
153         {
154         if (!node)
155                 return NULL;
156         return node->data->qualifier_set;
157         }

159 const X509_POLICY_NODE *
160                 X509_policy_node_get0_parent(const X509_POLICY_NODE *node)
161         {
162         if (!node)
163                 return NULL;
164         return node->parent;
165         }
166 #endif /* ! codereview */
```

```
**********************************************************
    4595 Wed Aug 13 19:53:27 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/pcy_map.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pcy_map.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/x509v3.h>
```



```
#include "pcy_int.h"

/* Set policy mapping entries in cache.
 * Note: this modifies the passed POLICY_MAPPINGS structure
 */

int policy_cache_set_mapping(X509 *x, POLICY_MAPPINGS *maps)
        {
        POLICY_MAPPING *map;
        X509_POLICY_DATA *data;
        X509_POLICY_CACHE *cache = x->policy_cache;
        int i;
        int ret = 0;
        if (sk_POLICY_MAPPING_num(maps) == 0)
                {
                ret = -1;
                goto bad_mapping;
                }
        for (i = 0; i < sk_POLICY_MAPPING_num(maps); i++)
                {
                map = sk_POLICY_MAPPING_value(maps, i);
                /* Reject if map to or from anyPolicy */
                if ((OBJ_obj2nid(map->subjectDomainPolicy) == NID_any_policy)
                   || (OBJ_obj2nid(map->issuerDomainPolicy) == NID_any_policy))
                        {
                        ret = -1;
                        goto bad_mapping;
                        }

                /* Attempt to find matching policy data */
                data = policy_cache_find_data(cache, map->issuerDomainPolicy);
                /* If we don't have anyPolicy can't map */
                if (!data && !cache->anyPolicy)
                        continue;

                /* Create a NODE from anyPolicy */
                if (!data)
                        {
                        data = policy_data_new(NULL, map->issuerDomainPolicy,
                                        cache->anyPolicy->flags
                                                & POLICY_DATA_FLAG_CRITICAL);
                        if (!data)
                                goto bad_mapping;
                        data->qualifier_set = cache->anyPolicy->qualifier_set;
                        /*map->issuerDomainPolicy = NULL;*/
                        data->flags |= POLICY_DATA_FLAG_MAPPED_ANY;
                        data->flags |= POLICY_DATA_FLAG_SHARED_QUALIFIERS;
                        if (!sk_X509_POLICY_DATA_push(cache->data, data))
                                {
                                policy_data_free(data);
                                goto bad_mapping;
                                }
                        }
                else
                        data->flags |= POLICY_DATA_FLAG_MAPPED;
                if (!sk_ASN1_OBJECT_push(data->expected_policy_set,
                                                map->subjectDomainPolicy))
                        goto bad_mapping;
                map->subjectDomainPolicy = NULL;

                }

        ret = 1;
        bad_mapping:
        if (ret == -1)
```

```
128                 x->ex_flags |= EXFLAG_INVALID_POLICY;
129         sk_POLICY_MAPPING_pop_free(maps, POLICY_MAPPING_free);
130         return ret;

132         }
133 #endif /* ! codereview */
```

```
**********************************************************
    5622 Wed Aug 13 19:53:28 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/pcy_node.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pcy_node.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <openssl/asn1.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>
```



```
#include "pcy_int.h"

static int node_cmp(const X509_POLICY_NODE * const *a,
                        const X509_POLICY_NODE * const *b)
        {
        return OBJ_cmp((*a)->data->valid_policy, (*b)->data->valid_policy);
        }

STACK_OF(X509_POLICY_NODE) *policy_node_cmp_new(void)
        {
        return sk_X509_POLICY_NODE_new(node_cmp);
        }

X509_POLICY_NODE *tree_find_sk(STACK_OF(X509_POLICY_NODE) *nodes,
                                        const ASN1_OBJECT *id)
        {
        X509_POLICY_DATA n;
        X509_POLICY_NODE l;
        int idx;

        n.valid_policy = (ASN1_OBJECT *)id;
        l.data = &n;

        idx = sk_X509_POLICY_NODE_find(nodes, &l);
        if (idx == -1)
                return NULL;

        return sk_X509_POLICY_NODE_value(nodes, idx);

        }

X509_POLICY_NODE *level_find_node(const X509_POLICY_LEVEL *level,
                                        const X509_POLICY_NODE *parent,
                                        const ASN1_OBJECT *id)
        {
        X509_POLICY_NODE *node;
        int i;
        for (i = 0; i < sk_X509_POLICY_NODE_num(level->nodes); i++)
                {
                node = sk_X509_POLICY_NODE_value(level->nodes, i);
                if (node->parent == parent)
                        {
                        if (!OBJ_cmp(node->data->valid_policy, id))
                                return node;
                        }
                }
        return NULL;
        }

X509_POLICY_NODE *level_add_node(X509_POLICY_LEVEL *level,
                        const X509_POLICY_DATA *data,
                        X509_POLICY_NODE *parent,
                        X509_POLICY_TREE *tree)
        {
        X509_POLICY_NODE *node;
        node = OPENSSL_malloc(sizeof(X509_POLICY_NODE));
        if (!node)
                return NULL;
        node->data = data;
        node->parent = parent;
        node->nchild = 0;
        if (level)
                {
                if (OBJ_obj2nid(data->valid_policy) == NID_any_policy)
                        {
```

```
128                         if (level->anyPolicy)
129                                 goto node_error;
130                         level->anyPolicy = node;
131                         }
132                 else
133                         {

135                         if (!level->nodes)
136                                 level->nodes = policy_node_cmp_new();
137                         if (!level->nodes)
138                                 goto node_error;
139                         if (!sk_X509_POLICY_NODE_push(level->nodes, node))
140                                 goto node_error;
141                         }
142                 }

144         if (tree)
145                 {
146                 if (!tree->extra_data)
147                          tree->extra_data = sk_X509_POLICY_DATA_new_null();
148                 if (!tree->extra_data)
149                         goto node_error;
150                 if (!sk_X509_POLICY_DATA_push(tree->extra_data, data))
151                         goto node_error;
152                 }

154         if (parent)
155                 parent->nchild++;

157         return node;

159         node_error:
160         policy_node_free(node);
161         return 0;

163         }

165 void policy_node_free(X509_POLICY_NODE *node)
166         {
167         OPENSSL_free(node);
168         }

170 /* See if a policy node matches a policy OID. If mapping enabled look through
171  * expected policy set otherwise just valid policy.
172  */

174 int policy_node_match(const X509_POLICY_LEVEL *lvl,
175                       const X509_POLICY_NODE *node, const ASN1_OBJECT *oid)
176         {
177         int i;
178         ASN1_OBJECT *policy_oid;
179         const X509_POLICY_DATA *x = node->data;

181         if (        (lvl->flags & X509_V_FLAG_INHIBIT_MAP)
182                 || !(x->flags & POLICY_DATA_FLAG_MAP_MASK))
183                 {
184                 if (!OBJ_cmp(x->valid_policy, oid))
185                         return 1;
186                 return 0;
187                 }

189         for (i = 0; i < sk_ASN1_OBJECT_num(x->expected_policy_set); i++)
190                 {
191                 policy_oid = sk_ASN1_OBJECT_value(x->expected_policy_set, i);
192                 if (!OBJ_cmp(policy_oid, oid))
193                         return 1;
```



```
194                 }
195         return 0;

197         }
198 #endif /* ! codereview */
```

```
**********************************************************
   21268 Wed Aug 13 19:53:28 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/pcy_tree.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* pcy_tree.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "cryptlib.h"
#include <openssl/x509.h>
#include <openssl/x509v3.h>
```



```
#include "pcy_int.h"

/* Enable this to print out the complete policy tree at various point during
 * evaluation.
 */

/*#define OPENSSL_POLICY_DEBUG*/

#ifdef OPENSSL_POLICY_DEBUG

static void expected_print(BIO *err, X509_POLICY_LEVEL *lev,
                                X509_POLICY_NODE *node, int indent)
        {
        if (        (lev->flags & X509_V_FLAG_INHIBIT_MAP)
                || !(node->data->flags & POLICY_DATA_FLAG_MAP_MASK))
                BIO_puts(err, "  Not Mapped\n");
        else
                {
                int i;
                STACK_OF(ASN1_OBJECT) *pset = node->data->expected_policy_set;
                ASN1_OBJECT *oid;
                BIO_puts(err, "  Expected: ");
                for (i = 0; i < sk_ASN1_OBJECT_num(pset); i++)
                        {
                        oid = sk_ASN1_OBJECT_value(pset, i);
                        if (i)
                                BIO_puts(err, ", ");
                        i2a_ASN1_OBJECT(err, oid);
                        }
                BIO_puts(err, "\n");
                }
        }

static void tree_print(char *str, X509_POLICY_TREE *tree,
                        X509_POLICY_LEVEL *curr)
        {
        X509_POLICY_LEVEL *plev;
        X509_POLICY_NODE *node;
        int i;
        BIO *err;
        err = BIO_new_fp(stderr, BIO_NOCLOSE);
        if (!curr)
                curr = tree->levels + tree->nlevel;
        else
                curr++;
        BIO_printf(err, "Level print after %s\n", str);
        BIO_printf(err, "Printing Up to Level %ld\n", curr - tree->levels);
        for (plev = tree->levels; plev != curr; plev++)
                {
                BIO_printf(err, "Level %ld, flags = %x\n",
                                plev - tree->levels, plev->flags);
                for (i = 0; i < sk_X509_POLICY_NODE_num(plev->nodes); i++)
                        {
                        node = sk_X509_POLICY_NODE_value(plev->nodes, i);
                        X509_POLICY_NODE_print(err, node, 2);
                        expected_print(err, plev, node, 2);
                        BIO_printf(err, "  Flags: %x\n", node->data->flags);
                        }
                if (plev->anyPolicy)
                        X509_POLICY_NODE_print(err, plev->anyPolicy, 2);
                }

        BIO_free(err);

        }
```

```
128 #else

130 #define tree_print(a,b,c) /* */

132 #endif

134 /* Initialize policy tree. Return values:
135  *  0 Some internal error occured.
136  * -1 Inconsistent or invalid extensions in certificates.
137  *  1 Tree initialized OK.
138  *  2 Policy tree is empty.
139  *  5 Tree OK and requireExplicitPolicy true.
140  *  6 Tree empty and requireExplicitPolicy true.
141  */

143 static int tree_init(X509_POLICY_TREE **ptree, STACK_OF(X509) *certs,
144                      unsigned int flags)
145         {
146         X509_POLICY_TREE *tree;
147         X509_POLICY_LEVEL *level;
148         const X509_POLICY_CACHE *cache;
149         X509_POLICY_DATA *data = NULL;
150         X509 *x;
151         int ret = 1;
152         int i, n;
153         int explicit_policy;
154         int any_skip;
155         int map_skip;
156         *ptree = NULL;
157         n = sk_X509_num(certs);

159 #if 0
160         /* Disable policy mapping for now... */
161         flags |= X509_V_FLAG_INHIBIT_MAP;
162 #endif

164         if (flags & X509_V_FLAG_EXPLICIT_POLICY)
165                 explicit_policy = 0;
166         else
167                 explicit_policy = n + 1;

169         if (flags & X509_V_FLAG_INHIBIT_ANY)
170                 any_skip = 0;
171         else
172                 any_skip = n + 1;

174         if (flags & X509_V_FLAG_INHIBIT_MAP)
175                 map_skip = 0;
176         else
177                 map_skip = n + 1;

179         /* Can't do anything with just a trust anchor */
180         if (n == 1)
181                 return 1;
182         /* First setup policy cache in all certificates apart from the
183          * trust anchor. Note any bad cache results on the way. Also can
184          * calculate explicit_policy value at this point.
185          */
186         for (i = n - 2; i >= 0; i--)
187                 {
188                 x = sk_X509_value(certs, i);
189                 X509_check_purpose(x, -1, -1);
190                 cache = policy_cache_set(x);
191                 /* If cache NULL something bad happened: return immediately */
192                 if (cache == NULL)
193                         return 0;
```



```
194                 /* If inconsistent extensions keep a note of it but continue */
195                 if (x->ex_flags & EXFLAG_INVALID_POLICY)
196                         ret = -1;
197                 /* Otherwise if we have no data (hence no CertificatePolicies)
198                  * and haven't already set an inconsistent code note it.
199                  */
200                 else if ((ret == 1) && !cache->data)
201                         ret = 2;
202                 if (explicit_policy > 0)
203                         {
204                         if (!(x->ex_flags & EXFLAG_SI))
205                                 explicit_policy--;
206                         if ((cache->explicit_skip != -1)
207                                 && (cache->explicit_skip < explicit_policy))
208                                 explicit_policy = cache->explicit_skip;
209                         }
210                 }

212         if (ret != 1)
213                 {
214                 if (ret == 2 && !explicit_policy)
215                         return 6;
216                 return ret;
217                 }


220         /* If we get this far initialize the tree */

222         tree = OPENSSL_malloc(sizeof(X509_POLICY_TREE));

224         if (!tree)
225                 return 0;

227         tree->flags = 0;
228         tree->levels = OPENSSL_malloc(sizeof(X509_POLICY_LEVEL) * n);
229         tree->nlevel = 0;
230         tree->extra_data = NULL;
231         tree->auth_policies = NULL;
232         tree->user_policies = NULL;

234         if (!tree->levels)
235                 {
236                 OPENSSL_free(tree);
237                 return 0;
238                 }

240         memset(tree->levels, 0, n * sizeof(X509_POLICY_LEVEL));

242         tree->nlevel = n;

244         level = tree->levels;

246         /* Root data: initialize to anyPolicy */

248         data = policy_data_new(NULL, OBJ_nid2obj(NID_any_policy), 0);

250         if (!data || !level_add_node(level, data, NULL, tree))
251                 goto bad_tree;

253         for (i = n - 2; i >= 0; i--)
254                 {
255                 level++;
256                 x = sk_X509_value(certs, i);
257                 cache = policy_cache_set(x);
258                 CRYPTO_add(&x->references, 1, CRYPTO_LOCK_X509);
259                 level->cert = x;
```

```
261                 if (!cache->anyPolicy)
262                         level->flags |= X509_V_FLAG_INHIBIT_ANY;

264                 /* Determine inhibit any and inhibit map flags */
265                 if (any_skip == 0)
266                         {
267                         /* Any matching allowed if certificate is self
268                          * issued and not the last in the chain.
269                          */
270                         if (!(x->ex_flags & EXFLAG_SI) || (i == 0))
271                                 level->flags |= X509_V_FLAG_INHIBIT_ANY;
272                         }
273                 else
274                         {
275                         if (!(x->ex_flags & EXFLAG_SI))
276                                 any_skip--;
277                         if ((cache->any_skip >= 0)
278                                 && (cache->any_skip < any_skip))
279                                 any_skip = cache->any_skip;
280                         }

282                 if (map_skip == 0)
283                         level->flags |= X509_V_FLAG_INHIBIT_MAP;
284                 else
285                         {
286                         if (!(x->ex_flags & EXFLAG_SI))
287                                 map_skip--;
288                         if ((cache->map_skip >= 0)
289                                 && (cache->map_skip < map_skip))
290                                 map_skip = cache->map_skip;
291                         }

293                 }

295         *ptree = tree;

297         if (explicit_policy)
298                 return 1;
299         else
300                 return 5;

302         bad_tree:

304         X509_policy_tree_free(tree);

306         return 0;

308         }

310 static int tree_link_matching_nodes(X509_POLICY_LEVEL *curr,
311                                 const X509_POLICY_DATA *data)
312         {
313         X509_POLICY_LEVEL *last = curr - 1;
314         X509_POLICY_NODE *node;
315         int i, matched = 0;
316         /* Iterate through all in nodes linking matches */
317         for (i = 0; i < sk_X509_POLICY_NODE_num(last->nodes); i++)
318                 {
319                 node = sk_X509_POLICY_NODE_value(last->nodes, i);
320                 if (policy_node_match(last, node, data->valid_policy))
321                         {
322                         if (!level_add_node(curr, data, node, NULL))
323                                 return 0;
324                         matched = 1;
325                         }
```



```
326                 }
327         if (!matched && last->anyPolicy)
328                 {
329                 if (!level_add_node(curr, data, last->anyPolicy, NULL))
330                         return 0;
331                 }
332         return 1;
333         }

335 /* This corresponds to RFC3280 6.1.3(d)(1):
336  * link any data from CertificatePolicies onto matching parent
337  * or anyPolicy if no match.
338  */

340 static int tree_link_nodes(X509_POLICY_LEVEL *curr,
341                                 const X509_POLICY_CACHE *cache)
342         {
343         int i;
344         X509_POLICY_DATA *data;

346         for (i = 0; i < sk_X509_POLICY_DATA_num(cache->data); i++)
347                 {
348                 data = sk_X509_POLICY_DATA_value(cache->data, i);
349                 /* If a node is mapped any it doesn't have a corresponding
350                  * CertificatePolicies entry.
351                  * However such an identical node would be created
352                  * if anyPolicy matching is enabled because there would be
353                  * no match with the parent valid_policy_set. So we create
354                  * link because then it will have the mapping flags
355                  * right and we can prune it later.
356                  */
357 #if 0
358                 if ((data->flags & POLICY_DATA_FLAG_MAPPED_ANY)
359                         && !(curr->flags & X509_V_FLAG_INHIBIT_ANY))
360                         continue;
361 #endif
362                 /* Look for matching nodes in previous level */
363                 if (!tree_link_matching_nodes(curr, data))
364                                 return 0;
365                 }
366         return 1;
367         }

369 /* This corresponds to RFC3280 6.1.3(d)(2):
370  * Create new data for any unmatched policies in the parent and link
371  * to anyPolicy.
372  */

374 static int tree_add_unmatched(X509_POLICY_LEVEL *curr,
375                         const X509_POLICY_CACHE *cache,
376                         const ASN1_OBJECT *id,
377                         X509_POLICY_NODE *node,
378                         X509_POLICY_TREE *tree)
379         {
380         X509_POLICY_DATA *data;
381         if (id == NULL)
382                 id = node->data->valid_policy;
383         /* Create a new node with qualifiers from anyPolicy and
384          * id from unmatched node.
385          */
386         data = policy_data_new(NULL, id, node_critical(node));

388         if (data == NULL)
389                 return 0;
390         /* Curr may not have anyPolicy */
391         data->qualifier_set = cache->anyPolicy->qualifier_set;
```

```
392         data->flags |= POLICY_DATA_FLAG_SHARED_QUALIFIERS;
393         if (!level_add_node(curr, data, node, tree))
394                 {
395                 policy_data_free(data);
396                 return 0;
397                 }

399         return 1;
400         }

402 static int tree_link_unmatched(X509_POLICY_LEVEL *curr,
403                         const X509_POLICY_CACHE *cache,
404                         X509_POLICY_NODE *node,
405                         X509_POLICY_TREE *tree)
406         {
407         const X509_POLICY_LEVEL *last = curr - 1;
408         int i;

410         if (        (last->flags & X509_V_FLAG_INHIBIT_MAP)
411                 || !(node->data->flags & POLICY_DATA_FLAG_MAPPED))
412                 {
413                 /* If no policy mapping: matched if one child present */
414                 if (node->nchild)
415                         return 1;
416                 if (!tree_add_unmatched(curr, cache, NULL, node, tree))
417                         return 0;
418                 /* Add it */
419                 }
420         else
421                 {
422                 /* If mapping: matched if one child per expected policy set */
423                 STACK_OF(ASN1_OBJECT) *expset = node->data->expected_policy_set;
424                 if (node->nchild == sk_ASN1_OBJECT_num(expset))
425                         return 1;
426                 /* Locate unmatched nodes */
427                 for (i = 0; i < sk_ASN1_OBJECT_num(expset); i++)
428                         {
429                         ASN1_OBJECT *oid = sk_ASN1_OBJECT_value(expset, i);
430                         if (level_find_node(curr, node, oid))
431                                 continue;
432                         if (!tree_add_unmatched(curr, cache, oid, node, tree))
433                                 return 0;
434                         }

436                 }

438         return 1;

440         }

442 static int tree_link_any(X509_POLICY_LEVEL *curr,
443                         const X509_POLICY_CACHE *cache,
444                         X509_POLICY_TREE *tree)
445         {
446         int i;
447         /*X509_POLICY_DATA *data;*/
448         X509_POLICY_NODE *node;
449         X509_POLICY_LEVEL *last = curr - 1;

451         for (i = 0; i < sk_X509_POLICY_NODE_num(last->nodes); i++)
452                 {
453                 node = sk_X509_POLICY_NODE_value(last->nodes, i);

455                 if (!tree_link_unmatched(curr, cache, node, tree))
456                         return 0;
```



```
458 #if 0

460                 /* Skip any node with any children: we only want unmathced
461                  * nodes.
462                  *
463                  * Note: need something better for policy mapping
464                  * because each node may have multiple children
465                  */
466                 if (node->nchild)
467                         continue;

469                 /* Create a new node with qualifiers from anyPolicy and
470                  * id from unmatched node.
471                  */
472                 data = policy_data_new(NULL, node->data->valid_policy,
473                                                 node_critical(node));

475                 if (data == NULL)
476                         return 0;
477                 /* Curr may not have anyPolicy */
478                 data->qualifier_set = cache->anyPolicy->qualifier_set;
479                 data->flags |= POLICY_DATA_FLAG_SHARED_QUALIFIERS;
480                 if (!level_add_node(curr, data, node, tree))
481                         {
482                         policy_data_free(data);
483                         return 0;
484                         }

486 #endif

488                 }
489         /* Finally add link to anyPolicy */
490         if (last->anyPolicy)
491                 {
492                 if (!level_add_node(curr, cache->anyPolicy,
493                                                 last->anyPolicy, NULL))
494                         return 0;
495                 }
496         return 1;
497         }

499 /* Prune the tree: delete any child mapped child data on the current level
500  * then proceed up the tree deleting any data with no children. If we ever
501  * have no data on a level we can halt because the tree will be empty.
502  */

504 static int tree_prune(X509_POLICY_TREE *tree, X509_POLICY_LEVEL *curr)
505         {
506         STACK_OF(X509_POLICY_NODE) *nodes;
507         X509_POLICY_NODE *node;
508         int i;
509         nodes = curr->nodes;
510         if (curr->flags & X509_V_FLAG_INHIBIT_MAP)
511                 {
512                 for (i = sk_X509_POLICY_NODE_num(nodes) - 1; i >= 0; i--)
513                         {
514                         node = sk_X509_POLICY_NODE_value(nodes, i);
515                         /* Delete any mapped data: see RFC3280 XXXX */
516                         if (node->data->flags & POLICY_DATA_FLAG_MAP_MASK)
517                                 {
518                                 node->parent->nchild--;
519                                 OPENSSL_free(node);
520                                 (void)sk_X509_POLICY_NODE_delete(nodes,i);
521                                 }
522                         }
523                 }
```

```
525         for(;;) {
526                 --curr;
527                 nodes = curr->nodes;
528                 for (i = sk_X509_POLICY_NODE_num(nodes) - 1; i >= 0; i--)
529                         {
530                         node = sk_X509_POLICY_NODE_value(nodes, i);
531                         if (node->nchild == 0)
532                                 {
533                                 node->parent->nchild--;
534                                 OPENSSL_free(node);
535                                 (void)sk_X509_POLICY_NODE_delete(nodes, i);
536                                 }
537                         }
538                 if (curr->anyPolicy && !curr->anyPolicy->nchild)
539                         {
540                         if (curr->anyPolicy->parent)
541                                 curr->anyPolicy->parent->nchild--;
542                         OPENSSL_free(curr->anyPolicy);
543                         curr->anyPolicy = NULL;
544                         }
545                 if (curr == tree->levels)
546                         {
547                         /* If we zapped anyPolicy at top then tree is empty */
548                         if (!curr->anyPolicy)
549                                         return 2;
550                         return 1;
551                         }
552                 }

554         return 1;

556         }

558 static int tree_add_auth_node(STACK_OF(X509_POLICY_NODE) **pnodes,
559                                                 X509_POLICY_NODE *pcy)
560         {
561         if (!*pnodes)
562                 {
563                 *pnodes = policy_node_cmp_new();
564                 if (!*pnodes)
565                         return 0;
566                 }
567         else if (sk_X509_POLICY_NODE_find(*pnodes, pcy) != -1)
568                 return 1;

570         if (!sk_X509_POLICY_NODE_push(*pnodes, pcy))
571                 return 0;

573         return 1;

575         }

577 /* Calculate the authority set based on policy tree.
578  * The 'pnodes' parameter is used as a store for the set of policy nodes
579  * used to calculate the user set. If the authority set is not anyPolicy
580  * then pnodes will just point to the authority set. If however the authority
581  * set is anyPolicy then the set of valid policies (other than anyPolicy)
582  * is store in pnodes. The return value of '2' is used in this case to indicate
583  * that pnodes should be freed.
584  */

586 static int tree_calculate_authority_set(X509_POLICY_TREE *tree,
587                                         STACK_OF(X509_POLICY_NODE) **pnodes)
588         {
589         X509_POLICY_LEVEL *curr;
```



```
590         X509_POLICY_NODE *node, *anyptr;
591         STACK_OF(X509_POLICY_NODE) **addnodes;
592         int i, j;
593         curr = tree->levels + tree->nlevel - 1;

595         /* If last level contains anyPolicy set is anyPolicy */
596         if (curr->anyPolicy)
597                 {
598                 if (!tree_add_auth_node(&tree->auth_policies, curr->anyPolicy))
599                         return 0;
600                 addnodes = pnodes;
601                 }
602         else
603                 /* Add policies to authority set */
604                 addnodes = &tree->auth_policies;

606         curr = tree->levels;
607         for (i = 1; i < tree->nlevel; i++)
608                 {
609                 /* If no anyPolicy node on this this level it can't
610                  * appear on lower levels so end search.
611                  */
612                 if (!(anyptr = curr->anyPolicy))
613                         break;
614                 curr++;
615                 for (j = 0; j < sk_X509_POLICY_NODE_num(curr->nodes); j++)
616                         {
617                         node = sk_X509_POLICY_NODE_value(curr->nodes, j);
618                         if ((node->parent == anyptr)
619                                 && !tree_add_auth_node(addnodes, node))
620                                         return 0;
621                         }
622                 }

624         if (addnodes == pnodes)
625                 return 2;

627         *pnodes = tree->auth_policies;

629         return 1;
630         }

632 static int tree_calculate_user_set(X509_POLICY_TREE *tree,
633                                    STACK_OF(ASN1_OBJECT) *policy_oids,
634                                    STACK_OF(X509_POLICY_NODE) *auth_nodes)
635         {
636         int i;
637         X509_POLICY_NODE *node;
638         ASN1_OBJECT *oid;

640         X509_POLICY_NODE *anyPolicy;
641         X509_POLICY_DATA *extra;

643         /* Check if anyPolicy present in authority constrained policy set:
644          * this will happen if it is a leaf node.
645          */

647         if (sk_ASN1_OBJECT_num(policy_oids) <= 0)
648                 return 1;

650         anyPolicy = tree->levels[tree->nlevel - 1].anyPolicy;

652         for (i = 0; i < sk_ASN1_OBJECT_num(policy_oids); i++)
653                 {
654                 oid = sk_ASN1_OBJECT_value(policy_oids, i);
655                 if (OBJ_obj2nid(oid) == NID_any_policy)
```

```
656                         {
657                         tree->flags |= POLICY_FLAG_ANY_POLICY;
658                         return 1;
659                         }
660                 }

662         for (i = 0; i < sk_ASN1_OBJECT_num(policy_oids); i++)
663                 {
664                 oid = sk_ASN1_OBJECT_value(policy_oids, i);
665                 node = tree_find_sk(auth_nodes, oid);
666                 if (!node)
667                         {
668                         if (!anyPolicy)
669                                 continue;
670                         /* Create a new node with policy ID from user set
671                          * and qualifiers from anyPolicy.
672                          */
673                         extra = policy_data_new(NULL, oid,
674                                                 node_critical(anyPolicy));
675                         if (!extra)
676                                 return 0;
677                         extra->qualifier_set = anyPolicy->data->qualifier_set;
678                         extra->flags = POLICY_DATA_FLAG_SHARED_QUALIFIERS
679                                         | POLICY_DATA_FLAG_EXTRA_NODE;
680                         node = level_add_node(NULL, extra, anyPolicy->parent,
681                                                 tree);
682                         }
683                 if (!tree->user_policies)
684                         {
685                         tree->user_policies = sk_X509_POLICY_NODE_new_null();
686                         if (!tree->user_policies)
687                                 return 1;
688                         }
689                 if (!sk_X509_POLICY_NODE_push(tree->user_policies, node))
690                         return 0;
691                 }
692         return 1;

694         }

696 static int tree_evaluate(X509_POLICY_TREE *tree)
697         {
698         int ret, i;
699         X509_POLICY_LEVEL *curr = tree->levels + 1;
700         const X509_POLICY_CACHE *cache;

702         for(i = 1; i < tree->nlevel; i++, curr++)
703                 {
704                 cache = policy_cache_set(curr->cert);
705                 if (!tree_link_nodes(curr, cache))
706                         return 0;

708                 if (!(curr->flags & X509_V_FLAG_INHIBIT_ANY)
709                         && !tree_link_any(curr, cache, tree))
710                         return 0;
711         tree_print("before tree_prune()", tree, curr);
712                 ret = tree_prune(tree, curr);
713                 if (ret != 1)
714                         return ret;
715                 }

717         return 1;

719         }

721 static void exnode_free(X509_POLICY_NODE *node)
```



```
722         {
723         if (node->data && (node->data->flags & POLICY_DATA_FLAG_EXTRA_NODE))
724                 OPENSSL_free(node);
725         }


728 void X509_policy_tree_free(X509_POLICY_TREE *tree)
729         {
730         X509_POLICY_LEVEL *curr;
731         int i;

733         if (!tree)
734                 return;

736         sk_X509_POLICY_NODE_free(tree->auth_policies);
737         sk_X509_POLICY_NODE_pop_free(tree->user_policies, exnode_free);

739         for(i = 0, curr = tree->levels; i < tree->nlevel; i++, curr++)
740                 {
741                 if (curr->cert)
742                         X509_free(curr->cert);
743                 if (curr->nodes)
744                         sk_X509_POLICY_NODE_pop_free(curr->nodes,
745                                                 policy_node_free);
746                 if (curr->anyPolicy)
747                         policy_node_free(curr->anyPolicy);
748                 }

750         if (tree->extra_data)
751                 sk_X509_POLICY_DATA_pop_free(tree->extra_data,
752                                                 policy_data_free);

754         OPENSSL_free(tree->levels);
755         OPENSSL_free(tree);

757         }

759 /* Application policy checking function.
760  * Return codes:
761  *  0   Internal Error.
762  *  1   Successful.
763  * -1   One or more certificates contain invalid or inconsistent extensions
764  * -2   User constrained policy set empty and requireExplicit true.
765  */

767 int X509_policy_check(X509_POLICY_TREE **ptree, int *pexplicit_policy,
768                         STACK_OF(X509) *certs,
769                         STACK_OF(ASN1_OBJECT) *policy_oids,
770                         unsigned int flags)
771         {
772         int ret;
773         X509_POLICY_TREE *tree = NULL;
774         STACK_OF(X509_POLICY_NODE) *nodes, *auth_nodes = NULL;
775         *ptree = NULL;

777         *pexplicit_policy = 0;
778         ret = tree_init(&tree, certs, flags);

780         switch (ret)
781                 {

783                 /* Tree empty requireExplicit False: OK */
784                 case 2:
785                 return 1;

787                 /* Some internal error */
```

```
788                 case -1:
789                 return -1;

791                 /* Some internal error */
792                 case 0:
793                 return 0;

795                 /* Tree empty requireExplicit True: Error */

797                 case 6:
798                 *pexplicit_policy = 1;
799                 return -2;

801                 /* Tree OK requireExplicit True: OK and continue */
802                 case 5:
803                 *pexplicit_policy = 1;
804                 break;

806                 /* Tree OK: continue */

808                 case 1:
809                 if (!tree)
810                         /*
811                          * tree_init() returns success and a null tree
812                          * if it's just looking at a trust anchor.
813                          * I'm not sure that returning success here is
814                          * correct, but I'm sure that reporting this
815                          * as an internal error which our caller
816                          * interprets as a malloc failure is wrong.
817                          */
818                         return 1;
819                 break;
820                 }

822         if (!tree) goto error;
823         ret = tree_evaluate(tree);

825         tree_print("tree_evaluate()", tree, NULL);

827         if (ret <= 0)
828                 goto error;

830         /* Return value 2 means tree empty */
831         if (ret == 2)
832                 {
833                 X509_policy_tree_free(tree);
834                 if (*pexplicit_policy)
835                         return -2;
836                 else
837                         return 1;
838                 }

840         /* Tree is not empty: continue */

842         ret = tree_calculate_authority_set(tree, &auth_nodes);

844         if (!ret)
845                 goto error;

847         if (!tree_calculate_user_set(tree, policy_oids, auth_nodes))
848                 goto error;

850         if (ret == 2)
851                 sk_X509_POLICY_NODE_free(auth_nodes);

853         if (tree)
```



```
854                 *ptree = tree;

856         if (*pexplicit_policy)
857                 {
858                 nodes = X509_policy_tree_get0_user_policies(tree);
859                 if (sk_X509_POLICY_NODE_num(nodes) <= 0)
860                         return -2;
861                 }

863         return 1;

865         error:

867         X509_policy_tree_free(tree);

869         return 0;

871         }
872 #endif /* ! codereview */
```

```
**********************************************************
   37236 Wed Aug 13 19:53:28 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_addr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Contributed to the OpenSSL Project by the American Registry for
 * Internet Numbers ("ARIN").
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 */

/*
 * Implementation of RFC 3779 section 2.2.
 */
```



```
#include <stdio.h>
#include <stdlib.h>

#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/buffer.h>
#include <openssl/x509v3.h>

#ifndef OPENSSL_NO_RFC3779

/*
 * OpenSSL ASN.1 template translation of RFC 3779 2.2.3.
 */

ASN1_SEQUENCE(IPAddressRange) = {
  ASN1_SIMPLE(IPAddressRange, min, ASN1_BIT_STRING),
  ASN1_SIMPLE(IPAddressRange, max, ASN1_BIT_STRING)
} ASN1_SEQUENCE_END(IPAddressRange)

ASN1_CHOICE(IPAddressOrRange) = {
  ASN1_SIMPLE(IPAddressOrRange, u.addressPrefix, ASN1_BIT_STRING),
  ASN1_SIMPLE(IPAddressOrRange, u.addressRange,  IPAddressRange)
} ASN1_CHOICE_END(IPAddressOrRange)

ASN1_CHOICE(IPAddressChoice) = {
  ASN1_SIMPLE(IPAddressChoice,      u.inherit,           ASN1_NULL),
  ASN1_SEQUENCE_OF(IPAddressChoice, u.addressesOrRanges, IPAddressOrRange)
} ASN1_CHOICE_END(IPAddressChoice)

ASN1_SEQUENCE(IPAddressFamily) = {
  ASN1_SIMPLE(IPAddressFamily, addressFamily,   ASN1_OCTET_STRING),
  ASN1_SIMPLE(IPAddressFamily, ipAddressChoice, IPAddressChoice)
} ASN1_SEQUENCE_END(IPAddressFamily)

ASN1_ITEM_TEMPLATE(IPAddrBlocks) =
  ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0,
                        IPAddrBlocks, IPAddressFamily)
ASN1_ITEM_TEMPLATE_END(IPAddrBlocks)

IMPLEMENT_ASN1_FUNCTIONS(IPAddressRange)
IMPLEMENT_ASN1_FUNCTIONS(IPAddressOrRange)
IMPLEMENT_ASN1_FUNCTIONS(IPAddressChoice)
IMPLEMENT_ASN1_FUNCTIONS(IPAddressFamily)

/*
 * How much buffer space do we need for a raw address?
 */
#define ADDR_RAW_BUF_LEN        16

/*
 * What's the address length associated with this AFI?
 */
static int length_from_afi(const unsigned afi)
{
  switch (afi) {
  case IANA_AFI_IPV4:
    return 4;
  case IANA_AFI_IPV6:
    return 16;
  default:
    return 0;
  }
}
```

```
/*
 * Extract the AFI from an IPAddressFamily.
 */
unsigned int v3_addr_get_afi(const IPAddressFamily *f)
{
  return ((f != NULL &&
           f->addressFamily != NULL &&
           f->addressFamily->data != NULL)
          ? ((f->addressFamily->data[0] << 8) |
             (f->addressFamily->data[1]))
          : 0);
}

/*
 * Expand the bitstring form of an address into a raw byte array.
 * At the moment this is coded for simplicity, not speed.
 */
static int addr_expand(unsigned char *addr,
                       const ASN1_BIT_STRING *bs,
                       const int length,
                       const unsigned char fill)
{
  if (bs->length < 0 || bs->length > length)
    return 0;
  if (bs->length > 0) {
    memcpy(addr, bs->data, bs->length);
    if ((bs->flags & 7) != 0) {
      unsigned char mask = 0xFF >> (8 - (bs->flags & 7));
      if (fill == 0)
        addr[bs->length - 1] &= ~mask;
      else
        addr[bs->length - 1] |= mask;
    }
  }
  memset(addr + bs->length, fill, length - bs->length);
  return 1;
}

/*
 * Extract the prefix length from a bitstring.
 */
#define addr_prefixlen(bs) ((int) ((bs)->length * 8 - ((bs)->flags & 7)))

/*
 * i2r handler for one address bitstring.
 */
static int i2r_address(BIO *out,
                       const unsigned afi,
                       const unsigned char fill,
                       const ASN1_BIT_STRING *bs)
{
  unsigned char addr[ADDR_RAW_BUF_LEN];
  int i, n;

  if (bs->length < 0)
    return 0;
  switch (afi) {
  case IANA_AFI_IPV4:
    if (!addr_expand(addr, bs, 4, fill))
      return 0;
    BIO_printf(out, "%d.%d.%d.%d", addr[0], addr[1], addr[2], addr[3]);
    break;
  case IANA_AFI_IPV6:
    if (!addr_expand(addr, bs, 16, fill))
      return 0;
    for (n = 16; n > 1 && addr[n-1] == 0x00 && addr[n-2] == 0x00; n -= 2)
```



```
      ;
    for (i = 0; i < n; i += 2)
      BIO_printf(out, "%x%s", (addr[i] << 8) | addr[i+1], (i < 14 ? ":" : ""));
    if (i < 16)
      BIO_puts(out, ":");
    if (i == 0)
      BIO_puts(out, ":");
    break;
  default:
    for (i = 0; i < bs->length; i++)
      BIO_printf(out, "%s%02x", (i > 0 ? ":" : ""), bs->data[i]);
    BIO_printf(out, "[%d]", (int) (bs->flags & 7));
    break;
  }
  return 1;
}

/*
 * i2r handler for a sequence of addresses and ranges.
 */
static int i2r_IPAddressOrRanges(BIO *out,
                                 const int indent,
                                 const IPAddressOrRanges *aors,
                                 const unsigned afi)
{
  int i;
  for (i = 0; i < sk_IPAddressOrRange_num(aors); i++) {
    const IPAddressOrRange *aor = sk_IPAddressOrRange_value(aors, i);
    BIO_printf(out, "%*s", indent, "");
    switch (aor->type) {
    case IPAddressOrRange_addressPrefix:
      if (!i2r_address(out, afi, 0x00, aor->u.addressPrefix))
        return 0;
      BIO_printf(out, "/%d\n", addr_prefixlen(aor->u.addressPrefix));
      continue;
    case IPAddressOrRange_addressRange:
      if (!i2r_address(out, afi, 0x00, aor->u.addressRange->min))
        return 0;
      BIO_puts(out, "-");
      if (!i2r_address(out, afi, 0xFF, aor->u.addressRange->max))
        return 0;
      BIO_puts(out, "\n");
      continue;
    }
  }
  return 1;
}

/*
 * i2r handler for an IPAddrBlocks extension.
 */
static int i2r_IPAddrBlocks(const X509V3_EXT_METHOD *method,
                            void *ext,
                            BIO *out,
                            int indent)
{
  const IPAddrBlocks *addr = ext;
  int i;
  for (i = 0; i < sk_IPAddressFamily_num(addr); i++) {
    IPAddressFamily *f = sk_IPAddressFamily_value(addr, i);
    const unsigned int afi = v3_addr_get_afi(f);
    switch (afi) {
    case IANA_AFI_IPV4:
      BIO_printf(out, "%*sIPv4", indent, "");
      break;
    case IANA_AFI_IPV6:
```

```
260       BIO_printf(out, "%*sIPv6", indent, "");
261       break;
262     default:
263       BIO_printf(out, "%*sUnknown AFI %u", indent, "", afi);
264       break;
265     }
266     if (f->addressFamily->length > 2) {
267       switch (f->addressFamily->data[2]) {
268       case   1:
269         BIO_puts(out, " (Unicast)");
270         break;
271       case   2:
272         BIO_puts(out, " (Multicast)");
273         break;
274       case   3:
275         BIO_puts(out, " (Unicast/Multicast)");
276         break;
277       case   4:
278         BIO_puts(out, " (MPLS)");
279         break;
280       case  64:
281         BIO_puts(out, " (Tunnel)");
282         break;
283       case  65:
284         BIO_puts(out, " (VPLS)");
285         break;
286       case  66:
287         BIO_puts(out, " (BGP MDT)");
288         break;
289       case 128:
290         BIO_puts(out, " (MPLS-labeled VPN)");
291         break;
292       default:
293         BIO_printf(out, " (Unknown SAFI %u)",
294                    (unsigned) f->addressFamily->data[2]);
295         break;
296       }
297     }
298     switch (f->ipAddressChoice->type) {
299     case IPAddressChoice_inherit:
300       BIO_puts(out, ": inherit\n");
301       break;
302     case IPAddressChoice_addressesOrRanges:
303       BIO_puts(out, ":\n");
304       if (!i2r_IPAddressOrRanges(out,
305                                  indent + 2,
306                                  f->ipAddressChoice->u.addressesOrRanges,
307                                  afi))
308         return 0;
309       break;
310     }
311   }
312   return 1;
313 }

315 /*
316  * Sort comparison function for a sequence of IPAddressOrRange
317  * elements.
318  *
319  * There's no sane answer we can give if addr_expand() fails, and an
320  * assertion failure on externally supplied data is seriously uncool,
321  * so we just arbitrarily declare that if given invalid inputs this
322  * function returns -1.  If this messes up your preferred sort order
323  * for garbage input, tough noogies.
324  */
325 static int IPAddressOrRange_cmp(const IPAddressOrRange *a,
```



```
326                                 const IPAddressOrRange *b,
327                                 const int length)
328 {
329   unsigned char addr_a[ADDR_RAW_BUF_LEN], addr_b[ADDR_RAW_BUF_LEN];
330   int prefixlen_a = 0, prefixlen_b = 0;
331   int r;

333   switch (a->type) {
334   case IPAddressOrRange_addressPrefix:
335     if (!addr_expand(addr_a, a->u.addressPrefix, length, 0x00))
336       return -1;
337     prefixlen_a = addr_prefixlen(a->u.addressPrefix);
338     break;
339   case IPAddressOrRange_addressRange:
340     if (!addr_expand(addr_a, a->u.addressRange->min, length, 0x00))
341       return -1;
342     prefixlen_a = length * 8;
343     break;
344   }

346   switch (b->type) {
347   case IPAddressOrRange_addressPrefix:
348     if (!addr_expand(addr_b, b->u.addressPrefix, length, 0x00))
349       return -1;
350     prefixlen_b = addr_prefixlen(b->u.addressPrefix);
351     break;
352   case IPAddressOrRange_addressRange:
353     if (!addr_expand(addr_b, b->u.addressRange->min, length, 0x00))
354       return -1;
355     prefixlen_b = length * 8;
356     break;
357   }

359   if ((r = memcmp(addr_a, addr_b, length)) != 0)
360     return r;
361   else
362     return prefixlen_a - prefixlen_b;
363 }

365 /*
366  * IPv4-specific closure over IPAddressOrRange_cmp, since sk_sort()
367  * comparision routines are only allowed two arguments.
368  */
369 static int v4IPAddressOrRange_cmp(const IPAddressOrRange * const *a,
370                                   const IPAddressOrRange * const *b)
371 {
372   return IPAddressOrRange_cmp(*a, *b, 4);
373 }

375 /*
376  * IPv6-specific closure over IPAddressOrRange_cmp, since sk_sort()
377  * comparision routines are only allowed two arguments.
378  */
379 static int v6IPAddressOrRange_cmp(const IPAddressOrRange * const *a,
380                                   const IPAddressOrRange * const *b)
381 {
382   return IPAddressOrRange_cmp(*a, *b, 16);
383 }

385 /*
386  * Calculate whether a range collapses to a prefix.
387  * See last paragraph of RFC 3779 2.2.3.7.
388  */
389 static int range_should_be_prefix(const unsigned char *min,
390                                   const unsigned char *max,
391                                   const int length)
```

```
{
  unsigned char mask;
  int i, j;

  OPENSSL_assert(memcmp(min, max, length) <= 0);
  for (i = 0; i < length && min[i] == max[i]; i++)
    ;
  for (j = length - 1; j >= 0 && min[j] == 0x00 && max[j] == 0xFF; j--)
    ;
  if (i < j)
    return -1;
  if (i > j)
    return i * 8;
  mask = min[i] ^ max[i];
  switch (mask) {
  case 0x01: j = 7; break;
  case 0x03: j = 6; break;
  case 0x07: j = 5; break;
  case 0x0F: j = 4; break;
  case 0x1F: j = 3; break;
  case 0x3F: j = 2; break;
  case 0x7F: j = 1; break;
  default:   return -1;
  }
  if ((min[i] & mask) != 0 || (max[i] & mask) != mask)
    return -1;
  else
    return i * 8 + j;
}

/*
 * Construct a prefix.
 */
static int make_addressPrefix(IPAddressOrRange **result,
                              unsigned char *addr,
                              const int prefixlen)
{
  int bytelen = (prefixlen + 7) / 8, bitlen = prefixlen % 8;
  IPAddressOrRange *aor = IPAddressOrRange_new();

  if (aor == NULL)
    return 0;
  aor->type = IPAddressOrRange_addressPrefix;
  if (aor->u.addressPrefix == NULL &&
      (aor->u.addressPrefix = ASN1_BIT_STRING_new()) == NULL)
    goto err;
  if (!ASN1_BIT_STRING_set(aor->u.addressPrefix, addr, bytelen))
    goto err;
  aor->u.addressPrefix->flags &= ~7;
  aor->u.addressPrefix->flags |= ASN1_STRING_FLAG_BITS_LEFT;
  if (bitlen > 0) {
    aor->u.addressPrefix->data[bytelen - 1] &= ~(0xFF >> bitlen);
    aor->u.addressPrefix->flags |= 8 - bitlen;
  }

  *result = aor;
  return 1;

 err:
  IPAddressOrRange_free(aor);
  return 0;
}

/*
 * Construct a range.  If it can be expressed as a prefix,
 * return a prefix instead.  Doing this here simplifies
```



```
 * the rest of the code considerably.
 */
static int make_addressRange(IPAddressOrRange **result,
                             unsigned char *min,
                             unsigned char *max,
                             const int length)
{
  IPAddressOrRange *aor;
  int i, prefixlen;

  if ((prefixlen = range_should_be_prefix(min, max, length)) >= 0)
    return make_addressPrefix(result, min, prefixlen);

  if ((aor = IPAddressOrRange_new()) == NULL)
    return 0;
  aor->type = IPAddressOrRange_addressRange;
  OPENSSL_assert(aor->u.addressRange == NULL);
  if ((aor->u.addressRange = IPAddressRange_new()) == NULL)
    goto err;
  if (aor->u.addressRange->min == NULL &&
      (aor->u.addressRange->min = ASN1_BIT_STRING_new()) == NULL)
    goto err;
  if (aor->u.addressRange->max == NULL &&
      (aor->u.addressRange->max = ASN1_BIT_STRING_new()) == NULL)
    goto err;

  for (i = length; i > 0 && min[i - 1] == 0x00; --i)
    ;
  if (!ASN1_BIT_STRING_set(aor->u.addressRange->min, min, i))
    goto err;
  aor->u.addressRange->min->flags &= ~7;
  aor->u.addressRange->min->flags |= ASN1_STRING_FLAG_BITS_LEFT;
  if (i > 0) {
    unsigned char b = min[i - 1];
    int j = 1;
    while ((b & (0xFFU >> j)) != 0)
      ++j;
    aor->u.addressRange->min->flags |= 8 - j;
  }

  for (i = length; i > 0 && max[i - 1] == 0xFF; --i)
    ;
  if (!ASN1_BIT_STRING_set(aor->u.addressRange->max, max, i))
    goto err;
  aor->u.addressRange->max->flags &= ~7;
  aor->u.addressRange->max->flags |= ASN1_STRING_FLAG_BITS_LEFT;
  if (i > 0) {
    unsigned char b = max[i - 1];
    int j = 1;
    while ((b & (0xFFU >> j)) != (0xFFU >> j))
      ++j;
    aor->u.addressRange->max->flags |= 8 - j;
  }

  *result = aor;
  return 1;

 err:
  IPAddressOrRange_free(aor);
  return 0;
}

/*
 * Construct a new address family or find an existing one.
 */
static IPAddressFamily *make_IPAddressFamily(IPAddrBlocks *addr,
```

```
524                                             const unsigned afi,
525                                             const unsigned *safi)
526 {
527   IPAddressFamily *f;
528   unsigned char key[3];
529   unsigned keylen;
530   int i;

532   key[0] = (afi >> 8) & 0xFF;
533   key[1] = afi & 0xFF;
534   if (safi != NULL) {
535     key[2] = *safi & 0xFF;
536     keylen = 3;
537   } else {
538     keylen = 2;
539   }

541   for (i = 0; i < sk_IPAddressFamily_num(addr); i++) {
542     f = sk_IPAddressFamily_value(addr, i);
543     OPENSSL_assert(f->addressFamily->data != NULL);
544     if (f->addressFamily->length == keylen &&
545         !memcmp(f->addressFamily->data, key, keylen))
546       return f;
547   }

549   if ((f = IPAddressFamily_new()) == NULL)
550     goto err;
551   if (f->ipAddressChoice == NULL &&
552       (f->ipAddressChoice = IPAddressChoice_new()) == NULL)
553     goto err;
554   if (f->addressFamily == NULL &&
555       (f->addressFamily = ASN1_OCTET_STRING_new()) == NULL)
556     goto err;
557   if (!ASN1_OCTET_STRING_set(f->addressFamily, key, keylen))
558     goto err;
559   if (!sk_IPAddressFamily_push(addr, f))
560     goto err;

562   return f;

564  err:
565   IPAddressFamily_free(f);
566   return NULL;
567 }

569 /*
570  * Add an inheritance element.
571  */
572 int v3_addr_add_inherit(IPAddrBlocks *addr,
573                         const unsigned afi,
574                         const unsigned *safi)
575 {
576   IPAddressFamily *f = make_IPAddressFamily(addr, afi, safi);
577   if (f == NULL ||
578       f->ipAddressChoice == NULL ||
579       (f->ipAddressChoice->type == IPAddressChoice_addressesOrRanges &&
580        f->ipAddressChoice->u.addressesOrRanges != NULL))
581     return 0;
582   if (f->ipAddressChoice->type == IPAddressChoice_inherit &&
583       f->ipAddressChoice->u.inherit != NULL)
584     return 1;
585   if (f->ipAddressChoice->u.inherit == NULL &&
586       (f->ipAddressChoice->u.inherit = ASN1_NULL_new()) == NULL)
587     return 0;
588   f->ipAddressChoice->type = IPAddressChoice_inherit;
589   return 1;
```



```
590 }

592 /*
593  * Construct an IPAddressOrRange sequence, or return an existing one.
594  */
595 static IPAddressOrRanges *make_prefix_or_range(IPAddrBlocks *addr,
596                                                const unsigned afi,
597                                                const unsigned *safi)
598 {
599   IPAddressFamily *f = make_IPAddressFamily(addr, afi, safi);
600   IPAddressOrRanges *aors = NULL;

602   if (f == NULL ||
603       f->ipAddressChoice == NULL ||
604       (f->ipAddressChoice->type == IPAddressChoice_inherit &&
605        f->ipAddressChoice->u.inherit != NULL))
606     return NULL;
607   if (f->ipAddressChoice->type == IPAddressChoice_addressesOrRanges)
608     aors = f->ipAddressChoice->u.addressesOrRanges;
609   if (aors != NULL)
610     return aors;
611   if ((aors = sk_IPAddressOrRange_new_null()) == NULL)
612     return NULL;
613   switch (afi) {
614   case IANA_AFI_IPV4:
615     (void) sk_IPAddressOrRange_set_cmp_func(aors, v4IPAddressOrRange_cmp);
616     break;
617   case IANA_AFI_IPV6:
618     (void) sk_IPAddressOrRange_set_cmp_func(aors, v6IPAddressOrRange_cmp);
619     break;
620   }
621   f->ipAddressChoice->type = IPAddressChoice_addressesOrRanges;
622   f->ipAddressChoice->u.addressesOrRanges = aors;
623   return aors;
624 }

626 /*
627  * Add a prefix.
628  */
629 int v3_addr_add_prefix(IPAddrBlocks *addr,
630                        const unsigned afi,
631                        const unsigned *safi,
632                        unsigned char *a,
633                        const int prefixlen)
634 {
635   IPAddressOrRanges *aors = make_prefix_or_range(addr, afi, safi);
636   IPAddressOrRange *aor;
637   if (aors == NULL || !make_addressPrefix(&aor, a, prefixlen))
638     return 0;
639   if (sk_IPAddressOrRange_push(aors, aor))
640     return 1;
641   IPAddressOrRange_free(aor);
642   return 0;
643 }

645 /*
646  * Add a range.
647  */
648 int v3_addr_add_range(IPAddrBlocks *addr,
649                       const unsigned afi,
650                       const unsigned *safi,
651                       unsigned char *min,
652                       unsigned char *max)
653 {
654   IPAddressOrRanges *aors = make_prefix_or_range(addr, afi, safi);
655   IPAddressOrRange *aor;
```

```
656   int length = length_from_afi(afi);
657   if (aors == NULL)
658     return 0;
659   if (!make_addressRange(&aor, min, max, length))
660     return 0;
661   if (sk_IPAddressOrRange_push(aors, aor))
662     return 1;
663   IPAddressOrRange_free(aor);
664   return 0;
665 }

667 /*
668  * Extract min and max values from an IPAddressOrRange.
669  */
670 static int extract_min_max(IPAddressOrRange *aor,
671                            unsigned char *min,
672                            unsigned char *max,
673                            int length)
674 {
675   if (aor == NULL || min == NULL || max == NULL)
676     return 0;
677   switch (aor->type) {
678   case IPAddressOrRange_addressPrefix:
679     return (addr_expand(min, aor->u.addressPrefix, length, 0x00) &&
680             addr_expand(max, aor->u.addressPrefix, length, 0xFF));
681   case IPAddressOrRange_addressRange:
682     return (addr_expand(min, aor->u.addressRange->min, length, 0x00) &&
683             addr_expand(max, aor->u.addressRange->max, length, 0xFF));
684   }
685   return 0;
686 }

688 /*
689  * Public wrapper for extract_min_max().
690  */
691 int v3_addr_get_range(IPAddressOrRange *aor,
692                       const unsigned afi,
693                       unsigned char *min,
694                       unsigned char *max,
695                       const int length)
696 {
697   int afi_length = length_from_afi(afi);
698   if (aor == NULL || min == NULL || max == NULL ||
699       afi_length == 0 || length < afi_length ||
700       (aor->type != IPAddressOrRange_addressPrefix &&
701        aor->type != IPAddressOrRange_addressRange) ||
702       !extract_min_max(aor, min, max, afi_length))
703     return 0;

705   return afi_length;
706 }

708 /*
709  * Sort comparision function for a sequence of IPAddressFamily.
710  *
711  * The last paragraph of RFC 3779 2.2.3.3 is slightly ambiguous about
712  * the ordering: I can read it as meaning that IPv6 without a SAFI
713  * comes before IPv4 with a SAFI, which seems pretty weird.  The
714  * examples in appendix B suggest that the author intended the
715  * null-SAFI rule to apply only within a single AFI, which is what I
716  * would have expected and is what the following code implements.
717  */
718 static int IPAddressFamily_cmp(const IPAddressFamily * const *a_,
719                                const IPAddressFamily * const *b_)
720 {
721   const ASN1_OCTET_STRING *a = (*a_)->addressFamily;
```



```
722   const ASN1_OCTET_STRING *b = (*b_)->addressFamily;
723   int len = ((a->length <= b->length) ? a->length : b->length);
724   int cmp = memcmp(a->data, b->data, len);
725   return cmp ? cmp : a->length - b->length;
726 }

728 /*
729  * Check whether an IPAddrBLocks is in canonical form.
730  */
731 int v3_addr_is_canonical(IPAddrBlocks *addr)
732 {
733   unsigned char a_min[ADDR_RAW_BUF_LEN], a_max[ADDR_RAW_BUF_LEN];
734   unsigned char b_min[ADDR_RAW_BUF_LEN], b_max[ADDR_RAW_BUF_LEN];
735   IPAddressOrRanges *aors;
736   int i, j, k;

738   /*
739    * Empty extension is cannonical.
740    */
741   if (addr == NULL)
742     return 1;

744   /*
745    * Check whether the top-level list is in order.
746    */
747   for (i = 0; i < sk_IPAddressFamily_num(addr) - 1; i++) {
748     const IPAddressFamily *a = sk_IPAddressFamily_value(addr, i);
749     const IPAddressFamily *b = sk_IPAddressFamily_value(addr, i + 1);
750     if (IPAddressFamily_cmp(&a, &b) >= 0)
751       return 0;
752   }

754   /*
755    * Top level's ok, now check each address family.
756    */
757   for (i = 0; i < sk_IPAddressFamily_num(addr); i++) {
758     IPAddressFamily *f = sk_IPAddressFamily_value(addr, i);
759     int length = length_from_afi(v3_addr_get_afi(f));

761     /*
762      * Inheritance is canonical.  Anything other than inheritance or
763      * a SEQUENCE OF IPAddressOrRange is an ASN.1 error or something.
764      */
765     if (f == NULL || f->ipAddressChoice == NULL)
766       return 0;
767     switch (f->ipAddressChoice->type) {
768     case IPAddressChoice_inherit:
769       continue;
770     case IPAddressChoice_addressesOrRanges:
771       break;
772     default:
773       return 0;
774     }

776     /*
777      * It's an IPAddressOrRanges sequence, check it.
778      */
779     aors = f->ipAddressChoice->u.addressesOrRanges;
780     if (sk_IPAddressOrRange_num(aors) == 0)
781       return 0;
782     for (j = 0; j < sk_IPAddressOrRange_num(aors) - 1; j++) {
783       IPAddressOrRange *a = sk_IPAddressOrRange_value(aors, j);
784       IPAddressOrRange *b = sk_IPAddressOrRange_value(aors, j + 1);

786       if (!extract_min_max(a, a_min, a_max, length) ||
787           !extract_min_max(b, b_min, b_max, length))
```

```
788         return 0;

790       /*
791        * Punt misordered list, overlapping start, or inverted range.
792        */
793       if (memcmp(a_min, b_min, length) >= 0 ||
794           memcmp(a_min, a_max, length) > 0 ||
795           memcmp(b_min, b_max, length) > 0)
796         return 0;

798       /*
799        * Punt if adjacent or overlapping.  Check for adjacency by
800        * subtracting one from b_min first.
801        */
802       for (k = length - 1; k >= 0 && b_min[k]-- == 0x00; k--)
803         ;
804       if (memcmp(a_max, b_min, length) >= 0)
805         return 0;

807       /*
808        * Check for range that should be expressed as a prefix.
809        */
810       if (a->type == IPAddressOrRange_addressRange &&
811           range_should_be_prefix(a_min, a_max, length) >= 0)
812         return 0;
813     }

815     /*
816      * Check range to see if it's inverted or should be a
817      * prefix.
818      */
819     j = sk_IPAddressOrRange_num(aors) - 1;
820     {
821       IPAddressOrRange *a = sk_IPAddressOrRange_value(aors, j);
822       if (a != NULL && a->type == IPAddressOrRange_addressRange) {
823         if (!extract_min_max(a, a_min, a_max, length))
824           return 0;
825         if (memcmp(a_min, a_max, length) > 0 ||
826             range_should_be_prefix(a_min, a_max, length) >= 0)
827           return 0;
828       }
829     }
830   }

832   /*
833    * If we made it through all that, we're happy.
834    */
835   return 1;
836 }

838 /*
839  * Whack an IPAddressOrRanges into canonical form.
840  */
841 static int IPAddressOrRanges_canonize(IPAddressOrRanges *aors,
842                                       const unsigned afi)
843 {
844   int i, j, length = length_from_afi(afi);

846   /*
847    * Sort the IPAddressOrRanges sequence.
848    */
849   sk_IPAddressOrRange_sort(aors);

851   /*
852    * Clean up representation issues, punt on duplicates or overlaps.
853    */
```



```
854   for (i = 0; i < sk_IPAddressOrRange_num(aors) - 1; i++) {
855     IPAddressOrRange *a = sk_IPAddressOrRange_value(aors, i);
856     IPAddressOrRange *b = sk_IPAddressOrRange_value(aors, i + 1);
857     unsigned char a_min[ADDR_RAW_BUF_LEN], a_max[ADDR_RAW_BUF_LEN];
858     unsigned char b_min[ADDR_RAW_BUF_LEN], b_max[ADDR_RAW_BUF_LEN];

860     if (!extract_min_max(a, a_min, a_max, length) ||
861         !extract_min_max(b, b_min, b_max, length))
862       return 0;

864     /*
865      * Punt inverted ranges.
866      */
867     if (memcmp(a_min, a_max, length) > 0 ||
868         memcmp(b_min, b_max, length) > 0)
869       return 0;

871     /*
872      * Punt overlaps.
873      */
874     if (memcmp(a_max, b_min, length) >= 0)
875       return 0;

877     /*
878      * Merge if a and b are adjacent.  We check for
879      * adjacency by subtracting one from b_min first.
880      */
881     for (j = length - 1; j >= 0 && b_min[j]-- == 0x00; j--)
882       ;
883     if (memcmp(a_max, b_min, length) == 0) {
884       IPAddressOrRange *merged;
885       if (!make_addressRange(&merged, a_min, b_max, length))
886         return 0;
887       (void) sk_IPAddressOrRange_set(aors, i, merged);
888       (void) sk_IPAddressOrRange_delete(aors, i + 1);
889       IPAddressOrRange_free(a);
890       IPAddressOrRange_free(b);
891       --i;
892       continue;
893     }
894   }

896   /*
897    * Check for inverted final range.
898    */
899   j = sk_IPAddressOrRange_num(aors) - 1;
900   {
901     IPAddressOrRange *a = sk_IPAddressOrRange_value(aors, j);
902     if (a != NULL && a->type == IPAddressOrRange_addressRange) {
903       unsigned char a_min[ADDR_RAW_BUF_LEN], a_max[ADDR_RAW_BUF_LEN];
904       extract_min_max(a, a_min, a_max, length);
905       if (memcmp(a_min, a_max, length) > 0)
906         return 0;
907     }
908   }

910   return 1;
911 }

913 /*
914  * Whack an IPAddrBlocks extension into canonical form.
915  */
916 int v3_addr_canonize(IPAddrBlocks *addr)
917 {
918   int i;
919   for (i = 0; i < sk_IPAddressFamily_num(addr); i++) {
```

```
    IPAddressFamily *f = sk_IPAddressFamily_value(addr, i);
    if (f->ipAddressChoice->type == IPAddressChoice_addressesOrRanges &&
        !IPAddressOrRanges_canonize(f->ipAddressChoice->u.addressesOrRanges,
                                    v3_addr_get_afi(f)))
      return 0;
  }
  (void) sk_IPAddressFamily_set_cmp_func(addr, IPAddressFamily_cmp);
  sk_IPAddressFamily_sort(addr);
  OPENSSL_assert(v3_addr_is_canonical(addr));
  return 1;
}

/*
 * v2i handler for the IPAddrBlocks extension.
 */
static void *v2i_IPAddrBlocks(const struct v3_ext_method *method,
                              struct v3_ext_ctx *ctx,
                              STACK_OF(CONF_VALUE) *values)
{
  static const char v4addr_chars[] = "0123456789.";
  static const char v6addr_chars[] = "0123456789.:abcdefABCDEF";
  IPAddrBlocks *addr = NULL;
  char *s = NULL, *t;
  int i;

  if ((addr = sk_IPAddressFamily_new(IPAddressFamily_cmp)) == NULL) {
    X509V3err(X509V3_F_V2I_IPADDRBLOCKS, ERR_R_MALLOC_FAILURE);
    return NULL;
  }

  for (i = 0; i < sk_CONF_VALUE_num(values); i++) {
    CONF_VALUE *val = sk_CONF_VALUE_value(values, i);
    unsigned char min[ADDR_RAW_BUF_LEN], max[ADDR_RAW_BUF_LEN];
    unsigned afi, *safi = NULL, safi_;
    const char *addr_chars;
    int prefixlen, i1, i2, delim, length;

    if (       !name_cmp(val->name, "IPv4")) {
      afi = IANA_AFI_IPV4;
    } else if (!name_cmp(val->name, "IPv6")) {
      afi = IANA_AFI_IPV6;
    } else if (!name_cmp(val->name, "IPv4-SAFI")) {
      afi = IANA_AFI_IPV4;
      safi = &safi_;
    } else if (!name_cmp(val->name, "IPv6-SAFI")) {
      afi = IANA_AFI_IPV6;
      safi = &safi_;
    } else {
      X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_EXTENSION_NAME_ERROR);
      X509V3_conf_err(val);
      goto err;
    }

    switch (afi) {
    case IANA_AFI_IPV4:
      addr_chars = v4addr_chars;
      break;
    case IANA_AFI_IPV6:
      addr_chars = v6addr_chars;
      break;
    }

    length = length_from_afi(afi);

    /*
     * Handle SAFI, if any, and BUF_strdup() so we can null-terminate
```



```
     * the other input values.
     */
    if (safi != NULL) {
      *safi = strtoul(val->value, &t, 0);
      t += strspn(t, " \t");
      if (*safi > 0xFF || *t++ != ':') {
        X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_INVALID_SAFI);
        X509V3_conf_err(val);
        goto err;
      }
      t += strspn(t, " \t");
      s = BUF_strdup(t);
    } else {
      s = BUF_strdup(val->value);
    }
    if (s == NULL) {
      X509V3err(X509V3_F_V2I_IPADDRBLOCKS, ERR_R_MALLOC_FAILURE);
      goto err;
    }

    /*
     * Check for inheritance.  Not worth additional complexity to
     * optimize this (seldom-used) case.
     */
    if (!strcmp(s, "inherit")) {
      if (!v3_addr_add_inherit(addr, afi, safi)) {
        X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_INVALID_INHERITANCE);
        X509V3_conf_err(val);
        goto err;
      }
      OPENSSL_free(s);
      s = NULL;
      continue;
    }

    i1 = strspn(s, addr_chars);
    i2 = i1 + strspn(s + i1, " \t");
    delim = s[i2++];
    s[i1] = '\0';

    if (a2i_ipadd(min, s) != length) {
      X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_INVALID_IPADDRESS);
      X509V3_conf_err(val);
      goto err;
    }

    switch (delim) {
    case '/':
      prefixlen = (int) strtoul(s + i2, &t, 10);
      if (t == s + i2 || *t != '\0') {
        X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_EXTENSION_VALUE_ERROR);
        X509V3_conf_err(val);
        goto err;
      }
      if (!v3_addr_add_prefix(addr, afi, safi, min, prefixlen)) {
        X509V3err(X509V3_F_V2I_IPADDRBLOCKS, ERR_R_MALLOC_FAILURE);
        goto err;
      }
      break;
    case '-':
      i1 = i2 + strspn(s + i2, " \t");
      i2 = i1 + strspn(s + i1, addr_chars);
      if (i1 == i2 || s[i2] != '\0') {
        X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_EXTENSION_VALUE_ERROR);
        X509V3_conf_err(val);
        goto err;
```

```
1052       }
1053       if (a2i_ipadd(max, s + i1) != length) {
1054         X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_INVALID_IPADDRESS);
1055         X509V3_conf_err(val);
1056         goto err;
1057       }
1058       if (memcmp(min, max, length_from_afi(afi)) > 0) {
1059         X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_EXTENSION_VALUE_ERROR);
1060         X509V3_conf_err(val);
1061         goto err;
1062       }
1063       if (!v3_addr_add_range(addr, afi, safi, min, max)) {
1064         X509V3err(X509V3_F_V2I_IPADDRBLOCKS, ERR_R_MALLOC_FAILURE);
1065         goto err;
1066       }
1067       break;
1068     case '\0':
1069       if (!v3_addr_add_prefix(addr, afi, safi, min, length * 8)) {
1070         X509V3err(X509V3_F_V2I_IPADDRBLOCKS, ERR_R_MALLOC_FAILURE);
1071         goto err;
1072       }
1073       break;
1074     default:
1075       X509V3err(X509V3_F_V2I_IPADDRBLOCKS, X509V3_R_EXTENSION_VALUE_ERROR);
1076       X509V3_conf_err(val);
1077       goto err;
1078     }

1080     OPENSSL_free(s);
1081     s = NULL;
1082   }

1084   /*
1085    * Canonize the result, then we're done.
1086    */
1087   if (!v3_addr_canonize(addr))
1088     goto err;
1089   return addr;

1091  err:
1092   OPENSSL_free(s);
1093   sk_IPAddressFamily_pop_free(addr, IPAddressFamily_free);
1094   return NULL;
1095 }

1097 /*
1098  * OpenSSL dispatch
1099  */
1100 const X509V3_EXT_METHOD v3_addr = {
1101   NID_sbgp_ipAddrBlock,         /* nid */
1102   0,                            /* flags */
1103   ASN1_ITEM_ref(IPAddrBlocks),  /* template */
1104   0, 0, 0, 0,                   /* old functions, ignored */
1105   0,                            /* i2s */
1106   0,                            /* s2i */
1107   0,                            /* i2v */
1108   v2i_IPAddrBlocks,             /* v2i */
1109   i2r_IPAddrBlocks,             /* i2r */
1110   0,                            /* r2i */
1111   NULL                          /* extension-specific data */
1112 };

1114 /*
1115  * Figure out whether extension sues inheritance.
1116  */
1117 int v3_addr_inherits(IPAddrBlocks *addr)
```



```
1118 {
1119   int i;
1120   if (addr == NULL)
1121     return 0;
1122   for (i = 0; i < sk_IPAddressFamily_num(addr); i++) {
1123     IPAddressFamily *f = sk_IPAddressFamily_value(addr, i);
1124     if (f->ipAddressChoice->type == IPAddressChoice_inherit)
1125       return 1;
1126   }
1127   return 0;
1128 }

1130 /*
1131  * Figure out whether parent contains child.
1132  */
1133 static int addr_contains(IPAddressOrRanges *parent,
1134                          IPAddressOrRanges *child,
1135                          int length)
1136 {
1137   unsigned char p_min[ADDR_RAW_BUF_LEN], p_max[ADDR_RAW_BUF_LEN];
1138   unsigned char c_min[ADDR_RAW_BUF_LEN], c_max[ADDR_RAW_BUF_LEN];
1139   int p, c;

1141   if (child == NULL || parent == child)
1142     return 1;
1143   if (parent == NULL)
1144     return 0;

1146   p = 0;
1147   for (c = 0; c < sk_IPAddressOrRange_num(child); c++) {
1148     if (!extract_min_max(sk_IPAddressOrRange_value(child, c),
1149                          c_min, c_max, length))
1150       return -1;
1151     for (;; p++) {
1152       if (p >= sk_IPAddressOrRange_num(parent))
1153         return 0;
1154       if (!extract_min_max(sk_IPAddressOrRange_value(parent, p),
1155                            p_min, p_max, length))
1156         return 0;
1157       if (memcmp(p_max, c_max, length) < 0)
1158         continue;
1159       if (memcmp(p_min, c_min, length) > 0)
1160         return 0;
1161       break;
1162     }
1163   }

1165   return 1;
1166 }

1168 /*
1169  * Test whether a is a subset of b.
1170  */
1171 int v3_addr_subset(IPAddrBlocks *a, IPAddrBlocks *b)
1172 {
1173   int i;
1174   if (a == NULL || a == b)
1175     return 1;
1176   if (b == NULL || v3_addr_inherits(a) || v3_addr_inherits(b))
1177     return 0;
1178   (void) sk_IPAddressFamily_set_cmp_func(b, IPAddressFamily_cmp);
1179   for (i = 0; i < sk_IPAddressFamily_num(a); i++) {
1180     IPAddressFamily *fa = sk_IPAddressFamily_value(a, i);
1181     int j = sk_IPAddressFamily_find(b, fa);
1182     IPAddressFamily *fb;
1183     fb = sk_IPAddressFamily_value(b, j);
```

```
1184     if (fb == NULL)
1185        return 0;
1186     if (!addr_contains(fb->ipAddressChoice->u.addressesOrRanges,
1187                        fa->ipAddressChoice->u.addressesOrRanges,
1188                        length_from_afi(v3_addr_get_afi(fb))))
1189       return 0;
1190   }
1191   return 1;
1192 }

1194 /*
1195  * Validation error handling via callback.
1196  */
1197 #define validation_err(_err_)           \
1198   do {                                  \
1199     if (ctx != NULL) {                  \
1200       ctx->error = _err_;               \
1201       ctx->error_depth = i;             \
1202       ctx->current_cert = x;            \
1203       ret = ctx->verify_cb(0, ctx);     \
1204     } else {                            \
1205       ret = 0;                          \
1206     }                                   \
1207     if (!ret)                           \
1208       goto done;                        \
1209   } while (0)

1211 /*
1212  * Core code for RFC 3779 2.3 path validation.
1213  */
1214 static int v3_addr_validate_path_internal(X509_STORE_CTX *ctx,
1215                                           STACK_OF(X509) *chain,
1216                                           IPAddrBlocks *ext)
1217 {
1218   IPAddrBlocks *child = NULL;
1219   int i, j, ret = 1;
1220   X509 *x;

1222   OPENSSL_assert(chain != NULL && sk_X509_num(chain) > 0);
1223   OPENSSL_assert(ctx != NULL || ext != NULL);
1224   OPENSSL_assert(ctx == NULL || ctx->verify_cb != NULL);

1226   /*
1227    * Figure out where to start.  If we don't have an extension to
1228    * check, we're done.  Otherwise, check canonical form and
1229    * set up for walking up the chain.
1230    */
1231   if (ext != NULL) {
1232     i = -1;
1233     x = NULL;
1234   } else {
1235     i = 0;
1236     x = sk_X509_value(chain, i);
1237     OPENSSL_assert(x != NULL);
1238     if ((ext = x->rfc3779_addr) == NULL)
1239       goto done;
1240   }
1241   if (!v3_addr_is_canonical(ext))
1242     validation_err(X509_V_ERR_INVALID_EXTENSION);
1243   (void) sk_IPAddressFamily_set_cmp_func(ext, IPAddressFamily_cmp);
1244   if ((child = sk_IPAddressFamily_dup(ext)) == NULL) {
1245     X509V3err(X509V3_F_V3_ADDR_VALIDATE_PATH_INTERNAL, ERR_R_MALLOC_FAILURE);
1246     ret = 0;
1247     goto done;
1248   }
```



```
1250   /*
1251    * Now walk up the chain.  No cert may list resources that its
1252    * parent doesn't list.
1253    */
1254   for (i++; i < sk_X509_num(chain); i++) {
1255     x = sk_X509_value(chain, i);
1256     OPENSSL_assert(x != NULL);
1257     if (!v3_addr_is_canonical(x->rfc3779_addr))
1258       validation_err(X509_V_ERR_INVALID_EXTENSION);
1259     if (x->rfc3779_addr == NULL) {
1260       for (j = 0; j < sk_IPAddressFamily_num(child); j++) {
1261         IPAddressFamily *fc = sk_IPAddressFamily_value(child, j);
1262         if (fc->ipAddressChoice->type != IPAddressChoice_inherit) {
1263           validation_err(X509_V_ERR_UNNESTED_RESOURCE);
1264           break;
1265         }
1266       }
1267       continue;
1268     }
1269     (void) sk_IPAddressFamily_set_cmp_func(x->rfc3779_addr, IPAddressFamily_cmp)
1270     for (j = 0; j < sk_IPAddressFamily_num(child); j++) {
1271       IPAddressFamily *fc = sk_IPAddressFamily_value(child, j);
1272       int k = sk_IPAddressFamily_find(x->rfc3779_addr, fc);
1273       IPAddressFamily *fp = sk_IPAddressFamily_value(x->rfc3779_addr, k);
1274       if (fp == NULL) {
1275         if (fc->ipAddressChoice->type == IPAddressChoice_addressesOrRanges) {
1276           validation_err(X509_V_ERR_UNNESTED_RESOURCE);
1277           break;
1278         }
1279         continue;
1280       }
1281       if (fp->ipAddressChoice->type == IPAddressChoice_addressesOrRanges) {
1282         if (fc->ipAddressChoice->type == IPAddressChoice_inherit ||
1283             addr_contains(fp->ipAddressChoice->u.addressesOrRanges,
1284                           fc->ipAddressChoice->u.addressesOrRanges,
1285                           length_from_afi(v3_addr_get_afi(fc))))
1286           sk_IPAddressFamily_set(child, j, fp);
1287         else
1288           validation_err(X509_V_ERR_UNNESTED_RESOURCE);
1289       }
1290     }
1291   }

1293   /*
1294    * Trust anchor can't inherit.
1295    */
1296   OPENSSL_assert(x != NULL);
1297   if (x->rfc3779_addr != NULL) {
1298     for (j = 0; j < sk_IPAddressFamily_num(x->rfc3779_addr); j++) {
1299       IPAddressFamily *fp = sk_IPAddressFamily_value(x->rfc3779_addr, j);
1300       if (fp->ipAddressChoice->type == IPAddressChoice_inherit &&
1301           sk_IPAddressFamily_find(child, fp) >= 0)
1302         validation_err(X509_V_ERR_UNNESTED_RESOURCE);
1303     }
1304   }

1306  done:
1307   sk_IPAddressFamily_free(child);
1308   return ret;
1309 }

1311 #undef validation_err

1313 /*
1314  * RFC 3779 2.3 path validation -- called from X509_verify_cert().
1315  */
```

```
1316 int v3_addr_validate_path(X509_STORE_CTX *ctx)
1317 {
1318   return v3_addr_validate_path_internal(ctx, ctx->chain, NULL);
1319 }

1321 /*
1322  * RFC 3779 2.3 path validation of an extension.
1323  * Test whether chain covers extension.
1324  */
1325 int v3_addr_validate_resource_set(STACK_OF(X509) *chain,
1326                                   IPAddrBlocks *ext,
1327                                   int allow_inheritance)
1328 {
1329   if (ext == NULL)
1330     return 1;
1331   if (chain == NULL || sk_X509_num(chain) == 0)
1332     return 0;
1333   if (!allow_inheritance && v3_addr_inherits(ext))
1334     return 0;
1335   return v3_addr_validate_path_internal(NULL, chain, ext);
1336 }

1338 #endif /* OPENSSL_NO_RFC3779 */
1339 #endif /* ! codereview */
```

```
**********************************************************
    6529 Wed Aug 13 19:53:28 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_akey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_akey.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

static STACK_OF(CONF_VALUE) *i2v_AUTHORITY_KEYID(X509V3_EXT_METHOD *method,
                        AUTHORITY_KEYID *akeyid, STACK_OF(CONF_VALUE) *extlist);
static AUTHORITY_KEYID *v2i_AUTHORITY_KEYID(X509V3_EXT_METHOD *method,
                        X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *values);

const X509V3_EXT_METHOD v3_akey_id =
        {
        NID_authority_key_identifier,
        X509V3_EXT_MULTILINE, ASN1_ITEM_ref(AUTHORITY_KEYID),
        0,0,0,0,
        0,0,
        (X509V3_EXT_I2V)i2v_AUTHORITY_KEYID,
        (X509V3_EXT_V2I)v2i_AUTHORITY_KEYID,
        0,0,
        NULL
        };

static STACK_OF(CONF_VALUE) *i2v_AUTHORITY_KEYID(X509V3_EXT_METHOD *method,
             AUTHORITY_KEYID *akeyid, STACK_OF(CONF_VALUE) *extlist)
{
        char *tmp;
        if(akeyid->keyid) {
                tmp = hex_to_string(akeyid->keyid->data, akeyid->keyid->length);
                X509V3_add_value("keyid", tmp, &extlist);
                OPENSSL_free(tmp);
        }
        if(akeyid->issuer)
                extlist = i2v_GENERAL_NAMES(NULL, akeyid->issuer, extlist);
        if(akeyid->serial) {
                tmp = hex_to_string(akeyid->serial->data,
                                                    akeyid->serial->length);
                X509V3_add_value("serial", tmp, &extlist);
                OPENSSL_free(tmp);
        }
        return extlist;
}

/* Currently two options:
 * keyid: use the issuers subject keyid, the value 'always' means its is
 * an error if the issuer certificate doesn't have a key id.
 * issuer: use the issuers cert issuer and serial number. The default is
 * to only use this if keyid is not present. With the option 'always'
 * this is always included.
 */

static AUTHORITY_KEYID *v2i_AUTHORITY_KEYID(X509V3_EXT_METHOD *method,
             X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *values)
        {
        char keyid=0, issuer=0;
        int i;
        CONF_VALUE *cnf;
        ASN1_OCTET_STRING *ikeyid = NULL;
        X509_NAME *isname = NULL;
        GENERAL_NAMES * gens = NULL;
        GENERAL_NAME *gen = NULL;
        ASN1_INTEGER *serial = NULL;
        X509_EXTENSION *ext;
        X509 *cert;
        AUTHORITY_KEYID *akeyid;

        for(i = 0; i < sk_CONF_VALUE_num(values); i++)
                {
```

```
128                 cnf = sk_CONF_VALUE_value(values, i);
129                 if(!strcmp(cnf->name, "keyid"))
130                         {
131                         keyid = 1;
132                         if(cnf->value && !strcmp(cnf->value, "always"))
133                                 keyid = 2;
134                         }
135                 else if(!strcmp(cnf->name, "issuer"))
136                         {
137                         issuer = 1;
138                         if(cnf->value && !strcmp(cnf->value, "always"))
139                                 issuer = 2;
140                         }
141                 else
142                         {
143                         X509V3err(X509V3_F_V2I_AUTHORITY_KEYID,X509V3_R_UNKNOWN_
144                         ERR_add_error_data(2, "name=", cnf->name);
145                         return NULL;
146                         }
147                 }

149         if(!ctx || !ctx->issuer_cert)
150                 {
151                 if(ctx && (ctx->flags==CTX_TEST))
152                         return AUTHORITY_KEYID_new();
153                 X509V3err(X509V3_F_V2I_AUTHORITY_KEYID,X509V3_R_NO_ISSUER_CERTIF
154                 return NULL;
155                 }

157         cert = ctx->issuer_cert;

159         if(keyid)
160                 {
161                 i = X509_get_ext_by_NID(cert, NID_subject_key_identifier, -1);
162                 if((i >= 0)  && (ext = X509_get_ext(cert, i)))
163                         ikeyid = X509V3_EXT_d2i(ext);
164                 if(keyid==2 && !ikeyid)
165                         {
166                         X509V3err(X509V3_F_V2I_AUTHORITY_KEYID,X509V3_R_UNABLE_T
167                         return NULL;
168                         }
169                 }

171         if((issuer && !ikeyid) || (issuer == 2))
172                 {
173                 isname = X509_NAME_dup(X509_get_issuer_name(cert));
174                 serial = M_ASN1_INTEGER_dup(X509_get_serialNumber(cert));
175                 if(!isname || !serial)
176                         {
177                         X509V3err(X509V3_F_V2I_AUTHORITY_KEYID,X509V3_R_UNABLE_T
178                         goto err;
179                         }
180                 }

182         if(!(akeyid = AUTHORITY_KEYID_new())) goto err;

184         if(isname)
185                 {
186                 if(!(gens = sk_GENERAL_NAME_new_null())
187                         || !(gen = GENERAL_NAME_new())
188                         || !sk_GENERAL_NAME_push(gens, gen))
189                         {
190                         X509V3err(X509V3_F_V2I_AUTHORITY_KEYID,ERR_R_MALLOC_FAIL
191                         goto err;
192                         }
193                 gen->type = GEN_DIRNAME;
```



```
194                 gen->d.dirn = isname;
195                 }

197         akeyid->issuer = gens;
198         akeyid->serial = serial;
199         akeyid->keyid = ikeyid;

201         return akeyid;

203  err:
204         X509_NAME_free(isname);
205         M_ASN1_INTEGER_free(serial);
206         M_ASN1_OCTET_STRING_free(ikeyid);
207         return NULL;
208         }
209 #endif /* ! codereview */
```

```
**********************************************************
    3188 Wed Aug 13 19:53:28 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_akeya.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_akey_asn1.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

ASN1_SEQUENCE(AUTHORITY_KEYID) = {
        ASN1_IMP_OPT(AUTHORITY_KEYID, keyid, ASN1_OCTET_STRING, 0),
        ASN1_IMP_SEQUENCE_OF_OPT(AUTHORITY_KEYID, issuer, GENERAL_NAME, 1),
        ASN1_IMP_OPT(AUTHORITY_KEYID, serial, ASN1_INTEGER, 2)
} ASN1_SEQUENCE_END(AUTHORITY_KEYID)

IMPLEMENT_ASN1_FUNCTIONS(AUTHORITY_KEYID)
#endif /* ! codereview */
```

```
**********************************************************
   15689 Wed Aug 13 19:53:29 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_alt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_alt.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 1999-2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/x509v3.h>

static GENERAL_NAMES *v2i_subject_alt(X509V3_EXT_METHOD *method, X509V3_CTX *ctx
static GENERAL_NAMES *v2i_issuer_alt(X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
static int copy_email(X509V3_CTX *ctx, GENERAL_NAMES *gens, int move_p);
static int copy_issuer(X509V3_CTX *ctx, GENERAL_NAMES *gens);
static int do_othername(GENERAL_NAME *gen, char *value, X509V3_CTX *ctx);
static int do_dirname(GENERAL_NAME *gen, char *value, X509V3_CTX *ctx);

const X509V3_EXT_METHOD v3_alt[] = {
{ NID_subject_alt_name, 0, ASN1_ITEM_ref(GENERAL_NAMES),
0,0,0,0,
0,0,
(X509V3_EXT_I2V)i2v_GENERAL_NAMES,
(X509V3_EXT_V2I)v2i_subject_alt,
NULL, NULL, NULL},

{ NID_issuer_alt_name, 0, ASN1_ITEM_ref(GENERAL_NAMES),
0,0,0,0,
0,0,
(X509V3_EXT_I2V)i2v_GENERAL_NAMES,
(X509V3_EXT_V2I)v2i_issuer_alt,
NULL, NULL, NULL},

{ NID_certificate_issuer, 0, ASN1_ITEM_ref(GENERAL_NAMES),
0,0,0,0,
0,0,
(X509V3_EXT_I2V)i2v_GENERAL_NAMES,
NULL, NULL, NULL, NULL},
};

STACK_OF(CONF_VALUE) *i2v_GENERAL_NAMES(X509V3_EXT_METHOD *method,
                GENERAL_NAMES *gens, STACK_OF(CONF_VALUE) *ret)
{
        int i;
        GENERAL_NAME *gen;
        for(i = 0; i < sk_GENERAL_NAME_num(gens); i++) {
                gen = sk_GENERAL_NAME_value(gens, i);
                ret = i2v_GENERAL_NAME(method, gen, ret);
        }
        if(!ret) return sk_CONF_VALUE_new_null();
        return ret;
}

STACK_OF(CONF_VALUE) *i2v_GENERAL_NAME(X509V3_EXT_METHOD *method,
                                GENERAL_NAME *gen, STACK_OF(CONF_VALUE) *ret)
{
        unsigned char *p;
        char oline[256], htmp[5];
        int i;
        switch (gen->type)
        {
                case GEN_OTHERNAME:
                X509V3_add_value("othername","<unsupported>", &ret);
                break;

                case GEN_X400:
                X509V3_add_value("X400Name","<unsupported>", &ret);
                break;

                case GEN_EDIPARTY:
                X509V3_add_value("EdiPartyName","<unsupported>", &ret);
                break;

                case GEN_EMAIL:
                X509V3_add_value_uchar("email",gen->d.ia5->data, &ret);
```

```
128                 break;
129
130                 case GEN_DNS:
131                 X509V3_add_value_uchar("DNS",gen->d.ia5->data, &ret);
132                 break;
133
134                 case GEN_URI:
135                 X509V3_add_value_uchar("URI",gen->d.ia5->data, &ret);
136                 break;
137
138                 case GEN_DIRNAME:
139                 X509_NAME_oneline(gen->d.dirn, oline, 256);
140                 X509V3_add_value("DirName",oline, &ret);
141                 break;
142
143                 case GEN_IPADD:
144                 p = gen->d.ip->data;
145                 if(gen->d.ip->length == 4)
146                         BIO_snprintf(oline, sizeof oline,
147                                         "%d.%d.%d.%d", p[0], p[1], p[2], p[3]);
148                 else if(gen->d.ip->length == 16)
149                         {
150                         oline[0] = 0;
151                         for (i = 0; i < 8; i++)
152                                 {
153                                 BIO_snprintf(htmp, sizeof htmp,
154                                                 "%X", p[0] << 8 | p[1]);
155                                 p += 2;
156                                 strcat(oline, htmp);
157                                 if (i != 7)
158                                         strcat(oline, ":");
159                                 }
160                         }
161                 else
162                         {
163                         X509V3_add_value("IP Address","<invalid>", &ret);
164                         break;
165                         }
166                 X509V3_add_value("IP Address",oline, &ret);
167                 break;
168
169                 case GEN_RID:
170                 i2t_ASN1_OBJECT(oline, 256, gen->d.rid);
171                 X509V3_add_value("Registered ID",oline, &ret);
172                 break;
173         }
174         return ret;
175 }
176
177 int GENERAL_NAME_print(BIO *out, GENERAL_NAME *gen)
178 {
179         unsigned char *p;
180         int i;
181         switch (gen->type)
182         {
183                 case GEN_OTHERNAME:
184                 BIO_printf(out, "othername:<unsupported>");
185                 break;
186
187                 case GEN_X400:
188                 BIO_printf(out, "X400Name:<unsupported>");
189                 break;
190
191                 case GEN_EDIPARTY:
192                 /* Maybe fix this: it is supported now */
193                 BIO_printf(out, "EdiPartyName:<unsupported>");
```



```
194                 break;
195
196                 case GEN_EMAIL:
197                 BIO_printf(out, "email:%s",gen->d.ia5->data);
198                 break;
199
200                 case GEN_DNS:
201                 BIO_printf(out, "DNS:%s",gen->d.ia5->data);
202                 break;
203
204                 case GEN_URI:
205                 BIO_printf(out, "URI:%s",gen->d.ia5->data);
206                 break;
207
208                 case GEN_DIRNAME:
209                 BIO_printf(out, "DirName: ");
210                 X509_NAME_print_ex(out, gen->d.dirn, 0, XN_FLAG_ONELINE);
211                 break;
212
213                 case GEN_IPADD:
214                 p = gen->d.ip->data;
215                 if(gen->d.ip->length == 4)
216                         BIO_printf(out, "IP Address:%d.%d.%d.%d",
217                                                 p[0], p[1], p[2], p[3]);
218                 else if(gen->d.ip->length == 16)
219                         {
220                         BIO_printf(out, "IP Address");
221                         for (i = 0; i < 8; i++)
222                                 {
223                                 BIO_printf(out, ":%X", p[0] << 8 | p[1]);
224                                 p += 2;
225                                 }
226                         BIO_puts(out, "\n");
227                         }
228                 else
229                         {
230                         BIO_printf(out,"IP Address:<invalid>");
231                         break;
232                         }
233                 break;
234
235                 case GEN_RID:
236                 BIO_printf(out, "Registered ID");
237                 i2a_ASN1_OBJECT(out, gen->d.rid);
238                 break;
239         }
240         return 1;
241 }
242
243 static GENERAL_NAMES *v2i_issuer_alt(X509V3_EXT_METHOD *method,
244                                      X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
245 {
246         GENERAL_NAMES *gens = NULL;
247         CONF_VALUE *cnf;
248         int i;
249         if(!(gens = sk_GENERAL_NAME_new_null())) {
250                 X509V3err(X509V3_F_V2I_ISSUER_ALT,ERR_R_MALLOC_FAILURE);
251                 return NULL;
252         }
253         for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
254                 cnf = sk_CONF_VALUE_value(nval, i);
255                 if(!name_cmp(cnf->name, "issuer") && cnf->value &&
256                                                 !strcmp(cnf->value, "copy")) {
257                         if(!copy_issuer(ctx, gens)) goto err;
258                 } else {
259                         GENERAL_NAME *gen;
```

```
260                         if(!(gen = v2i_GENERAL_NAME(method, ctx, cnf)))
261                                                                 goto err;
262                         sk_GENERAL_NAME_push(gens, gen);
263                 }
264         }
265         return gens;
266         err:
267         sk_GENERAL_NAME_pop_free(gens, GENERAL_NAME_free);
268         return NULL;
269 }

271 /* Append subject altname of issuer to issuer alt name of subject */

273 static int copy_issuer(X509V3_CTX *ctx, GENERAL_NAMES *gens)
274 {
275         GENERAL_NAMES *ialt;
276         GENERAL_NAME *gen;
277         X509_EXTENSION *ext;
278         int i;
279         if(ctx && (ctx->flags == CTX_TEST)) return 1;
280         if(!ctx || !ctx->issuer_cert) {
281                 X509V3err(X509V3_F_COPY_ISSUER,X509V3_R_NO_ISSUER_DETAILS);
282                 goto err;
283         }
284         i = X509_get_ext_by_NID(ctx->issuer_cert, NID_subject_alt_name, -1);
285         if(i < 0) return 1;
286         if(!(ext = X509_get_ext(ctx->issuer_cert, i)) ||
287                         !(ialt = X509V3_EXT_d2i(ext)) ) {
288                 X509V3err(X509V3_F_COPY_ISSUER,X509V3_R_ISSUER_DECODE_ERROR);
289                 goto err;
290         }

292         for(i = 0; i < sk_GENERAL_NAME_num(ialt); i++) {
293                 gen = sk_GENERAL_NAME_value(ialt, i);
294                 if(!sk_GENERAL_NAME_push(gens, gen)) {
295                         X509V3err(X509V3_F_COPY_ISSUER,ERR_R_MALLOC_FAILURE);
296                         goto err;
297                 }
298         }
299         sk_GENERAL_NAME_free(ialt);

301         return 1;

303         err:
304         return 0;

306 }

308 static GENERAL_NAMES *v2i_subject_alt(X509V3_EXT_METHOD *method,
309                                       X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
310 {
311         GENERAL_NAMES *gens = NULL;
312         CONF_VALUE *cnf;
313         int i;
314         if(!(gens = sk_GENERAL_NAME_new_null())) {
315                 X509V3err(X509V3_F_V2I_SUBJECT_ALT,ERR_R_MALLOC_FAILURE);
316                 return NULL;
317         }
318         for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
319                 cnf = sk_CONF_VALUE_value(nval, i);
320                 if(!name_cmp(cnf->name, "email") && cnf->value &&
321                                                 !strcmp(cnf->value, "copy")) {
322                         if(!copy_email(ctx, gens, 0)) goto err;
323                 } else if(!name_cmp(cnf->name, "email") && cnf->value &&
324                                                 !strcmp(cnf->value, "move")) {
325                         if(!copy_email(ctx, gens, 1)) goto err;
```



```
326                 } else {
327                         GENERAL_NAME *gen;
328                         if(!(gen = v2i_GENERAL_NAME(method, ctx, cnf)))
329                                                                 goto err;
330                         sk_GENERAL_NAME_push(gens, gen);
331                 }
332         }
333         return gens;
334         err:
335         sk_GENERAL_NAME_pop_free(gens, GENERAL_NAME_free);
336         return NULL;
337 }

339 /* Copy any email addresses in a certificate or request to
340  * GENERAL_NAMES
341  */

343 static int copy_email(X509V3_CTX *ctx, GENERAL_NAMES *gens, int move_p)
344 {
345         X509_NAME *nm;
346         ASN1_IA5STRING *email = NULL;
347         X509_NAME_ENTRY *ne;
348         GENERAL_NAME *gen = NULL;
349         int i;
350         if(ctx != NULL && ctx->flags == CTX_TEST)
351                 return 1;
352         if(!ctx || (!ctx->subject_cert && !ctx->subject_req)) {
353                 X509V3err(X509V3_F_COPY_EMAIL,X509V3_R_NO_SUBJECT_DETAILS);
354                 goto err;
355         }
356         /* Find the subject name */
357         if(ctx->subject_cert) nm = X509_get_subject_name(ctx->subject_cert);
358         else nm = X509_REQ_get_subject_name(ctx->subject_req);

360         /* Now add any email address(es) to STACK */
361         i = -1;
362         while((i = X509_NAME_get_index_by_NID(nm,
363                                         NID_pkcs9_emailAddress, i)) >= 0) {
364                 ne = X509_NAME_get_entry(nm, i);
365                 email = M_ASN1_IA5STRING_dup(X509_NAME_ENTRY_get_data(ne));
366                 if (move_p)
367                         {
368                         X509_NAME_delete_entry(nm, i);
369                         X509_NAME_ENTRY_free(ne);
370                         i--;
371                         }
372                 if(!email || !(gen = GENERAL_NAME_new())) {
373                         X509V3err(X509V3_F_COPY_EMAIL,ERR_R_MALLOC_FAILURE);
374                         goto err;
375                 }
376                 gen->d.ia5 = email;
377                 email = NULL;
378                 gen->type = GEN_EMAIL;
379                 if(!sk_GENERAL_NAME_push(gens, gen)) {
380                         X509V3err(X509V3_F_COPY_EMAIL,ERR_R_MALLOC_FAILURE);
381                         goto err;
382                 }
383                 gen = NULL;
384         }


387         return 1;

389         err:
390         GENERAL_NAME_free(gen);
391         M_ASN1_IA5STRING_free(email);
```

```
392         return 0;

394 }

396 GENERAL_NAMES *v2i_GENERAL_NAMES(const X509V3_EXT_METHOD *method,
397                                  X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
398 {
399         GENERAL_NAME *gen;
400         GENERAL_NAMES *gens = NULL;
401         CONF_VALUE *cnf;
402         int i;
403         if(!(gens = sk_GENERAL_NAME_new_null())) {
404                 X509V3err(X509V3_F_V2I_GENERAL_NAMES,ERR_R_MALLOC_FAILURE);
405                 return NULL;
406         }
407         for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
408                 cnf = sk_CONF_VALUE_value(nval, i);
409                 if(!(gen = v2i_GENERAL_NAME(method, ctx, cnf))) goto err;
410                 sk_GENERAL_NAME_push(gens, gen);
411         }
412         return gens;
413         err:
414         sk_GENERAL_NAME_pop_free(gens, GENERAL_NAME_free);
415         return NULL;
416 }

418 GENERAL_NAME *v2i_GENERAL_NAME(const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
419                                CONF_VALUE *cnf)
420         {
421         return v2i_GENERAL_NAME_ex(NULL, method, ctx, cnf, 0);
422         }

424 GENERAL_NAME *a2i_GENERAL_NAME(GENERAL_NAME *out,
425                                const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
426                                int gen_type, char *value, int is_nc)
427         {
428         char is_string = 0;
429         GENERAL_NAME *gen = NULL;

431         if(!value)
432                 {
433                 X509V3err(X509V3_F_A2I_GENERAL_NAME,X509V3_R_MISSING_VALUE);
434                 return NULL;
435                 }

437         if (out)
438                 gen = out;
439         else
440                 {
441                 gen = GENERAL_NAME_new();
442                 if(gen == NULL)
443                         {
444                         X509V3err(X509V3_F_A2I_GENERAL_NAME,ERR_R_MALLOC_FAILURE
445                         return NULL;
446                         }
447                 }

449         switch (gen_type)
450                 {
451                 case GEN_URI:
452                 case GEN_EMAIL:
453                 case GEN_DNS:
454                 is_string = 1;
455                 break;

457                 case GEN_RID:
```



```
458                 {
459                 ASN1_OBJECT *obj;
460                 if(!(obj = OBJ_txt2obj(value,0)))
461                         {
462                         X509V3err(X509V3_F_A2I_GENERAL_NAME,X509V3_R_BAD_OBJECT)
463                         ERR_add_error_data(2, "value=", value);
464                         goto err;
465                         }
466                 gen->d.rid = obj;
467                 }
468                 break;

470                 case GEN_IPADD:
471                 if (is_nc)
472                         gen->d.ip = a2i_IPADDRESS_NC(value);
473                 else
474                         gen->d.ip = a2i_IPADDRESS(value);
475                 if(gen->d.ip == NULL)
476                         {
477                         X509V3err(X509V3_F_A2I_GENERAL_NAME,X509V3_R_BAD_IP_ADDR
478                         ERR_add_error_data(2, "value=", value);
479                         goto err;
480                         }
481                 break;

483                 case GEN_DIRNAME:
484                 if (!do_dirname(gen, value, ctx))
485                         {
486                         X509V3err(X509V3_F_A2I_GENERAL_NAME,X509V3_R_DIRNAME_ERR
487                         goto err;
488                         }
489                 break;

491                 case GEN_OTHERNAME:
492                 if (!do_othername(gen, value, ctx))
493                         {
494                         X509V3err(X509V3_F_A2I_GENERAL_NAME,X509V3_R_OTHERNAME_E
495                         goto err;
496                         }
497                 break;
498                 default:
499                 X509V3err(X509V3_F_A2I_GENERAL_NAME,X509V3_R_UNSUPPORTED_TYPE);
500                 goto err;
501                 }

503         if(is_string)
504                 {
505                 if(!(gen->d.ia5 = M_ASN1_IA5STRING_new()) ||
506                                 !ASN1_STRING_set(gen->d.ia5, (unsigned char*)value
507                                                 strlen(value)))
508                         {
509                         X509V3err(X509V3_F_A2I_GENERAL_NAME,ERR_R_MALLOC_FAILURE
510                         goto err;
511                         }
512                 }

514         gen->type = gen_type;

516         return gen;

518         err:
519         if (!out)
520                 GENERAL_NAME_free(gen);
521         return NULL;
522         }
```

```
GENERAL_NAME *v2i_GENERAL_NAME_ex(GENERAL_NAME *out,
                                  const X509V3_EXT_METHOD *method,
                                  X509V3_CTX *ctx, CONF_VALUE *cnf, int is_nc)
        {
        int type;

        char *name, *value;

        name = cnf->name;
        value = cnf->value;

        if(!value)
                {
                X509V3err(X509V3_F_V2I_GENERAL_NAME_EX,X509V3_R_MISSING_VALUE);
                return NULL;
                }

        if(!name_cmp(name, "email"))
                type = GEN_EMAIL;
        else if(!name_cmp(name, "URI"))
                type = GEN_URI;
        else if(!name_cmp(name, "DNS"))
                type = GEN_DNS;
        else if(!name_cmp(name, "RID"))
                type = GEN_RID;
        else if(!name_cmp(name, "IP"))
                type = GEN_IPADD;
        else if(!name_cmp(name, "dirName"))
                type = GEN_DIRNAME;
        else if(!name_cmp(name, "otherName"))
                type = GEN_OTHERNAME;
        else
                {
                X509V3err(X509V3_F_V2I_GENERAL_NAME_EX,X509V3_R_UNSUPPORTED_OPTI
                ERR_add_error_data(2, "name=", name);
                return NULL;
                }

        return a2i_GENERAL_NAME(out, method, ctx, type, value, is_nc);

        }

static int do_othername(GENERAL_NAME *gen, char *value, X509V3_CTX *ctx)
        {
        char *objtmp = NULL, *p;
        int objlen;
        if (!(p = strchr(value, ';')))
                return 0;
        if (!(gen->d.otherName = OTHERNAME_new()))
                return 0;
        /* Free this up because we will overwrite it.
         * no need to free type_id because it is static
         */
        ASN1_TYPE_free(gen->d.otherName->value);
        if (!(gen->d.otherName->value = ASN1_generate_v3(p + 1, ctx)))
                return 0;
        objlen = p - value;
        objtmp = OPENSSL_malloc(objlen + 1);
        strncpy(objtmp, value, objlen);
        objtmp[objlen] = 0;
        gen->d.otherName->type_id = OBJ_txt2obj(objtmp, 0);
        OPENSSL_free(objtmp);
        if (!gen->d.otherName->type_id)
                return 0;
        return 1;
        }
```



```
static int do_dirname(GENERAL_NAME *gen, char *value, X509V3_CTX *ctx)
        {
        int ret;
        STACK_OF(CONF_VALUE) *sk;
        X509_NAME *nm;
        if (!(nm = X509_NAME_new()))
                return 0;
        sk = X509V3_get_section(ctx, value);
        if (!sk)
                {
                X509V3err(X509V3_F_DO_DIRNAME,X509V3_R_SECTION_NOT_FOUND);
                ERR_add_error_data(2, "section=", value);
                X509_NAME_free(nm);
                return 0;
                }
        /* FIXME: should allow other character types... */
        ret = X509V3_NAME_from_section(nm, sk, MBSTRING_ASC);
        if (!ret)
                X509_NAME_free(nm);
        gen->d.dirn = nm;
        X509V3_section_free(ctx, sk);

        return ret;
        }
#endif /* ! codereview */
```

```
**********************************************************
   24150 Wed Aug 13 19:53:29 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_asid.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*
 * Contributed to the OpenSSL Project by the American Registry for
 * Internet Numbers ("ARIN").
 */
/* ====================================================================
 * Copyright (c) 2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 */

/*
 * Implementation of RFC 3779 section 3.2.
 */
```



```
#include <stdio.h>
#include <string.h>
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>
#include <openssl/x509.h>
#include <openssl/bn.h>

#ifndef OPENSSL_NO_RFC3779

/*
 * OpenSSL ASN.1 template translation of RFC 3779 3.2.3.
 */

ASN1_SEQUENCE(ASRange) = {
  ASN1_SIMPLE(ASRange, min, ASN1_INTEGER),
  ASN1_SIMPLE(ASRange, max, ASN1_INTEGER)
} ASN1_SEQUENCE_END(ASRange)

ASN1_CHOICE(ASIdOrRange) = {
  ASN1_SIMPLE(ASIdOrRange, u.id,    ASN1_INTEGER),
  ASN1_SIMPLE(ASIdOrRange, u.range, ASRange)
} ASN1_CHOICE_END(ASIdOrRange)

ASN1_CHOICE(ASIdentifierChoice) = {
  ASN1_SIMPLE(ASIdentifierChoice,      u.inherit,       ASN1_NULL),
  ASN1_SEQUENCE_OF(ASIdentifierChoice, u.asIdsOrRanges, ASIdOrRange)
} ASN1_CHOICE_END(ASIdentifierChoice)

ASN1_SEQUENCE(ASIdentifiers) = {
  ASN1_EXP_OPT(ASIdentifiers, asnum, ASIdentifierChoice, 0),
  ASN1_EXP_OPT(ASIdentifiers, rdi,   ASIdentifierChoice, 1)
} ASN1_SEQUENCE_END(ASIdentifiers)

IMPLEMENT_ASN1_FUNCTIONS(ASRange)
IMPLEMENT_ASN1_FUNCTIONS(ASIdOrRange)
IMPLEMENT_ASN1_FUNCTIONS(ASIdentifierChoice)
IMPLEMENT_ASN1_FUNCTIONS(ASIdentifiers)

/*
 * i2r method for an ASIdentifierChoice.
 */
static int i2r_ASIdentifierChoice(BIO *out,
                                  ASIdentifierChoice *choice,
                                  int indent,
                                  const char *msg)
{
  int i;
  char *s;
  if (choice == NULL)
    return 1;
  BIO_printf(out, "%*s%s:\n", indent, "", msg);
  switch (choice->type) {
  case ASIdentifierChoice_inherit:
    BIO_printf(out, "%*sinherit\n", indent + 2, "");
    break;
  case ASIdentifierChoice_asIdsOrRanges:
    for (i = 0; i < sk_ASIdOrRange_num(choice->u.asIdsOrRanges); i++) {
      ASIdOrRange *aor = sk_ASIdOrRange_value(choice->u.asIdsOrRanges, i);
      switch (aor->type) {
      case ASIdOrRange_id:
        if ((s = i2s_ASN1_INTEGER(NULL, aor->u.id)) == NULL)
          return 0;
        BIO_printf(out, "%*s%s\n", indent + 2, "", s);
```

```
128         OPENSSL_free(s);
129         break;
130       case ASIdOrRange_range:
131         if ((s = i2s_ASN1_INTEGER(NULL, aor->u.range->min)) == NULL)
132           return 0;
133         BIO_printf(out, "%*s%s-", indent + 2, "", s);
134         OPENSSL_free(s);
135         if ((s = i2s_ASN1_INTEGER(NULL, aor->u.range->max)) == NULL)
136           return 0;
137         BIO_printf(out, "%s\n", s);
138         OPENSSL_free(s);
139         break;
140       default:
141         return 0;
142       }
143     }
144     break;
145   default:
146     return 0;
147   }
148   return 1;
149 }

151 /*
152  * i2r method for an ASIdentifier extension.
153  */
154 static int i2r_ASIdentifiers(const X509V3_EXT_METHOD *method,
155                              void *ext,
156                              BIO *out,
157                              int indent)
158 {
159   ASIdentifiers *asid = ext;
160   return (i2r_ASIdentifierChoice(out, asid->asnum, indent,
161                                  "Autonomous System Numbers") &&
162           i2r_ASIdentifierChoice(out, asid->rdi, indent,
163                                  "Routing Domain Identifiers"));
164 }

166 /*
167  * Sort comparision function for a sequence of ASIdOrRange elements.
168  */
169 static int ASIdOrRange_cmp(const ASIdOrRange * const *a_,
170                            const ASIdOrRange * const *b_)
171 {
172   const ASIdOrRange *a = *a_, *b = *b_;

174   OPENSSL_assert((a->type == ASIdOrRange_id && a->u.id != NULL) ||
175          (a->type == ASIdOrRange_range && a->u.range != NULL &&
176           a->u.range->min != NULL && a->u.range->max != NULL));

178   OPENSSL_assert((b->type == ASIdOrRange_id && b->u.id != NULL) ||
179          (b->type == ASIdOrRange_range && b->u.range != NULL &&
180           b->u.range->min != NULL && b->u.range->max != NULL));

182   if (a->type == ASIdOrRange_id && b->type == ASIdOrRange_id)
183     return ASN1_INTEGER_cmp(a->u.id, b->u.id);

185   if (a->type == ASIdOrRange_range && b->type == ASIdOrRange_range) {
186     int r = ASN1_INTEGER_cmp(a->u.range->min, b->u.range->min);
187     return r != 0 ? r : ASN1_INTEGER_cmp(a->u.range->max, b->u.range->max);
188   }

190   if (a->type == ASIdOrRange_id)
191     return ASN1_INTEGER_cmp(a->u.id, b->u.range->min);
192   else
193     return ASN1_INTEGER_cmp(a->u.range->min, b->u.id);
```



```
194 }

196 /*
197  * Add an inherit element.
198  */
199 int v3_asid_add_inherit(ASIdentifiers *asid, int which)
200 {
201   ASIdentifierChoice **choice;
202   if (asid == NULL)
203     return 0;
204   switch (which) {
205   case V3_ASID_ASNUM:
206     choice = &asid->asnum;
207     break;
208   case V3_ASID_RDI:
209     choice = &asid->rdi;
210     break;
211   default:
212     return 0;
213   }
214   if (*choice == NULL) {
215     if ((*choice = ASIdentifierChoice_new()) == NULL)
216       return 0;
217     OPENSSL_assert((*choice)->u.inherit == NULL);
218     if (((*choice)->u.inherit = ASN1_NULL_new()) == NULL)
219       return 0;
220     (*choice)->type = ASIdentifierChoice_inherit;
221   }
222   return (*choice)->type == ASIdentifierChoice_inherit;
223 }

225 /*
226  * Add an ID or range to an ASIdentifierChoice.
227  */
228 int v3_asid_add_id_or_range(ASIdentifiers *asid,
229                             int which,
230                             ASN1_INTEGER *min,
231                             ASN1_INTEGER *max)
232 {
233   ASIdentifierChoice **choice;
234   ASIdOrRange *aor;
235   if (asid == NULL)
236     return 0;
237   switch (which) {
238   case V3_ASID_ASNUM:
239     choice = &asid->asnum;
240     break;
241   case V3_ASID_RDI:
242     choice = &asid->rdi;
243     break;
244   default:
245     return 0;
246   }
247   if (*choice != NULL && (*choice)->type == ASIdentifierChoice_inherit)
248     return 0;
249   if (*choice == NULL) {
250     if ((*choice = ASIdentifierChoice_new()) == NULL)
251       return 0;
252     OPENSSL_assert((*choice)->u.asIdsOrRanges == NULL);
253     (*choice)->u.asIdsOrRanges = sk_ASIdOrRange_new(ASIdOrRange_cmp);
254     if ((*choice)->u.asIdsOrRanges == NULL)
255       return 0;
256     (*choice)->type = ASIdentifierChoice_asIdsOrRanges;
257   }
258   if ((aor = ASIdOrRange_new()) == NULL)
259     return 0;
```

```
  if (max == NULL) {
    aor->type = ASIdOrRange_id;
    aor->u.id = min;
  } else {
    aor->type = ASIdOrRange_range;
    if ((aor->u.range = ASRange_new()) == NULL)
      goto err;
    ASN1_INTEGER_free(aor->u.range->min);
    aor->u.range->min = min;
    ASN1_INTEGER_free(aor->u.range->max);
    aor->u.range->max = max;
  }
  if (!(sk_ASIdOrRange_push((*choice)->u.asIdsOrRanges, aor)))
    goto err;
  return 1;

 err:
  ASIdOrRange_free(aor);
  return 0;
}

/*
 * Extract min and max values from an ASIdOrRange.
 */
static void extract_min_max(ASIdOrRange *aor,
                            ASN1_INTEGER **min,
                            ASN1_INTEGER **max)
{
  OPENSSL_assert(aor != NULL && min != NULL && max != NULL);
  switch (aor->type) {
  case ASIdOrRange_id:
    *min = aor->u.id;
    *max = aor->u.id;
    return;
  case ASIdOrRange_range:
    *min = aor->u.range->min;
    *max = aor->u.range->max;
    return;
  }
}

/*
 * Check whether an ASIdentifierChoice is in canonical form.
 */
static int ASIdentifierChoice_is_canonical(ASIdentifierChoice *choice)
{
  ASN1_INTEGER *a_max_plus_one = NULL;
  BIGNUM *bn = NULL;
  int i, ret = 0;

  /*
   * Empty element or inheritance is canonical.
   */
  if (choice == NULL || choice->type == ASIdentifierChoice_inherit)
    return 1;

  /*
   * If not a list, or if empty list, it's broken.
   */
  if (choice->type != ASIdentifierChoice_asIdsOrRanges ||
      sk_ASIdOrRange_num(choice->u.asIdsOrRanges) == 0)
    return 0;

  /*
   * It's a list, check it.
   */
```



```
  for (i = 0; i < sk_ASIdOrRange_num(choice->u.asIdsOrRanges) - 1; i++) {
    ASIdOrRange *a = sk_ASIdOrRange_value(choice->u.asIdsOrRanges, i);
    ASIdOrRange *b = sk_ASIdOrRange_value(choice->u.asIdsOrRanges, i + 1);
    ASN1_INTEGER *a_min, *a_max, *b_min, *b_max;

    extract_min_max(a, &a_min, &a_max);
    extract_min_max(b, &b_min, &b_max);

    /*
     * Punt misordered list, overlapping start, or inverted range.
     */
    if (ASN1_INTEGER_cmp(a_min, b_min) >= 0 ||
        ASN1_INTEGER_cmp(a_min, a_max) > 0 ||
        ASN1_INTEGER_cmp(b_min, b_max) > 0)
      goto done;

    /*
     * Calculate a_max + 1 to check for adjacency.
     */
    if ((bn == NULL && (bn = BN_new()) == NULL) ||
        ASN1_INTEGER_to_BN(a_max, bn) == NULL ||
        !BN_add_word(bn, 1) ||
        (a_max_plus_one = BN_to_ASN1_INTEGER(bn, a_max_plus_one)) == NULL) {
      X509V3err(X509V3_F_ASIDENTIFIERCHOICE_IS_CANONICAL,
                ERR_R_MALLOC_FAILURE);
      goto done;
    }

    /*
     * Punt if adjacent or overlapping.
     */
    if (ASN1_INTEGER_cmp(a_max_plus_one, b_min) >= 0)
      goto done;
  }

  /*
   * Check for inverted range.
   */
  i = sk_ASIdOrRange_num(choice->u.asIdsOrRanges) - 1;
  {
    ASIdOrRange *a = sk_ASIdOrRange_value(choice->u.asIdsOrRanges, i);
    ASN1_INTEGER *a_min, *a_max;
    if (a != NULL && a->type == ASIdOrRange_range) {
      extract_min_max(a, &a_min, &a_max);
      if (ASN1_INTEGER_cmp(a_min, a_max) > 0)
        goto done;
    }
  }

  ret = 1;

 done:
  ASN1_INTEGER_free(a_max_plus_one);
  BN_free(bn);
  return ret;
}

/*
 * Check whether an ASIdentifier extension is in canonical form.
 */
int v3_asid_is_canonical(ASIdentifiers *asid)
{
  return (asid == NULL ||
          (ASIdentifierChoice_is_canonical(asid->asnum) &&
           ASIdentifierChoice_is_canonical(asid->rdi)));
}
```

```
393 /*
394  * Whack an ASIdentifierChoice into canonical form.
395  */
396 static int ASIdentifierChoice_canonize(ASIdentifierChoice *choice)
397 {
398   ASN1_INTEGER *a_max_plus_one = NULL;
399   BIGNUM *bn = NULL;
400   int i, ret = 0;

402   /*
403    * Nothing to do for empty element or inheritance.
404    */
405   if (choice == NULL || choice->type == ASIdentifierChoice_inherit)
406     return 1;

408   /*
409    * If not a list, or if empty list, it's broken.
410    */
411   if (choice->type != ASIdentifierChoice_asIdsOrRanges ||
412       sk_ASIdOrRange_num(choice->u.asIdsOrRanges) == 0) {
413     X509V3err(X509V3_F_ASIDENTIFIERCHOICE_CANONIZE,
414               X509V3_R_EXTENSION_VALUE_ERROR);
415     return 0;
416   }

418   /*
419    * We have a non-empty list.  Sort it.
420    */
421   sk_ASIdOrRange_sort(choice->u.asIdsOrRanges);

423   /*
424    * Now check for errors and suboptimal encoding, rejecting the
425    * former and fixing the latter.
426    */
427   for (i = 0; i < sk_ASIdOrRange_num(choice->u.asIdsOrRanges) - 1; i++) {
428     ASIdOrRange *a = sk_ASIdOrRange_value(choice->u.asIdsOrRanges, i);
429     ASIdOrRange *b = sk_ASIdOrRange_value(choice->u.asIdsOrRanges, i + 1);
430     ASN1_INTEGER *a_min, *a_max, *b_min, *b_max;

432     extract_min_max(a, &a_min, &a_max);
433     extract_min_max(b, &b_min, &b_max);

435     /*
436      * Make sure we're properly sorted (paranoia).
437      */
438     OPENSSL_assert(ASN1_INTEGER_cmp(a_min, b_min) <= 0);

440     /*
441      * Punt inverted ranges.
442      */
443     if (ASN1_INTEGER_cmp(a_min, a_max) > 0 ||
444         ASN1_INTEGER_cmp(b_min, b_max) > 0)
445       goto done;

447     /*
448      * Check for overlaps.
449      */
450     if (ASN1_INTEGER_cmp(a_max, b_min) >= 0) {
451       X509V3err(X509V3_F_ASIDENTIFIERCHOICE_CANONIZE,
452                 X509V3_R_EXTENSION_VALUE_ERROR);
453       goto done;
454     }

456     /*
457      * Calculate a_max + 1 to check for adjacency.
```



```
458      */
459     if ((bn == NULL && (bn = BN_new()) == NULL) ||
460         ASN1_INTEGER_to_BN(a_max, bn) == NULL ||
461         !BN_add_word(bn, 1) ||
462         (a_max_plus_one = BN_to_ASN1_INTEGER(bn, a_max_plus_one)) == NULL) {
463       X509V3err(X509V3_F_ASIDENTIFIERCHOICE_CANONIZE, ERR_R_MALLOC_FAILURE);
464       goto done;
465     }

467     /*
468      * If a and b are adjacent, merge them.
469      */
470     if (ASN1_INTEGER_cmp(a_max_plus_one, b_min) == 0) {
471       ASRange *r;
472       switch (a->type) {
473       case ASIdOrRange_id:
474         if ((r = OPENSSL_malloc(sizeof(ASRange))) == NULL) {
475           X509V3err(X509V3_F_ASIDENTIFIERCHOICE_CANONIZE,
476                     ERR_R_MALLOC_FAILURE);
477           goto done;
478         }
479         r->min = a_min;
480         r->max = b_max;
481         a->type = ASIdOrRange_range;
482         a->u.range = r;
483         break;
484       case ASIdOrRange_range:
485         ASN1_INTEGER_free(a->u.range->max);
486         a->u.range->max = b_max;
487         break;
488       }
489       switch (b->type) {
490       case ASIdOrRange_id:
491         b->u.id = NULL;
492         break;
493       case ASIdOrRange_range:
494         b->u.range->max = NULL;
495         break;
496       }
497       ASIdOrRange_free(b);
498       (void) sk_ASIdOrRange_delete(choice->u.asIdsOrRanges, i + 1);
499       i--;
500       continue;
501     }
502   }

504   /*
505    * Check for final inverted range.
506    */
507   i = sk_ASIdOrRange_num(choice->u.asIdsOrRanges) - 1;
508   {
509     ASIdOrRange *a = sk_ASIdOrRange_value(choice->u.asIdsOrRanges, i);
510     ASN1_INTEGER *a_min, *a_max;
511     if (a != NULL && a->type == ASIdOrRange_range) {
512       extract_min_max(a, &a_min, &a_max);
513       if (ASN1_INTEGER_cmp(a_min, a_max) > 0)
514         goto done;
515     }
516   }

518   OPENSSL_assert(ASIdentifierChoice_is_canonical(choice)); /* Paranoia */

520   ret = 1;

522 done:
523   ASN1_INTEGER_free(a_max_plus_one);
```

```
524   BN_free(bn);
525   return ret;
526 }

528 /*
529  * Whack an ASIdentifier extension into canonical form.
530  */
531 int v3_asid_canonize(ASIdentifiers *asid)
532 {
533   return (asid == NULL ||
534           (ASIdentifierChoice_canonize(asid->asnum) &&
535            ASIdentifierChoice_canonize(asid->rdi)));
536 }

538 /*
539  * v2i method for an ASIdentifier extension.
540  */
541 static void *v2i_ASIdentifiers(const struct v3_ext_method *method,
542                                struct v3_ext_ctx *ctx,
543                                STACK_OF(CONF_VALUE) *values)
544 {
545   ASN1_INTEGER *min = NULL, *max = NULL;
546   ASIdentifiers *asid = NULL;
547   int i;

549   if ((asid = ASIdentifiers_new()) == NULL) {
550     X509V3err(X509V3_F_V2I_ASIDENTIFIERS, ERR_R_MALLOC_FAILURE);
551     return NULL;
552   }

554   for (i = 0; i < sk_CONF_VALUE_num(values); i++) {
555     CONF_VALUE *val = sk_CONF_VALUE_value(values, i);
556     int i1, i2, i3, is_range, which;

558     /*
559      * Figure out whether this is an AS or an RDI.
560      */
561     if (       !name_cmp(val->name, "AS")) {
562       which = V3_ASID_ASNUM;
563     } else if (!name_cmp(val->name, "RDI")) {
564       which = V3_ASID_RDI;
565     } else {
566       X509V3err(X509V3_F_V2I_ASIDENTIFIERS, X509V3_R_EXTENSION_NAME_ERROR);
567       X509V3_conf_err(val);
568       goto err;
569     }

571     /*
572      * Handle inheritance.
573      */
574     if (!strcmp(val->value, "inherit")) {
575       if (v3_asid_add_inherit(asid, which))
576         continue;
577       X509V3err(X509V3_F_V2I_ASIDENTIFIERS, X509V3_R_INVALID_INHERITANCE);
578       X509V3_conf_err(val);
579       goto err;
580     }

582     /*
583      * Number, range, or mistake, pick it apart and figure out which.
584      */
585     i1 = strspn(val->value, "0123456789");
586     if (val->value[i1] == '\0') {
587       is_range = 0;
588     } else {
589       is_range = 1;
```



```
590       i2 = i1 + strspn(val->value + i1, " \t");
591       if (val->value[i2] != '-') {
592         X509V3err(X509V3_F_V2I_ASIDENTIFIERS, X509V3_R_INVALID_ASNUMBER);
593         X509V3_conf_err(val);
594         goto err;
595       }
596       i2++;
597       i2 = i2 + strspn(val->value + i2, " \t");
598       i3 = i2 + strspn(val->value + i2, "0123456789");
599       if (val->value[i3] != '\0') {
600         X509V3err(X509V3_F_V2I_ASIDENTIFIERS, X509V3_R_INVALID_ASRANGE);
601         X509V3_conf_err(val);
602         goto err;
603       }
604     }

606     /*
607      * Syntax is ok, read and add it.
608      */
609     if (!is_range) {
610       if (!X509V3_get_value_int(val, &min)) {
611         X509V3err(X509V3_F_V2I_ASIDENTIFIERS, ERR_R_MALLOC_FAILURE);
612         goto err;
613       }
614     } else {
615       char *s = BUF_strdup(val->value);
616       if (s == NULL) {
617         X509V3err(X509V3_F_V2I_ASIDENTIFIERS, ERR_R_MALLOC_FAILURE);
618         goto err;
619       }
620       s[i1] = '\0';
621       min = s2i_ASN1_INTEGER(NULL, s);
622       max = s2i_ASN1_INTEGER(NULL, s + i2);
623       OPENSSL_free(s);
624       if (min == NULL || max == NULL) {
625         X509V3err(X509V3_F_V2I_ASIDENTIFIERS, ERR_R_MALLOC_FAILURE);
626         goto err;
627       }
628       if (ASN1_INTEGER_cmp(min, max) > 0) {
629         X509V3err(X509V3_F_V2I_ASIDENTIFIERS, X509V3_R_EXTENSION_VALUE_ERROR);
630         goto err;
631       }
632     }
633     if (!v3_asid_add_id_or_range(asid, which, min, max)) {
634       X509V3err(X509V3_F_V2I_ASIDENTIFIERS, ERR_R_MALLOC_FAILURE);
635       goto err;
636     }
637     min = max = NULL;
638   }

640   /*
641    * Canonize the result, then we're done.
642    */
643   if (!v3_asid_canonize(asid))
644     goto err;
645   return asid;

647  err:
648   ASIdentifiers_free(asid);
649   ASN1_INTEGER_free(min);
650   ASN1_INTEGER_free(max);
651   return NULL;
652 }

654 /*
655  * OpenSSL dispatch.
```

```
 */
const X509V3_EXT_METHOD v3_asid = {
  NID_sbgp_autonomousSysNum,    /* nid */
  0,                            /* flags */
  ASN1_ITEM_ref(ASIdentifiers), /* template */
  0, 0, 0, 0,                   /* old functions, ignored */
  0,                            /* i2s */
  0,                            /* s2i */
  0,                            /* i2v */
  v2i_ASIdentifiers,            /* v2i */
  i2r_ASIdentifiers,            /* i2r */
  0,                            /* r2i */
  NULL                          /* extension-specific data */
};

/*
 * Figure out whether extension uses inheritance.
 */
int v3_asid_inherits(ASIdentifiers *asid)
{
  return (asid != NULL &&
          ((asid->asnum != NULL &&
            asid->asnum->type == ASIdentifierChoice_inherit) ||
           (asid->rdi != NULL &&
            asid->rdi->type == ASIdentifierChoice_inherit)));
}

/*
 * Figure out whether parent contains child.
 */
static int asid_contains(ASIdOrRanges *parent, ASIdOrRanges *child)
{
  ASN1_INTEGER *p_min, *p_max, *c_min, *c_max;
  int p, c;

  if (child == NULL || parent == child)
    return 1;
  if (parent == NULL)
    return 0;

  p = 0;
  for (c = 0; c < sk_ASIdOrRange_num(child); c++) {
    extract_min_max(sk_ASIdOrRange_value(child, c), &c_min, &c_max);
    for (;; p++) {
      if (p >= sk_ASIdOrRange_num(parent))
        return 0;
      extract_min_max(sk_ASIdOrRange_value(parent, p), &p_min, &p_max);
      if (ASN1_INTEGER_cmp(p_max, c_max) < 0)
        continue;
      if (ASN1_INTEGER_cmp(p_min, c_min) > 0)
        return 0;
      break;
    }
  }

  return 1;
}

/*
 * Test whether a is a subet of b.
 */
int v3_asid_subset(ASIdentifiers *a, ASIdentifiers *b)
{
  return (a == NULL ||
          a == b ||
          (b != NULL &&
```



```
           !v3_asid_inherits(a) &&
           !v3_asid_inherits(b) &&
           asid_contains(b->asnum->u.asIdsOrRanges,
                         a->asnum->u.asIdsOrRanges) &&
           asid_contains(b->rdi->u.asIdsOrRanges,
                         a->rdi->u.asIdsOrRanges)));
}

/*
 * Validation error handling via callback.
 */
#define validation_err(_err_)           \
  do {                                  \
    if (ctx != NULL) {                  \
      ctx->error = _err_;               \
      ctx->error_depth = i;             \
      ctx->current_cert = x;            \
      ret = ctx->verify_cb(0, ctx);     \
    } else {                            \
      ret = 0;                          \
    }                                   \
    if (!ret)                           \
      goto done;                        \
  } while (0)

/*
 * Core code for RFC 3779 3.3 path validation.
 */
static int v3_asid_validate_path_internal(X509_STORE_CTX *ctx,
                                          STACK_OF(X509) *chain,
                                          ASIdentifiers *ext)
{
  ASIdOrRanges *child_as = NULL, *child_rdi = NULL;
  int i, ret = 1, inherit_as = 0, inherit_rdi = 0;
  X509 *x;

  OPENSSL_assert(chain != NULL && sk_X509_num(chain) > 0);
  OPENSSL_assert(ctx != NULL || ext != NULL);
  OPENSSL_assert(ctx == NULL || ctx->verify_cb != NULL);

  /*
   * Figure out where to start.  If we don't have an extension to
   * check, we're done.  Otherwise, check canonical form and
   * set up for walking up the chain.
   */
  if (ext != NULL) {
    i = -1;
    x = NULL;
  } else {
    i = 0;
    x = sk_X509_value(chain, i);
    OPENSSL_assert(x != NULL);
    if ((ext = x->rfc3779_asid) == NULL)
      goto done;
  }
  if (!v3_asid_is_canonical(ext))
    validation_err(X509_V_ERR_INVALID_EXTENSION);
  if (ext->asnum != NULL)  {
    switch (ext->asnum->type) {
    case ASIdentifierChoice_inherit:
      inherit_as = 1;
      break;
    case ASIdentifierChoice_asIdsOrRanges:
      child_as = ext->asnum->u.asIdsOrRanges;
      break;
    }
```

```
788   }
789   if (ext->rdi != NULL) {
790     switch (ext->rdi->type) {
791     case ASIdentifierChoice_inherit:
792       inherit_rdi = 1;
793       break;
794     case ASIdentifierChoice_asIdsOrRanges:
795       child_rdi = ext->rdi->u.asIdsOrRanges;
796       break;
797     }
798   }

800   /*
801    * Now walk up the chain.  Extensions must be in canonical form, no
802    * cert may list resources that its parent doesn't list.
803    */
804   for (i++; i < sk_X509_num(chain); i++) {
805     x = sk_X509_value(chain, i);
806     OPENSSL_assert(x != NULL);
807     if (x->rfc3779_asid == NULL) {
808       if (child_as != NULL || child_rdi != NULL)
809         validation_err(X509_V_ERR_UNNESTED_RESOURCE);
810       continue;
811     }
812     if (!v3_asid_is_canonical(x->rfc3779_asid))
813       validation_err(X509_V_ERR_INVALID_EXTENSION);
814     if (x->rfc3779_asid->asnum == NULL && child_as != NULL) {
815       validation_err(X509_V_ERR_UNNESTED_RESOURCE);
816       child_as = NULL;
817       inherit_as = 0;
818     }
819     if (x->rfc3779_asid->asnum != NULL &&
820         x->rfc3779_asid->asnum->type == ASIdentifierChoice_asIdsOrRanges) {
821       if (inherit_as ||
822           asid_contains(x->rfc3779_asid->asnum->u.asIdsOrRanges, child_as)) {
823         child_as = x->rfc3779_asid->asnum->u.asIdsOrRanges;
824         inherit_as = 0;
825       } else {
826         validation_err(X509_V_ERR_UNNESTED_RESOURCE);
827       }
828     }
829     if (x->rfc3779_asid->rdi == NULL && child_rdi != NULL) {
830       validation_err(X509_V_ERR_UNNESTED_RESOURCE);
831       child_rdi = NULL;
832       inherit_rdi = 0;
833     }
834     if (x->rfc3779_asid->rdi != NULL &&
835         x->rfc3779_asid->rdi->type == ASIdentifierChoice_asIdsOrRanges) {
836       if (inherit_rdi ||
837           asid_contains(x->rfc3779_asid->rdi->u.asIdsOrRanges, child_rdi)) {
838         child_rdi = x->rfc3779_asid->rdi->u.asIdsOrRanges;
839         inherit_rdi = 0;
840       } else {
841         validation_err(X509_V_ERR_UNNESTED_RESOURCE);
842       }
843     }
844   }

846   /*
847    * Trust anchor can't inherit.
848    */
849   OPENSSL_assert(x != NULL);
850   if (x->rfc3779_asid != NULL) {
851     if (x->rfc3779_asid->asnum != NULL &&
852         x->rfc3779_asid->asnum->type == ASIdentifierChoice_inherit)
853       validation_err(X509_V_ERR_UNNESTED_RESOURCE);
```



```
854     if (x->rfc3779_asid->rdi != NULL &&
855         x->rfc3779_asid->rdi->type == ASIdentifierChoice_inherit)
856       validation_err(X509_V_ERR_UNNESTED_RESOURCE);
857   }

859  done:
860   return ret;
861 }

863 #undef validation_err

865 /*
866  * RFC 3779 3.3 path validation -- called from X509_verify_cert().
867  */
868 int v3_asid_validate_path(X509_STORE_CTX *ctx)
869 {
870   return v3_asid_validate_path_internal(ctx, ctx->chain, NULL);
871 }

873 /*
874  * RFC 3779 3.3 path validation of an extension.
875  * Test whether chain covers extension.
876  */
877 int v3_asid_validate_resource_set(STACK_OF(X509) *chain,
878                                   ASIdentifiers *ext,
879                                   int allow_inheritance)
880 {
881   if (ext == NULL)
882     return 1;
883   if (chain == NULL || sk_X509_num(chain) == 0)
884     return 0;
885   if (!allow_inheritance && v3_asid_inherits(ext))
886     return 0;
887   return v3_asid_validate_path_internal(NULL, chain, ext);
888 }

890 #endif /* OPENSSL_NO_RFC3779 */
891 #endif /* ! codereview */
```

```
**********************************************************
    4646 Wed Aug 13 19:53:29 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_bcons.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_bcons.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/conf.h>
#include <openssl/x509v3.h>

static STACK_OF(CONF_VALUE) *i2v_BASIC_CONSTRAINTS(X509V3_EXT_METHOD *method, BA
static BASIC_CONSTRAINTS *v2i_BASIC_CONSTRAINTS(X509V3_EXT_METHOD *method, X509V

const X509V3_EXT_METHOD v3_bcons = {
NID_basic_constraints, 0,
ASN1_ITEM_ref(BASIC_CONSTRAINTS),
0,0,0,0,
0,0,
(X509V3_EXT_I2V)i2v_BASIC_CONSTRAINTS,
(X509V3_EXT_V2I)v2i_BASIC_CONSTRAINTS,
NULL,NULL,
NULL
};

ASN1_SEQUENCE(BASIC_CONSTRAINTS) = {
        ASN1_OPT(BASIC_CONSTRAINTS, ca, ASN1_FBOOLEAN),
        ASN1_OPT(BASIC_CONSTRAINTS, pathlen, ASN1_INTEGER)
} ASN1_SEQUENCE_END(BASIC_CONSTRAINTS)

IMPLEMENT_ASN1_FUNCTIONS(BASIC_CONSTRAINTS)


static STACK_OF(CONF_VALUE) *i2v_BASIC_CONSTRAINTS(X509V3_EXT_METHOD *method,
             BASIC_CONSTRAINTS *bcons, STACK_OF(CONF_VALUE) *extlist)
{
        X509V3_add_value_bool("CA", bcons->ca, &extlist);
        X509V3_add_value_int("pathlen", bcons->pathlen, &extlist);
        return extlist;
}

static BASIC_CONSTRAINTS *v2i_BASIC_CONSTRAINTS(X509V3_EXT_METHOD *method,
             X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *values)
{
        BASIC_CONSTRAINTS *bcons=NULL;
        CONF_VALUE *val;
        int i;
        if(!(bcons = BASIC_CONSTRAINTS_new())) {
                X509V3err(X509V3_F_V2I_BASIC_CONSTRAINTS, ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        for(i = 0; i < sk_CONF_VALUE_num(values); i++) {
                val = sk_CONF_VALUE_value(values, i);
                if(!strcmp(val->name, "CA")) {
                        if(!X509V3_get_value_bool(val, &bcons->ca)) goto err;
                } else if(!strcmp(val->name, "pathlen")) {
                        if(!X509V3_get_value_int(val, &bcons->pathlen)) goto err
                } else {
                        X509V3err(X509V3_F_V2I_BASIC_CONSTRAINTS, X509V3_R_INVAL
                        X509V3_conf_err(val);
                        goto err;
                }
        }
        return bcons;
        err:
        BASIC_CONSTRAINTS_free(bcons);
        return NULL;
}
#endif /* ! codereview */
```

```
**********************************************************
    5019 Wed Aug 13 19:53:29 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_bitst.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_bitst.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/x509v3.h>

static BIT_STRING_BITNAME ns_cert_type_table[] = {
{0, "SSL Client", "client"},
{1, "SSL Server", "server"},
{2, "S/MIME", "email"},
{3, "Object Signing", "objsign"},
{4, "Unused", "reserved"},
{5, "SSL CA", "sslCA"},
{6, "S/MIME CA", "emailCA"},
{7, "Object Signing CA", "objCA"},
{-1, NULL, NULL}
};

static BIT_STRING_BITNAME key_usage_type_table[] = {
{0, "Digital Signature", "digitalSignature"},
{1, "Non Repudiation", "nonRepudiation"},
{2, "Key Encipherment", "keyEncipherment"},
{3, "Data Encipherment", "dataEncipherment"},
{4, "Key Agreement", "keyAgreement"},
{5, "Certificate Sign", "keyCertSign"},
{6, "CRL Sign", "cRLSign"},
{7, "Encipher Only", "encipherOnly"},
{8, "Decipher Only", "decipherOnly"},
{-1, NULL, NULL}
};



const X509V3_EXT_METHOD v3_nscert = EXT_BITSTRING(NID_netscape_cert_type, ns_cer
const X509V3_EXT_METHOD v3_key_usage = EXT_BITSTRING(NID_key_usage, key_usage_ty

STACK_OF(CONF_VALUE) *i2v_ASN1_BIT_STRING(X509V3_EXT_METHOD *method,
             ASN1_BIT_STRING *bits, STACK_OF(CONF_VALUE) *ret)
{
        BIT_STRING_BITNAME *bnam;
        for(bnam =method->usr_data; bnam->lname; bnam++) {
                if(ASN1_BIT_STRING_get_bit(bits, bnam->bitnum))
                        X509V3_add_value(bnam->lname, NULL, &ret);
        }
        return ret;
}

ASN1_BIT_STRING *v2i_ASN1_BIT_STRING(X509V3_EXT_METHOD *method,
             X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
{
        CONF_VALUE *val;
        ASN1_BIT_STRING *bs;
        int i;
        BIT_STRING_BITNAME *bnam;
        if(!(bs = M_ASN1_BIT_STRING_new())) {
                X509V3err(X509V3_F_V2I_ASN1_BIT_STRING,ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
                val = sk_CONF_VALUE_value(nval, i);
                for(bnam = method->usr_data; bnam->lname; bnam++) {
                        if(!strcmp(bnam->sname, val->name) ||
                                !strcmp(bnam->lname, val->name) ) {
                                if(!ASN1_BIT_STRING_set_bit(bs, bnam->bitnum, 1)
                                        X509V3err(X509V3_F_V2I_ASN1_BIT_STRING,
                                                ERR_R_MALLOC_FAILURE);
                                        M_ASN1_BIT_STRING_free(bs);
                                        return NULL;
                                }
                                break;
```

```
128                         }
129                 }
130                 if(!bnam->lname) {
131                         X509V3err(X509V3_F_V2I_ASN1_BIT_STRING,
132                                         X509V3_R_UNKNOWN_BIT_STRING_ARGUMENT);
133                         X509V3_conf_err(val);
134                         M_ASN1_BIT_STRING_free(bs);
135                         return NULL;
136                 }
137         }
138         return bs;
139 }
140 #endif /* ! codereview */
```

```
**********************************************************
   14850 Wed Aug 13 19:53:29 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_conf.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_conf.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* extension creation utilities */
```



```
#include <stdio.h>
#include <ctype.h>
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/x509.h>
#include <openssl/x509v3.h>

static int v3_check_critical(char **value);
static int v3_check_generic(char **value);
static X509_EXTENSION *do_ext_nconf(CONF *conf, X509V3_CTX *ctx, int ext_nid, in
static X509_EXTENSION *v3_generic_extension(const char *ext, char *value, int cr
static char *conf_lhash_get_string(void *db, char *section, char *value);
static STACK_OF(CONF_VALUE) *conf_lhash_get_section(void *db, char *section);
static X509_EXTENSION *do_ext_i2d(const X509V3_EXT_METHOD *method, int ext_nid,
                                  int crit, void *ext_struc);
static unsigned char *generic_asn1(char *value, X509V3_CTX *ctx, long *ext_len);
/* CONF *conf:  Config file    */
/* char *name:  Name    */
/* char *value:  Value    */
X509_EXTENSION *X509V3_EXT_nconf(CONF *conf, X509V3_CTX *ctx, char *name,
                                 char *value)
        {
        int crit;
        int ext_type;
        X509_EXTENSION *ret;
        crit = v3_check_critical(&value);
        if ((ext_type = v3_check_generic(&value)))
                return v3_generic_extension(name, value, crit, ext_type, ctx);
        ret = do_ext_nconf(conf, ctx, OBJ_sn2nid(name), crit, value);
        if (!ret)
                {
                X509V3err(X509V3_F_X509V3_EXT_NCONF,X509V3_R_ERROR_IN_EXTENSION)
                ERR_add_error_data(4,"name=", name, ", value=", value);
                }
        return ret;
        }

/* CONF *conf:  Config file    */
/* char *value:  Value    */
X509_EXTENSION *X509V3_EXT_nconf_nid(CONF *conf, X509V3_CTX *ctx, int ext_nid,
                                     char *value)
        {
        int crit;
        int ext_type;
        crit = v3_check_critical(&value);
        if ((ext_type = v3_check_generic(&value)))
                return v3_generic_extension(OBJ_nid2sn(ext_nid),
                                            value, crit, ext_type, ctx);
        return do_ext_nconf(conf, ctx, ext_nid, crit, value);
        }

/* CONF *conf:  Config file    */
/* char *value:  Value    */
static X509_EXTENSION *do_ext_nconf(CONF *conf, X509V3_CTX *ctx, int ext_nid,
                                    int crit, char *value)
        {
        const X509V3_EXT_METHOD *method;
        X509_EXTENSION *ext;
        STACK_OF(CONF_VALUE) *nval;
        void *ext_struc;
        if (ext_nid == NID_undef)
                {
                X509V3err(X509V3_F_DO_EXT_NCONF,X509V3_R_UNKNOWN_EXTENSION_NAME)
                return NULL;
                }
        if (!(method = X509V3_EXT_get_nid(ext_nid)))
```

```
128                 {
129                 X509V3err(X509V3_F_DO_EXT_NCONF,X509V3_R_UNKNOWN_EXTENSION);
130                 return NULL;
131                 }
132         /* Now get internal extension representation based on type */
133         if (method->v2i)
134                 {
135                 if(*value == '@') nval = NCONF_get_section(conf, value + 1);
136                 else nval = X509V3_parse_list(value);
137                 if(sk_CONF_VALUE_num(nval) <= 0)
138                         {
139                         X509V3err(X509V3_F_DO_EXT_NCONF,X509V3_R_INVALID_EXTENSI
140                         ERR_add_error_data(4, "name=", OBJ_nid2sn(ext_nid), ",se
141                         return NULL;
142                         }
143                 ext_struc = method->v2i(method, ctx, nval);
144                 if(*value != '@') sk_CONF_VALUE_pop_free(nval,
145                                                         X509V3_conf_free);
146                 if(!ext_struc) return NULL;
147                 }
148         else if(method->s2i)
149                 {
150                 if(!(ext_struc = method->s2i(method, ctx, value))) return NULL;
151                 }
152         else if(method->r2i)
153                 {
154                 if(!ctx->db || !ctx->db_meth)
155                         {
156                         X509V3err(X509V3_F_DO_EXT_NCONF,X509V3_R_NO_CONFIG_DATAB
157                         return NULL;
158                         }
159                 if(!(ext_struc = method->r2i(method, ctx, value))) return NULL;
160                 }
161         else
162                 {
163                 X509V3err(X509V3_F_DO_EXT_NCONF,X509V3_R_EXTENSION_SETTING_NOT_S
164                 ERR_add_error_data(2, "name=", OBJ_nid2sn(ext_nid));
165                 return NULL;
166                 }

168         ext  = do_ext_i2d(method, ext_nid, crit, ext_struc);
169         if(method->it) ASN1_item_free(ext_struc, ASN1_ITEM_ptr(method->it));
170         else method->ext_free(ext_struc);
171         return ext;

173         }

175 static X509_EXTENSION *do_ext_i2d(const X509V3_EXT_METHOD *method, int ext_nid,
176                                   int crit, void *ext_struc)
177         {
178         unsigned char *ext_der;
179         int ext_len;
180         ASN1_OCTET_STRING *ext_oct;
181         X509_EXTENSION *ext;
182         /* Convert internal representation to DER */
183         if (method->it)
184                 {
185                 ext_der = NULL;
186                 ext_len = ASN1_item_i2d(ext_struc, &ext_der, ASN1_ITEM_ptr(metho
187                 if (ext_len < 0) goto merr;
188                 }
189          else
190                 {
191                 unsigned char *p;
192                 ext_len = method->i2d(ext_struc, NULL);
193                 if(!(ext_der = OPENSSL_malloc(ext_len))) goto merr;
```



```
194                 p = ext_der;
195                 method->i2d(ext_struc, &p);
196                 }
197         if (!(ext_oct = M_ASN1_OCTET_STRING_new())) goto merr;
198         ext_oct->data = ext_der;
199         ext_oct->length = ext_len;

201         ext = X509_EXTENSION_create_by_NID(NULL, ext_nid, crit, ext_oct);
202         if (!ext) goto merr;
203         M_ASN1_OCTET_STRING_free(ext_oct);

205         return ext;

207         merr:
208         X509V3err(X509V3_F_DO_EXT_I2D,ERR_R_MALLOC_FAILURE);
209         return NULL;

211         }

213 /* Given an internal structure, nid and critical flag create an extension */

215 X509_EXTENSION *X509V3_EXT_i2d(int ext_nid, int crit, void *ext_struc)
216         {
217         const X509V3_EXT_METHOD *method;
218         if (!(method = X509V3_EXT_get_nid(ext_nid))) {
219                 X509V3err(X509V3_F_X509V3_EXT_I2D,X509V3_R_UNKNOWN_EXTENSION);
220                 return NULL;
221         }
222         return do_ext_i2d(method, ext_nid, crit, ext_struc);
223 }

225 /* Check the extension string for critical flag */
226 static int v3_check_critical(char **value)
227 {
228         char *p = *value;
229         if ((strlen(p) < 9) || strncmp(p, "critical,", 9)) return 0;
230         p+=9;
231         while(isspace((unsigned char)*p)) p++;
232         *value = p;
233         return 1;
234 }

236 /* Check extension string for generic extension and return the type */
237 static int v3_check_generic(char **value)
238 {
239         int gen_type = 0;
240         char *p = *value;
241         if ((strlen(p) >= 4) && !strncmp(p, "DER:", 4))
242                 {
243                 p+=4;
244                 gen_type = 1;
245                 }
246         else if ((strlen(p) >= 5) && !strncmp(p, "ASN1:", 5))
247                 {
248                 p+=5;
249                 gen_type = 2;
250                 }
251         else
252                 return 0;

254         while (isspace((unsigned char)*p)) p++;
255         *value = p;
256         return gen_type;
257 }

259 /* Create a generic extension: for now just handle DER type */
```

```
260 static X509_EXTENSION *v3_generic_extension(const char *ext, char *value,
261                                               int crit, int gen_type,
262                                               X509V3_CTX *ctx)
263         {
264         unsigned char *ext_der=NULL;
265         long ext_len;
266         ASN1_OBJECT *obj=NULL;
267         ASN1_OCTET_STRING *oct=NULL;
268         X509_EXTENSION *extension=NULL;
269         if (!(obj = OBJ_txt2obj(ext, 0)))
270                 {
271                 X509V3err(X509V3_F_V3_GENERIC_EXTENSION,X509V3_R_EXTENSION_NAME_
272                 ERR_add_error_data(2, "name=", ext);
273                 goto err;
274                 }

276         if (gen_type == 1)
277                 ext_der = string_to_hex(value, &ext_len);
278         else if (gen_type == 2)
279                 ext_der = generic_asn1(value, ctx, &ext_len);

281         if (ext_der == NULL)
282                 {
283                 X509V3err(X509V3_F_V3_GENERIC_EXTENSION,X509V3_R_EXTENSION_VALUE
284                 ERR_add_error_data(2, "value=", value);
285                 goto err;
286                 }

288         if (!(oct = M_ASN1_OCTET_STRING_new()))
289                 {
290                 X509V3err(X509V3_F_V3_GENERIC_EXTENSION,ERR_R_MALLOC_FAILURE);
291                 goto err;
292                 }

294         oct->data = ext_der;
295         oct->length = ext_len;
296         ext_der = NULL;

298         extension = X509_EXTENSION_create_by_OBJ(NULL, obj, crit, oct);

300         err:
301         ASN1_OBJECT_free(obj);
302         M_ASN1_OCTET_STRING_free(oct);
303         if(ext_der) OPENSSL_free(ext_der);
304         return extension;

306         }

308 static unsigned char *generic_asn1(char *value, X509V3_CTX *ctx, long *ext_len)
309         {
310         ASN1_TYPE *typ;
311         unsigned char *ext_der = NULL;
312         typ = ASN1_generate_v3(value, ctx);
313         if (typ == NULL)
314                 return NULL;
315         *ext_len = i2d_ASN1_TYPE(typ, &ext_der);
316         ASN1_TYPE_free(typ);
317         return ext_der;
318         }

320 /* This is the main function: add a bunch of extensions based on a config file
321  * section to an extension STACK.
322  */


325 int X509V3_EXT_add_nconf_sk(CONF *conf, X509V3_CTX *ctx, char *section,
```



```
326                             STACK_OF(X509_EXTENSION) **sk)
327         {
328         X509_EXTENSION *ext;
329         STACK_OF(CONF_VALUE) *nval;
330         CONF_VALUE *val;
331         int i;
332         if (!(nval = NCONF_get_section(conf, section))) return 0;
333         for (i = 0; i < sk_CONF_VALUE_num(nval); i++)
334                 {
335                 val = sk_CONF_VALUE_value(nval, i);
336                 if (!(ext = X509V3_EXT_nconf(conf, ctx, val->name, val->value)))
337                                                                 return 0;
338                 if (sk) X509v3_add_ext(sk, ext, -1);
339                 X509_EXTENSION_free(ext);
340                 }
341         return 1;
342         }

344 /* Convenience functions to add extensions to a certificate, CRL and request */

346 int X509V3_EXT_add_nconf(CONF *conf, X509V3_CTX *ctx, char *section,
347                          X509 *cert)
348         {
349         STACK_OF(X509_EXTENSION) **sk = NULL;
350         if (cert)
351                 sk = &cert->cert_info->extensions;
352         return X509V3_EXT_add_nconf_sk(conf, ctx, section, sk);
353         }

355 /* Same as above but for a CRL */

357 int X509V3_EXT_CRL_add_nconf(CONF *conf, X509V3_CTX *ctx, char *section,
358                              X509_CRL *crl)
359         {
360         STACK_OF(X509_EXTENSION) **sk = NULL;
361         if (crl)
362                 sk = &crl->crl->extensions;
363         return X509V3_EXT_add_nconf_sk(conf, ctx, section, sk);
364         }

366 /* Add extensions to certificate request */

368 int X509V3_EXT_REQ_add_nconf(CONF *conf, X509V3_CTX *ctx, char *section,
369              X509_REQ *req)
370         {
371         STACK_OF(X509_EXTENSION) *extlist = NULL, **sk = NULL;
372         int i;
373         if (req)
374                 sk = &extlist;
375         i = X509V3_EXT_add_nconf_sk(conf, ctx, section, sk);
376         if (!i || !sk)
377                 return i;
378         i = X509_REQ_add_extensions(req, extlist);
379         sk_X509_EXTENSION_pop_free(extlist, X509_EXTENSION_free);
380         return i;
381         }

383 /* Config database functions */

385 char * X509V3_get_string(X509V3_CTX *ctx, char *name, char *section)
386         {
387         if(!ctx->db || !ctx->db_meth || !ctx->db_meth->get_string)
388                 {
389                 X509V3err(X509V3_F_X509V3_GET_STRING,X509V3_R_OPERATION_NOT_DEFI
390                 return NULL;
391                 }
```

```
392         if (ctx->db_meth->get_string)
393                         return ctx->db_meth->get_string(ctx->db, name, section);
394         return NULL;
395         }

397 STACK_OF(CONF_VALUE) * X509V3_get_section(X509V3_CTX *ctx, char *section)
398         {
399         if(!ctx->db || !ctx->db_meth || !ctx->db_meth->get_section)
400                 {
401                 X509V3err(X509V3_F_X509V3_GET_SECTION,X509V3_R_OPERATION_NOT_DEF
402                 return NULL;
403                 }
404         if (ctx->db_meth->get_section)
405                         return ctx->db_meth->get_section(ctx->db, section);
406         return NULL;
407         }

409 void X509V3_string_free(X509V3_CTX *ctx, char *str)
410         {
411         if (!str) return;
412         if (ctx->db_meth->free_string)
413                         ctx->db_meth->free_string(ctx->db, str);
414         }

416 void X509V3_section_free(X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *section)
417         {
418         if (!section) return;
419         if (ctx->db_meth->free_section)
420                         ctx->db_meth->free_section(ctx->db, section);
421         }

423 static char *nconf_get_string(void *db, char *section, char *value)
424         {
425         return NCONF_get_string(db, section, value);
426         }

428 static STACK_OF(CONF_VALUE) *nconf_get_section(void *db, char *section)
429         {
430         return NCONF_get_section(db, section);
431         }

433 static X509V3_CONF_METHOD nconf_method = {
434 nconf_get_string,
435 nconf_get_section,
436 NULL,
437 NULL
438 };

440 void X509V3_set_nconf(X509V3_CTX *ctx, CONF *conf)
441         {
442         ctx->db_meth = &nconf_method;
443         ctx->db = conf;
444         }

446 void X509V3_set_ctx(X509V3_CTX *ctx, X509 *issuer, X509 *subj, X509_REQ *req,
447                         X509_CRL *crl, int flags)
448         {
449         ctx->issuer_cert = issuer;
450         ctx->subject_cert = subj;
451         ctx->crl = crl;
452         ctx->subject_req = req;
453         ctx->flags = flags;
454         }

456 /* Old conf compatibility functions */
```



```
458 X509_EXTENSION *X509V3_EXT_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
459                                 char *name, char *value)
460         {
461         CONF ctmp;
462         CONF_set_nconf(&ctmp, conf);
463         return X509V3_EXT_nconf(&ctmp, ctx, name, value);
464         }

466 /* LHASH *conf:  Config file    */
467 /* char *value:  Value    */
468 X509_EXTENSION *X509V3_EXT_conf_nid(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
469                                     int ext_nid, char *value)
470         {
471         CONF ctmp;
472         CONF_set_nconf(&ctmp, conf);
473         return X509V3_EXT_nconf_nid(&ctmp, ctx, ext_nid, value);
474         }

476 static char *conf_lhash_get_string(void *db, char *section, char *value)
477         {
478         return CONF_get_string(db, section, value);
479         }

481 static STACK_OF(CONF_VALUE) *conf_lhash_get_section(void *db, char *section)
482         {
483         return CONF_get_section(db, section);
484         }

486 static X509V3_CONF_METHOD conf_lhash_method = {
487 conf_lhash_get_string,
488 conf_lhash_get_section,
489 NULL,
490 NULL
491 };

493 void X509V3_set_conf_lhash(X509V3_CTX *ctx, LHASH_OF(CONF_VALUE) *lhash)
494         {
495         ctx->db_meth = &conf_lhash_method;
496         ctx->db = lhash;
497         }

499 int X509V3_EXT_add_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
500                         char *section, X509 *cert)
501         {
502         CONF ctmp;
503         CONF_set_nconf(&ctmp, conf);
504         return X509V3_EXT_add_nconf(&ctmp, ctx, section, cert);
505         }

507 /* Same as above but for a CRL */

509 int X509V3_EXT_CRL_add_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
510                             char *section, X509_CRL *crl)
511         {
512         CONF ctmp;
513         CONF_set_nconf(&ctmp, conf);
514         return X509V3_EXT_CRL_add_nconf(&ctmp, ctx, section, crl);
515         }

517 /* Add extensions to certificate request */

519 int X509V3_EXT_REQ_add_conf(LHASH_OF(CONF_VALUE) *conf, X509V3_CTX *ctx,
520                             char *section, X509_REQ *req)
521         {
522         CONF ctmp;
523         CONF_set_nconf(&ctmp, conf);
```

```
524         return X509V3_EXT_REQ_add_nconf(&ctmp, ctx, section, req);
525         }
526 #endif /* ! codereview */
```

```
**********************************************************
   13910 Wed Aug 13 19:53:29 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_cpols.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_cpols.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

#include "pcy_int.h"

/* Certificate policies extension support: this one is a bit complex... */

static int i2r_certpol(X509V3_EXT_METHOD *method, STACK_OF(POLICYINFO) *pol, BIO
static STACK_OF(POLICYINFO) *r2i_certpol(X509V3_EXT_METHOD *method, X509V3_CTX *
static void print_qualifiers(BIO *out, STACK_OF(POLICYQUALINFO) *quals, int inde
static void print_notice(BIO *out, USERNOTICE *notice, int indent);
static POLICYINFO *policy_section(X509V3_CTX *ctx,
                                  STACK_OF(CONF_VALUE) *polstrs, int ia5org);
static POLICYQUALINFO *notice_section(X509V3_CTX *ctx,
                                      STACK_OF(CONF_VALUE) *unot, int ia5org);
static int nref_nos(STACK_OF(ASN1_INTEGER) *nnums, STACK_OF(CONF_VALUE) *nos);

const X509V3_EXT_METHOD v3_cpols = {
NID_certificate_policies, 0,ASN1_ITEM_ref(CERTIFICATEPOLICIES),
0,0,0,0,
0,0,
0,0,
(X509V3_EXT_I2R)i2r_certpol,
(X509V3_EXT_R2I)r2i_certpol,
NULL
};

ASN1_ITEM_TEMPLATE(CERTIFICATEPOLICIES) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, CERTIFICATEPOLICIES, POL
ASN1_ITEM_TEMPLATE_END(CERTIFICATEPOLICIES)

IMPLEMENT_ASN1_FUNCTIONS(CERTIFICATEPOLICIES)

ASN1_SEQUENCE(POLICYINFO) = {
        ASN1_SIMPLE(POLICYINFO, policyid, ASN1_OBJECT),
        ASN1_SEQUENCE_OF_OPT(POLICYINFO, qualifiers, POLICYQUALINFO)
} ASN1_SEQUENCE_END(POLICYINFO)

IMPLEMENT_ASN1_FUNCTIONS(POLICYINFO)

ASN1_ADB_TEMPLATE(policydefault) = ASN1_SIMPLE(POLICYQUALINFO, d.other, ASN1_ANY

ASN1_ADB(POLICYQUALINFO) = {
        ADB_ENTRY(NID_id_qt_cps, ASN1_SIMPLE(POLICYQUALINFO, d.cpsuri, ASN1_IA5S
        ADB_ENTRY(NID_id_qt_unotice, ASN1_SIMPLE(POLICYQUALINFO, d.usernotice, U
} ASN1_ADB_END(POLICYQUALINFO, 0, pqualid, 0, &policydefault_tt, NULL);

ASN1_SEQUENCE(POLICYQUALINFO) = {
        ASN1_SIMPLE(POLICYQUALINFO, pqualid, ASN1_OBJECT),
        ASN1_ADB_OBJECT(POLICYQUALINFO)
} ASN1_SEQUENCE_END(POLICYQUALINFO)

IMPLEMENT_ASN1_FUNCTIONS(POLICYQUALINFO)

ASN1_SEQUENCE(USERNOTICE) = {
        ASN1_OPT(USERNOTICE, noticeref, NOTICEREF),
        ASN1_OPT(USERNOTICE, exptext, DISPLAYTEXT)
} ASN1_SEQUENCE_END(USERNOTICE)

IMPLEMENT_ASN1_FUNCTIONS(USERNOTICE)

ASN1_SEQUENCE(NOTICEREF) = {
        ASN1_SIMPLE(NOTICEREF, organization, DISPLAYTEXT),
        ASN1_SEQUENCE_OF(NOTICEREF, noticenos, ASN1_INTEGER)
} ASN1_SEQUENCE_END(NOTICEREF)
```

```
IMPLEMENT_ASN1_FUNCTIONS(NOTICEREF)

static STACK_OF(POLICYINFO) *r2i_certpol(X509V3_EXT_METHOD *method,
                X509V3_CTX *ctx, char *value)
{
        STACK_OF(POLICYINFO) *pols = NULL;
        char *pstr;
        POLICYINFO *pol;
        ASN1_OBJECT *pobj;
        STACK_OF(CONF_VALUE) *vals;
        CONF_VALUE *cnf;
        int i, ia5org;
        pols = sk_POLICYINFO_new_null();
        if (pols == NULL) {
                X509V3err(X509V3_F_R2I_CERTPOL, ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        vals =  X509V3_parse_list(value);
        if (vals == NULL) {
                X509V3err(X509V3_F_R2I_CERTPOL, ERR_R_X509V3_LIB);
                goto err;
        }
        ia5org = 0;
        for(i = 0; i < sk_CONF_VALUE_num(vals); i++) {
                cnf = sk_CONF_VALUE_value(vals, i);
                if(cnf->value || !cnf->name ) {
                        X509V3err(X509V3_F_R2I_CERTPOL,X509V3_R_INVALID_POLICY_I
                        X509V3_conf_err(cnf);
                        goto err;
                }
                pstr = cnf->name;
                if(!strcmp(pstr,"ia5org")) {
                        ia5org = 1;
                        continue;
                } else if(*pstr == '@') {
                        STACK_OF(CONF_VALUE) *polsect;
                        polsect = X509V3_get_section(ctx, pstr + 1);
                        if(!polsect) {
                                X509V3err(X509V3_F_R2I_CERTPOL,X509V3_R_INVALID_

                                X509V3_conf_err(cnf);
                                goto err;
                        }
                        pol = policy_section(ctx, polsect, ia5org);
                        X509V3_section_free(ctx, polsect);
                        if(!pol) goto err;
                } else {
                        if(!(pobj = OBJ_txt2obj(cnf->name, 0))) {
                                X509V3err(X509V3_F_R2I_CERTPOL,X509V3_R_INVALID_
                                X509V3_conf_err(cnf);
                                goto err;
                        }
                        pol = POLICYINFO_new();
                        pol->policyid = pobj;
                }
                if (!sk_POLICYINFO_push(pols, pol)){
                        POLICYINFO_free(pol);
                        X509V3err(X509V3_F_R2I_CERTPOL, ERR_R_MALLOC_FAILURE);
                        goto err;
                }
        }
        sk_CONF_VALUE_pop_free(vals, X509V3_conf_free);
        return pols;
        err:
        sk_CONF_VALUE_pop_free(vals, X509V3_conf_free);
```



```
        sk_POLICYINFO_pop_free(pols, POLICYINFO_free);
        return NULL;
}

static POLICYINFO *policy_section(X509V3_CTX *ctx,
                                STACK_OF(CONF_VALUE) *polstrs, int ia5org)
{
        int i;
        CONF_VALUE *cnf;
        POLICYINFO *pol;
        POLICYQUALINFO *qual;
        if(!(pol = POLICYINFO_new())) goto merr;
        for(i = 0; i < sk_CONF_VALUE_num(polstrs); i++) {
                cnf = sk_CONF_VALUE_value(polstrs, i);
                if(!strcmp(cnf->name, "policyIdentifier")) {
                        ASN1_OBJECT *pobj;
                        if(!(pobj = OBJ_txt2obj(cnf->value, 0))) {
                                X509V3err(X509V3_F_POLICY_SECTION,X509V3_R_INVAL
                                X509V3_conf_err(cnf);
                                goto err;
                        }
                        pol->policyid = pobj;

                } else if(!name_cmp(cnf->name, "CPS")) {
                        if(!pol->qualifiers) pol->qualifiers =
                                                sk_POLICYQUALINFO_new_null();
                        if(!(qual = POLICYQUALINFO_new())) goto merr;
                        if(!sk_POLICYQUALINFO_push(pol->qualifiers, qual))
                                                                goto merr;
                        qual->pqualid = OBJ_nid2obj(NID_id_qt_cps);
                        qual->d.cpsuri = M_ASN1_IA5STRING_new();
                        if(!ASN1_STRING_set(qual->d.cpsuri, cnf->value,
                                                strlen(cnf->value))) goto merr;
                } else if(!name_cmp(cnf->name, "userNotice")) {
                        STACK_OF(CONF_VALUE) *unot;
                        if(*cnf->value != '@') {
                                X509V3err(X509V3_F_POLICY_SECTION,X509V3_R_EXPEC
                                X509V3_conf_err(cnf);
                                goto err;
                        }
                        unot = X509V3_get_section(ctx, cnf->value + 1);
                        if(!unot) {
                                X509V3err(X509V3_F_POLICY_SECTION,X509V3_R_INVAL

                                X509V3_conf_err(cnf);
                                goto err;
                        }
                        qual = notice_section(ctx, unot, ia5org);
                        X509V3_section_free(ctx, unot);
                        if(!qual) goto err;
                        if(!pol->qualifiers) pol->qualifiers =
                                                sk_POLICYQUALINFO_new_null();
                        if(!sk_POLICYQUALINFO_push(pol->qualifiers, qual))
                                                                goto merr;
                } else {
                        X509V3err(X509V3_F_POLICY_SECTION,X509V3_R_INVALID_OPTIO

                        X509V3_conf_err(cnf);
                        goto err;
                }
        }
        if(!pol->policyid) {
                X509V3err(X509V3_F_POLICY_SECTION,X509V3_R_NO_POLICY_IDENTIFIER)
                goto err;
        }
```

```
260         return pol;

262         merr:
263         X509V3err(X509V3_F_POLICY_SECTION,ERR_R_MALLOC_FAILURE);

265         err:
266         POLICYINFO_free(pol);
267         return NULL;


270 }

272 static POLICYQUALINFO *notice_section(X509V3_CTX *ctx,
273                                       STACK_OF(CONF_VALUE) *unot, int ia5org)
274 {
275         int i, ret;
276         CONF_VALUE *cnf;
277         USERNOTICE *not;
278         POLICYQUALINFO *qual;
279         if(!(qual = POLICYQUALINFO_new())) goto merr;
280         qual->pqualid = OBJ_nid2obj(NID_id_qt_unotice);
281         if(!(not = USERNOTICE_new())) goto merr;
282         qual->d.usernotice = not;
283         for(i = 0; i < sk_CONF_VALUE_num(unot); i++) {
284                 cnf = sk_CONF_VALUE_value(unot, i);
285                 if(!strcmp(cnf->name, "explicitText")) {
286                         not->exptext = M_ASN1_VISIBLESTRING_new();
287                         if(!ASN1_STRING_set(not->exptext, cnf->value,
288                                             strlen(cnf->value))) goto merr;
289                 } else if(!strcmp(cnf->name, "organization")) {
290                         NOTICEREF *nref;
291                         if(!not->noticeref) {
292                                 if(!(nref = NOTICEREF_new())) goto merr;
293                                 not->noticeref = nref;
294                         } else nref = not->noticeref;
295                         if(ia5org) nref->organization->type = V_ASN1_IA5STRING;
296                         else nref->organization->type = V_ASN1_VISIBLESTRING;
297                         if(!ASN1_STRING_set(nref->organization, cnf->value,
298                                             strlen(cnf->value))) goto merr;
299                 } else if(!strcmp(cnf->name, "noticeNumbers")) {
300                         NOTICEREF *nref;
301                         STACK_OF(CONF_VALUE) *nos;
302                         if(!not->noticeref) {
303                                 if(!(nref = NOTICEREF_new())) goto merr;
304                                 not->noticeref = nref;
305                         } else nref = not->noticeref;
306                         nos = X509V3_parse_list(cnf->value);
307                         if(!nos || !sk_CONF_VALUE_num(nos)) {
308                                 X509V3err(X509V3_F_NOTICE_SECTION,X509V3_R_INVAL
309                                 X509V3_conf_err(cnf);
310                                 goto err;
311                         }
312                         ret = nref_nos(nref->noticenos, nos);
313                         sk_CONF_VALUE_pop_free(nos, X509V3_conf_free);
314                         if (!ret)
315                                 goto err;
316                 } else {
317                         X509V3err(X509V3_F_NOTICE_SECTION,X509V3_R_INVALID_OPTIO
318                         X509V3_conf_err(cnf);
319                         goto err;
320                 }
321         }

323         if(not->noticeref &&
324              (!not->noticeref->noticenos || !not->noticeref->organization)) {
325                         X509V3err(X509V3_F_NOTICE_SECTION,X509V3_R_NEED_ORGANIZA
```



```
326                         goto err;
327         }

329         return qual;

331         merr:
332         X509V3err(X509V3_F_NOTICE_SECTION,ERR_R_MALLOC_FAILURE);

334         err:
335         POLICYQUALINFO_free(qual);
336         return NULL;
337 }

339 static int nref_nos(STACK_OF(ASN1_INTEGER) *nnums, STACK_OF(CONF_VALUE) *nos)
340 {
341         CONF_VALUE *cnf;
342         ASN1_INTEGER *aint;

344         int i;

346         for(i = 0; i < sk_CONF_VALUE_num(nos); i++) {
347                 cnf = sk_CONF_VALUE_value(nos, i);
348                 if(!(aint = s2i_ASN1_INTEGER(NULL, cnf->name))) {
349                         X509V3err(X509V3_F_NREF_NOS,X509V3_R_INVALID_NUMBER);
350                         goto err;
351                 }
352                 if(!sk_ASN1_INTEGER_push(nnums, aint)) goto merr;
353         }
354         return 1;

356         merr:
357         X509V3err(X509V3_F_NREF_NOS,ERR_R_MALLOC_FAILURE);

359         err:
360         sk_ASN1_INTEGER_pop_free(nnums, ASN1_STRING_free);
361         return 0;
362 }


365 static int i2r_certpol(X509V3_EXT_METHOD *method, STACK_OF(POLICYINFO) *pol,
366                 BIO *out, int indent)
367 {
368         int i;
369         POLICYINFO *pinfo;
370         /* First print out the policy OIDs */
371         for(i = 0; i < sk_POLICYINFO_num(pol); i++) {
372                 pinfo = sk_POLICYINFO_value(pol, i);
373                 BIO_printf(out, "%*sPolicy: ", indent, "");
374                 i2a_ASN1_OBJECT(out, pinfo->policyid);
375                 BIO_puts(out, "\n");
376                 if(pinfo->qualifiers)
377                         print_qualifiers(out, pinfo->qualifiers, indent + 2);
378         }
379         return 1;
380 }

382 static void print_qualifiers(BIO *out, STACK_OF(POLICYQUALINFO) *quals,
383                 int indent)
384 {
385         POLICYQUALINFO *qualinfo;
386         int i;
387         for(i = 0; i < sk_POLICYQUALINFO_num(quals); i++) {
388                 qualinfo = sk_POLICYQUALINFO_value(quals, i);
389                 switch(OBJ_obj2nid(qualinfo->pqualid))
390                 {
391                         case NID_id_qt_cps:
```

```
392                         BIO_printf(out, "%*sCPS: %s\n", indent, "",
393                                                 qualinfo->d.cpsuri->data);
394                         break;

396                         case NID_id_qt_unotice:
397                         BIO_printf(out, "%*sUser Notice:\n", indent, "");
398                         print_notice(out, qualinfo->d.usernotice, indent + 2);
399                         break;

401                         default:
402                         BIO_printf(out, "%*sUnknown Qualifier: ",
403                                                         indent + 2, "");

405                         i2a_ASN1_OBJECT(out, qualinfo->pqualid);
406                         BIO_puts(out, "\n");
407                         break;
408                 }
409         }
410 }

412 static void print_notice(BIO *out, USERNOTICE *notice, int indent)
413 {
414         int i;
415         if(notice->noticeref) {
416                 NOTICEREF *ref;
417                 ref = notice->noticeref;
418                 BIO_printf(out, "%*sOrganization: %s\n", indent, "",
419                                                 ref->organization->data);
420                 BIO_printf(out, "%*sNumber%s: ", indent, "",
421                                 sk_ASN1_INTEGER_num(ref->noticenos) > 1 ? "s" : "");
422                 for(i = 0; i < sk_ASN1_INTEGER_num(ref->noticenos); i++) {
423                         ASN1_INTEGER *num;
424                         char *tmp;
425                         num = sk_ASN1_INTEGER_value(ref->noticenos, i);
426                         if(i) BIO_puts(out, ", ");
427                         tmp = i2s_ASN1_INTEGER(NULL, num);
428                         BIO_puts(out, tmp);
429                         OPENSSL_free(tmp);
430                 }
431                 BIO_puts(out, "\n");
432         }
433         if(notice->exptext)
434                 BIO_printf(out, "%*sExplicit Text: %s\n", indent, "",
435                                                         notice->exptext->data);
436 }

438 void X509_POLICY_NODE_print(BIO *out, X509_POLICY_NODE *node, int indent)
439         {
440         const X509_POLICY_DATA *dat = node->data;

442         BIO_printf(out, "%*sPolicy: ", indent, "");

444         i2a_ASN1_OBJECT(out, dat->valid_policy);
445         BIO_puts(out, "\n");
446         BIO_printf(out, "%*s%s\n", indent + 2, "",
447                 node_data_critical(dat) ? "Critical" : "Non Critical");
448         if (dat->qualifier_set)
449                 print_qualifiers(out, dat->qualifier_set, indent + 2);
450         else
451                 BIO_printf(out, "%*sNo Qualifiers\n", indent + 2, "");
452         }


455 IMPLEMENT_STACK_OF(X509_POLICY_NODE)
456 IMPLEMENT_STACK_OF(X509_POLICY_DATA)
457 #endif /* ! codereview */
```

```
**********************************************************
   15736 Wed Aug 13 19:53:30 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_crld.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_crld.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

static void *v2i_crld(const X509V3_EXT_METHOD *method,
                      X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval);
static int i2r_crldp(const X509V3_EXT_METHOD *method, void *pcrldp, BIO *out,
                     int indent);

const X509V3_EXT_METHOD v3_crld =
        {
        NID_crl_distribution_points, 0, ASN1_ITEM_ref(CRL_DIST_POINTS),
        0,0,0,0,
        0,0,
        0,
        v2i_crld,
        i2r_crldp,0,
        NULL
        };

const X509V3_EXT_METHOD v3_freshest_crl =
        {
        NID_freshest_crl, 0, ASN1_ITEM_ref(CRL_DIST_POINTS),
        0,0,0,0,
        0,0,
        0,
        v2i_crld,
        i2r_crldp,0,
        NULL
        };

static STACK_OF(GENERAL_NAME) *gnames_from_sectname(X509V3_CTX *ctx, char *sect)
        {
        STACK_OF(CONF_VALUE) *gnsect;
        STACK_OF(GENERAL_NAME) *gens;
        if (*sect == '@')
                gnsect = X509V3_get_section(ctx, sect + 1);
        else
                gnsect = X509V3_parse_list(sect);
        if (!gnsect)
                {
                X509V3err(X509V3_F_GNAMES_FROM_SECTNAME,
                                                X509V3_R_SECTION_NOT_FOUND);
                return NULL;
                }
        gens = v2i_GENERAL_NAMES(NULL, ctx, gnsect);
        if (*sect == '@')
                X509V3_section_free(ctx, gnsect);
        else
                sk_CONF_VALUE_pop_free(gnsect, X509V3_conf_free);
        return gens;
        }

static int set_dist_point_name(DIST_POINT_NAME **pdp, X509V3_CTX *ctx,
                                                        CONF_VALUE *cnf)
        {
        STACK_OF(GENERAL_NAME) *fnm = NULL;
        STACK_OF(X509_NAME_ENTRY) *rnm = NULL;
        if (!strncmp(cnf->name, "fullname", 9))
                {
                fnm = gnames_from_sectname(ctx, cnf->value);
                if (!fnm)
                        goto err;
                }
        else if (!strcmp(cnf->name, "relativename"))
                {
```

```
128                 int ret;
129                 STACK_OF(CONF_VALUE) *dnsect;
130                 X509_NAME *nm;
131                 nm = X509_NAME_new();
132                 if (!nm)
133                         return -1;
134                 dnsect = X509V3_get_section(ctx, cnf->value);
135                 if (!dnsect)
136                         {
137                         X509V3err(X509V3_F_SET_DIST_POINT_NAME,
138                                                 X509V3_R_SECTION_NOT_FOUND);
139                         return -1;
140                         }
141                 ret = X509V3_NAME_from_section(nm, dnsect, MBSTRING_ASC);
142                 X509V3_section_free(ctx, dnsect);
143                 rnm = nm->entries;
144                 nm->entries = NULL;
145                 X509_NAME_free(nm);
146                 if (!ret || sk_X509_NAME_ENTRY_num(rnm) <= 0)
147                         goto err;
148                 /* Since its a name fragment can't have more than one
149                  * RDNSequence
150                  */
151                 if (sk_X509_NAME_ENTRY_value(rnm,
152                                 sk_X509_NAME_ENTRY_num(rnm) - 1)->set)
153                         {
154                         X509V3err(X509V3_F_SET_DIST_POINT_NAME,
155                                                 X509V3_R_INVALID_MULTIPLE_RDNS);
156                         goto err;
157                         }
158                 }
159         else
160                 return 0;

162         if (*pdp)
163                 {
164                 X509V3err(X509V3_F_SET_DIST_POINT_NAME,
165                                                 X509V3_R_DISTPOINT_ALREADY_SET);
166                 goto err;
167                 }

169         *pdp = DIST_POINT_NAME_new();
170         if (!*pdp)
171                 goto err;
172         if (fnm)
173                 {
174                 (*pdp)->type = 0;
175                 (*pdp)->name.fullname = fnm;
176                 }
177         else
178                 {
179                 (*pdp)->type = 1;
180                 (*pdp)->name.relativename = rnm;
181                 }

183         return 1;

185         err:
186         if (fnm)
187                 sk_GENERAL_NAME_pop_free(fnm, GENERAL_NAME_free);
188         if (rnm)
189                 sk_X509_NAME_ENTRY_pop_free(rnm, X509_NAME_ENTRY_free);
190         return -1;
191         }

193 static const BIT_STRING_BITNAME reason_flags[] = {
```



```
194 {0, "Unused", "unused"},
195 {1, "Key Compromise", "keyCompromise"},
196 {2, "CA Compromise", "CACompromise"},
197 {3, "Affiliation Changed", "affiliationChanged"},
198 {4, "Superseded", "superseded"},
199 {5, "Cessation Of Operation", "cessationOfOperation"},
200 {6, "Certificate Hold", "certificateHold"},
201 {7, "Privilege Withdrawn", "privilegeWithdrawn"},
202 {8, "AA Compromise", "AACompromise"},
203 {-1, NULL, NULL}
204 };

206 static int set_reasons(ASN1_BIT_STRING **preas, char *value)
207         {
208         STACK_OF(CONF_VALUE) *rsk = NULL;
209         const BIT_STRING_BITNAME *pbn;
210         const char *bnam;
211         int i, ret = 0;
212         rsk = X509V3_parse_list(value);
213         if (!rsk)
214                 return 0;
215         if (*preas)
216                 return 0;
217         for (i = 0; i < sk_CONF_VALUE_num(rsk); i++)
218                 {
219                 bnam = sk_CONF_VALUE_value(rsk, i)->name;
220                 if (!*preas)
221                         {
222                         *preas = ASN1_BIT_STRING_new();
223                         if (!*preas)
224                                 goto err;
225                         }
226                 for (pbn = reason_flags; pbn->lname; pbn++)
227                         {
228                         if (!strcmp(pbn->sname, bnam))
229                                 {
230                                 if (!ASN1_BIT_STRING_set_bit(*preas,
231                                                         pbn->bitnum, 1))
232                                         goto err;
233                                 break;
234                                 }
235                         }
236                 if (!pbn->lname)
237                         goto err;
238                 }
239         ret = 1;

241         err:
242         sk_CONF_VALUE_pop_free(rsk, X509V3_conf_free);
243         return ret;
244         }

246 static int print_reasons(BIO *out, const char *rname,
247                         ASN1_BIT_STRING *rflags, int indent)
248         {
249         int first = 1;
250         const BIT_STRING_BITNAME *pbn;
251         BIO_printf(out, "%*s%s:\n%*s", indent, "", rname, indent + 2, "");
252         for (pbn = reason_flags; pbn->lname; pbn++)
253                 {
254                 if (ASN1_BIT_STRING_get_bit(rflags, pbn->bitnum))
255                         {
256                         if (first)
257                                 first = 0;
258                         else
259                                 BIO_puts(out, ", ");
```

```
260                         BIO_puts(out, pbn->lname);
261                         }
262                 }
263         if (first)
264                 BIO_puts(out, "<EMPTY>\n");
265         else
266                 BIO_puts(out, "\n");
267         return 1;
268         }

270 static DIST_POINT *crldp_from_section(X509V3_CTX *ctx,
271                                       STACK_OF(CONF_VALUE) *nval)
272         {
273         int i;
274         CONF_VALUE *cnf;
275         DIST_POINT *point = NULL;
276         point = DIST_POINT_new();
277         if (!point)
278                 goto err;
279         for(i = 0; i < sk_CONF_VALUE_num(nval); i++)
280                 {
281                 int ret;
282                 cnf = sk_CONF_VALUE_value(nval, i);
283                 ret = set_dist_point_name(&point->distpoint, ctx, cnf);
284                 if (ret > 0)
285                         continue;
286                 if (ret < 0)
287                         goto err;
288                 if (!strcmp(cnf->name, "reasons"))
289                         {
290                         if (!set_reasons(&point->reasons, cnf->value))
291                                 goto err;
292                         }
293                 else if (!strcmp(cnf->name, "CRLissuer"))
294                         {
295                         point->CRLissuer =
296                                 gnames_from_sectname(ctx, cnf->value);
297                         if (!point->CRLissuer)
298                                 goto err;
299                         }
300                 }

302         return point;


305         err:
306         if (point)
307                 DIST_POINT_free(point);
308         return NULL;
309         }

311 static void *v2i_crld(const X509V3_EXT_METHOD *method,
312                       X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
313         {
314         STACK_OF(DIST_POINT) *crld = NULL;
315         GENERAL_NAMES *gens = NULL;
316         GENERAL_NAME *gen = NULL;
317         CONF_VALUE *cnf;
318         int i;
319         if(!(crld = sk_DIST_POINT_new_null())) goto merr;
320         for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
321                 DIST_POINT *point;
322                 cnf = sk_CONF_VALUE_value(nval, i);
323                 if (!cnf->value)
324                         {
325                         STACK_OF(CONF_VALUE) *dpsect;
```



```
326                         dpsect = X509V3_get_section(ctx, cnf->name);
327                         if (!dpsect)
328                                 goto err;
329                         point = crldp_from_section(ctx, dpsect);
330                         X509V3_section_free(ctx, dpsect);
331                         if (!point)
332                                 goto err;
333                         if(!sk_DIST_POINT_push(crld, point))
334                                 {
335                                 DIST_POINT_free(point);
336                                 goto merr;
337                                 }
338                         }
339                 else
340                         {
341                         if(!(gen = v2i_GENERAL_NAME(method, ctx, cnf)))
342                                 goto err;
343                         if(!(gens = GENERAL_NAMES_new()))
344                                 goto merr;
345                         if(!sk_GENERAL_NAME_push(gens, gen))
346                                 goto merr;
347                         gen = NULL;
348                         if(!(point = DIST_POINT_new()))
349                                 goto merr;
350                         if(!sk_DIST_POINT_push(crld, point))
351                                 {
352                                 DIST_POINT_free(point);
353                                 goto merr;
354                                 }
355                         if(!(point->distpoint = DIST_POINT_NAME_new()))
356                                 goto merr;
357                         point->distpoint->name.fullname = gens;
358                         point->distpoint->type = 0;
359                         gens = NULL;
360                         }
361         }
362         return crld;

364         merr:
365         X509V3err(X509V3_F_V2I_CRLD,ERR_R_MALLOC_FAILURE);
366         err:
367         GENERAL_NAME_free(gen);
368         GENERAL_NAMES_free(gens);
369         sk_DIST_POINT_pop_free(crld, DIST_POINT_free);
370         return NULL;
371 }

373 IMPLEMENT_STACK_OF(DIST_POINT)
374 IMPLEMENT_ASN1_SET_OF(DIST_POINT)

376 static int dpn_cb(int operation, ASN1_VALUE **pval, const ASN1_ITEM *it,
377                                                                 void *exarg)
378         {
379         DIST_POINT_NAME *dpn = (DIST_POINT_NAME *)*pval;

381         switch(operation)
382                 {
383                 case ASN1_OP_NEW_POST:
384                 dpn->dpname = NULL;
385                 break;

387                 case ASN1_OP_FREE_POST:
388                 if (dpn->dpname)
389                         X509_NAME_free(dpn->dpname);
390                 break;
391                 }
```

```
392         return 1;
393         }


396 ASN1_CHOICE_cb(DIST_POINT_NAME, dpn_cb) = {
397         ASN1_IMP_SEQUENCE_OF(DIST_POINT_NAME, name.fullname, GENERAL_NAME, 0),
398         ASN1_IMP_SET_OF(DIST_POINT_NAME, name.relativename, X509_NAME_ENTRY, 1)
399 } ASN1_CHOICE_END_cb(DIST_POINT_NAME, DIST_POINT_NAME, type)


402 IMPLEMENT_ASN1_FUNCTIONS(DIST_POINT_NAME)

404 ASN1_SEQUENCE(DIST_POINT) = {
405         ASN1_EXP_OPT(DIST_POINT, distpoint, DIST_POINT_NAME, 0),
406         ASN1_IMP_OPT(DIST_POINT, reasons, ASN1_BIT_STRING, 1),
407         ASN1_IMP_SEQUENCE_OF_OPT(DIST_POINT, CRLissuer, GENERAL_NAME, 2)
408 } ASN1_SEQUENCE_END(DIST_POINT)

410 IMPLEMENT_ASN1_FUNCTIONS(DIST_POINT)

412 ASN1_ITEM_TEMPLATE(CRL_DIST_POINTS) =
413         ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, CRLDistributionPoints, D
414 ASN1_ITEM_TEMPLATE_END(CRL_DIST_POINTS)

416 IMPLEMENT_ASN1_FUNCTIONS(CRL_DIST_POINTS)

418 ASN1_SEQUENCE(ISSUING_DIST_POINT) = {
419         ASN1_EXP_OPT(ISSUING_DIST_POINT, distpoint, DIST_POINT_NAME, 0),
420         ASN1_IMP_OPT(ISSUING_DIST_POINT, onlyuser, ASN1_FBOOLEAN, 1),
421         ASN1_IMP_OPT(ISSUING_DIST_POINT, onlyCA, ASN1_FBOOLEAN, 2),
422         ASN1_IMP_OPT(ISSUING_DIST_POINT, onlysomereasons, ASN1_BIT_STRING, 3),
423         ASN1_IMP_OPT(ISSUING_DIST_POINT, indirectCRL, ASN1_FBOOLEAN, 4),
424         ASN1_IMP_OPT(ISSUING_DIST_POINT, onlyattr, ASN1_FBOOLEAN, 5)
425 } ASN1_SEQUENCE_END(ISSUING_DIST_POINT)

427 IMPLEMENT_ASN1_FUNCTIONS(ISSUING_DIST_POINT)

429 static int i2r_idp(const X509V3_EXT_METHOD *method, void *pidp, BIO *out,
430                     int indent);
431 static void *v2i_idp(const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
432                      STACK_OF(CONF_VALUE) *nval);

434 const X509V3_EXT_METHOD v3_idp =
435         {
436         NID_issuing_distribution_point, X509V3_EXT_MULTILINE,
437         ASN1_ITEM_ref(ISSUING_DIST_POINT),
438         0,0,0,0,
439         0,0,
440         0,
441         v2i_idp,
442         i2r_idp,0,
443         NULL
444         };

446 static void *v2i_idp(const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
447                      STACK_OF(CONF_VALUE) *nval)
448         {
449         ISSUING_DIST_POINT *idp = NULL;
450         CONF_VALUE *cnf;
451         char *name, *val;
452         int i, ret;
453         idp = ISSUING_DIST_POINT_new();
454         if (!idp)
455                 goto merr;
456         for(i = 0; i < sk_CONF_VALUE_num(nval); i++)
457                 {
```



```
458                 cnf = sk_CONF_VALUE_value(nval, i);
459                 name = cnf->name;
460                 val = cnf->value;
461                 ret = set_dist_point_name(&idp->distpoint, ctx, cnf);
462                 if (ret > 0)
463                         continue;
464                 if (ret < 0)
465                         goto err;
466                 if (!strcmp(name, "onlyuser"))
467                         {
468                         if (!X509V3_get_value_bool(cnf, &idp->onlyuser))
469                                 goto err;
470                         }
471                 else if (!strcmp(name, "onlyCA"))
472                         {
473                         if (!X509V3_get_value_bool(cnf, &idp->onlyCA))
474                                 goto err;
475                         }
476                 else if (!strcmp(name, "onlyAA"))
477                         {
478                         if (!X509V3_get_value_bool(cnf, &idp->onlyattr))
479                                 goto err;
480                         }
481                 else if (!strcmp(name, "indirectCRL"))
482                         {
483                         if (!X509V3_get_value_bool(cnf, &idp->indirectCRL))
484                                 goto err;
485                         }
486                 else if (!strcmp(name, "onlysomereasons"))
487                         {
488                         if (!set_reasons(&idp->onlysomereasons, val))
489                                 goto err;
490                         }
491                 else
492                         {
493                         X509V3err(X509V3_F_V2I_IDP, X509V3_R_INVALID_NAME);
494                         X509V3_conf_err(cnf);
495                         goto err;
496                         }
497                 }
498         return idp;

500         merr:
501         X509V3err(X509V3_F_V2I_IDP,ERR_R_MALLOC_FAILURE);
502         err:
503         ISSUING_DIST_POINT_free(idp);
504         return NULL;
505         }

507 static int print_gens(BIO *out, STACK_OF(GENERAL_NAME) *gens, int indent)
508         {
509         int i;
510         for (i = 0; i < sk_GENERAL_NAME_num(gens); i++)
511                 {
512                 BIO_printf(out, "%*s", indent + 2, "");
513                 GENERAL_NAME_print(out, sk_GENERAL_NAME_value(gens, i));
514                 BIO_puts(out, "\n");
515                 }
516         return 1;
517         }

519 static int print_distpoint(BIO *out, DIST_POINT_NAME *dpn, int indent)
520         {
521         if (dpn->type == 0)
522                 {
523                 BIO_printf(out, "%*sFull Name:\n", indent, "");
```

```
524                 print_gens(out, dpn->name.fullname, indent);
525                 }
526         else
527                 {
528                 X509_NAME ntmp;
529                 ntmp.entries = dpn->name.relativename;
530                 BIO_printf(out, "%*sRelative Name:\n%*s",
531                                                 indent, "", indent + 2, "");
532                 X509_NAME_print_ex(out, &ntmp, 0, XN_FLAG_ONELINE);
533                 BIO_puts(out, "\n");
534                 }
535         return 1;
536         }

538 static int i2r_idp(const X509V3_EXT_METHOD *method, void *pidp, BIO *out,
539                    int indent)
540         {
541         ISSUING_DIST_POINT *idp = pidp;
542         if (idp->distpoint)
543                 print_distpoint(out, idp->distpoint, indent);
544         if (idp->onlyuser > 0)
545                 BIO_printf(out, "%*sOnly User Certificates\n", indent, "");
546         if (idp->onlyCA > 0)
547                 BIO_printf(out, "%*sOnly CA Certificates\n", indent, "");
548         if (idp->indirectCRL > 0)
549                 BIO_printf(out, "%*sIndirect CRL\n", indent, "");
550         if (idp->onlysomereasons)
551                 print_reasons(out, "Only Some Reasons",
552                               idp->onlysomereasons, indent);
553         if (idp->onlyattr > 0)
554                 BIO_printf(out, "%*sOnly Attribute Certificates\n", indent, "");
555         if (!idp->distpoint && (idp->onlyuser <= 0) && (idp->onlyCA <= 0)
556                 && (idp->indirectCRL <= 0) && !idp->onlysomereasons
557                 && (idp->onlyattr <= 0))
558                 BIO_printf(out, "%*s<EMPTY>\n", indent, "");

560         return 1;
561         }

563 static int i2r_crldp(const X509V3_EXT_METHOD *method, void *pcrldp, BIO *out,
564                      int indent)
565         {
566         STACK_OF(DIST_POINT) *crld = pcrldp;
567         DIST_POINT *point;
568         int i;
569         for(i = 0; i < sk_DIST_POINT_num(crld); i++)
570                 {
571                 BIO_puts(out, "\n");
572                 point = sk_DIST_POINT_value(crld, i);
573                 if(point->distpoint)
574                         print_distpoint(out, point->distpoint, indent);
575                 if(point->reasons)
576                         print_reasons(out, "Reasons", point->reasons,
577                                                                 indent);
578                 if(point->CRLissuer)
579                         {
580                         BIO_printf(out, "%*sCRL Issuer:\n", indent, "");
581                         print_gens(out, point->CRLissuer, indent);
582                         }
583                 }
584         return 1;
585         }

587 int DIST_POINT_set_dpname(DIST_POINT_NAME *dpn, X509_NAME *iname)
588         {
589         int i;
```



```
590         STACK_OF(X509_NAME_ENTRY) *frag;
591         X509_NAME_ENTRY *ne;
592         if (!dpn || (dpn->type != 1))
593                 return 1;
594         frag = dpn->name.relativename;
595         dpn->dpname = X509_NAME_dup(iname);
596         if (!dpn->dpname)
597                 return 0;
598         for (i = 0; i < sk_X509_NAME_ENTRY_num(frag); i++)
599                 {
600                 ne = sk_X509_NAME_ENTRY_value(frag, i);
601                 if (!X509_NAME_add_entry(dpn->dpname, ne, -1, i ? 0 : 1))
602                         {
603                         X509_NAME_free(dpn->dpname);
604                         dpn->dpname = NULL;
605                         return 0;
606                         }
607                 }
608         /* generate cached encoding of name */
609         if (i2d_X509_NAME(dpn->dpname, NULL) < 0)
610                 {
611                 X509_NAME_free(dpn->dpname);
612                 dpn->dpname = NULL;
613                 return 0;
614                 }
615         return 1;
616         }
617 #endif /* ! codereview */
```

```
**********************************************************
    4046 Wed Aug 13 19:53:30 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_enum.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* v3_enum.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509v3.h>
```



```
static ENUMERATED_NAMES crl_reasons[] = {
{CRL_REASON_UNSPECIFIED,         "Unspecified", "unspecified"},
{CRL_REASON_KEY_COMPROMISE,      "Key Compromise", "keyCompromise"},
{CRL_REASON_CA_COMPROMISE,       "CA Compromise", "CACompromise"},
{CRL_REASON_AFFILIATION_CHANGED, "Affiliation Changed", "affiliationChanged"},
{CRL_REASON_SUPERSEDED,          "Superseded", "superseded"},
{CRL_REASON_CESSATION_OF_OPERATION,
                                 "Cessation Of Operation", "cessationOfOperation"},
{CRL_REASON_CERTIFICATE_HOLD,    "Certificate Hold", "certificateHold"},
{CRL_REASON_REMOVE_FROM_CRL,     "Remove From CRL", "removeFromCRL"},
{CRL_REASON_PRIVILEGE_WITHDRAWN, "Privilege Withdrawn", "privilegeWithdrawn"},
{CRL_REASON_AA_COMPROMISE,       "AA Compromise", "AACompromise"},
{-1, NULL, NULL}
};

const X509V3_EXT_METHOD v3_crl_reason = {
NID_crl_reason, 0, ASN1_ITEM_ref(ASN1_ENUMERATED),
0,0,0,0,
(X509V3_EXT_I2S)i2s_ASN1_ENUMERATED_TABLE,
0,
0,0,0,0,
crl_reasons};


char *i2s_ASN1_ENUMERATED_TABLE(X509V3_EXT_METHOD *method,
             ASN1_ENUMERATED *e)
{
        ENUMERATED_NAMES *enam;
        long strval;
        strval = ASN1_ENUMERATED_get(e);
        for(enam = method->usr_data; enam->lname; enam++) {
                if(strval == enam->bitnum) return BUF_strdup(enam->lname);
        }
        return i2s_ASN1_ENUMERATED(method, e);
}
#endif /* ! codereview */
```

```
**********************************************************
    4962 Wed Aug 13 19:53:30 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_extku.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_extku.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1t.h>
#include <openssl/conf.h>
#include <openssl/x509v3.h>

static void *v2i_EXTENDED_KEY_USAGE(const X509V3_EXT_METHOD *method,
                                    X509V3_CTX *ctx,
                                    STACK_OF(CONF_VALUE) *nval);
static STACK_OF(CONF_VALUE) *i2v_EXTENDED_KEY_USAGE(const X509V3_EXT_METHOD *met
                void *eku, STACK_OF(CONF_VALUE) *extlist);

const X509V3_EXT_METHOD v3_ext_ku = {
        NID_ext_key_usage, 0,
        ASN1_ITEM_ref(EXTENDED_KEY_USAGE),
        0,0,0,0,
        0,0,
        i2v_EXTENDED_KEY_USAGE,
        v2i_EXTENDED_KEY_USAGE,
        0,0,
        NULL
};

/* NB OCSP acceptable responses also is a SEQUENCE OF OBJECT */
const X509V3_EXT_METHOD v3_ocsp_accresp = {
        NID_id_pkix_OCSP_acceptableResponses, 0,
        ASN1_ITEM_ref(EXTENDED_KEY_USAGE),
        0,0,0,0,
        0,0,
        i2v_EXTENDED_KEY_USAGE,
        v2i_EXTENDED_KEY_USAGE,
        0,0,
        NULL
};

ASN1_ITEM_TEMPLATE(EXTENDED_KEY_USAGE) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, EXTENDED_KEY_USAGE, ASN1
ASN1_ITEM_TEMPLATE_END(EXTENDED_KEY_USAGE)

IMPLEMENT_ASN1_FUNCTIONS(EXTENDED_KEY_USAGE)

static STACK_OF(CONF_VALUE) *
  i2v_EXTENDED_KEY_USAGE(const X509V3_EXT_METHOD *method, void *a,
                         STACK_OF(CONF_VALUE) *ext_list)
{
        EXTENDED_KEY_USAGE *eku = a;
        int i;
        ASN1_OBJECT *obj;
        char obj_tmp[80];
        for(i = 0; i < sk_ASN1_OBJECT_num(eku); i++) {
                obj = sk_ASN1_OBJECT_value(eku, i);
                i2t_ASN1_OBJECT(obj_tmp, 80, obj);
                X509V3_add_value(NULL, obj_tmp, &ext_list);
        }
        return ext_list;
}

static void *v2i_EXTENDED_KEY_USAGE(const X509V3_EXT_METHOD *method,
                                    X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
{
        EXTENDED_KEY_USAGE *extku;
        char *extval;
        ASN1_OBJECT *objtmp;
        CONF_VALUE *val;
        int i;

        if(!(extku = sk_ASN1_OBJECT_new_null())) {
                X509V3err(X509V3_F_V2I_EXTENDED_KEY_USAGE,ERR_R_MALLOC_FAILURE);
```

```
128                 return NULL;
129         }

131         for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
132                 val = sk_CONF_VALUE_value(nval, i);
133                 if(val->value) extval = val->value;
134                 else extval = val->name;
135                 if(!(objtmp = OBJ_txt2obj(extval, 0))) {
136                         sk_ASN1_OBJECT_pop_free(extku, ASN1_OBJECT_free);
137                         X509V3err(X509V3_F_V2I_EXTENDED_KEY_USAGE,X509V3_R_INVAL
138                         X509V3_conf_err(val);
139                         return NULL;
140                 }
141                 sk_ASN1_OBJECT_push(extku, objtmp);
142         }
143         return extku;
144 }
145 #endif /* ! codereview */
```

```
**********************************************************
    6952 Wed Aug 13 19:53:30 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_genn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_genn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1t.h>
#include <openssl/conf.h>
#include <openssl/x509v3.h>

ASN1_SEQUENCE(OTHERNAME) = {
        ASN1_SIMPLE(OTHERNAME, type_id, ASN1_OBJECT),
        /* Maybe have a true ANY DEFINED BY later */
        ASN1_EXP(OTHERNAME, value, ASN1_ANY, 0)
} ASN1_SEQUENCE_END(OTHERNAME)

IMPLEMENT_ASN1_FUNCTIONS(OTHERNAME)

ASN1_SEQUENCE(EDIPARTYNAME) = {
        ASN1_IMP_OPT(EDIPARTYNAME, nameAssigner, DIRECTORYSTRING, 0),
        ASN1_IMP_OPT(EDIPARTYNAME, partyName, DIRECTORYSTRING, 1)
} ASN1_SEQUENCE_END(EDIPARTYNAME)

IMPLEMENT_ASN1_FUNCTIONS(EDIPARTYNAME)

ASN1_CHOICE(GENERAL_NAME) = {
        ASN1_IMP(GENERAL_NAME, d.otherName, OTHERNAME, GEN_OTHERNAME),
        ASN1_IMP(GENERAL_NAME, d.rfc822Name, ASN1_IA5STRING, GEN_EMAIL),
        ASN1_IMP(GENERAL_NAME, d.dNSName, ASN1_IA5STRING, GEN_DNS),
        /* Don't decode this */
        ASN1_IMP(GENERAL_NAME, d.x400Address, ASN1_SEQUENCE, GEN_X400),
        /* X509_NAME is a CHOICE type so use EXPLICIT */
        ASN1_EXP(GENERAL_NAME, d.directoryName, X509_NAME, GEN_DIRNAME),
        ASN1_IMP(GENERAL_NAME, d.ediPartyName, EDIPARTYNAME, GEN_EDIPARTY),
        ASN1_IMP(GENERAL_NAME, d.uniformResourceIdentifier, ASN1_IA5STRING, GEN_
        ASN1_IMP(GENERAL_NAME, d.iPAddress, ASN1_OCTET_STRING, GEN_IPADD),
        ASN1_IMP(GENERAL_NAME, d.registeredID, ASN1_OBJECT, GEN_RID)
} ASN1_CHOICE_END(GENERAL_NAME)

IMPLEMENT_ASN1_FUNCTIONS(GENERAL_NAME)

ASN1_ITEM_TEMPLATE(GENERAL_NAMES) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, GeneralNames, GENERAL_NA
ASN1_ITEM_TEMPLATE_END(GENERAL_NAMES)

IMPLEMENT_ASN1_FUNCTIONS(GENERAL_NAMES)

GENERAL_NAME *GENERAL_NAME_dup(GENERAL_NAME *a)
        {
        return (GENERAL_NAME *) ASN1_dup((i2d_of_void *) i2d_GENERAL_NAME,
                                         (d2i_of_void *) d2i_GENERAL_NAME,
                                         (char *) a);
        }

/* Returns 0 if they are equal, != 0 otherwise. */
int GENERAL_NAME_cmp(GENERAL_NAME *a, GENERAL_NAME *b)
        {
        int result = -1;

        if (!a || !b || a->type != b->type) return -1;
        switch(a->type)
                {
        case GEN_X400:
        case GEN_EDIPARTY:
                result = ASN1_TYPE_cmp(a->d.other, b->d.other);
                break;

        case GEN_OTHERNAME:
                result = OTHERNAME_cmp(a->d.otherName, b->d.otherName);
                break;

        case GEN_EMAIL:
```

```
128         case GEN_DNS:
129         case GEN_URI:
130                 result = ASN1_STRING_cmp(a->d.ia5, b->d.ia5);
131                 break;

133         case GEN_DIRNAME:
134                 result = X509_NAME_cmp(a->d.dirn, b->d.dirn);
135                 break;

137         case GEN_IPADD:
138                 result = ASN1_OCTET_STRING_cmp(a->d.ip, b->d.ip);
139                 break;

141         case GEN_RID:
142                 result = OBJ_cmp(a->d.rid, b->d.rid);
143                 break;
144                 }
145         return result;
146         }

148 /* Returns 0 if they are equal, != 0 otherwise. */
149 int OTHERNAME_cmp(OTHERNAME *a, OTHERNAME *b)
150         {
151         int result = -1;

153         if (!a || !b) return -1;
154         /* Check their type first. */
155         if ((result = OBJ_cmp(a->type_id, b->type_id)) != 0)
156                 return result;
157         /* Check the value. */
158         result = ASN1_TYPE_cmp(a->value, b->value);
159         return result;
160         }

162 void GENERAL_NAME_set0_value(GENERAL_NAME *a, int type, void *value)
163         {
164         switch(type)
165                 {
166         case GEN_X400:
167         case GEN_EDIPARTY:
168                 a->d.other = value;
169                 break;

171         case GEN_OTHERNAME:
172                 a->d.otherName = value;
173                 break;

175         case GEN_EMAIL:
176         case GEN_DNS:
177         case GEN_URI:
178                 a->d.ia5 = value;
179                 break;

181         case GEN_DIRNAME:
182                 a->d.dirn = value;
183                 break;

185         case GEN_IPADD:
186                 a->d.ip = value;
187                 break;

189         case GEN_RID:
190                 a->d.rid = value;
191                 break;
192                 }
193         a->type = type;
```



```
194         }

196 void *GENERAL_NAME_get0_value(GENERAL_NAME *a, int *ptype)
197         {
198         if (ptype)
199                 *ptype = a->type;
200         switch(a->type)
201                 {
202         case GEN_X400:
203         case GEN_EDIPARTY:
204                 return a->d.other;

206         case GEN_OTHERNAME:
207                 return a->d.otherName;

209         case GEN_EMAIL:
210         case GEN_DNS:
211         case GEN_URI:
212                 return a->d.ia5;

214         case GEN_DIRNAME:
215                 return a->d.dirn;

217         case GEN_IPADD:
218                 return a->d.ip;

220         case GEN_RID:
221                 return a->d.rid;

223         default:
224                 return NULL;
225                 }
226         }

228 int GENERAL_NAME_set0_othername(GENERAL_NAME *gen,
229                                 ASN1_OBJECT *oid, ASN1_TYPE *value)
230         {
231         OTHERNAME *oth;
232         oth = OTHERNAME_new();
233         if (!oth)
234                 return 0;
235         oth->type_id = oid;
236         oth->value = value;
237         GENERAL_NAME_set0_value(gen, GEN_OTHERNAME, oth);
238         return 1;
239         }

241 int GENERAL_NAME_get0_otherName(GENERAL_NAME *gen,
242                                 ASN1_OBJECT **poid, ASN1_TYPE **pvalue)
243         {
244         if (gen->type != GEN_OTHERNAME)
245                 return 0;
246         if (poid)
247                 *poid = gen->d.otherName->type_id;
248         if (pvalue)
249                 *pvalue = gen->d.otherName->value;
250         return 1;
251         }
252 #endif /* ! codereview */
```

```
**********************************************************
    4341 Wed Aug 13 19:53:30 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_ia5.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_ia5.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/conf.h>
#include <openssl/x509v3.h>

static char *i2s_ASN1_IA5STRING(X509V3_EXT_METHOD *method, ASN1_IA5STRING *ia5);
static ASN1_IA5STRING *s2i_ASN1_IA5STRING(X509V3_EXT_METHOD *method, X509V3_CTX
const X509V3_EXT_METHOD v3_ns_ia5_list[] = {
EXT_IA5STRING(NID_netscape_base_url),
EXT_IA5STRING(NID_netscape_revocation_url),
EXT_IA5STRING(NID_netscape_ca_revocation_url),
EXT_IA5STRING(NID_netscape_renewal_url),
EXT_IA5STRING(NID_netscape_ca_policy_url),
EXT_IA5STRING(NID_netscape_ssl_server_name),
EXT_IA5STRING(NID_netscape_comment),
EXT_END
};


static char *i2s_ASN1_IA5STRING(X509V3_EXT_METHOD *method,
             ASN1_IA5STRING *ia5)
{
        char *tmp;
        if(!ia5 || !ia5->length) return NULL;
        if(!(tmp = OPENSSL_malloc(ia5->length + 1))) {
                X509V3err(X509V3_F_I2S_ASN1_IA5STRING,ERR_R_MALLOC_FAILURE);
                return NULL;
        }
        memcpy(tmp, ia5->data, ia5->length);
        tmp[ia5->length] = 0;
        return tmp;
}

static ASN1_IA5STRING *s2i_ASN1_IA5STRING(X509V3_EXT_METHOD *method,
             X509V3_CTX *ctx, char *str)
{
        ASN1_IA5STRING *ia5;
        if(!str) {
                X509V3err(X509V3_F_S2I_ASN1_IA5STRING,X509V3_R_INVALID_NULL_ARGU
                return NULL;
        }
        if(!(ia5 = M_ASN1_IA5STRING_new())) goto err;
        if(!ASN1_STRING_set((ASN1_STRING *)ia5, (unsigned char*)str,
                                strlen(str))) {
                M_ASN1_IA5STRING_free(ia5);
                goto err;
        }
#ifdef CHARSET_EBCDIC
        ebcdic2ascii(ia5->data, ia5->data, ia5->length);
#endif /*CHARSET_EBCDIC*/
        return ia5;
        err:
        X509V3err(X509V3_F_S2I_ASN1_IA5STRING,ERR_R_MALLOC_FAILURE);
        return NULL;
}
#endif /* ! codereview */
```

```
**********************************************************
    6679 Wed Aug 13 19:53:30 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_info.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_info.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

static STACK_OF(CONF_VALUE) *i2v_AUTHORITY_INFO_ACCESS(X509V3_EXT_METHOD *method
                                AUTHORITY_INFO_ACCESS *ainfo,
                                        STACK_OF(CONF_VALUE) *ret);
static AUTHORITY_INFO_ACCESS *v2i_AUTHORITY_INFO_ACCESS(X509V3_EXT_METHOD *metho
                                 X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval);

const X509V3_EXT_METHOD v3_info =
{ NID_info_access, X509V3_EXT_MULTILINE, ASN1_ITEM_ref(AUTHORITY_INFO_ACCESS),
0,0,0,0,
0,0,
(X509V3_EXT_I2V)i2v_AUTHORITY_INFO_ACCESS,
(X509V3_EXT_V2I)v2i_AUTHORITY_INFO_ACCESS,
0,0,
NULL};

const X509V3_EXT_METHOD v3_sinfo =
{ NID_sinfo_access, X509V3_EXT_MULTILINE, ASN1_ITEM_ref(AUTHORITY_INFO_ACCESS),
0,0,0,0,
0,0,
(X509V3_EXT_I2V)i2v_AUTHORITY_INFO_ACCESS,
(X509V3_EXT_V2I)v2i_AUTHORITY_INFO_ACCESS,
0,0,
NULL};

ASN1_SEQUENCE(ACCESS_DESCRIPTION) = {
        ASN1_SIMPLE(ACCESS_DESCRIPTION, method, ASN1_OBJECT),
        ASN1_SIMPLE(ACCESS_DESCRIPTION, location, GENERAL_NAME)
} ASN1_SEQUENCE_END(ACCESS_DESCRIPTION)

IMPLEMENT_ASN1_FUNCTIONS(ACCESS_DESCRIPTION)

ASN1_ITEM_TEMPLATE(AUTHORITY_INFO_ACCESS) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, GeneralNames, ACCESS_DES
ASN1_ITEM_TEMPLATE_END(AUTHORITY_INFO_ACCESS)

IMPLEMENT_ASN1_FUNCTIONS(AUTHORITY_INFO_ACCESS)

static STACK_OF(CONF_VALUE) *i2v_AUTHORITY_INFO_ACCESS(X509V3_EXT_METHOD *method
                                AUTHORITY_INFO_ACCESS *ainfo,
                                        STACK_OF(CONF_VALUE) *ret)
{
        ACCESS_DESCRIPTION *desc;
        int i,nlen;
        char objtmp[80], *ntmp;
        CONF_VALUE *vtmp;
        for(i = 0; i < sk_ACCESS_DESCRIPTION_num(ainfo); i++) {
                desc = sk_ACCESS_DESCRIPTION_value(ainfo, i);
                ret = i2v_GENERAL_NAME(method, desc->location, ret);
                if(!ret) break;
                vtmp = sk_CONF_VALUE_value(ret, i);
                i2t_ASN1_OBJECT(objtmp, sizeof objtmp, desc->method);
                nlen = strlen(objtmp) + strlen(vtmp->name) + 5;
                ntmp = OPENSSL_malloc(nlen);
                if(!ntmp) {
                        X509V3err(X509V3_F_I2V_AUTHORITY_INFO_ACCESS,
                                        ERR_R_MALLOC_FAILURE);
                        return NULL;
                }
                BUF_strlcpy(ntmp, objtmp, nlen);
                BUF_strlcat(ntmp, " - ", nlen);
                BUF_strlcat(ntmp, vtmp->name, nlen);
                OPENSSL_free(vtmp->name);
```

```
128                 vtmp->name = ntmp;

130         }
131         if(!ret) return sk_CONF_VALUE_new_null();
132         return ret;
133 }

135 static AUTHORITY_INFO_ACCESS *v2i_AUTHORITY_INFO_ACCESS(X509V3_EXT_METHOD *metho
136                                 X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
137 {
138         AUTHORITY_INFO_ACCESS *ainfo = NULL;
139         CONF_VALUE *cnf, ctmp;
140         ACCESS_DESCRIPTION *acc;
141         int i, objlen;
142         char *objtmp, *ptmp;
143         if(!(ainfo = sk_ACCESS_DESCRIPTION_new_null())) {
144                 X509V3err(X509V3_F_V2I_AUTHORITY_INFO_ACCESS,ERR_R_MALLOC_FAILUR
145                 return NULL;
146         }
147         for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
148                 cnf = sk_CONF_VALUE_value(nval, i);
149                 if(!(acc = ACCESS_DESCRIPTION_new())
150                         || !sk_ACCESS_DESCRIPTION_push(ainfo, acc)) {
151                         X509V3err(X509V3_F_V2I_AUTHORITY_INFO_ACCESS,ERR_R_MALLO
152                         goto err;
153                 }
154                 ptmp = strchr(cnf->name, ';');
155                 if(!ptmp) {
156                         X509V3err(X509V3_F_V2I_AUTHORITY_INFO_ACCESS,X509V3_R_IN
157                         goto err;
158                 }
159                 objlen = ptmp - cnf->name;
160                 ctmp.name = ptmp + 1;
161                 ctmp.value = cnf->value;
162                 if(!v2i_GENERAL_NAME_ex(acc->location, method, ctx, &ctmp, 0))
163                                                                 goto err;
164                 if(!(objtmp = OPENSSL_malloc(objlen + 1))) {
165                         X509V3err(X509V3_F_V2I_AUTHORITY_INFO_ACCESS,ERR_R_MALLO
166                         goto err;
167                 }
168                 strncpy(objtmp, cnf->name, objlen);
169                 objtmp[objlen] = 0;
170                 acc->method = OBJ_txt2obj(objtmp, 0);
171                 if(!acc->method) {
172                         X509V3err(X509V3_F_V2I_AUTHORITY_INFO_ACCESS,X509V3_R_BA
173                         ERR_add_error_data(2, "value=", objtmp);
174                         OPENSSL_free(objtmp);
175                         goto err;
176                 }
177                 OPENSSL_free(objtmp);

179         }
180         return ainfo;
181         err:
182         sk_ACCESS_DESCRIPTION_pop_free(ainfo, ACCESS_DESCRIPTION_free);
183         return NULL;
184 }

186 int i2a_ACCESS_DESCRIPTION(BIO *bp, ACCESS_DESCRIPTION* a)
187         {
188         i2a_ASN1_OBJECT(bp, a->method);
189 #ifdef UNDEF
190         i2a_GENERAL_NAME(bp, a->location);
191 #endif
192         return 2;
193         }
```



```
194 #endif /* ! codereview */
```

```
**********************************************************
    3440 Wed Aug 13 19:53:31 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_int.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_int.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509v3.h>
```



```
const X509V3_EXT_METHOD v3_crl_num = {
        NID_crl_number, 0, ASN1_ITEM_ref(ASN1_INTEGER),
        0,0,0,0,
        (X509V3_EXT_I2S)i2s_ASN1_INTEGER,
        0,
        0,0,0,0, NULL};

const X509V3_EXT_METHOD v3_delta_crl = {
        NID_delta_crl, 0, ASN1_ITEM_ref(ASN1_INTEGER),
        0,0,0,0,
        (X509V3_EXT_I2S)i2s_ASN1_INTEGER,
        0,
        0,0,0,0, NULL};

static void * s2i_asn1_int(X509V3_EXT_METHOD *meth, X509V3_CTX *ctx, char *value
        {
        return s2i_ASN1_INTEGER(meth, value);
        }

const X509V3_EXT_METHOD v3_inhibit_anyp = {
        NID_inhibit_any_policy, 0, ASN1_ITEM_ref(ASN1_INTEGER),
        0,0,0,0,
        (X509V3_EXT_I2S)i2s_ASN1_INTEGER,
        (X509V3_EXT_S2I)s2i_asn1_int,
        0,0,0,0, NULL};
#endif /* ! codereview */
```

```
**********************************************************
    9349 Wed Aug 13 19:53:31 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_lib.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* X509 v3 extension utilities */

#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/conf.h>
#include <openssl/x509v3.h>

#include "ext_dat.h"

static STACK_OF(X509V3_EXT_METHOD) *ext_list = NULL;

static int ext_cmp(const X509V3_EXT_METHOD * const *a,
                   const X509V3_EXT_METHOD * const *b);
static void ext_list_free(X509V3_EXT_METHOD *ext);

int X509V3_EXT_add(X509V3_EXT_METHOD *ext)
{
        if(!ext_list && !(ext_list = sk_X509V3_EXT_METHOD_new(ext_cmp))) {
                X509V3err(X509V3_F_X509V3_EXT_ADD,ERR_R_MALLOC_FAILURE);
                return 0;
        }
        if(!sk_X509V3_EXT_METHOD_push(ext_list, ext)) {
                X509V3err(X509V3_F_X509V3_EXT_ADD,ERR_R_MALLOC_FAILURE);
                return 0;
        }
        return 1;
}

static int ext_cmp(const X509V3_EXT_METHOD * const *a,
                   const X509V3_EXT_METHOD * const *b)
{
        return ((*a)->ext_nid - (*b)->ext_nid);
}

DECLARE_OBJ_BSEARCH_CMP_FN(const X509V3_EXT_METHOD *, const X509V3_EXT_METHOD *,
                           ext);
IMPLEMENT_OBJ_BSEARCH_CMP_FN(const X509V3_EXT_METHOD *,
                             const X509V3_EXT_METHOD *, ext);

const X509V3_EXT_METHOD *X509V3_EXT_get_nid(int nid)
{
        X509V3_EXT_METHOD tmp;
        const X509V3_EXT_METHOD *t = &tmp, * const *ret;
        int idx;
        if(nid < 0) return NULL;
        tmp.ext_nid = nid;
        ret = OBJ_bsearch_ext(&t, standard_exts, STANDARD_EXTENSION_COUNT);
        if(ret) return *ret;
        if(!ext_list) return NULL;
        idx = sk_X509V3_EXT_METHOD_find(ext_list, &tmp);
        if(idx == -1) return NULL;
        return sk_X509V3_EXT_METHOD_value(ext_list, idx);
}

const X509V3_EXT_METHOD *X509V3_EXT_get(X509_EXTENSION *ext)
{
        int nid;
        if((nid = OBJ_obj2nid(ext->object)) == NID_undef) return NULL;
        return X509V3_EXT_get_nid(nid);
}


int X509V3_EXT_add_list(X509V3_EXT_METHOD *extlist)
{
        for(;extlist->ext_nid!=-1;extlist++)
                        if(!X509V3_EXT_add(extlist)) return 0;
        return 1;
}

int X509V3_EXT_add_alias(int nid_to, int nid_from)
```

```
128 {
129         const X509V3_EXT_METHOD *ext;
130         X509V3_EXT_METHOD *tmpext;

132         if(!(ext = X509V3_EXT_get_nid(nid_from))) {
133                 X509V3err(X509V3_F_X509V3_EXT_ADD_ALIAS,X509V3_R_EXTENSION_NOT_F
134                 return 0;
135         }
136         if(!(tmpext = (X509V3_EXT_METHOD *)OPENSSL_malloc(sizeof(X509V3_EXT_METH
137                 X509V3err(X509V3_F_X509V3_EXT_ADD_ALIAS,ERR_R_MALLOC_FAILURE);
138                 return 0;
139         }
140         *tmpext = *ext;
141         tmpext->ext_nid = nid_to;
142         tmpext->ext_flags |= X509V3_EXT_DYNAMIC;
143         return X509V3_EXT_add(tmpext);
144 }

146 void X509V3_EXT_cleanup(void)
147 {
148         sk_X509V3_EXT_METHOD_pop_free(ext_list, ext_list_free);
149         ext_list = NULL;
150 }

152 static void ext_list_free(X509V3_EXT_METHOD *ext)
153 {
154         if(ext->ext_flags & X509V3_EXT_DYNAMIC) OPENSSL_free(ext);
155 }

157 /* Legacy function: we don't need to add standard extensions
158  * any more because they are now kept in ext_dat.h.
159  */

161 int X509V3_add_standard_extensions(void)
162 {
163         return 1;
164 }

166 /* Return an extension internal structure */

168 void *X509V3_EXT_d2i(X509_EXTENSION *ext)
169 {
170         const X509V3_EXT_METHOD *method;
171         const unsigned char *p;

173         if(!(method = X509V3_EXT_get(ext))) return NULL;
174         p = ext->value->data;
175         if(method->it) return ASN1_item_d2i(NULL, &p, ext->value->length, ASN1_I
176         return method->d2i(NULL, &p, ext->value->length);
177 }

179 /* Get critical flag and decoded version of extension from a NID.
180  * The "idx" variable returns the last found extension and can
181  * be used to retrieve multiple extensions of the same NID.
182  * However multiple extensions with the same NID is usually
183  * due to a badly encoded certificate so if idx is NULL we
184  * choke if multiple extensions exist.
185  * The "crit" variable is set to the critical value.
186  * The return value is the decoded extension or NULL on
187  * error. The actual error can have several different causes,
188  * the value of *crit reflects the cause:
189  * >= 0, extension found but not decoded (reflects critical value).
190  * -1 extension not found.
191  * -2 extension occurs more than once.
192  */
```



```
194 void *X509V3_get_d2i(STACK_OF(X509_EXTENSION) *x, int nid, int *crit, int *idx)
195 {
196         int lastpos, i;
197         X509_EXTENSION *ex, *found_ex = NULL;
198         if(!x) {
199                 if(idx) *idx = -1;
200                 if(crit) *crit = -1;
201                 return NULL;
202         }
203         if(idx) lastpos = *idx + 1;
204         else lastpos = 0;
205         if(lastpos < 0) lastpos = 0;
206         for(i = lastpos; i < sk_X509_EXTENSION_num(x); i++)
207         {
208                 ex = sk_X509_EXTENSION_value(x, i);
209                 if(OBJ_obj2nid(ex->object) == nid) {
210                         if(idx) {
211                                 *idx = i;
212                                 found_ex = ex;
213                                 break;
214                         } else if(found_ex) {
215                                 /* Found more than one */
216                                 if(crit) *crit = -2;
217                                 return NULL;
218                         }
219                         found_ex = ex;
220                 }
221         }
222         if(found_ex) {
223                 /* Found it */
224                 if(crit) *crit = X509_EXTENSION_get_critical(found_ex);
225                 return X509V3_EXT_d2i(found_ex);
226         }

228         /* Extension not found */
229         if(idx) *idx = -1;
230         if(crit) *crit = -1;
231         return NULL;
232 }

234 /* This function is a general extension append, replace and delete utility.
235  * The precise operation is governed by the 'flags' value. The 'crit' and
236  * 'value' arguments (if relevant) are the extensions internal structure.
237  */

239 int X509V3_add1_i2d(STACK_OF(X509_EXTENSION) **x, int nid, void *value,
240                                         int crit, unsigned long flags)
241 {
242         int extidx = -1;
243         int errcode;
244         X509_EXTENSION *ext, *extmp;
245         unsigned long ext_op = flags & X509V3_ADD_OP_MASK;

247         /* If appending we don't care if it exists, otherwise
248          * look for existing extension.
249          */
250         if(ext_op != X509V3_ADD_APPEND)
251                 extidx = X509v3_get_ext_by_NID(*x, nid, -1);

253         /* See if extension exists */
254         if(extidx >= 0) {
255                 /* If keep existing, nothing to do */
256                 if(ext_op == X509V3_ADD_KEEP_EXISTING)
257                         return 1;
258                 /* If default then its an error */
259                 if(ext_op == X509V3_ADD_DEFAULT) {
```

```
260                         errcode = X509V3_R_EXTENSION_EXISTS;
261                         goto err;
262                 }
263                 /* If delete, just delete it */
264                 if(ext_op == X509V3_ADD_DELETE) {
265                         if(!sk_X509_EXTENSION_delete(*x, extidx)) return -1;
266                         return 1;
267                 }
268         } else {
269                 /* If replace existing or delete, error since
270                  * extension must exist
271                  */
272                 if((ext_op == X509V3_ADD_REPLACE_EXISTING) ||
273                    (ext_op == X509V3_ADD_DELETE)) {
274                         errcode = X509V3_R_EXTENSION_NOT_FOUND;
275                         goto err;
276                 }
277         }

279         /* If we get this far then we have to create an extension:
280          * could have some flags for alternative encoding schemes...
281          */

283         ext = X509V3_EXT_i2d(nid, crit, value);

285         if(!ext) {
286                 X509V3err(X509V3_F_X509V3_ADD1_I2D, X509V3_R_ERROR_CREATING_EXTE
287                 return 0;
288         }

290         /* If extension exists replace it.. */
291         if(extidx >= 0) {
292                 extmp = sk_X509_EXTENSION_value(*x, extidx);
293                 X509_EXTENSION_free(extmp);
294                 if(!sk_X509_EXTENSION_set(*x, extidx, ext)) return -1;
295                 return 1;
296         }

298         if(!*x && !(*x = sk_X509_EXTENSION_new_null())) return -1;
299         if(!sk_X509_EXTENSION_push(*x, ext)) return -1;

301         return 1;

303         err:
304         if(!(flags & X509V3_ADD_SILENT))
305                 X509V3err(X509V3_F_X509V3_ADD1_I2D, errcode);
306         return 0;
307 }

309 IMPLEMENT_STACK_OF(X509V3_EXT_METHOD)
310 #endif /* ! codereview */
```

```
**********************************************************
   14022 Wed Aug 13 19:53:31 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_ncons.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_ncons.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1t.h>
#include <openssl/conf.h>
#include <openssl/x509v3.h>

static void *v2i_NAME_CONSTRAINTS(const X509V3_EXT_METHOD *method,
                                  X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval);
static int i2r_NAME_CONSTRAINTS(const X509V3_EXT_METHOD *method,
                                void *a, BIO *bp, int ind);
static int do_i2r_name_constraints(const X509V3_EXT_METHOD *method,
                                   STACK_OF(GENERAL_SUBTREE) *trees,
                                   BIO *bp, int ind, char *name);
static int print_nc_ipadd(BIO *bp, ASN1_OCTET_STRING *ip);

static int nc_match(GENERAL_NAME *gen, NAME_CONSTRAINTS *nc);
static int nc_match_single(GENERAL_NAME *sub, GENERAL_NAME *gen);
static int nc_dn(X509_NAME *sub, X509_NAME *nm);
static int nc_dns(ASN1_IA5STRING *sub, ASN1_IA5STRING *dns);
static int nc_email(ASN1_IA5STRING *sub, ASN1_IA5STRING *eml);
static int nc_uri(ASN1_IA5STRING *uri, ASN1_IA5STRING *base);

const X509V3_EXT_METHOD v3_name_constraints = {
        NID_name_constraints, 0,
        ASN1_ITEM_ref(NAME_CONSTRAINTS),
        0,0,0,0,
        0,0,
        0, v2i_NAME_CONSTRAINTS,
        i2r_NAME_CONSTRAINTS,0,
        NULL
};

ASN1_SEQUENCE(GENERAL_SUBTREE) = {
        ASN1_SIMPLE(GENERAL_SUBTREE, base, GENERAL_NAME),
        ASN1_IMP_OPT(GENERAL_SUBTREE, minimum, ASN1_INTEGER, 0),
        ASN1_IMP_OPT(GENERAL_SUBTREE, maximum, ASN1_INTEGER, 1)
} ASN1_SEQUENCE_END(GENERAL_SUBTREE)

ASN1_SEQUENCE(NAME_CONSTRAINTS) = {
        ASN1_IMP_SEQUENCE_OF_OPT(NAME_CONSTRAINTS, permittedSubtrees,
                                 GENERAL_SUBTREE, 0),
        ASN1_IMP_SEQUENCE_OF_OPT(NAME_CONSTRAINTS, excludedSubtrees,
                                 GENERAL_SUBTREE, 1),
} ASN1_SEQUENCE_END(NAME_CONSTRAINTS)


IMPLEMENT_ASN1_ALLOC_FUNCTIONS(GENERAL_SUBTREE)
IMPLEMENT_ASN1_ALLOC_FUNCTIONS(NAME_CONSTRAINTS)

static void *v2i_NAME_CONSTRAINTS(const X509V3_EXT_METHOD *method,
                                  X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
        {
        int i;
        CONF_VALUE tval, *val;
        STACK_OF(GENERAL_SUBTREE) **ptree = NULL;
        NAME_CONSTRAINTS *ncons = NULL;
        GENERAL_SUBTREE *sub = NULL;
        ncons = NAME_CONSTRAINTS_new();
        if (!ncons)
                goto memerr;
        for(i = 0; i < sk_CONF_VALUE_num(nval); i++)
                {
                val = sk_CONF_VALUE_value(nval, i);
                if (!strncmp(val->name, "permitted", 9) && val->name[9])
                        {
                        ptree = &ncons->permittedSubtrees;
                        tval.name = val->name + 10;
                        }
```

```
128                 else if (!strncmp(val->name, "excluded", 8) && val->name[8])
129                         {
130                         ptree = &ncons->excludedSubtrees;
131                         tval.name = val->name + 9;
132                         }
133                 else
134                         {
135                         X509V3err(X509V3_F_V2I_NAME_CONSTRAINTS, X509V3_R_INVALI
136                         goto err;
137                         }
138                 tval.value = val->value;
139                 sub = GENERAL_SUBTREE_new();
140                 if (!v2i_GENERAL_NAME_ex(sub->base, method, ctx, &tval, 1))
141                         goto err;
142                 if (!*ptree)
143                         *ptree = sk_GENERAL_SUBTREE_new_null();
144                 if (!*ptree || !sk_GENERAL_SUBTREE_push(*ptree, sub))
145                         goto memerr;
146                 sub = NULL;
147                 }

149         return ncons;

151         memerr:
152         X509V3err(X509V3_F_V2I_NAME_CONSTRAINTS, ERR_R_MALLOC_FAILURE);
153         err:
154         if (ncons)
155                 NAME_CONSTRAINTS_free(ncons);
156         if (sub)
157                 GENERAL_SUBTREE_free(sub);

159         return NULL;
160         }



165 static int i2r_NAME_CONSTRAINTS(const X509V3_EXT_METHOD *method, void *a,
166                                  BIO *bp, int ind)
167         {
168         NAME_CONSTRAINTS *ncons = a;
169         do_i2r_name_constraints(method, ncons->permittedSubtrees,
170                                         bp, ind, "Permitted");
171         do_i2r_name_constraints(method, ncons->excludedSubtrees,
172                                         bp, ind, "Excluded");
173         return 1;
174         }

176 static int do_i2r_name_constraints(const X509V3_EXT_METHOD *method,
177                                    STACK_OF(GENERAL_SUBTREE) *trees,
178                                    BIO *bp, int ind, char *name)
179         {
180         GENERAL_SUBTREE *tree;
181         int i;
182         if (sk_GENERAL_SUBTREE_num(trees) > 0)
183                 BIO_printf(bp, "%*s%s:\n", ind, "", name);
184         for(i = 0; i < sk_GENERAL_SUBTREE_num(trees); i++)
185                 {
186                 tree = sk_GENERAL_SUBTREE_value(trees, i);
187                 BIO_printf(bp, "%*s", ind + 2, "");
188                 if (tree->base->type == GEN_IPADD)
189                         print_nc_ipadd(bp, tree->base->d.ip);
190                 else
191                         GENERAL_NAME_print(bp, tree->base);
192                 BIO_puts(bp, "\n");
193                 }
```



```
194         return 1;
195         }

197 static int print_nc_ipadd(BIO *bp, ASN1_OCTET_STRING *ip)
198         {
199         int i, len;
200         unsigned char *p;
201         p = ip->data;
202         len = ip->length;
203         BIO_puts(bp, "IP:");
204         if(len == 8)
205                 {
206                 BIO_printf(bp, "%d.%d.%d.%d/%d.%d.%d.%d",
207                                 p[0], p[1], p[2], p[3],
208                                 p[4], p[5], p[6], p[7]);
209                 }
210         else if(len == 32)
211                 {
212                 for (i = 0; i < 16; i++)
213                         {
214                         BIO_printf(bp, "%X", p[0] << 8 | p[1]);
215                         p += 2;
216                         if (i == 7)
217                                 BIO_puts(bp, "/");
218                         else if (i != 15)
219                                 BIO_puts(bp, ":");
220                         }
221                 }
222         else
223                 BIO_printf(bp, "IP Address:<invalid>");
224         return 1;
225         }

227 /* Check a certificate conforms to a specified set of constraints.
228  * Return values:
229  *  X509_V_OK: All constraints obeyed.
230  *  X509_V_ERR_PERMITTED_VIOLATION: Permitted subtree violation.
231  *  X509_V_ERR_EXCLUDED_VIOLATION: Excluded subtree violation.
232  *  X509_V_ERR_SUBTREE_MINMAX: Min or max values present and matching type.
233  *  X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE:  Unsupported constraint type.
234  *  X509_V_ERR_UNSUPPORTED_CONSTRAINT_SYNTAX: bad unsupported constraint syntax.
235  *  X509_V_ERR_UNSUPPORTED_NAME_SYNTAX: bad or unsupported syntax of name

237  */

239 int NAME_CONSTRAINTS_check(X509 *x, NAME_CONSTRAINTS *nc)
240         {
241         int r, i;
242         X509_NAME *nm;

244         nm = X509_get_subject_name(x);

246         if (X509_NAME_entry_count(nm) > 0)
247                 {
248                 GENERAL_NAME gntmp;
249                 gntmp.type = GEN_DIRNAME;
250                 gntmp.d.directoryName = nm;

252                 r = nc_match(&gntmp, nc);

254                 if (r != X509_V_OK)
255                         return r;

257                 gntmp.type = GEN_EMAIL;
```

```
260                 /* Process any email address attributes in subject name */

262                 for (i = -1;;)
263                         {
264                         X509_NAME_ENTRY *ne;
265                         i = X509_NAME_get_index_by_NID(nm,
266                                                        NID_pkcs9_emailAddress,
267                                                        i);
268                         if (i == -1)
269                                 break;
270                         ne = X509_NAME_get_entry(nm, i);
271                         gntmp.d.rfc822Name = X509_NAME_ENTRY_get_data(ne);
272                         if (gntmp.d.rfc822Name->type != V_ASN1_IA5STRING)
273                                 return X509_V_ERR_UNSUPPORTED_NAME_SYNTAX;

275                         r = nc_match(&gntmp, nc);

277                         if (r != X509_V_OK)
278                                 return r;
279                         }

281                 }

283         for (i = 0; i < sk_GENERAL_NAME_num(x->altname); i++)
284                 {
285                 GENERAL_NAME *gen = sk_GENERAL_NAME_value(x->altname, i);
286                 r = nc_match(gen, nc);
287                 if (r != X509_V_OK)
288                         return r;
289                 }

291         return X509_V_OK;

293         }

295 static int nc_match(GENERAL_NAME *gen, NAME_CONSTRAINTS *nc)
296         {
297         GENERAL_SUBTREE *sub;
298         int i, r, match = 0;

300         /* Permitted subtrees: if any subtrees exist of matching the type
301          * at least one subtree must match.
302          */

304         for (i = 0; i < sk_GENERAL_SUBTREE_num(nc->permittedSubtrees); i++)
305                 {
306                 sub = sk_GENERAL_SUBTREE_value(nc->permittedSubtrees, i);
307                 if (gen->type != sub->base->type)
308                         continue;
309                 if (sub->minimum || sub->maximum)
310                         return X509_V_ERR_SUBTREE_MINMAX;
311                 /* If we already have a match don't bother trying any more */
312                 if (match == 2)
313                         continue;
314                 if (match == 0)
315                         match = 1;
316                 r = nc_match_single(gen, sub->base);
317                 if (r == X509_V_OK)
318                         match = 2;
319                 else if (r != X509_V_ERR_PERMITTED_VIOLATION)
320                         return r;
321                 }

323         if (match == 1)
324                 return X509_V_ERR_PERMITTED_VIOLATION;
```



```
326         /* Excluded subtrees: must not match any of these */

328         for (i = 0; i < sk_GENERAL_SUBTREE_num(nc->excludedSubtrees); i++)
329                 {
330                 sub = sk_GENERAL_SUBTREE_value(nc->excludedSubtrees, i);
331                 if (gen->type != sub->base->type)
332                         continue;
333                 if (sub->minimum || sub->maximum)
334                         return X509_V_ERR_SUBTREE_MINMAX;

336                 r = nc_match_single(gen, sub->base);
337                 if (r == X509_V_OK)
338                         return X509_V_ERR_EXCLUDED_VIOLATION;
339                 else if (r != X509_V_ERR_PERMITTED_VIOLATION)
340                         return r;

342                 }

344         return X509_V_OK;

346         }

348 static int nc_match_single(GENERAL_NAME *gen, GENERAL_NAME *base)
349         {
350         switch(base->type)
351                 {
352                 case GEN_DIRNAME:
353                 return nc_dn(gen->d.directoryName, base->d.directoryName);

355                 case GEN_DNS:
356                 return nc_dns(gen->d.dNSName, base->d.dNSName);

358                 case GEN_EMAIL:
359                 return nc_email(gen->d.rfc822Name, base->d.rfc822Name);

361                 case GEN_URI:
362                 return nc_uri(gen->d.uniformResourceIdentifier,
363                                         base->d.uniformResourceIdentifier);

365                 default:
366                 return X509_V_ERR_UNSUPPORTED_CONSTRAINT_TYPE;
367                 }

369         }

371 /* directoryName name constraint matching.
372  * The canonical encoding of X509_NAME makes this comparison easy. It is
373  * matched if the subtree is a subset of the name.
374  */

376 static int nc_dn(X509_NAME *nm, X509_NAME *base)
377         {
378         /* Ensure canonical encodings are up to date.  */
379         if (nm->modified && i2d_X509_NAME(nm, NULL) < 0)
380                 return X509_V_ERR_OUT_OF_MEM;
381         if (base->modified && i2d_X509_NAME(base, NULL) < 0)
382                 return X509_V_ERR_OUT_OF_MEM;
383         if (base->canon_enclen > nm->canon_enclen)
384                 return X509_V_ERR_PERMITTED_VIOLATION;
385         if (memcmp(base->canon_enc, nm->canon_enc, base->canon_enclen))
386                 return X509_V_ERR_PERMITTED_VIOLATION;
387         return X509_V_OK;
388         }

390 static int nc_dns(ASN1_IA5STRING *dns, ASN1_IA5STRING *base)
391         {
```

```
392         char *baseptr = (char *)base->data;
393         char *dnsptr = (char *)dns->data;
394         /* Empty matches everything */
395         if (!*baseptr)
396                 return X509_V_OK;
397         /* Otherwise can add zero or more components on the left so
398          * compare RHS and if dns is longer and expect '.' as preceding
399          * character.
400          */
401         if (dns->length > base->length)
402                 {
403                 dnsptr += dns->length - base->length;
404                 if (dnsptr[-1] != '.')
405                         return X509_V_ERR_PERMITTED_VIOLATION;
406                 }

408         if (strcasecmp(baseptr, dnsptr))
409                         return X509_V_ERR_PERMITTED_VIOLATION;

411         return X509_V_OK;

413         }

415 static int nc_email(ASN1_IA5STRING *eml, ASN1_IA5STRING *base)
416         {
417         const char *baseptr = (char *)base->data;
418         const char *emlptr = (char *)eml->data;

420         const char *baseat = strchr(baseptr, '@');
421         const char *emlat = strchr(emlptr, '@');
422         if (!emlat)
423                 return X509_V_ERR_UNSUPPORTED_NAME_SYNTAX;
424         /* Special case: inital '.' is RHS match */
425         if (!baseat && (*baseptr == '.'))
426                 {
427                 if (eml->length > base->length)
428                         {
429                         emlptr += eml->length - base->length;
430                         if (!strcasecmp(baseptr, emlptr))
431                                 return X509_V_OK;
432                         }
433                 return X509_V_ERR_PERMITTED_VIOLATION;
434                 }

436         /* If we have anything before '@' match local part */

438         if (baseat)
439                 {
440                 if (baseat != baseptr)
441                         {
442                         if ((baseat - baseptr) != (emlat - emlptr))
443                                 return X509_V_ERR_PERMITTED_VIOLATION;
444                         /* Case sensitive match of local part */
445                         if (strncmp(baseptr, emlptr, emlat - emlptr))
446                                 return X509_V_ERR_PERMITTED_VIOLATION;
447                         }
448                 /* Position base after '@' */
449                 baseptr = baseat + 1;
450                 }
451         emlptr = emlat + 1;
452         /* Just have hostname left to match: case insensitive */
453         if (strcasecmp(baseptr, emlptr))
454                 return X509_V_ERR_PERMITTED_VIOLATION;

456         return X509_V_OK;
```



```
458         }

460 static int nc_uri(ASN1_IA5STRING *uri, ASN1_IA5STRING *base)
461         {
462         const char *baseptr = (char *)base->data;
463         const char *hostptr = (char *)uri->data;
464         const char *p = strchr(hostptr, ':');
465         int hostlen;
466         /* Check for foo:// and skip past it */
467         if (!p || (p[1] != '/') || (p[2] != '/'))
468                 return X509_V_ERR_UNSUPPORTED_NAME_SYNTAX;
469         hostptr = p + 3;

471         /* Determine length of hostname part of URI */

473         /* Look for a port indicator as end of hostname first */

475         p = strchr(hostptr, ':');
476         /* Otherwise look for trailing slash */
477         if (!p)
478                 p = strchr(hostptr, '/');

480         if (!p)
481                 hostlen = strlen(hostptr);
482         else
483                 hostlen = p - hostptr;

485         if (hostlen == 0)
486                 return X509_V_ERR_UNSUPPORTED_NAME_SYNTAX;

488         /* Special case: inital '.' is RHS match */
489         if (*baseptr == '.')
490                 {
491                 if (hostlen > base->length)
492                         {
493                         p = hostptr + hostlen - base->length;
494                         if (!strncasecmp(p, baseptr, base->length))
495                                 return X509_V_OK;
496                         }
497                 return X509_V_ERR_PERMITTED_VIOLATION;
498                 }

500         if ((base->length != (int)hostlen) || strncasecmp(hostptr, baseptr, host
501                 return X509_V_ERR_PERMITTED_VIOLATION;

503         return X509_V_OK;

505         }
506 #endif /* ! codereview */
```

```
**********************************************************
    8385 Wed Aug 13 19:53:31 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_ocsp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_ocsp.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#ifndef OPENSSL_NO_OCSP

#include <stdio.h>
```



```
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/asn1.h>
#include <openssl/ocsp.h>
#include <openssl/x509v3.h>

/* OCSP extensions and a couple of CRL entry extensions
 */

static int i2r_ocsp_crlid(const X509V3_EXT_METHOD *method, void *nonce,
                          BIO *out, int indent);
static int i2r_ocsp_acutoff(const X509V3_EXT_METHOD *method, void *nonce,
                            BIO *out, int indent);
static int i2r_object(const X509V3_EXT_METHOD *method, void *obj, BIO *out,
                      int indent);

static void *ocsp_nonce_new(void);
static int i2d_ocsp_nonce(void *a, unsigned char **pp);
static void *d2i_ocsp_nonce(void *a, const unsigned char **pp, long length);
static void ocsp_nonce_free(void *a);
static int i2r_ocsp_nonce(const X509V3_EXT_METHOD *method, void *nonce,
                          BIO *out, int indent);

static int i2r_ocsp_nocheck(const X509V3_EXT_METHOD *method,
                            void *nocheck, BIO *out, int indent);
static void *s2i_ocsp_nocheck(const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
                              const char *str);
static int i2r_ocsp_serviceloc(const X509V3_EXT_METHOD *method, void *in,
                               BIO *bp, int ind);

const X509V3_EXT_METHOD v3_ocsp_crlid = {
        NID_id_pkix_OCSP_CrlID, 0, ASN1_ITEM_ref(OCSP_CRLID),
        0,0,0,0,
        0,0,
        0,0,
        i2r_ocsp_crlid,0,
        NULL
};

const X509V3_EXT_METHOD v3_ocsp_acutoff = {
        NID_id_pkix_OCSP_archiveCutoff, 0, ASN1_ITEM_ref(ASN1_GENERALIZEDTIME),
        0,0,0,0,
        0,0,
        0,0,
        i2r_ocsp_acutoff,0,
        NULL
};

const X509V3_EXT_METHOD v3_crl_invdate = {
        NID_invalidity_date, 0, ASN1_ITEM_ref(ASN1_GENERALIZEDTIME),
        0,0,0,0,
        0,0,
        0,0,
        i2r_ocsp_acutoff,0,
        NULL
};

const X509V3_EXT_METHOD v3_crl_hold = {
        NID_hold_instruction_code, 0, ASN1_ITEM_ref(ASN1_OBJECT),
        0,0,0,0,
        0,0,
        0,0,
        i2r_object,0,
        NULL
};
```

```
128 const X509V3_EXT_METHOD v3_ocsp_nonce = {
129         NID_id_pkix_OCSP_Nonce, 0, NULL,
130         ocsp_nonce_new,
131         ocsp_nonce_free,
132         d2i_ocsp_nonce,
133         i2d_ocsp_nonce,
134         0,0,
135         0,0,
136         i2r_ocsp_nonce,0,
137         NULL
138 };

140 const X509V3_EXT_METHOD v3_ocsp_nocheck = {
141         NID_id_pkix_OCSP_noCheck, 0, ASN1_ITEM_ref(ASN1_NULL),
142         0,0,0,0,
143         0,s2i_ocsp_nocheck,
144         0,0,
145         i2r_ocsp_nocheck,0,
146         NULL
147 };

149 const X509V3_EXT_METHOD v3_ocsp_serviceloc = {
150         NID_id_pkix_OCSP_serviceLocator, 0, ASN1_ITEM_ref(OCSP_SERVICELOC),
151         0,0,0,0,
152         0,0,
153         0,0,
154         i2r_ocsp_serviceloc,0,
155         NULL
156 };

158 static int i2r_ocsp_crlid(const X509V3_EXT_METHOD *method, void *in, BIO *bp,
159                           int ind)
160 {
161         OCSP_CRLID *a = in;
162         if (a->crlUrl)
163                 {
164                 if (BIO_printf(bp, "%*scrlUrl: ", ind, "") <= 0) goto err;
165                 if (!ASN1_STRING_print(bp, (ASN1_STRING*)a->crlUrl)) goto err;
166                 if (BIO_write(bp, "\n", 1) <= 0) goto err;
167                 }
168         if (a->crlNum)
169                 {
170                 if (BIO_printf(bp, "%*scrlNum: ", ind, "") <= 0) goto err;
171                 if (i2a_ASN1_INTEGER(bp, a->crlNum) <= 0) goto err;
172                 if (BIO_write(bp, "\n", 1) <= 0) goto err;
173                 }
174         if (a->crlTime)
175                 {
176                 if (BIO_printf(bp, "%*scrlTime: ", ind, "") <= 0) goto err;
177                 if (!ASN1_GENERALIZEDTIME_print(bp, a->crlTime)) goto err;
178                 if (BIO_write(bp, "\n", 1) <= 0) goto err;
179                 }
180         return 1;
181         err:
182         return 0;
183 }

185 static int i2r_ocsp_acutoff(const X509V3_EXT_METHOD *method, void *cutoff,
186                             BIO *bp, int ind)
187 {
188         if (BIO_printf(bp, "%*s", ind, "") <= 0) return 0;
189         if(!ASN1_GENERALIZEDTIME_print(bp, cutoff)) return 0;
190         return 1;
191 }
```



```
194 static int i2r_object(const X509V3_EXT_METHOD *method, void *oid, BIO *bp,
195                       int ind)
196 {
197         if (BIO_printf(bp, "%*s", ind, "") <= 0) return 0;
198         if(i2a_ASN1_OBJECT(bp, oid) <= 0) return 0;
199         return 1;
200 }

202 /* OCSP nonce. This is needs special treatment because it doesn't have
203  * an ASN1 encoding at all: it just contains arbitrary data.
204  */

206 static void *ocsp_nonce_new(void)
207 {
208         return ASN1_OCTET_STRING_new();
209 }

211 static int i2d_ocsp_nonce(void *a, unsigned char **pp)
212 {
213         ASN1_OCTET_STRING *os = a;
214         if(pp) {
215                 memcpy(*pp, os->data, os->length);
216                 *pp += os->length;
217         }
218         return os->length;
219 }

221 static void *d2i_ocsp_nonce(void *a, const unsigned char **pp, long length)
222 {
223         ASN1_OCTET_STRING *os, **pos;
224         pos = a;
225         if(!pos || !*pos) os = ASN1_OCTET_STRING_new();
226         else os = *pos;
227         if(!ASN1_OCTET_STRING_set(os, *pp, length)) goto err;

229         *pp += length;

231         if(pos) *pos = os;
232         return os;

234         err:
235         if(os && (!pos || (*pos != os))) M_ASN1_OCTET_STRING_free(os);
236         OCSPerr(OCSP_F_D2I_OCSP_NONCE, ERR_R_MALLOC_FAILURE);
237         return NULL;
238 }

240 static void ocsp_nonce_free(void *a)
241 {
242         M_ASN1_OCTET_STRING_free(a);
243 }

245 static int i2r_ocsp_nonce(const X509V3_EXT_METHOD *method, void *nonce,
246                           BIO *out, int indent)
247 {
248         if(BIO_printf(out, "%*s", indent, "") <= 0) return 0;
249         if(i2a_ASN1_STRING(out, nonce, V_ASN1_OCTET_STRING) <= 0) return 0;
250         return 1;
251 }

253 /* Nocheck is just a single NULL. Don't print anything and always set it */

255 static int i2r_ocsp_nocheck(const X509V3_EXT_METHOD *method, void *nocheck,
256                             BIO *out, int indent)
257 {
258         return 1;
259 }
```

```
261 static void *s2i_ocsp_nocheck(const X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
262                                const char *str)
263 {
264         return ASN1_NULL_new();
265 }

267 static int i2r_ocsp_serviceloc(const X509V3_EXT_METHOD *method, void *in,
268                                BIO *bp, int ind)
269         {
270         int i;
271         OCSP_SERVICELOC *a = in;
272         ACCESS_DESCRIPTION *ad;

274         if (BIO_printf(bp, "%*sIssuer: ", ind, "") <= 0) goto err;
275         if (X509_NAME_print_ex(bp, a->issuer, 0, XN_FLAG_ONELINE) <= 0) goto err
276         for (i = 0; i < sk_ACCESS_DESCRIPTION_num(a->locator); i++)
277                 {
278                                 ad = sk_ACCESS_DESCRIPTION_value(a->locator,i);
279                                 if (BIO_printf(bp, "\n%*s", (2*ind), "") <= 0)
280                                         goto err;
281                                 if(i2a_ASN1_OBJECT(bp, ad->method) <= 0) goto er
282                                 if(BIO_puts(bp, " - ") <= 0) goto err;
283                                 if(GENERAL_NAME_print(bp, ad->location) <= 0) go
284                 }
285         return 1;
286 err:
287         return 0;
288         }
289 #endif
290 #endif /* ! codereview */
```

```
**********************************************************
    9050 Wed Aug 13 19:53:31 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_pci.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* v3_pci.c -*- mode:C; c-file-style: "eay" -*- */
/* Contributed to the OpenSSL Project 2004
 * by Richard Levitte (richard@levitte.org)
 */
/* Copyright (c) 2004 Kungliga Tekniska Högskolan
 * (Royal Institute of Technology, Stockholm, Sweden).
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the Institute nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/x509v3.h>

static int i2r_pci(X509V3_EXT_METHOD *method, PROXY_CERT_INFO_EXTENSION *ext,
         BIO *out, int indent);
static PROXY_CERT_INFO_EXTENSION *r2i_pci(X509V3_EXT_METHOD *method,
         X509V3_CTX *ctx, char *str);

const X509V3_EXT_METHOD v3_pci =
        { NID_proxyCertInfo, 0, ASN1_ITEM_ref(PROXY_CERT_INFO_EXTENSION),
          0,0,0,0,
          0,0,
          NULL, NULL,
          (X509V3_EXT_I2R)i2r_pci,
          (X509V3_EXT_R2I)r2i_pci,
          NULL,
        };

static int i2r_pci(X509V3_EXT_METHOD *method, PROXY_CERT_INFO_EXTENSION *pci,
        BIO *out, int indent)
        {
        BIO_printf(out, "%*sPath Length Constraint: ", indent, "");
        if (pci->pcPathLengthConstraint)
```



```
          i2a_ASN1_INTEGER(out, pci->pcPathLengthConstraint);
        else
          BIO_printf(out, "infinite");
        BIO_puts(out, "\n");
        BIO_printf(out, "%*sPolicy Language: ", indent, "");
        i2a_ASN1_OBJECT(out, pci->proxyPolicy->policyLanguage);
        BIO_puts(out, "\n");
        if (pci->proxyPolicy->policy && pci->proxyPolicy->policy->data)
          BIO_printf(out, "%*sPolicy Text: %s\n", indent, "",
                     pci->proxyPolicy->policy->data);
        return 1;
        }

static int process_pci_value(CONF_VALUE *val,
        ASN1_OBJECT **language, ASN1_INTEGER **pathlen,
        ASN1_OCTET_STRING **policy)
        {
        int free_policy = 0;

        if (strcmp(val->name, "language") == 0)
                {
                if (*language)
                        {
                        X509V3err(X509V3_F_PROCESS_PCI_VALUE,X509V3_R_POLICY_LAN
                        X509V3_conf_err(val);
                        return 0;
                        }
                if (!(*language = OBJ_txt2obj(val->value, 0)))
                        {
                        X509V3err(X509V3_F_PROCESS_PCI_VALUE,X509V3_R_INVALID_OB
                        X509V3_conf_err(val);
                        return 0;
                        }
                }
        else if (strcmp(val->name, "pathlen") == 0)
                {
                if (*pathlen)
                        {
                        X509V3err(X509V3_F_PROCESS_PCI_VALUE,X509V3_R_POLICY_PAT
                        X509V3_conf_err(val);
                        return 0;
                        }
                if (!X509V3_get_value_int(val, pathlen))
                        {
                        X509V3err(X509V3_F_PROCESS_PCI_VALUE,X509V3_R_POLICY_PAT
                        X509V3_conf_err(val);
                        return 0;
                        }
                }
        else if (strcmp(val->name, "policy") == 0)
                {
                unsigned char *tmp_data = NULL;
                long val_len;
                if (!*policy)
                        {
                        *policy = ASN1_OCTET_STRING_new();
                        if (!*policy)
                                {
                                X509V3err(X509V3_F_PROCESS_PCI_VALUE,ERR_R_MALLO
                                X509V3_conf_err(val);
                                return 0;
                                }
                        free_policy = 1;
                        }
                if (strncmp(val->value, "hex:", 4) == 0)
                        {
```

```
128                         unsigned char *tmp_data2 =
129                                 string_to_hex(val->value + 4, &val_len);
130
131                         if (!tmp_data2)
132                                 {
133                                 X509V3err(X509V3_F_PROCESS_PCI_VALUE,X509V3_R_IL
134                                 X509V3_conf_err(val);
135                                 goto err;
136                                 }
137
138                         tmp_data = OPENSSL_realloc((*policy)->data,
139                                 (*policy)->length + val_len + 1);
140                         if (tmp_data)
141                                 {
142                                 (*policy)->data = tmp_data;
143                                 memcpy(&(*policy)->data[(*policy)->length],
144                                         tmp_data2, val_len);
145                                 (*policy)->length += val_len;
146                                 (*policy)->data[(*policy)->length] = '\0';
147                                 }
148                         else
149                                 {
150                                 OPENSSL_free(tmp_data2);
151                                 /* realloc failure implies the original data spa
152                                 (*policy)->data = NULL;
153                                 (*policy)->length = 0;
154                                 X509V3err(X509V3_F_PROCESS_PCI_VALUE,ERR_R_MALLO
155                                 X509V3_conf_err(val);
156                                 goto err;
157                                 }
158                         OPENSSL_free(tmp_data2);
159                         }
160                 else if (strncmp(val->value, "file:", 5) == 0)
161                         {
162                         unsigned char buf[2048];
163                         int n;
164                         BIO *b = BIO_new_file(val->value + 5, "r");
165                         if (!b)
166                                 {
167                                 X509V3err(X509V3_F_PROCESS_PCI_VALUE,ERR_R_BIO_L
168                                 X509V3_conf_err(val);
169                                 goto err;
170                                 }
171                         while((n = BIO_read(b, buf, sizeof(buf))) > 0
172                                 || (n == 0 && BIO_should_retry(b)))
173                                 {
174                                 if (!n) continue;
175
176                                 tmp_data = OPENSSL_realloc((*policy)->data,
177                                         (*policy)->length + n + 1);
178
179                                 if (!tmp_data)
180                                         break;
181
182                                 (*policy)->data = tmp_data;
183                                 memcpy(&(*policy)->data[(*policy)->length],
184                                         buf, n);
185                                 (*policy)->length += n;
186                                 (*policy)->data[(*policy)->length] = '\0';
187                                 }
188                         BIO_free_all(b);
189
190                         if (n < 0)
191                                 {
192                                 X509V3err(X509V3_F_PROCESS_PCI_VALUE,ERR_R_BIO_L
193                                 X509V3_conf_err(val);
```



```
194                                 goto err;
195                                 }
196                         }
197                 else if (strncmp(val->value, "text:", 5) == 0)
198                         {
199                         val_len = strlen(val->value + 5);
200                         tmp_data = OPENSSL_realloc((*policy)->data,
201                                 (*policy)->length + val_len + 1);
202                         if (tmp_data)
203                                 {
204                                 (*policy)->data = tmp_data;
205                                 memcpy(&(*policy)->data[(*policy)->length],
206                                         val->value + 5, val_len);
207                                 (*policy)->length += val_len;
208                                 (*policy)->data[(*policy)->length] = '\0';
209                                 }
210                         else
211                                 {
212                                 /* realloc failure implies the original data spa
213                                 (*policy)->data = NULL;
214                                 (*policy)->length = 0;
215                                 X509V3err(X509V3_F_PROCESS_PCI_VALUE,ERR_R_MALLO
216                                 X509V3_conf_err(val);
217                                 goto err;
218                                 }
219                         }
220                 else
221                         {
222                         X509V3err(X509V3_F_PROCESS_PCI_VALUE,X509V3_R_INCORRECT_
223                         X509V3_conf_err(val);
224                         goto err;
225                         }
226                 if (!tmp_data)
227                         {
228                         X509V3err(X509V3_F_PROCESS_PCI_VALUE,ERR_R_MALLOC_FAILUR
229                         X509V3_conf_err(val);
230                         goto err;
231                         }
232                 }
233         return 1;
234 err:
235         if (free_policy)
236                 {
237                 ASN1_OCTET_STRING_free(*policy);
238                 *policy = NULL;
239                 }
240         return 0;
241         }
242
243 static PROXY_CERT_INFO_EXTENSION *r2i_pci(X509V3_EXT_METHOD *method,
244         X509V3_CTX *ctx, char *value)
245         {
246         PROXY_CERT_INFO_EXTENSION *pci = NULL;
247         STACK_OF(CONF_VALUE) *vals;
248         ASN1_OBJECT *language = NULL;
249         ASN1_INTEGER *pathlen = NULL;
250         ASN1_OCTET_STRING *policy = NULL;
251         int i, j;
252
253         vals = X509V3_parse_list(value);
254         for (i = 0; i < sk_CONF_VALUE_num(vals); i++)
255                 {
256                 CONF_VALUE *cnf = sk_CONF_VALUE_value(vals, i);
257                 if (!cnf->name || (*cnf->name != '@' && !cnf->value))
258                         {
259                         X509V3err(X509V3_F_R2I_PCI,X509V3_R_INVALID_PROXY_POLICY
```

```
260                         X509V3_conf_err(cnf);
261                         goto err;
262                         }
263                 if (*cnf->name == '@')
264                         {
265                         STACK_OF(CONF_VALUE) *sect;
266                         int success_p = 1;

268                         sect = X509V3_get_section(ctx, cnf->name + 1);
269                         if (!sect)
270                                 {
271                                 X509V3err(X509V3_F_R2I_PCI,X509V3_R_INVALID_SECT
272                                 X509V3_conf_err(cnf);
273                                 goto err;
274                                 }
275                         for (j = 0; success_p && j < sk_CONF_VALUE_num(sect); j+
276                                 {
277                                 success_p =
278                                         process_pci_value(sk_CONF_VALUE_value(se
279                                                 &language, &pathlen, &policy);
280                                 }
281                         X509V3_section_free(ctx, sect);
282                         if (!success_p)
283                                 goto err;
284                         }
285                 else
286                         {
287                         if (!process_pci_value(cnf,
288                                         &language, &pathlen, &policy))
289                                 {
290                                 X509V3_conf_err(cnf);
291                                 goto err;
292                                 }
293                         }
294                 }

296         /* Language is mandatory */
297         if (!language)
298                 {
299                 X509V3err(X509V3_F_R2I_PCI,X509V3_R_NO_PROXY_CERT_POLICY_LANGUAG
300                 goto err;
301                 }
302         i = OBJ_obj2nid(language);
303         if ((i == NID_Independent || i == NID_id_ppl_inheritAll) && policy)
304                 {
305                 X509V3err(X509V3_F_R2I_PCI,X509V3_R_POLICY_WHEN_PROXY_LANGUAGE_R
306                 goto err;
307                 }

309         pci = PROXY_CERT_INFO_EXTENSION_new();
310         if (!pci)
311                 {
312                 X509V3err(X509V3_F_R2I_PCI,ERR_R_MALLOC_FAILURE);
313                 goto err;
314                 }

316         pci->proxyPolicy->policyLanguage = language; language = NULL;
317         pci->proxyPolicy->policy = policy; policy = NULL;
318         pci->pcPathLengthConstraint = pathlen; pathlen = NULL;
319         goto end;
320 err:
321         if (language) { ASN1_OBJECT_free(language); language = NULL; }
322         if (pathlen) { ASN1_INTEGER_free(pathlen); pathlen = NULL; }
323         if (policy) { ASN1_OCTET_STRING_free(policy); policy = NULL; }
324         if (pci) { PROXY_CERT_INFO_EXTENSION_free(pci); pci = NULL; }
325 end:
```



```
326         sk_CONF_VALUE_pop_free(vals, X509V3_conf_free);
327         return pci;
328         }
329 #endif /* ! codereview */
```

****
2357 Wed Aug 13 19:53:31 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_pcia.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
/* v3_pcia.c -*- mode:C; c-file-style: "eay" -*- */
/* Contributed to the OpenSSL Project 2004
 * by Richard Levitte (richard@levitte.org)
 */
/* Copyright (c) 2004 Kungliga Tekniska Högskolan
 * (Royal Institute of Technology, Stockholm, Sweden).
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * 3. Neither the name of the Institute nor the names of its contributors
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */

#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

ASN1_SEQUENCE(PROXY_POLICY) =
        {
        ASN1_SIMPLE(PROXY_POLICY,policyLanguage,ASN1_OBJECT),
        ASN1_OPT(PROXY_POLICY,policy,ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END(PROXY_POLICY)

IMPLEMENT_ASN1_FUNCTIONS(PROXY_POLICY)

ASN1_SEQUENCE(PROXY_CERT_INFO_EXTENSION) =
        {
        ASN1_OPT(PROXY_CERT_INFO_EXTENSION,pcPathLengthConstraint,ASN1_INTEGER),
        ASN1_SIMPLE(PROXY_CERT_INFO_EXTENSION,proxyPolicy,PROXY_POLICY)
} ASN1_SEQUENCE_END(PROXY_CERT_INFO_EXTENSION)

IMPLEMENT_ASN1_FUNCTIONS(PROXY_CERT_INFO_EXTENSION)
#endif /* ! codereview */
```

```
**********************************************************
    5039 Wed Aug 13 19:53:32 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_pcons.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_pcons.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/conf.h>
#include <openssl/x509v3.h>

static STACK_OF(CONF_VALUE) *
i2v_POLICY_CONSTRAINTS(const X509V3_EXT_METHOD *method, void *bcons,
                       STACK_OF(CONF_VALUE) *extlist);
static void *v2i_POLICY_CONSTRAINTS(const X509V3_EXT_METHOD *method,
                                    X509V3_CTX *ctx,
                                    STACK_OF(CONF_VALUE) *values);

const X509V3_EXT_METHOD v3_policy_constraints = {
NID_policy_constraints, 0,
ASN1_ITEM_ref(POLICY_CONSTRAINTS),
0,0,0,0,
0,0,
i2v_POLICY_CONSTRAINTS,
v2i_POLICY_CONSTRAINTS,
NULL,NULL,
NULL
};

ASN1_SEQUENCE(POLICY_CONSTRAINTS) = {
        ASN1_IMP_OPT(POLICY_CONSTRAINTS, requireExplicitPolicy, ASN1_INTEGER,0),
        ASN1_IMP_OPT(POLICY_CONSTRAINTS, inhibitPolicyMapping, ASN1_INTEGER,1)
} ASN1_SEQUENCE_END(POLICY_CONSTRAINTS)

IMPLEMENT_ASN1_ALLOC_FUNCTIONS(POLICY_CONSTRAINTS)


static STACK_OF(CONF_VALUE) *
i2v_POLICY_CONSTRAINTS(const X509V3_EXT_METHOD *method, void *a,
                       STACK_OF(CONF_VALUE) *extlist)
{
        POLICY_CONSTRAINTS *pcons = a;
        X509V3_add_value_int("Require Explicit Policy",
                        pcons->requireExplicitPolicy, &extlist);
        X509V3_add_value_int("Inhibit Policy Mapping",
                        pcons->inhibitPolicyMapping, &extlist);
        return extlist;
}

static void *v2i_POLICY_CONSTRAINTS(const X509V3_EXT_METHOD *method,
                                    X509V3_CTX *ctx,
                                    STACK_OF(CONF_VALUE) *values)
{
        POLICY_CONSTRAINTS *pcons=NULL;
        CONF_VALUE *val;
        int i;
        if(!(pcons = POLICY_CONSTRAINTS_new())) {
                X509V3err(X509V3_F_V2I_POLICY_CONSTRAINTS, ERR_R_MALLOC_FAILURE)
                return NULL;
        }
        for(i = 0; i < sk_CONF_VALUE_num(values); i++) {
                val = sk_CONF_VALUE_value(values, i);
                if(!strcmp(val->name, "requireExplicitPolicy")) {
                        if(!X509V3_get_value_int(val,
                                &pcons->requireExplicitPolicy)) goto err;
                } else if(!strcmp(val->name, "inhibitPolicyMapping")) {
                        if(!X509V3_get_value_int(val,
                                &pcons->inhibitPolicyMapping)) goto err;
                } else {
                        X509V3err(X509V3_F_V2I_POLICY_CONSTRAINTS, X509V3_R_INVA
                        X509V3_conf_err(val);
                        goto err;
```

```
128                 }
129         }
130         if (!pcons->inhibitPolicyMapping && !pcons->requireExplicitPolicy) {
131                 X509V3err(X509V3_F_V2I_POLICY_CONSTRAINTS, X509V3_R_ILLEGAL_EMPT
132                 goto err;
133         }

135         return pcons;
136         err:
137         POLICY_CONSTRAINTS_free(pcons);
138         return NULL;
139 }
140 #endif /* ! codereview */
```

```
**********************************************************
    4154 Wed Aug 13 19:53:32 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_pku.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_pku.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/asn1.h>
```



```
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

static int i2r_PKEY_USAGE_PERIOD(X509V3_EXT_METHOD *method, PKEY_USAGE_PERIOD *u
/*
static PKEY_USAGE_PERIOD *v2i_PKEY_USAGE_PERIOD(X509V3_EXT_METHOD *method, X509V
*/
const X509V3_EXT_METHOD v3_pkey_usage_period = {
NID_private_key_usage_period, 0, ASN1_ITEM_ref(PKEY_USAGE_PERIOD),
0,0,0,0,
0,0,0,0,
(X509V3_EXT_I2R)i2r_PKEY_USAGE_PERIOD, NULL,
NULL
};

ASN1_SEQUENCE(PKEY_USAGE_PERIOD) = {
        ASN1_IMP_OPT(PKEY_USAGE_PERIOD, notBefore, ASN1_GENERALIZEDTIME, 0),
        ASN1_IMP_OPT(PKEY_USAGE_PERIOD, notAfter, ASN1_GENERALIZEDTIME, 1)
} ASN1_SEQUENCE_END(PKEY_USAGE_PERIOD)

IMPLEMENT_ASN1_FUNCTIONS(PKEY_USAGE_PERIOD)

static int i2r_PKEY_USAGE_PERIOD(X509V3_EXT_METHOD *method,
             PKEY_USAGE_PERIOD *usage, BIO *out, int indent)
{
        BIO_printf(out, "%*s", indent, "");
        if(usage->notBefore) {
                BIO_write(out, "Not Before: ", 12);
                ASN1_GENERALIZEDTIME_print(out, usage->notBefore);
                if(usage->notAfter) BIO_write(out, ", ", 2);
        }
        if(usage->notAfter) {
                BIO_write(out, "Not After: ", 11);
                ASN1_GENERALIZEDTIME_print(out, usage->notAfter);
        }
        return 1;
}

/*
static PKEY_USAGE_PERIOD *v2i_PKEY_USAGE_PERIOD(method, ctx, values)
X509V3_EXT_METHOD *method;
X509V3_CTX *ctx;
STACK_OF(CONF_VALUE) *values;
{
return NULL;
}
*/
#endif /* ! codereview */
```

```
**********************************************************
    5475 Wed Aug 13 19:53:32 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_pmaps.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_pmaps.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/asn1t.h>
#include <openssl/conf.h>
#include <openssl/x509v3.h>

static void *v2i_POLICY_MAPPINGS(const X509V3_EXT_METHOD *method,
                                 X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval);
static STACK_OF(CONF_VALUE) *
i2v_POLICY_MAPPINGS(const X509V3_EXT_METHOD *method, void *pmps,
                    STACK_OF(CONF_VALUE) *extlist);

const X509V3_EXT_METHOD v3_policy_mappings = {
        NID_policy_mappings, 0,
        ASN1_ITEM_ref(POLICY_MAPPINGS),
        0,0,0,0,
        0,0,
        i2v_POLICY_MAPPINGS,
        v2i_POLICY_MAPPINGS,
        0,0,
        NULL
};

ASN1_SEQUENCE(POLICY_MAPPING) = {
        ASN1_SIMPLE(POLICY_MAPPING, issuerDomainPolicy, ASN1_OBJECT),
        ASN1_SIMPLE(POLICY_MAPPING, subjectDomainPolicy, ASN1_OBJECT)
} ASN1_SEQUENCE_END(POLICY_MAPPING)

ASN1_ITEM_TEMPLATE(POLICY_MAPPINGS) =
        ASN1_EX_TEMPLATE_TYPE(ASN1_TFLG_SEQUENCE_OF, 0, POLICY_MAPPINGS,
                                                                POLICY_MAPPING)
ASN1_ITEM_TEMPLATE_END(POLICY_MAPPINGS)

IMPLEMENT_ASN1_ALLOC_FUNCTIONS(POLICY_MAPPING)


static STACK_OF(CONF_VALUE) *
i2v_POLICY_MAPPINGS(const X509V3_EXT_METHOD *method, void *a,
                    STACK_OF(CONF_VALUE) *ext_list)
{
        POLICY_MAPPINGS *pmaps = a;
        POLICY_MAPPING *pmap;
        int i;
        char obj_tmp1[80];
        char obj_tmp2[80];
        for(i = 0; i < sk_POLICY_MAPPING_num(pmaps); i++) {
                pmap = sk_POLICY_MAPPING_value(pmaps, i);
                i2t_ASN1_OBJECT(obj_tmp1, 80, pmap->issuerDomainPolicy);
                i2t_ASN1_OBJECT(obj_tmp2, 80, pmap->subjectDomainPolicy);
                X509V3_add_value(obj_tmp1, obj_tmp2, &ext_list);
        }
        return ext_list;
}

static void *v2i_POLICY_MAPPINGS(const X509V3_EXT_METHOD *method,
                                 X509V3_CTX *ctx, STACK_OF(CONF_VALUE) *nval)
{
        POLICY_MAPPINGS *pmaps;
        POLICY_MAPPING *pmap;
        ASN1_OBJECT *obj1, *obj2;
        CONF_VALUE *val;
        int i;

        if(!(pmaps = sk_POLICY_MAPPING_new_null())) {
                X509V3err(X509V3_F_V2I_POLICY_MAPPINGS,ERR_R_MALLOC_FAILURE);
                return NULL;
        }
```

```
128         for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
129                 val = sk_CONF_VALUE_value(nval, i);
130                 if(!val->value || !val->name) {
131                         sk_POLICY_MAPPING_pop_free(pmaps, POLICY_MAPPING_free);
132                         X509V3err(X509V3_F_V2I_POLICY_MAPPINGS,X509V3_R_INVALID_
133                         X509V3_conf_err(val);
134                         return NULL;
135                 }
136                 obj1 = OBJ_txt2obj(val->name, 0);
137                 obj2 = OBJ_txt2obj(val->value, 0);
138                 if(!obj1 || !obj2) {
139                         sk_POLICY_MAPPING_pop_free(pmaps, POLICY_MAPPING_free);
140                         X509V3err(X509V3_F_V2I_POLICY_MAPPINGS,X509V3_R_INVALID_
141                         X509V3_conf_err(val);
142                         return NULL;
143                 }
144                 pmap = POLICY_MAPPING_new();
145                 if (!pmap) {
146                         sk_POLICY_MAPPING_pop_free(pmaps, POLICY_MAPPING_free);
147                         X509V3err(X509V3_F_V2I_POLICY_MAPPINGS,ERR_R_MALLOC_FAIL
148                         return NULL;
149                 }
150                 pmap->issuerDomainPolicy = obj1;
151                 pmap->subjectDomainPolicy = obj2;
152                 sk_POLICY_MAPPING_push(pmaps, pmap);
153         }
154         return pmaps;
155 }
156 #endif /* ! codereview */
```

```
**********************************************************
    7211 Wed Aug 13 19:53:32 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_prn.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_prn.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* X509 v3 extension utilities */

#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/conf.h>
#include <openssl/x509v3.h>

/* Extension printing routines */

static int unknown_ext_print(BIO *out, X509_EXTENSION *ext, unsigned long flag,

/* Print out a name+value stack */

void X509V3_EXT_val_prn(BIO *out, STACK_OF(CONF_VALUE) *val, int indent, int ml)
{
        int i;
        CONF_VALUE *nval;
        if(!val) return;
        if(!ml || !sk_CONF_VALUE_num(val)) {
                BIO_printf(out, "%*s", indent, "");
                if(!sk_CONF_VALUE_num(val)) BIO_puts(out, "<EMPTY>\n");
        }
        for(i = 0; i < sk_CONF_VALUE_num(val); i++) {
                if(ml) BIO_printf(out, "%*s", indent, "");
                else if(i > 0) BIO_printf(out, ", ");
                nval = sk_CONF_VALUE_value(val, i);
                if(!nval->name) BIO_puts(out, nval->value);
                else if(!nval->value) BIO_puts(out, nval->name);
#ifndef CHARSET_EBCDIC
                else BIO_printf(out, "%s:%s", nval->name, nval->value);
#else
                else {
                        int len;
                        char *tmp;
                        len = strlen(nval->value)+1;
                        tmp = OPENSSL_malloc(len);
                        if (tmp)
                        {
                                ascii2ebcdic(tmp, nval->value, len);
                                BIO_printf(out, "%s:%s", nval->name, tmp);
                                OPENSSL_free(tmp);
                        }
                }
#endif
                if(ml) BIO_puts(out, "\n");
        }
}

/* Main routine: print out a general extension */

int X509V3_EXT_print(BIO *out, X509_EXTENSION *ext, unsigned long flag, int inde
{
        void *ext_str = NULL;
        char *value = NULL;
        const unsigned char *p;
        const X509V3_EXT_METHOD *method;
        STACK_OF(CONF_VALUE) *nval = NULL;
        int ok = 1;

        if(!(method = X509V3_EXT_get(ext)))
                return unknown_ext_print(out, ext, flag, indent, 0);
        p = ext->value->data;
        if(method->it) ext_str = ASN1_item_d2i(NULL, &p, ext->value->length, ASN
        else ext_str = method->d2i(NULL, &p, ext->value->length);

        if(!ext_str) return unknown_ext_print(out, ext, flag, indent, 1);

        if(method->i2s) {
                if(!(value = method->i2s(method, ext_str))) {
                        ok = 0;
```

```
128                         goto err;
129                 }
130 #ifndef CHARSET_EBCDIC
131                 BIO_printf(out, "%*s%s", indent, "", value);
132 #else
133                 {
134                         int len;
135                         char *tmp;
136                         len = strlen(value)+1;
137                         tmp = OPENSSL_malloc(len);
138                         if (tmp)
139                         {
140                                 ascii2ebcdic(tmp, value, len);
141                                 BIO_printf(out, "%*s%s", indent, "", tmp);
142                                 OPENSSL_free(tmp);
143                         }
144                 }
145 #endif
146         } else if(method->i2v) {
147                 if(!(nval = method->i2v(method, ext_str, NULL))) {
148                         ok = 0;
149                         goto err;
150                 }
151                 X509V3_EXT_val_prn(out, nval, indent,
152                                    method->ext_flags & X509V3_EXT_MULTILINE);
153         } else if(method->i2r) {
154                 if(!method->i2r(method, ext_str, out, indent)) ok = 0;
155         } else ok = 0;
156
157         err:
158                 sk_CONF_VALUE_pop_free(nval, X509V3_conf_free);
159                 if(value) OPENSSL_free(value);
160                 if(method->it) ASN1_item_free(ext_str, ASN1_ITEM_ptr(method->it)
161                 else method->ext_free(ext_str);
162                 return ok;
163 }
164
165 int X509V3_extensions_print(BIO *bp, char *title, STACK_OF(X509_EXTENSION) *exts
166 {
167         int i, j;
168
169         if(sk_X509_EXTENSION_num(exts) <= 0) return 1;
170
171         if(title)
172                 {
173                 BIO_printf(bp,"%*s%s:\n",indent, "", title);
174                 indent += 4;
175                 }
176
177         for (i=0; i<sk_X509_EXTENSION_num(exts); i++)
178                 {
179                 ASN1_OBJECT *obj;
180                 X509_EXTENSION *ex;
181                 ex=sk_X509_EXTENSION_value(exts, i);
182                 if (indent && BIO_printf(bp,"%*s",indent, "") <= 0) return 0;
183                 obj=X509_EXTENSION_get_object(ex);
184                 i2a_ASN1_OBJECT(bp,obj);
185                 j=X509_EXTENSION_get_critical(ex);
186                 if (BIO_printf(bp,": %s\n",j?"critical":"") <= 0)
187                         return 0;
188                 if(!X509V3_EXT_print(bp, ex, flag, indent + 4))
189                         {
190                         BIO_printf(bp, "%*s", indent + 4, "");
191                         M_ASN1_OCTET_STRING_print(bp,ex->value);
192                         }
193                 if (BIO_write(bp,"\n",1) <= 0) return 0;
```



```
194                 }
195         return 1;
196 }
197
198 static int unknown_ext_print(BIO *out, X509_EXTENSION *ext, unsigned long flag,
199 {
200         switch(flag & X509V3_EXT_UNKNOWN_MASK) {
201
202                 case X509V3_EXT_DEFAULT:
203                 return 0;
204
205                 case X509V3_EXT_ERROR_UNKNOWN:
206                 if(supported)
207                         BIO_printf(out, "%*s<Parse Error>", indent, "");
208                 else
209                         BIO_printf(out, "%*s<Not Supported>", indent, "");
210                 return 1;
211
212                 case X509V3_EXT_PARSE_UNKNOWN:
213                         return ASN1_parse_dump(out,
214                                 ext->value->data, ext->value->length, indent, -1
215                 case X509V3_EXT_DUMP_UNKNOWN:
216                         return BIO_dump_indent(out, (char *)ext->value->data, ex
217
218                 default:
219                 return 1;
220         }
221 }
222
223
224 #ifndef OPENSSL_NO_FP_API
225 int X509V3_EXT_print_fp(FILE *fp, X509_EXTENSION *ext, int flag, int indent)
226 {
227         BIO *bio_tmp;
228         int ret;
229         if(!(bio_tmp = BIO_new_fp(fp, BIO_NOCLOSE))) return 0;
230         ret = X509V3_EXT_print(bio_tmp, ext, flag, indent);
231         BIO_free(bio_tmp);
232         return ret;
233 }
234 #endif
235 #endif /* ! codereview */
```

```
**********************************************************
   22885 Wed Aug 13 19:53:32 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_purp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_purp.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 2001.
 */
/* ====================================================================
 * Copyright (c) 1999-2004 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/x509v3.h>
```



```
#include <openssl/x509_vfy.h>

static void x509v3_cache_extensions(X509 *x);

static int check_ssl_ca(const X509 *x);
static int check_purpose_ssl_client(const X509_PURPOSE *xp, const X509 *x, int c
static int check_purpose_ssl_server(const X509_PURPOSE *xp, const X509 *x, int c
static int check_purpose_ns_ssl_server(const X509_PURPOSE *xp, const X509 *x, in
static int purpose_smime(const X509 *x, int ca);
static int check_purpose_smime_sign(const X509_PURPOSE *xp, const X509 *x, int c
static int check_purpose_smime_encrypt(const X509_PURPOSE *xp, const X509 *x, in
static int check_purpose_crl_sign(const X509_PURPOSE *xp, const X509 *x, int ca)
static int check_purpose_timestamp_sign(const X509_PURPOSE *xp, const X509 *x, i
static int no_check(const X509_PURPOSE *xp, const X509 *x, int ca);
static int ocsp_helper(const X509_PURPOSE *xp, const X509 *x, int ca);

static int xp_cmp(const X509_PURPOSE * const *a,
                  const X509_PURPOSE * const *b);
static void xptable_free(X509_PURPOSE *p);

static X509_PURPOSE xstandard[] = {
        {X509_PURPOSE_SSL_CLIENT, X509_TRUST_SSL_CLIENT, 0, check_purpose_ssl_cl
        {X509_PURPOSE_SSL_SERVER, X509_TRUST_SSL_SERVER, 0, check_purpose_ssl_se
        {X509_PURPOSE_NS_SSL_SERVER, X509_TRUST_SSL_SERVER, 0, check_purpose_ns_
        {X509_PURPOSE_SMIME_SIGN, X509_TRUST_EMAIL, 0, check_purpose_smime_sign,
        {X509_PURPOSE_SMIME_ENCRYPT, X509_TRUST_EMAIL, 0, check_purpose_smime_en
        {X509_PURPOSE_CRL_SIGN, X509_TRUST_COMPAT, 0, check_purpose_crl_sign, "C
        {X509_PURPOSE_ANY, X509_TRUST_DEFAULT, 0, no_check, "Any Purpose", "any"
        {X509_PURPOSE_OCSP_HELPER, X509_TRUST_COMPAT, 0, ocsp_helper, "OCSP help
        {X509_PURPOSE_TIMESTAMP_SIGN, X509_TRUST_TSA, 0, check_purpose_timestamp
};

#define X509_PURPOSE_COUNT (sizeof(xstandard)/sizeof(X509_PURPOSE))

IMPLEMENT_STACK_OF(X509_PURPOSE)

static STACK_OF(X509_PURPOSE) *xptable = NULL;

static int xp_cmp(const X509_PURPOSE * const *a,
                  const X509_PURPOSE * const *b)
{
        return (*a)->purpose - (*b)->purpose;
}

/* As much as I'd like to make X509_check_purpose use a "const" X509*
 * I really can't because it does recalculate hashes and do other non-const
 * things. */
int X509_check_purpose(X509 *x, int id, int ca)
{
        int idx;
        const X509_PURPOSE *pt;
        if(!(x->ex_flags & EXFLAG_SET)) {
                CRYPTO_w_lock(CRYPTO_LOCK_X509);
                x509v3_cache_extensions(x);
                CRYPTO_w_unlock(CRYPTO_LOCK_X509);
        }
        if(id == -1) return 1;
        idx = X509_PURPOSE_get_by_id(id);
        if(idx == -1) return -1;
        pt = X509_PURPOSE_get0(idx);
        return pt->check_purpose(pt, x, ca);
}

int X509_PURPOSE_set(int *p, int purpose)
{
        if(X509_PURPOSE_get_by_id(purpose) == -1) {
```

```
128                 X509V3err(X509V3_F_X509_PURPOSE_SET, X509V3_R_INVALID_PURPOSE);
129                 return 0;
130         }
131         *p = purpose;
132         return 1;
133 }

135 int X509_PURPOSE_get_count(void)
136 {
137         if(!xptable) return X509_PURPOSE_COUNT;
138         return sk_X509_PURPOSE_num(xptable) + X509_PURPOSE_COUNT;
139 }

141 X509_PURPOSE * X509_PURPOSE_get0(int idx)
142 {
143         if(idx < 0) return NULL;
144         if(idx < (int)X509_PURPOSE_COUNT) return xstandard + idx;
145         return sk_X509_PURPOSE_value(xptable, idx - X509_PURPOSE_COUNT);
146 }

148 int X509_PURPOSE_get_by_sname(char *sname)
149 {
150         int i;
151         X509_PURPOSE *xptmp;
152         for(i = 0; i < X509_PURPOSE_get_count(); i++) {
153                 xptmp = X509_PURPOSE_get0(i);
154                 if(!strcmp(xptmp->sname, sname)) return i;
155         }
156         return -1;
157 }

159 int X509_PURPOSE_get_by_id(int purpose)
160 {
161         X509_PURPOSE tmp;
162         int idx;
163         if((purpose >= X509_PURPOSE_MIN) && (purpose <= X509_PURPOSE_MAX))
164                 return purpose - X509_PURPOSE_MIN;
165         tmp.purpose = purpose;
166         if(!xptable) return -1;
167         idx = sk_X509_PURPOSE_find(xptable, &tmp);
168         if(idx == -1) return -1;
169         return idx + X509_PURPOSE_COUNT;
170 }

172 int X509_PURPOSE_add(int id, int trust, int flags,
173                         int (*ck)(const X509_PURPOSE *, const X509 *, int),
174                                         char *name, char *sname, void *arg)
175 {
176         int idx;
177         X509_PURPOSE *ptmp;
178         /* This is set according to what we change: application can't set it */
179         flags &= ~X509_PURPOSE_DYNAMIC;
180         /* This will always be set for application modified trust entries */
181         flags |= X509_PURPOSE_DYNAMIC_NAME;
182         /* Get existing entry if any */
183         idx = X509_PURPOSE_get_by_id(id);
184         /* Need a new entry */
185         if(idx == -1) {
186                 if(!(ptmp = OPENSSL_malloc(sizeof(X509_PURPOSE)))) {
187                         X509V3err(X509V3_F_X509_PURPOSE_ADD,ERR_R_MALLOC_FAILURE
188                         return 0;
189                 }
190                 ptmp->flags = X509_PURPOSE_DYNAMIC;
191         } else ptmp = X509_PURPOSE_get0(idx);

193         /* OPENSSL_free existing name if dynamic */
```



```
194         if(ptmp->flags & X509_PURPOSE_DYNAMIC_NAME) {
195                 OPENSSL_free(ptmp->name);
196                 OPENSSL_free(ptmp->sname);
197         }
198         /* dup supplied name */
199         ptmp->name = BUF_strdup(name);
200         ptmp->sname = BUF_strdup(sname);
201         if(!ptmp->name || !ptmp->sname) {
202                 X509V3err(X509V3_F_X509_PURPOSE_ADD,ERR_R_MALLOC_FAILURE);
203                 return 0;
204         }
205         /* Keep the dynamic flag of existing entry */
206         ptmp->flags &= X509_PURPOSE_DYNAMIC;
207         /* Set all other flags */
208         ptmp->flags |= flags;

210         ptmp->purpose = id;
211         ptmp->trust = trust;
212         ptmp->check_purpose = ck;
213         ptmp->usr_data = arg;

215         /* If its a new entry manage the dynamic table */
216         if(idx == -1) {
217                 if(!xptable && !(xptable = sk_X509_PURPOSE_new(xp_cmp))) {
218                         X509V3err(X509V3_F_X509_PURPOSE_ADD,ERR_R_MALLOC_FAILURE
219                         return 0;
220                 }
221                 if (!sk_X509_PURPOSE_push(xptable, ptmp)) {
222                         X509V3err(X509V3_F_X509_PURPOSE_ADD,ERR_R_MALLOC_FAILURE
223                         return 0;
224                 }
225         }
226         return 1;
227 }

229 static void xptable_free(X509_PURPOSE *p)
230         {
231         if(!p) return;
232         if (p->flags & X509_PURPOSE_DYNAMIC)
233                 {
234                 if (p->flags & X509_PURPOSE_DYNAMIC_NAME) {
235                         OPENSSL_free(p->name);
236                         OPENSSL_free(p->sname);
237                 }
238                 OPENSSL_free(p);
239                 }
240         }

242 void X509_PURPOSE_cleanup(void)
243 {
244         unsigned int i;
245         sk_X509_PURPOSE_pop_free(xptable, xptable_free);
246         for(i = 0; i < X509_PURPOSE_COUNT; i++) xptable_free(xstandard + i);
247         xptable = NULL;
248 }

250 int X509_PURPOSE_get_id(X509_PURPOSE *xp)
251 {
252         return xp->purpose;
253 }

255 char *X509_PURPOSE_get0_name(X509_PURPOSE *xp)
256 {
257         return xp->name;
258 }
```

```
260 char *X509_PURPOSE_get0_sname(X509_PURPOSE *xp)
261 {
262         return xp->sname;
263 }

265 int X509_PURPOSE_get_trust(X509_PURPOSE *xp)
266 {
267         return xp->trust;
268 }

270 static int nid_cmp(const int *a, const int *b)
271         {
272         return *a - *b;
273         }

275 DECLARE_OBJ_BSEARCH_CMP_FN(int, int, nid);
276 IMPLEMENT_OBJ_BSEARCH_CMP_FN(int, int, nid);

278 int X509_supported_extension(X509_EXTENSION *ex)
279         {
280         /* This table is a list of the NIDs of supported extensions:
281          * that is those which are used by the verify process. If
282          * an extension is critical and doesn't appear in this list
283          * then the verify process will normally reject the certificate.
284          * The list must be kept in numerical order because it will be
285          * searched using bsearch.
286          */

288         static const int supported_nids[] = {
289                 NID_netscape_cert_type, /* 71 */
290                 NID_key_usage,          /* 83 */
291                 NID_subject_alt_name,   /* 85 */
292                 NID_basic_constraints,  /* 87 */
293                 NID_certificate_policies, /* 89 */
294                 NID_ext_key_usage,      /* 126 */
295 #ifndef OPENSSL_NO_RFC3779
296                 NID_sbgp_ipAddrBlock,   /* 290 */
297                 NID_sbgp_autonomousSysNum, /* 291 */
298 #endif
299                 NID_policy_constraints, /* 401 */
300                 NID_proxyCertInfo,      /* 663 */
301                 NID_name_constraints,   /* 666 */
302                 NID_policy_mappings,    /* 747 */
303                 NID_inhibit_any_policy  /* 748 */
304         };

306         int ex_nid = OBJ_obj2nid(X509_EXTENSION_get_object(ex));

308         if (ex_nid == NID_undef)
309                 return 0;

311         if (OBJ_bsearch_nid(&ex_nid, supported_nids,
312                         sizeof(supported_nids)/sizeof(int)))
313                 return 1;
314         return 0;
315         }

317 static void setup_dp(X509 *x, DIST_POINT *dp)
318         {
319         X509_NAME *iname = NULL;
320         int i;
321         if (dp->reasons)
322                 {
323                 if (dp->reasons->length > 0)
324                         dp->dp_reasons = dp->reasons->data[0];
325                 if (dp->reasons->length > 1)
```



```
326                         dp->dp_reasons |= (dp->reasons->data[1] << 8);
327                 dp->dp_reasons &= CRLDP_ALL_REASONS;
328                 }
329         else
330                 dp->dp_reasons = CRLDP_ALL_REASONS;
331         if (!dp->distpoint || (dp->distpoint->type != 1))
332                 return;
333         for (i = 0; i < sk_GENERAL_NAME_num(dp->CRLissuer); i++)
334                 {
335                 GENERAL_NAME *gen = sk_GENERAL_NAME_value(dp->CRLissuer, i);
336                 if (gen->type == GEN_DIRNAME)
337                         {
338                         iname = gen->d.directoryName;
339                         break;
340                         }
341                 }
342         if (!iname)
343                 iname = X509_get_issuer_name(x);

345         DIST_POINT_set_dpname(dp->distpoint, iname);

347         }

349 static void setup_crldp(X509 *x)
350         {
351         int i;
352         x->crldp = X509_get_ext_d2i(x, NID_crl_distribution_points, NULL, NULL);
353         for (i = 0; i < sk_DIST_POINT_num(x->crldp); i++)
354                 setup_dp(x, sk_DIST_POINT_value(x->crldp, i));
355         }

357 static void x509v3_cache_extensions(X509 *x)
358 {
359         BASIC_CONSTRAINTS *bs;
360         PROXY_CERT_INFO_EXTENSION *pci;
361         ASN1_BIT_STRING *usage;
362         ASN1_BIT_STRING *ns;
363         EXTENDED_KEY_USAGE *extusage;
364         X509_EXTENSION *ex;

366         int i;
367         if(x->ex_flags & EXFLAG_SET) return;
368 #ifndef OPENSSL_NO_SHA
369         X509_digest(x, EVP_sha1(), x->sha1_hash, NULL);
370 #endif
371         /* Does subject name match issuer ? */
372         if(!X509_NAME_cmp(X509_get_subject_name(x), X509_get_issuer_name(x)))
373                         x->ex_flags |= EXFLAG_SI;
374         /* V1 should mean no extensions ... */
375         if(!X509_get_version(x)) x->ex_flags |= EXFLAG_V1;
376         /* Handle basic constraints */
377         if((bs=X509_get_ext_d2i(x, NID_basic_constraints, NULL, NULL))) {
378                 if(bs->ca) x->ex_flags |= EXFLAG_CA;
379                 if(bs->pathlen) {
380                         if((bs->pathlen->type == V_ASN1_NEG_INTEGER)
381                                                 || !bs->ca) {
382                                 x->ex_flags |= EXFLAG_INVALID;
383                                 x->ex_pathlen = 0;
384                         } else x->ex_pathlen = ASN1_INTEGER_get(bs->pathlen);
385                 } else x->ex_pathlen = -1;
386                 BASIC_CONSTRAINTS_free(bs);
387                 x->ex_flags |= EXFLAG_BCONS;
388         }
389         /* Handle proxy certificates */
390         if((pci=X509_get_ext_d2i(x, NID_proxyCertInfo, NULL, NULL))) {
391                 if (x->ex_flags & EXFLAG_CA
```

```
392                     || X509_get_ext_by_NID(x, NID_subject_alt_name, -1) >= 0
393                     || X509_get_ext_by_NID(x, NID_issuer_alt_name, -1) >= 0) {
394                         x->ex_flags |= EXFLAG_INVALID;
395                 }
396                 if (pci->pcPathLengthConstraint) {
397                         x->ex_pcpathlen =
398                                 ASN1_INTEGER_get(pci->pcPathLengthConstraint);
399                 } else x->ex_pcpathlen = -1;
400                 PROXY_CERT_INFO_EXTENSION_free(pci);
401                 x->ex_flags |= EXFLAG_PROXY;
402         }
403         /* Handle key usage */
404         if((usage=X509_get_ext_d2i(x, NID_key_usage, NULL, NULL))) {
405                 if(usage->length > 0) {
406                         x->ex_kusage = usage->data[0];
407                         if(usage->length > 1)
408                                 x->ex_kusage |= usage->data[1] << 8;
409                 } else x->ex_kusage = 0;
410                 x->ex_flags |= EXFLAG_KUSAGE;
411                 ASN1_BIT_STRING_free(usage);
412         }
413         x->ex_xkusage = 0;
414         if((extusage=X509_get_ext_d2i(x, NID_ext_key_usage, NULL, NULL))) {
415                 x->ex_flags |= EXFLAG_XKUSAGE;
416                 for(i = 0; i < sk_ASN1_OBJECT_num(extusage); i++) {
417                         switch(OBJ_obj2nid(sk_ASN1_OBJECT_value(extusage,i))) {
418                                 case NID_server_auth:
419                                 x->ex_xkusage |= XKU_SSL_SERVER;
420                                 break;
421
422                                 case NID_client_auth:
423                                 x->ex_xkusage |= XKU_SSL_CLIENT;
424                                 break;
425
426                                 case NID_email_protect:
427                                 x->ex_xkusage |= XKU_SMIME;
428                                 break;
429
430                                 case NID_code_sign:
431                                 x->ex_xkusage |= XKU_CODE_SIGN;
432                                 break;
433
434                                 case NID_ms_sgc:
435                                 case NID_ns_sgc:
436                                 x->ex_xkusage |= XKU_SGC;
437                                 break;
438
439                                 case NID_OCSP_sign:
440                                 x->ex_xkusage |= XKU_OCSP_SIGN;
441                                 break;
442
443                                 case NID_time_stamp:
444                                 x->ex_xkusage |= XKU_TIMESTAMP;
445                                 break;
446
447                                 case NID_dvcs:
448                                 x->ex_xkusage |= XKU_DVCS;
449                                 break;
450                         }
451                 }
452                 sk_ASN1_OBJECT_pop_free(extusage, ASN1_OBJECT_free);
453         }
454
455         if((ns=X509_get_ext_d2i(x, NID_netscape_cert_type, NULL, NULL))) {
456                 if(ns->length > 0) x->ex_nscert = ns->data[0];
457                 else x->ex_nscert = 0;
```



```
458                 x->ex_flags |= EXFLAG_NSCERT;
459                 ASN1_BIT_STRING_free(ns);
460         }
461         x->skid =X509_get_ext_d2i(x, NID_subject_key_identifier, NULL, NULL);
462         x->akid =X509_get_ext_d2i(x, NID_authority_key_identifier, NULL, NULL);
463         x->altname = X509_get_ext_d2i(x, NID_subject_alt_name, NULL, NULL);
464         x->nc = X509_get_ext_d2i(x, NID_name_constraints, &i, NULL);
465         if (!x->nc && (i != -1))
466                 x->ex_flags |= EXFLAG_INVALID;
467         setup_crldp(x);
468
469 #ifndef OPENSSL_NO_RFC3779
470         x->rfc3779_addr =X509_get_ext_d2i(x, NID_sbgp_ipAddrBlock, NULL, NULL);
471         x->rfc3779_asid =X509_get_ext_d2i(x, NID_sbgp_autonomousSysNum,
472                                           NULL, NULL);
473 #endif
474         for (i = 0; i < X509_get_ext_count(x); i++)
475                 {
476                 ex = X509_get_ext(x, i);
477                 if (OBJ_obj2nid(X509_EXTENSION_get_object(ex))
478                                         == NID_freshest_crl)
479                         x->ex_flags |= EXFLAG_FRESHEST;
480                 if (!X509_EXTENSION_get_critical(ex))
481                         continue;
482                 if (!X509_supported_extension(ex))
483                         {
484                         x->ex_flags |= EXFLAG_CRITICAL;
485                         break;
486                         }
487                 }
488         x->ex_flags |= EXFLAG_SET;
489 }
490
491 /* CA checks common to all purposes
492  * return codes:
493  * 0 not a CA
494  * 1 is a CA
495  * 2 basicConstraints absent so "maybe" a CA
496  * 3 basicConstraints absent but self signed V1.
497  * 4 basicConstraints absent but keyUsage present and keyCertSign asserted.
498  */
499
500 #define V1_ROOT (EXFLAG_V1|EXFLAG_SS)
501 #define ku_reject(x, usage) \
502         (((x)->ex_flags & EXFLAG_KUSAGE) && !((x)->ex_kusage & (usage)))
503 #define xku_reject(x, usage) \
504         (((x)->ex_flags & EXFLAG_XKUSAGE) && !((x)->ex_xkusage & (usage)))
505 #define ns_reject(x, usage) \
506         (((x)->ex_flags & EXFLAG_NSCERT) && !((x)->ex_nscert & (usage)))
507
508 static int check_ca(const X509 *x)
509 {
510         /* keyUsage if present should allow cert signing */
511         if(ku_reject(x, KU_KEY_CERT_SIGN)) return 0;
512         if(x->ex_flags & EXFLAG_BCONS) {
513                 if(x->ex_flags & EXFLAG_CA) return 1;
514                 /* If basicConstraints says not a CA then say so */
515                 else return 0;
516         } else {
517                 /* we support V1 roots for...  uh, I don't really know why. */
518                 if((x->ex_flags & V1_ROOT) == V1_ROOT) return 3;
519                 /* If key usage present it must have certSign so tolerate it */
520                 else if (x->ex_flags & EXFLAG_KUSAGE) return 4;
521                 /* Older certificates could have Netscape-specific CA types */
522                 else if (x->ex_flags & EXFLAG_NSCERT
523                                 && x->ex_nscert & NS_ANY_CA) return 5;
```

```
524                 /* can this still be regarded a CA certificate?  I doubt it */
525                 return 0;
526         }
527 }

529 int X509_check_ca(X509 *x)
530 {
531         if(!(x->ex_flags & EXFLAG_SET)) {
532                 CRYPTO_w_lock(CRYPTO_LOCK_X509);
533                 x509v3_cache_extensions(x);
534                 CRYPTO_w_unlock(CRYPTO_LOCK_X509);
535         }

537         return check_ca(x);
538 }

540 /* Check SSL CA: common checks for SSL client and server */
541 static int check_ssl_ca(const X509 *x)
542 {
543         int ca_ret;
544         ca_ret = check_ca(x);
545         if(!ca_ret) return 0;
546         /* check nsCertType if present */
547         if(ca_ret != 5 || x->ex_nscert & NS_SSL_CA) return ca_ret;
548         else return 0;
549 }


552 static int check_purpose_ssl_client(const X509_PURPOSE *xp, const X509 *x, int c
553 {
554         if(xku_reject(x,XKU_SSL_CLIENT)) return 0;
555         if(ca) return check_ssl_ca(x);
556         /* We need to do digital signatures with it */
557         if(ku_reject(x,KU_DIGITAL_SIGNATURE)) return 0;
558         /* nsCertType if present should allow SSL client use */
559         if(ns_reject(x, NS_SSL_CLIENT)) return 0;
560         return 1;
561 }

563 static int check_purpose_ssl_server(const X509_PURPOSE *xp, const X509 *x, int c
564 {
565         if(xku_reject(x,XKU_SSL_SERVER|XKU_SGC)) return 0;
566         if(ca) return check_ssl_ca(x);

568         if(ns_reject(x, NS_SSL_SERVER)) return 0;
569         /* Now as for keyUsage: we'll at least need to sign OR encipher */
570         if(ku_reject(x, KU_DIGITAL_SIGNATURE|KU_KEY_ENCIPHERMENT)) return 0;

572         return 1;

574 }

576 static int check_purpose_ns_ssl_server(const X509_PURPOSE *xp, const X509 *x, in
577 {
578         int ret;
579         ret = check_purpose_ssl_server(xp, x, ca);
580         if(!ret || ca) return ret;
581         /* We need to encipher or Netscape complains */
582         if(ku_reject(x, KU_KEY_ENCIPHERMENT)) return 0;
583         return ret;
584 }

586 /* common S/MIME checks */
587 static int purpose_smime(const X509 *x, int ca)
588 {
589         if(xku_reject(x,XKU_SMIME)) return 0;
```



```
590         if(ca) {
591                 int ca_ret;
592                 ca_ret = check_ca(x);
593                 if(!ca_ret) return 0;
594                 /* check nsCertType if present */
595                 if(ca_ret != 5 || x->ex_nscert & NS_SMIME_CA) return ca_ret;
596                 else return 0;
597         }
598         if(x->ex_flags & EXFLAG_NSCERT) {
599                 if(x->ex_nscert & NS_SMIME) return 1;
600                 /* Workaround for some buggy certificates */
601                 if(x->ex_nscert & NS_SSL_CLIENT) return 2;
602                 return 0;
603         }
604         return 1;
605 }

607 static int check_purpose_smime_sign(const X509_PURPOSE *xp, const X509 *x, int c
608 {
609         int ret;
610         ret = purpose_smime(x, ca);
611         if(!ret || ca) return ret;
612         if(ku_reject(x, KU_DIGITAL_SIGNATURE|KU_NON_REPUDIATION)) return 0;
613         return ret;
614 }

616 static int check_purpose_smime_encrypt(const X509_PURPOSE *xp, const X509 *x, in
617 {
618         int ret;
619         ret = purpose_smime(x, ca);
620         if(!ret || ca) return ret;
621         if(ku_reject(x, KU_KEY_ENCIPHERMENT)) return 0;
622         return ret;
623 }

625 static int check_purpose_crl_sign(const X509_PURPOSE *xp, const X509 *x, int ca)
626 {
627         if(ca) {
628                 int ca_ret;
629                 if((ca_ret = check_ca(x)) != 2) return ca_ret;
630                 else return 0;
631         }
632         if(ku_reject(x, KU_CRL_SIGN)) return 0;
633         return 1;
634 }

636 /* OCSP helper: this is *not* a full OCSP check. It just checks that
637  * each CA is valid. Additional checks must be made on the chain.
638  */

640 static int ocsp_helper(const X509_PURPOSE *xp, const X509 *x, int ca)
641 {
642         /* Must be a valid CA.  Should we really support the "I don't know"
643            value (2)? */
644         if(ca) return check_ca(x);
645         /* leaf certificate is checked in OCSP_verify() */
646         return 1;
647 }

649 static int check_purpose_timestamp_sign(const X509_PURPOSE *xp, const X509 *x,
650                                         int ca)
651 {
652         int i_ext;

654         /* If ca is true we must return if this is a valid CA certificate. */
655         if (ca) return check_ca(x);
```

```
657         /*
658          * Check the optional key usage field:
659          * if Key Usage is present, it must be one of digitalSignature
660          * and/or nonRepudiation (other values are not consistent and shall
661          * be rejected).
662          */
663         if ((x->ex_flags & EXFLAG_KUSAGE)
664             && ((x->ex_kusage & ~(KU_NON_REPUDIATION | KU_DIGITAL_SIGNATURE)) ||
665                 !(x->ex_kusage & (KU_NON_REPUDIATION | KU_DIGITAL_SIGNATURE))))
666                 return 0;

668         /* Only time stamp key usage is permitted and it's required. */
669         if (!(x->ex_flags & EXFLAG_XKUSAGE) || x->ex_xkusage != XKU_TIMESTAMP)
670                 return 0;

672         /* Extended Key Usage MUST be critical */
673         i_ext = X509_get_ext_by_NID((X509 *) x, NID_ext_key_usage, -1);
674         if (i_ext >= 0)
675                 {
676                 X509_EXTENSION *ext = X509_get_ext((X509 *) x, i_ext);
677                 if (!X509_EXTENSION_get_critical(ext))
678                         return 0;
679                 }

681         return 1;
682 }

684 static int no_check(const X509_PURPOSE *xp, const X509 *x, int ca)
685 {
686         return 1;
687 }

689 /* Various checks to see if one certificate issued the second.
690  * This can be used to prune a set of possible issuer certificates
691  * which have been looked up using some simple method such as by
692  * subject name.
693  * These are:
694  * 1. Check issuer_name(subject) == subject_name(issuer)
695  * 2. If akid(subject) exists check it matches issuer
696  * 3. If key_usage(issuer) exists check it supports certificate signing
697  * returns 0 for OK, positive for reason for mismatch, reasons match
698  * codes for X509_verify_cert()
699  */

701 int X509_check_issued(X509 *issuer, X509 *subject)
702 {
703         if(X509_NAME_cmp(X509_get_subject_name(issuer),
704                         X509_get_issuer_name(subject)))
705                                 return X509_V_ERR_SUBJECT_ISSUER_MISMATCH;
706         x509v3_cache_extensions(issuer);
707         x509v3_cache_extensions(subject);

709         if(subject->akid)
710                 {
711                 int ret = X509_check_akid(issuer, subject->akid);
712                 if (ret != X509_V_OK)
713                         return ret;
714                 }

716         if(subject->ex_flags & EXFLAG_PROXY)
717                 {
718                 if(ku_reject(issuer, KU_DIGITAL_SIGNATURE))
719                         return X509_V_ERR_KEYUSAGE_NO_DIGITAL_SIGNATURE;
720                 }
721         else if(ku_reject(issuer, KU_KEY_CERT_SIGN))
```



```
722                 return X509_V_ERR_KEYUSAGE_NO_CERTSIGN;
723         return X509_V_OK;
724 }

726 int X509_check_akid(X509 *issuer, AUTHORITY_KEYID *akid)
727         {

729         if(!akid)
730                 return X509_V_OK;

732         /* Check key ids (if present) */
733         if(akid->keyid && issuer->skid &&
734                  ASN1_OCTET_STRING_cmp(akid->keyid, issuer->skid) )
735                                 return X509_V_ERR_AKID_SKID_MISMATCH;
736         /* Check serial number */
737         if(akid->serial &&
738                 ASN1_INTEGER_cmp(X509_get_serialNumber(issuer), akid->serial))
739                                 return X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH;
740         /* Check issuer name */
741         if(akid->issuer)
742                 {
743                 /* Ugh, for some peculiar reason AKID includes
744                  * SEQUENCE OF GeneralName. So look for a DirName.
745                  * There may be more than one but we only take any
746                  * notice of the first.
747                  */
748                 GENERAL_NAMES *gens;
749                 GENERAL_NAME *gen;
750                 X509_NAME *nm = NULL;
751                 int i;
752                 gens = akid->issuer;
753                 for(i = 0; i < sk_GENERAL_NAME_num(gens); i++)
754                         {
755                         gen = sk_GENERAL_NAME_value(gens, i);
756                         if(gen->type == GEN_DIRNAME)
757                                 {
758                                 nm = gen->d.dirn;
759                                 break;
760                                 }
761                         }
762                 if(nm && X509_NAME_cmp(nm, X509_get_issuer_name(issuer)))
763                         return X509_V_ERR_AKID_ISSUER_SERIAL_MISMATCH;
764                 }
765         return X509_V_OK;
766         }
767 #endif /* ! codereview */
```

```
**********************************************************
    4738 Wed Aug 13 19:53:32 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_skey.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_skey.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


#include <stdio.h>
#include "cryptlib.h"
```



```
#include <openssl/x509v3.h>

static ASN1_OCTET_STRING *s2i_skey_id(X509V3_EXT_METHOD *method, X509V3_CTX *ctx
const X509V3_EXT_METHOD v3_skey_id = {
NID_subject_key_identifier, 0, ASN1_ITEM_ref(ASN1_OCTET_STRING),
0,0,0,0,
(X509V3_EXT_I2S)i2s_ASN1_OCTET_STRING,
(X509V3_EXT_S2I)s2i_skey_id,
0,0,0,0,
NULL};

char *i2s_ASN1_OCTET_STRING(X509V3_EXT_METHOD *method,
             ASN1_OCTET_STRING *oct)
{
        return hex_to_string(oct->data, oct->length);
}

ASN1_OCTET_STRING *s2i_ASN1_OCTET_STRING(X509V3_EXT_METHOD *method,
             X509V3_CTX *ctx, char *str)
{
        ASN1_OCTET_STRING *oct;
        long length;

        if(!(oct = M_ASN1_OCTET_STRING_new())) {
                X509V3err(X509V3_F_S2I_ASN1_OCTET_STRING,ERR_R_MALLOC_FAILURE);
                return NULL;
        }

        if(!(oct->data = string_to_hex(str, &length))) {
                M_ASN1_OCTET_STRING_free(oct);
                return NULL;
        }

        oct->length = length;

        return oct;

}

static ASN1_OCTET_STRING *s2i_skey_id(X509V3_EXT_METHOD *method,
             X509V3_CTX *ctx, char *str)
{
        ASN1_OCTET_STRING *oct;
        ASN1_BIT_STRING *pk;
        unsigned char pkey_dig[EVP_MAX_MD_SIZE];
        unsigned int diglen;

        if(strcmp(str, "hash")) return s2i_ASN1_OCTET_STRING(method, ctx, str);

        if(!(oct = M_ASN1_OCTET_STRING_new())) {
                X509V3err(X509V3_F_S2I_SKEY_ID,ERR_R_MALLOC_FAILURE);
                return NULL;
        }

        if(ctx && (ctx->flags == CTX_TEST)) return oct;

        if(!ctx || (!ctx->subject_req && !ctx->subject_cert)) {
                X509V3err(X509V3_F_S2I_SKEY_ID,X509V3_R_NO_PUBLIC_KEY);
                goto err;
        }

        if(ctx->subject_req)
                pk = ctx->subject_req->req_info->pubkey->public_key;
        else pk = ctx->subject_cert->cert_info->key->public_key;

        if(!pk) {
```

```
128                 X509V3err(X509V3_F_S2I_SKEY_ID,X509V3_R_NO_PUBLIC_KEY);
129                 goto err;
130         }

132         if (!EVP_Digest(pk->data, pk->length, pkey_dig, &diglen, EVP_sha1(), NUL
133                 goto err;

135         if(!M_ASN1_OCTET_STRING_set(oct, pkey_dig, diglen)) {
136                 X509V3err(X509V3_F_S2I_SKEY_ID,ERR_R_MALLOC_FAILURE);
137                 goto err;
138         }

140         return oct;

142         err:
143         M_ASN1_OCTET_STRING_free(oct);
144         return NULL;
145 }
146 #endif /* ! codereview */
```

```
**********************************************************
    7827 Wed Aug 13 19:53:33 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_sxnet.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_sxnet.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project 1999.
 */
/* ====================================================================
 * Copyright (c) 1999 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "cryptlib.h"
#include <openssl/conf.h>
```



```
#include <openssl/asn1.h>
#include <openssl/asn1t.h>
#include <openssl/x509v3.h>

/* Support for Thawte strong extranet extension */

#define SXNET_TEST

static int sxnet_i2r(X509V3_EXT_METHOD *method, SXNET *sx, BIO *out, int indent)
#ifdef SXNET_TEST
static SXNET * sxnet_v2i(X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
                                                    STACK_OF(CONF_VALUE) *nval);
#endif
const X509V3_EXT_METHOD v3_sxnet = {
NID_sxnet, X509V3_EXT_MULTILINE, ASN1_ITEM_ref(SXNET),
0,0,0,0,
0,0,
0,
#ifdef SXNET_TEST
(X509V3_EXT_V2I)sxnet_v2i,
#else
0,
#endif
(X509V3_EXT_I2R)sxnet_i2r,
0,
NULL
};

ASN1_SEQUENCE(SXNETID) = {
        ASN1_SIMPLE(SXNETID, zone, ASN1_INTEGER),
        ASN1_SIMPLE(SXNETID, user, ASN1_OCTET_STRING)
} ASN1_SEQUENCE_END(SXNETID)

IMPLEMENT_ASN1_FUNCTIONS(SXNETID)

ASN1_SEQUENCE(SXNET) = {
        ASN1_SIMPLE(SXNET, version, ASN1_INTEGER),
        ASN1_SEQUENCE_OF(SXNET, ids, SXNETID)
} ASN1_SEQUENCE_END(SXNET)

IMPLEMENT_ASN1_FUNCTIONS(SXNET)

static int sxnet_i2r(X509V3_EXT_METHOD *method, SXNET *sx, BIO *out,
             int indent)
{
        long v;
        char *tmp;
        SXNETID *id;
        int i;
        v = ASN1_INTEGER_get(sx->version);
        BIO_printf(out, "%*sVersion: %ld (0x%lX)", indent, "", v + 1, v);
        for(i = 0; i < sk_SXNETID_num(sx->ids); i++) {
                id = sk_SXNETID_value(sx->ids, i);
                tmp = i2s_ASN1_INTEGER(NULL, id->zone);
                BIO_printf(out, "\n%*sZone: %s, User: ", indent, "", tmp);
                OPENSSL_free(tmp);
                M_ASN1_OCTET_STRING_print(out, id->user);
        }
        return 1;
}

#ifdef SXNET_TEST

/* NBB: this is used for testing only. It should *not* be used for anything
 * else because it will just take static IDs from the configuration file and
 * they should really be separate values for each user.
```

```
 */


static SXNET * sxnet_v2i(X509V3_EXT_METHOD *method, X509V3_CTX *ctx,
             STACK_OF(CONF_VALUE) *nval)
{
        CONF_VALUE *cnf;
        SXNET *sx = NULL;
        int i;
        for(i = 0; i < sk_CONF_VALUE_num(nval); i++) {
                cnf = sk_CONF_VALUE_value(nval, i);
                if(!SXNET_add_id_asc(&sx, cnf->name, cnf->value, -1))
                                                                return NULL;
        }
        return sx;
}


#endif

/* Strong Extranet utility functions */

/* Add an id given the zone as an ASCII number */

int SXNET_add_id_asc(SXNET **psx, char *zone, char *user,
             int userlen)
{
        ASN1_INTEGER *izone = NULL;
        if(!(izone = s2i_ASN1_INTEGER(NULL, zone))) {
                X509V3err(X509V3_F_SXNET_ADD_ID_ASC,X509V3_R_ERROR_CONVERTING_ZO
                return 0;
        }
        return SXNET_add_id_INTEGER(psx, izone, user, userlen);
}

/* Add an id given the zone as an unsigned long */

int SXNET_add_id_ulong(SXNET **psx, unsigned long lzone, char *user,
             int userlen)
{
        ASN1_INTEGER *izone = NULL;
        if(!(izone = M_ASN1_INTEGER_new()) || !ASN1_INTEGER_set(izone, lzone)) {
                X509V3err(X509V3_F_SXNET_ADD_ID_ULONG,ERR_R_MALLOC_FAILURE);
                M_ASN1_INTEGER_free(izone);
                return 0;
        }
        return SXNET_add_id_INTEGER(psx, izone, user, userlen);

}

/* Add an id given the zone as an ASN1_INTEGER.
 * Note this version uses the passed integer and doesn't make a copy so don't
 * free it up afterwards.
 */

int SXNET_add_id_INTEGER(SXNET **psx, ASN1_INTEGER *zone, char *user,
             int userlen)
{
        SXNET *sx = NULL;
        SXNETID *id = NULL;
        if(!psx || !zone || !user) {
                X509V3err(X509V3_F_SXNET_ADD_ID_INTEGER,X509V3_R_INVALID_NULL_AR
                return 0;
        }
        if(userlen == -1) userlen = strlen(user);
        if(userlen > 64) {
```



```
                X509V3err(X509V3_F_SXNET_ADD_ID_INTEGER,X509V3_R_USER_TOO_LONG);
                return 0;
        }
        if(!*psx) {
                if(!(sx = SXNET_new())) goto err;
                if(!ASN1_INTEGER_set(sx->version, 0)) goto err;
                *psx = sx;
        } else sx = *psx;
        if(SXNET_get_id_INTEGER(sx, zone)) {
                X509V3err(X509V3_F_SXNET_ADD_ID_INTEGER,X509V3_R_DUPLICATE_ZONE_
                return 0;
        }

        if(!(id = SXNETID_new())) goto err;
        if(userlen == -1) userlen = strlen(user);

        if(!M_ASN1_OCTET_STRING_set(id->user, user, userlen)) goto err;
        if(!sk_SXNETID_push(sx->ids, id)) goto err;
        id->zone = zone;
        return 1;

        err:
        X509V3err(X509V3_F_SXNET_ADD_ID_INTEGER,ERR_R_MALLOC_FAILURE);
        SXNETID_free(id);
        SXNET_free(sx);
        *psx = NULL;
        return 0;
}

ASN1_OCTET_STRING *SXNET_get_id_asc(SXNET *sx, char *zone)
{
        ASN1_INTEGER *izone = NULL;
        ASN1_OCTET_STRING *oct;
        if(!(izone = s2i_ASN1_INTEGER(NULL, zone))) {
                X509V3err(X509V3_F_SXNET_GET_ID_ASC,X509V3_R_ERROR_CONVERTING_ZO
                return NULL;
        }
        oct = SXNET_get_id_INTEGER(sx, izone);
        M_ASN1_INTEGER_free(izone);
        return oct;
}

ASN1_OCTET_STRING *SXNET_get_id_ulong(SXNET *sx, unsigned long lzone)
{
        ASN1_INTEGER *izone = NULL;
        ASN1_OCTET_STRING *oct;
        if(!(izone = M_ASN1_INTEGER_new()) || !ASN1_INTEGER_set(izone, lzone)) {
                X509V3err(X509V3_F_SXNET_GET_ID_ULONG,ERR_R_MALLOC_FAILURE);
                M_ASN1_INTEGER_free(izone);
                return NULL;
        }
        oct = SXNET_get_id_INTEGER(sx, izone);
        M_ASN1_INTEGER_free(izone);
        return oct;
}

ASN1_OCTET_STRING *SXNET_get_id_INTEGER(SXNET *sx, ASN1_INTEGER *zone)
{
        SXNETID *id;
        int i;
        for(i = 0; i < sk_SXNETID_num(sx->ids); i++) {
                id = sk_SXNETID_value(sx->ids, i);
                if(!M_ASN1_INTEGER_cmp(id->zone, zone)) return id->user;
        }
        return NULL;
}
```

```
261 IMPLEMENT_STACK_OF(SXNETID)
262 IMPLEMENT_ASN1_SET_OF(SXNETID)
263 #endif /* ! codereview */
```

```
**********************************************************
   21051 Wed Aug 13 19:53:33 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3_utl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* v3_utl.c */
/* Written by Dr Stephen N Henson (steve@openssl.org) for the OpenSSL
 * project.
 */
/* ====================================================================
 * Copyright (c) 1999-2003 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* X509 v3 extension utilities */


#include <stdio.h>
```



```
#include <ctype.h>
#include "cryptlib.h"
#include <openssl/conf.h>
#include <openssl/x509v3.h>
#include <openssl/bn.h>

static char *strip_spaces(char *name);
static int sk_strcmp(const char * const *a, const char * const *b);
static STACK_OF(OPENSSL_STRING) *get_email(X509_NAME *name, GENERAL_NAMES *gens);
static void str_free(OPENSSL_STRING str);
static int append_ia5(STACK_OF(OPENSSL_STRING) **sk, ASN1_IA5STRING *email);

static int ipv4_from_asc(unsigned char *v4, const char *in);
static int ipv6_from_asc(unsigned char *v6, const char *in);
static int ipv6_cb(const char *elem, int len, void *usr);
static int ipv6_hex(unsigned char *out, const char *in, int inlen);

/* Add a CONF_VALUE name value pair to stack */

int X509V3_add_value(const char *name, const char *value,
                                                STACK_OF(CONF_VALUE) **extlist)
{
        CONF_VALUE *vtmp = NULL;
        char *tname = NULL, *tvalue = NULL;
        if(name && !(tname = BUF_strdup(name))) goto err;
        if(value && !(tvalue = BUF_strdup(value))) goto err;
        if(!(vtmp = (CONF_VALUE *)OPENSSL_malloc(sizeof(CONF_VALUE)))) goto err;
        if(!*extlist && !(*extlist = sk_CONF_VALUE_new_null())) goto err;
        vtmp->section = NULL;
        vtmp->name = tname;
        vtmp->value = tvalue;
        if(!sk_CONF_VALUE_push(*extlist, vtmp)) goto err;
        return 1;
        err:
        X509V3err(X509V3_F_X509V3_ADD_VALUE,ERR_R_MALLOC_FAILURE);
        if(vtmp) OPENSSL_free(vtmp);
        if(tname) OPENSSL_free(tname);
        if(tvalue) OPENSSL_free(tvalue);
        return 0;
}

int X509V3_add_value_uchar(const char *name, const unsigned char *value,
                           STACK_OF(CONF_VALUE) **extlist)
    {
    return X509V3_add_value(name,(const char *)value,extlist);
    }

/* Free function for STACK_OF(CONF_VALUE) */

void X509V3_conf_free(CONF_VALUE *conf)
{
        if(!conf) return;
        if(conf->name) OPENSSL_free(conf->name);
        if(conf->value) OPENSSL_free(conf->value);
        if(conf->section) OPENSSL_free(conf->section);
        OPENSSL_free(conf);
}

int X509V3_add_value_bool(const char *name, int asn1_bool,
                                                STACK_OF(CONF_VALUE) **extlist)
{
        if(asn1_bool) return X509V3_add_value(name, "TRUE", extlist);
        return X509V3_add_value(name, "FALSE", extlist);
}

int X509V3_add_value_bool_nf(char *name, int asn1_bool,
```

```
128                                                 STACK_OF(CONF_VALUE) **extlist)
129 {
130         if(asn1_bool) return X509V3_add_value(name, "TRUE", extlist);
131         return 1;
132 }


135 char *i2s_ASN1_ENUMERATED(X509V3_EXT_METHOD *method, ASN1_ENUMERATED *a)
136 {
137         BIGNUM *bntmp = NULL;
138         char *strtmp = NULL;
139         if(!a) return NULL;
140         if(!(bntmp = ASN1_ENUMERATED_to_BN(a, NULL)) ||
141             !(strtmp = BN_bn2dec(bntmp)) )
142                 X509V3err(X509V3_F_I2S_ASN1_ENUMERATED,ERR_R_MALLOC_FAILURE);
143         BN_free(bntmp);
144         return strtmp;
145 }

147 char *i2s_ASN1_INTEGER(X509V3_EXT_METHOD *method, ASN1_INTEGER *a)
148 {
149         BIGNUM *bntmp = NULL;
150         char *strtmp = NULL;
151         if(!a) return NULL;
152         if(!(bntmp = ASN1_INTEGER_to_BN(a, NULL)) ||
153             !(strtmp = BN_bn2dec(bntmp)) )
154                 X509V3err(X509V3_F_I2S_ASN1_INTEGER,ERR_R_MALLOC_FAILURE);
155         BN_free(bntmp);
156         return strtmp;
157 }

159 ASN1_INTEGER *s2i_ASN1_INTEGER(X509V3_EXT_METHOD *method, char *value)
160 {
161         BIGNUM *bn = NULL;
162         ASN1_INTEGER *aint;
163         int isneg, ishex;
164         int ret;
165         if (!value) {
166                 X509V3err(X509V3_F_S2I_ASN1_INTEGER,X509V3_R_INVALID_NULL_VALUE)
167                 return 0;
168         }
169         bn = BN_new();
170         if (value[0] == '-') {
171                 value++;
172                 isneg = 1;
173         } else isneg = 0;

175         if (value[0] == '0' && ((value[1] == 'x') || (value[1] == 'X'))) {
176                 value += 2;
177                 ishex = 1;
178         } else ishex = 0;

180         if (ishex) ret = BN_hex2bn(&bn, value);
181         else ret = BN_dec2bn(&bn, value);

183         if (!ret || value[ret]) {
184                 BN_free(bn);
185                 X509V3err(X509V3_F_S2I_ASN1_INTEGER,X509V3_R_BN_DEC2BN_ERROR);
186                 return 0;
187         }

189         if (isneg && BN_is_zero(bn)) isneg = 0;

191         aint = BN_to_ASN1_INTEGER(bn, NULL);
192         BN_free(bn);
193         if (!aint) {
```



```
194                 X509V3err(X509V3_F_S2I_ASN1_INTEGER,X509V3_R_BN_TO_ASN1_INTEGER_
195                 return 0;
196         }
197         if (isneg) aint->type |= V_ASN1_NEG;
198         return aint;
199 }

201 int X509V3_add_value_int(const char *name, ASN1_INTEGER *aint,
202                 STACK_OF(CONF_VALUE) **extlist)
203 {
204         char *strtmp;
205         int ret;
206         if(!aint) return 1;
207         if(!(strtmp = i2s_ASN1_INTEGER(NULL, aint))) return 0;
208         ret = X509V3_add_value(name, strtmp, extlist);
209         OPENSSL_free(strtmp);
210         return ret;
211 }

213 int X509V3_get_value_bool(CONF_VALUE *value, int *asn1_bool)
214 {
215         char *btmp;
216         if(!(btmp = value->value)) goto err;
217         if(!strcmp(btmp, "TRUE") || !strcmp(btmp, "true")
218                 || !strcmp(btmp, "Y") || !strcmp(btmp, "y")
219                 || !strcmp(btmp, "YES") || !strcmp(btmp, "yes")) {
220                 *asn1_bool = 0xff;
221                 return 1;
222         } else if(!strcmp(btmp, "FALSE") || !strcmp(btmp, "false")
223                 || !strcmp(btmp, "N") || !strcmp(btmp, "n")
224                 || !strcmp(btmp, "NO") || !strcmp(btmp, "no")) {
225                 *asn1_bool = 0;
226                 return 1;
227         }
228         err:
229         X509V3err(X509V3_F_X509V3_GET_VALUE_BOOL,X509V3_R_INVALID_BOOLEAN_STRING
230         X509V3_conf_err(value);
231         return 0;
232 }

234 int X509V3_get_value_int(CONF_VALUE *value, ASN1_INTEGER **aint)
235 {
236         ASN1_INTEGER *itmp;
237         if(!(itmp = s2i_ASN1_INTEGER(NULL, value->value))) {
238                 X509V3_conf_err(value);
239                 return 0;
240         }
241         *aint = itmp;
242         return 1;
243 }

245 #define HDR_NAME        1
246 #define HDR_VALUE       2

248 /*#define DEBUG*/

250 STACK_OF(CONF_VALUE) *X509V3_parse_list(const char *line)
251 {
252         char *p, *q, c;
253         char *ntmp, *vtmp;
254         STACK_OF(CONF_VALUE) *values = NULL;
255         char *linebuf;
256         int state;
257         /* We are going to modify the line so copy it first */
258         linebuf = BUF_strdup(line);
259         state = HDR_NAME;
```

```
260         ntmp = NULL;
261         /* Go through all characters */
262         for(p = linebuf, q = linebuf; (c = *p) && (c!='\r') && (c!='\n'); p++) {

264                 switch(state) {
265                         case HDR_NAME:
266                         if(c == ':') {
267                                 state = HDR_VALUE;
268                                 *p = 0;
269                                 ntmp = strip_spaces(q);
270                                 if(!ntmp) {
271                                         X509V3err(X509V3_F_X509V3_PARSE_LIST, X5
272                                         goto err;
273                                 }
274                                 q = p + 1;
275                         } else if(c == ',') {
276                                 *p = 0;
277                                 ntmp = strip_spaces(q);
278                                 q = p + 1;
279 #if 0
280                                 printf("%s\n", ntmp);
281 #endif
282                                 if(!ntmp) {
283                                         X509V3err(X509V3_F_X509V3_PARSE_LIST, X5
284                                         goto err;
285                                 }
286                                 X509V3_add_value(ntmp, NULL, &values);
287                         }
288                         break ;

290                         case HDR_VALUE:
291                         if(c == ',') {
292                                 state = HDR_NAME;
293                                 *p = 0;
294                                 vtmp = strip_spaces(q);
295 #if 0
296                                 printf("%s\n", ntmp);
297 #endif
298                                 if(!vtmp) {
299                                         X509V3err(X509V3_F_X509V3_PARSE_LIST, X5
300                                         goto err;
301                                 }
302                                 X509V3_add_value(ntmp, vtmp, &values);
303                                 ntmp = NULL;
304                                 q = p + 1;
305                         }

307                 }
308         }

310         if(state == HDR_VALUE) {
311                 vtmp = strip_spaces(q);
312 #if 0
313                 printf("%s=%s\n", ntmp, vtmp);
314 #endif
315                 if(!vtmp) {
316                         X509V3err(X509V3_F_X509V3_PARSE_LIST, X509V3_R_INVALID_N
317                         goto err;
318                 }
319                 X509V3_add_value(ntmp, vtmp, &values);
320         } else {
321                 ntmp = strip_spaces(q);
322 #if 0
323                 printf("%s\n", ntmp);
324 #endif
325                 if(!ntmp) {
```



```
326                         X509V3err(X509V3_F_X509V3_PARSE_LIST, X509V3_R_INVALID_N
327                         goto err;
328                 }
329                 X509V3_add_value(ntmp, NULL, &values);
330         }
331 OPENSSL_free(linebuf);
332 return values;

334 err:
335 OPENSSL_free(linebuf);
336 sk_CONF_VALUE_pop_free(values, X509V3_conf_free);
337 return NULL;

339 }

341 /* Delete leading and trailing spaces from a string */
342 static char *strip_spaces(char *name)
343 {
344         char *p, *q;
345         /* Skip over leading spaces */
346         p = name;
347         while(*p && isspace((unsigned char)*p)) p++;
348         if(!*p) return NULL;
349         q = p + strlen(p) - 1;
350         while((q != p) && isspace((unsigned char)*q)) q--;
351         if(p != q) q[1] = 0;
352         if(!*p) return NULL;
353         return p;
354 }

356 /* hex string utilities */

358 /* Given a buffer of length 'len' return a OPENSSL_malloc'ed string with its
359  * hex representation
360  * @@@ (Contents of buffer are always kept in ASCII, also on EBCDIC machines)
361  */

363 char *hex_to_string(const unsigned char *buffer, long len)
364 {
365         char *tmp, *q;
366         const unsigned char *p;
367         int i;
368         static const char hexdig[] = "0123456789ABCDEF";
369         if(!buffer || !len) return NULL;
370         if(!(tmp = OPENSSL_malloc(len * 3 + 1))) {
371                 X509V3err(X509V3_F_HEX_TO_STRING,ERR_R_MALLOC_FAILURE);
372                 return NULL;
373         }
374         q = tmp;
375         for(i = 0, p = buffer; i < len; i++,p++) {
376                 *q++ = hexdig[(*p >> 4) & 0xf];
377                 *q++ = hexdig[*p & 0xf];
378                 *q++ = ':';
379         }
380         q[-1] = 0;
381 #ifdef CHARSET_EBCDIC
382         ebcdic2ascii(tmp, tmp, q - tmp - 1);
383 #endif

385         return tmp;
386 }

388 /* Give a string of hex digits convert to
389  * a buffer
390  */
```

```
392 unsigned char *string_to_hex(const char *str, long *len)
393 {
394         unsigned char *hexbuf, *q;
395         unsigned char ch, cl, *p;
396         if(!str) {
397                 X509V3err(X509V3_F_STRING_TO_HEX,X509V3_R_INVALID_NULL_ARGUMENT)
398                 return NULL;
399         }
400         if(!(hexbuf = OPENSSL_malloc(strlen(str) >> 1))) goto err;
401         for(p = (unsigned char *)str, q = hexbuf; *p;) {
402                 ch = *p++;
403 #ifdef CHARSET_EBCDIC
404                 ch = os_toebcdic[ch];
405 #endif
406                 if(ch == ':') continue;
407                 cl = *p++;
408 #ifdef CHARSET_EBCDIC
409                 cl = os_toebcdic[cl];
410 #endif
411                 if(!cl) {
412                         X509V3err(X509V3_F_STRING_TO_HEX,X509V3_R_ODD_NUMBER_OF_
413                         OPENSSL_free(hexbuf);
414                         return NULL;
415                 }
416                 if(isupper(ch)) ch = tolower(ch);
417                 if(isupper(cl)) cl = tolower(cl);

419                 if((ch >= '0') && (ch <= '9')) ch -= '0';
420                 else if ((ch >= 'a') && (ch <= 'f')) ch -= 'a' - 10;
421                 else goto badhex;

423                 if((cl >= '0') && (cl <= '9')) cl -= '0';
424                 else if ((cl >= 'a') && (cl <= 'f')) cl -= 'a' - 10;
425                 else goto badhex;

427                 *q++ = (ch << 4) | cl;
428         }

430         if(len) *len = q - hexbuf;

432         return hexbuf;

434         err:
435         if(hexbuf) OPENSSL_free(hexbuf);
436         X509V3err(X509V3_F_STRING_TO_HEX,ERR_R_MALLOC_FAILURE);
437         return NULL;

439         badhex:
440         OPENSSL_free(hexbuf);
441         X509V3err(X509V3_F_STRING_TO_HEX,X509V3_R_ILLEGAL_HEX_DIGIT);
442         return NULL;

444 }

446 /* V2I name comparison function: returns zero if 'name' matches
447  * cmp or cmp.*
448  */

450 int name_cmp(const char *name, const char *cmp)
451 {
452         int len, ret;
453         char c;
454         len = strlen(cmp);
455         if((ret = strncmp(name, cmp, len))) return ret;
456         c = name[len];
457         if(!c || (c=='.')) return 0;
```



```
458         return 1;
459 }

461 static int sk_strcmp(const char * const *a, const char * const *b)
462 {
463         return strcmp(*a, *b);
464 }

466 STACK_OF(OPENSSL_STRING) *X509_get1_email(X509 *x)
467 {
468         GENERAL_NAMES *gens;
469         STACK_OF(OPENSSL_STRING) *ret;

471         gens = X509_get_ext_d2i(x, NID_subject_alt_name, NULL, NULL);
472         ret = get_email(X509_get_subject_name(x), gens);
473         sk_GENERAL_NAME_pop_free(gens, GENERAL_NAME_free);
474         return ret;
475 }

477 STACK_OF(OPENSSL_STRING) *X509_get1_ocsp(X509 *x)
478 {
479         AUTHORITY_INFO_ACCESS *info;
480         STACK_OF(OPENSSL_STRING) *ret = NULL;
481         int i;

483         info = X509_get_ext_d2i(x, NID_info_access, NULL, NULL);
484         if (!info)
485                 return NULL;
486         for (i = 0; i < sk_ACCESS_DESCRIPTION_num(info); i++)
487                 {
488                 ACCESS_DESCRIPTION *ad = sk_ACCESS_DESCRIPTION_value(info, i);
489                 if (OBJ_obj2nid(ad->method) == NID_ad_OCSP)
490                         {
491                         if (ad->location->type == GEN_URI)
492                                 {
493                                 if (!append_ia5(&ret, ad->location->d.uniformRes
494                                         break;
495                                 }
496                         }
497                 }
498         AUTHORITY_INFO_ACCESS_free(info);
499         return ret;
500 }

502 STACK_OF(OPENSSL_STRING) *X509_REQ_get1_email(X509_REQ *x)
503 {
504         GENERAL_NAMES *gens;
505         STACK_OF(X509_EXTENSION) *exts;
506         STACK_OF(OPENSSL_STRING) *ret;

508         exts = X509_REQ_get_extensions(x);
509         gens = X509V3_get_d2i(exts, NID_subject_alt_name, NULL, NULL);
510         ret = get_email(X509_REQ_get_subject_name(x), gens);
511         sk_GENERAL_NAME_pop_free(gens, GENERAL_NAME_free);
512         sk_X509_EXTENSION_pop_free(exts, X509_EXTENSION_free);
513         return ret;
514 }


517 static STACK_OF(OPENSSL_STRING) *get_email(X509_NAME *name, GENERAL_NAMES *gens)
518 {
519         STACK_OF(OPENSSL_STRING) *ret = NULL;
520         X509_NAME_ENTRY *ne;
521         ASN1_IA5STRING *email;
522         GENERAL_NAME *gen;
523         int i;
```

```
524         /* Now add any email address(es) to STACK */
525         i = -1;
526         /* First supplied X509_NAME */
527         while((i = X509_NAME_get_index_by_NID(name,
528                                         NID_pkcs9_emailAddress, i)) >= 0) {
529                 ne = X509_NAME_get_entry(name, i);
530                 email = X509_NAME_ENTRY_get_data(ne);
531                 if(!append_ia5(&ret, email)) return NULL;
532         }
533         for(i = 0; i < sk_GENERAL_NAME_num(gens); i++)
534         {
535                 gen = sk_GENERAL_NAME_value(gens, i);
536                 if(gen->type != GEN_EMAIL) continue;
537                 if(!append_ia5(&ret, gen->d.ia5)) return NULL;
538         }
539         return ret;
540 }

542 static void str_free(OPENSSL_STRING str)
543 {
544         OPENSSL_free(str);
545 }

547 static int append_ia5(STACK_OF(OPENSSL_STRING) **sk, ASN1_IA5STRING *email)
548 {
549         char *emtmp;
550         /* First some sanity checks */
551         if(email->type != V_ASN1_IA5STRING) return 1;
552         if(!email->data || !email->length) return 1;
553         if(!*sk) *sk = sk_OPENSSL_STRING_new(sk_strcmp);
554         if(!*sk) return 0;
555         /* Don't add duplicates */
556         if(sk_OPENSSL_STRING_find(*sk, (char *)email->data) != -1) return 1;
557         emtmp = BUF_strdup((char *)email->data);
558         if(!emtmp || !sk_OPENSSL_STRING_push(*sk, emtmp)) {
559                 X509_email_free(*sk);
560                 *sk = NULL;
561                 return 0;
562         }
563         return 1;
564 }

566 void X509_email_free(STACK_OF(OPENSSL_STRING) *sk)
567 {
568         sk_OPENSSL_STRING_pop_free(sk, str_free);
569 }

571 /* Convert IP addresses both IPv4 and IPv6 into an
572  * OCTET STRING compatible with RFC3280.
573  */

575 ASN1_OCTET_STRING *a2i_IPADDRESS(const char *ipasc)
576         {
577         unsigned char ipout[16];
578         ASN1_OCTET_STRING *ret;
579         int iplen;

581         /* If string contains a ':' assume IPv6 */

583         iplen = a2i_ipadd(ipout, ipasc);

585         if (!iplen)
586                 return NULL;

588         ret = ASN1_OCTET_STRING_new();
589         if (!ret)
```



```
590                 return NULL;
591         if (!ASN1_OCTET_STRING_set(ret, ipout, iplen))
592                 {
593                 ASN1_OCTET_STRING_free(ret);
594                 return NULL;
595                 }
596         return ret;
597         }

599 ASN1_OCTET_STRING *a2i_IPADDRESS_NC(const char *ipasc)
600         {
601         ASN1_OCTET_STRING *ret = NULL;
602         unsigned char ipout[32];
603         char *iptmp = NULL, *p;
604         int iplen1, iplen2;
605         p = strchr(ipasc,'/');
606         if (!p)
607                 return NULL;
608         iptmp = BUF_strdup(ipasc);
609         if (!iptmp)
610                 return NULL;
611         p = iptmp + (p - ipasc);
612         *p++ = 0;

614         iplen1 = a2i_ipadd(ipout, iptmp);

616         if (!iplen1)
617                 goto err;

619         iplen2 = a2i_ipadd(ipout + iplen1, p);

621         OPENSSL_free(iptmp);
622         iptmp = NULL;

624         if (!iplen2 || (iplen1 != iplen2))
625                 goto err;

627         ret = ASN1_OCTET_STRING_new();
628         if (!ret)
629                 goto err;
630         if (!ASN1_OCTET_STRING_set(ret, ipout, iplen1 + iplen2))
631                 goto err;

633         return ret;

635         err:
636         if (iptmp)
637                 OPENSSL_free(iptmp);
638         if (ret)
639                 ASN1_OCTET_STRING_free(ret);
640         return NULL;
641         }


644 int a2i_ipadd(unsigned char *ipout, const char *ipasc)
645         {
646         /* If string contains a ':' assume IPv6 */

648         if (strchr(ipasc, ':'))
649                 {
650                 if (!ipv6_from_asc(ipout, ipasc))
651                         return 0;
652                 return 16;
653                 }
654         else
655                 {
```

```
656                 if (!ipv4_from_asc(ipout, ipasc))
657                         return 0;
658                 return 4;
659                 }
660         }

662 static int ipv4_from_asc(unsigned char *v4, const char *in)
663         {
664         int a0, a1, a2, a3;
665         if (sscanf(in, "%d.%d.%d.%d", &a0, &a1, &a2, &a3) != 4)
666                 return 0;
667         if ((a0 < 0) || (a0 > 255) || (a1 < 0) || (a1 > 255)
668                 || (a2 < 0) || (a2 > 255) || (a3 < 0) || (a3 > 255))
669                 return 0;
670         v4[0] = a0;
671         v4[1] = a1;
672         v4[2] = a2;
673         v4[3] = a3;
674         return 1;
675         }

677 typedef struct {
678                 /* Temporary store for IPV6 output */
679                 unsigned char tmp[16];
680                 /* Total number of bytes in tmp */
681                 int total;
682                 /* The position of a zero (corresponding to '::') */
683                 int zero_pos;
684                 /* Number of zeroes */
685                 int zero_cnt;
686         } IPV6_STAT;


689 static int ipv6_from_asc(unsigned char *v6, const char *in)
690         {
691         IPV6_STAT v6stat;
692         v6stat.total = 0;
693         v6stat.zero_pos = -1;
694         v6stat.zero_cnt = 0;
695         /* Treat the IPv6 representation as a list of values
696          * separated by ':'. The presence of a '::' will parse
697          * as one, two or three zero length elements.
698          */
699         if (!CONF_parse_list(in, ':', 0, ipv6_cb, &v6stat))
700                 return 0;

702         /* Now for some sanity checks */

704         if (v6stat.zero_pos == -1)
705                 {
706                 /* If no '::' must have exactly 16 bytes */
707                 if (v6stat.total != 16)
708                         return 0;
709                 }
710         else
711                 {
712                 /* If '::' must have less than 16 bytes */
713                 if (v6stat.total == 16)
714                         return 0;
715                 /* More than three zeroes is an error */
716                 if (v6stat.zero_cnt > 3)
717                         return 0;
718                 /* Can only have three zeroes if nothing else present */
719                 else if (v6stat.zero_cnt == 3)
720                         {
721                         if (v6stat.total > 0)
```



```
722                                 return 0;
723                         }
724                 /* Can only have two zeroes if at start or end */
725                 else if (v6stat.zero_cnt == 2)
726                         {
727                         if ((v6stat.zero_pos != 0)
728                                 && (v6stat.zero_pos != v6stat.total))
729                                 return 0;
730                         }
731                 else
732                 /* Can only have one zero if *not* start or end */
733                         {
734                         if ((v6stat.zero_pos == 0)
735                                 || (v6stat.zero_pos == v6stat.total))
736                                 return 0;
737                         }
738                 }

740         /* Format result */

742         if (v6stat.zero_pos >= 0)
743                 {
744                 /* Copy initial part */
745                 memcpy(v6, v6stat.tmp, v6stat.zero_pos);
746                 /* Zero middle */
747                 memset(v6 + v6stat.zero_pos, 0, 16 - v6stat.total);
748                 /* Copy final part */
749                 if (v6stat.total != v6stat.zero_pos)
750                         memcpy(v6 + v6stat.zero_pos + 16 - v6stat.total,
751                                 v6stat.tmp + v6stat.zero_pos,
752                                 v6stat.total - v6stat.zero_pos);
753                 }
754         else
755                 memcpy(v6, v6stat.tmp, 16);

757         return 1;
758         }

760 static int ipv6_cb(const char *elem, int len, void *usr)
761         {
762         IPV6_STAT *s = usr;
763         /* Error if 16 bytes written */
764         if (s->total == 16)
765                 return 0;
766         if (len == 0)
767                 {
768                 /* Zero length element, corresponds to '::' */
769                 if (s->zero_pos == -1)
770                         s->zero_pos = s->total;
771                 /* If we've already got a :: its an error */
772                 else if (s->zero_pos != s->total)
773                         return 0;
774                 s->zero_cnt++;
775                 }
776         else
777                 {
778                 /* If more than 4 characters could be final a.b.c.d form */
779                 if (len > 4)
780                         {
781                         /* Need at least 4 bytes left */
782                         if (s->total > 12)
783                                 return 0;
784                         /* Must be end of string */
785                         if (elem[len])
786                                 return 0;
787                         if (!ipv4_from_asc(s->tmp + s->total, elem))
```

```
788                                 return 0;
789                         s->total += 4;
790                         }
791                 else
792                         {
793                         if (!ipv6_hex(s->tmp + s->total, elem, len))
794                                 return 0;
795                         s->total += 2;
796                         }
797                 }
798         return 1;
799         }

801 /* Convert a string of up to 4 hex digits into the corresponding
802  * IPv6 form.
803  */

805 static int ipv6_hex(unsigned char *out, const char *in, int inlen)
806         {
807         unsigned char c;
808         unsigned int num = 0;
809         if (inlen > 4)
810                 return 0;
811         while(inlen--)
812                 {
813                 c = *in++;
814                 num <<= 4;
815                 if ((c >= '0') && (c <= '9'))
816                         num |= c - '0';
817                 else if ((c >= 'A') && (c <= 'F'))
818                         num |= c - 'A' + 10;
819                 else if ((c >= 'a') && (c <= 'f'))
820                         num |=  c - 'a' + 10;
821                 else
822                         return 0;
823                 }
824         out[0] = num >> 8;
825         out[1] = num & 0xff;
826         return 1;
827         }


830 int X509V3_NAME_from_section(X509_NAME *nm, STACK_OF(CONF_VALUE)*dn_sk,
831                                                 unsigned long chtype)
832         {
833         CONF_VALUE *v;
834         int i, mval;
835         char *p, *type;
836         if (!nm)
837                 return 0;

839         for (i = 0; i < sk_CONF_VALUE_num(dn_sk); i++)
840                 {
841                 v=sk_CONF_VALUE_value(dn_sk,i);
842                 type=v->name;
843                 /* Skip past any leading X. X: X, etc to allow for
844                  * multiple instances
845                  */
846                 for(p = type; *p ; p++)
847 #ifndef CHARSET_EBCDIC
848                         if ((*p == ':') || (*p == ',') || (*p == '.'))
849 #else
850                         if ((*p == os_toascii[':']) || (*p == os_toascii[',']) |
851 #endif
852                                 {
853                                 p++;
```



```
854                                 if(*p) type = p;
855                                 break;
856                                 }
857 #ifndef CHARSET_EBCDIC
858                 if (*type == '+')
859 #else
860                 if (*type == os_toascii['+'])
861 #endif
862                         {
863                         mval = -1;
864                         type++;
865                         }
866                 else
867                         mval = 0;
868                 if (!X509_NAME_add_entry_by_txt(nm,type, chtype,
869                                 (unsigned char *) v->value,-1,-1,mval))
870                                         return 0;

872                 }
873         return 1;
874         }
875 #endif /* ! codereview */
```

```
**********************************************************
   12171 Wed Aug 13 19:53:33 2014
new/usr/src/lib/openssl/libsunw_crypto/x509v3/v3err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* crypto/x509v3/v3err.c */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/x509v3.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_X509V3,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_X509V3,0,reason)

static ERR_STRING_DATA X509V3_str_functs[]=
        {
{ERR_FUNC(X509V3_F_A2I_GENERAL_NAME),   "A2I_GENERAL_NAME"},
{ERR_FUNC(X509V3_F_ASIDENTIFIERCHOICE_CANONIZE),        "ASIDENTIFIERCHOICE_CANO
{ERR_FUNC(X509V3_F_ASIDENTIFIERCHOICE_IS_CANONICAL),    "ASIDENTIFIERCHOICE_IS_C
{ERR_FUNC(X509V3_F_COPY_EMAIL), "COPY_EMAIL"},
{ERR_FUNC(X509V3_F_COPY_ISSUER),        "COPY_ISSUER"},
{ERR_FUNC(X509V3_F_DO_DIRNAME), "DO_DIRNAME"},
{ERR_FUNC(X509V3_F_DO_EXT_CONF),        "DO_EXT_CONF"},
{ERR_FUNC(X509V3_F_DO_EXT_I2D), "DO_EXT_I2D"},
{ERR_FUNC(X509V3_F_DO_EXT_NCONF),       "DO_EXT_NCONF"},
{ERR_FUNC(X509V3_F_DO_I2V_NAME_CONSTRAINTS),    "DO_I2V_NAME_CONSTRAINTS"},
{ERR_FUNC(X509V3_F_GNAMES_FROM_SECTNAME),       "GNAMES_FROM_SECTNAME"},
{ERR_FUNC(X509V3_F_HEX_TO_STRING),      "hex_to_string"},
{ERR_FUNC(X509V3_F_I2S_ASN1_ENUMERATED),        "i2s_ASN1_ENUMERATED"},
{ERR_FUNC(X509V3_F_I2S_ASN1_IA5STRING), "I2S_ASN1_IA5STRING"},
{ERR_FUNC(X509V3_F_I2S_ASN1_INTEGER),   "i2s_ASN1_INTEGER"},
{ERR_FUNC(X509V3_F_I2V_AUTHORITY_INFO_ACCESS),  "I2V_AUTHORITY_INFO_ACCESS"},
{ERR_FUNC(X509V3_F_NOTICE_SECTION),     "NOTICE_SECTION"},
{ERR_FUNC(X509V3_F_NREF_NOS),   "NREF_NOS"},
{ERR_FUNC(X509V3_F_POLICY_SECTION),     "POLICY_SECTION"},
{ERR_FUNC(X509V3_F_PROCESS_PCI_VALUE),  "PROCESS_PCI_VALUE"},
{ERR_FUNC(X509V3_F_R2I_CERTPOL),        "R2I_CERTPOL"},
{ERR_FUNC(X509V3_F_R2I_PCI),    "R2I_PCI"},
{ERR_FUNC(X509V3_F_S2I_ASN1_IA5STRING), "S2I_ASN1_IA5STRING"},
{ERR_FUNC(X509V3_F_S2I_ASN1_INTEGER),   "s2i_ASN1_INTEGER"},
{ERR_FUNC(X509V3_F_S2I_ASN1_OCTET_STRING),      "s2i_ASN1_OCTET_STRING"},
{ERR_FUNC(X509V3_F_S2I_ASN1_SKEY_ID),   "S2I_ASN1_SKEY_ID"},
{ERR_FUNC(X509V3_F_S2I_SKEY_ID),        "S2I_SKEY_ID"},
{ERR_FUNC(X509V3_F_SET_DIST_POINT_NAME),        "SET_DIST_POINT_NAME"},
{ERR_FUNC(X509V3_F_STRING_TO_HEX),      "string_to_hex"},
{ERR_FUNC(X509V3_F_SXNET_ADD_ID_ASC),   "SXNET_add_id_asc"},
{ERR_FUNC(X509V3_F_SXNET_ADD_ID_INTEGER),       "SXNET_add_id_INTEGER"},
{ERR_FUNC(X509V3_F_SXNET_ADD_ID_ULONG), "SXNET_add_id_ulong"},
{ERR_FUNC(X509V3_F_SXNET_GET_ID_ASC),   "SXNET_get_id_asc"},
{ERR_FUNC(X509V3_F_SXNET_GET_ID_ULONG), "SXNET_get_id_ulong"},
{ERR_FUNC(X509V3_F_V2I_ASIDENTIFIERS),  "V2I_ASIDENTIFIERS"},
{ERR_FUNC(X509V3_F_V2I_ASN1_BIT_STRING),        "v2i_ASN1_BIT_STRING"},
{ERR_FUNC(X509V3_F_V2I_AUTHORITY_INFO_ACCESS),  "V2I_AUTHORITY_INFO_ACCESS"},
{ERR_FUNC(X509V3_F_V2I_AUTHORITY_KEYID),        "V2I_AUTHORITY_KEYID"},
{ERR_FUNC(X509V3_F_V2I_BASIC_CONSTRAINTS),      "V2I_BASIC_CONSTRAINTS"},
{ERR_FUNC(X509V3_F_V2I_CRLD),   "V2I_CRLD"},
{ERR_FUNC(X509V3_F_V2I_EXTENDED_KEY_USAGE),     "V2I_EXTENDED_KEY_USAGE"},
{ERR_FUNC(X509V3_F_V2I_GENERAL_NAMES),  "v2i_GENERAL_NAMES"},
{ERR_FUNC(X509V3_F_V2I_GENERAL_NAME_EX),        "v2i_GENERAL_NAME_ex"},
{ERR_FUNC(X509V3_F_V2I_IDP),    "V2I_IDP"},
{ERR_FUNC(X509V3_F_V2I_IPADDRBLOCKS),   "V2I_IPADDRBLOCKS"},
{ERR_FUNC(X509V3_F_V2I_ISSUER_ALT),     "V2I_ISSUER_ALT"},
{ERR_FUNC(X509V3_F_V2I_NAME_CONSTRAINTS),       "V2I_NAME_CONSTRAINTS"},
{ERR_FUNC(X509V3_F_V2I_POLICY_CONSTRAINTS),     "V2I_POLICY_CONSTRAINTS"},
{ERR_FUNC(X509V3_F_V2I_POLICY_MAPPINGS),        "V2I_POLICY_MAPPINGS"},
{ERR_FUNC(X509V3_F_V2I_SUBJECT_ALT),    "V2I_SUBJECT_ALT"},
{ERR_FUNC(X509V3_F_V3_ADDR_VALIDATE_PATH_INTERNAL),     "V3_ADDR_VALIDATE_PATH_I
{ERR_FUNC(X509V3_F_V3_GENERIC_EXTENSION),       "V3_GENERIC_EXTENSION"},
{ERR_FUNC(X509V3_F_X509V3_ADD1_I2D),    "X509V3_add1_i2d"},
{ERR_FUNC(X509V3_F_X509V3_ADD_VALUE),   "X509V3_add_value"},
{ERR_FUNC(X509V3_F_X509V3_EXT_ADD),     "X509V3_EXT_add"},
```

```
{ERR_FUNC(X509V3_F_X509V3_EXT_ADD_ALIAS),       "X509V3_EXT_add_alias"},
{ERR_FUNC(X509V3_F_X509V3_EXT_CONF),    "X509V3_EXT_conf"},
{ERR_FUNC(X509V3_F_X509V3_EXT_I2D),     "X509V3_EXT_i2d"},
{ERR_FUNC(X509V3_F_X509V3_EXT_NCONF),   "X509V3_EXT_nconf"},
{ERR_FUNC(X509V3_F_X509V3_GET_SECTION), "X509V3_get_section"},
{ERR_FUNC(X509V3_F_X509V3_GET_STRING),  "X509V3_get_string"},
{ERR_FUNC(X509V3_F_X509V3_GET_VALUE_BOOL),      "X509V3_get_value_bool"},
{ERR_FUNC(X509V3_F_X509V3_PARSE_LIST),  "X509V3_parse_list"},
{ERR_FUNC(X509V3_F_X509_PURPOSE_ADD),   "X509_PURPOSE_add"},
{ERR_FUNC(X509V3_F_X509_PURPOSE_SET),   "X509_PURPOSE_set"},
{0,NULL}
        };

static ERR_STRING_DATA X509V3_str_reasons[]=
        {
{ERR_REASON(X509V3_R_BAD_IP_ADDRESS)     ,"bad ip address"},
{ERR_REASON(X509V3_R_BAD_OBJECT)         ,"bad object"},
{ERR_REASON(X509V3_R_BN_DEC2BN_ERROR)    ,"bn dec2bn error"},
{ERR_REASON(X509V3_R_BN_TO_ASN1_INTEGER_ERROR),"bn to asn1 integer error"},
{ERR_REASON(X509V3_R_DIRNAME_ERROR)      ,"dirname error"},
{ERR_REASON(X509V3_R_DISTPOINT_ALREADY_SET),"distpoint already set"},
{ERR_REASON(X509V3_R_DUPLICATE_ZONE_ID)  ,"duplicate zone id"},
{ERR_REASON(X509V3_R_ERROR_CONVERTING_ZONE),"error converting zone"},
{ERR_REASON(X509V3_R_ERROR_CREATING_EXTENSION),"error creating extension"},
{ERR_REASON(X509V3_R_ERROR_IN_EXTENSION) ,"error in extension"},
{ERR_REASON(X509V3_R_EXPECTED_A_SECTION_NAME),"expected a section name"},
{ERR_REASON(X509V3_R_EXTENSION_EXISTS)   ,"extension exists"},
{ERR_REASON(X509V3_R_EXTENSION_NAME_ERROR),"extension name error"},
{ERR_REASON(X509V3_R_EXTENSION_NOT_FOUND),"extension not found"},
{ERR_REASON(X509V3_R_EXTENSION_SETTING_NOT_SUPPORTED),"extension setting not sup
{ERR_REASON(X509V3_R_EXTENSION_VALUE_ERROR),"extension value error"},
{ERR_REASON(X509V3_R_ILLEGAL_EMPTY_EXTENSION),"illegal empty extension"},
{ERR_REASON(X509V3_R_ILLEGAL_HEX_DIGIT)  ,"illegal hex digit"},
{ERR_REASON(X509V3_R_INCORRECT_POLICY_SYNTAX_TAG),"incorrect policy syntax tag"}
{ERR_REASON(X509V3_R_INVALID_MULTIPLE_RDNS),"invalid multiple rdns"},
{ERR_REASON(X509V3_R_INVALID_ASNUMBER)   ,"invalid asnumber"},
{ERR_REASON(X509V3_R_INVALID_ASRANGE)    ,"invalid asrange"},
{ERR_REASON(X509V3_R_INVALID_BOOLEAN_STRING),"invalid boolean string"},
{ERR_REASON(X509V3_R_INVALID_EXTENSION_STRING),"invalid extension string"},
{ERR_REASON(X509V3_R_INVALID_INHERITANCE),"invalid inheritance"},
{ERR_REASON(X509V3_R_INVALID_IPADDRESS)  ,"invalid ipaddress"},
{ERR_REASON(X509V3_R_INVALID_NAME)       ,"invalid name"},
{ERR_REASON(X509V3_R_INVALID_NULL_ARGUMENT),"invalid null argument"},
{ERR_REASON(X509V3_R_INVALID_NULL_NAME)  ,"invalid null name"},
{ERR_REASON(X509V3_R_INVALID_NULL_VALUE) ,"invalid null value"},
{ERR_REASON(X509V3_R_INVALID_NUMBER)     ,"invalid number"},
{ERR_REASON(X509V3_R_INVALID_NUMBERS)    ,"invalid numbers"},
{ERR_REASON(X509V3_R_INVALID_OBJECT_IDENTIFIER),"invalid object identifier"},
{ERR_REASON(X509V3_R_INVALID_OPTION)     ,"invalid option"},
{ERR_REASON(X509V3_R_INVALID_POLICY_IDENTIFIER),"invalid policy identifier"},
{ERR_REASON(X509V3_R_INVALID_PROXY_POLICY_SETTING),"invalid proxy policy setting
{ERR_REASON(X509V3_R_INVALID_PURPOSE)    ,"invalid purpose"},
{ERR_REASON(X509V3_R_INVALID_SAFI)       ,"invalid safi"},
{ERR_REASON(X509V3_R_INVALID_SECTION)    ,"invalid section"},
{ERR_REASON(X509V3_R_INVALID_SYNTAX)     ,"invalid syntax"},
{ERR_REASON(X509V3_R_ISSUER_DECODE_ERROR),"issuer decode error"},
{ERR_REASON(X509V3_R_MISSING_VALUE)      ,"missing value"},
{ERR_REASON(X509V3_R_NEED_ORGANIZATION_AND_NUMBERS),"need organization and numbe
{ERR_REASON(X509V3_R_NO_CONFIG_DATABASE) ,"no config database"},
{ERR_REASON(X509V3_R_NO_ISSUER_CERTIFICATE),"no issuer certificate"},
{ERR_REASON(X509V3_R_NO_ISSUER_DETAILS)  ,"no issuer details"},
{ERR_REASON(X509V3_R_NO_POLICY_IDENTIFIER),"no policy identifier"},
{ERR_REASON(X509V3_R_NO_PROXY_CERT_POLICY_LANGUAGE_DEFINED),"no proxy cert polic
{ERR_REASON(X509V3_R_NO_PUBLIC_KEY)      ,"no public key"},
{ERR_REASON(X509V3_R_NO_SUBJECT_DETAILS) ,"no subject details"},
{ERR_REASON(X509V3_R_ODD_NUMBER_OF_DIGITS),"odd number of digits"},
```



```
{ERR_REASON(X509V3_R_OPERATION_NOT_DEFINED),"operation not defined"},
{ERR_REASON(X509V3_R_OTHERNAME_ERROR)    ,"othername error"},
{ERR_REASON(X509V3_R_POLICY_LANGUAGE_ALREADY_DEFINED),"policy language already d
{ERR_REASON(X509V3_R_POLICY_PATH_LENGTH) ,"policy path length"},
{ERR_REASON(X509V3_R_POLICY_PATH_LENGTH_ALREADY_DEFINED),"policy path length alr
{ERR_REASON(X509V3_R_POLICY_SYNTAX_NOT_CURRENTLY_SUPPORTED),"policy syntax not c
{ERR_REASON(X509V3_R_POLICY_WHEN_PROXY_LANGUAGE_REQUIRES_NO_POLICY),"policy when
{ERR_REASON(X509V3_R_SECTION_NOT_FOUND)  ,"section not found"},
{ERR_REASON(X509V3_R_UNABLE_TO_GET_ISSUER_DETAILS),"unable to get issuer details
{ERR_REASON(X509V3_R_UNABLE_TO_GET_ISSUER_KEYID),"unable to get issuer keyid"},
{ERR_REASON(X509V3_R_UNKNOWN_BIT_STRING_ARGUMENT),"unknown bit string argument"}
{ERR_REASON(X509V3_R_UNKNOWN_EXTENSION)  ,"unknown extension"},
{ERR_REASON(X509V3_R_UNKNOWN_EXTENSION_NAME),"unknown extension name"},
{ERR_REASON(X509V3_R_UNKNOWN_OPTION)     ,"unknown option"},
{ERR_REASON(X509V3_R_UNSUPPORTED_OPTION) ,"unsupported option"},
{ERR_REASON(X509V3_R_UNSUPPORTED_TYPE)   ,"unsupported type"},
{ERR_REASON(X509V3_R_USER_TOO_LONG)      ,"user too long"},
{0,NULL}
        };

#endif

void ERR_load_X509V3_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(X509V3_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,X509V3_str_functs);
                ERR_load_strings(0,X509V3_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

****
1362 Wed Aug 13 19:53:33 2014
new/usr/src/lib/openssl/libsunw_ssl/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
****

```
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2014 Alexander Pyhalov
# Use is subject to license terms.
#
#

include $(SRC)/lib/Makefile.lib

SUBDIRS = .WAIT $(MACH) $(BUILD64) $(MACH64)

# conditional assignments
all :=          TARGET= all
install :=      TARGET= install
clean :=        TARGET= clean
clobber :=      TARGET= clobber
lint :=         TARGET= lint
_msg :=         TARGET= _msg

HDRS=

.KEEP_STATE:

all install clean clobber lint: $(SUBDIRS)

_msg check:

$(MACH) $(MACH64): FRC
        @cd $@; pwd; $(MAKE) $(TARGET)

FRC:

#endif /* ! codereview */
```

```
**********************************************************
    2931 Wed Aug 13 19:53:33 2014
new/usr/src/lib/openssl/libsunw_ssl/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Copyright 2014 Alexander Pyhalov
# Use is subject to license terms.
#

LIBRARY=        libsunw_ssl.a
VERS=           .1

OBJECTS=        bio_ssl.o  \
                d1_both.o  \
                d1_clnt.o  \
                d1_enc.o   \
                d1_lib.o   \
                d1_meth.o  \
                d1_pkt.o   \
                d1_srtp.o  \
                d1_srvr.o  \
                kssl.o     \
                s23_clnt.o \
                s23_lib.o  \
                s23_meth.o \
                s23_pkt.o  \
                s23_srvr.o \
                s2_clnt.o  \
                s2_enc.o   \
                s2_lib.o   \
                s2_meth.o  \
                s2_pkt.o   \
                s2_srvr.o  \
                s3_both.o  \
                s3_cbc.o   \
                s3_clnt.o  \
                s3_enc.o   \
                s3_lib.o   \
                s3_meth.o  \
                s3_pkt.o   \
                s3_srvr.o  \
                ssl_algs.o \
                ssl_asn1.o \
                ssl_cert.o \
```



```
                ssl_ciph.o \
                ssl_err.o  \
                ssl_err2.o \
                ssl_lib.o  \
                ssl_rsa.o  \
                ssl_sess.o \
                ssl_stat.o \
                ssl_txt.o  \
                t1_clnt.o  \
                t1_enc.o   \
                t1_lib.o   \
                t1_meth.o  \
                t1_reneg.o \
                t1_srvr.o  \
                tls_srp.o

# include library definitions
include $(SRC)/lib/Makefile.lib

CLOBBERFILES += $(LIBLINKS)

LIBS =          $(DYNLIB)

LDLIBS += -lc -lsunw_crypto

LINTFLAGS =     -uxn
LINTFLAGS64 =   $(LINTFLAGS) -m64
LINTOUT=        lint.out
LINTSRC =       $(LINTLIB:%.ln=%)
ROOTLINTDIR =   $(ROOTLIBDIR)
ROOTLINT =      $(LINTSRC:%=$(ROOTLINTDIR)/%)

CPPFLAGS +=     -I.. \
                -I$(SRC)/lib/openssl/include

CPPFLAGS +=     -D_REENTRANT
CPPFLAGS +=     -DOPENSSL_THREADS
CPPFLAGS +=     -DDSO_DLFCN
CPPFLAGS +=     -DHAVE_DLFCN_H
CPPFLAGS +=     -DSOLARIS_OPENSSL
CPPFLAGS +=     -DNO_WINDOWS_BRAINDEATH
CPPFLAGS +=     -DOPENSSL_BN_ASM_GF2m
CPPFLAGS +=     -DSHA1_ASM
CPPFLAGS +=     -DSHA256_ASM
CPPFLAGS +=     -DSHA512_ASM
CPPFLAGS +=     -DMD5_ASM
CPPFLAGS +=     -DAES_ASM
CPPFLAGS +=     -DVPAES_ASM
CPPFLAGS +=     -DGHASH_ASM
CPPFLAGS +=     -DVPAES_ASM
CPPFLAGS +=     -DOPENSSL_BN_ASM_MONT

CFLAGS +=       $(CCVERBOSE)

CERRWARN +=     -erroff=E_END_OF_LOOP_CODE_NOT_REACHED
CERRWARN +=     -erroff=E_CONST_PROMOTED_UNSIGNED_LONG
CERRWARN +=     -erroff=E_INIT_DOES_NOT_FIT

$(LINTLIB) :=   LINTFLAGS = -nvx -I..
$(LINTLIB) :=   LINTFLAGS64 = -nvx -m64 -I..

.KEEP_STATE:

all : $(LIBS)

lint :  lintcheck
```

```
129 # include library targets
130 include $(SRC)/lib/Makefile.targ

132 pics/%.o:       ../%.c
133         $(COMPILE.c) -o $@ $<

135 $(ROOTLINTDIR)/%: ../%
136         $(INS.file)
137 #endif /* ! codereview */
```

```
**********************************************************
    1300 Wed Aug 13 19:53:33 2014
new/usr/src/lib/openssl/libsunw_ssl/amd64/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23 # Copyright 2014 Alexander Pyhalov
  24 # Use is subject to license terms.

  26 include ../Makefile.com
  27 include $(SRC)/lib/Makefile.lib.64

  29 .KEEP_STATE:

  31 CPPFLAGS += -DL_ENDIAN
  32 CPPFLAGS += -DOPENSSL_IA32_SSE2
  33 CPPFLAGS += -DOPENSSL_BN_ASM_MONT5
  34 CPPFLAGS += -DBSAES_ASM

  36 all:  $(ROOTLIBDIR64) $(LIBS) $(LIBLINKS)

  38 $(LIBLINKS):    FRC
  39         $(RM) $@; $(SYMLINK) $(DYNLIB) $@

  41 $(ROOTLIBDIR64):
  42         $(INS.dir)

  44 install: all $(ROOTLIBS64) $(ROOTLINKS64)

  46 FRC:
  47 #endif /* ! codereview */
```

```
**********************************************************
   14275 Wed Aug 13 19:53:34 2014
new/usr/src/lib/openssl/libsunw_ssl/bio_ssl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/bio_ssl.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```



```
#include <errno.h>
#include <openssl/opensslconf.h>
#include <openssl/crypto.h>
#include <openssl/bio.h>
#include <openssl/err.h>
#include <openssl/ssl.h>

static int ssl_write(BIO *h, const char *buf, int num);
static int ssl_read(BIO *h, char *buf, int size);
static int ssl_puts(BIO *h, const char *str);
static long ssl_ctrl(BIO *h, int cmd, long arg1, void *arg2);
static int ssl_new(BIO *h);
static int ssl_free(BIO *data);
static long ssl_callback_ctrl(BIO *h, int cmd, bio_info_cb *fp);
typedef struct bio_ssl_st
        {
        SSL *ssl; /* The ssl handle :-) */
        /* re-negotiate every time the total number of bytes is this size */
        int num_renegotiates;
        unsigned long renegotiate_count;
        unsigned long byte_count;
        unsigned long renegotiate_timeout;
        unsigned long last_time;
        } BIO_SSL;

static BIO_METHOD methods_sslp=
        {
        BIO_TYPE_SSL,"ssl",
        ssl_write,
        ssl_read,
        ssl_puts,
        NULL, /* ssl_gets, */
        ssl_ctrl,
        ssl_new,
        ssl_free,
        ssl_callback_ctrl,
        };

BIO_METHOD *BIO_f_ssl(void)
        {
        return(&methods_sslp);
        }

static int ssl_new(BIO *bi)
        {
        BIO_SSL *bs;

        bs=(BIO_SSL *)OPENSSL_malloc(sizeof(BIO_SSL));
        if (bs == NULL)
                {
                BIOerr(BIO_F_SSL_NEW,ERR_R_MALLOC_FAILURE);
                return(0);
                }
        memset(bs,0,sizeof(BIO_SSL));
        bi->init=0;
        bi->ptr=(char *)bs;
        bi->flags=0;
        return(1);
        }

static int ssl_free(BIO *a)
        {
        BIO_SSL *bs;

        if (a == NULL) return(0);
        bs=(BIO_SSL *)a->ptr;
```

```
128         if (bs->ssl != NULL) SSL_shutdown(bs->ssl);
129         if (a->shutdown)
130                 {
131                 if (a->init && (bs->ssl != NULL))
132                         SSL_free(bs->ssl);
133                 a->init=0;
134                 a->flags=0;
135                 }
136         if (a->ptr != NULL)
137                 OPENSSL_free(a->ptr);
138         return(1);
139         }

141 static int ssl_read(BIO *b, char *out, int outl)
142         {
143         int ret=1;
144         BIO_SSL *sb;
145         SSL *ssl;
146         int retry_reason=0;
147         int r=0;

149         if (out == NULL) return(0);
150         sb=(BIO_SSL *)b->ptr;
151         ssl=sb->ssl;

153         BIO_clear_retry_flags(b);

155 #if 0
156         if (!SSL_is_init_finished(ssl))
157                 {
158 /*              ret=SSL_do_handshake(ssl); */
159                 if (ret > 0)
160                         {

162                         outflags=(BIO_FLAGS_READ|BIO_FLAGS_SHOULD_RETRY);
163                         ret= -1;
164                         goto end;
165                         }
166                 }
167 #endif
168 /*      if (ret > 0) */
169         ret=SSL_read(ssl,out,outl);

171         switch (SSL_get_error(ssl,ret))
172                 {
173         case SSL_ERROR_NONE:
174                 if (ret <= 0) break;
175                 if (sb->renegotiate_count > 0)
176                         {
177                         sb->byte_count+=ret;
178                         if (sb->byte_count > sb->renegotiate_count)
179                                 {
180                                 sb->byte_count=0;
181                                 sb->num_renegotiates++;
182                                 SSL_renegotiate(ssl);
183                                 r=1;
184                                 }
185                         }
186                 if ((sb->renegotiate_timeout > 0) && (!r))
187                         {
188                         unsigned long tm;

190                         tm=(unsigned long)time(NULL);
191                         if (tm > sb->last_time+sb->renegotiate_timeout)
192                                 {
193                                 sb->last_time=tm;
```



```
194                                 sb->num_renegotiates++;
195                                 SSL_renegotiate(ssl);
196                                 }
197                         }

199                 break;
200         case SSL_ERROR_WANT_READ:
201                 BIO_set_retry_read(b);
202                 break;
203         case SSL_ERROR_WANT_WRITE:
204                 BIO_set_retry_write(b);
205                 break;
206         case SSL_ERROR_WANT_X509_LOOKUP:
207                 BIO_set_retry_special(b);
208                 retry_reason=BIO_RR_SSL_X509_LOOKUP;
209                 break;
210         case SSL_ERROR_WANT_ACCEPT:
211                 BIO_set_retry_special(b);
212                 retry_reason=BIO_RR_ACCEPT;
213                 break;
214         case SSL_ERROR_WANT_CONNECT:
215                 BIO_set_retry_special(b);
216                 retry_reason=BIO_RR_CONNECT;
217                 break;
218         case SSL_ERROR_SYSCALL:
219         case SSL_ERROR_SSL:
220         case SSL_ERROR_ZERO_RETURN:
221         default:
222                 break;
223                 }

225         b->retry_reason=retry_reason;
226         return(ret);
227         }

229 static int ssl_write(BIO *b, const char *out, int outl)
230         {
231         int ret,r=0;
232         int retry_reason=0;
233         SSL *ssl;
234         BIO_SSL *bs;

236         if (out == NULL) return(0);
237         bs=(BIO_SSL *)b->ptr;
238         ssl=bs->ssl;

240         BIO_clear_retry_flags(b);

242 /*      ret=SSL_do_handshake(ssl);
243         if (ret > 0) */
244         ret=SSL_write(ssl,out,outl);

246         switch (SSL_get_error(ssl,ret))
247                 {
248         case SSL_ERROR_NONE:
249                 if (ret <= 0) break;
250                 if (bs->renegotiate_count > 0)
251                         {
252                         bs->byte_count+=ret;
253                         if (bs->byte_count > bs->renegotiate_count)
254                                 {
255                                 bs->byte_count=0;
256                                 bs->num_renegotiates++;
257                                 SSL_renegotiate(ssl);
258                                 r=1;
259                                 }
```

```
260                     }
261             if ((bs->renegotiate_timeout > 0) && (!r))
262                     {
263                     unsigned long tm;

265                     tm=(unsigned long)time(NULL);
266                     if (tm > bs->last_time+bs->renegotiate_timeout)
267                             {
268                             bs->last_time=tm;
269                             bs->num_renegotiates++;
270                             SSL_renegotiate(ssl);
271                             }
272                     }
273             break;
274     case SSL_ERROR_WANT_WRITE:
275             BIO_set_retry_write(b);
276             break;
277     case SSL_ERROR_WANT_READ:
278             BIO_set_retry_read(b);
279             break;
280     case SSL_ERROR_WANT_X509_LOOKUP:
281             BIO_set_retry_special(b);
282             retry_reason=BIO_RR_SSL_X509_LOOKUP;
283             break;
284     case SSL_ERROR_WANT_CONNECT:
285             BIO_set_retry_special(b);
286             retry_reason=BIO_RR_CONNECT;
287     case SSL_ERROR_SYSCALL:
288     case SSL_ERROR_SSL:
289     default:
290             break;
291             }

293     b->retry_reason=retry_reason;
294     return(ret);
295     }

297 static long ssl_ctrl(BIO *b, int cmd, long num, void *ptr)
298     {
299     SSL **sslp,*ssl;
300     BIO_SSL *bs;
301     BIO *dbio,*bio;
302     long ret=1;

304     bs=(BIO_SSL *)b->ptr;
305     ssl=bs->ssl;
306     if ((ssl == NULL)  && (cmd != BIO_C_SET_SSL))
307             return(0);
308     switch (cmd)
309             {
310     case BIO_CTRL_RESET:
311             SSL_shutdown(ssl);

313             if (ssl->handshake_func == ssl->method->ssl_connect)
314                     SSL_set_connect_state(ssl);
315             else if (ssl->handshake_func == ssl->method->ssl_accept)
316                     SSL_set_accept_state(ssl);

318             SSL_clear(ssl);

320             if (b->next_bio != NULL)
321                     ret=BIO_ctrl(b->next_bio,cmd,num,ptr);
322             else if (ssl->rbio != NULL)
323                     ret=BIO_ctrl(ssl->rbio,cmd,num,ptr);
324             else
325                     ret=1;
```



```
326             break;
327     case BIO_CTRL_INFO:
328             ret=0;
329             break;
330     case BIO_C_SSL_MODE:
331             if (num) /* client mode */
332                     SSL_set_connect_state(ssl);
333             else
334                     SSL_set_accept_state(ssl);
335             break;
336     case BIO_C_SET_SSL_RENEGOTIATE_TIMEOUT:
337             ret=bs->renegotiate_timeout;
338             if (num < 60) num=5;
339             bs->renegotiate_timeout=(unsigned long)num;
340             bs->last_time=(unsigned long)time(NULL);
341             break;
342     case BIO_C_SET_SSL_RENEGOTIATE_BYTES:
343             ret=bs->renegotiate_count;
344             if ((long)num >=512)
345                     bs->renegotiate_count=(unsigned long)num;
346             break;
347     case BIO_C_GET_SSL_NUM_RENEGOTIATES:
348             ret=bs->num_renegotiates;
349             break;
350     case BIO_C_SET_SSL:
351             if (ssl != NULL)
352                     {
353                     ssl_free(b);
354                     if (!ssl_new(b))
355                             return 0;
356                     }
357             b->shutdown=(int)num;
358             ssl=(SSL *)ptr;
359             ((BIO_SSL *)b->ptr)->ssl=ssl;
360             bio=SSL_get_rbio(ssl);
361             if (bio != NULL)
362                     {
363                     if (b->next_bio != NULL)
364                             BIO_push(bio,b->next_bio);
365                     b->next_bio=bio;
366                     CRYPTO_add(&bio->references,1,CRYPTO_LOCK_BIO);
367                     }
368             b->init=1;
369             break;
370     case BIO_C_GET_SSL:
371             if (ptr != NULL)
372                     {
373                     sslp=(SSL **)ptr;
374                     *sslp=ssl;
375                     }
376             else
377                     ret=0;
378             break;
379     case BIO_CTRL_GET_CLOSE:
380             ret=b->shutdown;
381             break;
382     case BIO_CTRL_SET_CLOSE:
383             b->shutdown=(int)num;
384             break;
385     case BIO_CTRL_WPENDING:
386             ret=BIO_ctrl(ssl->wbio,cmd,num,ptr);
387             break;
388     case BIO_CTRL_PENDING:
389             ret=SSL_pending(ssl);
390             if (ret == 0)
391                     ret=BIO_pending(ssl->rbio);
```

```
392                 break;
393         case BIO_CTRL_FLUSH:
394                 BIO_clear_retry_flags(b);
395                 ret=BIO_ctrl(ssl->wbio,cmd,num,ptr);
396                 BIO_copy_next_retry(b);
397                 break;
398         case BIO_CTRL_PUSH:
399                 if ((b->next_bio != NULL) && (b->next_bio != ssl->rbio))
400                         {
401                         SSL_set_bio(ssl,b->next_bio,b->next_bio);
402                         CRYPTO_add(&b->next_bio->references,1,CRYPTO_LOCK_BIO);
403                         }
404                 break;
405         case BIO_CTRL_POP:
406                 /* Only detach if we are the BIO explicitly being popped */
407                 if (b == ptr)
408                         {
409                         /* Shouldn't happen in practice because the
410                          * rbio and wbio are the same when pushed.
411                          */
412                         if (ssl->rbio != ssl->wbio)
413                                 BIO_free_all(ssl->wbio);
414                         if (b->next_bio != NULL)
415                                 CRYPTO_add(&b->next_bio->references,-1,CRYPTO_LO
416                         ssl->wbio=NULL;
417                         ssl->rbio=NULL;
418                         }
419                 break;
420         case BIO_C_DO_STATE_MACHINE:
421                 BIO_clear_retry_flags(b);

423                 b->retry_reason=0;
424                 ret=(int)SSL_do_handshake(ssl);

426                 switch (SSL_get_error(ssl,(int)ret))
427                         {
428                 case SSL_ERROR_WANT_READ:
429                         BIO_set_flags(b,
430                                 BIO_FLAGS_READ|BIO_FLAGS_SHOULD_RETRY);
431                         break;
432                 case SSL_ERROR_WANT_WRITE:
433                         BIO_set_flags(b,
434                                 BIO_FLAGS_WRITE|BIO_FLAGS_SHOULD_RETRY);
435                         break;
436                 case SSL_ERROR_WANT_CONNECT:
437                         BIO_set_flags(b,
438                                 BIO_FLAGS_IO_SPECIAL|BIO_FLAGS_SHOULD_RETRY);
439                         b->retry_reason=b->next_bio->retry_reason;
440                         break;
441                 default:
442                         break;
443                         }
444                 break;
445         case BIO_CTRL_DUP:
446                 dbio=(BIO *)ptr;
447                 if (((BIO_SSL *)dbio->ptr)->ssl != NULL)
448                         SSL_free(((BIO_SSL *)dbio->ptr)->ssl);
449                 ((BIO_SSL *)dbio->ptr)->ssl=SSL_dup(ssl);
450                 ((BIO_SSL *)dbio->ptr)->renegotiate_count=
451                         ((BIO_SSL *)b->ptr)->renegotiate_count;
452                 ((BIO_SSL *)dbio->ptr)->byte_count=
453                         ((BIO_SSL *)b->ptr)->byte_count;
454                 ((BIO_SSL *)dbio->ptr)->renegotiate_timeout=
455                         ((BIO_SSL *)b->ptr)->renegotiate_timeout;
456                 ((BIO_SSL *)dbio->ptr)->last_time=
457                         ((BIO_SSL *)b->ptr)->last_time;
```



```
458                 ret=(((BIO_SSL *)dbio->ptr)->ssl != NULL);
459                 break;
460         case BIO_C_GET_FD:
461                 ret=BIO_ctrl(ssl->rbio,cmd,num,ptr);
462                 break;
463         case BIO_CTRL_SET_CALLBACK:
464                 {
465 #if 0 /* FIXME: Should this be used?  -- Richard Levitte */
466                 SSLerr(SSL_F_SSL_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
467                 ret = -1;
468 #else
469                 ret=0;
470 #endif
471                 }
472                 break;
473         case BIO_CTRL_GET_CALLBACK:
474                 {
475                 void (**fptr)(const SSL *xssl,int type,int val);

477                 fptr=(void (**)(const SSL *xssl,int type,int val))ptr;
478                 *fptr=SSL_get_info_callback(ssl);
479                 }
480                 break;
481         default:
482                 ret=BIO_ctrl(ssl->rbio,cmd,num,ptr);
483                 break;
484                 }
485         return(ret);
486         }

488 static long ssl_callback_ctrl(BIO *b, int cmd, bio_info_cb *fp)
489         {
490         SSL *ssl;
491         BIO_SSL *bs;
492         long ret=1;

494         bs=(BIO_SSL *)b->ptr;
495         ssl=bs->ssl;
496         switch (cmd)
497                 {
498         case BIO_CTRL_SET_CALLBACK:
499                 {
500                 /* FIXME: setting this via a completely different prototype
501                    seems like a crap idea */
502                 SSL_set_info_callback(ssl,(void (*)(const SSL *,int,int))fp);
503                 }
504                 break;
505         default:
506                 ret=BIO_callback_ctrl(ssl->rbio,cmd,fp);
507                 break;
508                 }
509         return(ret);
510         }

512 static int ssl_puts(BIO *bp, const char *str)
513         {
514         int n,ret;

516         n=strlen(str);
517         ret=BIO_write(bp,str,n);
518         return(ret);
519         }

521 BIO *BIO_new_buffer_ssl_connect(SSL_CTX *ctx)
522         {
523 #ifndef OPENSSL_NO_SOCK
```

```
524         BIO *ret=NULL,*buf=NULL,*ssl=NULL;

526         if ((buf=BIO_new(BIO_f_buffer())) == NULL)
527                 return(NULL);
528         if ((ssl=BIO_new_ssl_connect(ctx)) == NULL)
529                 goto err;
530         if ((ret=BIO_push(buf,ssl)) == NULL)
531                 goto err;
532         return(ret);
533 err:
534         if (buf != NULL) BIO_free(buf);
535         if (ssl != NULL) BIO_free(ssl);
536 #endif
537         return(NULL);
538         }

540 BIO *BIO_new_ssl_connect(SSL_CTX *ctx)
541         {
542 #ifndef OPENSSL_NO_SOCK
543         BIO *ret=NULL,*con=NULL,*ssl=NULL;

545         if ((con=BIO_new(BIO_s_connect())) == NULL)
546                 return(NULL);
547         if ((ssl=BIO_new_ssl(ctx,1)) == NULL)
548                 goto err;
549         if ((ret=BIO_push(ssl,con)) == NULL)
550                 goto err;
551         return(ret);
552 err:
553         if (con != NULL) BIO_free(con);
554 #endif
555         return(NULL);
556         }

558 BIO *BIO_new_ssl(SSL_CTX *ctx, int client)
559         {
560         BIO *ret;
561         SSL *ssl;

563         if ((ret=BIO_new(BIO_f_ssl())) == NULL)
564                 return(NULL);
565         if ((ssl=SSL_new(ctx)) == NULL)
566                 {
567                 BIO_free(ret);
568                 return(NULL);
569                 }
570         if (client)
571                 SSL_set_connect_state(ssl);
572         else
573                 SSL_set_accept_state(ssl);

575         BIO_set_ssl(ret,ssl,BIO_CLOSE);
576         return(ret);
577         }

579 int BIO_ssl_copy_session_id(BIO *t, BIO *f)
580         {
581         t=BIO_find_type(t,BIO_TYPE_SSL);
582         f=BIO_find_type(f,BIO_TYPE_SSL);
583         if ((t == NULL) || (f == NULL))
584                 return(0);
585         if (    (((BIO_SSL *)t->ptr)->ssl == NULL) ||
586                 (((BIO_SSL *)f->ptr)->ssl == NULL))
587                 return(0);
588         SSL_copy_session_id(((BIO_SSL *)t->ptr)->ssl,((BIO_SSL *)f->ptr)->ssl);
589         return(1);
```



```
590         }

592 void BIO_ssl_shutdown(BIO *b)
593         {
594         SSL *s;

596         while (b != NULL)
597                 {
598                 if (b->method->type == BIO_TYPE_SSL)
599                         {
600                         s=((BIO_SSL *)b->ptr)->ssl;
601                         SSL_shutdown(s);
602                         break;
603                         }
604                 b=b->next_bio;
605                 }
606         }
607 #endif /* ! codereview */
```

```
**********************************************************
   45549 Wed Aug 13 19:53:34 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_both.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/d1_both.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1998-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
```



```
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <limits.h>
#include <string.h>
#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>

#define RSMBLY_BITMASK_SIZE(msg_len) (((msg_len) + 7) / 8)
```

```
#define RSMBLY_BITMASK_MARK(bitmask, start, end) { \
                        if ((end) - (start) <= 8) { \
                                long ii; \
                                for (ii = (start); ii < (end); ii++) bitmask[((i
                        } else { \
                                long ii; \
                                bitmask[((start) >> 3)] |= bitmask_start_values[
                                for (ii = (((start) >> 3) + 1); ii < ((((end) -
                                bitmask[(((end) - 1) >> 3)] |= bitmask_end_value
                        } }

#define RSMBLY_BITMASK_IS_COMPLETE(bitmask, msg_len, is_complete) { \
                        long ii; \
                        OPENSSL_assert((msg_len) > 0); \
                        is_complete = 1; \
                        if (bitmask[(((msg_len) - 1) >> 3)] != bitmask_end_value
                        if (is_complete) for (ii = (((msg_len) - 1) >> 3) - 1; i
                                if (bitmask[ii] != 0xff) { is_complete = 0; brea

#if 0
#define RSMBLY_BITMASK_PRINT(bitmask, msg_len) { \
                        long ii; \
                        printf("bitmask: "); for (ii = 0; ii < (msg_len); ii++)
                        printf("%d ", (bitmask[ii >> 3] & (1 << (ii & 7))) >> (i
                        printf("\n"); }
#endif

static unsigned char bitmask_start_values[] = {0xff, 0xfe, 0xfc, 0xf8, 0xf0, 0xe
static unsigned char bitmask_end_values[]   = {0xff, 0x01, 0x03, 0x07, 0x0f, 0x1

/* XDTLS:  figure out the right values */
static unsigned int g_probable_mtu[] = {1500 - 28, 512 - 28, 256 - 28};

static unsigned int dtls1_guess_mtu(unsigned int curr_mtu);
static void dtls1_fix_message_header(SSL *s, unsigned long frag_off,
        unsigned long frag_len);
static unsigned char *dtls1_write_message_header(SSL *s,
        unsigned char *p);
static void dtls1_set_message_header_int(SSL *s, unsigned char mt,
        unsigned long len, unsigned short seq_num, unsigned long frag_off,
        unsigned long frag_len);
static long dtls1_get_message_fragment(SSL *s, int st1, int stn,
        long max, int *ok);

static hm_fragment *
dtls1_hm_fragment_new(unsigned long frag_len, int reassembly)
        {
        hm_fragment *frag = NULL;
        unsigned char *buf = NULL;
        unsigned char *bitmask = NULL;

        frag = (hm_fragment *)OPENSSL_malloc(sizeof(hm_fragment));
        if ( frag == NULL)
                return NULL;

        if (frag_len)
                {
                buf = (unsigned char *)OPENSSL_malloc(frag_len);
                if ( buf == NULL)
                        {
                        OPENSSL_free(frag);
                        return NULL;
                        }
                }

        /* zero length fragment gets zero frag->fragment */
```



```
        frag->fragment = buf;

        /* Initialize reassembly bitmask if necessary */
        if (reassembly)
                {
                bitmask = (unsigned char *)OPENSSL_malloc(RSMBLY_BITMASK_SIZE(fr
                if (bitmask == NULL)
                        {
                        if (buf != NULL) OPENSSL_free(buf);
                        OPENSSL_free(frag);
                        return NULL;
                        }
                memset(bitmask, 0, RSMBLY_BITMASK_SIZE(frag_len));
                }

        frag->reassembly = bitmask;

        return frag;
        }

static void
dtls1_hm_fragment_free(hm_fragment *frag)
        {

        if (frag->msg_header.is_ccs)
                {
                EVP_CIPHER_CTX_free(frag->msg_header.saved_retransmit_state.enc_
                EVP_MD_CTX_destroy(frag->msg_header.saved_retransmit_state.write
                }
        if (frag->fragment) OPENSSL_free(frag->fragment);
        if (frag->reassembly) OPENSSL_free(frag->reassembly);
        OPENSSL_free(frag);
        }

/* send s->init_buf in records of type 'type' (SSL3_RT_HANDSHAKE or SSL3_RT_CHAN
int dtls1_do_write(SSL *s, int type)
        {
        int ret;
        int curr_mtu;
        unsigned int len, frag_off, mac_size, blocksize;

        /* AHA!  Figure out the MTU, and stick to the right size */
        if (s->d1->mtu < dtls1_min_mtu() && !(SSL_get_options(s) & SSL_OP_NO_QUE
                {
                s->d1->mtu =
                        BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_QUERY_MTU, 0, N

                /* I've seen the kernel return bogus numbers when it doesn't kno
                 * (initial write), so just make sure we have a reasonable numbe
                if (s->d1->mtu < dtls1_min_mtu())
                        {
                        s->d1->mtu = 0;
                        s->d1->mtu = dtls1_guess_mtu(s->d1->mtu);
                        BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SET_MTU,
                                s->d1->mtu, NULL);
                        }
                }
#if 0
        mtu = s->d1->mtu;

        fprintf(stderr, "using MTU = %d\n", mtu);

        mtu -= (DTLS1_HM_HEADER_LENGTH + DTLS1_RT_HEADER_LENGTH);

        curr_mtu = mtu - BIO_wpending(SSL_get_wbio(s));
```

```
260         if ( curr_mtu > 0)
261                 mtu = curr_mtu;
262         else if ( ( ret = BIO_flush(SSL_get_wbio(s))) <= 0)
263                 return ret;
264
265         if ( BIO_wpending(SSL_get_wbio(s)) + s->init_num >= mtu)
266                 {
267                 ret = BIO_flush(SSL_get_wbio(s));
268                 if ( ret <= 0)
269                         return ret;
270                 mtu = s->d1->mtu - (DTLS1_HM_HEADER_LENGTH + DTLS1_RT_HEADER_LEN
271                 }
272 #endif
273
274         OPENSSL_assert(s->d1->mtu >= dtls1_min_mtu());  /* should have something
275
276         if ( s->init_off == 0  && type == SSL3_RT_HANDSHAKE)
277                 OPENSSL_assert(s->init_num ==
278                         (int)s->d1->w_msg_hdr.msg_len + DTLS1_HM_HEADER_LENGTH);
279
280         if (s->write_hash)
281                 mac_size = EVP_MD_CTX_size(s->write_hash);
282         else
283                 mac_size = 0;
284
285         if (s->enc_write_ctx &&
286                 (EVP_CIPHER_mode( s->enc_write_ctx->cipher) & EVP_CIPH_CBC_MODE)
287                 blocksize = 2 * EVP_CIPHER_block_size(s->enc_write_ctx->cipher);
288         else
289                 blocksize = 0;
290
291         frag_off = 0;
292         while( s->init_num)
293                 {
294                 curr_mtu = s->d1->mtu - BIO_wpending(SSL_get_wbio(s)) -
295                         DTLS1_RT_HEADER_LENGTH - mac_size - blocksize;
296
297                 if ( curr_mtu <= DTLS1_HM_HEADER_LENGTH)
298                         {
299                         /* grr.. we could get an error if MTU picked was wrong *
300                         ret = BIO_flush(SSL_get_wbio(s));
301                         if ( ret <= 0)
302                                 return ret;
303                         curr_mtu = s->d1->mtu - DTLS1_RT_HEADER_LENGTH -
304                                 mac_size - blocksize;
305                         }
306
307                 if ( s->init_num > curr_mtu)
308                         len = curr_mtu;
309                 else
310                         len = s->init_num;
311
312
313                 /* XDTLS: this function is too long.  split out the CCS part */
314                 if ( type == SSL3_RT_HANDSHAKE)
315                         {
316                         if ( s->init_off != 0)
317                                 {
318                                 OPENSSL_assert(s->init_off > DTLS1_HM_HEADER_LEN
319                                 s->init_off -= DTLS1_HM_HEADER_LENGTH;
320                                 s->init_num += DTLS1_HM_HEADER_LENGTH;
321
322                                 if ( s->init_num > curr_mtu)
323                                         len = curr_mtu;
324                                 else
325                                         len = s->init_num;
```



```
326                                 }
327
328                         dtls1_fix_message_header(s, frag_off,
329                                 len - DTLS1_HM_HEADER_LENGTH);
330
331                         dtls1_write_message_header(s, (unsigned char *)&s->init_
332
333                         OPENSSL_assert(len >= DTLS1_HM_HEADER_LENGTH);
334                         }
335
336                 ret=dtls1_write_bytes(s,type,&s->init_buf->data[s->init_off],
337                         len);
338                 if (ret < 0)
339                         {
340                         /* might need to update MTU here, but we don't know
341                          * which previous packet caused the failure -- so can't
342                          * really retransmit anything.  continue as if everythin
343                          * is fine and wait for an alert to handle the
344                          * retransmit
345                          */
346                         if ( BIO_ctrl(SSL_get_wbio(s),
347                                 BIO_CTRL_DGRAM_MTU_EXCEEDED, 0, NULL) > 0 )
348                                 s->d1->mtu = BIO_ctrl(SSL_get_wbio(s),
349                                         BIO_CTRL_DGRAM_QUERY_MTU, 0, NULL);
350                         else
351                                 return(-1);
352                         }
353                 else
354                         {
355
356                         /* bad if this assert fails, only part of the handshake
357                          * message got sent.  but why would this happen? */
358                         OPENSSL_assert(len == (unsigned int)ret);
359
360                         if (type == SSL3_RT_HANDSHAKE && ! s->d1->retransmitting
361                                 {
362                                 /* should not be done for 'Hello Request's, but
363                                  * we'll ignore the result anyway */
364                                 unsigned char *p = (unsigned char *)&s->init_buf
365                                 const struct hm_header_st *msg_hdr = &s->d1->w_m
366                                 int xlen;
367
368                                 if (frag_off == 0 && s->version != DTLS1_BAD_VER
369                                         {
370                                         /* reconstruct message header is if it
371                                          * is being sent in single fragment */
372                                         *p++ = msg_hdr->type;
373                                         l2n3(msg_hdr->msg_len,p);
374                                         s2n (msg_hdr->seq,p);
375                                         l2n3(0,p);
376                                         l2n3(msg_hdr->msg_len,p);
377                                         p  -= DTLS1_HM_HEADER_LENGTH;
378                                         xlen = ret;
379                                         }
380                                 else
381                                         {
382                                         p  += DTLS1_HM_HEADER_LENGTH;
383                                         xlen = ret - DTLS1_HM_HEADER_LENGTH;
384                                         }
385
386                                 ssl3_finish_mac(s, p, xlen);
387                                 }
388
389                         if (ret == s->init_num)
390                                 {
391                                 if (s->msg_callback)
```

```
392                                     s->msg_callback(1, s->version, type, s->
393                                             (size_t)(s->init_off + s->init_n
394                                             s->msg_callback_arg);

396                                 s->init_off = 0;  /* done writing this message *
397                                 s->init_num = 0;

399                                 return(1);
400                                 }
401                         s->init_off+=ret;
402                         s->init_num-=ret;
403                         frag_off += (ret -= DTLS1_HM_HEADER_LENGTH);
404                         }
405                 }
406         return(0);
407         }


410 /* Obtain handshake message of message type 'mt' (any if mt == -1),
411  * maximum acceptable body length 'max'.
412  * Read an entire handshake message.  Handshake messages arrive in
413  * fragments.
414  */
415 long dtls1_get_message(SSL *s, int st1, int stn, int mt, long max, int *ok)
416         {
417         int i, al;
418         struct hm_header_st *msg_hdr;
419         unsigned char *p;
420         unsigned long msg_len;

422         /* s3->tmp is used to store messages that are unexpected, caused
423          * by the absence of an optional handshake message */
424         if (s->s3->tmp.reuse_message)
425                 {
426                 s->s3->tmp.reuse_message=0;
427                 if ((mt >= 0) && (s->s3->tmp.message_type != mt))
428                         {
429                         al=SSL_AD_UNEXPECTED_MESSAGE;
430                         SSLerr(SSL_F_DTLS1_GET_MESSAGE,SSL_R_UNEXPECTED_MESSAGE)
431                         goto f_err;
432                         }
433                 *ok=1;
434                 s->init_msg = s->init_buf->data + DTLS1_HM_HEADER_LENGTH;
435                 s->init_num = (int)s->s3->tmp.message_size;
436                 return s->init_num;
437                 }

439         msg_hdr = &s->d1->r_msg_hdr;
440         memset(msg_hdr, 0x00, sizeof(struct hm_header_st));

442 again:
443         i = dtls1_get_message_fragment(s, st1, stn, max, ok);
444         if ( i == DTLS1_HM_BAD_FRAGMENT ||
445                 i == DTLS1_HM_FRAGMENT_RETRY)  /* bad fragment received */
446                 goto again;
447         else if ( i <= 0 && !*ok)
448                 return i;

450         p = (unsigned char *)s->init_buf->data;
451         msg_len = msg_hdr->msg_len;

453         /* reconstruct message header */
454         *(p++) = msg_hdr->type;
455         l2n3(msg_len,p);
456         s2n (msg_hdr->seq,p);
457         l2n3(0,p);
```



```
458         l2n3(msg_len,p);
459         if (s->version != DTLS1_BAD_VER) {
460                 p       -= DTLS1_HM_HEADER_LENGTH;
461                 msg_len += DTLS1_HM_HEADER_LENGTH;
462         }

464         ssl3_finish_mac(s, p, msg_len);
465         if (s->msg_callback)
466                 s->msg_callback(0, s->version, SSL3_RT_HANDSHAKE,
467                         p, msg_len,
468                         s, s->msg_callback_arg);

470         memset(msg_hdr, 0x00, sizeof(struct hm_header_st));

472         /* Don't change sequence numbers while listening */
473         if (!s->d1->listen)
474                 s->d1->handshake_read_seq++;

476         s->init_msg = s->init_buf->data + DTLS1_HM_HEADER_LENGTH;
477         return s->init_num;

479 f_err:
480         ssl3_send_alert(s,SSL3_AL_FATAL,al);
481         *ok = 0;
482         return -1;
483         }


486 static int dtls1_preprocess_fragment(SSL *s,struct hm_header_st *msg_hdr,int max
487         {
488         size_t frag_off,frag_len,msg_len;

490         msg_len  = msg_hdr->msg_len;
491         frag_off = msg_hdr->frag_off;
492         frag_len = msg_hdr->frag_len;

494         /* sanity checking */
495         if ( (frag_off+frag_len) > msg_len)
496                 {
497                 SSLerr(SSL_F_DTLS1_PREPROCESS_FRAGMENT,SSL_R_EXCESSIVE_MESSAGE_S
498                 return SSL_AD_ILLEGAL_PARAMETER;
499                 }

501         if ( (frag_off+frag_len) > (unsigned long)max)
502                 {
503                 SSLerr(SSL_F_DTLS1_PREPROCESS_FRAGMENT,SSL_R_EXCESSIVE_MESSAGE_S
504                 return SSL_AD_ILLEGAL_PARAMETER;
505                 }

507         if ( s->d1->r_msg_hdr.frag_off == 0) /* first fragment */
508                 {
509                 /* msg_len is limited to 2^24, but is effectively checked
510                  * against max above */
511                 if (!BUF_MEM_grow_clean(s->init_buf,msg_len+DTLS1_HM_HEADER_LENG
512                         {
513                         SSLerr(SSL_F_DTLS1_PREPROCESS_FRAGMENT,ERR_R_BUF_LIB);
514                         return SSL_AD_INTERNAL_ERROR;
515                         }

517                 s->s3->tmp.message_size  = msg_len;
518                 s->d1->r_msg_hdr.msg_len = msg_len;
519                 s->s3->tmp.message_type  = msg_hdr->type;
520                 s->d1->r_msg_hdr.type    = msg_hdr->type;
521                 s->d1->r_msg_hdr.seq     = msg_hdr->seq;
522                 }
523         else if (msg_len != s->d1->r_msg_hdr.msg_len)
```

```
524                 {
525                 /* They must be playing with us! BTW, failure to enforce
526                  * upper limit would open possibility for buffer overrun. */
527                 SSLerr(SSL_F_DTLS1_PREPROCESS_FRAGMENT,SSL_R_EXCESSIVE_MESSAGE_S
528                 return SSL_AD_ILLEGAL_PARAMETER;
529                 }

531         return 0; /* no error */
532         }


535 static int
536 dtls1_retrieve_buffered_fragment(SSL *s, long max, int *ok)
537         {
538         /* (0) check whether the desired fragment is available
539          * if so:
540          * (1) copy over the fragment to s->init_buf->data[]
541          * (2) update s->init_num
542          */
543         pitem *item;
544         hm_fragment *frag;
545         int al;

547         *ok = 0;
548         item = pqueue_peek(s->d1->buffered_messages);
549         if ( item == NULL)
550                 return 0;

552         frag = (hm_fragment *)item->data;
553
554         /* Don't return if reassembly still in progress */
555         if (frag->reassembly != NULL)
556                 return 0;

558         if ( s->d1->handshake_read_seq == frag->msg_header.seq)
559                 {
560                 unsigned long frag_len = frag->msg_header.frag_len;
561                 pqueue_pop(s->d1->buffered_messages);

563                 al=dtls1_preprocess_fragment(s,&frag->msg_header,max);

565                 if (al==0) /* no alert */
566                         {
567                         unsigned char *p = (unsigned char *)s->init_buf->data+DT
568                         memcpy(&p[frag->msg_header.frag_off],
569                                 frag->fragment,frag->msg_header.frag_len);
570                         }

572                 dtls1_hm_fragment_free(frag);
573                 pitem_free(item);

575                 if (al==0)
576                         {
577                         *ok = 1;
578                         return frag_len;
579                         }

581                 ssl3_send_alert(s,SSL3_AL_FATAL,al);
582                 s->init_num = 0;
583                 *ok = 0;
584                 return -1;
585                 }
586         else
587                 return 0;
588         }
```



```
590 /* dtls1_max_handshake_message_len returns the maximum number of bytes
591  * permitted in a DTLS handshake message for |s|. The minimum is 16KB, but may
592  * be greater if the maximum certificate list size requires it. */
593 static unsigned long dtls1_max_handshake_message_len(const SSL *s)
594         {
595         unsigned long max_len = DTLS1_HM_HEADER_LENGTH + SSL3_RT_MAX_ENCRYPTED_L
596         if (max_len < (unsigned long)s->max_cert_list)
597                 return s->max_cert_list;
598         return max_len;
599         }

601 static int
602 dtls1_reassemble_fragment(SSL *s, const struct hm_header_st* msg_hdr, int *ok)
603         {
604         hm_fragment *frag = NULL;
605         pitem *item = NULL;
606         int i = -1, is_complete;
607         unsigned char seq64be[8];
608         unsigned long frag_len = msg_hdr->frag_len;

610         if ((msg_hdr->frag_off+frag_len) > msg_hdr->msg_len ||
611                 msg_hdr->msg_len > dtls1_max_handshake_message_len(s))
612                 goto err;

614         if (frag_len == 0)
615                 return DTLS1_HM_FRAGMENT_RETRY;

617         /* Try to find item in queue */
618         memset(seq64be,0,sizeof(seq64be));
619         seq64be[6] = (unsigned char) (msg_hdr->seq>>8);
620         seq64be[7] = (unsigned char) msg_hdr->seq;
621         item = pqueue_find(s->d1->buffered_messages, seq64be);

623         if (item == NULL)
624                 {
625                 frag = dtls1_hm_fragment_new(msg_hdr->msg_len, 1);
626                 if ( frag == NULL)
627                         goto err;
628                 memcpy(&(frag->msg_header), msg_hdr, sizeof(*msg_hdr));
629                 frag->msg_header.frag_len = frag->msg_header.msg_len;
630                 frag->msg_header.frag_off = 0;
631                 }
632         else
633                 {
634                 frag = (hm_fragment*) item->data;
635                 if (frag->msg_header.msg_len != msg_hdr->msg_len)
636                         {
637                         item = NULL;
638                         frag = NULL;
639                         goto err;
640                         }
641                 }

643         /* If message is already reassembled, this must be a
644          * retransmit and can be dropped. In this case item != NULL and so frag
645          * does not need to be freed.
646          */
647         if (frag->reassembly == NULL)
648                 {
649                 unsigned char devnull [256];

651                 while (frag_len)
652                         {
653                         i = s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,
654                                 devnull,
655                                 frag_len>sizeof(devnull)?sizeof(devnull):frag_le
```

```
656                         if (i<=0) goto err;
657                         frag_len -= i;
658                         }
659                 return DTLS1_HM_FRAGMENT_RETRY;
660                 }

662         /* read the body of the fragment (header has already been read */
663         i = s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,
664                 frag->fragment + msg_hdr->frag_off,frag_len,0);
665         if ((unsigned long)i!=frag_len)
666                 i=-1;
667         if (i<=0)
668                 goto err;

670         RSMBLY_BITMASK_MARK(frag->reassembly, (long)msg_hdr->frag_off,
671                             (long)(msg_hdr->frag_off + frag_len));

673         RSMBLY_BITMASK_IS_COMPLETE(frag->reassembly, (long)msg_hdr->msg_len,
674                                    is_complete);

676         if (is_complete)
677                 {
678                 OPENSSL_free(frag->reassembly);
679                 frag->reassembly = NULL;
680                 }

682         if (item == NULL)
683                 {
684                 item = pitem_new(seq64be, frag);
685                 if (item == NULL)
686                         {
687                         i = -1;
688                         goto err;
689                         }

691                 item = pqueue_insert(s->d1->buffered_messages, item);
692                 /* pqueue_insert fails iff a duplicate item is inserted.
693                  * However, |item| cannot be a duplicate. If it were,
694                  * |pqueue_find|, above, would have returned it and control
695                  * would never have reached this branch. */
696                 OPENSSL_assert(item != NULL);
697                 }

699         return DTLS1_HM_FRAGMENT_RETRY;

701 err:
702         if (frag != NULL && item == NULL) dtls1_hm_fragment_free(frag);
703         *ok = 0;
704         return i;
705         }


708 static int
709 dtls1_process_out_of_seq_message(SSL *s, const struct hm_header_st* msg_hdr, int
710 {
711         int i=-1;
712         hm_fragment *frag = NULL;
713         pitem *item = NULL;
714         unsigned char seq64be[8];
715         unsigned long frag_len = msg_hdr->frag_len;

717         if ((msg_hdr->frag_off+frag_len) > msg_hdr->msg_len)
718                 goto err;

720         /* Try to find item in queue, to prevent duplicate entries */
721         memset(seq64be,0,sizeof(seq64be));
```



```
722         seq64be[6] = (unsigned char) (msg_hdr->seq>>8);
723         seq64be[7] = (unsigned char) msg_hdr->seq;
724         item = pqueue_find(s->d1->buffered_messages, seq64be);

726         /* If we already have an entry and this one is a fragment,
727          * don't discard it and rather try to reassemble it.
728          */
729         if (item != NULL && frag_len != msg_hdr->msg_len)
730                 item = NULL;

732         /* Discard the message if sequence number was already there, is
733          * too far in the future, already in the queue or if we received
734          * a FINISHED before the SERVER_HELLO, which then must be a stale
735          * retransmit.
736          */
737         if (msg_hdr->seq <= s->d1->handshake_read_seq ||
738                 msg_hdr->seq > s->d1->handshake_read_seq + 10 || item != NULL ||
739                 (s->d1->handshake_read_seq == 0 && msg_hdr->type == SSL3_MT_FINI
740                 {
741                 unsigned char devnull [256];

743                 while (frag_len)
744                         {
745                         i = s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,
746                                 devnull,
747                                 frag_len>sizeof(devnull)?sizeof(devnull):frag_le
748                         if (i<=0) goto err;
749                         frag_len -= i;
750                         }
751                 }
752         else
753                 {
754                 if (frag_len != msg_hdr->msg_len)
755                         return dtls1_reassemble_fragment(s, msg_hdr, ok);

757                 if (frag_len > dtls1_max_handshake_message_len(s))
758                         goto err;

760                 frag = dtls1_hm_fragment_new(frag_len, 0);
761                 if ( frag == NULL)
762                         goto err;

764                 memcpy(&(frag->msg_header), msg_hdr, sizeof(*msg_hdr));

766                 if (frag_len)
767                         {
768                         /* read the body of the fragment (header has already bee
769                         i = s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,
770                                 frag->fragment,frag_len,0);
771                         if ((unsigned long)i!=frag_len)
772                                 i=-1;
773                         if (i<=0)
774                                 goto err;
775                         }

777                 item = pitem_new(seq64be, frag);
778                 if ( item == NULL)
779                         goto err;

781                 item = pqueue_insert(s->d1->buffered_messages, item);
782                 /* pqueue_insert fails iff a duplicate item is inserted.
783                  * However, |item| cannot be a duplicate. If it were,
784                  * |pqueue_find|, above, would have returned it. Then, either
785                  * |frag_len| != |msg_hdr->msg_len| in which case |item| is set
786                  * to NULL and it will have been processed with
787                  * |dtls1_reassemble_fragment|, above, or the record will have
```

```
788                  * been discarded. */
789                 OPENSSL_assert(item != NULL);
790                 }

792         return DTLS1_HM_FRAGMENT_RETRY;

794 err:
795         if (frag != NULL && item == NULL) dtls1_hm_fragment_free(frag);
796         *ok = 0;
797         return i;
798         }


801 static long
802 dtls1_get_message_fragment(SSL *s, int st1, int stn, long max, int *ok)
803         {
804         unsigned char wire[DTLS1_HM_HEADER_LENGTH];
805         unsigned long len, frag_off, frag_len;
806         int i,al;
807         struct hm_header_st msg_hdr;

809         redo:
810         /* see if we have the required fragment already */
811         if ((frag_len = dtls1_retrieve_buffered_fragment(s,max,ok)) || *ok)
812                 {
813                 if (*ok)        s->init_num = frag_len;
814                 return frag_len;
815                 }

817         /* read handshake message header */
818         i=s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,wire,
819                 DTLS1_HM_HEADER_LENGTH, 0);
820         if (i <= 0)     /* nbio, or an error */
821                 {
822                 s->rwstate=SSL_READING;
823                 *ok = 0;
824                 return i;
825                 }
826         /* Handshake fails if message header is incomplete */
827         if (i != DTLS1_HM_HEADER_LENGTH)
828                 {
829                 al=SSL_AD_UNEXPECTED_MESSAGE;
830                 SSLerr(SSL_F_DTLS1_GET_MESSAGE_FRAGMENT,SSL_R_UNEXPECTED_MESSAGE
831                 goto f_err;
832                 }

834         /* parse the message fragment header */
835         dtls1_get_message_header(wire, &msg_hdr);

837         /*
838          * if this is a future (or stale) message it gets buffered
839          * (or dropped)--no further processing at this time
840          * While listening, we accept seq 1 (ClientHello with cookie)
841          * although we're still expecting seq 0 (ClientHello)
842          */
843         if (msg_hdr.seq != s->d1->handshake_read_seq && !(s->d1->listen && msg_h
844                 return dtls1_process_out_of_seq_message(s, &msg_hdr, ok);

846         len = msg_hdr.msg_len;
847         frag_off = msg_hdr.frag_off;
848         frag_len = msg_hdr.frag_len;

850         if (frag_len && frag_len < len)
851                 return dtls1_reassemble_fragment(s, &msg_hdr, ok);

853         if (!s->server && s->d1->r_msg_hdr.frag_off == 0 &&
```



```
854                 wire[0] == SSL3_MT_HELLO_REQUEST)
855                 {
856                 /* The server may always send 'Hello Request' messages --
857                  * we are doing a handshake anyway now, so ignore them
858                  * if their format is correct. Does not count for
859                  * 'Finished' MAC. */
860                 if (wire[1] == 0 && wire[2] == 0 && wire[3] == 0)
861                         {
862                         if (s->msg_callback)
863                                 s->msg_callback(0, s->version, SSL3_RT_HANDSHAKE
864                                         wire, DTLS1_HM_HEADER_LENGTH, s,
865                                         s->msg_callback_arg);
866
867                         s->init_num = 0;
868                         goto redo;
869                         }
870                 else /* Incorrectly formated Hello request */
871                         {
872                         al=SSL_AD_UNEXPECTED_MESSAGE;
873                         SSLerr(SSL_F_DTLS1_GET_MESSAGE_FRAGMENT,SSL_R_UNEXPECTED
874                         goto f_err;
875                         }
876                 }

878         if ((al=dtls1_preprocess_fragment(s,&msg_hdr,max)))
879                 goto f_err;

881         /* XDTLS:  ressurect this when restart is in place */
882         s->state=stn;

884         if ( frag_len > 0)
885                 {
886                 unsigned char *p=(unsigned char *)s->init_buf->data+DTLS1_HM_HEA

888                 i=s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,
889                         &p[frag_off],frag_len,0);
890                 /* XDTLS:  fix this--message fragments cannot span multiple pack
891                 if (i <= 0)
892                         {
893                         s->rwstate=SSL_READING;
894                         *ok = 0;
895                         return i;
896                         }
897                 }
898         else
899                 i = 0;

901         /* XDTLS:  an incorrectly formatted fragment should cause the
902          * handshake to fail */
903         if (i != (int)frag_len)
904                 {
905                 al=SSL3_AD_ILLEGAL_PARAMETER;
906                 SSLerr(SSL_F_DTLS1_GET_MESSAGE_FRAGMENT,SSL3_AD_ILLEGAL_PARAMETE
907                 goto f_err;
908                 }

910         *ok = 1;

912         /* Note that s->init_num is *not* used as current offset in
913          * s->init_buf->data, but as a counter summing up fragments'
914          * lengths: as soon as they sum up to handshake packet
915          * length, we assume we have got all the fragments. */
916         s->init_num = frag_len;
917         return frag_len;

919 f_err:
```

```
920         ssl3_send_alert(s,SSL3_AL_FATAL,al);
921         s->init_num = 0;

923         *ok=0;
924         return(-1);
925         }

927 int dtls1_send_finished(SSL *s, int a, int b, const char *sender, int slen)
928         {
929         unsigned char *p,*d;
930         int i;
931         unsigned long l;

933         if (s->state == a)
934                 {
935                 d=(unsigned char *)s->init_buf->data;
936                 p= &(d[DTLS1_HM_HEADER_LENGTH]);

938                 i=s->method->ssl3_enc->final_finish_mac(s,
939                         sender,slen,s->s3->tmp.finish_md);
940                 s->s3->tmp.finish_md_len = i;
941                 memcpy(p, s->s3->tmp.finish_md, i);
942                 p+=i;
943                 l=i;

945         /* Copy the finished so we can use it for
946          * renegotiation checks
947          */
948         if(s->type == SSL_ST_CONNECT)
949                 {
950                 OPENSSL_assert(i <= EVP_MAX_MD_SIZE);
951                 memcpy(s->s3->previous_client_finished,
952                         s->s3->tmp.finish_md, i);
953                 s->s3->previous_client_finished_len=i;
954                 }
955         else
956                 {
957                 OPENSSL_assert(i <= EVP_MAX_MD_SIZE);
958                 memcpy(s->s3->previous_server_finished,
959                        s->s3->tmp.finish_md, i);
960                 s->s3->previous_server_finished_len=i;
961                 }

963 #ifdef OPENSSL_SYS_WIN16
964                 /* MSVC 1.5 does not clear the top bytes of the word unless
965                  * I do this.
966                  */
967                 l&=0xffff;
968 #endif

970                 d = dtls1_set_message_header(s, d, SSL3_MT_FINISHED, l, 0, l);
971                 s->init_num=(int)l+DTLS1_HM_HEADER_LENGTH;
972                 s->init_off=0;

974                 /* buffer the message to handle re-xmits */
975                 dtls1_buffer_message(s, 0);

977                 s->state=b;
978                 }

980         /* SSL3_ST_SEND_xxxxxx_HELLO_B */
981         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
982         }

984 /* for these 2 messages, we need to
985  * ssl->enc_read_ctx                    re-init
```



```
986  * ssl->s3->read_sequence               zero
987  * ssl->s3->read_mac_secret             re-init
988  * ssl->session->read_sym_enc           assign
989  * ssl->session->read_compression       assign
990  * ssl->session->read_hash              assign
991  */
992 int dtls1_send_change_cipher_spec(SSL *s, int a, int b)
993         {
994         unsigned char *p;

996         if (s->state == a)
997                 {
998                 p=(unsigned char *)s->init_buf->data;
999                 *p++=SSL3_MT_CCS;
1000                s->d1->handshake_write_seq = s->d1->next_handshake_write_seq;
1001                s->init_num=DTLS1_CCS_HEADER_LENGTH;

1003                if (s->version == DTLS1_BAD_VER) {
1004                        s->d1->next_handshake_write_seq++;
1005                        s2n(s->d1->handshake_write_seq,p);
1006                        s->init_num+=2;
1007                }

1009                s->init_off=0;

1011                dtls1_set_message_header_int(s, SSL3_MT_CCS, 0,
1012                        s->d1->handshake_write_seq, 0, 0);

1014                /* buffer the message to handle re-xmits */
1015                dtls1_buffer_message(s, 1);

1017                s->state=b;
1018                }

1020        /* SSL3_ST_CW_CHANGE_B */
1021        return(dtls1_do_write(s,SSL3_RT_CHANGE_CIPHER_SPEC));
1022        }

1024 static int dtls1_add_cert_to_buf(BUF_MEM *buf, unsigned long *l, X509 *x)
1025        {
1026        int n;
1027        unsigned char *p;

1029        n=i2d_X509(x,NULL);
1030        if (!BUF_MEM_grow_clean(buf,(int)(n+(*l)+3)))
1031                {
1032                SSLerr(SSL_F_DTLS1_ADD_CERT_TO_BUF,ERR_R_BUF_LIB);
1033                return 0;
1034                }
1035        p=(unsigned char *)&(buf->data[*l]);
1036        l2n3(n,p);
1037        i2d_X509(x,&p);
1038        *l+=n+3;

1040        return 1;
1041        }
1042 unsigned long dtls1_output_cert_chain(SSL *s, X509 *x)
1043        {
1044        unsigned char *p;
1045        int i;
1046        unsigned long l= 3 + DTLS1_HM_HEADER_LENGTH;
1047        BUF_MEM *buf;

1049        /* TLSv1 sends a chain with nothing in it, instead of an alert */
1050        buf=s->init_buf;
1051        if (!BUF_MEM_grow_clean(buf,10))
```

```
1052                 {
1053                 SSLerr(SSL_F_DTLS1_OUTPUT_CERT_CHAIN,ERR_R_BUF_LIB);
1054                 return(0);
1055                 }
1056         if (x != NULL)
1057                 {
1058                 X509_STORE_CTX xs_ctx;

1060                 if (!X509_STORE_CTX_init(&xs_ctx,s->ctx->cert_store,x,NULL))
1061                         {
1062                         SSLerr(SSL_F_DTLS1_OUTPUT_CERT_CHAIN,ERR_R_X509_LIB);
1063                         return(0);
1064                         }
1065
1066                 X509_verify_cert(&xs_ctx);
1067                 /* Don't leave errors in the queue */
1068                 ERR_clear_error();
1069                 for (i=0; i < sk_X509_num(xs_ctx.chain); i++)
1070                         {
1071                         x = sk_X509_value(xs_ctx.chain, i);

1073                         if (!dtls1_add_cert_to_buf(buf, &l, x))
1074                                 {
1075                                 X509_STORE_CTX_cleanup(&xs_ctx);
1076                                 return 0;
1077                                 }
1078                         }
1079                 X509_STORE_CTX_cleanup(&xs_ctx);
1080                 }
1081         /* Thawte special :-) */
1082         for (i=0; i<sk_X509_num(s->ctx->extra_certs); i++)
1083                 {
1084                 x=sk_X509_value(s->ctx->extra_certs,i);
1085                 if (!dtls1_add_cert_to_buf(buf, &l, x))
1086                         return 0;
1087                 }

1089         l-= (3 + DTLS1_HM_HEADER_LENGTH);

1091         p=(unsigned char *)&(buf->data[DTLS1_HM_HEADER_LENGTH]);
1092         l2n3(l,p);
1093         l+=3;
1094         p=(unsigned char *)&(buf->data[0]);
1095         p = dtls1_set_message_header(s, p, SSL3_MT_CERTIFICATE, l, 0, l);

1097         l+=DTLS1_HM_HEADER_LENGTH;
1098         return(l);
1099         }

1101 int dtls1_read_failed(SSL *s, int code)
1102         {
1103         if ( code > 0)
1104                 {
1105                 fprintf( stderr, "invalid state reached %s:%d", __FILE__, __LINE
1106                 return 1;
1107                 }

1109         if (!dtls1_is_timer_expired(s))
1110                 {
1111                 /* not a timeout, none of our business,
1112                    let higher layers handle this.  in fact it's probably an erro
1113                 return code;
1114                 }

1116 #ifndef OPENSSL_NO_HEARTBEATS
1117         if (!SSL_in_init(s) && !s->tlsext_hb_pending)  /* done, no need to send
```



```
1118 #else
1119         if (!SSL_in_init(s))  /* done, no need to send a retransmit */
1120 #endif
1121                 {
1122                 BIO_set_flags(SSL_get_rbio(s), BIO_FLAGS_READ);
1123                 return code;
1124                 }

1126 #if 0 /* for now, each alert contains only one record number */
1127         item = pqueue_peek(state->rcvd_records);
1128         if ( item )
1129                 {
1130                 /* send an alert immediately for all the missing records */
1131                 }
1132         else
1133 #endif

1135 #if 0  /* no more alert sending, just retransmit the last set of messages */
1136         if ( state->timeout.read_timeouts >= DTLS1_TMO_READ_COUNT)
1137                 ssl3_send_alert(s,SSL3_AL_WARNING,
1138                         DTLS1_AD_MISSING_HANDSHAKE_MESSAGE);
1139 #endif

1141         return dtls1_handle_timeout(s);
1142         }

1144 int
1145 dtls1_get_queue_priority(unsigned short seq, int is_ccs)
1146         {
1147         /* The index of the retransmission queue actually is the message sequenc
1148          * since the queue only contains messages of a single handshake. However
1149          * ChangeCipherSpec has no message sequence number and so using only the
1150          * will result in the CCS and Finished having the same index. To prevent
1151          * the sequence number is multiplied by 2. In case of a CCS 1 is subtrac
1152          * This does not only differ CSS and Finished, it also maintains the ord
1153          * index (important for priority queues) and fits in the unsigned short
1154          */
1155         return seq * 2 - is_ccs;
1156         }

1158 int
1159 dtls1_retransmit_buffered_messages(SSL *s)
1160         {
1161         pqueue sent = s->d1->sent_messages;
1162         piterator iter;
1163         pitem *item;
1164         hm_fragment *frag;
1165         int found = 0;

1167         iter = pqueue_iterator(sent);

1169         for ( item = pqueue_next(&iter); item != NULL; item = pqueue_next(&iter)
1170                 {
1171                 frag = (hm_fragment *)item->data;
1172                         if ( dtls1_retransmit_message(s,
1173                                 (unsigned short)dtls1_get_queue_priority(frag->m
1174                                 0, &found) <= 0 && found)
1175                         {
1176                         fprintf(stderr, "dtls1_retransmit_message() failed\n");
1177                         return -1;
1178                         }
1179                 }

1181         return 1;
1182         }
```

```
1184 int
1185 dtls1_buffer_message(SSL *s, int is_ccs)
1186         {
1187         pitem *item;
1188         hm_fragment *frag;
1189         unsigned char seq64be[8];

1191         /* this function is called immediately after a message has
1192          * been serialized */
1193         OPENSSL_assert(s->init_off == 0);

1195         frag = dtls1_hm_fragment_new(s->init_num, 0);
1196         if (!frag)
1197                 return 0;

1199         memcpy(frag->fragment, s->init_buf->data, s->init_num);

1201         if ( is_ccs)
1202                 {
1203                 OPENSSL_assert(s->d1->w_msg_hdr.msg_len +
1204                                ((s->version==DTLS1_VERSION)?DTLS1_CCS_HEADER_LEN
1205                 }
1206         else
1207                 {
1208                 OPENSSL_assert(s->d1->w_msg_hdr.msg_len +
1209                         DTLS1_HM_HEADER_LENGTH == (unsigned int)s->init_num);
1210                 }

1212         frag->msg_header.msg_len = s->d1->w_msg_hdr.msg_len;
1213         frag->msg_header.seq = s->d1->w_msg_hdr.seq;
1214         frag->msg_header.type = s->d1->w_msg_hdr.type;
1215         frag->msg_header.frag_off = 0;
1216         frag->msg_header.frag_len = s->d1->w_msg_hdr.msg_len;
1217         frag->msg_header.is_ccs = is_ccs;

1219         /* save current state*/
1220         frag->msg_header.saved_retransmit_state.enc_write_ctx = s->enc_write_ctx
1221         frag->msg_header.saved_retransmit_state.write_hash = s->write_hash;
1222         frag->msg_header.saved_retransmit_state.compress = s->compress;
1223         frag->msg_header.saved_retransmit_state.session = s->session;
1224         frag->msg_header.saved_retransmit_state.epoch = s->d1->w_epoch;
1225
1226         memset(seq64be,0,sizeof(seq64be));
1227         seq64be[6] = (unsigned char)(dtls1_get_queue_priority(frag->msg_header.s
1228
1229         seq64be[7] = (unsigned char)(dtls1_get_queue_priority(frag->msg_header.s
1230


1232         item = pitem_new(seq64be, frag);
1233         if ( item == NULL)
1234                 {
1235                 dtls1_hm_fragment_free(frag);
1236                 return 0;
1237                 }

1239 #if 0
1240         fprintf( stderr, "buffered messge: \ttype = %xx\n", msg_buf->type);
1241         fprintf( stderr, "\t\t\t\t\tlen = %d\n", msg_buf->len);
1242         fprintf( stderr, "\t\t\t\t\tseq_num = %d\n", msg_buf->seq_num);
1243 #endif

1245         pqueue_insert(s->d1->sent_messages, item);
1246         return 1;
1247         }

1249 int
```



```
1250 dtls1_retransmit_message(SSL *s, unsigned short seq, unsigned long frag_off,
1251         int *found)
1252         {
1253         int ret;
1254         /* XDTLS: for now assuming that read/writes are blocking */
1255         pitem *item;
1256         hm_fragment *frag ;
1257         unsigned long header_length;
1258         unsigned char seq64be[8];
1259         struct dtls1_retransmit_state saved_state;
1260         unsigned char save_write_sequence[8];

1262         /*
1263           OPENSSL_assert(s->init_num == 0);
1264           OPENSSL_assert(s->init_off == 0);
1265          */

1267         /* XDTLS:  the requested message ought to be found, otherwise error */
1268         memset(seq64be,0,sizeof(seq64be));
1269         seq64be[6] = (unsigned char)(seq>>8);
1270         seq64be[7] = (unsigned char)seq;

1272         item = pqueue_find(s->d1->sent_messages, seq64be);
1273         if ( item == NULL)
1274                 {
1275                 fprintf(stderr, "retransmit:  message %d non-existant\n", seq);
1276                 *found = 0;
1277                 return 0;
1278                 }

1280         *found = 1;
1281         frag = (hm_fragment *)item->data;

1283         if ( frag->msg_header.is_ccs)
1284                 header_length = DTLS1_CCS_HEADER_LENGTH;
1285         else
1286                 header_length = DTLS1_HM_HEADER_LENGTH;

1288         memcpy(s->init_buf->data, frag->fragment,
1289                 frag->msg_header.msg_len + header_length);
1290                 s->init_num = frag->msg_header.msg_len + header_length;

1292         dtls1_set_message_header_int(s, frag->msg_header.type,
1293                 frag->msg_header.msg_len, frag->msg_header.seq, 0,
1294                 frag->msg_header.frag_len);

1296         /* save current state */
1297         saved_state.enc_write_ctx = s->enc_write_ctx;
1298         saved_state.write_hash = s->write_hash;
1299         saved_state.compress = s->compress;
1300         saved_state.session = s->session;
1301         saved_state.epoch = s->d1->w_epoch;
1302         saved_state.epoch = s->d1->w_epoch;
1303
1304         s->d1->retransmitting = 1;
1305
1306         /* restore state in which the message was originally sent */
1307         s->enc_write_ctx = frag->msg_header.saved_retransmit_state.enc_write_ctx
1308         s->write_hash = frag->msg_header.saved_retransmit_state.write_hash;
1309         s->compress = frag->msg_header.saved_retransmit_state.compress;
1310         s->session = frag->msg_header.saved_retransmit_state.session;
1311         s->d1->w_epoch = frag->msg_header.saved_retransmit_state.epoch;
1312
1313         if (frag->msg_header.saved_retransmit_state.epoch == saved_state.epoch -
1314         {
1315                 memcpy(save_write_sequence, s->s3->write_sequence, sizeof(s->s3-
```

```
1316                 memcpy(s->s3->write_sequence, s->d1->last_write_sequence, sizeof
1317         }
1318
1319         ret = dtls1_do_write(s, frag->msg_header.is_ccs ?
1320                                                 SSL3_RT_CHANGE_CIPHER_SPEC : SS
1321
1322         /* restore current state */
1323         s->enc_write_ctx = saved_state.enc_write_ctx;
1324         s->write_hash = saved_state.write_hash;
1325         s->compress = saved_state.compress;
1326         s->session = saved_state.session;
1327         s->d1->w_epoch = saved_state.epoch;
1328
1329         if (frag->msg_header.saved_retransmit_state.epoch == saved_state.epoch -
1330         {
1331                 memcpy(s->d1->last_write_sequence, s->s3->write_sequence, sizeof
1332                 memcpy(s->s3->write_sequence, save_write_sequence, sizeof(s->s3-
1333         }

1335         s->d1->retransmitting = 0;

1337         (void)BIO_flush(SSL_get_wbio(s));
1338         return ret;
1339         }

1341 /* call this function when the buffered messages are no longer needed */
1342 void
1343 dtls1_clear_record_buffer(SSL *s)
1344         {
1345         pitem *item;

1347         for(item = pqueue_pop(s->d1->sent_messages);
1348                 item != NULL; item = pqueue_pop(s->d1->sent_messages))
1349                 {
1350                 dtls1_hm_fragment_free((hm_fragment *)item->data);
1351                 pitem_free(item);
1352                 }
1353         }


1356 unsigned char *
1357 dtls1_set_message_header(SSL *s, unsigned char *p, unsigned char mt,
1358                         unsigned long len, unsigned long frag_off, unsigned long
1359         {
1360         /* Don't change sequence numbers while listening */
1361         if (frag_off == 0 && !s->d1->listen)
1362                 {
1363                 s->d1->handshake_write_seq = s->d1->next_handshake_write_seq;
1364                 s->d1->next_handshake_write_seq++;
1365                 }

1367         dtls1_set_message_header_int(s, mt, len, s->d1->handshake_write_seq,
1368                 frag_off, frag_len);

1370         return p += DTLS1_HM_HEADER_LENGTH;
1371         }


1374 /* don't actually do the writing, wait till the MTU has been retrieved */
1375 static void
1376 dtls1_set_message_header_int(SSL *s, unsigned char mt,
1377                              unsigned long len, unsigned short seq_num, unsigned
1378                              unsigned long frag_len)
1379         {
1380         struct hm_header_st *msg_hdr = &s->d1->w_msg_hdr;
```



```
1382         msg_hdr->type = mt;
1383         msg_hdr->msg_len = len;
1384         msg_hdr->seq = seq_num;
1385         msg_hdr->frag_off = frag_off;
1386         msg_hdr->frag_len = frag_len;
1387         }

1389 static void
1390 dtls1_fix_message_header(SSL *s, unsigned long frag_off,
1391                          unsigned long frag_len)
1392         {
1393         struct hm_header_st *msg_hdr = &s->d1->w_msg_hdr;

1395         msg_hdr->frag_off = frag_off;
1396         msg_hdr->frag_len = frag_len;
1397         }

1399 static unsigned char *
1400 dtls1_write_message_header(SSL *s, unsigned char *p)
1401         {
1402         struct hm_header_st *msg_hdr = &s->d1->w_msg_hdr;

1404         *p++ = msg_hdr->type;
1405         l2n3(msg_hdr->msg_len, p);

1407         s2n(msg_hdr->seq, p);
1408         l2n3(msg_hdr->frag_off, p);
1409         l2n3(msg_hdr->frag_len, p);

1411         return p;
1412         }

1414 unsigned int
1415 dtls1_min_mtu(void)
1416         {
1417         return (g_probable_mtu[(sizeof(g_probable_mtu) /
1418                 sizeof(g_probable_mtu[0])) - 1]);
1419         }

1421 static unsigned int
1422 dtls1_guess_mtu(unsigned int curr_mtu)
1423         {
1424         unsigned int i;

1426         if ( curr_mtu == 0 )
1427                 return g_probable_mtu[0] ;

1429         for ( i = 0; i < sizeof(g_probable_mtu)/sizeof(g_probable_mtu[0]); i++)
1430                 if ( curr_mtu > g_probable_mtu[i])
1431                         return g_probable_mtu[i];

1433         return curr_mtu;
1434         }

1436 void
1437 dtls1_get_message_header(unsigned char *data, struct hm_header_st *msg_hdr)
1438         {
1439         memset(msg_hdr, 0x00, sizeof(struct hm_header_st));
1440         msg_hdr->type = *(data++);
1441         n2l3(data, msg_hdr->msg_len);

1443         n2s(data, msg_hdr->seq);
1444         n2l3(data, msg_hdr->frag_off);
1445         n2l3(data, msg_hdr->frag_len);
1446         }
```

```
1448 void
1449 dtls1_get_ccs_header(unsigned char *data, struct ccs_header_st *ccs_hdr)
1450         {
1451         memset(ccs_hdr, 0x00, sizeof(struct ccs_header_st));

1453         ccs_hdr->type = *(data++);
1454         }

1456 int dtls1_shutdown(SSL *s)
1457         {
1458         int ret;
1459 #ifndef OPENSSL_NO_SCTP
1460         if (BIO_dgram_is_sctp(SSL_get_wbio(s)) &&
1461             !(s->shutdown & SSL_SENT_SHUTDOWN))
1462                 {
1463                 ret = BIO_dgram_sctp_wait_for_dry(SSL_get_wbio(s));
1464                 if (ret < 0) return -1;

1466                 if (ret == 0)
1467                         BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_SAVE_SHUTD
1468                 }
1469 #endif
1470         ret = ssl3_shutdown(s);
1471 #ifndef OPENSSL_NO_SCTP
1472         BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_SAVE_SHUTDOWN, 0, NULL);
1473 #endif
1474         return ret;
1475         }

1477 #ifndef OPENSSL_NO_HEARTBEATS
1478 int
1479 dtls1_process_heartbeat(SSL *s)
1480         {
1481         unsigned char *p = &s->s3->rrec.data[0], *pl;
1482         unsigned short hbtype;
1483         unsigned int payload;
1484         unsigned int padding = 16; /* Use minimum padding */

1486         if (s->msg_callback)
1487                 s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
1488                         &s->s3->rrec.data[0], s->s3->rrec.length,
1489                         s, s->msg_callback_arg);

1491         /* Read type and payload length first */
1492         if (1 + 2 + 16 > s->s3->rrec.length)
1493                 return 0; /* silently discard */
1494         hbtype = *p++;
1495         n2s(p, payload);
1496         if (1 + 2 + payload + 16 > s->s3->rrec.length)
1497                 return 0; /* silently discard per RFC 6520 sec. 4 */
1498         pl = p;

1500         if (hbtype == TLS1_HB_REQUEST)
1501                 {
1502                 unsigned char *buffer, *bp;
1503                 unsigned int write_length = 1 /* heartbeat type */ +
1504                                             2 /* heartbeat length */ +
1505                                             payload + padding;
1506                 int r;

1508                 if (write_length > SSL3_RT_MAX_PLAIN_LENGTH)
1509                         return 0;

1511                 /* Allocate memory for the response, size is 1 byte
1512                  * message type, plus 2 bytes payload length, plus
1513                  * payload, plus padding
```



```
1514                  */
1515                 buffer = OPENSSL_malloc(write_length);
1516                 bp = buffer;

1518                 /* Enter response type, length and copy payload */
1519                 *bp++ = TLS1_HB_RESPONSE;
1520                 s2n(payload, bp);
1521                 memcpy(bp, pl, payload);
1522                 bp += payload;
1523                 /* Random padding */
1524                 RAND_pseudo_bytes(bp, padding);

1526                 r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, write_length

1528                 if (r >= 0 && s->msg_callback)
1529                         s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
1530                                 buffer, write_length,
1531                                 s, s->msg_callback_arg);

1533                 OPENSSL_free(buffer);

1535                 if (r < 0)
1536                         return r;
1537                 }
1538         else if (hbtype == TLS1_HB_RESPONSE)
1539                 {
1540                 unsigned int seq;

1542                 /* We only send sequence numbers (2 bytes unsigned int),
1543                  * and 16 random bytes, so we just try to read the
1544                  * sequence number */
1545                 n2s(pl, seq);

1547                 if (payload == 18 && seq == s->tlsext_hb_seq)
1548                         {
1549                         dtls1_stop_timer(s);
1550                         s->tlsext_hb_seq++;
1551                         s->tlsext_hb_pending = 0;
1552                         }
1553                 }

1555         return 0;
1556         }

1558 int
1559 dtls1_heartbeat(SSL *s)
1560         {
1561         unsigned char *buf, *p;
1562         int ret;
1563         unsigned int payload = 18; /* Sequence number + random bytes */
1564         unsigned int padding = 16; /* Use minimum padding */

1566         /* Only send if peer supports and accepts HB requests... */
1567         if (!(s->tlsext_heartbeat & SSL_TLSEXT_HB_ENABLED) ||
1568             s->tlsext_heartbeat & SSL_TLSEXT_HB_DONT_SEND_REQUESTS)
1569                 {
1570                 SSLerr(SSL_F_DTLS1_HEARTBEAT,SSL_R_TLS_HEARTBEAT_PEER_DOESNT_ACC
1571                 return -1;
1572                 }

1574         /* ...and there is none in flight yet... */
1575         if (s->tlsext_hb_pending)
1576                 {
1577                 SSLerr(SSL_F_DTLS1_HEARTBEAT,SSL_R_TLS_HEARTBEAT_PENDING);
1578                 return -1;
1579                 }
```

```
1581         /* ...and no handshake in progress. */
1582         if (SSL_in_init(s) || s->in_handshake)
1583                 {
1584                 SSLerr(SSL_F_DTLS1_HEARTBEAT,SSL_R_UNEXPECTED_MESSAGE);
1585                 return -1;
1586                 }

1588         /* Check if padding is too long, payload and padding
1589          * must not exceed 2^14 - 3 = 16381 bytes in total.
1590          */
1591         OPENSSL_assert(payload + padding <= 16381);

1593         /* Create HeartBeat message, we just use a sequence number
1594          * as payload to distuingish different messages and add
1595          * some random stuff.
1596          *  - Message Type, 1 byte
1597          *  - Payload Length, 2 bytes (unsigned int)
1598          *  - Payload, the sequence number (2 bytes uint)
1599          *  - Payload, random bytes (16 bytes uint)
1600          *  - Padding
1601          */
1602         buf = OPENSSL_malloc(1 + 2 + payload + padding);
1603         p = buf;
1604         /* Message Type */
1605         *p++ = TLS1_HB_REQUEST;
1606         /* Payload length (18 bytes here) */
1607         s2n(payload, p);
1608         /* Sequence number */
1609         s2n(s->tlsext_hb_seq, p);
1610         /* 16 random bytes */
1611         RAND_pseudo_bytes(p, 16);
1612         p += 16;
1613         /* Random padding */
1614         RAND_pseudo_bytes(p, padding);

1616         ret = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buf, 3 + payload + padding
1617         if (ret >= 0)
1618                 {
1619                 if (s->msg_callback)
1620                         s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
1621                                 buf, 3 + payload + padding,
1622                                 s, s->msg_callback_arg);

1624                 dtls1_start_timer(s);
1625                 s->tlsext_hb_pending = 1;
1626                 }

1628         OPENSSL_free(buf);

1630         return ret;
1631         }
1632 #endif
1633 #endif /* ! codereview */
```

```
**********************************************************
   45387 Wed Aug 13 19:53:34 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_clnt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/d1_clnt.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
```



```
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "ssl_locl.h"
#ifndef OPENSSL_NO_KRB5
#include "kssl_lcl.h"
#endif
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/md5.h>
#include <openssl/bn.h>
#ifndef OPENSSL_NO_DH
```

```
#include <openssl/dh.h>
#endif

static const SSL_METHOD *dtls1_get_client_method(int ver);
static int dtls1_get_hello_verify(SSL *s);

static const SSL_METHOD *dtls1_get_client_method(int ver)
        {
        if (ver == DTLS1_VERSION || ver == DTLS1_BAD_VER)
                return(DTLSv1_client_method());
        else
                return(NULL);
        }

IMPLEMENT_dtls1_meth_func(DTLSv1_client_method,
                        ssl_undefined_function,
                        dtls1_connect,
                        dtls1_get_client_method)

int dtls1_connect(SSL *s)
        {
        BUF_MEM *buf=NULL;
        unsigned long Time=(unsigned long)time(NULL);
        void (*cb)(const SSL *ssl,int type,int val)=NULL;
        int ret= -1;
        int new_state,state,skip=0;
#ifndef OPENSSL_NO_SCTP
        unsigned char sctpauthkey[64];
        char labelbuffer[sizeof(DTLS1_SCTP_AUTH_LABEL)];
#endif

        RAND_add(&Time,sizeof(Time),0);
        ERR_clear_error();
        clear_sys_error();

        if (s->info_callback != NULL)
                cb=s->info_callback;
        else if (s->ctx->info_callback != NULL)
                cb=s->ctx->info_callback;

        s->in_handshake++;
        if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

#ifndef OPENSSL_NO_SCTP
        /* Notify SCTP BIO socket to enter handshake
         * mode and prevent stream identifier other
         * than 0. Will be ignored if no SCTP is used.
         */
        BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_SET_IN_HANDSHAKE, s->in_ha
#endif

#ifndef OPENSSL_NO_HEARTBEATS
        /* If we're awaiting a HeartbeatResponse, pretend we
         * already got and don't await it anymore, because
         * Heartbeats don't make sense during handshakes anyway.
         */
        if (s->tlsext_hb_pending)
                {
                dtls1_stop_timer(s);
                s->tlsext_hb_pending = 0;
                s->tlsext_hb_seq++;
                }
#endif

        for (;;)
                {
```



```
                state=s->state;

                switch(s->state)
                        {
                case SSL_ST_RENEGOTIATE:
                        s->renegotiate=1;
                        s->state=SSL_ST_CONNECT;
                        s->ctx->stats.sess_connect_renegotiate++;
                        /* break */
                case SSL_ST_BEFORE:
                case SSL_ST_CONNECT:
                case SSL_ST_BEFORE|SSL_ST_CONNECT:
                case SSL_ST_OK|SSL_ST_CONNECT:

                        s->server=0;
                        if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

                        if ((s->version & 0xff00 ) != (DTLS1_VERSION & 0xff00) &
                            (s->version & 0xff00 ) != (DTLS1_BAD_VER & 0xff00))
                                {
                                SSLerr(SSL_F_DTLS1_CONNECT, ERR_R_INTERNAL_ERROR
                                ret = -1;
                                goto end;
                                }

                        /* s->version=SSL3_VERSION; */
                        s->type=SSL_ST_CONNECT;

                        if (s->init_buf == NULL)
                                {
                                if ((buf=BUF_MEM_new()) == NULL)
                                        {
                                        ret= -1;
                                        goto end;
                                        }
                                if (!BUF_MEM_grow(buf,SSL3_RT_MAX_PLAIN_LENGTH))
                                        {
                                        ret= -1;
                                        goto end;
                                        }
                                s->init_buf=buf;
                                buf=NULL;
                                }

                        if (!ssl3_setup_buffers(s)) { ret= -1; goto end; }

                        /* setup buffing BIO */
                        if (!ssl_init_wbio_buffer(s,0)) { ret= -1; goto end; }

                        /* don't push the buffering BIO quite yet */

                        s->state=SSL3_ST_CW_CLNT_HELLO_A;
                        s->ctx->stats.sess_connect++;
                        s->init_num=0;
                        /* mark client_random uninitialized */
                        memset(s->s3->client_random,0,sizeof(s->s3->client_rando
                        s->d1->send_cookie = 0;
                        s->hit = 0;
                        break;

#ifndef OPENSSL_NO_SCTP
                case DTLS1_SCTP_ST_CR_READ_SOCK:

                        if (BIO_dgram_sctp_msg_waiting(SSL_get_rbio(s)))
                        {
                                s->s3->in_read_app_data=2;
```

```
260                                 s->rwstate=SSL_READING;
261                                 BIO_clear_retry_flags(SSL_get_rbio(s));
262                                 BIO_set_retry_read(SSL_get_rbio(s));
263                                 ret = -1;
264                                 goto end;
265                         }
267                         s->state=s->s3->tmp.next_state;
268                         break;
270                 case DTLS1_SCTP_ST_CW_WRITE_SOCK:
271                         /* read app data until dry event */
273                         ret = BIO_dgram_sctp_wait_for_dry(SSL_get_wbio(s));
274                         if (ret < 0) goto end;
276                         if (ret == 0)
277                         {
278                                 s->s3->in_read_app_data=2;
279                                 s->rwstate=SSL_READING;
280                                 BIO_clear_retry_flags(SSL_get_rbio(s));
281                                 BIO_set_retry_read(SSL_get_rbio(s));
282                                 ret = -1;
283                                 goto end;
284                         }
286                         s->state=s->d1->next_state;
287                         break;
288 #endif
290                 case SSL3_ST_CW_CLNT_HELLO_A:
291                 case SSL3_ST_CW_CLNT_HELLO_B:
293                         s->shutdown=0;
295                         /* every DTLS ClientHello resets Finished MAC */
296                         ssl3_init_finished_mac(s);
298                         dtls1_start_timer(s);
299                         ret=dtls1_client_hello(s);
300                         if (ret <= 0) goto end;
302                         if ( s->d1->send_cookie)
303                                 {
304                                 s->state=SSL3_ST_CW_FLUSH;
305                                 s->s3->tmp.next_state=SSL3_ST_CR_SRVR_HELLO_A;
306                                 }
307                         else
308                                 s->state=SSL3_ST_CR_SRVR_HELLO_A;
310                         s->init_num=0;
312 #ifndef OPENSSL_NO_SCTP
313                         /* Disable buffering for SCTP */
314                         if (!BIO_dgram_is_sctp(SSL_get_wbio(s)))
315                                 {
316 #endif
317                                 /* turn on buffering for the next lot of output
318                                 if (s->bbio != s->wbio)
319                                         s->wbio=BIO_push(s->bbio,s->wbio);
320 #ifndef OPENSSL_NO_SCTP
321                                 }
322 #endif
324                         break;
```



```
326                 case SSL3_ST_CR_SRVR_HELLO_A:
327                 case SSL3_ST_CR_SRVR_HELLO_B:
328                         ret=ssl3_get_server_hello(s);
329                         if (ret <= 0) goto end;
330                         else
331                                 {
332                                 if (s->hit)
333                                         {
334 #ifndef OPENSSL_NO_SCTP
335                                         /* Add new shared key for SCTP-Auth,
336                                          * will be ignored if no SCTP used.
337                                          */
338                                         snprintf((char*) labelbuffer, sizeof(DTL
339                                                 DTLS1_SCTP_AUTH_LABEL);
341                                         SSL_export_keying_material(s, sctpauthke
342                                                                    sizeof(sctpau
343                                                                    sizeof(labelb
345                                         BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM
346                                                         sizeof(sctpauthkey), sc
347 #endif
349                                         s->state=SSL3_ST_CR_FINISHED_A;
350                                         }
351                                 else
352                                         s->state=DTLS1_ST_CR_HELLO_VERIFY_REQUES
353                                 }
354                         s->init_num=0;
355                         break;
357                 case DTLS1_ST_CR_HELLO_VERIFY_REQUEST_A:
358                 case DTLS1_ST_CR_HELLO_VERIFY_REQUEST_B:
360                         ret = dtls1_get_hello_verify(s);
361                         if ( ret <= 0)
362                                 goto end;
363                         dtls1_stop_timer(s);
364                         if ( s->d1->send_cookie) /* start again, with a cookie *
365                                 s->state=SSL3_ST_CW_CLNT_HELLO_A;
366                         else
367                                 s->state = SSL3_ST_CR_CERT_A;
368                         s->init_num = 0;
369                         break;
371                 case SSL3_ST_CR_CERT_A:
372                 case SSL3_ST_CR_CERT_B:
373 #ifndef OPENSSL_NO_TLSEXT
374                         ret=ssl3_check_finished(s);
375                         if (ret <= 0) goto end;
376                         if (ret == 2)
377                                 {
378                                 s->hit = 1;
379                                 if (s->tlsext_ticket_expected)
380                                         s->state=SSL3_ST_CR_SESSION_TICKET_A;
381                                 else
382                                         s->state=SSL3_ST_CR_FINISHED_A;
383                                 s->init_num=0;
384                                 break;
385                                 }
386 #endif
387                         /* Check if it is anon DH or PSK */
388                         if (!(s->s3->tmp.new_cipher->algorithm_auth & SSL_aNULL)
389                             !(s->s3->tmp.new_cipher->algorithm_mkey & SSL_kPSK))
390                                 {
391                                 ret=ssl3_get_server_certificate(s);
```

```
392                                 if (ret <= 0) goto end;
393 #ifndef OPENSSL_NO_TLSEXT
394                                 if (s->tlsext_status_expected)
395                                         s->state=SSL3_ST_CR_CERT_STATUS_A;
396                                 else
397                                         s->state=SSL3_ST_CR_KEY_EXCH_A;
398                                 }
399                         else
400                                 {
401                                 skip = 1;
402                                 s->state=SSL3_ST_CR_KEY_EXCH_A;
403                                 }
404 #else
405                                 }
406                         else
407                                 skip=1;

409                         s->state=SSL3_ST_CR_KEY_EXCH_A;
410 #endif
411                         s->init_num=0;
412                         break;

414                 case SSL3_ST_CR_KEY_EXCH_A:
415                 case SSL3_ST_CR_KEY_EXCH_B:
416                         ret=ssl3_get_key_exchange(s);
417                         if (ret <= 0) goto end;
418                         s->state=SSL3_ST_CR_CERT_REQ_A;
419                         s->init_num=0;

421                         /* at this point we check that we have the
422                          * required stuff from the server */
423                         if (!ssl3_check_cert_and_algorithm(s))
424                                 {
425                                 ret= -1;
426                                 goto end;
427                                 }
428                         break;

430                 case SSL3_ST_CR_CERT_REQ_A:
431                 case SSL3_ST_CR_CERT_REQ_B:
432                         ret=ssl3_get_certificate_request(s);
433                         if (ret <= 0) goto end;
434                         s->state=SSL3_ST_CR_SRVR_DONE_A;
435                         s->init_num=0;
436                         break;

438                 case SSL3_ST_CR_SRVR_DONE_A:
439                 case SSL3_ST_CR_SRVR_DONE_B:
440                         ret=ssl3_get_server_done(s);
441                         if (ret <= 0) goto end;
442                         dtls1_stop_timer(s);
443                         if (s->s3->tmp.cert_req)
444                                 s->s3->tmp.next_state=SSL3_ST_CW_CERT_A;
445                         else
446                                 s->s3->tmp.next_state=SSL3_ST_CW_KEY_EXCH_A;
447                         s->init_num=0;

449 #ifndef OPENSSL_NO_SCTP
450                         if (BIO_dgram_is_sctp(SSL_get_wbio(s)) &&
451                             state == SSL_ST_RENEGOTIATE)
452                                 s->state=DTLS1_SCTP_ST_CR_READ_SOCK;
453                         else
454 #endif
455                         s->state=s->s3->tmp.next_state;
456                         break;
```



```
458                 case SSL3_ST_CW_CERT_A:
459                 case SSL3_ST_CW_CERT_B:
460                 case SSL3_ST_CW_CERT_C:
461                 case SSL3_ST_CW_CERT_D:
462                         dtls1_start_timer(s);
463                         ret=dtls1_send_client_certificate(s);
464                         if (ret <= 0) goto end;
465                         s->state=SSL3_ST_CW_KEY_EXCH_A;
466                         s->init_num=0;
467                         break;

469                 case SSL3_ST_CW_KEY_EXCH_A:
470                 case SSL3_ST_CW_KEY_EXCH_B:
471                         dtls1_start_timer(s);
472                         ret=dtls1_send_client_key_exchange(s);
473                         if (ret <= 0) goto end;

475 #ifndef OPENSSL_NO_SCTP
476                         /* Add new shared key for SCTP-Auth,
477                          * will be ignored if no SCTP used.
478                          */
479                         snprintf((char*) labelbuffer, sizeof(DTLS1_SCTP_AUTH_LAB
480                                 DTLS1_SCTP_AUTH_LABEL);

482                         SSL_export_keying_material(s, sctpauthkey,
483                                                    sizeof(sctpauthkey), labelbuf
484                                                    sizeof(labelbuffer), NULL, 0,

486                         BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_ADD_AUTH_K
487                                  sizeof(sctpauthkey), sctpauthkey);
488 #endif

490                         /* EAY EAY EAY need to check for DH fix cert
491                          * sent back */
492                         /* For TLS, cert_req is set to 2, so a cert chain
493                          * of nothing is sent, but no verify packet is sent */
494                         if (s->s3->tmp.cert_req == 1)
495                                 {
496                                 s->state=SSL3_ST_CW_CERT_VRFY_A;
497                                 }
498                         else
499                                 {
500 #ifndef OPENSSL_NO_SCTP
501                                 if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
502                                         {
503                                         s->d1->next_state=SSL3_ST_CW_CHANGE_A;
504                                         s->state=DTLS1_SCTP_ST_CW_WRITE_SOCK;
505                                         }
506                                 else
507 #endif
508                                         s->state=SSL3_ST_CW_CHANGE_A;
509                                 s->s3->change_cipher_spec=0;
510                                 }

512                         s->init_num=0;
513                         break;

515                 case SSL3_ST_CW_CERT_VRFY_A:
516                 case SSL3_ST_CW_CERT_VRFY_B:
517                         dtls1_start_timer(s);
518                         ret=dtls1_send_client_verify(s);
519                         if (ret <= 0) goto end;
520 #ifndef OPENSSL_NO_SCTP
521                         if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
522                         {
523                                 s->d1->next_state=SSL3_ST_CW_CHANGE_A;
```

```
524                             s->state=DTLS1_SCTP_ST_CW_WRITE_SOCK;
525                     }
526                     else
527 #endif
528                             s->state=SSL3_ST_CW_CHANGE_A;
529                     s->init_num=0;
530                     s->s3->change_cipher_spec=0;
531                     break;

533             case SSL3_ST_CW_CHANGE_A:
534             case SSL3_ST_CW_CHANGE_B:
535                     if (!s->hit)
536                             dtls1_start_timer(s);
537                     ret=dtls1_send_change_cipher_spec(s,
538                             SSL3_ST_CW_CHANGE_A,SSL3_ST_CW_CHANGE_B);
539                     if (ret <= 0) goto end;

541                     s->state=SSL3_ST_CW_FINISHED_A;
542                     s->init_num=0;

544                     s->session->cipher=s->s3->tmp.new_cipher;
545 #ifdef OPENSSL_NO_COMP
546                     s->session->compress_meth=0;
547 #else
548                     if (s->s3->tmp.new_compression == NULL)
549                             s->session->compress_meth=0;
550                     else
551                             s->session->compress_meth=
552                                     s->s3->tmp.new_compression->id;
553 #endif
554                     if (!s->method->ssl3_enc->setup_key_block(s))
555                             {
556                             ret= -1;
557                             goto end;
558                             }

560                     if (!s->method->ssl3_enc->change_cipher_state(s,
561                             SSL3_CHANGE_CIPHER_CLIENT_WRITE))
562                             {
563                             ret= -1;
564                             goto end;
565                             }

567 #ifndef OPENSSL_NO_SCTP
568                             if (s->hit)
569                                     {
570                                     /* Change to new shared key of SCTP-Auth
571                                      * will be ignored if no SCTP used.
572                                      */
573                                     BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM
574                                     }
575 #endif

577                     dtls1_reset_seq_numbers(s, SSL3_CC_WRITE);
578                     break;

580             case SSL3_ST_CW_FINISHED_A:
581             case SSL3_ST_CW_FINISHED_B:
582                     if (!s->hit)
583                             dtls1_start_timer(s);
584                     ret=dtls1_send_finished(s,
585                             SSL3_ST_CW_FINISHED_A,SSL3_ST_CW_FINISHED_B,
586                             s->method->ssl3_enc->client_finished_label,
587                             s->method->ssl3_enc->client_finished_label_len);
588                     if (ret <= 0) goto end;
589                     s->state=SSL3_ST_CW_FLUSH;
```



```
591                     /* clear flags */
592                     s->s3->flags&= ~SSL3_FLAGS_POP_BUFFER;
593                     if (s->hit)
594                             {
595                             s->s3->tmp.next_state=SSL_ST_OK;
596 #ifndef OPENSSL_NO_SCTP
597                             if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
598                                     {
599                                             s->d1->next_state = s->s3->tmp.n
600                                             s->s3->tmp.next_state=DTLS1_SCTP
601                                     }
602 #endif
603                             if (s->s3->flags & SSL3_FLAGS_DELAY_CLIENT_FINIS
604                                     {
605                                     s->state=SSL_ST_OK;
606 #ifndef OPENSSL_NO_SCTP
607                                     if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
608                                             {
609                                                     s->d1->next_state = SSL_
610                                                     s->state=DTLS1_SCTP_ST_C
611                                             }
612 #endif
613                                     s->s3->flags|=SSL3_FLAGS_POP_BUFFER;
614                                     s->s3->delay_buf_pop_ret=0;
615                                     }
616                             }
617                     else
618                             {
619 #ifndef OPENSSL_NO_SCTP
620                             /* Change to new shared key of SCTP-Auth,
621                              * will be ignored if no SCTP used.
622                              */
623                             BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_NE
624 #endif

626 #ifndef OPENSSL_NO_TLSEXT
627                             /* Allow NewSessionTicket if ticket expected */
628                             if (s->tlsext_ticket_expected)
629                                     s->s3->tmp.next_state=SSL3_ST_CR_SESSION
630                             else
631 #endif

633                             s->s3->tmp.next_state=SSL3_ST_CR_FINISHED_A;
634                             }
635                     s->init_num=0;
636                     break;

638 #ifndef OPENSSL_NO_TLSEXT
639             case SSL3_ST_CR_SESSION_TICKET_A:
640             case SSL3_ST_CR_SESSION_TICKET_B:
641                     ret=ssl3_get_new_session_ticket(s);
642                     if (ret <= 0) goto end;
643                     s->state=SSL3_ST_CR_FINISHED_A;
644                     s->init_num=0;
645             break;

647             case SSL3_ST_CR_CERT_STATUS_A:
648             case SSL3_ST_CR_CERT_STATUS_B:
649                     ret=ssl3_get_cert_status(s);
650                     if (ret <= 0) goto end;
651                     s->state=SSL3_ST_CR_KEY_EXCH_A;
652                     s->init_num=0;
653             break;
654 #endif
```

```
656                 case SSL3_ST_CR_FINISHED_A:
657                 case SSL3_ST_CR_FINISHED_B:
658                         s->d1->change_cipher_spec_ok = 1;
659                         ret=ssl3_get_finished(s,SSL3_ST_CR_FINISHED_A,
660                                 SSL3_ST_CR_FINISHED_B);
661                         if (ret <= 0) goto end;
662                         dtls1_stop_timer(s);
663
664                         if (s->hit)
665                                 s->state=SSL3_ST_CW_CHANGE_A;
666                         else
667                                 s->state=SSL_ST_OK;
668
669 #ifndef OPENSSL_NO_SCTP
670                         if (BIO_dgram_is_sctp(SSL_get_wbio(s)) &&
671                                 state == SSL_ST_RENEGOTIATE)
672                                 {
673                                 s->d1->next_state=s->state;
674                                 s->state=DTLS1_SCTP_ST_CW_WRITE_SOCK;
675                                 }
676 #endif
677
678                         s->init_num=0;
679                         break;
680
681                 case SSL3_ST_CW_FLUSH:
682                         s->rwstate=SSL_WRITING;
683                         if (BIO_flush(s->wbio) <= 0)
684                                 {
685                                 /* If the write error was fatal, stop trying */
686                                 if (!BIO_should_retry(s->wbio))
687                                         {
688                                         s->rwstate=SSL_NOTHING;
689                                         s->state=s->s3->tmp.next_state;
690                                         }
691
692                                 ret= -1;
693                                 goto end;
694                                 }
695                         s->rwstate=SSL_NOTHING;
696                         s->state=s->s3->tmp.next_state;
697                         break;
698
699                 case SSL_ST_OK:
700                         /* clean a few things up */
701                         ssl3_cleanup_key_block(s);
702
703 #if 0
704                         if (s->init_buf != NULL)
705                                 {
706                                 BUF_MEM_free(s->init_buf);
707                                 s->init_buf=NULL;
708                                 }
709 #endif
710
711                         /* If we are not 'joining' the last two packets,
712                          * remove the buffering now */
713                         if (!(s->s3->flags & SSL3_FLAGS_POP_BUFFER))
714                                 ssl_free_wbio_buffer(s);
715                         /* else do it later in ssl3_write */
716
717                         s->init_num=0;
718                         s->renegotiate=0;
719                         s->new_session=0;
720
721                         ssl_update_cache(s,SSL_SESS_CACHE_CLIENT);
```



```
722                         if (s->hit) s->ctx->stats.sess_hit++;
723
724                         ret=1;
725                         /* s->server=0; */
726                         s->handshake_func=dtls1_connect;
727                         s->ctx->stats.sess_connect_good++;
728
729                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_DONE,1);
730
731                         /* done with handshaking */
732                         s->d1->handshake_read_seq  = 0;
733                         s->d1->next_handshake_write_seq = 0;
734                         goto end;
735                         /* break; */
736
737                 default:
738                         SSLerr(SSL_F_DTLS1_CONNECT,SSL_R_UNKNOWN_STATE);
739                         ret= -1;
740                         goto end;
741                         /* break; */
742                         }
743
744                 /* did we do anything */
745                 if (!s->s3->tmp.reuse_message && !skip)
746                         {
747                         if (s->debug)
748                                 {
749                                 if ((ret=BIO_flush(s->wbio)) <= 0)
750                                         goto end;
751                                 }
752
753                         if ((cb != NULL) && (s->state != state))
754                                 {
755                                 new_state=s->state;
756                                 s->state=state;
757                                 cb(s,SSL_CB_CONNECT_LOOP,1);
758                                 s->state=new_state;
759                                 }
760                         }
761                 skip=0;
762                 }
763 end:
764         s->in_handshake--;
765
766 #ifndef OPENSSL_NO_SCTP
767         /* Notify SCTP BIO socket to leave handshake
768          * mode and allow stream identifier other
769          * than 0. Will be ignored if no SCTP is used.
770          */
771         BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_SET_IN_HANDSHAKE, s->in_ha
772 #endif
773
774         if (buf != NULL)
775                 BUF_MEM_free(buf);
776         if (cb != NULL)
777                 cb(s,SSL_CB_CONNECT_EXIT,ret);
778         return(ret);
779         }
780
781 int dtls1_client_hello(SSL *s)
782         {
783         unsigned char *buf;
784         unsigned char *p,*d;
785         unsigned int i,j;
786         unsigned long l;
787         SSL_COMP *comp;
```

```
789         buf=(unsigned char *)s->init_buf->data;
790         if (s->state == SSL3_ST_CW_CLNT_HELLO_A)
791                 {
792                 SSL_SESSION *sess = s->session;
793                 if ((s->session == NULL) ||
794                         (s->session->ssl_version != s->version) ||
795 #ifdef OPENSSL_NO_TLSEXT
796                         !sess->session_id_length ||
797 #else
798                         (!sess->session_id_length && !sess->tlsext_tick) ||
799 #endif
800                         (s->session->not_resumable))
801                         {
802                         if (!ssl_get_new_session(s,0))
803                                 goto err;
804                         }
805                 /* else use the pre-loaded session */

807                 p=s->s3->client_random;

809                 /* if client_random is initialized, reuse it, we are
810                  * required to use same upon reply to HelloVerify */
811                 for (i=0;p[i]==’\0’ && i<sizeof(s->s3->client_random);i++)
812                         ;
813                 if (i==sizeof(s->s3->client_random))
814                         ssl_fill_hello_random(s, 0, p,
815                                               sizeof(s->s3->client_random));

817                 /* Do the message type and length last */
818                 d=p= &(buf[DTLS1_HM_HEADER_LENGTH]);

820                 *(p++)=s->version>>8;
821                 *(p++)=s->version&0xff;
822                 s->client_version=s->version;

824                 /* Random stuff */
825                 memcpy(p,s->s3->client_random,SSL3_RANDOM_SIZE);
826                 p+=SSL3_RANDOM_SIZE;

828                 /* Session ID */
829                 if (s->new_session)
830                         i=0;
831                 else
832                         i=s->session->session_id_length;
833                 *(p++)=i;
834                 if (i != 0)
835                         {
836                         if (i > sizeof s->session->session_id)
837                                 {
838                                 SSLerr(SSL_F_DTLS1_CLIENT_HELLO, ERR_R_INTERNAL_
839                                 goto err;
840                                 }
841                         memcpy(p,s->session->session_id,i);
842                         p+=i;
843                         }

845                 /* cookie stuff */
846                 if ( s->d1->cookie_len > sizeof(s->d1->cookie))
847                         {
848                         SSLerr(SSL_F_DTLS1_CLIENT_HELLO, ERR_R_INTERNAL_ERROR);
849                         goto err;
850                         }
851                 *(p++) = s->d1->cookie_len;
852                 memcpy(p, s->d1->cookie, s->d1->cookie_len);
853                 p += s->d1->cookie_len;
```



```
855                 /* Ciphers supported */
856                 i=ssl_cipher_list_to_bytes(s,SSL_get_ciphers(s),&(p[2]),0);
857                 if (i == 0)
858                         {
859                         SSLerr(SSL_F_DTLS1_CLIENT_HELLO,SSL_R_NO_CIPHERS_AVAILAB
860                         goto err;
861                         }
862                 s2n(i,p);
863                 p+=i;

865                 /* COMPRESSION */
866                 if (s->ctx->comp_methods == NULL)
867                         j=0;
868                 else
869                         j=sk_SSL_COMP_num(s->ctx->comp_methods);
870                 *(p++)=1+j;
871                 for (i=0; i<j; i++)
872                         {
873                         comp=sk_SSL_COMP_value(s->ctx->comp_methods,i);
874                         *(p++)=comp->id;
875                         }
876                 *(p++)=0; /* Add the NULL method */

878 #ifndef OPENSSL_NO_TLSEXT
879                 /* TLS extensions*/
880                 if (ssl_prepare_clienthello_tlsext(s) <= 0)
881                         {
882                         SSLerr(SSL_F_DTLS1_CLIENT_HELLO,SSL_R_CLIENTHELLO_TLSEXT
883                         goto err;
884                         }
885                 if ((p = ssl_add_clienthello_tlsext(s, p, buf+SSL3_RT_MAX_PLAIN_
886                         {
887                         SSLerr(SSL_F_DTLS1_CLIENT_HELLO,ERR_R_INTERNAL_ERROR);
888                         goto err;
889                         }
890 #endif

892                 l=(p-d);
893                 d=buf;

895                 d = dtls1_set_message_header(s, d, SSL3_MT_CLIENT_HELLO, l, 0, l

897                 s->state=SSL3_ST_CW_CLNT_HELLO_B;
898                 /* number of bytes to write */
899                 s->init_num=p-buf;
900                 s->init_off=0;

902                 /* buffer the message to handle re-xmits */
903                 dtls1_buffer_message(s, 0);
904                 }

906         /* SSL3_ST_CW_CLNT_HELLO_B */
907         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
908 err:
909         return(-1);
910         }

912 static int dtls1_get_hello_verify(SSL *s)
913         {
914         int n, al, ok = 0;
915         unsigned char *data;
916         unsigned int cookie_len;

918         n=s->method->ssl_get_message(s,
919                 DTLS1_ST_CR_HELLO_VERIFY_REQUEST_A,
```

```
920                 DTLS1_ST_CR_HELLO_VERIFY_REQUEST_B,
921                 -1,
922                 s->max_cert_list,
923                 &ok);

925         if (!ok) return((int)n);

927         if (s->s3->tmp.message_type != DTLS1_MT_HELLO_VERIFY_REQUEST)
928                 {
929                 s->d1->send_cookie = 0;
930                 s->s3->tmp.reuse_message=1;
931                 return(1);
932                 }

934         data = (unsigned char *)s->init_msg;

936         if ((data[0] != (s->version>>8)) || (data[1] != (s->version&0xff)))
937                 {
938                 SSLerr(SSL_F_DTLS1_GET_HELLO_VERIFY,SSL_R_WRONG_SSL_VERSION);
939                 s->version=(s->version&0xff00)|data[1];
940                 al = SSL_AD_PROTOCOL_VERSION;
941                 goto f_err;
942                 }
943         data+=2;

945         cookie_len = *(data++);
946         if ( cookie_len > sizeof(s->d1->cookie))
947                 {
948                 al=SSL_AD_ILLEGAL_PARAMETER;
949                 goto f_err;
950                 }

952         memcpy(s->d1->cookie, data, cookie_len);
953         s->d1->cookie_len = cookie_len;

955         s->d1->send_cookie = 1;
956         return 1;

958 f_err:
959         ssl3_send_alert(s, SSL3_AL_FATAL, al);
960         return -1;
961         }

963 int dtls1_send_client_key_exchange(SSL *s)
964         {
965         unsigned char *p,*d;
966         int n;
967         unsigned long alg_k;
968 #ifndef OPENSSL_NO_RSA
969         unsigned char *q;
970         EVP_PKEY *pkey=NULL;
971 #endif
972 #ifndef OPENSSL_NO_KRB5
973         KSSL_ERR kssl_err;
974 #endif /* OPENSSL_NO_KRB5 */
975 #ifndef OPENSSL_NO_ECDH
976         EC_KEY *clnt_ecdh = NULL;
977         const EC_POINT *srvr_ecpoint = NULL;
978         EVP_PKEY *srvr_pub_pkey = NULL;
979         unsigned char *encodedPoint = NULL;
980         int encoded_pt_len = 0;
981         BN_CTX * bn_ctx = NULL;
982 #endif

984         if (s->state == SSL3_ST_CW_KEY_EXCH_A)
985                 {
```



```
986                 d=(unsigned char *)s->init_buf->data;
987                 p= &(d[DTLS1_HM_HEADER_LENGTH]);

989                 alg_k=s->s3->tmp.new_cipher->algorithm_mkey;

991                 /* Fool emacs indentation */
992                 if (0) {}
993 #ifndef OPENSSL_NO_RSA
994                 else if (alg_k & SSL_kRSA)
995                         {
996                         RSA *rsa;
997                         unsigned char tmp_buf[SSL_MAX_MASTER_KEY_LENGTH];

999                         if (s->session->sess_cert == NULL)
1000                                {
1001                                /* We should always have a server certificate wi
1002                                SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,ERR_
1003                                goto err;
1004                                }

1006                        if (s->session->sess_cert->peer_rsa_tmp != NULL)
1007                                rsa=s->session->sess_cert->peer_rsa_tmp;
1008                        else
1009                                {
1010                                pkey=X509_get_pubkey(s->session->sess_cert->peer
1011                                if ((pkey == NULL) ||
1012                                        (pkey->type != EVP_PKEY_RSA) ||
1013                                        (pkey->pkey.rsa == NULL))
1014                                        {
1015                                        SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHA
1016                                        goto err;
1017                                        }
1018                                rsa=pkey->pkey.rsa;
1019                                EVP_PKEY_free(pkey);
1020                                }

1022                        tmp_buf[0]=s->client_version>>8;
1023                        tmp_buf[1]=s->client_version&0xff;
1024                        if (RAND_bytes(&(tmp_buf[2]),sizeof tmp_buf-2) <= 0)
1025                                        goto err;

1027                        s->session->master_key_length=sizeof tmp_buf;

1029                        q=p;
1030                        /* Fix buf for TLS and [incidentally] DTLS */
1031                        if (s->version > SSL3_VERSION)
1032                                p+=2;
1033                        n=RSA_public_encrypt(sizeof tmp_buf,
1034                                tmp_buf,p,rsa,RSA_PKCS1_PADDING);
1035 #ifdef PKCS1_CHECK
1036                        if (s->options & SSL_OP_PKCS1_CHECK_1) p[1]++;
1037                        if (s->options & SSL_OP_PKCS1_CHECK_2) tmp_buf[0]=0x70;
1038 #endif
1039                        if (n <= 0)
1040                                {
1041                                SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,SSL_
1042                                goto err;
1043                                }

1045                        /* Fix buf for TLS and [incidentally] DTLS */
1046                        if (s->version > SSL3_VERSION)
1047                                {
1048                                s2n(n,q);
1049                                n+=2;
1050                                }
```

```
1052                        s->session->master_key_length=
1053                                s->method->ssl3_enc->generate_master_secret(s,
1054                                        s->session->master_key,
1055                                        tmp_buf,sizeof tmp_buf);
1056                        OPENSSL_cleanse(tmp_buf,sizeof tmp_buf);
1057                        }
1058 #endif
1059 #ifndef OPENSSL_NO_KRB5
1060                 else if (alg_k & SSL_kKRB5)
1061                        {
1062                        krb5_error_code krb5rc;
1063                        KSSL_CTX       *kssl_ctx = s->kssl_ctx;
1064                        /*  krb5_data   krb5_ap_req;  */
1065                        krb5_data       *enc_ticket;
1066                        krb5_data       authenticator, *authp = NULL;
1067                        EVP_CIPHER_CTX  ciph_ctx;
1068                        const EVP_CIPHER *enc = NULL;
1069                        unsigned char   iv[EVP_MAX_IV_LENGTH];
1070                        unsigned char   tmp_buf[SSL_MAX_MASTER_KEY_LENGTH];
1071                        unsigned char   epms[SSL_MAX_MASTER_KEY_LENGTH
1072                                          + EVP_MAX_IV_LENGTH];
1073                        int             padl, outl = sizeof(epms);

1075                        EVP_CIPHER_CTX_init(&ciph_ctx);

1077 #ifdef KSSL_DEBUG
1078                        printf("ssl3_send_client_key_exchange(%lx & %lx)\n",
1079                                alg_k, SSL_kKRB5);
1080 #endif  /* KSSL_DEBUG */

1082                        authp = NULL;
1083 #ifdef KRB5SENDAUTH
1084                        if (KRB5SENDAUTH)  authp = &authenticator;
1085 #endif  /* KRB5SENDAUTH */

1087                        krb5rc = kssl_cget_tkt(kssl_ctx, &enc_ticket, authp,
1088                                &kssl_err);
1089                        enc = kssl_map_enc(kssl_ctx->enctype);
1090                        if (enc == NULL)
1091                            goto err;
1092 #ifdef KSSL_DEBUG
1093                        {
1094                        printf("kssl_cget_tkt rtn %d\n", krb5rc);
1095                        if (krb5rc && kssl_err.text)
1096                          printf("kssl_cget_tkt kssl_err=%s\n", kssl_err.text);
1097                        }
1098 #endif  /* KSSL_DEBUG */

1100                        if (krb5rc)
1101                                {
1102                                ssl3_send_alert(s,SSL3_AL_FATAL,
1103                                                SSL_AD_HANDSHAKE_FAILURE);
1104                                SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1105                                                kssl_err.reason);
1106                                goto err;
1107                                }

1109                        /*  20010406 VRS - Earlier versions used KRB5 AP_REQ
1110                        **  in place of RFC 2712 KerberosWrapper, as in:
1111                        **
1112                        **  Send ticket (copy to *p, set n = length)
1113                        **  n = krb5_ap_req.length;
1114                        **  memcpy(p, krb5_ap_req.data, krb5_ap_req.length);
1115                        **  if (krb5_ap_req.data)
1116                        **    kssl_krb5_free_data_contents(NULL,&krb5_ap_req);
1117                        **
```



```
1118                        **  Now using real RFC 2712 KerberosWrapper
1119                        **  (Thanks to Simon Wilkinson <sxw@sxw.org.uk>)
1120                        **  Note: 2712 "opaque" types are here replaced
1121                        **  with a 2-byte length followed by the value.
1122                        **  Example:
1123                        **  KerberosWrapper= xx xx asn1ticket 0 0 xx xx encpms
1124                        **  Where "xx xx" = length bytes.  Shown here with
1125                        **  optional authenticator omitted.
1126                        */

1128                        /*  KerberosWrapper.Ticket              */
1129                        s2n(enc_ticket->length,p);
1130                        memcpy(p, enc_ticket->data, enc_ticket->length);
1131                        p+= enc_ticket->length;
1132                        n = enc_ticket->length + 2;

1134                        /*  KerberosWrapper.Authenticator       */
1135                        if (authp  &&  authp->length)
1136                                {
1137                                s2n(authp->length,p);
1138                                memcpy(p, authp->data, authp->length);
1139                                p+= authp->length;
1140                                n+= authp->length + 2;

1142                                free(authp->data);
1143                                authp->data = NULL;
1144                                authp->length = 0;
1145                                }
1146                        else
1147                                {
1148                                s2n(0,p);/*  null authenticator length  */
1149                                n+=2;
1150                                }

1152                        if (RAND_bytes(tmp_buf,sizeof tmp_buf) <= 0)
1153                            goto err;

1155                        /*  20010420 VRS.  Tried it this way; failed.
1156                        **      EVP_EncryptInit_ex(&ciph_ctx,enc, NULL,NULL);
1157                        **      EVP_CIPHER_CTX_set_key_length(&ciph_ctx,
1158                        **                              kssl_ctx->length);
1159                        **      EVP_EncryptInit_ex(&ciph_ctx,NULL, key,iv);
1160                        */

1162                        memset(iv, 0, sizeof iv);  /* per RFC 1510 */
1163                        EVP_EncryptInit_ex(&ciph_ctx,enc, NULL,
1164                                kssl_ctx->key,iv);
1165                        EVP_EncryptUpdate(&ciph_ctx,epms,&outl,tmp_buf,
1166                                sizeof tmp_buf);
1167                        EVP_EncryptFinal_ex(&ciph_ctx,&(epms[outl]),&padl);
1168                        outl += padl;
1169                        if (outl > (int)sizeof epms)
1170                                {
1171                                SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE, ERR
1172                                goto err;
1173                                }
1174                        EVP_CIPHER_CTX_cleanup(&ciph_ctx);

1176                        /*  KerberosWrapper.EncryptedPreMasterSecret    */
1177                        s2n(outl,p);
1178                        memcpy(p, epms, outl);
1179                        p+=outl;
1180                        n+=outl + 2;

1182                        s->session->master_key_length=
1183                                s->method->ssl3_enc->generate_master_secret(s,
```

```
1184                                    s->session->master_key,
1185                                    tmp_buf, sizeof tmp_buf);

1187                        OPENSSL_cleanse(tmp_buf, sizeof tmp_buf);
1188                        OPENSSL_cleanse(epms, outl);
1189                        }
1190 #endif
1191 #ifndef OPENSSL_NO_DH
1192                 else if (alg_k & (SSL_kEDH|SSL_kDHr|SSL_kDHd))
1193                         {
1194                         DH *dh_srvr,*dh_clnt;

1196                         if (s->session->sess_cert == NULL)
1197                                 {
1198                                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_UNEXPECTE
1199                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,SSL_
1200                                 goto err;
1201                                 }

1203                         if (s->session->sess_cert->peer_dh_tmp != NULL)
1204                                 dh_srvr=s->session->sess_cert->peer_dh_tmp;
1205                         else
1206                                 {
1207                                 /* we get them from the cert */
1208                                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_HANDSHAKE
1209                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,SSL_
1210                                 goto err;
1211                                 }

1213                         /* generate a new random key */
1214                         if ((dh_clnt=DHparams_dup(dh_srvr)) == NULL)
1215                                 {
1216                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,ERR_
1217                                 goto err;
1218                                 }
1219                         if (!DH_generate_key(dh_clnt))
1220                                 {
1221                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,ERR_
1222                                 goto err;
1223                                 }

1225                         /* use the 'p' output buffer for the DH key, but
1226                          * make sure to clear it out afterwards */

1228                         n=DH_compute_key(p,dh_srvr->pub_key,dh_clnt);

1230                         if (n <= 0)
1231                                 {
1232                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,ERR_
1233                                 goto err;
1234                                 }

1236                         /* generate master key from the result */
1237                         s->session->master_key_length=
1238                                 s->method->ssl3_enc->generate_master_secret(s,
1239                                         s->session->master_key,p,n);
1240                         /* clean up */
1241                         memset(p,0,n);

1243                         /* send off the data */
1244                         n=BN_num_bytes(dh_clnt->pub_key);
1245                         s2n(n,p);
1246                         BN_bn2bin(dh_clnt->pub_key,p);
1247                         n+=2;

1249                         DH_free(dh_clnt);
```



```
1251                         /* perhaps clean things up a bit EAY EAY EAY EAY*/
1252                         }
1253 #endif
1254 #ifndef OPENSSL_NO_ECDH
1255                 else if (alg_k & (SSL_kEECDH|SSL_kECDHr|SSL_kECDHe))
1256                         {
1257                         const EC_GROUP *srvr_group = NULL;
1258                         EC_KEY *tkey;
1259                         int ecdh_clnt_cert = 0;
1260                         int field_size = 0;

1262                         if (s->session->sess_cert == NULL)
1263                                 {
1264                                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_UNEXPECTE
1265                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,SSL_
1266                                 goto err;
1267                                 }

1269                         /* Did we send out the client's
1270                          * ECDH share for use in premaster
1271                          * computation as part of client certificate?
1272                          * If so, set ecdh_clnt_cert to 1.
1273                          */
1274                         if ((alg_k & (SSL_kECDHr|SSL_kECDHe)) && (s->cert != NUL
1275                                 {
1276                                 /* XXX: For now, we do not support client
1277                                  * authentication using ECDH certificates.
1278                                  * To add such support, one needs to add
1279                                  * code that checks for appropriate
1280                                  * conditions and sets ecdh_clnt_cert to 1.
1281                                  * For example, the cert have an ECC
1282                                  * key on the same curve as the server's
1283                                  * and the key should be authorized for
1284                                  * key agreement.
1285                                  *
1286                                  * One also needs to add code in ssl3_connect
1287                                  * to skip sending the certificate verify
1288                                  * message.
1289                                  *
1290                                  * if ((s->cert->key->privatekey != NULL) &&
1291                                  *     (s->cert->key->privatekey->type ==
1292                                  *      EVP_PKEY_EC) && ...)
1293                                  * ecdh_clnt_cert = 1;
1294                                  */
1295                                 }

1297                         if (s->session->sess_cert->peer_ecdh_tmp != NULL)
1298                                 {
1299                                 tkey = s->session->sess_cert->peer_ecdh_tmp;
1300                                 }
1301                         else
1302                                 {
1303                                 /* Get the Server Public Key from Cert */
1304                                 srvr_pub_pkey = X509_get_pubkey(s->session-> \
1305                                     sess_cert->peer_pkeys[SSL_PKEY_ECC].x509);
1306                                 if ((srvr_pub_pkey == NULL) ||
1307                                     (srvr_pub_pkey->type != EVP_PKEY_EC) ||
1308                                     (srvr_pub_pkey->pkey.ec == NULL))
1309                                         {
1310                                         SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHA
1311                                             ERR_R_INTERNAL_ERROR);
1312                                         goto err;
1313                                         }

1315                                 tkey = srvr_pub_pkey->pkey.ec;
```

```
1316                                    }
1318                            srvr_group   = EC_KEY_get0_group(tkey);
1319                            srvr_ecpoint = EC_KEY_get0_public_key(tkey);
1321                            if ((srvr_group == NULL) || (srvr_ecpoint == NULL))
1322                                    {
1323                                    SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1324                                        ERR_R_INTERNAL_ERROR);
1325                                    goto err;
1326                                    }
1328                            if ((clnt_ecdh=EC_KEY_new()) == NULL)
1329                                    {
1330                                    SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,ERR_
1331                                    goto err;
1332                                    }
1334                            if (!EC_KEY_set_group(clnt_ecdh, srvr_group))
1335                                    {
1336                                    SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,ERR_
1337                                    goto err;
1338                                    }
1339                            if (ecdh_clnt_cert)
1340                                    {
1341                                    /* Reuse key info from our certificate
1342                                     * We only need our private key to perform
1343                                     * the ECDH computation.
1344                                     */
1345                                    const BIGNUM *priv_key;
1346                                    tkey = s->cert->key->privatekey->pkey.ec;
1347                                    priv_key = EC_KEY_get0_private_key(tkey);
1348                                    if (priv_key == NULL)
1349                                            {
1350                                            SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHA
1351                                            goto err;
1352                                            }
1353                                    if (!EC_KEY_set_private_key(clnt_ecdh, priv_key)
1354                                            {
1355                                            SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHA
1356                                            goto err;
1357                                            }
1358                                    }
1359                            else
1360                                    {
1361                                    /* Generate a new ECDH key pair */
1362                                    if (!(EC_KEY_generate_key(clnt_ecdh)))
1363                                            {
1364                                            SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHA
1365                                            goto err;
1366                                            }
1367                                    }
1369                            /* use the 'p' output buffer for the ECDH key, but
1370                             * make sure to clear it out afterwards
1371                             */
1373                            field_size = EC_GROUP_get_degree(srvr_group);
1374                            if (field_size <= 0)
1375                                    {
1376                                    SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1377                                           ERR_R_ECDH_LIB);
1378                                    goto err;
1379                                    }
1380                            n=ECDH_compute_key(p, (field_size+7)/8, srvr_ecpoint, cl
1381                            if (n <= 0)
```



```
1382                                    {
1383                                    SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1384                                           ERR_R_ECDH_LIB);
1385                                    goto err;
1386                                    }
1388                            /* generate master key from the result */
1389                            s->session->master_key_length = s->method->ssl3_enc \
1390                                -> generate_master_secret(s,
1391                                    s->session->master_key,
1392                                    p, n);
1394                            memset(p, 0, n); /* clean up */
1396                            if (ecdh_clnt_cert)
1397                                    {
1398                                    /* Send empty client key exch message */
1399                                    n = 0;
1400                                    }
1401                            else
1402                                    {
1403                                    /* First check the size of encoding and
1404                                     * allocate memory accordingly.
1405                                     */
1406                                    encoded_pt_len =
1407                                        EC_POINT_point2oct(srvr_group,
1408                                            EC_KEY_get0_public_key(clnt_ecdh),
1409                                            POINT_CONVERSION_UNCOMPRESSED,
1410                                            NULL, 0, NULL);
1412                                    encodedPoint = (unsigned char *)
1413                                        OPENSSL_malloc(encoded_pt_len *
1414                                            sizeof(unsigned char));
1415                                    bn_ctx = BN_CTX_new();
1416                                    if ((encodedPoint == NULL) ||
1417                                        (bn_ctx == NULL))
1418                                            {
1419                                            SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHA
1420                                            goto err;
1421                                            }
1423                                    /* Encode the public key */
1424                                    n = EC_POINT_point2oct(srvr_group,
1425                                        EC_KEY_get0_public_key(clnt_ecdh),
1426                                        POINT_CONVERSION_UNCOMPRESSED,
1427                                        encodedPoint, encoded_pt_len, bn_ctx);
1429                                    *p = n; /* length of encoded point */
1430                                    /* Encoded point will be copied here */
1431                                    p += 1;
1432                                    /* copy the point */
1433                                    memcpy((unsigned char *)p, encodedPoint, n);
1434                                    /* increment n to account for length field */
1435                                    n += 1;
1436                                    }
1438                            /* Free allocated memory */
1439                            BN_CTX_free(bn_ctx);
1440                            if (encodedPoint != NULL) OPENSSL_free(encodedPoint);
1441                            if (clnt_ecdh != NULL)
1442                                     EC_KEY_free(clnt_ecdh);
1443                            EVP_PKEY_free(srvr_pub_pkey);
1444                            }
1445 #endif /* !OPENSSL_NO_ECDH */
1447 #ifndef OPENSSL_NO_PSK
```

```
1448                 else if (alg_k & SSL_kPSK)
1449                         {
1450                         char identity[PSK_MAX_IDENTITY_LEN];
1451                         unsigned char *t = NULL;
1452                         unsigned char psk_or_pre_ms[PSK_MAX_PSK_LEN*2+4];
1453                         unsigned int pre_ms_len = 0, psk_len = 0;
1454                         int psk_err = 1;

1456                         n = 0;
1457                         if (s->psk_client_callback == NULL)
1458                                 {
1459                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1460                                         SSL_R_PSK_NO_CLIENT_CB);
1461                                 goto err;
1462                                 }

1464                         psk_len = s->psk_client_callback(s, s->ctx->psk_identity
1465                                 identity, PSK_MAX_IDENTITY_LEN,
1466                                 psk_or_pre_ms, sizeof(psk_or_pre_ms));
1467                         if (psk_len > PSK_MAX_PSK_LEN)
1468                                 {
1469                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1470                                         ERR_R_INTERNAL_ERROR);
1471                                 goto psk_err;
1472                                 }
1473                         else if (psk_len == 0)
1474                                 {
1475                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1476                                         SSL_R_PSK_IDENTITY_NOT_FOUND);
1477                                 goto psk_err;
1478                                 }

1480                         /* create PSK pre_master_secret */
1481                         pre_ms_len = 2+psk_len+2+psk_len;
1482                         t = psk_or_pre_ms;
1483                         memmove(psk_or_pre_ms+psk_len+4, psk_or_pre_ms, psk_len)
1484                         s2n(psk_len, t);
1485                         memset(t, 0, psk_len);
1486                         t+=psk_len;
1487                         s2n(psk_len, t);

1489                         if (s->session->psk_identity_hint != NULL)
1490                                 OPENSSL_free(s->session->psk_identity_hint);
1491                         s->session->psk_identity_hint = BUF_strdup(s->ctx->psk_i
1492                         if (s->ctx->psk_identity_hint != NULL &&
1493                                 s->session->psk_identity_hint == NULL)
1494                                 {
1495                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1496                                         ERR_R_MALLOC_FAILURE);
1497                                 goto psk_err;
1498                                 }

1500                         if (s->session->psk_identity != NULL)
1501                                 OPENSSL_free(s->session->psk_identity);
1502                         s->session->psk_identity = BUF_strdup(identity);
1503                         if (s->session->psk_identity == NULL)
1504                                 {
1505                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,
1506                                         ERR_R_MALLOC_FAILURE);
1507                                 goto psk_err;
1508                                 }

1510                         s->session->master_key_length =
1511                                 s->method->ssl3_enc->generate_master_secret(s,
1512                                         s->session->master_key,
1513                                         psk_or_pre_ms, pre_ms_len);
```



```
1514                         n = strlen(identity);
1515                         s2n(n, p);
1516                         memcpy(p, identity, n);
1517                         n+=2;
1518                         psk_err = 0;
1519                 psk_err:
1520                         OPENSSL_cleanse(identity, PSK_MAX_IDENTITY_LEN);
1521                         OPENSSL_cleanse(psk_or_pre_ms, sizeof(psk_or_pre_ms));
1522                         if (psk_err != 0)
1523                                 {
1524                                 ssl3_send_alert(s, SSL3_AL_FATAL, SSL_AD_HANDSHA
1525                                 goto err;
1526                                 }
1527                         }
1528 #endif
1529                 else
1530                         {
1531                         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_HANDSHAKE_FAILURE
1532                         SSLerr(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE,ERR_R_INTERN
1533                         goto err;
1534                         }

1536                 d = dtls1_set_message_header(s, d,
1537                 SSL3_MT_CLIENT_KEY_EXCHANGE, n, 0, n);
1538                 /*
1539                  *(d++)=SSL3_MT_CLIENT_KEY_EXCHANGE;
1540                  l2n3(n,d);
1541                  l2n(s->d1->handshake_write_seq,d);
1542                  s->d1->handshake_write_seq++;
1543                 */

1545                 s->state=SSL3_ST_CW_KEY_EXCH_B;
1546                 /* number of bytes to write */
1547                 s->init_num=n+DTLS1_HM_HEADER_LENGTH;
1548                 s->init_off=0;

1550                 /* buffer the message to handle re-xmits */
1551                 dtls1_buffer_message(s, 0);
1552                 }

1554         /* SSL3_ST_CW_KEY_EXCH_B */
1555         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1556 err:
1557 #ifndef OPENSSL_NO_ECDH
1558         BN_CTX_free(bn_ctx);
1559         if (encodedPoint != NULL) OPENSSL_free(encodedPoint);
1560         if (clnt_ecdh != NULL)
1561                 EC_KEY_free(clnt_ecdh);
1562         EVP_PKEY_free(srvr_pub_pkey);
1563 #endif
1564         return(-1);
1565         }

1567 int dtls1_send_client_verify(SSL *s)
1568         {
1569         unsigned char *p,*d;
1570         unsigned char data[MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH];
1571         EVP_PKEY *pkey;
1572 #ifndef OPENSSL_NO_RSA
1573         unsigned u=0;
1574 #endif
1575         unsigned long n;
1576 #if !defined(OPENSSL_NO_DSA) || !defined(OPENSSL_NO_ECDSA)
1577         int j;
1578 #endif
```

```
1580         if (s->state == SSL3_ST_CW_CERT_VRFY_A)
1581                 {
1582                 d=(unsigned char *)s->init_buf->data;
1583                 p= &(d[DTLS1_HM_HEADER_LENGTH]);
1584                 pkey=s->cert->key->privatekey;

1586                 s->method->ssl3_enc->cert_verify_mac(s,
1587                 NID_sha1,
1588                         &(data[MD5_DIGEST_LENGTH]));

1590 #ifndef OPENSSL_NO_RSA
1591                 if (pkey->type == EVP_PKEY_RSA)
1592                         {
1593                         s->method->ssl3_enc->cert_verify_mac(s,
1594                                 NID_md5,
1595                                 &(data[0]));
1596                         if (RSA_sign(NID_md5_sha1, data,
1597                                         MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH,
1598                                         &(p[2]), &u, pkey->pkey.rsa) <= 0 )
1599                                 {
1600                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_VERIFY,ERR_R_RSA_
1601                                 goto err;
1602                                 }
1603                         s2n(u,p);
1604                         n=u+2;
1605                         }
1606                 else
1607 #endif
1608 #ifndef OPENSSL_NO_DSA
1609                         if (pkey->type == EVP_PKEY_DSA)
1610                         {
1611                         if (!DSA_sign(pkey->save_type,
1612                                 &(data[MD5_DIGEST_LENGTH]),
1613                                 SHA_DIGEST_LENGTH,&(p[2]),
1614                                 (unsigned int *)&j,pkey->pkey.dsa))
1615                                 {
1616                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_VERIFY,ERR_R_DSA_
1617                                 goto err;
1618                                 }
1619                         s2n(j,p);
1620                         n=j+2;
1621                         }
1622                 else
1623 #endif
1624 #ifndef OPENSSL_NO_ECDSA
1625                         if (pkey->type == EVP_PKEY_EC)
1626                         {
1627                         if (!ECDSA_sign(pkey->save_type,
1628                                 &(data[MD5_DIGEST_LENGTH]),
1629                                 SHA_DIGEST_LENGTH,&(p[2]),
1630                                 (unsigned int *)&j,pkey->pkey.ec))
1631                                 {
1632                                 SSLerr(SSL_F_DTLS1_SEND_CLIENT_VERIFY,
1633                                     ERR_R_ECDSA_LIB);
1634                                 goto err;
1635                                 }
1636                         s2n(j,p);
1637                         n=j+2;
1638                         }
1639                 else
1640 #endif
1641                         {
1642                         SSLerr(SSL_F_DTLS1_SEND_CLIENT_VERIFY,ERR_R_INTERNAL_ERR
1643                         goto err;
1644                         }
```



```
1646                 d = dtls1_set_message_header(s, d,
1647                         SSL3_MT_CERTIFICATE_VERIFY, n, 0, n) ;

1649                 s->init_num=(int)n+DTLS1_HM_HEADER_LENGTH;
1650                 s->init_off=0;

1652                 /* buffer the message to handle re-xmits */
1653                 dtls1_buffer_message(s, 0);

1655                 s->state = SSL3_ST_CW_CERT_VRFY_B;
1656                 }

1658         /* s->state = SSL3_ST_CW_CERT_VRFY_B */
1659         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1660 err:
1661         return(-1);
1662         }

1664 int dtls1_send_client_certificate(SSL *s)
1665         {
1666         X509 *x509=NULL;
1667         EVP_PKEY *pkey=NULL;
1668         int i;
1669         unsigned long l;

1671         if (s->state == SSL3_ST_CW_CERT_A)
1672                 {
1673                 if ((s->cert == NULL) ||
1674                         (s->cert->key->x509 == NULL) ||
1675                         (s->cert->key->privatekey == NULL))
1676                         s->state=SSL3_ST_CW_CERT_B;
1677                 else
1678                         s->state=SSL3_ST_CW_CERT_C;
1679                 }

1681         /* We need to get a client cert */
1682         if (s->state == SSL3_ST_CW_CERT_B)
1683                 {
1684                 /* If we get an error, we need to
1685                  * ssl->rwstate=SSL_X509_LOOKUP; return(-1);
1686                  * We then get retied later */
1687                 i=0;
1688                 i = ssl_do_client_cert_cb(s, &x509, &pkey);
1689                 if (i < 0)
1690                         {
1691                         s->rwstate=SSL_X509_LOOKUP;
1692                         return(-1);
1693                         }
1694                 s->rwstate=SSL_NOTHING;
1695                 if ((i == 1) && (pkey != NULL) && (x509 != NULL))
1696                         {
1697                         s->state=SSL3_ST_CW_CERT_B;
1698                         if (    !SSL_use_certificate(s,x509) ||
1699                                 !SSL_use_PrivateKey(s,pkey))
1700                                 i=0;
1701                         }
1702                 else if (i == 1)
1703                         {
1704                         i=0;
1705                         SSLerr(SSL_F_DTLS1_SEND_CLIENT_CERTIFICATE,SSL_R_BAD_DAT
1706                         }

1708                 if (x509 != NULL) X509_free(x509);
1709                 if (pkey != NULL) EVP_PKEY_free(pkey);
1710                 if (i == 0)
1711                         {
```

```
1712                        if (s->version == SSL3_VERSION)
1713                                {
1714                                s->s3->tmp.cert_req=0;
1715                                ssl3_send_alert(s,SSL3_AL_WARNING,SSL_AD_NO_CERT
1716                                return(1);
1717                                }
1718                        else
1719                                {
1720                                s->s3->tmp.cert_req=2;
1721                                }
1722                        }

1724                /* Ok, we have a cert */
1725                s->state=SSL3_ST_CW_CERT_C;
1726                }

1728        if (s->state == SSL3_ST_CW_CERT_C)
1729                {
1730                s->state=SSL3_ST_CW_CERT_D;
1731                l=dtls1_output_cert_chain(s,
1732                        (s->s3->tmp.cert_req == 2)?NULL:s->cert->key->x509);
1733                s->init_num=(int)l;
1734                s->init_off=0;

1736                /* set header called by dtls1_output_cert_chain() */

1738                /* buffer the message to handle re-xmits */
1739                dtls1_buffer_message(s, 0);
1740                }
1741        /* SSL3_ST_CW_CERT_D */
1742        return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1743        }
1744 #endif /* ! codereview */
```

```
**********************************************************
    9356 Wed Aug 13 19:53:34 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/d1_enc.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1998-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
```



```
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "ssl_locl.h"
#ifndef OPENSSL_NO_COMP
#include <openssl/comp.h>
#endif
#include <openssl/evp.h>
#include <openssl/hmac.h>
#include <openssl/md5.h>
#include <openssl/rand.h>
#ifdef KSSL_DEBUG
#include <openssl/des.h>
#endif
```

```
/* dtls1_enc encrypts/decrypts the record in |s->wrec| / |s->rrec|, respectively
 *
 * Returns:
 *   0: (in non-constant time) if the record is publically invalid (i.e. too
 *      short etc).
 *   1: if the record's padding is valid / the encryption was successful.
 *   -1: if the record's padding/AEAD-authenticator is invalid or, if sending,
 *      an internal error occured. */
int dtls1_enc(SSL *s, int send)
        {
        SSL3_RECORD *rec;
        EVP_CIPHER_CTX *ds;
        unsigned long l;
        int bs,i,j,k,mac_size=0;
        const EVP_CIPHER *enc;

        if (send)
                {
                if (EVP_MD_CTX_md(s->write_hash))
                        {
                        mac_size=EVP_MD_CTX_size(s->write_hash);
                        if (mac_size < 0)
                                return -1;
                        }
                ds=s->enc_write_ctx;
                rec= &(s->s3->wrec);
                if (s->enc_write_ctx == NULL)
                        enc=NULL;
                else
                        {
                        enc=EVP_CIPHER_CTX_cipher(s->enc_write_ctx);
                        if ( rec->data != rec->input)
                                /* we can't write into the input stream */
                                fprintf(stderr, "%s:%d: rec->data != rec->input\
                                        __FILE__, __LINE__);
                        else if ( EVP_CIPHER_block_size(ds->cipher) > 1)
                                {
                                if (RAND_bytes(rec->input, EVP_CIPHER_block_size
                                        return -1;
                                }
                        }
                }
        else
                {
                if (EVP_MD_CTX_md(s->read_hash))
                        {
                        mac_size=EVP_MD_CTX_size(s->read_hash);
                        OPENSSL_assert(mac_size >= 0);
                        }
                ds=s->enc_read_ctx;
                rec= &(s->s3->rrec);
                if (s->enc_read_ctx == NULL)
                        enc=NULL;
                else
                        enc=EVP_CIPHER_CTX_cipher(s->enc_read_ctx);
                }

#ifdef KSSL_DEBUG
        printf("dtls1_enc(%d)\n", send);
#endif    /* KSSL_DEBUG */

        if ((s->session == NULL) || (ds == NULL) ||
                (enc == NULL))
                {
                memmove(rec->data,rec->input,rec->length);
```



```
                rec->input=rec->data;
                }
        else
                {
                l=rec->length;
                bs=EVP_CIPHER_block_size(ds->cipher);

                if ((bs != 1) && send)
                        {
                        i=bs-((int)l%bs);

                        /* Add weird padding of upto 256 bytes */

                        /* we need to add 'i' padding bytes of value j */
                        j=i-1;
                        if (s->options & SSL_OP_TLS_BLOCK_PADDING_BUG)
                                {
                                if (s->s3->flags & TLS1_FLAGS_TLS_PADDING_BUG)
                                        j++;
                                }
                        for (k=(int)l; k<(int)(l+i); k++)
                                rec->input[k]=j;
                        l+=i;
                        rec->length+=i;
                        }

#ifdef KSSL_DEBUG
                {
                unsigned long ui;
                printf("EVP_Cipher(ds=%p,rec->data=%p,rec->input=%p,l=%ld) ==>\n
                        ds,rec->data,rec->input,l);
                printf("\tEVP_CIPHER_CTX: %d buf_len, %d key_len [%d %d], %d iv_
                        ds->buf_len, ds->cipher->key_len,
                        DES_KEY_SZ, DES_SCHEDULE_SZ,
                        ds->cipher->iv_len);
                printf("\t\tIV: ");
                for (i=0; i<ds->cipher->iv_len; i++) printf("%02X", ds->iv[i]);
                printf("\n");
                printf("\trec->input=");
                for (ui=0; ui<l; ui++) printf(" %02x", rec->input[ui]);
                printf("\n");
                }
#endif  /* KSSL_DEBUG */

                if (!send)
                        {
                        if (l == 0 || l%bs != 0)
                                return 0;
                        }

                EVP_Cipher(ds,rec->data,rec->input,l);

#ifdef KSSL_DEBUG
                {
                unsigned long i;
                printf("\trec->data=");
                for (i=0; i<l; i++)
                        printf(" %02x", rec->data[i]);  printf("\n");
                }
#endif  /* KSSL_DEBUG */

                if ((bs != 1) && !send)
                        return tls1_cbc_remove_padding(s, rec, bs, mac_size);
                }
        return(1);
        }
```

```
260 #endif /* ! codereview */
```

```
**********************************************************
   12668 Wed Aug 13 19:53:34 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/d1_lib.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#define USE_SOCKETS
```



```
#include <openssl/objects.h>
#include "ssl_locl.h"

#if defined(OPENSSL_SYS_WIN32) || defined(OPENSSL_SYS_VMS)
#include <sys/timeb.h>
#endif

static void get_current_time(struct timeval *t);
const char dtls1_version_str[]="DTLSv1" OPENSSL_VERSION_PTEXT;
int dtls1_listen(SSL *s, struct sockaddr *client);

SSL3_ENC_METHOD DTLSv1_enc_data={
    dtls1_enc,
        tls1_mac,
        tls1_setup_key_block,
        tls1_generate_master_secret,
        tls1_change_cipher_state,
        tls1_final_finish_mac,
        TLS1_FINISH_MAC_LENGTH,
        tls1_cert_verify_mac,
        TLS_MD_CLIENT_FINISH_CONST,TLS_MD_CLIENT_FINISH_CONST_SIZE,
        TLS_MD_SERVER_FINISH_CONST,TLS_MD_SERVER_FINISH_CONST_SIZE,
        tls1_alert_code,
        tls1_export_keying_material,
        };

long dtls1_default_timeout(void)
        {
        /* 2 hours, the 24 hours mentioned in the DTLSv1 spec
         * is way too long for http, the cache would over fill */
        return(60*60*2);
        }

int dtls1_new(SSL *s)
        {
        DTLS1_STATE *d1;

        if (!ssl3_new(s)) return(0);
        if ((d1=OPENSSL_malloc(sizeof *d1)) == NULL) return (0);
        memset(d1,0, sizeof *d1);

        /* d1->handshake_epoch=0; */

        d1->unprocessed_rcds.q=pqueue_new();
        d1->processed_rcds.q=pqueue_new();
        d1->buffered_messages = pqueue_new();
        d1->sent_messages=pqueue_new();
        d1->buffered_app_data.q=pqueue_new();

        if ( s->server)
                {
                d1->cookie_len = sizeof(s->d1->cookie);
                }

        if( ! d1->unprocessed_rcds.q || ! d1->processed_rcds.q
        || ! d1->buffered_messages || ! d1->sent_messages || ! d1->buffered_app_
                {
        if ( d1->unprocessed_rcds.q) pqueue_free(d1->unprocessed_rcds.q);
        if ( d1->processed_rcds.q) pqueue_free(d1->processed_rcds.q);
        if ( d1->buffered_messages) pqueue_free(d1->buffered_messages);
                if ( d1->sent_messages) pqueue_free(d1->sent_messages);
                if ( d1->buffered_app_data.q) pqueue_free(d1->buffered_app_data.
                OPENSSL_free(d1);
                return (0);
                }
```

```
128         s->d1=d1;
129         s->method->ssl_clear(s);
130         return(1);
131         }

133 static void dtls1_clear_queues(SSL *s)
134         {
135     pitem *item = NULL;
136     hm_fragment *frag = NULL;
137         DTLS1_RECORD_DATA *rdata;

139     while( (item = pqueue_pop(s->d1->unprocessed_rcds.q)) != NULL)
140         {
141                 rdata = (DTLS1_RECORD_DATA *) item->data;
142                 if (rdata->rbuf.buf)
143                         {
144                         OPENSSL_free(rdata->rbuf.buf);
145                         }
146         OPENSSL_free(item->data);
147         pitem_free(item);
148         }

150     while( (item = pqueue_pop(s->d1->processed_rcds.q)) != NULL)
151         {
152                 rdata = (DTLS1_RECORD_DATA *) item->data;
153                 if (rdata->rbuf.buf)
154                         {
155                         OPENSSL_free(rdata->rbuf.buf);
156                         }
157         OPENSSL_free(item->data);
158         pitem_free(item);
159         }

161     while( (item = pqueue_pop(s->d1->buffered_messages)) != NULL)
162         {
163         frag = (hm_fragment *)item->data;
164         OPENSSL_free(frag->fragment);
165         OPENSSL_free(frag);
166         pitem_free(item);
167         }

169     while ( (item = pqueue_pop(s->d1->sent_messages)) != NULL)
170         {
171         frag = (hm_fragment *)item->data;
172         OPENSSL_free(frag->fragment);
173         OPENSSL_free(frag);
174         pitem_free(item);
175         }

177         while ( (item = pqueue_pop(s->d1->buffered_app_data.q)) != NULL)
178                 {
179                 rdata = (DTLS1_RECORD_DATA *) item->data;
180                 if (rdata->rbuf.buf)
181                         {
182                         OPENSSL_free(rdata->rbuf.buf);
183                         }
184                 OPENSSL_free(item->data);
185                 pitem_free(item);
186                 }
187         }

189 void dtls1_free(SSL *s)
190         {
191         ssl3_free(s);

193         dtls1_clear_queues(s);
```



```
195     pqueue_free(s->d1->unprocessed_rcds.q);
196     pqueue_free(s->d1->processed_rcds.q);
197     pqueue_free(s->d1->buffered_messages);
198         pqueue_free(s->d1->sent_messages);
199         pqueue_free(s->d1->buffered_app_data.q);

201         OPENSSL_free(s->d1);
202         s->d1 = NULL;
203         }

205 void dtls1_clear(SSL *s)
206         {
207     pqueue unprocessed_rcds;
208     pqueue processed_rcds;
209     pqueue buffered_messages;
210         pqueue sent_messages;
211         pqueue buffered_app_data;
212         unsigned int mtu;

214         if (s->d1)
215                 {
216                 unprocessed_rcds = s->d1->unprocessed_rcds.q;
217                 processed_rcds = s->d1->processed_rcds.q;
218                 buffered_messages = s->d1->buffered_messages;
219                 sent_messages = s->d1->sent_messages;
220                 buffered_app_data = s->d1->buffered_app_data.q;
221                 mtu = s->d1->mtu;

223                 dtls1_clear_queues(s);

225                 memset(s->d1, 0, sizeof(*(s->d1)));

227                 if (s->server)
228                         {
229                         s->d1->cookie_len = sizeof(s->d1->cookie);
230                         }

232                 if (SSL_get_options(s) & SSL_OP_NO_QUERY_MTU)
233                         {
234                         s->d1->mtu = mtu;
235                         }

237                 s->d1->unprocessed_rcds.q = unprocessed_rcds;
238                 s->d1->processed_rcds.q = processed_rcds;
239                 s->d1->buffered_messages = buffered_messages;
240                 s->d1->sent_messages = sent_messages;
241                 s->d1->buffered_app_data.q = buffered_app_data;
242                 }

244         ssl3_clear(s);
245         if (s->options & SSL_OP_CISCO_ANYCONNECT)
246                 s->version=DTLS1_BAD_VER;
247         else
248                 s->version=DTLS1_VERSION;
249         }

251 long dtls1_ctrl(SSL *s, int cmd, long larg, void *parg)
252         {
253         int ret=0;

255         switch (cmd)
256                 {
257         case DTLS_CTRL_GET_TIMEOUT:
258                 if (dtls1_get_timeout(s, (struct timeval*) parg) != NULL)
259                         {
```

```
260                         ret = 1;
261                         }
262                 break;
263         case DTLS_CTRL_HANDLE_TIMEOUT:
264                 ret = dtls1_handle_timeout(s);
265                 break;
266         case DTLS_CTRL_LISTEN:
267                 ret = dtls1_listen(s, parg);
268                 break;

270         default:
271                 ret = ssl3_ctrl(s, cmd, larg, parg);
272                 break;
273                 }
274         return(ret);
275         }

277 /*
278  * As it's impossible to use stream ciphers in "datagram" mode, this
279  * simple filter is designed to disengage them in DTLS. Unfortunately
280  * there is no universal way to identify stream SSL_CIPHER, so we have
281  * to explicitly list their SSL_* codes. Currently RC4 is the only one
282  * available, but if new ones emerge, they will have to be added...
283  */
284 const SSL_CIPHER *dtls1_get_cipher(unsigned int u)
285         {
286         const SSL_CIPHER *ciph = ssl3_get_cipher(u);

288         if (ciph != NULL)
289                 {
290                 if (ciph->algorithm_enc == SSL_RC4)
291                         return NULL;
292                 }

294         return ciph;
295         }

297 void dtls1_start_timer(SSL *s)
298         {
299 #ifndef OPENSSL_NO_SCTP
300         /* Disable timer for SCTP */
301         if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
302                 {
303                 memset(&(s->d1->next_timeout), 0, sizeof(struct timeval));
304                 return;
305                 }
306 #endif

308         /* If timer is not set, initialize duration with 1 second */
309         if (s->d1->next_timeout.tv_sec == 0 && s->d1->next_timeout.tv_usec == 0)
310                 {
311                 s->d1->timeout_duration = 1;
312                 }

314         /* Set timeout to current time */
315         get_current_time(&(s->d1->next_timeout));

317         /* Add duration to current time */
318         s->d1->next_timeout.tv_sec += s->d1->timeout_duration;
319         BIO_ctrl(SSL_get_rbio(s), BIO_CTRL_DGRAM_SET_NEXT_TIMEOUT, 0, &(s->d1->n
320         }

322 struct timeval* dtls1_get_timeout(SSL *s, struct timeval* timeleft)
323         {
324         struct timeval timenow;
```



```
326         /* If no timeout is set, just return NULL */
327         if (s->d1->next_timeout.tv_sec == 0 && s->d1->next_timeout.tv_usec == 0)
328                 {
329                 return NULL;
330                 }

332         /* Get current time */
333         get_current_time(&timenow);

335         /* If timer already expired, set remaining time to 0 */
336         if (s->d1->next_timeout.tv_sec < timenow.tv_sec ||
337                 (s->d1->next_timeout.tv_sec == timenow.tv_sec &&
338                  s->d1->next_timeout.tv_usec <= timenow.tv_usec))
339                 {
340                 memset(timeleft, 0, sizeof(struct timeval));
341                 return timeleft;
342                 }

344         /* Calculate time left until timer expires */
345         memcpy(timeleft, &(s->d1->next_timeout), sizeof(struct timeval));
346         timeleft->tv_sec -= timenow.tv_sec;
347         timeleft->tv_usec -= timenow.tv_usec;
348         if (timeleft->tv_usec < 0)
349                 {
350                 timeleft->tv_sec--;
351                 timeleft->tv_usec += 1000000;
352                 }

354         /* If remaining time is less than 15 ms, set it to 0
355          * to prevent issues because of small devergences with
356          * socket timeouts.
357          */
358         if (timeleft->tv_sec == 0 && timeleft->tv_usec < 15000)
359                 {
360                 memset(timeleft, 0, sizeof(struct timeval));
361                 }


364         return timeleft;
365         }

367 int dtls1_is_timer_expired(SSL *s)
368         {
369         struct timeval timeleft;

371         /* Get time left until timeout, return false if no timer running */
372         if (dtls1_get_timeout(s, &timeleft) == NULL)
373                 {
374                 return 0;
375                 }

377         /* Return false if timer is not expired yet */
378         if (timeleft.tv_sec > 0 || timeleft.tv_usec > 0)
379                 {
380                 return 0;
381                 }

383         /* Timer expired, so return true */
384         return 1;
385         }

387 void dtls1_double_timeout(SSL *s)
388         {
389         s->d1->timeout_duration *= 2;
390         if (s->d1->timeout_duration > 60)
391                 s->d1->timeout_duration = 60;
```

```
392         dtls1_start_timer(s);
393         }

395 void dtls1_stop_timer(SSL *s)
396         {
397         /* Reset everything */
398         memset(&(s->d1->timeout), 0, sizeof(struct dtls1_timeout_st));
399         memset(&(s->d1->next_timeout), 0, sizeof(struct timeval));
400         s->d1->timeout_duration = 1;
401         BIO_ctrl(SSL_get_rbio(s), BIO_CTRL_DGRAM_SET_NEXT_TIMEOUT, 0, &(s->d1->n
402         /* Clear retransmission buffer */
403         dtls1_clear_record_buffer(s);
404         }

406 int dtls1_check_timeout_num(SSL *s)
407         {
408         s->d1->timeout.num_alerts++;

410         /* Reduce MTU after 2 unsuccessful retransmissions */
411         if (s->d1->timeout.num_alerts > 2)
412                 {
413                 s->d1->mtu = BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_GET_FALLBA
414                 }

416         if (s->d1->timeout.num_alerts > DTLS1_TMO_ALERT_COUNT)
417                 {
418                 /* fail the connection, enough alerts have been sent */
419                 SSLerr(SSL_F_DTLS1_CHECK_TIMEOUT_NUM,SSL_R_READ_TIMEOUT_EXPIRED)
420                 return -1;
421                 }

423         return 0;
424         }

426 int dtls1_handle_timeout(SSL *s)
427         {
428         /* if no timer is expired, don't do anything */
429         if (!dtls1_is_timer_expired(s))
430                 {
431                 return 0;
432                 }

434         dtls1_double_timeout(s);

436         if (dtls1_check_timeout_num(s) < 0)
437                 return -1;

439         s->d1->timeout.read_timeouts++;
440         if (s->d1->timeout.read_timeouts > DTLS1_TMO_READ_COUNT)
441                 {
442                 s->d1->timeout.read_timeouts = 1;
443                 }

445 #ifndef OPENSSL_NO_HEARTBEATS
446         if (s->tlsext_hb_pending)
447                 {
448                 s->tlsext_hb_pending = 0;
449                 return dtls1_heartbeat(s);
450                 }
451 #endif

453         dtls1_start_timer(s);
454         return dtls1_retransmit_buffered_messages(s);
455         }

457 static void get_current_time(struct timeval *t)
```



```
458 {
459 #ifdef OPENSSL_SYS_WIN32
460         struct _timeb tb;
461         _ftime(&tb);
462         t->tv_sec = (long)tb.time;
463         t->tv_usec = (long)tb.millitm * 1000;
464 #elif defined(OPENSSL_SYS_VMS)
465         struct timeb tb;
466         ftime(&tb);
467         t->tv_sec = (long)tb.time;
468         t->tv_usec = (long)tb.millitm * 1000;
469 #else
470         gettimeofday(t, NULL);
471 #endif
472 }

474 int dtls1_listen(SSL *s, struct sockaddr *client)
475         {
476         int ret;

478         SSL_set_options(s, SSL_OP_COOKIE_EXCHANGE);
479         s->d1->listen = 1;

481         ret = SSL_accept(s);
482         if (ret <= 0) return ret;

484         (void) BIO_dgram_get_peer(SSL_get_rbio(s), client);
485         return 1;
486         }
487 #endif /* ! codereview */
```

```
**********************************************************
    3127 Wed Aug 13 19:53:34 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_meth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/d1_meth.h */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/objects.h>
```



```
#include "ssl_locl.h"

static const SSL_METHOD *dtls1_get_method(int ver);
static const SSL_METHOD *dtls1_get_method(int ver)
        {
        if (ver == DTLS1_VERSION)
                return(DTLSv1_method());
        else
                return(NULL);
        }

IMPLEMENT_dtls1_meth_func(DTLSv1_method,
                          dtls1_accept,
                          dtls1_connect,
                          dtls1_get_method)
#endif /* ! codereview */
```

```
**********************************************************
   52563 Wed Aug 13 19:53:35 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_pkt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/d1_pkt.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1998-2005 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
```



```
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#define USE_SOCKETS
#include "ssl_locl.h"
#include <openssl/evp.h>
#include <openssl/buffer.h>
#include <openssl/pqueue.h>
#include <openssl/rand.h>

/* mod 128 saturating subtract of two 64-bit values in big-endian order */
static int satsub64be(const unsigned char *v1,const unsigned char *v2)
{       int ret,sat,brw,i;
```

```
129         if (sizeof(long) == 8) do
130         {       const union { long one; char little; } is_endian = {1};
131                 long l;

133                 if (is_endian.little)                   break;
134                 /* not reached on little-endians */
135                 /* following test is redundant, because input is
136                  * always aligned, but I take no chances... */
137                 if (((size_t)v1|(size_t)v2)&0x7)        break;

139                 l  = *((long *)v1);
140                 l -= *((long *)v2);
141                 if (l>128)              return 128;
142                 else if (l<-128)        return -128;
143                 else                    return (int)l;
144         } while (0);

146         ret = (int)v1[7]-(int)v2[7];
147         sat = 0;
148         brw = ret>>8;   /* brw is either 0 or -1 */
149         if (ret & 0x80)
150         {       for (i=6;i>=0;i--)
151                 {       brw += (int)v1[i]-(int)v2[i];
152                         sat |= ~brw;
153                         brw >>= 8;
154                 }
155         }
156         else
157         {       for (i=6;i>=0;i--)
158                 {       brw += (int)v1[i]-(int)v2[i];
159                         sat |= brw;
160                         brw >>= 8;
161                 }
162         }
163         brw <<= 8;      /* brw is either 0 or -256 */

165         if (sat&0xff)   return brw | 0x80;
166         else            return brw + (ret&0xFF);
167 }

169 static int have_handshake_fragment(SSL *s, int type, unsigned char *buf,
170         int len, int peek);
171 static int dtls1_record_replay_check(SSL *s, DTLS1_BITMAP *bitmap);
172 static void dtls1_record_bitmap_update(SSL *s, DTLS1_BITMAP *bitmap);
173 static DTLS1_BITMAP *dtls1_get_bitmap(SSL *s, SSL3_RECORD *rr,
174     unsigned int *is_next_epoch);
175 #if 0
176 static int dtls1_record_needs_buffering(SSL *s, SSL3_RECORD *rr,
177         unsigned short *priority, unsigned long *offset);
178 #endif
179 static int dtls1_buffer_record(SSL *s, record_pqueue *q,
180         unsigned char *priority);
181 static int dtls1_process_record(SSL *s);

183 /* copy buffered record into SSL structure */
184 static int
185 dtls1_copy_record(SSL *s, pitem *item)
186     {
187     DTLS1_RECORD_DATA *rdata;

189     rdata = (DTLS1_RECORD_DATA *)item->data;

191     if (s->s3->rbuf.buf != NULL)
192         OPENSSL_free(s->s3->rbuf.buf);
```



```
194     s->packet = rdata->packet;
195     s->packet_length = rdata->packet_length;
196     memcpy(&(s->s3->rbuf), &(rdata->rbuf), sizeof(SSL3_BUFFER));
197     memcpy(&(s->s3->rrec), &(rdata->rrec), sizeof(SSL3_RECORD));

199         /* Set proper sequence number for mac calculation */
200         memcpy(&(s->s3->read_sequence[2]), &(rdata->packet[5]), 6);

202     return(1);
203     }


206 static int
207 dtls1_buffer_record(SSL *s, record_pqueue *queue, unsigned char *priority)
208         {
209         DTLS1_RECORD_DATA *rdata;
210         pitem *item;

212         /* Limit the size of the queue to prevent DOS attacks */
213         if (pqueue_size(queue->q) >= 100)
214                 return 0;

216         rdata = OPENSSL_malloc(sizeof(DTLS1_RECORD_DATA));
217         item = pitem_new(priority, rdata);
218         if (rdata == NULL || item == NULL)
219                 {
220                 if (rdata != NULL) OPENSSL_free(rdata);
221                 if (item != NULL) pitem_free(item);

223                 SSLerr(SSL_F_DTLS1_BUFFER_RECORD, ERR_R_INTERNAL_ERROR);
224                 return(0);
225                 }

227         rdata->packet = s->packet;
228         rdata->packet_length = s->packet_length;
229         memcpy(&(rdata->rbuf), &(s->s3->rbuf), sizeof(SSL3_BUFFER));
230         memcpy(&(rdata->rrec), &(s->s3->rrec), sizeof(SSL3_RECORD));

232         item->data = rdata;

234 #ifndef OPENSSL_NO_SCTP
235         /* Store bio_dgram_sctp_rcvinfo struct */
236         if (BIO_dgram_is_sctp(SSL_get_rbio(s)) &&
237             (s->state == SSL3_ST_SR_FINISHED_A || s->state == SSL3_ST_CR_FINISHE
238                 BIO_ctrl(SSL_get_rbio(s), BIO_CTRL_DGRAM_SCTP_GET_RCVINFO, sizeo
239         }
240 #endif

242         s->packet = NULL;
243         s->packet_length = 0;
244         memset(&(s->s3->rbuf), 0, sizeof(SSL3_BUFFER));
245         memset(&(s->s3->rrec), 0, sizeof(SSL3_RECORD));

247         if (!ssl3_setup_buffers(s))
248                 {
249                 SSLerr(SSL_F_DTLS1_BUFFER_RECORD, ERR_R_INTERNAL_ERROR);
250                 OPENSSL_free(rdata);
251                 pitem_free(item);
252                 return(0);
253                 }

255         /* insert should not fail, since duplicates are dropped */
256         if (pqueue_insert(queue->q, item) == NULL)
257                 {
258                 SSLerr(SSL_F_DTLS1_BUFFER_RECORD, ERR_R_INTERNAL_ERROR);
259                 OPENSSL_free(rdata);
```

```
260                 pitem_free(item);
261                 return(0);
262                 }

264         return(1);
265         }


268 static int
269 dtls1_retrieve_buffered_record(SSL *s, record_pqueue *queue)
270     {
271     pitem *item;

273     item = pqueue_pop(queue->q);
274     if (item)
275         {
276         dtls1_copy_record(s, item);

278         OPENSSL_free(item->data);
279                 pitem_free(item);

281         return(1);
282         }

284     return(0);
285     }


288 /* retrieve a buffered record that belongs to the new epoch, i.e., not processed
289  * yet */
290 #define dtls1_get_unprocessed_record(s) \
291                    dtls1_retrieve_buffered_record((s), \
292                    &((s)->d1->unprocessed_rcds))

294 /* retrieve a buffered record that belongs to the current epoch, ie, processed *
295 #define dtls1_get_processed_record(s) \
296                    dtls1_retrieve_buffered_record((s), \
297                    &((s)->d1->processed_rcds))

299 static int
300 dtls1_process_buffered_records(SSL *s)
301     {
302     pitem *item;

304     item = pqueue_peek(s->d1->unprocessed_rcds.q);
305     if (item)
306         {
307         /* Check if epoch is current. */
308         if (s->d1->unprocessed_rcds.epoch != s->d1->r_epoch)
309             return(1);  /* Nothing to do. */

311         /* Process all the records. */
312         while (pqueue_peek(s->d1->unprocessed_rcds.q))
313             {
314             dtls1_get_unprocessed_record(s);
315             if ( ! dtls1_process_record(s))
316                 return(0);
317             dtls1_buffer_record(s, &(s->d1->processed_rcds),
318                 s->s3->rrec.seq_num);
319             }
320         }

322     /* sync epoch numbers once all the unprocessed records
323      * have been processed */
324     s->d1->processed_rcds.epoch = s->d1->r_epoch;
325     s->d1->unprocessed_rcds.epoch = s->d1->r_epoch + 1;
```



```
327     return(1);
328     }


331 #if 0

333 static int
334 dtls1_get_buffered_record(SSL *s)
335         {
336         pitem *item;
337         PQ_64BIT priority =
338                 (((PQ_64BIT)s->d1->handshake_read_seq) << 32) |
339                 ((PQ_64BIT)s->d1->r_msg_hdr.frag_off);

341         if ( ! SSL_in_init(s))  /* if we're not (re)negotiating,
342                                                         nothing buffered */
343                 return 0;


346         item = pqueue_peek(s->d1->rcvd_records);
347         if (item && item->priority == priority)
348                 {
349                 /* Check if we've received the record of interest.  It must be
350                  * a handshake record, since data records as passed up without
351                  * buffering */
352                 DTLS1_RECORD_DATA *rdata;
353                 item = pqueue_pop(s->d1->rcvd_records);
354                 rdata = (DTLS1_RECORD_DATA *)item->data;

356                 if (s->s3->rbuf.buf != NULL)
357                         OPENSSL_free(s->s3->rbuf.buf);

359                 s->packet = rdata->packet;
360                 s->packet_length = rdata->packet_length;
361                 memcpy(&(s->s3->rbuf), &(rdata->rbuf), sizeof(SSL3_BUFFER));
362                 memcpy(&(s->s3->rrec), &(rdata->rrec), sizeof(SSL3_RECORD));

364                 OPENSSL_free(item->data);
365                 pitem_free(item);

367                 /* s->d1->next_expected_seq_num++; */
368                 return(1);
369                 }

371         return 0;
372         }

374 #endif

376 static int
377 dtls1_process_record(SSL *s)
378 {
379         int i,al;
380         int enc_err;
381         SSL_SESSION *sess;
382         SSL3_RECORD *rr;
383         unsigned int mac_size, orig_len;
384         unsigned char md[EVP_MAX_MD_SIZE];

386         rr= &(s->s3->rrec);
387         sess = s->session;

389         /* At this point, s->packet_length == SSL3_RT_HEADER_LNGTH + rr->length,
390          * and we have that many bytes in s->packet
391          */
```

```
392         rr->input= &(s->packet[DTLS1_RT_HEADER_LENGTH]);

394         /* ok, we can now read from 's->packet' data into 'rr'
395          * rr->input points at rr->length bytes, which
396          * need to be copied into rr->data by either
397          * the decryption or by the decompression
398          * When the data is 'copied' into the rr->data buffer,
399          * rr->input will be pointed at the new buffer */

401         /* We now have - encrypted [ MAC [ compressed [ plain ] ] ]
402          * rr->length bytes of encrypted compressed stuff. */

404         /* check is not needed I believe */
405         if (rr->length > SSL3_RT_MAX_ENCRYPTED_LENGTH)
406                 {
407                 al=SSL_AD_RECORD_OVERFLOW;
408                 SSLerr(SSL_F_DTLS1_PROCESS_RECORD,SSL_R_ENCRYPTED_LENGTH_TOO_LON
409                 goto f_err;
410                 }

412         /* decrypt in place in 'rr->input' */
413         rr->data=rr->input;

415         enc_err = s->method->ssl3_enc->enc(s,0);
416         /* enc_err is:
417          *    0: (in non-constant time) if the record is publically invalid.
418          *    1: if the padding is valid
419          *    -1: if the padding is invalid */
420         if (enc_err == 0)
421                 {
422                 /* For DTLS we simply ignore bad packets. */
423                 rr->length = 0;
424                 s->packet_length = 0;
425                 goto err;
426                 }

428 #ifdef TLS_DEBUG
429 printf("dec %d\n",rr->length);
430 { unsigned int z; for (z=0; z<rr->length; z++) printf("%02X%c",rr->data[z],((z+1
431 printf("\n");
432 #endif

434         /* r->length is now the compressed data plus mac */
435         if ((sess != NULL) &&
436             (s->enc_read_ctx != NULL) &&
437             (EVP_MD_CTX_md(s->read_hash) != NULL))
438                 {
439                 /* s->read_hash != NULL => mac_size != -1 */
440                 unsigned char *mac = NULL;
441                 unsigned char mac_tmp[EVP_MAX_MD_SIZE];
442                 mac_size=EVP_MD_CTX_size(s->read_hash);
443                 OPENSSL_assert(mac_size <= EVP_MAX_MD_SIZE);

445                 /* kludge: *_cbc_remove_padding passes padding length in rr->typ
446                 orig_len = rr->length+((unsigned int)rr->type>>8);

448                 /* orig_len is the length of the record before any padding was
449                  * removed. This is public information, as is the MAC in use,
450                  * therefore we can safely process the record in a different
451                  * amount of time if it's too short to possibly contain a MAC.
452                  */
453                 if (orig_len < mac_size ||
454                     /* CBC records must have a padding length byte too. */
455                     (EVP_CIPHER_CTX_mode(s->enc_read_ctx) == EVP_CIPH_CBC_MODE &
456                      orig_len < mac_size+1))
457                         {
```



```
458                         al=SSL_AD_DECODE_ERROR;
459                         SSLerr(SSL_F_DTLS1_PROCESS_RECORD,SSL_R_LENGTH_TOO_SHORT
460                         goto f_err;
461                         }

463                 if (EVP_CIPHER_CTX_mode(s->enc_read_ctx) == EVP_CIPH_CBC_MODE)
464                         {
465                         /* We update the length so that the TLS header bytes
466                          * can be constructed correctly but we need to extract
467                          * the MAC in constant time from within the record,
468                          * without leaking the contents of the padding bytes.
469                          * */
470                         mac = mac_tmp;
471                         ssl3_cbc_copy_mac(mac_tmp, rr, mac_size, orig_len);
472                         rr->length -= mac_size;
473                         }
474                 else
475                         {
476                         /* In this case there's no padding, so |orig_len|
477                          * equals |rec->length| and we checked that there's
478                          * enough bytes for |mac_size| above. */
479                         rr->length -= mac_size;
480                         mac = &rr->data[rr->length];
481                         }

483                 i=s->method->ssl3_enc->mac(s,md,0 /* not send */);
484                 if (i < 0 || mac == NULL || CRYPTO_memcmp(md, mac, (size_t)mac_s
485                         enc_err = -1;
486                 if (rr->length > SSL3_RT_MAX_COMPRESSED_LENGTH+mac_size)
487                         enc_err = -1;
488                 }

490         if (enc_err < 0)
491                 {
492                 /* decryption failed, silently discard message */
493                 rr->length = 0;
494                 s->packet_length = 0;
495                 goto err;
496                 }

498         /* r->length is now just compressed */
499         if (s->expand != NULL)
500                 {
501                 if (rr->length > SSL3_RT_MAX_COMPRESSED_LENGTH)
502                         {
503                         al=SSL_AD_RECORD_OVERFLOW;
504                         SSLerr(SSL_F_DTLS1_PROCESS_RECORD,SSL_R_COMPRESSED_LENGT
505                         goto f_err;
506                         }
507                 if (!ssl3_do_uncompress(s))
508                         {
509                         al=SSL_AD_DECOMPRESSION_FAILURE;
510                         SSLerr(SSL_F_DTLS1_PROCESS_RECORD,SSL_R_BAD_DECOMPRESSIO
511                         goto f_err;
512                         }
513                 }

515         if (rr->length > SSL3_RT_MAX_PLAIN_LENGTH)
516                 {
517                 al=SSL_AD_RECORD_OVERFLOW;
518                 SSLerr(SSL_F_DTLS1_PROCESS_RECORD,SSL_R_DATA_LENGTH_TOO_LONG);
519                 goto f_err;
520                 }

522         rr->off=0;
523         /* So at this point the following is true
```

```
524          * ssl->s3->rrec.type   is the type of record
525          * ssl->s3->rrec.length == number of bytes in record
526          * ssl->s3->rrec.off    == offset to first valid byte
527          * ssl->s3->rrec.data   == where to take bytes from, increment
528          *                         after use :-).
529          */

531         /* we have pulled in a full packet so zero things */
532         s->packet_length=0;
533         dtls1_record_bitmap_update(s, &(s->d1->bitmap));/* Mark receipt of recor
534         return(1);

536 f_err:
537         ssl3_send_alert(s,SSL3_AL_FATAL,al);
538 err:
539         return(0);
540 }


543 /* Call this to get a new input record.
544  * It will return <= 0 if more data is needed, normally due to an error
545  * or non-blocking IO.
546  * When it finishes, one packet has been decoded and can be found in
547  * ssl->s3->rrec.type    - is the type of record
548  * ssl->s3->rrec.data,   - data
549  * ssl->s3->rrec.length, - number of bytes
550  */
551 /* used only by dtls1_read_bytes */
552 int dtls1_get_record(SSL *s)
553         {
554         int ssl_major,ssl_minor;
555         int i,n;
556         SSL3_RECORD *rr;
557         unsigned char *p = NULL;
558         unsigned short version;
559         DTLS1_BITMAP *bitmap;
560         unsigned int is_next_epoch;

562         rr= &(s->s3->rrec);

564         /* The epoch may have changed.  If so, process all the
565          * pending records.  This is a non-blocking operation. */
566         dtls1_process_buffered_records(s);

568         /* if we're renegotiating, then there may be buffered records */
569         if (dtls1_get_processed_record(s))
570                 return 1;

572         /* get something from the wire */
573 again:
574         /* check if we have the header */
575         if (    (s->rstate != SSL_ST_READ_BODY) ||
576                 (s->packet_length < DTLS1_RT_HEADER_LENGTH))
577                 {
578                 n=ssl3_read_n(s, DTLS1_RT_HEADER_LENGTH, s->s3->rbuf.len, 0);
579                 /* read timeout is handled by dtls1_read_bytes */
580                 if (n <= 0) return(n); /* error or non-blocking */

582                 /* this packet contained a partial record, dump it */
583                 if (s->packet_length != DTLS1_RT_HEADER_LENGTH)
584                         {
585                         s->packet_length = 0;
586                         goto again;
587                         }

589                 s->rstate=SSL_ST_READ_BODY;
```



```
591                 p=s->packet;

593                 /* Pull apart the header into the DTLS1_RECORD */
594                 rr->type= *(p++);
595                 ssl_major= *(p++);
596                 ssl_minor= *(p++);
597                 version=(ssl_major<<8)|ssl_minor;

599                 /* sequence number is 64 bits, with top 2 bytes = epoch */
600                 n2s(p,rr->epoch);

602                 memcpy(&(s->s3->read_sequence[2]), p, 6);
603                 p+=6;

605                 n2s(p,rr->length);

607                 /* Lets check version */
608                 if (!s->first_packet)
609                         {
610                         if (version != s->version)
611                                 {
612                                 /* unexpected version, silently discard */
613                                 rr->length = 0;
614                                 s->packet_length = 0;
615                                 goto again;
616                                 }
617                         }

619                 if ((version & 0xff00) != (s->version & 0xff00))
620                         {
621                         /* wrong version, silently discard record */
622                         rr->length = 0;
623                         s->packet_length = 0;
624                         goto again;
625                         }

627                 if (rr->length > SSL3_RT_MAX_ENCRYPTED_LENGTH)
628                         {
629                         /* record too long, silently discard it */
630                         rr->length = 0;
631                         s->packet_length = 0;
632                         goto again;
633                         }

635                 /* now s->rstate == SSL_ST_READ_BODY */
636                 }

638         /* s->rstate == SSL_ST_READ_BODY, get and decode the data */

640         if (rr->length > s->packet_length-DTLS1_RT_HEADER_LENGTH)
641                 {
642                 /* now s->packet_length == DTLS1_RT_HEADER_LENGTH */
643                 i=rr->length;
644                 n=ssl3_read_n(s,i,i,1);
645                 if (n <= 0) return(n); /* error or non-blocking io */

647                 /* this packet contained a partial record, dump it */
648                 if ( n != i)
649                         {
650                         rr->length = 0;
651                         s->packet_length = 0;
652                         goto again;
653                         }

655                 /* now n == rr->length,
```

```
656                  * and s->packet_length == DTLS1_RT_HEADER_LENGTH + rr->length *
657                 }
658         s->rstate=SSL_ST_READ_HEADER; /* set state for later operations */

660         /* match epochs.  NULL means the packet is dropped on the floor */
661         bitmap = dtls1_get_bitmap(s, rr, &is_next_epoch);
662         if ( bitmap == NULL)
663                 {
664                 rr->length = 0;
665                 s->packet_length = 0;  /* dump this record */
666                 goto again;   /* get another record */
667                 }

669 #ifndef OPENSSL_NO_SCTP
670         /* Only do replay check if no SCTP bio */
671         if (!BIO_dgram_is_sctp(SSL_get_rbio(s)))
672                 {
673 #endif
674                 /* Check whether this is a repeat, or aged record.
675                  * Don't check if we're listening and this message is
676                  * a ClientHello. They can look as if they're replayed,
677                  * since they arrive from different connections and
678                  * would be dropped unnecessarily.
679                  */
680                 if (!(s->d1->listen && rr->type == SSL3_RT_HANDSHAKE &&
681                     *p == SSL3_MT_CLIENT_HELLO) &&
682                     !dtls1_record_replay_check(s, bitmap))
683                         {
684                         rr->length = 0;
685                         s->packet_length=0; /* dump this record */
686                         goto again;     /* get another record */
687                         }
688 #ifndef OPENSSL_NO_SCTP
689                 }
690 #endif

692         /* just read a 0 length packet */
693         if (rr->length == 0) goto again;

695         /* If this record is from the next epoch (either HM or ALERT),
696          * and a handshake is currently in progress, buffer it since it
697          * cannot be processed at this time. However, do not buffer
698          * anything while listening.
699          */
700         if (is_next_epoch)
701                 {
702                 if ((SSL_in_init(s) || s->in_handshake) && !s->d1->listen)
703                         {
704                         dtls1_buffer_record(s, &(s->d1->unprocessed_rcds), rr->s
705                         }
706                 rr->length = 0;
707                 s->packet_length = 0;
708                 goto again;
709                 }

711         if (!dtls1_process_record(s))
712                 {
713                 rr->length = 0;
714                 s->packet_length = 0;  /* dump this record */
715                 goto again;   /* get another record */
716                 }

718         return(1);

720         }
```



```
722 /* Return up to 'len' payload bytes received in 'type' records.
723  * 'type' is one of the following:
724  *
725  *   -  SSL3_RT_HANDSHAKE (when ssl3_get_message calls us)
726  *   -  SSL3_RT_APPLICATION_DATA (when ssl3_read calls us)
727  *   -  0 (during a shutdown, no data has to be returned)
728  *
729  * If we don't have stored data to work from, read a SSL/TLS record first
730  * (possibly multiple records if we still don't have anything to return).
731  *
732  * This function must handle any surprises the peer may have for us, such as
733  * Alert records (e.g. close_notify), ChangeCipherSpec records (not really
734  * a surprise, but handled as if it were), or renegotiation requests.
735  * Also if record payloads contain fragments too small to process, we store
736  * them until there is enough for the respective protocol (the record protocol
737  * may use arbitrary fragmentation and even interleaving):
738  *     Change cipher spec protocol
739  *             just 1 byte needed, no need for keeping anything stored
740  *     Alert protocol
741  *             2 bytes needed (AlertLevel, AlertDescription)
742  *     Handshake protocol
743  *             4 bytes needed (HandshakeType, uint24 length) -- we just have
744  *             to detect unexpected Client Hello and Hello Request messages
745  *             here, anything else is handled by higher layers
746  *     Application data protocol
747  *             none of our business
748  */
749 int dtls1_read_bytes(SSL *s, int type, unsigned char *buf, int len, int peek)
750         {
751         int al,i,j,ret;
752         unsigned int n;
753         SSL3_RECORD *rr;
754         void (*cb)(const SSL *ssl,int type2,int val)=NULL;

756         if (s->s3->rbuf.buf == NULL) /* Not initialized yet */
757                 if (!ssl3_setup_buffers(s))
758                         return(-1);

760     /* XXX: check what the second '&& type' is about */
761         if ((type && (type != SSL3_RT_APPLICATION_DATA) &&
762                 (type != SSL3_RT_HANDSHAKE) && type) ||
763             (peek && (type != SSL3_RT_APPLICATION_DATA)))
764                 {
765                 SSLerr(SSL_F_DTLS1_READ_BYTES, ERR_R_INTERNAL_ERROR);
766                 return -1;
767                 }

769         /* check whether there's a handshake message (client hello?) waiting */
770         if ( (ret = have_handshake_fragment(s, type, buf, len, peek)))
771                 return ret;

773         /* Now s->d1->handshake_fragment_len == 0 if type == SSL3_RT_HANDSHAKE.

775 #ifndef OPENSSL_NO_SCTP
776         /* Continue handshake if it had to be interrupted to read
777          * app data with SCTP.
778          */
779         if ((!s->in_handshake && SSL_in_init(s)) ||
780             (BIO_dgram_is_sctp(SSL_get_rbio(s)) &&
781              (s->state == DTLS1_SCTP_ST_SR_READ_SOCK || s->state == DTLS1_SCTP_S
782              s->s3->in_read_app_data != 2))
783 #else
784         if (!s->in_handshake && SSL_in_init(s))
785 #endif
786                 {
787                 /* type == SSL3_RT_APPLICATION_DATA */
```

```
788                 i=s->handshake_func(s);
789                 if (i < 0) return(i);
790                 if (i == 0)
791                         {
792                         SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_SSL_HANDSHAKE_FAILUR
793                         return(-1);
794                         }
795                 }

797 start:
798         s->rwstate=SSL_NOTHING;

800         /* s->s3->rrec.type         - is the type of record
801          * s->s3->rrec.data,    - data
802          * s->s3->rrec.off,     - offset into 'data' for next read
803          * s->s3->rrec.length,  - number of bytes. */
804         rr = &(s->s3->rrec);

806         /* We are not handshaking and have no data yet,
807          * so process data buffered during the last handshake
808          * in advance, if any.
809          */
810         if (s->state == SSL_ST_OK && rr->length == 0)
811                 {
812                 pitem *item;
813                 item = pqueue_pop(s->d1->buffered_app_data.q);
814                 if (item)
815                         {
816 #ifndef OPENSSL_NO_SCTP
817                         /* Restore bio_dgram_sctp_rcvinfo struct */
818                         if (BIO_dgram_is_sctp(SSL_get_rbio(s)))
819                                 {
820                                 DTLS1_RECORD_DATA *rdata = (DTLS1_RECORD_DATA *)
821                                 BIO_ctrl(SSL_get_rbio(s), BIO_CTRL_DGRAM_SCTP_SE
822                                 }
823 #endif

825                         dtls1_copy_record(s, item);

827                         OPENSSL_free(item->data);
828                         pitem_free(item);
829                         }
830                 }

832         /* Check for timeout */
833         if (dtls1_handle_timeout(s) > 0)
834                 goto start;

836         /* get new packet if necessary */
837         if ((rr->length == 0) || (s->rstate == SSL_ST_READ_BODY))
838                 {
839                 ret=dtls1_get_record(s);
840                 if (ret <= 0)
841                         {
842                         ret = dtls1_read_failed(s, ret);
843                         /* anything other than a timeout is an error */
844                         if (ret <= 0)
845                                 return(ret);
846                         else
847                                 goto start;
848                         }
849                 }

851         if (s->d1->listen && rr->type != SSL3_RT_HANDSHAKE)
852                 {
853                 rr->length = 0;
```



```
854                 goto start;
855                 }

857         /* we now have a packet which can be read and processed */

859         if (s->s3->change_cipher_spec /* set when we receive ChangeCipherSpec,
860                                        * reset by ssl3_get_finished */
861                 && (rr->type != SSL3_RT_HANDSHAKE))
862                 {
863                 /* We now have application data between CCS and Finished.
864                  * Most likely the packets were reordered on their way, so
865                  * buffer the application data for later processing rather
866                  * than dropping the connection.
867                  */
868                 dtls1_buffer_record(s, &(s->d1->buffered_app_data), rr->seq_num)
869                 rr->length = 0;
870                 goto start;
871                 }

873         /* If the other end has shut down, throw anything we read away
874          * (even in 'peek' mode) */
875         if (s->shutdown & SSL_RECEIVED_SHUTDOWN)
876                 {
877                 rr->length=0;
878                 s->rwstate=SSL_NOTHING;
879                 return(0);
880                 }


883         if (type == rr->type) /* SSL3_RT_APPLICATION_DATA or SSL3_RT_HANDSHAKE *
884                 {
885                 /* make sure that we are not getting application data when we
886                  * are doing a handshake for the first time */
887                 if (SSL_in_init(s) && (type == SSL3_RT_APPLICATION_DATA) &&
888                         (s->enc_read_ctx == NULL))
889                         {
890                         al=SSL_AD_UNEXPECTED_MESSAGE;
891                         SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_APP_DATA_IN_HANDSHAK
892                         goto f_err;
893                         }

895                 if (len <= 0) return(len);

897                 if ((unsigned int)len > rr->length)
898                         n = rr->length;
899                 else
900                         n = (unsigned int)len;

902                 memcpy(buf,&(rr->data[rr->off]),n);
903                 if (!peek)
904                         {
905                         rr->length-=n;
906                         rr->off+=n;
907                         if (rr->length == 0)
908                                 {
909                                 s->rstate=SSL_ST_READ_HEADER;
910                                 rr->off=0;
911                                 }
912                         }

914 #ifndef OPENSSL_NO_SCTP
915                         /* We were about to renegotiate but had to read
916                          * belated application data first, so retry.
917                          */
918                         if (BIO_dgram_is_sctp(SSL_get_rbio(s)) &&
919                             rr->type == SSL3_RT_APPLICATION_DATA &&
```

```
920                             (s->state == DTLS1_SCTP_ST_SR_READ_SOCK || s->state
921                                 {
922                                 s->rwstate=SSL_READING;
923                                 BIO_clear_retry_flags(SSL_get_rbio(s));
924                                 BIO_set_retry_read(SSL_get_rbio(s));
925                                 }

927                         /* We might had to delay a close_notify alert because
928                          * of reordered app data. If there was an alert and ther
929                          * is no message to read anymore, finally set shutdown.
930                          */
931                         if (BIO_dgram_is_sctp(SSL_get_rbio(s)) &&
932                             s->d1->shutdown_received && !BIO_dgram_sctp_msg_wait
933                                 {
934                                 s->shutdown |= SSL_RECEIVED_SHUTDOWN;
935                                 return(0);
936                                 }
937 #endif
938                 return(n);
939                 }


942         /* If we get here, then type != rr->type; if we have a handshake
943          * message, then it was unexpected (Hello Request or Client Hello). */

945         /* In case of record types for which we have 'fragment' storage,
946          * fill that so that we can process the data at a fixed place.
947          */
948                 {
949                 unsigned int k, dest_maxlen = 0;
950                 unsigned char *dest = NULL;
951                 unsigned int *dest_len = NULL;

953                 if (rr->type == SSL3_RT_HANDSHAKE)
954                         {
955                         dest_maxlen = sizeof s->d1->handshake_fragment;
956                         dest = s->d1->handshake_fragment;
957                         dest_len = &s->d1->handshake_fragment_len;
958                         }
959                 else if (rr->type == SSL3_RT_ALERT)
960                         {
961                         dest_maxlen = sizeof(s->d1->alert_fragment);
962                         dest = s->d1->alert_fragment;
963                         dest_len = &s->d1->alert_fragment_len;
964                         }
965 #ifndef OPENSSL_NO_HEARTBEATS
966                 else if (rr->type == TLS1_RT_HEARTBEAT)
967                         {
968                         dtls1_process_heartbeat(s);

970                         /* Exit and notify application to read again */
971                         rr->length = 0;
972                         s->rwstate=SSL_READING;
973                         BIO_clear_retry_flags(SSL_get_rbio(s));
974                         BIO_set_retry_read(SSL_get_rbio(s));
975                         return(-1);
976                         }
977 #endif
978                 /* else it's a CCS message, or application data or wrong */
979                 else if (rr->type != SSL3_RT_CHANGE_CIPHER_SPEC)
980                         {
981                         /* Application data while renegotiating
982                          * is allowed. Try again reading.
983                          */
984                         if (rr->type == SSL3_RT_APPLICATION_DATA)
985                                 {
```



```
986                                 BIO *bio;
987                                 s->s3->in_read_app_data=2;
988                                 bio=SSL_get_rbio(s);
989                                 s->rwstate=SSL_READING;
990                                 BIO_clear_retry_flags(bio);
991                                 BIO_set_retry_read(bio);
992                                 return(-1);
993                                 }

995                         /* Not certain if this is the right error handling */
996                         al=SSL_AD_UNEXPECTED_MESSAGE;
997                         SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_UNEXPECTED_RECORD);
998                         goto f_err;
999                         }

1001                if (dest_maxlen > 0)
1002                        {
1003            /* XDTLS:  In a pathalogical case, the Client Hello
1004             *  may be fragmented--don't always expect dest_maxlen bytes */
1005                        if ( rr->length < dest_maxlen)
1006                                {
1007 #ifdef DTLS1_AD_MISSING_HANDSHAKE_MESSAGE
1008                                /*
1009                                 * for normal alerts rr->length is 2, while
1010                                 * dest_maxlen is 7 if we were to handle this
1011                                 * non-existing alert...
1012                                 */
1013                                FIX ME
1014 #endif
1015                                s->rstate=SSL_ST_READ_HEADER;
1016                                rr->length = 0;
1017                                goto start;
1018                                }

1020                        /* now move 'n' bytes: */
1021                        for ( k = 0; k < dest_maxlen; k++)
1022                                {
1023                                dest[k] = rr->data[rr->off++];
1024                                rr->length--;
1025                                }
1026                        *dest_len = dest_maxlen;
1027                        }
1028                }

1030        /* s->d1->handshake_fragment_len == 12  iff  rr->type == SSL3_RT_HANDSHA
1031         * s->d1->alert_fragment_len == 7      iff  rr->type == SSL3_RT_ALERT.
1032         * (Possibly rr is 'empty' now, i.e. rr->length may be 0.) */

1034        /* If we are a client, check for an incoming 'Hello Request': */
1035        if ((!s->server) &&
1036                (s->d1->handshake_fragment_len >= DTLS1_HM_HEADER_LENGTH) &&
1037                (s->d1->handshake_fragment[0] == SSL3_MT_HELLO_REQUEST) &&
1038                (s->session != NULL) && (s->session->cipher != NULL))
1039                {
1040                s->d1->handshake_fragment_len = 0;

1042                if ((s->d1->handshake_fragment[1] != 0) ||
1043                        (s->d1->handshake_fragment[2] != 0) ||
1044                        (s->d1->handshake_fragment[3] != 0))
1045                        {
1046                        al=SSL_AD_DECODE_ERROR;
1047                        SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_BAD_HELLO_REQUEST);
1048                        goto err;
1049                        }

1051                /* no need to check sequence number on HELLO REQUEST messages */
```

```
1053                if (s->msg_callback)
1054                        s->msg_callback(0, s->version, SSL3_RT_HANDSHAKE,
1055                                s->d1->handshake_fragment, 4, s, s->msg_callback

1057                if (SSL_is_init_finished(s) &&
1058                        !(s->s3->flags & SSL3_FLAGS_NO_RENEGOTIATE_CIPHERS) &&
1059                        !s->s3->renegotiate)
1060                        {
1061                        s->d1->handshake_read_seq++;
1062                        s->new_session = 1;
1063                        ssl3_renegotiate(s);
1064                        if (ssl3_renegotiate_check(s))
1065                                {
1066                                i=s->handshake_func(s);
1067                                if (i < 0) return(i);
1068                                if (i == 0)
1069                                        {
1070                                        SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_SSL_
1071                                        return(-1);
1072                                        }

1074                                if (!(s->mode & SSL_MODE_AUTO_RETRY))
1075                                        {
1076                                        if (s->s3->rbuf.left == 0) /* no read-ah
1077                                                {
1078                                                BIO *bio;
1079                                                /* In the case where we try to r
1080                                                 * but we trigger an SSL handsha
1081                                                 * the retry option set.  Otherw
1082                                                 * cause nasty problems in the b
1083                                                s->rwstate=SSL_READING;
1084                                                bio=SSL_get_rbio(s);
1085                                                BIO_clear_retry_flags(bio);
1086                                                BIO_set_retry_read(bio);
1087                                                return(-1);
1088                                                }
1089                                        }
1090                                }
1091                        }
1092                /* we either finished a handshake or ignored the request,
1093                 * now try again to obtain the (application) data we were asked
1094                goto start;
1095                }

1097        if (s->d1->alert_fragment_len >= DTLS1_AL_HEADER_LENGTH)
1098                {
1099                int alert_level = s->d1->alert_fragment[0];
1100                int alert_descr = s->d1->alert_fragment[1];

1102                s->d1->alert_fragment_len = 0;

1104                if (s->msg_callback)
1105                        s->msg_callback(0, s->version, SSL3_RT_ALERT,
1106                                s->d1->alert_fragment, 2, s, s->msg_callback_arg

1108                if (s->info_callback != NULL)
1109                        cb=s->info_callback;
1110                else if (s->ctx->info_callback != NULL)
1111                        cb=s->ctx->info_callback;

1113                if (cb != NULL)
1114                        {
1115                        j = (alert_level << 8) | alert_descr;
1116                        cb(s, SSL_CB_READ_ALERT, j);
1117                        }
```



```
1119                if (alert_level == 1) /* warning */
1120                        {
1121                        s->s3->warn_alert = alert_descr;
1122                        if (alert_descr == SSL_AD_CLOSE_NOTIFY)
1123                                {
1124 #ifndef OPENSSL_NO_SCTP
1125                                /* With SCTP and streams the socket may deliver
1126                                 * after a close_notify alert. We have to check
1127                                 * first so that nothing gets discarded.
1128                                 */
1129                                if (BIO_dgram_is_sctp(SSL_get_rbio(s)) &&
1130                                        BIO_dgram_sctp_msg_waiting(SSL_get_rbio(
1131                                        {
1132                                        s->d1->shutdown_received = 1;
1133                                        s->rwstate=SSL_READING;
1134                                        BIO_clear_retry_flags(SSL_get_rbio(s));
1135                                        BIO_set_retry_read(SSL_get_rbio(s));
1136                                        return -1;
1137                                        }
1138 #endif
1139                                s->shutdown |= SSL_RECEIVED_SHUTDOWN;
1140                                return(0);
1141                                }
1142 #if 0
1143                /* XXX: this is a possible improvement in the future */
1144                                /* now check if it's a missing record */
1145                                if (alert_descr == DTLS1_AD_MISSING_HANDSHAKE_MESSAGE)
1146                                        {
1147                                        unsigned short seq;
1148                                        unsigned int frag_off;
1149                                        unsigned char *p = &(s->d1->alert_fragment[2]);

1151                                        n2s(p, seq);
1152                                        n2l3(p, frag_off);

1154                                        dtls1_retransmit_message(s,
1155
1156
1157                                        if ( ! found  && SSL_in_init(s))
1158                                                {
1159                                                /* fprintf( stderr,"in init = %d\n", SSL
1160                                                /* requested a message not yet sent,
1161                                                   send an alert ourselves */
1162                                                ssl3_send_alert(s,SSL3_AL_WARNING,
1163                                                        DTLS1_AD_MISSING_HANDSHAKE_MESSA
1164                                                }
1165                                        }
1166 #endif
1167                                }
1168                        else if (alert_level == 2) /* fatal */
1169                                {
1170                                char tmp[16];

1172                                s->rwstate=SSL_NOTHING;
1173                                s->s3->fatal_alert = alert_descr;
1174                                SSLerr(SSL_F_DTLS1_READ_BYTES, SSL_AD_REASON_OFFSET + al
1175                                BIO_snprintf(tmp,sizeof tmp,"%d",alert_descr);
1176                                ERR_add_error_data(2,"SSL alert number ",tmp);
1177                                s->shutdown|=SSL_RECEIVED_SHUTDOWN;
1178                                SSL_CTX_remove_session(s->ctx,s->session);
1179                                return(0);
1180                                }
1181                        else
1182                                {
1183                                al=SSL_AD_ILLEGAL_PARAMETER;
```

```
1184                        SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_UNKNOWN_ALERT_TYPE);
1185                        goto f_err;
1186                        }

1188                goto start;
1189                }

1191        if (s->shutdown & SSL_SENT_SHUTDOWN) /* but we have not received a shutd
1192                {
1193                s->rwstate=SSL_NOTHING;
1194                rr->length=0;
1195                return(0);
1196                }

1198        if (rr->type == SSL3_RT_CHANGE_CIPHER_SPEC)
1199                {
1200                struct ccs_header_st ccs_hdr;
1201                unsigned int ccs_hdr_len = DTLS1_CCS_HEADER_LENGTH;

1203                dtls1_get_ccs_header(rr->data, &ccs_hdr);

1205                if (s->version == DTLS1_BAD_VER)
1206                        ccs_hdr_len = 3;

1208                /* 'Change Cipher Spec' is just a single byte, so we know
1209                 * exactly what the record payload has to look like */
1210                /* XDTLS: check that epoch is consistent */
1211                if (    (rr->length != ccs_hdr_len) ||
1212                        (rr->off != 0) || (rr->data[0] != SSL3_MT_CCS))
1213                        {
1214                        i=SSL_AD_ILLEGAL_PARAMETER;
1215                        SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_BAD_CHANGE_CIPHER_SP
1216                        goto err;
1217                        }

1219                rr->length=0;

1221                if (s->msg_callback)
1222                        s->msg_callback(0, s->version, SSL3_RT_CHANGE_CIPHER_SPE
1223                                rr->data, 1, s, s->msg_callback_arg);

1225                /* We can't process a CCS now, because previous handshake
1226                 * messages are still missing, so just drop it.
1227                 */
1228                if (!s->d1->change_cipher_spec_ok)
1229                        {
1230                        goto start;
1231                        }

1233                s->d1->change_cipher_spec_ok = 0;

1235                s->s3->change_cipher_spec=1;
1236                if (!ssl3_do_change_cipher_spec(s))
1237                        goto err;

1239                /* do this whenever CCS is processed */
1240                dtls1_reset_seq_numbers(s, SSL3_CC_READ);

1242                if (s->version == DTLS1_BAD_VER)
1243                        s->d1->handshake_read_seq++;

1245 #ifndef OPENSSL_NO_SCTP
1246                /* Remember that a CCS has been received,
1247                 * so that an old key of SCTP-Auth can be
1248                 * deleted when a CCS is sent. Will be ignored
1249                 * if no SCTP is used
```



```
1250                 */
1251                BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_AUTH_CCS_RCVD, 1,
1252 #endif

1254                goto start;
1255                }

1257        /* Unexpected handshake message (Client Hello, or protocol violation) */
1258        if ((s->d1->handshake_fragment_len >= DTLS1_HM_HEADER_LENGTH) &&
1259                !s->in_handshake)
1260                {
1261                struct hm_header_st msg_hdr;

1263                /* this may just be a stale retransmit */
1264                dtls1_get_message_header(rr->data, &msg_hdr);
1265                if( rr->epoch != s->d1->r_epoch)
1266                        {
1267                        rr->length = 0;
1268                        goto start;
1269                        }

1271                /* If we are server, we may have a repeated FINISHED of the
1272                 * client here, then retransmit our CCS and FINISHED.
1273                 */
1274                if (msg_hdr.type == SSL3_MT_FINISHED)
1275                        {
1276                        if (dtls1_check_timeout_num(s) < 0)
1277                                return -1;

1279                        dtls1_retransmit_buffered_messages(s);
1280                        rr->length = 0;
1281                        goto start;
1282                        }

1284                if (((s->state&SSL_ST_MASK) == SSL_ST_OK) &&
1285                        !(s->s3->flags & SSL3_FLAGS_NO_RENEGOTIATE_CIPHERS))
1286                        {
1287 #if 0 /* worked only because C operator preferences are not as expected (and
1288        * because this is not really needed for clients except for detecting
1289        * protocol violations): */
1290                        s->state=SSL_ST_BEFORE|(s->server)
1291                                ?SSL_ST_ACCEPT
1292                                :SSL_ST_CONNECT;
1293 #else
1294                        s->state = s->server ? SSL_ST_ACCEPT : SSL_ST_CONNECT;
1295 #endif
1296                        s->renegotiate=1;
1297                        s->new_session=1;
1298                        }
1299                i=s->handshake_func(s);
1300                if (i < 0) return(i);
1301                if (i == 0)
1302                        {
1303                        SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_SSL_HANDSHAKE_FAILUR
1304                        return(-1);
1305                        }

1307                if (!(s->mode & SSL_MODE_AUTO_RETRY))
1308                        {
1309                        if (s->s3->rbuf.left == 0) /* no read-ahead left? */
1310                                {
1311                                BIO *bio;
1312                                /* In the case where we try to read application
1313                                 * but we trigger an SSL handshake, we return -1
1314                                 * the retry option set.  Otherwise renegotiatio
1315                                 * cause nasty problems in the blocking world */
```

```
1316                                s->rwstate=SSL_READING;
1317                                bio=SSL_get_rbio(s);
1318                                BIO_clear_retry_flags(bio);
1319                                BIO_set_retry_read(bio);
1320                                return(-1);
1321                                }
1322                        }
1323                goto start;
1324                }

1326        switch (rr->type)
1327                {
1328        default:
1329 #ifndef OPENSSL_NO_TLS
1330                /* TLS just ignores unknown message types */
1331                if (s->version == TLS1_VERSION)
1332                        {
1333                        rr->length = 0;
1334                        goto start;
1335                        }
1336 #endif
1337                al=SSL_AD_UNEXPECTED_MESSAGE;
1338                SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_UNEXPECTED_RECORD);
1339                goto f_err;
1340        case SSL3_RT_CHANGE_CIPHER_SPEC:
1341        case SSL3_RT_ALERT:
1342        case SSL3_RT_HANDSHAKE:
1343                /* we already handled all of these, with the possible exception
1344                 * of SSL3_RT_HANDSHAKE when s->in_handshake is set, but that
1345                 * should not happen when type != rr->type */
1346                al=SSL_AD_UNEXPECTED_MESSAGE;
1347                SSLerr(SSL_F_DTLS1_READ_BYTES,ERR_R_INTERNAL_ERROR);
1348                goto f_err;
1349        case SSL3_RT_APPLICATION_DATA:
1350                /* At this point, we were expecting handshake data,
1351                 * but have application data.  If the library was
1352                 * running inside ssl3_read() (i.e. in_read_app_data
1353                 * is set) and it makes sense to read application data
1354                 * at this point (session renegotiation not yet started),
1355                 * we will indulge it.
1356                 */
1357                if (s->s3->in_read_app_data &&
1358                        (s->s3->total_renegotiations != 0) &&
1359                        ((
1360                                (s->state & SSL_ST_CONNECT) &&
1361                                (s->state >= SSL3_ST_CW_CLNT_HELLO_A) &&
1362                                (s->state <= SSL3_ST_CR_SRVR_HELLO_A)
1363                                ) || (
1364                                        (s->state & SSL_ST_ACCEPT) &&
1365                                        (s->state <= SSL3_ST_SW_HELLO_REQ_A) &&
1366                                        (s->state >= SSL3_ST_SR_CLNT_HELLO_A)
1367                                        )
1368                                ))
1369                        {
1370                        s->s3->in_read_app_data=2;
1371                        return(-1);
1372                        }
1373                else
1374                        {
1375                        al=SSL_AD_UNEXPECTED_MESSAGE;
1376                        SSLerr(SSL_F_DTLS1_READ_BYTES,SSL_R_UNEXPECTED_RECORD);
1377                        goto f_err;
1378                        }
1379                }
1380        /* not reached */
```



```
1382 f_err:
1383        ssl3_send_alert(s,SSL3_AL_FATAL,al);
1384 err:
1385        return(-1);
1386        }

1388 int
1389 dtls1_write_app_data_bytes(SSL *s, int type, const void *buf_, int len)
1390        {
1391        int i;

1393 #ifndef OPENSSL_NO_SCTP
1394                /* Check if we have to continue an interrupted handshake
1395                 * for reading belated app data with SCTP.
1396                 */
1397                if ((SSL_in_init(s) && !s->in_handshake) ||
1398                    (BIO_dgram_is_sctp(SSL_get_wbio(s)) &&
1399                     (s->state == DTLS1_SCTP_ST_SR_READ_SOCK || s->state == DTLS
1400 #else
1401                if (SSL_in_init(s) && !s->in_handshake)
1402 #endif
1403                {
1404                i=s->handshake_func(s);
1405                if (i < 0) return(i);
1406                if (i == 0)
1407                        {
1408                        SSLerr(SSL_F_DTLS1_WRITE_APP_DATA_BYTES,SSL_R_SSL_HANDSH
1409                        return -1;
1410                        }
1411                }

1413        if (len > SSL3_RT_MAX_PLAIN_LENGTH)
1414                {
1415                        SSLerr(SSL_F_DTLS1_WRITE_APP_DATA_BYTES,SSL_R_DTLS_MESSA
1416                        return -1;
1417                }

1419        i = dtls1_write_bytes(s, type, buf_, len);
1420        return i;
1421        }


1424        /* this only happens when a client hello is received and a handshake
1425         * is started. */
1426 static int
1427 have_handshake_fragment(SSL *s, int type, unsigned char *buf,
1428        int len, int peek)
1429        {

1431        if ((type == SSL3_RT_HANDSHAKE) && (s->d1->handshake_fragment_len > 0))
1432                /* (partially) satisfy request from storage */
1433                {
1434                unsigned char *src = s->d1->handshake_fragment;
1435                unsigned char *dst = buf;
1436                unsigned int k,n;

1438                /* peek == 0 */
1439                n = 0;
1440                while ((len > 0) && (s->d1->handshake_fragment_len > 0))
1441                        {
1442                        *dst++ = *src++;
1443                        len--; s->d1->handshake_fragment_len--;
1444                        n++;
1445                        }
1446                /* move any remaining fragment bytes: */
1447                for (k = 0; k < s->d1->handshake_fragment_len; k++)
```

```
1448                        s->d1->handshake_fragment[k] = *src++;
1449                return n;
1450                }

1452        return 0;
1453        }




1458 /* Call this to write data in records of type 'type'
1459  * It will return <= 0 if not all data has been sent or non-blocking IO.
1460  */
1461 int dtls1_write_bytes(SSL *s, int type, const void *buf, int len)
1462        {
1463        int i;

1465        OPENSSL_assert(len <= SSL3_RT_MAX_PLAIN_LENGTH);
1466        s->rwstate=SSL_NOTHING;
1467        i=do_dtls1_write(s, type, buf, len, 0);
1468        return i;
1469        }

1471 int do_dtls1_write(SSL *s, int type, const unsigned char *buf, unsigned int len,
1472        {
1473        unsigned char *p,*pseq;
1474        int i,mac_size,clear=0;
1475        int prefix_len = 0;
1476        SSL3_RECORD *wr;
1477        SSL3_BUFFER *wb;
1478        SSL_SESSION *sess;
1479        int bs;

1481        /* first check if there is a SSL3_BUFFER still being written
1482         * out.  This will happen with non blocking IO */
1483        if (s->s3->wbuf.left != 0)
1484                {
1485                OPENSSL_assert(0); /* XDTLS:  want to see if we ever get here */
1486                return(ssl3_write_pending(s,type,buf,len));
1487                }

1489        /* If we have an alert to send, lets send it */
1490        if (s->s3->alert_dispatch)
1491                {
1492                i=s->method->ssl_dispatch_alert(s);
1493                if (i <= 0)
1494                        return(i);
1495                /* if it went, fall through and send more stuff */
1496                }

1498        if (len == 0 && !create_empty_fragment)
1499                return 0;

1501        wr= &(s->s3->wrec);
1502        wb= &(s->s3->wbuf);
1503        sess=s->session;

1505        if (    (sess == NULL) ||
1506                (s->enc_write_ctx == NULL) ||
1507                (EVP_MD_CTX_md(s->write_hash) == NULL))
1508                clear=1;

1510        if (clear)
1511                mac_size=0;
1512        else
1513                {
```



```
1514                mac_size=EVP_MD_CTX_size(s->write_hash);
1515                if (mac_size < 0)
1516                        goto err;
1517                }

1519        /* DTLS implements explicit IV, so no need for empty fragments */
1520 #if 0
1521        /* 'create_empty_fragment' is true only when this function calls itself
1522        if (!clear && !create_empty_fragment && !s->s3->empty_fragment_done
1523            && SSL_version(s) != DTLS1_VERSION && SSL_version(s) != DTLS1_BAD_VE
1524                {
1525                /* countermeasure against known-IV weakness in CBC ciphersuites
1526                 * (see http://www.openssl.org/~bodo/tls-cbc.txt)
1527                 */

1529                if (s->s3->need_empty_fragments && type == SSL3_RT_APPLICATION_D
1530                        {
1531                        /* recursive function call with 'create_empty_fragment'
1532                         * this prepares and buffers the data for an empty fragm
1533                         * (these 'prefix_len' bytes are sent out later
1534                         * together with the actual payload) */
1535                        prefix_len = s->method->do_ssl_write(s, type, buf, 0, 1)
1536                        if (prefix_len <= 0)
1537                                goto err;

1539                        if (s->s3->wbuf.len < (size_t)prefix_len + SSL3_RT_MAX_P
1540                                {
1541                                /* insufficient space */
1542                                SSLerr(SSL_F_DO_DTLS1_WRITE, ERR_R_INTERNAL_ERRO
1543                                goto err;
1544                                }
1545                        }

1547                s->s3->empty_fragment_done = 1;
1548                }
1549 #endif
1550        p = wb->buf + prefix_len;

1552        /* write the header */

1554        *(p++)=type&0xff;
1555        wr->type=type;

1557        *(p++)=(s->version>>8);
1558        *(p++)=s->version&0xff;

1560        /* field where we are to write out packet epoch, seq num and len */
1561        pseq=p;
1562        p+=10;

1564        /* lets setup the record stuff. */

1566        /* Make space for the explicit IV in case of CBC.
1567         * (this is a bit of a boundary violation, but what the heck).
1568         */
1569        if ( s->enc_write_ctx &&
1570                (EVP_CIPHER_mode( s->enc_write_ctx->cipher ) & EVP_CIPH_CBC_MODE
1571                bs = EVP_CIPHER_block_size(s->enc_write_ctx->cipher);
1572        else
1573                bs = 0;

1575        wr->data=p + bs;  /* make room for IV in case of CBC */
1576        wr->length=(int)len;
1577        wr->input=(unsigned char *)buf;

1579        /* we now 'read' from wr->input, wr->length bytes into
```

```
1580          * wr->data */

1582         /* first we compress */
1583         if (s->compress != NULL)
1584                 {
1585                 if (!ssl3_do_compress(s))
1586                         {
1587                         SSLerr(SSL_F_DO_DTLS1_WRITE,SSL_R_COMPRESSION_FAILURE);
1588                         goto err;
1589                         }
1590                 }
1591         else
1592                 {
1593                 memcpy(wr->data,wr->input,wr->length);
1594                 wr->input=wr->data;
1595                 }

1597         /* we should still have the output to wr->data and the input
1598          * from wr->input.  Length should be wr->length.
1599          * wr->data still points in the wb->buf */

1601         if (mac_size != 0)
1602                 {
1603                 if(s->method->ssl3_enc->mac(s,&(p[wr->length + bs]),1) < 0)
1604                         goto err;
1605                 wr->length+=mac_size;
1606                 }

1608         /* this is true regardless of mac size */
1609         wr->input=p;
1610         wr->data=p;


1613         /* ssl3_enc can only have an error on read */
1614         if (bs) /* bs != 0 in case of CBC */
1615                 {
1616                 RAND_pseudo_bytes(p,bs);
1617                 /* master IV and last CBC residue stand for
1618                  * the rest of randomness */
1619                 wr->length += bs;
1620                 }

1622         s->method->ssl3_enc->enc(s,1);

1624         /* record length after mac and block padding */
1625 /*      if (type == SSL3_RT_APPLICATION_DATA ||
1626         (type == SSL3_RT_ALERT && ! SSL_in_init(s))) */

1628         /* there's only one epoch between handshake and app data */

1630         s2n(s->d1->w_epoch, pseq);

1632         /* XDTLS: ?? */
1633 /*      else
1634         s2n(s->d1->handshake_epoch, pseq); */

1636         memcpy(pseq, &(s->s3->write_sequence[2]), 6);
1637         pseq+=6;
1638         s2n(wr->length,pseq);

1640         /* we should now have
1641          * wr->data pointing to the encrypted data, which is
1642          * wr->length long */
1643         wr->type=type; /* not needed but helps for debugging */
1644         wr->length+=DTLS1_RT_HEADER_LENGTH;
```



```
1646 #if 0  /* this is now done at the message layer */
1647         /* buffer the record, making it easy to handle retransmits */
1648         if ( type == SSL3_RT_HANDSHAKE || type == SSL3_RT_CHANGE_CIPHER_SPEC)
1649                 dtls1_buffer_record(s, wr->data, wr->length,
1650                         *((PQ_64BIT *)&(s->s3->write_sequence[0])));
1651 #endif

1653         ssl3_record_sequence_update(&(s->s3->write_sequence[0]));

1655         if (create_empty_fragment)
1656                 {
1657                 /* we are in a recursive call;
1658                  * just return the length, don't write out anything here
1659                  */
1660                 return wr->length;
1661                 }

1663         /* now let's set up wb */
1664         wb->left = prefix_len + wr->length;
1665         wb->offset = 0;

1667         /* memorize arguments so that ssl3_write_pending can detect bad write re
1668         s->s3->wpend_tot=len;
1669         s->s3->wpend_buf=buf;
1670         s->s3->wpend_type=type;
1671         s->s3->wpend_ret=len;

1673         /* we now just need to write the buffer */
1674         return ssl3_write_pending(s,type,buf,len);
1675 err:
1676         return -1;
1677         }



1681 static int dtls1_record_replay_check(SSL *s, DTLS1_BITMAP *bitmap)
1682         {
1683         int cmp;
1684         unsigned int shift;
1685         const unsigned char *seq = s->s3->read_sequence;

1687         cmp = satsub64be(seq,bitmap->max_seq_num);
1688         if (cmp > 0)
1689                 {
1690                 memcpy (s->s3->rrec.seq_num,seq,8);
1691                 return 1; /* this record in new */
1692                 }
1693         shift = -cmp;
1694         if (shift >= sizeof(bitmap->map)*8)
1695                 return 0; /* stale, outside the window */
1696         else if (bitmap->map & (1UL<<shift))
1697                 return 0; /* record previously received */

1699         memcpy (s->s3->rrec.seq_num,seq,8);
1700         return 1;
1701         }


1704 static void dtls1_record_bitmap_update(SSL *s, DTLS1_BITMAP *bitmap)
1705         {
1706         int cmp;
1707         unsigned int shift;
1708         const unsigned char *seq = s->s3->read_sequence;

1710         cmp = satsub64be(seq,bitmap->max_seq_num);
1711         if (cmp > 0)
```

```
1712                 {
1713                 shift = cmp;
1714                 if (shift < sizeof(bitmap->map)*8)
1715                         bitmap->map <<= shift, bitmap->map |= 1UL;
1716                 else
1717                         bitmap->map = 1UL;
1718                 memcpy(bitmap->max_seq_num,seq,8);
1719                 }
1720         else    {
1721                 shift = -cmp;
1722                 if (shift < sizeof(bitmap->map)*8)
1723                         bitmap->map |= 1UL<<shift;
1724                 }
1725         }


1728 int dtls1_dispatch_alert(SSL *s)
1729         {
1730         int i,j;
1731         void (*cb)(const SSL *ssl,int type,int val)=NULL;
1732         unsigned char buf[DTLS1_AL_HEADER_LENGTH];
1733         unsigned char *ptr = &buf[0];

1735         s->s3->alert_dispatch=0;

1737         memset(buf, 0x00, sizeof(buf));
1738         *ptr++ = s->s3->send_alert[0];
1739         *ptr++ = s->s3->send_alert[1];

1741 #ifdef DTLS1_AD_MISSING_HANDSHAKE_MESSAGE
1742         if (s->s3->send_alert[1] == DTLS1_AD_MISSING_HANDSHAKE_MESSAGE)
1743                 {
1744                 s2n(s->d1->handshake_read_seq, ptr);
1745 #if 0
1746                 if ( s->d1->r_msg_hdr.frag_off == 0)  /* waiting for a new msg *

1748                 else
1749                         s2n(s->d1->r_msg_hdr.seq, ptr); /* partial msg read */
1750 #endif

1752 #if 0
1753                 fprintf(stderr, "s->d1->handshake_read_seq = %d, s->d1->r_msg_hd
1754 #endif
1755                 l2n3(s->d1->r_msg_hdr.frag_off, ptr);
1756                 }
1757 #endif

1759         i = do_dtls1_write(s, SSL3_RT_ALERT, &buf[0], sizeof(buf), 0);
1760         if (i <= 0)
1761                 {
1762                 s->s3->alert_dispatch=1;
1763                 /* fprintf( stderr, "not done with alert\n" ); */
1764                 }
1765         else
1766                 {
1767                 if (s->s3->send_alert[0] == SSL3_AL_FATAL
1768 #ifdef DTLS1_AD_MISSING_HANDSHAKE_MESSAGE
1769                     || s->s3->send_alert[1] == DTLS1_AD_MISSING_HANDSHAKE_MESSAG
1770 #endif
1771                     )
1772                         (void)BIO_flush(s->wbio);

1774                 if (s->msg_callback)
1775                         s->msg_callback(1, s->version, SSL3_RT_ALERT, s->s3->sen
1776                                 2, s, s->msg_callback_arg);
```



```
1778                 if (s->info_callback != NULL)
1779                         cb=s->info_callback;
1780                 else if (s->ctx->info_callback != NULL)
1781                         cb=s->ctx->info_callback;

1783                 if (cb != NULL)
1784                         {
1785                         j=(s->s3->send_alert[0]<<8)|s->s3->send_alert[1];
1786                         cb(s,SSL_CB_WRITE_ALERT,j);
1787                         }
1788                 }
1789         return(i);
1790         }


1793 static DTLS1_BITMAP *
1794 dtls1_get_bitmap(SSL *s, SSL3_RECORD *rr, unsigned int *is_next_epoch)
1795     {

1797     *is_next_epoch = 0;

1799     /* In current epoch, accept HM, CCS, DATA, & ALERT */
1800     if (rr->epoch == s->d1->r_epoch)
1801         return &s->d1->bitmap;

1803     /* Only HM and ALERT messages can be from the next epoch */
1804     else if (rr->epoch == (unsigned long)(s->d1->r_epoch + 1) &&
1805         (rr->type == SSL3_RT_HANDSHAKE ||
1806             rr->type == SSL3_RT_ALERT))
1807         {
1808         *is_next_epoch = 1;
1809         return &s->d1->next_bitmap;
1810         }

1812     return NULL;
1813     }

1815 #if 0
1816 static int
1817 dtls1_record_needs_buffering(SSL *s, SSL3_RECORD *rr, unsigned short *priority,
1818         unsigned long *offset)
1819         {

1821         /* alerts are passed up immediately */
1822         if ( rr->type == SSL3_RT_APPLICATION_DATA ||
1823                 rr->type == SSL3_RT_ALERT)
1824                 return 0;

1826         /* Only need to buffer if a handshake is underway.
1827          * (this implies that Hello Request and Client Hello are passed up
1828          * immediately) */
1829         if ( SSL_in_init(s))
1830                 {
1831                 unsigned char *data = rr->data;
1832                 /* need to extract the HM/CCS sequence number here */
1833                 if ( rr->type == SSL3_RT_HANDSHAKE ||
1834                         rr->type == SSL3_RT_CHANGE_CIPHER_SPEC)
1835                         {
1836                         unsigned short seq_num;
1837                         struct hm_header_st msg_hdr;
1838                         struct ccs_header_st ccs_hdr;

1840                         if ( rr->type == SSL3_RT_HANDSHAKE)
1841                                 {
1842                                 dtls1_get_message_header(data, &msg_hdr);
1843                                 seq_num = msg_hdr.seq;
```

```
1844                                *offset = msg_hdr.frag_off;
1845                                }
1846                        else
1847                                {
1848                                dtls1_get_ccs_header(data, &ccs_hdr);
1849                                seq_num = ccs_hdr.seq;
1850                                *offset = 0;
1851                                }

1853                        /* this is either a record we're waiting for, or a
1854                         * retransmit of something we happened to previously
1855                         * receive (higher layers will drop the repeat silently
1856                        if ( seq_num < s->d1->handshake_read_seq)
1857                                return 0;
1858                        if (rr->type == SSL3_RT_HANDSHAKE &&
1859                                seq_num == s->d1->handshake_read_seq &&
1860                                msg_hdr.frag_off < s->d1->r_msg_hdr.frag_off)
1861                                return 0;
1862                        else if ( seq_num == s->d1->handshake_read_seq &&
1863                                (rr->type == SSL3_RT_CHANGE_CIPHER_SPEC ||
1864                                        msg_hdr.frag_off == s->d1->r_msg_hdr.fra
1865                                return 0;
1866                        else
1867                                {
1868                                *priority = seq_num;
1869                                return 1;
1870                                }
1871                        }
1872                else /* unknown record type */
1873                        return 0;
1874                }

1876        return 0;
1877        }
1878 #endif

1880 void
1881 dtls1_reset_seq_numbers(SSL *s, int rw)
1882         {
1883         unsigned char *seq;
1884         unsigned int seq_bytes = sizeof(s->s3->read_sequence);

1886         if ( rw & SSL3_CC_READ)
1887                 {
1888                 seq = s->s3->read_sequence;
1889                 s->d1->r_epoch++;
1890                 memcpy(&(s->d1->bitmap), &(s->d1->next_bitmap), sizeof(DTLS1_BIT
1891                 memset(&(s->d1->next_bitmap), 0x00, sizeof(DTLS1_BITMAP));
1892                 }
1893         else
1894                 {
1895                 seq = s->s3->write_sequence;
1896                 memcpy(s->d1->last_write_sequence, seq, sizeof(s->s3->write_sequ
1897                 s->d1->w_epoch++;
1898                 }

1900         memset(seq, 0x00, seq_bytes);
1901         }
1902 #endif /* ! codereview */
```

```
**********************************************************
   13303 Wed Aug 13 19:53:35 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_srtp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/t1_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/*
  DTLS code by Eric Rescorla <ekr@rtfm.com>

  Copyright (C) 2006, Network Resonance, Inc.
  Copyright (C) 2011, RTFM, Inc.
*/

#include <stdio.h>
#include <openssl/objects.h>
#include "ssl_locl.h"

#ifndef OPENSSL_NO_SRTP

#include <openssl/srtp.h>


static SRTP_PROTECTION_PROFILE srtp_known_profiles[]=
```

```
128     {
129     {
130     "SRTP_AES128_CM_SHA1_80",
131     SRTP_AES128_CM_SHA1_80,
132     },
133     {
134     "SRTP_AES128_CM_SHA1_32",
135     SRTP_AES128_CM_SHA1_32,
136     },
137 #if 0
138     {
139     "SRTP_NULL_SHA1_80",
140     SRTP_NULL_SHA1_80,
141     },
142     {
143     "SRTP_NULL_SHA1_32",
144     SRTP_NULL_SHA1_32,
145     },
146 #endif
147     {0}
148     };

150 static int find_profile_by_name(char *profile_name,
151                                 SRTP_PROTECTION_PROFILE **pptr,unsigned len)
152         {
153         SRTP_PROTECTION_PROFILE *p;

155         p=srtp_known_profiles;
156         while(p->name)
157                 {
158                 if((len == strlen(p->name)) && !strncmp(p->name,profile_name,
159                                                         len))
160                         {
161                         *pptr=p;
162                         return 0;
163                         }

165                 p++;
166                 }

168         return 1;
169         }

171 static int find_profile_by_num(unsigned profile_num,
172                                SRTP_PROTECTION_PROFILE **pptr)
173         {
174         SRTP_PROTECTION_PROFILE *p;

176         p=srtp_known_profiles;
177         while(p->name)
178                 {
179                 if(p->id == profile_num)
180                         {
181                         *pptr=p;
182                         return 0;
183                         }
184                 p++;
185                 }

187         return 1;
188         }

190 static int ssl_ctx_make_profiles(const char *profiles_string,STACK_OF(SRTP_PROTE
191         {
192         STACK_OF(SRTP_PROTECTION_PROFILE) *profiles;
```



```
194         char *col;
195         char *ptr=(char *)profiles_string;

197         SRTP_PROTECTION_PROFILE *p;

199         if(!(profiles=sk_SRTP_PROTECTION_PROFILE_new_null()))
200                 {
201                 SSLerr(SSL_F_SSL_CTX_MAKE_PROFILES, SSL_R_SRTP_COULD_NOT_ALLOCAT
202                 return 1;
203                 }

205         do
206                 {
207                 col=strchr(ptr,':');

209                 if(!find_profile_by_name(ptr,&p,
210                                          col ? col-ptr : (int)strlen(ptr)))
211                         {
212                         sk_SRTP_PROTECTION_PROFILE_push(profiles,p);
213                         }
214                 else
215                         {
216                         SSLerr(SSL_F_SSL_CTX_MAKE_PROFILES,SSL_R_SRTP_UNKNOWN_PR
217                         return 1;
218                         }

220                 if(col) ptr=col+1;
221                 } while (col);

223         *out=profiles;

225         return 0;
226         }

228 int SSL_CTX_set_tlsext_use_srtp(SSL_CTX *ctx,const char *profiles)
229         {
230         return ssl_ctx_make_profiles(profiles,&ctx->srtp_profiles);
231         }

233 int SSL_set_tlsext_use_srtp(SSL *s,const char *profiles)
234         {
235         return ssl_ctx_make_profiles(profiles,&s->srtp_profiles);
236         }


239 STACK_OF(SRTP_PROTECTION_PROFILE) *SSL_get_srtp_profiles(SSL *s)
240         {
241         if(s != NULL)
242                 {
243                 if(s->srtp_profiles != NULL)
244                         {
245                         return s->srtp_profiles;
246                         }
247                 else if((s->ctx != NULL) &&
248                         (s->ctx->srtp_profiles != NULL))
249                         {
250                         return s->ctx->srtp_profiles;
251                         }
252                 }

254         return NULL;
255         }

257 SRTP_PROTECTION_PROFILE *SSL_get_selected_srtp_profile(SSL *s)
258         {
259         return s->srtp_profile;
```

```
260         }

262 /* Note: this function returns 0 length if there are no
263    profiles specified */
264 int ssl_add_clienthello_use_srtp_ext(SSL *s, unsigned char *p, int *len, int max
265         {
266         int ct=0;
267         int i;
268         STACK_OF(SRTP_PROTECTION_PROFILE) *clnt=0;
269         SRTP_PROTECTION_PROFILE *prof;

271         clnt=SSL_get_srtp_profiles(s);
272         ct=sk_SRTP_PROTECTION_PROFILE_num(clnt); /* -1 if clnt == 0 */

274         if(p)
275                 {
276                 if(ct==0)
277                         {
278                         SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_USE_SRTP_EXT,SSL_R_EMPT
279                         return 1;
280                         }

282                 if((2 + ct*2 + 1) > maxlen)
283                         {
284                         SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_USE_SRTP_EXT,SSL_R_SRTP
285                         return 1;
286                         }

288                 /* Add the length */
289                 s2n(ct * 2, p);
290                 for(i=0;i<ct;i++)
291                         {
292                         prof=sk_SRTP_PROTECTION_PROFILE_value(clnt,i);
293                         s2n(prof->id,p);
294                         }

296                 /* Add an empty use_mki value */
297                 *p++ = 0;
298                 }

300         *len=2 + ct*2 + 1;

302         return 0;
303         }


306 int ssl_parse_clienthello_use_srtp_ext(SSL *s, unsigned char *d, int len,int *al
307         {
308         SRTP_PROTECTION_PROFILE *cprof,*sprof;
309         STACK_OF(SRTP_PROTECTION_PROFILE) *clnt=0,*srvr;
310         int ct;
311         int mki_len;
312         int i,j;
313         int id;
314         int ret;

316          /* Length value + the MKI length */
317         if(len < 3)
318                 {
319                 SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_P
320                 *al=SSL_AD_DECODE_ERROR;
321                 return 1;
322                 }

324         /* Pull off the length of the cipher suite list */
325         n2s(d, ct);
```



```
326         len -= 2;

328         /* Check that it is even */
329         if(ct%2)
330                 {
331                 SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_P
332                 *al=SSL_AD_DECODE_ERROR;
333                 return 1;
334                 }

336         /* Check that lengths are consistent */
337         if(len < (ct + 1))
338                 {
339                 SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_P
340                 *al=SSL_AD_DECODE_ERROR;
341                 return 1;
342                 }


345         clnt=sk_SRTP_PROTECTION_PROFILE_new_null();

347         while(ct)
348                 {
349                 n2s(d,id);
350                 ct-=2;
351                 len-=2;

353                 if(!find_profile_by_num(id,&cprof))
354                         {
355                         sk_SRTP_PROTECTION_PROFILE_push(clnt,cprof);
356                         }
357                 else
358                         {
359                         ; /* Ignore */
360                         }
361                 }

363         /* Now extract the MKI value as a sanity check, but discard it for now *
364         mki_len = *d;
365         d++; len--;

367         if (mki_len != len)
368                 {
369                 SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_M
370                 *al=SSL_AD_DECODE_ERROR;
371                 return 1;
372                 }

374         srvr=SSL_get_srtp_profiles(s);

376         /* Pick our most preferred profile. If no profiles have been
377          configured then the outer loop doesn't run
378          (sk_SRTP_PROTECTION_PROFILE_num() = -1)
379          and so we just return without doing anything */
380         for(i=0;i<sk_SRTP_PROTECTION_PROFILE_num(srvr);i++)
381                 {
382                 sprof=sk_SRTP_PROTECTION_PROFILE_value(srvr,i);

384                 for(j=0;j<sk_SRTP_PROTECTION_PROFILE_num(clnt);j++)
385                         {
386                         cprof=sk_SRTP_PROTECTION_PROFILE_value(clnt,j);

388                         if(cprof->id==sprof->id)
389                                 {
390                                 s->srtp_profile=sprof;
391                                 *al=0;
```

```
392                                 ret=0;
393                                 goto done;
394                                 }
395                         }
396                 }

398         ret=0;

400 done:
401         if(clnt) sk_SRTP_PROTECTION_PROFILE_free(clnt);

403         return ret;
404         }

406 int ssl_add_serverhello_use_srtp_ext(SSL *s, unsigned char *p, int *len, int max
407         {
408         if(p)
409                 {
410                 if(maxlen < 5)
411                         {
412                         SSLerr(SSL_F_SSL_ADD_SERVERHELLO_USE_SRTP_EXT,SSL_R_SRTP
413                         return 1;
414                         }

416                 if(s->srtp_profile==0)
417                         {
418                         SSLerr(SSL_F_SSL_ADD_SERVERHELLO_USE_SRTP_EXT,SSL_R_USE_
419                         return 1;
420                         }
421                 s2n(2, p);
422                 s2n(s->srtp_profile->id,p);
423                 *p++ = 0;
424                 }
425         *len=5;

427         return 0;
428         }


431 int ssl_parse_serverhello_use_srtp_ext(SSL *s, unsigned char *d, int len,int *al
432         {
433         unsigned id;
434         int i;
435         int ct;

437         STACK_OF(SRTP_PROTECTION_PROFILE) *clnt;
438         SRTP_PROTECTION_PROFILE *prof;

440         if(len!=5)
441                 {
442                 SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_P
443                 *al=SSL_AD_DECODE_ERROR;
444                 return 1;
445                 }

447         n2s(d, ct);
448         if(ct!=2)
449                 {
450                 SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_P
451                 *al=SSL_AD_DECODE_ERROR;
452                 return 1;
453                 }

455         n2s(d,id);
456         if (*d)  /* Must be no MKI, since we never offer one */
457                 {
```



```
458                 SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_M
459                 *al=SSL_AD_ILLEGAL_PARAMETER;
460                 return 1;
461                 }

463         clnt=SSL_get_srtp_profiles(s);

465         /* Throw an error if the server gave us an unsolicited extension */
466         if (clnt == NULL)
467                 {
468                 SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_USE_SRTP_EXT,SSL_R_NO_SRTP_PR
469                 *al=SSL_AD_DECODE_ERROR;
470                 return 1;
471                 }

473         /* Check to see if the server gave us something we support
474            (and presumably offered)
475         */
476         for(i=0;i<sk_SRTP_PROTECTION_PROFILE_num(clnt);i++)
477                 {
478                 prof=sk_SRTP_PROTECTION_PROFILE_value(clnt,i);

480                 if(prof->id == id)
481                         {
482                         s->srtp_profile=prof;
483                         *al=0;
484                         return 0;
485                         }
486                 }

488         SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_USE_SRTP_EXT,SSL_R_BAD_SRTP_PROTECTIO
489         *al=SSL_AD_DECODE_ERROR;
490         return 1;
491         }


494 #endif
495 #endif /* ! codereview */
```

```
**********************************************************
   45121 Wed Aug 13 19:53:35 2014
new/usr/src/lib/openssl/libsunw_ssl/d1_srvr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/d1_srvr.c */
/*
 * DTLS implementation written by Nagendra Modadugu
 * (nagendra@cs.stanford.edu) for the OpenSSL project 2005.
 */
/* ====================================================================
 * Copyright (c) 1999-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
```



```
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/md5.h>
#include <openssl/bn.h>
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
```

```
129 static const SSL_METHOD *dtls1_get_server_method(int ver);
130 static int dtls1_send_hello_verify_request(SSL *s);

132 static const SSL_METHOD *dtls1_get_server_method(int ver)
133         {
134         if (ver == DTLS1_VERSION)
135                 return(DTLSv1_server_method());
136         else
137                 return(NULL);
138         }

140 IMPLEMENT_dtls1_meth_func(DTLSv1_server_method,
141                           dtls1_accept,
142                           ssl_undefined_function,
143                           dtls1_get_server_method)

145 int dtls1_accept(SSL *s)
146         {
147         BUF_MEM *buf;
148         unsigned long Time=(unsigned long)time(NULL);
149         void (*cb)(const SSL *ssl,int type,int val)=NULL;
150         unsigned long alg_k;
151         int ret= -1;
152         int new_state,state,skip=0;
153         int listen;
154 #ifndef OPENSSL_NO_SCTP
155         unsigned char sctpauthkey[64];
156         char labelbuffer[sizeof(DTLS1_SCTP_AUTH_LABEL)];
157 #endif

159         RAND_add(&Time,sizeof(Time),0);
160         ERR_clear_error();
161         clear_sys_error();

163         if (s->info_callback != NULL)
164                 cb=s->info_callback;
165         else if (s->ctx->info_callback != NULL)
166                 cb=s->ctx->info_callback;

168         listen = s->d1->listen;

170         /* init things to blank */
171         s->in_handshake++;
172         if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

174         s->d1->listen = listen;
175 #ifndef OPENSSL_NO_SCTP
176         /* Notify SCTP BIO socket to enter handshake
177          * mode and prevent stream identifier other
178          * than 0. Will be ignored if no SCTP is used.
179          */
180         BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_SET_IN_HANDSHAKE, s->in_ha
181 #endif

183         if (s->cert == NULL)
184                 {
185                 SSLerr(SSL_F_DTLS1_ACCEPT,SSL_R_NO_CERTIFICATE_SET);
186                 return(-1);
187                 }

189 #ifndef OPENSSL_NO_HEARTBEATS
190         /* If we're awaiting a HeartbeatResponse, pretend we
191          * already got and don't await it anymore, because
192          * Heartbeats don't make sense during handshakes anyway.
193          */
```



```
194         if (s->tlsext_hb_pending)
195                 {
196                 dtls1_stop_timer(s);
197                 s->tlsext_hb_pending = 0;
198                 s->tlsext_hb_seq++;
199                 }
200 #endif

202         for (;;)
203                 {
204                 state=s->state;

206                 switch (s->state)
207                         {
208                 case SSL_ST_RENEGOTIATE:
209                         s->renegotiate=1;
210                         /* s->state=SSL_ST_ACCEPT; */

212                 case SSL_ST_BEFORE:
213                 case SSL_ST_ACCEPT:
214                 case SSL_ST_BEFORE|SSL_ST_ACCEPT:
215                 case SSL_ST_OK|SSL_ST_ACCEPT:

217                         s->server=1;
218                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

220                         if ((s->version & 0xff00) != (DTLS1_VERSION & 0xff00))
221                                 {
222                                 SSLerr(SSL_F_DTLS1_ACCEPT, ERR_R_INTERNAL_ERROR)
223                                 return -1;
224                                 }
225                         s->type=SSL_ST_ACCEPT;

227                         if (s->init_buf == NULL)
228                                 {
229                                 if ((buf=BUF_MEM_new()) == NULL)
230                                         {
231                                         ret= -1;
232                                         goto end;
233                                         }
234                                 if (!BUF_MEM_grow(buf,SSL3_RT_MAX_PLAIN_LENGTH))
235                                         {
236                                         ret= -1;
237                                         goto end;
238                                         }
239                                 s->init_buf=buf;
240                                 }

242                         if (!ssl3_setup_buffers(s))
243                                 {
244                                 ret= -1;
245                                 goto end;
246                                 }

248                         s->init_num=0;

250                         if (s->state != SSL_ST_RENEGOTIATE)
251                                 {
252                                 /* Ok, we now need to push on a buffering BIO so
253                                  * the output is sent in a way that TCP likes :-
254                                  * ...but not with SCTP :-)
255                                  */
256 #ifndef OPENSSL_NO_SCTP
257                                 if (!BIO_dgram_is_sctp(SSL_get_wbio(s)))
258 #endif
259                                         if (!ssl_init_wbio_buffer(s,1)) { ret= -
```

```
261                             ssl3_init_finished_mac(s);
262                             s->state=SSL3_ST_SR_CLNT_HELLO_A;
263                             s->ctx->stats.sess_accept++;
264                             }
265                     else
266                             {
267                             /* s->state == SSL_ST_RENEGOTIATE,
268                              * we will just send a HelloRequest */
269                             s->ctx->stats.sess_accept_renegotiate++;
270                             s->state=SSL3_ST_SW_HELLO_REQ_A;
271                             }

273                     break;

275             case SSL3_ST_SW_HELLO_REQ_A:
276             case SSL3_ST_SW_HELLO_REQ_B:

278                     s->shutdown=0;
279                     dtls1_clear_record_buffer(s);
280                     dtls1_start_timer(s);
281                     ret=dtls1_send_hello_request(s);
282                     if (ret <= 0) goto end;
283                     s->s3->tmp.next_state=SSL3_ST_SR_CLNT_HELLO_A;
284                     s->state=SSL3_ST_SW_FLUSH;
285                     s->init_num=0;

287                     ssl3_init_finished_mac(s);
288                     break;

290             case SSL3_ST_SW_HELLO_REQ_C:
291                     s->state=SSL_ST_OK;
292                     break;

294             case SSL3_ST_SR_CLNT_HELLO_A:
295             case SSL3_ST_SR_CLNT_HELLO_B:
296             case SSL3_ST_SR_CLNT_HELLO_C:

298                     s->shutdown=0;
299                     ret=ssl3_get_client_hello(s);
300                     if (ret <= 0) goto end;
301                     dtls1_stop_timer(s);

303                     if (ret == 1 && (SSL_get_options(s) & SSL_OP_COOKIE_EXCH
304                             s->state = DTLS1_ST_SW_HELLO_VERIFY_REQUEST_A;
305                     else
306                             s->state = SSL3_ST_SW_SRVR_HELLO_A;

308                     s->init_num=0;

310                     /* Reflect ClientHello sequence to remain stateless whil
311                     if (listen)
312                             {
313                             memcpy(s->s3->write_sequence, s->s3->read_sequen
314                             }

316                     /* If we're just listening, stop here */
317                     if (listen && s->state == SSL3_ST_SW_SRVR_HELLO_A)
318                             {
319                             ret = 2;
320                             s->d1->listen = 0;
321                             /* Set expected sequence numbers
322                              * to continue the handshake.
323                              */
324                             s->d1->handshake_read_seq = 2;
325                             s->d1->handshake_write_seq = 1;
```



```
326                             s->d1->next_handshake_write_seq = 1;
327                             goto end;
328                             }

330                     break;

332             case DTLS1_ST_SW_HELLO_VERIFY_REQUEST_A:
333             case DTLS1_ST_SW_HELLO_VERIFY_REQUEST_B:

335                     ret = dtls1_send_hello_verify_request(s);
336                     if ( ret <= 0) goto end;
337                     s->state=SSL3_ST_SW_FLUSH;
338                     s->s3->tmp.next_state=SSL3_ST_SR_CLNT_HELLO_A;

340                     /* HelloVerifyRequest resets Finished MAC */
341                     if (s->version != DTLS1_BAD_VER)
342                             ssl3_init_finished_mac(s);
343                     break;

345 #ifndef OPENSSL_NO_SCTP
346             case DTLS1_SCTP_ST_SR_READ_SOCK:

348                     if (BIO_dgram_sctp_msg_waiting(SSL_get_rbio(s)))
349                             {
350                             s->s3->in_read_app_data=2;
351                             s->rwstate=SSL_READING;
352                             BIO_clear_retry_flags(SSL_get_rbio(s));
353                             BIO_set_retry_read(SSL_get_rbio(s));
354                             ret = -1;
355                             goto end;
356                             }

358                     s->state=SSL3_ST_SR_FINISHED_A;
359                     break;

361             case DTLS1_SCTP_ST_SW_WRITE_SOCK:
362                     ret = BIO_dgram_sctp_wait_for_dry(SSL_get_wbio(s));
363                     if (ret < 0) goto end;

365                     if (ret == 0)
366                             {
367                             if (s->d1->next_state != SSL_ST_OK)
368                                     {
369                                     s->s3->in_read_app_data=2;
370                                     s->rwstate=SSL_READING;
371                                     BIO_clear_retry_flags(SSL_get_rbio(s));
372                                     BIO_set_retry_read(SSL_get_rbio(s));
373                                     ret = -1;
374                                     goto end;
375                                     }
376                             }

378                     s->state=s->d1->next_state;
379                     break;
380 #endif

382             case SSL3_ST_SW_SRVR_HELLO_A:
383             case SSL3_ST_SW_SRVR_HELLO_B:
384                     s->renegotiate = 2;
385                     dtls1_start_timer(s);
386                     ret=dtls1_send_server_hello(s);
387                     if (ret <= 0) goto end;

389                     if (s->hit)
390                             {
391 #ifndef OPENSSL_NO_SCTP
```

```
392                                 /* Add new shared key for SCTP-Auth,
393                                  * will be ignored if no SCTP used.
394                                  */
395                                 snprintf((char*) labelbuffer, sizeof(DTLS1_SCTP_
396                                         DTLS1_SCTP_AUTH_LABEL);

398                                 SSL_export_keying_material(s, sctpauthkey,
399                                                            sizeof(sctpauthkey),
400                                                            sizeof(labelbuffer),

402                                 BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_AD
403                              sizeof(sctpauthkey), sctpauthkey);
404 #endif
405 #ifndef OPENSSL_NO_TLSEXT
406                                 if (s->tlsext_ticket_expected)
407                                         s->state=SSL3_ST_SW_SESSION_TICKET_A;
408                                 else
409                                         s->state=SSL3_ST_SW_CHANGE_A;
410 #else
411                                 s->state=SSL3_ST_SW_CHANGE_A;
412 #endif
413                                 }
414                         else
415                                 s->state=SSL3_ST_SW_CERT_A;
416                         s->init_num=0;
417                         break;

419                 case SSL3_ST_SW_CERT_A:
420                 case SSL3_ST_SW_CERT_B:
421                         /* Check if it is anon DH or normal PSK */
422                         if (!(s->s3->tmp.new_cipher->algorithm_auth & SSL_aNULL)
423                                 && !(s->s3->tmp.new_cipher->algorithm_mkey & SSL
424                                 {
425                                 dtls1_start_timer(s);
426                                 ret=dtls1_send_server_certificate(s);
427                                 if (ret <= 0) goto end;
428 #ifndef OPENSSL_NO_TLSEXT
429                                 if (s->tlsext_status_expected)
430                                         s->state=SSL3_ST_SW_CERT_STATUS_A;
431                                 else
432                                         s->state=SSL3_ST_SW_KEY_EXCH_A;
433                                 }
434                         else
435                                 {
436                                 skip = 1;
437                                 s->state=SSL3_ST_SW_KEY_EXCH_A;
438                                 }
439 #else
440                                 }
441                         else
442                                 skip=1;

444                         s->state=SSL3_ST_SW_KEY_EXCH_A;
445 #endif
446                         s->init_num=0;
447                         break;

449                 case SSL3_ST_SW_KEY_EXCH_A:
450                 case SSL3_ST_SW_KEY_EXCH_B:
451                         alg_k = s->s3->tmp.new_cipher->algorithm_mkey;

453                         /* clear this, it may get reset by
454                          * send_server_key_exchange */
455                         if ((s->options & SSL_OP_EPHEMERAL_RSA)
456 #ifndef OPENSSL_NO_KRB5
457                                 && !(alg_k & SSL_kKRB5)
```



```
458 #endif /* OPENSSL_NO_KRB5 */
459                                 )
460                                 /* option SSL_OP_EPHEMERAL_RSA sends temporary R
461                                  * even when forbidden by protocol specs
462                                  * (handshake may fail as clients are not requir
463                                  * be able to handle this) */
464                                 s->s3->tmp.use_rsa_tmp=1;
465                         else
466                                 s->s3->tmp.use_rsa_tmp=0;

468                         /* only send if a DH key exchange or
469                          * RSA but we have a sign only certificate */
470                         if (s->s3->tmp.use_rsa_tmp
471                         /* PSK: send ServerKeyExchange if PSK identity
472                          * hint if provided */
473 #ifndef OPENSSL_NO_PSK
474                             || ((alg_k & SSL_kPSK) && s->ctx->psk_identity_hint)
475 #endif
476                             || (alg_k & (SSL_kEDH|SSL_kDHr|SSL_kDHd))
477                             || (alg_k & SSL_kEECDH)
478                             || ((alg_k & SSL_kRSA)
479                                 && (s->cert->pkeys[SSL_PKEY_RSA_ENC].privatekey
480                                     || (SSL_C_IS_EXPORT(s->s3->tmp.new_cipher)
481                                         && EVP_PKEY_size(s->cert->pkeys[SSL_PKEY
482                                         )
483                                     )
484                                 )
485                             )
486                                 {
487                                 dtls1_start_timer(s);
488                                 ret=dtls1_send_server_key_exchange(s);
489                                 if (ret <= 0) goto end;
490                                 }
491                         else
492                                 skip=1;

494                         s->state=SSL3_ST_SW_CERT_REQ_A;
495                         s->init_num=0;
496                         break;

498                 case SSL3_ST_SW_CERT_REQ_A:
499                 case SSL3_ST_SW_CERT_REQ_B:
500                         if (/* don't request cert unless asked for it: */
501                                 !(s->verify_mode & SSL_VERIFY_PEER) ||
502                                 /* if SSL_VERIFY_CLIENT_ONCE is set,
503                                  * don't request cert during re-negotiation: */
504                                 ((s->session->peer != NULL) &&
505                                  (s->verify_mode & SSL_VERIFY_CLIENT_ONCE)) ||
506                                 /* never request cert in anonymous ciphersuites
507                                  * (see section "Certificate request" in SSL 3 d
508                                  * and in RFC 2246): */
509                                 ((s->s3->tmp.new_cipher->algorithm_auth & SSL_aN
510                                  /* ... except when the application insists on v
511                                   * (against the specs, but s3_clnt.c accepts th
512                                  !(s->verify_mode & SSL_VERIFY_FAIL_IF_NO_PEER_C
513                                  /* never request cert in Kerberos ciphersuites
514                                 (s->s3->tmp.new_cipher->algorithm_auth & SSL_aKR
515                                 /* With normal PSK Certificates and
516                                  * Certificate Requests are omitted */
517                                 || (s->s3->tmp.new_cipher->algorithm_mkey & SSL_
518                                 {
519                                 /* no cert request */
520                                 skip=1;
521                                 s->s3->tmp.cert_request=0;
522                                 s->state=SSL3_ST_SW_SRVR_DONE_A;
523 #ifndef OPENSSL_NO_SCTP
```

```
524                                 if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
525                                         {
526                                         s->d1->next_state = SSL3_ST_SW_SRVR_DONE
527                                         s->state = DTLS1_SCTP_ST_SW_WRITE_SOCK;
528                                         }
529 #endif
530                                 }
531                         else
532                                 {
533                                 s->s3->tmp.cert_request=1;
534                                 dtls1_start_timer(s);
535                                 ret=dtls1_send_certificate_request(s);
536                                 if (ret <= 0) goto end;
537 #ifndef NETSCAPE_HANG_BUG
538                                 s->state=SSL3_ST_SW_SRVR_DONE_A;
539 #ifndef OPENSSL_NO_SCTP
540                                 if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
541                                         {
542                                         s->d1->next_state = SSL3_ST_SW_SRVR_DONE
543                                         s->state = DTLS1_SCTP_ST_SW_WRITE_SOCK;
544                                         }
545 #endif
546 #else
547                                 s->state=SSL3_ST_SW_FLUSH;
548                                 s->s3->tmp.next_state=SSL3_ST_SR_CERT_A;
549 #ifndef OPENSSL_NO_SCTP
550                                 if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
551                                         {
552                                         s->d1->next_state = s->s3->tmp.next_stat
553                                         s->s3->tmp.next_state=DTLS1_SCTP_ST_SW_W
554                                         }
555 #endif
556 #endif
557                                 s->init_num=0;
558                                 }
559                         break;

561                 case SSL3_ST_SW_SRVR_DONE_A:
562                 case SSL3_ST_SW_SRVR_DONE_B:
563                         dtls1_start_timer(s);
564                         ret=dtls1_send_server_done(s);
565                         if (ret <= 0) goto end;
566                         s->s3->tmp.next_state=SSL3_ST_SR_CERT_A;
567                         s->state=SSL3_ST_SW_FLUSH;
568                         s->init_num=0;
569                         break;

571                 case SSL3_ST_SW_FLUSH:
572                         s->rwstate=SSL_WRITING;
573                         if (BIO_flush(s->wbio) <= 0)
574                                 {
575                                 /* If the write error was fatal, stop trying */
576                                 if (!BIO_should_retry(s->wbio))
577                                         {
578                                         s->rwstate=SSL_NOTHING;
579                                         s->state=s->s3->tmp.next_state;
580                                         }

582                                 ret= -1;
583                                 goto end;
584                                 }
585                         s->rwstate=SSL_NOTHING;
586                         s->state=s->s3->tmp.next_state;
587                         break;

589                 case SSL3_ST_SR_CERT_A:
```



```
590                 case SSL3_ST_SR_CERT_B:
591                         /* Check for second client hello (MS SGC) */
592                         ret = ssl3_check_client_hello(s);
593                         if (ret <= 0)
594                                 goto end;
595                         if (ret == 2)
596                                 {
597                                 dtls1_stop_timer(s);
598                                 s->state = SSL3_ST_SR_CLNT_HELLO_C;
599                                 }
600                         else {
601                                 if (s->s3->tmp.cert_request)
602                                         {
603                                 ret=ssl3_get_client_certificate(s);
604                                 if (ret <= 0) goto end;
605                                         }
606                                 s->init_num=0;
607                                 s->state=SSL3_ST_SR_KEY_EXCH_A;
608                         }
609                         break;

611                 case SSL3_ST_SR_KEY_EXCH_A:
612                 case SSL3_ST_SR_KEY_EXCH_B:
613                         ret=ssl3_get_client_key_exchange(s);
614                         if (ret <= 0) goto end;
615 #ifndef OPENSSL_NO_SCTP
616                         /* Add new shared key for SCTP-Auth,
617                          * will be ignored if no SCTP used.
618                          */
619                         snprintf((char *) labelbuffer, sizeof(DTLS1_SCTP_AUTH_LA
620                                  DTLS1_SCTP_AUTH_LABEL);

622                         SSL_export_keying_material(s, sctpauthkey,
623                                                    sizeof(sctpauthkey), labelbuf
624                                                    sizeof(labelbuffer), NULL, 0,

626                         BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_ADD_AUTH_K
627                                  sizeof(sctpauthkey), sctpauthkey);
628 #endif

630                         s->state=SSL3_ST_SR_CERT_VRFY_A;
631                         s->init_num=0;

633                         if (ret == 2)
634                                 {
635                                 /* For the ECDH ciphersuites when
636                                  * the client sends its ECDH pub key in
637                                  * a certificate, the CertificateVerify
638                                  * message is not sent.
639                                  */
640                                 s->state=SSL3_ST_SR_FINISHED_A;
641                                 s->init_num = 0;
642                                 }
643                         else
644                                 {
645                                 s->state=SSL3_ST_SR_CERT_VRFY_A;
646                                 s->init_num=0;

648                                 /* We need to get hashes here so if there is
649                                  * a client cert, it can be verified */
650                                 s->method->ssl3_enc->cert_verify_mac(s,
651                                         NID_md5,
652                                         &(s->s3->tmp.cert_verify_md[0]));
653                                 s->method->ssl3_enc->cert_verify_mac(s,
654                                         NID_sha1,
655                                         &(s->s3->tmp.cert_verify_md[MD5_DIGEST_L
```

```
656                             }
657                         break;

659                 case SSL3_ST_SR_CERT_VRFY_A:
660                 case SSL3_ST_SR_CERT_VRFY_B:

662                         s->d1->change_cipher_spec_ok = 1;
663                         /* we should decide if we expected this one */
664                         ret=ssl3_get_cert_verify(s);
665                         if (ret <= 0) goto end;
666 #ifndef OPENSSL_NO_SCTP
667                         if (BIO_dgram_is_sctp(SSL_get_wbio(s)) &&
668                             state == SSL_ST_RENEGOTIATE)
669                                 s->state=DTLS1_SCTP_ST_SR_READ_SOCK;
670                         else
671 #endif
672                                 s->state=SSL3_ST_SR_FINISHED_A;
673                         s->init_num=0;
674                         break;

676                 case SSL3_ST_SR_FINISHED_A:
677                 case SSL3_ST_SR_FINISHED_B:
678                         s->d1->change_cipher_spec_ok = 1;
679                         ret=ssl3_get_finished(s,SSL3_ST_SR_FINISHED_A,
680                                 SSL3_ST_SR_FINISHED_B);
681                         if (ret <= 0) goto end;
682                         dtls1_stop_timer(s);
683                         if (s->hit)
684                                 s->state=SSL_ST_OK;
685 #ifndef OPENSSL_NO_TLSEXT
686                         else if (s->tlsext_ticket_expected)
687                                 s->state=SSL3_ST_SW_SESSION_TICKET_A;
688 #endif
689                         else
690                                 s->state=SSL3_ST_SW_CHANGE_A;
691                         s->init_num=0;
692                         break;

694 #ifndef OPENSSL_NO_TLSEXT
695                 case SSL3_ST_SW_SESSION_TICKET_A:
696                 case SSL3_ST_SW_SESSION_TICKET_B:
697                         ret=dtls1_send_newsession_ticket(s);
698                         if (ret <= 0) goto end;
699                         s->state=SSL3_ST_SW_CHANGE_A;
700                         s->init_num=0;
701                         break;

703                 case SSL3_ST_SW_CERT_STATUS_A:
704                 case SSL3_ST_SW_CERT_STATUS_B:
705                         ret=ssl3_send_cert_status(s);
706                         if (ret <= 0) goto end;
707                         s->state=SSL3_ST_SW_KEY_EXCH_A;
708                         s->init_num=0;
709                         break;

711 #endif

713                 case SSL3_ST_SW_CHANGE_A:
714                 case SSL3_ST_SW_CHANGE_B:

716                         s->session->cipher=s->s3->tmp.new_cipher;
717                         if (!s->method->ssl3_enc->setup_key_block(s))
718                                 { ret= -1; goto end; }

720                         ret=dtls1_send_change_cipher_spec(s,
721                                 SSL3_ST_SW_CHANGE_A,SSL3_ST_SW_CHANGE_B);
```



```
723                         if (ret <= 0) goto end;

725 #ifndef OPENSSL_NO_SCTP
726                         if (!s->hit)
727                                 {
728                                 /* Change to new shared key of SCTP-Auth,
729                                  * will be ignored if no SCTP used.
730                                  */
731                                 BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_NE
732                                 }
733 #endif

735                         s->state=SSL3_ST_SW_FINISHED_A;
736                         s->init_num=0;

738                         if (!s->method->ssl3_enc->change_cipher_state(s,
739                                 SSL3_CHANGE_CIPHER_SERVER_WRITE))
740                                 {
741                                 ret= -1;
742                                 goto end;
743                                 }

745                         dtls1_reset_seq_numbers(s, SSL3_CC_WRITE);
746                         break;

748                 case SSL3_ST_SW_FINISHED_A:
749                 case SSL3_ST_SW_FINISHED_B:
750                         ret=dtls1_send_finished(s,
751                                 SSL3_ST_SW_FINISHED_A,SSL3_ST_SW_FINISHED_B,
752                                 s->method->ssl3_enc->server_finished_label,
753                                 s->method->ssl3_enc->server_finished_label_len);
754                         if (ret <= 0) goto end;
755                         s->state=SSL3_ST_SW_FLUSH;
756                         if (s->hit)
757                                 {
758                                 s->s3->tmp.next_state=SSL3_ST_SR_FINISHED_A;

760 #ifndef OPENSSL_NO_SCTP
761                                 /* Change to new shared key of SCTP-Auth,
762                                  * will be ignored if no SCTP used.
763                                  */
764                                 BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_NE
765 #endif
766                                 }
767                         else
768                                 {
769                                 s->s3->tmp.next_state=SSL_ST_OK;
770 #ifndef OPENSSL_NO_SCTP
771                                 if (BIO_dgram_is_sctp(SSL_get_wbio(s)))
772                                         {
773                                         s->d1->next_state = s->s3->tmp.next_stat
774                                         s->s3->tmp.next_state=DTLS1_SCTP_ST_SW_W
775                                         }
776 #endif
777                                 }
778                         s->init_num=0;
779                         break;

781                 case SSL_ST_OK:
782                         /* clean a few things up */
783                         ssl3_cleanup_key_block(s);

785 #if 0
786                         BUF_MEM_free(s->init_buf);
787                         s->init_buf=NULL;
```

```
788 #endif
790                         /* remove buffering on output */
791                         ssl_free_wbio_buffer(s);
793                         s->init_num=0;
795                         if (s->renegotiate == 2) /* skipped if we just sent a He
796                                 {
797                                 s->renegotiate=0;
798                                 s->new_session=0;
800                                 ssl_update_cache(s,SSL_SESS_CACHE_SERVER);
802                                 s->ctx->stats.sess_accept_good++;
803                                 /* s->server=1; */
804                                 s->handshake_func=dtls1_accept;
806                                 if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_DONE,1);
807                                 }
809                         ret = 1;
811                         /* done handshaking, next message is client hello */
812                         s->d1->handshake_read_seq = 0;
813                         /* next message is server hello */
814                         s->d1->handshake_write_seq = 0;
815                         s->d1->next_handshake_write_seq = 0;
816                         goto end;
817                         /* break; */
819                 default:
820                         SSLerr(SSL_F_DTLS1_ACCEPT,SSL_R_UNKNOWN_STATE);
821                         ret= -1;
822                         goto end;
823                         /* break; */
824                         }
826                 if (!s->s3->tmp.reuse_message && !skip)
827                         {
828                         if (s->debug)
829                                 {
830                                 if ((ret=BIO_flush(s->wbio)) <= 0)
831                                         goto end;
832                                 }
835                         if ((cb != NULL) && (s->state != state))
836                                 {
837                                 new_state=s->state;
838                                 s->state=state;
839                                 cb(s,SSL_CB_ACCEPT_LOOP,1);
840                                 s->state=new_state;
841                                 }
842                         }
843                 skip=0;
844                 }
845 end:
846         /* BIO_flush(s->wbio); */
848         s->in_handshake--;
849 #ifndef OPENSSL_NO_SCTP
850                 /* Notify SCTP BIO socket to leave handshake
851                  * mode and prevent stream identifier other
852                  * than 0. Will be ignored if no SCTP is used.
853                  */
```



```
854                 BIO_ctrl(SSL_get_wbio(s), BIO_CTRL_DGRAM_SCTP_SET_IN_HANDSHAKE,
855 #endif
857         if (cb != NULL)
858                 cb(s,SSL_CB_ACCEPT_EXIT,ret);
859         return(ret);
860         }
862 int dtls1_send_hello_request(SSL *s)
863         {
864         unsigned char *p;
866         if (s->state == SSL3_ST_SW_HELLO_REQ_A)
867                 {
868                 p=(unsigned char *)s->init_buf->data;
869                 p = dtls1_set_message_header(s, p, SSL3_MT_HELLO_REQUEST, 0, 0,
871                 s->state=SSL3_ST_SW_HELLO_REQ_B;
872                 /* number of bytes to write */
873                 s->init_num=DTLS1_HM_HEADER_LENGTH;
874                 s->init_off=0;
876                 /* no need to buffer this message, since there are no retransmit
877                  * requests for it */
878                 }
880         /* SSL3_ST_SW_HELLO_REQ_B */
881         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
882         }
884 int dtls1_send_hello_verify_request(SSL *s)
885         {
886         unsigned int msg_len;
887         unsigned char *msg, *buf, *p;
889         if (s->state == DTLS1_ST_SW_HELLO_VERIFY_REQUEST_A)
890                 {
891                 buf = (unsigned char *)s->init_buf->data;
893                 msg = p = &(buf[DTLS1_HM_HEADER_LENGTH]);
894                 *(p++) = s->version >> 8;
895                 *(p++) = s->version & 0xFF;
897                 if (s->ctx->app_gen_cookie_cb == NULL ||
898                      s->ctx->app_gen_cookie_cb(s, s->d1->cookie,
899                          &(s->d1->cookie_len)) == 0)
900                         {
901                         SSLerr(SSL_F_DTLS1_SEND_HELLO_VERIFY_REQUEST,ERR_R_INTER
902                         return 0;
903                         }
905                 *(p++) = (unsigned char) s->d1->cookie_len;
906                 memcpy(p, s->d1->cookie, s->d1->cookie_len);
907                 p += s->d1->cookie_len;
908                 msg_len = p - msg;
910                 dtls1_set_message_header(s, buf,
911                         DTLS1_MT_HELLO_VERIFY_REQUEST, msg_len, 0, msg_len);
913                 s->state=DTLS1_ST_SW_HELLO_VERIFY_REQUEST_B;
914                 /* number of bytes to write */
915                 s->init_num=p-buf;
916                 s->init_off=0;
917                 }
919         /* s->state = DTLS1_ST_SW_HELLO_VERIFY_REQUEST_B */
```

```
920         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
921         }

923 int dtls1_send_server_hello(SSL *s)
924         {
925         unsigned char *buf;
926         unsigned char *p,*d;
927         int i;
928         unsigned int sl;
929         unsigned long l;

931         if (s->state == SSL3_ST_SW_SRVR_HELLO_A)
932                 {
933                 buf=(unsigned char *)s->init_buf->data;
934                 p=s->s3->server_random;
935                 ssl_fill_hello_random(s, 1, p, SSL3_RANDOM_SIZE);
936                 /* Do the message type and length last */
937                 d=p= &(buf[DTLS1_HM_HEADER_LENGTH]);

939                 *(p++)=s->version>>8;
940                 *(p++)=s->version&0xff;

942                 /* Random stuff */
943                 memcpy(p,s->s3->server_random,SSL3_RANDOM_SIZE);
944                 p+=SSL3_RANDOM_SIZE;

946                 /* now in theory we have 3 options to sending back the
947                  * session id.  If it is a re-use, we send back the
948                  * old session-id, if it is a new session, we send
949                  * back the new session-id or we send back a 0 length
950                  * session-id if we want it to be single use.
951                  * Currently I will not implement the '0' length session-id
952                  * 12-Jan-98 - I'll now support the '0' length stuff.
953                  */
954                 if (!(s->ctx->session_cache_mode & SSL_SESS_CACHE_SERVER))
955                         s->session->session_id_length=0;

957                 sl=s->session->session_id_length;
958                 if (sl > sizeof s->session->session_id)
959                         {
960                         SSLerr(SSL_F_DTLS1_SEND_SERVER_HELLO, ERR_R_INTERNAL_ERR
961                         return -1;
962                         }
963                 *(p++)=sl;
964                 memcpy(p,s->session->session_id,sl);
965                 p+=sl;

967                 /* put the cipher */
968                 if (s->s3->tmp.new_cipher == NULL)
969                         return -1;
970                 i=ssl3_put_cipher_by_char(s->s3->tmp.new_cipher,p);
971                 p+=i;

973                 /* put the compression method */
974 #ifdef OPENSSL_NO_COMP
975                 *(p++)=0;
976 #else
977                 if (s->s3->tmp.new_compression == NULL)
978                         *(p++)=0;
979                 else
980                         *(p++)=s->s3->tmp.new_compression->id;
981 #endif

983 #ifndef OPENSSL_NO_TLSEXT
984                 if (ssl_prepare_serverhello_tlsext(s) <= 0)
985                         {
```



```
986                         SSLerr(SSL_F_DTLS1_SEND_SERVER_HELLO,SSL_R_SERVERHELLO_T
987                         return -1;
988                         }
989                 if ((p = ssl_add_serverhello_tlsext(s, p, buf+SSL3_RT_MAX_PLAIN_
990                         {
991                         SSLerr(SSL_F_DTLS1_SEND_SERVER_HELLO,ERR_R_INTERNAL_ERRO
992                         return -1;
993                         }
994 #endif

996                 /* do the header */
997                 l=(p-d);
998                 d=buf;

1000                d = dtls1_set_message_header(s, d, SSL3_MT_SERVER_HELLO, l, 0, l

1002                s->state=SSL3_ST_SW_SRVR_HELLO_B;
1003                /* number of bytes to write */
1004                s->init_num=p-buf;
1005                s->init_off=0;

1007                /* buffer the message to handle re-xmits */
1008                dtls1_buffer_message(s, 0);
1009                }

1011        /* SSL3_ST_SW_SRVR_HELLO_B */
1012        return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1013        }

1015 int dtls1_send_server_done(SSL *s)
1016        {
1017        unsigned char *p;

1019        if (s->state == SSL3_ST_SW_SRVR_DONE_A)
1020                {
1021                p=(unsigned char *)s->init_buf->data;

1023                /* do the header */
1024                p = dtls1_set_message_header(s, p, SSL3_MT_SERVER_DONE, 0, 0, 0)

1026                s->state=SSL3_ST_SW_SRVR_DONE_B;
1027                /* number of bytes to write */
1028                s->init_num=DTLS1_HM_HEADER_LENGTH;
1029                s->init_off=0;

1031                /* buffer the message to handle re-xmits */
1032                dtls1_buffer_message(s, 0);
1033                }

1035        /* SSL3_ST_SW_SRVR_DONE_B */
1036        return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1037        }

1039 int dtls1_send_server_key_exchange(SSL *s)
1040        {
1041 #ifndef OPENSSL_NO_RSA
1042        unsigned char *q;
1043        int j,num;
1044        RSA *rsa;
1045        unsigned char md_buf[MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH];
1046        unsigned int u;
1047 #endif
1048 #ifndef OPENSSL_NO_DH
1049        DH *dh=NULL,*dhp;
1050 #endif
1051 #ifndef OPENSSL_NO_ECDH
```

```
1052         EC_KEY *ecdh=NULL, *ecdhp;
1053         unsigned char *encodedPoint = NULL;
1054         int encodedlen = 0;
1055         int curve_id = 0;
1056         BN_CTX *bn_ctx = NULL;
1057 #endif
1058         EVP_PKEY *pkey;
1059         unsigned char *p,*d;
1060         int al,i;
1061         unsigned long type;
1062         int n;
1063         CERT *cert;
1064         BIGNUM *r[4];
1065         int nr[4],kn;
1066         BUF_MEM *buf;
1067         EVP_MD_CTX md_ctx;

1069         EVP_MD_CTX_init(&md_ctx);
1070         if (s->state == SSL3_ST_SW_KEY_EXCH_A)
1071                 {
1072                 type=s->s3->tmp.new_cipher->algorithm_mkey;
1073                 cert=s->cert;

1075                 buf=s->init_buf;

1077                 r[0]=r[1]=r[2]=r[3]=NULL;
1078                 n=0;
1079 #ifndef OPENSSL_NO_RSA
1080                 if (type & SSL_kRSA)
1081                         {
1082                         rsa=cert->rsa_tmp;
1083                         if ((rsa == NULL) && (s->cert->rsa_tmp_cb != NULL))
1084                                 {
1085                                 rsa=s->cert->rsa_tmp_cb(s,
1086                                       SSL_C_IS_EXPORT(s->s3->tmp.new_cipher),
1087                                       SSL_C_EXPORT_PKEYLENGTH(s->s3->tmp.new_cip
1088                                 if(rsa == NULL)
1089                                 {
1090                                        al=SSL_AD_HANDSHAKE_FAILURE;
1091                                        SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHA
1092                                        goto f_err;
1093                                 }
1094                                 RSA_up_ref(rsa);
1095                                 cert->rsa_tmp=rsa;
1096                                 }
1097                         if (rsa == NULL)
1098                                 {
1099                                 al=SSL_AD_HANDSHAKE_FAILURE;
1100                                 SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,SSL_
1101                                 goto f_err;
1102                                 }
1103                         r[0]=rsa->n;
1104                         r[1]=rsa->e;
1105                         s->s3->tmp.use_rsa_tmp=1;
1106                         }
1107                 else
1108 #endif
1109 #ifndef OPENSSL_NO_DH
1110                         if (type & SSL_kEDH)
1111                         {
1112                         dhp=cert->dh_tmp;
1113                         if ((dhp == NULL) && (s->cert->dh_tmp_cb != NULL))
1114                                 dhp=s->cert->dh_tmp_cb(s,
1115                                       SSL_C_IS_EXPORT(s->s3->tmp.new_cipher),
1116                                       SSL_C_EXPORT_PKEYLENGTH(s->s3->tmp.new_cip
1117                         if (dhp == NULL)
```



```
1118                                 {
1119                                 al=SSL_AD_HANDSHAKE_FAILURE;
1120                                 SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,SSL_
1121                                 goto f_err;
1122                                 }

1124                         if (s->s3->tmp.dh != NULL)
1125                                 {
1126                                 DH_free(dh);
1127                                 SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE, ERR
1128                                 goto err;
1129                                 }

1131                         if ((dh=DHparams_dup(dhp)) == NULL)
1132                                 {
1133                                 SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,ERR_
1134                                 goto err;
1135                                 }

1137                         s->s3->tmp.dh=dh;
1138                         if ((dhp->pub_key == NULL ||
1139                              dhp->priv_key == NULL ||
1140                              (s->options & SSL_OP_SINGLE_DH_USE)))
1141                                 {
1142                                 if(!DH_generate_key(dh))
1143                                     {
1144                                     SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,
1145                                            ERR_R_DH_LIB);
1146                                     goto err;
1147                                     }
1148                                 }
1149                         else
1150                                 {
1151                                 dh->pub_key=BN_dup(dhp->pub_key);
1152                                 dh->priv_key=BN_dup(dhp->priv_key);
1153                                 if ((dh->pub_key == NULL) ||
1154                                        (dh->priv_key == NULL))
1155                                        {
1156                                        SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHA
1157                                        goto err;
1158                                        }
1159                                 }
1160                         r[0]=dh->p;
1161                         r[1]=dh->g;
1162                         r[2]=dh->pub_key;
1163                         }
1164                 else
1165 #endif
1166 #ifndef OPENSSL_NO_ECDH
1167                         if (type & SSL_kEECDH)
1168                         {
1169                         const EC_GROUP *group;

1171                         ecdhp=cert->ecdh_tmp;
1172                         if ((ecdhp == NULL) && (s->cert->ecdh_tmp_cb != NULL))
1173                                 {
1174                                 ecdhp=s->cert->ecdh_tmp_cb(s,
1175                                       SSL_C_IS_EXPORT(s->s3->tmp.new_cipher),
1176                                       SSL_C_EXPORT_PKEYLENGTH(s->s3->tmp.new_cip
1177                                 }
1178                         if (ecdhp == NULL)
1179                                 {
1180                                 al=SSL_AD_HANDSHAKE_FAILURE;
1181                                 SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,SSL_
1182                                 goto f_err;
1183                                 }
```

```
1185                        if (s->s3->tmp.ecdh != NULL)
1186                                {
1187                                EC_KEY_free(s->s3->tmp.ecdh);
1188                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE, ERR
1189                                goto err;
1190                                }
1191
1192                        /* Duplicate the ECDH structure. */
1193                        if (ecdhp == NULL)
1194                                {
1195                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,ERR_
1196                                goto err;
1197                                }
1198                        if ((ecdh = EC_KEY_dup(ecdhp)) == NULL)
1199                                {
1200                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,ERR_
1201                                goto err;
1202                                }
1203
1204                        s->s3->tmp.ecdh=ecdh;
1205                        if ((EC_KEY_get0_public_key(ecdh) == NULL) ||
1206                            (EC_KEY_get0_private_key(ecdh) == NULL) ||
1207                            (s->options & SSL_OP_SINGLE_ECDH_USE))
1208                                {
1209                                if(!EC_KEY_generate_key(ecdh))
1210                                    {
1211                                    SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,
1212                                    goto err;
1213                                    }
1214                                }
1215
1216                        if (((group = EC_KEY_get0_group(ecdh)) == NULL) ||
1217                            (EC_KEY_get0_public_key(ecdh)  == NULL) ||
1218                            (EC_KEY_get0_private_key(ecdh) == NULL))
1219                                {
1220                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,ERR_
1221                                goto err;
1222                                }
1223
1224                        if (SSL_C_IS_EXPORT(s->s3->tmp.new_cipher) &&
1225                            (EC_GROUP_get_degree(group) > 163))
1226                                {
1227                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,SSL_
1228                                goto err;
1229                                }
1230
1231                        /* XXX: For now, we only support ephemeral ECDH
1232                         * keys over named (not generic) curves. For
1233                         * supported named curves, curve_id is non-zero.
1234                         */
1235                        if ((curve_id =
1236                            tls1_ec_nid2curve_id(EC_GROUP_get_curve_name(group))
1237                            == 0)
1238                                {
1239                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,SSL_
1240                                goto err;
1241                                }
1242
1243                        /* Encode the public key.
1244                         * First check the size of encoding and
1245                         * allocate memory accordingly.
1246                         */
1247                        encodedlen = EC_POINT_point2oct(group,
1248                            EC_KEY_get0_public_key(ecdh),
1249                            POINT_CONVERSION_UNCOMPRESSED,
```



```
1250                                NULL, 0, NULL);
1251
1252                        encodedPoint = (unsigned char *)
1253                            OPENSSL_malloc(encodedlen*sizeof(unsigned char));
1254                        bn_ctx = BN_CTX_new();
1255                        if ((encodedPoint == NULL) || (bn_ctx == NULL))
1256                                {
1257                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,ERR_
1258                                goto err;
1259                                }
1260
1261                        encodedlen = EC_POINT_point2oct(group,
1262                            EC_KEY_get0_public_key(ecdh),
1263                            POINT_CONVERSION_UNCOMPRESSED,
1264                            encodedPoint, encodedlen, bn_ctx);
1265
1266                        if (encodedlen == 0)
1267                                {
1268                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,ERR_
1269                                goto err;
1270                                }
1271
1272                        BN_CTX_free(bn_ctx);  bn_ctx=NULL;
1273
1274                        /* XXX: For now, we only support named (not
1275                         * generic) curves in ECDH ephemeral key exchanges.
1276                         * In this situation, we need four additional bytes
1277                         * to encode the entire ServerECDHParams
1278                         * structure.
1279                         */
1280                        n = 4 + encodedlen;
1281
1282                        /* We'll generate the serverKeyExchange message
1283                         * explicitly so we can set these to NULLs
1284                         */
1285                        r[0]=NULL;
1286                        r[1]=NULL;
1287                        r[2]=NULL;
1288                        r[3]=NULL;
1289                        }
1290                else
1291 #endif /* !OPENSSL_NO_ECDH */
1292 #ifndef OPENSSL_NO_PSK
1293                        if (type & SSL_kPSK)
1294                                {
1295                                /* reserve size for record length and PSK identi
1296                                n+=2+strlen(s->ctx->psk_identity_hint);
1297                                }
1298                        else
1299 #endif /* !OPENSSL_NO_PSK */
1300                        {
1301                        al=SSL_AD_HANDSHAKE_FAILURE;
1302                        SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,SSL_R_UNKNOW
1303                        goto f_err;
1304                        }
1305                for (i=0; r[i] != NULL; i++)
1306                        {
1307                        nr[i]=BN_num_bytes(r[i]);
1308                        n+=2+nr[i];
1309                        }
1310
1311                if (!(s->s3->tmp.new_cipher->algorithm_auth & SSL_aNULL)
1312                        && !(s->s3->tmp.new_cipher->algorithm_mkey & SSL_kPSK))
1313                        {
1314                        if ((pkey=ssl_get_sign_pkey(s,s->s3->tmp.new_cipher, NUL
1315
```

```
1316                                == NULL)
1317                                {
1318                                al=SSL_AD_DECODE_ERROR;
1319                                goto f_err;
1320                                }
1321                        kn=EVP_PKEY_size(pkey);
1322                        }
1323                else
1324                        {
1325                        pkey=NULL;
1326                        kn=0;
1327                        }

1329                if (!BUF_MEM_grow_clean(buf,n+DTLS1_HM_HEADER_LENGTH+kn))
1330                        {
1331                        SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,ERR_LIB_BUF)
1332                        goto err;
1333                        }
1334                d=(unsigned char *)s->init_buf->data;
1335                p= &(d[DTLS1_HM_HEADER_LENGTH]);

1337                for (i=0; r[i] != NULL; i++)
1338                        {
1339                        s2n(nr[i],p);
1340                        BN_bn2bin(r[i],p);
1341                        p+=nr[i];
1342                        }

1344 #ifndef OPENSSL_NO_ECDH
1345                if (type & SSL_kEECDH)
1346                        {
1347                        /* XXX: For now, we only support named (not generic) cur
1348                         * In this situation, the serverKeyExchange message has:
1349                         * [1 byte CurveType], [2 byte CurveName]
1350                         * [1 byte length of encoded point], followed by
1351                         * the actual encoded point itself
1352                         */
1353                        *p = NAMED_CURVE_TYPE;
1354                        p += 1;
1355                        *p = 0;
1356                        p += 1;
1357                        *p = curve_id;
1358                        p += 1;
1359                        *p = encodedlen;
1360                        p += 1;
1361                        memcpy((unsigned char*)p,
1362                            (unsigned char *)encodedPoint,
1363                            encodedlen);
1364                        OPENSSL_free(encodedPoint);
1365                        encodedPoint = NULL;
1366                        p += encodedlen;
1367                        }
1368 #endif

1370 #ifndef OPENSSL_NO_PSK
1371                if (type & SSL_kPSK)
1372                        {
1373                        /* copy PSK identity hint */
1374                        s2n(strlen(s->ctx->psk_identity_hint), p);
1375                        strncpy((char *)p, s->ctx->psk_identity_hint, strlen(s->
1376                        p+=strlen(s->ctx->psk_identity_hint);
1377                        }
1378 #endif

1380                /* not anonymous */
1381                if (pkey != NULL)
```



```
1382                        {
1383                        /* n is the length of the params, they start at
1384                         * &(d[DTLS1_HM_HEADER_LENGTH]) and p points to the spac
1385                         * at the end. */
1386 #ifndef OPENSSL_NO_RSA
1387                        if (pkey->type == EVP_PKEY_RSA)
1388                                {
1389                                q=md_buf;
1390                                j=0;
1391                                for (num=2; num > 0; num--)
1392                                        {
1393                                        EVP_DigestInit_ex(&md_ctx,(num == 2)
1394                                                ?s->ctx->md5:s->ctx->sha1, NULL)
1395                                        EVP_DigestUpdate(&md_ctx,&(s->s3->client
1396                                        EVP_DigestUpdate(&md_ctx,&(s->s3->server
1397                                        EVP_DigestUpdate(&md_ctx,&(d[DTLS1_HM_HE
1398                                        EVP_DigestFinal_ex(&md_ctx,q,
1399                                                (unsigned int *)&i);
1400                                        q+=i;
1401                                        j+=i;
1402                                        }
1403                                if (RSA_sign(NID_md5_sha1, md_buf, j,
1404                                        &(p[2]), &u, pkey->pkey.rsa) <= 0)
1405                                        {
1406                                        SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHA
1407                                        goto err;
1408                                        }
1409                                s2n(u,p);
1410                                n+=u+2;
1411                                }
1412                        else
1413 #endif
1414 #if !defined(OPENSSL_NO_DSA)
1415                                if (pkey->type == EVP_PKEY_DSA)
1416                                {
1417                                /* lets do DSS */
1418                                EVP_SignInit_ex(&md_ctx,EVP_dss1(), NULL);
1419                                EVP_SignUpdate(&md_ctx,&(s->s3->client_random[0]
1420                                EVP_SignUpdate(&md_ctx,&(s->s3->server_random[0]
1421                                EVP_SignUpdate(&md_ctx,&(d[DTLS1_HM_HEADER_LENGT
1422                                if (!EVP_SignFinal(&md_ctx,&(p[2]),
1423                                        (unsigned int *)&i,pkey))
1424                                        {
1425                                        SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHA
1426                                        goto err;
1427                                        }
1428                                s2n(i,p);
1429                                n+=i+2;
1430                                }
1431                        else
1432 #endif
1433 #if !defined(OPENSSL_NO_ECDSA)
1434                                if (pkey->type == EVP_PKEY_EC)
1435                                {
1436                                /* let's do ECDSA */
1437                                EVP_SignInit_ex(&md_ctx,EVP_ecdsa(), NULL);
1438                                EVP_SignUpdate(&md_ctx,&(s->s3->client_random[0]
1439                                EVP_SignUpdate(&md_ctx,&(s->s3->server_random[0]
1440                                EVP_SignUpdate(&md_ctx,&(d[DTLS1_HM_HEADER_LENGT
1441                                if (!EVP_SignFinal(&md_ctx,&(p[2]),
1442                                        (unsigned int *)&i,pkey))
1443                                        {
1444                                        SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHA
1445                                        goto err;
1446                                        }
1447                                s2n(i,p);
```

```
1448                                n+=i+2;
1449                                }
1450                        else
1451 #endif
1452                                {
1453                                /* Is this error check actually needed? */
1454                                al=SSL_AD_HANDSHAKE_FAILURE;
1455                                SSLerr(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE,SSL_
1456                                goto f_err;
1457                                }
1458                        }

1460                d = dtls1_set_message_header(s, d,
1461                        SSL3_MT_SERVER_KEY_EXCHANGE, n, 0, n);

1463                /* we should now have things packed up, so lets send
1464                 * it off */
1465                s->init_num=n+DTLS1_HM_HEADER_LENGTH;
1466                s->init_off=0;

1468                /* buffer the message to handle re-xmits */
1469                dtls1_buffer_message(s, 0);
1470                }

1472        s->state = SSL3_ST_SW_KEY_EXCH_B;
1473        EVP_MD_CTX_cleanup(&md_ctx);
1474        return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1475 f_err:
1476        ssl3_send_alert(s,SSL3_AL_FATAL,al);
1477 err:
1478 #ifndef OPENSSL_NO_ECDH
1479        if (encodedPoint != NULL) OPENSSL_free(encodedPoint);
1480        BN_CTX_free(bn_ctx);
1481 #endif
1482        EVP_MD_CTX_cleanup(&md_ctx);
1483        return(-1);
1484        }

1486 int dtls1_send_certificate_request(SSL *s)
1487        {
1488        unsigned char *p,*d;
1489        int i,j,nl,off,n;
1490        STACK_OF(X509_NAME) *sk=NULL;
1491        X509_NAME *name;
1492        BUF_MEM *buf;
1493        unsigned int msg_len;

1495        if (s->state == SSL3_ST_SW_CERT_REQ_A)
1496                {
1497                buf=s->init_buf;

1499                d=p=(unsigned char *)&(buf->data[DTLS1_HM_HEADER_LENGTH]);

1501                /* get the list of acceptable cert types */
1502                p++;
1503                n=ssl3_get_req_cert_type(s,p);
1504                d[0]=n;
1505                p+=n;
1506                n++;

1508                off=n;
1509                p+=2;
1510                n+=2;

1512                sk=SSL_get_client_CA_list(s);
1513                nl=0;
```



```
1514                if (sk != NULL)
1515                        {
1516                        for (i=0; i<sk_X509_NAME_num(sk); i++)
1517                                {
1518                                name=sk_X509_NAME_value(sk,i);
1519                                j=i2d_X509_NAME(name,NULL);
1520                                if (!BUF_MEM_grow_clean(buf,DTLS1_HM_HEADER_LENG
1521                                        {
1522                                        SSLerr(SSL_F_DTLS1_SEND_CERTIFICATE_REQU
1523                                        goto err;
1524                                        }
1525                                p=(unsigned char *)&(buf->data[DTLS1_HM_HEADER_L
1526                                if (!(s->options & SSL_OP_NETSCAPE_CA_DN_BUG))
1527                                        {
1528                                        s2n(j,p);
1529                                        i2d_X509_NAME(name,&p);
1530                                        n+=2+j;
1531                                        nl+=2+j;
1532                                        }
1533                                else
1534                                        {
1535                                        d=p;
1536                                        i2d_X509_NAME(name,&p);
1537                                        j-=2; s2n(j,d); j+=2;
1538                                        n+=j;
1539                                        nl+=j;
1540                                        }
1541                                }
1542                        }
1543                /* else no CA names */
1544                p=(unsigned char *)&(buf->data[DTLS1_HM_HEADER_LENGTH+off]);
1545                s2n(nl,p);

1547                d=(unsigned char *)buf->data;
1548                *(d++)=SSL3_MT_CERTIFICATE_REQUEST;
1549                l2n3(n,d);
1550                s2n(s->d1->handshake_write_seq,d);
1551                s->d1->handshake_write_seq++;

1553                /* we should now have things packed up, so lets send
1554                 * it off */

1556                s->init_num=n+DTLS1_HM_HEADER_LENGTH;
1557                s->init_off=0;
1558 #ifdef NETSCAPE_HANG_BUG
1559 /* XXX: what to do about this? */
1560                p=(unsigned char *)s->init_buf->data + s->init_num;

1562                /* do the header */
1563                *(p++)=SSL3_MT_SERVER_DONE;
1564                *(p++)=0;
1565                *(p++)=0;
1566                *(p++)=0;
1567                s->init_num += 4;
1568 #endif

1570                /* XDTLS:  set message header ? */
1571                msg_len = s->init_num - DTLS1_HM_HEADER_LENGTH;
1572                dtls1_set_message_header(s, (void *)s->init_buf->data,
1573                        SSL3_MT_CERTIFICATE_REQUEST, msg_len, 0, msg_len);

1575                /* buffer the message to handle re-xmits */
1576                dtls1_buffer_message(s, 0);

1578                s->state = SSL3_ST_SW_CERT_REQ_B;
1579                }
```

```
1581         /* SSL3_ST_SW_CERT_REQ_B */
1582         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1583 err:
1584         return(-1);
1585         }
1587 int dtls1_send_server_certificate(SSL *s)
1588         {
1589         unsigned long l;
1590         X509 *x;
1592         if (s->state == SSL3_ST_SW_CERT_A)
1593                 {
1594                 x=ssl_get_server_send_cert(s);
1595                 if (x == NULL)
1596                         {
1597                         /* VRS: allow null cert if auth == KRB5 */
1598                         if ((s->s3->tmp.new_cipher->algorithm_mkey != SSL_kKRB5)
1599                             (s->s3->tmp.new_cipher->algorithm_auth != SSL_aKRB5)
1600                                 {
1601                                 SSLerr(SSL_F_DTLS1_SEND_SERVER_CERTIFICATE,ERR_R
1602                                 return(0);
1603                                 }
1604                         }
1606                 l=dtls1_output_cert_chain(s,x);
1607                 s->state=SSL3_ST_SW_CERT_B;
1608                 s->init_num=(int)l;
1609                 s->init_off=0;
1611                 /* buffer the message to handle re-xmits */
1612                 dtls1_buffer_message(s, 0);
1613                 }
1615         /* SSL3_ST_SW_CERT_B */
1616         return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1617         }
1619 #ifndef OPENSSL_NO_TLSEXT
1620 int dtls1_send_newsession_ticket(SSL *s)
1621         {
1622         if (s->state == SSL3_ST_SW_SESSION_TICKET_A)
1623                 {
1624                 unsigned char *p, *senc, *macstart;
1625                 int len, slen;
1626                 unsigned int hlen, msg_len;
1627                 EVP_CIPHER_CTX ctx;
1628                 HMAC_CTX hctx;
1629                 SSL_CTX *tctx = s->initial_ctx;
1630                 unsigned char iv[EVP_MAX_IV_LENGTH];
1631                 unsigned char key_name[16];
1633                 /* get session encoding length */
1634                 slen = i2d_SSL_SESSION(s->session, NULL);
1635                 /* Some length values are 16 bits, so forget it if session is
1636                  * too long
1637                  */
1638                 if (slen > 0xFF00)
1639                         return -1;
1640                 /* Grow buffer if need be: the length calculation is as
1641                  * follows 12 (DTLS handshake message header) +
1642                  * 4 (ticket lifetime hint) + 2 (ticket length) +
1643                  * 16 (key name) + max_iv_len (iv length) +
1644                  * session_length + max_enc_block_size (max encrypted session
1645                  * length) + max_md_size (HMAC).
```



```
1646                  */
1647                 if (!BUF_MEM_grow(s->init_buf,
1648                         DTLS1_HM_HEADER_LENGTH + 22 + EVP_MAX_IV_LENGTH +
1649                         EVP_MAX_BLOCK_LENGTH + EVP_MAX_MD_SIZE + slen))
1650                         return -1;
1651                 senc = OPENSSL_malloc(slen);
1652                 if (!senc)
1653                         return -1;
1654                 p = senc;
1655                 i2d_SSL_SESSION(s->session, &p);
1657                 p=(unsigned char *)&(s->init_buf->data[DTLS1_HM_HEADER_LENGTH]);
1658                 EVP_CIPHER_CTX_init(&ctx);
1659                 HMAC_CTX_init(&hctx);
1660                 /* Initialize HMAC and cipher contexts. If callback present
1661                  * it does all the work otherwise use generated values
1662                  * from parent ctx.
1663                  */
1664                 if (tctx->tlsext_ticket_key_cb)
1665                         {
1666                         if (tctx->tlsext_ticket_key_cb(s, key_name, iv, &ctx,
1667                                                        &hctx, 1) < 0)
1668                                 {
1669                                 OPENSSL_free(senc);
1670                                 return -1;
1671                                 }
1672                         }
1673                 else
1674                         {
1675                         RAND_pseudo_bytes(iv, 16);
1676                         EVP_EncryptInit_ex(&ctx, EVP_aes_128_cbc(), NULL,
1677                                            tctx->tlsext_tick_aes_key, iv);
1678                         HMAC_Init_ex(&hctx, tctx->tlsext_tick_hmac_key, 16,
1679                                      tlsext_tick_md(), NULL);
1680                         memcpy(key_name, tctx->tlsext_tick_key_name, 16);
1681                         }
1682                 l2n(s->session->tlsext_tick_lifetime_hint, p);
1683                 /* Skip ticket length for now */
1684                 p += 2;
1685                 /* Output key name */
1686                 macstart = p;
1687                 memcpy(p, key_name, 16);
1688                 p += 16;
1689                 /* output IV */
1690                 memcpy(p, iv, EVP_CIPHER_CTX_iv_length(&ctx));
1691                 p += EVP_CIPHER_CTX_iv_length(&ctx);
1692                 /* Encrypt session data */
1693                 EVP_EncryptUpdate(&ctx, p, &len, senc, slen);
1694                 p += len;
1695                 EVP_EncryptFinal(&ctx, p, &len);
1696                 p += len;
1697                 EVP_CIPHER_CTX_cleanup(&ctx);
1699                 HMAC_Update(&hctx, macstart, p - macstart);
1700                 HMAC_Final(&hctx, p, &hlen);
1701                 HMAC_CTX_cleanup(&hctx);
1703                 p += hlen;
1704                 /* Now write out lengths: p points to end of data written */
1705                 /* Total length */
1706                 len = p - (unsigned char *)(s->init_buf->data);
1707                 /* Ticket length */
1708                 p=(unsigned char *)&(s->init_buf->data[DTLS1_HM_HEADER_LENGTH])
1709                 s2n(len - DTLS1_HM_HEADER_LENGTH - 6, p);
1711                 /* number of bytes to write */
```

```
1712                s->init_num= len;
1713                s->state=SSL3_ST_SW_SESSION_TICKET_B;
1714                s->init_off=0;
1715                OPENSSL_free(senc);

1717                /* XDTLS:  set message header ? */
1718                msg_len = s->init_num - DTLS1_HM_HEADER_LENGTH;
1719                dtls1_set_message_header(s, (void *)s->init_buf->data,
1720                        SSL3_MT_NEWSESSION_TICKET, msg_len, 0, msg_len);

1722                /* buffer the message to handle re-xmits */
1723                dtls1_buffer_message(s, 0);
1724                }

1726        /* SSL3_ST_SW_SESSION_TICKET_B */
1727        return(dtls1_do_write(s,SSL3_RT_HANDSHAKE));
1728        }
1729 #endif
1730 #endif /* ! codereview */
```

```
**********************************************************
    1323 Wed Aug 13 19:53:35 2014
new/usr/src/lib/openssl/libsunw_ssl/i386/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #
  21 #
  22 # Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23 # Copyright 2014 Alexander Pyhalov
  24 # Use is subject to license terms.
  25 #

  27 include ../Makefile.com

  29 CPPFLAGS += -DL_ENDIAN
  30 CPPFLAGS += -DOPENSSL_NO_INLINE_ASM
  31 CPPFLAGS += -DOPENSSL_BN_ASM_PART_WORDS
  32 CPPFLAGS += -DOPENSSL_IA32_SSE2
  33 CPPFLAGS += -DRMD160_ASM
  34 CPPFLAGS += -DAES_ASM

  36 .KEEP_STATE:

  38 all:  $(ROOTLIBDIR) $(LIBS) $(LIBLINKS)

  40 $(LIBLINKS):    FRC
  41         $(RM) $@; $(SYMLINK) $(DYNLIB) $@

  43 $(ROOTLIBDIR):
  44         $(INS.dir)

  46 install: all $(ROOTLIBS) $(ROOTLINKS)

  48 FRC:
  49 #endif /* ! codereview */
```

```
**********************************************************
   69379 Wed Aug 13 19:53:35 2014
new/usr/src/lib/openssl/libsunw_ssl/kssl.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/kssl.c -*- mode: C; c-file-style: "eay" -*- */
/* Written by Vern Staats <staatsvr@asc.hpc.mil> for the OpenSSL project 2000.
 */
/* ====================================================================
 * Copyright (c) 2000 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */


/*  ssl/kssl.c  --  Routines to support (& debug) Kerberos5 auth for openssl
**
**  19990701    VRS     Started.
```



```
**  200011??    Jeffrey Altman, Richard Levitte
**                      Generalized for Heimdal, Newer MIT, & Win32.
**                      Integrated into main OpenSSL 0.9.7 snapshots.
**  20010413    Simon Wilkinson, VRS
**                      Real RFC2712 KerberosWrapper replaces AP_REQ.
*/

#include <openssl/opensslconf.h>

#include <string.h>

#define KRB5_PRIVATE    1

#include <openssl/ssl.h>
#include <openssl/evp.h>
#include <openssl/objects.h>
#include <openssl/krb5_asn.h>
#include "kssl_lcl.h"

#ifndef OPENSSL_NO_KRB5

#ifndef ENOMEM
#define ENOMEM KRB5KRB_ERR_GENERIC
#endif

/*
 * When OpenSSL is built on Windows, we do not want to require that
 * the Kerberos DLLs be available in order for the OpenSSL DLLs to
 * work.  Therefore, all Kerberos routines are loaded at run time
 * and we do not link to a .LIB file.
 */

#if defined(OPENSSL_SYS_WINDOWS) || defined(OPENSSL_SYS_WIN32)
/*
 * The purpose of the following pre-processor statements is to provide
 * compatibility with different releases of MIT Kerberos for Windows.
 * All versions up to 1.2 used macros.  But macros do not allow for
 * a binary compatible interface for DLLs.  Therefore, all macros are
 * being replaced by function calls.  The following code will allow
 * an OpenSSL DLL built on Windows to work whether or not the macro
 * or function form of the routines are utilized.
 */
#ifdef  krb5_cc_get_principal
#define NO_DEF_KRB5_CCACHE
#undef  krb5_cc_get_principal
#endif
#define krb5_cc_get_principal    kssl_krb5_cc_get_principal

#define krb5_free_data_contents  kssl_krb5_free_data_contents
#define krb5_free_context        kssl_krb5_free_context
#define krb5_auth_con_free       kssl_krb5_auth_con_free
#define krb5_free_principal      kssl_krb5_free_principal
#define krb5_mk_req_extended     kssl_krb5_mk_req_extended
#define krb5_get_credentials     kssl_krb5_get_credentials
#define krb5_cc_default          kssl_krb5_cc_default
#define krb5_sname_to_principal  kssl_krb5_sname_to_principal
#define krb5_init_context        kssl_krb5_init_context
#define krb5_free_ticket         kssl_krb5_free_ticket
#define krb5_rd_req              kssl_krb5_rd_req
#define krb5_kt_default          kssl_krb5_kt_default
#define krb5_kt_resolve          kssl_krb5_kt_resolve
/* macros in mit 1.2.2 and earlier; functions in mit 1.2.3 and greater */
#ifndef krb5_kt_close
#define krb5_kt_close            kssl_krb5_kt_close
#endif /* krb5_kt_close */
#ifndef krb5_kt_get_entry
```

```
128 #define krb5_kt_get_entry        kssl_krb5_kt_get_entry
129 #endif /* krb5_kt_get_entry */
130 #define krb5_auth_con_init       kssl_krb5_auth_con_init

132 #define krb5_principal_compare   kssl_krb5_principal_compare
133 #define krb5_decrypt_tkt_part    kssl_krb5_decrypt_tkt_part
134 #define krb5_timeofday           kssl_krb5_timeofday
135 #define krb5_rc_default          kssl_krb5_rc_default

137 #ifdef krb5_rc_initialize
138 #undef krb5_rc_initialize
139 #endif
140 #define krb5_rc_initialize   kssl_krb5_rc_initialize

142 #ifdef krb5_rc_get_lifespan
143 #undef krb5_rc_get_lifespan
144 #endif
145 #define krb5_rc_get_lifespan kssl_krb5_rc_get_lifespan

147 #ifdef krb5_rc_destroy
148 #undef krb5_rc_destroy
149 #endif
150 #define krb5_rc_destroy      kssl_krb5_rc_destroy

152 #define valid_cksumtype      kssl_valid_cksumtype
153 #define krb5_checksum_size   kssl_krb5_checksum_size
154 #define krb5_kt_free_entry   kssl_krb5_kt_free_entry
155 #define krb5_auth_con_setrcache  kssl_krb5_auth_con_setrcache
156 #define krb5_auth_con_getrcache  kssl_krb5_auth_con_getrcache
157 #define krb5_get_server_rcache   kssl_krb5_get_server_rcache

159 /* Prototypes for built in stubs */
160 void kssl_krb5_free_data_contents(krb5_context, krb5_data *);
161 void kssl_krb5_free_principal(krb5_context, krb5_principal );
162 krb5_error_code kssl_krb5_kt_resolve(krb5_context,
163                                      krb5_const char *,
164                                      krb5_keytab *);
165 krb5_error_code kssl_krb5_kt_default(krb5_context,
166                                      krb5_keytab *);
167 krb5_error_code kssl_krb5_free_ticket(krb5_context, krb5_ticket *);
168 krb5_error_code kssl_krb5_rd_req(krb5_context, krb5_auth_context *,
169                                  krb5_const krb5_data *,
170                                  krb5_const_principal, krb5_keytab,
171                                  krb5_flags *,krb5_ticket **);

173 krb5_boolean kssl_krb5_principal_compare(krb5_context, krb5_const_principal,
174                                          krb5_const_principal);
175 krb5_error_code kssl_krb5_mk_req_extended(krb5_context,
176                                           krb5_auth_context  *,
177                                           krb5_const krb5_flags,
178                                           krb5_data  *,
179                                           krb5_creds  *,
180                                           krb5_data  * );
181 krb5_error_code kssl_krb5_init_context(krb5_context *);
182 void kssl_krb5_free_context(krb5_context);
183 krb5_error_code kssl_krb5_cc_default(krb5_context,krb5_ccache  *);
184 krb5_error_code kssl_krb5_sname_to_principal(krb5_context,
185                                              krb5_const char  *,
186                                              krb5_const char  *,
187                                              krb5_int32,
188                                              krb5_principal  *);
189 krb5_error_code kssl_krb5_get_credentials(krb5_context,
190                                           krb5_const krb5_flags,
191                                           krb5_ccache,
192                                           krb5_creds  *,
193                                           krb5_creds  *  *);
```



```
194 krb5_error_code kssl_krb5_auth_con_init(krb5_context,
195                                          krb5_auth_context  *);
196 krb5_error_code kssl_krb5_cc_get_principal(krb5_context context,
197                                            krb5_ccache cache,
198                                            krb5_principal *principal);
199 krb5_error_code kssl_krb5_auth_con_free(krb5_context,krb5_auth_context);
200 size_t kssl_krb5_checksum_size(krb5_context context,krb5_cksumtype ctype);
201 krb5_boolean kssl_valid_cksumtype(krb5_cksumtype ctype);
202 krb5_error_code krb5_kt_free_entry(krb5_context,krb5_keytab_entry FAR * );
203 krb5_error_code kssl_krb5_auth_con_setrcache(krb5_context,
204                                              krb5_auth_context,
205                                              krb5_rcache);
206 krb5_error_code kssl_krb5_get_server_rcache(krb5_context,
207                                             krb5_const krb5_data *,
208                                             krb5_rcache *);
209 krb5_error_code kssl_krb5_auth_con_getrcache(krb5_context,
210                                              krb5_auth_context,
211                                              krb5_rcache *);

213 /* Function pointers (almost all Kerberos functions are _stdcall) */
214 static void (_stdcall *p_krb5_free_data_contents)(krb5_context, krb5_data *)
215         =NULL;
216 static void (_stdcall *p_krb5_free_principal)(krb5_context, krb5_principal )
217         =NULL;
218 static krb5_error_code(_stdcall *p_krb5_kt_resolve)
219                        (krb5_context, krb5_const char *, krb5_keytab *)=NULL;
220 static krb5_error_code (_stdcall *p_krb5_kt_default)(krb5_context,
221                                                      krb5_keytab *)=NULL;
222 static krb5_error_code (_stdcall *p_krb5_free_ticket)(krb5_context,
223                                                       krb5_ticket *)=NULL;
224 static krb5_error_code (_stdcall *p_krb5_rd_req)(krb5_context,
225                                                  krb5_auth_context *,
226                                                  krb5_const krb5_data *,
227                                                  krb5_const_principal,
228                                                  krb5_keytab, krb5_flags *,
229                                                  krb5_ticket **)=NULL;
230 static krb5_error_code (_stdcall *p_krb5_mk_req_extended)
231                         (krb5_context, krb5_auth_context *,
232                          krb5_const krb5_flags, krb5_data *, krb5_creds *,
233                          krb5_data * )=NULL;
234 static krb5_error_code (_stdcall *p_krb5_init_context)(krb5_context *)=NULL;
235 static void (_stdcall *p_krb5_free_context)(krb5_context)=NULL;
236 static krb5_error_code (_stdcall *p_krb5_cc_default)(krb5_context,
237                                                      krb5_ccache  *)=NULL;
238 static krb5_error_code (_stdcall *p_krb5_sname_to_principal)
239                         (krb5_context, krb5_const char *, krb5_const char *,
240                          krb5_int32, krb5_principal *)=NULL;
241 static krb5_error_code (_stdcall *p_krb5_get_credentials)
242                         (krb5_context, krb5_const krb5_flags, krb5_ccache,
243                          krb5_creds *, krb5_creds **)=NULL;
244 static krb5_error_code (_stdcall *p_krb5_auth_con_init)
245                         (krb5_context, krb5_auth_context *)=NULL;
246 static krb5_error_code (_stdcall *p_krb5_cc_get_principal)
247                         (krb5_context context, krb5_ccache cache,
248                          krb5_principal *principal)=NULL;
249 static krb5_error_code (_stdcall *p_krb5_auth_con_free)
250                         (krb5_context, krb5_auth_context)=NULL;
251 static krb5_error_code (_stdcall *p_krb5_decrypt_tkt_part)
252                         (krb5_context, krb5_const krb5_keyblock *,
253                                        krb5_ticket *)=NULL;
254 static krb5_error_code (_stdcall *p_krb5_timeofday)
255                         (krb5_context context, krb5_int32 *timeret)=NULL;
256 static krb5_error_code (_stdcall *p_krb5_rc_default)
257                         (krb5_context context, krb5_rcache *rc)=NULL;
258 static krb5_error_code (_stdcall *p_krb5_rc_initialize)
259                         (krb5_context context, krb5_rcache rc,
```

```
260                                     krb5_deltat lifespan)=NULL;
261 static krb5_error_code (_stdcall *p_krb5_rc_get_lifespan)
262                          (krb5_context context, krb5_rcache rc,
263                           krb5_deltat *lifespan)=NULL;
264 static krb5_error_code (_stdcall *p_krb5_rc_destroy)
265                          (krb5_context context, krb5_rcache rc)=NULL;
266 static krb5_boolean (_stdcall *p_krb5_principal_compare)
267                     (krb5_context, krb5_const_principal, krb5_const_principal)=
268 static size_t (_stdcall *p_krb5_checksum_size)(krb5_context context,krb5_cksumty
269 static krb5_boolean (_stdcall *p_valid_cksumtype)(krb5_cksumtype ctype)=NULL;
270 static krb5_error_code (_stdcall *p_krb5_kt_free_entry)
271                          (krb5_context,krb5_keytab_entry * )=NULL;
272 static krb5_error_code (_stdcall * p_krb5_auth_con_setrcache)(krb5_context,
273                                                               krb5_auth_context
274                                                               krb5_rcache)=NULL
275 static krb5_error_code (_stdcall * p_krb5_get_server_rcache)(krb5_context,
276                                                             krb5_const krb5_da
277                                                             krb5_rcache *)=NUL
278 static krb5_error_code (* p_krb5_auth_con_getrcache)(krb5_context,
279                                                      krb5_auth_context,
280                                                      krb5_rcache *)=NULL;
281 static krb5_error_code (_stdcall * p_krb5_kt_close)(krb5_context context,
282                                                     krb5_keytab keytab)=NULL;
283 static krb5_error_code (_stdcall * p_krb5_kt_get_entry)(krb5_context context,
284                                                         krb5_keytab keytab,
285                       krb5_const_principal principal, krb5_kvno vno,
286                       krb5_enctype enctype, krb5_keytab_entry *entry)=NULL;
287 static int krb5_loaded = 0;     /* only attempt to initialize func ptrs once */

289 /* Function to Load the Kerberos 5 DLL and initialize function pointers */
290 void
291 load_krb5_dll(void)
292         {
293         HANDLE hKRB5_32;

295         krb5_loaded++;
296         hKRB5_32 = LoadLibrary(TEXT("KRB5_32"));
297         if (!hKRB5_32)
298                 return;

300         (FARPROC) p_krb5_free_data_contents =
301                 GetProcAddress( hKRB5_32, "krb5_free_data_contents" );
302         (FARPROC) p_krb5_free_context =
303                 GetProcAddress( hKRB5_32, "krb5_free_context" );
304         (FARPROC) p_krb5_auth_con_free =
305                 GetProcAddress( hKRB5_32, "krb5_auth_con_free" );
306         (FARPROC) p_krb5_free_principal =
307                 GetProcAddress( hKRB5_32, "krb5_free_principal" );
308         (FARPROC) p_krb5_mk_req_extended =
309                 GetProcAddress( hKRB5_32, "krb5_mk_req_extended" );
310         (FARPROC) p_krb5_get_credentials =
311                 GetProcAddress( hKRB5_32, "krb5_get_credentials" );
312         (FARPROC) p_krb5_cc_get_principal =
313                 GetProcAddress( hKRB5_32, "krb5_cc_get_principal" );
314         (FARPROC) p_krb5_cc_default =
315                 GetProcAddress( hKRB5_32, "krb5_cc_default" );
316         (FARPROC) p_krb5_sname_to_principal =
317                 GetProcAddress( hKRB5_32, "krb5_sname_to_principal" );
318         (FARPROC) p_krb5_init_context =
319                 GetProcAddress( hKRB5_32, "krb5_init_context" );
320         (FARPROC) p_krb5_free_ticket =
321                 GetProcAddress( hKRB5_32, "krb5_free_ticket" );
322         (FARPROC) p_krb5_rd_req =
323                 GetProcAddress( hKRB5_32, "krb5_rd_req" );
324         (FARPROC) p_krb5_principal_compare =
325                 GetProcAddress( hKRB5_32, "krb5_principal_compare" );
```



```
326         (FARPROC) p_krb5_decrypt_tkt_part =
327                 GetProcAddress( hKRB5_32, "krb5_decrypt_tkt_part" );
328         (FARPROC) p_krb5_timeofday =
329                 GetProcAddress( hKRB5_32, "krb5_timeofday" );
330         (FARPROC) p_krb5_rc_default =
331                 GetProcAddress( hKRB5_32, "krb5_rc_default" );
332         (FARPROC) p_krb5_rc_initialize =
333                 GetProcAddress( hKRB5_32, "krb5_rc_initialize" );
334         (FARPROC) p_krb5_rc_get_lifespan =
335                 GetProcAddress( hKRB5_32, "krb5_rc_get_lifespan" );
336         (FARPROC) p_krb5_rc_destroy =
337                 GetProcAddress( hKRB5_32, "krb5_rc_destroy" );
338         (FARPROC) p_krb5_kt_default =
339                 GetProcAddress( hKRB5_32, "krb5_kt_default" );
340         (FARPROC) p_krb5_kt_resolve =
341                 GetProcAddress( hKRB5_32, "krb5_kt_resolve" );
342         (FARPROC) p_krb5_auth_con_init =
343                 GetProcAddress( hKRB5_32, "krb5_auth_con_init" );
344         (FARPROC) p_valid_cksumtype =
345                 GetProcAddress( hKRB5_32, "valid_cksumtype" );
346         (FARPROC) p_krb5_checksum_size =
347                 GetProcAddress( hKRB5_32, "krb5_checksum_size" );
348         (FARPROC) p_krb5_kt_free_entry =
349                 GetProcAddress( hKRB5_32, "krb5_kt_free_entry" );
350         (FARPROC) p_krb5_auth_con_setrcache =
351                 GetProcAddress( hKRB5_32, "krb5_auth_con_setrcache" );
352         (FARPROC) p_krb5_get_server_rcache =
353                 GetProcAddress( hKRB5_32, "krb5_get_server_rcache" );
354         (FARPROC) p_krb5_auth_con_getrcache =
355                 GetProcAddress( hKRB5_32, "krb5_auth_con_getrcache" );
356         (FARPROC) p_krb5_kt_close =
357                 GetProcAddress( hKRB5_32, "krb5_kt_close" );
358         (FARPROC) p_krb5_kt_get_entry =
359                 GetProcAddress( hKRB5_32, "krb5_kt_get_entry" );
360         }

362 /* Stubs for each function to be dynamicly loaded */
363 void
364 kssl_krb5_free_data_contents(krb5_context CO, krb5_data  * data)
365         {
366         if (!krb5_loaded)
367                 load_krb5_dll();

369         if ( p_krb5_free_data_contents )
370                 p_krb5_free_data_contents(CO,data);
371         }

373 krb5_error_code
374 kssl_krb5_mk_req_extended (krb5_context CO,
375                            krb5_auth_context  * pACO,
376                            krb5_const krb5_flags F,
377                            krb5_data  * pD1,
378                            krb5_creds  * pC,
379                            krb5_data  * pD2)
380         {
381         if (!krb5_loaded)
382                 load_krb5_dll();

384         if ( p_krb5_mk_req_extended )
385                 return(p_krb5_mk_req_extended(CO,pACO,F,pD1,pC,pD2));
386         else
387                 return KRB5KRB_ERR_GENERIC;
388         }
389 krb5_error_code
390 kssl_krb5_auth_con_init(krb5_context CO,
391                         krb5_auth_context  * pACO)
```

```
392         {
393         if (!krb5_loaded)
394                 load_krb5_dll();

396         if ( p_krb5_auth_con_init )
397                 return(p_krb5_auth_con_init(CO,pACO));
398         else
399                 return KRB5KRB_ERR_GENERIC;
400         }
401 krb5_error_code
402 kssl_krb5_auth_con_free (krb5_context CO,
403                          krb5_auth_context ACO)
404         {
405         if (!krb5_loaded)
406                 load_krb5_dll();

408         if ( p_krb5_auth_con_free )
409                 return(p_krb5_auth_con_free(CO,ACO));
410         else
411                 return KRB5KRB_ERR_GENERIC;
412         }
413 krb5_error_code
414 kssl_krb5_get_credentials(krb5_context CO,
415                           krb5_const krb5_flags F,
416                           krb5_ccache CC,
417                           krb5_creds  * pCR,
418                           krb5_creds  ** ppCR)
419         {
420         if (!krb5_loaded)
421                 load_krb5_dll();

423         if ( p_krb5_get_credentials )
424                 return(p_krb5_get_credentials(CO,F,CC,pCR,ppCR));
425         else
426                 return KRB5KRB_ERR_GENERIC;
427         }
428 krb5_error_code
429 kssl_krb5_sname_to_principal(krb5_context CO,
430                              krb5_const char  * pC1,
431                              krb5_const char  * pC2,
432                              krb5_int32 I,
433                              krb5_principal  * pPR)
434         {
435         if (!krb5_loaded)
436                 load_krb5_dll();

438         if ( p_krb5_sname_to_principal )
439                 return(p_krb5_sname_to_principal(CO,pC1,pC2,I,pPR));
440         else
441                 return KRB5KRB_ERR_GENERIC;
442         }

444 krb5_error_code
445 kssl_krb5_cc_default(krb5_context CO,
446                      krb5_ccache  * pCC)
447         {
448         if (!krb5_loaded)
449                 load_krb5_dll();

451         if ( p_krb5_cc_default )
452                 return(p_krb5_cc_default(CO,pCC));
453         else
454                 return KRB5KRB_ERR_GENERIC;
455         }

457 krb5_error_code
```



```
458 kssl_krb5_init_context(krb5_context * pCO)
459         {
460         if (!krb5_loaded)
461                 load_krb5_dll();

463         if ( p_krb5_init_context )
464                 return(p_krb5_init_context(pCO));
465         else
466                 return KRB5KRB_ERR_GENERIC;
467         }

469 void
470 kssl_krb5_free_context(krb5_context CO)
471         {
472         if (!krb5_loaded)
473                 load_krb5_dll();

475         if ( p_krb5_free_context )
476                 p_krb5_free_context(CO);
477         }

479 void
480 kssl_krb5_free_principal(krb5_context c, krb5_principal p)
481         {
482         if (!krb5_loaded)
483                 load_krb5_dll();

485         if ( p_krb5_free_principal )
486                 p_krb5_free_principal(c,p);
487         }

489 krb5_error_code
490 kssl_krb5_kt_resolve(krb5_context con,
491                      krb5_const char * sz,
492                      krb5_keytab * kt)
493         {
494         if (!krb5_loaded)
495                 load_krb5_dll();

497         if ( p_krb5_kt_resolve )
498                 return(p_krb5_kt_resolve(con,sz,kt));
499         else
500                 return KRB5KRB_ERR_GENERIC;
501         }

503 krb5_error_code
504 kssl_krb5_kt_default(krb5_context con,
505                      krb5_keytab * kt)
506         {
507         if (!krb5_loaded)
508                 load_krb5_dll();

510         if ( p_krb5_kt_default )
511                 return(p_krb5_kt_default(con,kt));
512         else
513                 return KRB5KRB_ERR_GENERIC;
514         }

516 krb5_error_code
517 kssl_krb5_free_ticket(krb5_context con,
518                       krb5_ticket * kt)
519         {
520         if (!krb5_loaded)
521                 load_krb5_dll();

523         if ( p_krb5_free_ticket )
```

```
524                 return(p_krb5_free_ticket(con,kt));
525         else
526                 return KRB5KRB_ERR_GENERIC;
527         }

529 krb5_error_code
530 kssl_krb5_rd_req(krb5_context con, krb5_auth_context * pacon,
531                 krb5_const krb5_data * data,
532                 krb5_const_principal princ, krb5_keytab keytab,
533                 krb5_flags * flags, krb5_ticket ** pptkt)
534         {
535         if (!krb5_loaded)
536                 load_krb5_dll();

538         if ( p_krb5_rd_req )
539                 return(p_krb5_rd_req(con,pacon,data,princ,keytab,flags,pptkt));
540         else
541                 return KRB5KRB_ERR_GENERIC;
542         }

544 krb5_boolean
545 krb5_principal_compare(krb5_context con, krb5_const_principal princ1,
546                 krb5_const_principal princ2)
547         {
548         if (!krb5_loaded)
549                 load_krb5_dll();

551         if ( p_krb5_principal_compare )
552                 return(p_krb5_principal_compare(con,princ1,princ2));
553         else
554                 return KRB5KRB_ERR_GENERIC;
555         }

557 krb5_error_code
558 krb5_decrypt_tkt_part(krb5_context con, krb5_const krb5_keyblock *keys,
559                 krb5_ticket *ticket)
560         {
561         if (!krb5_loaded)
562                 load_krb5_dll();

564         if ( p_krb5_decrypt_tkt_part )
565                 return(p_krb5_decrypt_tkt_part(con,keys,ticket));
566         else
567                 return KRB5KRB_ERR_GENERIC;
568         }

570 krb5_error_code
571 krb5_timeofday(krb5_context con, krb5_int32 *timeret)
572         {
573         if (!krb5_loaded)
574                 load_krb5_dll();

576         if ( p_krb5_timeofday )
577                 return(p_krb5_timeofday(con,timeret));
578         else
579                 return KRB5KRB_ERR_GENERIC;
580         }

582 krb5_error_code
583 krb5_rc_default(krb5_context con, krb5_rcache *rc)
584         {
585         if (!krb5_loaded)
586                 load_krb5_dll();

588         if ( p_krb5_rc_default )
589                 return(p_krb5_rc_default(con,rc));
```



```
590         else
591                 return KRB5KRB_ERR_GENERIC;
592         }

594 krb5_error_code
595 krb5_rc_initialize(krb5_context con, krb5_rcache rc, krb5_deltat lifespan)
596         {
597         if (!krb5_loaded)
598                 load_krb5_dll();

600         if ( p_krb5_rc_initialize )
601                 return(p_krb5_rc_initialize(con, rc, lifespan));
602         else
603                 return KRB5KRB_ERR_GENERIC;
604         }

606 krb5_error_code
607 krb5_rc_get_lifespan(krb5_context con, krb5_rcache rc, krb5_deltat *lifespanp)
608         {
609         if (!krb5_loaded)
610                 load_krb5_dll();

612         if ( p_krb5_rc_get_lifespan )
613                 return(p_krb5_rc_get_lifespan(con, rc, lifespanp));
614         else
615                 return KRB5KRB_ERR_GENERIC;
616         }

618 krb5_error_code
619 krb5_rc_destroy(krb5_context con, krb5_rcache rc)
620         {
621         if (!krb5_loaded)
622                 load_krb5_dll();

624         if ( p_krb5_rc_destroy )
625                 return(p_krb5_rc_destroy(con, rc));
626         else
627                 return KRB5KRB_ERR_GENERIC;
628         }

630 size_t
631 krb5_checksum_size(krb5_context context,krb5_cksumtype ctype)
632         {
633         if (!krb5_loaded)
634                 load_krb5_dll();

636         if ( p_krb5_checksum_size )
637                 return(p_krb5_checksum_size(context, ctype));
638         else
639                 return KRB5KRB_ERR_GENERIC;
640         }

642 krb5_boolean
643 valid_cksumtype(krb5_cksumtype ctype)
644         {
645         if (!krb5_loaded)
646                 load_krb5_dll();

648         if ( p_valid_cksumtype )
649                 return(p_valid_cksumtype(ctype));
650         else
651                 return KRB5KRB_ERR_GENERIC;
652         }

654 krb5_error_code
655 krb5_kt_free_entry(krb5_context con,krb5_keytab_entry * entry)
```

```
656         {
657         if (!krb5_loaded)
658                 load_krb5_dll();

660         if ( p_krb5_kt_free_entry )
661                 return(p_krb5_kt_free_entry(con,entry));
662         else
663                 return KRB5KRB_ERR_GENERIC;
664         }

666 /* Structure definitions  */
667 #ifndef NO_DEF_KRB5_CCACHE
668 #ifndef krb5_x
669 #define krb5_x(ptr,args) ((ptr)?((*(ptr)) args):(abort(),1))
670 #define krb5_xc(ptr,args) ((ptr)?((*(ptr)) args):(abort(),(char*)0))
671 #endif

673 typedef krb5_pointer    krb5_cc_cursor; /* cursor for sequential lookup */

675 typedef struct _krb5_ccache
676         {
677         krb5_magic magic;
678         struct _krb5_cc_ops FAR *ops;
679         krb5_pointer data;
680         } *krb5_ccache;

682 typedef struct _krb5_cc_ops
683         {
684         krb5_magic magic;
685         char  *prefix;
686         char  * (KRB5_CALLCONV *get_name)
687                 (krb5_context, krb5_ccache);
688         krb5_error_code (KRB5_CALLCONV *resolve)
689                 (krb5_context, krb5_ccache  *, const char  *);
690         krb5_error_code (KRB5_CALLCONV *gen_new)
691                 (krb5_context, krb5_ccache  *);
692         krb5_error_code (KRB5_CALLCONV *init)
693                 (krb5_context, krb5_ccache, krb5_principal);
694         krb5_error_code (KRB5_CALLCONV *destroy)
695                 (krb5_context, krb5_ccache);
696         krb5_error_code (KRB5_CALLCONV *close)
697                 (krb5_context, krb5_ccache);
698         krb5_error_code (KRB5_CALLCONV *store)
699                 (krb5_context, krb5_ccache, krb5_creds  *);
700         krb5_error_code (KRB5_CALLCONV *retrieve)
701                 (krb5_context, krb5_ccache,
702                 krb5_flags, krb5_creds  *, krb5_creds  *);
703         krb5_error_code (KRB5_CALLCONV *get_princ)
704                 (krb5_context, krb5_ccache, krb5_principal  *);
705         krb5_error_code (KRB5_CALLCONV *get_first)
706                 (krb5_context, krb5_ccache, krb5_cc_cursor  *);
707         krb5_error_code (KRB5_CALLCONV *get_next)
708                 (krb5_context, krb5_ccache,
709                 krb5_cc_cursor  *, krb5_creds  *);
710         krb5_error_code (KRB5_CALLCONV *end_get)
711                 (krb5_context, krb5_ccache, krb5_cc_cursor  *);
712         krb5_error_code (KRB5_CALLCONV *remove_cred)
713                 (krb5_context, krb5_ccache,
714                 krb5_flags, krb5_creds  *);
715         krb5_error_code (KRB5_CALLCONV *set_flags)
716                 (krb5_context, krb5_ccache, krb5_flags);
717         } krb5_cc_ops;
718 #endif /* NO_DEF_KRB5_CCACHE */

720 krb5_error_code
721 kssl_krb5_cc_get_principal
```



```
722     (krb5_context context, krb5_ccache cache,
723       krb5_principal *principal)
724         {
725         if ( p_krb5_cc_get_principal )
726                 return(p_krb5_cc_get_principal(context,cache,principal));
727         else
728                 return(krb5_x
729                         ((cache)->ops->get_princ,(context, cache, principal)));
730         }

732 krb5_error_code
733 kssl_krb5_auth_con_setrcache(krb5_context con, krb5_auth_context acon,
734                              krb5_rcache rcache)
735         {
736         if ( p_krb5_auth_con_setrcache )
737                  return(p_krb5_auth_con_setrcache(con,acon,rcache));
738         else
739                  return KRB5KRB_ERR_GENERIC;
740         }

742 krb5_error_code
743 kssl_krb5_get_server_rcache(krb5_context con, krb5_const krb5_data * data,
744                             krb5_rcache * rcache)
745         {
746         if ( p_krb5_get_server_rcache )
747                 return(p_krb5_get_server_rcache(con,data,rcache));
748         else
749                 return KRB5KRB_ERR_GENERIC;
750         }

752 krb5_error_code
753 kssl_krb5_auth_con_getrcache(krb5_context con, krb5_auth_context acon,
754                              krb5_rcache * prcache)
755         {
756         if ( p_krb5_auth_con_getrcache )
757                 return(p_krb5_auth_con_getrcache(con,acon, prcache));
758         else
759                 return KRB5KRB_ERR_GENERIC;
760         }

762 krb5_error_code
763 kssl_krb5_kt_close(krb5_context context, krb5_keytab keytab)
764         {
765         if ( p_krb5_kt_close )
766                 return(p_krb5_kt_close(context,keytab));
767         else
768                 return KRB5KRB_ERR_GENERIC;
769         }

771 krb5_error_code
772 kssl_krb5_kt_get_entry(krb5_context context, krb5_keytab keytab,
773                        krb5_const_principal principal, krb5_kvno vno,
774                        krb5_enctype enctype, krb5_keytab_entry *entry)
775         {
776         if ( p_krb5_kt_get_entry )
777                 return(p_krb5_kt_get_entry(context,keytab,principal,vno,enctype,
778         else
779                 return KRB5KRB_ERR_GENERIC;
780         }
781 #endif  /* OPENSSL_SYS_WINDOWS || OPENSSL_SYS_WIN32 */


784 /* memory allocation functions for non-temporary storage
785  * (e.g. stuff that gets saved into the kssl context) */
786 static void* kssl_calloc(size_t nmemb, size_t size)
787 {
```

```
788         void* p;

790         p=OPENSSL_malloc(nmemb*size);
791         if (p){
792                 memset(p, 0, nmemb*size);
793         }
794         return p;
795 }

797 #define kssl_malloc(size) OPENSSL_malloc((size))
798 #define kssl_realloc(ptr, size) OPENSSL_realloc(ptr, size)
799 #define kssl_free(ptr) OPENSSL_free((ptr))


802 char
803 *kstring(char *string)
804         {
805         static char     *null = "[NULL]";

807         return ((string == NULL)? null: string);
808         }

810 /*      Given KRB5 enctype (basically DES or 3DES),
811 **      return closest match openssl EVP_ encryption algorithm.
812 **      Return NULL for unknown or problematic (krb5_dk_encrypt) enctypes.
813 **      Assume ENCTYPE_*_RAW (krb5_raw_encrypt) are OK.
814 */
815 const EVP_CIPHER *
816 kssl_map_enc(krb5_enctype enctype)
817         {
818         switch (enctype)
819                 {
820         case ENCTYPE_DES_HMAC_SHA1:             /*    EVP_des_cbc();       */
821         case ENCTYPE_DES_CBC_CRC:
822         case ENCTYPE_DES_CBC_MD4:
823         case ENCTYPE_DES_CBC_MD5:
824         case ENCTYPE_DES_CBC_RAW:
825                                 return EVP_des_cbc();
826                                 break;
827         case ENCTYPE_DES3_CBC_SHA1:             /*    EVP_des_ede3_cbc();  */
828         case ENCTYPE_DES3_CBC_SHA:
829         case ENCTYPE_DES3_CBC_RAW:
830                                 return EVP_des_ede3_cbc();
831                                 break;
832         default:                return NULL;
833                                 break;
834                 }
835         }


838 /*      Return true:1 if p "looks like" the start of the real authenticator
839 **      described in kssl_skip_confound() below.  The ASN.1 pattern is
840 **      "62 xx 30 yy" (APPLICATION-2, SEQUENCE), where xx-yy =~ 2, and
841 **      xx and yy are possibly multi-byte length fields.
842 */
843 static int      kssl_test_confound(unsigned char *p)
844         {
845         int     len = 2;
846         int     xx = 0, yy = 0;

848         if (*p++ != 0x62)  return 0;
849         if (*p > 0x82)  return 0;
850         switch(*p)  {
851                 case 0x82:  p++;          xx = (*p++ << 8);  xx += *p++;  break;
852                 case 0x81:  p++;          xx =  *p++;  break;
853                 case 0x80:  return 0;
```



```
854                 default:    xx = *p++;  break;
855                 }
856         if (*p++ != 0x30)  return 0;
857         if (*p > 0x82)  return 0;
858         switch(*p)  {
859                 case 0x82:  p++; len+=2;  yy = (*p++ << 8);  yy += *p++;  break;
860                 case 0x81:  p++; len++;   yy =  *p++;  break;
861                 case 0x80:  return 0;
862                 default:    yy = *p++;  break;
863                 }

865         return (xx - len == yy)? 1: 0;
866         }

868 /*      Allocate, fill, and return cksumlens array of checksum lengths.
869 **      This array holds just the unique elements from the krb5_cksumarray[].
870 **      array[n] == 0 signals end of data.
871 **
872 **      The krb5_cksumarray[] was an internal variable that has since been
873 **      replaced by a more general method for storing the data.  It should
874 **      not be used.  Instead we use real API calls and make a guess for
875 **      what the highest assigned CKSUMTYPE_ constant is.  As of 1.2.2
876 **      it is 0x000c (CKSUMTYPE_HMAC_SHA1_DES3).  So we will use 0x0010.
877 */
878 static size_t  *populate_cksumlens(void)
879         {
880         int             i, j, n;
881         static size_t   *cklens = NULL;

883 #ifdef KRB5_MIT_OLD11
884         n = krb5_max_cksum;
885 #else
886         n = 0x0010;
887 #endif  /* KRB5_MIT_OLD11 */

889 #ifdef KRB5CHECKAUTH
890         if (!cklens && !(cklens = (size_t *) calloc(sizeof(int),n+1)))  return N

892         for (i=0; i < n; i++)  {
893                 if (!valid_cksumtype(i))  continue;     /*  array has holes  */
894                 for (j=0; j < n; j++)  {
895                         if (cklens[j] == 0)  {
896                                 cklens[j] = krb5_checksum_size(NULL,i);
897                                 break;          /*  krb5 elem was new: add   */
898                                 }
899                         if (cklens[j] == krb5_checksum_size(NULL,i))  {
900                                 break;          /*  ignore duplicate elements */
901                                 }
902                         }
903                 }
904 #endif  /* KRB5CHECKAUTH */

906         return cklens;
907         }

909 /*      Return pointer to start of real authenticator within authenticator, or
910 **      return NULL on error.
911 **      Decrypted authenticator looks like this:
912 **              [0 or 8 byte confounder] [4-24 byte checksum] [real authent'r]
913 **      This hackery wouldn't be necessary if MIT KRB5 1.0.6 had the
914 **      krb5_auth_con_getcksumtype() function advertised in its krb5.h.
915 */
916 unsigned char   *kssl_skip_confound(krb5_enctype etype, unsigned char *a)
917         {
918         int             i, conlen;
919         size_t          cklen;
```

```
920         static size_t   *cksumlens = NULL;
921         unsigned char   *test_auth;

923         conlen = (etype)? 8: 0;

925         if (!cksumlens  &&  !(cksumlens = populate_cksumlens()))  return NULL;
926         for (i=0; (cklen = cksumlens[i]) != 0; i++)
927                 {
928                 test_auth = a + conlen + cklen;
929                 if (kssl_test_confound(test_auth))  return test_auth;
930                 }

932         return NULL;
933         }


936 /*      Set kssl_err error info when reason text is a simple string
937 **              kssl_err = struct { int reason; char text[KSSL_ERR_MAX+1]; }
938 */
939 void
940 kssl_err_set(KSSL_ERR *kssl_err, int reason, char *text)
941         {
942         if (kssl_err == NULL)  return;

944         kssl_err->reason = reason;
945         BIO_snprintf(kssl_err->text, KSSL_ERR_MAX, "%s", text);
946         return;
947         }


950 /*      Display contents of krb5_data struct, for debugging
951 */
952 void
953 print_krb5_data(char *label, krb5_data *kdata)
954         {
955         int i;

957         printf("%s[%d] ", label, kdata->length);
958         for (i=0; i < (int)kdata->length; i++)
959                 {
960                 if (0 &&  isprint((int) kdata->data[i]))
961                         printf( "%c ",  kdata->data[i]);
962                 else
963                         printf( "%02x ", (unsigned char) kdata->data[i]);
964                 }
965         printf("\n");
966         }


969 /*      Display contents of krb5_authdata struct, for debugging
970 */
971 void
972 print_krb5_authdata(char *label, krb5_authdata **adata)
973         {
974         if (adata == NULL)
975                 {
976                 printf("%s, authdata==0\n", label);
977                 return;
978                 }
979         printf("%s [%p]\n", label, (void *)adata);
980 #if 0
981         {
982         int     i;
983         printf("%s[at%d:%d] ", label, adata->ad_type, adata->length);
984         for (i=0; i < adata->length; i++)
985                 {
```



```
986                 printf((isprint(adata->contents[i]))? "%c ": "%02x",
987                         adata->contents[i]);
988                 }
989         printf("\n");
990         }
991 #endif
992         }


995 /*      Display contents of krb5_keyblock struct, for debugging
996 */
997 void
998 print_krb5_keyblock(char *label, krb5_keyblock *keyblk)
999         {
1000        int i;

1002        if (keyblk == NULL)
1003                {
1004                printf("%s, keyblk==0\n", label);
1005                return;
1006                }
1007 #ifdef KRB5_HEIMDAL
1008        printf("%s\n\t[et%d:%d]: ", label, keyblk->keytype,
1009                                            keyblk->keyvalue->length);
1010        for (i=0; i < (int)keyblk->keyvalue->length; i++)
1011                {
1012                printf("%02x",(unsigned char *)(keyblk->keyvalue->contents)[i]);
1013                }
1014        printf("\n");
1015 #else
1016        printf("%s\n\t[et%d:%d]: ", label, keyblk->enctype, keyblk->length);
1017        for (i=0; i < (int)keyblk->length; i++)
1018                {
1019                printf("%02x",keyblk->contents[i]);
1020                }
1021        printf("\n");
1022 #endif
1023        }


1026 /*      Display contents of krb5_principal_data struct, for debugging
1027 **      (krb5_principal is typedef'd == krb5_principal_data *)
1028 */
1029 static void
1030 print_krb5_princ(char *label, krb5_principal_data *princ)
1031        {
1032        int i, ui, uj;

1034        printf("%s principal Realm: ", label);
1035        if (princ == NULL)  return;
1036        for (ui=0; ui < (int)princ->realm.length; ui++)  putchar(princ->realm.da
1037        printf(" (nametype %d) has %d strings:\n", princ->type,princ->length);
1038        for (i=0; i < (int)princ->length; i++)
1039                {
1040                printf("\t%d [%d]: ", i, princ->data[i].length);
1041                for (uj=0; uj < (int)princ->data[i].length; uj++)  {
1042                        putchar(princ->data[i].data[uj]);
1043                        }
1044                printf("\n");
1045                }
1046        return;
1047        }


1050 /*      Given krb5 service (typically "kssl") and hostname in kssl_ctx,
1051 **      Return encrypted Kerberos ticket for service @ hostname.
```

```
1052 **      If authenp is non-NULL, also return encrypted authenticator,
1053 **      whose data should be freed by caller.
1054 **      (Originally was: Create Kerberos AP_REQ message for SSL Client.)
1055 **
1056 **      19990628        VRS     Started; Returns Kerberos AP_REQ message.
1057 **      20010409        VRS     Modified for RFC2712; Returns enc tkt.
1058 **      20010606        VRS     May also return optional authenticator.
1059 */
1060 krb5_error_code
1061 kssl_cget_tkt(  /* UPDATE */    KSSL_CTX *kssl_ctx,
1062                 /* OUT    */    krb5_data **enc_ticketp,
1063                 /* UPDATE */    krb5_data *authenp,
1064                 /* OUT    */    KSSL_ERR *kssl_err)
1065         {
1066         krb5_error_code         krb5rc = KRB5KRB_ERR_GENERIC;
1067         krb5_context            krb5context = NULL;
1068         krb5_auth_context       krb5auth_context = NULL;
1069         krb5_ccache             krb5ccdef = NULL;
1070         krb5_creds              krb5creds, *krb5credsp = NULL;
1071         krb5_data               krb5_app_req;

1073         kssl_err_set(kssl_err, 0, "");
1074         memset((char *)&krb5creds, 0, sizeof(krb5creds));

1076         if (!kssl_ctx)
1077                 {
1078                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1079                         "No kssl_ctx defined.\n");
1080                 goto err;
1081                 }
1082         else if (!kssl_ctx->service_host)
1083                 {
1084                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1085                         "kssl_ctx service_host undefined.\n");
1086                 goto err;
1087                 }

1089         if ((krb5rc = krb5_init_context(&krb5context)) != 0)
1090                 {
1091                 BIO_snprintf(kssl_err->text,KSSL_ERR_MAX,
1092                         "krb5_init_context() fails: %d\n", krb5rc);
1093                 kssl_err->reason = SSL_R_KRB5_C_INIT;
1094                 goto err;
1095                 }

1097         if ((krb5rc = krb5_sname_to_principal(krb5context,
1098                 kssl_ctx->service_host,
1099                 (kssl_ctx->service_name)? kssl_ctx->service_name: KRB5SVC,
1100                 KRB5_NT_SRV_HST, &krb5creds.server)) != 0)
1101                 {
1102                 BIO_snprintf(kssl_err->text,KSSL_ERR_MAX,
1103                         "krb5_sname_to_principal() fails for %s/%s\n",
1104                         kssl_ctx->service_host,
1105                         (kssl_ctx->service_name)? kssl_ctx->service_name:
1106                                                   KRB5SVC);
1107                 kssl_err->reason = SSL_R_KRB5_C_INIT;
1108                 goto err;
1109                 }

1111         if ((krb5rc = krb5_cc_default(krb5context, &krb5ccdef)) != 0)
1112                 {
1113                 kssl_err_set(kssl_err, SSL_R_KRB5_C_CC_PRINC,
1114                         "krb5_cc_default fails.\n");
1115                 goto err;
1116                 }
```



```
1118         if ((krb5rc = krb5_cc_get_principal(krb5context, krb5ccdef,
1119                 &krb5creds.client)) != 0)
1120                 {
1121                 kssl_err_set(kssl_err, SSL_R_KRB5_C_CC_PRINC,
1122                         "krb5_cc_get_principal() fails.\n");
1123                 goto err;
1124                 }

1126         if ((krb5rc = krb5_get_credentials(krb5context, 0, krb5ccdef,
1127                 &krb5creds, &krb5credsp)) != 0)
1128                 {
1129                 kssl_err_set(kssl_err, SSL_R_KRB5_C_GET_CRED,
1130                         "krb5_get_credentials() fails.\n");
1131                 goto err;
1132                 }

1134         *enc_ticketp = &krb5credsp->ticket;
1135 #ifdef KRB5_HEIMDAL
1136         kssl_ctx->enctype = krb5credsp->session.keytype;
1137 #else
1138         kssl_ctx->enctype = krb5credsp->keyblock.enctype;
1139 #endif

1141         krb5rc = KRB5KRB_ERR_GENERIC;
1142         /*      caller should free data of krb5_app_req  */
1143         /*  20010406 VRS deleted for real KerberosWrapper
1144         **  20010605 VRS reinstated to offer Authenticator to KerberosWrapper
1145         */
1146         krb5_app_req.length = 0;
1147         if (authenp)
1148                 {
1149                 krb5_data       krb5in_data;
1150                 const unsigned char     *p;
1151                 long            arlen;
1152                 KRB5_APREQBODY  *ap_req;

1154                 authenp->length = 0;
1155                 krb5in_data.data = NULL;
1156                 krb5in_data.length = 0;
1157                 if ((krb5rc = krb5_mk_req_extended(krb5context,
1158                         &krb5auth_context, 0, &krb5in_data, krb5credsp,
1159                         &krb5_app_req)) != 0)
1160                         {
1161                         kssl_err_set(kssl_err, SSL_R_KRB5_C_MK_REQ,
1162                                 "krb5_mk_req_extended() fails.\n");
1163                         goto err;
1164                         }

1166                 arlen = krb5_app_req.length;
1167                 p = (unsigned char *)krb5_app_req.data;
1168                 ap_req = (KRB5_APREQBODY *) d2i_KRB5_APREQ(NULL, &p, arlen);
1169                 if (ap_req)
1170                         {
1171                         authenp->length = i2d_KRB5_ENCDATA(
1172                                         ap_req->authenticator, NULL);
1173                         if (authenp->length  &&
1174                                 (authenp->data = malloc(authenp->length)))
1175                                 {
1176                                 unsigned char   *adp = (unsigned char *)authenp-
1177                                 authenp->length = i2d_KRB5_ENCDATA(
1178                                                 ap_req->authenticator, &adp);
1179                                 }
1180                         }

1182                 if (ap_req)  KRB5_APREQ_free((KRB5_APREQ *) ap_req);
1183                 if (krb5_app_req.length)
```

```
1184                        kssl_krb5_free_data_contents(krb5context,&krb5_app_req);
1185                }
1186 #ifdef KRB5_HEIMDAL
1187         if (kssl_ctx_setkey(kssl_ctx, &krb5credsp->session))
1188                 {
1189                 kssl_err_set(kssl_err, SSL_R_KRB5_C_INIT,
1190                         "kssl_ctx_setkey() fails.\n");
1191                 }
1192 #else
1193         if (kssl_ctx_setkey(kssl_ctx, &krb5credsp->keyblock))
1194                 {
1195                 kssl_err_set(kssl_err, SSL_R_KRB5_C_INIT,
1196                         "kssl_ctx_setkey() fails.\n");
1197                 }
1198 #endif
1199         else    krb5rc = 0;

1201  err:
1202 #ifdef KSSL_DEBUG
1203         kssl_ctx_show(kssl_ctx);
1204 #endif  /* KSSL_DEBUG */

1206         if (krb5creds.client)   krb5_free_principal(krb5context,
1207                                                     krb5creds.client);
1208         if (krb5creds.server)   krb5_free_principal(krb5context,
1209                                                     krb5creds.server);
1210         if (krb5auth_context)   krb5_auth_con_free(krb5context,
1211                                                    krb5auth_context);
1212         if (krb5context)        krb5_free_context(krb5context);
1213         return (krb5rc);
1214         }


1217 /*  Given d2i_-decoded asn1ticket, allocate and return a new krb5_ticket.
1218 **  Return Kerberos error code and kssl_err struct on error.
1219 **  Allocates krb5_ticket and krb5_principal; caller should free these.
1220 **
1221 **      20010410        VRS     Implemented krb5_decode_ticket() as
1222 **                              old_krb5_decode_ticket(). Missing from MIT1.0.6.
1223 **      20010615        VRS     Re-cast as openssl/asn1 d2i_*() functions.
1224 **                              Re-used some of the old krb5_decode_ticket()
1225 **                              code here.  This tkt should alloc/free just
1226 **                              like the real thing.
1227 */
1228 static krb5_error_code
1229 kssl_TKT2tkt(   /* IN     */    krb5_context    krb5context,
1230                 /* IN     */    KRB5_TKTBODY    *asn1ticket,
1231                 /* OUT    */    krb5_ticket     **krb5ticket,
1232                 /* OUT    */    KSSL_ERR *kssl_err  )
1233         {
1234         krb5_error_code                 krb5rc = KRB5KRB_ERR_GENERIC;
1235         krb5_ticket                     *new5ticket = NULL;
1236         ASN1_GENERALSTRING              *gstr_svc, *gstr_host;

1238         *krb5ticket = NULL;

1240         if (asn1ticket == NULL  ||  asn1ticket->realm == NULL  ||
1241                 asn1ticket->sname == NULL  ||
1242                 sk_ASN1_GENERALSTRING_num(asn1ticket->sname->namestring) < 2)
1243                 {
1244                 BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1245                         "Null field in asn1ticket.\n");
1246                 kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1247                 return KRB5KRB_ERR_GENERIC;
1248                 }
```



```
1250         if ((new5ticket = (krb5_ticket *) calloc(1, sizeof(krb5_ticket)))==NULL)
1251                 {
1252                 BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1253                         "Unable to allocate new krb5_ticket.\n");
1254                 kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1255                 return ENOMEM;          /*  or  KRB5KRB_ERR_GENERIC;    */
1256                 }

1258         gstr_svc  = sk_ASN1_GENERALSTRING_value(asn1ticket->sname->namestring, 0
1259         gstr_host = sk_ASN1_GENERALSTRING_value(asn1ticket->sname->namestring, 1

1261         if ((krb5rc = kssl_build_principal_2(krb5context,
1262                         &new5ticket->server,
1263                         asn1ticket->realm->length, (char *)asn1ticket->realm->da
1264                         gstr_svc->length,  (char *)gstr_svc->data,
1265                         gstr_host->length, (char *)gstr_host->data)) != 0)
1266                 {
1267                 free(new5ticket);
1268                 BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1269                         "Error building ticket server principal.\n");
1270                 kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1271                 return krb5rc;          /*  or  KRB5KRB_ERR_GENERIC;    */
1272                 }

1274         krb5_princ_type(krb5context, new5ticket->server) =
1275                         asn1ticket->sname->nametype->data[0];
1276         new5ticket->enc_part.enctype = asn1ticket->encdata->etype->data[0];
1277         new5ticket->enc_part.kvno = asn1ticket->encdata->kvno->data[0];
1278         new5ticket->enc_part.ciphertext.length =
1279                         asn1ticket->encdata->cipher->length;
1280         if ((new5ticket->enc_part.ciphertext.data =
1281                 calloc(1, asn1ticket->encdata->cipher->length)) == NULL)
1282                 {
1283                 free(new5ticket);
1284                 BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1285                         "Error allocating cipher in krb5ticket.\n");
1286                 kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1287                 return KRB5KRB_ERR_GENERIC;
1288                 }
1289         else
1290                 {
1291                 memcpy(new5ticket->enc_part.ciphertext.data,
1292                         asn1ticket->encdata->cipher->data,
1293                         asn1ticket->encdata->cipher->length);
1294                 }

1296         *krb5ticket = new5ticket;
1297         return 0;
1298         }


1301 /*      Given krb5 service name in KSSL_CTX *kssl_ctx (typically "kssl"),
1302 **              and krb5 AP_REQ message & message length,
1303 **      Return Kerberos session key and client principle
1304 **              to SSL Server in KSSL_CTX *kssl_ctx.
1305 **
1306 **      19990702        VRS     Started.
1307 */
1308 krb5_error_code
1309 kssl_sget_tkt(  /* UPDATE */    KSSL_CTX                *kssl_ctx,
1310                 /* IN     */    krb5_data               *indata,
1311                 /* OUT    */    krb5_ticket_times       *ttimes,
1312                 /* OUT    */    KSSL_ERR                *kssl_err  )
1313         {
1314         krb5_error_code                 krb5rc = KRB5KRB_ERR_GENERIC;
1315         static krb5_context             krb5context = NULL;
```

```
1316         static krb5_auth_context        krb5auth_context = NULL;
1317         krb5_ticket                     *krb5ticket = NULL;
1318         KRB5_TKTBODY                    *asn1ticket = NULL;
1319         const unsigned char             *p;
1320         krb5_keytab                     krb5keytab = NULL;
1321         krb5_keytab_entry               kt_entry;
1322         krb5_principal                  krb5server;
1323         krb5_rcache                     rcache = NULL;

1325         kssl_err_set(kssl_err, 0, "");

1327         if (!kssl_ctx)
1328                 {
1329                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1330                         "No kssl_ctx defined.\n");
1331                 goto err;
1332                 }

1334 #ifdef KSSL_DEBUG
1335         printf("in kssl_sget_tkt(%s)\n", kstring(kssl_ctx->service_name));
1336 #endif  /* KSSL_DEBUG */

1338         if (!krb5context  &&  (krb5rc = krb5_init_context(&krb5context)))
1339                 {
1340                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1341                         "krb5_init_context() fails.\n");
1342                 goto err;
1343                 }
1344         if (krb5auth_context  &&
1345                 (krb5rc = krb5_auth_con_free(krb5context, krb5auth_context)))
1346                 {
1347                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1348                         "krb5_auth_con_free() fails.\n");
1349                 goto err;
1350                 }
1351         else  krb5auth_context = NULL;
1352         if (!krb5auth_context  &&
1353                 (krb5rc = krb5_auth_con_init(krb5context, &krb5auth_context)))
1354                 {
1355                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1356                         "krb5_auth_con_init() fails.\n");
1357                 goto err;
1358                 }


1361         if ((krb5rc = krb5_auth_con_getrcache(krb5context, krb5auth_context,
1362                 &rcache)))
1363                 {
1364                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1365                         "krb5_auth_con_getrcache() fails.\n");
1366                 goto err;
1367                 }

1369         if ((krb5rc = krb5_sname_to_principal(krb5context, NULL,
1370                 (kssl_ctx->service_name)? kssl_ctx->service_name: KRB5SVC,
1371                 KRB5_NT_SRV_HST, &krb5server)) != 0)
1372                 {
1373                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1374                         "krb5_sname_to_principal() fails.\n");
1375                 goto err;
1376                 }

1378         if (rcache == NULL)
1379                 {
1380                 if ((krb5rc = krb5_get_server_rcache(krb5context,
1381                         krb5_princ_component(krb5context, krb5server, 0),
```



```
1382                         &rcache)))
1383                         {
1384                         kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1385                                 "krb5_get_server_rcache() fails.\n");
1386                         goto err;
1387                         }
1388                 }

1390         if ((krb5rc = krb5_auth_con_setrcache(krb5context, krb5auth_context, rca
1391                 {
1392                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1393                         "krb5_auth_con_setrcache() fails.\n");
1394                 goto err;
1395                 }


1398         /*      kssl_ctx->keytab_file == NULL ==> use Kerberos default
1399         */
1400         if (kssl_ctx->keytab_file)
1401                 {
1402                 krb5rc = krb5_kt_resolve(krb5context, kssl_ctx->keytab_file,
1403                         &krb5keytab);
1404                 if (krb5rc)
1405                         {
1406                         kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1407                                 "krb5_kt_resolve() fails.\n");
1408                         goto err;
1409                         }
1410                 }
1411         else
1412                 {
1413                 krb5rc = krb5_kt_default(krb5context,&krb5keytab);
1414                 if (krb5rc)
1415                         {
1416                         kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
1417                                 "krb5_kt_default() fails.\n");
1418                         goto err;
1419                         }
1420                 }

1422         /*      Actual Kerberos5 krb5_recvauth() has initial conversation here
1423         **      o       check KRB5_SENDAUTH_BADAUTHVERS
1424         **              unless KRB5_RECVAUTH_SKIP_VERSION
1425         **      o       check KRB5_SENDAUTH_BADAPPLVERS
1426         **      o       send "0" msg if all OK
1427         */

1429         /*  20010411 was using AP_REQ instead of true KerberosWrapper
1430         **
1431         **  if ((krb5rc = krb5_rd_req(krb5context, &krb5auth_context,
1432         **                      &krb5in_data, krb5server, krb5keytab,
1433         **                      &ap_option, &krb5ticket)) != 0)  { Error }
1434         */

1436         p = (unsigned char *)indata->data;
1437         if ((asn1ticket = (KRB5_TKTBODY *) d2i_KRB5_TICKET(NULL, &p,
1438                                                 (long) indata->length)) == NULL)
1439                 {
1440                 BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1441                         "d2i_KRB5_TICKET() ASN.1 decode failure.\n");
1442                 kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1443                 goto err;
1444                 }

1446         /* Was:  krb5rc = krb5_decode_ticket(krb5in_data,&krb5ticket)) != 0) */
1447         if ((krb5rc = kssl_TKT2tkt(krb5context, asn1ticket, &krb5ticket,
```

```
1448                                        kssl_err)) != 0)
1449                {
1450                BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1451                        "Error converting ASN.1 ticket to krb5_ticket.\n");
1452                kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1453                goto err;
1454                }

1456        if (! krb5_principal_compare(krb5context, krb5server,
1457                                                krb5ticket->server))  {
1458                krb5rc = KRB5_PRINC_NOMATCH;
1459                BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1460                        "server principal != ticket principal\n");
1461                kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1462                goto err;
1463                }
1464        if ((krb5rc = krb5_kt_get_entry(krb5context, krb5keytab,
1465                        krb5ticket->server, krb5ticket->enc_part.kvno,
1466                        krb5ticket->enc_part.enctype, &kt_entry)) != 0)  {
1467                BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1468                        "krb5_kt_get_entry() fails with %x.\n", krb5rc);
1469                kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1470                goto err;
1471                }
1472        if ((krb5rc = krb5_decrypt_tkt_part(krb5context, &kt_entry.key,
1473                        krb5ticket)) != 0)  {
1474                BIO_snprintf(kssl_err->text, KSSL_ERR_MAX,
1475                        "krb5_decrypt_tkt_part() failed.\n");
1476                kssl_err->reason = SSL_R_KRB5_S_RD_REQ;
1477                goto err;
1478                }
1479        else  {
1480                krb5_kt_free_entry(krb5context, &kt_entry);
1481 #ifdef KSSL_DEBUG
1482                {
1483                int i; krb5_address **paddr = krb5ticket->enc_part2->caddrs;
1484                printf("Decrypted ticket fields:\n");
1485                printf("\tflags: %X, transit-type: %X",
1486                        krb5ticket->enc_part2->flags,
1487                        krb5ticket->enc_part2->transited.tr_type);
1488                print_krb5_data("\ttransit-data: ",
1489                        &(krb5ticket->enc_part2->transited.tr_contents));
1490                printf("\tcaddrs: %p, authdata: %p\n",
1491                        krb5ticket->enc_part2->caddrs,
1492                        krb5ticket->enc_part2->authorization_data);
1493                if (paddr)
1494                        {
1495                        printf("\tcaddrs:\n");
1496                        for (i=0; paddr[i] != NULL; i++)
1497                                {
1498                                krb5_data d;
1499                                d.length=paddr[i]->length;
1500                                d.data=paddr[i]->contents;
1501                                print_krb5_data("\t\tIP: ", &d);
1502                                }
1503                        }
1504                printf("\tstart/auth/end times: %d / %d / %d\n",
1505                        krb5ticket->enc_part2->times.starttime,
1506                        krb5ticket->enc_part2->times.authtime,
1507                        krb5ticket->enc_part2->times.endtime);
1508                }
1509 #endif  /* KSSL_DEBUG */
1510                }

1512        krb5rc = KRB5_NO_TKT_SUPPLIED;
1513        if (!krb5ticket  ||     !krb5ticket->enc_part2  ||
```



```
1514                !krb5ticket->enc_part2->client  ||
1515                !krb5ticket->enc_part2->client->data  ||
1516                !krb5ticket->enc_part2->session)
1517                {
1518                kssl_err_set(kssl_err, SSL_R_KRB5_S_BAD_TICKET,
1519                        "bad ticket from krb5_rd_req.\n");
1520                }
1521        else if (kssl_ctx_setprinc(kssl_ctx, KSSL_CLIENT,
1522                 &krb5ticket->enc_part2->client->realm,
1523                 krb5ticket->enc_part2->client->data,
1524                 krb5ticket->enc_part2->client->length))
1525                {
1526                kssl_err_set(kssl_err, SSL_R_KRB5_S_BAD_TICKET,
1527                        "kssl_ctx_setprinc() fails.\n");
1528                }
1529        else if (kssl_ctx_setkey(kssl_ctx, krb5ticket->enc_part2->session))
1530                {
1531                kssl_err_set(kssl_err, SSL_R_KRB5_S_BAD_TICKET,
1532                        "kssl_ctx_setkey() fails.\n");
1533                }
1534        else if (krb5ticket->enc_part2->flags & TKT_FLG_INVALID)
1535                {
1536                krb5rc = KRB5KRB_AP_ERR_TKT_INVALID;
1537                kssl_err_set(kssl_err, SSL_R_KRB5_S_BAD_TICKET,
1538                        "invalid ticket from krb5_rd_req.\n");
1539                }
1540        else    krb5rc = 0;

1542        kssl_ctx->enctype       = krb5ticket->enc_part.enctype;
1543        ttimes->authtime        = krb5ticket->enc_part2->times.authtime;
1544        ttimes->starttime       = krb5ticket->enc_part2->times.starttime;
1545        ttimes->endtime         = krb5ticket->enc_part2->times.endtime;
1546        ttimes->renew_till      = krb5ticket->enc_part2->times.renew_till;

1548  err:
1549 #ifdef KSSL_DEBUG
1550        kssl_ctx_show(kssl_ctx);
1551 #endif  /* KSSL_DEBUG */

1553        if (asn1ticket)         KRB5_TICKET_free((KRB5_TICKET *) asn1ticket);
1554        if (krb5keytab)         krb5_kt_close(krb5context, krb5keytab);
1555        if (krb5ticket)         krb5_free_ticket(krb5context, krb5ticket);
1556        if (krb5server)         krb5_free_principal(krb5context, krb5server);
1557        return (krb5rc);
1558        }


1561 /*      Allocate & return a new kssl_ctx struct.
1562 */
1563 KSSL_CTX        *
1564 kssl_ctx_new(void)
1565        {
1566        return ((KSSL_CTX *) kssl_calloc(1, sizeof(KSSL_CTX)));
1567        }


1570 /*      Frees a kssl_ctx struct and any allocated memory it holds.
1571 **      Returns NULL.
1572 */
1573 KSSL_CTX        *
1574 kssl_ctx_free(KSSL_CTX *kssl_ctx)
1575        {
1576        if (kssl_ctx == NULL)  return kssl_ctx;

1578        if (kssl_ctx->key)              OPENSSL_cleanse(kssl_ctx->key,
1579                                                        kssl_ctx->length);
```

```
1580         if (kssl_ctx->key)              kssl_free(kssl_ctx->key);
1581         if (kssl_ctx->client_princ)     kssl_free(kssl_ctx->client_princ);
1582         if (kssl_ctx->service_host)     kssl_free(kssl_ctx->service_host);
1583         if (kssl_ctx->service_name)     kssl_free(kssl_ctx->service_name);
1584         if (kssl_ctx->keytab_file)      kssl_free(kssl_ctx->keytab_file);

1586         kssl_free(kssl_ctx);
1587         return (KSSL_CTX *) NULL;
1588         }


1591 /*      Given an array of (krb5_data *) entity (and optional realm),
1592 **      set the plain (char *) client_princ or service_host member
1593 **      of the kssl_ctx struct.
1594 */
1595 krb5_error_code
1596 kssl_ctx_setprinc(KSSL_CTX *kssl_ctx, int which,
1597         krb5_data *realm, krb5_data *entity, int nentities)
1598         {
1599         char    **princ;
1600         int     length;
1601         int i;

1603         if (kssl_ctx == NULL  ||  entity == NULL)  return KSSL_CTX_ERR;

1605         switch (which)
1606                 {
1607         case KSSL_CLIENT:       princ = &kssl_ctx->client_princ;        break;
1608         case KSSL_SERVER:       princ = &kssl_ctx->service_host;        break;
1609         default:                return KSSL_CTX_ERR;                    break;
1610                 }
1611         if (*princ)  kssl_free(*princ);

1613         /* Add up all the entity->lengths */
1614         length = 0;
1615         for (i=0; i < nentities; i++)
1616                 {
1617                 length += entity[i].length;
1618                 }
1619         /* Add in space for the '/' character(s) (if any) */
1620         length += nentities-1;
1621         /* Space for the ('@'+realm+NULL | NULL) */
1622         length += ((realm)? realm->length + 2: 1);

1624         if ((*princ = kssl_calloc(1, length)) == NULL)
1625                 return KSSL_CTX_ERR;
1626         else
1627                 {
1628                 for (i = 0; i < nentities; i++)
1629                         {
1630                         strncat(*princ, entity[i].data, entity[i].length);
1631                         if (i < nentities-1)
1632                                 {
1633                                 strcat (*princ, "/");
1634                                 }
1635                         }
1636                 if (realm)
1637                         {
1638                         strcat (*princ, "@");
1639                         (void) strncat(*princ, realm->data, realm->length);
1640                         }
1641                 }

1643         return KSSL_CTX_OK;
1644         }
```



```
1647 /*      Set one of the plain (char *) string members of the kssl_ctx struct.
1648 **      Default values should be:
1649 **              which == KSSL_SERVICE   =>      "khost" (KRB5SVC)
1650 **              which == KSSL_KEYTAB    =>      "/etc/krb5.keytab" (KRB5KEYTAB)
1651 */
1652 krb5_error_code
1653 kssl_ctx_setstring(KSSL_CTX *kssl_ctx, int which, char *text)
1654         {
1655         char    **string;

1657         if (!kssl_ctx)  return KSSL_CTX_ERR;

1659         switch (which)
1660                 {
1661         case KSSL_SERVICE:      string = &kssl_ctx->service_name;       break;
1662         case KSSL_SERVER:       string = &kssl_ctx->service_host;       break;
1663         case KSSL_CLIENT:       string = &kssl_ctx->client_princ;       break;
1664         case KSSL_KEYTAB:       string = &kssl_ctx->keytab_file;        break;
1665         default:                return KSSL_CTX_ERR;                    break;
1666                 }
1667         if (*string)  kssl_free(*string);

1669         if (!text)
1670                 {
1671                 *string = '\0';
1672                 return KSSL_CTX_OK;
1673                 }

1675         if ((*string = kssl_calloc(1, strlen(text) + 1)) == NULL)
1676                 return KSSL_CTX_ERR;
1677         else
1678                 strcpy(*string, text);

1680         return KSSL_CTX_OK;
1681         }


1684 /*      Copy the Kerberos session key from a (krb5_keyblock *) to a kssl_ctx
1685 **      struct.  Clear kssl_ctx->key if Kerberos session key is NULL.
1686 */
1687 krb5_error_code
1688 kssl_ctx_setkey(KSSL_CTX *kssl_ctx, krb5_keyblock *session)
1689         {
1690         int             length;
1691         krb5_enctype    enctype;
1692         krb5_octet FAR  *contents = NULL;

1694         if (!kssl_ctx)  return KSSL_CTX_ERR;

1696         if (kssl_ctx->key)
1697                 {
1698                 OPENSSL_cleanse(kssl_ctx->key, kssl_ctx->length);
1699                 kssl_free(kssl_ctx->key);
1700                 }

1702         if (session)
1703                 {

1705 #ifdef KRB5_HEIMDAL
1706                 length = session->keyvalue->length;
1707                 enctype = session->keytype;
1708                 contents = session->keyvalue->contents;
1709 #else
1710                 length = session->length;
1711                 enctype = session->enctype;
```

```
1712                 contents = session->contents;
1713 #endif
1714                 kssl_ctx->enctype = enctype;
1715                 kssl_ctx->length  = length;
1716                 }
1717         else
1718                 {
1719                 kssl_ctx->enctype = ENCTYPE_UNKNOWN;
1720                 kssl_ctx->length  = 0;
1721                 return KSSL_CTX_OK;
1722                 }

1724         if ((kssl_ctx->key =
1725                 (krb5_octet FAR *) kssl_calloc(1, kssl_ctx->length)) == NULL)
1726                 {
1727                 kssl_ctx->length  = 0;
1728                 return KSSL_CTX_ERR;
1729                 }
1730         else
1731                 memcpy(kssl_ctx->key, contents, length);

1733         return KSSL_CTX_OK;
1734         }


1737 /*      Display contents of kssl_ctx struct
1738 */
1739 void
1740 kssl_ctx_show(KSSL_CTX *kssl_ctx)
1741         {
1742         int     i;

1744         printf("kssl_ctx: ");
1745         if (kssl_ctx == NULL)
1746                 {
1747                 printf("NULL\n");
1748                 return;
1749                 }
1750         else
1751                 printf("%p\n", (void *)kssl_ctx);

1753         printf("\tservice:\t%s\n",
1754                 (kssl_ctx->service_name)? kssl_ctx->service_name: "NULL");
1755         printf("\tclient:\t%s\n",
1756                 (kssl_ctx->client_princ)? kssl_ctx->client_princ: "NULL");
1757         printf("\tserver:\t%s\n",
1758                 (kssl_ctx->service_host)? kssl_ctx->service_host: "NULL");
1759         printf("\tkeytab:\t%s\n",
1760                 (kssl_ctx->keytab_file)? kssl_ctx->keytab_file: "NULL");
1761         printf("\tkey [%d:%d]:\t",
1762                 kssl_ctx->enctype, kssl_ctx->length);

1764         for (i=0; i < kssl_ctx->length  &&  kssl_ctx->key; i++)
1765                 {
1766                 printf("%02x", kssl_ctx->key[i]);
1767                 }
1768         printf("\n");
1769         return;
1770         }

1772     int
1773     kssl_keytab_is_available(KSSL_CTX *kssl_ctx)
1774 {
1775     krb5_context                krb5context = NULL;
1776     krb5_keytab                 krb5keytab = NULL;
1777     krb5_keytab_entry           entry;
```



```
1778     krb5_principal              princ = NULL;
1779     krb5_error_code             krb5rc = KRB5KRB_ERR_GENERIC;
1780     int rc = 0;

1782     if ((krb5rc = krb5_init_context(&krb5context)))
1783         return(0);

1785     /*  kssl_ctx->keytab_file == NULL ==> use Kerberos default
1786     */
1787     if (kssl_ctx->keytab_file)
1788     {
1789         krb5rc = krb5_kt_resolve(krb5context, kssl_ctx->keytab_file,
1790                                  &krb5keytab);
1791         if (krb5rc)
1792             goto exit;
1793     }
1794     else
1795     {
1796         krb5rc = krb5_kt_default(krb5context,&krb5keytab);
1797         if (krb5rc)
1798             goto exit;
1799     }

1801     /* the host key we are looking for */
1802     krb5rc = krb5_sname_to_principal(krb5context, NULL,
1803                                      kssl_ctx->service_name ? kssl_ctx->service_
1804                                      KRB5_NT_SRV_HST, &princ);

1806     if (krb5rc)
1807         goto exit;

1809     krb5rc = krb5_kt_get_entry(krb5context, krb5keytab,
1810                                princ,
1811                                0 /* IGNORE_VNO */,
1812                                0 /* IGNORE_ENCTYPE */,
1813                                &entry);
1814     if ( krb5rc == KRB5_KT_NOTFOUND ) {
1815         rc = 1;
1816         goto exit;
1817     } else if ( krb5rc )
1818         goto exit;

1820     krb5_kt_free_entry(krb5context, &entry);
1821     rc = 1;

1823   exit:
1824     if (krb5keytab)     krb5_kt_close(krb5context, krb5keytab);
1825     if (princ)          krb5_free_principal(krb5context, princ);
1826     if (krb5context)    krb5_free_context(krb5context);
1827     return(rc);
1828 }

1830 int
1831 kssl_tgt_is_available(KSSL_CTX *kssl_ctx)
1832         {
1833         krb5_error_code         krb5rc = KRB5KRB_ERR_GENERIC;
1834         krb5_context            krb5context = NULL;
1835         krb5_ccache             krb5ccdef = NULL;
1836         krb5_creds              krb5creds, *krb5credsp = NULL;
1837         int                     rc = 0;

1839         memset((char *)&krb5creds, 0, sizeof(krb5creds));

1841         if (!kssl_ctx)
1842             return(0);
```

```
1844         if (!kssl_ctx->service_host)
1845             return(0);

1847         if ((krb5rc = krb5_init_context(&krb5context)) != 0)
1848             goto err;

1850         if ((krb5rc = krb5_sname_to_principal(krb5context,
1851                                               kssl_ctx->service_host,
1852                                               (kssl_ctx->service_name)? kssl_ctx
1853                                               KRB5_NT_SRV_HST, &krb5creds.server
1854             goto err;

1856         if ((krb5rc = krb5_cc_default(krb5context, &krb5ccdef)) != 0)
1857             goto err;

1859         if ((krb5rc = krb5_cc_get_principal(krb5context, krb5ccdef,
1860                                             &krb5creds.client)) != 0)
1861             goto err;

1863         if ((krb5rc = krb5_get_credentials(krb5context, 0, krb5ccdef,
1864                                            &krb5creds, &krb5credsp)) != 0)
1865             goto err;

1867         rc = 1;

1869       err:
1870 #ifdef KSSL_DEBUG
1871         kssl_ctx_show(kssl_ctx);
1872 #endif  /* KSSL_DEBUG */

1874         if (krb5creds.client)   krb5_free_principal(krb5context, krb5creds.clien
1875         if (krb5creds.server)   krb5_free_principal(krb5context, krb5creds.serve
1876         if (krb5context)        krb5_free_context(krb5context);
1877         return(rc);
1878         }

1880 #if !defined(OPENSSL_SYS_WINDOWS) && !defined(OPENSSL_SYS_WIN32)
1881 void kssl_krb5_free_data_contents(krb5_context context, krb5_data *data)
1882         {
1883 #ifdef KRB5_HEIMDAL
1884         data->length = 0;
1885         if (data->data)
1886             free(data->data);
1887 #elif defined(KRB5_MIT_OLD11)
1888         if (data->data)  {
1889                 krb5_xfree(data->data);
1890                 data->data = 0;
1891                 }
1892 #else
1893         krb5_free_data_contents(NULL, data);
1894 #endif
1895         }
1896 #endif /* !OPENSSL_SYS_WINDOWS && !OPENSSL_SYS_WIN32 */


1899 /*  Given pointers to KerberosTime and struct tm structs, convert the
1900 **  KerberosTime string to struct tm.  Note that KerberosTime is a
1901 **  ASN1_GENERALIZEDTIME value, constrained to GMT with no fractional
1902 **  seconds as defined in RFC 1510.
1903 **  Return pointer to the (partially) filled in struct tm on success,
1904 **  return NULL on failure.
1905 */
1906 static struct tm *k_gmtime(ASN1_GENERALIZEDTIME *gtime, struct tm *k_tm)
1907         {
1908         char            c, *p;
```



```
1910         if (!k_tm)  return NULL;
1911         if (gtime == NULL  ||  gtime->length < 14)  return NULL;
1912         if (gtime->data == NULL)  return NULL;

1914         p = (char *)&gtime->data[14];

1916         c = *p;  *p = '\0';  p -= 2;  k_tm->tm_sec  = atoi(p);      *(p+2) = c;
1917         c = *p;  *p = '\0';  p -= 2;  k_tm->tm_min  = atoi(p);      *(p+2) = c;
1918         c = *p;  *p = '\0';  p -= 2;  k_tm->tm_hour = atoi(p);      *(p+2) = c;
1919         c = *p;  *p = '\0';  p -= 2;  k_tm->tm_mday = atoi(p);      *(p+2) = c;
1920         c = *p;  *p = '\0';  p -= 2;  k_tm->tm_mon  = atoi(p)-1;    *(p+2) = c;
1921         c = *p;  *p = '\0';  p -= 4;  k_tm->tm_year = atoi(p)-1900; *(p+4) = c;

1923         return k_tm;
1924         }


1927 /*  Helper function for kssl_validate_times().
1928 **  We need context->clockskew, but krb5_context is an opaque struct.
1929 **  So we try to sneek the clockskew out through the replay cache.
1930 **      If that fails just return a likely default (300 seconds).
1931 */
1932 static krb5_deltat get_rc_clockskew(krb5_context context)
1933         {
1934         krb5_rcache     rc;
1935         krb5_deltat     clockskew;

1937         if (krb5_rc_default(context, &rc))  return KSSL_CLOCKSKEW;
1938         if (krb5_rc_initialize(context, rc, 0))  return KSSL_CLOCKSKEW;
1939         if (krb5_rc_get_lifespan(context, rc, &clockskew))  {
1940                 clockskew = KSSL_CLOCKSKEW;
1941                 }
1942         (void) krb5_rc_destroy(context, rc);
1943         return clockskew;
1944         }


1947 /*  kssl_validate_times() combines (and more importantly exposes)
1948 **  the MIT KRB5 internal function krb5_validate_times() and the
1949 **  in_clock_skew() macro.  The authenticator client time is checked
1950 **  to be within clockskew secs of the current time and the current
1951 **  time is checked to be within the ticket start and expire times.
1952 **  Either check may be omitted by supplying a NULL value.
1953 **  Returns 0 for valid times, SSL_R_KRB5* error codes otherwise.
1954 **  See Also: (Kerberos source)/krb5/lib/krb5/krb/valid_times.c
1955 **  20010420 VRS
1956 */
1957 krb5_error_code  kssl_validate_times(   krb5_timestamp atime,
1958                                         krb5_ticket_times *ttimes)
1959         {
1960         krb5_deltat     skew;
1961         krb5_timestamp  start, now;
1962         krb5_error_code rc;
1963         krb5_context    context;

1965         if ((rc = krb5_init_context(&context)))  return SSL_R_KRB5_S_BAD_TICKET;
1966         skew = get_rc_clockskew(context);
1967         if ((rc = krb5_timeofday(context,&now))) return SSL_R_KRB5_S_BAD_TICKET;
1968         krb5_free_context(context);

1970         if (atime  &&  labs(atime - now) >= skew)  return SSL_R_KRB5_S_TKT_SKEW;

1972         if (! ttimes)  return 0;

1974         start = (ttimes->starttime != 0)? ttimes->starttime: ttimes->authtime;
1975         if (start - now > skew)  return SSL_R_KRB5_S_TKT_NYV;
```

```
1976         if ((now - ttimes->endtime) > skew)  return SSL_R_KRB5_S_TKT_EXPIRED;

1978 #ifdef KSSL_DEBUG
1979         printf("kssl_validate_times: %d |<-  | %d - %d | < %d  ->| %d\n",
1980                 start, atime, now, skew, ttimes->endtime);
1981 #endif  /* KSSL_DEBUG */

1983         return 0;
1984         }


1987 /*  Decode and decrypt given DER-encoded authenticator, then pass
1988 **  authenticator ctime back in *atimep (or 0 if time unavailable).
1989 **  Returns krb5_error_code and kssl_err on error.  A NULL
1990 **  authenticator (authentp->length == 0) is not considered an error.
1991 **  Note that kssl_check_authent() makes use of the KRB5 session key;
1992 **  you must call kssl_sget_tkt() to get the key before calling this routine.
1993 */
1994 krb5_error_code  kssl_check_authent(
1995                         /* IN     */    KSSL_CTX        *kssl_ctx,
1996                         /* IN     */    krb5_data       *authentp,
1997                         /* OUT    */    krb5_timestamp  *atimep,
1998                         /* OUT    */    KSSL_ERR        *kssl_err  )
1999         {
2000         krb5_error_code         krb5rc = 0;
2001         KRB5_ENCDATA            *dec_authent = NULL;
2002         KRB5_AUTHENTBODY        *auth = NULL;
2003         krb5_enctype            enctype;
2004         EVP_CIPHER_CTX          ciph_ctx;
2005         const EVP_CIPHER        *enc = NULL;
2006         unsigned char           iv[EVP_MAX_IV_LENGTH];
2007         const unsigned char     *p;
2008         unsigned char           *unenc_authent;
2009         int                     outl, unencbufsize;
2010         struct tm               tm_time, *tm_l, *tm_g;
2011         time_t                  now, tl, tg, tr, tz_offset;

2013         EVP_CIPHER_CTX_init(&ciph_ctx);
2014         *atimep = 0;
2015         kssl_err_set(kssl_err, 0, "");

2017 #ifndef KRB5CHECKAUTH
2018         authentp = NULL;
2019 #else
2020 #if     KRB5CHECKAUTH == 0
2021         authentp = NULL;
2022 #endif
2023 #endif  /* KRB5CHECKAUTH */

2025         if (authentp == NULL  ||  authentp->length == 0)  return 0;

2027 #ifdef KSSL_DEBUG
2028         {
2029         unsigned int ui;
2030         printf("kssl_check_authent: authenticator[%d]:\n",authentp->length);
2031         p = authentp->data;
2032         for (ui=0; ui < authentp->length; ui++)  printf("%02x ",p[ui]);
2033         printf("\n");
2034         }
2035 #endif  /* KSSL_DEBUG */

2037         unencbufsize = 2 * authentp->length;
2038         if ((unenc_authent = calloc(1, unencbufsize)) == NULL)
2039                 {
2040                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
2041                         "Unable to allocate authenticator buffer.\n");
```



```
2042                 krb5rc = KRB5KRB_ERR_GENERIC;
2043                 goto err;
2044                 }

2046         p = (unsigned char *)authentp->data;
2047         if ((dec_authent = d2i_KRB5_ENCDATA(NULL, &p,
2048                                         (long) authentp->length)) == NULL)
2049                 {
2050                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
2051                         "Error decoding authenticator.\n");
2052                 krb5rc = KRB5KRB_AP_ERR_BAD_INTEGRITY;
2053                 goto err;
2054                 }

2056         enctype = dec_authent->etype->data[0];  /* should = kssl_ctx->enctype */
2057 #if !defined(KRB5_MIT_OLD11)
2058             switch ( enctype ) {
2059             case ENCTYPE_DES3_CBC_SHA1:         /*    EVP_des_ede3_cbc();  */
2060             case ENCTYPE_DES3_CBC_SHA:
2061             case ENCTYPE_DES3_CBC_RAW:
2062                 krb5rc = 0;                     /* Skip, can't handle derived ke
2063                 goto err;
2064             }
2065 #endif
2066         enc = kssl_map_enc(enctype);
2067         memset(iv, 0, sizeof iv);       /* per RFC 1510 */

2069         if (enc == NULL)
2070                 {
2071                 /*  Disable kssl_check_authent for ENCTYPE_DES3_CBC_SHA1.
2072                 **  This enctype indicates the authenticator was encrypted
2073                 **  using key-usage derived keys which openssl cannot decrypt.
2074                 */
2075                 goto err;
2076                 }

2078         if (!EVP_CipherInit(&ciph_ctx,enc,kssl_ctx->key,iv,0))
2079                 {
2080                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
2081                         "EVP_CipherInit error decrypting authenticator.\n");
2082                 krb5rc = KRB5KRB_AP_ERR_BAD_INTEGRITY;
2083                 goto err;
2084                 }
2085         outl = dec_authent->cipher->length;
2086         if (!EVP_Cipher(&ciph_ctx,unenc_authent,dec_authent->cipher->data,outl))
2087                 {
2088                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
2089                         "EVP_Cipher error decrypting authenticator.\n");
2090                 krb5rc = KRB5KRB_AP_ERR_BAD_INTEGRITY;
2091                 goto err;
2092                 }
2093         EVP_CIPHER_CTX_cleanup(&ciph_ctx);

2095 #ifdef KSSL_DEBUG
2096         {
2097         int padl;
2098         printf("kssl_check_authent: decrypted authenticator[%d] =\n", outl);
2099         for (padl=0; padl < outl; padl++) printf("%02x ",unenc_authent[padl]);
2100         printf("\n");
2101         }
2102 #endif  /* KSSL_DEBUG */

2104         if ((p = kssl_skip_confound(enctype, unenc_authent)) == NULL)
2105                 {
2106                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
2107                         "confounded by authenticator.\n");
```

```
2108                 krb5rc = KRB5KRB_AP_ERR_BAD_INTEGRITY;
2109                 goto err;
2110                 }
2111         outl -= p - unenc_authent;

2113         if ((auth = (KRB5_AUTHENTBODY *) d2i_KRB5_AUTHENT(NULL, &p,
2114                                                           (long) outl))==NULL)
2115                 {
2116                 kssl_err_set(kssl_err, SSL_R_KRB5_S_INIT,
2117                         "Error decoding authenticator body.\n");
2118                 krb5rc = KRB5KRB_AP_ERR_BAD_INTEGRITY;
2119                 goto err;
2120                 }

2122         memset(&tm_time,0,sizeof(struct tm));
2123         if (k_gmtime(auth->ctime, &tm_time)  &&
2124                 ((tr = mktime(&tm_time)) != (time_t)(-1)))
2125                 {
2126                 now  = time(&now);
2127                 tm_l = localtime(&now);         tl = mktime(tm_l);
2128                 tm_g = gmtime(&now);            tg = mktime(tm_g);
2129                 tz_offset = tg - tl;

2131                 *atimep = (krb5_timestamp)(tr - tz_offset);
2132                 }

2134 #ifdef KSSL_DEBUG
2135         printf("kssl_check_authent: returns %d for client time ", *atimep);
2136         if (auth && auth->ctime && auth->ctime->length && auth->ctime->data)
2137                 printf("%.*s\n", auth->ctime->length, auth->ctime->data);
2138         else    printf("NULL\n");
2139 #endif  /* KSSL_DEBUG */

2141  err:
2142         if (auth)               KRB5_AUTHENT_free((KRB5_AUTHENT *) auth);
2143         if (dec_authent)        KRB5_ENCDATA_free(dec_authent);
2144         if (unenc_authent)      free(unenc_authent);
2145         EVP_CIPHER_CTX_cleanup(&ciph_ctx);
2146         return krb5rc;
2147         }


2150 /*  Replaces krb5_build_principal_ext(), with varargs length == 2 (svc, host),
2151 **  because I dont't know how to stub varargs.
2152 **  Returns krb5_error_code == ENOMEM on alloc error, otherwise
2153 **  passes back newly constructed principal, which should be freed by caller.
2154 */
2155 krb5_error_code  kssl_build_principal_2(
2156                         /* UPDATE */    krb5_context    context,
2157                         /* OUT    */    krb5_principal  *princ,
2158                         /* IN     */    int rlen,  const char *realm,
2159                         /* IN     */    int slen,  const char *svc,
2160                         /* IN     */    int hlen,  const char *host)
2161         {
2162         krb5_data               *p_data = NULL;
2163         krb5_principal          new_p = NULL;
2164         char                    *new_r = NULL;

2166         if ((p_data = (krb5_data *) calloc(2, sizeof(krb5_data))) == NULL  ||
2167             (new_p = (krb5_principal) calloc(1, sizeof(krb5_principal_data)))
2168                         == NULL)  goto err;
2169         new_p->length = 2;
2170         new_p->data = p_data;

2172         if ((new_r = calloc(1, rlen + 1)) == NULL)  goto err;
2173         memcpy(new_r, realm, rlen);
```



```
2174         krb5_princ_set_realm_length(context, new_p, rlen);
2175         krb5_princ_set_realm_data(context, new_p, new_r);

2177         if ((new_p->data[0].data = calloc(1, slen + 1)) == NULL)  goto err;
2178         memcpy(new_p->data[0].data, svc, slen);
2179         new_p->data[0].length = slen;

2181         if ((new_p->data[1].data = calloc(1, hlen + 1)) == NULL)  goto err;
2182         memcpy(new_p->data[1].data, host, hlen);
2183         new_p->data[1].length = hlen;

2185         krb5_princ_type(context, new_p) = KRB5_NT_UNKNOWN;
2186         *princ = new_p;
2187         return 0;

2189  err:
2190         if (new_p  &&  new_p[0].data)   free(new_p[0].data);
2191         if (new_p  &&  new_p[1].data)   free(new_p[1].data);
2192         if (new_p)      free(new_p);
2193         if (new_r)      free(new_r);
2194         return ENOMEM;
2195         }

2197 void SSL_set0_kssl_ctx(SSL *s, KSSL_CTX *kctx)
2198         {
2199         s->kssl_ctx = kctx;
2200         }

2202 KSSL_CTX * SSL_get0_kssl_ctx(SSL *s)
2203         {
2204         return s->kssl_ctx;
2205         }

2207 char *kssl_ctx_get0_client_princ(KSSL_CTX *kctx)
2208         {
2209         if (kctx)
2210                 return kctx->client_princ;
2211         return NULL;
2212         }

2214 #else /* !OPENSSL_NO_KRB5 */

2216 #if defined(PEDANTIC) || defined(OPENSSL_SYS_VMS)
2217 static void *dummy=&dummy;
2218 #endif

2220 #endif  /* !OPENSSL_NO_KRB5     */
2221 #endif /* ! codereview */
```

```
**********************************************************
   15597 Wed Aug 13 19:53:36 2014
new/usr/src/lib/openssl/libsunw_ssl/mapfile-vers
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
$mapfile_version 2

SYMBOL_VERSION SUNWprivate_1.1 {
    global:
        sunw_BIO_f_ssl;
        sunw_BIO_new_buffer_ssl_connect;
        sunw_BIO_new_ssl;
        sunw_BIO_new_ssl_connect;
        sunw_BIO_ssl_copy_session_id;
        sunw_BIO_ssl_shutdown;
        sunw_d2i_SSL_SESSION;
        sunw_do_dtls1_write;
        sunw_dtls1_accept;
        sunw_dtls1_buffer_message;
        sunw_dtls1_check_timeout_num;
        sunw_dtls1_clear;
        sunw_dtls1_clear_record_buffer;
        sunw_dtls1_client_hello;
        sunw_dtls1_connect;
        sunw_dtls1_ctrl;
        sunw_dtls1_default_timeout;
        sunw_dtls1_dispatch_alert;
        sunw_dtls1_do_write;
        sunw_dtls1_double_timeout;
        sunw_dtls1_enc;
        sunw_dtls1_free;
        sunw_dtls1_get_ccs_header;
        sunw_dtls1_get_cipher;
        sunw_dtls1_get_message;
        sunw_dtls1_get_message_header;
        sunw_dtls1_get_queue_priority;
        sunw_dtls1_get_record;
        sunw_dtls1_get_timeout;
        sunw_dtls1_handle_timeout;
        sunw_dtls1_heartbeat;
        sunw_dtls1_is_timer_expired;
        sunw_dtls1_listen;
        sunw_dtls1_min_mtu;
        sunw_dtls1_new;
        sunw_dtls1_output_cert_chain;
        sunw_dtls1_process_heartbeat;
        sunw_dtls1_read_bytes;
        sunw_dtls1_read_failed;
        sunw_dtls1_reset_seq_numbers;
        sunw_dtls1_retransmit_buffered_messages;
        sunw_dtls1_retransmit_message;
        sunw_dtls1_send_certificate_request;
        sunw_dtls1_send_change_cipher_spec;
        sunw_dtls1_send_client_certificate;
        sunw_dtls1_send_client_key_exchange;
        sunw_dtls1_send_client_verify;
        sunw_dtls1_send_finished;
        sunw_dtls1_send_hello_request;
        sunw_dtls1_send_newsession_ticket;
        sunw_dtls1_send_server_certificate;
        sunw_dtls1_send_server_done;
        sunw_dtls1_send_server_hello;
        sunw_dtls1_send_server_key_exchange;
        sunw_dtls1_set_message_header;
        sunw_dtls1_shutdown;
        sunw_dtls1_start_timer;
```



```
        sunw_dtls1_stop_timer;
        sunw_dtls1_version_str;
        sunw_dtls1_write_app_data_bytes;
        sunw_dtls1_write_bytes;
        sunw_DTLSv1_client_method;
        sunw_DTLSv1_enc_data;
        sunw_DTLSv1_method;
        sunw_DTLSv1_server_method;
        sunw_ERR_load_SSL_strings;
        sunw_i2d_SSL_SESSION;
        sunw_n_ssl3_mac;
        sunw_OBJ_bsearch_ssl_cipher_id;
        sunw_PEM_read_bio_SSL_SESSION;
        sunw_PEM_read_SSL_SESSION;
        sunw_PEM_write_bio_SSL_SESSION;
        sunw_PEM_write_SSL_SESSION;
        sunw_SRP_Calc_A_param;
        sunw_SRP_generate_client_master_secret;
        sunw_SRP_generate_server_master_secret;
        sunw_SSL_accept;
        sunw_SSL_add_client_CA;
        sunw_ssl_add_clienthello_renegotiate_ext;
        sunw_ssl_add_clienthello_tlsext;
        sunw_ssl_add_clienthello_use_srtp_ext;
        sunw_SSL_add_dir_cert_subjects_to_stack;
        sunw_SSL_add_file_cert_subjects_to_stack;
        sunw_ssl_add_serverhello_renegotiate_ext;
        sunw_ssl_add_serverhello_tlsext;
        sunw_ssl_add_serverhello_use_srtp_ext;
        sunw_SSL_alert_desc_string;
        sunw_SSL_alert_desc_string_long;
        sunw_SSL_alert_type_string;
        sunw_SSL_alert_type_string_long;
        sunw_ssl_bad_method;
        sunw_ssl_bytes_to_cipher_list;
        sunw_SSL_cache_hit;
        sunw_SSL_callback_ctrl;
        sunw_ssl_cert_dup;
        sunw_ssl_cert_free;
        sunw_ssl_cert_inst;
        sunw_ssl_cert_new;
        sunw_ssl_cert_type;
        sunw_ssl_check_clienthello_tlsext_early;
        sunw_ssl_check_clienthello_tlsext_late;
        sunw_SSL_check_private_key;
        sunw_ssl_check_serverhello_tlsext;
        sunw_SSL_CIPHER_description;
        sunw_SSL_CIPHER_get_bits;
        sunw_ssl_cipher_get_evp;
        sunw_SSL_CIPHER_get_id;
        sunw_SSL_CIPHER_get_name;
        sunw_SSL_CIPHER_get_version;
        sunw_ssl_cipher_id_cmp;
        sunw_ssl_cipher_list_to_bytes;
        sunw_ssl_cipher_ptr_id_cmp;
        sunw_SSL_clear;
        sunw_ssl_clear_bad_session;
        sunw_ssl_clear_cipher_ctx;
        sunw_ssl_clear_hash_ctx;
        sunw_SSL_COMP_add_compression_method;
        sunw_SSL_COMP_get_compression_methods;
        sunw_SSL_COMP_get_name;
        sunw_SSL_connect;
        sunw_SSL_copy_session_id;
        sunw_ssl_create_cipher_list;
        sunw_SSL_ctrl;
```

```
128         sunw_SSL_CTX_add_client_CA;
129         sunw_SSL_CTX_add_session;
130         sunw_SSL_CTX_callback_ctrl;
131         sunw_SSL_CTX_check_private_key;
132         sunw_SSL_CTX_ctrl;
133         sunw_SSL_CTX_flush_sessions;
134         sunw_SSL_CTX_free;
135         sunw_SSL_CTX_get_cert_store;
136         sunw_SSL_CTX_get_client_CA_list;
137         sunw_SSL_CTX_get_client_cert_cb;
138         sunw_SSL_CTX_get_ex_data;
139         sunw_SSL_CTX_get_ex_new_index;
140         sunw_SSL_CTX_get_info_callback;
141         sunw_SSL_CTX_get_quiet_shutdown;
142         sunw_SSL_CTX_get_timeout;
143         sunw_SSL_CTX_get_verify_callback;
144         sunw_SSL_CTX_get_verify_depth;
145         sunw_SSL_CTX_get_verify_mode;
146         sunw_SSL_CTX_load_verify_locations;
147         sunw_SSL_CTX_new;
148         sunw_SSL_CTX_remove_session;
149         sunw_SSL_CTX_sess_get_get_cb;
150         sunw_SSL_CTX_sess_get_new_cb;
151         sunw_SSL_CTX_sess_get_remove_cb;
152         sunw_SSL_CTX_sess_set_get_cb;
153         sunw_SSL_CTX_sess_set_new_cb;
154         sunw_SSL_CTX_sess_set_remove_cb;
155         sunw_SSL_CTX_sessions;
156         sunw_SSL_CTX_set_cert_store;
157         sunw_SSL_CTX_set_cert_verify_callback;
158         sunw_SSL_CTX_set_cipher_list;
159         sunw_SSL_CTX_set_client_CA_list;
160         sunw_SSL_CTX_set_client_cert_cb;
161         sunw_SSL_CTX_set_client_cert_engine;
162         sunw_SSL_CTX_set_cookie_generate_cb;
163         sunw_SSL_CTX_set_cookie_verify_cb;
164         sunw_SSL_CTX_set_default_passwd_cb;
165         sunw_SSL_CTX_set_default_passwd_cb_userdata;
166         sunw_SSL_CTX_set_default_verify_paths;
167         sunw_SSL_CTX_set_ex_data;
168         sunw_SSL_CTX_set_generate_session_id;
169         sunw_SSL_CTX_set_info_callback;
170         sunw_SSL_CTX_set_msg_callback;
171         sunw_SSL_CTX_set_next_proto_select_cb;
172         sunw_SSL_CTX_set_next_protos_advertised_cb;
173         sunw_SSL_CTX_set_psk_client_callback;
174         sunw_SSL_CTX_set_psk_server_callback;
175         sunw_SSL_CTX_set_purpose;
176         sunw_SSL_CTX_set_quiet_shutdown;
177         sunw_SSL_CTX_set_session_id_context;
178         sunw_SSL_CTX_set_srp_cb_arg;
179         sunw_SSL_CTX_set_srp_client_pwd_callback;
180         sunw_SSL_CTX_set_srp_password;
181         sunw_SSL_CTX_set_srp_strength;
182         sunw_SSL_CTX_set_srp_username;
183         sunw_SSL_CTX_set_srp_username_callback;
184         sunw_SSL_CTX_set_srp_verify_param_callback;
185         sunw_SSL_CTX_set_ssl_version;
186         sunw_SSL_CTX_set_timeout;
187         sunw_SSL_CTX_set_tlsext_use_srtp;
188         sunw_SSL_CTX_set_tmp_dh_callback;
189         sunw_SSL_CTX_set_tmp_rsa_callback;
190         sunw_SSL_CTX_set_trust;
191         sunw_SSL_CTX_set_verify;
192         sunw_SSL_CTX_set_verify_depth;
193         sunw_SSL_CTX_set1_param;
```



```
194         sunw_SSL_CTX_SRP_CTX_free;
195         sunw_SSL_CTX_SRP_CTX_init;
196         sunw_SSL_CTX_use_certificate;
197         sunw_SSL_CTX_use_certificate_ASN1;
198         sunw_SSL_CTX_use_certificate_chain_file;
199         sunw_SSL_CTX_use_certificate_file;
200         sunw_SSL_CTX_use_PrivateKey;
201         sunw_SSL_CTX_use_PrivateKey_ASN1;
202         sunw_SSL_CTX_use_PrivateKey_file;
203         sunw_SSL_CTX_use_psk_identity_hint;
204         sunw_SSL_CTX_use_RSAPrivateKey;
205         sunw_SSL_CTX_use_RSAPrivateKey_ASN1;
206         sunw_SSL_CTX_use_RSAPrivateKey_file;
207         sunw_ssl_do_client_cert_cb;
208         sunw_SSL_do_handshake;
209         sunw_SSL_dup;
210         sunw_SSL_dup_CA_list;
211         sunw_SSL_export_keying_material;
212         sunw_ssl_fill_hello_random;
213         sunw_SSL_free;
214         sunw_ssl_free_wbio_buffer;
215         sunw_ssl_get_algorithm2;
216         sunw_SSL_get_certificate;
217         sunw_SSL_get_cipher_list;
218         sunw_SSL_get_ciphers;
219         sunw_ssl_get_ciphers_by_id;
220         sunw_SSL_get_client_CA_list;
221         sunw_SSL_get_current_cipher;
222         sunw_SSL_get_current_compression;
223         sunw_SSL_get_current_expansion;
224         sunw_SSL_get_default_timeout;
225         sunw_SSL_get_error;
226         sunw_SSL_get_ex_data;
227         sunw_SSL_get_ex_data_X509_STORE_CTX_idx;
228         sunw_SSL_get_ex_new_index;
229         sunw_SSL_get_fd;
230         sunw_SSL_get_finished;
231         sunw_ssl_get_handshake_digest;
232         sunw_SSL_get_info_callback;
233         sunw_ssl_get_new_session;
234         sunw_SSL_get_peer_cert_chain;
235         sunw_SSL_get_peer_certificate;
236         sunw_SSL_get_peer_finished;
237         sunw_ssl_get_prev_session;
238         sunw_SSL_get_privatekey;
239         sunw_SSL_get_psk_identity;
240         sunw_SSL_get_psk_identity_hint;
241         sunw_SSL_get_quiet_shutdown;
242         sunw_SSL_get_rbio;
243         sunw_SSL_get_read_ahead;
244         sunw_SSL_get_rfd;
245         sunw_SSL_get_selected_srtp_profile;
246         sunw_ssl_get_server_send_cert;
247         sunw_ssl_get_server_send_pkey;
248         sunw_SSL_get_servername;
249         sunw_SSL_get_servername_type;
250         sunw_SSL_get_session;
251         sunw_SSL_get_shared_ciphers;
252         sunw_SSL_get_shutdown;
253         sunw_ssl_get_sign_pkey;
254         sunw_SSL_get_srp_g;
255         sunw_SSL_get_srp_N;
256         sunw_SSL_get_srp_userinfo;
257         sunw_SSL_get_srp_username;
258         sunw_SSL_get_srtp_profiles;
259         sunw_SSL_get_SSL_CTX;
```

```
260         sunw_SSL_get_ssl_method;
261         sunw_SSL_get_verify_callback;
262         sunw_SSL_get_verify_depth;
263         sunw_SSL_get_verify_mode;
264         sunw_SSL_get_verify_result;
265         sunw_SSL_get_version;
266         sunw_SSL_get_wbio;
267         sunw_SSL_get_wfd;
268         sunw_SSL_get0_next_proto_negotiated;
269         sunw_SSL_get1_session;
270         sunw_SSL_has_matching_session_id;
271         sunw_ssl_init_wbio_buffer;
272         sunw_SSL_library_init;
273         sunw_ssl_load_ciphers;
274         sunw_SSL_load_client_CA_file;
275         sunw_SSL_load_error_strings;
276         sunw_SSL_new;
277         sunw_ssl_ok;
278         sunw_ssl_parse_clienthello_renegotiate_ext;
279         sunw_ssl_parse_clienthello_tlsext;
280         sunw_ssl_parse_clienthello_use_srtp_ext;
281         sunw_ssl_parse_serverhello_renegotiate_ext;
282         sunw_ssl_parse_serverhello_tlsext;
283         sunw_ssl_parse_serverhello_use_srtp_ext;
284         sunw_SSL_peek;
285         sunw_SSL_pending;
286         sunw_ssl_prepare_clienthello_tlsext;
287         sunw_ssl_prepare_serverhello_tlsext;
288         sunw_SSL_read;
289         sunw_SSL_renegotiate;
290         sunw_SSL_renegotiate_abbreviated;
291         sunw_SSL_renegotiate_pending;
292         sunw_ssl_replace_hash;
293         sunw_SSL_rstate_string;
294         sunw_SSL_rstate_string_long;
295         sunw_SSL_select_next_proto;
296         sunw_ssl_sess_cert_free;
297         sunw_ssl_sess_cert_new;
298         sunw_SSL_SESSION_free;
299         sunw_SSL_SESSION_get_compress_id;
300         sunw_SSL_SESSION_get_ex_data;
301         sunw_SSL_SESSION_get_ex_new_index;
302         sunw_SSL_SESSION_get_id;
303         sunw_SSL_SESSION_get_time;
304         sunw_SSL_SESSION_get_timeout;
305         sunw_SSL_SESSION_get0_peer;
306         sunw_SSL_SESSION_new;
307         sunw_SSL_SESSION_print;
308         sunw_SSL_SESSION_print_fp;
309         sunw_SSL_SESSION_set_ex_data;
310         sunw_SSL_SESSION_set_time;
311         sunw_SSL_SESSION_set_timeout;
312         sunw_SSL_SESSION_set1_id_context;
313         sunw_SSL_set_accept_state;
314         sunw_SSL_set_bio;
315         sunw_ssl_set_cert_masks;
316         sunw_SSL_set_cipher_list;
317         sunw_SSL_set_client_CA_list;
318         sunw_SSL_set_connect_state;
319         sunw_SSL_set_debug;
320         sunw_SSL_set_ex_data;
321         sunw_SSL_set_fd;
322         sunw_SSL_set_generate_session_id;
323         sunw_SSL_set_info_callback;
324         sunw_SSL_set_msg_callback;
325         sunw_ssl_set_peer_cert_type;
```



```
326         sunw_SSL_set_psk_client_callback;
327         sunw_SSL_set_psk_server_callback;
328         sunw_SSL_set_purpose;
329         sunw_SSL_set_quiet_shutdown;
330         sunw_SSL_set_read_ahead;
331         sunw_SSL_set_rfd;
332         sunw_SSL_set_session;
333         sunw_SSL_set_session_id_context;
334         sunw_SSL_set_session_secret_cb;
335         sunw_SSL_set_session_ticket_ext;
336         sunw_SSL_set_session_ticket_ext_cb;
337         sunw_SSL_set_shutdown;
338         sunw_SSL_set_srp_server_param;
339         sunw_SSL_set_srp_server_param_pw;
340         sunw_SSL_set_SSL_CTX;
341         sunw_SSL_set_ssl_method;
342         sunw_SSL_set_state;
343         sunw_SSL_set_tlsext_use_srtp;
344         sunw_SSL_set_tmp_dh_callback;
345         sunw_SSL_set_tmp_rsa_callback;
346         sunw_SSL_set_trust;
347         sunw_SSL_set_verify;
348         sunw_SSL_set_verify_depth;
349         sunw_SSL_set_verify_result;
350         sunw_SSL_set_wfd;
351         sunw_SSL_set1_param;
352         sunw_SSL_shutdown;
353         sunw_SSL_SRP_CTX_free;
354         sunw_SSL_SRP_CTX_init;
355         sunw_SSL_srp_server_param_with_username;
356         sunw_SSL_state;
357         sunw_SSL_state_string;
358         sunw_SSL_state_string_long;
359         sunw_ssl_undefined_const_function;
360         sunw_ssl_undefined_function;
361         sunw_ssl_undefined_void_function;
362         sunw_ssl_update_cache;
363         sunw_SSL_use_certificate;
364         sunw_SSL_use_certificate_ASN1;
365         sunw_SSL_use_certificate_file;
366         sunw_SSL_use_PrivateKey;
367         sunw_SSL_use_PrivateKey_ASN1;
368         sunw_SSL_use_PrivateKey_file;
369         sunw_SSL_use_psk_identity_hint;
370         sunw_SSL_use_RSAPrivateKey;
371         sunw_SSL_use_RSAPrivateKey_ASN1;
372         sunw_SSL_use_RSAPrivateKey_file;
373         sunw_ssl_verify_alarm_type;
374         sunw_ssl_verify_cert_chain;
375         sunw_SSL_version;
376         sunw_SSL_version_str;
377         sunw_SSL_want;
378         sunw_SSL_write;
379         sunw_ssl2_accept;
380         sunw_ssl2_callback_ctrl;
381         sunw_ssl2_ciphers;
382         sunw_ssl2_clear;
383         sunw_ssl2_connect;
384         sunw_ssl2_ctrl;
385         sunw_ssl2_ctx_callback_ctrl;
386         sunw_ssl2_ctx_ctrl;
387         sunw_ssl2_default_timeout;
388         sunw_ssl2_do_write;
389         sunw_ssl2_enc;
390         sunw_ssl2_enc_init;
391         sunw_ssl2_free;
```

```
392         sunw_ssl2_generate_key_material;
393         sunw_ssl2_get_cipher;
394         sunw_ssl2_get_cipher_by_char;
395         sunw_ssl2_mac;
396         sunw_ssl2_new;
397         sunw_ssl2_num_ciphers;
398         sunw_ssl2_part_read;
399         sunw_ssl2_peek;
400         sunw_ssl2_pending;
401         sunw_ssl2_put_cipher_by_char;
402         sunw_ssl2_read;
403         sunw_ssl2_return_error;
404         sunw_ssl2_set_certificate;
405         sunw_ssl2_shutdown;
406         sunw_ssl2_version_str;
407         sunw_ssl2_write;
408         sunw_ssl2_write_error;
409         sunw_ssl23_accept;
410         sunw_ssl23_connect;
411         sunw_ssl23_default_timeout;
412         sunw_ssl23_get_cipher;
413         sunw_ssl23_get_cipher_by_char;
414         sunw_ssl23_get_client_hello;
415         sunw_ssl23_num_ciphers;
416         sunw_ssl23_peek;
417         sunw_ssl23_put_cipher_by_char;
418         sunw_ssl23_read;
419         sunw_ssl23_read_bytes;
420         sunw_ssl23_write;
421         sunw_ssl23_write_bytes;
422         sunw_ssl3_accept;
423         sunw_ssl3_alert_code;
424         sunw_ssl3_callback_ctrl;
425         sunw_ssl3_cbc_copy_mac;
426         sunw_ssl3_cbc_digest_record;
427         sunw_ssl3_cbc_record_digest_supported;
428         sunw_ssl3_cbc_remove_padding;
429         sunw_ssl3_cert_verify_mac;
430         sunw_ssl3_change_cipher_state;
431         sunw_ssl3_check_cert_and_algorithm;
432         sunw_ssl3_check_client_hello;
433         sunw_ssl3_check_finished;
434         sunw_ssl3_choose_cipher;
435         sunw_ssl3_ciphers;
436         sunw_ssl3_cleanup_key_block;
437         sunw_ssl3_clear;
438         sunw_ssl3_client_hello;
439         sunw_ssl3_comp_find;
440         sunw_ssl3_connect;
441         sunw_ssl3_ctrl;
442         sunw_ssl3_ctx_callback_ctrl;
443         sunw_ssl3_ctx_ctrl;
444         sunw_ssl3_default_timeout;
445         sunw_ssl3_digest_cached_records;
446         sunw_ssl3_dispatch_alert;
447         sunw_ssl3_do_change_cipher_spec;
448         sunw_ssl3_do_compress;
449         sunw_ssl3_do_uncompress;
450         sunw_ssl3_do_write;
451         sunw_ssl3_enc;
452         sunw_ssl3_final_finish_mac;
453         sunw_ssl3_finish_mac;
454         sunw_ssl3_free;
455         sunw_ssl3_free_digest_list;
456         sunw_ssl3_generate_master_secret;
457         sunw_ssl3_get_cert_status;
```



```
458         sunw_ssl3_get_cert_verify;
459         sunw_ssl3_get_certificate_request;
460         sunw_ssl3_get_cipher;
461         sunw_ssl3_get_cipher_by_char;
462         sunw_ssl3_get_client_certificate;
463         sunw_ssl3_get_client_hello;
464         sunw_ssl3_get_client_key_exchange;
465         sunw_ssl3_get_finished;
466         sunw_ssl3_get_key_exchange;
467         sunw_ssl3_get_message;
468         sunw_ssl3_get_new_session_ticket;
469         sunw_ssl3_get_next_proto;
470         sunw_ssl3_get_req_cert_type;
471         sunw_ssl3_get_server_certificate;
472         sunw_ssl3_get_server_done;
473         sunw_ssl3_get_server_hello;
474         sunw_ssl3_init_finished_mac;
475         sunw_ssl3_new;
476         sunw_ssl3_num_ciphers;
477         sunw_ssl3_output_cert_chain;
478         sunw_ssl3_peek;
479         sunw_ssl3_pending;
480         sunw_ssl3_put_cipher_by_char;
481         sunw_ssl3_read;
482         sunw_ssl3_read_bytes;
483         sunw_ssl3_read_n;
484         sunw_ssl3_record_sequence_update;
485         sunw_ssl3_release_read_buffer;
486         sunw_ssl3_release_write_buffer;
487         sunw_ssl3_renegotiate;
488         sunw_ssl3_renegotiate_check;
489         sunw_ssl3_send_alert;
490         sunw_ssl3_send_cert_status;
491         sunw_ssl3_send_certificate_request;
492         sunw_ssl3_send_change_cipher_spec;
493         sunw_ssl3_send_client_certificate;
494         sunw_ssl3_send_client_key_exchange;
495         sunw_ssl3_send_client_verify;
496         sunw_ssl3_send_finished;
497         sunw_ssl3_send_hello_request;
498         sunw_ssl3_send_newsession_ticket;
499         sunw_ssl3_send_next_proto;
500         sunw_ssl3_send_server_certificate;
501         sunw_ssl3_send_server_done;
502         sunw_ssl3_send_server_hello;
503         sunw_ssl3_send_server_key_exchange;
504         sunw_ssl3_setup_buffers;
505         sunw_ssl3_setup_key_block;
506         sunw_ssl3_setup_read_buffer;
507         sunw_ssl3_setup_write_buffer;
508         sunw_ssl3_shutdown;
509         sunw_ssl3_undef_enc_method;
510         sunw_ssl3_version_str;
511         sunw_ssl3_write;
512         sunw_ssl3_write_bytes;
513         sunw_ssl3_write_pending;
514         sunw_SSLv2_client_method;
515         sunw_SSLv2_method;
516         sunw_SSLv2_server_method;
517         sunw_SSLv23_client_method;
518         sunw_SSLv23_method;
519         sunw_SSLv23_server_method;
520         sunw_SSLv3_client_method;
521         sunw_SSLv3_enc_data;
522         sunw_SSLv3_method;
523         sunw_SSLv3_server_method;
```

```
524         sunw_tls1_alert_code;
525         sunw_tls1_cbc_remove_padding;
526         sunw_tls1_cert_verify_mac;
527         sunw_tls1_change_cipher_state;
528         sunw_tls1_clear;
529         sunw_tls1_default_timeout;
530         sunw_tls1_enc;
531         sunw_tls1_export_keying_material;
532         sunw_tls1_final_finish_mac;
533         sunw_tls1_free;
534         sunw_tls1_generate_master_secret;
535         sunw_tls1_heartbeat;
536         sunw_tls1_mac;
537         sunw_tls1_new;
538         sunw_tls1_process_heartbeat;
539         sunw_tls1_process_sigalgs;
540         sunw_tls1_process_ticket;
541         sunw_tls1_setup_key_block;
542         sunw_tls1_version_str;
543         sunw_tls12_get_hash;
544         sunw_tls12_get_req_sig_algs;
545         sunw_tls12_get_sigandhash;
546         sunw_tls12_get_sigid;
547         sunw_TLSv1_1_client_method;
548         sunw_TLSv1_1_method;
549         sunw_TLSv1_1_server_method;
550         sunw_TLSv1_2_client_method;
551         sunw_TLSv1_2_method;
552         sunw_TLSv1_2_server_method;
553         sunw_TLSv1_client_method;
554         sunw_TLSv1_enc_data;
555         sunw_TLSv1_method;
556         sunw_TLSv1_server_method;
557     local:
558         *;
559 };
560 #endif /* ! codereview */
```

```
**********************************************************
   21926 Wed Aug 13 19:53:36 2014
new/usr/src/lib/openssl/libsunw_ssl/s23_clnt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s23_clnt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>

static const SSL_METHOD *ssl23_get_client_method(int ver);
static int ssl23_client_hello(SSL *s);
static int ssl23_get_server_hello(SSL *s);
static const SSL_METHOD *ssl23_get_client_method(int ver)
        {
#ifndef OPENSSL_NO_SSL2
        if (ver == SSL2_VERSION)
                return(SSLv2_client_method());
#endif
```

```
128         if (ver == SSL3_VERSION)
129                 return(SSLv3_client_method());
130         else if (ver == TLS1_VERSION)
131                 return(TLSv1_client_method());
132         else if (ver == TLS1_1_VERSION)
133                 return(TLSv1_1_client_method());
134         else if (ver == TLS1_2_VERSION)
135                 return(TLSv1_2_client_method());
136         else
137                 return(NULL);
138         }

140 IMPLEMENT_ssl23_meth_func(SSLv23_client_method,
141                           ssl_undefined_function,
142                           ssl23_connect,
143                           ssl23_get_client_method)

145 int ssl23_connect(SSL *s)
146         {
147         BUF_MEM *buf=NULL;
148         unsigned long Time=(unsigned long)time(NULL);
149         void (*cb)(const SSL *ssl,int type,int val)=NULL;
150         int ret= -1;
151         int new_state,state;

153         RAND_add(&Time,sizeof(Time),0);
154         ERR_clear_error();
155         clear_sys_error();

157         if (s->info_callback != NULL)
158                 cb=s->info_callback;
159         else if (s->ctx->info_callback != NULL)
160                 cb=s->ctx->info_callback;

162         s->in_handshake++;
163         if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

165         for (;;)
166                 {
167                 state=s->state;

169                 switch(s->state)
170                         {
171                 case SSL_ST_BEFORE:
172                 case SSL_ST_CONNECT:
173                 case SSL_ST_BEFORE|SSL_ST_CONNECT:
174                 case SSL_ST_OK|SSL_ST_CONNECT:

176                         if (s->session != NULL)
177                                 {
178                                 SSLerr(SSL_F_SSL23_CONNECT,SSL_R_SSL23_DOING_SES
179                                 ret= -1;
180                                 goto end;
181                                 }
182                         s->server=0;
183                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

185                         /* s->version=TLS1_VERSION; */
186                         s->type=SSL_ST_CONNECT;

188                         if (s->init_buf == NULL)
189                                 {
190                                 if ((buf=BUF_MEM_new()) == NULL)
191                                         {
192                                         ret= -1;
193                                         goto end;
```



```
194                                         }
195                                 if (!BUF_MEM_grow(buf,SSL3_RT_MAX_PLAIN_LENGTH))
196                                         {
197                                         ret= -1;
198                                         goto end;
199                                         }
200                                 s->init_buf=buf;
201                                 buf=NULL;
202                                 }

204                         if (!ssl3_setup_buffers(s)) { ret= -1; goto end; }

206                         ssl3_init_finished_mac(s);

208                         s->state=SSL23_ST_CW_CLNT_HELLO_A;
209                         s->ctx->stats.sess_connect++;
210                         s->init_num=0;
211                         break;

213                 case SSL23_ST_CW_CLNT_HELLO_A:
214                 case SSL23_ST_CW_CLNT_HELLO_B:

216                         s->shutdown=0;
217                         ret=ssl23_client_hello(s);
218                         if (ret <= 0) goto end;
219                         s->state=SSL23_ST_CR_SRVR_HELLO_A;
220                         s->init_num=0;

222                         break;

224                 case SSL23_ST_CR_SRVR_HELLO_A:
225                 case SSL23_ST_CR_SRVR_HELLO_B:
226                         ret=ssl23_get_server_hello(s);
227                         if (ret >= 0) cb=NULL;
228                         goto end;
229                         /* break; */

231                 default:
232                         SSLerr(SSL_F_SSL23_CONNECT,SSL_R_UNKNOWN_STATE);
233                         ret= -1;
234                         goto end;
235                         /* break; */
236                         }

238                 if (s->debug) { (void)BIO_flush(s->wbio); }

240                 if ((cb != NULL) && (s->state != state))
241                         {
242                         new_state=s->state;
243                         s->state=state;
244                         cb(s,SSL_CB_CONNECT_LOOP,1);
245                         s->state=new_state;
246                         }
247                 }
248 end:
249         s->in_handshake--;
250         if (buf != NULL)
251                 BUF_MEM_free(buf);
252         if (cb != NULL)
253                 cb(s,SSL_CB_CONNECT_EXIT,ret);
254         return(ret);
255         }

257 static int ssl23_no_ssl2_ciphers(SSL *s)
258         {
259         SSL_CIPHER *cipher;
```

```
260         STACK_OF(SSL_CIPHER) *ciphers;
261         int i;
262         ciphers = SSL_get_ciphers(s);
263         for (i = 0; i < sk_SSL_CIPHER_num(ciphers); i++)
264                 {
265                 cipher = sk_SSL_CIPHER_value(ciphers, i);
266                 if (cipher->algorithm_ssl == SSL_SSLV2)
267                         return 0;
268                 }
269         return 1;
270         }

272 /* Fill a ClientRandom or ServerRandom field of length len. Returns <= 0
273  * on failure, 1 on success. */
274 int ssl_fill_hello_random(SSL *s, int server, unsigned char *result, int len)
275         {
276         int send_time = 0;

278         if (len < 4)
279                 return 0;
280         if (server)
281                 send_time = (s->mode & SSL_MODE_SEND_SERVERHELLO_TIME) != 0;
282         else
283                 send_time = (s->mode & SSL_MODE_SEND_CLIENTHELLO_TIME) != 0;
284         if (send_time)
285                 {
286                 unsigned long Time = (unsigned long)time(NULL);
287                 unsigned char *p = result;
288                 l2n(Time, p);
289                 return RAND_pseudo_bytes(p, len-4);
290                 }
291         else
292                 return RAND_pseudo_bytes(result, len);
293         }

295 static int ssl23_client_hello(SSL *s)
296         {
297         unsigned char *buf;
298         unsigned char *p,*d;
299         int i,ch_len;
300         unsigned long l;
301         int ssl2_compat;
302         int version = 0, version_major, version_minor;
303 #ifndef OPENSSL_NO_COMP
304         int j;
305         SSL_COMP *comp;
306 #endif
307         int ret;
308         unsigned long mask, options = s->options;

310         ssl2_compat = (options & SSL_OP_NO_SSLv2) ? 0 : 1;

312         if (ssl2_compat && ssl23_no_ssl2_ciphers(s))
313                 ssl2_compat = 0;

315         /*
316          * SSL_OP_NO_X disables all protocols above X *if* there are
317          * some protocols below X enabled. This is required in order
318          * to maintain "version capability" vector contiguous. So
319          * that if application wants to disable TLS1.0 in favour of
320          * TLS1>=1, it would be insufficient to pass SSL_NO_TLSv1, the
321          * answer is SSL_OP_NO_TLSv1|SSL_OP_NO_SSLv3|SSL_OP_NO_SSLv2.
322          */
323         mask =  SSL_OP_NO_TLSv1_1|SSL_OP_NO_TLSv1
324 #if !defined(OPENSSL_NO_SSL3)
325                 |SSL_OP_NO_SSLv3
```



```
326 #endif
327 #if !defined(OPENSSL_NO_SSL2)
328                 |(ssl2_compat?SSL_OP_NO_SSLv2:0)
329 #endif
330                 ;
331 #if !defined(OPENSSL_NO_TLS1_2_CLIENT)
332         version = TLS1_2_VERSION;

334         if ((options & SSL_OP_NO_TLSv1_2) && (options & mask) != mask)
335                 version = TLS1_1_VERSION;
336 #else
337         version = TLS1_1_VERSION;
338 #endif
339         mask &= ~SSL_OP_NO_TLSv1_1;
340         if ((options & SSL_OP_NO_TLSv1_1) && (options & mask) != mask)
341                 version = TLS1_VERSION;
342         mask &= ~SSL_OP_NO_TLSv1;
343 #if !defined(OPENSSL_NO_SSL3)
344         if ((options & SSL_OP_NO_TLSv1) && (options & mask) != mask)
345                 version = SSL3_VERSION;
346         mask &= ~SSL_OP_NO_SSLv3;
347 #endif
348 #if !defined(OPENSSL_NO_SSL2)
349         if ((options & SSL_OP_NO_SSLv3) && (options & mask) != mask)
350                 version = SSL2_VERSION;
351 #endif

353 #ifndef OPENSSL_NO_TLSEXT
354         if (version != SSL2_VERSION)
355                 {
356                 /* have to disable SSL 2.0 compatibility if we need TLS extensio

358                 if (s->tlsext_hostname != NULL)
359                         ssl2_compat = 0;
360                 if (s->tlsext_status_type != -1)
361                         ssl2_compat = 0;
362 #ifdef TLSEXT_TYPE_opaque_prf_input
363                 if (s->ctx->tlsext_opaque_prf_input_callback != 0 || s->tlsext_o
364                         ssl2_compat = 0;
365 #endif
366                 }
367 #endif

369         buf=(unsigned char *)s->init_buf->data;
370         if (s->state == SSL23_ST_CW_CLNT_HELLO_A)
371                 {
372 #if 0
373                 /* don't reuse session-id's */
374                 if (!ssl_get_new_session(s,0))
375                         {
376                         return(-1);
377                         }
378 #endif

380                 p=s->s3->client_random;
381                 if (ssl_fill_hello_random(s, 0, p, SSL3_RANDOM_SIZE) <= 0)
382                         return -1;

384                 if (version == TLS1_2_VERSION)
385                         {
386                         version_major = TLS1_2_VERSION_MAJOR;
387                         version_minor = TLS1_2_VERSION_MINOR;
388                         }
389                 else if (version == TLS1_1_VERSION)
390                         {
391                         version_major = TLS1_1_VERSION_MAJOR;
```

```
392                         version_minor = TLS1_1_VERSION_MINOR;
393                         }
394                 else if (version == TLS1_VERSION)
395                         {
396                         version_major = TLS1_VERSION_MAJOR;
397                         version_minor = TLS1_VERSION_MINOR;
398                         }
399 #ifdef OPENSSL_FIPS
400                 else if(FIPS_mode())
401                         {
402                         SSLerr(SSL_F_SSL23_CLIENT_HELLO,
403                                       SSL_R_ONLY_TLS_ALLOWED_IN_FIPS_MODE);
404                         return -1;
405                         }
406 #endif
407                 else if (version == SSL3_VERSION)
408                         {
409                         version_major = SSL3_VERSION_MAJOR;
410                         version_minor = SSL3_VERSION_MINOR;
411                         }
412                 else if (version == SSL2_VERSION)
413                         {
414                         version_major = SSL2_VERSION_MAJOR;
415                         version_minor = SSL2_VERSION_MINOR;
416                         }
417                 else
418                         {
419                         SSLerr(SSL_F_SSL23_CLIENT_HELLO,SSL_R_NO_PROTOCOLS_AVAIL
420                         return(-1);
421                         }

423                 s->client_version = version;

425                 if (ssl2_compat)
426                         {
427                         /* create SSL 2.0 compatible Client Hello */

429                         /* two byte record header will be written last */
430                         d = &(buf[2]);
431                         p = d + 9; /* leave space for message type, version, ind

433                         *(d++) = SSL2_MT_CLIENT_HELLO;
434                         *(d++) = version_major;
435                         *(d++) = version_minor;

437                         /* Ciphers supported */
438                         i=ssl_cipher_list_to_bytes(s,SSL_get_ciphers(s),p,0);
439                         if (i == 0)
440                                 {
441                                 /* no ciphers */
442                                 SSLerr(SSL_F_SSL23_CLIENT_HELLO,SSL_R_NO_CIPHERS
443                                 return -1;
444                                 }
445                         s2n(i,d);
446                         p+=i;

448                         /* put in the session-id length (zero since there is no
449 #if 0
450                         s->session->session_id_length=0;
451 #endif
452                         s2n(0,d);

454                         if (s->options & SSL_OP_NETSCAPE_CHALLENGE_BUG)
455                                 ch_len=SSL2_CHALLENGE_LENGTH;
456                         else
457                                 ch_len=SSL2_MAX_CHALLENGE_LENGTH;
```



```
459                         /* write out sslv2 challenge */
460                         /* Note that ch_len must be <= SSL3_RANDOM_SIZE (32),
461                            because it is one of SSL2_MAX_CHALLENGE_LENGTH (32)
462                            or SSL2_MAX_CHALLENGE_LENGTH (16), but leave the
463                            check in for futurproofing */
464                         if (SSL3_RANDOM_SIZE < ch_len)
465                                 i=SSL3_RANDOM_SIZE;
466                         else
467                                 i=ch_len;
468                         s2n(i,d);
469                         memset(&(s->s3->client_random[0]),0,SSL3_RANDOM_SIZE);
470                         if (RAND_pseudo_bytes(&(s->s3->client_random[SSL3_RANDOM
471                                 return -1;

473                         memcpy(p,&(s->s3->client_random[SSL3_RANDOM_SIZE-i]),i);
474                         p+=i;

476                         i= p- &(buf[2]);
477                         buf[0]=((i>>8)&0xff)|0x80;
478                         buf[1]=(i&0xff);

480                         /* number of bytes to write */
481                         s->init_num=i+2;
482                         s->init_off=0;

484                         ssl3_finish_mac(s,&(buf[2]),i);
485                         }
486                 else
487                         {
488                         /* create Client Hello in SSL 3.0/TLS 1.0 format */

490                         /* do the record header (5 bytes) and handshake message
491                         d = p = &(buf[9]);

493                         *(p++) = version_major;
494                         *(p++) = version_minor;

496                         /* Random stuff */
497                         memcpy(p, s->s3->client_random, SSL3_RANDOM_SIZE);
498                         p += SSL3_RANDOM_SIZE;

500                         /* Session ID (zero since there is no reuse) */
501                         *(p++) = 0;

503                         /* Ciphers supported (using SSL 3.0/TLS 1.0 format) */
504                         i=ssl_cipher_list_to_bytes(s,SSL_get_ciphers(s),&(p[2]),
505                         if (i == 0)
506                                 {
507                                 SSLerr(SSL_F_SSL23_CLIENT_HELLO,SSL_R_NO_CIPHERS
508                                 return -1;
509                                 }
510 #ifdef OPENSSL_MAX_TLS1_2_CIPHER_LENGTH
511                         /* Some servers hang if client hello > 256 bytes
512                          * as hack workaround chop number of supported ciphers
513                          * to keep it well below this if we use TLS v1.2
514                          */
515                         if (TLS1_get_version(s) >= TLS1_2_VERSION
516                                 && i > OPENSSL_MAX_TLS1_2_CIPHER_LENGTH)
517                                 i = OPENSSL_MAX_TLS1_2_CIPHER_LENGTH & ~1;
518 #endif
519                         s2n(i,p);
520                         p+=i;

522                         /* COMPRESSION */
523 #ifdef OPENSSL_NO_COMP
```

```
524                         *(p++)=1;
525 #else
526                         if ((s->options & SSL_OP_NO_COMPRESSION)
527                                                 || !s->ctx->comp_methods)
528                                 j=0;
529                         else
530                                 j=sk_SSL_COMP_num(s->ctx->comp_methods);
531                         *(p++)=1+j;
532                         for (i=0; i<j; i++)
533                                 {
534                                 comp=sk_SSL_COMP_value(s->ctx->comp_methods,i);
535                                 *(p++)=comp->id;
536                                 }
537 #endif
538                         *(p++)=0; /* Add the NULL method */
539
540 #ifndef OPENSSL_NO_TLSEXT
541                         /* TLS extensions*/
542                         if (ssl_prepare_clienthello_tlsext(s) <= 0)
543                                 {
544                                 SSLerr(SSL_F_SSL23_CLIENT_HELLO,SSL_R_CLIENTHELL
545                                 return -1;
546                                 }
547                         if ((p = ssl_add_clienthello_tlsext(s, p, buf+SSL3_RT_MA
548                                 {
549                                 SSLerr(SSL_F_SSL23_CLIENT_HELLO,ERR_R_INTERNAL_E
550                                 return -1;
551                                 }
552 #endif
553
554                         l = p-d;
555
556                         /* fill in 4-byte handshake header */
557                         d=&(buf[5]);
558                         *(d++)=SSL3_MT_CLIENT_HELLO;
559                         l2n3(l,d);
560
561                         l += 4;
562
563                         if (l > SSL3_RT_MAX_PLAIN_LENGTH)
564                                 {
565                                 SSLerr(SSL_F_SSL23_CLIENT_HELLO,ERR_R_INTERNAL_E
566                                 return -1;
567                                 }
568
569                         /* fill in 5-byte record header */
570                         d=buf;
571                         *(d++) = SSL3_RT_HANDSHAKE;
572                         *(d++) = version_major;
573                         /* Some servers hang if we use long client hellos
574                          * and a record number > TLS 1.0.
575                          */
576                         if (TLS1_get_client_version(s) > TLS1_VERSION)
577                                 *(d++) = 1;
578                         else
579                                 *(d++) = version_minor;
580                         s2n((int)l,d);
581
582                         /* number of bytes to write */
583                         s->init_num=p-buf;
584                         s->init_off=0;
585
586                         ssl3_finish_mac(s,&(buf[5]), s->init_num - 5);
587                         }
588
589                 s->state=SSL23_ST_CW_CLNT_HELLO_B;
```



```
590                 s->init_off=0;
591                 }
592
593         /* SSL3_ST_CW_CLNT_HELLO_B */
594         ret = ssl23_write_bytes(s);
595
596         if ((ret >= 2) && s->msg_callback)
597                 {
598                 /* Client Hello has been sent; tell msg_callback */
599
600                 if (ssl2_compat)
601                         s->msg_callback(1, SSL2_VERSION, 0, s->init_buf->data+2,
602                 else
603                         s->msg_callback(1, version, SSL3_RT_HANDSHAKE, s->init_b
604                 }
605
606         return ret;
607         }
608
609 static int ssl23_get_server_hello(SSL *s)
610         {
611         char buf[8];
612         unsigned char *p;
613         int i;
614         int n;
615
616         n=ssl23_read_bytes(s,7);
617
618         if (n != 7) return(n);
619         p=s->packet;
620
621         memcpy(buf,p,n);
622
623         if ((p[0] & 0x80) && (p[2] == SSL2_MT_SERVER_HELLO) &&
624                 (p[5] == 0x00) && (p[6] == 0x02))
625                 {
626 #ifdef OPENSSL_NO_SSL2
627                 SSLerr(SSL_F_SSL23_GET_SERVER_HELLO,SSL_R_UNSUPPORTED_PROTOCOL);
628                 goto err;
629 #else
630                 /* we are talking sslv2 */
631                 /* we need to clean up the SSLv3 setup and put in the
632                  * sslv2 stuff. */
633                 int ch_len;
634
635                 if (s->options & SSL_OP_NO_SSLv2)
636                         {
637                         SSLerr(SSL_F_SSL23_GET_SERVER_HELLO,SSL_R_UNSUPPORTED_PR
638                         goto err;
639                         }
640                 if (s->s2 == NULL)
641                         {
642                         if (!ssl2_new(s))
643                                 goto err;
644                         }
645                 else
646                         ssl2_clear(s);
647
648                 if (s->options & SSL_OP_NETSCAPE_CHALLENGE_BUG)
649                         ch_len=SSL2_CHALLENGE_LENGTH;
650                 else
651                         ch_len=SSL2_MAX_CHALLENGE_LENGTH;
652
653                 /* write out sslv2 challenge */
654                 /* Note that ch_len must be <= SSL3_RANDOM_SIZE (32), because
655                    it is one of SSL2_MAX_CHALLENGE_LENGTH (32) or
```

```
656                     SSL2_MAX_CHALLENGE_LENGTH (16), but leave the check in for
657                     futurproofing */
658                 i=(SSL3_RANDOM_SIZE < ch_len)
659                         ?SSL3_RANDOM_SIZE:ch_len;
660                 s->s2->challenge_length=i;
661                 memcpy(s->s2->challenge,
662                         &(s->s3->client_random[SSL3_RANDOM_SIZE-i]),i);
663
664                 if (s->s3 != NULL) ssl3_free(s);
665
666                 if (!BUF_MEM_grow_clean(s->init_buf,
667                         SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER))
668                         {
669                         SSLerr(SSL_F_SSL23_GET_SERVER_HELLO,ERR_R_BUF_LIB);
670                         goto err;
671                         }
672
673                 s->state=SSL2_ST_GET_SERVER_HELLO_A;
674                 if (!(s->client_version == SSL2_VERSION))
675                         /* use special padding (SSL 3.0 draft/RFC 2246, App. E.2
676                         s->s2->ssl2_rollback=1;
677
678                 /* setup the 7 bytes we have read so we get them from
679                  * the sslv2 buffer */
680                 s->rstate=SSL_ST_READ_HEADER;
681                 s->packet_length=n;
682                 s->packet= &(s->s2->rbuf[0]);
683                 memcpy(s->packet,buf,n);
684                 s->s2->rbuf_left=n;
685                 s->s2->rbuf_offs=0;
686
687                 /* we have already written one */
688                 s->s2->write_sequence=1;
689
690                 s->method=SSLv2_client_method();
691                 s->handshake_func=s->method->ssl_connect;
692 #endif
693                 }
694         else if (p[1] == SSL3_VERSION_MAJOR &&
695                  p[2] <= TLS1_2_VERSION_MINOR &&
696                  ((p[0] == SSL3_RT_HANDSHAKE && p[5] == SSL3_MT_SERVER_HELLO) ||
697                   (p[0] == SSL3_RT_ALERT && p[3] == 0 && p[4] == 2)))
698                 {
699                 /* we have sslv3 or tls1 (server hello or alert) */
700
701                 if ((p[2] == SSL3_VERSION_MINOR) &&
702                         !(s->options & SSL_OP_NO_SSLv3))
703                         {
704 #ifdef OPENSSL_FIPS
705                         if(FIPS_mode())
706                                 {
707                                 SSLerr(SSL_F_SSL23_GET_SERVER_HELLO,
708                                         SSL_R_ONLY_TLS_ALLOWED_IN_FIPS_MODE);
709                                 goto err;
710                                 }
711 #endif
712                         s->version=SSL3_VERSION;
713                         s->method=SSLv3_client_method();
714                         }
715                 else if ((p[2] == TLS1_VERSION_MINOR) &&
716                         !(s->options & SSL_OP_NO_TLSv1))
717                         {
718                         s->version=TLS1_VERSION;
719                         s->method=TLSv1_client_method();
720                         }
721                 else if ((p[2] == TLS1_1_VERSION_MINOR) &&
```



```
722                         !(s->options & SSL_OP_NO_TLSv1_1))
723                         {
724                         s->version=TLS1_1_VERSION;
725                         s->method=TLSv1_1_client_method();
726                         }
727                 else if ((p[2] == TLS1_2_VERSION_MINOR) &&
728                         !(s->options & SSL_OP_NO_TLSv1_2))
729                         {
730                         s->version=TLS1_2_VERSION;
731                         s->method=TLSv1_2_client_method();
732                         }
733                 else
734                         {
735                         SSLerr(SSL_F_SSL23_GET_SERVER_HELLO,SSL_R_UNSUPPORTED_PR
736                         goto err;
737                         }
738
739                 if (p[0] == SSL3_RT_ALERT && p[5] != SSL3_AL_WARNING)
740                         {
741                         /* fatal alert */
742
743                         void (*cb)(const SSL *ssl,int type,int val)=NULL;
744                         int j;
745
746                         if (s->info_callback != NULL)
747                                 cb=s->info_callback;
748                         else if (s->ctx->info_callback != NULL)
749                                 cb=s->ctx->info_callback;
750
751                         i=p[5];
752                         if (cb != NULL)
753                                 {
754                                 j=(i<<8)|p[6];
755                                 cb(s,SSL_CB_READ_ALERT,j);
756                                 }
757
758                         if (s->msg_callback)
759                                 s->msg_callback(0, s->version, SSL3_RT_ALERT, p+
760
761                         s->rwstate=SSL_NOTHING;
762                         SSLerr(SSL_F_SSL23_GET_SERVER_HELLO,SSL_AD_REASON_OFFSET
763                         goto err;
764                         }
765
766                 if (!ssl_init_wbio_buffer(s,1)) goto err;
767
768                 /* we are in this state */
769                 s->state=SSL3_ST_CR_SRVR_HELLO_A;
770
771                 /* put the 7 bytes we have read into the input buffer
772                  * for SSLv3 */
773                 s->rstate=SSL_ST_READ_HEADER;
774                 s->packet_length=n;
775                 if (s->s3->rbuf.buf == NULL)
776                         if (!ssl3_setup_read_buffer(s))
777                                 goto err;
778                 s->packet= &(s->s3->rbuf.buf[0]);
779                 memcpy(s->packet,buf,n);
780                 s->s3->rbuf.left=n;
781                 s->s3->rbuf.offset=0;
782
783                 s->handshake_func=s->method->ssl_connect;
784                 }
785         else
786                 {
787                 SSLerr(SSL_F_SSL23_GET_SERVER_HELLO,SSL_R_UNKNOWN_PROTOCOL);
```

```
788                 goto err;
789                 }
790         s->init_num=0;

792         /* Since, if we are sending a ssl23 client hello, we are not
793          * reusing a session-id */
794         if (!ssl_get_new_session(s,0))
795                 goto err;

797         return(SSL_connect(s));
798 err:
799         return(-1);
800         }
801 #endif /* ! codereview */
```

```
**********************************************************
    5595 Wed Aug 13 19:53:36 2014
new/usr/src/lib/openssl/libsunw_ssl/s23_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s23_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/objects.h>
#include "ssl_locl.h"
```



```
long ssl23_default_timeout(void)
        {
        return(300);
        }

int ssl23_num_ciphers(void)
        {
        return(ssl3_num_ciphers()
#ifndef OPENSSL_NO_SSL2
                + ssl2_num_ciphers()
#endif
            );
        }

const SSL_CIPHER *ssl23_get_cipher(unsigned int u)
        {
        unsigned int uu=ssl3_num_ciphers();

        if (u < uu)
                return(ssl3_get_cipher(u));
        else
#ifndef OPENSSL_NO_SSL2
                return(ssl2_get_cipher(u-uu));
#else
                return(NULL);
#endif
        }

/* This function needs to check if the ciphers required are actually
 * available */
const SSL_CIPHER *ssl23_get_cipher_by_char(const unsigned char *p)
        {
        const SSL_CIPHER *cp;

        cp=ssl3_get_cipher_by_char(p);
#ifndef OPENSSL_NO_SSL2
        if (cp == NULL)
                cp=ssl2_get_cipher_by_char(p);
#endif
        return(cp);
        }

int ssl23_put_cipher_by_char(const SSL_CIPHER *c, unsigned char *p)
        {
        long l;

        /* We can write SSLv2 and SSLv3 ciphers */
        /* but no ECC ciphers */
        if (c->algorithm_mkey == SSL_kECDHr ||
                c->algorithm_mkey == SSL_kECDHe ||
                c->algorithm_mkey == SSL_kEECDH ||
                c->algorithm_auth == SSL_aECDH ||
                c->algorithm_auth == SSL_aECDSA)
                return 0;
        if (p != NULL)
                {
                l=c->id;
                p[0]=((unsigned char)(l>>16L))&0xFF;
                p[1]=((unsigned char)(l>> 8L))&0xFF;
                p[2]=((unsigned char)(l     ))&0xFF;
                }
        return(3);
        }

int ssl23_read(SSL *s, void *buf, int len)
```

```
128         {
129         int n;

131         clear_sys_error();
132         if (SSL_in_init(s) && (!s->in_handshake))
133                 {
134                 n=s->handshake_func(s);
135                 if (n < 0) return(n);
136                 if (n == 0)
137                         {
138                         SSLerr(SSL_F_SSL23_READ,SSL_R_SSL_HANDSHAKE_FAILURE);
139                         return(-1);
140                         }
141                 return(SSL_read(s,buf,len));
142                 }
143         else
144                 {
145                 ssl_undefined_function(s);
146                 return(-1);
147                 }
148         }

150 int ssl23_peek(SSL *s, void *buf, int len)
151         {
152         int n;

154         clear_sys_error();
155         if (SSL_in_init(s) && (!s->in_handshake))
156                 {
157                 n=s->handshake_func(s);
158                 if (n < 0) return(n);
159                 if (n == 0)
160                         {
161                         SSLerr(SSL_F_SSL23_PEEK,SSL_R_SSL_HANDSHAKE_FAILURE);
162                         return(-1);
163                         }
164                 return(SSL_peek(s,buf,len));
165                 }
166         else
167                 {
168                 ssl_undefined_function(s);
169                 return(-1);
170                 }
171         }

173 int ssl23_write(SSL *s, const void *buf, int len)
174         {
175         int n;

177         clear_sys_error();
178         if (SSL_in_init(s) && (!s->in_handshake))
179                 {
180                 n=s->handshake_func(s);
181                 if (n < 0) return(n);
182                 if (n == 0)
183                         {
184                         SSLerr(SSL_F_SSL23_WRITE,SSL_R_SSL_HANDSHAKE_FAILURE);
185                         return(-1);
186                         }
187                 return(SSL_write(s,buf,len));
188                 }
189         else
190                 {
191                 ssl_undefined_function(s);
192                 return(-1);
193                 }
```



```
194         }
195 #endif /* ! codereview */
```

```
**********************************************************
    3868 Wed Aug 13 19:53:36 2014
new/usr/src/lib/openssl/libsunw_ssl/s23_meth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s23_meth.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/objects.h>
#include "ssl_locl.h"
```



```
static const SSL_METHOD *ssl23_get_method(int ver);
static const SSL_METHOD *ssl23_get_method(int ver)
        {
#ifndef OPENSSL_NO_SSL2
        if (ver == SSL2_VERSION)
                return(SSLv2_method());
        else
#endif
#ifndef OPENSSL_NO_SSL3
        if (ver == SSL3_VERSION)
                return(SSLv3_method());
        else
#endif
#ifndef OPENSSL_NO_TLS1
        if (ver == TLS1_VERSION)
                return(TLSv1_method());
        else if (ver == TLS1_1_VERSION)
                return(TLSv1_1_method());
        else if (ver == TLS1_2_VERSION)
                return(TLSv1_2_method());
        else
#endif
                return(NULL);
        }

IMPLEMENT_ssl23_meth_func(SSLv23_method,
                          ssl23_accept,
                          ssl23_connect,
                          ssl23_get_method)
#endif /* ! codereview */
```

```
**********************************************************
    4180 Wed Aug 13 19:53:36 2014
new/usr/src/lib/openssl/libsunw_ssl/s23_pkt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s23_pkt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <errno.h>
#define USE_SOCKETS
```



```
#include "ssl_locl.h"
#include <openssl/evp.h>
#include <openssl/buffer.h>

int ssl23_write_bytes(SSL *s)
        {
        int i,num,tot;
        char *buf;

        buf=s->init_buf->data;
        tot=s->init_off;
        num=s->init_num;
        for (;;)
                {
                s->rwstate=SSL_WRITING;
                i=BIO_write(s->wbio,&(buf[tot]),num);
                if (i <= 0)
                        {
                        s->init_off=tot;
                        s->init_num=num;
                        return(i);
                        }
                s->rwstate=SSL_NOTHING;
                if (i == num) return(tot+i);

                num-=i;
                tot+=i;
                }
        }

/* return regularly only when we have read (at least) 'n' bytes */
int ssl23_read_bytes(SSL *s, int n)
        {
        unsigned char *p;
        int j;

        if (s->packet_length < (unsigned int)n)
                {
                p=s->packet;

                for (;;)
                        {
                        s->rwstate=SSL_READING;
                        j=BIO_read(s->rbio,(char *)&(p[s->packet_length]),
                                n-s->packet_length);
                        if (j <= 0)
                                return(j);
                        s->rwstate=SSL_NOTHING;
                        s->packet_length+=j;
                        if (s->packet_length >= (unsigned int)n)
                                return(s->packet_length);
                        }
                }
        return(n);
        }
#endif /* ! codereview */
```

```
**********************************************************
   18904 Wed Aug 13 19:53:36 2014
new/usr/src/lib/openssl/libsunw_ssl/s23_srvr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/s23_srvr.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif

static const SSL_METHOD *ssl23_get_server_method(int ver);
int ssl23_get_client_hello(SSL *s);
static const SSL_METHOD *ssl23_get_server_method(int ver)
        {
#ifndef OPENSSL_NO_SSL2
        if (ver == SSL2_VERSION)
```

```
128                 return(SSLv2_server_method());
129 #endif
130         if (ver == SSL3_VERSION)
131                 return(SSLv3_server_method());
132         else if (ver == TLS1_VERSION)
133                 return(TLSv1_server_method());
134         else if (ver == TLS1_1_VERSION)
135                 return(TLSv1_1_server_method());
136         else if (ver == TLS1_2_VERSION)
137                 return(TLSv1_2_server_method());
138         else
139                 return(NULL);
140         }

142 IMPLEMENT_ssl23_meth_func(SSLv23_server_method,
143                         ssl23_accept,
144                         ssl_undefined_function,
145                         ssl23_get_server_method)

147 int ssl23_accept(SSL *s)
148         {
149         BUF_MEM *buf;
150         unsigned long Time=(unsigned long)time(NULL);
151         void (*cb)(const SSL *ssl,int type,int val)=NULL;
152         int ret= -1;
153         int new_state,state;

155         RAND_add(&Time,sizeof(Time),0);
156         ERR_clear_error();
157         clear_sys_error();

159         if (s->info_callback != NULL)
160                 cb=s->info_callback;
161         else if (s->ctx->info_callback != NULL)
162                 cb=s->ctx->info_callback;

164         s->in_handshake++;
165         if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

167         for (;;)
168                 {
169                 state=s->state;

171                 switch(s->state)
172                         {
173                 case SSL_ST_BEFORE:
174                 case SSL_ST_ACCEPT:
175                 case SSL_ST_BEFORE|SSL_ST_ACCEPT:
176                 case SSL_ST_OK|SSL_ST_ACCEPT:

178                         s->server=1;
179                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

181                         /* s->version=SSL3_VERSION; */
182                         s->type=SSL_ST_ACCEPT;

184                         if (s->init_buf == NULL)
185                                 {
186                                 if ((buf=BUF_MEM_new()) == NULL)
187                                         {
188                                         ret= -1;
189                                         goto end;
190                                         }
191                                 if (!BUF_MEM_grow(buf,SSL3_RT_MAX_PLAIN_LENGTH))
192                                         {
193                                         ret= -1;
```



```
194                                         goto end;
195                                         }
196                                 s->init_buf=buf;
197                                 }

199                         ssl3_init_finished_mac(s);

201                         s->state=SSL23_ST_SR_CLNT_HELLO_A;
202                         s->ctx->stats.sess_accept++;
203                         s->init_num=0;
204                         break;

206                 case SSL23_ST_SR_CLNT_HELLO_A:
207                 case SSL23_ST_SR_CLNT_HELLO_B:

209                         s->shutdown=0;
210                         ret=ssl23_get_client_hello(s);
211                         if (ret >= 0) cb=NULL;
212                         goto end;
213                         /* break; */

215                 default:
216                         SSLerr(SSL_F_SSL23_ACCEPT,SSL_R_UNKNOWN_STATE);
217                         ret= -1;
218                         goto end;
219                         /* break; */
220                         }

222                 if ((cb != NULL) && (s->state != state))
223                         {
224                         new_state=s->state;
225                         s->state=state;
226                         cb(s,SSL_CB_ACCEPT_LOOP,1);
227                         s->state=new_state;
228                         }
229                 }
230 end:
231         s->in_handshake--;
232         if (cb != NULL)
233                 cb(s,SSL_CB_ACCEPT_EXIT,ret);
234         return(ret);
235         }


238 int ssl23_get_client_hello(SSL *s)
239         {
240         char buf_space[11]; /* Request this many bytes in initial read.
241                              * We can detect SSL 3.0/TLS 1.0 Client Hellos
242                              * ('type == 3') correctly only when the following
243                              * is in a single record, which is not guaranteed by
244                              * the protocol specification:
245                              * Byte  Content
246                              *  0     type            \
247                              *  1/2   version          > record header
248                              *  3/4   length          /
249                              *  5     msg_type        \
250                              *  6-8   length           > Client Hello message
251                              *  9/10  client_version  /
252                              */
253         char *buf= &(buf_space[0]);
254         unsigned char *p,*d,*d_len,*dd;
255         unsigned int i;
256         unsigned int csl,sil,cl;
257         int n=0,j;
258         int type=0;
259         int v[2];
```

```
261         if (s->state == SSL23_ST_SR_CLNT_HELLO_A)
262                 {
263                 /* read the initial header */
264                 v[0]=v[1]=0;

266                 if (!ssl3_setup_buffers(s)) goto err;

268                 n=ssl23_read_bytes(s, sizeof buf_space);
269                 if (n != sizeof buf_space) return(n); /* n == -1 || n == 0 */

271                 p=s->packet;

273                 memcpy(buf,p,n);

275                 if ((p[0] & 0x80) && (p[2] == SSL2_MT_CLIENT_HELLO))
276                         {
277                         /*
278                          * SSLv2 header
279                          */
280                         if ((p[3] == 0x00) && (p[4] == 0x02))
281                                 {
282                                 v[0]=p[3]; v[1]=p[4];
283                                 /* SSLv2 */
284                                 if (!(s->options & SSL_OP_NO_SSLv2))
285                                         type=1;
286                                 }
287                         else if (p[3] == SSL3_VERSION_MAJOR)
288                                 {
289                                 v[0]=p[3]; v[1]=p[4];
290                                 /* SSLv3/TLSv1 */
291                                 if (p[4] >= TLS1_VERSION_MINOR)
292                                         {
293                                         if (p[4] >= TLS1_2_VERSION_MINOR &&
294                                            !(s->options & SSL_OP_NO_TLSv1_2))
295                                                 {
296                                                 s->version=TLS1_2_VERSION;
297                                                 s->state=SSL23_ST_SR_CLNT_HELLO_
298                                                 }
299                                         else if (p[4] >= TLS1_1_VERSION_MINOR &&
300                                            !(s->options & SSL_OP_NO_TLSv1_1))
301                                                 {
302                                                 s->version=TLS1_1_VERSION;
303                                                 /* type=2; */ /* done later to s
304                                                 s->state=SSL23_ST_SR_CLNT_HELLO_
305                                                 }
306                                         else if (!(s->options & SSL_OP_NO_TLSv1)
307                                                 {
308                                                 s->version=TLS1_VERSION;
309                                                 /* type=2; */ /* done later to s
310                                                 s->state=SSL23_ST_SR_CLNT_HELLO_
311                                                 }
312                                         else if (!(s->options & SSL_OP_NO_SSLv3)
313                                                 {
314                                                 s->version=SSL3_VERSION;
315                                                 /* type=2; */
316                                                 s->state=SSL23_ST_SR_CLNT_HELLO_
317                                                 }
318                                         else if (!(s->options & SSL_OP_NO_SSLv2)
319                                                 {
320                                                 type=1;
321                                                 }
322                                         }
323                                 else if (!(s->options & SSL_OP_NO_SSLv3))
324                                         {
325                                         s->version=SSL3_VERSION;
```



```
326                                         /* type=2; */
327                                         s->state=SSL23_ST_SR_CLNT_HELLO_B;
328                                         }
329                                 else if (!(s->options & SSL_OP_NO_SSLv2))
330                                         type=1;

332                                 }
333                         }
334                 else if ((p[0] == SSL3_RT_HANDSHAKE) &&
335                          (p[1] == SSL3_VERSION_MAJOR) &&
336                          (p[5] == SSL3_MT_CLIENT_HELLO) &&
337                          ((p[3] == 0 && p[4] < 5 /* silly record length? */)
338                                 || (p[9] >= p[1])))
339                         {
340                         /*
341                          * SSLv3 or tls1 header
342                          */

344                         v[0]=p[1]; /* major version (= SSL3_VERSION_MAJOR) */
345                         /* We must look at client_version inside the Client Hell
346                          * to get the correct minor version.
347                          * However if we have only a pathologically small fragme
348                          * Client Hello message, this would be difficult, and we
349                          * to read more records to find out.
350                          * No known SSL 3.0 client fragments ClientHello like th
351                          * so we simply reject such connections to avoid
352                          * protocol version downgrade attacks. */
353                         if (p[3] == 0 && p[4] < 6)
354                                 {
355                                 SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_RECORD
356                                 goto err;
357                                 }
358                         /* if major version number > 3 set minor to a value
359                          * which will use the highest version 3 we support.
360                          * If TLS 2.0 ever appears we will need to revise
361                          * this....
362                          */
363                         if (p[9] > SSL3_VERSION_MAJOR)
364                                 v[1]=0xff;
365                         else
366                                 v[1]=p[10]; /* minor version according to client
367                         if (v[1] >= TLS1_VERSION_MINOR)
368                                 {
369                                 if (v[1] >= TLS1_2_VERSION_MINOR &&
370                                         !(s->options & SSL_OP_NO_TLSv1_2))
371                                         {
372                                         s->version=TLS1_2_VERSION;
373                                         type=3;
374                                         }
375                                 else if (v[1] >= TLS1_1_VERSION_MINOR &&
376                                         !(s->options & SSL_OP_NO_TLSv1_1))
377                                         {
378                                         s->version=TLS1_1_VERSION;
379                                         type=3;
380                                         }
381                                 else if (!(s->options & SSL_OP_NO_TLSv1))
382                                         {
383                                         s->version=TLS1_VERSION;
384                                         type=3;
385                                         }
386                                 else if (!(s->options & SSL_OP_NO_SSLv3))
387                                         {
388                                         s->version=SSL3_VERSION;
389                                         type=3;
390                                         }
391                                 }
```

```
392                         else
393                                 {
394                                 /* client requests SSL 3.0 */
395                                 if (!(s->options & SSL_OP_NO_SSLv3))
396                                         {
397                                         s->version=SSL3_VERSION;
398                                         type=3;
399                                         }
400                                 else if (!(s->options & SSL_OP_NO_TLSv1))
401                                         {
402                                         /* we won't be able to use TLS of course
403                                          * but this will send an appropriate ale
404                                         s->version=TLS1_VERSION;
405                                         type=3;
406                                         }
407                                 }
408                         }
409                 else if ((strncmp("GET ", (char *)p,4) == 0) ||
410                          (strncmp("POST ",(char *)p,5) == 0) ||
411                          (strncmp("HEAD ",(char *)p,5) == 0) ||
412                          (strncmp("PUT ", (char *)p,4) == 0))
413                         {
414                         SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_HTTP_REQUEST);
415                         goto err;
416                         }
417                 else if (strncmp("CONNECT",(char *)p,7) == 0)
418                         {
419                         SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_HTTPS_PROXY_RE
420                         goto err;
421                         }
422                 }

424 #ifdef OPENSSL_FIPS
425         if (FIPS_mode() && (s->version < TLS1_VERSION))
426                 {
427                 SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,
428                                         SSL_R_ONLY_TLS_ALLOWED_IN_FIPS_MODE);
429                 goto err;
430                 }
431 #endif

433         if (s->state == SSL23_ST_SR_CLNT_HELLO_B)
434                 {
435                 /* we have SSLv3/TLSv1 in an SSLv2 header
436                  * (other cases skip this state) */

438                 type=2;
439                 p=s->packet;
440                 v[0] = p[3]; /* == SSL3_VERSION_MAJOR */
441                 v[1] = p[4];

443                 /* An SSLv3/TLSv1 backwards-compatible CLIENT-HELLO in an SSLv2
444                  * header is sent directly on the wire, not wrapped as a TLS
445                  * record. It's format is:
446                  * Byte  Content
447                  * 0-1   msg_length
448                  * 2     msg_type
449                  * 3-4   version
450                  * 5-6   cipher_spec_length
451                  * 7-8   session_id_length
452                  * 9-10  challenge_length
453                  * ...   ...
454                  */
455                 n=((p[0]&0x7f)<<8)|p[1];
456                 if (n > (1024*4))
457                         {
```



```
458                         SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_RECORD_TOO_LAR
459                         goto err;
460                         }
461                 if (n < 9)
462                         {
463                         SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_RECORD_LENGTH_
464                         goto err;
465                         }

467                 j=ssl23_read_bytes(s,n+2);
468                 /* We previously read 11 bytes, so if j > 0, we must have
469                  * j == n+2 == s->packet_length. We have at least 11 valid
470                  * packet bytes. */
471                 if (j <= 0) return(j);

473                 ssl3_finish_mac(s, s->packet+2, s->packet_length-2);
474                 if (s->msg_callback)
475                         s->msg_callback(0, SSL2_VERSION, 0, s->packet+2, s->pack

477                 p=s->packet;
478                 p+=5;
479                 n2s(p,csl);
480                 n2s(p,sil);
481                 n2s(p,cl);
482                 d=(unsigned char *)s->init_buf->data;
483                 if ((csl+sil+cl+11) != s->packet_length) /* We can't have TLS ex
484                                                          * Client Hello, can we
485                                                          * '>' otherweise */
486                         {
487                         SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_RECORD_LENGTH_
488                         goto err;
489                         }

491                 /* record header: msg_type ... */
492                 *(d++) = SSL3_MT_CLIENT_HELLO;
493                 /* ... and length (actual value will be written later) */
494                 d_len = d;
495                 d += 3;

497                 /* client_version */
498                 *(d++) = SSL3_VERSION_MAJOR; /* == v[0] */
499                 *(d++) = v[1];

501                 /* lets populate the random area */
502                 /* get the challenge_length */
503                 i=(cl > SSL3_RANDOM_SIZE)?SSL3_RANDOM_SIZE:cl;
504                 memset(d,0,SSL3_RANDOM_SIZE);
505                 memcpy(&(d[SSL3_RANDOM_SIZE-i]),&(p[csl+sil]),i);
506                 d+=SSL3_RANDOM_SIZE;

508                 /* no session-id reuse */
509                 *(d++)=0;

511                 /* ciphers */
512                 j=0;
513                 dd=d;
514                 d+=2;
515                 for (i=0; i<csl; i+=3)
516                         {
517                         if (p[i] != 0) continue;
518                         *(d++)=p[i+1];
519                         *(d++)=p[i+2];
520                         j+=2;
521                         }
522                 s2n(j,dd);
```

```
524                 /* COMPRESSION */
525                 *(d++)=1;
526                 *(d++)=0;

528 #if 0
529                 /* copy any remaining data with may be extensions */
530                 p = p+csl+sil+cl;
531                 while (p <  s->packet+s->packet_length)
532                         {
533                         *(d++)=*(p++);
534                         }
535 #endif

537                 i = (d-(unsigned char *)s->init_buf->data) - 4;
538                 l2n3((long)i, d_len);

540                 /* get the data reused from the init_buf */
541                 s->s3->tmp.reuse_message=1;
542                 s->s3->tmp.message_type=SSL3_MT_CLIENT_HELLO;
543                 s->s3->tmp.message_size=i;
544                 }

546         /* imaginary new state (for program structure): */
547         /* s->state = SSL23_SR_CLNT_HELLO_C */

549         if (type == 1)
550                 {
551 #ifdef OPENSSL_NO_SSL2
552                 SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_UNSUPPORTED_PROTOCOL);
553                 goto err;
554 #else
555                 /* we are talking sslv2 */
556                 /* we need to clean up the SSLv3/TLSv1 setup and put in the
557                  * sslv2 stuff. */

559                 if (s->s2 == NULL)
560                         {
561                         if (!ssl2_new(s))
562                                 goto err;
563                         }
564                 else
565                         ssl2_clear(s);

567                 if (s->s3 != NULL) ssl3_free(s);

569                 if (!BUF_MEM_grow_clean(s->init_buf,
570                         SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER))
571                         {
572                         goto err;
573                         }

575                 s->state=SSL2_ST_GET_CLIENT_HELLO_A;
576                 if (s->options & SSL_OP_NO_TLSv1 && s->options & SSL_OP_NO_SSLv3
577                         s->s2->ssl2_rollback=0;
578                 else
579                         /* reject SSL 2.0 session if client supports SSL 3.0 or
580                          * (SSL 3.0 draft/RFC 2246, App. E.2) */
581                         s->s2->ssl2_rollback=1;

583                 /* setup the n bytes we have read so we get them from
584                  * the sslv2 buffer */
585                 s->rstate=SSL_ST_READ_HEADER;
586                 s->packet_length=n;
587                 s->packet= &(s->s2->rbuf[0]);
588                 memcpy(s->packet,buf,n);
589                 s->s2->rbuf_left=n;
```



```
590                 s->s2->rbuf_offs=0;

592                 s->method=SSLv2_server_method();
593                 s->handshake_func=s->method->ssl_accept;
594 #endif
595                 }

597         if ((type == 2) || (type == 3))
598                 {
599                 /* we have SSLv3/TLSv1 (type 2: SSL2 style, type 3: SSL3/TLS sty

601                 if (!ssl_init_wbio_buffer(s,1)) goto err;

603                 /* we are in this state */
604                 s->state=SSL3_ST_SR_CLNT_HELLO_A;

606                 if (type == 3)
607                         {
608                         /* put the 'n' bytes we have read into the input buffer
609                          * for SSLv3 */
610                         s->rstate=SSL_ST_READ_HEADER;
611                         s->packet_length=n;
612                         if (s->s3->rbuf.buf == NULL)
613                                 if (!ssl3_setup_read_buffer(s))
614                                         goto err;

616                         s->packet= &(s->s3->rbuf.buf[0]);
617                         memcpy(s->packet,buf,n);
618                         s->s3->rbuf.left=n;
619                         s->s3->rbuf.offset=0;
620                         }
621                 else
622                         {
623                         s->packet_length=0;
624                         s->s3->rbuf.left=0;
625                         s->s3->rbuf.offset=0;
626                         }
627                 if (s->version == TLS1_2_VERSION)
628                         s->method = TLSv1_2_server_method();
629                 else if (s->version == TLS1_1_VERSION)
630                         s->method = TLSv1_1_server_method();
631                 else if (s->version == TLS1_VERSION)
632                         s->method = TLSv1_server_method();
633                 else
634                         s->method = SSLv3_server_method();
635 #if 0 /* ssl3_get_client_hello does this */
636                 s->client_version=(v[0]<<8)|v[1];
637 #endif
638                 s->handshake_func=s->method->ssl_accept;
639                 }

641         if ((type < 1) || (type > 3))
642                 {
643                 /* bad, very bad */
644                 SSLerr(SSL_F_SSL23_GET_CLIENT_HELLO,SSL_R_UNKNOWN_PROTOCOL);
645                 goto err;
646                 }
647         s->init_num=0;

649         if (buf != buf_space) OPENSSL_free(buf);
650         return(SSL_accept(s));
651 err:
652         if (buf != buf_space) OPENSSL_free(buf);
653         return(-1);
654         }
655 #endif /* ! codereview */
```

```
**********************************************************
   30825 Wed Aug 13 19:53:37 2014
new/usr/src/lib/openssl/libsunw_ssl/s2_clnt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s2_clnt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "ssl_locl.h"
#ifndef OPENSSL_NO_SSL2
#include <stdio.h>
#include <openssl/rand.h>
#include <openssl/buffer.h>
#include <openssl/objects.h>
#include <openssl/evp.h>

static const SSL_METHOD *ssl2_get_client_method(int ver);
static int get_server_finished(SSL *s);
static int get_server_verify(SSL *s);
static int get_server_hello(SSL *s);
static int client_hello(SSL *s);
static int client_master_key(SSL *s);
static int client_finished(SSL *s);
static int client_certificate(SSL *s);
```

```
128 static int ssl_rsa_public_encrypt(SESS_CERT *sc, int len, unsigned char *from,
129         unsigned char *to,int padding);
130 #define BREAK   break

132 static const SSL_METHOD *ssl2_get_client_method(int ver)
133         {
134         if (ver == SSL2_VERSION)
135                 return(SSLv2_client_method());
136         else
137                 return(NULL);
138         }

140 IMPLEMENT_ssl2_meth_func(SSLv2_client_method,
141                          ssl_undefined_function,
142                          ssl2_connect,
143                          ssl2_get_client_method)

145 int ssl2_connect(SSL *s)
146         {
147         unsigned long l=(unsigned long)time(NULL);
148         BUF_MEM *buf=NULL;
149         int ret= -1;
150         void (*cb)(const SSL *ssl,int type,int val)=NULL;
151         int new_state,state;

153         RAND_add(&l,sizeof(l),0);
154         ERR_clear_error();
155         clear_sys_error();

157         if (s->info_callback != NULL)
158                 cb=s->info_callback;
159         else if (s->ctx->info_callback != NULL)
160                 cb=s->ctx->info_callback;

162         /* init things to blank */
163         s->in_handshake++;
164         if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

166         for (;;)
167                 {
168                 state=s->state;

170                 switch (s->state)
171                         {
172                 case SSL_ST_BEFORE:
173                 case SSL_ST_CONNECT:
174                 case SSL_ST_BEFORE|SSL_ST_CONNECT:
175                 case SSL_ST_OK|SSL_ST_CONNECT:

177                         s->server=0;
178                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

180                         s->version=SSL2_VERSION;
181                         s->type=SSL_ST_CONNECT;

183                         buf=s->init_buf;
184                         if ((buf == NULL) && ((buf=BUF_MEM_new()) == NULL))
185                                 {
186                                 ret= -1;
187                                 goto end;
188                                 }
189                         if (!BUF_MEM_grow(buf,
190                                 SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER))
191                                 {
192                                 if (buf == s->init_buf)
193                                         buf=NULL;
```



```
194                                 ret= -1;
195                                 goto end;
196                                 }
197                         s->init_buf=buf;
198                         buf=NULL;
199                         s->init_num=0;
200                         s->state=SSL2_ST_SEND_CLIENT_HELLO_A;
201                         s->ctx->stats.sess_connect++;
202                         s->handshake_func=ssl2_connect;
203                         BREAK;

205                 case SSL2_ST_SEND_CLIENT_HELLO_A:
206                 case SSL2_ST_SEND_CLIENT_HELLO_B:
207                         s->shutdown=0;
208                         ret=client_hello(s);
209                         if (ret <= 0) goto end;
210                         s->init_num=0;
211                         s->state=SSL2_ST_GET_SERVER_HELLO_A;
212                         BREAK;

214                 case SSL2_ST_GET_SERVER_HELLO_A:
215                 case SSL2_ST_GET_SERVER_HELLO_B:
216                         ret=get_server_hello(s);
217                         if (ret <= 0) goto end;
218                         s->init_num=0;
219                         if (!s->hit) /* new session */
220                                 {
221                                 s->state=SSL2_ST_SEND_CLIENT_MASTER_KEY_A;
222                                 BREAK;
223                                 }
224                         else
225                                 {
226                                 s->state=SSL2_ST_CLIENT_START_ENCRYPTION;
227                                 break;
228                                 }

230                 case SSL2_ST_SEND_CLIENT_MASTER_KEY_A:
231                 case SSL2_ST_SEND_CLIENT_MASTER_KEY_B:
232                         ret=client_master_key(s);
233                         if (ret <= 0) goto end;
234                         s->init_num=0;
235                         s->state=SSL2_ST_CLIENT_START_ENCRYPTION;
236                         break;

238                 case SSL2_ST_CLIENT_START_ENCRYPTION:
239                         /* Ok, we now have all the stuff needed to
240                          * start encrypting, so lets fire it up :-) */
241                         if (!ssl2_enc_init(s,1))
242                                 {
243                                 ret= -1;
244                                 goto end;
245                                 }
246                         s->s2->clear_text=0;
247                         s->state=SSL2_ST_SEND_CLIENT_FINISHED_A;
248                         break;

250                 case SSL2_ST_SEND_CLIENT_FINISHED_A:
251                 case SSL2_ST_SEND_CLIENT_FINISHED_B:
252                         ret=client_finished(s);
253                         if (ret <= 0) goto end;
254                         s->init_num=0;
255                         s->state=SSL2_ST_GET_SERVER_VERIFY_A;
256                         break;

258                 case SSL2_ST_GET_SERVER_VERIFY_A:
259                 case SSL2_ST_GET_SERVER_VERIFY_B:
```

```
260                         ret=get_server_verify(s);
261                         if (ret <= 0) goto end;
262                         s->init_num=0;
263                         s->state=SSL2_ST_GET_SERVER_FINISHED_A;
264                         break;

266                 case SSL2_ST_GET_SERVER_FINISHED_A:
267                 case SSL2_ST_GET_SERVER_FINISHED_B:
268                         ret=get_server_finished(s);
269                         if (ret <= 0) goto end;
270                         break;

272                 case SSL2_ST_SEND_CLIENT_CERTIFICATE_A:
273                 case SSL2_ST_SEND_CLIENT_CERTIFICATE_B:
274                 case SSL2_ST_SEND_CLIENT_CERTIFICATE_C:
275                 case SSL2_ST_SEND_CLIENT_CERTIFICATE_D:
276                 case SSL2_ST_X509_GET_CLIENT_CERTIFICATE:
277                         ret=client_certificate(s);
278                         if (ret <= 0) goto end;
279                         s->init_num=0;
280                         s->state=SSL2_ST_GET_SERVER_FINISHED_A;
281                         break;

283                 case SSL_ST_OK:
284                         if (s->init_buf != NULL)
285                                 {
286                                 BUF_MEM_free(s->init_buf);
287                                 s->init_buf=NULL;
288                                 }
289                         s->init_num=0;
290                 /*      ERR_clear_error();*/

292                         /* If we want to cache session-ids in the client
293                          * and we successfully add the session-id to the
294                          * cache, and there is a callback, then pass it out.
295                          * 26/11/96 - eay - only add if not a re-used session.
296                          */

298                         ssl_update_cache(s,SSL_SESS_CACHE_CLIENT);
299                         if (s->hit) s->ctx->stats.sess_hit++;

301                         ret=1;
302                         /* s->server=0; */
303                         s->ctx->stats.sess_connect_good++;

305                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_DONE,1);

307                         goto end;
308                         /* break; */
309                 default:
310                         SSLerr(SSL_F_SSL2_CONNECT,SSL_R_UNKNOWN_STATE);
311                         return(-1);
312                         /* break; */
313                         }

315                 if ((cb != NULL) && (s->state != state))
316                         {
317                         new_state=s->state;
318                         s->state=state;
319                         cb(s,SSL_CB_CONNECT_LOOP,1);
320                         s->state=new_state;
321                         }
322                 }
323 end:
324         s->in_handshake--;
325         if (buf != NULL)
```



```
326                 BUF_MEM_free(buf);
327         if (cb != NULL)
328                 cb(s,SSL_CB_CONNECT_EXIT,ret);
329         return(ret);
330         }

332 static int get_server_hello(SSL *s)
333         {
334         unsigned char *buf;
335         unsigned char *p;
336         int i,j;
337         unsigned long len;
338         STACK_OF(SSL_CIPHER) *sk=NULL,*cl, *prio, *allow;

340         buf=(unsigned char *)s->init_buf->data;
341         p=buf;
342         if (s->state == SSL2_ST_GET_SERVER_HELLO_A)
343                 {
344                 i=ssl2_read(s,(char *)&(buf[s->init_num]),11-s->init_num);
345                 if (i < (11-s->init_num))
346                         return(ssl2_part_read(s,SSL_F_GET_SERVER_HELLO,i));
347                 s->init_num = 11;

349                 if (*(p++) != SSL2_MT_SERVER_HELLO)
350                         {
351                         if (p[-1] != SSL2_MT_ERROR)
352                                 {
353                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
354                                 SSLerr(SSL_F_GET_SERVER_HELLO,
355                                         SSL_R_READ_WRONG_PACKET_TYPE);
356                                 }
357                         else
358                                 SSLerr(SSL_F_GET_SERVER_HELLO,
359                                         SSL_R_PEER_ERROR);
360                         return(-1);
361                         }
362 #if 0
363                 s->hit=(*(p++))?1:0;
364                 /* Some [PPC?] compilers fail to increment p in above
365                    statement, e.g. one provided with Rhapsody 5.5, but
366                    most recent example XL C 11.1 for AIX, even without
367                    optimization flag... */
368 #else
369                 s->hit=(*p)?1:0; p++;
370 #endif
371                 s->s2->tmp.cert_type= *(p++);
372                 n2s(p,i);
373                 if (i < s->version) s->version=i;
374                 n2s(p,i); s->s2->tmp.cert_length=i;
375                 n2s(p,i); s->s2->tmp.csl=i;
376                 n2s(p,i); s->s2->tmp.conn_id_length=i;
377                 s->state=SSL2_ST_GET_SERVER_HELLO_B;
378                 }

380         /* SSL2_ST_GET_SERVER_HELLO_B */
381         len = 11 + (unsigned long)s->s2->tmp.cert_length + (unsigned long)s->s2-
382         if (len > SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER)
383                 {
384                 SSLerr(SSL_F_GET_SERVER_HELLO,SSL_R_MESSAGE_TOO_LONG);
385                 return -1;
386                 }
387         j = (int)len - s->init_num;
388         i = ssl2_read(s,(char *)&(buf[s->init_num]),j);
389         if (i != j) return(ssl2_part_read(s,SSL_F_GET_SERVER_HELLO,i));
390         if (s->msg_callback)
391                 s->msg_callback(0, s->version, 0, buf, (size_t)len, s, s->msg_ca
```

```
393         /* things are looking good */

395         p = buf + 11;
396         if (s->hit)
397                 {
398                 if (s->s2->tmp.cert_length != 0)
399                         {
400                         SSLerr(SSL_F_GET_SERVER_HELLO,SSL_R_REUSE_CERT_LENGTH_NO
401                         return(-1);
402                         }
403                 if (s->s2->tmp.cert_type != 0)
404                         {
405                         if (!(s->options &
406                                 SSL_OP_SSLREF2_REUSE_CERT_TYPE_BUG))
407                                 {
408                                 SSLerr(SSL_F_GET_SERVER_HELLO,SSL_R_REUSE_CERT_T
409                                 return(-1);
410                                 }
411                         }
412                 if (s->s2->tmp.csl != 0)
413                         {
414                         SSLerr(SSL_F_GET_SERVER_HELLO,SSL_R_REUSE_CIPHER_LIST_NO
415                         return(-1);
416                         }
417                 }
418         else
419                 {
420 #ifdef undef
421                 /* very bad */
422                 memset(s->session->session_id,0,
423                         SSL_MAX_SSL_SESSION_ID_LENGTH_IN_BYTES);
424                 s->session->session_id_length=0;
425                 */
426 #endif

428                 /* we need to do this in case we were trying to reuse a
429                  * client session but others are already reusing it.
430                  * If this was a new 'blank' session ID, the session-id
431                  * length will still be 0 */
432                 if (s->session->session_id_length > 0)
433                         {
434                         if (!ssl_get_new_session(s,0))
435                                 {
436                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
437                                 return(-1);
438                                 }
439                         }

441                 if (ssl2_set_certificate(s,s->s2->tmp.cert_type,
442                         s->s2->tmp.cert_length,p) <= 0)
443                         {
444                         ssl2_return_error(s,SSL2_PE_BAD_CERTIFICATE);
445                         return(-1);
446                         }
447                 p+=s->s2->tmp.cert_length;

449                 if (s->s2->tmp.csl == 0)
450                         {
451                         ssl2_return_error(s,SSL2_PE_NO_CIPHER);
452                         SSLerr(SSL_F_GET_SERVER_HELLO,SSL_R_NO_CIPHER_LIST);
453                         return(-1);
454                         }

456                 /* We have just received a list of ciphers back from the
457                  * server.  We need to get the ones that match, then select
```



```
458                  * the one we want the most :-). */

460                 /* load the ciphers */
461                 sk=ssl_bytes_to_cipher_list(s,p,s->s2->tmp.csl,
462                                             &s->session->ciphers);
463                 p+=s->s2->tmp.csl;
464                 if (sk == NULL)
465                         {
466                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
467                         SSLerr(SSL_F_GET_SERVER_HELLO,ERR_R_MALLOC_FAILURE);
468                         return(-1);
469                         }

471                 (void)sk_SSL_CIPHER_set_cmp_func(sk,ssl_cipher_ptr_id_cmp);

473                 /* get the array of ciphers we will accept */
474                 cl=SSL_get_ciphers(s);
475                 (void)sk_SSL_CIPHER_set_cmp_func(cl,ssl_cipher_ptr_id_cmp);

477                 /*
478                  * If server preference flag set, choose the first
479                  * (highest priority) cipher the server sends, otherwise
480                  * client preference has priority.
481                  */
482                 if (s->options & SSL_OP_CIPHER_SERVER_PREFERENCE)
483                     {
484                     prio = sk;
485                     allow = cl;
486                     }
487                 else
488                     {
489                     prio = cl;
490                     allow = sk;
491                     }
492                 /* In theory we could have ciphers sent back that we
493                  * don't want to use but that does not matter since we
494                  * will check against the list we originally sent and
495                  * for performance reasons we should not bother to match
496                  * the two lists up just to check. */
497                 for (i=0; i<sk_SSL_CIPHER_num(prio); i++)
498                         {
499                         if (sk_SSL_CIPHER_find(allow,
500                                                sk_SSL_CIPHER_value(prio,i)) >= 0)
501                                 break;
502                         }

504                 if (i >= sk_SSL_CIPHER_num(prio))
505                         {
506                         ssl2_return_error(s,SSL2_PE_NO_CIPHER);
507                         SSLerr(SSL_F_GET_SERVER_HELLO,SSL_R_NO_CIPHER_MATCH);
508                         return(-1);
509                         }
510                 s->session->cipher=sk_SSL_CIPHER_value(prio,i);


513                 if (s->session->peer != NULL) /* can't happen*/
514                         {
515                         ssl2_return_error(s, SSL2_PE_UNDEFINED_ERROR);
516                         SSLerr(SSL_F_GET_SERVER_HELLO, ERR_R_INTERNAL_ERROR);
517                         return(-1);
518                         }

520                 s->session->peer = s->session->sess_cert->peer_key->x509;
521                 /* peer_key->x509 has been set by ssl2_set_certificate. */
522                 CRYPTO_add(&s->session->peer->references, 1, CRYPTO_LOCK_X509);
523                 }
```

```
525         if (s->session->sess_cert == NULL
526       || s->session->peer != s->session->sess_cert->peer_key->x509)
527                 /* can't happen */
528                 {
529                 ssl2_return_error(s, SSL2_PE_UNDEFINED_ERROR);
530                 SSLerr(SSL_F_GET_SERVER_HELLO, ERR_R_INTERNAL_ERROR);
531                 return(-1);
532                 }

534         s->s2->conn_id_length=s->s2->tmp.conn_id_length;
535         if (s->s2->conn_id_length > sizeof s->s2->conn_id)
536                 {
537                 ssl2_return_error(s, SSL2_PE_UNDEFINED_ERROR);
538                 SSLerr(SSL_F_GET_SERVER_HELLO, SSL_R_SSL2_CONNECTION_ID_TOO_LONG
539                 return -1;
540                 }
541         memcpy(s->s2->conn_id,p,s->s2->tmp.conn_id_length);
542         return(1);
543         }

545 static int client_hello(SSL *s)
546         {
547         unsigned char *buf;
548         unsigned char *p,*d;
549 /*      CIPHER **cipher;*/
550         int i,n,j;

552         buf=(unsigned char *)s->init_buf->data;
553         if (s->state == SSL2_ST_SEND_CLIENT_HELLO_A)
554                 {
555                 if ((s->session == NULL) ||
556                         (s->session->ssl_version != s->version))
557                         {
558                         if (!ssl_get_new_session(s,0))
559                                 {
560                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
561                                 return(-1);
562                                 }
563                         }
564                 /* else use the pre-loaded session */

566                 p=buf;                                  /* header */
567                 d=p+9;                                  /* data section */
568                 *(p++)=SSL2_MT_CLIENT_HELLO;            /* type */
569                 s2n(SSL2_VERSION,p);                    /* version */
570                 n=j=0;

572                 n=ssl_cipher_list_to_bytes(s,SSL_get_ciphers(s),d,0);
573                 d+=n;

575                 if (n == 0)
576                         {
577                         SSLerr(SSL_F_CLIENT_HELLO,SSL_R_NO_CIPHERS_AVAILABLE);
578                         return(-1);
579                         }

581                 s2n(n,p);                       /* cipher spec num bytes */

583                 if ((s->session->session_id_length > 0) &&
584                         (s->session->session_id_length <=
585                         SSL2_MAX_SSL_SESSION_ID_LENGTH))
586                         {
587                         i=s->session->session_id_length;
588                         s2n(i,p);               /* session id length */
589                         memcpy(d,s->session->session_id,(unsigned int)i);
```



```
590                         d+=i;
591                         }
592                 else
593                         {
594                         s2n(0,p);
595                         }

597                 s->s2->challenge_length=SSL2_CHALLENGE_LENGTH;
598                 s2n(SSL2_CHALLENGE_LENGTH,p);           /* challenge length */
599                 /*challenge id data*/
600                 if (RAND_pseudo_bytes(s->s2->challenge,SSL2_CHALLENGE_LENGTH) <=
601                         return -1;
602                 memcpy(d,s->s2->challenge,SSL2_CHALLENGE_LENGTH);
603                 d+=SSL2_CHALLENGE_LENGTH;

605                 s->state=SSL2_ST_SEND_CLIENT_HELLO_B;
606                 s->init_num=d-buf;
607                 s->init_off=0;
608                 }
609         /* SSL2_ST_SEND_CLIENT_HELLO_B */
610         return(ssl2_do_write(s));
611         }

613 static int client_master_key(SSL *s)
614         {
615         unsigned char *buf;
616         unsigned char *p,*d;
617         int clear,enc,karg,i;
618         SSL_SESSION *sess;
619         const EVP_CIPHER *c;
620         const EVP_MD *md;

622         buf=(unsigned char *)s->init_buf->data;
623         if (s->state == SSL2_ST_SEND_CLIENT_MASTER_KEY_A)
624                 {

626                 if (!ssl_cipher_get_evp(s->session,&c,&md,NULL,NULL,NULL))
627                         {
628                         ssl2_return_error(s,SSL2_PE_NO_CIPHER);
629                         SSLerr(SSL_F_CLIENT_MASTER_KEY,SSL_R_PROBLEMS_MAPPING_CI
630                         return(-1);
631                         }
632                 sess=s->session;
633                 p=buf;
634                 d=p+10;
635                 *(p++)=SSL2_MT_CLIENT_MASTER_KEY;/* type */

637                 i=ssl_put_cipher_by_char(s,sess->cipher,p);
638                 p+=i;

640                 /* make key_arg data */
641                 i=EVP_CIPHER_iv_length(c);
642                 sess->key_arg_length=i;
643                 if (i > SSL_MAX_KEY_ARG_LENGTH)
644                         {
645                         ssl2_return_error(s, SSL2_PE_UNDEFINED_ERROR);
646                         SSLerr(SSL_F_CLIENT_MASTER_KEY, ERR_R_INTERNAL_ERROR);
647                         return -1;
648                         }
649                 if (i > 0)
650                         if (RAND_pseudo_bytes(sess->key_arg,i) <= 0)
651                                 return -1;

653                 /* make a master key */
654                 i=EVP_CIPHER_key_length(c);
655                 sess->master_key_length=i;
```

```
656                 if (i > 0)
657                         {
658                         if (i > (int)sizeof(sess->master_key))
659                                 {
660                                 ssl2_return_error(s, SSL2_PE_UNDEFINED_ERROR);
661                                 SSLerr(SSL_F_CLIENT_MASTER_KEY, ERR_R_INTERNAL_E
662                                 return -1;
663                                 }
664                         if (RAND_bytes(sess->master_key,i) <= 0)
665                                 {
666                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
667                                 return(-1);
668                                 }
669                         }

671                 if (sess->cipher->algorithm2 & SSL2_CF_8_BYTE_ENC)
672                         enc=8;
673                 else if (SSL_C_IS_EXPORT(sess->cipher))
674                         enc=5;
675                 else
676                         enc=i;

678                 if ((int)i < enc)
679                         {
680                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
681                         SSLerr(SSL_F_CLIENT_MASTER_KEY,SSL_R_CIPHER_TABLE_SRC_ER
682                         return(-1);
683                         }
684                 clear=i-enc;
685                 s2n(clear,p);
686                 memcpy(d,sess->master_key,(unsigned int)clear);
687                 d+=clear;

689                 enc=ssl_rsa_public_encrypt(sess->sess_cert,enc,
690                         &(sess->master_key[clear]),d,
691                         (s->s2->ssl2_rollback)?RSA_SSLV23_PADDING:RSA_PKCS1_PADD
692                 if (enc <= 0)
693                         {
694                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
695                         SSLerr(SSL_F_CLIENT_MASTER_KEY,SSL_R_PUBLIC_KEY_ENCRYPT_
696                         return(-1);
697                         }
698 #ifdef PKCS1_CHECK
699                 if (s->options & SSL_OP_PKCS1_CHECK_1) d[1]++;
700                 if (s->options & SSL_OP_PKCS1_CHECK_2)
701                         sess->master_key[clear]++;
702 #endif
703                 s2n(enc,p);
704                 d+=enc;
705                 karg=sess->key_arg_length;
706                 s2n(karg,p); /* key arg size */
707                 if (karg > (int)sizeof(sess->key_arg))
708                         {
709                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
710                         SSLerr(SSL_F_CLIENT_MASTER_KEY, ERR_R_INTERNAL_ERROR);
711                         return -1;
712                         }
713                 memcpy(d,sess->key_arg,(unsigned int)karg);
714                 d+=karg;

716                 s->state=SSL2_ST_SEND_CLIENT_MASTER_KEY_B;
717                 s->init_num=d-buf;
718                 s->init_off=0;
719                 }

721         /* SSL2_ST_SEND_CLIENT_MASTER_KEY_B */
```



```
722         return(ssl2_do_write(s));
723         }

725 static int client_finished(SSL *s)
726         {
727         unsigned char *p;

729         if (s->state == SSL2_ST_SEND_CLIENT_FINISHED_A)
730                 {
731                 p=(unsigned char *)s->init_buf->data;
732                 *(p++)=SSL2_MT_CLIENT_FINISHED;
733                 if (s->s2->conn_id_length > sizeof s->s2->conn_id)
734                         {
735                         SSLerr(SSL_F_CLIENT_FINISHED, ERR_R_INTERNAL_ERROR);
736                         return -1;
737                         }
738                 memcpy(p,s->s2->conn_id,(unsigned int)s->s2->conn_id_length);

740                 s->state=SSL2_ST_SEND_CLIENT_FINISHED_B;
741                 s->init_num=s->s2->conn_id_length+1;
742                 s->init_off=0;
743                 }
744         return(ssl2_do_write(s));
745         }

747 /* read the data and then respond */
748 static int client_certificate(SSL *s)
749         {
750         unsigned char *buf;
751         unsigned char *p,*d;
752         int i;
753         unsigned int n;
754         int cert_ch_len;
755         unsigned char *cert_ch;

757         buf=(unsigned char *)s->init_buf->data;

759         /* We have a cert associated with the SSL, so attach it to
760          * the session if it does not have one */

762         if (s->state == SSL2_ST_SEND_CLIENT_CERTIFICATE_A)
763                 {
764                 i=ssl2_read(s,(char *)&(buf[s->init_num]),
765                         SSL2_MAX_CERT_CHALLENGE_LENGTH+2-s->init_num);
766                 if (i<(SSL2_MIN_CERT_CHALLENGE_LENGTH+2-s->init_num))
767                         return(ssl2_part_read(s,SSL_F_CLIENT_CERTIFICATE,i));
768                 s->init_num += i;
769                 if (s->msg_callback)
770                         s->msg_callback(0, s->version, 0, buf, (size_t)s->init_n

772                 /* type=buf[0]; */
773                 /* type eq x509 */
774                 if (buf[1] != SSL2_AT_MD5_WITH_RSA_ENCRYPTION)
775                         {
776                         ssl2_return_error(s,SSL2_PE_UNSUPPORTED_CERTIFICATE_TYPE
777                         SSLerr(SSL_F_CLIENT_CERTIFICATE,SSL_R_BAD_AUTHENTICATION
778                         return(-1);
779                         }

781                 if ((s->cert == NULL) ||
782                         (s->cert->key->x509 == NULL) ||
783                         (s->cert->key->privatekey == NULL))
784                         {
785                         s->state=SSL2_ST_X509_GET_CLIENT_CERTIFICATE;
786                         }
787                 else
```

```
788                         s->state=SSL2_ST_SEND_CLIENT_CERTIFICATE_C;
789                 }
790
791         cert_ch = buf + 2;
792         cert_ch_len = s->init_num - 2;
793
794         if (s->state == SSL2_ST_X509_GET_CLIENT_CERTIFICATE)
795                 {
796                 X509 *x509=NULL;
797                 EVP_PKEY *pkey=NULL;
798
799                 /* If we get an error we need to
800                  * ssl->rwstate=SSL_X509_LOOKUP;
801                  * return(error);
802                  * We should then be retried when things are ok and we
803                  * can get a cert or not */
804
805                 i=0;
806                 if (s->ctx->client_cert_cb != NULL)
807                         {
808                         i=s->ctx->client_cert_cb(s,&(x509),&(pkey));
809                         }
810
811                 if (i < 0)
812                         {
813                         s->rwstate=SSL_X509_LOOKUP;
814                         return(-1);
815                         }
816                 s->rwstate=SSL_NOTHING;
817
818                 if ((i == 1) && (pkey != NULL) && (x509 != NULL))
819                         {
820                         s->state=SSL2_ST_SEND_CLIENT_CERTIFICATE_C;
821                         if (    !SSL_use_certificate(s,x509) ||
822                                 !SSL_use_PrivateKey(s,pkey))
823                                 {
824                                 i=0;
825                                 }
826                         X509_free(x509);
827                         EVP_PKEY_free(pkey);
828                         }
829                 else if (i == 1)
830                         {
831                         if (x509 != NULL) X509_free(x509);
832                         if (pkey != NULL) EVP_PKEY_free(pkey);
833                         SSLerr(SSL_F_CLIENT_CERTIFICATE,SSL_R_BAD_DATA_RETURNED_
834                         i=0;
835                         }
836
837                 if (i == 0)
838                         {
839                         /* We have no client certificate to respond with
840                          * so send the correct error message back */
841                         s->state=SSL2_ST_SEND_CLIENT_CERTIFICATE_B;
842                         p=buf;
843                         *(p++)=SSL2_MT_ERROR;
844                         s2n(SSL2_PE_NO_CERTIFICATE,p);
845                         s->init_off=0;
846                         s->init_num=3;
847                         /* Write is done at the end */
848                         }
849                 }
850
851         if (s->state == SSL2_ST_SEND_CLIENT_CERTIFICATE_B)
852                 {
853                 return(ssl2_do_write(s));
```



```
854                 }
855
856         if (s->state == SSL2_ST_SEND_CLIENT_CERTIFICATE_C)
857                 {
858                 EVP_MD_CTX ctx;
859
860                 /* ok, now we calculate the checksum
861                  * do it first so we can reuse buf :-) */
862                 p=buf;
863                 EVP_MD_CTX_init(&ctx);
864                 EVP_SignInit_ex(&ctx,s->ctx->rsa_md5, NULL);
865                 EVP_SignUpdate(&ctx,s->s2->key_material,
866                                 s->s2->key_material_length);
867                 EVP_SignUpdate(&ctx,cert_ch,(unsigned int)cert_ch_len);
868                 i=i2d_X509(s->session->sess_cert->peer_key->x509,&p);
869                 /* Don't update the signature if it fails - FIXME: probably shou
870                 if(i > 0)
871                         EVP_SignUpdate(&ctx,buf,(unsigned int)i);
872
873                 p=buf;
874                 d=p+6;
875                 *(p++)=SSL2_MT_CLIENT_CERTIFICATE;
876                 *(p++)=SSL2_CT_X509_CERTIFICATE;
877                 n=i2d_X509(s->cert->key->x509,&d);
878                 s2n(n,p);
879
880                 if (!EVP_SignFinal(&ctx,d,&n,s->cert->key->privatekey))
881                         {
882                         /* this is not good.  If things have failed it
883                          * means there so something wrong with the key.
884                          * We will continue with a 0 length signature
885                          */
886                         }
887                 EVP_MD_CTX_cleanup(&ctx);
888                 s2n(n,p);
889                 d+=n;
890
891                 s->state=SSL2_ST_SEND_CLIENT_CERTIFICATE_D;
892                 s->init_num=d-buf;
893                 s->init_off=0;
894                 }
895         /* if (s->state == SSL2_ST_SEND_CLIENT_CERTIFICATE_D) */
896         return(ssl2_do_write(s));
897         }
898
899 static int get_server_verify(SSL *s)
900         {
901         unsigned char *p;
902         int i, n, len;
903
904         p=(unsigned char *)s->init_buf->data;
905         if (s->state == SSL2_ST_GET_SERVER_VERIFY_A)
906                 {
907                 i=ssl2_read(s,(char *)&(p[s->init_num]),1-s->init_num);
908                 if (i < (1-s->init_num))
909                         return(ssl2_part_read(s,SSL_F_GET_SERVER_VERIFY,i));
910                 s->init_num += i;
911
912                 s->state= SSL2_ST_GET_SERVER_VERIFY_B;
913                 if (*p != SSL2_MT_SERVER_VERIFY)
914                         {
915                         if (p[0] != SSL2_MT_ERROR)
916                                 {
917                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
918                                 SSLerr(SSL_F_GET_SERVER_VERIFY,
919                                         SSL_R_READ_WRONG_PACKET_TYPE);
```

```
920                             }
921                         else
922                             {
923                             SSLerr(SSL_F_GET_SERVER_VERIFY,SSL_R_PEER_ERROR)
924                             /* try to read the error message */
925                             i=ssl2_read(s,(char *)&(p[s->init_num]),3-s->ini
926                             return ssl2_part_read(s,SSL_F_GET_SERVER_VERIFY,
927                             }
928                         return(-1);
929                         }
930                 }

932         p=(unsigned char *)s->init_buf->data;
933         len = 1 + s->s2->challenge_length;
934         n =  len - s->init_num;
935         i = ssl2_read(s,(char *)&(p[s->init_num]),n);
936         if (i < n)
937                 return(ssl2_part_read(s,SSL_F_GET_SERVER_VERIFY,i));
938         if (s->msg_callback)
939                 s->msg_callback(0, s->version, 0, p, len, s, s->msg_callback_arg
940         p += 1;

942         if (CRYPTO_memcmp(p,s->s2->challenge,s->s2->challenge_length) != 0)
943                 {
944                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
945                 SSLerr(SSL_F_GET_SERVER_VERIFY,SSL_R_CHALLENGE_IS_DIFFERENT);
946                 return(-1);
947                 }
948         return(1);
949         }

951 static int get_server_finished(SSL *s)
952         {
953         unsigned char *buf;
954         unsigned char *p;
955         int i, n, len;

957         buf=(unsigned char *)s->init_buf->data;
958         p=buf;
959         if (s->state == SSL2_ST_GET_SERVER_FINISHED_A)
960                 {
961                 i=ssl2_read(s,(char *)&(buf[s->init_num]),1-s->init_num);
962                 if (i < (1-s->init_num))
963                         return(ssl2_part_read(s,SSL_F_GET_SERVER_FINISHED,i));
964                 s->init_num += i;

966                 if (*p == SSL2_MT_REQUEST_CERTIFICATE)
967                         {
968                         s->state=SSL2_ST_SEND_CLIENT_CERTIFICATE_A;
969                         return(1);
970                         }
971                 else if (*p != SSL2_MT_SERVER_FINISHED)
972                         {
973                         if (p[0] != SSL2_MT_ERROR)
974                                 {
975                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
976                                 SSLerr(SSL_F_GET_SERVER_FINISHED,SSL_R_READ_WRON
977                                 }
978                         else
979                                 {
980                                 SSLerr(SSL_F_GET_SERVER_FINISHED,SSL_R_PEER_ERRO
981                                 /* try to read the error message */
982                                 i=ssl2_read(s,(char *)&(p[s->init_num]),3-s->ini
983                                 return ssl2_part_read(s,SSL_F_GET_SERVER_VERIFY,
984                                 }
985                         return(-1);
```



```
986                         }
987                 s->state=SSL2_ST_GET_SERVER_FINISHED_B;
988                 }

990         len = 1 + SSL2_SSL_SESSION_ID_LENGTH;
991         n = len - s->init_num;
992         i = ssl2_read(s,(char *)&(buf[s->init_num]), n);
993         if (i < n) /* XXX could be shorter than SSL2_SSL_SESSION_ID_LENGTH, that
994                 return(ssl2_part_read(s,SSL_F_GET_SERVER_FINISHED,i));
995         s->init_num += i;
996         if (s->msg_callback)
997                 s->msg_callback(0, s->version, 0, buf, (size_t)s->init_num, s, s

999         if (!s->hit) /* new session */
1000                {
1001                /* new session-id */
1002                /* Make sure we were not trying to re-use an old SSL_SESSION
1003                 * or bad things can happen */
1004                /* ZZZZZZZZZZZZZ */
1005                s->session->session_id_length=SSL2_SSL_SESSION_ID_LENGTH;
1006                memcpy(s->session->session_id,p+1,SSL2_SSL_SESSION_ID_LENGTH);
1007                }
1008        else
1009                {
1010                if (!(s->options & SSL_OP_MICROSOFT_SESS_ID_BUG))
1011                        {
1012                        if ((s->session->session_id_length > sizeof s->session->
1013                            || (0 != memcmp(buf + 1, s->session->session_id,
1014                                            (unsigned int)s->session->session_id
1015                                {
1016                                ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
1017                                SSLerr(SSL_F_GET_SERVER_FINISHED,SSL_R_SSL_SESSI
1018                                return(-1);
1019                                }
1020                        }
1021                }
1022        s->state = SSL_ST_OK;
1023        return(1);
1024        }

1026 /* loads in the certificate from the server */
1027 int ssl2_set_certificate(SSL *s, int type, int len, const unsigned char *data)
1028        {
1029        STACK_OF(X509) *sk=NULL;
1030        EVP_PKEY *pkey=NULL;
1031        SESS_CERT *sc=NULL;
1032        int i;
1033        X509 *x509=NULL;
1034        int ret=0;

1036        x509=d2i_X509(NULL,&data,(long)len);
1037        if (x509 == NULL)
1038                {
1039                SSLerr(SSL_F_SSL2_SET_CERTIFICATE,ERR_R_X509_LIB);
1040                goto err;
1041                }

1043        if ((sk=sk_X509_new_null()) == NULL || !sk_X509_push(sk,x509))
1044                {
1045                SSLerr(SSL_F_SSL2_SET_CERTIFICATE,ERR_R_MALLOC_FAILURE);
1046                goto err;
1047                }

1049        i=ssl_verify_cert_chain(s,sk);

1051        if ((s->verify_mode != SSL_VERIFY_NONE) && (i <= 0))
```

```
1052                 {
1053                 SSLerr(SSL_F_SSL2_SET_CERTIFICATE,SSL_R_CERTIFICATE_VERIFY_FAILE
1054                 goto err;
1055                 }
1056         ERR_clear_error(); /* but we keep s->verify_result */
1057         s->session->verify_result = s->verify_result;
1059         /* server's cert for this session */
1060         sc=ssl_sess_cert_new();
1061         if (sc == NULL)
1062                 {
1063                 ret= -1;
1064                 goto err;
1065                 }
1066         if (s->session->sess_cert) ssl_sess_cert_free(s->session->sess_cert);
1067         s->session->sess_cert=sc;
1069         sc->peer_pkeys[SSL_PKEY_RSA_ENC].x509=x509;
1070         sc->peer_key= &(sc->peer_pkeys[SSL_PKEY_RSA_ENC]);
1072         pkey=X509_get_pubkey(x509);
1073         x509=NULL;
1074         if (pkey == NULL)
1075                 {
1076                 SSLerr(SSL_F_SSL2_SET_CERTIFICATE,SSL_R_UNABLE_TO_EXTRACT_PUBLIC
1077                 goto err;
1078                 }
1079         if (pkey->type != EVP_PKEY_RSA)
1080                 {
1081                 SSLerr(SSL_F_SSL2_SET_CERTIFICATE,SSL_R_PUBLIC_KEY_NOT_RSA);
1082                 goto err;
1083                 }
1085         if (!ssl_set_peer_cert_type(sc,SSL2_CT_X509_CERTIFICATE))
1086                 goto err;
1087         ret=1;
1088 err:
1089         sk_X509_free(sk);
1090         X509_free(x509);
1091         EVP_PKEY_free(pkey);
1092         return(ret);
1093         }
1095 static int ssl_rsa_public_encrypt(SESS_CERT *sc, int len, unsigned char *from,
1096             unsigned char *to, int padding)
1097         {
1098         EVP_PKEY *pkey=NULL;
1099         int i= -1;
1101         if ((sc == NULL) || (sc->peer_key->x509 == NULL) ||
1102                 ((pkey=X509_get_pubkey(sc->peer_key->x509)) == NULL))
1103                 {
1104                 SSLerr(SSL_F_SSL_RSA_PUBLIC_ENCRYPT,SSL_R_NO_PUBLICKEY);
1105                 return(-1);
1106                 }
1107         if (pkey->type != EVP_PKEY_RSA)
1108                 {
1109                 SSLerr(SSL_F_SSL_RSA_PUBLIC_ENCRYPT,SSL_R_PUBLIC_KEY_IS_NOT_RSA)
1110                 goto end;
1111                 }
1113         /* we have the public key */
1114         i=RSA_public_encrypt(len,from,to,pkey->pkey.rsa,padding);
1115         if (i < 0)
1116                 SSLerr(SSL_F_SSL_RSA_PUBLIC_ENCRYPT,ERR_R_RSA_LIB);
1117 end:
```



```
1118         EVP_PKEY_free(pkey);
1119         return(i);
1120         }
1121 #else /* !OPENSSL_NO_SSL2 */
1123 # if PEDANTIC
1124 static void *dummy=&dummy;
1125 # endif
1127 #endif
1128 #endif /* ! codereview */
```

```
**********************************************************
    6254 Wed Aug 13 19:53:37 2014
new/usr/src/lib/openssl/libsunw_ssl/s2_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s2_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "ssl_locl.h"
#ifndef OPENSSL_NO_SSL2
#include <stdio.h>
```



```
int ssl2_enc_init(SSL *s, int client)
        {
        /* Max number of bytes needed */
        EVP_CIPHER_CTX *rs,*ws;
        const EVP_CIPHER *c;
        const EVP_MD *md;
        int num;

        if (!ssl_cipher_get_evp(s->session,&c,&md,NULL,NULL,NULL))
                {
                ssl2_return_error(s,SSL2_PE_NO_CIPHER);
                SSLerr(SSL_F_SSL2_ENC_INIT,SSL_R_PROBLEMS_MAPPING_CIPHER_FUNCTIO
                return(0);
                }
        ssl_replace_hash(&s->read_hash,md);
        ssl_replace_hash(&s->write_hash,md);

        if ((s->enc_read_ctx == NULL) &&
                ((s->enc_read_ctx=(EVP_CIPHER_CTX *)
                OPENSSL_malloc(sizeof(EVP_CIPHER_CTX))) == NULL))
                goto err;

        /* make sure it's intialized in case the malloc for enc_write_ctx fails
         * and we exit with an error */
        rs= s->enc_read_ctx;
        EVP_CIPHER_CTX_init(rs);

        if ((s->enc_write_ctx == NULL) &&
                ((s->enc_write_ctx=(EVP_CIPHER_CTX *)
                OPENSSL_malloc(sizeof(EVP_CIPHER_CTX))) == NULL))
                goto err;

        ws= s->enc_write_ctx;
        EVP_CIPHER_CTX_init(ws);

        num=c->key_len;
        s->s2->key_material_length=num*2;
        OPENSSL_assert(s->s2->key_material_length <= sizeof s->s2->key_material)

        if (ssl2_generate_key_material(s) <= 0)
                return 0;

        OPENSSL_assert(c->iv_len <= (int)sizeof(s->session->key_arg));
        EVP_EncryptInit_ex(ws,c,NULL,&(s->s2->key_material[(client)?num:0]),
                s->session->key_arg);
        EVP_DecryptInit_ex(rs,c,NULL,&(s->s2->key_material[(client)?0:num]),
                s->session->key_arg);
        s->s2->read_key=  &(s->s2->key_material[(client)?0:num]);
        s->s2->write_key= &(s->s2->key_material[(client)?num:0]);
        return(1);
err:
        SSLerr(SSL_F_SSL2_ENC_INIT,ERR_R_MALLOC_FAILURE);
        return(0);
        }

/* read/writes from s->s2->mac_data using length for encrypt and
 * decrypt.  It sets s->s2->padding and s->[rw]length
 * if we are encrypting */
void ssl2_enc(SSL *s, int send)
        {
        EVP_CIPHER_CTX *ds;
        unsigned long l;
        int bs;

        if (send)
```

```
128                 {
129                 ds=s->enc_write_ctx;
130                 l=s->s2->wlength;
131                 }
132         else
133                 {
134                 ds=s->enc_read_ctx;
135                 l=s->s2->rlength;
136                 }

138         /* check for NULL cipher */
139         if (ds == NULL) return;


142         bs=ds->cipher->block_size;
143         /* This should be using (bs-1) and bs instead of 7 and 8, but
144          * what the hell. */
145         if (bs == 8)
146                 l=(l+7)/8*8;

148         EVP_Cipher(ds,s->s2->mac_data,s->s2->mac_data,l);
149         }

151 void ssl2_mac(SSL *s, unsigned char *md, int send)
152         {
153         EVP_MD_CTX c;
154         unsigned char sequence[4],*p,*sec,*act;
155         unsigned long seq;
156         unsigned int len;

158         if (send)
159                 {
160                 seq=s->s2->write_sequence;
161                 sec=s->s2->write_key;
162                 len=s->s2->wact_data_length;
163                 act=s->s2->wact_data;
164                 }
165         else
166                 {
167                 seq=s->s2->read_sequence;
168                 sec=s->s2->read_key;
169                 len=s->s2->ract_data_length;
170                 act=s->s2->ract_data;
171                 }

173         p= &(sequence[0]);
174         l2n(seq,p);

176         /* There has to be a MAC algorithm. */
177         EVP_MD_CTX_init(&c);
178         EVP_MD_CTX_copy(&c, s->read_hash);
179         EVP_DigestUpdate(&c,sec,
180                 EVP_CIPHER_CTX_key_length(s->enc_read_ctx));
181         EVP_DigestUpdate(&c,act,len);
182         /* the above line also does the pad data */
183         EVP_DigestUpdate(&c,sequence,4);
184         EVP_DigestFinal_ex(&c,md,NULL);
185         EVP_MD_CTX_cleanup(&c);
186         }
187 #else /* !OPENSSL_NO_SSL2 */

189 # if PEDANTIC
190 static void *dummy=&dummy;
191 # endif

193 #endif
```



```
194 #endif /* ! codereview */
```

```
**********************************************************
   13411 Wed Aug 13 19:53:37 2014
new/usr/src/lib/openssl/libsunw_ssl/s2_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s2_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "ssl_locl.h"
#ifndef OPENSSL_NO_SSL2
#include <stdio.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/md5.h>

const char ssl2_version_str[]="SSLv2" OPENSSL_VERSION_PTEXT;

#define SSL2_NUM_CIPHERS (sizeof(ssl2_ciphers)/sizeof(SSL_CIPHER))

/* list of available SSLv2 ciphers (sorted by id) */
OPENSSL_GLOBAL const SSL_CIPHER ssl2_ciphers[]={
#if 0
/* NULL_WITH_MD5 v3 */
        {
```

```
128         1,
129         SSL2_TXT_NULL_WITH_MD5,
130         SSL2_CK_NULL_WITH_MD5,
131         SSL_kRSA,
132         SSL_aRSA,
133         SSL_eNULL,
134         SSL_MD5,
135         SSL_SSLV2,
136         SSL_EXPORT|SSL_EXP40|SSL_STRONG_NONE,
137         0,
138         0,
139         0,
140         },
141 #endif

143 /* RC4_128_WITH_MD5 */
144         {
145         1,
146         SSL2_TXT_RC4_128_WITH_MD5,
147         SSL2_CK_RC4_128_WITH_MD5,
148         SSL_kRSA,
149         SSL_aRSA,
150         SSL_RC4,
151         SSL_MD5,
152         SSL_SSLV2,
153         SSL_NOT_EXP|SSL_MEDIUM,
154         0,
155         128,
156         128,
157         },

159 /* RC4_128_EXPORT40_WITH_MD5 */
160         {
161         1,
162         SSL2_TXT_RC4_128_EXPORT40_WITH_MD5,
163         SSL2_CK_RC4_128_EXPORT40_WITH_MD5,
164         SSL_kRSA,
165         SSL_aRSA,
166         SSL_RC4,
167         SSL_MD5,
168         SSL_SSLV2,
169         SSL_EXPORT|SSL_EXP40,
170         SSL2_CF_5_BYTE_ENC,
171         40,
172         128,
173         },

175 /* RC2_128_CBC_WITH_MD5 */
176         {
177         1,
178         SSL2_TXT_RC2_128_CBC_WITH_MD5,
179         SSL2_CK_RC2_128_CBC_WITH_MD5,
180         SSL_kRSA,
181         SSL_aRSA,
182         SSL_RC2,
183         SSL_MD5,
184         SSL_SSLV2,
185         SSL_NOT_EXP|SSL_MEDIUM,
186         0,
187         128,
188         128,
189         },

191 /* RC2_128_CBC_EXPORT40_WITH_MD5 */
192         {
193         1,
```



```
194         SSL2_TXT_RC2_128_CBC_EXPORT40_WITH_MD5,
195         SSL2_CK_RC2_128_CBC_EXPORT40_WITH_MD5,
196         SSL_kRSA,
197         SSL_aRSA,
198         SSL_RC2,
199         SSL_MD5,
200         SSL_SSLV2,
201         SSL_EXPORT|SSL_EXP40,
202         SSL2_CF_5_BYTE_ENC,
203         40,
204         128,
205         },

207 #ifndef OPENSSL_NO_IDEA
208 /* IDEA_128_CBC_WITH_MD5 */
209         {
210         1,
211         SSL2_TXT_IDEA_128_CBC_WITH_MD5,
212         SSL2_CK_IDEA_128_CBC_WITH_MD5,
213         SSL_kRSA,
214         SSL_aRSA,
215         SSL_IDEA,
216         SSL_MD5,
217         SSL_SSLV2,
218         SSL_NOT_EXP|SSL_MEDIUM,
219         0,
220         128,
221         128,
222         },
223 #endif

225 /* DES_64_CBC_WITH_MD5 */
226         {
227         1,
228         SSL2_TXT_DES_64_CBC_WITH_MD5,
229         SSL2_CK_DES_64_CBC_WITH_MD5,
230         SSL_kRSA,
231         SSL_aRSA,
232         SSL_DES,
233         SSL_MD5,
234         SSL_SSLV2,
235         SSL_NOT_EXP|SSL_LOW,
236         0,
237         56,
238         56,
239         },

241 /* DES_192_EDE3_CBC_WITH_MD5 */
242         {
243         1,
244         SSL2_TXT_DES_192_EDE3_CBC_WITH_MD5,
245         SSL2_CK_DES_192_EDE3_CBC_WITH_MD5,
246         SSL_kRSA,
247         SSL_aRSA,
248         SSL_3DES,
249         SSL_MD5,
250         SSL_SSLV2,
251         SSL_NOT_EXP|SSL_HIGH,
252         0,
253         112,
254         168,
255         },

257 #if 0
258 /* RC4_64_WITH_MD5 */
259         {
```

```
260         1,
261         SSL2_TXT_RC4_64_WITH_MD5,
262         SSL2_CK_RC4_64_WITH_MD5,
263         SSL_kRSA,
264         SSL_aRSA,
265         SSL_RC4,
266         SSL_MD5,
267         SSL_SSLV2,
268         SSL_NOT_EXP|SSL_LOW,
269         SSL2_CF_8_BYTE_ENC,
270         64,
271         64,
272         },
273 #endif

275 #if 0
276 /* NULL SSLeay (testing) */
277         {
278         0,
279         SSL2_TXT_NULL,
280         SSL2_CK_NULL,
281         0,
282         0,
283         0,
284         0,
285         SSL_SSLV2,
286         SSL_STRONG_NONE,
287         0,
288         0,
289         0,
290         },
291 #endif

293 /* end of list :-) */
294         };

296 long ssl2_default_timeout(void)
297         {
298         return(300);
299         }

301 int ssl2_num_ciphers(void)
302         {
303         return(SSL2_NUM_CIPHERS);
304         }

306 const SSL_CIPHER *ssl2_get_cipher(unsigned int u)
307         {
308         if (u < SSL2_NUM_CIPHERS)
309                 return(&(ssl2_ciphers[SSL2_NUM_CIPHERS-1-u]));
310         else
311                 return(NULL);
312         }

314 int ssl2_pending(const SSL *s)
315         {
316         return SSL_in_init(s) ? 0 : s->s2->ract_data_length;
317         }

319 int ssl2_new(SSL *s)
320         {
321         SSL2_STATE *s2;

323         if ((s2=OPENSSL_malloc(sizeof *s2)) == NULL) goto err;
324         memset(s2,0,sizeof *s2);
```



```
326 #if SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER + 3 > SSL2_MAX_RECORD_LENGTH_2_BYTE_HEA
327 #  error "assertion failed"
328 #endif

330         if ((s2->rbuf=OPENSSL_malloc(
331                 SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER+2)) == NULL) goto err;
332         /* wbuf needs one byte more because when using two-byte headers,
333          * we leave the first byte unused in do_ssl_write (s2_pkt.c) */
334         if ((s2->wbuf=OPENSSL_malloc(
335                 SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER+3)) == NULL) goto err;
336         s->s2=s2;

338         ssl2_clear(s);
339         return(1);
340 err:
341         if (s2 != NULL)
342                 {
343                 if (s2->wbuf != NULL) OPENSSL_free(s2->wbuf);
344                 if (s2->rbuf != NULL) OPENSSL_free(s2->rbuf);
345                 OPENSSL_free(s2);
346                 }
347         return(0);
348         }

350 void ssl2_free(SSL *s)
351         {
352         SSL2_STATE *s2;

354         if(s == NULL)
355             return;

357         s2=s->s2;
358         if (s2->rbuf != NULL) OPENSSL_free(s2->rbuf);
359         if (s2->wbuf != NULL) OPENSSL_free(s2->wbuf);
360         OPENSSL_cleanse(s2,sizeof *s2);
361         OPENSSL_free(s2);
362         s->s2=NULL;
363         }

365 void ssl2_clear(SSL *s)
366         {
367         SSL2_STATE *s2;
368         unsigned char *rbuf,*wbuf;

370         s2=s->s2;

372         rbuf=s2->rbuf;
373         wbuf=s2->wbuf;

375         memset(s2,0,sizeof *s2);

377         s2->rbuf=rbuf;
378         s2->wbuf=wbuf;
379         s2->clear_text=1;
380         s->packet=s2->rbuf;
381         s->version=SSL2_VERSION;
382         s->packet_length=0;
383         }

385 long ssl2_ctrl(SSL *s, int cmd, long larg, void *parg)
386         {
387         int ret=0;

389         switch(cmd)
390                 {
391         case SSL_CTRL_GET_SESSION_REUSED:
```

```
392                 ret=s->hit;
393                 break;
394         default:
395                 break;
396                 }
397         return(ret);
398         }

400 long ssl2_callback_ctrl(SSL *s, int cmd, void (*fp)(void))
401         {
402         return(0);
403         }

405 long ssl2_ctx_ctrl(SSL_CTX *ctx, int cmd, long larg, void *parg)
406         {
407         return(0);
408         }

410 long ssl2_ctx_callback_ctrl(SSL_CTX *ctx, int cmd, void (*fp)(void))
411         {
412         return(0);
413         }

415 /* This function needs to check if the ciphers required are actually
416  * available */
417 const SSL_CIPHER *ssl2_get_cipher_by_char(const unsigned char *p)
418         {
419         SSL_CIPHER c;
420         const SSL_CIPHER *cp;
421         unsigned long id;

423         id=0x02000000L|((unsigned long)p[0]<<16L)|
424                 ((unsigned long)p[1]<<8L)|(unsigned long)p[2];
425         c.id=id;
426         cp = OBJ_bsearch_ssl_cipher_id(&c, ssl2_ciphers, SSL2_NUM_CIPHERS);
427         if ((cp == NULL) || (cp->valid == 0))
428                 return NULL;
429         else
430                 return cp;
431         }

433 int ssl2_put_cipher_by_char(const SSL_CIPHER *c, unsigned char *p)
434         {
435         long l;

437         if (p != NULL)
438                 {
439                 l=c->id;
440                 if ((l & 0xff000000) != 0x02000000) return(0);
441                 p[0]=((unsigned char)(l>>16L))&0xFF;
442                 p[1]=((unsigned char)(l>> 8L))&0xFF;
443                 p[2]=((unsigned char)(l     ))&0xFF;
444                 }
445         return(3);
446         }

448 int ssl2_generate_key_material(SSL *s)
449         {
450         unsigned int i;
451         EVP_MD_CTX ctx;
452         unsigned char *km;
453         unsigned char c='0';
454         const EVP_MD *md5;
455         int md_size;

457         md5 = EVP_md5();
```



```
459 #ifdef CHARSET_EBCDIC
460         c = os_toascii['0']; /* Must be an ASCII '0', not EBCDIC '0',
461                                 see SSLv2 docu */
462 #endif
463         EVP_MD_CTX_init(&ctx);
464         km=s->s2->key_material;

466         if (s->session->master_key_length < 0 ||
467                 s->session->master_key_length > (int)sizeof(s->session->
468                 {
469                 SSLerr(SSL_F_SSL2_GENERATE_KEY_MATERIAL, ERR_R_INTERNAL_ERROR);
470                 return 0;
471                 }
472         md_size = EVP_MD_size(md5);
473         if (md_size < 0)
474             return 0;
475         for (i=0; i<s->s2->key_material_length; i += md_size)
476                 {
477                 if (((km - s->s2->key_material) + md_size) >
478                         (int)sizeof(s->s2->key_material))
479                         {
480                         /* EVP_DigestFinal_ex() below would write beyond buffer
481                         SSLerr(SSL_F_SSL2_GENERATE_KEY_MATERIAL, ERR_R_INTERNAL_
482                         return 0;
483                         }

485                 EVP_DigestInit_ex(&ctx, md5, NULL);

487                 OPENSSL_assert(s->session->master_key_length >= 0
488                     && s->session->master_key_length
489                     < (int)sizeof(s->session->master_key));
490                 EVP_DigestUpdate(&ctx,s->session->master_key,s->session->master_
491                 EVP_DigestUpdate(&ctx,&c,1);
492                 c++;
493                 EVP_DigestUpdate(&ctx,s->s2->challenge,s->s2->challenge_length);
494                 EVP_DigestUpdate(&ctx,s->s2->conn_id,s->s2->conn_id_length);
495                 EVP_DigestFinal_ex(&ctx,km,NULL);
496                 km += md_size;
497                 }

499         EVP_MD_CTX_cleanup(&ctx);
500         return 1;
501         }

503 void ssl2_return_error(SSL *s, int err)
504         {
505         if (!s->error)
506                 {
507                 s->error=3;
508                 s->error_code=err;

510                 ssl2_write_error(s);
511                 }
512         }


515 void ssl2_write_error(SSL *s)
516         {
517         unsigned char buf[3];
518         int i,error;

520         buf[0]=SSL2_MT_ERROR;
521         buf[1]=(s->error_code>>8)&0xff;
522         buf[2]=(s->error_code)&0xff;
```

```
524 /*      state=s->rwstate;*/
526         error=s->error; /* number of bytes left to write */
527         s->error=0;
528         OPENSSL_assert(error >= 0 && error <= (int)sizeof(buf));
529         i=ssl2_write(s,&(buf[3-error]),error);
531 /*      if (i == error) s->rwstate=state; */
533         if (i < 0)
534                 s->error=error;
535         else
536                 {
537                 s->error=error-i;
539                 if (s->error == 0)
540                         if (s->msg_callback)
541                                 s->msg_callback(1, s->version, 0, buf, 3, s, s->
542                 }
543         }
545 int ssl2_shutdown(SSL *s)
546         {
547         s->shutdown=(SSL_SENT_SHUTDOWN|SSL_RECEIVED_SHUTDOWN);
548         return(1);
549         }
550 #else /* !OPENSSL_NO_SSL2 */
552 # if PEDANTIC
553 static void *dummy=&dummy;
554 # endif
556 #endif
557 #endif /* ! codereview */
```

```
**********************************************************
    3644 Wed Aug 13 19:53:37 2014
new/usr/src/lib/openssl/libsunw_ssl/s2_meth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s2_meth.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include "ssl_locl.h"
#ifndef OPENSSL_NO_SSL2
#include <stdio.h>
```



```
#include <openssl/objects.h>

static const SSL_METHOD *ssl2_get_method(int ver);
static const SSL_METHOD *ssl2_get_method(int ver)
        {
        if (ver == SSL2_VERSION)
                return(SSLv2_method());
        else
                return(NULL);
        }

IMPLEMENT_ssl2_meth_func(SSLv2_method,
                         ssl2_accept,
                         ssl2_connect,
                         ssl2_get_method)

#else /* !OPENSSL_NO_SSL2 */

# if PEDANTIC
static void *dummy=&dummy;
# endif

#endif
#endif /* ! codereview */
```

```
**********************************************************
   20186 Wed Aug 13 19:53:37 2014
new/usr/src/lib/openssl/libsunw_ssl/s2_pkt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s2_pkt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "ssl_locl.h"
#ifndef OPENSSL_NO_SSL2
#include <stdio.h>
#include <errno.h>
#define USE_SOCKETS

static int read_n(SSL *s,unsigned int n,unsigned int max,unsigned int extend);
static int n_do_ssl_write(SSL *s, const unsigned char *buf, unsigned int len);
static int write_pending(SSL *s, const unsigned char *buf, unsigned int len);
static int ssl_mt_error(int n);


/* SSL 2.0 imlementation for SSL_read/SSL_peek -
 * This routine will return 0 to len bytes, decrypted etc if required.
 */
static int ssl2_read_internal(SSL *s, void *buf, int len, int peek)
```

```
128         {
129         int n;
130         unsigned char mac[MAX_MAC_SIZE];
131         unsigned char *p;
132         int i;
133         int mac_size;

135 ssl2_read_again:
136         if (SSL_in_init(s) && !s->in_handshake)
137                 {
138                 n=s->handshake_func(s);
139                 if (n < 0) return(n);
140                 if (n == 0)
141                         {
142                         SSLerr(SSL_F_SSL2_READ_INTERNAL,SSL_R_SSL_HANDSHAKE_FAIL
143                         return(-1);
144                         }
145                 }

147         clear_sys_error();
148         s->rwstate=SSL_NOTHING;
149         if (len <= 0) return(len);

151         if (s->s2->ract_data_length != 0) /* read from buffer */
152                 {
153                 if (len > s->s2->ract_data_length)
154                         n=s->s2->ract_data_length;
155                 else
156                         n=len;

158                 memcpy(buf,s->s2->ract_data,(unsigned int)n);
159                 if (!peek)
160                         {
161                         s->s2->ract_data_length-=n;
162                         s->s2->ract_data+=n;
163                         if (s->s2->ract_data_length == 0)
164                                 s->rstate=SSL_ST_READ_HEADER;
165                         }

167                 return(n);
168                 }

170         /* s->s2->ract_data_length == 0
171          *
172          * Fill the buffer, then goto ssl2_read_again.
173          */

175         if (s->rstate == SSL_ST_READ_HEADER)
176                 {
177                 if (s->first_packet)
178                         {
179                         n=read_n(s,5,SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER+2,0);
180                         if (n <= 0) return(n); /* error or non-blocking */
181                         s->first_packet=0;
182                         p=s->packet;
183                         if (!((p[0] & 0x80) && (
184                                 (p[2] == SSL2_MT_CLIENT_HELLO) ||
185                                 (p[2] == SSL2_MT_SERVER_HELLO))))
186                                 {
187                                 SSLerr(SSL_F_SSL2_READ_INTERNAL,SSL_R_NON_SSLV2_
188                                 return(-1);
189                                 }
190                         }
191                 else
192                         {
193                         n=read_n(s,2,SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER+2,0);
```



```
194                         if (n <= 0) return(n); /* error or non-blocking */
195                         }
196                 /* part read stuff */

198                 s->rstate=SSL_ST_READ_BODY;
199                 p=s->packet;
200                 /* Do header */
201                 /*s->s2->padding=0;*/
202                 s->s2->escape=0;
203                 s->s2->rlength=(((unsigned int)p[0])<<8)|((unsigned int)p[1]);
204                 if ((p[0] & TWO_BYTE_BIT))              /* Two byte header? */
205                         {
206                         s->s2->three_byte_header=0;
207                         s->s2->rlength&=TWO_BYTE_MASK;
208                         }
209                 else
210                         {
211                         s->s2->three_byte_header=1;
212                         s->s2->rlength&=THREE_BYTE_MASK;

214                         /* security >s2->escape */
215                         s->s2->escape=((p[0] & SEC_ESC_BIT))?1:0;
216                         }
217                 }

219         if (s->rstate == SSL_ST_READ_BODY)
220                 {
221                 n=s->s2->rlength+2+s->s2->three_byte_header;
222                 if (n > (int)s->packet_length)
223                         {
224                         n-=s->packet_length;
225                         i=read_n(s,(unsigned int)n,(unsigned int)n,1);
226                         if (i <= 0) return(i); /* ERROR */
227                         }

229                 p= &(s->packet[2]);
230                 s->rstate=SSL_ST_READ_HEADER;
231                 if (s->s2->three_byte_header)
232                         s->s2->padding= *(p++);
233                 else    s->s2->padding=0;

235                 /* Data portion */
236                 if (s->s2->clear_text)
237                         {
238                         mac_size = 0;
239                         s->s2->mac_data=p;
240                         s->s2->ract_data=p;
241                         if (s->s2->padding)
242                                 {
243                                 SSLerr(SSL_F_SSL2_READ_INTERNAL,SSL_R_ILLEGAL_PA
244                                 return(-1);
245                                 }
246                         }
247                 else
248                         {
249                         mac_size=EVP_MD_CTX_size(s->read_hash);
250                         if (mac_size < 0)
251                                 return -1;
252                         OPENSSL_assert(mac_size <= MAX_MAC_SIZE);
253                         s->s2->mac_data=p;
254                         s->s2->ract_data= &p[mac_size];
255                         if (s->s2->padding + mac_size > s->s2->rlength)
256                                 {
257                                 SSLerr(SSL_F_SSL2_READ_INTERNAL,SSL_R_ILLEGAL_PA
258                                 return(-1);
259                                 }
```

```
260                         }

262                 s->s2->ract_data_length=s->s2->rlength;
263                 /* added a check for length > max_size in case
264                  * encryption was not turned on yet due to an error */
265                 if ((!s->s2->clear_text) &&
266                         (s->s2->rlength >= (unsigned int)mac_size))
267                         {
268                         ssl2_enc(s,0);
269                         s->s2->ract_data_length-=mac_size;
270                         ssl2_mac(s,mac,0);
271                         s->s2->ract_data_length-=s->s2->padding;
272                         if (    (CRYPTO_memcmp(mac,s->s2->mac_data,mac_size) !=
273                                 (s->s2->rlength%EVP_CIPHER_CTX_block_size(s->enc
274                                 {
275                                 SSLerr(SSL_F_SSL2_READ_INTERNAL,SSL_R_BAD_MAC_DE
276                                 return(-1);
277                                 }
278                         }
279                 INC32(s->s2->read_sequence); /* expect next number */
280                 /* s->s2->ract_data is now available for processing */

282                 /* Possibly the packet that we just read had 0 actual data bytes
283                  * (SSLeay/OpenSSL itself never sends such packets; see ssl2_wri
284                  * In this case, returning 0 would be interpreted by the caller
285                  * as indicating EOF, so it's not a good idea.  Instead, we just
286                  * continue reading; thus ssl2_read_internal may have to process
287                  * multiple packets before it can return.
288                  *
289                  * [Note that using select() for blocking sockets *never* guaran
290                  * that the next SSL_read will not block -- the available
291                  * data may contain incomplete packets, and except for SSL 2,
292                  * renegotiation can confuse things even more.] */

294                 goto ssl2_read_again; /* This should really be
295                                        * "return ssl2_read(s,buf,len)",
296                                        * but that would allow for
297                                        * denial-of-service attacks if a
298                                        * C compiler is used that does not
299                                        * recognize end-recursion. */
300                 }
301         else
302                 {
303                 SSLerr(SSL_F_SSL2_READ_INTERNAL,SSL_R_BAD_STATE);
304                         return(-1);
305                 }
306         }

308 int ssl2_read(SSL *s, void *buf, int len)
309         {
310         return ssl2_read_internal(s, buf, len, 0);
311         }

313 int ssl2_peek(SSL *s, void *buf, int len)
314         {
315         return ssl2_read_internal(s, buf, len, 1);
316         }

318 static int read_n(SSL *s, unsigned int n, unsigned int max,
319                   unsigned int extend)
320         {
321         int i,off,newb;

323         /* if there is stuff still in the buffer from a previous read,
324          * and there is more than we want, take some. */
325         if (s->s2->rbuf_left >= (int)n)
```



```
326                 {
327                 if (extend)
328                         s->packet_length+=n;
329                 else
330                         {
331                         s->packet= &(s->s2->rbuf[s->s2->rbuf_offs]);
332                         s->packet_length=n;
333                         }
334                 s->s2->rbuf_left-=n;
335                 s->s2->rbuf_offs+=n;
336                 return(n);
337                 }

339         if (!s->read_ahead) max=n;
340         if (max > (unsigned int)(SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER+2))
341                 max=SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER+2;


344         /* Else we want more than we have.
345          * First, if there is some left or we want to extend */
346         off=0;
347         if ((s->s2->rbuf_left != 0) || ((s->packet_length != 0) && extend))
348                 {
349                 newb=s->s2->rbuf_left;
350                 if (extend)
351                         {
352                         off=s->packet_length;
353                         if (s->packet != s->s2->rbuf)
354                                 memcpy(s->s2->rbuf,s->packet,
355                                         (unsigned int)newb+off);
356                         }
357                 else if (s->s2->rbuf_offs != 0)
358                         {
359                         memcpy(s->s2->rbuf,&(s->s2->rbuf[s->s2->rbuf_offs]),
360                                 (unsigned int)newb);
361                         s->s2->rbuf_offs=0;
362                         }
363                 s->s2->rbuf_left=0;
364                 }
365         else
366                 newb=0;

368         /* off is the offset to start writing too.
369          * r->s2->rbuf_offs is the 'unread data', now 0.
370          * newb is the number of new bytes so far
371          */
372         s->packet=s->s2->rbuf;
373         while (newb < (int)n)
374                 {
375                 clear_sys_error();
376                 if (s->rbio != NULL)
377                         {
378                         s->rwstate=SSL_READING;
379                         i=BIO_read(s->rbio,(char *)&(s->s2->rbuf[off+newb]),
380                                 max-newb);
381                         }
382                 else
383                         {
384                         SSLerr(SSL_F_READ_N,SSL_R_READ_BIO_NOT_SET);
385                         i= -1;
386                         }
387 #ifdef PKT_DEBUG
388                 if (s->debug & 0x01) sleep(1);
389 #endif
390                 if (i <= 0)
391                         {
```

```
392                         s->s2->rbuf_left+=newb;
393                         return(i);
394                         }
395                 newb+=i;
396                 }

398         /* record unread data */
399         if (newb > (int)n)
400                 {
401                 s->s2->rbuf_offs=n+off;
402                 s->s2->rbuf_left=newb-n;
403                 }
404         else
405                 {
406                 s->s2->rbuf_offs=0;
407                 s->s2->rbuf_left=0;
408                 }
409         if (extend)
410                 s->packet_length+=n;
411         else
412                 s->packet_length=n;
413         s->rwstate=SSL_NOTHING;
414         return(n);
415         }

417 int ssl2_write(SSL *s, const void *_buf, int len)
418         {
419         const unsigned char *buf=_buf;
420         unsigned int n,tot;
421         int i;

423         if (SSL_in_init(s) && !s->in_handshake)
424                 {
425                 i=s->handshake_func(s);
426                 if (i < 0) return(i);
427                 if (i == 0)
428                         {
429                         SSLerr(SSL_F_SSL2_WRITE,SSL_R_SSL_HANDSHAKE_FAILURE);
430                         return(-1);
431                         }
432                 }

434         if (s->error)
435                 {
436                 ssl2_write_error(s);
437                 if (s->error)
438                         return(-1);
439                 }

441         clear_sys_error();
442         s->rwstate=SSL_NOTHING;
443         if (len <= 0) return(len);

445         tot=s->s2->wnum;
446         s->s2->wnum=0;

448         n=(len-tot);
449         for (;;)
450                 {
451                 i=n_do_ssl_write(s,&(buf[tot]),n);
452                 if (i <= 0)
453                         {
454                         s->s2->wnum=tot;
455                         return(i);
456                         }
457                 if ((i == (int)n) ||
```



```
458                         (s->mode & SSL_MODE_ENABLE_PARTIAL_WRITE))
459                         {
460                         return(tot+i);
461                         }

463                 n-=i;
464                 tot+=i;
465                 }
466         }

468 static int write_pending(SSL *s, const unsigned char *buf, unsigned int len)
469         {
470         int i;

472         /* s->s2->wpend_len != 0 MUST be true. */

474         /* check that they have given us the same buffer to
475          * write */
476         if ((s->s2->wpend_tot > (int)len) ||
477                 ((s->s2->wpend_buf != buf) &&
478                  !(s->mode & SSL_MODE_ACCEPT_MOVING_WRITE_BUFFER)))
479                 {
480                 SSLerr(SSL_F_WRITE_PENDING,SSL_R_BAD_WRITE_RETRY);
481                 return(-1);
482                 }

484         for (;;)
485                 {
486                 clear_sys_error();
487                 if (s->wbio != NULL)
488                         {
489                         s->rwstate=SSL_WRITING;
490                         i=BIO_write(s->wbio,
491                                 (char *)&(s->s2->write_ptr[s->s2->wpend_off]),
492                                 (unsigned int)s->s2->wpend_len);
493                         }
494                 else
495                         {
496                         SSLerr(SSL_F_WRITE_PENDING,SSL_R_WRITE_BIO_NOT_SET);
497                         i= -1;
498                         }
499 #ifdef PKT_DEBUG
500                 if (s->debug & 0x01) sleep(1);
501 #endif
502                 if (i == s->s2->wpend_len)
503                         {
504                         s->s2->wpend_len=0;
505                         s->rwstate=SSL_NOTHING;
506                         return(s->s2->wpend_ret);
507                         }
508                 else if (i <= 0)
509                         return(i);
510                 s->s2->wpend_off+=i;
511                 s->s2->wpend_len-=i;
512                 }
513         }

515 static int n_do_ssl_write(SSL *s, const unsigned char *buf, unsigned int len)
516         {
517         unsigned int j,k,olen,p,bs;
518         int mac_size;
519         register unsigned char *pp;

521         olen=len;

523         /* first check if there is data from an encryption waiting to
```

```
524          * be sent - it must be sent because the other end is waiting.
525          * This will happen with non-blocking IO.  We print it and then
526          * return.
527          */
528         if (s->s2->wpend_len != 0) return(write_pending(s,buf,len));
529
530         /* set mac_size to mac size */
531         if (s->s2->clear_text)
532                 mac_size=0;
533         else
534                 {
535                 mac_size=EVP_MD_CTX_size(s->write_hash);
536                 if (mac_size < 0)
537                         return -1;
538                 }
539
540         /* lets set the pad p */
541         if (s->s2->clear_text)
542                 {
543                 if (len > SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER)
544                         len=SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER;
545                 p=0;
546                 s->s2->three_byte_header=0;
547                 /* len=len; */
548                 }
549         else
550                 {
551                 bs=EVP_CIPHER_CTX_block_size(s->enc_read_ctx);
552                 j=len+mac_size;
553                 /* Two-byte headers allow for a larger record length than
554                  * three-byte headers, but we can't use them if we need
555                  * padding or if we have to set the escape bit. */
556                 if ((j > SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER) &&
557                         (!s->s2->escape))
558                         {
559                         if (j > SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER)
560                                 j=SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER;
561                         /* set k to the max number of bytes with 2
562                          * byte header */
563                         k=j-(j%bs);
564                         /* how many data bytes? */
565                         len=k-mac_size;
566                         s->s2->three_byte_header=0;
567                         p=0;
568                         }
569                 else if ((bs <= 1) && (!s->s2->escape))
570                         {
571                         /* j <= SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER, thus
572                          * j < SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER */
573                         s->s2->three_byte_header=0;
574                         p=0;
575                         }
576                 else /* we may have to use a 3 byte header */
577                         {
578                         /* If s->s2->escape is not set, then
579                          * j <= SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER, and thus
580                          * j < SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER. */
581                         p=(j%bs);
582                         p=(p == 0)?0:(bs-p);
583                         if (s->s2->escape)
584                                 {
585                                 s->s2->three_byte_header=1;
586                                 if (j > SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER)
587                                         j=SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER;
588                                 }
589                         else
```



```
590                                 s->s2->three_byte_header=(p == 0)?0:1;
591                         }
592                 }
593
594         /* Now
595          *      j <= SSL2_MAX_RECORD_LENGTH_2_BYTE_HEADER
596          * holds, and if s->s2->three_byte_header is set, then even
597          *      j <= SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER.
598          */
599
600         /* mac_size is the number of MAC bytes
601          * len is the number of data bytes we are going to send
602          * p is the number of padding bytes
603          * (if it is a two-byte header, then p == 0) */
604
605         s->s2->wlength=len;
606         s->s2->padding=p;
607         s->s2->mac_data= &(s->s2->wbuf[3]);
608         s->s2->wact_data= &(s->s2->wbuf[3+mac_size]);
609         /* we copy the data into s->s2->wbuf */
610         memcpy(s->s2->wact_data,buf,len);
611         if (p)
612                 memset(&(s->s2->wact_data[len]),0,p); /* arbitrary padding */
613
614         if (!s->s2->clear_text)
615                 {
616                 s->s2->wact_data_length=len+p;
617                 ssl2_mac(s,s->s2->mac_data,1);
618                 s->s2->wlength+=p+mac_size;
619                 ssl2_enc(s,1);
620                 }
621
622         /* package up the header */
623         s->s2->wpend_len=s->s2->wlength;
624         if (s->s2->three_byte_header) /* 3 byte header */
625                 {
626                 pp=s->s2->mac_data;
627                 pp-=3;
628                 pp[0]=(s->s2->wlength>>8)&(THREE_BYTE_MASK>>8);
629                 if (s->s2->escape) pp[0]|=SEC_ESC_BIT;
630                 pp[1]=s->s2->wlength&0xff;
631                 pp[2]=s->s2->padding;
632                 s->s2->wpend_len+=3;
633                 }
634         else
635                 {
636                 pp=s->s2->mac_data;
637                 pp-=2;
638                 pp[0]=((s->s2->wlength>>8)&(TWO_BYTE_MASK>>8))|TWO_BYTE_BIT;
639                 pp[1]=s->s2->wlength&0xff;
640                 s->s2->wpend_len+=2;
641                 }
642         s->s2->write_ptr=pp;
643
644         INC32(s->s2->write_sequence); /* expect next number */
645
646         /* lets try to actually write the data */
647         s->s2->wpend_tot=olen;
648         s->s2->wpend_buf=buf;
649
650         s->s2->wpend_ret=len;
651
652         s->s2->wpend_off=0;
653         return(write_pending(s,buf,olen));
654         }
```

```
656 int ssl2_part_read(SSL *s, unsigned long f, int i)
657         {
658         unsigned char *p;
659         int j;
660
661         if (i < 0)
662                 {
663                 /* ssl2_return_error(s); */
664                 /* for non-blocking io,
665                  * this is not necessarily fatal */
666                 return(i);
667                 }
668         else
669                 {
670                 s->init_num+=i;
671
672                 /* Check for error.  While there are recoverable errors,
673                  * this function is not called when those must be expected;
674                  * any error detected here is fatal. */
675                 if (s->init_num >= 3)
676                         {
677                         p=(unsigned char *)s->init_buf->data;
678                         if (p[0] == SSL2_MT_ERROR)
679                                 {
680                                 j=(p[1]<<8)|p[2];
681                                 SSLerr((int)f,ssl_mt_error(j));
682                                 s->init_num -= 3;
683                                 if (s->init_num > 0)
684                                         memmove(p, p+3, s->init_num);
685                                 }
686                         }
687
688                 /* If it's not an error message, we have some error anyway --
689                  * the message was shorter than expected.  This too is treated
690                  * as fatal (at least if SSL_get_error is asked for its opinion)
691                 return(0);
692                 }
693         }
694
695 int ssl2_do_write(SSL *s)
696         {
697         int ret;
698
699         ret=ssl2_write(s,&s->init_buf->data[s->init_off],s->init_num);
700         if (ret == s->init_num)
701                 {
702                 if (s->msg_callback)
703                         s->msg_callback(1, s->version, 0, s->init_buf->data, (si
704                 return(1);
705                 }
706         if (ret < 0)
707                 return(-1);
708         s->init_off+=ret;
709         s->init_num-=ret;
710         return(0);
711         }
712
713 static int ssl_mt_error(int n)
714         {
715         int ret;
716
717         switch (n)
718                 {
719         case SSL2_PE_NO_CIPHER:
720                 ret=SSL_R_PEER_ERROR_NO_CIPHER;
721                 break;
```



```
722         case SSL2_PE_NO_CERTIFICATE:
723                 ret=SSL_R_PEER_ERROR_NO_CERTIFICATE;
724                 break;
725         case SSL2_PE_BAD_CERTIFICATE:
726                 ret=SSL_R_PEER_ERROR_CERTIFICATE;
727                 break;
728         case SSL2_PE_UNSUPPORTED_CERTIFICATE_TYPE:
729                 ret=SSL_R_PEER_ERROR_UNSUPPORTED_CERTIFICATE_TYPE;
730                 break;
731         default:
732                 ret=SSL_R_UNKNOWN_REMOTE_ERROR_TYPE;
733                 break;
734                 }
735         return(ret);
736         }
737 #else /* !OPENSSL_NO_SSL2 */
738
739 # if PEDANTIC
740 static void *dummy=&dummy;
741 # endif
742
743 #endif
744 #endif /* ! codereview */
```

```
**********************************************************
   32224 Wed Aug 13 19:53:37 2014
new/usr/src/lib/openssl/libsunw_ssl/s2_srvr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s2_srvr.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2001 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "ssl_locl.h"
#ifndef OPENSSL_NO_SSL2
#include <stdio.h>
#include <openssl/bio.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>

static const SSL_METHOD *ssl2_get_server_method(int ver);
static int get_client_master_key(SSL *s);
static int get_client_hello(SSL *s);
static int server_hello(SSL *s);
static int get_client_finished(SSL *s);
static int server_verify(SSL *s);
static int server_finish(SSL *s);
static int request_certificate(SSL *s);
```

```
128 static int ssl_rsa_private_decrypt(CERT *c, int len, unsigned char *from,
129         unsigned char *to,int padding);
130 #define BREAK   break

132 static const SSL_METHOD *ssl2_get_server_method(int ver)
133         {
134         if (ver == SSL2_VERSION)
135                 return(SSLv2_server_method());
136         else
137                 return(NULL);
138         }

140 IMPLEMENT_ssl2_meth_func(SSLv2_server_method,
141                          ssl2_accept,
142                          ssl_undefined_function,
143                          ssl2_get_server_method)

145 int ssl2_accept(SSL *s)
146         {
147         unsigned long l=(unsigned long)time(NULL);
148         BUF_MEM *buf=NULL;
149         int ret= -1;
150         long num1;
151         void (*cb)(const SSL *ssl,int type,int val)=NULL;
152         int new_state,state;

154         RAND_add(&l,sizeof(l),0);
155         ERR_clear_error();
156         clear_sys_error();

158         if (s->info_callback != NULL)
159                 cb=s->info_callback;
160         else if (s->ctx->info_callback != NULL)
161                 cb=s->ctx->info_callback;

163         /* init things to blank */
164         s->in_handshake++;
165         if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

167         if (s->cert == NULL)
168                 {
169                 SSLerr(SSL_F_SSL2_ACCEPT,SSL_R_NO_CERTIFICATE_SET);
170                 return(-1);
171                 }

173         clear_sys_error();
174         for (;;)
175                 {
176                 state=s->state;

178                 switch (s->state)
179                         {
180                 case SSL_ST_BEFORE:
181                 case SSL_ST_ACCEPT:
182                 case SSL_ST_BEFORE|SSL_ST_ACCEPT:
183                 case SSL_ST_OK|SSL_ST_ACCEPT:

185                         s->server=1;
186                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

188                         s->version=SSL2_VERSION;
189                         s->type=SSL_ST_ACCEPT;

191                         buf=s->init_buf;
192                         if ((buf == NULL) && ((buf=BUF_MEM_new()) == NULL))
193                                 { ret= -1; goto end; }
```



```
194                         if (!BUF_MEM_grow(buf,(int)
195                                 SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER))
196                                 { ret= -1; goto end; }
197                         s->init_buf=buf;
198                         s->init_num=0;
199                         s->ctx->stats.sess_accept++;
200                         s->handshake_func=ssl2_accept;
201                         s->state=SSL2_ST_GET_CLIENT_HELLO_A;
202                         BREAK;

204                 case SSL2_ST_GET_CLIENT_HELLO_A:
205                 case SSL2_ST_GET_CLIENT_HELLO_B:
206                 case SSL2_ST_GET_CLIENT_HELLO_C:
207                         s->shutdown=0;
208                         ret=get_client_hello(s);
209                         if (ret <= 0) goto end;
210                         s->init_num=0;
211                         s->state=SSL2_ST_SEND_SERVER_HELLO_A;
212                         BREAK;

214                 case SSL2_ST_SEND_SERVER_HELLO_A:
215                 case SSL2_ST_SEND_SERVER_HELLO_B:
216                         ret=server_hello(s);
217                         if (ret <= 0) goto end;
218                         s->init_num=0;
219                         if (!s->hit)
220                                 {
221                                 s->state=SSL2_ST_GET_CLIENT_MASTER_KEY_A;
222                                 BREAK;
223                                 }
224                         else
225                                 {
226                                 s->state=SSL2_ST_SERVER_START_ENCRYPTION;
227                                 BREAK;
228                                 }
229                 case SSL2_ST_GET_CLIENT_MASTER_KEY_A:
230                 case SSL2_ST_GET_CLIENT_MASTER_KEY_B:
231                         ret=get_client_master_key(s);
232                         if (ret <= 0) goto end;
233                         s->init_num=0;
234                         s->state=SSL2_ST_SERVER_START_ENCRYPTION;
235                         BREAK;

237                 case SSL2_ST_SERVER_START_ENCRYPTION:
238                         /* Ok we how have sent all the stuff needed to
239                          * start encrypting, the next packet back will
240                          * be encrypted. */
241                         if (!ssl2_enc_init(s,0))
242                                 { ret= -1; goto end; }
243                         s->s2->clear_text=0;
244                         s->state=SSL2_ST_SEND_SERVER_VERIFY_A;
245                         BREAK;

247                 case SSL2_ST_SEND_SERVER_VERIFY_A:
248                 case SSL2_ST_SEND_SERVER_VERIFY_B:
249                         ret=server_verify(s);
250                         if (ret <= 0) goto end;
251                         s->init_num=0;
252                         if (s->hit)
253                                 {
254                                 /* If we are in here, we have been
255                                  * buffering the output, so we need to
256                                  * flush it and remove buffering from
257                                  * future traffic */
258                                 s->state=SSL2_ST_SEND_SERVER_VERIFY_C;
259                                 BREAK;
```

```
260                             }
261                     else
262                             {
263                             s->state=SSL2_ST_GET_CLIENT_FINISHED_A;
264                             break;
265                             }

267             case SSL2_ST_SEND_SERVER_VERIFY_C:
268                     /* get the number of bytes to write */
269                     num1=BIO_ctrl(s->wbio,BIO_CTRL_INFO,0,NULL);
270                     if (num1 > 0)
271                             {
272                             s->rwstate=SSL_WRITING;
273                             num1=BIO_flush(s->wbio);
274                             if (num1 <= 0) { ret= -1; goto end; }
275                             s->rwstate=SSL_NOTHING;
276                             }

278                     /* flushed and now remove buffering */
279                     s->wbio=BIO_pop(s->wbio);

281                     s->state=SSL2_ST_GET_CLIENT_FINISHED_A;
282                     BREAK;

284             case SSL2_ST_GET_CLIENT_FINISHED_A:
285             case SSL2_ST_GET_CLIENT_FINISHED_B:
286                     ret=get_client_finished(s);
287                     if (ret <= 0)
288                             goto end;
289                     s->init_num=0;
290                     s->state=SSL2_ST_SEND_REQUEST_CERTIFICATE_A;
291                     BREAK;

293             case SSL2_ST_SEND_REQUEST_CERTIFICATE_A:
294             case SSL2_ST_SEND_REQUEST_CERTIFICATE_B:
295             case SSL2_ST_SEND_REQUEST_CERTIFICATE_C:
296             case SSL2_ST_SEND_REQUEST_CERTIFICATE_D:
297                     /* don't do a 'request certificate' if we
298                      * don't want to, or we already have one, and
299                      * we only want to do it once. */
300                     if (!(s->verify_mode & SSL_VERIFY_PEER) ||
301                             ((s->session->peer != NULL) &&
302                             (s->verify_mode & SSL_VERIFY_CLIENT_ONCE)))
303                             {
304                             s->state=SSL2_ST_SEND_SERVER_FINISHED_A;
305                             break;
306                             }
307                     else
308                             {
309                             ret=request_certificate(s);
310                             if (ret <= 0) goto end;
311                             s->init_num=0;
312                             s->state=SSL2_ST_SEND_SERVER_FINISHED_A;
313                             }
314                     BREAK;

316             case SSL2_ST_SEND_SERVER_FINISHED_A:
317             case SSL2_ST_SEND_SERVER_FINISHED_B:
318                     ret=server_finish(s);
319                     if (ret <= 0) goto end;
320                     s->init_num=0;
321                     s->state=SSL_ST_OK;
322                     break;

324             case SSL_ST_OK:
325                     BUF_MEM_free(s->init_buf);
```



```
326                             ssl_free_wbio_buffer(s);
327                             s->init_buf=NULL;
328                             s->init_num=0;
329                     /*      ERR_clear_error();*/

331                             ssl_update_cache(s,SSL_SESS_CACHE_SERVER);

333                             s->ctx->stats.sess_accept_good++;
334                             /* s->server=1; */
335                             ret=1;

337                             if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_DONE,1);

339                             goto end;
340                             /* BREAK; */

342                     default:
343                             SSLerr(SSL_F_SSL2_ACCEPT,SSL_R_UNKNOWN_STATE);
344                             ret= -1;
345                             goto end;
346                             /* BREAK; */
347                             }

349                     if ((cb != NULL) && (s->state != state))
350                             {
351                             new_state=s->state;
352                             s->state=state;
353                             cb(s,SSL_CB_ACCEPT_LOOP,1);
354                             s->state=new_state;
355                             }
356                     }
357     end:
358             s->in_handshake--;
359             if (cb != NULL)
360                     cb(s,SSL_CB_ACCEPT_EXIT,ret);
361             return(ret);
362             }

364     static int get_client_master_key(SSL *s)
365             {
366             int is_export,i,n,keya,ek;
367             unsigned long len;
368             unsigned char *p;
369             const SSL_CIPHER *cp;
370             const EVP_CIPHER *c;
371             const EVP_MD *md;

373             p=(unsigned char *)s->init_buf->data;
374             if (s->state == SSL2_ST_GET_CLIENT_MASTER_KEY_A)
375                     {
376                     i=ssl2_read(s,(char *)&(p[s->init_num]),10-s->init_num);

378                     if (i < (10-s->init_num))
379                             return(ssl2_part_read(s,SSL_F_GET_CLIENT_MASTER_KEY,i));
380                     s->init_num = 10;

382                     if (*(p++) != SSL2_MT_CLIENT_MASTER_KEY)
383                             {
384                             if (p[-1] != SSL2_MT_ERROR)
385                                     {
386                                     ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
387                                     SSLerr(SSL_F_GET_CLIENT_MASTER_KEY,SSL_R_READ_WR
388                                     }
389                             else
390                                     SSLerr(SSL_F_GET_CLIENT_MASTER_KEY, SSL_R_PEER_E
391                             return(-1);
```

```
392                         }

394                 cp=ssl2_get_cipher_by_char(p);
395                 if (cp == NULL)
396                         {
397                         ssl2_return_error(s,SSL2_PE_NO_CIPHER);
398                         SSLerr(SSL_F_GET_CLIENT_MASTER_KEY, SSL_R_NO_CIPHER_MATC
399                         return(-1);
400                         }
401                 s->session->cipher= cp;

403                 p+=3;
404                 n2s(p,i); s->s2->tmp.clear=i;
405                 n2s(p,i); s->s2->tmp.enc=i;
406                 n2s(p,i);
407                 if(i > SSL_MAX_KEY_ARG_LENGTH)
408                         {
409                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
410                         SSLerr(SSL_F_GET_CLIENT_MASTER_KEY, SSL_R_KEY_ARG_TOO_LO
411                         return -1;
412                         }
413                 s->session->key_arg_length=i;
414                 s->state=SSL2_ST_GET_CLIENT_MASTER_KEY_B;
415                 }

417         /* SSL2_ST_GET_CLIENT_MASTER_KEY_B */
418         p=(unsigned char *)s->init_buf->data;
419         if (s->init_buf->length < SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER)
420                 {
421                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
422                 SSLerr(SSL_F_GET_CLIENT_MASTER_KEY, ERR_R_INTERNAL_ERROR);
423                 return -1;
424                 }
425         keya=s->session->key_arg_length;
426         len = 10 + (unsigned long)s->s2->tmp.clear + (unsigned long)s->s2->tmp.e
427         if (len > SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER)
428                 {
429                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
430                 SSLerr(SSL_F_GET_CLIENT_MASTER_KEY,SSL_R_MESSAGE_TOO_LONG);
431                 return -1;
432                 }
433         n = (int)len - s->init_num;
434         i = ssl2_read(s,(char *)&(p[s->init_num]),n);
435         if (i != n) return(ssl2_part_read(s,SSL_F_GET_CLIENT_MASTER_KEY,i));
436         if (s->msg_callback)
437                 s->msg_callback(0, s->version, 0, p, (size_t)len, s, s->msg_call
438         p += 10;

440         memcpy(s->session->key_arg,&(p[s->s2->tmp.clear+s->s2->tmp.enc]),
441                 (unsigned int)keya);

443         if (s->cert->pkeys[SSL_PKEY_RSA_ENC].privatekey == NULL)
444                 {
445                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
446                 SSLerr(SSL_F_GET_CLIENT_MASTER_KEY,SSL_R_NO_PRIVATEKEY);
447                 return(-1);
448                 }
449         i=ssl_rsa_private_decrypt(s->cert,s->s2->tmp.enc,
450                 &(p[s->s2->tmp.clear]),&(p[s->s2->tmp.clear]),
451                 (s->s2->ssl2_rollback)?RSA_SSLV23_PADDING:RSA_PKCS1_PADDING);

453         is_export=SSL_C_IS_EXPORT(s->session->cipher);

455         if (!ssl_cipher_get_evp(s->session,&c,&md,NULL,NULL,NULL))
456                 {
457                 ssl2_return_error(s,SSL2_PE_NO_CIPHER);
```



```
458                 SSLerr(SSL_F_GET_CLIENT_MASTER_KEY,SSL_R_PROBLEMS_MAPPING_CIPHER
459                 return(0);
460                 }

462         if (s->session->cipher->algorithm2 & SSL2_CF_8_BYTE_ENC)
463                 {
464                 is_export=1;
465                 ek=8;
466                 }
467         else
468                 ek=5;

470         /* bad decrypt */
471 #if 1
472         /* If a bad decrypt, continue with protocol but with a
473          * random master secret (Bleichenbacher attack) */
474         if ((i < 0) ||
475                 ((!is_export && (i != EVP_CIPHER_key_length(c)))
476                 || (is_export && ((i != ek) || (s->s2->tmp.clear+(unsigned int)i
477                         (unsigned int)EVP_CIPHER_key_length(c))))))
478                 {
479                 ERR_clear_error();
480                 if (is_export)
481                         i=ek;
482                 else
483                         i=EVP_CIPHER_key_length(c);
484                 if (RAND_pseudo_bytes(p,i) <= 0)
485                         return 0;
486                 }
487 #else
488         if (i < 0)
489                 {
490                 error=1;
491                 SSLerr(SSL_F_GET_CLIENT_MASTER_KEY,SSL_R_BAD_RSA_DECRYPT);
492                 }
493         /* incorrect number of key bytes for non export cipher */
494         else if ((!is_export && (i != EVP_CIPHER_key_length(c)))
495                 || (is_export && ((i != ek) || (s->s2->tmp.clear+i !=
496                         EVP_CIPHER_key_length(c)))))
497                 {
498                 error=1;
499                 SSLerr(SSL_F_GET_CLIENT_MASTER_KEY,SSL_R_WRONG_NUMBER_OF_KEY_BIT
500                 }
501         if (error)
502                 {
503                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
504                 return(-1);
505                 }
506 #endif

508         if (is_export) i+=s->s2->tmp.clear;

510         if (i > SSL_MAX_MASTER_KEY_LENGTH)
511                 {
512                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
513                 SSLerr(SSL_F_GET_CLIENT_MASTER_KEY, ERR_R_INTERNAL_ERROR);
514                 return -1;
515                 }
516         s->session->master_key_length=i;
517         memcpy(s->session->master_key,p,(unsigned int)i);
518         return(1);
519         }

521 static int get_client_hello(SSL *s)
522         {
523         int i,n;
```

```
524         unsigned long len;
525         unsigned char *p;
526         STACK_OF(SSL_CIPHER) *cs; /* a stack of SSL_CIPHERS */
527         STACK_OF(SSL_CIPHER) *cl; /* the ones we want to use */
528         STACK_OF(SSL_CIPHER) *prio, *allow;
529         int z;

531         /* This is a bit of a hack to check for the correct packet
532          * type the first time round. */
533         if (s->state == SSL2_ST_GET_CLIENT_HELLO_A)
534                 {
535                 s->first_packet=1;
536                 s->state=SSL2_ST_GET_CLIENT_HELLO_B;
537                 }

539         p=(unsigned char *)s->init_buf->data;
540         if (s->state == SSL2_ST_GET_CLIENT_HELLO_B)
541                 {
542                 i=ssl2_read(s,(char *)&(p[s->init_num]),9-s->init_num);
543                 if (i < (9-s->init_num))
544                         return(ssl2_part_read(s,SSL_F_GET_CLIENT_HELLO,i));
545                 s->init_num = 9;

547                 if (*(p++) != SSL2_MT_CLIENT_HELLO)
548                         {
549                         if (p[-1] != SSL2_MT_ERROR)
550                                 {
551                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
552                                 SSLerr(SSL_F_GET_CLIENT_HELLO,SSL_R_READ_WRONG_P
553                                 }
554                         else
555                                 SSLerr(SSL_F_GET_CLIENT_HELLO,SSL_R_PEER_ERROR);
556                         return(-1);
557                         }
558                 n2s(p,i);
559                 if (i < s->version) s->version=i;
560                 n2s(p,i); s->s2->tmp.cipher_spec_length=i;
561                 n2s(p,i); s->s2->tmp.session_id_length=i;
562                 n2s(p,i); s->s2->challenge_length=i;
563                 if (    (i < SSL2_MIN_CHALLENGE_LENGTH) ||
564                         (i > SSL2_MAX_CHALLENGE_LENGTH))
565                         {
566                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
567                         SSLerr(SSL_F_GET_CLIENT_HELLO,SSL_R_INVALID_CHALLENGE_LE
568                         return(-1);
569                         }
570                 s->state=SSL2_ST_GET_CLIENT_HELLO_C;
571                 }

573         /* SSL2_ST_GET_CLIENT_HELLO_C */
574         p=(unsigned char *)s->init_buf->data;
575         len = 9 + (unsigned long)s->s2->tmp.cipher_spec_length + (unsigned long)
576         if (len > SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER)
577                 {
578                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
579                 SSLerr(SSL_F_GET_CLIENT_HELLO,SSL_R_MESSAGE_TOO_LONG);
580                 return -1;
581                 }
582         n = (int)len - s->init_num;
583         i = ssl2_read(s,(char *)&(p[s->init_num]),n);
584         if (i != n) return(ssl2_part_read(s,SSL_F_GET_CLIENT_HELLO,i));
585         if (s->msg_callback)
586                 s->msg_callback(0, s->version, 0, p, (size_t)len, s, s->msg_call
587         p += 9;

589         /* get session-id before cipher stuff so we can get out session
```



```
590          * structure if it is cached */
591         /* session-id */
592         if ((s->s2->tmp.session_id_length != 0) &&
593                 (s->s2->tmp.session_id_length != SSL2_SSL_SESSION_ID_LENGTH))
594                 {
595                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
596                 SSLerr(SSL_F_GET_CLIENT_HELLO,SSL_R_BAD_SSL_SESSION_ID_LENGTH);
597                 return(-1);
598                 }

600         if (s->s2->tmp.session_id_length == 0)
601                 {
602                 if (!ssl_get_new_session(s,1))
603                         {
604                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
605                         return(-1);
606                         }
607                 }
608         else
609                 {
610                 i=ssl_get_prev_session(s,&(p[s->s2->tmp.cipher_spec_length]),
611                         s->s2->tmp.session_id_length, NULL);
612                 if (i == 1)
613                         { /* previous session */
614                         s->hit=1;
615                         }
616                 else if (i == -1)
617                         {
618                         ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
619                         return(-1);
620                         }
621                 else
622                         {
623                         if (s->cert == NULL)
624                                 {
625                                 ssl2_return_error(s,SSL2_PE_NO_CERTIFICATE);
626                                 SSLerr(SSL_F_GET_CLIENT_HELLO,SSL_R_NO_CERTIFICA
627                                 return(-1);
628                                 }

630                         if (!ssl_get_new_session(s,1))
631                                 {
632                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
633                                 return(-1);
634                                 }
635                         }
636                 }

638         if (!s->hit)
639                 {
640                 cs=ssl_bytes_to_cipher_list(s,p,s->s2->tmp.cipher_spec_length,
641                         &s->session->ciphers);
642                 if (cs == NULL) goto mem_err;

644                 cl=SSL_get_ciphers(s);

646                 if (s->options & SSL_OP_CIPHER_SERVER_PREFERENCE)
647                     {
648                     prio=sk_SSL_CIPHER_dup(cl);
649                     if (prio == NULL) goto mem_err;
650                     allow = cs;
651                     }
652                 else
653                     {
654                     prio = cs;
655                     allow = cl;
```

```
656                     }
657                 for (z=0; z<sk_SSL_CIPHER_num(prio); z++)
658                         {
659                         if (sk_SSL_CIPHER_find(allow,sk_SSL_CIPHER_value(prio,z)
660                                 {
661                                 (void)sk_SSL_CIPHER_delete(prio,z);
662                                 z--;
663                                 }
664                         }
665                 if (s->options & SSL_OP_CIPHER_SERVER_PREFERENCE)
666                     {
667                     sk_SSL_CIPHER_free(s->session->ciphers);
668                     s->session->ciphers = prio;
669                     }
670                 /* s->session->ciphers should now have a list of
671                  * ciphers that are on both the client and server.
672                  * This list is ordered by the order the client sent
673                  * the ciphers or in the order of the server's preference
674                  * if SSL_OP_CIPHER_SERVER_PREFERENCE was set.
675                  */
676                 }
677         p+=s->s2->tmp.cipher_spec_length;
678         /* done cipher selection */
679
680         /* session id extracted already */
681         p+=s->s2->tmp.session_id_length;
682
683         /* challenge */
684         if (s->s2->challenge_length > sizeof s->s2->challenge)
685                 {
686                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
687                 SSLerr(SSL_F_GET_CLIENT_HELLO, ERR_R_INTERNAL_ERROR);
688                 return -1;
689                 }
690         memcpy(s->s2->challenge,p,(unsigned int)s->s2->challenge_length);
691         return(1);
692 mem_err:
693         SSLerr(SSL_F_GET_CLIENT_HELLO,ERR_R_MALLOC_FAILURE);
694         return(0);
695         }
696
697 static int server_hello(SSL *s)
698         {
699         unsigned char *p,*d;
700         int n,hit;
701
702         p=(unsigned char *)s->init_buf->data;
703         if (s->state == SSL2_ST_SEND_SERVER_HELLO_A)
704                 {
705                 d=p+11;
706                 *(p++)=SSL2_MT_SERVER_HELLO;            /* type */
707                 hit=s->hit;
708                 *(p++)=(unsigned char)hit;
709 #if 1
710                 if (!hit)
711                         {
712                         if (s->session->sess_cert != NULL)
713                                 /* This can't really happen because get_client_h
714                                  * has called ssl_get_new_session, which does no
715                                  * sess_cert. */
716                                 ssl_sess_cert_free(s->session->sess_cert);
717                         s->session->sess_cert = ssl_sess_cert_new();
718                         if (s->session->sess_cert == NULL)
719                                 {
720                                 SSLerr(SSL_F_SERVER_HELLO, ERR_R_MALLOC_FAILURE)
721                                 return(-1);
```



```
722                                 }
723                         }
724                 /* If 'hit' is set, then s->sess_cert may be non-NULL or NULL,
725                  * depending on whether it survived in the internal cache
726                  * or was retrieved from an external cache.
727                  * If it is NULL, we cannot put any useful data in it anyway,
728                  * so we don't touch it.
729                  */
730 #else /* That's what used to be done when cert_st and sess_cert_st were
731            * the same. */
732                 if (!hit)
733                         {                       /* else add cert to session */
734                         CRYPTO_add(&s->cert->references,1,CRYPTO_LOCK_SSL_CERT);
735                         if (s->session->sess_cert != NULL)
736                                 ssl_cert_free(s->session->sess_cert);
737                         s->session->sess_cert=s->cert;
738                         }
739                 else    /* We have a session id-cache hit, if the
740                          * session-id has no certificate listed against
741                          * the 'cert' structure, grab the 'old' one
742                          * listed against the SSL connection */
743                         {
744                         if (s->session->sess_cert == NULL)
745                                 {
746                                 CRYPTO_add(&s->cert->references,1,
747                                         CRYPTO_LOCK_SSL_CERT);
748                                 s->session->sess_cert=s->cert;
749                                 }
750                         }
751 #endif
752
753                 if (s->cert == NULL)
754                         {
755                         ssl2_return_error(s,SSL2_PE_NO_CERTIFICATE);
756                         SSLerr(SSL_F_SERVER_HELLO,SSL_R_NO_CERTIFICATE_SPECIFIED
757                         return(-1);
758                         }
759
760                 if (hit)
761                         {
762                         *(p++)=0;               /* no certificate type */
763                         s2n(s->version,p);      /* version */
764                         s2n(0,p);               /* cert len */
765                         s2n(0,p);               /* ciphers len */
766                         }
767                 else
768                         {
769                         /* EAY EAY */
770                         /* put certificate type */
771                         *(p++)=SSL2_CT_X509_CERTIFICATE;
772                         s2n(s->version,p);      /* version */
773                         n=i2d_X509(s->cert->pkeys[SSL_PKEY_RSA_ENC].x509,NULL);
774                         s2n(n,p);               /* certificate length */
775                         i2d_X509(s->cert->pkeys[SSL_PKEY_RSA_ENC].x509,&d);
776                         n=0;
777
778                         /* lets send out the ciphers we like in the
779                          * prefered order */
780                         n=ssl_cipher_list_to_bytes(s,s->session->ciphers,d,0);
781                         d+=n;
782                         s2n(n,p);               /* add cipher length */
783                         }
784
785                 /* make and send conn_id */
786                 s2n(SSL2_CONNECTION_ID_LENGTH,p);       /* add conn_id length */
787
```

```
788                 s->s2->conn_id_length=SSL2_CONNECTION_ID_LENGTH;
789                 if (RAND_pseudo_bytes(s->s2->conn_id,(int)s->s2->conn_id_length)
790                         return -1;
791                 memcpy(d,s->s2->conn_id,SSL2_CONNECTION_ID_LENGTH);
792                 d+=SSL2_CONNECTION_ID_LENGTH;

794                 s->state=SSL2_ST_SEND_SERVER_HELLO_B;
795                 s->init_num=d-(unsigned char *)s->init_buf->data;
796                 s->init_off=0;
797                 }
798         /* SSL2_ST_SEND_SERVER_HELLO_B */
799         /* If we are using TCP/IP, the performance is bad if we do 2
800          * writes without a read between them.  This occurs when
801          * Session-id reuse is used, so I will put in a buffering module
802          */
803         if (s->hit)
804                 {
805                 if (!ssl_init_wbio_buffer(s,1)) return(-1);
806                 }

808         return(ssl2_do_write(s));
809         }

811 static int get_client_finished(SSL *s)
812         {
813         unsigned char *p;
814         int i, n;
815         unsigned long len;

817         p=(unsigned char *)s->init_buf->data;
818         if (s->state == SSL2_ST_GET_CLIENT_FINISHED_A)
819                 {
820                 i=ssl2_read(s,(char *)&(p[s->init_num]),1-s->init_num);
821                 if (i < 1-s->init_num)
822                         return(ssl2_part_read(s,SSL_F_GET_CLIENT_FINISHED,i));
823                 s->init_num += i;

825                 if (*p != SSL2_MT_CLIENT_FINISHED)
826                         {
827                         if (*p != SSL2_MT_ERROR)
828                                 {
829                                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
830                                 SSLerr(SSL_F_GET_CLIENT_FINISHED,SSL_R_READ_WRON
831                                 }
832                         else
833                                 {
834                                 SSLerr(SSL_F_GET_CLIENT_FINISHED,SSL_R_PEER_ERRO
835                                 /* try to read the error message */
836                                 i=ssl2_read(s,(char *)&(p[s->init_num]),3-s->ini
837                                 return ssl2_part_read(s,SSL_F_GET_SERVER_VERIFY,
838                                 }
839                         return(-1);
840                         }
841                 s->state=SSL2_ST_GET_CLIENT_FINISHED_B;
842                 }

844         /* SSL2_ST_GET_CLIENT_FINISHED_B */
845         if (s->s2->conn_id_length > sizeof s->s2->conn_id)
846                 {
847                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
848                 SSLerr(SSL_F_GET_CLIENT_FINISHED, ERR_R_INTERNAL_ERROR);
849                 return -1;
850                 }
851         len = 1 + (unsigned long)s->s2->conn_id_length;
852         n = (int)len - s->init_num;
853         i = ssl2_read(s,(char *)&(p[s->init_num]),n);
```



```
854         if (i < n)
855                 {
856                 return(ssl2_part_read(s,SSL_F_GET_CLIENT_FINISHED,i));
857                 }
858         if (s->msg_callback)
859                 s->msg_callback(0, s->version, 0, p, len, s, s->msg_callback_arg
860         p += 1;
861         if (memcmp(p,s->s2->conn_id,s->s2->conn_id_length) != 0)
862                 {
863                 ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
864                 SSLerr(SSL_F_GET_CLIENT_FINISHED,SSL_R_CONNECTION_ID_IS_DIFFEREN
865                 return(-1);
866                 }
867         return(1);
868         }

870 static int server_verify(SSL *s)
871         {
872         unsigned char *p;

874         if (s->state == SSL2_ST_SEND_SERVER_VERIFY_A)
875                 {
876                 p=(unsigned char *)s->init_buf->data;
877                 *(p++)=SSL2_MT_SERVER_VERIFY;
878                 if (s->s2->challenge_length > sizeof s->s2->challenge)
879                         {
880                         SSLerr(SSL_F_SERVER_VERIFY, ERR_R_INTERNAL_ERROR);
881                         return -1;
882                         }
883                 memcpy(p,s->s2->challenge,(unsigned int)s->s2->challenge_length)
884                 /* p+=s->s2->challenge_length; */

886                 s->state=SSL2_ST_SEND_SERVER_VERIFY_B;
887                 s->init_num=s->s2->challenge_length+1;
888                 s->init_off=0;
889                 }
890         return(ssl2_do_write(s));
891         }

893 static int server_finish(SSL *s)
894         {
895         unsigned char *p;

897         if (s->state == SSL2_ST_SEND_SERVER_FINISHED_A)
898                 {
899                 p=(unsigned char *)s->init_buf->data;
900                 *(p++)=SSL2_MT_SERVER_FINISHED;

902                 if (s->session->session_id_length > sizeof s->session->session_i
903                         {
904                         SSLerr(SSL_F_SERVER_FINISH, ERR_R_INTERNAL_ERROR);
905                         return -1;
906                         }
907                 memcpy(p,s->session->session_id, (unsigned int)s->session->sessi
908                 /* p+=s->session->session_id_length; */

910                 s->state=SSL2_ST_SEND_SERVER_FINISHED_B;
911                 s->init_num=s->session->session_id_length+1;
912                 s->init_off=0;
913                 }

915         /* SSL2_ST_SEND_SERVER_FINISHED_B */
916         return(ssl2_do_write(s));
917         }

919 /* send the request and check the response */
```

```
static int request_certificate(SSL *s)
        {
        const unsigned char *cp;
        unsigned char *p,*p2,*buf2;
        unsigned char *ccd;
        int i,j,ctype,ret= -1;
        unsigned long len;
        X509 *x509=NULL;
        STACK_OF(X509) *sk=NULL;

        ccd=s->s2->tmp.ccl;
        if (s->state == SSL2_ST_SEND_REQUEST_CERTIFICATE_A)
                {
                p=(unsigned char *)s->init_buf->data;
                *(p++)=SSL2_MT_REQUEST_CERTIFICATE;
                *(p++)=SSL2_AT_MD5_WITH_RSA_ENCRYPTION;
                if (RAND_pseudo_bytes(ccd,SSL2_MIN_CERT_CHALLENGE_LENGTH) <= 0)
                        return -1;
                memcpy(p,ccd,SSL2_MIN_CERT_CHALLENGE_LENGTH);

                s->state=SSL2_ST_SEND_REQUEST_CERTIFICATE_B;
                s->init_num=SSL2_MIN_CERT_CHALLENGE_LENGTH+2;
                s->init_off=0;
                }

        if (s->state == SSL2_ST_SEND_REQUEST_CERTIFICATE_B)
                {
                i=ssl2_do_write(s);
                if (i <= 0)
                        {
                        ret=i;
                        goto end;
                        }

                s->init_num=0;
                s->state=SSL2_ST_SEND_REQUEST_CERTIFICATE_C;
                }

        if (s->state == SSL2_ST_SEND_REQUEST_CERTIFICATE_C)
                {
                p=(unsigned char *)s->init_buf->data;
                i=ssl2_read(s,(char *)&(p[s->init_num]),6-s->init_num); /* try t
                if (i < 3-s->init_num) /* ... but don't call ssl2_part_read now
                                       * (probably NO-CERTIFICATE-ERROR) */
                        {
                        ret=ssl2_part_read(s,SSL_F_REQUEST_CERTIFICATE,i);
                        goto end;
                        }
                s->init_num += i;

                if ((s->init_num >= 3) && (p[0] == SSL2_MT_ERROR))
                        {
                        n2s(p,i);
                        if (i != SSL2_PE_NO_CERTIFICATE)
                                {
                                /* not the error message we expected -- let ssl2
                                s->init_num -= 3;
                                ret = ssl2_part_read(s,SSL_F_REQUEST_CERTIFICATE
                                goto end;
                                }

                        if (s->msg_callback)
                                s->msg_callback(0, s->version, 0, p, 3, s, s->ms

                        /* this is the one place where we can recover from an SS
```



```
                        if (s->verify_mode & SSL_VERIFY_FAIL_IF_NO_PEER_CERT)
                                {
                                ssl2_return_error(s,SSL2_PE_BAD_CERTIFICATE);
                                SSLerr(SSL_F_REQUEST_CERTIFICATE,SSL_R_PEER_DID_
                                goto end;
                                }
                        ret=1;
                        goto end;
                        }
                if ((*(p++) != SSL2_MT_CLIENT_CERTIFICATE) || (s->init_num < 6))
                        {
                        ssl2_return_error(s,SSL2_PE_UNDEFINED_ERROR);
                        SSLerr(SSL_F_REQUEST_CERTIFICATE,SSL_R_SHORT_READ);
                        goto end;
                        }
                if (s->init_num != 6)
                        {
                        SSLerr(SSL_F_REQUEST_CERTIFICATE, ERR_R_INTERNAL_ERROR);
                        goto end;
                        }

                /* ok we have a response */
                /* certificate type, there is only one right now. */
                ctype= *(p++);
                if (ctype != SSL2_AT_MD5_WITH_RSA_ENCRYPTION)
                        {
                        ssl2_return_error(s,SSL2_PE_UNSUPPORTED_CERTIFICATE_TYPE
                        SSLerr(SSL_F_REQUEST_CERTIFICATE,SSL_R_BAD_RESPONSE_ARGU
                        goto end;
                        }
                n2s(p,i); s->s2->tmp.clen=i;
                n2s(p,i); s->s2->tmp.rlen=i;
                s->state=SSL2_ST_SEND_REQUEST_CERTIFICATE_D;
                }

        /* SSL2_ST_SEND_REQUEST_CERTIFICATE_D */
        p=(unsigned char *)s->init_buf->data;
        len = 6 + (unsigned long)s->s2->tmp.clen + (unsigned long)s->s2->tmp.rle
        if (len > SSL2_MAX_RECORD_LENGTH_3_BYTE_HEADER)
                {
                SSLerr(SSL_F_REQUEST_CERTIFICATE,SSL_R_MESSAGE_TOO_LONG);
                goto end;
                }
        j = (int)len - s->init_num;
        i = ssl2_read(s,(char *)&(p[s->init_num]),j);
        if (i < j)
                {
                ret=ssl2_part_read(s,SSL_F_REQUEST_CERTIFICATE,i);
                goto end;
                }
        if (s->msg_callback)
                s->msg_callback(0, s->version, 0, p, len, s, s->msg_callback_arg
        p += 6;

        cp = p;
        x509=(X509 *)d2i_X509(NULL,&cp,(long)s->s2->tmp.clen);
        if (x509 == NULL)
                {
                SSLerr(SSL_F_REQUEST_CERTIFICATE,ERR_R_X509_LIB);
                goto msg_end;
                }

        if (((sk=sk_X509_new_null()) == NULL) || (!sk_X509_push(sk,x509)))
                {
                SSLerr(SSL_F_REQUEST_CERTIFICATE,ERR_R_MALLOC_FAILURE);
                goto msg_end;
```

```
1052                 }

1054         i=ssl_verify_cert_chain(s,sk);

1056         if (i > 0)      /* we like the packet, now check the chksum */
1057                 {
1058                 EVP_MD_CTX ctx;
1059                 EVP_PKEY *pkey=NULL;

1061                 EVP_MD_CTX_init(&ctx);
1062                 if (!EVP_VerifyInit_ex(&ctx,s->ctx->rsa_md5, NULL)
1063                     || !EVP_VerifyUpdate(&ctx,s->s2->key_material,
1064                                          s->s2->key_material_length)
1065                     || !EVP_VerifyUpdate(&ctx,ccd,
1066                                          SSL2_MIN_CERT_CHALLENGE_LENGTH))
1067                         goto msg_end;

1069                 i=i2d_X509(s->cert->pkeys[SSL_PKEY_RSA_ENC].x509,NULL);
1070                 buf2=OPENSSL_malloc((unsigned int)i);
1071                 if (buf2 == NULL)
1072                         {
1073                         SSLerr(SSL_F_REQUEST_CERTIFICATE,ERR_R_MALLOC_FAILURE);
1074                         goto msg_end;
1075                         }
1076                 p2=buf2;
1077                 i=i2d_X509(s->cert->pkeys[SSL_PKEY_RSA_ENC].x509,&p2);
1078                 if (!EVP_VerifyUpdate(&ctx,buf2,(unsigned int)i))
1079                         {
1080                         OPENSSL_free(buf2);
1081                         goto msg_end;
1082                         }
1083                 OPENSSL_free(buf2);

1085                 pkey=X509_get_pubkey(x509);
1086                 if (pkey == NULL) goto end;
1087                 i=EVP_VerifyFinal(&ctx,cp,s->s2->tmp.rlen,pkey);
1088                 EVP_PKEY_free(pkey);
1089                 EVP_MD_CTX_cleanup(&ctx);

1091                 if (i > 0)
1092                         {
1093                         if (s->session->peer != NULL)
1094                                 X509_free(s->session->peer);
1095                         s->session->peer=x509;
1096                         CRYPTO_add(&x509->references,1,CRYPTO_LOCK_X509);
1097                         s->session->verify_result = s->verify_result;
1098                         ret=1;
1099                         goto end;
1100                         }
1101                 else
1102                         {
1103                         SSLerr(SSL_F_REQUEST_CERTIFICATE,SSL_R_BAD_CHECKSUM);
1104                         goto msg_end;
1105                         }
1106                 }
1107         else
1108                 {
1109 msg_end:
1110                 ssl2_return_error(s,SSL2_PE_BAD_CERTIFICATE);
1111                 }
1112 end:
1113         sk_X509_free(sk);
1114         X509_free(x509);
1115         return(ret);
1116         }
```



```
1118 static int ssl_rsa_private_decrypt(CERT *c, int len, unsigned char *from,
1119                                    unsigned char *to, int padding)
1120         {
1121         RSA *rsa;
1122         int i;

1124         if ((c == NULL) || (c->pkeys[SSL_PKEY_RSA_ENC].privatekey == NULL))
1125                 {
1126                 SSLerr(SSL_F_SSL_RSA_PRIVATE_DECRYPT,SSL_R_NO_PRIVATEKEY);
1127                 return(-1);
1128                 }
1129         if (c->pkeys[SSL_PKEY_RSA_ENC].privatekey->type != EVP_PKEY_RSA)
1130                 {
1131                 SSLerr(SSL_F_SSL_RSA_PRIVATE_DECRYPT,SSL_R_PUBLIC_KEY_IS_NOT_RSA
1132                 return(-1);
1133                 }
1134         rsa=c->pkeys[SSL_PKEY_RSA_ENC].privatekey->pkey.rsa;

1136         /* we have the public key */
1137         i=RSA_private_decrypt(len,from,to,rsa,padding);
1138         if (i < 0)
1139                 SSLerr(SSL_F_SSL_RSA_PRIVATE_DECRYPT,ERR_R_RSA_LIB);
1140         return(i);
1141         }
1142 #else /* !OPENSSL_NO_SSL2 */

1144 # if PEDANTIC
1145 static void *dummy=&dummy;
1146 # endif

1148 #endif
1149 #endif /* ! codereview */
```

```
**********************************************************
   24091 Wed Aug 13 19:53:38 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_both.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s3_both.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECC cipher suite support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */

#include <limits.h>
#include <string.h>
#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>

/* send s->init_buf in records of type 'type' (SSL3_RT_HANDSHAKE or SSL3_RT_CHAN
```

```
128 int ssl3_do_write(SSL *s, int type)
129         {
130         int ret;

132         ret=ssl3_write_bytes(s,type,&s->init_buf->data[s->init_off],
133                              s->init_num);
134         if (ret < 0) return(-1);
135         if (type == SSL3_RT_HANDSHAKE)
136                 /* should not be done for 'Hello Request's, but in that case
137                  * we'll ignore the result anyway */
138                 ssl3_finish_mac(s,(unsigned char *)&s->init_buf->data[s->init_of

140         if (ret == s->init_num)
141                 {
142                 if (s->msg_callback)
143                         s->msg_callback(1, s->version, type, s->init_buf->data,
144                 return(1);
145                 }
146         s->init_off+=ret;
147         s->init_num-=ret;
148         return(0);
149         }

151 int ssl3_send_finished(SSL *s, int a, int b, const char *sender, int slen)
152         {
153         unsigned char *p,*d;
154         int i;
155         unsigned long l;

157         if (s->state == a)
158                 {
159                 d=(unsigned char *)s->init_buf->data;
160                 p= &(d[4]);

162                 i=s->method->ssl3_enc->final_finish_mac(s,
163                         sender,slen,s->s3->tmp.finish_md);
164                 if (i == 0)
165                         return 0;
166                 s->s3->tmp.finish_md_len = i;
167                 memcpy(p, s->s3->tmp.finish_md, i);
168                 p+=i;
169                 l=i;

171                 /* Copy the finished so we can use it for
172                    renegotiation checks */
173                 if(s->type == SSL_ST_CONNECT)
174                         {
175                         OPENSSL_assert(i <= EVP_MAX_MD_SIZE);
176                         memcpy(s->s3->previous_client_finished,
177                             s->s3->tmp.finish_md, i);
178                         s->s3->previous_client_finished_len=i;
179                         }
180                 else
181                         {
182                         OPENSSL_assert(i <= EVP_MAX_MD_SIZE);
183                         memcpy(s->s3->previous_server_finished,
184                             s->s3->tmp.finish_md, i);
185                         s->s3->previous_server_finished_len=i;
186                         }

188 #ifdef OPENSSL_SYS_WIN16
189                 /* MSVC 1.5 does not clear the top bytes of the word unless
190                  * I do this.
191                  */
192                 l&=0xffff;
193 #endif
```



```
195                 *(d++)=SSL3_MT_FINISHED;
196                 l2n3(l,d);
197                 s->init_num=(int)l+4;
198                 s->init_off=0;

200                 s->state=b;
201                 }

203         /* SSL3_ST_SEND_xxxxxx_HELLO_B */
204         return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
205         }

207 #ifndef OPENSSL_NO_NEXTPROTONEG
208 /* ssl3_take_mac calculates the Finished MAC for the handshakes messages seen to
209 static void ssl3_take_mac(SSL *s)
210         {
211         const char *sender;
212         int slen;
213         /* If no new cipher setup return immediately: other functions will
214          * set the appropriate error.
215          */
216         if (s->s3->tmp.new_cipher == NULL)
217                 return;
218         if (s->state & SSL_ST_CONNECT)
219                 {
220                 sender=s->method->ssl3_enc->server_finished_label;
221                 slen=s->method->ssl3_enc->server_finished_label_len;
222                 }
223         else
224                 {
225                 sender=s->method->ssl3_enc->client_finished_label;
226                 slen=s->method->ssl3_enc->client_finished_label_len;
227                 }

229         s->s3->tmp.peer_finish_md_len = s->method->ssl3_enc->final_finish_mac(s,
230                 sender,slen,s->s3->tmp.peer_finish_md);
231         }
232 #endif

234 int ssl3_get_finished(SSL *s, int a, int b)
235         {
236         int al,i,ok;
237         long n;
238         unsigned char *p;

240 #ifdef OPENSSL_NO_NEXTPROTONEG
241         /* the mac has already been generated when we received the
242          * change cipher spec message and is in s->s3->tmp.peer_finish_md.
243          */
244 #endif

246         n=s->method->ssl_get_message(s,
247                 a,
248                 b,
249                 SSL3_MT_FINISHED,
250                 64, /* should actually be 36+4 :-) */
251                 &ok);

253         if (!ok) return((int)n);

255         /* If this occurs, we have missed a message */
256         if (!s->s3->change_cipher_spec)
257                 {
258                 al=SSL_AD_UNEXPECTED_MESSAGE;
259                 SSLerr(SSL_F_SSL3_GET_FINISHED,SSL_R_GOT_A_FIN_BEFORE_A_CCS);
```

```
260                 goto f_err;
261                 }
262         s->s3->change_cipher_spec=0;

264         p = (unsigned char *)s->init_msg;
265         i = s->s3->tmp.peer_finish_md_len;

267         if (i != n)
268                 {
269                 al=SSL_AD_DECODE_ERROR;
270                 SSLerr(SSL_F_SSL3_GET_FINISHED,SSL_R_BAD_DIGEST_LENGTH);
271                 goto f_err;
272                 }

274         if (CRYPTO_memcmp(p, s->s3->tmp.peer_finish_md, i) != 0)
275                 {
276                 al=SSL_AD_DECRYPT_ERROR;
277                 SSLerr(SSL_F_SSL3_GET_FINISHED,SSL_R_DIGEST_CHECK_FAILED);
278                 goto f_err;
279                 }

281         /* Copy the finished so we can use it for
282            renegotiation checks */
283         if(s->type == SSL_ST_ACCEPT)
284                 {
285                 OPENSSL_assert(i <= EVP_MAX_MD_SIZE);
286                 memcpy(s->s3->previous_client_finished,
287                     s->s3->tmp.peer_finish_md, i);
288                 s->s3->previous_client_finished_len=i;
289                 }
290         else
291                 {
292                 OPENSSL_assert(i <= EVP_MAX_MD_SIZE);
293                 memcpy(s->s3->previous_server_finished,
294                     s->s3->tmp.peer_finish_md, i);
295                 s->s3->previous_server_finished_len=i;
296                 }

298         return(1);
299 f_err:
300         ssl3_send_alert(s,SSL3_AL_FATAL,al);
301         return(0);
302         }

304 /* for these 2 messages, we need to
305  * ssl->enc_read_ctx                    re-init
306  * ssl->s3->read_sequence               zero
307  * ssl->s3->read_mac_secret             re-init
308  * ssl->session->read_sym_enc           assign
309  * ssl->session->read_compression       assign
310  * ssl->session->read_hash              assign
311  */
312 int ssl3_send_change_cipher_spec(SSL *s, int a, int b)
313         {
314         unsigned char *p;

316         if (s->state == a)
317                 {
318                 p=(unsigned char *)s->init_buf->data;
319                 *p=SSL3_MT_CCS;
320                 s->init_num=1;
321                 s->init_off=0;

323                 s->state=b;
324                 }
```



```
326         /* SSL3_ST_CW_CHANGE_B */
327         return(ssl3_do_write(s,SSL3_RT_CHANGE_CIPHER_SPEC));
328         }

330 static int ssl3_add_cert_to_buf(BUF_MEM *buf, unsigned long *l, X509 *x)
331         {
332         int n;
333         unsigned char *p;

335         n=i2d_X509(x,NULL);
336         if (!BUF_MEM_grow_clean(buf,(int)(n+(*l)+3)))
337                 {
338                 SSLerr(SSL_F_SSL3_ADD_CERT_TO_BUF,ERR_R_BUF_LIB);
339                 return(-1);
340                 }
341         p=(unsigned char *)&(buf->data[*l]);
342         l2n3(n,p);
343         i2d_X509(x,&p);
344         *l+=n+3;

346         return(0);
347         }

349 unsigned long ssl3_output_cert_chain(SSL *s, X509 *x)
350         {
351         unsigned char *p;
352         int i;
353         unsigned long l=7;
354         BUF_MEM *buf;
355         int no_chain;

357         if ((s->mode & SSL_MODE_NO_AUTO_CHAIN) || s->ctx->extra_certs)
358                 no_chain = 1;
359         else
360                 no_chain = 0;

362         /* TLSv1 sends a chain with nothing in it, instead of an alert */
363         buf=s->init_buf;
364         if (!BUF_MEM_grow_clean(buf,10))
365                 {
366                 SSLerr(SSL_F_SSL3_OUTPUT_CERT_CHAIN,ERR_R_BUF_LIB);
367                 return(0);
368                 }
369         if (x != NULL)
370                 {
371                 if (no_chain)
372                         {
373                         if (ssl3_add_cert_to_buf(buf, &l, x))
374                                 return(0);
375                         }
376                 else
377                         {
378                         X509_STORE_CTX xs_ctx;

380                         if (!X509_STORE_CTX_init(&xs_ctx,s->ctx->cert_store,x,NU
381                                 {
382                                 SSLerr(SSL_F_SSL3_OUTPUT_CERT_CHAIN,ERR_R_X509_L
383                                 return(0);
384                                 }
385                         X509_verify_cert(&xs_ctx);
386                         /* Don't leave errors in the queue */
387                         ERR_clear_error();
388                         for (i=0; i < sk_X509_num(xs_ctx.chain); i++)
389                                 {
390                                 x = sk_X509_value(xs_ctx.chain, i);
```

```
392                                 if (ssl3_add_cert_to_buf(buf, &l, x))
393                                         {
394                                         X509_STORE_CTX_cleanup(&xs_ctx);
395                                         return 0;
396                                         }
397                                 }
398                         X509_STORE_CTX_cleanup(&xs_ctx);
399                         }
400                 }
401         /* Thawte special :-) */
402         for (i=0; i<sk_X509_num(s->ctx->extra_certs); i++)
403                 {
404                 x=sk_X509_value(s->ctx->extra_certs,i);
405                 if (ssl3_add_cert_to_buf(buf, &l, x))
406                         return(0);
407                 }
408
409         l-=7;
410         p=(unsigned char *)&(buf->data[4]);
411         l2n3(l,p);
412         l+=3;
413         p=(unsigned char *)&(buf->data[0]);
414         *(p++)=SSL3_MT_CERTIFICATE;
415         l2n3(l,p);
416         l+=4;
417         return(l);
418         }
419
420 /* Obtain handshake message of message type 'mt' (any if mt == -1),
421  * maximum acceptable body length 'max'.
422  * The first four bytes (msg_type and length) are read in state 'st1',
423  * the body is read in state 'stn'.
424  */
425 long ssl3_get_message(SSL *s, int st1, int stn, int mt, long max, int *ok)
426         {
427         unsigned char *p;
428         unsigned long l;
429         long n;
430         int i,al;
431
432         if (s->s3->tmp.reuse_message)
433                 {
434                 s->s3->tmp.reuse_message=0;
435                 if ((mt >= 0) && (s->s3->tmp.message_type != mt))
436                         {
437                         al=SSL_AD_UNEXPECTED_MESSAGE;
438                         SSLerr(SSL_F_SSL3_GET_MESSAGE,SSL_R_UNEXPECTED_MESSAGE);
439                         goto f_err;
440                         }
441                 *ok=1;
442                 s->init_msg = s->init_buf->data + 4;
443                 s->init_num = (int)s->s3->tmp.message_size;
444                 return s->init_num;
445                 }
446
447         p=(unsigned char *)s->init_buf->data;
448
449         if (s->state == st1) /* s->init_num < 4 */
450                 {
451                 int skip_message;
452
453                 do
454                         {
455                         while (s->init_num < 4)
456                                 {
457                                 i=s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,
```



```
458                                         &p[s->init_num],4 - s->init_num, 0);
459                                 if (i <= 0)
460                                         {
461                                         s->rwstate=SSL_READING;
462                                         *ok = 0;
463                                         return i;
464                                         }
465                                 s->init_num+=i;
466                                 }
467
468                         skip_message = 0;
469                         if (!s->server)
470                                 if (p[0] == SSL3_MT_HELLO_REQUEST)
471                                         /* The server may always send 'Hello Req
472                                          * we are doing a handshake anyway now,
473                                          * if their format is correct. Does not
474                                          * 'Finished' MAC. */
475                                         if (p[1] == 0 && p[2] == 0 &&p[3] == 0)
476                                                 {
477                                                 s->init_num = 0;
478                                                 skip_message = 1;
479
480                                                 if (s->msg_callback)
481                                                         s->msg_callback(0, s->ve
482                                                 }
483                         }
484                 while (skip_message);
485
486                 /* s->init_num == 4 */
487
488                 if ((mt >= 0) && (*p != mt))
489                         {
490                         al=SSL_AD_UNEXPECTED_MESSAGE;
491                         SSLerr(SSL_F_SSL3_GET_MESSAGE,SSL_R_UNEXPECTED_MESSAGE);
492                         goto f_err;
493                         }
494                 if ((mt < 0) && (*p == SSL3_MT_CLIENT_HELLO) &&
495                                         (st1 == SSL3_ST_SR_CERT_A) &&
496                                         (stn == SSL3_ST_SR_CERT_B))
497                         {
498                         /* At this point we have got an MS SGC second client
499                          * hello (maybe we should always allow the client to
500                          * start a new handshake?). We need to restart the mac.
501                          * Don't increment {num,total}_renegotiations because
502                          * we have not completed the handshake. */
503                         ssl3_init_finished_mac(s);
504                         }
505
506                 s->s3->tmp.message_type= *(p++);
507
508                 n2l3(p,l);
509                 if (l > (unsigned long)max)
510                         {
511                         al=SSL_AD_ILLEGAL_PARAMETER;
512                         SSLerr(SSL_F_SSL3_GET_MESSAGE,SSL_R_EXCESSIVE_MESSAGE_SI
513                         goto f_err;
514                         }
515                 if (l > (INT_MAX-4)) /* BUF_MEM_grow takes an 'int' parameter */
516                         {
517                         al=SSL_AD_ILLEGAL_PARAMETER;
518                         SSLerr(SSL_F_SSL3_GET_MESSAGE,SSL_R_EXCESSIVE_MESSAGE_SI
519                         goto f_err;
520                         }
521                 if (l && !BUF_MEM_grow_clean(s->init_buf,(int)l+4))
522                         {
523                         SSLerr(SSL_F_SSL3_GET_MESSAGE,ERR_R_BUF_LIB);
```

```
524                         goto err;
525                         }
526                 s->s3->tmp.message_size=l;
527                 s->state=stn;
528
529                 s->init_msg = s->init_buf->data + 4;
530                 s->init_num = 0;
531                 }
532
533         /* next state (stn) */
534         p = s->init_msg;
535         n = s->s3->tmp.message_size - s->init_num;
536         while (n > 0)
537                 {
538                 i=s->method->ssl_read_bytes(s,SSL3_RT_HANDSHAKE,&p[s->init_num],
539                 if (i <= 0)
540                         {
541                         s->rwstate=SSL_READING;
542                         *ok = 0;
543                         return i;
544                         }
545                 s->init_num += i;
546                 n -= i;
547                 }
548
549 #ifndef OPENSSL_NO_NEXTPROTONEG
550         /* If receiving Finished, record MAC of prior handshake messages for
551          * Finished verification. */
552         if (*s->init_buf->data == SSL3_MT_FINISHED)
553                 ssl3_take_mac(s);
554 #endif
555
556         /* Feed this message into MAC computation. */
557         ssl3_finish_mac(s, (unsigned char *)s->init_buf->data, s->init_num + 4);
558         if (s->msg_callback)
559                 s->msg_callback(0, s->version, SSL3_RT_HANDSHAKE, s->init_buf->d
560         *ok=1;
561         return s->init_num;
562 f_err:
563         ssl3_send_alert(s,SSL3_AL_FATAL,al);
564 err:
565         *ok=0;
566         return(-1);
567         }
568
569 int ssl_cert_type(X509 *x, EVP_PKEY *pkey)
570         {
571         EVP_PKEY *pk;
572         int ret= -1,i;
573
574         if (pkey == NULL)
575                 pk=X509_get_pubkey(x);
576         else
577                 pk=pkey;
578         if (pk == NULL) goto err;
579
580         i=pk->type;
581         if (i == EVP_PKEY_RSA)
582                 {
583                 ret=SSL_PKEY_RSA_ENC;
584                 }
585         else if (i == EVP_PKEY_DSA)
586                 {
587                 ret=SSL_PKEY_DSA_SIGN;
588                 }
589 #ifndef OPENSSL_NO_EC
```



```
590         else if (i == EVP_PKEY_EC)
591                 {
592                 ret = SSL_PKEY_ECC;
593                 }
594 #endif
595         else if (i == NID_id_GostR3410_94 || i == NID_id_GostR3410_94_cc)
596                 {
597                 ret = SSL_PKEY_GOST94;
598                 }
599         else if (i == NID_id_GostR3410_2001 || i == NID_id_GostR3410_2001_cc)
600                 {
601                 ret = SSL_PKEY_GOST01;
602                 }
603 err:
604         if(!pkey) EVP_PKEY_free(pk);
605         return(ret);
606         }
607
608 int ssl_verify_alarm_type(long type)
609         {
610         int al;
611
612         switch(type)
613                 {
614         case X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT:
615         case X509_V_ERR_UNABLE_TO_GET_CRL:
616         case X509_V_ERR_UNABLE_TO_GET_CRL_ISSUER:
617                 al=SSL_AD_UNKNOWN_CA;
618                 break;
619         case X509_V_ERR_UNABLE_TO_DECRYPT_CERT_SIGNATURE:
620         case X509_V_ERR_UNABLE_TO_DECRYPT_CRL_SIGNATURE:
621         case X509_V_ERR_UNABLE_TO_DECODE_ISSUER_PUBLIC_KEY:
622         case X509_V_ERR_ERROR_IN_CERT_NOT_BEFORE_FIELD:
623         case X509_V_ERR_ERROR_IN_CERT_NOT_AFTER_FIELD:
624         case X509_V_ERR_ERROR_IN_CRL_LAST_UPDATE_FIELD:
625         case X509_V_ERR_ERROR_IN_CRL_NEXT_UPDATE_FIELD:
626         case X509_V_ERR_CERT_NOT_YET_VALID:
627         case X509_V_ERR_CRL_NOT_YET_VALID:
628         case X509_V_ERR_CERT_UNTRUSTED:
629         case X509_V_ERR_CERT_REJECTED:
630                 al=SSL_AD_BAD_CERTIFICATE;
631                 break;
632         case X509_V_ERR_CERT_SIGNATURE_FAILURE:
633         case X509_V_ERR_CRL_SIGNATURE_FAILURE:
634                 al=SSL_AD_DECRYPT_ERROR;
635                 break;
636         case X509_V_ERR_CERT_HAS_EXPIRED:
637         case X509_V_ERR_CRL_HAS_EXPIRED:
638                 al=SSL_AD_CERTIFICATE_EXPIRED;
639                 break;
640         case X509_V_ERR_CERT_REVOKED:
641                 al=SSL_AD_CERTIFICATE_REVOKED;
642                 break;
643         case X509_V_ERR_OUT_OF_MEM:
644                 al=SSL_AD_INTERNAL_ERROR;
645                 break;
646         case X509_V_ERR_DEPTH_ZERO_SELF_SIGNED_CERT:
647         case X509_V_ERR_SELF_SIGNED_CERT_IN_CHAIN:
648         case X509_V_ERR_UNABLE_TO_GET_ISSUER_CERT_LOCALLY:
649         case X509_V_ERR_UNABLE_TO_VERIFY_LEAF_SIGNATURE:
650         case X509_V_ERR_CERT_CHAIN_TOO_LONG:
651         case X509_V_ERR_PATH_LENGTH_EXCEEDED:
652         case X509_V_ERR_INVALID_CA:
653                 al=SSL_AD_UNKNOWN_CA;
654                 break;
655         case X509_V_ERR_APPLICATION_VERIFICATION:
```

```
656                 al=SSL_AD_HANDSHAKE_FAILURE;
657                 break;
658         case X509_V_ERR_INVALID_PURPOSE:
659                 al=SSL_AD_UNSUPPORTED_CERTIFICATE;
660                 break;
661         default:
662                 al=SSL_AD_CERTIFICATE_UNKNOWN;
663                 break;
664                 }
665         return(al);
666         }

668 #ifndef OPENSSL_NO_BUF_FREELISTS
669 /* On some platforms, malloc() performance is bad enough that you can't just
670  * free() and malloc() buffers all the time, so we need to use freelists from
671  * unused buffers.  Currently, each freelist holds memory chunks of only a
672  * given size (list->chunklen); other sized chunks are freed and malloced.
673  * This doesn't help much if you're using many different SSL option settings
674  * with a given context.  (The options affecting buffer size are
675  * max_send_fragment, read buffer vs write buffer,
676  * SSL_OP_MICROSOFT_BIG_WRITE_BUFFER, SSL_OP_NO_COMPRESSION, and
677  * SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS.)  Using a separate freelist for every
678  * possible size is not an option, since max_send_fragment can take on many
679  * different values.
680  *
681  * If you are on a platform with a slow malloc(), and you're using SSL
682  * connections with many different settings for these options, and you need to
683  * use the SSL_MOD_RELEASE_BUFFERS feature, you have a few options:
684  *    - Link against a faster malloc implementation.
685  *    - Use a separate SSL_CTX for each option set.
686  *    - Improve this code.
687  */
688 static void *
689 freelist_extract(SSL_CTX *ctx, int for_read, int sz)
690         {
691         SSL3_BUF_FREELIST *list;
692         SSL3_BUF_FREELIST_ENTRY *ent = NULL;
693         void *result = NULL;

695         CRYPTO_w_lock(CRYPTO_LOCK_SSL_CTX);
696         list = for_read ? ctx->rbuf_freelist : ctx->wbuf_freelist;
697         if (list != NULL && sz == (int)list->chunklen)
698                 ent = list->head;
699         if (ent != NULL)
700                 {
701                 list->head = ent->next;
702                 result = ent;
703                 if (--list->len == 0)
704                         list->chunklen = 0;
705                 }
706         CRYPTO_w_unlock(CRYPTO_LOCK_SSL_CTX);
707         if (!result)
708                 result = OPENSSL_malloc(sz);
709         return result;
710 }

712 static void
713 freelist_insert(SSL_CTX *ctx, int for_read, size_t sz, void *mem)
714         {
715         SSL3_BUF_FREELIST *list;
716         SSL3_BUF_FREELIST_ENTRY *ent;

718         CRYPTO_w_lock(CRYPTO_LOCK_SSL_CTX);
719         list = for_read ? ctx->rbuf_freelist : ctx->wbuf_freelist;
720         if (list != NULL &&
721             (sz == list->chunklen || list->chunklen == 0) &&
```



```
722             list->len < ctx->freelist_max_len &&
723             sz >= sizeof(*ent))
724                 {
725                 list->chunklen = sz;
726                 ent = mem;
727                 ent->next = list->head;
728                 list->head = ent;
729                 ++list->len;
730                 mem = NULL;
731                 }

733         CRYPTO_w_unlock(CRYPTO_LOCK_SSL_CTX);
734         if (mem)
735                 OPENSSL_free(mem);
736         }
737 #else
738 #define freelist_extract(c,fr,sz) OPENSSL_malloc(sz)
739 #define freelist_insert(c,fr,sz,m) OPENSSL_free(m)
740 #endif

742 int ssl3_setup_read_buffer(SSL *s)
743         {
744         unsigned char *p;
745         size_t len,align=0,headerlen;

747         if (SSL_version(s) == DTLS1_VERSION || SSL_version(s) == DTLS1_BAD_VER)
748                 headerlen = DTLS1_RT_HEADER_LENGTH;
749         else
750                 headerlen = SSL3_RT_HEADER_LENGTH;

752 #if defined(SSL3_ALIGN_PAYLOAD) && SSL3_ALIGN_PAYLOAD!=0
753         align = (-SSL3_RT_HEADER_LENGTH)&(SSL3_ALIGN_PAYLOAD-1);
754 #endif

756         if (s->s3->rbuf.buf == NULL)
757                 {
758                 len = SSL3_RT_MAX_PLAIN_LENGTH
759                         + SSL3_RT_MAX_ENCRYPTED_OVERHEAD
760                         + headerlen + align;
761                 if (s->options & SSL_OP_MICROSOFT_BIG_SSLV3_BUFFER)
762                         {
763                         s->s3->init_extra = 1;
764                         len += SSL3_RT_MAX_EXTRA;
765                         }
766 #ifndef OPENSSL_NO_COMP
767                 if (!(s->options & SSL_OP_NO_COMPRESSION))
768                         len += SSL3_RT_MAX_COMPRESSED_OVERHEAD;
769 #endif
770                 if ((p=freelist_extract(s->ctx, 1, len)) == NULL)
771                         goto err;
772                 s->s3->rbuf.buf = p;
773                 s->s3->rbuf.len = len;
774                 }

776         s->packet= &(s->s3->rbuf.buf[0]);
777         return 1;

779 err:
780         SSLerr(SSL_F_SSL3_SETUP_READ_BUFFER,ERR_R_MALLOC_FAILURE);
781         return 0;
782         }

784 int ssl3_setup_write_buffer(SSL *s)
785         {
786         unsigned char *p;
787         size_t len,align=0,headerlen;
```

```
789         if (SSL_version(s) == DTLS1_VERSION || SSL_version(s) == DTLS1_BAD_VER)
790                 headerlen = DTLS1_RT_HEADER_LENGTH + 1;
791         else
792                 headerlen = SSL3_RT_HEADER_LENGTH;

794 #if defined(SSL3_ALIGN_PAYLOAD) && SSL3_ALIGN_PAYLOAD!=0
795         align = (-SSL3_RT_HEADER_LENGTH)&(SSL3_ALIGN_PAYLOAD-1);
796 #endif

798         if (s->s3->wbuf.buf == NULL)
799                 {
800                 len = s->max_send_fragment
801                         + SSL3_RT_SEND_MAX_ENCRYPTED_OVERHEAD
802                         + headerlen + align;
803 #ifndef OPENSSL_NO_COMP
804                 if (!(s->options & SSL_OP_NO_COMPRESSION))
805                         len += SSL3_RT_MAX_COMPRESSED_OVERHEAD;
806 #endif
807                 if (!(s->options & SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS))
808                         len += headerlen + align
809                                 + SSL3_RT_SEND_MAX_ENCRYPTED_OVERHEAD;

811                 if ((p=freelist_extract(s->ctx, 0, len)) == NULL)
812                         goto err;
813                 s->s3->wbuf.buf = p;
814                 s->s3->wbuf.len = len;
815                 }

817         return 1;

819 err:
820         SSLerr(SSL_F_SSL3_SETUP_WRITE_BUFFER,ERR_R_MALLOC_FAILURE);
821         return 0;
822         }


825 int ssl3_setup_buffers(SSL *s)
826         {
827         if (!ssl3_setup_read_buffer(s))
828                 return 0;
829         if (!ssl3_setup_write_buffer(s))
830                 return 0;
831         return 1;
832         }

834 int ssl3_release_write_buffer(SSL *s)
835         {
836         if (s->s3->wbuf.buf != NULL)
837                 {
838                 freelist_insert(s->ctx, 0, s->s3->wbuf.len, s->s3->wbuf.buf);
839                 s->s3->wbuf.buf = NULL;
840                 }
841         return 1;
842         }

844 int ssl3_release_read_buffer(SSL *s)
845         {
846         if (s->s3->rbuf.buf != NULL)
847                 {
848                 freelist_insert(s->ctx, 1, s->s3->rbuf.len, s->s3->rbuf.buf);
849                 s->s3->rbuf.buf = NULL;
850                 }
851         return 1;
852         }
853 #endif /* ! codereview */
```

```
**********************************************************
   28213 Wed Aug 13 19:53:38 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_cbc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s3_cbc.c */
/* ====================================================================
 * Copyright (c) 2012 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include "ssl_locl.h"

#include <openssl/md5.h>
#include <openssl/sha.h>

/* MAX_HASH_BIT_COUNT_BYTES is the maximum number of bytes in the hash's length
```



```
 * field. (SHA-384/512 have 128-bit length.) */
#define MAX_HASH_BIT_COUNT_BYTES 16

/* MAX_HASH_BLOCK_SIZE is the maximum hash block size that we'll support.
 * Currently SHA-384/512 has a 128-byte block size and that's the largest
 * supported by TLS.) */
#define MAX_HASH_BLOCK_SIZE 128

/* Some utility functions are needed:
 *
 * These macros return the given value with the MSB copied to all the other
 * bits. They use the fact that arithmetic shift shifts-in the sign bit.
 * However, this is not ensured by the C standard so you may need to replace
 * them with something else on odd CPUs. */
#define DUPLICATE_MSB_TO_ALL(x) ( (unsigned)( (int)(x) >> (sizeof(int)*8-1) ) )
#define DUPLICATE_MSB_TO_ALL_8(x) ((unsigned char)(DUPLICATE_MSB_TO_ALL(x)))

/* constant_time_lt returns 0xff if a<b and 0x00 otherwise. */
static unsigned constant_time_lt(unsigned a, unsigned b)
        {
        a -= b;
        return DUPLICATE_MSB_TO_ALL(a);
        }

/* constant_time_ge returns 0xff if a>=b and 0x00 otherwise. */
static unsigned constant_time_ge(unsigned a, unsigned b)
        {
        a -= b;
        return DUPLICATE_MSB_TO_ALL(~a);
        }

/* constant_time_eq_8 returns 0xff if a==b and 0x00 otherwise. */
static unsigned char constant_time_eq_8(unsigned a, unsigned b)
        {
        unsigned c = a ^ b;
        c--;
        return DUPLICATE_MSB_TO_ALL_8(c);
        }

/* ssl3_cbc_remove_padding removes padding from the decrypted, SSLv3, CBC
 * record in |rec| by updating |rec->length| in constant time.
 *
 * block_size: the block size of the cipher used to encrypt the record.
 * returns:
 *   0: (in non-constant time) if the record is publicly invalid.
 *   1: if the padding was valid
 *  -1: otherwise. */
int ssl3_cbc_remove_padding(const SSL* s,
                            SSL3_RECORD *rec,
                            unsigned block_size,
                            unsigned mac_size)
        {
        unsigned padding_length, good;
        const unsigned overhead = 1 /* padding length byte */ + mac_size;

        /* These lengths are all public so we can test them in non-constant
         * time. */
        if (overhead > rec->length)
                return 0;

        padding_length = rec->data[rec->length-1];
        good = constant_time_ge(rec->length, padding_length+overhead);
        /* SSLv3 requires that the padding is minimal. */
        good &= constant_time_ge(block_size, padding_length+1);
        padding_length = good & (padding_length+1);
        rec->length -= padding_length;
```

```
128         rec->type |= padding_length<<8; /* kludge: pass padding length */
129         return (int)((good & 1) | (~good & -1));
130 }

132 /* tls1_cbc_remove_padding removes the CBC padding from the decrypted, TLS, CBC
133  * record in |rec| in constant time and returns 1 if the padding is valid and
134  * -1 otherwise. It also removes any explicit IV from the start of the record
135  * without leaking any timing about whether there was enough space after the
136  * padding was removed.
137  *
138  * block_size: the block size of the cipher used to encrypt the record.
139  * returns:
140  *   0: (in non-constant time) if the record is publicly invalid.
141  *   1: if the padding was valid
142  *  -1: otherwise. */
143 int tls1_cbc_remove_padding(const SSL* s,
144                             SSL3_RECORD *rec,
145                             unsigned block_size,
146                             unsigned mac_size)
147         {
148         unsigned padding_length, good, to_check, i;
149         const unsigned overhead = 1 /* padding length byte */ + mac_size;
150         /* Check if version requires explicit IV */
151         if (s->version >= TLS1_1_VERSION || s->version == DTLS1_BAD_VER)
152                 {
153                 /* These lengths are all public so we can test them in
154                  * non-constant time.
155                  */
156                 if (overhead + block_size > rec->length)
157                         return 0;
158                 /* We can now safely skip explicit IV */
159                 rec->data += block_size;
160                 rec->input += block_size;
161                 rec->length -= block_size;
162                 }
163         else if (overhead > rec->length)
164                 return 0;

166         padding_length = rec->data[rec->length-1];

168         /* NB: if compression is in operation the first packet may not be of
169          * even length so the padding bug check cannot be performed. This bug
170          * workaround has been around since SSLeay so hopefully it is either
171          * fixed now or no buggy implementation supports compression [steve]
172          */
173         if ( (s->options&SSL_OP_TLS_BLOCK_PADDING_BUG) && !s->expand)
174                 {
175                 /* First packet is even in size, so check */
176                 if ((memcmp(s->s3->read_sequence, "\0\0\0\0\0\0\0\0",8) == 0) &&
177                     !(padding_length & 1))
178                         {
179                         s->s3->flags|=TLS1_FLAGS_TLS_PADDING_BUG;
180                         }
181                 if ((s->s3->flags & TLS1_FLAGS_TLS_PADDING_BUG) &&
182                     padding_length > 0)
183                         {
184                         padding_length--;
185                         }
186                 }

188         if (EVP_CIPHER_flags(s->enc_read_ctx->cipher)&EVP_CIPH_FLAG_AEAD_CIPHER)
189                 {
190                 /* padding is already verified */
191                 rec->length -= padding_length + 1;
192                 return 1;
193                 }
```



```
195         good = constant_time_ge(rec->length, overhead+padding_length);
196         /* The padding consists of a length byte at the end of the record and
197          * then that many bytes of padding, all with the same value as the
198          * length byte. Thus, with the length byte included, there are i+1
199          * bytes of padding.
200          *
201          * We can't check just |padding_length+1| bytes because that leaks
202          * decrypted information. Therefore we always have to check the maximum
203          * amount of padding possible. (Again, the length of the record is
204          * public information so we can use it.) */
205         to_check = 255; /* maximum amount of padding. */
206         if (to_check > rec->length-1)
207                 to_check = rec->length-1;

209         for (i = 0; i < to_check; i++)
210                 {
211                 unsigned char mask = constant_time_ge(padding_length, i);
212                 unsigned char b = rec->data[rec->length-1-i];
213                 /* The final |padding_length+1| bytes should all have the value
214                  * |padding_length|. Therefore the XOR should be zero. */
215                 good &= ~(mask&(padding_length ^ b));
216                 }

218         /* If any of the final |padding_length+1| bytes had the wrong value,
219          * one or more of the lower eight bits of |good| will be cleared. We
220          * AND the bottom 8 bits together and duplicate the result to all the
221          * bits. */
222         good &= good >> 4;
223         good &= good >> 2;
224         good &= good >> 1;
225         good <<= sizeof(good)*8-1;
226         good = DUPLICATE_MSB_TO_ALL(good);

228         padding_length = good & (padding_length+1);
229         rec->length -= padding_length;
230         rec->type |= padding_length<<8; /* kludge: pass padding length */

232         return (int)((good & 1) | (~good & -1));
233         }

235 /* ssl3_cbc_copy_mac copies |md_size| bytes from the end of |rec| to |out| in
236  * constant time (independent of the concrete value of rec->length, which may
237  * vary within a 256-byte window).
238  *
239  * ssl3_cbc_remove_padding or tls1_cbc_remove_padding must be called prior to
240  * this function.
241  *
242  * On entry:
243  *   rec->orig_len >= md_size
244  *   md_size <= EVP_MAX_MD_SIZE
245  *
246  * If CBC_MAC_ROTATE_IN_PLACE is defined then the rotation is performed with
247  * variable accesses in a 64-byte-aligned buffer. Assuming that this fits into
248  * a single or pair of cache-lines, then the variable memory accesses don't
249  * actually affect the timing. CPUs with smaller cache-lines [if any] are
250  * not multi-core and are not considered vulnerable to cache-timing attacks.
251  */
252 #define CBC_MAC_ROTATE_IN_PLACE

254 void ssl3_cbc_copy_mac(unsigned char* out,
255                        const SSL3_RECORD *rec,
256                        unsigned md_size,unsigned orig_len)
257         {
258 #if defined(CBC_MAC_ROTATE_IN_PLACE)
259         unsigned char rotated_mac_buf[64+EVP_MAX_MD_SIZE];
```

```
260         unsigned char *rotated_mac;
261 #else
262         unsigned char rotated_mac[EVP_MAX_MD_SIZE];
263 #endif

265         /* mac_end is the index of |rec->data| just after the end of the MAC. */
266         unsigned mac_end = rec->length;
267         unsigned mac_start = mac_end - md_size;
268         /* scan_start contains the number of bytes that we can ignore because
269          * the MAC's position can only vary by 255 bytes. */
270         unsigned scan_start = 0;
271         unsigned i, j;
272         unsigned div_spoiler;
273         unsigned rotate_offset;

275         OPENSSL_assert(orig_len >= md_size);
276         OPENSSL_assert(md_size <= EVP_MAX_MD_SIZE);

278 #if defined(CBC_MAC_ROTATE_IN_PLACE)
279         rotated_mac = rotated_mac_buf + ((0-(size_t)rotated_mac_buf)&63);
280 #endif

282         /* This information is public so it's safe to branch based on it. */
283         if (orig_len > md_size + 255 + 1)
284                 scan_start = orig_len - (md_size + 255 + 1);
285         /* div_spoiler contains a multiple of md_size that is used to cause the
286          * modulo operation to be constant time. Without this, the time varies
287          * based on the amount of padding when running on Intel chips at least.
288          *
289          * The aim of right-shifting md_size is so that the compiler doesn't
290          * figure out that it can remove div_spoiler as that would require it
291          * to prove that md_size is always even, which I hope is beyond it. */
292         div_spoiler = md_size >> 1;
293         div_spoiler <<= (sizeof(div_spoiler)-1)*8;
294         rotate_offset = (div_spoiler + mac_start - scan_start) % md_size;

296         memset(rotated_mac, 0, md_size);
297         for (i = scan_start, j = 0; i < orig_len; i++)
298                 {
299                 unsigned char mac_started = constant_time_ge(i, mac_start);
300                 unsigned char mac_ended = constant_time_ge(i, mac_end);
301                 unsigned char b = rec->data[i];
302                 rotated_mac[j++] |= b & mac_started & ~mac_ended;
303                 j &= constant_time_lt(j,md_size);
304                 }

306         /* Now rotate the MAC */
307 #if defined(CBC_MAC_ROTATE_IN_PLACE)
308         j = 0;
309         for (i = 0; i < md_size; i++)
310                 {
311                 /* in case cache-line is 32 bytes, touch second line */
312                 ((volatile unsigned char *)rotated_mac)[rotate_offset^32];
313                 out[j++] = rotated_mac[rotate_offset++];
314                 rotate_offset &= constant_time_lt(rotate_offset,md_size);
315                 }
316 #else
317         memset(out, 0, md_size);
318         rotate_offset = md_size - rotate_offset;
319         rotate_offset &= constant_time_lt(rotate_offset,md_size);
320         for (i = 0; i < md_size; i++)
321                 {
322                 for (j = 0; j < md_size; j++)
323                         out[j] |= rotated_mac[i] & constant_time_eq_8(j, rotate_
324                 rotate_offset++;
325                 rotate_offset &= constant_time_lt(rotate_offset,md_size);
```



```
326                 }
327 #endif
328         }

330 /* u32toLE serialises an unsigned, 32-bit number (n) as four bytes at (p) in
331  * little-endian order. The value of p is advanced by four. */
332 #define u32toLE(n, p) \
333         (*((p)++)=(unsigned char)(n), \
334          *((p)++)=(unsigned char)(n>>8), \
335          *((p)++)=(unsigned char)(n>>16), \
336          *((p)++)=(unsigned char)(n>>24))

338 /* These functions serialize the state of a hash and thus perform the standard
339  * "final" operation without adding the padding and length that such a function
340  * typically does. */
341 static void tls1_md5_final_raw(void* ctx, unsigned char *md_out)
342         {
343         MD5_CTX *md5 = ctx;
344         u32toLE(md5->A, md_out);
345         u32toLE(md5->B, md_out);
346         u32toLE(md5->C, md_out);
347         u32toLE(md5->D, md_out);
348         }

350 static void tls1_sha1_final_raw(void* ctx, unsigned char *md_out)
351         {
352         SHA_CTX *sha1 = ctx;
353         l2n(sha1->h0, md_out);
354         l2n(sha1->h1, md_out);
355         l2n(sha1->h2, md_out);
356         l2n(sha1->h3, md_out);
357         l2n(sha1->h4, md_out);
358         }
359 #define LARGEST_DIGEST_CTX SHA_CTX

361 #ifndef OPENSSL_NO_SHA256
362 static void tls1_sha256_final_raw(void* ctx, unsigned char *md_out)
363         {
364         SHA256_CTX *sha256 = ctx;
365         unsigned i;

367         for (i = 0; i < 8; i++)
368                 {
369                 l2n(sha256->h[i], md_out);
370                 }
371         }
372 #undef  LARGEST_DIGEST_CTX
373 #define LARGEST_DIGEST_CTX SHA256_CTX
374 #endif

376 #ifndef OPENSSL_NO_SHA512
377 static void tls1_sha512_final_raw(void* ctx, unsigned char *md_out)
378         {
379         SHA512_CTX *sha512 = ctx;
380         unsigned i;

382         for (i = 0; i < 8; i++)
383                 {
384                 l2n8(sha512->h[i], md_out);
385                 }
386         }
387 #undef  LARGEST_DIGEST_CTX
388 #define LARGEST_DIGEST_CTX SHA512_CTX
389 #endif

391 /* ssl3_cbc_record_digest_supported returns 1 iff |ctx| uses a hash function
```

```
392  * which ssl3_cbc_digest_record supports. */
393 char ssl3_cbc_record_digest_supported(const EVP_MD_CTX *ctx)
394         {
395 #ifdef OPENSSL_FIPS
396         if (FIPS_mode())
397                 return 0;
398 #endif
399         switch (EVP_MD_CTX_type(ctx))
400                 {
401                 case NID_md5:
402                 case NID_sha1:
403 #ifndef OPENSSL_NO_SHA256
404                 case NID_sha224:
405                 case NID_sha256:
406 #endif
407 #ifndef OPENSSL_NO_SHA512
408                 case NID_sha384:
409                 case NID_sha512:
410 #endif
411                         return 1;
412                 default:
413                         return 0;
414                 }
415         }

417 /* ssl3_cbc_digest_record computes the MAC of a decrypted, padded SSLv3/TLS
418  * record.
419  *
420  *   ctx: the EVP_MD_CTX from which we take the hash function.
421  *     ssl3_cbc_record_digest_supported must return true for this EVP_MD_CTX.
422  *   md_out: the digest output. At most EVP_MAX_MD_SIZE bytes will be written.
423  *   md_out_size: if non-NULL, the number of output bytes is written here.
424  *   header: the 13-byte, TLS record header.
425  *   data: the record data itself, less any preceeding explicit IV.
426  *   data_plus_mac_size: the secret, reported length of the data and MAC
427  *     once the padding has been removed.
428  *   data_plus_mac_plus_padding_size: the public length of the whole
429  *     record, including padding.
430  *   is_sslv3: non-zero if we are to use SSLv3. Otherwise, TLS.
431  *
432  * On entry: by virtue of having been through one of the remove_padding
433  * functions, above, we know that data_plus_mac_size is large enough to contain
434  * a padding byte and MAC. (If the padding was invalid, it might contain the
435  * padding too. ) */
436 void ssl3_cbc_digest_record(
437         const EVP_MD_CTX *ctx,
438         unsigned char* md_out,
439         size_t* md_out_size,
440         const unsigned char header[13],
441         const unsigned char *data,
442         size_t data_plus_mac_size,
443         size_t data_plus_mac_plus_padding_size,
444         const unsigned char *mac_secret,
445         unsigned mac_secret_length,
446         char is_sslv3)
447         {
448         union { double align;
449                 unsigned char c[sizeof(LARGEST_DIGEST_CTX)]; } md_state;
450         void (*md_final_raw)(void *ctx, unsigned char *md_out);
451         void (*md_transform)(void *ctx, const unsigned char *block);
452         unsigned md_size, md_block_size = 64;
453         unsigned sslv3_pad_length = 40, header_length, variance_blocks,
454                  len, max_mac_bytes, num_blocks,
455                  num_starting_blocks, k, mac_end_offset, c, index_a, index_b;
456         unsigned int bits;      /* at most 18 bits */
457         unsigned char length_bytes[MAX_HASH_BIT_COUNT_BYTES];
```



```
458         /* hmac_pad is the masked HMAC key. */
459         unsigned char hmac_pad[MAX_HASH_BLOCK_SIZE];
460         unsigned char first_block[MAX_HASH_BLOCK_SIZE];
461         unsigned char mac_out[EVP_MAX_MD_SIZE];
462         unsigned i, j, md_out_size_u;
463         EVP_MD_CTX md_ctx;
464         /* mdLengthSize is the number of bytes in the length field that terminat
465         * the hash. */
466         unsigned md_length_size = 8;
467         char length_is_big_endian = 1;

469         /* This is a, hopefully redundant, check that allows us to forget about
470          * many possible overflows later in this function. */
471         OPENSSL_assert(data_plus_mac_plus_padding_size < 1024*1024);

473         switch (EVP_MD_CTX_type(ctx))
474                 {
475                 case NID_md5:
476                         MD5_Init((MD5_CTX*)md_state.c);
477                         md_final_raw = tls1_md5_final_raw;
478                         md_transform = (void(*)(void *ctx, const unsigned char *
479                         md_size = 16;
480                         sslv3_pad_length = 48;
481                         length_is_big_endian = 0;
482                         break;
483                 case NID_sha1:
484                         SHA1_Init((SHA_CTX*)md_state.c);
485                         md_final_raw = tls1_sha1_final_raw;
486                         md_transform = (void(*)(void *ctx, const unsigned char *
487                         md_size = 20;
488                         break;
489 #ifndef OPENSSL_NO_SHA256
490                 case NID_sha224:
491                         SHA224_Init((SHA256_CTX*)md_state.c);
492                         md_final_raw = tls1_sha256_final_raw;
493                         md_transform = (void(*)(void *ctx, const unsigned char *
494                         md_size = 224/8;
495                         break;
496                 case NID_sha256:
497                         SHA256_Init((SHA256_CTX*)md_state.c);
498                         md_final_raw = tls1_sha256_final_raw;
499                         md_transform = (void(*)(void *ctx, const unsigned char *
500                         md_size = 32;
501                         break;
502 #endif
503 #ifndef OPENSSL_NO_SHA512
504                 case NID_sha384:
505                         SHA384_Init((SHA512_CTX*)md_state.c);
506                         md_final_raw = tls1_sha512_final_raw;
507                         md_transform = (void(*)(void *ctx, const unsigned char *
508                         md_size = 384/8;
509                         md_block_size = 128;
510                         md_length_size = 16;
511                         break;
512                 case NID_sha512:
513                         SHA512_Init((SHA512_CTX*)md_state.c);
514                         md_final_raw = tls1_sha512_final_raw;
515                         md_transform = (void(*)(void *ctx, const unsigned char *
516                         md_size = 64;
517                         md_block_size = 128;
518                         md_length_size = 16;
519                         break;
520 #endif
521                 default:
522                         /* ssl3_cbc_record_digest_supported should have been
523                          * called first to check that the hash function is
```

```
524                          * supported. */
525                         OPENSSL_assert(0);
526                         if (md_out_size)
527                                 *md_out_size = -1;
528                         return;
529                 }

531         OPENSSL_assert(md_length_size <= MAX_HASH_BIT_COUNT_BYTES);
532         OPENSSL_assert(md_block_size <= MAX_HASH_BLOCK_SIZE);
533         OPENSSL_assert(md_size <= EVP_MAX_MD_SIZE);

535         header_length = 13;
536         if (is_sslv3)
537                 {
538                 header_length =
539                         mac_secret_length +
540                         sslv3_pad_length +
541                         8 /* sequence number */ +
542                         1 /* record type */ +
543                         2 /* record length */;
544                 }

546         /* variance_blocks is the number of blocks of the hash that we have to
547          * calculate in constant time because they could be altered by the
548          * padding value.
549          *
550          * In SSLv3, the padding must be minimal so the end of the plaintext
551          * varies by, at most, 15+20 = 35 bytes. (We conservatively assume that
552          * the MAC size varies from 0..20 bytes.) In case the 9 bytes of hash
553          * termination (0x80 + 64-bit length) don't fit in the final block, we
554          * say that the final two blocks can vary based on the padding.
555          *
556          * TLSv1 has MACs up to 48 bytes long (SHA-384) and the padding is not
557          * required to be minimal. Therefore we say that the final six blocks
558          * can vary based on the padding.
559          *
560          * Later in the function, if the message is short and there obviously
561          * cannot be this many blocks then variance_blocks can be reduced. */
562         variance_blocks = is_sslv3 ? 2 : 6;
563         /* From now on we're dealing with the MAC, which conceptually has 13
564          * bytes of 'header' before the start of the data (TLS) or 71/75 bytes
565          * (SSLv3) */
566         len = data_plus_mac_plus_padding_size + header_length;
567         /* max_mac_bytes contains the maximum bytes of bytes in the MAC, includi
568         * |header|, assuming that there's no padding. */
569         max_mac_bytes = len - md_size - 1;
570         /* num_blocks is the maximum number of hash blocks. */
571         num_blocks = (max_mac_bytes + 1 + md_length_size + md_block_size - 1) /
572         /* In order to calculate the MAC in constant time we have to handle
573          * the final blocks specially because the padding value could cause the
574          * end to appear somewhere in the final |variance_blocks| blocks and we
575          * can't leak where. However, |num_starting_blocks| worth of data can
576          * be hashed right away because no padding value can affect whether
577          * they are plaintext. */
578         num_starting_blocks = 0;
579         /* k is the starting byte offset into the conceptual header||data where
580          * we start processing. */
581         k = 0;
582         /* mac_end_offset is the index just past the end of the data to be
583          * MACed. */
584         mac_end_offset = data_plus_mac_size + header_length - md_size;
585         /* c is the index of the 0x80 byte in the final hash block that
586          * contains application data. */
587         c = mac_end_offset % md_block_size;
588         /* index_a is the hash block number that contains the 0x80 terminating
589          * value. */
```



```
590         index_a = mac_end_offset / md_block_size;
591         /* index_b is the hash block number that contains the 64-bit hash
592          * length, in bits. */
593         index_b = (mac_end_offset + md_length_size) / md_block_size;
594         /* bits is the hash-length in bits. It includes the additional hash
595          * block for the masked HMAC key, or whole of |header| in the case of
596          * SSLv3. */

598         /* For SSLv3, if we're going to have any starting blocks then we need
599          * at least two because the header is larger than a single block. */
600         if (num_blocks > variance_blocks + (is_sslv3 ? 1 : 0))
601                 {
602                 num_starting_blocks = num_blocks - variance_blocks;
603                 k = md_block_size*num_starting_blocks;
604                 }

606         bits = 8*mac_end_offset;
607         if (!is_sslv3)
608                 {
609                 /* Compute the initial HMAC block. For SSLv3, the padding and
610                  * secret bytes are included in |header| because they take more
611                  * than a single block. */
612                 bits += 8*md_block_size;
613                 memset(hmac_pad, 0, md_block_size);
614                 OPENSSL_assert(mac_secret_length <= sizeof(hmac_pad));
615                 memcpy(hmac_pad, mac_secret, mac_secret_length);
616                 for (i = 0; i < md_block_size; i++)
617                         hmac_pad[i] ^= 0x36;

619                 md_transform(md_state.c, hmac_pad);
620                 }

622         if (length_is_big_endian)
623                 {
624                 memset(length_bytes,0,md_length_size-4);
625                 length_bytes[md_length_size-4] = (unsigned char)(bits>>24);
626                 length_bytes[md_length_size-3] = (unsigned char)(bits>>16);
627                 length_bytes[md_length_size-2] = (unsigned char)(bits>>8);
628                 length_bytes[md_length_size-1] = (unsigned char)bits;
629                 }
630         else
631                 {
632                 memset(length_bytes,0,md_length_size);
633                 length_bytes[md_length_size-5] = (unsigned char)(bits>>24);
634                 length_bytes[md_length_size-6] = (unsigned char)(bits>>16);
635                 length_bytes[md_length_size-7] = (unsigned char)(bits>>8);
636                 length_bytes[md_length_size-8] = (unsigned char)bits;
637                 }

639         if (k > 0)
640                 {
641                 if (is_sslv3)
642                         {
643                         /* The SSLv3 header is larger than a single block.
644                          * overhang is the number of bytes beyond a single
645                          * block that the header consumes: either 7 bytes
646                          * (SHA1) or 11 bytes (MD5). */
647                         unsigned overhang = header_length-md_block_size;
648                         md_transform(md_state.c, header);
649                         memcpy(first_block, header + md_block_size, overhang);
650                         memcpy(first_block + overhang, data, md_block_size-overh
651                         md_transform(md_state.c, first_block);
652                         for (i = 1; i < k/md_block_size - 1; i++)
653                                 md_transform(md_state.c, data + md_block_size*i
654                         }
655                 else
```

```
656                         {
657                         /* k is a multiple of md_block_size. */
658                         memcpy(first_block, header, 13);
659                         memcpy(first_block+13, data, md_block_size-13);
660                         md_transform(md_state.c, first_block);
661                         for (i = 1; i < k/md_block_size; i++)
662                                 md_transform(md_state.c, data + md_block_size*i
663                         }
664                 }

666         memset(mac_out, 0, sizeof(mac_out));

668         /* We now process the final hash blocks. For each block, we construct
669          * it in constant time. If the |i==index_a| then we'll include the 0x80
670          * bytes and zero pad etc. For each block we selectively copy it, in
671          * constant time, to |mac_out|. */
672         for (i = num_starting_blocks; i <= num_starting_blocks+variance_blocks;
673                 {
674                 unsigned char block[MAX_HASH_BLOCK_SIZE];
675                 unsigned char is_block_a = constant_time_eq_8(i, index_a);
676                 unsigned char is_block_b = constant_time_eq_8(i, index_b);
677                 for (j = 0; j < md_block_size; j++)
678                         {
679                         unsigned char b = 0, is_past_c, is_past_cp1;
680                         if (k < header_length)
681                                 b = header[k];
682                         else if (k < data_plus_mac_plus_padding_size + header_le
683                                 b = data[k-header_length];
684                         k++;

686                         is_past_c = is_block_a & constant_time_ge(j, c);
687                         is_past_cp1 = is_block_a & constant_time_ge(j, c+1);
688                         /* If this is the block containing the end of the
689                          * application data, and we are at the offset for the
690                          * 0x80 value, then overwrite b with 0x80. */
691                         b = (b&~is_past_c) | (0x80&is_past_c);
692                         /* If this the the block containing the end of the
693                          * application data and we're past the 0x80 value then
694                          * just write zero. */
695                         b = b&~is_past_cp1;
696                         /* If this is index_b (the final block), but not
697                          * index_a (the end of the data), then the 64-bit
698                          * length didn't fit into index_a and we're having to
699                          * add an extra block of zeros. */
700                         b &= ~is_block_b | is_block_a;

702                         /* The final bytes of one of the blocks contains the
703                          * length. */
704                         if (j >= md_block_size - md_length_size)
705                                 {
706                                 /* If this is index_b, write a length byte. */
707                                 b = (b&~is_block_b) | (is_block_b&length_bytes[j
708                                 }
709                         block[j] = b;
710                         }

712                 md_transform(md_state.c, block);
713                 md_final_raw(md_state.c, block);
714                 /* If this is index_b, copy the hash value to |mac_out|. */
715                 for (j = 0; j < md_size; j++)
716                         mac_out[j] |= block[j]&is_block_b;
717                 }

719         EVP_MD_CTX_init(&md_ctx);
720         EVP_DigestInit_ex(&md_ctx, ctx->digest, NULL /* engine */);
721         if (is_sslv3)
```



```
722                 {
723                 /* We repurpose |hmac_pad| to contain the SSLv3 pad2 block. */
724                 memset(hmac_pad, 0x5c, sslv3_pad_length);

726                 EVP_DigestUpdate(&md_ctx, mac_secret, mac_secret_length);
727                 EVP_DigestUpdate(&md_ctx, hmac_pad, sslv3_pad_length);
728                 EVP_DigestUpdate(&md_ctx, mac_out, md_size);
729                 }
730         else
731                 {
732                 /* Complete the HMAC in the standard manner. */
733                 for (i = 0; i < md_block_size; i++)
734                         hmac_pad[i] ^= 0x6a;

736                 EVP_DigestUpdate(&md_ctx, hmac_pad, md_block_size);
737                 EVP_DigestUpdate(&md_ctx, mac_out, md_size);
738                 }
739         EVP_DigestFinal(&md_ctx, md_out, &md_out_size_u);
740         if (md_out_size)
741                 *md_out_size = md_out_size_u;
742         EVP_MD_CTX_cleanup(&md_ctx);
743         }

745 #ifdef OPENSSL_FIPS

747 /* Due to the need to use EVP in FIPS mode we can't reimplement digests but
748  * we can ensure the number of blocks processed is equal for all cases
749  * by digesting additional data.
750  */

752 void tls_fips_digest_extra(
753         const EVP_CIPHER_CTX *cipher_ctx, EVP_MD_CTX *mac_ctx,
754         const unsigned char *data, size_t data_len, size_t orig_len)
755         {
756         size_t block_size, digest_pad, blocks_data, blocks_orig;
757         if (EVP_CIPHER_CTX_mode(cipher_ctx) != EVP_CIPH_CBC_MODE)
758                 return;
759         block_size = EVP_MD_CTX_block_size(mac_ctx);
760         /* We are in FIPS mode if we get this far so we know we have only SHA*
761          * digests and TLS to deal with.
762          * Minimum digest padding length is 17 for SHA384/SHA512 and 9
763          * otherwise.
764          * Additional header is 13 bytes. To get the number of digest blocks
765          * processed round up the amount of data plus padding to the nearest
766          * block length. Block length is 128 for SHA384/SHA512 and 64 otherwise.
767          * So we have:
768          * blocks = (payload_len + digest_pad + 13 + block_size - 1)/block_size
769          * equivalently:
770          * blocks = (payload_len + digest_pad + 12)/block_size + 1
771          * HMAC adds a constant overhead.
772          * We're ultimately only interested in differences so this becomes
773          * blocks = (payload_len + 29)/128
774          * for SHA384/SHA512 and
775          * blocks = (payload_len + 21)/64
776          * otherwise.
777          */
778         digest_pad = block_size == 64 ? 21 : 29;
779         blocks_orig = (orig_len + digest_pad)/block_size;
780         blocks_data = (data_len + digest_pad)/block_size;
781         /* MAC enough blocks to make up the difference between the original
782          * and actual lengths plus one extra block to ensure this is never a
783          * no op. The "data" pointer should always have enough space to
784          * perform this operation as it is large enough for a maximum
785          * length TLS buffer.
786          */
787         EVP_DigestSignUpdate(mac_ctx, data,
```

```
788                                 (blocks_orig - blocks_data + 1) * block_size);
789         }
790 #endif
791 #endif /* ! codereview */
```

```
**********************************************************
   86215 Wed Aug 13 19:53:38 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_clnt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s3_clnt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * Portions of the attached software ("Contribution") are developed by
 * SUN MICROSYSTEMS, INC., and are contributed to the OpenSSL project.
 *
 * The Contribution is licensed pursuant to the OpenSSL open source
 * license provided above.
 *
 * ECC cipher suite support in OpenSSL originally written by
 * Vipul Gupta and Sumit Gupta of Sun Microsystems Laboratories.
 *
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
```

```
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include "ssl_locl.h"
#include "kssl_lcl.h"
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/md5.h>
#ifdef OPENSSL_FIPS
#include <openssl/fips.h>
#endif
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#include <openssl/bn.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

static const SSL_METHOD *ssl3_get_client_method(int ver);
static int ca_dn_cmp(const X509_NAME * const *a,const X509_NAME * const *b);

static const SSL_METHOD *ssl3_get_client_method(int ver)
        {
        if (ver == SSL3_VERSION)
                return(SSLv3_client_method());
        else
                return(NULL);
        }

IMPLEMENT_ssl3_meth_func(SSLv3_client_method,
                         ssl_undefined_function,
                         ssl3_connect,
                         ssl3_get_client_method)

int ssl3_connect(SSL *s)
        {
        BUF_MEM *buf=NULL;
        unsigned long Time=(unsigned long)time(NULL);
        void (*cb)(const SSL *ssl,int type,int val)=NULL;
        int ret= -1;
        int new_state,state,skip=0;
```



```
        RAND_add(&Time,sizeof(Time),0);
        ERR_clear_error();
        clear_sys_error();

        if (s->info_callback != NULL)
                cb=s->info_callback;
        else if (s->ctx->info_callback != NULL)
                cb=s->ctx->info_callback;

        s->in_handshake++;
        if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

#ifndef OPENSSL_NO_HEARTBEATS
        /* If we're awaiting a HeartbeatResponse, pretend we
         * already got and don't await it anymore, because
         * Heartbeats don't make sense during handshakes anyway.
         */
        if (s->tlsext_hb_pending)
                {
                s->tlsext_hb_pending = 0;
                s->tlsext_hb_seq++;
                }
#endif

        for (;;)
                {
                state=s->state;

                switch(s->state)
                        {
                case SSL_ST_RENEGOTIATE:
                        s->renegotiate=1;
                        s->state=SSL_ST_CONNECT;
                        s->ctx->stats.sess_connect_renegotiate++;
                        /* break */
                case SSL_ST_BEFORE:
                case SSL_ST_CONNECT:
                case SSL_ST_BEFORE|SSL_ST_CONNECT:
                case SSL_ST_OK|SSL_ST_CONNECT:

                        s->server=0;
                        if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

                        if ((s->version & 0xff00 ) != 0x0300)
                                {
                                SSLerr(SSL_F_SSL3_CONNECT, ERR_R_INTERNAL_ERROR);
                                ret = -1;
                                goto end;
                                }

                        /* s->version=SSL3_VERSION; */
                        s->type=SSL_ST_CONNECT;

                        if (s->init_buf == NULL)
                                {
                                if ((buf=BUF_MEM_new()) == NULL)
                                        {
                                        ret= -1;
                                        goto end;
                                        }
                                if (!BUF_MEM_grow(buf,SSL3_RT_MAX_PLAIN_LENGTH))
                                        {
                                        ret= -1;
                                        goto end;
                                        }
                                s->init_buf=buf;
```

```
260                                 buf=NULL;
261                                 }
263                         if (!ssl3_setup_buffers(s)) { ret= -1; goto end; }
265                         /* setup buffing BIO */
266                         if (!ssl_init_wbio_buffer(s,0)) { ret= -1; goto end; }
268                         /* don't push the buffering BIO quite yet */
270                         ssl3_init_finished_mac(s);
272                         s->state=SSL3_ST_CW_CLNT_HELLO_A;
273                         s->ctx->stats.sess_connect++;
274                         s->init_num=0;
275                         break;
277                 case SSL3_ST_CW_CLNT_HELLO_A:
278                 case SSL3_ST_CW_CLNT_HELLO_B:
280                         s->shutdown=0;
281                         ret=ssl3_client_hello(s);
282                         if (ret <= 0) goto end;
283                         s->state=SSL3_ST_CR_SRVR_HELLO_A;
284                         s->init_num=0;
286                         /* turn on buffering for the next lot of output */
287                         if (s->bbio != s->wbio)
288                                 s->wbio=BIO_push(s->bbio,s->wbio);
290                         break;
292                 case SSL3_ST_CR_SRVR_HELLO_A:
293                 case SSL3_ST_CR_SRVR_HELLO_B:
294                         ret=ssl3_get_server_hello(s);
295                         if (ret <= 0) goto end;
297                         if (s->hit)
298                                 {
299                                 s->state=SSL3_ST_CR_FINISHED_A;
300 #ifndef OPENSSL_NO_TLSEXT
301                                 if (s->tlsext_ticket_expected)
302                                         {
303                                         /* receive renewed session ticket */
304                                         s->state=SSL3_ST_CR_SESSION_TICKET_A;
305                                         }
306 #endif
307                                 }
308                         else
309                                 s->state=SSL3_ST_CR_CERT_A;
310                         s->init_num=0;
311                         break;
313                 case SSL3_ST_CR_CERT_A:
314                 case SSL3_ST_CR_CERT_B:
315 #ifndef OPENSSL_NO_TLSEXT
316                         ret=ssl3_check_finished(s);
317                         if (ret <= 0) goto end;
318                         if (ret == 2)
319                                 {
320                                 s->hit = 1;
321                                 if (s->tlsext_ticket_expected)
322                                         s->state=SSL3_ST_CR_SESSION_TICKET_A;
323                                 else
324                                         s->state=SSL3_ST_CR_FINISHED_A;
325                                 s->init_num=0;
```



```
326                                 break;
327                                 }
328 #endif
329                         /* Check if it is anon DH/ECDH */
330                         /* or PSK */
331                         if (!(s->s3->tmp.new_cipher->algorithm_auth & SSL_aNULL)
332                             !(s->s3->tmp.new_cipher->algorithm_mkey & SSL_kPSK))
333                                 {
334                                 ret=ssl3_get_server_certificate(s);
335                                 if (ret <= 0) goto end;
336 #ifndef OPENSSL_NO_TLSEXT
337                                 if (s->tlsext_status_expected)
338                                         s->state=SSL3_ST_CR_CERT_STATUS_A;
339                                 else
340                                         s->state=SSL3_ST_CR_KEY_EXCH_A;
341                                 }
342                         else
343                                 {
344                                 skip = 1;
345                                 s->state=SSL3_ST_CR_KEY_EXCH_A;
346                                 }
347 #else
348                                 }
349                         else
350                                 skip=1;
352                         s->state=SSL3_ST_CR_KEY_EXCH_A;
353 #endif
354                         s->init_num=0;
355                         break;
357                 case SSL3_ST_CR_KEY_EXCH_A:
358                 case SSL3_ST_CR_KEY_EXCH_B:
359                         ret=ssl3_get_key_exchange(s);
360                         if (ret <= 0) goto end;
361                         s->state=SSL3_ST_CR_CERT_REQ_A;
362                         s->init_num=0;
364                         /* at this point we check that we have the
365                          * required stuff from the server */
366                         if (!ssl3_check_cert_and_algorithm(s))
367                                 {
368                                 ret= -1;
369                                 goto end;
370                                 }
371                         break;
373                 case SSL3_ST_CR_CERT_REQ_A:
374                 case SSL3_ST_CR_CERT_REQ_B:
375                         ret=ssl3_get_certificate_request(s);
376                         if (ret <= 0) goto end;
377                         s->state=SSL3_ST_CR_SRVR_DONE_A;
378                         s->init_num=0;
379                         break;
381                 case SSL3_ST_CR_SRVR_DONE_A:
382                 case SSL3_ST_CR_SRVR_DONE_B:
383                         ret=ssl3_get_server_done(s);
384                         if (ret <= 0) goto end;
385 #ifndef OPENSSL_NO_SRP
386                         if (s->s3->tmp.new_cipher->algorithm_mkey & SSL_kSRP)
387                                 {
388                                 if ((ret = SRP_Calc_A_param(s))<=0)
389                                         {
390                                         SSLerr(SSL_F_SSL3_CONNECT,SSL_R_SRP_A_CA
391                                         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_I
```

```
392                                     goto end;
393                                     }
394                             }
395 #endif
396                         if (s->s3->tmp.cert_req)
397                                 s->state=SSL3_ST_CW_CERT_A;
398                         else
399                                 s->state=SSL3_ST_CW_KEY_EXCH_A;
400                         s->init_num=0;

402                         break;

404                 case SSL3_ST_CW_CERT_A:
405                 case SSL3_ST_CW_CERT_B:
406                 case SSL3_ST_CW_CERT_C:
407                 case SSL3_ST_CW_CERT_D:
408                         ret=ssl3_send_client_certificate(s);
409                         if (ret <= 0) goto end;
410                         s->state=SSL3_ST_CW_KEY_EXCH_A;
411                         s->init_num=0;
412                         break;

414                 case SSL3_ST_CW_KEY_EXCH_A:
415                 case SSL3_ST_CW_KEY_EXCH_B:
416                         ret=ssl3_send_client_key_exchange(s);
417                         if (ret <= 0) goto end;
418                         /* EAY EAY EAY need to check for DH fix cert
419                          * sent back */
420                         /* For TLS, cert_req is set to 2, so a cert chain
421                          * of nothing is sent, but no verify packet is sent */
422                         /* XXX: For now, we do not support client
423                          * authentication in ECDH cipher suites with
424                          * ECDH (rather than ECDSA) certificates.
425                          * We need to skip the certificate verify
426                          * message when client's ECDH public key is sent
427                          * inside the client certificate.
428                          */
429                         if (s->s3->tmp.cert_req == 1)
430                                 {
431                                 s->state=SSL3_ST_CW_CERT_VRFY_A;
432                                 }
433                         else
434                                 {
435                                 s->state=SSL3_ST_CW_CHANGE_A;
436                                 s->s3->change_cipher_spec=0;
437                                 }
438                         if (s->s3->flags & TLS1_FLAGS_SKIP_CERT_VERIFY)
439                                 {
440                                 s->state=SSL3_ST_CW_CHANGE_A;
441                                 s->s3->change_cipher_spec=0;
442                                 }

444                         s->init_num=0;
445                         break;

447                 case SSL3_ST_CW_CERT_VRFY_A:
448                 case SSL3_ST_CW_CERT_VRFY_B:
449                         ret=ssl3_send_client_verify(s);
450                         if (ret <= 0) goto end;
451                         s->state=SSL3_ST_CW_CHANGE_A;
452                         s->init_num=0;
453                         s->s3->change_cipher_spec=0;
454                         break;

456                 case SSL3_ST_CW_CHANGE_A:
457                 case SSL3_ST_CW_CHANGE_B:
```



```
458                         ret=ssl3_send_change_cipher_spec(s,
459                                 SSL3_ST_CW_CHANGE_A,SSL3_ST_CW_CHANGE_B);
460                         if (ret <= 0) goto end;

462 #if defined(OPENSSL_NO_TLSEXT) || defined(OPENSSL_NO_NEXTPROTONEG)
463                         s->state=SSL3_ST_CW_FINISHED_A;
464 #else
465                         if (s->s3->next_proto_neg_seen)
466                                 s->state=SSL3_ST_CW_NEXT_PROTO_A;
467                         else
468                                 s->state=SSL3_ST_CW_FINISHED_A;
469 #endif
470                         s->init_num=0;

472                         s->session->cipher=s->s3->tmp.new_cipher;
473 #ifdef OPENSSL_NO_COMP
474                         s->session->compress_meth=0;
475 #else
476                         if (s->s3->tmp.new_compression == NULL)
477                                 s->session->compress_meth=0;
478                         else
479                                 s->session->compress_meth=
480                                         s->s3->tmp.new_compression->id;
481 #endif
482                         if (!s->method->ssl3_enc->setup_key_block(s))
483                                 {
484                                 ret= -1;
485                                 goto end;
486                                 }

488                         if (!s->method->ssl3_enc->change_cipher_state(s,
489                                 SSL3_CHANGE_CIPHER_CLIENT_WRITE))
490                                 {
491                                 ret= -1;
492                                 goto end;
493                                 }

495                         break;

497 #if !defined(OPENSSL_NO_TLSEXT) && !defined(OPENSSL_NO_NEXTPROTONEG)
498                 case SSL3_ST_CW_NEXT_PROTO_A:
499                 case SSL3_ST_CW_NEXT_PROTO_B:
500                         ret=ssl3_send_next_proto(s);
501                         if (ret <= 0) goto end;
502                         s->state=SSL3_ST_CW_FINISHED_A;
503                         break;
504 #endif

506                 case SSL3_ST_CW_FINISHED_A:
507                 case SSL3_ST_CW_FINISHED_B:
508                         ret=ssl3_send_finished(s,
509                                 SSL3_ST_CW_FINISHED_A,SSL3_ST_CW_FINISHED_B,
510                                 s->method->ssl3_enc->client_finished_label,
511                                 s->method->ssl3_enc->client_finished_label_len);
512                         if (ret <= 0) goto end;
513                         s->s3->flags |= SSL3_FLAGS_CCS_OK;
514                         s->state=SSL3_ST_CW_FLUSH;

516                         /* clear flags */
517                         s->s3->flags&= ~SSL3_FLAGS_POP_BUFFER;
518                         if (s->hit)
519                                 {
520                                 s->s3->tmp.next_state=SSL_ST_OK;
521                                 if (s->s3->flags & SSL3_FLAGS_DELAY_CLIENT_FINIS
522                                         {
523                                         s->state=SSL_ST_OK;
```

```
524                                     s->s3->flags|=SSL3_FLAGS_POP_BUFFER;
525                                     s->s3->delay_buf_pop_ret=0;
526                                     }
527                             }
528                     else
529                             {
530 #ifndef OPENSSL_NO_TLSEXT
531                             /* Allow NewSessionTicket if ticket expected */
532                             if (s->tlsext_ticket_expected)
533                                     s->s3->tmp.next_state=SSL3_ST_CR_SESSION
534                             else
535 #endif
536
537                             s->s3->tmp.next_state=SSL3_ST_CR_FINISHED_A;
538                             }
539                     s->init_num=0;
540                     break;
541
542 #ifndef OPENSSL_NO_TLSEXT
543                 case SSL3_ST_CR_SESSION_TICKET_A:
544                 case SSL3_ST_CR_SESSION_TICKET_B:
545                         ret=ssl3_get_new_session_ticket(s);
546                         if (ret <= 0) goto end;
547                         s->state=SSL3_ST_CR_FINISHED_A;
548                         s->init_num=0;
549                 break;
550
551                 case SSL3_ST_CR_CERT_STATUS_A:
552                 case SSL3_ST_CR_CERT_STATUS_B:
553                         ret=ssl3_get_cert_status(s);
554                         if (ret <= 0) goto end;
555                         s->state=SSL3_ST_CR_KEY_EXCH_A;
556                         s->init_num=0;
557                 break;
558 #endif
559
560                 case SSL3_ST_CR_FINISHED_A:
561                 case SSL3_ST_CR_FINISHED_B:
562
563                         s->s3->flags |= SSL3_FLAGS_CCS_OK;
564                         ret=ssl3_get_finished(s,SSL3_ST_CR_FINISHED_A,
565                                 SSL3_ST_CR_FINISHED_B);
566                         if (ret <= 0) goto end;
567
568                         if (s->hit)
569                                 s->state=SSL3_ST_CW_CHANGE_A;
570                         else
571                                 s->state=SSL_ST_OK;
572                         s->init_num=0;
573                         break;
574
575                 case SSL3_ST_CW_FLUSH:
576                         s->rwstate=SSL_WRITING;
577                         if (BIO_flush(s->wbio) <= 0)
578                                 {
579                                 ret= -1;
580                                 goto end;
581                                 }
582                         s->rwstate=SSL_NOTHING;
583                         s->state=s->s3->tmp.next_state;
584                         break;
585
586                 case SSL_ST_OK:
587                         /* clean a few things up */
588                         ssl3_cleanup_key_block(s);
```



```
590                         if (s->init_buf != NULL)
591                                 {
592                                 BUF_MEM_free(s->init_buf);
593                                 s->init_buf=NULL;
594                                 }
595
596                         /* If we are not 'joining' the last two packets,
597                          * remove the buffering now */
598                         if (!(s->s3->flags & SSL3_FLAGS_POP_BUFFER))
599                                 ssl_free_wbio_buffer(s);
600                         /* else do it later in ssl3_write */
601
602                         s->init_num=0;
603                         s->renegotiate=0;
604                         s->new_session=0;
605
606                         ssl_update_cache(s,SSL_SESS_CACHE_CLIENT);
607                         if (s->hit) s->ctx->stats.sess_hit++;
608
609                         ret=1;
610                         /* s->server=0; */
611                         s->handshake_func=ssl3_connect;
612                         s->ctx->stats.sess_connect_good++;
613
614                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_DONE,1);
615
616                         goto end;
617                         /* break; */
618
619                 default:
620                         SSLerr(SSL_F_SSL3_CONNECT,SSL_R_UNKNOWN_STATE);
621                         ret= -1;
622                         goto end;
623                         /* break; */
624                         }
625
626                 /* did we do anything */
627                 if (!s->s3->tmp.reuse_message && !skip)
628                         {
629                         if (s->debug)
630                                 {
631                                 if ((ret=BIO_flush(s->wbio)) <= 0)
632                                         goto end;
633                                 }
634
635                         if ((cb != NULL) && (s->state != state))
636                                 {
637                                 new_state=s->state;
638                                 s->state=state;
639                                 cb(s,SSL_CB_CONNECT_LOOP,1);
640                                 s->state=new_state;
641                                 }
642                         }
643                 skip=0;
644                 }
645 end:
646         s->in_handshake--;
647         if (buf != NULL)
648                 BUF_MEM_free(buf);
649         if (cb != NULL)
650                 cb(s,SSL_CB_CONNECT_EXIT,ret);
651         return(ret);
652         }
653
654
655 int ssl3_client_hello(SSL *s)
```

```
656         {
657         unsigned char *buf;
658         unsigned char *p,*d;
659         int i;
660         unsigned long l;
661 #ifndef OPENSSL_NO_COMP
662         int j;
663         SSL_COMP *comp;
664 #endif

666         buf=(unsigned char *)s->init_buf->data;
667         if (s->state == SSL3_ST_CW_CLNT_HELLO_A)
668                 {
669                 SSL_SESSION *sess = s->session;
670                 if ((sess == NULL) ||
671                         (sess->ssl_version != s->version) ||
672 #ifdef OPENSSL_NO_TLSEXT
673                         !sess->session_id_length ||
674 #else
675                         (!sess->session_id_length && !sess->tlsext_tick) ||
676 #endif
677                         (sess->not_resumable))
678                         {
679                         if (!ssl_get_new_session(s,0))
680                                 goto err;
681                         }
682                 /* else use the pre-loaded session */

684                 p=s->s3->client_random;

686                 if (ssl_fill_hello_random(s, 0, p, SSL3_RANDOM_SIZE) <= 0)
687                         goto err;

689                 /* Do the message type and length last */
690                 d=p= &(buf[4]);

692                 /* version indicates the negotiated version: for example from
693                  * an SSLv2/v3 compatible client hello). The client_version
694                  * field is the maximum version we permit and it is also
695                  * used in RSA encrypted premaster secrets. Some servers can
696                  * choke if we initially report a higher version then
697                  * renegotiate to a lower one in the premaster secret. This
698                  * didn't happen with TLS 1.0 as most servers supported it
699                  * but it can with TLS 1.1 or later if the server only supports
700                  * 1.0.
701                  *
702                  * Possible scenario with previous logic:
703                  *      1. Client hello indicates TLS 1.2
704                  *      2. Server hello says TLS 1.0
705                  *      3. RSA encrypted premaster secret uses 1.2.
706                  *      4. Handhaked proceeds using TLS 1.0.
707                  *      5. Server sends hello request to renegotiate.
708                  *      6. Client hello indicates TLS v1.0 as we now
709                  *         know that is maximum server supports.
710                  *      7. Server chokes on RSA encrypted premaster secret
711                  *         containing version 1.0.
712                  *
713                  * For interoperability it should be OK to always use the
714                  * maximum version we support in client hello and then rely
715                  * on the checking of version to ensure the servers isn't
716                  * being inconsistent: for example initially negotiating with
717                  * TLS 1.0 and renegotiating with TLS 1.2. We do this by using
718                  * client_version in client hello and not resetting it to
719                  * the negotiated version.
720                  */
721 #if 0
```



```
722                 *(p++)=s->version>>8;
723                 *(p++)=s->version&0xff;
724                 s->client_version=s->version;
725 #else
726                 *(p++)=s->client_version>>8;
727                 *(p++)=s->client_version&0xff;
728 #endif

730                 /* Random stuff */
731                 memcpy(p,s->s3->client_random,SSL3_RANDOM_SIZE);
732                 p+=SSL3_RANDOM_SIZE;

734                 /* Session ID */
735                 if (s->new_session)
736                         i=0;
737                 else
738                         i=s->session->session_id_length;
739                 *(p++)=i;
740                 if (i != 0)
741                         {
742                         if (i > (int)sizeof(s->session->session_id))
743                                 {
744                                 SSLerr(SSL_F_SSL3_CLIENT_HELLO, ERR_R_INTERNAL_E
745                                 goto err;
746                                 }
747                         memcpy(p,s->session->session_id,i);
748                         p+=i;
749                         }

751                 /* Ciphers supported */
752                 i=ssl_cipher_list_to_bytes(s,SSL_get_ciphers(s),&(p[2]),0);
753                 if (i == 0)
754                         {
755                         SSLerr(SSL_F_SSL3_CLIENT_HELLO,SSL_R_NO_CIPHERS_AVAILABL
756                         goto err;
757                         }
758 #ifdef OPENSSL_MAX_TLS1_2_CIPHER_LENGTH
759                         /* Some servers hang if client hello > 256 bytes
760                          * as hack workaround chop number of supported ciphers
761                          * to keep it well below this if we use TLS v1.2
762                          */
763                         if (TLS1_get_version(s) >= TLS1_2_VERSION
764                                 && i > OPENSSL_MAX_TLS1_2_CIPHER_LENGTH)
765                                 i = OPENSSL_MAX_TLS1_2_CIPHER_LENGTH & ~1;
766 #endif
767                 s2n(i,p);
768                 p+=i;

770                 /* COMPRESSION */
771 #ifdef OPENSSL_NO_COMP
772                 *(p++)=1;
773 #else

775                 if ((s->options & SSL_OP_NO_COMPRESSION)
776                                         || !s->ctx->comp_methods)
777                         j=0;
778                 else
779                         j=sk_SSL_COMP_num(s->ctx->comp_methods);
780                 *(p++)=1+j;
781                 for (i=0; i<j; i++)
782                         {
783                         comp=sk_SSL_COMP_value(s->ctx->comp_methods,i);
784                         *(p++)=comp->id;
785                         }
786 #endif
787                 *(p++)=0; /* Add the NULL method */
```

```
#ifndef OPENSSL_NO_TLSEXT
                /* TLS extensions*/
                if (ssl_prepare_clienthello_tlsext(s) <= 0)
                        {
                        SSLerr(SSL_F_SSL3_CLIENT_HELLO,SSL_R_CLIENTHELLO_TLSEXT)
                        goto err;
                        }
                if ((p = ssl_add_clienthello_tlsext(s, p, buf+SSL3_RT_MAX_PLAIN_
                        {
                        SSLerr(SSL_F_SSL3_CLIENT_HELLO,ERR_R_INTERNAL_ERROR);
                        goto err;
                        }
#endif

                l=(p-d);
                d=buf;
                *(d++)=SSL3_MT_CLIENT_HELLO;
                l2n3(l,d);

                s->state=SSL3_ST_CW_CLNT_HELLO_B;
                /* number of bytes to write */
                s->init_num=p-buf;
                s->init_off=0;
                }

        /* SSL3_ST_CW_CLNT_HELLO_B */
        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
err:
        return(-1);
        }

int ssl3_get_server_hello(SSL *s)
        {
        STACK_OF(SSL_CIPHER) *sk;
        const SSL_CIPHER *c;
        unsigned char *p,*d;
        int i,al,ok;
        unsigned int j;
        long n;
#ifndef OPENSSL_NO_COMP
        SSL_COMP *comp;
#endif

        n=s->method->ssl_get_message(s,
                SSL3_ST_CR_SRVR_HELLO_A,
                SSL3_ST_CR_SRVR_HELLO_B,
                -1,
                20000, /* ?? */
                &ok);

        if (!ok) return((int)n);

        if ( SSL_version(s) == DTLS1_VERSION || SSL_version(s) == DTLS1_BAD_VER)
                {
                if ( s->s3->tmp.message_type == DTLS1_MT_HELLO_VERIFY_REQUEST)
                        {
                        if ( s->d1->send_cookie == 0)
                                {
                                s->s3->tmp.reuse_message = 1;
                                return 1;
                                }
                        else /* already sent a cookie */
                                {
                                al=SSL_AD_UNEXPECTED_MESSAGE;
                                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_BAD_MES
```



```
                                goto f_err;
                                }
                        }
                }

        if ( s->s3->tmp.message_type != SSL3_MT_SERVER_HELLO)
                {
                al=SSL_AD_UNEXPECTED_MESSAGE;
                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_BAD_MESSAGE_TYPE);
                goto f_err;
                }

        d=p=(unsigned char *)s->init_msg;

        if ((p[0] != (s->version>>8)) || (p[1] != (s->version&0xff)))
                {
                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_WRONG_SSL_VERSION);
                s->version=(s->version&0xff00)|p[1];
                al=SSL_AD_PROTOCOL_VERSION;
                goto f_err;
                }
        p+=2;

        /* load the server hello data */
        /* load the server random */
        memcpy(s->s3->server_random,p,SSL3_RANDOM_SIZE);
        p+=SSL3_RANDOM_SIZE;

        /* get the session-id */
        j= *(p++);

        if ((j > sizeof s->session->session_id) || (j > SSL3_SESSION_ID_SIZE))
                {
                al=SSL_AD_ILLEGAL_PARAMETER;
                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_SSL3_SESSION_ID_TOO_LON
                goto f_err;
                }

#ifndef OPENSSL_NO_TLSEXT
        /* check if we want to resume the session based on external pre-shared s
        if (s->version >= TLS1_VERSION && s->tls_session_secret_cb)
                {
                SSL_CIPHER *pref_cipher=NULL;
                s->session->master_key_length=sizeof(s->session->master_key);
                if (s->tls_session_secret_cb(s, s->session->master_key,
                                                &s->session->master_key_length,
                                                NULL, &pref_cipher,
                                                s->tls_session_secret_cb_arg))
                        {
                        s->session->cipher = pref_cipher ?
                                pref_cipher : ssl_get_cipher_by_char(s, p+j);
                        s->s3->flags |= SSL3_FLAGS_CCS_OK;
                        }
                }
#endif /* OPENSSL_NO_TLSEXT */

        if (j != 0 && j == s->session->session_id_length
            && memcmp(p,s->session->session_id,j) == 0)
            {
            if(s->sid_ctx_length != s->session->sid_ctx_length
               || memcmp(s->session->sid_ctx,s->sid_ctx,s->sid_ctx_length))
                {
                /* actually a client application bug */
                al=SSL_AD_ILLEGAL_PARAMETER;
                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_ATTEMPT_TO_REUSE_SESSIO
                goto f_err;
```

```
920                 }
921             s->s3->flags |= SSL3_FLAGS_CCS_OK;
922             s->hit=1;
923             }
924         else    /* a miss or crap from the other end */
925                 {
926                 /* If we were trying for session-id reuse, make a new
927                  * SSL_SESSION so we don't stuff up other people */
928                 s->hit=0;
929                 if (s->session->session_id_length > 0)
930                         {
931                         if (!ssl_get_new_session(s,0))
932                                 {
933                                 al=SSL_AD_INTERNAL_ERROR;
934                                 goto f_err;
935                                 }
936                         }
937                 s->session->session_id_length=j;
938                 memcpy(s->session->session_id,p,j); /* j could be 0 */
939                 }
940         p+=j;
941         c=ssl_get_cipher_by_char(s,p);
942         if (c == NULL)
943                 {
944                 /* unknown cipher */
945                 al=SSL_AD_ILLEGAL_PARAMETER;
946                 SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_UNKNOWN_CIPHER_RETURNED
947                 goto f_err;
948                 }
949         /* TLS v1.2 only ciphersuites require v1.2 or later */
950         if ((c->algorithm_ssl & SSL_TLSV1_2) &&
951                 (TLS1_get_version(s) < TLS1_2_VERSION))
952                 {
953                 al=SSL_AD_ILLEGAL_PARAMETER;
954                 SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_WRONG_CIPHER_RETURNED);
955                 goto f_err;
956                 }
957 #ifndef OPENSSL_NO_SRP
958         if (((c->algorithm_mkey & SSL_kSRP) || (c->algorithm_auth & SSL_aSRP)) &
959                     !(s->srp_ctx.srp_Mask & SSL_kSRP))
960                 {
961                 al=SSL_AD_ILLEGAL_PARAMETER;
962                 SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_WRONG_CIPHER_RETURNED);
963                 goto f_err;
964                 }
965 #endif /* OPENSSL_NO_SRP */
966         p+=ssl_put_cipher_by_char(s,NULL,NULL);

968         sk=ssl_get_ciphers_by_id(s);
969         i=sk_SSL_CIPHER_find(sk,c);
970         if (i < 0)
971                 {
972                 /* we did not say we would use this cipher */
973                 al=SSL_AD_ILLEGAL_PARAMETER;
974                 SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_WRONG_CIPHER_RETURNED);
975                 goto f_err;
976                 }

978         /* Depending on the session caching (internal/external), the cipher
979            and/or cipher_id values may not be set. Make sure that
980            cipher_id is set and use it for comparison. */
981         if (s->session->cipher)
982                 s->session->cipher_id = s->session->cipher->id;
983         if (s->hit && (s->session->cipher_id != c->id))
984                 {
985 /* Workaround is now obsolete */
```



```
986 #if 0
987                 if (!(s->options &
988                         SSL_OP_NETSCAPE_REUSE_CIPHER_CHANGE_BUG))
989 #endif
990                         {
991                         al=SSL_AD_ILLEGAL_PARAMETER;
992                         SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_OLD_SESSION_CIP
993                         goto f_err;
994                         }
995                 }
996         s->s3->tmp.new_cipher=c;
997         /* Don't digest cached records if TLS v1.2: we may need them for
998          * client authentication.
999          */
1000        if (TLS1_get_version(s) < TLS1_2_VERSION && !ssl3_digest_cached_records(
1001                {
1002                al = SSL_AD_INTERNAL_ERROR;
1003                goto f_err;
1004                }
1005        /* lets get the compression algorithm */
1006        /* COMPRESSION */
1007 #ifdef OPENSSL_NO_COMP
1008        if (*(p++) != 0)
1009                {
1010                al=SSL_AD_ILLEGAL_PARAMETER;
1011                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_UNSUPPORTED_COMPRESSION
1012                goto f_err;
1013                }
1014        /* If compression is disabled we'd better not try to resume a session
1015         * using compression.
1016         */
1017        if (s->session->compress_meth != 0)
1018                {
1019                al=SSL_AD_INTERNAL_ERROR;
1020                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_INCONSISTENT_COMPRESSIO
1021                goto f_err;
1022                }
1023 #else
1024        j= *(p++);
1025        if (s->hit && j != s->session->compress_meth)
1026                {
1027                al=SSL_AD_ILLEGAL_PARAMETER;
1028                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_OLD_SESSION_COMPRESSION
1029                goto f_err;
1030                }
1031        if (j == 0)
1032                comp=NULL;
1033        else if (s->options & SSL_OP_NO_COMPRESSION)
1034                {
1035                al=SSL_AD_ILLEGAL_PARAMETER;
1036                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_COMPRESSION_DISABLED);
1037                goto f_err;
1038                }
1039        else
1040                comp=ssl3_comp_find(s->ctx->comp_methods,j);

1042        if ((j != 0) && (comp == NULL))
1043                {
1044                al=SSL_AD_ILLEGAL_PARAMETER;
1045                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_UNSUPPORTED_COMPRESSION
1046                goto f_err;
1047                }
1048        else
1049                {
1050                s->s3->tmp.new_compression=comp;
1051                }
```

```
#endif

#ifndef OPENSSL_NO_TLSEXT
        /* TLS extensions*/
        if (s->version >= SSL3_VERSION)
                {
                if (!ssl_parse_serverhello_tlsext(s,&p,d,n, &al))
                        {
                        /* 'al' set by ssl_parse_serverhello_tlsext */
                        SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_PARSE_TLSEXT);
                        goto f_err;
                        }
                if (ssl_check_serverhello_tlsext(s) <= 0)
                        {
                        SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_SERVERHELLO_TLS
                                goto err;
                        }
                }
#endif

        if (p != (d+n))
                {
                /* wrong packet length */
                al=SSL_AD_DECODE_ERROR;
                SSLerr(SSL_F_SSL3_GET_SERVER_HELLO,SSL_R_BAD_PACKET_LENGTH);
                goto f_err;
                }

        return(1);
f_err:
        ssl3_send_alert(s,SSL3_AL_FATAL,al);
err:
        return(-1);
        }

int ssl3_get_server_certificate(SSL *s)
        {
        int al,i,ok,ret= -1;
        unsigned long n,nc,llen,l;
        X509 *x=NULL;
        const unsigned char *q,*p;
        unsigned char *d;
        STACK_OF(X509) *sk=NULL;
        SESS_CERT *sc;
        EVP_PKEY *pkey=NULL;
        int need_cert = 1; /* VRS: 0=> will allow null cert if auth == KRB5 */

        n=s->method->ssl_get_message(s,
                SSL3_ST_CR_CERT_A,
                SSL3_ST_CR_CERT_B,
                -1,
                s->max_cert_list,
                &ok);

        if (!ok) return((int)n);

        if ((s->s3->tmp.message_type == SSL3_MT_SERVER_KEY_EXCHANGE) ||
                ((s->s3->tmp.new_cipher->algorithm_auth & SSL_aKRB5) &&
                (s->s3->tmp.message_type == SSL3_MT_SERVER_DONE)))
                {
                s->s3->tmp.reuse_message=1;
                return(1);
                }

        if (s->s3->tmp.message_type != SSL3_MT_CERTIFICATE)
                {
```



```
                al=SSL_AD_UNEXPECTED_MESSAGE;
                SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,SSL_R_BAD_MESSAGE_TYPE)
                goto f_err;
                }
        p=d=(unsigned char *)s->init_msg;

        if ((sk=sk_X509_new_null()) == NULL)
                {
                SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,ERR_R_MALLOC_FAILURE);
                goto err;
                }

        n2l3(p,llen);
        if (llen+3 != n)
                {
                al=SSL_AD_DECODE_ERROR;
                SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,SSL_R_LENGTH_MISMATCH);
                goto f_err;
                }
        for (nc=0; nc<llen; )
                {
                n2l3(p,l);
                if ((l+nc+3) > llen)
                        {
                        al=SSL_AD_DECODE_ERROR;
                        SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,SSL_R_CERT_LENG
                        goto f_err;
                        }

                q=p;
                x=d2i_X509(NULL,&q,l);
                if (x == NULL)
                        {
                        al=SSL_AD_BAD_CERTIFICATE;
                        SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,ERR_R_ASN1_LIB)
                        goto f_err;
                        }
                if (q != (p+l))
                        {
                        al=SSL_AD_DECODE_ERROR;
                        SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,SSL_R_CERT_LENG
                        goto f_err;
                        }
                if (!sk_X509_push(sk,x))
                        {
                        SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,ERR_R_MALLOC_FA
                        goto err;
                        }
                x=NULL;
                nc+=l+3;
                p=q;
                }

        i=ssl_verify_cert_chain(s,sk);
        if ((s->verify_mode != SSL_VERIFY_NONE) && (i <= 0)
#ifndef OPENSSL_NO_KRB5
            && !((s->s3->tmp.new_cipher->algorithm_mkey & SSL_kKRB5) &&
                 (s->s3->tmp.new_cipher->algorithm_auth & SSL_aKRB5))
#endif /* OPENSSL_NO_KRB5 */
                )
                {
                al=ssl_verify_alarm_type(s->verify_result);
                SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,SSL_R_CERTIFICATE_VERIF
                goto f_err;
                }
        ERR_clear_error(); /* but we keep s->verify_result */
```

```
1185         sc=ssl_sess_cert_new();
1186         if (sc == NULL) goto err;

1188         if (s->session->sess_cert) ssl_sess_cert_free(s->session->sess_cert);
1189         s->session->sess_cert=sc;

1191         sc->cert_chain=sk;
1192         /* Inconsistency alert: cert_chain does include the peer's
1193          * certificate, which we don't include in s3_srvr.c */
1194         x=sk_X509_value(sk,0);
1195         sk=NULL;
1196         /* VRS 19990621: possible memory leak; sk=null ==> !sk_pop_free() @end*/

1198         pkey=X509_get_pubkey(x);

1200         /* VRS: allow null cert if auth == KRB5 */
1201         need_cert = ((s->s3->tmp.new_cipher->algorithm_mkey & SSL_kKRB5) &&
1202                      (s->s3->tmp.new_cipher->algorithm_auth & SSL_aKRB5))
1203                      ? 0 : 1;

1205 #ifdef KSSL_DEBUG
1206         printf("pkey,x = %p, %p\n", pkey,x);
1207         printf("ssl_cert_type(x,pkey) = %d\n", ssl_cert_type(x,pkey));
1208         printf("cipher, alg, nc = %s, %lx, %lx, %d\n", s->s3->tmp.new_cipher->na
1209                s->s3->tmp.new_cipher->algorithm_mkey, s->s3->tmp.new_cipher->al
1210 #endif    /* KSSL_DEBUG */

1212         if (need_cert && ((pkey == NULL) || EVP_PKEY_missing_parameters(pkey)))
1213                 {
1214                 x=NULL;
1215                 al=SSL3_AL_FATAL;
1216                 SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,
1217                        SSL_R_UNABLE_TO_FIND_PUBLIC_KEY_PARAMETERS);
1218                 goto f_err;
1219                 }

1221         i=ssl_cert_type(x,pkey);
1222         if (need_cert && i < 0)
1223                 {
1224                 x=NULL;
1225                 al=SSL3_AL_FATAL;
1226                 SSLerr(SSL_F_SSL3_GET_SERVER_CERTIFICATE,
1227                        SSL_R_UNKNOWN_CERTIFICATE_TYPE);
1228                 goto f_err;
1229                 }

1231         if (need_cert)
1232                 {
1233                 sc->peer_cert_type=i;
1234                 CRYPTO_add(&x->references,1,CRYPTO_LOCK_X509);
1235                 /* Why would the following ever happen?
1236                  * We just created sc a couple of lines ago. */
1237                 if (sc->peer_pkeys[i].x509 != NULL)
1238                        X509_free(sc->peer_pkeys[i].x509);
1239                 sc->peer_pkeys[i].x509=x;
1240                 sc->peer_key= &(sc->peer_pkeys[i]);

1242                 if (s->session->peer != NULL)
1243                        X509_free(s->session->peer);
1244                 CRYPTO_add(&x->references,1,CRYPTO_LOCK_X509);
1245                 s->session->peer=x;
1246                 }
1247         else
1248                 {
1249                 sc->peer_cert_type=i;
```



```
1250                 sc->peer_key= NULL;

1252                 if (s->session->peer != NULL)
1253                        X509_free(s->session->peer);
1254                 s->session->peer=NULL;
1255                 }
1256         s->session->verify_result = s->verify_result;

1258         x=NULL;
1259         ret=1;

1261         if (0)
1262                 {
1263 f_err:
1264                 ssl3_send_alert(s,SSL3_AL_FATAL,al);
1265                 }
1266 err:
1267         EVP_PKEY_free(pkey);
1268         X509_free(x);
1269         sk_X509_pop_free(sk,X509_free);
1270         return(ret);
1271         }

1273 int ssl3_get_key_exchange(SSL *s)
1274         {
1275 #ifndef OPENSSL_NO_RSA
1276         unsigned char *q,md_buf[EVP_MAX_MD_SIZE*2];
1277 #endif
1278         EVP_MD_CTX md_ctx;
1279         unsigned char *param,*p;
1280         int al,i,j,param_len,ok;
1281         long n,alg_k,alg_a;
1282         EVP_PKEY *pkey=NULL;
1283         const EVP_MD *md = NULL;
1284 #ifndef OPENSSL_NO_RSA
1285         RSA *rsa=NULL;
1286 #endif
1287 #ifndef OPENSSL_NO_DH
1288         DH *dh=NULL;
1289 #endif
1290 #ifndef OPENSSL_NO_ECDH
1291         EC_KEY *ecdh = NULL;
1292         BN_CTX *bn_ctx = NULL;
1293         EC_POINT *srvr_ecpoint = NULL;
1294         int curve_nid = 0;
1295         int encoded_pt_len = 0;
1296 #endif

1298         /* use same message size as in ssl3_get_certificate_request()
1299          * as ServerKeyExchange message may be skipped */
1300         n=s->method->ssl_get_message(s,
1301                SSL3_ST_CR_KEY_EXCH_A,
1302                SSL3_ST_CR_KEY_EXCH_B,
1303                -1,
1304                s->max_cert_list,
1305                &ok);
1306         if (!ok) return((int)n);

1308         if (s->s3->tmp.message_type != SSL3_MT_SERVER_KEY_EXCHANGE)
1309                 {
1310 #ifndef OPENSSL_NO_PSK
1311                 /* In plain PSK ciphersuite, ServerKeyExchange can be
1312                    omitted if no identity hint is sent. Set
1313                    session->sess_cert anyway to avoid problems
1314                    later.*/
1315                 if (s->s3->tmp.new_cipher->algorithm_mkey & SSL_kPSK)
```

```
1316                        {
1317                        s->session->sess_cert=ssl_sess_cert_new();
1318                        if (s->ctx->psk_identity_hint)
1319                                OPENSSL_free(s->ctx->psk_identity_hint);
1320                        s->ctx->psk_identity_hint = NULL;
1321                        }
1322 #endif
1323                s->s3->tmp.reuse_message=1;
1324                return(1);
1325                }

1327        param=p=(unsigned char *)s->init_msg;
1328        if (s->session->sess_cert != NULL)
1329                {
1330 #ifndef OPENSSL_NO_RSA
1331                if (s->session->sess_cert->peer_rsa_tmp != NULL)
1332                        {
1333                        RSA_free(s->session->sess_cert->peer_rsa_tmp);
1334                        s->session->sess_cert->peer_rsa_tmp=NULL;
1335                        }
1336 #endif
1337 #ifndef OPENSSL_NO_DH
1338                if (s->session->sess_cert->peer_dh_tmp)
1339                        {
1340                        DH_free(s->session->sess_cert->peer_dh_tmp);
1341                        s->session->sess_cert->peer_dh_tmp=NULL;
1342                        }
1343 #endif
1344 #ifndef OPENSSL_NO_ECDH
1345                if (s->session->sess_cert->peer_ecdh_tmp)
1346                        {
1347                        EC_KEY_free(s->session->sess_cert->peer_ecdh_tmp);
1348                        s->session->sess_cert->peer_ecdh_tmp=NULL;
1349                        }
1350 #endif
1351                }
1352        else
1353                {
1354                s->session->sess_cert=ssl_sess_cert_new();
1355                }

1357        param_len=0;
1358        alg_k=s->s3->tmp.new_cipher->algorithm_mkey;
1359        alg_a=s->s3->tmp.new_cipher->algorithm_auth;
1360        EVP_MD_CTX_init(&md_ctx);

1362 #ifndef OPENSSL_NO_PSK
1363        if (alg_k & SSL_kPSK)
1364                {
1365                char tmp_id_hint[PSK_MAX_IDENTITY_LEN+1];

1367                al=SSL_AD_HANDSHAKE_FAILURE;
1368                n2s(p,i);
1369                param_len=i+2;
1370                /* Store PSK identity hint for later use, hint is used
1371                 * in ssl3_send_client_key_exchange.  Assume that the
1372                 * maximum length of a PSK identity hint can be as
1373                 * long as the maximum length of a PSK identity. */
1374                if (i > PSK_MAX_IDENTITY_LEN)
1375                        {
1376                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,
1377                               SSL_R_DATA_LENGTH_TOO_LONG);
1378                        goto f_err;
1379                        }
1380                if (param_len > n)
1381                        {
```



```
1382                        al=SSL_AD_DECODE_ERROR;
1383                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,
1384                               SSL_R_BAD_PSK_IDENTITY_HINT_LENGTH);
1385                        goto f_err;
1386                        }
1387                /* If received PSK identity hint contains NULL
1388                 * characters, the hint is truncated from the first
1389                 * NULL. p may not be ending with NULL, so create a
1390                 * NULL-terminated string. */
1391                memcpy(tmp_id_hint, p, i);
1392                memset(tmp_id_hint+i, 0, PSK_MAX_IDENTITY_LEN+1-i);
1393                if (s->ctx->psk_identity_hint != NULL)
1394                        OPENSSL_free(s->ctx->psk_identity_hint);
1395                s->ctx->psk_identity_hint = BUF_strdup(tmp_id_hint);
1396                if (s->ctx->psk_identity_hint == NULL)
1397                        {
1398                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE, ERR_R_MALLOC_FAILURE
1399                        goto f_err;
1400                        }

1402                p+=i;
1403                n-=param_len;
1404                }
1405        else
1406 #endif /* !OPENSSL_NO_PSK */
1407 #ifndef OPENSSL_NO_SRP
1408        if (alg_k & SSL_kSRP)
1409                {
1410                n2s(p,i);
1411                param_len=i+2;
1412                if (param_len > n)
1413                        {
1414                        al=SSL_AD_DECODE_ERROR;
1415                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_SRP_N_LENGT
1416                        goto f_err;
1417                        }
1418                if (!(s->srp_ctx.N=BN_bin2bn(p,i,NULL)))
1419                        {
1420                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1421                        goto err;
1422                        }
1423                p+=i;

1425                n2s(p,i);
1426                param_len+=i+2;
1427                if (param_len > n)
1428                        {
1429                        al=SSL_AD_DECODE_ERROR;
1430                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_SRP_G_LENGT
1431                        goto f_err;
1432                        }
1433                if (!(s->srp_ctx.g=BN_bin2bn(p,i,NULL)))
1434                        {
1435                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1436                        goto err;
1437                        }
1438                p+=i;

1440                i = (unsigned int)(p[0]);
1441                p++;
1442                param_len+=i+1;
1443                if (param_len > n)
1444                        {
1445                        al=SSL_AD_DECODE_ERROR;
1446                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_SRP_S_LENGT
1447                        goto f_err;
```

```
1448                    }
1449                if (!(s->srp_ctx.s=BN_bin2bn(p,i,NULL)))
1450                    {
1451                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1452                    goto err;
1453                    }
1454                p+=i;

1456                n2s(p,i);
1457                param_len+=i+2;
1458                if (param_len > n)
1459                    {
1460                    al=SSL_AD_DECODE_ERROR;
1461                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_SRP_B_LENGT
1462                    goto f_err;
1463                    }
1464                if (!(s->srp_ctx.B=BN_bin2bn(p,i,NULL)))
1465                    {
1466                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1467                    goto err;
1468                    }
1469                p+=i;
1470                n-=param_len;

1472                if (!srp_verify_server_param(s, &al))
1473                    {
1474                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_SRP_PARAMET
1475                    goto f_err;
1476                    }

1478 /* We must check if there is a certificate */
1479 #ifndef OPENSSL_NO_RSA
1480                if (alg_a & SSL_aRSA)
1481                    pkey=X509_get_pubkey(s->session->sess_cert->peer_pkeys[S
1482 #else
1483                if (0)
1484                    ;
1485 #endif
1486 #ifndef OPENSSL_NO_DSA
1487                else if (alg_a & SSL_aDSS)
1488                    pkey=X509_get_pubkey(s->session->sess_cert->peer_pkeys[S
1489 #endif
1490                }
1491        else
1492 #endif /* !OPENSSL_NO_SRP */
1493 #ifndef OPENSSL_NO_RSA
1494        if (alg_k & SSL_kRSA)
1495                {
1496                if ((rsa=RSA_new()) == NULL)
1497                    {
1498                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_MALLOC_FAILURE)
1499                    goto err;
1500                    }
1501                n2s(p,i);
1502                param_len=i+2;
1503                if (param_len > n)
1504                    {
1505                    al=SSL_AD_DECODE_ERROR;
1506                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_RSA_MODULUS
1507                    goto f_err;
1508                    }
1509                if (!(rsa->n=BN_bin2bn(p,i,rsa->n)))
1510                    {
1511                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1512                    goto err;
1513                    }
```



```
1514                p+=i;

1516                n2s(p,i);
1517                param_len+=i+2;
1518                if (param_len > n)
1519                    {
1520                    al=SSL_AD_DECODE_ERROR;
1521                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_RSA_E_LENGT
1522                    goto f_err;
1523                    }
1524                if (!(rsa->e=BN_bin2bn(p,i,rsa->e)))
1525                    {
1526                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1527                    goto err;
1528                    }
1529                p+=i;
1530                n-=param_len;

1532                /* this should be because we are using an export cipher */
1533                if (alg_a & SSL_aRSA)
1534                    pkey=X509_get_pubkey(s->session->sess_cert->peer_pkeys[S
1535                else
1536                    {
1537                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_INTERNAL_ERROR)
1538                    goto err;
1539                    }
1540                s->session->sess_cert->peer_rsa_tmp=rsa;
1541                rsa=NULL;
1542                }
1543 #else /* OPENSSL_NO_RSA */
1544        if (0)
1545                ;
1546 #endif
1547 #ifndef OPENSSL_NO_DH
1548        else if (alg_k & SSL_kEDH)
1549                {
1550                if ((dh=DH_new()) == NULL)
1551                    {
1552                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_DH_LIB);
1553                    goto err;
1554                    }
1555                n2s(p,i);
1556                param_len=i+2;
1557                if (param_len > n)
1558                    {
1559                    al=SSL_AD_DECODE_ERROR;
1560                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_DH_P_LENGTH
1561                    goto f_err;
1562                    }
1563                if (!(dh->p=BN_bin2bn(p,i,NULL)))
1564                    {
1565                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1566                    goto err;
1567                    }
1568                p+=i;

1570                n2s(p,i);
1571                param_len+=i+2;
1572                if (param_len > n)
1573                    {
1574                    al=SSL_AD_DECODE_ERROR;
1575                    SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_DH_G_LENGTH
1576                    goto f_err;
1577                    }
1578                if (!(dh->g=BN_bin2bn(p,i,NULL)))
1579                    {
```

```
1580                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1581                        goto err;
1582                        }
1583                p+=i;

1585                n2s(p,i);
1586                param_len+=i+2;
1587                if (param_len > n)
1588                        {
1589                        al=SSL_AD_DECODE_ERROR;
1590                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_DH_PUB_KEY_
1591                        goto f_err;
1592                        }
1593                if (!(dh->pub_key=BN_bin2bn(p,i,NULL)))
1594                        {
1595                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_BN_LIB);
1596                        goto err;
1597                        }
1598                p+=i;
1599                n-=param_len;

1601 #ifndef OPENSSL_NO_RSA
1602                if (alg_a & SSL_aRSA)
1603                        pkey=X509_get_pubkey(s->session->sess_cert->peer_pkeys[S
1604 #else
1605                if (0)
1606                        ;
1607 #endif
1608 #ifndef OPENSSL_NO_DSA
1609                else if (alg_a & SSL_aDSS)
1610                        pkey=X509_get_pubkey(s->session->sess_cert->peer_pkeys[S
1611 #endif
1612                /* else anonymous DH, so no certificate or pkey. */

1614                s->session->sess_cert->peer_dh_tmp=dh;
1615                dh=NULL;
1616                }
1617        else if ((alg_k & SSL_kDHr) || (alg_k & SSL_kDHd))
1618                {
1619                al=SSL_AD_ILLEGAL_PARAMETER;
1620                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_TRIED_TO_USE_UNSUPPORTE
1621                goto f_err;
1622                }
1623 #endif /* !OPENSSL_NO_DH */

1625 #ifndef OPENSSL_NO_ECDH
1626        else if (alg_k & SSL_kEECDH)
1627                {
1628                EC_GROUP *ngroup;
1629                const EC_GROUP *group;

1631                if ((ecdh=EC_KEY_new()) == NULL)
1632                        {
1633                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_MALLOC_FAILURE)
1634                        goto err;
1635                        }

1637                /* Extract elliptic curve parameters and the
1638                 * server's ephemeral ECDH public key.
1639                 * Keep accumulating lengths of various components in
1640                 * param_len and make sure it never exceeds n.
1641                 */

1643                /* XXX: For now we only support named (not generic) curves
1644                 * and the ECParameters in this case is just three bytes.
1645                 */
```



```
1646                param_len=3;
1647                if ((param_len > n) ||
1648                    (*p != NAMED_CURVE_TYPE) ||
1649                    ((curve_nid = tls1_ec_curve_id2nid(*(p + 2))) == 0))
1650                        {
1651                        al=SSL_AD_INTERNAL_ERROR;
1652                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_UNABLE_TO_FIND_
1653                        goto f_err;
1654                        }

1656                ngroup = EC_GROUP_new_by_curve_name(curve_nid);
1657                if (ngroup == NULL)
1658                        {
1659                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_EC_LIB);
1660                        goto err;
1661                        }
1662                if (EC_KEY_set_group(ecdh, ngroup) == 0)
1663                        {
1664                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_EC_LIB);
1665                        goto err;
1666                        }
1667                EC_GROUP_free(ngroup);

1669                group = EC_KEY_get0_group(ecdh);

1671                if (SSL_C_IS_EXPORT(s->s3->tmp.new_cipher) &&
1672                    (EC_GROUP_get_degree(group) > 163))
1673                        {
1674                        al=SSL_AD_EXPORT_RESTRICTION;
1675                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_ECGROUP_TOO_LAR
1676                        goto f_err;
1677                        }

1679                p+=3;

1681                /* Next, get the encoded ECPoint */
1682                if (((srvr_ecpoint = EC_POINT_new(group)) == NULL) ||
1683                    ((bn_ctx = BN_CTX_new()) == NULL))
1684                        {
1685                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_MALLOC_FAILURE)
1686                        goto err;
1687                        }

1689                encoded_pt_len = *p;  /* length of encoded point */
1690                p+=1;
1691                param_len += (1 + encoded_pt_len);
1692                if ((param_len > n) ||
1693                    (EC_POINT_oct2point(group, srvr_ecpoint,
1694                        p, encoded_pt_len, bn_ctx) == 0))
1695                        {
1696                        al=SSL_AD_DECODE_ERROR;
1697                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_ECPOINT);
1698                        goto f_err;
1699                        }

1701                n-=param_len;
1702                p+=encoded_pt_len;

1704                /* The ECC/TLS specification does not mention
1705                 * the use of DSA to sign ECParameters in the server
1706                 * key exchange message. We do support RSA and ECDSA.
1707                 */
1708                if (0) ;
1709 #ifndef OPENSSL_NO_RSA
1710                else if (alg_a & SSL_aRSA)
1711                        pkey=X509_get_pubkey(s->session->sess_cert->peer_pkeys[S
```

```
#endif
#ifndef OPENSSL_NO_ECDSA
                else if (alg_a & SSL_aECDSA)
                        pkey=X509_get_pubkey(s->session->sess_cert->peer_pkeys[S
#endif
                /* else anonymous ECDH, so no certificate or pkey. */
                EC_KEY_set_public_key(ecdh, srvr_ecpoint);
                s->session->sess_cert->peer_ecdh_tmp=ecdh;
                ecdh=NULL;
                BN_CTX_free(bn_ctx);
                bn_ctx = NULL;
                EC_POINT_free(srvr_ecpoint);
                srvr_ecpoint = NULL;
                }
        else if (alg_k)
                {
                al=SSL_AD_UNEXPECTED_MESSAGE;
                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_UNEXPECTED_MESSAGE);
                goto f_err;
                }
#endif /* !OPENSSL_NO_ECDH */


        /* p points to the next byte, there are 'n' bytes left */

        /* if it was signed, check the signature */
        if (pkey != NULL)
                {
                if (TLS1_get_version(s) >= TLS1_2_VERSION)
                        {
                        int sigalg = tls12_get_sigid(pkey);
                        /* Should never happen */
                        if (sigalg == -1)
                                {
                                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_INTERNA
                                goto err;
                                }
                        /* Check key type is consistent with signature */
                        if (sigalg != (int)p[1])
                                {
                                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_WRONG_S
                                al=SSL_AD_DECODE_ERROR;
                                goto f_err;
                                }
                        md = tls12_get_hash(p[0]);
                        if (md == NULL)
                                {
                                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_UNKNOWN
                                al=SSL_AD_DECODE_ERROR;
                                goto f_err;
                                }
#ifdef SSL_DEBUG
fprintf(stderr, "USING TLSv1.2 HASH %s\n", EVP_MD_name(md));
#endif
                        p += 2;
                        n -= 2;
                        }
                else
                        md = EVP_sha1();

                n2s(p,i);
                n-=2;
                j=EVP_PKEY_size(pkey);

                if ((i != n) || (n > j) || (n <= 0))
                        {
```



```
                        /* wrong packet length */
                        al=SSL_AD_DECODE_ERROR;
                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_WRONG_SIGNATURE
                        goto f_err;
                        }

#ifndef OPENSSL_NO_RSA
                if (pkey->type == EVP_PKEY_RSA && TLS1_get_version(s) < TLS1_2_V
                        {
                        int num;

                        j=0;
                        q=md_buf;
                        for (num=2; num > 0; num--)
                                {
                                EVP_MD_CTX_set_flags(&md_ctx,
                                        EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
                                EVP_DigestInit_ex(&md_ctx,(num == 2)
                                        ?s->ctx->md5:s->ctx->sha1, NULL);
                                EVP_DigestUpdate(&md_ctx,&(s->s3->client_random[
                                EVP_DigestUpdate(&md_ctx,&(s->s3->server_random[
                                EVP_DigestUpdate(&md_ctx,param,param_len);
                                EVP_DigestFinal_ex(&md_ctx,q,(unsigned int *)&i)
                                q+=i;
                                j+=i;
                                }
                        i=RSA_verify(NID_md5_sha1, md_buf, j, p, n,
                                                                pkey->pkey.rsa);
                        if (i < 0)
                                {
                                al=SSL_AD_DECRYPT_ERROR;
                                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_RSA
                                goto f_err;
                                }
                        if (i == 0)
                                {
                                /* bad signature */
                                al=SSL_AD_DECRYPT_ERROR;
                                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_SIG
                                goto f_err;
                                }
                        }
                else
#endif
                        {
                        EVP_VerifyInit_ex(&md_ctx, md, NULL);
                        EVP_VerifyUpdate(&md_ctx,&(s->s3->client_random[0]),SSL3
                        EVP_VerifyUpdate(&md_ctx,&(s->s3->server_random[0]),SSL3
                        EVP_VerifyUpdate(&md_ctx,param,param_len);
                        if (EVP_VerifyFinal(&md_ctx,p,(int)n,pkey) <= 0)
                                {
                                /* bad signature */
                                al=SSL_AD_DECRYPT_ERROR;
                                SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_BAD_SIG
                                goto f_err;
                                }
                        }
                }
        else
                {
                if (!(alg_a & SSL_aNULL) && !(alg_k & SSL_kPSK))
                        /* aNULL or kPSK do not need public keys */
                        {
                        SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,ERR_R_INTERNAL_ERROR)
                        goto err;
                        }
```

```
1844                 /* still data left over */
1845                 if (n != 0)
1846                         {
1847                         al=SSL_AD_DECODE_ERROR;
1848                         SSLerr(SSL_F_SSL3_GET_KEY_EXCHANGE,SSL_R_EXTRA_DATA_IN_M
1849                         goto f_err;
1850                         }
1851                 }
1852         EVP_PKEY_free(pkey);
1853         EVP_MD_CTX_cleanup(&md_ctx);
1854         return(1);
1855 f_err:
1856         ssl3_send_alert(s,SSL3_AL_FATAL,al);
1857 err:
1858         EVP_PKEY_free(pkey);
1859 #ifndef OPENSSL_NO_RSA
1860         if (rsa != NULL)
1861                 RSA_free(rsa);
1862 #endif
1863 #ifndef OPENSSL_NO_DH
1864         if (dh != NULL)
1865                 DH_free(dh);
1866 #endif
1867 #ifndef OPENSSL_NO_ECDH
1868         BN_CTX_free(bn_ctx);
1869         EC_POINT_free(srvr_ecpoint);
1870         if (ecdh != NULL)
1871                 EC_KEY_free(ecdh);
1872 #endif
1873         EVP_MD_CTX_cleanup(&md_ctx);
1874         return(-1);
1875         }

1877 int ssl3_get_certificate_request(SSL *s)
1878         {
1879         int ok,ret=0;
1880         unsigned long n,nc,l;
1881         unsigned int llen, ctype_num,i;
1882         X509_NAME *xn=NULL;
1883         const unsigned char *p,*q;
1884         unsigned char *d;
1885         STACK_OF(X509_NAME) *ca_sk=NULL;

1887         n=s->method->ssl_get_message(s,
1888                 SSL3_ST_CR_CERT_REQ_A,
1889                 SSL3_ST_CR_CERT_REQ_B,
1890                 -1,
1891                 s->max_cert_list,
1892                 &ok);

1894         if (!ok) return((int)n);

1896         s->s3->tmp.cert_req=0;

1898         if (s->s3->tmp.message_type == SSL3_MT_SERVER_DONE)
1899                 {
1900                 s->s3->tmp.reuse_message=1;
1901                 /* If we get here we don't need any cached handshake records
1902                  * as we wont be doing client auth.
1903                  */
1904                 if (s->s3->handshake_buffer)
1905                         {
1906                         if (!ssl3_digest_cached_records(s))
1907                                 goto err;
1908                         }
1909                 return(1);
```



```
1910                 }

1912         if (s->s3->tmp.message_type != SSL3_MT_CERTIFICATE_REQUEST)
1913                 {
1914                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_UNEXPECTED_MESSAGE);
1915                 SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,SSL_R_WRONG_MESSAGE_TY
1916                 goto err;
1917                 }

1919         /* TLS does not like anon-DH with client cert */
1920         if (s->version > SSL3_VERSION)
1921                 {
1922                 if (s->s3->tmp.new_cipher->algorithm_auth & SSL_aNULL)
1923                         {
1924                         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_UNEXPECTED_MESSAG
1925                         SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,SSL_R_TLS_CLIE
1926                         goto err;
1927                         }
1928                 }

1930         p=d=(unsigned char *)s->init_msg;

1932         if ((ca_sk=sk_X509_NAME_new(ca_dn_cmp)) == NULL)
1933                 {
1934                 SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,ERR_R_MALLOC_FAILURE);
1935                 goto err;
1936                 }

1938         /* get the certificate types */
1939         ctype_num= *(p++);
1940         if (ctype_num > SSL3_CT_NUMBER)
1941                 ctype_num=SSL3_CT_NUMBER;
1942         for (i=0; i<ctype_num; i++)
1943                 s->s3->tmp.ctype[i]= p[i];
1944         p+=ctype_num;
1945         if (TLS1_get_version(s) >= TLS1_2_VERSION)
1946                 {
1947                 n2s(p, llen);
1948                 /* Check we have enough room for signature algorithms and
1949                  * following length value.
1950                  */
1951                 if ((unsigned long)(p - d + llen + 2) > n)
1952                         {
1953                         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_DECODE_ERROR);
1954                         SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,SSL_R_DATA_LEN
1955                         goto err;
1956                         }
1957                 if ((llen & 1) || !tls1_process_sigalgs(s, p, llen))
1958                         {
1959                         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_DECODE_ERROR);
1960                         SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,SSL_R_SIGNATUR
1961                         goto err;
1962                         }
1963                 p += llen;
1964                 }

1966         /* get the CA RDNs */
1967         n2s(p,llen);
1968 #if 0
1969 {
1970 FILE *out;
1971 out=fopen("/tmp/vsign.der","w");
1972 fwrite(p,1,llen,out);
1973 fclose(out);
1974 }
1975 #endif
```

```
1977         if ((unsigned long)(p - d + llen) != n)
1978                 {
1979                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_DECODE_ERROR);
1980                 SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,SSL_R_LENGTH_MISMATCH)
1981                 goto err;
1982                 }
1983
1984         for (nc=0; nc<llen; )
1985                 {
1986                 n2s(p,l);
1987                 if ((l+nc+2) > llen)
1988                         {
1989                         if ((s->options & SSL_OP_NETSCAPE_CA_DN_BUG))
1990                                 goto cont; /* netscape bugs */
1991                         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_DECODE_ERROR);
1992                         SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,SSL_R_CA_DN_TO
1993                         goto err;
1994                         }
1995
1996                 q=p;
1997
1998                 if ((xn=d2i_X509_NAME(NULL,&q,l)) == NULL)
1999                         {
2000                         /* If netscape tolerance is on, ignore errors */
2001                         if (s->options & SSL_OP_NETSCAPE_CA_DN_BUG)
2002                                 goto cont;
2003                         else
2004                                 {
2005                                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_DECODE_ER
2006                                 SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,ERR_R_
2007                                 goto err;
2008                                 }
2009                         }
2010
2011                 if (q != (p+l))
2012                         {
2013                         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_DECODE_ERROR);
2014                         SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,SSL_R_CA_DN_LE
2015                         goto err;
2016                         }
2017                 if (!sk_X509_NAME_push(ca_sk,xn))
2018                         {
2019                         SSLerr(SSL_F_SSL3_GET_CERTIFICATE_REQUEST,ERR_R_MALLOC_F
2020                         goto err;
2021                         }
2022
2023                 p+=l;
2024                 nc+=l+2;
2025                 }
2026
2027         if (0)
2028                 {
2029 cont:
2030                 ERR_clear_error();
2031                 }
2032
2033         /* we should setup a certificate to return.... */
2034         s->s3->tmp.cert_req=1;
2035         s->s3->tmp.ctype_num=ctype_num;
2036         if (s->s3->tmp.ca_names != NULL)
2037                 sk_X509_NAME_pop_free(s->s3->tmp.ca_names,X509_NAME_free);
2038         s->s3->tmp.ca_names=ca_sk;
2039         ca_sk=NULL;
2040
2041         ret=1;
```



```
2042 err:
2043         if (ca_sk != NULL) sk_X509_NAME_pop_free(ca_sk,X509_NAME_free);
2044         return(ret);
2045         }
2046
2047 static int ca_dn_cmp(const X509_NAME * const *a, const X509_NAME * const *b)
2048         {
2049         return(X509_NAME_cmp(*a,*b));
2050         }
2051 #ifndef OPENSSL_NO_TLSEXT
2052 int ssl3_get_new_session_ticket(SSL *s)
2053         {
2054         int ok,al,ret=0, ticklen;
2055         long n;
2056         const unsigned char *p;
2057         unsigned char *d;
2058
2059         n=s->method->ssl_get_message(s,
2060                 SSL3_ST_CR_SESSION_TICKET_A,
2061                 SSL3_ST_CR_SESSION_TICKET_B,
2062                 -1,
2063                 16384,
2064                 &ok);
2065
2066         if (!ok)
2067                 return((int)n);
2068
2069         if (s->s3->tmp.message_type == SSL3_MT_FINISHED)
2070                 {
2071                 s->s3->tmp.reuse_message=1;
2072                 return(1);
2073                 }
2074         if (s->s3->tmp.message_type != SSL3_MT_NEWSESSION_TICKET)
2075                 {
2076                 al=SSL_AD_UNEXPECTED_MESSAGE;
2077                 SSLerr(SSL_F_SSL3_GET_NEW_SESSION_TICKET,SSL_R_BAD_MESSAGE_TYPE)
2078                 goto f_err;
2079                 }
2080         if (n < 6)
2081                 {
2082                 /* need at least ticket_lifetime_hint + ticket length */
2083                 al = SSL_AD_DECODE_ERROR;
2084                 SSLerr(SSL_F_SSL3_GET_NEW_SESSION_TICKET,SSL_R_LENGTH_MISMATCH);
2085                 goto f_err;
2086                 }
2087
2088         p=d=(unsigned char *)s->init_msg;
2089         n2l(p, s->session->tlsext_tick_lifetime_hint);
2090         n2s(p, ticklen);
2091         /* ticket_lifetime_hint + ticket_length + ticket */
2092         if (ticklen + 6 != n)
2093                 {
2094                 al = SSL_AD_DECODE_ERROR;
2095                 SSLerr(SSL_F_SSL3_GET_NEW_SESSION_TICKET,SSL_R_LENGTH_MISMATCH);
2096                 goto f_err;
2097                 }
2098         if (s->session->tlsext_tick)
2099                 {
2100                 OPENSSL_free(s->session->tlsext_tick);
2101                 s->session->tlsext_ticklen = 0;
2102                 }
2103         s->session->tlsext_tick = OPENSSL_malloc(ticklen);
2104         if (!s->session->tlsext_tick)
2105                 {
2106                 SSLerr(SSL_F_SSL3_GET_NEW_SESSION_TICKET,ERR_R_MALLOC_FAILURE);
2107                 goto err;
```

```
2108                 }
2109         memcpy(s->session->tlsext_tick, p, ticklen);
2110         s->session->tlsext_ticklen = ticklen;
2111         /* There are two ways to detect a resumed ticket sesion.
2112          * One is to set an appropriate session ID and then the server
2113          * must return a match in ServerHello. This allows the normal
2114          * client session ID matching to work and we know much
2115          * earlier that the ticket has been accepted.
2116          *
2117          * The other way is to set zero length session ID when the
2118          * ticket is presented and rely on the handshake to determine
2119          * session resumption.
2120          *
2121          * We choose the former approach because this fits in with
2122          * assumptions elsewhere in OpenSSL. The session ID is set
2123          * to the SHA256 (or SHA1 is SHA256 is disabled) hash of the
2124          * ticket.
2125          */
2126         EVP_Digest(p, ticklen,
2127                         s->session->session_id, &s->session->session_id_length,
2128 #ifndef OPENSSL_NO_SHA256
2129                                                 EVP_sha256(), NULL);
2130 #else
2131                                                 EVP_sha1(), NULL);
2132 #endif
2133         ret=1;
2134         return(ret);
2135 f_err:
2136         ssl3_send_alert(s,SSL3_AL_FATAL,al);
2137 err:
2138         return(-1);
2139         }

2141 int ssl3_get_cert_status(SSL *s)
2142         {
2143         int ok, al;
2144         unsigned long resplen,n;
2145         const unsigned char *p;

2147         n=s->method->ssl_get_message(s,
2148                 SSL3_ST_CR_CERT_STATUS_A,
2149                 SSL3_ST_CR_CERT_STATUS_B,
2150                 SSL3_MT_CERTIFICATE_STATUS,
2151                 16384,
2152                 &ok);

2154         if (!ok) return((int)n);
2155         if (n < 4)
2156                 {
2157                 /* need at least status type + length */
2158                 al = SSL_AD_DECODE_ERROR;
2159                 SSLerr(SSL_F_SSL3_GET_CERT_STATUS,SSL_R_LENGTH_MISMATCH);
2160                 goto f_err;
2161                 }
2162         p = (unsigned char *)s->init_msg;
2163         if (*p++ != TLSEXT_STATUSTYPE_ocsp)
2164                 {
2165                 al = SSL_AD_DECODE_ERROR;
2166                 SSLerr(SSL_F_SSL3_GET_CERT_STATUS,SSL_R_UNSUPPORTED_STATUS_TYPE)
2167                 goto f_err;
2168                 }
2169         n2l3(p, resplen);
2170         if (resplen + 4 != n)
2171                 {
2172                 al = SSL_AD_DECODE_ERROR;
2173                 SSLerr(SSL_F_SSL3_GET_CERT_STATUS,SSL_R_LENGTH_MISMATCH);
```



```
2174                 goto f_err;
2175                 }
2176         if (s->tlsext_ocsp_resp)
2177                 OPENSSL_free(s->tlsext_ocsp_resp);
2178         s->tlsext_ocsp_resp = BUF_memdup(p, resplen);
2179         if (!s->tlsext_ocsp_resp)
2180                 {
2181                 al = SSL_AD_INTERNAL_ERROR;
2182                 SSLerr(SSL_F_SSL3_GET_CERT_STATUS,ERR_R_MALLOC_FAILURE);
2183                 goto f_err;
2184                 }
2185         s->tlsext_ocsp_resplen = resplen;
2186         if (s->ctx->tlsext_status_cb)
2187                 {
2188                 int ret;
2189                 ret = s->ctx->tlsext_status_cb(s, s->ctx->tlsext_status_arg);
2190                 if (ret == 0)
2191                         {
2192                         al = SSL_AD_BAD_CERTIFICATE_STATUS_RESPONSE;
2193                         SSLerr(SSL_F_SSL3_GET_CERT_STATUS,SSL_R_INVALID_STATUS_R
2194                         goto f_err;
2195                         }
2196                 if (ret < 0)
2197                         {
2198                         al = SSL_AD_INTERNAL_ERROR;
2199                         SSLerr(SSL_F_SSL3_GET_CERT_STATUS,ERR_R_MALLOC_FAILURE);
2200                         goto f_err;
2201                         }
2202                 }
2203         return 1;
2204 f_err:
2205         ssl3_send_alert(s,SSL3_AL_FATAL,al);
2206         return(-1);
2207         }
2208 #endif

2210 int ssl3_get_server_done(SSL *s)
2211         {
2212         int ok,ret=0;
2213         long n;

2215         n=s->method->ssl_get_message(s,
2216                 SSL3_ST_CR_SRVR_DONE_A,
2217                 SSL3_ST_CR_SRVR_DONE_B,
2218                 SSL3_MT_SERVER_DONE,
2219                 30, /* should be very small, like 0 :-) */
2220                 &ok);

2222         if (!ok) return((int)n);
2223         if (n > 0)
2224                 {
2225                 /* should contain no data */
2226                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_DECODE_ERROR);
2227                 SSLerr(SSL_F_SSL3_GET_SERVER_DONE,SSL_R_LENGTH_MISMATCH);
2228                 return -1;
2229                 }
2230         ret=1;
2231         return(ret);
2232         }


2235 int ssl3_send_client_key_exchange(SSL *s)
2236         {
2237         unsigned char *p,*d;
2238         int n;
2239         unsigned long alg_k;
```

```
#ifndef OPENSSL_NO_RSA
        unsigned char *q;
        EVP_PKEY *pkey=NULL;
#endif
#ifndef OPENSSL_NO_KRB5
        KSSL_ERR kssl_err;
#endif /* OPENSSL_NO_KRB5 */
#ifndef OPENSSL_NO_ECDH
        EC_KEY *clnt_ecdh = NULL;
        const EC_POINT *srvr_ecpoint = NULL;
        EVP_PKEY *srvr_pub_pkey = NULL;
        unsigned char *encodedPoint = NULL;
        int encoded_pt_len = 0;
        BN_CTX * bn_ctx = NULL;
#endif

        if (s->state == SSL3_ST_CW_KEY_EXCH_A)
                {
                d=(unsigned char *)s->init_buf->data;
                p= &(d[4]);

                alg_k=s->s3->tmp.new_cipher->algorithm_mkey;

                /* Fool emacs indentation */
                if (0) {}
#ifndef OPENSSL_NO_RSA
                else if (alg_k & SSL_kRSA)
                        {
                        RSA *rsa;
                        unsigned char tmp_buf[SSL_MAX_MASTER_KEY_LENGTH];

                        if (s->session->sess_cert == NULL)
                                {
                                /* We should always have a server certificate wi
                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
                                goto err;
                                }


                        if (s->session->sess_cert->peer_rsa_tmp != NULL)
                                rsa=s->session->sess_cert->peer_rsa_tmp;
                        else
                                {
                                pkey=X509_get_pubkey(s->session->sess_cert->peer
                                if ((pkey == NULL) ||
                                        (pkey->type != EVP_PKEY_RSA) ||
                                        (pkey->pkey.rsa == NULL))
                                        {
                                        SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
                                        goto err;
                                        }
                                rsa=pkey->pkey.rsa;
                                EVP_PKEY_free(pkey);
                                }

                        tmp_buf[0]=s->client_version>>8;
                        tmp_buf[1]=s->client_version&0xff;
                        if (RAND_bytes(&(tmp_buf[2]),sizeof tmp_buf-2) <= 0)
                                        goto err;

                        s->session->master_key_length=sizeof tmp_buf;

                        q=p;
                        /* Fix buf for TLS and beyond */
                        if (s->version > SSL3_VERSION)
                                p+=2;
```



```
                        n=RSA_public_encrypt(sizeof tmp_buf,
                                tmp_buf,p,rsa,RSA_PKCS1_PADDING);
#ifdef PKCS1_CHECK
                        if (s->options & SSL_OP_PKCS1_CHECK_1) p[1]++;
                        if (s->options & SSL_OP_PKCS1_CHECK_2) tmp_buf[0]=0x70;
#endif
                        if (n <= 0)
                                {
                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,SSL_R
                                goto err;
                                }

                        /* Fix buf for TLS and beyond */
                        if (s->version > SSL3_VERSION)
                                {
                                s2n(n,q);
                                n+=2;
                                }

                        s->session->master_key_length=
                                s->method->ssl3_enc->generate_master_secret(s,
                                        s->session->master_key,
                                        tmp_buf,sizeof tmp_buf);
                        OPENSSL_cleanse(tmp_buf,sizeof tmp_buf);
                        }
#endif
#ifndef OPENSSL_NO_KRB5
                else if (alg_k & SSL_kKRB5)
                        {
                        krb5_error_code krb5rc;
                        KSSL_CTX        *kssl_ctx = s->kssl_ctx;
                        /*  krb5_data   krb5_ap_req;  */
                        krb5_data       *enc_ticket;
                        krb5_data       authenticator, *authp = NULL;
                        EVP_CIPHER_CTX  ciph_ctx;
                        const EVP_CIPHER *enc = NULL;
                        unsigned char   iv[EVP_MAX_IV_LENGTH];
                        unsigned char   tmp_buf[SSL_MAX_MASTER_KEY_LENGTH];
                        unsigned char   epms[SSL_MAX_MASTER_KEY_LENGTH
                                                + EVP_MAX_IV_LENGTH];
                        int             padl, outl = sizeof(epms);

                        EVP_CIPHER_CTX_init(&ciph_ctx);

#ifdef KSSL_DEBUG
                        printf("ssl3_send_client_key_exchange(%lx & %lx)\n",
                                alg_k, SSL_kKRB5);
#endif  /* KSSL_DEBUG */

                        authp = NULL;
#ifdef KRB5SENDAUTH
                        if (KRB5SENDAUTH)  authp = &authenticator;
#endif  /* KRB5SENDAUTH */

                        krb5rc = kssl_cget_tkt(kssl_ctx, &enc_ticket, authp,
                                &kssl_err);
                        enc = kssl_map_enc(kssl_ctx->enctype);
                        if (enc == NULL)
                            goto err;
#ifdef KSSL_DEBUG
                        {
                        printf("kssl_cget_tkt rtn %d\n", krb5rc);
                        if (krb5rc && kssl_err.text)
                          printf("kssl_cget_tkt kssl_err=%s\n", kssl_err.text);
                        }
#endif  /* KSSL_DEBUG */
```

```
2373                        if (krb5rc)
2374                                {
2375                                ssl3_send_alert(s,SSL3_AL_FATAL,
2376                                                SSL_AD_HANDSHAKE_FAILURE);
2377                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2378                                                kssl_err.reason);
2379                                goto err;
2380                                }

2382                        /*  20010406 VRS - Earlier versions used KRB5 AP_REQ
2383                        **  in place of RFC 2712 KerberosWrapper, as in:
2384                        **
2385                        **  Send ticket (copy to *p, set n = length)
2386                        **  n = krb5_ap_req.length;
2387                        **  memcpy(p, krb5_ap_req.data, krb5_ap_req.length);
2388                        **  if (krb5_ap_req.data)
2389                        **    kssl_krb5_free_data_contents(NULL,&krb5_ap_req);
2390                        **
2391                        **  Now using real RFC 2712 KerberosWrapper
2392                        **  (Thanks to Simon Wilkinson <sxw@sxw.org.uk>)
2393                        **  Note: 2712 "opaque" types are here replaced
2394                        **  with a 2-byte length followed by the value.
2395                        **  Example:
2396                        **  KerberosWrapper= xx xx asn1ticket 0 0 xx xx encpms
2397                        **  Where "xx xx" = length bytes.  Shown here with
2398                        **  optional authenticator omitted.
2399                        */

2401                        /*  KerberosWrapper.Ticket              */
2402                        s2n(enc_ticket->length,p);
2403                        memcpy(p, enc_ticket->data, enc_ticket->length);
2404                        p+= enc_ticket->length;
2405                        n = enc_ticket->length + 2;

2407                        /*  KerberosWrapper.Authenticator       */
2408                        if (authp  &&  authp->length)
2409                                {
2410                                s2n(authp->length,p);
2411                                memcpy(p, authp->data, authp->length);
2412                                p+= authp->length;
2413                                n+= authp->length + 2;

2415                                free(authp->data);
2416                                authp->data = NULL;
2417                                authp->length = 0;
2418                                }
2419                        else
2420                                {
2421                                s2n(0,p);/*  null authenticator length  */
2422                                n+=2;
2423                                }

2425                            tmp_buf[0]=s->client_version>>8;
2426                            tmp_buf[1]=s->client_version&0xff;
2427                            if (RAND_bytes(&(tmp_buf[2]),sizeof tmp_buf-2) <= 0)
2428                                goto err;

2430                        /*  20010420 VRS.  Tried it this way; failed.
2431                        **      EVP_EncryptInit_ex(&ciph_ctx,enc, NULL,NULL);
2432                        **      EVP_CIPHER_CTX_set_key_length(&ciph_ctx,
2433                        **                              kssl_ctx->length);
2434                        **      EVP_EncryptInit_ex(&ciph_ctx,NULL, key,iv);
2435                        */

2437                        memset(iv, 0, sizeof iv);  /* per RFC 1510 */
```



```
2438                        EVP_EncryptInit_ex(&ciph_ctx,enc, NULL,
2439                                kssl_ctx->key,iv);
2440                        EVP_EncryptUpdate(&ciph_ctx,epms,&outl,tmp_buf,
2441                                sizeof tmp_buf);
2442                        EVP_EncryptFinal_ex(&ciph_ctx,&(epms[outl]),&padl);
2443                        outl += padl;
2444                        if (outl > (int)sizeof epms)
2445                                {
2446                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE, ERR_
2447                                goto err;
2448                                }
2449                        EVP_CIPHER_CTX_cleanup(&ciph_ctx);

2451                        /*  KerberosWrapper.EncryptedPreMasterSecret    */
2452                        s2n(outl,p);
2453                        memcpy(p, epms, outl);
2454                        p+=outl;
2455                        n+=outl + 2;

2457                        s->session->master_key_length=
2458                                s->method->ssl3_enc->generate_master_secret(s,
2459                                        s->session->master_key,
2460                                        tmp_buf, sizeof tmp_buf);

2462                        OPENSSL_cleanse(tmp_buf, sizeof tmp_buf);
2463                        OPENSSL_cleanse(epms, outl);
2464                        }
2465 #endif
2466 #ifndef OPENSSL_NO_DH
2467                     else if (alg_k & (SSL_kEDH|SSL_kDHr|SSL_kDHd))
2468                        {
2469                        DH *dh_srvr,*dh_clnt;

2471                        if (s->session->sess_cert == NULL)
2472                                {
2473                                ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_UNEXPECTE
2474                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,SSL_R
2475                                goto err;
2476                                }

2478                        if (s->session->sess_cert->peer_dh_tmp != NULL)
2479                                dh_srvr=s->session->sess_cert->peer_dh_tmp;
2480                        else
2481                                {
2482                                /* we get them from the cert */
2483                                ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_HANDSHAKE
2484                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,SSL_R
2485                                goto err;
2486                                }

2488                        /* generate a new random key */
2489                        if ((dh_clnt=DHparams_dup(dh_srvr)) == NULL)
2490                                {
2491                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
2492                                goto err;
2493                                }
2494                        if (!DH_generate_key(dh_clnt))
2495                                {
2496                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
2497                                DH_free(dh_clnt);
2498                                goto err;
2499                                }

2501                        /* use the 'p' output buffer for the DH key, but
2502                         * make sure to clear it out afterwards */
```

```
2504                        n=DH_compute_key(p,dh_srvr->pub_key,dh_clnt);

2506                        if (n <= 0)
2507                                {
2508                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
2509                                DH_free(dh_clnt);
2510                                goto err;
2511                                }

2513                        /* generate master key from the result */
2514                        s->session->master_key_length=
2515                                s->method->ssl3_enc->generate_master_secret(s,
2516                                        s->session->master_key,p,n);
2517                        /* clean up */
2518                        memset(p,0,n);

2520                        /* send off the data */
2521                        n=BN_num_bytes(dh_clnt->pub_key);
2522                        s2n(n,p);
2523                        BN_bn2bin(dh_clnt->pub_key,p);
2524                        n+=2;

2526                        DH_free(dh_clnt);

2528                        /* perhaps clean things up a bit EAY EAY EAY EAY*/
2529                        }
2530 #endif

2532 #ifndef OPENSSL_NO_ECDH
2533                 else if (alg_k & (SSL_kEECDH|SSL_kECDHr|SSL_kECDHe))
2534                         {
2535                         const EC_GROUP *srvr_group = NULL;
2536                         EC_KEY *tkey;
2537                         int ecdh_clnt_cert = 0;
2538                         int field_size = 0;

2540                         if (s->session->sess_cert == NULL)
2541                                 {
2542                                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_UNEXPECTE
2543                                 SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,SSL_R
2544                                 goto err;
2545                                 }

2547                         /* Did we send out the client's
2548                          * ECDH share for use in premaster
2549                          * computation as part of client certificate?
2550                          * If so, set ecdh_clnt_cert to 1.
2551                          */
2552                         if ((alg_k & (SSL_kECDHr|SSL_kECDHe)) && (s->cert != NUL
2553                                 {
2554                                 /* XXX: For now, we do not support client
2555                                  * authentication using ECDH certificates.
2556                                  * To add such support, one needs to add
2557                                  * code that checks for appropriate
2558                                  * conditions and sets ecdh_clnt_cert to 1.
2559                                  * For example, the cert have an ECC
2560                                  * key on the same curve as the server's
2561                                  * and the key should be authorized for
2562                                  * key agreement.
2563                                  *
2564                                  * One also needs to add code in ssl3_connect
2565                                  * to skip sending the certificate verify
2566                                  * message.
2567                                  *
2568                                  * if ((s->cert->key->privatekey != NULL) &&
2569                                  *     (s->cert->key->privatekey->type ==
```



```
2570                                  *      EVP_PKEY_EC) && ...)
2571                                  * ecdh_clnt_cert = 1;
2572                                  */
2573                                 }

2575                         if (s->session->sess_cert->peer_ecdh_tmp != NULL)
2576                                 {
2577                                 tkey = s->session->sess_cert->peer_ecdh_tmp;
2578                                 }
2579                         else
2580                                 {
2581                                 /* Get the Server Public Key from Cert */
2582                                 srvr_pub_pkey = X509_get_pubkey(s->session-> \
2583                                     sess_cert->peer_pkeys[SSL_PKEY_ECC].x509);
2584                                 if ((srvr_pub_pkey == NULL) ||
2585                                     (srvr_pub_pkey->type != EVP_PKEY_EC) ||
2586                                     (srvr_pub_pkey->pkey.ec == NULL))
2587                                         {
2588                                         SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
2589                                             ERR_R_INTERNAL_ERROR);
2590                                         goto err;
2591                                         }

2593                                 tkey = srvr_pub_pkey->pkey.ec;
2594                                 }

2596                         srvr_group   = EC_KEY_get0_group(tkey);
2597                         srvr_ecpoint = EC_KEY_get0_public_key(tkey);

2599                         if ((srvr_group == NULL) || (srvr_ecpoint == NULL))
2600                                 {
2601                                 SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2602                                     ERR_R_INTERNAL_ERROR);
2603                                 goto err;
2604                                 }

2606                         if ((clnt_ecdh=EC_KEY_new()) == NULL)
2607                                 {
2608                                 SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
2609                                 goto err;
2610                                 }

2612                         if (!EC_KEY_set_group(clnt_ecdh, srvr_group))
2613                                 {
2614                                 SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
2615                                 goto err;
2616                                 }
2617                         if (ecdh_clnt_cert)
2618                                 {
2619                                 /* Reuse key info from our certificate
2620                                  * We only need our private key to perform
2621                                  * the ECDH computation.
2622                                  */
2623                                 const BIGNUM *priv_key;
2624                                 tkey = s->cert->key->privatekey->pkey.ec;
2625                                 priv_key = EC_KEY_get0_private_key(tkey);
2626                                 if (priv_key == NULL)
2627                                         {
2628                                         SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
2629                                         goto err;
2630                                         }
2631                                 if (!EC_KEY_set_private_key(clnt_ecdh, priv_key)
2632                                         {
2633                                         SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
2634                                         goto err;
2635                                         }
```

```
2636                            }
2637                        else
2638                            {
2639                            /* Generate a new ECDH key pair */
2640                            if (!(EC_KEY_generate_key(clnt_ecdh)))
2641                                    {
2642                                    SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
2643                                    goto err;
2644                                    }
2645                            }

2647                        /* use the 'p' output buffer for the ECDH key, but
2648                         * make sure to clear it out afterwards
2649                         */

2651                        field_size = EC_GROUP_get_degree(srvr_group);
2652                        if (field_size <= 0)
2653                            {
2654                            SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2655                                   ERR_R_ECDH_LIB);
2656                            goto err;
2657                            }
2658                        n=ECDH_compute_key(p, (field_size+7)/8, srvr_ecpoint, cl
2659                        if (n <= 0)
2660                            {
2661                            SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2662                                   ERR_R_ECDH_LIB);
2663                            goto err;
2664                            }

2666                        /* generate master key from the result */
2667                        s->session->master_key_length = s->method->ssl3_enc \
2668                            -> generate_master_secret(s,
2669                                s->session->master_key,
2670                                p, n);

2672                        memset(p, 0, n); /* clean up */

2674                        if (ecdh_clnt_cert)
2675                            {
2676                            /* Send empty client key exch message */
2677                            n = 0;
2678                            }
2679                        else
2680                            {
2681                            /* First check the size of encoding and
2682                             * allocate memory accordingly.
2683                             */
2684                            encoded_pt_len =
2685                                EC_POINT_point2oct(srvr_group,
2686                                    EC_KEY_get0_public_key(clnt_ecdh),
2687                                    POINT_CONVERSION_UNCOMPRESSED,
2688                                    NULL, 0, NULL);

2690                            encodedPoint = (unsigned char *)
2691                                OPENSSL_malloc(encoded_pt_len *
2692                                    sizeof(unsigned char));
2693                            bn_ctx = BN_CTX_new();
2694                            if ((encodedPoint == NULL) ||
2695                                (bn_ctx == NULL))
2696                                    {
2697                                    SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
2698                                    goto err;
2699                                    }

2701                            /* Encode the public key */
```



```
2702                                n = EC_POINT_point2oct(srvr_group,
2703                                    EC_KEY_get0_public_key(clnt_ecdh),
2704                                    POINT_CONVERSION_UNCOMPRESSED,
2705                                    encodedPoint, encoded_pt_len, bn_ctx);

2707                                *p = n; /* length of encoded point */
2708                                /* Encoded point will be copied here */
2709                                p += 1;
2710                                /* copy the point */
2711                                memcpy((unsigned char *)p, encodedPoint, n);
2712                                /* increment n to account for length field */
2713                                n += 1;
2714                                }

2716                            /* Free allocated memory */
2717                            BN_CTX_free(bn_ctx);
2718                            if (encodedPoint != NULL) OPENSSL_free(encodedPoint);
2719                            if (clnt_ecdh != NULL)
2720                                     EC_KEY_free(clnt_ecdh);
2721                            EVP_PKEY_free(srvr_pub_pkey);
2722                            }
2723 #endif /* !OPENSSL_NO_ECDH */
2724                    else if (alg_k & SSL_kGOST)
2725                            {
2726                            /* GOST key exchange message creation */
2727                            EVP_PKEY_CTX *pkey_ctx;
2728                            X509 *peer_cert;
2729                            size_t msglen;
2730                            unsigned int md_len;
2731                            int keytype;
2732                            unsigned char premaster_secret[32],shared_ukm[32], tmp[2
2733                            EVP_MD_CTX *ukm_hash;
2734                            EVP_PKEY *pub_key;

2736                            /* Get server sertificate PKEY and create ctx from it */
2737                            peer_cert=s->session->sess_cert->peer_pkeys[(keytype=SSL
2738                            if (!peer_cert)
2739                                    peer_cert=s->session->sess_cert->peer_pkeys[(key
2740                            if (!peer_cert)         {
2741                                            SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
2742                                            goto err;
2743                                    }

2745                            pkey_ctx=EVP_PKEY_CTX_new(pub_key=X509_get_pubkey(peer_c
2746                            /* If we have send a certificate, and certificate key

2748                             * parameters match those of server certificate, use
2749                             * certificate key for key exchange
2750                             */

2752                             /* Otherwise, generate ephemeral key pair */

2754                            EVP_PKEY_encrypt_init(pkey_ctx);
2755                              /* Generate session key */
2756                        RAND_bytes(premaster_secret,32);
2757                            /* If we have client certificate, use its secret as peer
2758                            if (s->s3->tmp.cert_req && s->cert->key->privatekey) {
2759                                    if (EVP_PKEY_derive_set_peer(pkey_ctx,s->cert->k
2760                                            /* If there was an error - just ignore i
2761                                            * would be used
2762                                            */
2763                                            ERR_clear_error();
2764                                    }
2765                            }
2766                            /* Compute shared IV and store it in algorithm-specific
2767                             * context data */
```

```
2768                        ukm_hash = EVP_MD_CTX_create();
2769                        EVP_DigestInit(ukm_hash,EVP_get_digestbynid(NID_id_GostR
2770                        EVP_DigestUpdate(ukm_hash,s->s3->client_random,SSL3_RAND
2771                        EVP_DigestUpdate(ukm_hash,s->s3->server_random,SSL3_RAND
2772                        EVP_DigestFinal_ex(ukm_hash, shared_ukm, &md_len);
2773                        EVP_MD_CTX_destroy(ukm_hash);
2774                        if (EVP_PKEY_CTX_ctrl(pkey_ctx,-1,EVP_PKEY_OP_ENCRYPT,EV
2775                                8,shared_ukm)<0) {
2776                                        SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHAN
2777                                                SSL_R_LIBRARY_BUG);
2778                                        goto err;
2779                                }
2780                        /* Make GOST keytransport blob message */
2781                        /*Encapsulate it into sequence */
2782                        *(p++)=V_ASN1_SEQUENCE | V_ASN1_CONSTRUCTED;
2783                        msglen=255;
2784                        if (EVP_PKEY_encrypt(pkey_ctx,tmp,&msglen,premaster_secr
2785                        SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2786                                        SSL_R_LIBRARY_BUG);
2787                                goto err;
2788                        }
2789                        if (msglen >= 0x80)
2790                                {
2791                                *(p++)=0x81;
2792                                *(p++)= msglen & 0xff;
2793                                n=msglen+3;
2794                                }
2795                        else
2796                                {
2797                                *(p++)= msglen & 0xff;
2798                                n=msglen+2;
2799                                }
2800                        memcpy(p, tmp, msglen);
2801                        /* Check if pubkey from client certificate was used */
2802                        if (EVP_PKEY_CTX_ctrl(pkey_ctx, -1, -1, EVP_PKEY_CTRL_PE
2803                                {
2804                                /* Set flag "skip certificate verify" */
2805                                s->s3->flags |= TLS1_FLAGS_SKIP_CERT_VERIFY;
2806                                }
2807                        EVP_PKEY_CTX_free(pkey_ctx);
2808                        s->session->master_key_length=
2809                                s->method->ssl3_enc->generate_master_secret(s,
2810                                        s->session->master_key,premaster_secret,
2811                        EVP_PKEY_free(pub_key);

2813                        }
2814 #ifndef OPENSSL_NO_SRP
2815                 else if (alg_k & SSL_kSRP)
2816                        {
2817                        if (s->srp_ctx.A != NULL)
2818                                {
2819                                /* send off the data */
2820                                n=BN_num_bytes(s->srp_ctx.A);
2821                                s2n(n,p);
2822                                BN_bn2bin(s->srp_ctx.A,p);
2823                                n+=2;
2824                                }
2825                        else
2826                                {
2827                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
2828                                goto err;
2829                                }
2830                        if (s->session->srp_username != NULL)
2831                                OPENSSL_free(s->session->srp_username);
2832                        s->session->srp_username = BUF_strdup(s->srp_ctx.login);
2833                        if (s->session->srp_username == NULL)
```



```
2834                                {
2835                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2836                                        ERR_R_MALLOC_FAILURE);
2837                                goto err;
2838                                }

2840                        if ((s->session->master_key_length = SRP_generate_client
2841                                {
2842                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,ERR_R
2843                                goto err;
2844                                }
2845                        }
2846 #endif
2847 #ifndef OPENSSL_NO_PSK
2848                 else if (alg_k & SSL_kPSK)
2849                        {
2850                        char identity[PSK_MAX_IDENTITY_LEN];
2851                        unsigned char *t = NULL;
2852                        unsigned char psk_or_pre_ms[PSK_MAX_PSK_LEN*2+4];
2853                        unsigned int pre_ms_len = 0, psk_len = 0;
2854                        int psk_err = 1;

2856                        n = 0;
2857                        if (s->psk_client_callback == NULL)
2858                                {
2859                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2860                                        SSL_R_PSK_NO_CLIENT_CB);
2861                                goto err;
2862                                }

2864                        psk_len = s->psk_client_callback(s, s->ctx->psk_identity
2865                                identity, PSK_MAX_IDENTITY_LEN,
2866                                psk_or_pre_ms, sizeof(psk_or_pre_ms));
2867                        if (psk_len > PSK_MAX_PSK_LEN)
2868                                {
2869                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2870                                        ERR_R_INTERNAL_ERROR);
2871                                goto psk_err;
2872                                }
2873                        else if (psk_len == 0)
2874                                {
2875                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2876                                        SSL_R_PSK_IDENTITY_NOT_FOUND);
2877                                goto psk_err;
2878                                }

2880                        /* create PSK pre_master_secret */
2881                        pre_ms_len = 2+psk_len+2+psk_len;
2882                        t = psk_or_pre_ms;
2883                        memmove(psk_or_pre_ms+psk_len+4, psk_or_pre_ms, psk_len)
2884                        s2n(psk_len, t);
2885                        memset(t, 0, psk_len);
2886                        t+=psk_len;
2887                        s2n(psk_len, t);

2889                        if (s->session->psk_identity_hint != NULL)
2890                                OPENSSL_free(s->session->psk_identity_hint);
2891                        s->session->psk_identity_hint = BUF_strdup(s->ctx->psk_i
2892                        if (s->ctx->psk_identity_hint != NULL &&
2893                                s->session->psk_identity_hint == NULL)
2894                                {
2895                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2896                                        ERR_R_MALLOC_FAILURE);
2897                                goto psk_err;
2898                                }
```

```
2900                        if (s->session->psk_identity != NULL)
2901                                OPENSSL_free(s->session->psk_identity);
2902                        s->session->psk_identity = BUF_strdup(identity);
2903                        if (s->session->psk_identity == NULL)
2904                                {
2905                                SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2906                                        ERR_R_MALLOC_FAILURE);
2907                                goto psk_err;
2908                                }

2910                        s->session->master_key_length =
2911                                s->method->ssl3_enc->generate_master_secret(s,
2912                                        s->session->master_key,
2913                                        psk_or_pre_ms, pre_ms_len);
2914                        n = strlen(identity);
2915                        s2n(n, p);
2916                        memcpy(p, identity, n);
2917                        n+=2;
2918                        psk_err = 0;
2919                psk_err:
2920                        OPENSSL_cleanse(identity, PSK_MAX_IDENTITY_LEN);
2921                        OPENSSL_cleanse(psk_or_pre_ms, sizeof(psk_or_pre_ms));
2922                        if (psk_err != 0)
2923                                {
2924                                ssl3_send_alert(s, SSL3_AL_FATAL, SSL_AD_HANDSHA
2925                                goto err;
2926                                }
2927                        }
2928 #endif
2929                else
2930                        {
2931                        ssl3_send_alert(s, SSL3_AL_FATAL,
2932                            SSL_AD_HANDSHAKE_FAILURE);
2933                        SSLerr(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE,
2934                            ERR_R_INTERNAL_ERROR);
2935                        goto err;
2936                        }

2938                *(d++)=SSL3_MT_CLIENT_KEY_EXCHANGE;
2939                l2n3(n,d);

2941                s->state=SSL3_ST_CW_KEY_EXCH_B;
2942                /* number of bytes to write */
2943                s->init_num=n+4;
2944                s->init_off=0;
2945                }

2947        /* SSL3_ST_CW_KEY_EXCH_B */
2948        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
2949 err:
2950 #ifndef OPENSSL_NO_ECDH
2951        BN_CTX_free(bn_ctx);
2952        if (encodedPoint != NULL) OPENSSL_free(encodedPoint);
2953        if (clnt_ecdh != NULL)
2954                EC_KEY_free(clnt_ecdh);
2955        EVP_PKEY_free(srvr_pub_pkey);
2956 #endif
2957        return(-1);
2958        }

2960 int ssl3_send_client_verify(SSL *s)
2961        {
2962        unsigned char *p,*d;
2963        unsigned char data[MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH];
2964        EVP_PKEY *pkey;
2965        EVP_PKEY_CTX *pctx=NULL;
```



```
2966        EVP_MD_CTX mctx;
2967        unsigned u=0;
2968        unsigned long n;
2969        int j;

2971        EVP_MD_CTX_init(&mctx);

2973        if (s->state == SSL3_ST_CW_CERT_VRFY_A)
2974                {
2975                d=(unsigned char *)s->init_buf->data;
2976                p= &(d[4]);
2977                pkey=s->cert->key->privatekey;
2978 /* Create context from key and test if sha1 is allowed as digest */
2979                pctx = EVP_PKEY_CTX_new(pkey,NULL);
2980                EVP_PKEY_sign_init(pctx);
2981                if (EVP_PKEY_CTX_set_signature_md(pctx, EVP_sha1())>0)
2982                        {
2983                        if (TLS1_get_version(s) < TLS1_2_VERSION)
2984                                s->method->ssl3_enc->cert_verify_mac(s,
2985                                                NID_sha1,
2986                                                &(data[MD5_DIGEST_LENGTH]));
2987                        }
2988                else
2989                        {
2990                        ERR_clear_error();
2991                        }
2992                /* For TLS v1.2 send signature algorithm and signature
2993                 * using agreed digest and cached handshake records.
2994                 */
2995                if (TLS1_get_version(s) >= TLS1_2_VERSION)
2996                        {
2997                        long hdatalen = 0;
2998                        void *hdata;
2999                        const EVP_MD *md = s->cert->key->digest;
3000                        hdatalen = BIO_get_mem_data(s->s3->handshake_buffer,
3001                                                        &hdata);
3002                        if (hdatalen <= 0 || !tls12_get_sigandhash(p, pkey, md))
3003                                {
3004                                SSLerr(SSL_F_SSL3_SEND_CLIENT_VERIFY,
3005                                                ERR_R_INTERNAL_ERROR);
3006                                goto err;
3007                                }
3008                        p += 2;
3009 #ifdef SSL_DEBUG
3010                        fprintf(stderr, "Using TLS 1.2 with client alg %s\n",
3011                                                EVP_MD_name(md));
3012 #endif
3013                        if (!EVP_SignInit_ex(&mctx, md, NULL)
3014                                || !EVP_SignUpdate(&mctx, hdata, hdatalen)
3015                                || !EVP_SignFinal(&mctx, p + 2, &u, pkey))
3016                                {
3017                                SSLerr(SSL_F_SSL3_SEND_CLIENT_VERIFY,
3018                                                ERR_R_EVP_LIB);
3019                                goto err;
3020                                }
3021                        s2n(u,p);
3022                        n = u + 4;
3023                        if (!ssl3_digest_cached_records(s))
3024                                goto err;
3025                        }
3026                else
3027 #ifndef OPENSSL_NO_RSA
3028                if (pkey->type == EVP_PKEY_RSA)
3029                        {
3030                        s->method->ssl3_enc->cert_verify_mac(s,
3031                                NID_md5,
```

```
3032                                &(data[0]));
3033                        if (RSA_sign(NID_md5_sha1, data,
3034                                        MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH,
3035                                       &(p[2]), &u, pkey->pkey.rsa) <= 0 )
3036                                {
3037                                SSLerr(SSL_F_SSL3_SEND_CLIENT_VERIFY,ERR_R_RSA_L
3038                                goto err;
3039                                }
3040                        s2n(u,p);
3041                        n=u+2;
3042                        }
3043                else
3044 #endif
3045 #ifndef OPENSSL_NO_DSA
3046                        if (pkey->type == EVP_PKEY_DSA)
3047                        {
3048                        if (!DSA_sign(pkey->save_type,
3049                                &(data[MD5_DIGEST_LENGTH]),
3050                                SHA_DIGEST_LENGTH,&(p[2]),
3051                                (unsigned int *)&j,pkey->pkey.dsa))
3052                                {
3053                                SSLerr(SSL_F_SSL3_SEND_CLIENT_VERIFY,ERR_R_DSA_L
3054                                goto err;
3055                                }
3056                        s2n(j,p);
3057                        n=j+2;
3058                        }
3059                else
3060 #endif
3061 #ifndef OPENSSL_NO_ECDSA
3062                        if (pkey->type == EVP_PKEY_EC)
3063                        {
3064                        if (!ECDSA_sign(pkey->save_type,
3065                                &(data[MD5_DIGEST_LENGTH]),
3066                                SHA_DIGEST_LENGTH,&(p[2]),
3067                                (unsigned int *)&j,pkey->pkey.ec))
3068                                {
3069                                SSLerr(SSL_F_SSL3_SEND_CLIENT_VERIFY,
3070                                    ERR_R_ECDSA_LIB);
3071                                goto err;
3072                                }
3073                        s2n(j,p);
3074                        n=j+2;
3075                        }
3076                else
3077 #endif
3078                if (pkey->type == NID_id_GostR3410_94 || pkey->type == NID_id_Go
3079                {
3080                unsigned char signbuf[64];
3081                int i;
3082                size_t sigsize=64;
3083                s->method->ssl3_enc->cert_verify_mac(s,
3084                        NID_id_GostR3411_94,
3085                        data);
3086                if (EVP_PKEY_sign(pctx, signbuf, &sigsize, data, 32) <= 0) {
3087                        SSLerr(SSL_F_SSL3_SEND_CLIENT_VERIFY,
3088                        ERR_R_INTERNAL_ERROR);
3089                        goto err;
3090                }
3091                for (i=63,j=0; i>=0; j++, i--) {
3092                        p[2+j]=signbuf[i];
3093                }
3094                s2n(j,p);
3095                n=j+2;
3096                }
3097                else
```



```
3098                {
3099                        SSLerr(SSL_F_SSL3_SEND_CLIENT_VERIFY,ERR_R_INTERNAL_ERRO
3100                        goto err;
3101                }
3102                *(d++)=SSL3_MT_CERTIFICATE_VERIFY;
3103                l2n3(n,d);
3104
3105                s->state=SSL3_ST_CW_CERT_VRFY_B;
3106                s->init_num=(int)n+4;
3107                s->init_off=0;
3108                }
3109        EVP_MD_CTX_cleanup(&mctx);
3110        EVP_PKEY_CTX_free(pctx);
3111        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
3112 err:
3113        EVP_MD_CTX_cleanup(&mctx);
3114        EVP_PKEY_CTX_free(pctx);
3115        return(-1);
3116        }
3117
3118 int ssl3_send_client_certificate(SSL *s)
3119        {
3120        X509 *x509=NULL;
3121        EVP_PKEY *pkey=NULL;
3122        int i;
3123        unsigned long l;
3124
3125        if (s->state == SSL3_ST_CW_CERT_A)
3126                {
3127                if ((s->cert == NULL) ||
3128                        (s->cert->key->x509 == NULL) ||
3129                        (s->cert->key->privatekey == NULL))
3130                        s->state=SSL3_ST_CW_CERT_B;
3131                else
3132                        s->state=SSL3_ST_CW_CERT_C;
3133                }
3134
3135        /* We need to get a client cert */
3136        if (s->state == SSL3_ST_CW_CERT_B)
3137                {
3138                /* If we get an error, we need to
3139                 * ssl->rwstate=SSL_X509_LOOKUP; return(-1);
3140                 * We then get retied later */
3141                i=0;
3142                i = ssl_do_client_cert_cb(s, &x509, &pkey);
3143                if (i < 0)
3144                        {
3145                        s->rwstate=SSL_X509_LOOKUP;
3146                        return(-1);
3147                        }
3148                s->rwstate=SSL_NOTHING;
3149                if ((i == 1) && (pkey != NULL) && (x509 != NULL))
3150                        {
3151                        s->state=SSL3_ST_CW_CERT_B;
3152                        if (    !SSL_use_certificate(s,x509) ||
3153                                !SSL_use_PrivateKey(s,pkey))
3154                                i=0;
3155                        }
3156                else if (i == 1)
3157                        {
3158                        i=0;
3159                        SSLerr(SSL_F_SSL3_SEND_CLIENT_CERTIFICATE,SSL_R_BAD_DATA
3160                        }
3161
3162                if (x509 != NULL) X509_free(x509);
3163                if (pkey != NULL) EVP_PKEY_free(pkey);
```

```
3164                 if (i == 0)
3165                         {
3166                         if (s->version == SSL3_VERSION)
3167                                 {
3168                                 s->s3->tmp.cert_req=0;
3169                                 ssl3_send_alert(s,SSL3_AL_WARNING,SSL_AD_NO_CERT
3170                                 return(1);
3171                                 }
3172                         else
3173                                 {
3174                                 s->s3->tmp.cert_req=2;
3175                                 }
3176                         }

3178                 /* Ok, we have a cert */
3179                 s->state=SSL3_ST_CW_CERT_C;
3180                 }

3182         if (s->state == SSL3_ST_CW_CERT_C)
3183                 {
3184                 s->state=SSL3_ST_CW_CERT_D;
3185                 l=ssl3_output_cert_chain(s,
3186                         (s->s3->tmp.cert_req == 2)?NULL:s->cert->key->x509);
3187                 s->init_num=(int)l;
3188                 s->init_off=0;
3189                 }
3190         /* SSL3_ST_CW_CERT_D */
3191         return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
3192         }

3194 #define has_bits(i,m)   (((i)&(m)) == (m))

3196 int ssl3_check_cert_and_algorithm(SSL *s)
3197         {
3198         int i,idx;
3199         long alg_k,alg_a;
3200         EVP_PKEY *pkey=NULL;
3201         SESS_CERT *sc;
3202 #ifndef OPENSSL_NO_RSA
3203         RSA *rsa;
3204 #endif
3205 #ifndef OPENSSL_NO_DH
3206         DH *dh;
3207 #endif

3209         alg_k=s->s3->tmp.new_cipher->algorithm_mkey;
3210         alg_a=s->s3->tmp.new_cipher->algorithm_auth;

3212         /* we don't have a certificate */
3213         if ((alg_a & (SSL_aDH|SSL_aNULL|SSL_aKRB5)) || (alg_k & SSL_kPSK))
3214                 return(1);

3216         sc=s->session->sess_cert;
3217         if (sc == NULL)
3218                 {
3219                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,ERR_R_INTERNAL_ERROR)
3220                 goto err;
3221                 }

3223 #ifndef OPENSSL_NO_RSA
3224         rsa=s->session->sess_cert->peer_rsa_tmp;
3225 #endif
3226 #ifndef OPENSSL_NO_DH
3227         dh=s->session->sess_cert->peer_dh_tmp;
3228 #endif
```



```
3230         /* This is the passed certificate */

3232         idx=sc->peer_cert_type;
3233 #ifndef OPENSSL_NO_ECDH
3234         if (idx == SSL_PKEY_ECC)
3235                 {
3236                 if (ssl_check_srvr_ecc_cert_and_alg(sc->peer_pkeys[idx].x509,
3237                                                                 s) == 0)
3238                         { /* check failed */
3239                         SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_BAD_ECC
3240                         goto f_err;
3241                         }
3242                 else
3243                         {
3244                         return 1;
3245                         }
3246                 }
3247 #endif
3248         pkey=X509_get_pubkey(sc->peer_pkeys[idx].x509);
3249         i=X509_certificate_type(sc->peer_pkeys[idx].x509,pkey);
3250         EVP_PKEY_free(pkey);


3253         /* Check that we have a certificate if we require one */
3254         if ((alg_a & SSL_aRSA) && !has_bits(i,EVP_PK_RSA|EVP_PKT_SIGN))
3255                 {
3256                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_MISSING_RSA_SIG
3257                 goto f_err;
3258                 }
3259 #ifndef OPENSSL_NO_DSA
3260         else if ((alg_a & SSL_aDSS) && !has_bits(i,EVP_PK_DSA|EVP_PKT_SIGN))
3261                 {
3262                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_MISSING_DSA_SIG
3263                 goto f_err;
3264                 }
3265 #endif
3266 #ifndef OPENSSL_NO_RSA
3267         if ((alg_k & SSL_kRSA) &&
3268                 !(has_bits(i,EVP_PK_RSA|EVP_PKT_ENC) || (rsa != NULL)))
3269                 {
3270                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_MISSING_RSA_ENC
3271                 goto f_err;
3272                 }
3273 #endif
3274 #ifndef OPENSSL_NO_DH
3275         if ((alg_k & SSL_kEDH) &&
3276                 !(has_bits(i,EVP_PK_DH|EVP_PKT_EXCH) || (dh != NULL)))
3277                 {
3278                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_MISSING_DH_KEY)
3279                 goto f_err;
3280                 }
3281         else if ((alg_k & SSL_kDHr) && !has_bits(i,EVP_PK_DH|EVP_PKS_RSA))
3282                 {
3283                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_MISSING_DH_RSA_
3284                 goto f_err;
3285                 }
3286 #ifndef OPENSSL_NO_DSA
3287         else if ((alg_k & SSL_kDHd) && !has_bits(i,EVP_PK_DH|EVP_PKS_DSA))
3288                 {
3289                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_MISSING_DH_DSA_
3290                 goto f_err;
3291                 }
3292 #endif
3293 #endif

3295         if (SSL_C_IS_EXPORT(s->s3->tmp.new_cipher) && !has_bits(i,EVP_PKT_EXP))
```

```
3296                 {
3297 #ifndef OPENSSL_NO_RSA
3298                 if (alg_k & SSL_kRSA)
3299                         {
3300                         if (rsa == NULL
3301                             || RSA_size(rsa)*8 > SSL_C_EXPORT_PKEYLENGTH(s->s3->
3302                                 {
3303                                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R
3304                                 goto f_err;
3305                                 }
3306                         }
3307                 else
3308 #endif
3309 #ifndef OPENSSL_NO_DH
3310                         if (alg_k & (SSL_kEDH|SSL_kDHr|SSL_kDHd))
3311                             {
3312                             if (dh == NULL
3313                                 || DH_size(dh)*8 > SSL_C_EXPORT_PKEYLENGTH(s->s3
3314                                 {
3315                                 SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R
3316                                 goto f_err;
3317                                 }
3318                         }
3319                 else
3320 #endif
3321                         {
3322                         SSLerr(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM,SSL_R_UNKNOWN
3323                         goto f_err;
3324                         }
3325                 }
3326         return(1);
3327 f_err:
3328         ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_HANDSHAKE_FAILURE);
3329 err:
3330         return(0);
3331         }

3333 #if !defined(OPENSSL_NO_TLSEXT) && !defined(OPENSSL_NO_NEXTPROTONEG)
3334 int ssl3_send_next_proto(SSL *s)
3335         {
3336         unsigned int len, padding_len;
3337         unsigned char *d;

3339         if (s->state == SSL3_ST_CW_NEXT_PROTO_A)
3340                 {
3341                 len = s->next_proto_negotiated_len;
3342                 padding_len = 32 - ((len + 2) % 32);
3343                 d = (unsigned char *)s->init_buf->data;
3344                 d[4] = len;
3345                 memcpy(d + 5, s->next_proto_negotiated, len);
3346                 d[5 + len] = padding_len;
3347                 memset(d + 6 + len, 0, padding_len);
3348                 *(d++)=SSL3_MT_NEXT_PROTO;
3349                 l2n3(2 + len + padding_len, d);
3350                 s->state = SSL3_ST_CW_NEXT_PROTO_B;
3351                 s->init_num = 4 + 2 + len + padding_len;
3352                 s->init_off = 0;
3353                 }

3355         return ssl3_do_write(s, SSL3_RT_HANDSHAKE);
3356 }
3357 #endif  /* !OPENSSL_NO_TLSEXT && !OPENSSL_NO_NEXTPROTONEG */

3359 /* Check to see if handshake is full or resumed. Usually this is just a
3360  * case of checking to see if a cache hit has occurred. In the case of
3361  * session tickets we have to check the next message to be sure.
```



```
3362  */

3364 #ifndef OPENSSL_NO_TLSEXT
3365 int ssl3_check_finished(SSL *s)
3366         {
3367         int ok;
3368         long n;
3369         /* If we have no ticket it cannot be a resumed session. */
3370         if (!s->session->tlsext_tick)
3371                 return 1;
3372         /* this function is called when we really expect a Certificate
3373          * message, so permit appropriate message length */
3374         n=s->method->ssl_get_message(s,
3375                 SSL3_ST_CR_CERT_A,
3376                 SSL3_ST_CR_CERT_B,
3377                 -1,
3378                 s->max_cert_list,
3379                 &ok);
3380         if (!ok) return((int)n);
3381         s->s3->tmp.reuse_message = 1;
3382         if ((s->s3->tmp.message_type == SSL3_MT_FINISHED)
3383                 || (s->s3->tmp.message_type == SSL3_MT_NEWSESSION_TICKET))
3384                 return 2;

3386         return 1;
3387         }
3388 #endif

3390 int ssl_do_client_cert_cb(SSL *s, X509 **px509, EVP_PKEY **ppkey)
3391         {
3392         int i = 0;
3393 #ifndef OPENSSL_NO_ENGINE
3394         if (s->ctx->client_cert_engine)
3395                 {
3396                 i = ENGINE_load_ssl_client_cert(s->ctx->client_cert_engine, s,
3397                                                 SSL_get_client_CA_list(s),
3398                                                 px509, ppkey, NULL, NULL, NULL);
3399                 if (i != 0)
3400                         return i;
3401                 }
3402 #endif
3403         if (s->ctx->client_cert_cb)
3404                 i = s->ctx->client_cert_cb(s,px509,ppkey);
3405         return i;
3406         }
3407 #endif /* ! codereview */
```

```
**********************************************************
   26637 Wed Aug 13 19:53:38 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s3_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
```

```
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/evp.h>
#include <openssl/md5.h>

static unsigned char ssl3_pad_1[48]={
        0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,
        0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,
        0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,
        0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,
        0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,
        0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36,0x36 };

static unsigned char ssl3_pad_2[48]={
        0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,
        0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,
        0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,
        0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,
        0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,
        0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c,0x5c };
static int ssl3_handshake_mac(SSL *s, int md_nid,
        const char *sender, int len, unsigned char *p);
static int ssl3_generate_key_block(SSL *s, unsigned char *km, int num)
        {
        EVP_MD_CTX m5;
        EVP_MD_CTX s1;
        unsigned char buf[16],smd[SHA_DIGEST_LENGTH];
        unsigned char c='A';
        unsigned int i,j,k;

#ifdef CHARSET_EBCDIC
        c = os_toascii[c]; /*'A' in ASCII */
#endif
        k=0;
        EVP_MD_CTX_init(&m5);
        EVP_MD_CTX_set_flags(&m5, EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
        EVP_MD_CTX_init(&s1);
        for (i=0; (int)i<num; i+=MD5_DIGEST_LENGTH)
                {
                k++;
                if (k > sizeof buf)
                        {
                        /* bug: 'buf' is too small for this ciphersuite */
                        SSLerr(SSL_F_SSL3_GENERATE_KEY_BLOCK, ERR_R_INTERNAL_ERR
                        return 0;
                        }

                for (j=0; j<k; j++)
                        buf[j]=c;
                c++;
                EVP_DigestInit_ex(&s1,EVP_sha1(), NULL);
                EVP_DigestUpdate(&s1,buf,k);
                EVP_DigestUpdate(&s1,s->session->master_key,
                        s->session->master_key_length);
                EVP_DigestUpdate(&s1,s->s3->server_random,SSL3_RANDOM_SIZE);
                EVP_DigestUpdate(&s1,s->s3->client_random,SSL3_RANDOM_SIZE);
```



```
                EVP_DigestFinal_ex(&s1,smd,NULL);

                EVP_DigestInit_ex(&m5,EVP_md5(), NULL);
                EVP_DigestUpdate(&m5,s->session->master_key,
                        s->session->master_key_length);
                EVP_DigestUpdate(&m5,smd,SHA_DIGEST_LENGTH);
                if ((int)(i+MD5_DIGEST_LENGTH) > num)
                        {
                        EVP_DigestFinal_ex(&m5,smd,NULL);
                        memcpy(km,smd,(num-i));
                        }
                else
                        EVP_DigestFinal_ex(&m5,km,NULL);

                km+=MD5_DIGEST_LENGTH;
                }
        OPENSSL_cleanse(smd,SHA_DIGEST_LENGTH);
        EVP_MD_CTX_cleanup(&m5);
        EVP_MD_CTX_cleanup(&s1);
        return 1;
        }

int ssl3_change_cipher_state(SSL *s, int which)
        {
        unsigned char *p,*mac_secret;
        unsigned char exp_key[EVP_MAX_KEY_LENGTH];
        unsigned char exp_iv[EVP_MAX_IV_LENGTH];
        unsigned char *ms,*key,*iv,*er1,*er2;
        EVP_CIPHER_CTX *dd;
        const EVP_CIPHER *c;
#ifndef OPENSSL_NO_COMP
        COMP_METHOD *comp;
#endif
        const EVP_MD *m;
        EVP_MD_CTX md;
        int is_exp,n,i,j,k,cl;
        int reuse_dd = 0;

        is_exp=SSL_C_IS_EXPORT(s->s3->tmp.new_cipher);
        c=s->s3->tmp.new_sym_enc;
        m=s->s3->tmp.new_hash;
        /* m == NULL will lead to a crash later */
        OPENSSL_assert(m);
#ifndef OPENSSL_NO_COMP
        if (s->s3->tmp.new_compression == NULL)
                comp=NULL;
        else
                comp=s->s3->tmp.new_compression->method;
#endif

        if (which & SSL3_CC_READ)
                {
                if (s->enc_read_ctx != NULL)
                        reuse_dd = 1;
                else if ((s->enc_read_ctx=OPENSSL_malloc(sizeof(EVP_CIPHER_CTX))
                        goto err;
                else
                        /* make sure it's intialized in case we exit later with
                        EVP_CIPHER_CTX_init(s->enc_read_ctx);
                dd= s->enc_read_ctx;

                ssl_replace_hash(&s->read_hash,m);
#ifndef OPENSSL_NO_COMP
                /* COMPRESS */
                if (s->expand != NULL)
                        {
```

```
260                         COMP_CTX_free(s->expand);
261                         s->expand=NULL;
262                         }
263                 if (comp != NULL)
264                         {
265                         s->expand=COMP_CTX_new(comp);
266                         if (s->expand == NULL)
267                                 {
268                                 SSLerr(SSL_F_SSL3_CHANGE_CIPHER_STATE,SSL_R_COMP
269                                 goto err2;
270                                 }
271                         if (s->s3->rrec.comp == NULL)
272                                 s->s3->rrec.comp=(unsigned char *)
273                                         OPENSSL_malloc(SSL3_RT_MAX_PLAIN_LENGTH)
274                         if (s->s3->rrec.comp == NULL)
275                                 goto err;
276                         }
277 #endif
278                 memset(&(s->s3->read_sequence[0]),0,8);
279                 mac_secret= &(s->s3->read_mac_secret[0]);
280                 }
281         else
282                 {
283                 if (s->enc_write_ctx != NULL)
284                         reuse_dd = 1;
285                 else if ((s->enc_write_ctx=OPENSSL_malloc(sizeof(EVP_CIPHER_CTX)
286                         goto err;
287                 else
288                         /* make sure it's intialized in case we exit later with
289                         EVP_CIPHER_CTX_init(s->enc_write_ctx);
290                 dd= s->enc_write_ctx;
291                 ssl_replace_hash(&s->write_hash,m);
292 #ifndef OPENSSL_NO_COMP
293                 /* COMPRESS */
294                 if (s->compress != NULL)
295                         {
296                         COMP_CTX_free(s->compress);
297                         s->compress=NULL;
298                         }
299                 if (comp != NULL)
300                         {
301                         s->compress=COMP_CTX_new(comp);
302                         if (s->compress == NULL)
303                                 {
304                                 SSLerr(SSL_F_SSL3_CHANGE_CIPHER_STATE,SSL_R_COMP
305                                 goto err2;
306                                 }
307                         }
308 #endif
309                 memset(&(s->s3->write_sequence[0]),0,8);
310                 mac_secret= &(s->s3->write_mac_secret[0]);
311                 }

313         if (reuse_dd)
314                 EVP_CIPHER_CTX_cleanup(dd);

316         p=s->s3->tmp.key_block;
317         i=EVP_MD_size(m);
318         if (i < 0)
319                 goto err2;
320         cl=EVP_CIPHER_key_length(c);
321         j=is_exp ? (cl < SSL_C_EXPORT_KEYLENGTH(s->s3->tmp.new_cipher) ?
322                  cl : SSL_C_EXPORT_KEYLENGTH(s->s3->tmp.new_cipher)) : cl;
323         /* Was j=(is_exp)?5:EVP_CIPHER_key_length(c); */
324         k=EVP_CIPHER_iv_length(c);
325         if (    (which == SSL3_CHANGE_CIPHER_CLIENT_WRITE) ||
```



```
326                 (which == SSL3_CHANGE_CIPHER_SERVER_READ))
327                 {
328                 ms=  &(p[ 0]); n=i+i;
329                 key= &(p[ n]); n+=j+j;
330                 iv=  &(p[ n]); n+=k+k;
331                 er1= &(s->s3->client_random[0]);
332                 er2= &(s->s3->server_random[0]);
333                 }
334         else
335                 {
336                 n=i;
337                 ms=  &(p[ n]); n+=i+j;
338                 key= &(p[ n]); n+=j+k;
339                 iv=  &(p[ n]); n+=k;
340                 er1= &(s->s3->server_random[0]);
341                 er2= &(s->s3->client_random[0]);
342                 }

344         if (n > s->s3->tmp.key_block_length)
345                 {
346                 SSLerr(SSL_F_SSL3_CHANGE_CIPHER_STATE,ERR_R_INTERNAL_ERROR);
347                 goto err2;
348                 }

350         EVP_MD_CTX_init(&md);
351         memcpy(mac_secret,ms,i);
352         if (is_exp)
353                 {
354                 /* In here I set both the read and write key/iv to the
355                  * same value since only the correct one will be used :-).
356                  */
357                 EVP_DigestInit_ex(&md,EVP_md5(), NULL);
358                 EVP_DigestUpdate(&md,key,j);
359                 EVP_DigestUpdate(&md,er1,SSL3_RANDOM_SIZE);
360                 EVP_DigestUpdate(&md,er2,SSL3_RANDOM_SIZE);
361                 EVP_DigestFinal_ex(&md,&(exp_key[0]),NULL);
362                 key= &(exp_key[0]);

364                 if (k > 0)
365                         {
366                         EVP_DigestInit_ex(&md,EVP_md5(), NULL);
367                         EVP_DigestUpdate(&md,er1,SSL3_RANDOM_SIZE);
368                         EVP_DigestUpdate(&md,er2,SSL3_RANDOM_SIZE);
369                         EVP_DigestFinal_ex(&md,&(exp_iv[0]),NULL);
370                         iv= &(exp_iv[0]);
371                         }
372                 }

374         s->session->key_arg_length=0;

376         EVP_CipherInit_ex(dd,c,NULL,key,iv,(which & SSL3_CC_WRITE));

378         OPENSSL_cleanse(&(exp_key[0]),sizeof(exp_key));
379         OPENSSL_cleanse(&(exp_iv[0]),sizeof(exp_iv));
380         EVP_MD_CTX_cleanup(&md);
381         return(1);
382 err:
383         SSLerr(SSL_F_SSL3_CHANGE_CIPHER_STATE,ERR_R_MALLOC_FAILURE);
384 err2:
385         return(0);
386         }

388 int ssl3_setup_key_block(SSL *s)
389         {
390         unsigned char *p;
391         const EVP_CIPHER *c;
```

```
392         const EVP_MD *hash;
393         int num;
394         int ret = 0;
395         SSL_COMP *comp;

397         if (s->s3->tmp.key_block_length != 0)
398                 return(1);

400         if (!ssl_cipher_get_evp(s->session,&c,&hash,NULL,NULL,&comp))
401                 {
402                 SSLerr(SSL_F_SSL3_SETUP_KEY_BLOCK,SSL_R_CIPHER_OR_HASH_UNAVAILAB
403                 return(0);
404                 }

406         s->s3->tmp.new_sym_enc=c;
407         s->s3->tmp.new_hash=hash;
408 #ifdef OPENSSL_NO_COMP
409         s->s3->tmp.new_compression=NULL;
410 #else
411         s->s3->tmp.new_compression=comp;
412 #endif

414         num=EVP_MD_size(hash);
415         if (num < 0)
416                 return 0;

418         num=EVP_CIPHER_key_length(c)+num+EVP_CIPHER_iv_length(c);
419         num*=2;

421         ssl3_cleanup_key_block(s);

423         if ((p=OPENSSL_malloc(num)) == NULL)
424                 goto err;

426         s->s3->tmp.key_block_length=num;
427         s->s3->tmp.key_block=p;

429         ret = ssl3_generate_key_block(s,p,num);

431         if (!(s->options & SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS))
432                 {
433                 /* enable vulnerability countermeasure for CBC ciphers with
434                  * known-IV problem (http://www.openssl.org/~bodo/tls-cbc.txt)
435                  */
436                 s->s3->need_empty_fragments = 1;

438                 if (s->session->cipher != NULL)
439                         {
440                         if (s->session->cipher->algorithm_enc == SSL_eNULL)
441                                 s->s3->need_empty_fragments = 0;

443 #ifndef OPENSSL_NO_RC4
444                         if (s->session->cipher->algorithm_enc == SSL_RC4)
445                                 s->s3->need_empty_fragments = 0;
446 #endif
447                         }
448                 }

450         return ret;

452 err:
453         SSLerr(SSL_F_SSL3_SETUP_KEY_BLOCK,ERR_R_MALLOC_FAILURE);
454         return(0);
455         }

457 void ssl3_cleanup_key_block(SSL *s)
```



```
458         {
459         if (s->s3->tmp.key_block != NULL)
460                 {
461                 OPENSSL_cleanse(s->s3->tmp.key_block,
462                         s->s3->tmp.key_block_length);
463                 OPENSSL_free(s->s3->tmp.key_block);
464                 s->s3->tmp.key_block=NULL;
465                 }
466         s->s3->tmp.key_block_length=0;
467         }

469 /* ssl3_enc encrypts/decrypts the record in |s->wrec| / |s->rrec|, respectively.
470  *
471  * Returns:
472  *   0: (in non-constant time) if the record is publically invalid (i.e. too
473  *       short etc).
474  *   1: if the record's padding is valid / the encryption was successful.
475  *   -1: if the record's padding is invalid or, if sending, an internal error
476  *       occured.
477  */
478 int ssl3_enc(SSL *s, int send)
479         {
480         SSL3_RECORD *rec;
481         EVP_CIPHER_CTX *ds;
482         unsigned long l;
483         int bs,i,mac_size=0;
484         const EVP_CIPHER *enc;

486         if (send)
487                 {
488                 ds=s->enc_write_ctx;
489                 rec= &(s->s3->wrec);
490                 if (s->enc_write_ctx == NULL)
491                         enc=NULL;
492                 else
493                         enc=EVP_CIPHER_CTX_cipher(s->enc_write_ctx);
494                 }
495         else
496                 {
497                 ds=s->enc_read_ctx;
498                 rec= &(s->s3->rrec);
499                 if (s->enc_read_ctx == NULL)
500                         enc=NULL;
501                 else
502                         enc=EVP_CIPHER_CTX_cipher(s->enc_read_ctx);
503                 }

505         if ((s->session == NULL) || (ds == NULL) ||
506                 (enc == NULL))
507                 {
508                 memmove(rec->data,rec->input,rec->length);
509                 rec->input=rec->data;
510                 }
511         else
512                 {
513                 l=rec->length;
514                 bs=EVP_CIPHER_block_size(ds->cipher);

516                 /* COMPRESS */

518                 if ((bs != 1) && send)
519                         {
520                         i=bs-((int)l%bs);

522                         /* we need to add 'i-1' padding bytes */
523                         l+=i;
```

```
524                         /* the last of these zero bytes will be overwritten
525                          * with the padding length. */
526                         memset(&rec->input[rec->length], 0, i);
527                         rec->length+=i;
528                         rec->input[l-1]=(i-1);
529                         }

531                 if (!send)
532                         {
533                         if (l == 0 || l%bs != 0)
534                                 return 0;
535                         /* otherwise, rec->length >= bs */
536                         }

538                 EVP_Cipher(ds,rec->data,rec->input,l);

540                 if (EVP_MD_CTX_md(s->read_hash) != NULL)
541                         mac_size = EVP_MD_CTX_size(s->read_hash);
542                 if ((bs != 1) && !send)
543                         return ssl3_cbc_remove_padding(s, rec, bs, mac_size);
544                 }
545         return(1);
546         }

548 void ssl3_init_finished_mac(SSL *s)
549         {
550         if (s->s3->handshake_buffer) BIO_free(s->s3->handshake_buffer);
551         if (s->s3->handshake_dgst) ssl3_free_digest_list(s);
552     s->s3->handshake_buffer=BIO_new(BIO_s_mem());
553         (void)BIO_set_close(s->s3->handshake_buffer,BIO_CLOSE);
554         }

556 void ssl3_free_digest_list(SSL *s)
557         {
558         int i;
559         if (!s->s3->handshake_dgst) return;
560         for (i=0;i<SSL_MAX_DIGEST;i++)
561                 {
562                 if (s->s3->handshake_dgst[i])
563                         EVP_MD_CTX_destroy(s->s3->handshake_dgst[i]);
564                 }
565         OPENSSL_free(s->s3->handshake_dgst);
566         s->s3->handshake_dgst=NULL;
567         }



571 void ssl3_finish_mac(SSL *s, const unsigned char *buf, int len)
572         {
573         if (s->s3->handshake_buffer && !(s->s3->flags & TLS1_FLAGS_KEEP_HANDSHAK
574                 {
575                 BIO_write (s->s3->handshake_buffer,(void *)buf,len);
576                 }
577         else
578                 {
579                 int i;
580                 for (i=0;i< SSL_MAX_DIGEST;i++)
581                         {
582                         if (s->s3->handshake_dgst[i]!= NULL)
583                         EVP_DigestUpdate(s->s3->handshake_dgst[i],buf,len);
584                         }
585                 }
586         }

588 int ssl3_digest_cached_records(SSL *s)
589         {
```



```
590         int i;
591         long mask;
592         const EVP_MD *md;
593         long hdatalen;
594         void *hdata;

596         /* Allocate handshake_dgst array */
597         ssl3_free_digest_list(s);
598         s->s3->handshake_dgst = OPENSSL_malloc(SSL_MAX_DIGEST * sizeof(EVP_MD_CT
599         memset(s->s3->handshake_dgst,0,SSL_MAX_DIGEST *sizeof(EVP_MD_CTX *));
600         hdatalen = BIO_get_mem_data(s->s3->handshake_buffer,&hdata);
601         if (hdatalen <= 0)
602                 {
603                 SSLerr(SSL_F_SSL3_DIGEST_CACHED_RECORDS, SSL_R_BAD_HANDSHAKE_LEN
604                 return 0;
605                 }

607         /* Loop through bitso of algorithm2 field and create MD_CTX-es */
608         for (i=0;ssl_get_handshake_digest(i,&mask,&md); i++)
609                 {
610                 if ((mask & ssl_get_algorithm2(s)) && md)
611                         {
612                         s->s3->handshake_dgst[i]=EVP_MD_CTX_create();
613 #ifdef OPENSSL_FIPS
614                         if (EVP_MD_nid(md) == NID_md5)
615                                 {
616                                 EVP_MD_CTX_set_flags(s->s3->handshake_dgst[i],
617                                                 EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
618                                 }
619 #endif
620                         EVP_DigestInit_ex(s->s3->handshake_dgst[i],md,NULL);
621                         EVP_DigestUpdate(s->s3->handshake_dgst[i],hdata,hdatalen
622                         }
623                 else
624                         {
625                         s->s3->handshake_dgst[i]=NULL;
626                         }
627                 }
628         if (!(s->s3->flags & TLS1_FLAGS_KEEP_HANDSHAKE))
629                 {
630                 /* Free handshake_buffer BIO */
631                 BIO_free(s->s3->handshake_buffer);
632                 s->s3->handshake_buffer = NULL;
633                 }

635         return 1;
636         }

638 int ssl3_cert_verify_mac(SSL *s, int md_nid, unsigned char *p)
639         {
640         return(ssl3_handshake_mac(s,md_nid,NULL,0,p));
641         }
642 int ssl3_final_finish_mac(SSL *s,
643              const char *sender, int len, unsigned char *p)
644         {
645         int ret, sha1len;
646         ret=ssl3_handshake_mac(s,NID_md5,sender,len,p);
647         if(ret == 0)
648                 return 0;

650         p+=ret;

652         sha1len=ssl3_handshake_mac(s,NID_sha1,sender,len,p);
653         if(sha1len == 0)
654                 return 0;
```

```
656         ret+=sha1len;
657         return(ret);
658         }
659 static int ssl3_handshake_mac(SSL *s, int md_nid,
660              const char *sender, int len, unsigned char *p)
661         {
662         unsigned int ret;
663         int npad,n;
664         unsigned int i;
665         unsigned char md_buf[EVP_MAX_MD_SIZE];
666         EVP_MD_CTX ctx,*d=NULL;

668         if (s->s3->handshake_buffer)
669                 if (!ssl3_digest_cached_records(s))
670                         return 0;

672         /* Search for digest of specified type in the handshake_dgst
673          * array*/
674         for (i=0;i<SSL_MAX_DIGEST;i++)
675                 {
676                   if (s->s3->handshake_dgst[i]&&EVP_MD_CTX_type(s->s3->handshake
677                         {
678                         d=s->s3->handshake_dgst[i];
679                         break;
680                         }
681                 }
682         if (!d) {
683                 SSLerr(SSL_F_SSL3_HANDSHAKE_MAC,SSL_R_NO_REQUIRED_DIGEST);
684                 return 0;
685         }
686         EVP_MD_CTX_init(&ctx);
687         EVP_MD_CTX_set_flags(&ctx, EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
688         EVP_MD_CTX_copy_ex(&ctx,d);
689         n=EVP_MD_CTX_size(&ctx);
690         if (n < 0)
691                 return 0;

693         npad=(48/n)*n;
694         if (sender != NULL)
695                 EVP_DigestUpdate(&ctx,sender,len);
696         EVP_DigestUpdate(&ctx,s->session->master_key,
697                 s->session->master_key_length);
698         EVP_DigestUpdate(&ctx,ssl3_pad_1,npad);
699         EVP_DigestFinal_ex(&ctx,md_buf,&i);

701         EVP_DigestInit_ex(&ctx,EVP_MD_CTX_md(&ctx), NULL);
702         EVP_DigestUpdate(&ctx,s->session->master_key,
703                 s->session->master_key_length);
704         EVP_DigestUpdate(&ctx,ssl3_pad_2,npad);
705         EVP_DigestUpdate(&ctx,md_buf,i);
706         EVP_DigestFinal_ex(&ctx,p,&ret);

708         EVP_MD_CTX_cleanup(&ctx);

710         return((int)ret);
711         }

713 int n_ssl3_mac(SSL *ssl, unsigned char *md, int send)
714         {
715         SSL3_RECORD *rec;
716         unsigned char *mac_sec,*seq;
717         EVP_MD_CTX md_ctx;
718         const EVP_MD_CTX *hash;
719         unsigned char *p,rec_char;
720         size_t md_size, orig_len;
721         int npad;
```



```
722         int t;

724         if (send)
725                 {
726                 rec= &(ssl->s3->wrec);
727                 mac_sec= &(ssl->s3->write_mac_secret[0]);
728                 seq= &(ssl->s3->write_sequence[0]);
729                 hash=ssl->write_hash;
730                 }
731         else
732                 {
733                 rec= &(ssl->s3->rrec);
734                 mac_sec= &(ssl->s3->read_mac_secret[0]);
735                 seq= &(ssl->s3->read_sequence[0]);
736                 hash=ssl->read_hash;
737                 }

739         t=EVP_MD_CTX_size(hash);
740         if (t < 0)
741                 return -1;
742         md_size=t;
743         npad=(48/md_size)*md_size;

745         /* kludge: ssl3_cbc_remove_padding passes padding length in rec->type */
746         orig_len = rec->length+md_size+((unsigned int)rec->type>>8);
747         rec->type &= 0xff;

749         if (!send &&
750             EVP_CIPHER_CTX_mode(ssl->enc_read_ctx) == EVP_CIPH_CBC_MODE &&
751             ssl3_cbc_record_digest_supported(hash))
752                 {
753                 /* This is a CBC-encrypted record. We must avoid leaking any
754                  * timing-side channel information about how many blocks of
755                  * data we are hashing because that gives an attacker a
756                  * timing-oracle. */

758                 /* npad is, at most, 48 bytes and that's with MD5:
759                  *   16 + 48 + 8 (sequence bytes) + 1 + 2 = 75.
760                  *
761                  * With SHA-1 (the largest hash speced for SSLv3) the hash size
762                  * goes up 4, but npad goes down by 8, resulting in a smaller
763                  * total size. */
764                 unsigned char header[75];
765                 unsigned j = 0;
766                 memcpy(header+j, mac_sec, md_size);
767                 j += md_size;
768                 memcpy(header+j, ssl3_pad_1, npad);
769                 j += npad;
770                 memcpy(header+j, seq, 8);
771                 j += 8;
772                 header[j++] = rec->type;
773                 header[j++] = rec->length >> 8;
774                 header[j++] = rec->length & 0xff;

776                 ssl3_cbc_digest_record(
777                         hash,
778                         md, &md_size,
779                         header, rec->input,
780                         rec->length + md_size, orig_len,
781                         mac_sec, md_size,
782                         1 /* is SSLv3 */);
783                 }
784         else
785                 {
786                 unsigned int md_size_u;
787                 /* Chop the digest off the end :-) */
```

```
788                 EVP_MD_CTX_init(&md_ctx);

790                 EVP_MD_CTX_copy_ex( &md_ctx,hash);
791                 EVP_DigestUpdate(&md_ctx,mac_sec,md_size);
792                 EVP_DigestUpdate(&md_ctx,ssl3_pad_1,npad);
793                 EVP_DigestUpdate(&md_ctx,seq,8);
794                 rec_char=rec->type;
795                 EVP_DigestUpdate(&md_ctx,&rec_char,1);
796                 p=md;
797                 s2n(rec->length,p);
798                 EVP_DigestUpdate(&md_ctx,md,2);
799                 EVP_DigestUpdate(&md_ctx,rec->input,rec->length);
800                 EVP_DigestFinal_ex( &md_ctx,md,NULL);

802                 EVP_MD_CTX_copy_ex( &md_ctx,hash);
803                 EVP_DigestUpdate(&md_ctx,mac_sec,md_size);
804                 EVP_DigestUpdate(&md_ctx,ssl3_pad_2,npad);
805                 EVP_DigestUpdate(&md_ctx,md,md_size);
806                 EVP_DigestFinal_ex( &md_ctx,md,&md_size_u);
807                 md_size = md_size_u;

809                 EVP_MD_CTX_cleanup(&md_ctx);
810         }

812         ssl3_record_sequence_update(seq);
813         return(md_size);
814         }

816 void ssl3_record_sequence_update(unsigned char *seq)
817         {
818         int i;

820         for (i=7; i>=0; i--)
821                 {
822                 ++seq[i];
823                 if (seq[i] != 0) break;
824                 }
825         }

827 int ssl3_generate_master_secret(SSL *s, unsigned char *out, unsigned char *p,
828                 int len)
829         {
830         static const unsigned char *salt[3]={
831 #ifndef CHARSET_EBCDIC
832                 (const unsigned char *)"A",
833                 (const unsigned char *)"BB",
834                 (const unsigned char *)"CCC",
835 #else
836                 (const unsigned char *)"\x41",
837                 (const unsigned char *)"\x42\x42",
838                 (const unsigned char *)"\x43\x43\x43",
839 #endif
840                 };
841         unsigned char buf[EVP_MAX_MD_SIZE];
842         EVP_MD_CTX ctx;
843         int i,ret=0;
844         unsigned int n;

846         EVP_MD_CTX_init(&ctx);
847         for (i=0; i<3; i++)
848                 {
849                 EVP_DigestInit_ex(&ctx,s->ctx->sha1, NULL);
850                 EVP_DigestUpdate(&ctx,salt[i],strlen((const char *)salt[i]));
851                 EVP_DigestUpdate(&ctx,p,len);
852                 EVP_DigestUpdate(&ctx,&(s->s3->client_random[0]),
853                         SSL3_RANDOM_SIZE);
```



```
854                 EVP_DigestUpdate(&ctx,&(s->s3->server_random[0]),
855                         SSL3_RANDOM_SIZE);
856                 EVP_DigestFinal_ex(&ctx,buf,&n);

858                 EVP_DigestInit_ex(&ctx,s->ctx->md5, NULL);
859                 EVP_DigestUpdate(&ctx,p,len);
860                 EVP_DigestUpdate(&ctx,buf,n);
861                 EVP_DigestFinal_ex(&ctx,out,&n);
862                 out+=n;
863                 ret+=n;
864                 }
865         EVP_MD_CTX_cleanup(&ctx);
866         return(ret);
867         }

869 int ssl3_alert_code(int code)
870         {
871         switch (code)
872                 {
873         case SSL_AD_CLOSE_NOTIFY:       return(SSL3_AD_CLOSE_NOTIFY);
874         case SSL_AD_UNEXPECTED_MESSAGE: return(SSL3_AD_UNEXPECTED_MESSAGE);
875         case SSL_AD_BAD_RECORD_MAC:     return(SSL3_AD_BAD_RECORD_MAC);
876         case SSL_AD_DECRYPTION_FAILED:  return(SSL3_AD_BAD_RECORD_MAC);
877         case SSL_AD_RECORD_OVERFLOW:    return(SSL3_AD_BAD_RECORD_MAC);
878         case SSL_AD_DECOMPRESSION_FAILURE:return(SSL3_AD_DECOMPRESSION_FAILURE);
879         case SSL_AD_HANDSHAKE_FAILURE:  return(SSL3_AD_HANDSHAKE_FAILURE);
880         case SSL_AD_NO_CERTIFICATE:     return(SSL3_AD_NO_CERTIFICATE);
881         case SSL_AD_BAD_CERTIFICATE:    return(SSL3_AD_BAD_CERTIFICATE);
882         case SSL_AD_UNSUPPORTED_CERTIFICATE:return(SSL3_AD_UNSUPPORTED_CERTIFICA
883         case SSL_AD_CERTIFICATE_REVOKED:return(SSL3_AD_CERTIFICATE_REVOKED);
884         case SSL_AD_CERTIFICATE_EXPIRED:return(SSL3_AD_CERTIFICATE_EXPIRED);
885         case SSL_AD_CERTIFICATE_UNKNOWN:return(SSL3_AD_CERTIFICATE_UNKNOWN);
886         case SSL_AD_ILLEGAL_PARAMETER:  return(SSL3_AD_ILLEGAL_PARAMETER);
887         case SSL_AD_UNKNOWN_CA:         return(SSL3_AD_BAD_CERTIFICATE);
888         case SSL_AD_ACCESS_DENIED:      return(SSL3_AD_HANDSHAKE_FAILURE);
889         case SSL_AD_DECODE_ERROR:       return(SSL3_AD_HANDSHAKE_FAILURE);
890         case SSL_AD_DECRYPT_ERROR:      return(SSL3_AD_HANDSHAKE_FAILURE);
891         case SSL_AD_EXPORT_RESTRICTION: return(SSL3_AD_HANDSHAKE_FAILURE);
892         case SSL_AD_PROTOCOL_VERSION:   return(SSL3_AD_HANDSHAKE_FAILURE);
893         case SSL_AD_INSUFFICIENT_SECURITY:return(SSL3_AD_HANDSHAKE_FAILURE);
894         case SSL_AD_INTERNAL_ERROR:     return(SSL3_AD_HANDSHAKE_FAILURE);
895         case SSL_AD_USER_CANCELLED:     return(SSL3_AD_HANDSHAKE_FAILURE);
896         case SSL_AD_NO_RENEGOTIATION:   return(-1); /* Don't send it :-) */
897         case SSL_AD_UNSUPPORTED_EXTENSION: return(SSL3_AD_HANDSHAKE_FAILURE);
898         case SSL_AD_CERTIFICATE_UNOBTAINABLE: return(SSL3_AD_HANDSHAKE_FAILURE);
899         case SSL_AD_UNRECOGNIZED_NAME:  return(SSL3_AD_HANDSHAKE_FAILURE);
900         case SSL_AD_BAD_CERTIFICATE_STATUS_RESPONSE: return(SSL3_AD_HANDSHAKE_FA
901         case SSL_AD_BAD_CERTIFICATE_HASH_VALUE: return(SSL3_AD_HANDSHAKE_FAILURE
902         case SSL_AD_UNKNOWN_PSK_IDENTITY:return(TLS1_AD_UNKNOWN_PSK_IDENTITY);
903         default:                        return(-1);
904                 }
905         }
906 #endif /* ! codereview */
```

```
**********************************************************
   83244 Wed Aug 13 19:53:38 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s3_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * Portions of the attached software ("Contribution") are developed by
 * SUN MICROSYSTEMS, INC., and are contributed to the OpenSSL project.
 *
 * The Contribution is licensed pursuant to the OpenSSL open source
 * license provided above.
 *
 * ECC cipher suite support in OpenSSL originally written by
 * Vipul Gupta and Sumit Gupta of Sun Microsystems Laboratories.
 *
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
```

```
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include <openssl/objects.h>
#include "ssl_locl.h"
#include "kssl_lcl.h"
#ifndef OPENSSL_NO_TLSEXT
#ifndef OPENSSL_NO_EC
#include <ec_lcl.h>
#endif /* OPENSSL_NO_EC */
#endif /* OPENSSL_NO_TLSEXT */
#include <openssl/md5.h>
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif

const char ssl3_version_str[]="SSLv3" OPENSSL_VERSION_PTEXT;

#define SSL3_NUM_CIPHERS        (sizeof(ssl3_ciphers)/sizeof(SSL_CIPHER))

/* list of available SSLv3 ciphers (sorted by id) */
OPENSSL_GLOBAL SSL_CIPHER ssl3_ciphers[]={

/* The RSA ciphers */
/* Cipher 01 */
        {
        1,
        SSL3_TXT_RSA_NULL_MD5,
        SSL3_CK_RSA_NULL_MD5,
        SSL_kRSA,
        SSL_aRSA,
        SSL_eNULL,
        SSL_MD5,
        SSL_SSLV3,
        SSL_NOT_EXP|SSL_STRONG_NONE,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        0,
        0,
        },

/* Cipher 02 */
        {
        1,
        SSL3_TXT_RSA_NULL_SHA,
        SSL3_CK_RSA_NULL_SHA,
```



```
        SSL_kRSA,
        SSL_aRSA,
        SSL_eNULL,
        SSL_SHA1,
        SSL_SSLV3,
        SSL_NOT_EXP|SSL_STRONG_NONE|SSL_FIPS,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        0,
        0,
        },

/* Cipher 03 */
        {
        1,
        SSL3_TXT_RSA_RC4_40_MD5,
        SSL3_CK_RSA_RC4_40_MD5,
        SSL_kRSA,
        SSL_aRSA,
        SSL_RC4,
        SSL_MD5,
        SSL_SSLV3,
        SSL_EXPORT|SSL_EXP40,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        40,
        128,
        },

/* Cipher 04 */
        {
        1,
        SSL3_TXT_RSA_RC4_128_MD5,
        SSL3_CK_RSA_RC4_128_MD5,
        SSL_kRSA,
        SSL_aRSA,
        SSL_RC4,
        SSL_MD5,
        SSL_SSLV3,
        SSL_NOT_EXP|SSL_MEDIUM,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        128,
        128,
        },

/* Cipher 05 */
        {
        1,
        SSL3_TXT_RSA_RC4_128_SHA,
        SSL3_CK_RSA_RC4_128_SHA,
        SSL_kRSA,
        SSL_aRSA,
        SSL_RC4,
        SSL_SHA1,
        SSL_SSLV3,
        SSL_NOT_EXP|SSL_MEDIUM,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        128,
        128,
        },

/* Cipher 06 */
        {
        1,
        SSL3_TXT_RSA_RC2_40_MD5,
        SSL3_CK_RSA_RC2_40_MD5,
        SSL_kRSA,
        SSL_aRSA,
```

```
260         SSL_RC2,
261         SSL_MD5,
262         SSL_SSLV3,
263         SSL_EXPORT|SSL_EXP40,
264         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
265         40,
266         128,
267         },

269 /* Cipher 07 */
270 #ifndef OPENSSL_NO_IDEA
271         {
272         1,
273         SSL3_TXT_RSA_IDEA_128_SHA,
274         SSL3_CK_RSA_IDEA_128_SHA,
275         SSL_kRSA,
276         SSL_aRSA,
277         SSL_IDEA,
278         SSL_SHA1,
279         SSL_SSLV3,
280         SSL_NOT_EXP|SSL_MEDIUM,
281         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
282         128,
283         128,
284         },
285 #endif

287 /* Cipher 08 */
288         {
289         1,
290         SSL3_TXT_RSA_DES_40_CBC_SHA,
291         SSL3_CK_RSA_DES_40_CBC_SHA,
292         SSL_kRSA,
293         SSL_aRSA,
294         SSL_DES,
295         SSL_SHA1,
296         SSL_SSLV3,
297         SSL_EXPORT|SSL_EXP40,
298         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
299         40,
300         56,
301         },

303 /* Cipher 09 */
304         {
305         1,
306         SSL3_TXT_RSA_DES_64_CBC_SHA,
307         SSL3_CK_RSA_DES_64_CBC_SHA,
308         SSL_kRSA,
309         SSL_aRSA,
310         SSL_DES,
311         SSL_SHA1,
312         SSL_SSLV3,
313         SSL_NOT_EXP|SSL_LOW,
314         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
315         56,
316         56,
317         },

319 /* Cipher 0A */
320         {
321         1,
322         SSL3_TXT_RSA_DES_192_CBC3_SHA,
323         SSL3_CK_RSA_DES_192_CBC3_SHA,
324         SSL_kRSA,
325         SSL_aRSA,
```



```
326         SSL_3DES,
327         SSL_SHA1,
328         SSL_SSLV3,
329         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
330         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
331         112,
332         168,
333         },

335 /* The DH ciphers */
336 /* Cipher 0B */
337         {
338         0,
339         SSL3_TXT_DH_DSS_DES_40_CBC_SHA,
340         SSL3_CK_DH_DSS_DES_40_CBC_SHA,
341         SSL_kDHd,
342         SSL_aDH,
343         SSL_DES,
344         SSL_SHA1,
345         SSL_SSLV3,
346         SSL_EXPORT|SSL_EXP40,
347         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
348         40,
349         56,
350         },

352 /* Cipher 0C */
353         {
354         0, /* not implemented (non-ephemeral DH) */
355         SSL3_TXT_DH_DSS_DES_64_CBC_SHA,
356         SSL3_CK_DH_DSS_DES_64_CBC_SHA,
357         SSL_kDHd,
358         SSL_aDH,
359         SSL_DES,
360         SSL_SHA1,
361         SSL_SSLV3,
362         SSL_NOT_EXP|SSL_LOW,
363         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
364         56,
365         56,
366         },

368 /* Cipher 0D */
369         {
370         0, /* not implemented (non-ephemeral DH) */
371         SSL3_TXT_DH_DSS_DES_192_CBC3_SHA,
372         SSL3_CK_DH_DSS_DES_192_CBC3_SHA,
373         SSL_kDHd,
374         SSL_aDH,
375         SSL_3DES,
376         SSL_SHA1,
377         SSL_SSLV3,
378         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
379         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
380         112,
381         168,
382         },

384 /* Cipher 0E */
385         {
386         0, /* not implemented (non-ephemeral DH) */
387         SSL3_TXT_DH_RSA_DES_40_CBC_SHA,
388         SSL3_CK_DH_RSA_DES_40_CBC_SHA,
389         SSL_kDHr,
390         SSL_aDH,
391         SSL_DES,
```

```
392         SSL_SHA1,
393         SSL_SSLV3,
394         SSL_EXPORT|SSL_EXP40,
395         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
396         40,
397         56,
398         },

400 /* Cipher 0F */
401         {
402         0, /* not implemented (non-ephemeral DH) */
403         SSL3_TXT_DH_RSA_DES_64_CBC_SHA,
404         SSL3_CK_DH_RSA_DES_64_CBC_SHA,
405         SSL_kDHr,
406         SSL_aDH,
407         SSL_DES,
408         SSL_SHA1,
409         SSL_SSLV3,
410         SSL_NOT_EXP|SSL_LOW,
411         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
412         56,
413         56,
414         },

416 /* Cipher 10 */
417         {
418         0, /* not implemented (non-ephemeral DH) */
419         SSL3_TXT_DH_RSA_DES_192_CBC3_SHA,
420         SSL3_CK_DH_RSA_DES_192_CBC3_SHA,
421         SSL_kDHr,
422         SSL_aDH,
423         SSL_3DES,
424         SSL_SHA1,
425         SSL_SSLV3,
426         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
427         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
428         112,
429         168,
430         },

432 /* The Ephemeral DH ciphers */
433 /* Cipher 11 */
434         {
435         1,
436         SSL3_TXT_EDH_DSS_DES_40_CBC_SHA,
437         SSL3_CK_EDH_DSS_DES_40_CBC_SHA,
438         SSL_kEDH,
439         SSL_aDSS,
440         SSL_DES,
441         SSL_SHA1,
442         SSL_SSLV3,
443         SSL_EXPORT|SSL_EXP40,
444         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
445         40,
446         56,
447         },

449 /* Cipher 12 */
450         {
451         1,
452         SSL3_TXT_EDH_DSS_DES_64_CBC_SHA,
453         SSL3_CK_EDH_DSS_DES_64_CBC_SHA,
454         SSL_kEDH,
455         SSL_aDSS,
456         SSL_DES,
457         SSL_SHA1,
```



```
458         SSL_SSLV3,
459         SSL_NOT_EXP|SSL_LOW,
460         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
461         56,
462         56,
463         },

465 /* Cipher 13 */
466         {
467         1,
468         SSL3_TXT_EDH_DSS_DES_192_CBC3_SHA,
469         SSL3_CK_EDH_DSS_DES_192_CBC3_SHA,
470         SSL_kEDH,
471         SSL_aDSS,
472         SSL_3DES,
473         SSL_SHA1,
474         SSL_SSLV3,
475         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
476         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
477         112,
478         168,
479         },

481 /* Cipher 14 */
482         {
483         1,
484         SSL3_TXT_EDH_RSA_DES_40_CBC_SHA,
485         SSL3_CK_EDH_RSA_DES_40_CBC_SHA,
486         SSL_kEDH,
487         SSL_aRSA,
488         SSL_DES,
489         SSL_SHA1,
490         SSL_SSLV3,
491         SSL_EXPORT|SSL_EXP40,
492         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
493         40,
494         56,
495         },

497 /* Cipher 15 */
498         {
499         1,
500         SSL3_TXT_EDH_RSA_DES_64_CBC_SHA,
501         SSL3_CK_EDH_RSA_DES_64_CBC_SHA,
502         SSL_kEDH,
503         SSL_aRSA,
504         SSL_DES,
505         SSL_SHA1,
506         SSL_SSLV3,
507         SSL_NOT_EXP|SSL_LOW,
508         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
509         56,
510         56,
511         },

513 /* Cipher 16 */
514         {
515         1,
516         SSL3_TXT_EDH_RSA_DES_192_CBC3_SHA,
517         SSL3_CK_EDH_RSA_DES_192_CBC3_SHA,
518         SSL_kEDH,
519         SSL_aRSA,
520         SSL_3DES,
521         SSL_SHA1,
522         SSL_SSLV3,
523         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
```

```
524         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
525         112,
526         168,
527         },

529 /* Cipher 17 */
530         {
531         1,
532         SSL3_TXT_ADH_RC4_40_MD5,
533         SSL3_CK_ADH_RC4_40_MD5,
534         SSL_kEDH,
535         SSL_aNULL,
536         SSL_RC4,
537         SSL_MD5,
538         SSL_SSLV3,
539         SSL_EXPORT|SSL_EXP40,
540         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
541         40,
542         128,
543         },

545 /* Cipher 18 */
546         {
547         1,
548         SSL3_TXT_ADH_RC4_128_MD5,
549         SSL3_CK_ADH_RC4_128_MD5,
550         SSL_kEDH,
551         SSL_aNULL,
552         SSL_RC4,
553         SSL_MD5,
554         SSL_SSLV3,
555         SSL_NOT_EXP|SSL_MEDIUM,
556         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
557         128,
558         128,
559         },

561 /* Cipher 19 */
562         {
563         1,
564         SSL3_TXT_ADH_DES_40_CBC_SHA,
565         SSL3_CK_ADH_DES_40_CBC_SHA,
566         SSL_kEDH,
567         SSL_aNULL,
568         SSL_DES,
569         SSL_SHA1,
570         SSL_SSLV3,
571         SSL_EXPORT|SSL_EXP40,
572         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
573         40,
574         128,
575         },

577 /* Cipher 1A */
578         {
579         1,
580         SSL3_TXT_ADH_DES_64_CBC_SHA,
581         SSL3_CK_ADH_DES_64_CBC_SHA,
582         SSL_kEDH,
583         SSL_aNULL,
584         SSL_DES,
585         SSL_SHA1,
586         SSL_SSLV3,
587         SSL_NOT_EXP|SSL_LOW,
588         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
589         56,
```



```
590         56,
591         },

593 /* Cipher 1B */
594         {
595         1,
596         SSL3_TXT_ADH_DES_192_CBC_SHA,
597         SSL3_CK_ADH_DES_192_CBC_SHA,
598         SSL_kEDH,
599         SSL_aNULL,
600         SSL_3DES,
601         SSL_SHA1,
602         SSL_SSLV3,
603         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
604         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
605         112,
606         168,
607         },

609 /* Fortezza ciphersuite from SSL 3.0 spec */
610 #if 0
611 /* Cipher 1C */
612         {
613         0,
614         SSL3_TXT_FZA_DMS_NULL_SHA,
615         SSL3_CK_FZA_DMS_NULL_SHA,
616         SSL_kFZA,
617         SSL_aFZA,
618         SSL_eNULL,
619         SSL_SHA1,
620         SSL_SSLV3,
621         SSL_NOT_EXP|SSL_STRONG_NONE,
622         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
623         0,
624         0,
625         },

627 /* Cipher 1D */
628         {
629         0,
630         SSL3_TXT_FZA_DMS_FZA_SHA,
631         SSL3_CK_FZA_DMS_FZA_SHA,
632         SSL_kFZA,
633         SSL_aFZA,
634         SSL_eFZA,
635         SSL_SHA1,
636         SSL_SSLV3,
637         SSL_NOT_EXP|SSL_STRONG_NONE,
638         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
639         0,
640         0,
641         },

643 /* Cipher 1E */
644         {
645         0,
646         SSL3_TXT_FZA_DMS_RC4_SHA,
647         SSL3_CK_FZA_DMS_RC4_SHA,
648         SSL_kFZA,
649         SSL_aFZA,
650         SSL_RC4,
651         SSL_SHA1,
652         SSL_SSLV3,
653         SSL_NOT_EXP|SSL_MEDIUM,
654         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
655         128,
```

```
656         128,
657         },
658 #endif

660 #ifndef OPENSSL_NO_KRB5
661 /* The Kerberos ciphers*/
662 /* Cipher 1E */
663         {
664         1,
665         SSL3_TXT_KRB5_DES_64_CBC_SHA,
666         SSL3_CK_KRB5_DES_64_CBC_SHA,
667         SSL_kKRB5,
668         SSL_aKRB5,
669         SSL_DES,
670         SSL_SHA1,
671         SSL_SSLV3,
672         SSL_NOT_EXP|SSL_LOW,
673         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
674         56,
675         56,
676         },

678 /* Cipher 1F */
679         {
680         1,
681         SSL3_TXT_KRB5_DES_192_CBC3_SHA,
682         SSL3_CK_KRB5_DES_192_CBC3_SHA,
683         SSL_kKRB5,
684         SSL_aKRB5,
685         SSL_3DES,
686         SSL_SHA1,
687         SSL_SSLV3,
688         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
689         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
690         112,
691         168,
692         },

694 /* Cipher 20 */
695         {
696         1,
697         SSL3_TXT_KRB5_RC4_128_SHA,
698         SSL3_CK_KRB5_RC4_128_SHA,
699         SSL_kKRB5,
700         SSL_aKRB5,
701         SSL_RC4,
702         SSL_SHA1,
703         SSL_SSLV3,
704         SSL_NOT_EXP|SSL_MEDIUM,
705         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
706         128,
707         128,
708         },

710 /* Cipher 21 */
711         {
712         1,
713         SSL3_TXT_KRB5_IDEA_128_CBC_SHA,
714         SSL3_CK_KRB5_IDEA_128_CBC_SHA,
715         SSL_kKRB5,
716         SSL_aKRB5,
717         SSL_IDEA,
718         SSL_SHA1,
719         SSL_SSLV3,
720         SSL_NOT_EXP|SSL_MEDIUM,
721         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
```



```
722         128,
723         128,
724         },

726 /* Cipher 22 */
727         {
728         1,
729         SSL3_TXT_KRB5_DES_64_CBC_MD5,
730         SSL3_CK_KRB5_DES_64_CBC_MD5,
731         SSL_kKRB5,
732         SSL_aKRB5,
733         SSL_DES,
734         SSL_MD5,
735         SSL_SSLV3,
736         SSL_NOT_EXP|SSL_LOW,
737         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
738         56,
739         56,
740         },

742 /* Cipher 23 */
743         {
744         1,
745         SSL3_TXT_KRB5_DES_192_CBC3_MD5,
746         SSL3_CK_KRB5_DES_192_CBC3_MD5,
747         SSL_kKRB5,
748         SSL_aKRB5,
749         SSL_3DES,
750         SSL_MD5,
751         SSL_SSLV3,
752         SSL_NOT_EXP|SSL_HIGH,
753         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
754         112,
755         168,
756         },

758 /* Cipher 24 */
759         {
760         1,
761         SSL3_TXT_KRB5_RC4_128_MD5,
762         SSL3_CK_KRB5_RC4_128_MD5,
763         SSL_kKRB5,
764         SSL_aKRB5,
765         SSL_RC4,
766         SSL_MD5,
767         SSL_SSLV3,
768         SSL_NOT_EXP|SSL_MEDIUM,
769         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
770         128,
771         128,
772         },

774 /* Cipher 25 */
775         {
776         1,
777         SSL3_TXT_KRB5_IDEA_128_CBC_MD5,
778         SSL3_CK_KRB5_IDEA_128_CBC_MD5,
779         SSL_kKRB5,
780         SSL_aKRB5,
781         SSL_IDEA,
782         SSL_MD5,
783         SSL_SSLV3,
784         SSL_NOT_EXP|SSL_MEDIUM,
785         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
786         128,
787         128,
```

```
788         },

790 /* Cipher 26 */
791         {
792         1,
793         SSL3_TXT_KRB5_DES_40_CBC_SHA,
794         SSL3_CK_KRB5_DES_40_CBC_SHA,
795         SSL_kKRB5,
796         SSL_aKRB5,
797         SSL_DES,
798         SSL_SHA1,
799         SSL_SSLV3,
800         SSL_EXPORT|SSL_EXP40,
801         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
802         40,
803         56,
804         },

806 /* Cipher 27 */
807         {
808         1,
809         SSL3_TXT_KRB5_RC2_40_CBC_SHA,
810         SSL3_CK_KRB5_RC2_40_CBC_SHA,
811         SSL_kKRB5,
812         SSL_aKRB5,
813         SSL_RC2,
814         SSL_SHA1,
815         SSL_SSLV3,
816         SSL_EXPORT|SSL_EXP40,
817         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
818         40,
819         128,
820         },

822 /* Cipher 28 */
823         {
824         1,
825         SSL3_TXT_KRB5_RC4_40_SHA,
826         SSL3_CK_KRB5_RC4_40_SHA,
827         SSL_kKRB5,
828         SSL_aKRB5,
829         SSL_RC4,
830         SSL_SHA1,
831         SSL_SSLV3,
832         SSL_EXPORT|SSL_EXP40,
833         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
834         40,
835         128,
836         },

838 /* Cipher 29 */
839         {
840         1,
841         SSL3_TXT_KRB5_DES_40_CBC_MD5,
842         SSL3_CK_KRB5_DES_40_CBC_MD5,
843         SSL_kKRB5,
844         SSL_aKRB5,
845         SSL_DES,
846         SSL_MD5,
847         SSL_SSLV3,
848         SSL_EXPORT|SSL_EXP40,
849         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
850         40,
851         56,
852         },
```



```
854 /* Cipher 2A */
855         {
856         1,
857         SSL3_TXT_KRB5_RC2_40_CBC_MD5,
858         SSL3_CK_KRB5_RC2_40_CBC_MD5,
859         SSL_kKRB5,
860         SSL_aKRB5,
861         SSL_RC2,
862         SSL_MD5,
863         SSL_SSLV3,
864         SSL_EXPORT|SSL_EXP40,
865         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
866         40,
867         128,
868         },

870 /* Cipher 2B */
871         {
872         1,
873         SSL3_TXT_KRB5_RC4_40_MD5,
874         SSL3_CK_KRB5_RC4_40_MD5,
875         SSL_kKRB5,
876         SSL_aKRB5,
877         SSL_RC4,
878         SSL_MD5,
879         SSL_SSLV3,
880         SSL_EXPORT|SSL_EXP40,
881         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
882         40,
883         128,
884         },
885 #endif  /* OPENSSL_NO_KRB5 */

887 /* New AES ciphersuites */
888 /* Cipher 2F */
889         {
890         1,
891         TLS1_TXT_RSA_WITH_AES_128_SHA,
892         TLS1_CK_RSA_WITH_AES_128_SHA,
893         SSL_kRSA,
894         SSL_aRSA,
895         SSL_AES128,
896         SSL_SHA1,
897         SSL_TLSV1,
898         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
899         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
900         128,
901         128,
902         },
903 /* Cipher 30 */
904         {
905         0,
906         TLS1_TXT_DH_DSS_WITH_AES_128_SHA,
907         TLS1_CK_DH_DSS_WITH_AES_128_SHA,
908         SSL_kDHd,
909         SSL_aDH,
910         SSL_AES128,
911         SSL_SHA1,
912         SSL_TLSV1,
913         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
914         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
915         128,
916         128,
917         },
918 /* Cipher 31 */
919         {
```

```
920         0,
921         TLS1_TXT_DH_RSA_WITH_AES_128_SHA,
922         TLS1_CK_DH_RSA_WITH_AES_128_SHA,
923         SSL_kDHr,
924         SSL_aDH,
925         SSL_AES128,
926         SSL_SHA1,
927         SSL_TLSV1,
928         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
929         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
930         128,
931         128,
932         },
933 /* Cipher 32 */
934         {
935         1,
936         TLS1_TXT_DHE_DSS_WITH_AES_128_SHA,
937         TLS1_CK_DHE_DSS_WITH_AES_128_SHA,
938         SSL_kEDH,
939         SSL_aDSS,
940         SSL_AES128,
941         SSL_SHA1,
942         SSL_TLSV1,
943         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
944         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
945         128,
946         128,
947         },
948 /* Cipher 33 */
949         {
950         1,
951         TLS1_TXT_DHE_RSA_WITH_AES_128_SHA,
952         TLS1_CK_DHE_RSA_WITH_AES_128_SHA,
953         SSL_kEDH,
954         SSL_aRSA,
955         SSL_AES128,
956         SSL_SHA1,
957         SSL_TLSV1,
958         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
959         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
960         128,
961         128,
962         },
963 /* Cipher 34 */
964         {
965         1,
966         TLS1_TXT_ADH_WITH_AES_128_SHA,
967         TLS1_CK_ADH_WITH_AES_128_SHA,
968         SSL_kEDH,
969         SSL_aNULL,
970         SSL_AES128,
971         SSL_SHA1,
972         SSL_TLSV1,
973         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
974         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
975         128,
976         128,
977         },

979 /* Cipher 35 */
980         {
981         1,
982         TLS1_TXT_RSA_WITH_AES_256_SHA,
983         TLS1_CK_RSA_WITH_AES_256_SHA,
984         SSL_kRSA,
985         SSL_aRSA,
```



```
986         SSL_AES256,
987         SSL_SHA1,
988         SSL_TLSV1,
989         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
990         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
991         256,
992         256,
993         },
994 /* Cipher 36 */
995         {
996         0,
997         TLS1_TXT_DH_DSS_WITH_AES_256_SHA,
998         TLS1_CK_DH_DSS_WITH_AES_256_SHA,
999         SSL_kDHd,
1000        SSL_aDH,
1001        SSL_AES256,
1002        SSL_SHA1,
1003        SSL_TLSV1,
1004        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1005        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1006        256,
1007        256,
1008        },

1010 /* Cipher 37 */
1011        {
1012        0, /* not implemented (non-ephemeral DH) */
1013        TLS1_TXT_DH_RSA_WITH_AES_256_SHA,
1014        TLS1_CK_DH_RSA_WITH_AES_256_SHA,
1015        SSL_kDHr,
1016        SSL_aDH,
1017        SSL_AES256,
1018        SSL_SHA1,
1019        SSL_TLSV1,
1020        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1021        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1022        256,
1023        256,
1024        },

1026 /* Cipher 38 */
1027        {
1028        1,
1029        TLS1_TXT_DHE_DSS_WITH_AES_256_SHA,
1030        TLS1_CK_DHE_DSS_WITH_AES_256_SHA,
1031        SSL_kEDH,
1032        SSL_aDSS,
1033        SSL_AES256,
1034        SSL_SHA1,
1035        SSL_TLSV1,
1036        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1037        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1038        256,
1039        256,
1040        },

1042 /* Cipher 39 */
1043        {
1044        1,
1045        TLS1_TXT_DHE_RSA_WITH_AES_256_SHA,
1046        TLS1_CK_DHE_RSA_WITH_AES_256_SHA,
1047        SSL_kEDH,
1048        SSL_aRSA,
1049        SSL_AES256,
1050        SSL_SHA1,
1051        SSL_TLSV1,
```

```
1052         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1053         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1054         256,
1055         256,
1056         },

1058         /* Cipher 3A */
1059         {
1060         1,
1061         TLS1_TXT_ADH_WITH_AES_256_SHA,
1062         TLS1_CK_ADH_WITH_AES_256_SHA,
1063         SSL_kEDH,
1064         SSL_aNULL,
1065         SSL_AES256,
1066         SSL_SHA1,
1067         SSL_TLSV1,
1068         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1069         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1070         256,
1071         256,
1072         },

1074         /* TLS v1.2 ciphersuites */
1075         /* Cipher 3B */
1076         {
1077         1,
1078         TLS1_TXT_RSA_WITH_NULL_SHA256,
1079         TLS1_CK_RSA_WITH_NULL_SHA256,
1080         SSL_kRSA,
1081         SSL_aRSA,
1082         SSL_eNULL,
1083         SSL_SHA256,
1084         SSL_TLSV1_2,
1085         SSL_NOT_EXP|SSL_STRONG_NONE|SSL_FIPS,
1086         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1087         0,
1088         0,
1089         },

1091         /* Cipher 3C */
1092         {
1093         1,
1094         TLS1_TXT_RSA_WITH_AES_128_SHA256,
1095         TLS1_CK_RSA_WITH_AES_128_SHA256,
1096         SSL_kRSA,
1097         SSL_aRSA,
1098         SSL_AES128,
1099         SSL_SHA256,
1100         SSL_TLSV1_2,
1101         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1102         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1103         128,
1104         128,
1105         },

1107         /* Cipher 3D */
1108         {
1109         1,
1110         TLS1_TXT_RSA_WITH_AES_256_SHA256,
1111         TLS1_CK_RSA_WITH_AES_256_SHA256,
1112         SSL_kRSA,
1113         SSL_aRSA,
1114         SSL_AES256,
1115         SSL_SHA256,
1116         SSL_TLSV1_2,
1117         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
```



```
1118         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1119         256,
1120         256,
1121         },

1123         /* Cipher 3E */
1124         {
1125         0, /* not implemented (non-ephemeral DH) */
1126         TLS1_TXT_DH_DSS_WITH_AES_128_SHA256,
1127         TLS1_CK_DH_DSS_WITH_AES_128_SHA256,
1128         SSL_kDHd,
1129         SSL_aDH,
1130         SSL_AES128,
1131         SSL_SHA256,
1132         SSL_TLSV1_2,
1133         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1134         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1135         128,
1136         128,
1137         },

1139         /* Cipher 3F */
1140         {
1141         0, /* not implemented (non-ephemeral DH) */
1142         TLS1_TXT_DH_RSA_WITH_AES_128_SHA256,
1143         TLS1_CK_DH_RSA_WITH_AES_128_SHA256,
1144         SSL_kDHr,
1145         SSL_aDH,
1146         SSL_AES128,
1147         SSL_SHA256,
1148         SSL_TLSV1_2,
1149         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1150         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1151         128,
1152         128,
1153         },

1155         /* Cipher 40 */
1156         {
1157         1,
1158         TLS1_TXT_DHE_DSS_WITH_AES_128_SHA256,
1159         TLS1_CK_DHE_DSS_WITH_AES_128_SHA256,
1160         SSL_kEDH,
1161         SSL_aDSS,
1162         SSL_AES128,
1163         SSL_SHA256,
1164         SSL_TLSV1_2,
1165         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1166         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1167         128,
1168         128,
1169         },

1171 #ifndef OPENSSL_NO_CAMELLIA
1172         /* Camellia ciphersuites from RFC4132 (128-bit portion) */

1174         /* Cipher 41 */
1175         {
1176         1,
1177         TLS1_TXT_RSA_WITH_CAMELLIA_128_CBC_SHA,
1178         TLS1_CK_RSA_WITH_CAMELLIA_128_CBC_SHA,
1179         SSL_kRSA,
1180         SSL_aRSA,
1181         SSL_CAMELLIA128,
1182         SSL_SHA1,
1183         SSL_TLSV1,
```

```
        SSL_NOT_EXP|SSL_HIGH,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        128,
        128,
        },

        /* Cipher 42 */
        {
        0, /* not implemented (non-ephemeral DH) */
        TLS1_TXT_DH_DSS_WITH_CAMELLIA_128_CBC_SHA,
        TLS1_CK_DH_DSS_WITH_CAMELLIA_128_CBC_SHA,
        SSL_kDHd,
        SSL_aDH,
        SSL_CAMELLIA128,
        SSL_SHA1,
        SSL_TLSV1,
        SSL_NOT_EXP|SSL_HIGH,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        128,
        128,
        },

        /* Cipher 43 */
        {
        0, /* not implemented (non-ephemeral DH) */
        TLS1_TXT_DH_RSA_WITH_CAMELLIA_128_CBC_SHA,
        TLS1_CK_DH_RSA_WITH_CAMELLIA_128_CBC_SHA,
        SSL_kDHr,
        SSL_aDH,
        SSL_CAMELLIA128,
        SSL_SHA1,
        SSL_TLSV1,
        SSL_NOT_EXP|SSL_HIGH,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        128,
        128,
        },

        /* Cipher 44 */
        {
        1,
        TLS1_TXT_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA,
        TLS1_CK_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA,
        SSL_kEDH,
        SSL_aDSS,
        SSL_CAMELLIA128,
        SSL_SHA1,
        SSL_TLSV1,
        SSL_NOT_EXP|SSL_HIGH,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        128,
        128,
        },

        /* Cipher 45 */
        {
        1,
        TLS1_TXT_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA,
        TLS1_CK_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA,
        SSL_kEDH,
        SSL_aRSA,
        SSL_CAMELLIA128,
        SSL_SHA1,
        SSL_TLSV1,
        SSL_NOT_EXP|SSL_HIGH,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
```



```
        128,
        128,
        },

        /* Cipher 46 */
        {
        1,
        TLS1_TXT_ADH_WITH_CAMELLIA_128_CBC_SHA,
        TLS1_CK_ADH_WITH_CAMELLIA_128_CBC_SHA,
        SSL_kEDH,
        SSL_aNULL,
        SSL_CAMELLIA128,
        SSL_SHA1,
        SSL_TLSV1,
        SSL_NOT_EXP|SSL_HIGH,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        128,
        128,
        },
#endif /* OPENSSL_NO_CAMELLIA */

#if TLS1_ALLOW_EXPERIMENTAL_CIPHERSUITES
        /* New TLS Export CipherSuites from expired ID */
#if 0
        /* Cipher 60 */
        {
        1,
        TLS1_TXT_RSA_EXPORT1024_WITH_RC4_56_MD5,
        TLS1_CK_RSA_EXPORT1024_WITH_RC4_56_MD5,
        SSL_kRSA,
        SSL_aRSA,
        SSL_RC4,
        SSL_MD5,
        SSL_TLSV1,
        SSL_EXPORT|SSL_EXP56,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        56,
        128,
        },

        /* Cipher 61 */
        {
        1,
        TLS1_TXT_RSA_EXPORT1024_WITH_RC2_CBC_56_MD5,
        TLS1_CK_RSA_EXPORT1024_WITH_RC2_CBC_56_MD5,
        SSL_kRSA,
        SSL_aRSA,
        SSL_RC2,
        SSL_MD5,
        SSL_TLSV1,
        SSL_EXPORT|SSL_EXP56,
        SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
        56,
        128,
        },
#endif

        /* Cipher 62 */
        {
        1,
        TLS1_TXT_RSA_EXPORT1024_WITH_DES_CBC_SHA,
        TLS1_CK_RSA_EXPORT1024_WITH_DES_CBC_SHA,
        SSL_kRSA,
        SSL_aRSA,
        SSL_DES,
        SSL_SHA1,
```

```
1316         SSL_TLSV1,
1317         SSL_EXPORT|SSL_EXP56,
1318         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1319         56,
1320         56,
1321         },

1323         /* Cipher 63 */
1324         {
1325         1,
1326         TLS1_TXT_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA,
1327         TLS1_CK_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA,
1328         SSL_kEDH,
1329         SSL_aDSS,
1330         SSL_DES,
1331         SSL_SHA1,
1332         SSL_TLSV1,
1333         SSL_EXPORT|SSL_EXP56,
1334         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1335         56,
1336         56,
1337         },

1339         /* Cipher 64 */
1340         {
1341         1,
1342         TLS1_TXT_RSA_EXPORT1024_WITH_RC4_56_SHA,
1343         TLS1_CK_RSA_EXPORT1024_WITH_RC4_56_SHA,
1344         SSL_kRSA,
1345         SSL_aRSA,
1346         SSL_RC4,
1347         SSL_SHA1,
1348         SSL_TLSV1,
1349         SSL_EXPORT|SSL_EXP56,
1350         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1351         56,
1352         128,
1353         },

1355         /* Cipher 65 */
1356         {
1357         1,
1358         TLS1_TXT_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA,
1359         TLS1_CK_DHE_DSS_EXPORT1024_WITH_RC4_56_SHA,
1360         SSL_kEDH,
1361         SSL_aDSS,
1362         SSL_RC4,
1363         SSL_SHA1,
1364         SSL_TLSV1,
1365         SSL_EXPORT|SSL_EXP56,
1366         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1367         56,
1368         128,
1369         },

1371         /* Cipher 66 */
1372         {
1373         1,
1374         TLS1_TXT_DHE_DSS_WITH_RC4_128_SHA,
1375         TLS1_CK_DHE_DSS_WITH_RC4_128_SHA,
1376         SSL_kEDH,
1377         SSL_aDSS,
1378         SSL_RC4,
1379         SSL_SHA1,
1380         SSL_TLSV1,
1381         SSL_NOT_EXP|SSL_MEDIUM,
```



```
1382         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1383         128,
1384         128,
1385         },
1386 #endif

1388         /* TLS v1.2 ciphersuites */
1389         /* Cipher 67 */
1390         {
1391         1,
1392         TLS1_TXT_DHE_RSA_WITH_AES_128_SHA256,
1393         TLS1_CK_DHE_RSA_WITH_AES_128_SHA256,
1394         SSL_kEDH,
1395         SSL_aRSA,
1396         SSL_AES128,
1397         SSL_SHA256,
1398         SSL_TLSV1_2,
1399         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1400         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1401         128,
1402         128,
1403         },

1405         /* Cipher 68 */
1406         {
1407         0, /* not implemented (non-ephemeral DH) */
1408         TLS1_TXT_DH_DSS_WITH_AES_256_SHA256,
1409         TLS1_CK_DH_DSS_WITH_AES_256_SHA256,
1410         SSL_kDHd,
1411         SSL_aDH,
1412         SSL_AES256,
1413         SSL_SHA256,
1414         SSL_TLSV1_2,
1415         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1416         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1417         256,
1418         256,
1419         },

1421         /* Cipher 69 */
1422         {
1423         0, /* not implemented (non-ephemeral DH) */
1424         TLS1_TXT_DH_RSA_WITH_AES_256_SHA256,
1425         TLS1_CK_DH_RSA_WITH_AES_256_SHA256,
1426         SSL_kDHr,
1427         SSL_aDH,
1428         SSL_AES256,
1429         SSL_SHA256,
1430         SSL_TLSV1_2,
1431         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1432         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1433         256,
1434         256,
1435         },

1437         /* Cipher 6A */
1438         {
1439         1,
1440         TLS1_TXT_DHE_DSS_WITH_AES_256_SHA256,
1441         TLS1_CK_DHE_DSS_WITH_AES_256_SHA256,
1442         SSL_kEDH,
1443         SSL_aDSS,
1444         SSL_AES256,
1445         SSL_SHA256,
1446         SSL_TLSV1_2,
1447         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
```

```
1448         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1449         256,
1450         256,
1451         },

1453         /* Cipher 6B */
1454         {
1455         1,
1456         TLS1_TXT_DHE_RSA_WITH_AES_256_SHA256,
1457         TLS1_CK_DHE_RSA_WITH_AES_256_SHA256,
1458         SSL_kEDH,
1459         SSL_aRSA,
1460         SSL_AES256,
1461         SSL_SHA256,
1462         SSL_TLSV1_2,
1463         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1464         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1465         256,
1466         256,
1467         },

1469         /* Cipher 6C */
1470         {
1471         1,
1472         TLS1_TXT_ADH_WITH_AES_128_SHA256,
1473         TLS1_CK_ADH_WITH_AES_128_SHA256,
1474         SSL_kEDH,
1475         SSL_aNULL,
1476         SSL_AES128,
1477         SSL_SHA256,
1478         SSL_TLSV1_2,
1479         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1480         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1481         128,
1482         128,
1483         },

1485         /* Cipher 6D */
1486         {
1487         1,
1488         TLS1_TXT_ADH_WITH_AES_256_SHA256,
1489         TLS1_CK_ADH_WITH_AES_256_SHA256,
1490         SSL_kEDH,
1491         SSL_aNULL,
1492         SSL_AES256,
1493         SSL_SHA256,
1494         SSL_TLSV1_2,
1495         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1496         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1497         256,
1498         256,
1499         },

1501         /* GOST Ciphersuites */

1503         {
1504         1,
1505         "GOST94-GOST89-GOST89",
1506         0x3000080,
1507         SSL_kGOST,
1508         SSL_aGOST94,
1509         SSL_eGOST2814789CNT,
1510         SSL_GOST89MAC,
1511         SSL_TLSV1,
1512         SSL_NOT_EXP|SSL_HIGH,
1513         SSL_HANDSHAKE_MAC_GOST94|TLS1_PRF_GOST94|TLS1_STREAM_MAC,
```



```
1514         256,
1515         256
1516         },
1517         {
1518         1,
1519         "GOST2001-GOST89-GOST89",
1520         0x3000081,
1521         SSL_kGOST,
1522         SSL_aGOST01,
1523         SSL_eGOST2814789CNT,
1524         SSL_GOST89MAC,
1525         SSL_TLSV1,
1526         SSL_NOT_EXP|SSL_HIGH,
1527         SSL_HANDSHAKE_MAC_GOST94|TLS1_PRF_GOST94|TLS1_STREAM_MAC,
1528         256,
1529         256
1530         },
1531         {
1532         1,
1533         "GOST94-NULL-GOST94",
1534         0x3000082,
1535         SSL_kGOST,
1536         SSL_aGOST94,
1537         SSL_eNULL,
1538         SSL_GOST94,
1539         SSL_TLSV1,
1540         SSL_NOT_EXP|SSL_STRONG_NONE,
1541         SSL_HANDSHAKE_MAC_GOST94|TLS1_PRF_GOST94,
1542         0,
1543         0
1544         },
1545         {
1546         1,
1547         "GOST2001-NULL-GOST94",
1548         0x3000083,
1549         SSL_kGOST,
1550         SSL_aGOST01,
1551         SSL_eNULL,
1552         SSL_GOST94,
1553         SSL_TLSV1,
1554         SSL_NOT_EXP|SSL_STRONG_NONE,
1555         SSL_HANDSHAKE_MAC_GOST94|TLS1_PRF_GOST94,
1556         0,
1557         0
1558         },

1560 #ifndef OPENSSL_NO_CAMELLIA
1561         /* Camellia ciphersuites from RFC4132 (256-bit portion) */

1563         /* Cipher 84 */
1564         {
1565         1,
1566         TLS1_TXT_RSA_WITH_CAMELLIA_256_CBC_SHA,
1567         TLS1_CK_RSA_WITH_CAMELLIA_256_CBC_SHA,
1568         SSL_kRSA,
1569         SSL_aRSA,
1570         SSL_CAMELLIA256,
1571         SSL_SHA1,
1572         SSL_TLSV1,
1573         SSL_NOT_EXP|SSL_HIGH,
1574         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1575         256,
1576         256,
1577         },
1578         /* Cipher 85 */
1579         {
```

```
1580         0, /* not implemented (non-ephemeral DH) */
1581         TLS1_TXT_DH_DSS_WITH_CAMELLIA_256_CBC_SHA,
1582         TLS1_CK_DH_DSS_WITH_CAMELLIA_256_CBC_SHA,
1583         SSL_kDHd,
1584         SSL_aDH,
1585         SSL_CAMELLIA256,
1586         SSL_SHA1,
1587         SSL_TLSV1,
1588         SSL_NOT_EXP|SSL_HIGH,
1589         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1590         256,
1591         256,
1592         },

1594         /* Cipher 86 */
1595         {
1596         0, /* not implemented (non-ephemeral DH) */
1597         TLS1_TXT_DH_RSA_WITH_CAMELLIA_256_CBC_SHA,
1598         TLS1_CK_DH_RSA_WITH_CAMELLIA_256_CBC_SHA,
1599         SSL_kDHr,
1600         SSL_aDH,
1601         SSL_CAMELLIA256,
1602         SSL_SHA1,
1603         SSL_TLSV1,
1604         SSL_NOT_EXP|SSL_HIGH,
1605         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1606         256,
1607         256,
1608         },

1610         /* Cipher 87 */
1611         {
1612         1,
1613         TLS1_TXT_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA,
1614         TLS1_CK_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA,
1615         SSL_kEDH,
1616         SSL_aDSS,
1617         SSL_CAMELLIA256,
1618         SSL_SHA1,
1619         SSL_TLSV1,
1620         SSL_NOT_EXP|SSL_HIGH,
1621         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1622         256,
1623         256,
1624         },

1626         /* Cipher 88 */
1627         {
1628         1,
1629         TLS1_TXT_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA,
1630         TLS1_CK_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA,
1631         SSL_kEDH,
1632         SSL_aRSA,
1633         SSL_CAMELLIA256,
1634         SSL_SHA1,
1635         SSL_TLSV1,
1636         SSL_NOT_EXP|SSL_HIGH,
1637         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1638         256,
1639         256,
1640         },

1642         /* Cipher 89 */
1643         {
1644         1,
1645         TLS1_TXT_ADH_WITH_CAMELLIA_256_CBC_SHA,
```



```
1646         TLS1_CK_ADH_WITH_CAMELLIA_256_CBC_SHA,
1647         SSL_kEDH,
1648         SSL_aNULL,
1649         SSL_CAMELLIA256,
1650         SSL_SHA1,
1651         SSL_TLSV1,
1652         SSL_NOT_EXP|SSL_HIGH,
1653         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1654         256,
1655         256,
1656         },
1657 #endif /* OPENSSL_NO_CAMELLIA */

1659 #ifndef OPENSSL_NO_PSK
1660         /* Cipher 8A */
1661         {
1662         1,
1663         TLS1_TXT_PSK_WITH_RC4_128_SHA,
1664         TLS1_CK_PSK_WITH_RC4_128_SHA,
1665         SSL_kPSK,
1666         SSL_aPSK,
1667         SSL_RC4,
1668         SSL_SHA1,
1669         SSL_TLSV1,
1670         SSL_NOT_EXP|SSL_MEDIUM,
1671         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1672         128,
1673         128,
1674         },

1676         /* Cipher 8B */
1677         {
1678         1,
1679         TLS1_TXT_PSK_WITH_3DES_EDE_CBC_SHA,
1680         TLS1_CK_PSK_WITH_3DES_EDE_CBC_SHA,
1681         SSL_kPSK,
1682         SSL_aPSK,
1683         SSL_3DES,
1684         SSL_SHA1,
1685         SSL_TLSV1,
1686         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1687         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1688         112,
1689         168,
1690         },

1692         /* Cipher 8C */
1693         {
1694         1,
1695         TLS1_TXT_PSK_WITH_AES_128_CBC_SHA,
1696         TLS1_CK_PSK_WITH_AES_128_CBC_SHA,
1697         SSL_kPSK,
1698         SSL_aPSK,
1699         SSL_AES128,
1700         SSL_SHA1,
1701         SSL_TLSV1,
1702         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1703         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1704         128,
1705         128,
1706         },

1708         /* Cipher 8D */
1709         {
1710         1,
1711         TLS1_TXT_PSK_WITH_AES_256_CBC_SHA,
```

```
1712         TLS1_CK_PSK_WITH_AES_256_CBC_SHA,
1713         SSL_kPSK,
1714         SSL_aPSK,
1715         SSL_AES256,
1716         SSL_SHA1,
1717         SSL_TLSV1,
1718         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1719         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1720         256,
1721         256,
1722         },
1723 #endif  /* OPENSSL_NO_PSK */

1725 #ifndef OPENSSL_NO_SEED
1726         /* SEED ciphersuites from RFC4162 */

1728         /* Cipher 96 */
1729         {
1730         1,
1731         TLS1_TXT_RSA_WITH_SEED_SHA,
1732         TLS1_CK_RSA_WITH_SEED_SHA,
1733         SSL_kRSA,
1734         SSL_aRSA,
1735         SSL_SEED,
1736         SSL_SHA1,
1737         SSL_TLSV1,
1738         SSL_NOT_EXP|SSL_MEDIUM,
1739         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1740         128,
1741         128,
1742         },

1744         /* Cipher 97 */
1745         {
1746         0, /* not implemented (non-ephemeral DH) */
1747         TLS1_TXT_DH_DSS_WITH_SEED_SHA,
1748         TLS1_CK_DH_DSS_WITH_SEED_SHA,
1749         SSL_kDHd,
1750         SSL_aDH,
1751         SSL_SEED,
1752         SSL_SHA1,
1753         SSL_TLSV1,
1754         SSL_NOT_EXP|SSL_MEDIUM,
1755         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1756         128,
1757         128,
1758         },

1760         /* Cipher 98 */
1761         {
1762         0, /* not implemented (non-ephemeral DH) */
1763         TLS1_TXT_DH_RSA_WITH_SEED_SHA,
1764         TLS1_CK_DH_RSA_WITH_SEED_SHA,
1765         SSL_kDHr,
1766         SSL_aDH,
1767         SSL_SEED,
1768         SSL_SHA1,
1769         SSL_TLSV1,
1770         SSL_NOT_EXP|SSL_MEDIUM,
1771         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1772         128,
1773         128,
1774         },

1776         /* Cipher 99 */
1777         {
```



```
1778         1,
1779         TLS1_TXT_DHE_DSS_WITH_SEED_SHA,
1780         TLS1_CK_DHE_DSS_WITH_SEED_SHA,
1781         SSL_kEDH,
1782         SSL_aDSS,
1783         SSL_SEED,
1784         SSL_SHA1,
1785         SSL_TLSV1,
1786         SSL_NOT_EXP|SSL_MEDIUM,
1787         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1788         128,
1789         128,
1790         },

1792         /* Cipher 9A */
1793         {
1794         1,
1795         TLS1_TXT_DHE_RSA_WITH_SEED_SHA,
1796         TLS1_CK_DHE_RSA_WITH_SEED_SHA,
1797         SSL_kEDH,
1798         SSL_aRSA,
1799         SSL_SEED,
1800         SSL_SHA1,
1801         SSL_TLSV1,
1802         SSL_NOT_EXP|SSL_MEDIUM,
1803         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1804         128,
1805         128,
1806         },

1808         /* Cipher 9B */
1809         {
1810         1,
1811         TLS1_TXT_ADH_WITH_SEED_SHA,
1812         TLS1_CK_ADH_WITH_SEED_SHA,
1813         SSL_kEDH,
1814         SSL_aNULL,
1815         SSL_SEED,
1816         SSL_SHA1,
1817         SSL_TLSV1,
1818         SSL_NOT_EXP|SSL_MEDIUM,
1819         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
1820         128,
1821         128,
1822         },

1824 #endif /* OPENSSL_NO_SEED */

1826         /* GCM ciphersuites from RFC5288 */

1828         /* Cipher 9C */
1829         {
1830         1,
1831         TLS1_TXT_RSA_WITH_AES_128_GCM_SHA256,
1832         TLS1_CK_RSA_WITH_AES_128_GCM_SHA256,
1833         SSL_kRSA,
1834         SSL_aRSA,
1835         SSL_AES128GCM,
1836         SSL_AEAD,
1837         SSL_TLSV1_2,
1838         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1839         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
1840         128,
1841         128,
1842         },
```

```
1844         /* Cipher 9D */
1845         {
1846         1,
1847         TLS1_TXT_RSA_WITH_AES_256_GCM_SHA384,
1848         TLS1_CK_RSA_WITH_AES_256_GCM_SHA384,
1849         SSL_kRSA,
1850         SSL_aRSA,
1851         SSL_AES256GCM,
1852         SSL_AEAD,
1853         SSL_TLSV1_2,
1854         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1855         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
1856         256,
1857         256,
1858         },

1860         /* Cipher 9E */
1861         {
1862         1,
1863         TLS1_TXT_DHE_RSA_WITH_AES_128_GCM_SHA256,
1864         TLS1_CK_DHE_RSA_WITH_AES_128_GCM_SHA256,
1865         SSL_kEDH,
1866         SSL_aRSA,
1867         SSL_AES128GCM,
1868         SSL_AEAD,
1869         SSL_TLSV1_2,
1870         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1871         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
1872         128,
1873         128,
1874         },

1876         /* Cipher 9F */
1877         {
1878         1,
1879         TLS1_TXT_DHE_RSA_WITH_AES_256_GCM_SHA384,
1880         TLS1_CK_DHE_RSA_WITH_AES_256_GCM_SHA384,
1881         SSL_kEDH,
1882         SSL_aRSA,
1883         SSL_AES256GCM,
1884         SSL_AEAD,
1885         SSL_TLSV1_2,
1886         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1887         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
1888         256,
1889         256,
1890         },

1892         /* Cipher A0 */
1893         {
1894         0,
1895         TLS1_TXT_DH_RSA_WITH_AES_128_GCM_SHA256,
1896         TLS1_CK_DH_RSA_WITH_AES_128_GCM_SHA256,
1897         SSL_kDHr,
1898         SSL_aDH,
1899         SSL_AES128GCM,
1900         SSL_AEAD,
1901         SSL_TLSV1_2,
1902         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1903         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
1904         128,
1905         128,
1906         },

1908         /* Cipher A1 */
1909         {
```



```
1910         0,
1911         TLS1_TXT_DH_RSA_WITH_AES_256_GCM_SHA384,
1912         TLS1_CK_DH_RSA_WITH_AES_256_GCM_SHA384,
1913         SSL_kDHr,
1914         SSL_aDH,
1915         SSL_AES256GCM,
1916         SSL_AEAD,
1917         SSL_TLSV1_2,
1918         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1919         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
1920         256,
1921         256,
1922         },

1924         /* Cipher A2 */
1925         {
1926         1,
1927         TLS1_TXT_DHE_DSS_WITH_AES_128_GCM_SHA256,
1928         TLS1_CK_DHE_DSS_WITH_AES_128_GCM_SHA256,
1929         SSL_kEDH,
1930         SSL_aDSS,
1931         SSL_AES128GCM,
1932         SSL_AEAD,
1933         SSL_TLSV1_2,
1934         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1935         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
1936         128,
1937         128,
1938         },

1940         /* Cipher A3 */
1941         {
1942         1,
1943         TLS1_TXT_DHE_DSS_WITH_AES_256_GCM_SHA384,
1944         TLS1_CK_DHE_DSS_WITH_AES_256_GCM_SHA384,
1945         SSL_kEDH,
1946         SSL_aDSS,
1947         SSL_AES256GCM,
1948         SSL_AEAD,
1949         SSL_TLSV1_2,
1950         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1951         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
1952         256,
1953         256,
1954         },

1956         /* Cipher A4 */
1957         {
1958         0,
1959         TLS1_TXT_DH_DSS_WITH_AES_128_GCM_SHA256,
1960         TLS1_CK_DH_DSS_WITH_AES_128_GCM_SHA256,
1961         SSL_kDHd,
1962         SSL_aDH,
1963         SSL_AES128GCM,
1964         SSL_AEAD,
1965         SSL_TLSV1_2,
1966         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1967         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
1968         128,
1969         128,
1970         },

1972         /* Cipher A5 */
1973         {
1974         0,
1975         TLS1_TXT_DH_DSS_WITH_AES_256_GCM_SHA384,
```

```
1976         TLS1_CK_DH_DSS_WITH_AES_256_GCM_SHA384,
1977         SSL_kDHd,
1978         SSL_aDH,
1979         SSL_AES256GCM,
1980         SSL_AEAD,
1981         SSL_TLSV1_2,
1982         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1983         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
1984         256,
1985         256,
1986         },

1988         /* Cipher A6 */
1989         {
1990         1,
1991         TLS1_TXT_ADH_WITH_AES_128_GCM_SHA256,
1992         TLS1_CK_ADH_WITH_AES_128_GCM_SHA256,
1993         SSL_kEDH,
1994         SSL_aNULL,
1995         SSL_AES128GCM,
1996         SSL_AEAD,
1997         SSL_TLSV1_2,
1998         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
1999         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2000         128,
2001         128,
2002         },

2004         /* Cipher A7 */
2005         {
2006         1,
2007         TLS1_TXT_ADH_WITH_AES_256_GCM_SHA384,
2008         TLS1_CK_ADH_WITH_AES_256_GCM_SHA384,
2009         SSL_kEDH,
2010         SSL_aNULL,
2011         SSL_AES256GCM,
2012         SSL_AEAD,
2013         SSL_TLSV1_2,
2014         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2015         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2016         256,
2017         256,
2018         },

2020 #ifndef OPENSSL_NO_ECDH
2021         /* Cipher C001 */
2022         {
2023         1,
2024         TLS1_TXT_ECDH_ECDSA_WITH_NULL_SHA,
2025         TLS1_CK_ECDH_ECDSA_WITH_NULL_SHA,
2026         SSL_kECDHe,
2027         SSL_aECDH,
2028         SSL_eNULL,
2029         SSL_SHA1,
2030         SSL_TLSV1,
2031         SSL_NOT_EXP|SSL_STRONG_NONE|SSL_FIPS,
2032         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2033         0,
2034         0,
2035         },

2037         /* Cipher C002 */
2038         {
2039         1,
2040         TLS1_TXT_ECDH_ECDSA_WITH_RC4_128_SHA,
2041         TLS1_CK_ECDH_ECDSA_WITH_RC4_128_SHA,
```



```
2042         SSL_kECDHe,
2043         SSL_aECDH,
2044         SSL_RC4,
2045         SSL_SHA1,
2046         SSL_TLSV1,
2047         SSL_NOT_EXP|SSL_MEDIUM,
2048         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2049         128,
2050         128,
2051         },

2053         /* Cipher C003 */
2054         {
2055         1,
2056         TLS1_TXT_ECDH_ECDSA_WITH_DES_192_CBC3_SHA,
2057         TLS1_CK_ECDH_ECDSA_WITH_DES_192_CBC3_SHA,
2058         SSL_kECDHe,
2059         SSL_aECDH,
2060         SSL_3DES,
2061         SSL_SHA1,
2062         SSL_TLSV1,
2063         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2064         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2065         112,
2066         168,
2067         },

2069         /* Cipher C004 */
2070         {
2071         1,
2072         TLS1_TXT_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
2073         TLS1_CK_ECDH_ECDSA_WITH_AES_128_CBC_SHA,
2074         SSL_kECDHe,
2075         SSL_aECDH,
2076         SSL_AES128,
2077         SSL_SHA1,
2078         SSL_TLSV1,
2079         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2080         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2081         128,
2082         128,
2083         },

2085         /* Cipher C005 */
2086         {
2087         1,
2088         TLS1_TXT_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
2089         TLS1_CK_ECDH_ECDSA_WITH_AES_256_CBC_SHA,
2090         SSL_kECDHe,
2091         SSL_aECDH,
2092         SSL_AES256,
2093         SSL_SHA1,
2094         SSL_TLSV1,
2095         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2096         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2097         256,
2098         256,
2099         },

2101         /* Cipher C006 */
2102         {
2103         1,
2104         TLS1_TXT_ECDHE_ECDSA_WITH_NULL_SHA,
2105         TLS1_CK_ECDHE_ECDSA_WITH_NULL_SHA,
2106         SSL_kEECDH,
2107         SSL_aECDSA,
```

```
2108         SSL_eNULL,
2109         SSL_SHA1,
2110         SSL_TLSV1,
2111         SSL_NOT_EXP|SSL_STRONG_NONE|SSL_FIPS,
2112         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2113         0,
2114         0,
2115         },

2117         /* Cipher C007 */
2118         {
2119         1,
2120         TLS1_TXT_ECDHE_ECDSA_WITH_RC4_128_SHA,
2121         TLS1_CK_ECDHE_ECDSA_WITH_RC4_128_SHA,
2122         SSL_kEECDH,
2123         SSL_aECDSA,
2124         SSL_RC4,
2125         SSL_SHA1,
2126         SSL_TLSV1,
2127         SSL_NOT_EXP|SSL_MEDIUM,
2128         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2129         128,
2130         128,
2131         },

2133         /* Cipher C008 */
2134         {
2135         1,
2136         TLS1_TXT_ECDHE_ECDSA_WITH_DES_192_CBC3_SHA,
2137         TLS1_CK_ECDHE_ECDSA_WITH_DES_192_CBC3_SHA,
2138         SSL_kEECDH,
2139         SSL_aECDSA,
2140         SSL_3DES,
2141         SSL_SHA1,
2142         SSL_TLSV1,
2143         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2144         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2145         112,
2146         168,
2147         },

2149         /* Cipher C009 */
2150         {
2151         1,
2152         TLS1_TXT_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
2153         TLS1_CK_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
2154         SSL_kEECDH,
2155         SSL_aECDSA,
2156         SSL_AES128,
2157         SSL_SHA1,
2158         SSL_TLSV1,
2159         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2160         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2161         128,
2162         128,
2163         },

2165         /* Cipher C00A */
2166         {
2167         1,
2168         TLS1_TXT_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
2169         TLS1_CK_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
2170         SSL_kEECDH,
2171         SSL_aECDSA,
2172         SSL_AES256,
2173         SSL_SHA1,
```



```
2174         SSL_TLSV1,
2175         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2176         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2177         256,
2178         256,
2179         },

2181         /* Cipher C00B */
2182         {
2183         1,
2184         TLS1_TXT_ECDH_RSA_WITH_NULL_SHA,
2185         TLS1_CK_ECDH_RSA_WITH_NULL_SHA,
2186         SSL_kECDHr,
2187         SSL_aECDH,
2188         SSL_eNULL,
2189         SSL_SHA1,
2190         SSL_TLSV1,
2191         SSL_NOT_EXP|SSL_STRONG_NONE|SSL_FIPS,
2192         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2193         0,
2194         0,
2195         },

2197         /* Cipher C00C */
2198         {
2199         1,
2200         TLS1_TXT_ECDH_RSA_WITH_RC4_128_SHA,
2201         TLS1_CK_ECDH_RSA_WITH_RC4_128_SHA,
2202         SSL_kECDHr,
2203         SSL_aECDH,
2204         SSL_RC4,
2205         SSL_SHA1,
2206         SSL_TLSV1,
2207         SSL_NOT_EXP|SSL_MEDIUM,
2208         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2209         128,
2210         128,
2211         },

2213         /* Cipher C00D */
2214         {
2215         1,
2216         TLS1_TXT_ECDH_RSA_WITH_DES_192_CBC3_SHA,
2217         TLS1_CK_ECDH_RSA_WITH_DES_192_CBC3_SHA,
2218         SSL_kECDHr,
2219         SSL_aECDH,
2220         SSL_3DES,
2221         SSL_SHA1,
2222         SSL_TLSV1,
2223         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2224         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2225         112,
2226         168,
2227         },

2229         /* Cipher C00E */
2230         {
2231         1,
2232         TLS1_TXT_ECDH_RSA_WITH_AES_128_CBC_SHA,
2233         TLS1_CK_ECDH_RSA_WITH_AES_128_CBC_SHA,
2234         SSL_kECDHr,
2235         SSL_aECDH,
2236         SSL_AES128,
2237         SSL_SHA1,
2238         SSL_TLSV1,
2239         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
```

```
2240         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2241         128,
2242         128,
2243         },

2245         /* Cipher C00F */
2246         {
2247         1,
2248         TLS1_TXT_ECDH_RSA_WITH_AES_256_CBC_SHA,
2249         TLS1_CK_ECDH_RSA_WITH_AES_256_CBC_SHA,
2250         SSL_kECDHr,
2251         SSL_aECDH,
2252         SSL_AES256,
2253         SSL_SHA1,
2254         SSL_TLSV1,
2255         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2256         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2257         256,
2258         256,
2259         },

2261         /* Cipher C010 */
2262         {
2263         1,
2264         TLS1_TXT_ECDHE_RSA_WITH_NULL_SHA,
2265         TLS1_CK_ECDHE_RSA_WITH_NULL_SHA,
2266         SSL_kEECDH,
2267         SSL_aRSA,
2268         SSL_eNULL,
2269         SSL_SHA1,
2270         SSL_TLSV1,
2271         SSL_NOT_EXP|SSL_STRONG_NONE|SSL_FIPS,
2272         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2273         0,
2274         0,
2275         },

2277         /* Cipher C011 */
2278         {
2279         1,
2280         TLS1_TXT_ECDHE_RSA_WITH_RC4_128_SHA,
2281         TLS1_CK_ECDHE_RSA_WITH_RC4_128_SHA,
2282         SSL_kEECDH,
2283         SSL_aRSA,
2284         SSL_RC4,
2285         SSL_SHA1,
2286         SSL_TLSV1,
2287         SSL_NOT_EXP|SSL_MEDIUM,
2288         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2289         128,
2290         128,
2291         },

2293         /* Cipher C012 */
2294         {
2295         1,
2296         TLS1_TXT_ECDHE_RSA_WITH_DES_192_CBC3_SHA,
2297         TLS1_CK_ECDHE_RSA_WITH_DES_192_CBC3_SHA,
2298         SSL_kEECDH,
2299         SSL_aRSA,
2300         SSL_3DES,
2301         SSL_SHA1,
2302         SSL_TLSV1,
2303         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2304         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2305         112,
```



```
2306         168,
2307         },

2309         /* Cipher C013 */
2310         {
2311         1,
2312         TLS1_TXT_ECDHE_RSA_WITH_AES_128_CBC_SHA,
2313         TLS1_CK_ECDHE_RSA_WITH_AES_128_CBC_SHA,
2314         SSL_kEECDH,
2315         SSL_aRSA,
2316         SSL_AES128,
2317         SSL_SHA1,
2318         SSL_TLSV1,
2319         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2320         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2321         128,
2322         128,
2323         },

2325         /* Cipher C014 */
2326         {
2327         1,
2328         TLS1_TXT_ECDHE_RSA_WITH_AES_256_CBC_SHA,
2329         TLS1_CK_ECDHE_RSA_WITH_AES_256_CBC_SHA,
2330         SSL_kEECDH,
2331         SSL_aRSA,
2332         SSL_AES256,
2333         SSL_SHA1,
2334         SSL_TLSV1,
2335         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2336         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2337         256,
2338         256,
2339         },

2341         /* Cipher C015 */
2342         {
2343         1,
2344         TLS1_TXT_ECDH_anon_WITH_NULL_SHA,
2345         TLS1_CK_ECDH_anon_WITH_NULL_SHA,
2346         SSL_kEECDH,
2347         SSL_aNULL,
2348         SSL_eNULL,
2349         SSL_SHA1,
2350         SSL_TLSV1,
2351         SSL_NOT_EXP|SSL_STRONG_NONE|SSL_FIPS,
2352         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2353         0,
2354         0,
2355         },

2357         /* Cipher C016 */
2358         {
2359         1,
2360         TLS1_TXT_ECDH_anon_WITH_RC4_128_SHA,
2361         TLS1_CK_ECDH_anon_WITH_RC4_128_SHA,
2362         SSL_kEECDH,
2363         SSL_aNULL,
2364         SSL_RC4,
2365         SSL_SHA1,
2366         SSL_TLSV1,
2367         SSL_NOT_EXP|SSL_MEDIUM,
2368         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2369         128,
2370         128,
2371         },
```

```
2373         /* Cipher C017 */
2374         {
2375         1,
2376         TLS1_TXT_ECDH_anon_WITH_DES_192_CBC3_SHA,
2377         TLS1_CK_ECDH_anon_WITH_DES_192_CBC3_SHA,
2378         SSL_kEECDH,
2379         SSL_aNULL,
2380         SSL_3DES,
2381         SSL_SHA1,
2382         SSL_TLSV1,
2383         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2384         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2385         112,
2386         168,
2387         },

2389         /* Cipher C018 */
2390         {
2391         1,
2392         TLS1_TXT_ECDH_anon_WITH_AES_128_CBC_SHA,
2393         TLS1_CK_ECDH_anon_WITH_AES_128_CBC_SHA,
2394         SSL_kEECDH,
2395         SSL_aNULL,
2396         SSL_AES128,
2397         SSL_SHA1,
2398         SSL_TLSV1,
2399         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2400         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2401         128,
2402         128,
2403         },

2405         /* Cipher C019 */
2406         {
2407         1,
2408         TLS1_TXT_ECDH_anon_WITH_AES_256_CBC_SHA,
2409         TLS1_CK_ECDH_anon_WITH_AES_256_CBC_SHA,
2410         SSL_kEECDH,
2411         SSL_aNULL,
2412         SSL_AES256,
2413         SSL_SHA1,
2414         SSL_TLSV1,
2415         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2416         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2417         256,
2418         256,
2419         },
2420 #endif  /* OPENSSL_NO_ECDH */

2422 #ifndef OPENSSL_NO_SRP
2423         /* Cipher C01A */
2424         {
2425         1,
2426         TLS1_TXT_SRP_SHA_WITH_3DES_EDE_CBC_SHA,
2427         TLS1_CK_SRP_SHA_WITH_3DES_EDE_CBC_SHA,
2428         SSL_kSRP,
2429         SSL_aSRP,
2430         SSL_3DES,
2431         SSL_SHA1,
2432         SSL_TLSV1,
2433         SSL_NOT_EXP|SSL_HIGH,
2434         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2435         112,
2436         168,
2437         },
```



```
2439         /* Cipher C01B */
2440         {
2441         1,
2442         TLS1_TXT_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA,
2443         TLS1_CK_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA,
2444         SSL_kSRP,
2445         SSL_aRSA,
2446         SSL_3DES,
2447         SSL_SHA1,
2448         SSL_TLSV1,
2449         SSL_NOT_EXP|SSL_HIGH,
2450         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2451         112,
2452         168,
2453         },

2455         /* Cipher C01C */
2456         {
2457         1,
2458         TLS1_TXT_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA,
2459         TLS1_CK_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA,
2460         SSL_kSRP,
2461         SSL_aDSS,
2462         SSL_3DES,
2463         SSL_SHA1,
2464         SSL_TLSV1,
2465         SSL_NOT_EXP|SSL_HIGH,
2466         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2467         112,
2468         168,
2469         },

2471         /* Cipher C01D */
2472         {
2473         1,
2474         TLS1_TXT_SRP_SHA_WITH_AES_128_CBC_SHA,
2475         TLS1_CK_SRP_SHA_WITH_AES_128_CBC_SHA,
2476         SSL_kSRP,
2477         SSL_aSRP,
2478         SSL_AES128,
2479         SSL_SHA1,
2480         SSL_TLSV1,
2481         SSL_NOT_EXP|SSL_HIGH,
2482         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2483         128,
2484         128,
2485         },

2487         /* Cipher C01E */
2488         {
2489         1,
2490         TLS1_TXT_SRP_SHA_RSA_WITH_AES_128_CBC_SHA,
2491         TLS1_CK_SRP_SHA_RSA_WITH_AES_128_CBC_SHA,
2492         SSL_kSRP,
2493         SSL_aRSA,
2494         SSL_AES128,
2495         SSL_SHA1,
2496         SSL_TLSV1,
2497         SSL_NOT_EXP|SSL_HIGH,
2498         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2499         128,
2500         128,
2501         },

2503         /* Cipher C01F */
```

```
2504         {
2505         1,
2506         TLS1_TXT_SRP_SHA_DSS_WITH_AES_128_CBC_SHA,
2507         TLS1_CK_SRP_SHA_DSS_WITH_AES_128_CBC_SHA,
2508         SSL_kSRP,
2509         SSL_aDSS,
2510         SSL_AES128,
2511         SSL_SHA1,
2512         SSL_TLSV1,
2513         SSL_NOT_EXP|SSL_HIGH,
2514         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2515         128,
2516         128,
2517         },

2519         /* Cipher C020 */
2520         {
2521         1,
2522         TLS1_TXT_SRP_SHA_WITH_AES_256_CBC_SHA,
2523         TLS1_CK_SRP_SHA_WITH_AES_256_CBC_SHA,
2524         SSL_kSRP,
2525         SSL_aSRP,
2526         SSL_AES256,
2527         SSL_SHA1,
2528         SSL_TLSV1,
2529         SSL_NOT_EXP|SSL_HIGH,
2530         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2531         256,
2532         256,
2533         },

2535         /* Cipher C021 */
2536         {
2537         1,
2538         TLS1_TXT_SRP_SHA_RSA_WITH_AES_256_CBC_SHA,
2539         TLS1_CK_SRP_SHA_RSA_WITH_AES_256_CBC_SHA,
2540         SSL_kSRP,
2541         SSL_aRSA,
2542         SSL_AES256,
2543         SSL_SHA1,
2544         SSL_TLSV1,
2545         SSL_NOT_EXP|SSL_HIGH,
2546         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2547         256,
2548         256,
2549         },

2551         /* Cipher C022 */
2552         {
2553         1,
2554         TLS1_TXT_SRP_SHA_DSS_WITH_AES_256_CBC_SHA,
2555         TLS1_CK_SRP_SHA_DSS_WITH_AES_256_CBC_SHA,
2556         SSL_kSRP,
2557         SSL_aDSS,
2558         SSL_AES256,
2559         SSL_SHA1,
2560         SSL_TLSV1,
2561         SSL_NOT_EXP|SSL_HIGH,
2562         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2563         256,
2564         256,
2565         },
2566 #endif  /* OPENSSL_NO_SRP */
2567 #ifndef OPENSSL_NO_ECDH

2569         /* HMAC based TLS v1.2 ciphersuites from RFC5289 */
```



```
2571         /* Cipher C023 */
2572         {
2573         1,
2574         TLS1_TXT_ECDHE_ECDSA_WITH_AES_128_SHA256,
2575         TLS1_CK_ECDHE_ECDSA_WITH_AES_128_SHA256,
2576         SSL_kEECDH,
2577         SSL_aECDSA,
2578         SSL_AES128,
2579         SSL_SHA256,
2580         SSL_TLSV1_2,
2581         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2582         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2583         128,
2584         128,
2585         },

2587         /* Cipher C024 */
2588         {
2589         1,
2590         TLS1_TXT_ECDHE_ECDSA_WITH_AES_256_SHA384,
2591         TLS1_CK_ECDHE_ECDSA_WITH_AES_256_SHA384,
2592         SSL_kEECDH,
2593         SSL_aECDSA,
2594         SSL_AES256,
2595         SSL_SHA384,
2596         SSL_TLSV1_2,
2597         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2598         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2599         256,
2600         256,
2601         },

2603         /* Cipher C025 */
2604         {
2605         1,
2606         TLS1_TXT_ECDH_ECDSA_WITH_AES_128_SHA256,
2607         TLS1_CK_ECDH_ECDSA_WITH_AES_128_SHA256,
2608         SSL_kECDHe,
2609         SSL_aECDH,
2610         SSL_AES128,
2611         SSL_SHA256,
2612         SSL_TLSV1_2,
2613         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2614         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2615         128,
2616         128,
2617         },

2619         /* Cipher C026 */
2620         {
2621         1,
2622         TLS1_TXT_ECDH_ECDSA_WITH_AES_256_SHA384,
2623         TLS1_CK_ECDH_ECDSA_WITH_AES_256_SHA384,
2624         SSL_kECDHe,
2625         SSL_aECDH,
2626         SSL_AES256,
2627         SSL_SHA384,
2628         SSL_TLSV1_2,
2629         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2630         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2631         256,
2632         256,
2633         },

2635         /* Cipher C027 */
```

```
2636        {
2637        1,
2638        TLS1_TXT_ECDHE_RSA_WITH_AES_128_SHA256,
2639        TLS1_CK_ECDHE_RSA_WITH_AES_128_SHA256,
2640        SSL_kEECDH,
2641        SSL_aRSA,
2642        SSL_AES128,
2643        SSL_SHA256,
2644        SSL_TLSV1_2,
2645        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2646        SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2647        128,
2648        128,
2649        },

2651        /* Cipher C028 */
2652        {
2653        1,
2654        TLS1_TXT_ECDHE_RSA_WITH_AES_256_SHA384,
2655        TLS1_CK_ECDHE_RSA_WITH_AES_256_SHA384,
2656        SSL_kEECDH,
2657        SSL_aRSA,
2658        SSL_AES256,
2659        SSL_SHA384,
2660        SSL_TLSV1_2,
2661        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2662        SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2663        256,
2664        256,
2665        },

2667        /* Cipher C029 */
2668        {
2669        1,
2670        TLS1_TXT_ECDH_RSA_WITH_AES_128_SHA256,
2671        TLS1_CK_ECDH_RSA_WITH_AES_128_SHA256,
2672        SSL_kECDHr,
2673        SSL_aECDH,
2674        SSL_AES128,
2675        SSL_SHA256,
2676        SSL_TLSV1_2,
2677        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2678        SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2679        128,
2680        128,
2681        },

2683        /* Cipher C02A */
2684        {
2685        1,
2686        TLS1_TXT_ECDH_RSA_WITH_AES_256_SHA384,
2687        TLS1_CK_ECDH_RSA_WITH_AES_256_SHA384,
2688        SSL_kECDHr,
2689        SSL_aECDH,
2690        SSL_AES256,
2691        SSL_SHA384,
2692        SSL_TLSV1_2,
2693        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2694        SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2695        256,
2696        256,
2697        },

2699        /* GCM based TLS v1.2 ciphersuites from RFC5289 */

2701        /* Cipher C02B */
```



```
2702        {
2703        1,
2704        TLS1_TXT_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
2705        TLS1_CK_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
2706        SSL_kEECDH,
2707        SSL_aECDSA,
2708        SSL_AES128GCM,
2709        SSL_AEAD,
2710        SSL_TLSV1_2,
2711        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2712        SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2713        128,
2714        128,
2715        },

2717        /* Cipher C02C */
2718        {
2719        1,
2720        TLS1_TXT_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
2721        TLS1_CK_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
2722        SSL_kEECDH,
2723        SSL_aECDSA,
2724        SSL_AES256GCM,
2725        SSL_AEAD,
2726        SSL_TLSV1_2,
2727        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2728        SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2729        256,
2730        256,
2731        },

2733        /* Cipher C02D */
2734        {
2735        1,
2736        TLS1_TXT_ECDH_ECDSA_WITH_AES_128_GCM_SHA256,
2737        TLS1_CK_ECDH_ECDSA_WITH_AES_128_GCM_SHA256,
2738        SSL_kECDHe,
2739        SSL_aECDH,
2740        SSL_AES128GCM,
2741        SSL_AEAD,
2742        SSL_TLSV1_2,
2743        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2744        SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2745        128,
2746        128,
2747        },

2749        /* Cipher C02E */
2750        {
2751        1,
2752        TLS1_TXT_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,
2753        TLS1_CK_ECDH_ECDSA_WITH_AES_256_GCM_SHA384,
2754        SSL_kECDHe,
2755        SSL_aECDH,
2756        SSL_AES256GCM,
2757        SSL_AEAD,
2758        SSL_TLSV1_2,
2759        SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2760        SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2761        256,
2762        256,
2763        },

2765        /* Cipher C02F */
2766        {
2767        1,
```

```
2768         TLS1_TXT_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
2769         TLS1_CK_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
2770         SSL_kEECDH,
2771         SSL_aRSA,
2772         SSL_AES128GCM,
2773         SSL_AEAD,
2774         SSL_TLSV1_2,
2775         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2776         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2777         128,
2778         128,
2779         },

2781         /* Cipher C030 */
2782         {
2783         1,
2784         TLS1_TXT_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
2785         TLS1_CK_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
2786         SSL_kEECDH,
2787         SSL_aRSA,
2788         SSL_AES256GCM,
2789         SSL_AEAD,
2790         SSL_TLSV1_2,
2791         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2792         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2793         256,
2794         256,
2795         },

2797         /* Cipher C031 */
2798         {
2799         1,
2800         TLS1_TXT_ECDH_RSA_WITH_AES_128_GCM_SHA256,
2801         TLS1_CK_ECDH_RSA_WITH_AES_128_GCM_SHA256,
2802         SSL_kECDHr,
2803         SSL_aECDH,
2804         SSL_AES128GCM,
2805         SSL_AEAD,
2806         SSL_TLSV1_2,
2807         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2808         SSL_HANDSHAKE_MAC_SHA256|TLS1_PRF_SHA256,
2809         128,
2810         128,
2811         },

2813         /* Cipher C032 */
2814         {
2815         1,
2816         TLS1_TXT_ECDH_RSA_WITH_AES_256_GCM_SHA384,
2817         TLS1_CK_ECDH_RSA_WITH_AES_256_GCM_SHA384,
2818         SSL_kECDHr,
2819         SSL_aECDH,
2820         SSL_AES256GCM,
2821         SSL_AEAD,
2822         SSL_TLSV1_2,
2823         SSL_NOT_EXP|SSL_HIGH|SSL_FIPS,
2824         SSL_HANDSHAKE_MAC_SHA384|TLS1_PRF_SHA384,
2825         256,
2826         256,
2827         },

2829 #endif /* OPENSSL_NO_ECDH */


2832 #ifdef TEMP_GOST_TLS
2833 /* Cipher FF00 */
```



```
2834         {
2835         1,
2836         "GOST-MD5",
2837         0x0300ff00,
2838         SSL_kRSA,
2839         SSL_aRSA,
2840         SSL_eGOST2814789CNT,
2841         SSL_MD5,
2842         SSL_TLSV1,
2843         SSL_NOT_EXP|SSL_HIGH,
2844         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2845         256,
2846         256,
2847         },
2848         {
2849         1,
2850         "GOST-GOST94",
2851         0x0300ff01,
2852         SSL_kRSA,
2853         SSL_aRSA,
2854         SSL_eGOST2814789CNT,
2855         SSL_GOST94,
2856         SSL_TLSV1,
2857         SSL_NOT_EXP|SSL_HIGH,
2858         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2859         256,
2860         256
2861         },
2862         {
2863         1,
2864         "GOST-GOST89MAC",
2865         0x0300ff02,
2866         SSL_kRSA,
2867         SSL_aRSA,
2868         SSL_eGOST2814789CNT,
2869         SSL_GOST89MAC,
2870         SSL_TLSV1,
2871         SSL_NOT_EXP|SSL_HIGH,
2872         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF,
2873         256,
2874         256
2875         },
2876         {
2877         1,
2878         "GOST-GOST89STREAM",
2879         0x0300ff03,
2880         SSL_kRSA,
2881         SSL_aRSA,
2882         SSL_eGOST2814789CNT,
2883         SSL_GOST89MAC,
2884         SSL_TLSV1,
2885         SSL_NOT_EXP|SSL_HIGH,
2886         SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF|TLS1_STREAM_MAC,
2887         256,
2888         256
2889         },
2890 #endif

2892 /* end of list */
2893         };

2895 SSL3_ENC_METHOD SSLv3_enc_data={
2896         ssl3_enc,
2897         n_ssl3_mac,
2898         ssl3_setup_key_block,
2899         ssl3_generate_master_secret,
```

```
2900         ssl3_change_cipher_state,
2901         ssl3_final_finish_mac,
2902         MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH,
2903         ssl3_cert_verify_mac,
2904         SSL3_MD_CLIENT_FINISHED_CONST,4,
2905         SSL3_MD_SERVER_FINISHED_CONST,4,
2906         ssl3_alert_code,
2907         (int (*)(SSL *, unsigned char *, size_t, const char *,
2908                 size_t, const unsigned char *, size_t,
2909                 int use_context))ssl_undefined_function,
2910         };

2912 long ssl3_default_timeout(void)
2913         {
2914         /* 2 hours, the 24 hours mentioned in the SSLv3 spec
2915          * is way too long for http, the cache would over fill */
2916         return(60*60*2);
2917         }

2919 int ssl3_num_ciphers(void)
2920         {
2921         return(SSL3_NUM_CIPHERS);
2922         }

2924 const SSL_CIPHER *ssl3_get_cipher(unsigned int u)
2925         {
2926         if (u < SSL3_NUM_CIPHERS)
2927                 return(&(ssl3_ciphers[SSL3_NUM_CIPHERS-1-u]));
2928         else
2929                 return(NULL);
2930         }

2932 int ssl3_pending(const SSL *s)
2933         {
2934         if (s->rstate == SSL_ST_READ_BODY)
2935                 return 0;

2937         return (s->s3->rrec.type == SSL3_RT_APPLICATION_DATA) ? s->s3->rrec.leng
2938         }

2940 int ssl3_new(SSL *s)
2941         {
2942         SSL3_STATE *s3;

2944         if ((s3=OPENSSL_malloc(sizeof *s3)) == NULL) goto err;
2945         memset(s3,0,sizeof *s3);
2946         memset(s3->rrec.seq_num,0,sizeof(s3->rrec.seq_num));
2947         memset(s3->wrec.seq_num,0,sizeof(s3->wrec.seq_num));

2949         s->s3=s3;

2951 #ifndef OPENSSL_NO_SRP
2952         SSL_SRP_CTX_init(s);
2953 #endif
2954         s->method->ssl_clear(s);
2955         return(1);
2956 err:
2957         return(0);
2958         }

2960 void ssl3_free(SSL *s)
2961         {
2962         if(s == NULL)
2963             return;

2965 #ifdef TLSEXT_TYPE_opaque_prf_input
```



```
2966         if (s->s3->client_opaque_prf_input != NULL)
2967                 OPENSSL_free(s->s3->client_opaque_prf_input);
2968         if (s->s3->server_opaque_prf_input != NULL)
2969                 OPENSSL_free(s->s3->server_opaque_prf_input);
2970 #endif

2972         ssl3_cleanup_key_block(s);
2973         if (s->s3->rbuf.buf != NULL)
2974                 ssl3_release_read_buffer(s);
2975         if (s->s3->wbuf.buf != NULL)
2976                 ssl3_release_write_buffer(s);
2977         if (s->s3->rrec.comp != NULL)
2978                 OPENSSL_free(s->s3->rrec.comp);
2979 #ifndef OPENSSL_NO_DH
2980         if (s->s3->tmp.dh != NULL)
2981                 DH_free(s->s3->tmp.dh);
2982 #endif
2983 #ifndef OPENSSL_NO_ECDH
2984         if (s->s3->tmp.ecdh != NULL)
2985                 EC_KEY_free(s->s3->tmp.ecdh);
2986 #endif

2988         if (s->s3->tmp.ca_names != NULL)
2989                 sk_X509_NAME_pop_free(s->s3->tmp.ca_names,X509_NAME_free);
2990         if (s->s3->handshake_buffer) {
2991                 BIO_free(s->s3->handshake_buffer);
2992         }
2993         if (s->s3->handshake_dgst) ssl3_free_digest_list(s);
2994 #ifndef OPENSSL_NO_SRP
2995         SSL_SRP_CTX_free(s);
2996 #endif
2997         OPENSSL_cleanse(s->s3,sizeof *s->s3);
2998         OPENSSL_free(s->s3);
2999         s->s3=NULL;
3000         }

3002 void ssl3_clear(SSL *s)
3003         {
3004         unsigned char *rp,*wp;
3005         size_t rlen, wlen;
3006         int init_extra;

3008 #ifdef TLSEXT_TYPE_opaque_prf_input
3009         if (s->s3->client_opaque_prf_input != NULL)
3010                 OPENSSL_free(s->s3->client_opaque_prf_input);
3011         s->s3->client_opaque_prf_input = NULL;
3012         if (s->s3->server_opaque_prf_input != NULL)
3013                 OPENSSL_free(s->s3->server_opaque_prf_input);
3014         s->s3->server_opaque_prf_input = NULL;
3015 #endif

3017         ssl3_cleanup_key_block(s);
3018         if (s->s3->tmp.ca_names != NULL)
3019                 sk_X509_NAME_pop_free(s->s3->tmp.ca_names,X509_NAME_free);

3021         if (s->s3->rrec.comp != NULL)
3022                 {
3023                 OPENSSL_free(s->s3->rrec.comp);
3024                 s->s3->rrec.comp=NULL;
3025                 }
3026 #ifndef OPENSSL_NO_DH
3027         if (s->s3->tmp.dh != NULL)
3028                 {
3029                 DH_free(s->s3->tmp.dh);
3030                 s->s3->tmp.dh = NULL;
3031                 }
```

```
#endif
#ifndef OPENSSL_NO_ECDH
        if (s->s3->tmp.ecdh != NULL)
                {
                EC_KEY_free(s->s3->tmp.ecdh);
                s->s3->tmp.ecdh = NULL;
                }
#endif
#ifndef OPENSSL_NO_TLSEXT
#ifndef OPENSSL_NO_EC
        s->s3->is_probably_safari = 0;
#endif /* !OPENSSL_NO_EC */
#endif /* !OPENSSL_NO_TLSEXT */

        rp = s->s3->rbuf.buf;
        wp = s->s3->wbuf.buf;
        rlen = s->s3->rbuf.len;
        wlen = s->s3->wbuf.len;
        init_extra = s->s3->init_extra;
        if (s->s3->handshake_buffer) {
                BIO_free(s->s3->handshake_buffer);
                s->s3->handshake_buffer = NULL;
        }
        if (s->s3->handshake_dgst) {
                ssl3_free_digest_list(s);
        }
        memset(s->s3,0,sizeof *s->s3);
        s->s3->rbuf.buf = rp;
        s->s3->wbuf.buf = wp;
        s->s3->rbuf.len = rlen;
        s->s3->wbuf.len = wlen;
        s->s3->init_extra = init_extra;

        ssl_free_wbio_buffer(s);

        s->packet_length=0;
        s->s3->renegotiate=0;
        s->s3->total_renegotiations=0;
        s->s3->num_renegotiations=0;
        s->s3->in_read_app_data=0;
        s->version=SSL3_VERSION;

#if !defined(OPENSSL_NO_TLSEXT) && !defined(OPENSSL_NO_NEXTPROTONEG)
        if (s->next_proto_negotiated)
                {
                OPENSSL_free(s->next_proto_negotiated);
                s->next_proto_negotiated = NULL;
                s->next_proto_negotiated_len = 0;
                }
#endif
        }

#ifndef OPENSSL_NO_SRP
static char * MS_CALLBACK srp_password_from_info_cb(SSL *s, void *arg)
        {
        return BUF_strdup(s->srp_ctx.info) ;
        }
#endif

long ssl3_ctrl(SSL *s, int cmd, long larg, void *parg)
        {
        int ret=0;

#if !defined(OPENSSL_NO_DSA) || !defined(OPENSSL_NO_RSA)
        if (
#ifndef OPENSSL_NO_RSA
```



```
            cmd == SSL_CTRL_SET_TMP_RSA ||
            cmd == SSL_CTRL_SET_TMP_RSA_CB ||
#endif
#ifndef OPENSSL_NO_DSA
            cmd == SSL_CTRL_SET_TMP_DH ||
            cmd == SSL_CTRL_SET_TMP_DH_CB ||
#endif
                0)
                {
                if (!ssl_cert_inst(&s->cert))
                        {
                        SSLerr(SSL_F_SSL3_CTRL, ERR_R_MALLOC_FAILURE);
                        return(0);
                        }
                }
#endif

        switch (cmd)
                {
        case SSL_CTRL_GET_SESSION_REUSED:
                ret=s->hit;
                break;
        case SSL_CTRL_GET_CLIENT_CERT_REQUEST:
                break;
        case SSL_CTRL_GET_NUM_RENEGOTIATIONS:
                ret=s->s3->num_renegotiations;
                break;
        case SSL_CTRL_CLEAR_NUM_RENEGOTIATIONS:
                ret=s->s3->num_renegotiations;
                s->s3->num_renegotiations=0;
                break;
        case SSL_CTRL_GET_TOTAL_RENEGOTIATIONS:
                ret=s->s3->total_renegotiations;
                break;
        case SSL_CTRL_GET_FLAGS:
                ret=(int)(s->s3->flags);
                break;
#ifndef OPENSSL_NO_RSA
        case SSL_CTRL_NEED_TMP_RSA:
                if ((s->cert != NULL) && (s->cert->rsa_tmp == NULL) &&
                    ((s->cert->pkeys[SSL_PKEY_RSA_ENC].privatekey == NULL) ||
                     (EVP_PKEY_size(s->cert->pkeys[SSL_PKEY_RSA_ENC].privatekey)
                        ret = 1;
                break;
        case SSL_CTRL_SET_TMP_RSA:
                {
                        RSA *rsa = (RSA *)parg;
                        if (rsa == NULL)
                                {
                                SSLerr(SSL_F_SSL3_CTRL, ERR_R_PASSED_NULL_PARAME
                                return(ret);
                                }
                        if ((rsa = RSAPrivateKey_dup(rsa)) == NULL)
                                {
                                SSLerr(SSL_F_SSL3_CTRL, ERR_R_RSA_LIB);
                                return(ret);
                                }
                        if (s->cert->rsa_tmp != NULL)
                                RSA_free(s->cert->rsa_tmp);
                        s->cert->rsa_tmp = rsa;
                        ret = 1;
                }
                break;
        case SSL_CTRL_SET_TMP_RSA_CB:
                {
                SSLerr(SSL_F_SSL3_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
```

```
3164                 return(ret);
3165                 }
3166                 break;
3167 #endif
3168 #ifndef OPENSSL_NO_DH
3169         case SSL_CTRL_SET_TMP_DH:
3170                 {
3171                         DH *dh = (DH *)parg;
3172                         if (dh == NULL)
3173                                 {
3174                                 SSLerr(SSL_F_SSL3_CTRL, ERR_R_PASSED_NULL_PARAME
3175                                 return(ret);
3176                                 }
3177                         if ((dh = DHparams_dup(dh)) == NULL)
3178                                 {
3179                                 SSLerr(SSL_F_SSL3_CTRL, ERR_R_DH_LIB);
3180                                 return(ret);
3181                                 }
3182                         if (!(s->options & SSL_OP_SINGLE_DH_USE))
3183                                 {
3184                                 if (!DH_generate_key(dh))
3185                                         {
3186                                         DH_free(dh);
3187                                         SSLerr(SSL_F_SSL3_CTRL, ERR_R_DH_LIB);
3188                                         return(ret);
3189                                         }
3190                                 }
3191                         if (s->cert->dh_tmp != NULL)
3192                                 DH_free(s->cert->dh_tmp);
3193                         s->cert->dh_tmp = dh;
3194                         ret = 1;
3195                 }
3196                 break;
3197         case SSL_CTRL_SET_TMP_DH_CB:
3198                 {
3199                 SSLerr(SSL_F_SSL3_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
3200                 return(ret);
3201                 }
3202                 break;
3203 #endif
3204 #ifndef OPENSSL_NO_ECDH
3205         case SSL_CTRL_SET_TMP_ECDH:
3206                 {
3207                 EC_KEY *ecdh = NULL;

3209                 if (parg == NULL)
3210                         {
3211                         SSLerr(SSL_F_SSL3_CTRL, ERR_R_PASSED_NULL_PARAMETER);
3212                         return(ret);
3213                         }
3214                 if (!EC_KEY_up_ref((EC_KEY *)parg))
3215                         {
3216                         SSLerr(SSL_F_SSL3_CTRL,ERR_R_ECDH_LIB);
3217                         return(ret);
3218                         }
3219                 ecdh = (EC_KEY *)parg;
3220                 if (!(s->options & SSL_OP_SINGLE_ECDH_USE))
3221                         {
3222                         if (!EC_KEY_generate_key(ecdh))
3223                                 {
3224                                 EC_KEY_free(ecdh);
3225                                 SSLerr(SSL_F_SSL3_CTRL,ERR_R_ECDH_LIB);
3226                                 return(ret);
3227                                 }
3228                         }
3229                 if (s->cert->ecdh_tmp != NULL)
```



```
3230                         EC_KEY_free(s->cert->ecdh_tmp);
3231                 s->cert->ecdh_tmp = ecdh;
3232                 ret = 1;
3233                 }
3234                 break;
3235         case SSL_CTRL_SET_TMP_ECDH_CB:
3236                 {
3237                 SSLerr(SSL_F_SSL3_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
3238                 return(ret);
3239                 }
3240                 break;
3241 #endif /* !OPENSSL_NO_ECDH */
3242 #ifndef OPENSSL_NO_TLSEXT
3243         case SSL_CTRL_SET_TLSEXT_HOSTNAME:
3244                 if (larg == TLSEXT_NAMETYPE_host_name)
3245                         {
3246                         if (s->tlsext_hostname != NULL)
3247                                 OPENSSL_free(s->tlsext_hostname);
3248                         s->tlsext_hostname = NULL;

3250                         ret = 1;
3251                         if (parg == NULL)
3252                                 break;
3253                         if (strlen((char *)parg) > TLSEXT_MAXLEN_host_name)
3254                                 {
3255                                 SSLerr(SSL_F_SSL3_CTRL, SSL_R_SSL3_EXT_INVALID_S
3256                                 return 0;
3257                                 }
3258                         if ((s->tlsext_hostname = BUF_strdup((char *)parg)) == N
3259                                 {
3260                                 SSLerr(SSL_F_SSL3_CTRL, ERR_R_INTERNAL_ERROR);
3261                                 return 0;
3262                                 }
3263                         }
3264                 else
3265                         {
3266                         SSLerr(SSL_F_SSL3_CTRL, SSL_R_SSL3_EXT_INVALID_SERVERNAM
3267                         return 0;
3268                         }
3269                 break;
3270         case SSL_CTRL_SET_TLSEXT_DEBUG_ARG:
3271                 s->tlsext_debug_arg=parg;
3272                 ret = 1;
3273                 break;

3275 #ifdef TLSEXT_TYPE_opaque_prf_input
3276         case SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT:
3277                 if (larg > 12288) /* actual internal limit is 2^16 for the compl
3278                                      * (including the cert chain and everything) *
3279                         {
3280                         SSLerr(SSL_F_SSL3_CTRL, SSL_R_OPAQUE_PRF_INPUT_TOO_LONG)
3281                         break;
3282                         }
3283                 if (s->tlsext_opaque_prf_input != NULL)
3284                         OPENSSL_free(s->tlsext_opaque_prf_input);
3285                 if ((size_t)larg == 0)
3286                         s->tlsext_opaque_prf_input = OPENSSL_malloc(1); /* dummy
3287                 else
3288                         s->tlsext_opaque_prf_input = BUF_memdup(parg, (size_t)la
3289                 if (s->tlsext_opaque_prf_input != NULL)
3290                         {
3291                         s->tlsext_opaque_prf_input_len = (size_t)larg;
3292                         ret = 1;
3293                         }
3294                 else
3295                         s->tlsext_opaque_prf_input_len = 0;
```

```
3296                 break;
3297 #endif

3299         case SSL_CTRL_SET_TLSEXT_STATUS_REQ_TYPE:
3300                 s->tlsext_status_type=larg;
3301                 ret = 1;
3302                 break;

3304         case SSL_CTRL_GET_TLSEXT_STATUS_REQ_EXTS:
3305                 *(STACK_OF(X509_EXTENSION) **)parg = s->tlsext_ocsp_exts;
3306                 ret = 1;
3307                 break;

3309         case SSL_CTRL_SET_TLSEXT_STATUS_REQ_EXTS:
3310                 s->tlsext_ocsp_exts = parg;
3311                 ret = 1;
3312                 break;

3314         case SSL_CTRL_GET_TLSEXT_STATUS_REQ_IDS:
3315                 *(STACK_OF(OCSP_RESPID) **)parg = s->tlsext_ocsp_ids;
3316                 ret = 1;
3317                 break;

3319         case SSL_CTRL_SET_TLSEXT_STATUS_REQ_IDS:
3320                 s->tlsext_ocsp_ids = parg;
3321                 ret = 1;
3322                 break;

3324         case SSL_CTRL_GET_TLSEXT_STATUS_REQ_OCSP_RESP:
3325                 *(unsigned char **)parg = s->tlsext_ocsp_resp;
3326                 return s->tlsext_ocsp_resplen;

3328         case SSL_CTRL_SET_TLSEXT_STATUS_REQ_OCSP_RESP:
3329                 if (s->tlsext_ocsp_resp)
3330                         OPENSSL_free(s->tlsext_ocsp_resp);
3331                 s->tlsext_ocsp_resp = parg;
3332                 s->tlsext_ocsp_resplen = larg;
3333                 ret = 1;
3334                 break;

3336 #ifndef OPENSSL_NO_HEARTBEATS
3337         case SSL_CTRL_TLS_EXT_SEND_HEARTBEAT:
3338                 if (SSL_version(s) == DTLS1_VERSION || SSL_version(s) == DTLS1_B
3339                         ret = dtls1_heartbeat(s);
3340                 else
3341                         ret = tls1_heartbeat(s);
3342                 break;

3344         case SSL_CTRL_GET_TLS_EXT_HEARTBEAT_PENDING:
3345                 ret = s->tlsext_hb_pending;
3346                 break;

3348         case SSL_CTRL_SET_TLS_EXT_HEARTBEAT_NO_REQUESTS:
3349                 if (larg)
3350                         s->tlsext_heartbeat |= SSL_TLSEXT_HB_DONT_RECV_REQUESTS;
3351                 else
3352                         s->tlsext_heartbeat &= ~SSL_TLSEXT_HB_DONT_RECV_REQUESTS
3353                 ret = 1;
3354                 break;
3355 #endif

3357 #endif /* !OPENSSL_NO_TLSEXT */
3358         default:
3359                 break;
3360                 }
3361         return(ret);
```



```
3362         }

3364 long ssl3_callback_ctrl(SSL *s, int cmd, void (*fp)(void))
3365         {
3366         int ret=0;

3368 #if !defined(OPENSSL_NO_DSA) || !defined(OPENSSL_NO_RSA)
3369         if (
3370 #ifndef OPENSSL_NO_RSA
3371             cmd == SSL_CTRL_SET_TMP_RSA_CB ||
3372 #endif
3373 #ifndef OPENSSL_NO_DSA
3374             cmd == SSL_CTRL_SET_TMP_DH_CB ||
3375 #endif
3376                 0)
3377                 {
3378                 if (!ssl_cert_inst(&s->cert))
3379                         {
3380                         SSLerr(SSL_F_SSL3_CALLBACK_CTRL, ERR_R_MALLOC_FAILURE);
3381                         return(0);
3382                         }
3383                 }
3384 #endif

3386         switch (cmd)
3387                 {
3388 #ifndef OPENSSL_NO_RSA
3389         case SSL_CTRL_SET_TMP_RSA_CB:
3390                 {
3391                 s->cert->rsa_tmp_cb = (RSA *(*)(SSL *, int, int))fp;
3392                 }
3393                 break;
3394 #endif
3395 #ifndef OPENSSL_NO_DH
3396         case SSL_CTRL_SET_TMP_DH_CB:
3397                 {
3398                 s->cert->dh_tmp_cb = (DH *(*)(SSL *, int, int))fp;
3399                 }
3400                 break;
3401 #endif
3402 #ifndef OPENSSL_NO_ECDH
3403         case SSL_CTRL_SET_TMP_ECDH_CB:
3404                 {
3405                 s->cert->ecdh_tmp_cb = (EC_KEY *(*)(SSL *, int, int))fp;
3406                 }
3407                 break;
3408 #endif
3409 #ifndef OPENSSL_NO_TLSEXT
3410         case SSL_CTRL_SET_TLSEXT_DEBUG_CB:
3411                 s->tlsext_debug_cb=(void (*)(SSL *,int ,int,
3412                                         unsigned char *, int, void *))fp;
3413                 break;
3414 #endif
3415         default:
3416                 break;
3417                 }
3418         return(ret);
3419         }

3421 long ssl3_ctx_ctrl(SSL_CTX *ctx, int cmd, long larg, void *parg)
3422         {
3423         CERT *cert;

3425         cert=ctx->cert;

3427         switch (cmd)
```

```
3428                 {
3429 #ifndef OPENSSL_NO_RSA
3430         case SSL_CTRL_NEED_TMP_RSA:
3431                 if (    (cert->rsa_tmp == NULL) &&
3432                         ((cert->pkeys[SSL_PKEY_RSA_ENC].privatekey == NULL) ||
3433                          (EVP_PKEY_size(cert->pkeys[SSL_PKEY_RSA_ENC].privatekey
3434                         )
3435                         return(1);
3436                 else
3437                         return(0);
3438                 /* break; */
3439         case SSL_CTRL_SET_TMP_RSA:
3440                 {
3441                 RSA *rsa;
3442                 int i;

3444                 rsa=(RSA *)parg;
3445                 i=1;
3446                 if (rsa == NULL)
3447                         i=0;
3448                 else
3449                         {
3450                         if ((rsa=RSAPrivateKey_dup(rsa)) == NULL)
3451                                 i=0;
3452                         }
3453                 if (!i)
3454                         {
3455                         SSLerr(SSL_F_SSL3_CTX_CTRL,ERR_R_RSA_LIB);
3456                         return(0);
3457                         }
3458                 else
3459                         {
3460                         if (cert->rsa_tmp != NULL)
3461                                 RSA_free(cert->rsa_tmp);
3462                         cert->rsa_tmp=rsa;
3463                         return(1);
3464                         }
3465                 }
3466                 /* break; */
3467         case SSL_CTRL_SET_TMP_RSA_CB:
3468                 {
3469                 SSLerr(SSL_F_SSL3_CTX_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
3470                 return(0);
3471                 }
3472                 break;
3473 #endif
3474 #ifndef OPENSSL_NO_DH
3475         case SSL_CTRL_SET_TMP_DH:
3476                 {
3477                 DH *new=NULL,*dh;

3479                 dh=(DH *)parg;
3480                 if ((new=DHparams_dup(dh)) == NULL)
3481                         {
3482                         SSLerr(SSL_F_SSL3_CTX_CTRL,ERR_R_DH_LIB);
3483                         return 0;
3484                         }
3485                 if (!(ctx->options & SSL_OP_SINGLE_DH_USE))
3486                         {
3487                         if (!DH_generate_key(new))
3488                                 {
3489                                 SSLerr(SSL_F_SSL3_CTX_CTRL,ERR_R_DH_LIB);
3490                                 DH_free(new);
3491                                 return 0;
3492                                 }
3493                         }
```



```
3494                 if (cert->dh_tmp != NULL)
3495                         DH_free(cert->dh_tmp);
3496                 cert->dh_tmp=new;
3497                 return 1;
3498                 }
3499                 /*break; */
3500         case SSL_CTRL_SET_TMP_DH_CB:
3501                 {
3502                 SSLerr(SSL_F_SSL3_CTX_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
3503                 return(0);
3504                 }
3505                 break;
3506 #endif
3507 #ifndef OPENSSL_NO_ECDH
3508         case SSL_CTRL_SET_TMP_ECDH:
3509                 {
3510                 EC_KEY *ecdh = NULL;

3512                 if (parg == NULL)
3513                         {
3514                         SSLerr(SSL_F_SSL3_CTX_CTRL,ERR_R_ECDH_LIB);
3515                         return 0;
3516                         }
3517                 ecdh = EC_KEY_dup((EC_KEY *)parg);
3518                 if (ecdh == NULL)
3519                         {
3520                         SSLerr(SSL_F_SSL3_CTX_CTRL,ERR_R_EC_LIB);
3521                         return 0;
3522                         }
3523                 if (!(ctx->options & SSL_OP_SINGLE_ECDH_USE))
3524                         {
3525                         if (!EC_KEY_generate_key(ecdh))
3526                                 {
3527                                 EC_KEY_free(ecdh);
3528                                 SSLerr(SSL_F_SSL3_CTX_CTRL,ERR_R_ECDH_LIB);
3529                                 return 0;
3530                                 }
3531                         }

3533                 if (cert->ecdh_tmp != NULL)
3534                         {
3535                         EC_KEY_free(cert->ecdh_tmp);
3536                         }
3537                 cert->ecdh_tmp = ecdh;
3538                 return 1;
3539                 }
3540                 /* break; */
3541         case SSL_CTRL_SET_TMP_ECDH_CB:
3542                 {
3543                 SSLerr(SSL_F_SSL3_CTX_CTRL, ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
3544                 return(0);
3545                 }
3546                 break;
3547 #endif /* !OPENSSL_NO_ECDH */
3548 #ifndef OPENSSL_NO_TLSEXT
3549         case SSL_CTRL_SET_TLSEXT_SERVERNAME_ARG:
3550                 ctx->tlsext_servername_arg=parg;
3551                 break;
3552         case SSL_CTRL_SET_TLSEXT_TICKET_KEYS:
3553         case SSL_CTRL_GET_TLSEXT_TICKET_KEYS:
3554                 {
3555                 unsigned char *keys = parg;
3556                 if (!keys)
3557                         return 48;
3558                 if (larg != 48)
3559                         {
```

```
3560                        SSLerr(SSL_F_SSL3_CTX_CTRL, SSL_R_INVALID_TICKET_KEYS_LE
3561                        return 0;
3562                        }
3563                if (cmd == SSL_CTRL_SET_TLSEXT_TICKET_KEYS)
3564                        {
3565                        memcpy(ctx->tlsext_tick_key_name, keys, 16);
3566                        memcpy(ctx->tlsext_tick_hmac_key, keys + 16, 16);
3567                        memcpy(ctx->tlsext_tick_aes_key, keys + 32, 16);
3568                        }
3569                else
3570                        {
3571                        memcpy(keys, ctx->tlsext_tick_key_name, 16);
3572                        memcpy(keys + 16, ctx->tlsext_tick_hmac_key, 16);
3573                        memcpy(keys + 32, ctx->tlsext_tick_aes_key, 16);
3574                        }
3575                return 1;
3576                }

3578 #ifdef TLSEXT_TYPE_opaque_prf_input
3579        case SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT_CB_ARG:
3580                ctx->tlsext_opaque_prf_input_callback_arg = parg;
3581                return 1;
3582 #endif

3584        case SSL_CTRL_SET_TLSEXT_STATUS_REQ_CB_ARG:
3585                ctx->tlsext_status_arg=parg;
3586                return 1;
3587                break;

3589 #ifndef OPENSSL_NO_SRP
3590        case SSL_CTRL_SET_TLS_EXT_SRP_USERNAME:
3591                ctx->srp_ctx.srp_Mask|=SSL_kSRP;
3592                if (ctx->srp_ctx.login != NULL)
3593                        OPENSSL_free(ctx->srp_ctx.login);
3594                ctx->srp_ctx.login = NULL;
3595                if (parg == NULL)
3596                        break;
3597                if (strlen((const char *)parg) > 255 || strlen((const char *)par
3598                        {
3599                        SSLerr(SSL_F_SSL3_CTX_CTRL, SSL_R_INVALID_SRP_USERNAME);
3600                        return 0;
3601                        }
3602                if ((ctx->srp_ctx.login = BUF_strdup((char *)parg)) == NULL)
3603                        {
3604                        SSLerr(SSL_F_SSL3_CTX_CTRL, ERR_R_INTERNAL_ERROR);
3605                        return 0;
3606                        }
3607                break;
3608        case SSL_CTRL_SET_TLS_EXT_SRP_PASSWORD:
3609                ctx->srp_ctx.SRP_give_srp_client_pwd_callback=srp_password_from_
3610                ctx->srp_ctx.info=parg;
3611                break;
3612        case SSL_CTRL_SET_SRP_ARG:
3613                ctx->srp_ctx.srp_Mask|=SSL_kSRP;
3614                ctx->srp_ctx.SRP_cb_arg=parg;
3615                break;

3617        case SSL_CTRL_SET_TLS_EXT_SRP_STRENGTH:
3618                ctx->srp_ctx.strength=larg;
3619                break;
3620 #endif
3621 #endif /* !OPENSSL_NO_TLSEXT */

3623        /* A Thawte special :-) */
3624        case SSL_CTRL_EXTRA_CHAIN_CERT:
3625                if (ctx->extra_certs == NULL)
```



```
3626                        {
3627                        if ((ctx->extra_certs=sk_X509_new_null()) == NULL)
3628                                return(0);
3629                        }
3630                sk_X509_push(ctx->extra_certs,(X509 *)parg);
3631                break;

3633        case SSL_CTRL_GET_EXTRA_CHAIN_CERTS:
3634                *(STACK_OF(X509) **)parg =  ctx->extra_certs;
3635                break;

3637        case SSL_CTRL_CLEAR_EXTRA_CHAIN_CERTS:
3638                if (ctx->extra_certs)
3639                        {
3640                        sk_X509_pop_free(ctx->extra_certs, X509_free);
3641                        ctx->extra_certs = NULL;
3642                        }
3643                break;

3645        default:
3646                return(0);
3647                }
3648        return(1);
3649        }

3651 long ssl3_ctx_callback_ctrl(SSL_CTX *ctx, int cmd, void (*fp)(void))
3652        {
3653        CERT *cert;

3655        cert=ctx->cert;

3657        switch (cmd)
3658                {
3659 #ifndef OPENSSL_NO_RSA
3660        case SSL_CTRL_SET_TMP_RSA_CB:
3661                {
3662                cert->rsa_tmp_cb = (RSA *(*)(SSL *, int, int))fp;
3663                }
3664                break;
3665 #endif
3666 #ifndef OPENSSL_NO_DH
3667        case SSL_CTRL_SET_TMP_DH_CB:
3668                {
3669                cert->dh_tmp_cb = (DH *(*)(SSL *, int, int))fp;
3670                }
3671                break;
3672 #endif
3673 #ifndef OPENSSL_NO_ECDH
3674        case SSL_CTRL_SET_TMP_ECDH_CB:
3675                {
3676                cert->ecdh_tmp_cb = (EC_KEY *(*)(SSL *, int, int))fp;
3677                }
3678                break;
3679 #endif
3680 #ifndef OPENSSL_NO_TLSEXT
3681        case SSL_CTRL_SET_TLSEXT_SERVERNAME_CB:
3682                ctx->tlsext_servername_callback=(int (*)(SSL *,int *,void *))fp;
3683                break;

3685 #ifdef TLSEXT_TYPE_opaque_prf_input
3686        case SSL_CTRL_SET_TLSEXT_OPAQUE_PRF_INPUT_CB:
3687                ctx->tlsext_opaque_prf_input_callback = (int (*)(SSL *,void *, s
3688                break;
3689 #endif

3691        case SSL_CTRL_SET_TLSEXT_STATUS_REQ_CB:
```

```
3692                 ctx->tlsext_status_cb=(int (*)(SSL *,void *))fp;
3693                 break;

3695         case SSL_CTRL_SET_TLSEXT_TICKET_KEY_CB:
3696                 ctx->tlsext_ticket_key_cb=(int (*)(SSL *,unsigned char  *,
3697                                                    unsigned char *,
3698                                                    EVP_CIPHER_CTX *,
3699                                                    HMAC_CTX *, int))fp;
3700                 break;

3702 #ifndef OPENSSL_NO_SRP
3703         case SSL_CTRL_SET_SRP_VERIFY_PARAM_CB:
3704                 ctx->srp_ctx.srp_Mask|=SSL_kSRP;
3705                 ctx->srp_ctx.SRP_verify_param_callback=(int (*)(SSL *,void *))fp
3706                 break;
3707         case SSL_CTRL_SET_TLS_EXT_SRP_USERNAME_CB:
3708                 ctx->srp_ctx.srp_Mask|=SSL_kSRP;
3709                 ctx->srp_ctx.TLS_ext_srp_username_callback=(int (*)(SSL *,int *,
3710                 break;
3711         case SSL_CTRL_SET_SRP_GIVE_CLIENT_PWD_CB:
3712                 ctx->srp_ctx.srp_Mask|=SSL_kSRP;
3713                 ctx->srp_ctx.SRP_give_srp_client_pwd_callback=(char *(*)(SSL *,v
3714                 break;
3715 #endif
3716 #endif
3717         default:
3718                 return(0);
3719                 }
3720         return(1);
3721         }

3723 /* This function needs to check if the ciphers required are actually
3724  * available */
3725 const SSL_CIPHER *ssl3_get_cipher_by_char(const unsigned char *p)
3726         {
3727         SSL_CIPHER c;
3728         const SSL_CIPHER *cp;
3729         unsigned long id;

3731         id=0x03000000L|((unsigned long)p[0]<<8L)|(unsigned long)p[1];
3732         c.id=id;
3733         cp = OBJ_bsearch_ssl_cipher_id(&c, ssl3_ciphers, SSL3_NUM_CIPHERS);
3734 #ifdef DEBUG_PRINT_UNKNOWN_CIPHERSUITES
3735 if (cp == NULL) fprintf(stderr, "Unknown cipher ID %x\n", (p[0] << 8) | p[1]);
3736 #endif
3737         if (cp == NULL || cp->valid == 0)
3738                 return NULL;
3739         else
3740                 return cp;
3741         }

3743 int ssl3_put_cipher_by_char(const SSL_CIPHER *c, unsigned char *p)
3744         {
3745         long l;

3747         if (p != NULL)
3748                 {
3749                 l=c->id;
3750                 if ((l & 0xff000000) != 0x03000000) return(0);
3751                 p[0]=((unsigned char)(l>> 8L))&0xFF;
3752                 p[1]=((unsigned char)(l     ))&0xFF;
3753                 }
3754         return(2);
3755         }

3757 SSL_CIPHER *ssl3_choose_cipher(SSL *s, STACK_OF(SSL_CIPHER) *clnt,
```



```
3758              STACK_OF(SSL_CIPHER) *srvr)
3759         {
3760         SSL_CIPHER *c,*ret=NULL;
3761         STACK_OF(SSL_CIPHER) *prio, *allow;
3762         int i,ii,ok;
3763 #if !defined(OPENSSL_NO_TLSEXT) && !defined(OPENSSL_NO_EC)
3764         unsigned int j;
3765         int ec_ok, ec_nid;
3766         unsigned char ec_search1 = 0, ec_search2 = 0;
3767 #endif
3768         CERT *cert;
3769         unsigned long alg_k,alg_a,mask_k,mask_a,emask_k,emask_a;

3771         /* Let's see which ciphers we can support */
3772         cert=s->cert;

3774 #if 0
3775         /* Do not set the compare functions, because this may lead to a
3776          * reordering by "id". We want to keep the original ordering.
3777          * We may pay a price in performance during sk_SSL_CIPHER_find(),
3778          * but would have to pay with the price of sk_SSL_CIPHER_dup().
3779          */
3780         sk_SSL_CIPHER_set_cmp_func(srvr, ssl_cipher_ptr_id_cmp);
3781         sk_SSL_CIPHER_set_cmp_func(clnt, ssl_cipher_ptr_id_cmp);
3782 #endif

3784 #ifdef CIPHER_DEBUG
3785         printf("Server has %d from %p:\n", sk_SSL_CIPHER_num(srvr), (void *)srvr
3786         for(i=0 ; i < sk_SSL_CIPHER_num(srvr) ; ++i)
3787                 {
3788                 c=sk_SSL_CIPHER_value(srvr,i);
3789                 printf("%p:%s\n",(void *)c,c->name);
3790                 }
3791         printf("Client sent %d from %p:\n", sk_SSL_CIPHER_num(clnt), (void *)cln
3792         for(i=0 ; i < sk_SSL_CIPHER_num(clnt) ; ++i)
3793             {
3794             c=sk_SSL_CIPHER_value(clnt,i);
3795             printf("%p:%s\n",(void *)c,c->name);
3796             }
3797 #endif

3799         if (s->options & SSL_OP_CIPHER_SERVER_PREFERENCE)
3800                 {
3801                 prio = srvr;
3802                 allow = clnt;
3803                 }
3804         else
3805                 {
3806                 prio = clnt;
3807                 allow = srvr;
3808                 }

3810         for (i=0; i<sk_SSL_CIPHER_num(prio); i++)
3811                 {
3812                 c=sk_SSL_CIPHER_value(prio,i);

3814                 /* Skip TLS v1.2 only ciphersuites if lower than v1.2 */
3815                 if ((c->algorithm_ssl & SSL_TLSV1_2) &&
3816                         (TLS1_get_version(s) < TLS1_2_VERSION))
3817                         continue;

3819                 ssl_set_cert_masks(cert,c);
3820                 mask_k = cert->mask_k;
3821                 mask_a = cert->mask_a;
3822                 emask_k = cert->export_mask_k;
3823                 emask_a = cert->export_mask_a;
```

```
#ifndef OPENSSL_NO_SRP
                mask_k=cert->mask_k | s->srp_ctx.srp_Mask;
                emask_k=cert->export_mask_k | s->srp_ctx.srp_Mask;
#endif

#ifdef KSSL_DEBUG
/*              printf("ssl3_choose_cipher %d alg= %lx\n", i,c->algorithms);*/
#endif    /* KSSL_DEBUG */

                alg_k=c->algorithm_mkey;
                alg_a=c->algorithm_auth;

#ifndef OPENSSL_NO_KRB5
                if (alg_k & SSL_kKRB5)
                        {
                        if ( !kssl_keytab_is_available(s->kssl_ctx) )
                            continue;
                        }
#endif /* OPENSSL_NO_KRB5 */
#ifndef OPENSSL_NO_PSK
                /* with PSK there must be server callback set */
                if ((alg_k & SSL_kPSK) && s->psk_server_callback == NULL)
                        continue;
#endif /* OPENSSL_NO_PSK */

                if (SSL_C_IS_EXPORT(c))
                        {
                        ok = (alg_k & emask_k) && (alg_a & emask_a);
#ifdef CIPHER_DEBUG
                        printf("%d:[%08lX:%08lX:%08lX:%08lX]%p:%s (export)\n",ok
                               (void *)c,c->name);
#endif
                        }
                else
                        {
                        ok = (alg_k & mask_k) && (alg_a & mask_a);
#ifdef CIPHER_DEBUG
                        printf("%d:[%08lX:%08lX:%08lX:%08lX]%p:%s\n",ok,alg_k,al
                               c->name);
#endif
                        }

#ifndef OPENSSL_NO_TLSEXT
#ifndef OPENSSL_NO_EC
                if (
                        /* if we are considering an ECC cipher suite that uses o
                        (alg_a & SSL_aECDSA || alg_a & SSL_aECDH)
                        /* and we have an ECC certificate */
                        && (s->cert->pkeys[SSL_PKEY_ECC].x509 != NULL)
                        /* and the client specified a Supported Point Formats ex
                        && ((s->session->tlsext_ecpointformatlist_length > 0) &&
                        /* and our certificate's point is compressed */
                        && (
                                (s->cert->pkeys[SSL_PKEY_ECC].x509->cert_info !=
                                && (s->cert->pkeys[SSL_PKEY_ECC].x509->cert_info
                                && (s->cert->pkeys[SSL_PKEY_ECC].x509->cert_info
                                && (s->cert->pkeys[SSL_PKEY_ECC].x509->cert_info
                                && (
                                        (*(s->cert->pkeys[SSL_PKEY_ECC].x509->ce
                                        || (*(s->cert->pkeys[SSL_PKEY_ECC].x509-
                                        )
                                )
                )
                        {
                        ec_ok = 0;
                        /* if our certificate's curve is over a field type that
```



```
                         * then do not allow this cipher suite to be negotiated
                        if (
                                (s->cert->pkeys[SSL_PKEY_ECC].privatekey->pkey.e
                                && (s->cert->pkeys[SSL_PKEY_ECC].privatekey->pke
                                && (s->cert->pkeys[SSL_PKEY_ECC].privatekey->pke
                                && (EC_METHOD_get_field_type(s->cert->pkeys[SSL_
                        )
                                {
                                for (j = 0; j < s->session->tlsext_ecpointformat
                                        {
                                        if (s->session->tlsext_ecpointformatlist
                                                {
                                                ec_ok = 1;
                                                break;
                                                }
                                        }
                                }
                        else if (EC_METHOD_get_field_type(s->cert->pkeys[SSL_PKE
                                {
                                for (j = 0; j < s->session->tlsext_ecpointformat
                                        {
                                        if (s->session->tlsext_ecpointformatlist
                                                {
                                                ec_ok = 1;
                                                break;
                                                }
                                        }
                                }
                        ok = ok && ec_ok;
                        }
                if (
                        /* if we are considering an ECC cipher suite that uses o
                        (alg_a & SSL_aECDSA || alg_a & SSL_aECDH)
                        /* and we have an ECC certificate */
                        && (s->cert->pkeys[SSL_PKEY_ECC].x509 != NULL)
                        /* and the client specified an EllipticCurves extension
                        && ((s->session->tlsext_ellipticcurvelist_length > 0) &&
                )
                        {
                        ec_ok = 0;
                        if (
                                (s->cert->pkeys[SSL_PKEY_ECC].privatekey->pkey.e
                                && (s->cert->pkeys[SSL_PKEY_ECC].privatekey->pke
                        )
                                {
                                ec_nid = EC_GROUP_get_curve_name(s->cert->pkeys[
                                if ((ec_nid == 0)
                                        && (s->cert->pkeys[SSL_PKEY_ECC].private
                                )
                                        {
                                        if (EC_METHOD_get_field_type(s->cert->pk
                                                {
                                                ec_search1 = 0xFF;
                                                ec_search2 = 0x01;
                                                }
                                        else if (EC_METHOD_get_field_type(s->cer
                                                {
                                                ec_search1 = 0xFF;
                                                ec_search2 = 0x02;
                                                }
                                        }
                                else
                                        {
                                        ec_search1 = 0x00;
                                        ec_search2 = tls1_ec_nid2curve_id(ec_nid
                                        }
```

```
3956                                if ((ec_search1 != 0) || (ec_search2 != 0))
3957                                        {
3958                                        for (j = 0; j < s->session->tlsext_ellip
3959                                                {
3960                                                if ((s->session->tlsext_elliptic
3961                                                        {
3962                                                        ec_ok = 1;
3963                                                        break;
3964                                                        }
3965                                                }
3966                                        }
3967                                }
3968                        ok = ok && ec_ok;
3969                        }
3970                if (
3971                        /* if we are considering an ECC cipher suite that uses a
3972                        (alg_k & SSL_kEECDH)
3973                        /* and we have an ephemeral EC key */
3974                        && (s->cert->ecdh_tmp != NULL)
3975                        /* and the client specified an EllipticCurves extension
3976                        && ((s->session->tlsext_ellipticcurvelist_length > 0) &&
3977                )
3978                        {
3979                        ec_ok = 0;
3980                        if (s->cert->ecdh_tmp->group != NULL)
3981                                {
3982                                ec_nid = EC_GROUP_get_curve_name(s->cert->ecdh_t
3983                                if ((ec_nid == 0)
3984                                        && (s->cert->ecdh_tmp->group->meth != NU
3985                                )
3986                                        {
3987                                        if (EC_METHOD_get_field_type(s->cert->ec
3988                                                {
3989                                                ec_search1 = 0xFF;
3990                                                ec_search2 = 0x01;
3991                                                }
3992                                        else if (EC_METHOD_get_field_type(s->cer
3993                                                {
3994                                                ec_search1 = 0xFF;
3995                                                ec_search2 = 0x02;
3996                                                }
3997                                        }
3998                                else
3999                                        {
4000                                        ec_search1 = 0x00;
4001                                        ec_search2 = tls1_ec_nid2curve_id(ec_nid
4002                                        }
4003                                if ((ec_search1 != 0) || (ec_search2 != 0))
4004                                        {
4005                                        for (j = 0; j < s->session->tlsext_ellip
4006                                                {
4007                                                if ((s->session->tlsext_elliptic
4008                                                        {
4009                                                        ec_ok = 1;
4010                                                        break;
4011                                                        }
4012                                                }
4013                                        }
4014                                }
4015                        ok = ok && ec_ok;
4016                        }
4017 #endif /* OPENSSL_NO_EC */
4018 #endif /* OPENSSL_NO_TLSEXT */

4020                if (!ok) continue;
4021                ii=sk_SSL_CIPHER_find(allow,c);
```



```
4022                if (ii >= 0)
4023                        {
4024 #if !defined(OPENSSL_NO_EC) && !defined(OPENSSL_NO_TLSEXT)
4025                        if ((alg_k & SSL_kEECDH) && (alg_a & SSL_aECDSA) && s->s
4026                                {
4027                                if (!ret) ret=sk_SSL_CIPHER_value(allow,ii);
4028                                continue;
4029                                }
4030 #endif
4031                        ret=sk_SSL_CIPHER_value(allow,ii);
4032                        break;
4033                        }
4034                }
4035        return(ret);
4036        }

4038 int ssl3_get_req_cert_type(SSL *s, unsigned char *p)
4039        {
4040        int ret=0;
4041        unsigned long alg_k;

4043        alg_k = s->s3->tmp.new_cipher->algorithm_mkey;

4045 #ifndef OPENSSL_NO_GOST
4046        if (s->version >= TLS1_VERSION)
4047                {
4048                if (alg_k & SSL_kGOST)
4049                        {
4050                        p[ret++]=TLS_CT_GOST94_SIGN;
4051                        p[ret++]=TLS_CT_GOST01_SIGN;
4052                        return(ret);
4053                        }
4054                }
4055 #endif

4057 #ifndef OPENSSL_NO_DH
4058        if (alg_k & (SSL_kDHr|SSL_kEDH))
4059                {
4060 #  ifndef OPENSSL_NO_RSA
4061                p[ret++]=SSL3_CT_RSA_FIXED_DH;
4062 #  endif
4063 #  ifndef OPENSSL_NO_DSA
4064                p[ret++]=SSL3_CT_DSS_FIXED_DH;
4065 #  endif
4066                }
4067        if ((s->version == SSL3_VERSION) &&
4068                (alg_k & (SSL_kEDH|SSL_kDHd|SSL_kDHr)))
4069                {
4070 #  ifndef OPENSSL_NO_RSA
4071                p[ret++]=SSL3_CT_RSA_EPHEMERAL_DH;
4072 #  endif
4073 #  ifndef OPENSSL_NO_DSA
4074                p[ret++]=SSL3_CT_DSS_EPHEMERAL_DH;
4075 #  endif
4076                }
4077 #endif /* !OPENSSL_NO_DH */
4078 #ifndef OPENSSL_NO_RSA
4079        p[ret++]=SSL3_CT_RSA_SIGN;
4080 #endif
4081 #ifndef OPENSSL_NO_DSA
4082        p[ret++]=SSL3_CT_DSS_SIGN;
4083 #endif
4084 #ifndef OPENSSL_NO_ECDH
4085        if ((alg_k & (SSL_kECDHr|SSL_kECDHe)) && (s->version >= TLS1_VERSION))
4086                {
4087                p[ret++]=TLS_CT_RSA_FIXED_ECDH;
```

```
4088                 p[ret++]=TLS_CT_ECDSA_FIXED_ECDH;
4089                 }
4090 #endif

4092 #ifndef OPENSSL_NO_ECDSA
4093         /* ECDSA certs can be used with RSA cipher suites as well
4094          * so we don't need to check for SSL_kECDH or SSL_kEECDH
4095          */
4096         if (s->version >= TLS1_VERSION)
4097                 {
4098                 p[ret++]=TLS_CT_ECDSA_SIGN;
4099                 }
4100 #endif
4101         return(ret);
4102         }

4104 int ssl3_shutdown(SSL *s)
4105         {
4106         int ret;

4108         /* Don't do anything much if we have not done the handshake or
4109          * we don't want to send messages :-) */
4110         if ((s->quiet_shutdown) || (s->state == SSL_ST_BEFORE))
4111                 {
4112                 s->shutdown=(SSL_SENT_SHUTDOWN|SSL_RECEIVED_SHUTDOWN);
4113                 return(1);
4114                 }

4116         if (!(s->shutdown & SSL_SENT_SHUTDOWN))
4117                 {
4118                 s->shutdown|=SSL_SENT_SHUTDOWN;
4119 #if 1
4120                 ssl3_send_alert(s,SSL3_AL_WARNING,SSL_AD_CLOSE_NOTIFY);
4121 #endif
4122                 /* our shutdown alert has been sent now, and if it still needs
4123                  * to be written, s->s3->alert_dispatch will be true */
4124                 if (s->s3->alert_dispatch)
4125                         return(-1);     /* return WANT_WRITE */
4126                 }
4127         else if (s->s3->alert_dispatch)
4128                 {
4129                 /* resend it if not sent */
4130 #if 1
4131                 ret=s->method->ssl_dispatch_alert(s);
4132                 if(ret == -1)
4133                         {
4134                         /* we only get to return -1 here the 2nd/Nth
4135                          * invocation, we must  have already signalled
4136                          * return 0 upon a previous invoation,
4137                          * return WANT_WRITE */
4138                         return(ret);
4139                         }
4140 #endif
4141                 }
4142         else if (!(s->shutdown & SSL_RECEIVED_SHUTDOWN))
4143                 {
4144                 /* If we are waiting for a close from our peer, we are closed */
4145                 s->method->ssl_read_bytes(s,0,NULL,0,0);
4146                 if(!(s->shutdown & SSL_RECEIVED_SHUTDOWN))
4147                         {
4148                         return(-1);     /* return WANT_READ */
4149                         }
4150                 }

4152         if ((s->shutdown == (SSL_SENT_SHUTDOWN|SSL_RECEIVED_SHUTDOWN)) &&
4153                 !s->s3->alert_dispatch)
```



```
4154                 return(1);
4155         else
4156                 return(0);
4157         }

4159 int ssl3_write(SSL *s, const void *buf, int len)
4160         {
4161         int ret,n;

4163 #if 0
4164         if (s->shutdown & SSL_SEND_SHUTDOWN)
4165                 {
4166                 s->rwstate=SSL_NOTHING;
4167                 return(0);
4168                 }
4169 #endif
4170         clear_sys_error();
4171         if (s->s3->renegotiate) ssl3_renegotiate_check(s);

4173         /* This is an experimental flag that sends the
4174          * last handshake message in the same packet as the first
4175          * use data - used to see if it helps the TCP protocol during
4176          * session-id reuse */
4177         /* The second test is because the buffer may have been removed */
4178         if ((s->s3->flags & SSL3_FLAGS_POP_BUFFER) && (s->wbio == s->bbio))
4179                 {
4180                 /* First time through, we write into the buffer */
4181                 if (s->s3->delay_buf_pop_ret == 0)
4182                         {
4183                         ret=ssl3_write_bytes(s,SSL3_RT_APPLICATION_DATA,
4184                                              buf,len);
4185                         if (ret <= 0) return(ret);

4187                         s->s3->delay_buf_pop_ret=ret;
4188                         }

4190                 s->rwstate=SSL_WRITING;
4191                 n=BIO_flush(s->wbio);
4192                 if (n <= 0) return(n);
4193                 s->rwstate=SSL_NOTHING;

4195                 /* We have flushed the buffer, so remove it */
4196                 ssl_free_wbio_buffer(s);
4197                 s->s3->flags&= ~SSL3_FLAGS_POP_BUFFER;

4199                 ret=s->s3->delay_buf_pop_ret;
4200                 s->s3->delay_buf_pop_ret=0;
4201                 }
4202         else
4203                 {
4204                 ret=s->method->ssl_write_bytes(s,SSL3_RT_APPLICATION_DATA,
4205                         buf,len);
4206                 if (ret <= 0) return(ret);
4207                 }

4209         return(ret);
4210         }

4212 static int ssl3_read_internal(SSL *s, void *buf, int len, int peek)
4213         {
4214         int ret;

4216         clear_sys_error();
4217         if (s->s3->renegotiate) ssl3_renegotiate_check(s);
4218         s->s3->in_read_app_data=1;
4219         ret=s->method->ssl_read_bytes(s,SSL3_RT_APPLICATION_DATA,buf,len,peek);
```

```
4220         if ((ret == -1) && (s->s3->in_read_app_data == 2))
4221                 {
4222                 /* ssl3_read_bytes decided to call s->handshake_func, which
4223                  * called ssl3_read_bytes to read handshake data.
4224                  * However, ssl3_read_bytes actually found application data
4225                  * and thinks that application data makes sense here; so disable
4226                  * handshake processing and try to read application data again.
4227                 s->in_handshake++;
4228                 ret=s->method->ssl_read_bytes(s,SSL3_RT_APPLICATION_DATA,buf,len
4229                 s->in_handshake--;
4230                 }
4231         else
4232                 s->s3->in_read_app_data=0;

4234         return(ret);
4235         }

4237 int ssl3_read(SSL *s, void *buf, int len)
4238         {
4239         return ssl3_read_internal(s, buf, len, 0);
4240         }

4242 int ssl3_peek(SSL *s, void *buf, int len)
4243         {
4244         return ssl3_read_internal(s, buf, len, 1);
4245         }

4247 int ssl3_renegotiate(SSL *s)
4248         {
4249         if (s->handshake_func == NULL)
4250                 return(1);

4252         if (s->s3->flags & SSL3_FLAGS_NO_RENEGOTIATE_CIPHERS)
4253                 return(0);

4255         s->s3->renegotiate=1;
4256         return(1);
4257         }

4259 int ssl3_renegotiate_check(SSL *s)
4260         {
4261         int ret=0;

4263         if (s->s3->renegotiate)
4264                 {
4265                 if (    (s->s3->rbuf.left == 0) &&
4266                         (s->s3->wbuf.left == 0) &&
4267                         !SSL_in_init(s))
4268                         {
4269 /*
4270 if we are the server, and we have sent a 'RENEGOTIATE' message, we
4271 need to go to SSL_ST_ACCEPT.
4272 */
4273                         /* SSL_ST_ACCEPT */
4274                         s->state=SSL_ST_RENEGOTIATE;
4275                         s->s3->renegotiate=0;
4276                         s->s3->num_renegotiations++;
4277                         s->s3->total_renegotiations++;
4278                         ret=1;
4279                         }
4280                 }
4281         return(ret);
4282         }
4283 /* If we are using TLS v1.2 or later and default SHA1+MD5 algorithms switch
4284  * to new SHA256 PRF and handshake macs
4285  */
```



```
4286 long ssl_get_algorithm2(SSL *s)
4287         {
4288         long alg2 = s->s3->tmp.new_cipher->algorithm2;
4289         if (s->method->version == TLS1_2_VERSION &&
4290             alg2 == (SSL_HANDSHAKE_MAC_DEFAULT|TLS1_PRF))
4291                 return SSL_HANDSHAKE_MAC_SHA256 | TLS1_PRF_SHA256;
4292         return alg2;
4293         }
4294 #endif /* ! codereview */
```

```
**********************************************************
    3532 Wed Aug 13 19:53:39 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_meth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/s3_meth.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/objects.h>
#include "ssl_locl.h"
```



```
static const SSL_METHOD *ssl3_get_method(int ver);
static const SSL_METHOD *ssl3_get_method(int ver)
        {
        if (ver == SSL3_VERSION)
                return(SSLv3_method());
        else
                return(NULL);
        }

IMPLEMENT_ssl3_meth_func(SSLv3_method,
                         ssl3_accept,
                         ssl3_connect,
                         ssl3_get_method)
#endif /* ! codereview */
```

```
**********************************************************
   44146 Wed Aug 13 19:53:39 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_pkt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/s3_pkt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2002 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <limits.h>
#include <errno.h>
#define USE_SOCKETS
#include "ssl_locl.h"
#include <openssl/evp.h>
#include <openssl/buffer.h>
#include <openssl/rand.h>

static int do_ssl3_write(SSL *s, int type, const unsigned char *buf,
                         unsigned int len, int create_empty_fragment);
static int ssl3_get_record(SSL *s);

int ssl3_read_n(SSL *s, int n, int max, int extend)
        {
        /* If extend == 0, obtain new n-byte packet; if extend == 1, increase
```

```
128          * packet by another n bytes.
129          * The packet will be in the sub-array of s->s3->rbuf.buf specified
130          * by s->packet and s->packet_length.
131          * (If s->read_ahead is set, 'max' bytes may be stored in rbuf
132          * [plus s->packet_length bytes if extend == 1].)
133          */
134         int i,len,left;
135         long align=0;
136         unsigned char *pkt;
137         SSL3_BUFFER *rb;
138
139         if (n <= 0) return n;
140
141         rb    = &(s->s3->rbuf);
142         if (rb->buf == NULL)
143                 if (!ssl3_setup_read_buffer(s))
144                         return -1;
145
146         left  = rb->left;
147 #if defined(SSL3_ALIGN_PAYLOAD) && SSL3_ALIGN_PAYLOAD!=0
148         align = (long)rb->buf + SSL3_RT_HEADER_LENGTH;
149         align = (-align)&(SSL3_ALIGN_PAYLOAD-1);
150 #endif
151
152         if (!extend)
153                 {
154                 /* start with empty packet ... */
155                 if (left == 0)
156                         rb->offset = align;
157                 else if (align != 0 && left >= SSL3_RT_HEADER_LENGTH)
158                         {
159                         /* check if next packet length is large
160                          * enough to justify payload alignment... */
161                         pkt = rb->buf + rb->offset;
162                         if (pkt[0] == SSL3_RT_APPLICATION_DATA
163                             && (pkt[3]<<8|pkt[4]) >= 128)
164                                 {
165                                 /* Note that even if packet is corrupted
166                                  * and its length field is insane, we can
167                                  * only be led to wrong decision about
168                                  * whether memmove will occur or not.
169                                  * Header values has no effect on memmove
170                                  * arguments and therefore no buffer
171                                  * overrun can be triggered. */
172                                 memmove (rb->buf+align,pkt,left);
173                                 rb->offset = align;
174                                 }
175                         }
176                 s->packet = rb->buf + rb->offset;
177                 s->packet_length = 0;
178                 /* ... now we can act as if 'extend' was set */
179                 }
180
181         /* For DTLS/UDP reads should not span multiple packets
182          * because the read operation returns the whole packet
183          * at once (as long as it fits into the buffer). */
184         if (SSL_version(s) == DTLS1_VERSION || SSL_version(s) == DTLS1_BAD_VER)
185                 {
186                 if (left > 0 && n > left)
187                         n = left;
188                 }
189
190         /* if there is enough in the buffer from a previous read, take some */
191         if (left >= n)
192                 {
193                 s->packet_length+=n;
```



```
194                 rb->left=left-n;
195                 rb->offset+=n;
196                 return(n);
197                 }
198
199         /* else we need to read more data */
200
201         len = s->packet_length;
202         pkt = rb->buf+align;
203         /* Move any available bytes to front of buffer:
204          * 'len' bytes already pointed to by 'packet',
205          * 'left' extra ones at the end */
206         if (s->packet != pkt) /* len > 0 */
207                 {
208                 memmove(pkt, s->packet, len+left);
209                 s->packet = pkt;
210                 rb->offset = len + align;
211                 }
212
213         if (n > (int)(rb->len - rb->offset)) /* does not happen */
214                 {
215                 SSLerr(SSL_F_SSL3_READ_N,ERR_R_INTERNAL_ERROR);
216                 return -1;
217                 }
218
219         if (!s->read_ahead)
220                 /* ignore max parameter */
221                 max = n;
222         else
223                 {
224                 if (max < n)
225                         max = n;
226                 if (max > (int)(rb->len - rb->offset))
227                         max = rb->len - rb->offset;
228                 }
229
230         while (left < n)
231                 {
232                 /* Now we have len+left bytes at the front of s->s3->rbuf.buf
233                  * and need to read in more until we have len+n (up to
234                  * len+max if possible) */
235
236                 clear_sys_error();
237                 if (s->rbio != NULL)
238                         {
239                         s->rwstate=SSL_READING;
240                         i=BIO_read(s->rbio,pkt+len+left, max-left);
241                         }
242                 else
243                         {
244                         SSLerr(SSL_F_SSL3_READ_N,SSL_R_READ_BIO_NOT_SET);
245                         i = -1;
246                         }
247
248                 if (i <= 0)
249                         {
250                         rb->left = left;
251                         if (s->mode & SSL_MODE_RELEASE_BUFFERS &&
252                             SSL_version(s) != DTLS1_VERSION && SSL_version(s) !=
253                                 if (len+left == 0)
254                                         ssl3_release_read_buffer(s);
255                         return(i);
256                         }
257                 left+=i;
258                 /* reads should *never* span multiple packets for DTLS because
259                  * the underlying transport protocol is message oriented as oppo
```

```
260                  * to byte oriented as in the TLS case. */
261                 if (SSL_version(s) == DTLS1_VERSION || SSL_version(s) == DTLS1_B
262                         {
263                         if (n > left)
264                                 n = left; /* makes the while condition false */
265                         }
266                 }

268         /* done reading, now the book-keeping */
269         rb->offset += n;
270         rb->left = left - n;
271         s->packet_length += n;
272         s->rwstate=SSL_NOTHING;
273         return(n);
274         }

276 /* Call this to get a new input record.
277  * It will return <= 0 if more data is needed, normally due to an error
278  * or non-blocking IO.
279  * When it finishes, one packet has been decoded and can be found in
280  * ssl->s3->rrec.type    - is the type of record
281  * ssl->s3->rrec.data,   - data
282  * ssl->s3->rrec.length, - number of bytes
283  */
284 /* used only by ssl3_read_bytes */
285 static int ssl3_get_record(SSL *s)
286         {
287         int ssl_major,ssl_minor,al;
288         int enc_err,n,i,ret= -1;
289         SSL3_RECORD *rr;
290         SSL_SESSION *sess;
291         unsigned char *p;
292         unsigned char md[EVP_MAX_MD_SIZE];
293         short version;
294         unsigned mac_size, orig_len;
295         size_t extra;

297         rr= &(s->s3->rrec);
298         sess=s->session;

300         if (s->options & SSL_OP_MICROSOFT_BIG_SSLV3_BUFFER)
301                 extra=SSL3_RT_MAX_EXTRA;
302         else
303                 extra=0;
304         if (extra && !s->s3->init_extra)
305                 {
306                 /* An application error: SLS_OP_MICROSOFT_BIG_SSLV3_BUFFER
307                  * set after ssl3_setup_buffers() was done */
308                 SSLerr(SSL_F_SSL3_GET_RECORD, ERR_R_INTERNAL_ERROR);
309                 return -1;
310                 }

312 again:
313         /* check if we have the header */
314         if (    (s->rstate != SSL_ST_READ_BODY) ||
315                 (s->packet_length < SSL3_RT_HEADER_LENGTH))
316                 {
317                 n=ssl3_read_n(s, SSL3_RT_HEADER_LENGTH, s->s3->rbuf.len, 0);
318                 if (n <= 0) return(n); /* error or non-blocking */
319                 s->rstate=SSL_ST_READ_BODY;

321                 p=s->packet;

323                 /* Pull apart the header into the SSL3_RECORD */
324                 rr->type= *(p++);
325                 ssl_major= *(p++);
```



```
326                 ssl_minor= *(p++);
327                 version=(ssl_major<<8)|ssl_minor;
328                 n2s(p,rr->length);
329 #if 0
330 fprintf(stderr, "Record type=%d, Length=%d\n", rr->type, rr->length);
331 #endif

333                 /* Lets check version */
334                 if (!s->first_packet)
335                         {
336                         if (version != s->version)
337                                 {
338                                 SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_WRONG_VERSION
339                                 if ((s->version & 0xFF00) == (version & 0xFF00)
340                                         /* Send back error using their minor ver
341                                         s->version = (unsigned short)version;
342                                 al=SSL_AD_PROTOCOL_VERSION;
343                                 goto f_err;
344                                 }
345                         }

347                 if ((version>>8) != SSL3_VERSION_MAJOR)
348                         {
349                         SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_WRONG_VERSION_NUMBER)
350                         goto err;
351                         }

353                 if (rr->length > s->s3->rbuf.len - SSL3_RT_HEADER_LENGTH)
354                         {
355                         al=SSL_AD_RECORD_OVERFLOW;
356                         SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_PACKET_LENGTH_TOO_LON
357                         goto f_err;
358                         }

360                 /* now s->rstate == SSL_ST_READ_BODY */
361                 }

363         /* s->rstate == SSL_ST_READ_BODY, get and decode the data */

365         if (rr->length > s->packet_length-SSL3_RT_HEADER_LENGTH)
366                 {
367                 /* now s->packet_length == SSL3_RT_HEADER_LENGTH */
368                 i=rr->length;
369                 n=ssl3_read_n(s,i,i,1);
370                 if (n <= 0) return(n); /* error or non-blocking io */
371                 /* now n == rr->length,
372                  * and s->packet_length == SSL3_RT_HEADER_LENGTH + rr->length */
373                 }

375         s->rstate=SSL_ST_READ_HEADER; /* set state for later operations */

377         /* At this point, s->packet_length == SSL3_RT_HEADER_LNGTH + rr->length,
378          * and we have that many bytes in s->packet
379          */
380         rr->input= &(s->packet[SSL3_RT_HEADER_LENGTH]);

382         /* ok, we can now read from 's->packet' data into 'rr'
383          * rr->input points at rr->length bytes, which
384          * need to be copied into rr->data by either
385          * the decryption or by the decompression
386          * When the data is 'copied' into the rr->data buffer,
387          * rr->input will be pointed at the new buffer */

389         /* We now have - encrypted [ MAC [ compressed [ plain ] ] ]
390          * rr->length bytes of encrypted compressed stuff. */
```

```
392         /* check is not needed I believe */
393         if (rr->length > SSL3_RT_MAX_ENCRYPTED_LENGTH+extra)
394                 {
395                 al=SSL_AD_RECORD_OVERFLOW;
396                 SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_ENCRYPTED_LENGTH_TOO_LONG);
397                 goto f_err;
398                 }
399
400         /* decrypt in place in 'rr->input' */
401         rr->data=rr->input;
402
403         enc_err = s->method->ssl3_enc->enc(s,0);
404         /* enc_err is:
405          *    0: (in non-constant time) if the record is publically invalid.
406          *    1: if the padding is valid
407          *    -1: if the padding is invalid */
408         if (enc_err == 0)
409                 {
410                 al=SSL_AD_DECRYPTION_FAILED;
411                 SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_BLOCK_CIPHER_PAD_IS_WRONG);
412                 goto f_err;
413                 }
414
415 #ifdef TLS_DEBUG
416 printf("dec %d\n",rr->length);
417 { unsigned int z; for (z=0; z<rr->length; z++) printf("%02X%c",rr->data[z],((z+1
418 printf("\n");
419 #endif
420
421         /* r->length is now the compressed data plus mac */
422         if ((sess != NULL) &&
423             (s->enc_read_ctx != NULL) &&
424             (EVP_MD_CTX_md(s->read_hash) != NULL))
425                 {
426                 /* s->read_hash != NULL => mac_size != -1 */
427                 unsigned char *mac = NULL;
428                 unsigned char mac_tmp[EVP_MAX_MD_SIZE];
429                 mac_size=EVP_MD_CTX_size(s->read_hash);
430                 OPENSSL_assert(mac_size <= EVP_MAX_MD_SIZE);
431
432                 /* kludge: *_cbc_remove_padding passes padding length in rr->typ
433                 orig_len = rr->length+((unsigned int)rr->type>>8);
434
435                 /* orig_len is the length of the record before any padding was
436                  * removed. This is public information, as is the MAC in use,
437                  * therefore we can safely process the record in a different
438                  * amount of time if it's too short to possibly contain a MAC.
439                  */
440                 if (orig_len < mac_size ||
441                     /* CBC records must have a padding length byte too. */
442                     (EVP_CIPHER_CTX_mode(s->enc_read_ctx) == EVP_CIPH_CBC_MODE &
443                      orig_len < mac_size+1))
444                         {
445                         al=SSL_AD_DECODE_ERROR;
446                         SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_LENGTH_TOO_SHORT);
447                         goto f_err;
448                         }
449
450                 if (EVP_CIPHER_CTX_mode(s->enc_read_ctx) == EVP_CIPH_CBC_MODE)
451                         {
452                         /* We update the length so that the TLS header bytes
453                          * can be constructed correctly but we need to extract
454                          * the MAC in constant time from within the record,
455                          * without leaking the contents of the padding bytes.
456                          * */
457                         mac = mac_tmp;
```



```
458                         ssl3_cbc_copy_mac(mac_tmp, rr, mac_size, orig_len);
459                         rr->length -= mac_size;
460                         }
461                 else
462                         {
463                         /* In this case there's no padding, so |orig_len|
464                          * equals |rec->length| and we checked that there's
465                          * enough bytes for |mac_size| above. */
466                         rr->length -= mac_size;
467                         mac = &rr->data[rr->length];
468                         }
469
470                 i=s->method->ssl3_enc->mac(s,md,0 /* not send */);
471                 if (i < 0 || mac == NULL || CRYPTO_memcmp(md, mac, (size_t)mac_s
472                         enc_err = -1;
473                 if (rr->length > SSL3_RT_MAX_COMPRESSED_LENGTH+extra+mac_size)
474                         enc_err = -1;
475                 }
476
477         if (enc_err < 0)
478                 {
479                 /* A separate 'decryption_failed' alert was introduced with TLS
480                  * SSL 3.0 only has 'bad_record_mac'.  But unless a decryption
481                  * failure is directly visible from the ciphertext anyway,
482                  * we should not reveal which kind of error occured -- this
483                  * might become visible to an attacker (e.g. via a logfile) */
484                 al=SSL_AD_BAD_RECORD_MAC;
485                 SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_DECRYPTION_FAILED_OR_BAD_RECO
486                 goto f_err;
487                 }
488
489         /* r->length is now just compressed */
490         if (s->expand != NULL)
491                 {
492                 if (rr->length > SSL3_RT_MAX_COMPRESSED_LENGTH+extra)
493                         {
494                         al=SSL_AD_RECORD_OVERFLOW;
495                         SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_COMPRESSED_LENGTH_TOO
496                         goto f_err;
497                         }
498                 if (!ssl3_do_uncompress(s))
499                         {
500                         al=SSL_AD_DECOMPRESSION_FAILURE;
501                         SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_BAD_DECOMPRESSION);
502                         goto f_err;
503                         }
504                 }
505
506         if (rr->length > SSL3_RT_MAX_PLAIN_LENGTH+extra)
507                 {
508                 al=SSL_AD_RECORD_OVERFLOW;
509                 SSLerr(SSL_F_SSL3_GET_RECORD,SSL_R_DATA_LENGTH_TOO_LONG);
510                 goto f_err;
511                 }
512
513         rr->off=0;
514         /* So at this point the following is true
515          * ssl->s3->rrec.type   is the type of record
516          * ssl->s3->rrec.length == number of bytes in record
517          * ssl->s3->rrec.off    == offset to first valid byte
518          * ssl->s3->rrec.data   == where to take bytes from, increment
519          *                         after use :-).
520          */
521
522         /* we have pulled in a full packet so zero things */
523         s->packet_length=0;
```

```
525         /* just read a 0 length packet */
526         if (rr->length == 0) goto again;

528 #if 0
529 fprintf(stderr, "Ultimate Record type=%d, Length=%d\n", rr->type, rr->length);
530 #endif

532         return(1);

534 f_err:
535         ssl3_send_alert(s,SSL3_AL_FATAL,al);
536 err:
537         return(ret);
538         }

540 int ssl3_do_uncompress(SSL *ssl)
541         {
542 #ifndef OPENSSL_NO_COMP
543         int i;
544         SSL3_RECORD *rr;

546         rr= &(ssl->s3->rrec);
547         i=COMP_expand_block(ssl->expand,rr->comp,
548                 SSL3_RT_MAX_PLAIN_LENGTH,rr->data,(int)rr->length);
549         if (i < 0)
550                 return(0);
551         else
552                 rr->length=i;
553         rr->data=rr->comp;
554 #endif
555         return(1);
556         }

558 int ssl3_do_compress(SSL *ssl)
559         {
560 #ifndef OPENSSL_NO_COMP
561         int i;
562         SSL3_RECORD *wr;

564         wr= &(ssl->s3->wrec);
565         i=COMP_compress_block(ssl->compress,wr->data,
566                 SSL3_RT_MAX_COMPRESSED_LENGTH,
567                 wr->input,(int)wr->length);
568         if (i < 0)
569                 return(0);
570         else
571                 wr->length=i;

573         wr->input=wr->data;
574 #endif
575         return(1);
576         }

578 /* Call this to write data in records of type 'type'
579  * It will return <= 0 if not all data has been sent or non-blocking IO.
580  */
581 int ssl3_write_bytes(SSL *s, int type, const void *buf_, int len)
582         {
583         const unsigned char *buf=buf_;
584         unsigned int n,nw;
585         int i,tot;

587         s->rwstate=SSL_NOTHING;
588         OPENSSL_assert(s->s3->wnum <= INT_MAX);
589         tot=s->s3->wnum;
```



```
590         s->s3->wnum=0;

592         if (SSL_in_init(s) && !s->in_handshake)
593                 {
594                 i=s->handshake_func(s);
595                 if (i < 0) return(i);
596                 if (i == 0)
597                         {
598                         SSLerr(SSL_F_SSL3_WRITE_BYTES,SSL_R_SSL_HANDSHAKE_FAILUR
599                         return -1;
600                         }
601                 }

603         /* ensure that if we end up with a smaller value of data to write
604          * out than the the original len from a write which didn't complete
605          * for non-blocking I/O and also somehow ended up avoiding
606          * the check for this in ssl3_write_pending/SSL_R_BAD_WRITE_RETRY as
607          * it must never be possible to end up with (len-tot) as a large
608          * number that will then promptly send beyond the end of the users
609          * buffer ... so we trap and report the error in a way the user
610          * will notice
611          */
612         if (len < tot)
613                 {
614                 SSLerr(SSL_F_SSL3_WRITE_BYTES,SSL_R_BAD_LENGTH);
615                 return(-1);
616                 }


619         n=(len-tot);
620         for (;;)
621                 {
622                 if (n > s->max_send_fragment)
623                         nw=s->max_send_fragment;
624                 else
625                         nw=n;

627                 i=do_ssl3_write(s, type, &(buf[tot]), nw, 0);
628                 if (i <= 0)
629                         {
630                         s->s3->wnum=tot;
631                         return i;
632                         }

634                 if ((i == (int)n) ||
635                         (type == SSL3_RT_APPLICATION_DATA &&
636                          (s->mode & SSL_MODE_ENABLE_PARTIAL_WRITE)))
637                         {
638                         /* next chunk of data should get another prepended empty
639                          * in ciphersuites with known-IV weakness: */
640                         s->s3->empty_fragment_done = 0;
641
642                         return tot+i;
643                         }

645                 n-=i;
646                 tot+=i;
647                 }
648         }

650 static int do_ssl3_write(SSL *s, int type, const unsigned char *buf,
651                          unsigned int len, int create_empty_fragment)
652         {
653         unsigned char *p,*plen;
654         int i,mac_size,clear=0;
655         int prefix_len=0;
```

```
656         int eivlen;
657         long align=0;
658         SSL3_RECORD *wr;
659         SSL3_BUFFER *wb=&(s->s3->wbuf);
660         SSL_SESSION *sess;


663         /* first check if there is a SSL3_BUFFER still being written
664          * out.  This will happen with non blocking IO */
665         if (wb->left != 0)
666                 return(ssl3_write_pending(s,type,buf,len));

668         /* If we have an alert to send, lets send it */
669         if (s->s3->alert_dispatch)
670                 {
671                 i=s->method->ssl_dispatch_alert(s);
672                 if (i <= 0)
673                         return(i);
674                 /* if it went, fall through and send more stuff */
675                 }

677         if (wb->buf == NULL)
678                 if (!ssl3_setup_write_buffer(s))
679                         return -1;

681         if (len == 0 && !create_empty_fragment)
682                 return 0;

684         wr= &(s->s3->wrec);
685         sess=s->session;

687         if (    (sess == NULL) ||
688                 (s->enc_write_ctx == NULL) ||
689                 (EVP_MD_CTX_md(s->write_hash) == NULL))
690                 {
691 #if 1
692                 clear=s->enc_write_ctx?0:1;     /* must be AEAD cipher */
693 #else
694                 clear=1;
695 #endif
696                 mac_size=0;
697                 }
698         else
699                 {
700                 mac_size=EVP_MD_CTX_size(s->write_hash);
701                 if (mac_size < 0)
702                         goto err;
703                 }

705         /* 'create_empty_fragment' is true only when this function calls itself
706         if (!clear && !create_empty_fragment && !s->s3->empty_fragment_done)
707                 {
708                 /* countermeasure against known-IV weakness in CBC ciphersuites
709                  * (see http://www.openssl.org/~bodo/tls-cbc.txt) */

711                 if (s->s3->need_empty_fragments && type == SSL3_RT_APPLICATION_D
712                         {
713                         /* recursive function call with 'create_empty_fragment'
714                          * this prepares and buffers the data for an empty fragm
715                          * (these 'prefix_len' bytes are sent out later
716                          * together with the actual payload) */
717                         prefix_len = do_ssl3_write(s, type, buf, 0, 1);
718                         if (prefix_len <= 0)
719                                 goto err;

721                         if (prefix_len >
```



```
722                 (SSL3_RT_HEADER_LENGTH + SSL3_RT_SEND_MAX_ENCRYPTED_OVERHEAD))
723                                 {
724                                 /* insufficient space */
725                                 SSLerr(SSL_F_DO_SSL3_WRITE, ERR_R_INTERNAL_ERROR
726                                 goto err;
727                                 }
728                         }
729
730                 s->s3->empty_fragment_done = 1;
731                 }

733         if (create_empty_fragment)
734                 {
735 #if defined(SSL3_ALIGN_PAYLOAD) && SSL3_ALIGN_PAYLOAD!=0
736                 /* extra fragment would be couple of cipher blocks,
737                  * which would be multiple of SSL3_ALIGN_PAYLOAD, so
738                  * if we want to align the real payload, then we can
739                  * just pretent we simply have two headers. */
740                 align = (long)wb->buf + 2*SSL3_RT_HEADER_LENGTH;
741                 align = (-align)&(SSL3_ALIGN_PAYLOAD-1);
742 #endif
743                 p = wb->buf + align;
744                 wb->offset  = align;
745                 }
746         else if (prefix_len)
747                 {
748                 p = wb->buf + wb->offset + prefix_len;
749                 }
750         else
751                 {
752 #if defined(SSL3_ALIGN_PAYLOAD) && SSL3_ALIGN_PAYLOAD!=0
753                 align = (long)wb->buf + SSL3_RT_HEADER_LENGTH;
754                 align = (-align)&(SSL3_ALIGN_PAYLOAD-1);
755 #endif
756                 p = wb->buf + align;
757                 wb->offset  = align;
758                 }

760         /* write the header */

762         *(p++)=type&0xff;
763         wr->type=type;

765         *(p++)=(s->version>>8);
766         /* Some servers hang if iniatial client hello is larger than 256
767          * bytes and record version number > TLS 1.0
768          */
769         if (s->state == SSL3_ST_CW_CLNT_HELLO_B
770                                 && !s->renegotiate
771                                 && TLS1_get_version(s) > TLS1_VERSION)
772                 *(p++) = 0x1;
773         else
774                 *(p++)=s->version&0xff;

776         /* field where we are to write out packet length */
777         plen=p;
778         p+=2;
779         /* Explicit IV length, block ciphers and TLS version 1.1 or later */
780         if (s->enc_write_ctx && s->version >= TLS1_1_VERSION)
781                 {
782                 int mode = EVP_CIPHER_CTX_mode(s->enc_write_ctx);
783                 if (mode == EVP_CIPH_CBC_MODE)
784                         {
785                         eivlen = EVP_CIPHER_CTX_iv_length(s->enc_write_ctx);
786                         if (eivlen <= 1)
787                                 eivlen = 0;
```

```
788                     }
789             /* Need explicit part of IV for GCM mode */
790             else if (mode == EVP_CIPH_GCM_MODE)
791                     eivlen = EVP_GCM_TLS_EXPLICIT_IV_LEN;
792             else
793                     eivlen = 0;
794             }
795     else
796             eivlen = 0;

798     /* lets setup the record stuff. */
799     wr->data=p + eivlen;
800     wr->length=(int)len;
801     wr->input=(unsigned char *)buf;

803     /* we now 'read' from wr->input, wr->length bytes into
804      * wr->data */

806     /* first we compress */
807     if (s->compress != NULL)
808             {
809             if (!ssl3_do_compress(s))
810                     {
811                     SSLerr(SSL_F_DO_SSL3_WRITE,SSL_R_COMPRESSION_FAILURE);
812                     goto err;
813                     }
814             }
815     else
816             {
817             memcpy(wr->data,wr->input,wr->length);
818             wr->input=wr->data;
819             }

821     /* we should still have the output to wr->data and the input
822      * from wr->input.  Length should be wr->length.
823      * wr->data still points in the wb->buf */

825     if (mac_size != 0)
826             {
827             if (s->method->ssl3_enc->mac(s,&(p[wr->length + eivlen]),1) < 0)
828                     goto err;
829             wr->length+=mac_size;
830             }

832     wr->input=p;
833     wr->data=p;

835     if (eivlen)
836             {
837     /*      if (RAND_pseudo_bytes(p, eivlen) <= 0)
838                     goto err; */
839             wr->length += eivlen;
840             }

842     /* ssl3_enc can only have an error on read */
843     s->method->ssl3_enc->enc(s,1);

845     /* record length after mac and block padding */
846     s2n(wr->length,plen);

848     /* we should now have
849      * wr->data pointing to the encrypted data, which is
850      * wr->length long */
851     wr->type=type; /* not needed but helps for debugging */
852     wr->length+=SSL3_RT_HEADER_LENGTH;
```



```
854     if (create_empty_fragment)
855             {
856             /* we are in a recursive call;
857              * just return the length, don't write out anything here
858              */
859             return wr->length;
860             }

862     /* now let's set up wb */
863     wb->left = prefix_len + wr->length;

865     /* memorize arguments so that ssl3_write_pending can detect bad write re
866     s->s3->wpend_tot=len;
867     s->s3->wpend_buf=buf;
868     s->s3->wpend_type=type;
869     s->s3->wpend_ret=len;

871     /* we now just need to write the buffer */
872     return ssl3_write_pending(s,type,buf,len);
873 err:
874     return -1;
875     }

877 /* if s->s3->wbuf.left != 0, we need to call this */
878 int ssl3_write_pending(SSL *s, int type, const unsigned char *buf,
879     unsigned int len)
880     {
881     int i;
882     SSL3_BUFFER *wb=&(s->s3->wbuf);

884 /* XXXX */
885     if ((s->s3->wpend_tot > (int)len)
886             || ((s->s3->wpend_buf != buf) &&
887                     !(s->mode & SSL_MODE_ACCEPT_MOVING_WRITE_BUFFER))
888             || (s->s3->wpend_type != type))
889             {
890             SSLerr(SSL_F_SSL3_WRITE_PENDING,SSL_R_BAD_WRITE_RETRY);
891             return(-1);
892             }

894     for (;;)
895             {
896             clear_sys_error();
897             if (s->wbio != NULL)
898                     {
899                     s->rwstate=SSL_WRITING;
900                     i=BIO_write(s->wbio,
901                             (char *)&(wb->buf[wb->offset]),
902                             (unsigned int)wb->left);
903                     }
904             else
905                     {
906                     SSLerr(SSL_F_SSL3_WRITE_PENDING,SSL_R_BIO_NOT_SET);
907                     i= -1;
908                     }
909             if (i == wb->left)
910                     {
911                     wb->left=0;
912                     wb->offset+=i;
913                     if (s->mode & SSL_MODE_RELEASE_BUFFERS &&
914                         SSL_version(s) != DTLS1_VERSION && SSL_version(s) !=
915                             ssl3_release_write_buffer(s);
916                     s->rwstate=SSL_NOTHING;
917                     return(s->s3->wpend_ret);
918                     }
919             else if (i <= 0) {
```

```
920                         if (s->version == DTLS1_VERSION ||
921                             s->version == DTLS1_BAD_VER) {
922                                 /* For DTLS, just drop it. That's kind of the wh
923                                    point in using a datagram service */
924                                 wb->left = 0;
925                         }
926                         return(i);
927                 }
928                 wb->offset+=i;
929                 wb->left-=i;
930                 }
931         }

933 /* Return up to 'len' payload bytes received in 'type' records.
934  * 'type' is one of the following:
935  *
936  *   -  SSL3_RT_HANDSHAKE (when ssl3_get_message calls us)
937  *   -  SSL3_RT_APPLICATION_DATA (when ssl3_read calls us)
938  *   -  0 (during a shutdown, no data has to be returned)
939  *
940  * If we don't have stored data to work from, read a SSL/TLS record first
941  * (possibly multiple records if we still don't have anything to return).
942  *
943  * This function must handle any surprises the peer may have for us, such as
944  * Alert records (e.g. close_notify), ChangeCipherSpec records (not really
945  * a surprise, but handled as if it were), or renegotiation requests.
946  * Also if record payloads contain fragments too small to process, we store
947  * them until there is enough for the respective protocol (the record protocol
948  * may use arbitrary fragmentation and even interleaving):
949  *     Change cipher spec protocol
950  *             just 1 byte needed, no need for keeping anything stored
951  *     Alert protocol
952  *             2 bytes needed (AlertLevel, AlertDescription)
953  *     Handshake protocol
954  *             4 bytes needed (HandshakeType, uint24 length) -- we just have
955  *             to detect unexpected Client Hello and Hello Request messages
956  *             here, anything else is handled by higher layers
957  *     Application data protocol
958  *             none of our business
959  */
960 int ssl3_read_bytes(SSL *s, int type, unsigned char *buf, int len, int peek)
961         {
962         int al,i,j,ret;
963         unsigned int n;
964         SSL3_RECORD *rr;
965         void (*cb)(const SSL *ssl,int type2,int val)=NULL;

967         if (s->s3->rbuf.buf == NULL) /* Not initialized yet */
968                 if (!ssl3_setup_read_buffer(s))
969                         return(-1);

971         if ((type && (type != SSL3_RT_APPLICATION_DATA) && (type != SSL3_RT_HAND
972             (peek && (type != SSL3_RT_APPLICATION_DATA)))
973                 {
974                 SSLerr(SSL_F_SSL3_READ_BYTES, ERR_R_INTERNAL_ERROR);
975                 return -1;
976                 }

978         if ((type == SSL3_RT_HANDSHAKE) && (s->s3->handshake_fragment_len > 0))
979                 /* (partially) satisfy request from storage */
980                 {
981                 unsigned char *src = s->s3->handshake_fragment;
982                 unsigned char *dst = buf;
983                 unsigned int k;

985                 /* peek == 0 */
```



```
986                 n = 0;
987                 while ((len > 0) && (s->s3->handshake_fragment_len > 0))
988                         {
989                         *dst++ = *src++;
990                         len--; s->s3->handshake_fragment_len--;
991                         n++;
992                         }
993                 /* move any remaining fragment bytes: */
994                 for (k = 0; k < s->s3->handshake_fragment_len; k++)
995                         s->s3->handshake_fragment[k] = *src++;
996                 return n;
997         }

999         /* Now s->s3->handshake_fragment_len == 0 if type == SSL3_RT_HANDSHAKE.

1001        if (!s->in_handshake && SSL_in_init(s))
1002                {
1003                /* type == SSL3_RT_APPLICATION_DATA */
1004                i=s->handshake_func(s);
1005                if (i < 0) return(i);
1006                if (i == 0)
1007                        {
1008                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_SSL_HANDSHAKE_FAILURE
1009                        return(-1);
1010                        }
1011                }
1012 start:
1013        s->rwstate=SSL_NOTHING;

1015        /* s->s3->rrec.type         - is the type of record
1016         * s->s3->rrec.data,    - data
1017         * s->s3->rrec.off,     - offset into 'data' for next read
1018         * s->s3->rrec.length,  - number of bytes. */
1019        rr = &(s->s3->rrec);

1021        /* get new packet if necessary */
1022        if ((rr->length == 0) || (s->rstate == SSL_ST_READ_BODY))
1023                {
1024                ret=ssl3_get_record(s);
1025                if (ret <= 0) return(ret);
1026                }

1028        /* we now have a packet which can be read and processed */

1030        if (s->s3->change_cipher_spec /* set when we receive ChangeCipherSpec,
1031                                       * reset by ssl3_get_finished */
1032                && (rr->type != SSL3_RT_HANDSHAKE))
1033                {
1034                al=SSL_AD_UNEXPECTED_MESSAGE;
1035                SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_DATA_BETWEEN_CCS_AND_FINISHED
1036                goto f_err;
1037                }

1039        /* If the other end has shut down, throw anything we read away
1040         * (even in 'peek' mode) */
1041        if (s->shutdown & SSL_RECEIVED_SHUTDOWN)
1042                {
1043                rr->length=0;
1044                s->rwstate=SSL_NOTHING;
1045                return(0);
1046                }


1049        if (type == rr->type) /* SSL3_RT_APPLICATION_DATA or SSL3_RT_HANDSHAKE *
1050                {
1051                /* make sure that we are not getting application data when we
```

```
1052                 * are doing a handshake for the first time */
1053                if (SSL_in_init(s) && (type == SSL3_RT_APPLICATION_DATA) &&
1054                        (s->enc_read_ctx == NULL))
1055                        {
1056                        al=SSL_AD_UNEXPECTED_MESSAGE;
1057                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_APP_DATA_IN_HANDSHAKE
1058                        goto f_err;
1059                        }

1061                if (len <= 0) return(len);

1063                if ((unsigned int)len > rr->length)
1064                        n = rr->length;
1065                else
1066                        n = (unsigned int)len;

1068                memcpy(buf,&(rr->data[rr->off]),n);
1069                if (!peek)
1070                        {
1071                        rr->length-=n;
1072                        rr->off+=n;
1073                        if (rr->length == 0)
1074                                {
1075                                s->rstate=SSL_ST_READ_HEADER;
1076                                rr->off=0;
1077                                if (s->mode & SSL_MODE_RELEASE_BUFFERS && s->s3-
1078                                        ssl3_release_read_buffer(s);
1079                                }
1080                        }
1081                return(n);
1082                }


1085        /* If we get here, then type != rr->type; if we have a handshake
1086         * message, then it was unexpected (Hello Request or Client Hello). */

1088        /* In case of record types for which we have 'fragment' storage,
1089         * fill that so that we can process the data at a fixed place.
1090         */
1091                {
1092                unsigned int dest_maxlen = 0;
1093                unsigned char *dest = NULL;
1094                unsigned int *dest_len = NULL;

1096                if (rr->type == SSL3_RT_HANDSHAKE)
1097                        {
1098                        dest_maxlen = sizeof s->s3->handshake_fragment;
1099                        dest = s->s3->handshake_fragment;
1100                        dest_len = &s->s3->handshake_fragment_len;
1101                        }
1102                else if (rr->type == SSL3_RT_ALERT)
1103                        {
1104                        dest_maxlen = sizeof s->s3->alert_fragment;
1105                        dest = s->s3->alert_fragment;
1106                        dest_len = &s->s3->alert_fragment_len;
1107                        }
1108 #ifndef OPENSSL_NO_HEARTBEATS
1109                else if (rr->type == TLS1_RT_HEARTBEAT)
1110                        {
1111                        tls1_process_heartbeat(s);

1113                        /* Exit and notify application to read again */
1114                        rr->length = 0;
1115                        s->rwstate=SSL_READING;
1116                        BIO_clear_retry_flags(SSL_get_rbio(s));
1117                        BIO_set_retry_read(SSL_get_rbio(s));
```



```
1118                        return(-1);
1119                        }
1120 #endif

1122                if (dest_maxlen > 0)
1123                        {
1124                        n = dest_maxlen - *dest_len; /* available space in 'dest
1125                        if (rr->length < n)
1126                                n = rr->length; /* available bytes */

1128                        /* now move 'n' bytes: */
1129                        while (n-- > 0)
1130                                {
1131                                dest[(*dest_len)++] = rr->data[rr->off++];
1132                                rr->length--;
1133                                }

1135                        if (*dest_len < dest_maxlen)
1136                                goto start; /* fragment was too small */
1137                        }
1138                }

1140        /* s->s3->handshake_fragment_len == 4  iff  rr->type == SSL3_RT_HANDSHAK
1141         * s->s3->alert_fragment_len == 2      iff  rr->type == SSL3_RT_ALERT.
1142         * (Possibly rr is 'empty' now, i.e. rr->length may be 0.) */

1144        /* If we are a client, check for an incoming 'Hello Request': */
1145        if ((!s->server) &&
1146                (s->s3->handshake_fragment_len >= 4) &&
1147                (s->s3->handshake_fragment[0] == SSL3_MT_HELLO_REQUEST) &&
1148                (s->session != NULL) && (s->session->cipher != NULL))
1149                {
1150                s->s3->handshake_fragment_len = 0;

1152                if ((s->s3->handshake_fragment[1] != 0) ||
1153                        (s->s3->handshake_fragment[2] != 0) ||
1154                        (s->s3->handshake_fragment[3] != 0))
1155                        {
1156                        al=SSL_AD_DECODE_ERROR;
1157                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_BAD_HELLO_REQUEST);
1158                        goto f_err;
1159                        }

1161                if (s->msg_callback)
1162                        s->msg_callback(0, s->version, SSL3_RT_HANDSHAKE, s->s3-

1164                if (SSL_is_init_finished(s) &&
1165                        !(s->s3->flags & SSL3_FLAGS_NO_RENEGOTIATE_CIPHERS) &&
1166                        !s->s3->renegotiate)
1167                        {
1168                        ssl3_renegotiate(s);
1169                        if (ssl3_renegotiate_check(s))
1170                                {
1171                                i=s->handshake_func(s);
1172                                if (i < 0) return(i);
1173                                if (i == 0)
1174                                        {
1175                                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_SSL_H
1176                                        return(-1);
1177                                        }

1179                                if (!(s->mode & SSL_MODE_AUTO_RETRY))
1180                                        {
1181                                        if (s->s3->rbuf.left == 0) /* no read-ah
1182                                                {
1183                                                BIO *bio;
```

```
1184                                            /* In the case where we try to r
1185                                             * but we trigger an SSL handsha
1186                                             * the retry option set.  Otherw
1187                                             * cause nasty problems in the b
1188                                            s->rwstate=SSL_READING;
1189                                            bio=SSL_get_rbio(s);
1190                                            BIO_clear_retry_flags(bio);
1191                                            BIO_set_retry_read(bio);
1192                                            return(-1);
1193                                            }
1194                                    }
1195                            }
1196                    }
1197            /* we either finished a handshake or ignored the request,
1198             * now try again to obtain the (application) data we were asked
1199            goto start;
1200            }
1201        /* If we are a server and get a client hello when renegotiation isn't
1202         * allowed send back a no renegotiation alert and carry on.
1203         * WARNING: experimental code, needs reviewing (steve)
1204         */
1205        if (s->server &&
1206                SSL_is_init_finished(s) &&
1207                !s->s3->send_connection_binding &&
1208                (s->version > SSL3_VERSION) &&
1209                (s->s3->handshake_fragment_len >= 4) &&
1210                (s->s3->handshake_fragment[0] == SSL3_MT_CLIENT_HELLO) &&
1211                (s->session != NULL) && (s->session->cipher != NULL) &&
1212                !(s->ctx->options & SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION))
1213
1214                {
1215                /*s->s3->handshake_fragment_len = 0;*/
1216                rr->length = 0;
1217                ssl3_send_alert(s,SSL3_AL_WARNING, SSL_AD_NO_RENEGOTIATION);
1218                goto start;
1219                }
1220        if (s->s3->alert_fragment_len >= 2)
1221                {
1222                int alert_level = s->s3->alert_fragment[0];
1223                int alert_descr = s->s3->alert_fragment[1];

1225                s->s3->alert_fragment_len = 0;

1227                if (s->msg_callback)
1228                        s->msg_callback(0, s->version, SSL3_RT_ALERT, s->s3->ale

1230                if (s->info_callback != NULL)
1231                        cb=s->info_callback;
1232                else if (s->ctx->info_callback != NULL)
1233                        cb=s->ctx->info_callback;

1235                if (cb != NULL)
1236                        {
1237                        j = (alert_level << 8) | alert_descr;
1238                        cb(s, SSL_CB_READ_ALERT, j);
1239                        }

1241                if (alert_level == 1) /* warning */
1242                        {
1243                        s->s3->warn_alert = alert_descr;
1244                        if (alert_descr == SSL_AD_CLOSE_NOTIFY)
1245                                {
1246                                s->shutdown |= SSL_RECEIVED_SHUTDOWN;
1247                                return(0);
1248                                }
1249                        /* This is a warning but we receive it if we requested
```



```
1250                         * renegotiation and the peer denied it. Terminate with
1251                         * a fatal alert because if application tried to
1252                         * renegotiatie it presumably had a good reason and
1253                         * expects it to succeed.
1254                         *
1255                         * In future we might have a renegotiation where we
1256                         * don't care if the peer refused it where we carry on.
1257                         */
1258                        else if (alert_descr == SSL_AD_NO_RENEGOTIATION)
1259                                {
1260                                al = SSL_AD_HANDSHAKE_FAILURE;
1261                                SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_NO_RENEGOTIAT
1262                                goto f_err;
1263                                }
1264 #ifdef SSL_AD_MISSING_SRP_USERNAME
1265                        else if (alert_descr == SSL_AD_MISSING_SRP_USERNAME)
1266                                return(0);
1267 #endif
1268                        }
1269                else if (alert_level == 2) /* fatal */
1270                        {
1271                        char tmp[16];

1273                        s->rwstate=SSL_NOTHING;
1274                        s->s3->fatal_alert = alert_descr;
1275                        SSLerr(SSL_F_SSL3_READ_BYTES, SSL_AD_REASON_OFFSET + ale
1276                        BIO_snprintf(tmp,sizeof tmp,"%d",alert_descr);
1277                        ERR_add_error_data(2,"SSL alert number ",tmp);
1278                        s->shutdown|=SSL_RECEIVED_SHUTDOWN;
1279                        SSL_CTX_remove_session(s->ctx,s->session);
1280                        return(0);
1281                        }
1282                else
1283                        {
1284                        al=SSL_AD_ILLEGAL_PARAMETER;
1285                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_UNKNOWN_ALERT_TYPE);
1286                        goto f_err;
1287                        }

1289                goto start;
1290                }

1292        if (s->shutdown & SSL_SENT_SHUTDOWN) /* but we have not received a shutd
1293                {
1294                s->rwstate=SSL_NOTHING;
1295                rr->length=0;
1296                return(0);
1297                }

1299        if (rr->type == SSL3_RT_CHANGE_CIPHER_SPEC)
1300                {
1301                /* 'Change Cipher Spec' is just a single byte, so we know
1302                 * exactly what the record payload has to look like */
1303                if (    (rr->length != 1) || (rr->off != 0) ||
1304                        (rr->data[0] != SSL3_MT_CCS))
1305                        {
1306                        al=SSL_AD_ILLEGAL_PARAMETER;
1307                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_BAD_CHANGE_CIPHER_SPE
1308                        goto f_err;
1309                        }

1311                /* Check we have a cipher to change to */
1312                if (s->s3->tmp.new_cipher == NULL)
1313                        {
1314                        al=SSL_AD_UNEXPECTED_MESSAGE;
1315                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_CCS_RECEIVED_EARLY);
```

```
1316                        goto f_err;
1317                        }

1319                if (!(s->s3->flags & SSL3_FLAGS_CCS_OK))
1320                        {
1321                        al=SSL_AD_UNEXPECTED_MESSAGE;
1322                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_CCS_RECEIVED_EARLY);
1323                        goto f_err;
1324                        }

1326                s->s3->flags &= ~SSL3_FLAGS_CCS_OK;

1328                rr->length=0;

1330                if (s->msg_callback)
1331                        s->msg_callback(0, s->version, SSL3_RT_CHANGE_CIPHER_SPE

1333                s->s3->change_cipher_spec=1;
1334                if (!ssl3_do_change_cipher_spec(s))
1335                        goto err;
1336                else
1337                        goto start;
1338                }

1340        /* Unexpected handshake message (Client Hello, or protocol violation) */
1341        if ((s->s3->handshake_fragment_len >= 4) &&     !s->in_handshake)
1342                {
1343                if (((s->state&SSL_ST_MASK) == SSL_ST_OK) &&
1344                        !(s->s3->flags & SSL3_FLAGS_NO_RENEGOTIATE_CIPHERS))
1345                        {
1346 #if 0 /* worked only because C operator preferences are not as expected (and
1347        * because this is not really needed for clients except for detecting
1348        * protocol violations): */
1349                        s->state=SSL_ST_BEFORE|(s->server)
1350                                ?SSL_ST_ACCEPT
1351                                :SSL_ST_CONNECT;
1352 #else
1353                        s->state = s->server ? SSL_ST_ACCEPT : SSL_ST_CONNECT;
1354 #endif
1355                        s->renegotiate=1;
1356                        s->new_session=1;
1357                        }
1358                i=s->handshake_func(s);
1359                if (i < 0) return(i);
1360                if (i == 0)
1361                        {
1362                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_SSL_HANDSHAKE_FAILURE
1363                        return(-1);
1364                        }

1366                if (!(s->mode & SSL_MODE_AUTO_RETRY))
1367                        {
1368                        if (s->s3->rbuf.left == 0) /* no read-ahead left? */
1369                                {
1370                                BIO *bio;
1371                                /* In the case where we try to read application
1372                                 * but we trigger an SSL handshake, we return -1
1373                                 * the retry option set.  Otherwise renegotiatio
1374                                 * cause nasty problems in the blocking world */
1375                                s->rwstate=SSL_READING;
1376                                bio=SSL_get_rbio(s);
1377                                BIO_clear_retry_flags(bio);
1378                                BIO_set_retry_read(bio);
1379                                return(-1);
1380                                }
1381                        }
```



```
1382                goto start;
1383                }

1385        switch (rr->type)
1386                {
1387        default:
1388 #ifndef OPENSSL_NO_TLS
1389                /* TLS up to v1.1 just ignores unknown message types:
1390                 * TLS v1.2 give an unexpected message alert.
1391                 */
1392                if (s->version >= TLS1_VERSION && s->version <= TLS1_1_VERSION)
1393                        {
1394                        rr->length = 0;
1395                        goto start;
1396                        }
1397 #endif
1398                al=SSL_AD_UNEXPECTED_MESSAGE;
1399                SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_UNEXPECTED_RECORD);
1400                goto f_err;
1401        case SSL3_RT_CHANGE_CIPHER_SPEC:
1402        case SSL3_RT_ALERT:
1403        case SSL3_RT_HANDSHAKE:
1404                /* we already handled all of these, with the possible exception
1405                 * of SSL3_RT_HANDSHAKE when s->in_handshake is set, but that
1406                 * should not happen when type != rr->type */
1407                al=SSL_AD_UNEXPECTED_MESSAGE;
1408                SSLerr(SSL_F_SSL3_READ_BYTES,ERR_R_INTERNAL_ERROR);
1409                goto f_err;
1410        case SSL3_RT_APPLICATION_DATA:
1411                /* At this point, we were expecting handshake data,
1412                 * but have application data.  If the library was
1413                 * running inside ssl3_read() (i.e. in_read_app_data
1414                 * is set) and it makes sense to read application data
1415                 * at this point (session renegotiation not yet started),
1416                 * we will indulge it.
1417                 */
1418                if (s->s3->in_read_app_data &&
1419                        (s->s3->total_renegotiations != 0) &&
1420                        ((
1421                                (s->state & SSL_ST_CONNECT) &&
1422                                (s->state >= SSL3_ST_CW_CLNT_HELLO_A) &&
1423                                (s->state <= SSL3_ST_CR_SRVR_HELLO_A)
1424                                ) || (
1425                                        (s->state & SSL_ST_ACCEPT) &&
1426                                        (s->state <= SSL3_ST_SW_HELLO_REQ_A) &&
1427                                        (s->state >= SSL3_ST_SR_CLNT_HELLO_A)
1428                                        )
1429                                ))
1430                        {
1431                        s->s3->in_read_app_data=2;
1432                        return(-1);
1433                        }
1434                else
1435                        {
1436                        al=SSL_AD_UNEXPECTED_MESSAGE;
1437                        SSLerr(SSL_F_SSL3_READ_BYTES,SSL_R_UNEXPECTED_RECORD);
1438                        goto f_err;
1439                        }
1440                }
1441        /* not reached */

1443 f_err:
1444        ssl3_send_alert(s,SSL3_AL_FATAL,al);
1445 err:
1446        return(-1);
1447        }
```

```
1449 int ssl3_do_change_cipher_spec(SSL *s)
1450         {
1451         int i;
1452         const char *sender;
1453         int slen;

1455         if (s->state & SSL_ST_ACCEPT)
1456                 i=SSL3_CHANGE_CIPHER_SERVER_READ;
1457         else
1458                 i=SSL3_CHANGE_CIPHER_CLIENT_READ;

1460         if (s->s3->tmp.key_block == NULL)
1461                 {
1462                 if (s->session == NULL || s->session->master_key_length == 0)
1463                         {
1464                         /* might happen if dtls1_read_bytes() calls this */
1465                         SSLerr(SSL_F_SSL3_DO_CHANGE_CIPHER_SPEC,SSL_R_CCS_RECEIV
1466                         return (0);
1467                         }

1469                 s->session->cipher=s->s3->tmp.new_cipher;
1470                 if (!s->method->ssl3_enc->setup_key_block(s)) return(0);
1471                 }

1473         if (!s->method->ssl3_enc->change_cipher_state(s,i))
1474                 return(0);

1476         /* we have to record the message digest at
1477          * this point so we can get it before we read
1478          * the finished message */
1479         if (s->state & SSL_ST_CONNECT)
1480                 {
1481                 sender=s->method->ssl3_enc->server_finished_label;
1482                 slen=s->method->ssl3_enc->server_finished_label_len;
1483                 }
1484         else
1485                 {
1486                 sender=s->method->ssl3_enc->client_finished_label;
1487                 slen=s->method->ssl3_enc->client_finished_label_len;
1488                 }

1490         i = s->method->ssl3_enc->final_finish_mac(s,
1491                 sender,slen,s->s3->tmp.peer_finish_md);
1492         if (i == 0)
1493                 {
1494                 SSLerr(SSL_F_SSL3_DO_CHANGE_CIPHER_SPEC, ERR_R_INTERNAL_ERROR);
1495                 return 0;
1496                 }
1497         s->s3->tmp.peer_finish_md_len = i;

1499         return(1);
1500         }

1502 int ssl3_send_alert(SSL *s, int level, int desc)
1503         {
1504         /* Map tls/ssl alert value to correct one */
1505         desc=s->method->ssl3_enc->alert_value(desc);
1506         if (s->version == SSL3_VERSION && desc == SSL_AD_PROTOCOL_VERSION)
1507                 desc = SSL_AD_HANDSHAKE_FAILURE; /* SSL 3.0 does not have protoc
1508         if (desc < 0) return -1;
1509         /* If a fatal one, remove from cache */
1510         if ((level == 2) && (s->session != NULL))
1511                 SSL_CTX_remove_session(s->ctx,s->session);

1513         s->s3->alert_dispatch=1;
```



```
1514         s->s3->send_alert[0]=level;
1515         s->s3->send_alert[1]=desc;
1516         if (s->s3->wbuf.left == 0) /* data still being written out? */
1517                 return s->method->ssl_dispatch_alert(s);
1518         /* else data is still being written out, we will get written
1519          * some time in the future */
1520         return -1;
1521         }

1523 int ssl3_dispatch_alert(SSL *s)
1524         {
1525         int i,j;
1526         void (*cb)(const SSL *ssl,int type,int val)=NULL;

1528         s->s3->alert_dispatch=0;
1529         i = do_ssl3_write(s, SSL3_RT_ALERT, &s->s3->send_alert[0], 2, 0);
1530         if (i <= 0)
1531                 {
1532                 s->s3->alert_dispatch=1;
1533                 }
1534         else
1535                 {
1536                 /* Alert sent to BIO.  If it is important, flush it now.
1537                  * If the message does not get sent due to non-blocking IO,
1538                  * we will not worry too much. */
1539                 if (s->s3->send_alert[0] == SSL3_AL_FATAL)
1540                         (void)BIO_flush(s->wbio);

1542                 if (s->msg_callback)
1543                         s->msg_callback(1, s->version, SSL3_RT_ALERT, s->s3->sen

1545                 if (s->info_callback != NULL)
1546                         cb=s->info_callback;
1547                 else if (s->ctx->info_callback != NULL)
1548                         cb=s->ctx->info_callback;

1550                 if (cb != NULL)
1551                         {
1552                         j=(s->s3->send_alert[0]<<8)|s->s3->send_alert[1];
1553                         cb(s,SSL_CB_WRITE_ALERT,j);
1554                         }
1555                 }
1556         return(i);
1557         }
1558 #endif /* ! codereview */
```

```
**********************************************************
   91853 Wed Aug 13 19:53:39 2014
new/usr/src/lib/openssl/libsunw_ssl/s3_srvr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/s3_srvr.c -*- mode:C; c-file-style: "eay" -*- */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 *
 * Portions of the attached software ("Contribution") are developed by
 * SUN MICROSYSTEMS, INC., and are contributed to the OpenSSL project.
 *
 * The Contribution is licensed pursuant to the OpenSSL open source
 * license provided above.
 *
 * ECC cipher suite support in OpenSSL originally written by
 * Vipul Gupta and Sumit Gupta of Sun Microsystems Laboratories.
 *
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
```

```
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#define REUSE_CIPHER_BUG
#define NETSCAPE_HANG_BUG

#include <stdio.h>
#include "ssl_locl.h"
#include "kssl_lcl.h"
#include <openssl/buffer.h>
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/hmac.h>
#include <openssl/x509.h>
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#include <openssl/bn.h>
#ifndef OPENSSL_NO_KRB5
#include <openssl/krb5_asn.h>
#endif
#include <openssl/md5.h>

static const SSL_METHOD *ssl3_get_server_method(int ver);

static const SSL_METHOD *ssl3_get_server_method(int ver)
        {
        if (ver == SSL3_VERSION)
                return(SSLv3_server_method());
        else
                return(NULL);
        }

#ifndef OPENSSL_NO_SRP
static int ssl_check_srp_ext_ClientHello(SSL *s, int *al)
        {
        int ret = SSL_ERROR_NONE;

        *al = SSL_AD_UNRECOGNIZED_NAME;

        if ((s->s3->tmp.new_cipher->algorithm_mkey & SSL_kSRP) &&
            (s->srp_ctx.TLS_ext_srp_username_callback != NULL))
                {
                if(s->srp_ctx.login == NULL)
                        {
```



```
                        /* RFC 5054 says SHOULD reject,
                           we do so if There is no srp login name */
                        ret = SSL3_AL_FATAL;
                        *al = SSL_AD_UNKNOWN_PSK_IDENTITY;
                        }
                else
                        {
                        ret = SSL_srp_server_param_with_username(s,al);
                        }
                }
        return ret;
        }
#endif

IMPLEMENT_ssl3_meth_func(SSLv3_server_method,
                        ssl3_accept,
                        ssl_undefined_function,
                        ssl3_get_server_method)

int ssl3_accept(SSL *s)
        {
        BUF_MEM *buf;
        unsigned long alg_k,Time=(unsigned long)time(NULL);
        void (*cb)(const SSL *ssl,int type,int val)=NULL;
        int ret= -1;
        int new_state,state,skip=0;

        RAND_add(&Time,sizeof(Time),0);
        ERR_clear_error();
        clear_sys_error();

        if (s->info_callback != NULL)
                cb=s->info_callback;
        else if (s->ctx->info_callback != NULL)
                cb=s->ctx->info_callback;

        /* init things to blank */
        s->in_handshake++;
        if (!SSL_in_init(s) || SSL_in_before(s)) SSL_clear(s);

        if (s->cert == NULL)
                {
                SSLerr(SSL_F_SSL3_ACCEPT,SSL_R_NO_CERTIFICATE_SET);
                return(-1);
                }

#ifndef OPENSSL_NO_HEARTBEATS
        /* If we're awaiting a HeartbeatResponse, pretend we
         * already got and don't await it anymore, because
         * Heartbeats don't make sense during handshakes anyway.
         */
        if (s->tlsext_hb_pending)
                {
                s->tlsext_hb_pending = 0;
                s->tlsext_hb_seq++;
                }
#endif

        for (;;)
                {
                state=s->state;

                switch (s->state)
                        {
                case SSL_ST_RENEGOTIATE:
                        s->renegotiate=1;
```

```
260                         /* s->state=SSL_ST_ACCEPT; */

262                 case SSL_ST_BEFORE:
263                 case SSL_ST_ACCEPT:
264                 case SSL_ST_BEFORE|SSL_ST_ACCEPT:
265                 case SSL_ST_OK|SSL_ST_ACCEPT:

267                         s->server=1;
268                         if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_START,1);

270                         if ((s->version>>8) != 3)
271                                 {
272                                 SSLerr(SSL_F_SSL3_ACCEPT, ERR_R_INTERNAL_ERROR);
273                                 return -1;
274                                 }
275                         s->type=SSL_ST_ACCEPT;

277                         if (s->init_buf == NULL)
278                                 {
279                                 if ((buf=BUF_MEM_new()) == NULL)
280                                         {
281                                         ret= -1;
282                                         goto end;
283                                         }
284                                 if (!BUF_MEM_grow(buf,SSL3_RT_MAX_PLAIN_LENGTH))
285                                         {
286                                         ret= -1;
287                                         goto end;
288                                         }
289                                 s->init_buf=buf;
290                                 }

292                         if (!ssl3_setup_buffers(s))
293                                 {
294                                 ret= -1;
295                                 goto end;
296                                 }

298                         s->init_num=0;
299                         s->s3->flags &= ~SSL3_FLAGS_SGC_RESTART_DONE;

301                         if (s->state != SSL_ST_RENEGOTIATE)
302                                 {
303                                 /* Ok, we now need to push on a buffering BIO so
304                                  * the output is sent in a way that TCP likes :-
305                                  */
306                                 if (!ssl_init_wbio_buffer(s,1)) { ret= -1; goto
307
308                                 ssl3_init_finished_mac(s);
309                                 s->state=SSL3_ST_SR_CLNT_HELLO_A;
310                                 s->ctx->stats.sess_accept++;
311                                 }
312                         else if (!s->s3->send_connection_binding &&
313                                 !(s->options & SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGO
314                                 {
315                                 /* Server attempting to renegotiate with
316                                  * client that doesn't support secure
317                                  * renegotiation.
318                                  */
319                                 SSLerr(SSL_F_SSL3_ACCEPT, SSL_R_UNSAFE_LEGACY_RE
320                                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_HANDSHAKE
321                                 ret = -1;
322                                 goto end;
323                                 }
324                         else
325                                 {
```



```
326                                 /* s->state == SSL_ST_RENEGOTIATE,
327                                  * we will just send a HelloRequest */
328                                 s->ctx->stats.sess_accept_renegotiate++;
329                                 s->state=SSL3_ST_SW_HELLO_REQ_A;
330                                 }
331                         break;

333                 case SSL3_ST_SW_HELLO_REQ_A:
334                 case SSL3_ST_SW_HELLO_REQ_B:

336                         s->shutdown=0;
337                         ret=ssl3_send_hello_request(s);
338                         if (ret <= 0) goto end;
339                         s->s3->tmp.next_state=SSL3_ST_SW_HELLO_REQ_C;
340                         s->state=SSL3_ST_SW_FLUSH;
341                         s->init_num=0;

343                         ssl3_init_finished_mac(s);
344                         break;

346                 case SSL3_ST_SW_HELLO_REQ_C:
347                         s->state=SSL_ST_OK;
348                         break;

350                 case SSL3_ST_SR_CLNT_HELLO_A:
351                 case SSL3_ST_SR_CLNT_HELLO_B:
352                 case SSL3_ST_SR_CLNT_HELLO_C:

354                         s->shutdown=0;
355                         if (s->rwstate != SSL_X509_LOOKUP)
356                         {
357                                 ret=ssl3_get_client_hello(s);
358                                 if (ret <= 0) goto end;
359                         }
360 #ifndef OPENSSL_NO_SRP
361                         {
362                         int al;
363                         if ((ret = ssl_check_srp_ext_ClientHello(s,&al))  < 0)
364                                 {
365                                 /* callback indicates firther work to be
366                                 s->rwstate=SSL_X509_LOOKUP;
367                                 goto end;
368                                 }
369                         if (ret != SSL_ERROR_NONE)
370                                 {
371                                 ssl3_send_alert(s,SSL3_AL_FATAL,al);
372                                 /* This is not really an error but the only mean
373                                    for a client to detect whether srp is support
374                                    if (al != TLS1_AD_UNKNOWN_PSK_IDENTITY)
375                                         SSLerr(SSL_F_SSL3_ACCEPT,SSL_R_CLIENTHEL
376                                 ret = SSL_TLSEXT_ERR_ALERT_FATAL;
377                                 ret= -1;
378                                 goto end;
379                                 }
380                         }
381 #endif
382
383                         s->renegotiate = 2;
384                         s->state=SSL3_ST_SW_SRVR_HELLO_A;
385                         s->init_num=0;
386                         break;

388                 case SSL3_ST_SW_SRVR_HELLO_A:
389                 case SSL3_ST_SW_SRVR_HELLO_B:
390                         ret=ssl3_send_server_hello(s);
391                         if (ret <= 0) goto end;
```

```
392 #ifndef OPENSSL_NO_TLSEXT
393                         if (s->hit)
394                                 {
395                                 if (s->tlsext_ticket_expected)
396                                         s->state=SSL3_ST_SW_SESSION_TICKET_A;
397                                 else
398                                         s->state=SSL3_ST_SW_CHANGE_A;
399                                 }
400 #else
401                         if (s->hit)
402                                         s->state=SSL3_ST_SW_CHANGE_A;
403 #endif
404                         else
405                                 s->state=SSL3_ST_SW_CERT_A;
406                         s->init_num=0;
407                         break;

409                 case SSL3_ST_SW_CERT_A:
410                 case SSL3_ST_SW_CERT_B:
411                         /* Check if it is anon DH or anon ECDH, */
412                         /* normal PSK or KRB5 or SRP */
413                         if (!(s->s3->tmp.new_cipher->algorithm_auth & SSL_aNULL)
414                                 && !(s->s3->tmp.new_cipher->algorithm_mkey & SSL
415                                 && !(s->s3->tmp.new_cipher->algorithm_auth & SSL
416                                 {
417                                 ret=ssl3_send_server_certificate(s);
418                                 if (ret <= 0) goto end;
419 #ifndef OPENSSL_NO_TLSEXT
420                                 if (s->tlsext_status_expected)
421                                         s->state=SSL3_ST_SW_CERT_STATUS_A;
422                                 else
423                                         s->state=SSL3_ST_SW_KEY_EXCH_A;
424                                 }
425                         else
426                                 {
427                                 skip = 1;
428                                 s->state=SSL3_ST_SW_KEY_EXCH_A;
429                                 }
430 #else
431                                 }
432                         else
433                                 skip=1;

435                         s->state=SSL3_ST_SW_KEY_EXCH_A;
436 #endif
437                         s->init_num=0;
438                         break;

440                 case SSL3_ST_SW_KEY_EXCH_A:
441                 case SSL3_ST_SW_KEY_EXCH_B:
442                         alg_k = s->s3->tmp.new_cipher->algorithm_mkey;

444                         /* clear this, it may get reset by
445                          * send_server_key_exchange */
446                         if ((s->options & SSL_OP_EPHEMERAL_RSA)
447 #ifndef OPENSSL_NO_KRB5
448                                 && !(alg_k & SSL_kKRB5)
449 #endif /* OPENSSL_NO_KRB5 */
450                                 )
451                                 /* option SSL_OP_EPHEMERAL_RSA sends temporary R
452                                  * even when forbidden by protocol specs
453                                  * (handshake may fail as clients are not requir
454                                  * be able to handle this) */
455                                 s->s3->tmp.use_rsa_tmp=1;
456                         else
457                                 s->s3->tmp.use_rsa_tmp=0;
```



```
460                         /* only send if a DH key exchange, fortezza or
461                          * RSA but we have a sign only certificate
462                          *
463                          * PSK: may send PSK identity hints
464                          *
465                          * For ECC ciphersuites, we send a serverKeyExchange
466                          * message only if the cipher suite is either
467                          * ECDH-anon or ECDHE. In other cases, the
468                          * server certificate contains the server's
469                          * public key for key exchange.
470                          */
471                         if (s->s3->tmp.use_rsa_tmp
472                         /* PSK: send ServerKeyExchange if PSK identity
473                          * hint if provided */
474 #ifndef OPENSSL_NO_PSK
475                             || ((alg_k & SSL_kPSK) && s->ctx->psk_identity_hint)
476 #endif
477 #ifndef OPENSSL_NO_SRP
478                             /* SRP: send ServerKeyExchange */
479                             || (alg_k & SSL_kSRP)
480 #endif
481                             || (alg_k & (SSL_kDHr|SSL_kDHd|SSL_kEDH))
482                             || (alg_k & SSL_kEECDH)
483                             || ((alg_k & SSL_kRSA)
484                                 && (s->cert->pkeys[SSL_PKEY_RSA_ENC].privatekey
485                                     || (SSL_C_IS_EXPORT(s->s3->tmp.new_cipher)
486                                         && EVP_PKEY_size(s->cert->pkeys[SSL_PKEY
487                                         )
488                                     )
489                                 )
490                             )
491                                 {
492                                 ret=ssl3_send_server_key_exchange(s);
493                                 if (ret <= 0) goto end;
494                                 }
495                         else
496                                 skip=1;

498                         s->state=SSL3_ST_SW_CERT_REQ_A;
499                         s->init_num=0;
500                         break;

502                 case SSL3_ST_SW_CERT_REQ_A:
503                 case SSL3_ST_SW_CERT_REQ_B:
504                         if (/* don't request cert unless asked for it: */
505                                 !(s->verify_mode & SSL_VERIFY_PEER) ||
506                                 /* if SSL_VERIFY_CLIENT_ONCE is set,
507                                  * don't request cert during re-negotiation: */
508                                 ((s->session->peer != NULL) &&
509                                  (s->verify_mode & SSL_VERIFY_CLIENT_ONCE)) ||
510                                 /* never request cert in anonymous ciphersuites
511                                  * (see section "Certificate request" in SSL 3 d
512                                  * and in RFC 2246): */
513                                 ((s->s3->tmp.new_cipher->algorithm_auth & SSL_aN
514                                  /* ... except when the application insists on v
515                                   * (against the specs, but s3_clnt.c accepts th
516                                   !(s->verify_mode & SSL_VERIFY_FAIL_IF_NO_PEER_C
517                                   /* never request cert in Kerberos ciphersuites
518                                 (s->s3->tmp.new_cipher->algorithm_auth & SSL_aKR
519                                 /* With normal PSK Certificates and
520                                  * Certificate Requests are omitted */
521                                 || (s->s3->tmp.new_cipher->algorithm_mkey & SSL_
522                                 {
523                                 /* no cert request */
```

```
524                                 skip=1;
525                                 s->s3->tmp.cert_request=0;
526                                 s->state=SSL3_ST_SW_SRVR_DONE_A;
527                                 if (s->s3->handshake_buffer)
528                                         if (!ssl3_digest_cached_records(s))
529                                                 return -1;
530                                 }
531                         else
532                                 {
533                                 s->s3->tmp.cert_request=1;
534                                 ret=ssl3_send_certificate_request(s);
535                                 if (ret <= 0) goto end;
536 #ifndef NETSCAPE_HANG_BUG
537                                 s->state=SSL3_ST_SW_SRVR_DONE_A;
538 #else
539                                 s->state=SSL3_ST_SW_FLUSH;
540                                 s->s3->tmp.next_state=SSL3_ST_SR_CERT_A;
541 #endif
542                                 s->init_num=0;
543                                 }
544                         break;

546                 case SSL3_ST_SW_SRVR_DONE_A:
547                 case SSL3_ST_SW_SRVR_DONE_B:
548                         ret=ssl3_send_server_done(s);
549                         if (ret <= 0) goto end;
550                         s->s3->tmp.next_state=SSL3_ST_SR_CERT_A;
551                         s->state=SSL3_ST_SW_FLUSH;
552                         s->init_num=0;
553                         break;
554
555                 case SSL3_ST_SW_FLUSH:

557                         /* This code originally checked to see if
558                          * any data was pending using BIO_CTRL_INFO
559                          * and then flushed. This caused problems
560                          * as documented in PR#1939. The proposed
561                          * fix doesn't completely resolve this issue
562                          * as buggy implementations of BIO_CTRL_PENDING
563                          * still exist. So instead we just flush
564                          * unconditionally.
565                          */

567                         s->rwstate=SSL_WRITING;
568                         if (BIO_flush(s->wbio) <= 0)
569                                 {
570                                 ret= -1;
571                                 goto end;
572                                 }
573                         s->rwstate=SSL_NOTHING;

575                         s->state=s->s3->tmp.next_state;
576                         break;

578                 case SSL3_ST_SR_CERT_A:
579                 case SSL3_ST_SR_CERT_B:
580                         /* Check for second client hello (MS SGC) */
581                         ret = ssl3_check_client_hello(s);
582                         if (ret <= 0)
583                                 goto end;
584                         if (ret == 2)
585                                 s->state = SSL3_ST_SR_CLNT_HELLO_C;
586                         else {
587                                 if (s->s3->tmp.cert_request)
588                                         {
589                                         ret=ssl3_get_client_certificate(s);
```



```
590                                         if (ret <= 0) goto end;
591                                         }
592                                 s->init_num=0;
593                                 s->state=SSL3_ST_SR_KEY_EXCH_A;
594                         }
595                         break;

597                 case SSL3_ST_SR_KEY_EXCH_A:
598                 case SSL3_ST_SR_KEY_EXCH_B:
599                         ret=ssl3_get_client_key_exchange(s);
600                         if (ret <= 0)
601                                 goto end;
602                         if (ret == 2)
603                                 {
604                                 /* For the ECDH ciphersuites when
605                                  * the client sends its ECDH pub key in
606                                  * a certificate, the CertificateVerify
607                                  * message is not sent.
608                                  * Also for GOST ciphersuites when
609                                  * the client uses its key from the certificate
610                                  * for key exchange.
611                                  */
612 #if defined(OPENSSL_NO_TLSEXT) || defined(OPENSSL_NO_NEXTPROTONEG)
613                                 s->state=SSL3_ST_SR_FINISHED_A;
614 #else
615                                 if (s->s3->next_proto_neg_seen)
616                                         s->state=SSL3_ST_SR_NEXT_PROTO_A;
617                                 else
618                                         s->state=SSL3_ST_SR_FINISHED_A;
619 #endif
620                                 s->init_num = 0;
621                                 }
622                         else if (TLS1_get_version(s) >= TLS1_2_VERSION)
623                                 {
624                                 s->state=SSL3_ST_SR_CERT_VRFY_A;
625                                 s->init_num=0;
626                                 if (!s->session->peer)
627                                         break;
628                                 /* For TLS v1.2 freeze the handshake buffer
629                                  * at this point and digest cached records.
630                                  */
631                                 if (!s->s3->handshake_buffer)
632                                         {
633                                         SSLerr(SSL_F_SSL3_ACCEPT,ERR_R_INTERNAL_
634                                         return -1;
635                                         }
636                                 s->s3->flags |= TLS1_FLAGS_KEEP_HANDSHAKE;
637                                 if (!ssl3_digest_cached_records(s))
638                                         return -1;
639                                 }
640                         else
641                                 {
642                                 int offset=0;
643                                 int dgst_num;

645                                 s->state=SSL3_ST_SR_CERT_VRFY_A;
646                                 s->init_num=0;

648                                 /* We need to get hashes here so if there is
649                                  * a client cert, it can be verified
650                                  * FIXME - digest processing for CertificateVeri
651                                  * should be generalized. But it is next step
652                                  */
653                                 if (s->s3->handshake_buffer)
654                                         if (!ssl3_digest_cached_records(s))
655                                                 return -1;
```

```
656                                 for (dgst_num=0; dgst_num<SSL_MAX_DIGEST;dgst_nu
657                                         if (s->s3->handshake_dgst[dgst_num])
658                                                 {
659                                                 int dgst_size;
660
661                                                 s->method->ssl3_enc->cert_verify
662                                                 dgst_size=EVP_MD_CTX_size(s->s3-
663                                                 if (dgst_size < 0)
664                                                         {
665                                                         ret = -1;
666                                                         goto end;
667                                                         }
668                                                 offset+=dgst_size;
669                                                 }
670                                 }
671                         break;
672
673                 case SSL3_ST_SR_CERT_VRFY_A:
674                 case SSL3_ST_SR_CERT_VRFY_B:
675
676                         s->s3->flags |= SSL3_FLAGS_CCS_OK;
677                         /* we should decide if we expected this one */
678                         ret=ssl3_get_cert_verify(s);
679                         if (ret <= 0) goto end;
680
681 #if defined(OPENSSL_NO_TLSEXT) || defined(OPENSSL_NO_NEXTPROTONEG)
682                         s->state=SSL3_ST_SR_FINISHED_A;
683 #else
684                         if (s->s3->next_proto_neg_seen)
685                                 s->state=SSL3_ST_SR_NEXT_PROTO_A;
686                         else
687                                 s->state=SSL3_ST_SR_FINISHED_A;
688 #endif
689                         s->init_num=0;
690                         break;
691
692 #if !defined(OPENSSL_NO_TLSEXT) && !defined(OPENSSL_NO_NEXTPROTONEG)
693                 case SSL3_ST_SR_NEXT_PROTO_A:
694                 case SSL3_ST_SR_NEXT_PROTO_B:
695                         ret=ssl3_get_next_proto(s);
696                         if (ret <= 0) goto end;
697                         s->init_num = 0;
698                         s->state=SSL3_ST_SR_FINISHED_A;
699                         break;
700 #endif
701
702                 case SSL3_ST_SR_FINISHED_A:
703                 case SSL3_ST_SR_FINISHED_B:
704                         s->s3->flags |= SSL3_FLAGS_CCS_OK;
705                         ret=ssl3_get_finished(s,SSL3_ST_SR_FINISHED_A,
706                                 SSL3_ST_SR_FINISHED_B);
707                         if (ret <= 0) goto end;
708                         if (s->hit)
709                                 s->state=SSL_ST_OK;
710 #ifndef OPENSSL_NO_TLSEXT
711                         else if (s->tlsext_ticket_expected)
712                                 s->state=SSL3_ST_SW_SESSION_TICKET_A;
713 #endif
714                         else
715                                 s->state=SSL3_ST_SW_CHANGE_A;
716                         s->init_num=0;
717                         break;
718
719 #ifndef OPENSSL_NO_TLSEXT
720                 case SSL3_ST_SW_SESSION_TICKET_A:
721                 case SSL3_ST_SW_SESSION_TICKET_B:
```



```
722                         ret=ssl3_send_newsession_ticket(s);
723                         if (ret <= 0) goto end;
724                         s->state=SSL3_ST_SW_CHANGE_A;
725                         s->init_num=0;
726                         break;
727
728                 case SSL3_ST_SW_CERT_STATUS_A:
729                 case SSL3_ST_SW_CERT_STATUS_B:
730                         ret=ssl3_send_cert_status(s);
731                         if (ret <= 0) goto end;
732                         s->state=SSL3_ST_SW_KEY_EXCH_A;
733                         s->init_num=0;
734                         break;
735
736 #endif
737
738                 case SSL3_ST_SW_CHANGE_A:
739                 case SSL3_ST_SW_CHANGE_B:
740
741                         s->session->cipher=s->s3->tmp.new_cipher;
742                         if (!s->method->ssl3_enc->setup_key_block(s))
743                                 { ret= -1; goto end; }
744
745                         ret=ssl3_send_change_cipher_spec(s,
746                                 SSL3_ST_SW_CHANGE_A,SSL3_ST_SW_CHANGE_B);
747
748                         if (ret <= 0) goto end;
749                         s->state=SSL3_ST_SW_FINISHED_A;
750                         s->init_num=0;
751
752                         if (!s->method->ssl3_enc->change_cipher_state(s,
753                                 SSL3_CHANGE_CIPHER_SERVER_WRITE))
754                                 {
755                                 ret= -1;
756                                 goto end;
757                                 }
758
759                         break;
760
761                 case SSL3_ST_SW_FINISHED_A:
762                 case SSL3_ST_SW_FINISHED_B:
763                         ret=ssl3_send_finished(s,
764                                 SSL3_ST_SW_FINISHED_A,SSL3_ST_SW_FINISHED_B,
765                                 s->method->ssl3_enc->server_finished_label,
766                                 s->method->ssl3_enc->server_finished_label_len);
767                         if (ret <= 0) goto end;
768                         s->state=SSL3_ST_SW_FLUSH;
769                         if (s->hit)
770                                 {
771 #if defined(OPENSSL_NO_TLSEXT) || defined(OPENSSL_NO_NEXTPROTONEG)
772                                 s->s3->tmp.next_state=SSL3_ST_SR_FINISHED_A;
773 #else
774                                 if (s->s3->next_proto_neg_seen)
775                                         {
776                                         s->s3->flags |= SSL3_FLAGS_CCS_OK;
777                                         s->s3->tmp.next_state=SSL3_ST_SR_NEXT_PR
778                                         }
779                                 else
780                                         s->s3->tmp.next_state=SSL3_ST_SR_FINISHE
781 #endif
782                                 }
783                         else
784                                 s->s3->tmp.next_state=SSL_ST_OK;
785                         s->init_num=0;
786                         break;
```

```
788                 case SSL_ST_OK:
789                         /* clean a few things up */
790                         ssl3_cleanup_key_block(s);
791
792                         BUF_MEM_free(s->init_buf);
793                         s->init_buf=NULL;
794
795                         /* remove buffering on output */
796                         ssl_free_wbio_buffer(s);
797
798                         s->init_num=0;
799
800                         if (s->renegotiate == 2) /* skipped if we just sent a He
801                                 {
802                                 s->renegotiate=0;
803                                 s->new_session=0;
804
805                                 ssl_update_cache(s,SSL_SESS_CACHE_SERVER);
806
807                                 s->ctx->stats.sess_accept_good++;
808                                 /* s->server=1; */
809                                 s->handshake_func=ssl3_accept;
810
811                                 if (cb != NULL) cb(s,SSL_CB_HANDSHAKE_DONE,1);
812                                 }
813
814                         ret = 1;
815                         goto end;
816                         /* break; */
817
818                 default:
819                         SSLerr(SSL_F_SSL3_ACCEPT,SSL_R_UNKNOWN_STATE);
820                         ret= -1;
821                         goto end;
822                         /* break; */
823                         }
824
825                 if (!s->s3->tmp.reuse_message && !skip)
826                         {
827                         if (s->debug)
828                                 {
829                                 if ((ret=BIO_flush(s->wbio)) <= 0)
830                                         goto end;
831                                 }
832
833
834                         if ((cb != NULL) && (s->state != state))
835                                 {
836                                 new_state=s->state;
837                                 s->state=state;
838                                 cb(s,SSL_CB_ACCEPT_LOOP,1);
839                                 s->state=new_state;
840                                 }
841                         }
842                 skip=0;
843                 }
844 end:
845         /* BIO_flush(s->wbio); */
846
847         s->in_handshake--;
848         if (cb != NULL)
849                 cb(s,SSL_CB_ACCEPT_EXIT,ret);
850         return(ret);
851         }
852
853 int ssl3_send_hello_request(SSL *s)
```



```
854         {
855         unsigned char *p;
856
857         if (s->state == SSL3_ST_SW_HELLO_REQ_A)
858                 {
859                 p=(unsigned char *)s->init_buf->data;
860                 *(p++)=SSL3_MT_HELLO_REQUEST;
861                 *(p++)=0;
862                 *(p++)=0;
863                 *(p++)=0;
864
865                 s->state=SSL3_ST_SW_HELLO_REQ_B;
866                 /* number of bytes to write */
867                 s->init_num=4;
868                 s->init_off=0;
869                 }
870
871         /* SSL3_ST_SW_HELLO_REQ_B */
872         return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
873         }
874
875 int ssl3_check_client_hello(SSL *s)
876         {
877         int ok;
878         long n;
879
880         /* this function is called when we really expect a Certificate message,
881          * so permit appropriate message length */
882         n=s->method->ssl_get_message(s,
883                 SSL3_ST_SR_CERT_A,
884                 SSL3_ST_SR_CERT_B,
885                 -1,
886                 s->max_cert_list,
887                 &ok);
888         if (!ok) return((int)n);
889         s->s3->tmp.reuse_message = 1;
890         if (s->s3->tmp.message_type == SSL3_MT_CLIENT_HELLO)
891                 {
892                 /* We only allow the client to restart the handshake once per
893                  * negotiation. */
894                 if (s->s3->flags & SSL3_FLAGS_SGC_RESTART_DONE)
895                         {
896                         SSLerr(SSL_F_SSL3_CHECK_CLIENT_HELLO, SSL_R_MULTIPLE_SGC
897                         return -1;
898                         }
899                 /* Throw away what we have done so far in the current handshake,
900                  * which will now be aborted. (A full SSL_clear would be too muc
901 #ifndef OPENSSL_NO_DH
902                 if (s->s3->tmp.dh != NULL)
903                         {
904                         DH_free(s->s3->tmp.dh);
905                         s->s3->tmp.dh = NULL;
906                         }
907 #endif
908 #ifndef OPENSSL_NO_ECDH
909                 if (s->s3->tmp.ecdh != NULL)
910                         {
911                         EC_KEY_free(s->s3->tmp.ecdh);
912                         s->s3->tmp.ecdh = NULL;
913                         }
914 #endif
915                 s->s3->flags |= SSL3_FLAGS_SGC_RESTART_DONE;
916                 return 2;
917                 }
918         return 1;
919 }
```

```
int ssl3_get_client_hello(SSL *s)
        {
        int i,j,ok,al,ret= -1;
        unsigned int cookie_len;
        long n;
        unsigned long id;
        unsigned char *p,*d,*q;
        SSL_CIPHER *c;
#ifndef OPENSSL_NO_COMP
        SSL_COMP *comp=NULL;
#endif
        STACK_OF(SSL_CIPHER) *ciphers=NULL;

        /* We do this so that we will respond with our native type.
         * If we are TLSv1 and we get SSLv3, we will respond with TLSv1,
         * This down switching should be handled by a different method.
         * If we are SSLv3, we will respond with SSLv3, even if prompted with
         * TLSv1.
         */
        if (s->state == SSL3_ST_SR_CLNT_HELLO_A
                )
                {
                s->state=SSL3_ST_SR_CLNT_HELLO_B;
                }
        s->first_packet=1;
        n=s->method->ssl_get_message(s,
                SSL3_ST_SR_CLNT_HELLO_B,
                SSL3_ST_SR_CLNT_HELLO_C,
                SSL3_MT_CLIENT_HELLO,
                SSL3_RT_MAX_PLAIN_LENGTH,
                &ok);

        if (!ok) return((int)n);
        s->first_packet=0;
        d=p=(unsigned char *)s->init_msg;

        /* use version from inside client hello, not from record header
         * (may differ: see RFC 2246, Appendix E, second paragraph) */
        s->client_version=(((int)p[0])<<8)|(int)p[1];
        p+=2;

        if ((s->version == DTLS1_VERSION && s->client_version > s->version) ||
            (s->version != DTLS1_VERSION && s->client_version < s->version))
                {
                SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO, SSL_R_WRONG_VERSION_NUMBER);
                if ((s->client_version>>8) == SSL3_VERSION_MAJOR &&
                        !s->enc_write_ctx && !s->write_hash)
                        {
                        /* similar to ssl3_get_record, send alert using remote v
                        s->version = s->client_version;
                        }
                al = SSL_AD_PROTOCOL_VERSION;
                goto f_err;
                }

        /* If we require cookies and this ClientHello doesn't
         * contain one, just return since we do not want to
         * allocate any memory yet. So check cookie length...
         */
        if (SSL_get_options(s) & SSL_OP_COOKIE_EXCHANGE)
                {
                unsigned int session_length, cookie_length;

                session_length = *(p + SSL3_RANDOM_SIZE);
                cookie_length = *(p + SSL3_RANDOM_SIZE + session_length + 1);
```



```
                if (cookie_length == 0)
                        return 1;
                }

        /* load the client random */
        memcpy(s->s3->client_random,p,SSL3_RANDOM_SIZE);
        p+=SSL3_RANDOM_SIZE;

        /* get the session-id */
        j= *(p++);

        s->hit=0;
        /* Versions before 0.9.7 always allow clients to resume sessions in rene
         * 0.9.7 and later allow this by default, but optionally ignore resumpti
         * with flag SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION (it's a new f
         * than a change to default behavior so that applications relying on thi
         * won't even compile against older library versions).
         *
         * 1.0.1 and later also have a function SSL_renegotiate_abbreviated() to
         * renegotiation but not a new session (s->new_session remains unset): f
         * this essentially just means that the SSL_OP_NO_SESSION_RESUMPTION_ON_
         * setting will be ignored.
         */
        if ((s->new_session && (s->options & SSL_OP_NO_SESSION_RESUMPTION_ON_REN
                {
                if (!ssl_get_new_session(s,1))
                        goto err;
                }
        else
                {
                i=ssl_get_prev_session(s, p, j, d + n);
                if (i == 1)
                        { /* previous session */
                        s->hit=1;
                        }
                else if (i == -1)
                        goto err;
                else /* i == 0 */
                        {
                        if (!ssl_get_new_session(s,1))
                                goto err;
                        }
                }

        p+=j;

        if (s->version == DTLS1_VERSION || s->version == DTLS1_BAD_VER)
                {
                /* cookie stuff */
                cookie_len = *(p++);

                /*
                 * The ClientHello may contain a cookie even if the
                 * HelloVerify message has not been sent--make sure that it
                 * does not cause an overflow.
                 */
                if ( cookie_len > sizeof(s->d1->rcvd_cookie))
                        {
                        /* too much data */
                        al = SSL_AD_DECODE_ERROR;
                        SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO, SSL_R_COOKIE_MISMATC
                        goto f_err;
                        }

                /* verify the cookie if appropriate option is set. */
```

```
1052                 if ((SSL_get_options(s) & SSL_OP_COOKIE_EXCHANGE) &&
1053                         cookie_len > 0)
1054                         {
1055                         memcpy(s->d1->rcvd_cookie, p, cookie_len);
1056
1057                         if ( s->ctx->app_verify_cookie_cb != NULL)
1058                                 {
1059                                 if ( s->ctx->app_verify_cookie_cb(s, s->d1->rcvd
1060                                         cookie_len) == 0)
1061                                         {
1062                                         al=SSL_AD_HANDSHAKE_FAILURE;
1063                                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,
1064                                                 SSL_R_COOKIE_MISMATCH);
1065                                         goto f_err;
1066                                         }
1067                                 /* else cookie verification succeeded */
1068                                 }
1069                         else if ( memcmp(s->d1->rcvd_cookie, s->d1->cookie,
1070                                                 s->d1->cookie_len) != 0) /* de
1071                                 {
1072                                         al=SSL_AD_HANDSHAKE_FAILURE;
1073                                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,
1074                                                 SSL_R_COOKIE_MISMATCH);
1075                                         goto f_err;
1076                                 }
1077
1078                         ret = 2;
1079                         }
1080
1081                 p += cookie_len;
1082                 }
1083
1084         n2s(p,i);
1085         if ((i == 0) && (j != 0))
1086                 {
1087                 /* we need a cipher if we are not resuming a session */
1088                 al=SSL_AD_ILLEGAL_PARAMETER;
1089                 SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_NO_CIPHERS_SPECIFIED);
1090                 goto f_err;
1091                 }
1092         if ((p+i) >= (d+n))
1093                 {
1094                 /* not enough data */
1095                 al=SSL_AD_DECODE_ERROR;
1096                 SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_LENGTH_MISMATCH);
1097                 goto f_err;
1098                 }
1099         if ((i > 0) && (ssl_bytes_to_cipher_list(s,p,i,&(ciphers))
1100                 == NULL))
1101                 {
1102                 goto err;
1103                 }
1104         p+=i;
1105
1106         /* If it is a hit, check that the cipher is in the list */
1107         if ((s->hit) && (i > 0))
1108                 {
1109                 j=0;
1110                 id=s->session->cipher->id;
1111
1112 #ifdef CIPHER_DEBUG
1113                 printf("client sent %d ciphers\n",sk_num(ciphers));
1114 #endif
1115                 for (i=0; i<sk_SSL_CIPHER_num(ciphers); i++)
1116                         {
1117                         c=sk_SSL_CIPHER_value(ciphers,i);
```



```
1118 #ifdef CIPHER_DEBUG
1119                         printf("client [%2d of %2d]:%s\n",
1120                                 i,sk_num(ciphers),SSL_CIPHER_get_name(c));
1121 #endif
1122                         if (c->id == id)
1123                                 {
1124                                 j=1;
1125                                 break;
1126                                 }
1127                         }
1128 /* Disabled because it can be used in a ciphersuite downgrade
1129  * attack: CVE-2010-4180.
1130  */
1131 #if 0
1132                 if (j == 0 && (s->options & SSL_OP_NETSCAPE_REUSE_CIPHER_CHANGE_
1133                         {
1134                         /* Special case as client bug workaround: the previously
1135                          * not be in the current list, the client instead might
1136                          * continue using a cipher that before wasn't chosen due
1137                          * preferences.  We'll have to reject the connection if
1138                          * enabled, though. */
1139                         c = sk_SSL_CIPHER_value(ciphers, 0);
1140                         if (sk_SSL_CIPHER_find(SSL_get_ciphers(s), c) >= 0)
1141                                 {
1142                                 s->session->cipher = c;
1143                                 j = 1;
1144                                 }
1145                         }
1146 #endif
1147                 if (j == 0)
1148                         {
1149                         /* we need to have the cipher in the cipher
1150                          * list if we are asked to reuse it */
1151                         al=SSL_AD_ILLEGAL_PARAMETER;
1152                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_REQUIRED_CIPHER
1153                         goto f_err;
1154                         }
1155                 }
1156
1157         /* compression */
1158         i= *(p++);
1159         if ((p+i) > (d+n))
1160                 {
1161                 /* not enough data */
1162                 al=SSL_AD_DECODE_ERROR;
1163                 SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_LENGTH_MISMATCH);
1164                 goto f_err;
1165                 }
1166         q=p;
1167         for (j=0; j<i; j++)
1168                 {
1169                 if (p[j] == 0) break;
1170                 }
1171
1172         p+=i;
1173         if (j >= i)
1174                 {
1175                 /* no compress */
1176                 al=SSL_AD_DECODE_ERROR;
1177                 SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_NO_COMPRESSION_SPECIFIE
1178                 goto f_err;
1179                 }
1180
1181 #ifndef OPENSSL_NO_TLSEXT
1182         /* TLS extensions*/
1183         if (s->version >= SSL3_VERSION)
```

```
1184                 {
1185                 if (!ssl_parse_clienthello_tlsext(s,&p,d,n, &al))
1186                         {
1187                         /* 'al' set by ssl_parse_clienthello_tlsext */
1188                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_PARSE_TLSEXT);
1189                         goto f_err;
1190                         }
1191                 }
1192                 if (ssl_check_clienthello_tlsext_early(s) <= 0) {
1193                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_CLIENTHELLO_TLS
1194                         goto err;
1195                 }

1197         /* Check if we want to use external pre-shared secret for this
1198          * handshake for not reused session only. We need to generate
1199          * server_random before calling tls_session_secret_cb in order to allow
1200          * SessionTicket processing to use it in key derivation. */
1201         {
1202                 unsigned char *pos;
1203                 pos=s->s3->server_random;
1204                 if (ssl_fill_hello_random(s, 1, pos, SSL3_RANDOM_SIZE) <= 0)
1205                         {
1206                         al=SSL_AD_INTERNAL_ERROR;
1207                         goto f_err;
1208                         }
1209         }

1211         if (!s->hit && s->version >= TLS1_VERSION && s->tls_session_secret_cb)
1212                 {
1213                 SSL_CIPHER *pref_cipher=NULL;

1215                 s->session->master_key_length=sizeof(s->session->master_key);
1216                 if(s->tls_session_secret_cb(s, s->session->master_key, &s->sessi
1217                         ciphers, &pref_cipher, s->tls_session_secret_cb_arg))
1218                         {
1219                         s->hit=1;
1220                         s->session->ciphers=ciphers;
1221                         s->session->verify_result=X509_V_OK;

1223                         ciphers=NULL;

1225                         /* check if some cipher was preferred by call back */
1226                         pref_cipher=pref_cipher ? pref_cipher : ssl3_choose_ciph
1227                         if (pref_cipher == NULL)
1228                                 {
1229                                 al=SSL_AD_HANDSHAKE_FAILURE;
1230                                 SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_NO_SHAR
1231                                 goto f_err;
1232                                 }

1234                         s->session->cipher=pref_cipher;

1236                         if (s->cipher_list)
1237                                 sk_SSL_CIPHER_free(s->cipher_list);

1239                         if (s->cipher_list_by_id)
1240                                 sk_SSL_CIPHER_free(s->cipher_list_by_id);

1242                         s->cipher_list = sk_SSL_CIPHER_dup(s->session->ciphers);
1243                         s->cipher_list_by_id = sk_SSL_CIPHER_dup(s->session->cip
1244                         }
1245                 }
1246 #endif

1248         /* Worst case, we will use the NULL compression, but if we have other
1249          * options, we will now look for them.  We have i-1 compression
```



```
1250          * algorithms from the client, starting at q. */
1251         s->s3->tmp.new_compression=NULL;
1252 #ifndef OPENSSL_NO_COMP
1253         /* This only happens if we have a cache hit */
1254         if (s->session->compress_meth != 0)
1255                 {
1256                 int m, comp_id = s->session->compress_meth;
1257                 /* Perform sanity checks on resumed compression algorithm */
1258                 /* Can't disable compression */
1259                 if (s->options & SSL_OP_NO_COMPRESSION)
1260                         {
1261                         al=SSL_AD_INTERNAL_ERROR;
1262                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_INCONSISTENT_CO
1263                         goto f_err;
1264                         }
1265                 /* Look for resumed compression method */
1266                 for (m = 0; m < sk_SSL_COMP_num(s->ctx->comp_methods); m++)
1267                         {
1268                         comp=sk_SSL_COMP_value(s->ctx->comp_methods,m);
1269                         if (comp_id == comp->id)
1270                                 {
1271                                 s->s3->tmp.new_compression=comp;
1272                                 break;
1273                                 }
1274                         }
1275                 if (s->s3->tmp.new_compression == NULL)
1276                         {
1277                         al=SSL_AD_INTERNAL_ERROR;
1278                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_INVALID_COMPRES
1279                         goto f_err;
1280                         }
1281                 /* Look for resumed method in compression list */
1282                 for (m = 0; m < i; m++)
1283                         {
1284                         if (q[m] == comp_id)
1285                                 break;
1286                         }
1287                 if (m >= i)
1288                         {
1289                         al=SSL_AD_ILLEGAL_PARAMETER;
1290                         SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_REQUIRED_COMPRE
1291                         goto f_err;
1292                         }
1293                 }
1294         else if (s->hit)
1295                 comp = NULL;
1296         else if (!(s->options & SSL_OP_NO_COMPRESSION) && s->ctx->comp_methods)
1297                 { /* See if we have a match */
1298                 int m,nn,o,v,done=0;

1300                 nn=sk_SSL_COMP_num(s->ctx->comp_methods);
1301                 for (m=0; m<nn; m++)
1302                         {
1303                         comp=sk_SSL_COMP_value(s->ctx->comp_methods,m);
1304                         v=comp->id;
1305                         for (o=0; o<i; o++)
1306                                 {
1307                                 if (v == q[o])
1308                                         {
1309                                         done=1;
1310                                         break;
1311                                         }
1312                                 }
1313                         if (done) break;
1314                         }
1315                 if (done)
```

```
1316                        s->s3->tmp.new_compression=comp;
1317                else
1318                        comp=NULL;
1319                }
1320 #else
1321        /* If compression is disabled we'd better not try to resume a session
1322         * using compression.
1323         */
1324        if (s->session->compress_meth != 0)
1325                {
1326                al=SSL_AD_INTERNAL_ERROR;
1327                SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_INCONSISTENT_COMPRESSIO
1328                goto f_err;
1329                }
1330 #endif

1332        /* Given s->session->ciphers and SSL_get_ciphers, we must
1333         * pick a cipher */

1335        if (!s->hit)
1336                {
1337 #ifdef OPENSSL_NO_COMP
1338                s->session->compress_meth=0;
1339 #else
1340                s->session->compress_meth=(comp == NULL)?0:comp->id;
1341 #endif
1342                if (s->session->ciphers != NULL)
1343                        sk_SSL_CIPHER_free(s->session->ciphers);
1344                s->session->ciphers=ciphers;
1345                if (ciphers == NULL)
1346                        {
1347                        al=SSL_AD_ILLEGAL_PARAMETER;
1348                        SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_NO_CIPHERS_PASS
1349                        goto f_err;
1350                        }
1351                ciphers=NULL;
1352                c=ssl3_choose_cipher(s,s->session->ciphers,
1353                                     SSL_get_ciphers(s));

1355                if (c == NULL)
1356                        {
1357                        al=SSL_AD_HANDSHAKE_FAILURE;
1358                        SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO,SSL_R_NO_SHARED_CIPHE
1359                        goto f_err;
1360                        }
1361                s->s3->tmp.new_cipher=c;
1362                }
1363        else
1364                {
1365                /* Session-id reuse */
1366 #ifdef REUSE_CIPHER_BUG
1367                STACK_OF(SSL_CIPHER) *sk;
1368                SSL_CIPHER *nc=NULL;
1369                SSL_CIPHER *ec=NULL;

1371                if (s->options & SSL_OP_NETSCAPE_DEMO_CIPHER_CHANGE_BUG)
1372                        {
1373                        sk=s->session->ciphers;
1374                        for (i=0; i<sk_SSL_CIPHER_num(sk); i++)
1375                                {
1376                                c=sk_SSL_CIPHER_value(sk,i);
1377                                if (c->algorithm_enc & SSL_eNULL)
1378                                        nc=c;
1379                                if (SSL_C_IS_EXPORT(c))
1380                                        ec=c;
1381                                }
```



```
1382                        if (nc != NULL)
1383                                s->s3->tmp.new_cipher=nc;
1384                        else if (ec != NULL)
1385                                s->s3->tmp.new_cipher=ec;
1386                        else
1387                                s->s3->tmp.new_cipher=s->session->cipher;
1388                        }
1389                else
1390 #endif
1391                s->s3->tmp.new_cipher=s->session->cipher;
1392                }

1394        if (TLS1_get_version(s) < TLS1_2_VERSION || !(s->verify_mode & SSL_VERIF
1395                {
1396                if (!ssl3_digest_cached_records(s))
1397                        {
1398                        al = SSL_AD_INTERNAL_ERROR;
1399                        goto f_err;
1400                        }
1401                }
1402
1403        /* we now have the following setup.
1404         * client_random
1405         * cipher_list          - our prefered list of ciphers
1406         * ciphers              - the clients prefered list of ciphers
1407         * compression          - basically ignored right now
1408         * ssl version is set   - sslv3
1409         * s->session           - The ssl session has been setup.
1410         * s->hit               - session reuse flag
1411         * s->tmp.new_cipher    - the new cipher to use.
1412         */

1414        /* Handles TLS extensions that we couldn't check earlier */
1415        if (s->version >= SSL3_VERSION)
1416                {
1417                if (ssl_check_clienthello_tlsext_late(s) <= 0)
1418                        {
1419                        SSLerr(SSL_F_SSL3_GET_CLIENT_HELLO, SSL_R_CLIENTHELLO_TL
1420                        goto err;
1421                        }
1422                }

1424        if (ret < 0) ret=1;
1425        if (0)
1426                {
1427 f_err:
1428                ssl3_send_alert(s,SSL3_AL_FATAL,al);
1429                }
1430 err:
1431        if (ciphers != NULL) sk_SSL_CIPHER_free(ciphers);
1432        return(ret);
1433        }

1435 int ssl3_send_server_hello(SSL *s)
1436        {
1437        unsigned char *buf;
1438        unsigned char *p,*d;
1439        int i,sl;
1440        unsigned long l;

1442        if (s->state == SSL3_ST_SW_SRVR_HELLO_A)
1443                {
1444                buf=(unsigned char *)s->init_buf->data;
1445 #ifdef OPENSSL_NO_TLSEXT
1446                p=s->s3->server_random;
1447                if (ssl_fill_hello_random(s, 1, p, SSL3_RANDOM_SIZE) <= 0)
```

```
1448                        return -1;
1449 #endif
1450                /* Do the message type and length last */
1451                d=p= &(buf[4]);

1453                *(p++)=s->version>>8;
1454                *(p++)=s->version&0xff;

1456                /* Random stuff */
1457                memcpy(p,s->s3->server_random,SSL3_RANDOM_SIZE);
1458                p+=SSL3_RANDOM_SIZE;

1460                /* There are several cases for the session ID to send
1461                 * back in the server hello:
1462                 * - For session reuse from the session cache,
1463                 *   we send back the old session ID.
1464                 * - If stateless session reuse (using a session ticket)
1465                 *   is successful, we send back the client's "session ID"
1466                 *   (which doesn't actually identify the session).
1467                 * - If it is a new session, we send back the new
1468                 *   session ID.
1469                 * - However, if we want the new session to be single-use,
1470                 *   we send back a 0-length session ID.
1471                 * s->hit is non-zero in either case of session reuse,
1472                 * so the following won't overwrite an ID that we're supposed
1473                 * to send back.
1474                 */
1475                if (!(s->ctx->session_cache_mode & SSL_SESS_CACHE_SERVER)
1476                        && !s->hit)
1477                        s->session->session_id_length=0;

1479                sl=s->session->session_id_length;
1480                if (sl > (int)sizeof(s->session->session_id))
1481                        {
1482                        SSLerr(SSL_F_SSL3_SEND_SERVER_HELLO, ERR_R_INTERNAL_ERRO
1483                        return -1;
1484                        }
1485                *(p++)=sl;
1486                memcpy(p,s->session->session_id,sl);
1487                p+=sl;

1489                /* put the cipher */
1490                i=ssl3_put_cipher_by_char(s->s3->tmp.new_cipher,p);
1491                p+=i;

1493                /* put the compression method */
1494 #ifdef OPENSSL_NO_COMP
1495                        *(p++)=0;
1496 #else
1497                if (s->s3->tmp.new_compression == NULL)
1498                        *(p++)=0;
1499                else
1500                        *(p++)=s->s3->tmp.new_compression->id;
1501 #endif
1502 #ifndef OPENSSL_NO_TLSEXT
1503                if (ssl_prepare_serverhello_tlsext(s) <= 0)
1504                        {
1505                        SSLerr(SSL_F_SSL3_SEND_SERVER_HELLO,SSL_R_SERVERHELLO_TL
1506                        return -1;
1507                        }
1508                if ((p = ssl_add_serverhello_tlsext(s, p, buf+SSL3_RT_MAX_PLAIN_
1509                        {
1510                        SSLerr(SSL_F_SSL3_SEND_SERVER_HELLO,ERR_R_INTERNAL_ERROR
1511                        return -1;
1512                        }
1513 #endif
```



```
1514                /* do the header */
1515                l=(p-d);
1516                d=buf;
1517                *(d++)=SSL3_MT_SERVER_HELLO;
1518                l2n3(l,d);

1520                s->state=SSL3_ST_SW_SRVR_HELLO_B;
1521                /* number of bytes to write */
1522                s->init_num=p-buf;
1523                s->init_off=0;
1524                }

1526        /* SSL3_ST_SW_SRVR_HELLO_B */
1527        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
1528        }

1530 int ssl3_send_server_done(SSL *s)
1531        {
1532        unsigned char *p;

1534        if (s->state == SSL3_ST_SW_SRVR_DONE_A)
1535                {
1536                p=(unsigned char *)s->init_buf->data;

1538                /* do the header */
1539                *(p++)=SSL3_MT_SERVER_DONE;
1540                *(p++)=0;
1541                *(p++)=0;
1542                *(p++)=0;

1544                s->state=SSL3_ST_SW_SRVR_DONE_B;
1545                /* number of bytes to write */
1546                s->init_num=4;
1547                s->init_off=0;
1548                }

1550        /* SSL3_ST_SW_SRVR_DONE_B */
1551        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
1552        }

1554 int ssl3_send_server_key_exchange(SSL *s)
1555        {
1556 #ifndef OPENSSL_NO_RSA
1557        unsigned char *q;
1558        int j,num;
1559        RSA *rsa;
1560        unsigned char md_buf[MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH];
1561        unsigned int u;
1562 #endif
1563 #ifndef OPENSSL_NO_DH
1564        DH *dh=NULL,*dhp;
1565 #endif
1566 #ifndef OPENSSL_NO_ECDH
1567        EC_KEY *ecdh=NULL, *ecdhp;
1568        unsigned char *encodedPoint = NULL;
1569        int encodedlen = 0;
1570        int curve_id = 0;
1571        BN_CTX *bn_ctx = NULL;
1572 #endif
1573        EVP_PKEY *pkey;
1574        const EVP_MD *md = NULL;
1575        unsigned char *p,*d;
1576        int al,i;
1577        unsigned long type;
1578        int n;
1579        CERT *cert;
```

```
1580         BIGNUM *r[4];
1581         int nr[4],kn;
1582         BUF_MEM *buf;
1583         EVP_MD_CTX md_ctx;

1585         EVP_MD_CTX_init(&md_ctx);
1586         if (s->state == SSL3_ST_SW_KEY_EXCH_A)
1587                 {
1588                 type=s->s3->tmp.new_cipher->algorithm_mkey;
1589                 cert=s->cert;

1591                 buf=s->init_buf;

1593                 r[0]=r[1]=r[2]=r[3]=NULL;
1594                 n=0;
1595 #ifndef OPENSSL_NO_RSA
1596                 if (type & SSL_kRSA)
1597                         {
1598                         rsa=cert->rsa_tmp;
1599                         if ((rsa == NULL) && (s->cert->rsa_tmp_cb != NULL))
1600                                 {
1601                                 rsa=s->cert->rsa_tmp_cb(s,
1602                                       SSL_C_IS_EXPORT(s->s3->tmp.new_cipher),
1603                                       SSL_C_EXPORT_PKEYLENGTH(s->s3->tmp.new_cip
1604                                 if(rsa == NULL)
1605                                 {
1606                                        al=SSL_AD_HANDSHAKE_FAILURE;
1607                                        SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHAN
1608                                        goto f_err;
1609                                 }
1610                                 RSA_up_ref(rsa);
1611                                 cert->rsa_tmp=rsa;
1612                                 }
1613                         if (rsa == NULL)
1614                                 {
1615                                 al=SSL_AD_HANDSHAKE_FAILURE;
1616                                 SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R
1617                                 goto f_err;
1618                                 }
1619                         r[0]=rsa->n;
1620                         r[1]=rsa->e;
1621                         s->s3->tmp.use_rsa_tmp=1;
1622                         }
1623                 else
1624 #endif
1625 #ifndef OPENSSL_NO_DH
1626                         if (type & SSL_kEDH)
1627                         {
1628                         dhp=cert->dh_tmp;
1629                         if ((dhp == NULL) && (s->cert->dh_tmp_cb != NULL))
1630                                 dhp=s->cert->dh_tmp_cb(s,
1631                                       SSL_C_IS_EXPORT(s->s3->tmp.new_cipher),
1632                                       SSL_C_EXPORT_PKEYLENGTH(s->s3->tmp.new_cip
1633                         if (dhp == NULL)
1634                                 {
1635                                 al=SSL_AD_HANDSHAKE_FAILURE;
1636                                 SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R
1637                                 goto f_err;
1638                                 }

1640                         if (s->s3->tmp.dh != NULL)
1641                                 {
1642                                 SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE, ERR_
1643                                 goto err;
1644                                 }
```



```
1646                         if ((dh=DHparams_dup(dhp)) == NULL)
1647                                 {
1648                                 SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,ERR_R
1649                                 goto err;
1650                                 }

1652                         s->s3->tmp.dh=dh;
1653                         if ((dhp->pub_key == NULL ||
1654                              dhp->priv_key == NULL ||
1655                              (s->options & SSL_OP_SINGLE_DH_USE)))
1656                                 {
1657                                 if(!DH_generate_key(dh))
1658                                     {
1659                                     SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,
1660                                            ERR_R_DH_LIB);
1661                                     goto err;
1662                                     }
1663                                 }
1664                         else
1665                                 {
1666                                 dh->pub_key=BN_dup(dhp->pub_key);
1667                                 dh->priv_key=BN_dup(dhp->priv_key);
1668                                 if ((dh->pub_key == NULL) ||
1669                                        (dh->priv_key == NULL))
1670                                        {
1671                                        SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHAN
1672                                        goto err;
1673                                        }
1674                                 }
1675                         r[0]=dh->p;
1676                         r[1]=dh->g;
1677                         r[2]=dh->pub_key;
1678                         }
1679                 else
1680 #endif
1681 #ifndef OPENSSL_NO_ECDH
1682                         if (type & SSL_kEECDH)
1683                         {
1684                         const EC_GROUP *group;

1686                         ecdhp=cert->ecdh_tmp;
1687                         if ((ecdhp == NULL) && (s->cert->ecdh_tmp_cb != NULL))
1688                                 {
1689                                 ecdhp=s->cert->ecdh_tmp_cb(s,
1690                                       SSL_C_IS_EXPORT(s->s3->tmp.new_cipher),
1691                                       SSL_C_EXPORT_PKEYLENGTH(s->s3->tmp.new_cip
1692                                 }
1693                         if (ecdhp == NULL)
1694                                 {
1695                                 al=SSL_AD_HANDSHAKE_FAILURE;
1696                                 SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R
1697                                 goto f_err;
1698                                 }

1700                         if (s->s3->tmp.ecdh != NULL)
1701                                 {
1702                                 SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE, ERR_
1703                                 goto err;
1704                                 }

1706                         /* Duplicate the ECDH structure. */
1707                         if (ecdhp == NULL)
1708                                 {
1709                                 SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,ERR_R
1710                                 goto err;
1711                                 }
```

```
1712                        if ((ecdh = EC_KEY_dup(ecdhp)) == NULL)
1713                            {
1714                            SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,ERR_R
1715                            goto err;
1716                            }
1718                        s->s3->tmp.ecdh=ecdh;
1719                        if ((EC_KEY_get0_public_key(ecdh) == NULL) ||
1720                            (EC_KEY_get0_private_key(ecdh) == NULL) ||
1721                            (s->options & SSL_OP_SINGLE_ECDH_USE))
1722                            {
1723                            if(!EC_KEY_generate_key(ecdh))
1724                                {
1725                                SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,E
1726                                goto err;
1727                                }
1728                            }
1730                        if (((group = EC_KEY_get0_group(ecdh)) == NULL) ||
1731                            (EC_KEY_get0_public_key(ecdh)  == NULL) ||
1732                            (EC_KEY_get0_private_key(ecdh) == NULL))
1733                            {
1734                            SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,ERR_R
1735                            goto err;
1736                            }
1738                        if (SSL_C_IS_EXPORT(s->s3->tmp.new_cipher) &&
1739                            (EC_GROUP_get_degree(group) > 163))
1740                            {
1741                            SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R
1742                            goto err;
1743                            }
1745                        /* XXX: For now, we only support ephemeral ECDH
1746                         * keys over named (not generic) curves. For
1747                         * supported named curves, curve_id is non-zero.
1748                         */
1749                        if ((curve_id =
1750                            tls1_ec_nid2curve_id(EC_GROUP_get_curve_name(group))
1751                            == 0)
1752                            {
1753                            SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R
1754                            goto err;
1755                            }
1757                        /* Encode the public key.
1758                         * First check the size of encoding and
1759                         * allocate memory accordingly.
1760                         */
1761                        encodedlen = EC_POINT_point2oct(group,
1762                            EC_KEY_get0_public_key(ecdh),
1763                            POINT_CONVERSION_UNCOMPRESSED,
1764                            NULL, 0, NULL);
1766                        encodedPoint = (unsigned char *)
1767                            OPENSSL_malloc(encodedlen*sizeof(unsigned char));
1768                        bn_ctx = BN_CTX_new();
1769                        if ((encodedPoint == NULL) || (bn_ctx == NULL))
1770                            {
1771                            SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,ERR_R
1772                            goto err;
1773                            }
1776                        encodedlen = EC_POINT_point2oct(group,
1777                            EC_KEY_get0_public_key(ecdh),
```



```
1778                            POINT_CONVERSION_UNCOMPRESSED,
1779                            encodedPoint, encodedlen, bn_ctx);
1781                        if (encodedlen == 0)
1782                            {
1783                            SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,ERR_R
1784                            goto err;
1785                            }
1787                        BN_CTX_free(bn_ctx);  bn_ctx=NULL;
1789                        /* XXX: For now, we only support named (not
1790                         * generic) curves in ECDH ephemeral key exchanges.
1791                         * In this situation, we need four additional bytes
1792                         * to encode the entire ServerECDHParams
1793                         * structure.
1794                         */
1795                        n = 4 + encodedlen;
1797                        /* We'll generate the serverKeyExchange message
1798                         * explicitly so we can set these to NULLs
1799                         */
1800                        r[0]=NULL;
1801                        r[1]=NULL;
1802                        r[2]=NULL;
1803                        r[3]=NULL;
1804                        }
1805                    else
1806 #endif /* !OPENSSL_NO_ECDH */
1807 #ifndef OPENSSL_NO_PSK
1808                        if (type & SSL_kPSK)
1809                            {
1810                            /* reserve size for record length and PSK identi
1811                            n+=2+strlen(s->ctx->psk_identity_hint);
1812                            }
1813                        else
1814 #endif /* !OPENSSL_NO_PSK */
1815 #ifndef OPENSSL_NO_SRP
1816                    if (type & SSL_kSRP)
1817                            {
1818                            if ((s->srp_ctx.N == NULL) ||
1819                                (s->srp_ctx.g == NULL) ||
1820                                (s->srp_ctx.s == NULL) ||
1821                                (s->srp_ctx.B == NULL))
1822                                {
1823                                SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R
1824                                goto err;
1825                                }
1826                            r[0]=s->srp_ctx.N;
1827                            r[1]=s->srp_ctx.g;
1828                            r[2]=s->srp_ctx.s;
1829                            r[3]=s->srp_ctx.B;
1830                            }
1831                    else
1832 #endif
1833                            {
1834                            al=SSL_AD_HANDSHAKE_FAILURE;
1835                            SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R_UNKNOWN
1836                            goto f_err;
1837                            }
1838                    for (i=0; i < 4 && r[i] != NULL; i++)
1839                            {
1840                            nr[i]=BN_num_bytes(r[i]);
1841 #ifndef OPENSSL_NO_SRP
1842                            if ((i == 2) && (type & SSL_kSRP))
1843                                n+=1+nr[i];
```

```
1844                        else
1845 #endif
1846                        n+=2+nr[i];
1847                        }
1848
1849                if (!(s->s3->tmp.new_cipher->algorithm_auth & SSL_aNULL)
1850                        && !(s->s3->tmp.new_cipher->algorithm_mkey & SSL_kPSK))
1851                        {
1852                        if ((pkey=ssl_get_sign_pkey(s,s->s3->tmp.new_cipher,&md)
1853                                == NULL)
1854                                {
1855                                al=SSL_AD_DECODE_ERROR;
1856                                goto f_err;
1857                                }
1858                        kn=EVP_PKEY_size(pkey);
1859                        }
1860                else
1861                        {
1862                        pkey=NULL;
1863                        kn=0;
1864                        }
1865
1866                if (!BUF_MEM_grow_clean(buf,n+4+kn))
1867                        {
1868                        SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,ERR_LIB_BUF);
1869                        goto err;
1870                        }
1871                d=(unsigned char *)s->init_buf->data;
1872                p= &(d[4]);
1873
1874                for (i=0; i < 4 && r[i] != NULL; i++)
1875                        {
1876 #ifndef OPENSSL_NO_SRP
1877                        if ((i == 2) && (type & SSL_kSRP))
1878                                {
1879                                *p = nr[i];
1880                                p++;
1881                                }
1882                        else
1883 #endif
1884                        s2n(nr[i],p);
1885                        BN_bn2bin(r[i],p);
1886                        p+=nr[i];
1887                        }
1888
1889 #ifndef OPENSSL_NO_ECDH
1890                if (type & SSL_kEECDH)
1891                        {
1892                        /* XXX: For now, we only support named (not generic) cur
1893                         * In this situation, the serverKeyExchange message has:
1894                         * [1 byte CurveType], [2 byte CurveName]
1895                         * [1 byte length of encoded point], followed by
1896                         * the actual encoded point itself
1897                         */
1898                        *p = NAMED_CURVE_TYPE;
1899                        p += 1;
1900                        *p = 0;
1901                        p += 1;
1902                        *p = curve_id;
1903                        p += 1;
1904                        *p = encodedlen;
1905                        p += 1;
1906                        memcpy((unsigned char*)p,
1907                            (unsigned char *)encodedPoint,
1908                            encodedlen);
1909                        OPENSSL_free(encodedPoint);
```



```
1910                        encodedPoint = NULL;
1911                        p += encodedlen;
1912                        }
1913 #endif
1914
1915 #ifndef OPENSSL_NO_PSK
1916                if (type & SSL_kPSK)
1917                        {
1918                        /* copy PSK identity hint */
1919                        s2n(strlen(s->ctx->psk_identity_hint), p);
1920                        strncpy((char *)p, s->ctx->psk_identity_hint, strlen(s->
1921                        p+=strlen(s->ctx->psk_identity_hint);
1922                        }
1923 #endif
1924
1925                /* not anonymous */
1926                if (pkey != NULL)
1927                        {
1928                        /* n is the length of the params, they start at &(d[4])
1929                         * and p points to the space at the end. */
1930 #ifndef OPENSSL_NO_RSA
1931                        if (pkey->type == EVP_PKEY_RSA
1932                                && TLS1_get_version(s) < TLS1_2_VERSION)
1933                                {
1934                                q=md_buf;
1935                                j=0;
1936                                for (num=2; num > 0; num--)
1937                                        {
1938                                        EVP_MD_CTX_set_flags(&md_ctx,
1939                                                EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
1940                                        EVP_DigestInit_ex(&md_ctx,(num == 2)
1941                                                ?s->ctx->md5:s->ctx->sha1, NULL)
1942                                        EVP_DigestUpdate(&md_ctx,&(s->s3->client
1943                                        EVP_DigestUpdate(&md_ctx,&(s->s3->server
1944                                        EVP_DigestUpdate(&md_ctx,&(d[4]),n);
1945                                        EVP_DigestFinal_ex(&md_ctx,q,
1946                                                (unsigned int *)&i);
1947                                        q+=i;
1948                                        j+=i;
1949                                        }
1950                                if (RSA_sign(NID_md5_sha1, md_buf, j,
1951                                        &(p[2]), &u, pkey->pkey.rsa) <= 0)
1952                                        {
1953                                        SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHAN
1954                                        goto err;
1955                                        }
1956                                s2n(u,p);
1957                                n+=u+2;
1958                                }
1959                        else
1960 #endif
1961                        if (md)
1962                                {
1963                                /* For TLS1.2 and later send signature
1964                                 * algorithm */
1965                                if (TLS1_get_version(s) >= TLS1_2_VERSION)
1966                                        {
1967                                        if (!tls12_get_sigandhash(p, pkey, md))
1968                                                {
1969                                                /* Should never happen */
1970                                                al=SSL_AD_INTERNAL_ERROR;
1971                                                SSLerr(SSL_F_SSL3_SEND_SERVER_KE
1972                                                goto f_err;
1973                                                }
1974                                        p+=2;
1975                                        }
```

```
#ifdef SSL_DEBUG
                                fprintf(stderr, "Using hash %s\n",
                                                        EVP_MD_name(md));
#endif
                                EVP_SignInit_ex(&md_ctx, md, NULL);
                                EVP_SignUpdate(&md_ctx,&(s->s3->client_random[0]
                                EVP_SignUpdate(&md_ctx,&(s->s3->server_random[0]
                                EVP_SignUpdate(&md_ctx,&(d[4]),n);
                                if (!EVP_SignFinal(&md_ctx,&(p[2]),
                                        (unsigned int *)&i,pkey))
                                        {
                                        SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHAN
                                        goto err;
                                        }
                                s2n(i,p);
                                n+=i+2;
                                if (TLS1_get_version(s) >= TLS1_2_VERSION)
                                        n+= 2;
                                }
                        else
                                {
                                /* Is this error check actually needed? */
                                al=SSL_AD_HANDSHAKE_FAILURE;
                                SSLerr(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE,SSL_R
                                goto f_err;
                                }
                        }

                *(d++)=SSL3_MT_SERVER_KEY_EXCHANGE;
                l2n3(n,d);

                /* we should now have things packed up, so lets send
                 * it off */
                s->init_num=n+4;
                s->init_off=0;
                }

        s->state = SSL3_ST_SW_KEY_EXCH_B;
        EVP_MD_CTX_cleanup(&md_ctx);
        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
f_err:
        ssl3_send_alert(s,SSL3_AL_FATAL,al);
err:
#ifndef OPENSSL_NO_ECDH
        if (encodedPoint != NULL) OPENSSL_free(encodedPoint);
        BN_CTX_free(bn_ctx);
#endif
        EVP_MD_CTX_cleanup(&md_ctx);
        return(-1);
        }

int ssl3_send_certificate_request(SSL *s)
        {
        unsigned char *p,*d;
        int i,j,nl,off,n;
        STACK_OF(X509_NAME) *sk=NULL;
        X509_NAME *name;
        BUF_MEM *buf;

        if (s->state == SSL3_ST_SW_CERT_REQ_A)
                {
                buf=s->init_buf;

                d=p=(unsigned char *)&(buf->data[4]);

                /* get the list of acceptable cert types */
```



```
                p++;
                n=ssl3_get_req_cert_type(s,p);
                d[0]=n;
                p+=n;
                n++;

                if (TLS1_get_version(s) >= TLS1_2_VERSION)
                        {
                        nl = tls12_get_req_sig_algs(s, p + 2);
                        s2n(nl, p);
                        p += nl + 2;
                        n += nl + 2;
                        }

                off=n;
                p+=2;
                n+=2;

                sk=SSL_get_client_CA_list(s);
                nl=0;
                if (sk != NULL)
                        {
                        for (i=0; i<sk_X509_NAME_num(sk); i++)
                                {
                                name=sk_X509_NAME_value(sk,i);
                                j=i2d_X509_NAME(name,NULL);
                                if (!BUF_MEM_grow_clean(buf,4+n+j+2))
                                        {
                                        SSLerr(SSL_F_SSL3_SEND_CERTIFICATE_REQUE
                                        goto err;
                                        }
                                p=(unsigned char *)&(buf->data[4+n]);
                                if (!(s->options & SSL_OP_NETSCAPE_CA_DN_BUG))
                                        {
                                        s2n(j,p);
                                        i2d_X509_NAME(name,&p);
                                        n+=2+j;
                                        nl+=2+j;
                                        }
                                else
                                        {
                                        d=p;
                                        i2d_X509_NAME(name,&p);
                                        j-=2; s2n(j,d); j+=2;
                                        n+=j;
                                        nl+=j;
                                        }
                                }
                        }
                /* else no CA names */
                p=(unsigned char *)&(buf->data[4+off]);
                s2n(nl,p);

                d=(unsigned char *)buf->data;
                *(d++)=SSL3_MT_CERTIFICATE_REQUEST;
                l2n3(n,d);

                /* we should now have things packed up, so lets send
                 * it off */

                s->init_num=n+4;
                s->init_off=0;
#ifdef NETSCAPE_HANG_BUG
                if (!BUF_MEM_grow_clean(buf, s->init_num + 4))
                        {
                        SSLerr(SSL_F_SSL3_SEND_CERTIFICATE_REQUEST,ERR_R_BUF_LIB
```

```
2108                        goto err;
2109                        }
2110                p=(unsigned char *)s->init_buf->data + s->init_num;

2112                /* do the header */
2113                *(p++)=SSL3_MT_SERVER_DONE;
2114                *(p++)=0;
2115                *(p++)=0;
2116                *(p++)=0;
2117                s->init_num += 4;
2118 #endif

2120                s->state = SSL3_ST_SW_CERT_REQ_B;
2121                }

2123        /* SSL3_ST_SW_CERT_REQ_B */
2124        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
2125 err:
2126        return(-1);
2127        }

2129 int ssl3_get_client_key_exchange(SSL *s)
2130        {
2131        int i,al,ok;
2132        long n;
2133        unsigned long alg_k;
2134        unsigned char *p;
2135 #ifndef OPENSSL_NO_RSA
2136        RSA *rsa=NULL;
2137        EVP_PKEY *pkey=NULL;
2138 #endif
2139 #ifndef OPENSSL_NO_DH
2140        BIGNUM *pub=NULL;
2141        DH *dh_srvr;
2142 #endif
2143 #ifndef OPENSSL_NO_KRB5
2144        KSSL_ERR kssl_err;
2145 #endif /* OPENSSL_NO_KRB5 */

2147 #ifndef OPENSSL_NO_ECDH
2148        EC_KEY *srvr_ecdh = NULL;
2149        EVP_PKEY *clnt_pub_pkey = NULL;
2150        EC_POINT *clnt_ecpoint = NULL;
2151        BN_CTX *bn_ctx = NULL;
2152 #endif

2154        n=s->method->ssl_get_message(s,
2155                SSL3_ST_SR_KEY_EXCH_A,
2156                SSL3_ST_SR_KEY_EXCH_B,
2157                SSL3_MT_CLIENT_KEY_EXCHANGE,
2158                2048, /* ??? */
2159                &ok);

2161        if (!ok) return((int)n);
2162        p=(unsigned char *)s->init_msg;

2164        alg_k=s->s3->tmp.new_cipher->algorithm_mkey;

2166 #ifndef OPENSSL_NO_RSA
2167        if (alg_k & SSL_kRSA)
2168                {
2169                /* FIX THIS UP EAY EAY EAY EAY */
2170                if (s->s3->tmp.use_rsa_tmp)
2171                        {
2172                        if ((s->cert != NULL) && (s->cert->rsa_tmp != NULL))
2173                                rsa=s->cert->rsa_tmp;
```



```
2174                        /* Don't do a callback because rsa_tmp should
2175                         * be sent already */
2176                        if (rsa == NULL)
2177                                {
2178                                al=SSL_AD_HANDSHAKE_FAILURE;
2179                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2180                                goto f_err;

2182                                }
2183                        }
2184                else
2185                        {
2186                        pkey=s->cert->pkeys[SSL_PKEY_RSA_ENC].privatekey;
2187                        if (    (pkey == NULL) ||
2188                                (pkey->type != EVP_PKEY_RSA) ||
2189                                (pkey->pkey.rsa == NULL))
2190                                {
2191                                al=SSL_AD_HANDSHAKE_FAILURE;
2192                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2193                                goto f_err;
2194                                }
2195                        rsa=pkey->pkey.rsa;
2196                        }

2198                /* TLS and [incidentally] DTLS{0xFEFF} */
2199                if (s->version > SSL3_VERSION && s->version != DTLS1_BAD_VER)
2200                        {
2201                        n2s(p,i);
2202                        if (n != i+2)
2203                                {
2204                                if (!(s->options & SSL_OP_TLS_D5_BUG))
2205                                        {
2206                                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANG
2207                                        goto err;
2208                                        }
2209                                else
2210                                        p-=2;
2211                                }
2212                        else
2213                                n=i;
2214                        }

2216                i=RSA_private_decrypt((int)n,p,p,rsa,RSA_PKCS1_PADDING);

2218                al = -1;
2219
2220                if (i != SSL_MAX_MASTER_KEY_LENGTH)
2221                        {
2222                        al=SSL_AD_DECODE_ERROR;
2223                        /* SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_BAD_R
2224                        }

2226                if ((al == -1) && !((p[0] == (s->client_version>>8)) && (p[1] ==
2227                        {
2228                        /* The premaster secret must contain the same version nu
2229                         * ClientHello to detect version rollback attacks (stran
2230                         * protocol does not offer such protection for DH cipher
2231                         * However, buggy clients exist that send the negotiated
2232                         * version instead if the server does not support the re
2233                         * protocol version.
2234                         * If SSL_OP_TLS_ROLLBACK_BUG is set, tolerate such clie
2235                        if (!((s->options & SSL_OP_TLS_ROLLBACK_BUG) &&
2236                                (p[0] == (s->version>>8)) && (p[1] == (s->versio
2237                                {
2238                                al=SSL_AD_DECODE_ERROR;
2239                                /* SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL
```

```
2241                                /* The Klima-Pokorny-Rosa extension of Bleichenb
2242                                 * (http://eprint.iacr.org/2003/052/) exploits t
2243                                 * number check as a "bad version oracle" -- an
2244                                 * reveal that the plaintext corresponding to so
2245                                 * made up by the adversary is properly formatte
2246                                 * that the version number is wrong.  To avoid s
2247                                 * we should treat this just like any other decr
2248                                */
2249                        }

2251                if (al != -1)
2252                        {
2253                        /* Some decryption failure -- use random value instead a
2254                         * against Bleichenbacher's attack on PKCS #1 v1.5 RSA p
2255                         * (see RFC 2246, section 7.4.7.1). */
2256                        ERR_clear_error();
2257                        i = SSL_MAX_MASTER_KEY_LENGTH;
2258                        p[0] = s->client_version >> 8;
2259                        p[1] = s->client_version & 0xff;
2260                        if (RAND_pseudo_bytes(p+2, i-2) <= 0) /* should be RAND_
2261                                goto err;
2262                        }
2263
2264                s->session->master_key_length=
2265                        s->method->ssl3_enc->generate_master_secret(s,
2266                                s->session->master_key,
2267                                p,i);
2268                OPENSSL_cleanse(p,i);
2269                }
2270        else
2271 #endif
2272 #ifndef OPENSSL_NO_DH
2273                if (alg_k & (SSL_kEDH|SSL_kDHr|SSL_kDHd))
2274                {
2275                n2s(p,i);
2276                if (n != i+2)
2277                        {
2278                        if (!(s->options & SSL_OP_SSLEAY_080_CLIENT_DH_BUG))
2279                                {
2280                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2281                                goto err;
2282                                }
2283                        else
2284                                {
2285                                p-=2;
2286                                i=(int)n;
2287                                }
2288                        }

2290                if (n == 0L) /* the parameters are in the cert */
2291                        {
2292                        al=SSL_AD_HANDSHAKE_FAILURE;
2293                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_UNABLE_T
2294                        goto f_err;
2295                        }
2296                else
2297                        {
2298                        if (s->s3->tmp.dh == NULL)
2299                                {
2300                                al=SSL_AD_HANDSHAKE_FAILURE;
2301                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2302                                goto f_err;
2303                                }
2304                        else
2305                                dh_srvr=s->s3->tmp.dh;
```



```
2306                        }

2308                pub=BN_bin2bn(p,i,NULL);
2309                if (pub == NULL)
2310                        {
2311                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_BN_LIB);
2312                        goto err;
2313                        }

2315                i=DH_compute_key(p,pub,dh_srvr);

2317                if (i <= 0)
2318                        {
2319                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,ERR_R_DH_LIB);
2320                        BN_clear_free(pub);
2321                        goto err;
2322                        }

2324                DH_free(s->s3->tmp.dh);
2325                s->s3->tmp.dh=NULL;

2327                BN_clear_free(pub);
2328                pub=NULL;
2329                s->session->master_key_length=
2330                        s->method->ssl3_enc->generate_master_secret(s,
2331                                s->session->master_key,p,i);
2332                OPENSSL_cleanse(p,i);
2333                }
2334        else
2335 #endif
2336 #ifndef OPENSSL_NO_KRB5
2337        if (alg_k & SSL_kKRB5)
2338                {
2339                krb5_error_code         krb5rc;
2340                krb5_data               enc_ticket;
2341                krb5_data               authenticator;
2342                krb5_data               enc_pms;
2343                KSSL_CTX                *kssl_ctx = s->kssl_ctx;
2344                EVP_CIPHER_CTX          ciph_ctx;
2345                const EVP_CIPHER        *enc = NULL;
2346                unsigned char           iv[EVP_MAX_IV_LENGTH];
2347                unsigned char           pms[SSL_MAX_MASTER_KEY_LENGTH
2348                                                + EVP_MAX_BLOCK_LENGTH];
2349                int                     padl, outl;
2350                krb5_timestamp          authtime = 0;
2351                krb5_ticket_times       ttimes;

2353                EVP_CIPHER_CTX_init(&ciph_ctx);

2355                if (!kssl_ctx)  kssl_ctx = kssl_ctx_new();

2357                n2s(p,i);
2358                enc_ticket.length = i;

2360                if (n < (long)(enc_ticket.length + 6))
2361                        {
2362                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2363                                SSL_R_DATA_LENGTH_TOO_LONG);
2364                        goto err;
2365                        }

2367                enc_ticket.data = (char *)p;
2368                p+=enc_ticket.length;

2370                n2s(p,i);
2371                authenticator.length = i;
```

```
2373                 if (n < (long)(enc_ticket.length + authenticator.length + 6))
2374                         {
2375                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2376                                 SSL_R_DATA_LENGTH_TOO_LONG);
2377                         goto err;
2378                         }

2380                 authenticator.data = (char *)p;
2381                 p+=authenticator.length;

2383                 n2s(p,i);
2384                 enc_pms.length = i;
2385                 enc_pms.data = (char *)p;
2386                 p+=enc_pms.length;

2388                 /* Note that the length is checked again below,
2389                 ** after decryption
2390                 */
2391                 if(enc_pms.length > sizeof pms)
2392                         {
2393                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2394                                SSL_R_DATA_LENGTH_TOO_LONG);
2395                         goto err;
2396                         }

2398                 if (n != (long)(enc_ticket.length + authenticator.length +
2399                                                 enc_pms.length + 6))
2400                         {
2401                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2402                                 SSL_R_DATA_LENGTH_TOO_LONG);
2403                         goto err;
2404                         }

2406                 if ((krb5rc = kssl_sget_tkt(kssl_ctx, &enc_ticket, &ttimes,
2407                                         &kssl_err)) != 0)
2408                         {
2409 #ifdef KSSL_DEBUG
2410                         printf("kssl_sget_tkt rtn %d [%d]\n",
2411                                 krb5rc, kssl_err.reason);
2412                         if (kssl_err.text)
2413                                 printf("kssl_err text= %s\n", kssl_err.text);
2414 #endif  /* KSSL_DEBUG */
2415                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2416                                 kssl_err.reason);
2417                         goto err;
2418                         }

2420                 /*  Note: no authenticator is not considered an error,
2421                 **  but will return authtime == 0.
2422                 */
2423                 if ((krb5rc = kssl_check_authent(kssl_ctx, &authenticator,
2424                                         &authtime, &kssl_err)) != 0)
2425                         {
2426 #ifdef KSSL_DEBUG
2427                         printf("kssl_check_authent rtn %d [%d]\n",
2428                                 krb5rc, kssl_err.reason);
2429                         if (kssl_err.text)
2430                                 printf("kssl_err text= %s\n", kssl_err.text);
2431 #endif  /* KSSL_DEBUG */
2432                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2433                                 kssl_err.reason);
2434                         goto err;
2435                         }

2437                 if ((krb5rc = kssl_validate_times(authtime, &ttimes)) != 0)
```



```
2438                         {
2439                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE, krb5rc);
2440                         goto err;
2441                         }

2443 #ifdef KSSL_DEBUG
2444                 kssl_ctx_show(kssl_ctx);
2445 #endif  /* KSSL_DEBUG */

2447                 enc = kssl_map_enc(kssl_ctx->enctype);
2448                 if (enc == NULL)
2449                     goto err;

2451                 memset(iv, 0, sizeof iv);       /* per RFC 1510 */

2453                 if (!EVP_DecryptInit_ex(&ciph_ctx,enc,NULL,kssl_ctx->key,iv))
2454                         {
2455                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2456                                 SSL_R_DECRYPTION_FAILED);
2457                         goto err;
2458                         }
2459                 if (!EVP_DecryptUpdate(&ciph_ctx, pms,&outl,
2460                                         (unsigned char *)enc_pms.data, enc_pms.l
2461                         {
2462                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2463                                 SSL_R_DECRYPTION_FAILED);
2464                         goto err;
2465                         }
2466                 if (outl > SSL_MAX_MASTER_KEY_LENGTH)
2467                         {
2468                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2469                                 SSL_R_DATA_LENGTH_TOO_LONG);
2470                         goto err;
2471                         }
2472                 if (!EVP_DecryptFinal_ex(&ciph_ctx,&(pms[outl]),&padl))
2473                         {
2474                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2475                                 SSL_R_DECRYPTION_FAILED);
2476                         goto err;
2477                         }
2478                 outl += padl;
2479                 if (outl > SSL_MAX_MASTER_KEY_LENGTH)
2480                         {
2481                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2482                                 SSL_R_DATA_LENGTH_TOO_LONG);
2483                         goto err;
2484                         }
2485                 if (!((pms[0] == (s->client_version>>8)) && (pms[1] == (s->clien
2486                     {
2487                     /* The premaster secret must contain the same version number
2488                      * ClientHello to detect version rollback attacks (strangely
2489                      * protocol does not offer such protection for DH ciphersuit
2490                      * However, buggy clients exist that send random bytes inste
2491                      * the protocol version.
2492                      * If SSL_OP_TLS_ROLLBACK_BUG is set, tolerate such clients.
2493                      * (Perhaps we should have a separate BUG value for the Kerb
2494                      */
2495                     if (!(s->options & SSL_OP_TLS_ROLLBACK_BUG))
2496                         {
2497                         SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2498                                SSL_AD_DECODE_ERROR);
2499                         goto err;
2500                         }
2501                     }

2503                 EVP_CIPHER_CTX_cleanup(&ciph_ctx);
```

```
2505                s->session->master_key_length=
2506                        s->method->ssl3_enc->generate_master_secret(s,
2507                                s->session->master_key, pms, outl);
2509                if (kssl_ctx->client_princ)
2510                        {
2511                        size_t len = strlen(kssl_ctx->client_princ);
2512                        if ( len < SSL_MAX_KRB5_PRINCIPAL_LENGTH )
2513                                {
2514                                s->session->krb5_client_princ_len = len;
2515                                memcpy(s->session->krb5_client_princ,kssl_ctx->c
2516                                }
2517                        }
2520                /*  Was doing kssl_ctx_free() here,
2521                **  but it caused problems for apache.
2522                **  kssl_ctx = kssl_ctx_free(kssl_ctx);
2523                **  if (s->kssl_ctx)  s->kssl_ctx = NULL;
2524                */
2525                }
2526        else
2527 #endif  /* OPENSSL_NO_KRB5 */
2529 #ifndef OPENSSL_NO_ECDH
2530                if (alg_k & (SSL_kEECDH|SSL_kECDHr|SSL_kECDHe))
2531                {
2532                int ret = 1;
2533                int field_size = 0;
2534                const EC_KEY   *tkey;
2535                const EC_GROUP *group;
2536                const BIGNUM *priv_key;
2538                /* initialize structures for server's ECDH key pair */
2539                if ((srvr_ecdh = EC_KEY_new()) == NULL)
2540                        {
2541                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2542                            ERR_R_MALLOC_FAILURE);
2543                        goto err;
2544                        }
2546                /* Let's get server private key and group information */
2547                if (alg_k & (SSL_kECDHr|SSL_kECDHe))
2548                        {
2549                        /* use the certificate */
2550                        tkey = s->cert->pkeys[SSL_PKEY_ECC].privatekey->pkey.ec;
2551                        }
2552                else
2553                        {
2554                        /* use the ephermeral values we saved when
2555                         * generating the ServerKeyExchange msg.
2556                         */
2557                        tkey = s->s3->tmp.ecdh;
2558                        }
2560                group    = EC_KEY_get0_group(tkey);
2561                priv_key = EC_KEY_get0_private_key(tkey);
2563                if (!EC_KEY_set_group(srvr_ecdh, group) ||
2564                    !EC_KEY_set_private_key(srvr_ecdh, priv_key))
2565                        {
2566                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2567                               ERR_R_EC_LIB);
2568                        goto err;
2569                        }
```



```
2571                /* Let's get client's public key */
2572                if ((clnt_ecpoint = EC_POINT_new(group)) == NULL)
2573                        {
2574                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2575                            ERR_R_MALLOC_FAILURE);
2576                        goto err;
2577                        }
2579                if (n == 0L)
2580                        {
2581                        /* Client Publickey was in Client Certificate */
2583                         if (alg_k & SSL_kEECDH)
2584                                 {
2585                                 al=SSL_AD_HANDSHAKE_FAILURE;
2586                                 SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R
2587                                 goto f_err;
2588                                 }
2589                        if (((clnt_pub_pkey=X509_get_pubkey(s->session->peer))
2590                            == NULL) ||
2591                            (clnt_pub_pkey->type != EVP_PKEY_EC))
2592                                {
2593                                /* XXX: For now, we do not support client
2594                                 * authentication using ECDH certificates
2595                                 * so this branch (n == 0L) of the code is
2596                                 * never executed. When that support is
2597                                 * added, we ought to ensure the key
2598                                 * received in the certificate is
2599                                 * authorized for key agreement.
2600                                 * ECDH_compute_key implicitly checks that
2601                                 * the two ECDH shares are for the same
2602                                 * group.
2603                                 */
2604                                al=SSL_AD_HANDSHAKE_FAILURE;
2605                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2606                                    SSL_R_UNABLE_TO_DECODE_ECDH_CERTS);
2607                                goto f_err;
2608                                }
2610                        if (EC_POINT_copy(clnt_ecpoint,
2611                            EC_KEY_get0_public_key(clnt_pub_pkey->pkey.ec)) == 0
2612                                {
2613                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2614                                       ERR_R_EC_LIB);
2615                                goto err;
2616                                }
2617                        ret = 2; /* Skip certificate verify processing */
2618                        }
2619                else
2620                        {
2621                        /* Get client's public key from encoded point
2622                         * in the ClientKeyExchange message.
2623                         */
2624                        if ((bn_ctx = BN_CTX_new()) == NULL)
2625                                {
2626                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2627                                    ERR_R_MALLOC_FAILURE);
2628                                goto err;
2629                                }
2631                        /* Get encoded point length */
2632                        i = *p;
2633                        p += 1;
2634                        if (n != 1 + i)
2635                                {
```

```
2636                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2637                                    ERR_R_EC_LIB);
2638                                goto err;
2639                                }
2640                        if (EC_POINT_oct2point(group,
2641                            clnt_ecpoint, p, i, bn_ctx) == 0)
2642                                {
2643                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2644                                    ERR_R_EC_LIB);
2645                                goto err;
2646                                }
2647                        /* p is pointing to somewhere in the buffer
2648                         * currently, so set it to the start
2649                         */
2650                        p=(unsigned char *)s->init_buf->data;
2651                        }

2653                /* Compute the shared pre-master secret */
2654                field_size = EC_GROUP_get_degree(group);
2655                if (field_size <= 0)
2656                        {
2657                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2658                               ERR_R_ECDH_LIB);
2659                        goto err;
2660                        }
2661                i = ECDH_compute_key(p, (field_size+7)/8, clnt_ecpoint, srvr_ecd
2662                if (i <= 0)
2663                        {
2664                        SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2665                            ERR_R_ECDH_LIB);
2666                        goto err;
2667                        }

2669                EVP_PKEY_free(clnt_pub_pkey);
2670                EC_POINT_free(clnt_ecpoint);
2671                EC_KEY_free(srvr_ecdh);
2672                BN_CTX_free(bn_ctx);
2673                EC_KEY_free(s->s3->tmp.ecdh);
2674                s->s3->tmp.ecdh = NULL;

2676                /* Compute the master secret */
2677                s->session->master_key_length = s->method->ssl3_enc-> \
2678                    generate_master_secret(s, s->session->master_key, p, i);
2679
2680                OPENSSL_cleanse(p, i);
2681                return (ret);
2682                }
2683        else
2684 #endif
2685 #ifndef OPENSSL_NO_PSK
2686                if (alg_k & SSL_kPSK)
2687                        {
2688                        unsigned char *t = NULL;
2689                        unsigned char psk_or_pre_ms[PSK_MAX_PSK_LEN*2+4];
2690                        unsigned int pre_ms_len = 0, psk_len = 0;
2691                        int psk_err = 1;
2692                        char tmp_id[PSK_MAX_IDENTITY_LEN+1];

2694                        al=SSL_AD_HANDSHAKE_FAILURE;

2696                        n2s(p,i);
2697                        if (n != i+2)
2698                                {
2699                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2700                                        SSL_R_LENGTH_MISMATCH);
2701                                goto psk_err;
```



```
2702                                }
2703                        if (i > PSK_MAX_IDENTITY_LEN)
2704                                {
2705                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2706                                        SSL_R_DATA_LENGTH_TOO_LONG);
2707                                goto psk_err;
2708                                }
2709                        if (s->psk_server_callback == NULL)
2710                                {
2711                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2712                                       SSL_R_PSK_NO_SERVER_CB);
2713                                goto psk_err;
2714                                }

2716                        /* Create guaranteed NULL-terminated identity
2717                         * string for the callback */
2718                        memcpy(tmp_id, p, i);
2719                        memset(tmp_id+i, 0, PSK_MAX_IDENTITY_LEN+1-i);
2720                        psk_len = s->psk_server_callback(s, tmp_id,
2721                                psk_or_pre_ms, sizeof(psk_or_pre_ms));
2722                        OPENSSL_cleanse(tmp_id, PSK_MAX_IDENTITY_LEN+1);

2724                        if (psk_len > PSK_MAX_PSK_LEN)
2725                                {
2726                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2727                                        ERR_R_INTERNAL_ERROR);
2728                                goto psk_err;
2729                                }
2730                        else if (psk_len == 0)
2731                                {
2732                                /* PSK related to the given identity not found *
2733                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2734                                        SSL_R_PSK_IDENTITY_NOT_FOUND);
2735                                al=SSL_AD_UNKNOWN_PSK_IDENTITY;
2736                                goto psk_err;
2737                                }

2739                        /* create PSK pre_master_secret */
2740                        pre_ms_len=2+psk_len+2+psk_len;
2741                        t = psk_or_pre_ms;
2742                        memmove(psk_or_pre_ms+psk_len+4, psk_or_pre_ms, psk_len)
2743                        s2n(psk_len, t);
2744                        memset(t, 0, psk_len);
2745                        t+=psk_len;
2746                        s2n(psk_len, t);

2748                        if (s->session->psk_identity != NULL)
2749                                OPENSSL_free(s->session->psk_identity);
2750                        s->session->psk_identity = BUF_strdup((char *)p);
2751                        if (s->session->psk_identity == NULL)
2752                                {
2753                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2754                                        ERR_R_MALLOC_FAILURE);
2755                                goto psk_err;
2756                                }

2758                        if (s->session->psk_identity_hint != NULL)
2759                                OPENSSL_free(s->session->psk_identity_hint);
2760                        s->session->psk_identity_hint = BUF_strdup(s->ctx->psk_i
2761                        if (s->ctx->psk_identity_hint != NULL &&
2762                                s->session->psk_identity_hint == NULL)
2763                                {
2764                                SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2765                                        ERR_R_MALLOC_FAILURE);
2766                                goto psk_err;
2767                                }
```

```
2769                            s->session->master_key_length=
2770                                    s->method->ssl3_enc->generate_master_secret(s,
2771                                            s->session->master_key, psk_or_pre_ms, p
2772                            psk_err = 0;
2773                    psk_err:
2774                            OPENSSL_cleanse(psk_or_pre_ms, sizeof(psk_or_pre_ms));
2775                            if (psk_err != 0)
2776                                    goto f_err;
2777                            }
2778                    else
2779 #endif
2780 #ifndef OPENSSL_NO_SRP
2781                    if (alg_k & SSL_kSRP)
2782                            {
2783                            int param_len;

2785                            n2s(p,i);
2786                            param_len=i+2;
2787                            if (param_len > n)
2788                                    {
2789                                    al=SSL_AD_DECODE_ERROR;
2790                                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2791                                    goto f_err;
2792                                    }
2793                            if (!(s->srp_ctx.A=BN_bin2bn(p,i,NULL)))
2794                                    {
2795                                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,ERR_R_
2796                                    goto err;
2797                                    }
2798                            if (BN_ucmp(s->srp_ctx.A, s->srp_ctx.N) >= 0
2799                                    || BN_is_zero(s->srp_ctx.A))
2800                                    {
2801                                    al=SSL_AD_ILLEGAL_PARAMETER;
2802                                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2803                                    goto f_err;
2804                                    }
2805                            if (s->session->srp_username != NULL)
2806                                    OPENSSL_free(s->session->srp_username);
2807                            s->session->srp_username = BUF_strdup(s->srp_ctx.login);
2808                            if (s->session->srp_username == NULL)
2809                                    {
2810                                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2811                                            ERR_R_MALLOC_FAILURE);
2812                                    goto err;
2813                                    }

2815                            if ((s->session->master_key_length = SRP_generate_server
2816                                    {
2817                                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,ERR_R_
2818                                    goto err;
2819                                    }

2821                            p+=i;
2822                            }
2823                    else
2824 #endif  /* OPENSSL_NO_SRP */
2825                    if (alg_k & SSL_kGOST)
2826                            {
2827                            int ret = 0;
2828                            EVP_PKEY_CTX *pkey_ctx;
2829                            EVP_PKEY *client_pub_pkey = NULL, *pk = NULL;
2830                            unsigned char premaster_secret[32], *start;
2831                            size_t outlen=32, inlen;
2832                            unsigned long alg_a;
2833                            int Ttag, Tclass;
```



```
2834                            long Tlen;

2836                            /* Get our certificate private key*/
2837                            alg_a = s->s3->tmp.new_cipher->algorithm_auth;
2838                            if (alg_a & SSL_aGOST94)
2839                                    pk = s->cert->pkeys[SSL_PKEY_GOST94].privatekey;
2840                            else if (alg_a & SSL_aGOST01)
2841                                    pk = s->cert->pkeys[SSL_PKEY_GOST01].privatekey;

2843                            pkey_ctx = EVP_PKEY_CTX_new(pk,NULL);
2844                            EVP_PKEY_decrypt_init(pkey_ctx);
2845                            /* If client certificate is present and is of the same t
2846                             * use it for key exchange.  Don't mind errors from
2847                             * EVP_PKEY_derive_set_peer, because it is completely va
2848                             * a client certificate for authorization only. */
2849                            client_pub_pkey = X509_get_pubkey(s->session->peer);
2850                            if (client_pub_pkey)
2851                                    {
2852                                    if (EVP_PKEY_derive_set_peer(pkey_ctx, client_pu
2853                                            ERR_clear_error();
2854                                    }
2855                            /* Decrypt session key */
2856                            if (ASN1_get_object((const unsigned char **)&p, &Tlen, &
2857                                    Ttag != V_ASN1_SEQUENCE ||
2858                                    Tclass != V_ASN1_UNIVERSAL)
2859                                    {
2860                                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2861                                    goto gerr;
2862                                    }
2863                            start = p;
2864                            inlen = Tlen;
2865                            if (EVP_PKEY_decrypt(pkey_ctx,premaster_secret,&outlen,s

2867                                    {
2868                                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,SSL_R_
2869                                    goto gerr;
2870                                    }
2871                            /* Generate master secret */
2872                            s->session->master_key_length=
2873                                    s->method->ssl3_enc->generate_master_secret(s,
2874                                            s->session->master_key,premaster_secret,
2875                            /* Check if pubkey from client certificate was used */
2876                            if (EVP_PKEY_CTX_ctrl(pkey_ctx, -1, -1, EVP_PKEY_CTRL_PE
2877                                    ret = 2;
2878                            else
2879                                    ret = 1;
2880                    gerr:
2881                            EVP_PKEY_free(client_pub_pkey);
2882                            EVP_PKEY_CTX_free(pkey_ctx);
2883                            if (ret)
2884                                    return ret;
2885                            else
2886                                    goto err;
2887                            }
2888                    else
2889                    {
2890                    al=SSL_AD_HANDSHAKE_FAILURE;
2891                    SSLerr(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE,
2892                                    SSL_R_UNKNOWN_CIPHER_TYPE);
2893                    goto f_err;
2894                    }

2896            return(1);
2897 f_err:
2898            ssl3_send_alert(s,SSL3_AL_FATAL,al);
2899 #if !defined(OPENSSL_NO_DH) || !defined(OPENSSL_NO_RSA) || !defined(OPENSSL_NO_E
```

```
2900 err:
2901 #endif
2902 #ifndef OPENSSL_NO_ECDH
2903         EVP_PKEY_free(clnt_pub_pkey);
2904         EC_POINT_free(clnt_ecpoint);
2905         if (srvr_ecdh != NULL)
2906                 EC_KEY_free(srvr_ecdh);
2907         BN_CTX_free(bn_ctx);
2908 #endif
2909         return(-1);
2910         }

2912 int ssl3_get_cert_verify(SSL *s)
2913         {
2914         EVP_PKEY *pkey=NULL;
2915         unsigned char *p;
2916         int al,ok,ret=0;
2917         long n;
2918         int type=0,i,j;
2919         X509 *peer;
2920         const EVP_MD *md = NULL;
2921         EVP_MD_CTX mctx;
2922         EVP_MD_CTX_init(&mctx);

2924         n=s->method->ssl_get_message(s,
2925                 SSL3_ST_SR_CERT_VRFY_A,
2926                 SSL3_ST_SR_CERT_VRFY_B,
2927                 -1,
2928                 SSL3_RT_MAX_PLAIN_LENGTH,
2929                 &ok);

2931         if (!ok) return((int)n);

2933         if (s->session->peer != NULL)
2934                 {
2935                 peer=s->session->peer;
2936                 pkey=X509_get_pubkey(peer);
2937                 type=X509_certificate_type(peer,pkey);
2938                 }
2939         else
2940                 {
2941                 peer=NULL;
2942                 pkey=NULL;
2943                 }

2945         if (s->s3->tmp.message_type != SSL3_MT_CERTIFICATE_VERIFY)
2946                 {
2947                 s->s3->tmp.reuse_message=1;
2948                 if ((peer != NULL) && (type & EVP_PKT_SIGN))
2949                         {
2950                         al=SSL_AD_UNEXPECTED_MESSAGE;
2951                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_MISSING_VERIFY_M
2952                         goto f_err;
2953                         }
2954                 ret=1;
2955                 goto end;
2956                 }

2958         if (peer == NULL)
2959                 {
2960                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_NO_CLIENT_CERT_RECEIVED)
2961                 al=SSL_AD_UNEXPECTED_MESSAGE;
2962                 goto f_err;
2963                 }

2965         if (!(type & EVP_PKT_SIGN))
```



```
2966                 {
2967                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_SIGNATURE_FOR_NON_SIGNIN
2968                 al=SSL_AD_ILLEGAL_PARAMETER;
2969                 goto f_err;
2970                 }

2972         if (s->s3->change_cipher_spec)
2973                 {
2974                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_CCS_RECEIVED_EARLY);
2975                 al=SSL_AD_UNEXPECTED_MESSAGE;
2976                 goto f_err;
2977                 }

2979         /* we now have a signature that we need to verify */
2980         p=(unsigned char *)s->init_msg;
2981         /* Check for broken implementations of GOST ciphersuites */
2982         /* If key is GOST and n is exactly 64, it is bare
2983          * signature without length field */
2984         if (n==64 && (pkey->type==NID_id_GostR3410_94 ||
2985                 pkey->type == NID_id_GostR3410_2001) )
2986                 {
2987                 i=64;
2988                 }
2989         else
2990                 {
2991                 if (TLS1_get_version(s) >= TLS1_2_VERSION)
2992                         {
2993                         int sigalg = tls12_get_sigid(pkey);
2994                         /* Should never happen */
2995                         if (sigalg == -1)
2996                                 {
2997                                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,ERR_R_INTERNAL
2998                                 al=SSL_AD_INTERNAL_ERROR;
2999                                 goto f_err;
3000                                 }
3001                         /* Check key type is consistent with signature */
3002                         if (sigalg != (int)p[1])
3003                                 {
3004                                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_WRONG_SI
3005                                 al=SSL_AD_DECODE_ERROR;
3006                                 goto f_err;
3007                                 }
3008                         md = tls12_get_hash(p[0]);
3009                         if (md == NULL)
3010                                 {
3011                                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_UNKNOWN_
3012                                 al=SSL_AD_DECODE_ERROR;
3013                                 goto f_err;
3014                                 }
3015 #ifdef SSL_DEBUG
3016 fprintf(stderr, "USING TLSv1.2 HASH %s\n", EVP_MD_name(md));
3017 #endif
3018                         p += 2;
3019                         n -= 2;
3020                         }
3021                 n2s(p,i);
3022                 n-=2;
3023                 if (i > n)
3024                         {
3025                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_LENGTH_MISMATCH)
3026                         al=SSL_AD_DECODE_ERROR;
3027                         goto f_err;
3028                         }
3029         }
3030         j=EVP_PKEY_size(pkey);
3031         if ((i > j) || (n > j) || (n <= 0))
```

```
3032                 {
3033                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_WRONG_SIGNATURE_SIZE);
3034                 al=SSL_AD_DECODE_ERROR;
3035                 goto f_err;
3036                 }

3038         if (TLS1_get_version(s) >= TLS1_2_VERSION)
3039                 {
3040                 long hdatalen = 0;
3041                 void *hdata;
3042                 hdatalen = BIO_get_mem_data(s->s3->handshake_buffer, &hdata);
3043                 if (hdatalen <= 0)
3044                         {
3045                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY, ERR_R_INTERNAL_ERROR)
3046                         al=SSL_AD_INTERNAL_ERROR;
3047                         goto f_err;
3048                         }
3049 #ifdef SSL_DEBUG
3050                 fprintf(stderr, "Using TLS 1.2 with client verify alg %s\n",
3051                                                         EVP_MD_name(md));
3052 #endif
3053                 if (!EVP_VerifyInit_ex(&mctx, md, NULL)
3054                         || !EVP_VerifyUpdate(&mctx, hdata, hdatalen))
3055                         {
3056                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY, ERR_R_EVP_LIB);
3057                         al=SSL_AD_INTERNAL_ERROR;
3058                         goto f_err;
3059                         }

3061                 if (EVP_VerifyFinal(&mctx, p , i, pkey) <= 0)
3062                         {
3063                         al=SSL_AD_DECRYPT_ERROR;
3064                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_BAD_SIGNATURE);
3065                         goto f_err;
3066                         }
3067                 }
3068         else
3069 #ifndef OPENSSL_NO_RSA
3070         if (pkey->type == EVP_PKEY_RSA)
3071                 {
3072                 i=RSA_verify(NID_md5_sha1, s->s3->tmp.cert_verify_md,
3073                         MD5_DIGEST_LENGTH+SHA_DIGEST_LENGTH, p, i,
3074                                                         pkey->pkey.rsa);
3075                 if (i < 0)
3076                         {
3077                         al=SSL_AD_DECRYPT_ERROR;
3078                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_BAD_RSA_DECRYPT)
3079                         goto f_err;
3080                         }
3081                 if (i == 0)
3082                         {
3083                         al=SSL_AD_DECRYPT_ERROR;
3084                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_BAD_RSA_SIGNATUR
3085                         goto f_err;
3086                         }
3087                 }
3088         else
3089 #endif
3090 #ifndef OPENSSL_NO_DSA
3091                 if (pkey->type == EVP_PKEY_DSA)
3092                 {
3093                 j=DSA_verify(pkey->save_type,
3094                         &(s->s3->tmp.cert_verify_md[MD5_DIGEST_LENGTH]),
3095                         SHA_DIGEST_LENGTH,p,i,pkey->pkey.dsa);
3096                 if (j <= 0)
3097                         {
```



```
3098                         /* bad signature */
3099                         al=SSL_AD_DECRYPT_ERROR;
3100                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,SSL_R_BAD_DSA_SIGNATUR
3101                         goto f_err;
3102                         }
3103                 }
3104         else
3105 #endif
3106 #ifndef OPENSSL_NO_ECDSA
3107                 if (pkey->type == EVP_PKEY_EC)
3108                 {
3109                 j=ECDSA_verify(pkey->save_type,
3110                         &(s->s3->tmp.cert_verify_md[MD5_DIGEST_LENGTH]),
3111                         SHA_DIGEST_LENGTH,p,i,pkey->pkey.ec);
3112                 if (j <= 0)
3113                         {
3114                         /* bad signature */
3115                         al=SSL_AD_DECRYPT_ERROR;
3116                         SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,
3117                             SSL_R_BAD_ECDSA_SIGNATURE);
3118                         goto f_err;
3119                         }
3120                 }
3121         else
3122 #endif
3123         if (pkey->type == NID_id_GostR3410_94 || pkey->type == NID_id_GostR3410_
3124                 {   unsigned char signature[64];
3125                         int idx;
3126                         EVP_PKEY_CTX *pctx = EVP_PKEY_CTX_new(pkey,NULL);
3127                         EVP_PKEY_verify_init(pctx);
3128                         if (i!=64) {
3129                                 fprintf(stderr,"GOST signature length is %d",i);
3130                         }
3131                         for (idx=0;idx<64;idx++) {
3132                                 signature[63-idx]=p[idx];
3133                         }
3134                         j=EVP_PKEY_verify(pctx,signature,64,s->s3->tmp.cert_veri
3135                         EVP_PKEY_CTX_free(pctx);
3136                         if (j<=0)
3137                                 {
3138                                 al=SSL_AD_DECRYPT_ERROR;
3139                                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,
3140                                         SSL_R_BAD_ECDSA_SIGNATURE);
3141                                 goto f_err;
3142                                 }
3143                 }
3144         else
3145                 {
3146                 SSLerr(SSL_F_SSL3_GET_CERT_VERIFY,ERR_R_INTERNAL_ERROR);
3147                 al=SSL_AD_UNSUPPORTED_CERTIFICATE;
3148                 goto f_err;
3149                 }


3152         ret=1;
3153         if (0)
3154                 {
3155 f_err:
3156                 ssl3_send_alert(s,SSL3_AL_FATAL,al);
3157                 }
3158 end:
3159         if (s->s3->handshake_buffer)
3160                 {
3161                 BIO_free(s->s3->handshake_buffer);
3162                 s->s3->handshake_buffer = NULL;
3163                 s->s3->flags &= ~TLS1_FLAGS_KEEP_HANDSHAKE;
```

```
3164                 }
3165         EVP_MD_CTX_cleanup(&mctx);
3166         EVP_PKEY_free(pkey);
3167         return(ret);
3168         }

3170 int ssl3_get_client_certificate(SSL *s)
3171         {
3172         int i,ok,al,ret= -1;
3173         X509 *x=NULL;
3174         unsigned long l,nc,llen,n;
3175         const unsigned char *p,*q;
3176         unsigned char *d;
3177         STACK_OF(X509) *sk=NULL;

3179         n=s->method->ssl_get_message(s,
3180                 SSL3_ST_SR_CERT_A,
3181                 SSL3_ST_SR_CERT_B,
3182                 -1,
3183                 s->max_cert_list,
3184                 &ok);

3186         if (!ok) return((int)n);

3188         if      (s->s3->tmp.message_type == SSL3_MT_CLIENT_KEY_EXCHANGE)
3189                 {
3190                 if (    (s->verify_mode & SSL_VERIFY_PEER) &&
3191                         (s->verify_mode & SSL_VERIFY_FAIL_IF_NO_PEER_CERT))
3192                         {
3193                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_PEER_DID_
3194                         al=SSL_AD_HANDSHAKE_FAILURE;
3195                         goto f_err;
3196                         }
3197                 /* If tls asked for a client cert, the client must return a 0 li
3198                 if ((s->version > SSL3_VERSION) && s->s3->tmp.cert_request)
3199                         {
3200                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_TLS_PEER_
3201                         al=SSL_AD_UNEXPECTED_MESSAGE;
3202                         goto f_err;
3203                         }
3204                 s->s3->tmp.reuse_message=1;
3205                 return(1);
3206                 }

3208         if (s->s3->tmp.message_type != SSL3_MT_CERTIFICATE)
3209                 {
3210                 al=SSL_AD_UNEXPECTED_MESSAGE;
3211                 SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_WRONG_MESSAGE_TYP
3212                 goto f_err;
3213                 }
3214         p=d=(unsigned char *)s->init_msg;

3216         if ((sk=sk_X509_new_null()) == NULL)
3217                 {
3218                 SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,ERR_R_MALLOC_FAILURE);
3219                 goto err;
3220                 }

3222         n2l3(p,llen);
3223         if (llen+3 != n)
3224                 {
3225                 al=SSL_AD_DECODE_ERROR;
3226                 SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_LENGTH_MISMATCH);
3227                 goto f_err;
3228                 }
3229         for (nc=0; nc<llen; )
```



```
3230                 {
3231                 n2l3(p,l);
3232                 if ((l+nc+3) > llen)
3233                         {
3234                         al=SSL_AD_DECODE_ERROR;
3235                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_CERT_LENG
3236                         goto f_err;
3237                         }

3239                 q=p;
3240                 x=d2i_X509(NULL,&p,l);
3241                 if (x == NULL)
3242                         {
3243                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,ERR_R_ASN1_LIB)
3244                         goto err;
3245                         }
3246                 if (p != (q+l))
3247                         {
3248                         al=SSL_AD_DECODE_ERROR;
3249                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_CERT_LENG
3250                         goto f_err;
3251                         }
3252                 if (!sk_X509_push(sk,x))
3253                         {
3254                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,ERR_R_MALLOC_FA
3255                         goto err;
3256                         }
3257                 x=NULL;
3258                 nc+=l+3;
3259                 }

3261         if (sk_X509_num(sk) <= 0)
3262                 {
3263                 /* TLS does not mind 0 certs returned */
3264                 if (s->version == SSL3_VERSION)
3265                         {
3266                         al=SSL_AD_HANDSHAKE_FAILURE;
3267                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_NO_CERTIF
3268                         goto f_err;
3269                         }
3270                 /* Fail for TLS only if we required a certificate */
3271                 else if ((s->verify_mode & SSL_VERIFY_PEER) &&
3272                          (s->verify_mode & SSL_VERIFY_FAIL_IF_NO_PEER_CERT))
3273                         {
3274                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_PEER_DID_
3275                         al=SSL_AD_HANDSHAKE_FAILURE;
3276                         goto f_err;
3277                         }
3278                 /* No client certificate so digest cached records */
3279                 if (s->s3->handshake_buffer && !ssl3_digest_cached_records(s))
3280                         {
3281                         al=SSL_AD_INTERNAL_ERROR;
3282                         goto f_err;
3283                         }
3284                 }
3285         else
3286                 {
3287                 i=ssl_verify_cert_chain(s,sk);
3288                 if (i <= 0)
3289                         {
3290                         al=ssl_verify_alarm_type(s->verify_result);
3291                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE,SSL_R_NO_CERTIF
3292                         goto f_err;
3293                         }
3294                 }
```

```
3296         if (s->session->peer != NULL) /* This should not be needed */
3297                 X509_free(s->session->peer);
3298         s->session->peer=sk_X509_shift(sk);
3299         s->session->verify_result = s->verify_result;

3301         /* With the current implementation, sess_cert will always be NULL
3302          * when we arrive here. */
3303         if (s->session->sess_cert == NULL)
3304                 {
3305                 s->session->sess_cert = ssl_sess_cert_new();
3306                 if (s->session->sess_cert == NULL)
3307                         {
3308                         SSLerr(SSL_F_SSL3_GET_CLIENT_CERTIFICATE, ERR_R_MALLOC_F
3309                         goto err;
3310                         }
3311                 }
3312         if (s->session->sess_cert->cert_chain != NULL)
3313                 sk_X509_pop_free(s->session->sess_cert->cert_chain, X509_free);
3314         s->session->sess_cert->cert_chain=sk;
3315         /* Inconsistency alert: cert_chain does *not* include the
3316          * peer's own certificate, while we do include it in s3_clnt.c */

3318         sk=NULL;

3320         ret=1;
3321         if (0)
3322                 {
3323 f_err:
3324                 ssl3_send_alert(s,SSL3_AL_FATAL,al);
3325                 }
3326 err:
3327         if (x != NULL) X509_free(x);
3328         if (sk != NULL) sk_X509_pop_free(sk,X509_free);
3329         return(ret);
3330         }

3332 int ssl3_send_server_certificate(SSL *s)
3333         {
3334         unsigned long l;
3335         X509 *x;

3337         if (s->state == SSL3_ST_SW_CERT_A)
3338                 {
3339                 x=ssl_get_server_send_cert(s);
3340                 if (x == NULL)
3341                         {
3342                         /* VRS: allow null cert if auth == KRB5 */
3343                         if ((s->s3->tmp.new_cipher->algorithm_auth != SSL_aKRB5)
3344                             (s->s3->tmp.new_cipher->algorithm_mkey & SSL_kKRB5))
3345                                 {
3346                                 SSLerr(SSL_F_SSL3_SEND_SERVER_CERTIFICATE,ERR_R_
3347                                 return(0);
3348                                 }
3349                         }

3351                 l=ssl3_output_cert_chain(s,x);
3352                 s->state=SSL3_ST_SW_CERT_B;
3353                 s->init_num=(int)l;
3354                 s->init_off=0;
3355                 }

3357         /* SSL3_ST_SW_CERT_B */
3358         return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
3359         }

3361 #ifndef OPENSSL_NO_TLSEXT
```



```
3362 /* send a new session ticket (not necessarily for a new session) */
3363 int ssl3_send_newsession_ticket(SSL *s)
3364         {
3365         if (s->state == SSL3_ST_SW_SESSION_TICKET_A)
3366                 {
3367                 unsigned char *p, *senc, *macstart;
3368                 const unsigned char *const_p;
3369                 int len, slen_full, slen;
3370                 SSL_SESSION *sess;
3371                 unsigned int hlen;
3372                 EVP_CIPHER_CTX ctx;
3373                 HMAC_CTX hctx;
3374                 SSL_CTX *tctx = s->initial_ctx;
3375                 unsigned char iv[EVP_MAX_IV_LENGTH];
3376                 unsigned char key_name[16];

3378                 /* get session encoding length */
3379                 slen_full = i2d_SSL_SESSION(s->session, NULL);
3380                 /* Some length values are 16 bits, so forget it if session is
3381                  * too long
3382                  */
3383                 if (slen_full > 0xFF00)
3384                         return -1;
3385                 senc = OPENSSL_malloc(slen_full);
3386                 if (!senc)
3387                         return -1;
3388                 p = senc;
3389                 i2d_SSL_SESSION(s->session, &p);

3391                 /* create a fresh copy (not shared with other threads) to clean
3392                 const_p = senc;
3393                 sess = d2i_SSL_SESSION(NULL, &const_p, slen_full);
3394                 if (sess == NULL)
3395                         {
3396                         OPENSSL_free(senc);
3397                         return -1;
3398                         }
3399                 sess->session_id_length = 0; /* ID is irrelevant for the ticket

3401                 slen = i2d_SSL_SESSION(sess, NULL);
3402                 if (slen > slen_full) /* shouldn't ever happen */
3403                         {
3404                         OPENSSL_free(senc);
3405                         return -1;
3406                         }
3407                 p = senc;
3408                 i2d_SSL_SESSION(sess, &p);
3409                 SSL_SESSION_free(sess);

3411                 /* Grow buffer if need be: the length calculation is as
3412                  * follows 1 (size of message name) + 3 (message length
3413                  * bytes) + 4 (ticket lifetime hint) + 2 (ticket length) +
3414                  * 16 (key name) + max_iv_len (iv length) +
3415                  * session_length + max_enc_block_size (max encrypted session
3416                  * length) + max_md_size (HMAC).
3417                  */
3418                 if (!BUF_MEM_grow(s->init_buf,
3419                         26 + EVP_MAX_IV_LENGTH + EVP_MAX_BLOCK_LENGTH +
3420                         EVP_MAX_MD_SIZE + slen))
3421                         return -1;

3423                 p=(unsigned char *)s->init_buf->data;
3424                 /* do the header */
3425                 *(p++)=SSL3_MT_NEWSESSION_TICKET;
3426                 /* Skip message length for now */
3427                 p += 3;
```

```
3428                EVP_CIPHER_CTX_init(&ctx);
3429                HMAC_CTX_init(&hctx);
3430                /* Initialize HMAC and cipher contexts. If callback present
3431                 * it does all the work otherwise use generated values
3432                 * from parent ctx.
3433                 */
3434                if (tctx->tlsext_ticket_key_cb)
3435                        {
3436                        if (tctx->tlsext_ticket_key_cb(s, key_name, iv, &ctx,
3437                                                       &hctx, 1) < 0)
3438                                {
3439                                OPENSSL_free(senc);
3440                                return -1;
3441                                }
3442                        }
3443                else
3444                        {
3445                        RAND_pseudo_bytes(iv, 16);
3446                        EVP_EncryptInit_ex(&ctx, EVP_aes_128_cbc(), NULL,
3447                                           tctx->tlsext_tick_aes_key, iv);
3448                        HMAC_Init_ex(&hctx, tctx->tlsext_tick_hmac_key, 16,
3449                                     tlsext_tick_md(), NULL);
3450                        memcpy(key_name, tctx->tlsext_tick_key_name, 16);
3451                        }

3453                /* Ticket lifetime hint (advisory only):
3454                 * We leave this unspecified for resumed session (for simplicity
3455                 * and guess that tickets for new sessions will live as long
3456                 * as their sessions. */
3457                l2n(s->hit ? 0 : s->session->timeout, p);

3459                /* Skip ticket length for now */
3460                p += 2;
3461                /* Output key name */
3462                macstart = p;
3463                memcpy(p, key_name, 16);
3464                p += 16;
3465                /* output IV */
3466                memcpy(p, iv, EVP_CIPHER_CTX_iv_length(&ctx));
3467                p += EVP_CIPHER_CTX_iv_length(&ctx);
3468                /* Encrypt session data */
3469                EVP_EncryptUpdate(&ctx, p, &len, senc, slen);
3470                p += len;
3471                EVP_EncryptFinal(&ctx, p, &len);
3472                p += len;
3473                EVP_CIPHER_CTX_cleanup(&ctx);

3475                HMAC_Update(&hctx, macstart, p - macstart);
3476                HMAC_Final(&hctx, p, &hlen);
3477                HMAC_CTX_cleanup(&hctx);

3479                p += hlen;
3480                /* Now write out lengths: p points to end of data written */
3481                /* Total length */
3482                len = p - (unsigned char *)s->init_buf->data;
3483                p=(unsigned char *)s->init_buf->data + 1;
3484                l2n3(len - 4, p); /* Message length */
3485                p += 4;
3486                s2n(len - 10, p);  /* Ticket length */

3488                /* number of bytes to write */
3489                s->init_num= len;
3490                s->state=SSL3_ST_SW_SESSION_TICKET_B;
3491                s->init_off=0;
3492                OPENSSL_free(senc);
3493                }
```



```
3495        /* SSL3_ST_SW_SESSION_TICKET_B */
3496        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
3497        }

3499 int ssl3_send_cert_status(SSL *s)
3500        {
3501        if (s->state == SSL3_ST_SW_CERT_STATUS_A)
3502                {
3503                unsigned char *p;
3504                /* Grow buffer if need be: the length calculation is as
3505                 * follows 1 (message type) + 3 (message length) +
3506                 * 1 (ocsp response type) + 3 (ocsp response length)
3507                 * + (ocsp response)
3508                 */
3509                if (!BUF_MEM_grow(s->init_buf, 8 + s->tlsext_ocsp_resplen))
3510                        return -1;

3512                p=(unsigned char *)s->init_buf->data;

3514                /* do the header */
3515                *(p++)=SSL3_MT_CERTIFICATE_STATUS;
3516                /* message length */
3517                l2n3(s->tlsext_ocsp_resplen + 4, p);
3518                /* status type */
3519                *(p++)= s->tlsext_status_type;
3520                /* length of OCSP response */
3521                l2n3(s->tlsext_ocsp_resplen, p);
3522                /* actual response */
3523                memcpy(p, s->tlsext_ocsp_resp, s->tlsext_ocsp_resplen);
3524                /* number of bytes to write */
3525                s->init_num = 8 + s->tlsext_ocsp_resplen;
3526                s->state=SSL3_ST_SW_CERT_STATUS_B;
3527                s->init_off = 0;
3528                }

3530        /* SSL3_ST_SW_CERT_STATUS_B */
3531        return(ssl3_do_write(s,SSL3_RT_HANDSHAKE));
3532        }

3534 # ifndef OPENSSL_NO_NEXTPROTONEG
3535 /* ssl3_get_next_proto reads a Next Protocol Negotiation handshake message. It
3536  * sets the next_proto member in s if found */
3537 int ssl3_get_next_proto(SSL *s)
3538        {
3539        int ok;
3540        int proto_len, padding_len;
3541        long n;
3542        const unsigned char *p;

3544        /* Clients cannot send a NextProtocol message if we didn't see the
3545         * extension in their ClientHello */
3546        if (!s->s3->next_proto_neg_seen)
3547                {
3548                SSLerr(SSL_F_SSL3_GET_NEXT_PROTO,SSL_R_GOT_NEXT_PROTO_WITHOUT_EX
3549                return -1;
3550                }

3552        n=s->method->ssl_get_message(s,
3553                SSL3_ST_SR_NEXT_PROTO_A,
3554                SSL3_ST_SR_NEXT_PROTO_B,
3555                SSL3_MT_NEXT_PROTO,
3556                514,  /* See the payload format below */
3557                &ok);

3559        if (!ok)
```

```
3560                 return((int)n);

3562         /* s->state doesn't reflect whether ChangeCipherSpec has been received
3563          * in this handshake, but s->s3->change_cipher_spec does (will be reset
3564          * by ssl3_get_finished). */
3565         if (!s->s3->change_cipher_spec)
3566                 {
3567                 SSLerr(SSL_F_SSL3_GET_NEXT_PROTO,SSL_R_GOT_NEXT_PROTO_BEFORE_A_C
3568                 return -1;
3569                 }

3571         if (n < 2)
3572                 return 0;  /* The body must be > 1 bytes long */

3574         p=(unsigned char *)s->init_msg;

3576         /* The payload looks like:
3577          *   uint8 proto_len;
3578          *   uint8 proto[proto_len];
3579          *   uint8 padding_len;
3580          *   uint8 padding[padding_len];
3581          */
3582         proto_len = p[0];
3583         if (proto_len + 2 > s->init_num)
3584                 return 0;
3585         padding_len = p[proto_len + 1];
3586         if (proto_len + padding_len + 2 != s->init_num)
3587                 return 0;

3589         s->next_proto_negotiated = OPENSSL_malloc(proto_len);
3590         if (!s->next_proto_negotiated)
3591                 {
3592                 SSLerr(SSL_F_SSL3_GET_NEXT_PROTO,ERR_R_MALLOC_FAILURE);
3593                 return 0;
3594                 }
3595         memcpy(s->next_proto_negotiated, p + 1, proto_len);
3596         s->next_proto_negotiated_len = proto_len;

3598         return 1;
3599         }
3600 # endif
3601 #endif
3602 #endif /* ! codereview */
```

```
**********************************************************
    5708 Wed Aug 13 19:53:39 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_algs.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_algs.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/objects.h>
#include <openssl/lhash.h>
```



```
#include "ssl_locl.h"

int SSL_library_init(void)
        {

#ifndef OPENSSL_NO_DES
        EVP_add_cipher(EVP_des_cbc());
        EVP_add_cipher(EVP_des_ede3_cbc());
#endif
#ifndef OPENSSL_NO_IDEA
        EVP_add_cipher(EVP_idea_cbc());
#endif
#ifndef OPENSSL_NO_RC4
        EVP_add_cipher(EVP_rc4());
#if !defined(OPENSSL_NO_MD5) && (defined(__x86_64) || defined(__x86_64__))
        EVP_add_cipher(EVP_rc4_hmac_md5());
#endif
#endif
#ifndef OPENSSL_NO_RC2
        EVP_add_cipher(EVP_rc2_cbc());
        /* Not actually used for SSL/TLS but this makes PKCS#12 work
         * if an application only calls SSL_library_init().
         */
        EVP_add_cipher(EVP_rc2_40_cbc());
#endif
#ifndef OPENSSL_NO_AES
        EVP_add_cipher(EVP_aes_128_cbc());
        EVP_add_cipher(EVP_aes_192_cbc());
        EVP_add_cipher(EVP_aes_256_cbc());
        EVP_add_cipher(EVP_aes_128_gcm());
        EVP_add_cipher(EVP_aes_256_gcm());
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_SHA1)
        EVP_add_cipher(EVP_aes_128_cbc_hmac_sha1());
        EVP_add_cipher(EVP_aes_256_cbc_hmac_sha1());
#endif

#endif
#ifndef OPENSSL_NO_CAMELLIA
        EVP_add_cipher(EVP_camellia_128_cbc());
        EVP_add_cipher(EVP_camellia_256_cbc());
#endif

#ifndef OPENSSL_NO_SEED
        EVP_add_cipher(EVP_seed_cbc());
#endif

#ifndef OPENSSL_NO_MD5
        EVP_add_digest(EVP_md5());
        EVP_add_digest_alias(SN_md5,"ssl2-md5");
        EVP_add_digest_alias(SN_md5,"ssl3-md5");
#endif
#ifndef OPENSSL_NO_SHA
        EVP_add_digest(EVP_sha1()); /* RSA with sha1 */
        EVP_add_digest_alias(SN_sha1,"ssl3-sha1");
        EVP_add_digest_alias(SN_sha1WithRSAEncryption,SN_sha1WithRSA);
#endif
#ifndef OPENSSL_NO_SHA256
        EVP_add_digest(EVP_sha224());
        EVP_add_digest(EVP_sha256());
#endif
#ifndef OPENSSL_NO_SHA512
        EVP_add_digest(EVP_sha384());
        EVP_add_digest(EVP_sha512());
#endif
#if !defined(OPENSSL_NO_SHA) && !defined(OPENSSL_NO_DSA)
        EVP_add_digest(EVP_dss1()); /* DSA with sha1 */
```

```
128         EVP_add_digest_alias(SN_dsaWithSHA1,SN_dsaWithSHA1_2);
129         EVP_add_digest_alias(SN_dsaWithSHA1,"DSS1");
130         EVP_add_digest_alias(SN_dsaWithSHA1,"dss1");
131 #endif
132 #ifndef OPENSSL_NO_ECDSA
133         EVP_add_digest(EVP_ecdsa());
134 #endif
135         /* If you want support for phased out ciphers, add the following */
136 #if 0
137         EVP_add_digest(EVP_sha());
138         EVP_add_digest(EVP_dss());
139 #endif
140 #ifndef OPENSSL_NO_COMP
141         /* This will initialise the built-in compression algorithms.
142            The value returned is a STACK_OF(SSL_COMP), but that can
143            be discarded safely */
144         (void)SSL_COMP_get_compression_methods();
145 #endif
146         /* initialize cipher/digest methods table */
147         ssl_load_ciphers();
148         return(1);
149         }
150 #endif /* ! codereview */
```

```
**********************************************************
   19964 Wed Aug 13 19:53:39 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_asn1.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_asn1.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
```



```
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include <stdlib.h>
#include "ssl_locl.h"
#include <openssl/asn1_mac.h>
#include <openssl/objects.h>
#include <openssl/x509.h>

typedef struct ssl_session_asn1_st
        {
        ASN1_INTEGER version;
        ASN1_INTEGER ssl_version;
        ASN1_OCTET_STRING cipher;
        ASN1_OCTET_STRING comp_id;
        ASN1_OCTET_STRING master_key;
        ASN1_OCTET_STRING session_id;
        ASN1_OCTET_STRING session_id_context;
        ASN1_OCTET_STRING key_arg;
#ifndef OPENSSL_NO_KRB5
        ASN1_OCTET_STRING krb5_princ;
#endif /* OPENSSL_NO_KRB5 */
        ASN1_INTEGER time;
        ASN1_INTEGER timeout;
        ASN1_INTEGER verify_result;
#ifndef OPENSSL_NO_TLSEXT
        ASN1_OCTET_STRING tlsext_hostname;
        ASN1_INTEGER tlsext_tick_lifetime;
        ASN1_OCTET_STRING tlsext_tick;
#endif /* OPENSSL_NO_TLSEXT */
#ifndef OPENSSL_NO_PSK
        ASN1_OCTET_STRING psk_identity_hint;
        ASN1_OCTET_STRING psk_identity;
#endif /* OPENSSL_NO_PSK */
#ifndef OPENSSL_NO_SRP
        ASN1_OCTET_STRING srp_username;
#endif /* OPENSSL_NO_SRP */
        } SSL_SESSION_ASN1;

int i2d_SSL_SESSION(SSL_SESSION *in, unsigned char **pp)
        {
#define LSIZE2 (sizeof(long)*2)
        int v1=0,v2=0,v3=0,v4=0,v5=0,v7=0,v8=0;
        unsigned char buf[4],ibuf1[LSIZE2],ibuf2[LSIZE2];
        unsigned char ibuf3[LSIZE2],ibuf4[LSIZE2],ibuf5[LSIZE2];
```

```
#ifndef OPENSSL_NO_TLSEXT
        int v6=0,v9=0,v10=0;
        unsigned char ibuf6[LSIZE2];
#endif
#ifndef OPENSSL_NO_COMP
        unsigned char cbuf;
        int v11=0;
#endif
#ifndef OPENSSL_NO_SRP
        int v12=0;
#endif
        long l;
        SSL_SESSION_ASN1 a;
        M_ASN1_I2D_vars(in);

        if ((in == NULL) || ((in->cipher == NULL) && (in->cipher_id == 0)))
                return(0);

        /* Note that I cheat in the following 2 assignments.  I know
         * that if the ASN1_INTEGER passed to ASN1_INTEGER_set
         * is > sizeof(long)+1, the buffer will not be re-OPENSSL_malloc()ed.
         * This is a bit evil but makes things simple, no dynamic allocation
         * to clean up :-) */
        a.version.length=LSIZE2;
        a.version.type=V_ASN1_INTEGER;
        a.version.data=ibuf1;
        ASN1_INTEGER_set(&(a.version),SSL_SESSION_ASN1_VERSION);

        a.ssl_version.length=LSIZE2;
        a.ssl_version.type=V_ASN1_INTEGER;
        a.ssl_version.data=ibuf2;
        ASN1_INTEGER_set(&(a.ssl_version),in->ssl_version);

        a.cipher.type=V_ASN1_OCTET_STRING;
        a.cipher.data=buf;

        if (in->cipher == NULL)
                l=in->cipher_id;
        else
                l=in->cipher->id;
        if (in->ssl_version == SSL2_VERSION)
                {
                a.cipher.length=3;
                buf[0]=((unsigned char)(l>>16L))&0xff;
                buf[1]=((unsigned char)(l>> 8L))&0xff;
                buf[2]=((unsigned char)(l     ))&0xff;
                }
        else
                {
                a.cipher.length=2;
                buf[0]=((unsigned char)(l>>8L))&0xff;
                buf[1]=((unsigned char)(l    ))&0xff;
                }

#ifndef OPENSSL_NO_COMP
        if (in->compress_meth)
                {
                cbuf = (unsigned char)in->compress_meth;
                a.comp_id.length = 1;
                a.comp_id.type = V_ASN1_OCTET_STRING;
                a.comp_id.data = &cbuf;
                }
#endif

        a.master_key.length=in->master_key_length;
        a.master_key.type=V_ASN1_OCTET_STRING;
```



```
        a.master_key.data=in->master_key;

        a.session_id.length=in->session_id_length;
        a.session_id.type=V_ASN1_OCTET_STRING;
        a.session_id.data=in->session_id;

        a.session_id_context.length=in->sid_ctx_length;
        a.session_id_context.type=V_ASN1_OCTET_STRING;
        a.session_id_context.data=in->sid_ctx;

        a.key_arg.length=in->key_arg_length;
        a.key_arg.type=V_ASN1_OCTET_STRING;
        a.key_arg.data=in->key_arg;

#ifndef OPENSSL_NO_KRB5
        if (in->krb5_client_princ_len)
                {
                a.krb5_princ.length=in->krb5_client_princ_len;
                a.krb5_princ.type=V_ASN1_OCTET_STRING;
                a.krb5_princ.data=in->krb5_client_princ;
                }
#endif /* OPENSSL_NO_KRB5 */

        if (in->time != 0L)
                {
                a.time.length=LSIZE2;
                a.time.type=V_ASN1_INTEGER;
                a.time.data=ibuf3;
                ASN1_INTEGER_set(&(a.time),in->time);
                }

        if (in->timeout != 0L)
                {
                a.timeout.length=LSIZE2;
                a.timeout.type=V_ASN1_INTEGER;
                a.timeout.data=ibuf4;
                ASN1_INTEGER_set(&(a.timeout),in->timeout);
                }

        if (in->verify_result != X509_V_OK)
                {
                a.verify_result.length=LSIZE2;
                a.verify_result.type=V_ASN1_INTEGER;
                a.verify_result.data=ibuf5;
                ASN1_INTEGER_set(&a.verify_result,in->verify_result);
                }

#ifndef OPENSSL_NO_TLSEXT
        if (in->tlsext_hostname)
                {
                a.tlsext_hostname.length=strlen(in->tlsext_hostname);
                a.tlsext_hostname.type=V_ASN1_OCTET_STRING;
                a.tlsext_hostname.data=(unsigned char *)in->tlsext_hostname;
                }
        if (in->tlsext_tick)
                {
                a.tlsext_tick.length= in->tlsext_ticklen;
                a.tlsext_tick.type=V_ASN1_OCTET_STRING;
                a.tlsext_tick.data=(unsigned char *)in->tlsext_tick;
                }
        if (in->tlsext_tick_lifetime_hint > 0)
                {
                a.tlsext_tick_lifetime.length=LSIZE2;
                a.tlsext_tick_lifetime.type=V_ASN1_INTEGER;
                a.tlsext_tick_lifetime.data=ibuf6;
                ASN1_INTEGER_set(&a.tlsext_tick_lifetime,in->tlsext_tick_lifetim
```

```
260                 }
261 #endif /* OPENSSL_NO_TLSEXT */
262 #ifndef OPENSSL_NO_PSK
263         if (in->psk_identity_hint)
264                 {
265                 a.psk_identity_hint.length=strlen(in->psk_identity_hint);
266                 a.psk_identity_hint.type=V_ASN1_OCTET_STRING;
267                 a.psk_identity_hint.data=(unsigned char *)(in->psk_identity_hint
268                 }
269         if (in->psk_identity)
270                 {
271                 a.psk_identity.length=strlen(in->psk_identity);
272                 a.psk_identity.type=V_ASN1_OCTET_STRING;
273                 a.psk_identity.data=(unsigned char *)(in->psk_identity);
274                 }
275 #endif /* OPENSSL_NO_PSK */
276 #ifndef OPENSSL_NO_SRP
277         if (in->srp_username)
278                 {
279                 a.srp_username.length=strlen(in->srp_username);
280                 a.srp_username.type=V_ASN1_OCTET_STRING;
281                 a.srp_username.data=(unsigned char *)(in->srp_username);
282                 }
283 #endif /* OPENSSL_NO_SRP */

285         M_ASN1_I2D_len(&(a.version),            i2d_ASN1_INTEGER);
286         M_ASN1_I2D_len(&(a.ssl_version),        i2d_ASN1_INTEGER);
287         M_ASN1_I2D_len(&(a.cipher),             i2d_ASN1_OCTET_STRING);
288         M_ASN1_I2D_len(&(a.session_id),         i2d_ASN1_OCTET_STRING);
289         M_ASN1_I2D_len(&(a.master_key),         i2d_ASN1_OCTET_STRING);
290 #ifndef OPENSSL_NO_KRB5
291         if (in->krb5_client_princ_len)
292                 M_ASN1_I2D_len(&(a.krb5_princ), i2d_ASN1_OCTET_STRING);
293 #endif /* OPENSSL_NO_KRB5 */
294         if (in->key_arg_length > 0)
295                 M_ASN1_I2D_len_IMP_opt(&(a.key_arg),i2d_ASN1_OCTET_STRING);
296         if (in->time != 0L)
297                 M_ASN1_I2D_len_EXP_opt(&(a.time),i2d_ASN1_INTEGER,1,v1);
298         if (in->timeout != 0L)
299                 M_ASN1_I2D_len_EXP_opt(&(a.timeout),i2d_ASN1_INTEGER,2,v2);
300         if (in->peer != NULL)
301                 M_ASN1_I2D_len_EXP_opt(in->peer,i2d_X509,3,v3);
302         M_ASN1_I2D_len_EXP_opt(&a.session_id_context,i2d_ASN1_OCTET_STRING,4,v4)
303         if (in->verify_result != X509_V_OK)
304                 M_ASN1_I2D_len_EXP_opt(&(a.verify_result),i2d_ASN1_INTEGER,5,v5)

306 #ifndef OPENSSL_NO_TLSEXT
307         if (in->tlsext_tick_lifetime_hint > 0)
308                 M_ASN1_I2D_len_EXP_opt(&a.tlsext_tick_lifetime, i2d_ASN1_INTEGER
309         if (in->tlsext_tick)
310                 M_ASN1_I2D_len_EXP_opt(&(a.tlsext_tick), i2d_ASN1_OCTET_STRING,1
311         if (in->tlsext_hostname)
312                 M_ASN1_I2D_len_EXP_opt(&(a.tlsext_hostname), i2d_ASN1_OCTET_STRI
313 #ifndef OPENSSL_NO_COMP
314         if (in->compress_meth)
315                 M_ASN1_I2D_len_EXP_opt(&(a.comp_id), i2d_ASN1_OCTET_STRING,11,v1
316 #endif
317 #endif /* OPENSSL_NO_TLSEXT */
318 #ifndef OPENSSL_NO_PSK
319         if (in->psk_identity_hint)
320                 M_ASN1_I2D_len_EXP_opt(&(a.psk_identity_hint), i2d_ASN1_OCTET_ST
321         if (in->psk_identity)
322                 M_ASN1_I2D_len_EXP_opt(&(a.psk_identity), i2d_ASN1_OCTET_STRING,
323 #endif /* OPENSSL_NO_PSK */
324 #ifndef OPENSSL_NO_SRP
325         if (in->srp_username)
```



```
326                 M_ASN1_I2D_len_EXP_opt(&(a.srp_username), i2d_ASN1_OCTET_STRING,
327 #endif /* OPENSSL_NO_SRP */

329         M_ASN1_I2D_seq_total();

331         M_ASN1_I2D_put(&(a.version),            i2d_ASN1_INTEGER);
332         M_ASN1_I2D_put(&(a.ssl_version),        i2d_ASN1_INTEGER);
333         M_ASN1_I2D_put(&(a.cipher),             i2d_ASN1_OCTET_STRING);
334         M_ASN1_I2D_put(&(a.session_id),         i2d_ASN1_OCTET_STRING);
335         M_ASN1_I2D_put(&(a.master_key),         i2d_ASN1_OCTET_STRING);
336 #ifndef OPENSSL_NO_KRB5
337         if (in->krb5_client_princ_len)
338                 M_ASN1_I2D_put(&(a.krb5_princ), i2d_ASN1_OCTET_STRING);
339 #endif /* OPENSSL_NO_KRB5 */
340         if (in->key_arg_length > 0)
341                 M_ASN1_I2D_put_IMP_opt(&(a.key_arg),i2d_ASN1_OCTET_STRING,0);
342         if (in->time != 0L)
343                 M_ASN1_I2D_put_EXP_opt(&(a.time),i2d_ASN1_INTEGER,1,v1);
344         if (in->timeout != 0L)
345                 M_ASN1_I2D_put_EXP_opt(&(a.timeout),i2d_ASN1_INTEGER,2,v2);
346         if (in->peer != NULL)
347                 M_ASN1_I2D_put_EXP_opt(in->peer,i2d_X509,3,v3);
348         M_ASN1_I2D_put_EXP_opt(&a.session_id_context,i2d_ASN1_OCTET_STRING,4,
349                         v4);
350         if (in->verify_result != X509_V_OK)
351                 M_ASN1_I2D_put_EXP_opt(&a.verify_result,i2d_ASN1_INTEGER,5,v5);
352 #ifndef OPENSSL_NO_TLSEXT
353         if (in->tlsext_hostname)
354                 M_ASN1_I2D_put_EXP_opt(&(a.tlsext_hostname), i2d_ASN1_OCTET_STRI
355 #endif /* OPENSSL_NO_TLSEXT */
356 #ifndef OPENSSL_NO_PSK
357         if (in->psk_identity_hint)
358                 M_ASN1_I2D_put_EXP_opt(&(a.psk_identity_hint), i2d_ASN1_OCTET_ST
359         if (in->psk_identity)
360                 M_ASN1_I2D_put_EXP_opt(&(a.psk_identity), i2d_ASN1_OCTET_STRING,
361 #endif /* OPENSSL_NO_PSK */
362 #ifndef OPENSSL_NO_TLSEXT
363         if (in->tlsext_tick_lifetime_hint > 0)
364                 M_ASN1_I2D_put_EXP_opt(&a.tlsext_tick_lifetime, i2d_ASN1_INTEGER
365         if (in->tlsext_tick)
366                 M_ASN1_I2D_put_EXP_opt(&(a.tlsext_tick), i2d_ASN1_OCTET_STRING,1
367 #endif /* OPENSSL_NO_TLSEXT */
368 #ifndef OPENSSL_NO_COMP
369         if (in->compress_meth)
370                 M_ASN1_I2D_put_EXP_opt(&(a.comp_id), i2d_ASN1_OCTET_STRING,11,v1
371 #endif
372 #ifndef OPENSSL_NO_SRP
373         if (in->srp_username)
374                 M_ASN1_I2D_put_EXP_opt(&(a.srp_username), i2d_ASN1_OCTET_STRING,
375 #endif /* OPENSSL_NO_SRP */
376         M_ASN1_I2D_finish();
377         }

379 SSL_SESSION *d2i_SSL_SESSION(SSL_SESSION **a, const unsigned char **pp,
380                              long length)
381         {
382         int ssl_version=0,i;
383         long id;
384         ASN1_INTEGER ai,*aip;
385         ASN1_OCTET_STRING os,*osp;
386         M_ASN1_D2I_vars(a,SSL_SESSION *,SSL_SESSION_new);

388         aip= &ai;
389         osp= &os;

391         M_ASN1_D2I_Init();
```

```
392         M_ASN1_D2I_start_sequence();

394         ai.data=NULL; ai.length=0;
395         M_ASN1_D2I_get_x(ASN1_INTEGER,aip,d2i_ASN1_INTEGER);
396         if (ai.data != NULL) { OPENSSL_free(ai.data); ai.data=NULL; ai.length=0;

398         /* we don't care about the version right now :-) */
399         M_ASN1_D2I_get_x(ASN1_INTEGER,aip,d2i_ASN1_INTEGER);
400         ssl_version=(int)ASN1_INTEGER_get(aip);
401         ret->ssl_version=ssl_version;
402         if (ai.data != NULL) { OPENSSL_free(ai.data); ai.data=NULL; ai.length=0;

404         os.data=NULL; os.length=0;
405         M_ASN1_D2I_get_x(ASN1_OCTET_STRING,osp,d2i_ASN1_OCTET_STRING);
406         if (ssl_version == SSL2_VERSION)
407                 {
408                 if (os.length != 3)
409                         {
410                         c.error=SSL_R_CIPHER_CODE_WRONG_LENGTH;
411                         c.line=__LINE__;
412                         goto err;
413                         }
414                 id=0x02000000L|
415                         ((unsigned long)os.data[0]<<16L)|
416                         ((unsigned long)os.data[1]<< 8L)|
417                          (unsigned long)os.data[2];
418                 }
419         else if ((ssl_version>>8) >= SSL3_VERSION_MAJOR)
420                 {
421                 if (os.length != 2)
422                         {
423                         c.error=SSL_R_CIPHER_CODE_WRONG_LENGTH;
424                         c.line=__LINE__;
425                         goto err;
426                         }
427                 id=0x03000000L|
428                         ((unsigned long)os.data[0]<<8L)|
429                          (unsigned long)os.data[1];
430                 }
431         else
432                 {
433                 c.error=SSL_R_UNKNOWN_SSL_VERSION;
434                 c.line=__LINE__;
435                 goto err;
436                 }

438         ret->cipher=NULL;
439         ret->cipher_id=id;

441         M_ASN1_D2I_get_x(ASN1_OCTET_STRING,osp,d2i_ASN1_OCTET_STRING);
442         if ((ssl_version>>8) >= SSL3_VERSION_MAJOR)
443                 i=SSL3_MAX_SSL_SESSION_ID_LENGTH;
444         else /* if (ssl_version>>8 == SSL2_VERSION_MAJOR) */
445                 i=SSL2_MAX_SSL_SESSION_ID_LENGTH;

447         if (os.length > i)
448                 os.length = i;
449         if (os.length > (int)sizeof(ret->session_id)) /* can't happen */
450                 os.length = sizeof(ret->session_id);

452         ret->session_id_length=os.length;
453         OPENSSL_assert(os.length <= (int)sizeof(ret->session_id));
454         memcpy(ret->session_id,os.data,os.length);

456         M_ASN1_D2I_get_x(ASN1_OCTET_STRING,osp,d2i_ASN1_OCTET_STRING);
457         if (os.length > SSL_MAX_MASTER_KEY_LENGTH)
```



```
458                 ret->master_key_length=SSL_MAX_MASTER_KEY_LENGTH;
459         else
460                 ret->master_key_length=os.length;
461         memcpy(ret->master_key,os.data,ret->master_key_length);

463         os.length=0;

465 #ifndef OPENSSL_NO_KRB5
466         os.length=0;
467         M_ASN1_D2I_get_opt(osp,d2i_ASN1_OCTET_STRING,V_ASN1_OCTET_STRING);
468         if (os.data)
469                 {
470                 if (os.length > SSL_MAX_KRB5_PRINCIPAL_LENGTH)
471                         ret->krb5_client_princ_len=0;
472                 else
473                         ret->krb5_client_princ_len=os.length;
474                 memcpy(ret->krb5_client_princ,os.data,ret->krb5_client_princ_len
475                 OPENSSL_free(os.data);
476                 os.data = NULL;
477                 os.length = 0;
478                 }
479         else
480                 ret->krb5_client_princ_len=0;
481 #endif /* OPENSSL_NO_KRB5 */

483         M_ASN1_D2I_get_IMP_opt(osp,d2i_ASN1_OCTET_STRING,0,V_ASN1_OCTET_STRING);
484         if (os.length > SSL_MAX_KEY_ARG_LENGTH)
485                 ret->key_arg_length=SSL_MAX_KEY_ARG_LENGTH;
486         else
487                 ret->key_arg_length=os.length;
488         memcpy(ret->key_arg,os.data,ret->key_arg_length);
489         if (os.data != NULL) OPENSSL_free(os.data);

491         ai.length=0;
492         M_ASN1_D2I_get_EXP_opt(aip,d2i_ASN1_INTEGER,1);
493         if (ai.data != NULL)
494                 {
495                 ret->time=ASN1_INTEGER_get(aip);
496                 OPENSSL_free(ai.data); ai.data=NULL; ai.length=0;
497                 }
498         else
499                 ret->time=(unsigned long)time(NULL);

501         ai.length=0;
502         M_ASN1_D2I_get_EXP_opt(aip,d2i_ASN1_INTEGER,2);
503         if (ai.data != NULL)
504                 {
505                 ret->timeout=ASN1_INTEGER_get(aip);
506                 OPENSSL_free(ai.data); ai.data=NULL; ai.length=0;
507                 }
508         else
509                 ret->timeout=3;

511         if (ret->peer != NULL)
512                 {
513                 X509_free(ret->peer);
514                 ret->peer=NULL;
515                 }
516         M_ASN1_D2I_get_EXP_opt(ret->peer,d2i_X509,3);

518         os.length=0;
519         os.data=NULL;
520         M_ASN1_D2I_get_EXP_opt(osp,d2i_ASN1_OCTET_STRING,4);

522         if(os.data != NULL)
523             {
```

```
524             if (os.length > SSL_MAX_SID_CTX_LENGTH)
525                 {
526                 c.error=SSL_R_BAD_LENGTH;
527                 c.line=__LINE__;
528                 goto err;
529                 }
530             else
531                 {
532                 ret->sid_ctx_length=os.length;
533                 memcpy(ret->sid_ctx,os.data,os.length);
534                 }
535             OPENSSL_free(os.data); os.data=NULL; os.length=0;
536             }
537         else
538             ret->sid_ctx_length=0;

540         ai.length=0;
541         M_ASN1_D2I_get_EXP_opt(aip,d2i_ASN1_INTEGER,5);
542         if (ai.data != NULL)
543                 {
544                 ret->verify_result=ASN1_INTEGER_get(aip);
545                 OPENSSL_free(ai.data); ai.data=NULL; ai.length=0;
546                 }
547         else
548                 ret->verify_result=X509_V_OK;

550 #ifndef OPENSSL_NO_TLSEXT
551         os.length=0;
552         os.data=NULL;
553         M_ASN1_D2I_get_EXP_opt(osp,d2i_ASN1_OCTET_STRING,6);
554         if (os.data)
555                 {
556                 ret->tlsext_hostname = BUF_strndup((char *)os.data, os.length);
557                 OPENSSL_free(os.data);
558                 os.data = NULL;
559                 os.length = 0;
560                 }
561         else
562                 ret->tlsext_hostname=NULL;
563 #endif /* OPENSSL_NO_TLSEXT */

565 #ifndef OPENSSL_NO_PSK
566         os.length=0;
567         os.data=NULL;
568         M_ASN1_D2I_get_EXP_opt(osp,d2i_ASN1_OCTET_STRING,7);
569         if (os.data)
570                 {
571                 ret->psk_identity_hint = BUF_strndup((char *)os.data, os.length)
572                 OPENSSL_free(os.data);
573                 os.data = NULL;
574                 os.length = 0;
575                 }
576         else
577                 ret->psk_identity_hint=NULL;

579         os.length=0;
580         os.data=NULL;
581         M_ASN1_D2I_get_EXP_opt(osp,d2i_ASN1_OCTET_STRING,8);
582         if (os.data)
583                 {
584                 ret->psk_identity = BUF_strndup((char *)os.data, os.length);
585                 OPENSSL_free(os.data);
586                 os.data = NULL;
587                 os.length = 0;
588                 }
589         else
```



```
590                 ret->psk_identity=NULL;
591 #endif /* OPENSSL_NO_PSK */

593 #ifndef OPENSSL_NO_TLSEXT
594         ai.length=0;
595         M_ASN1_D2I_get_EXP_opt(aip,d2i_ASN1_INTEGER,9);
596         if (ai.data != NULL)
597                 {
598                 ret->tlsext_tick_lifetime_hint=ASN1_INTEGER_get(aip);
599                 OPENSSL_free(ai.data); ai.data=NULL; ai.length=0;
600                 }
601         else if (ret->tlsext_ticklen && ret->session_id_length)
602                 ret->tlsext_tick_lifetime_hint = -1;
603         else
604                 ret->tlsext_tick_lifetime_hint=0;
605         os.length=0;
606         os.data=NULL;
607         M_ASN1_D2I_get_EXP_opt(osp,d2i_ASN1_OCTET_STRING,10);
608         if (os.data)
609                 {
610                 ret->tlsext_tick = os.data;
611                 ret->tlsext_ticklen = os.length;
612                 os.data = NULL;
613                 os.length = 0;
614                 }
615         else
616                 ret->tlsext_tick=NULL;
617 #endif /* OPENSSL_NO_TLSEXT */
618 #ifndef OPENSSL_NO_COMP
619         os.length=0;
620         os.data=NULL;
621         M_ASN1_D2I_get_EXP_opt(osp,d2i_ASN1_OCTET_STRING,11);
622         if (os.data)
623                 {
624                 ret->compress_meth = os.data[0];
625                 OPENSSL_free(os.data);
626                 os.data = NULL;
627                 }
628 #endif

630 #ifndef OPENSSL_NO_SRP
631         os.length=0;
632         os.data=NULL;
633         M_ASN1_D2I_get_EXP_opt(osp,d2i_ASN1_OCTET_STRING,12);
634         if (os.data)
635                 {
636                 ret->srp_username = BUF_strndup((char *)os.data, os.length);
637                 OPENSSL_free(os.data);
638                 os.data = NULL;
639                 os.length = 0;
640                 }
641         else
642                 ret->srp_username=NULL;
643 #endif /* OPENSSL_NO_SRP */

645         M_ASN1_D2I_Finish(a,SSL_SESSION_free,SSL_F_D2I_SSL_SESSION);
646         }
647 #endif /* ! codereview */
```

```
**********************************************************
   22050 Wed Aug 13 19:53:39 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_cert.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/*! \file ssl/ssl_cert.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECC cipher suite support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */

#include <stdio.h>

#include "e_os.h"
#ifndef NO_SYS_TYPES_H
# include <sys/types.h>
#endif

#include "o_dir.h"
#include <openssl/objects.h>
#include <openssl/bio.h>
#include <openssl/pem.h>
```

```
#include <openssl/x509v3.h>
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#include <openssl/bn.h>
#include "ssl_locl.h"

int SSL_get_ex_data_X509_STORE_CTX_idx(void)
        {
        static volatile int ssl_x509_store_ctx_idx= -1;
        int got_write_lock = 0;

        CRYPTO_r_lock(CRYPTO_LOCK_SSL_CTX);

        if (ssl_x509_store_ctx_idx < 0)
                {
                CRYPTO_r_unlock(CRYPTO_LOCK_SSL_CTX);
                CRYPTO_w_lock(CRYPTO_LOCK_SSL_CTX);
                got_write_lock = 1;

                if (ssl_x509_store_ctx_idx < 0)
                        {
                        ssl_x509_store_ctx_idx=X509_STORE_CTX_get_ex_new_index(
                                0,"SSL for verify callback",NULL,NULL,NULL);
                        }
                }

        if (got_write_lock)
                CRYPTO_w_unlock(CRYPTO_LOCK_SSL_CTX);
        else
                CRYPTO_r_unlock(CRYPTO_LOCK_SSL_CTX);

        return ssl_x509_store_ctx_idx;
        }

static void ssl_cert_set_default_md(CERT *cert)
        {
        /* Set digest values to defaults */
#ifndef OPENSSL_NO_DSA
        cert->pkeys[SSL_PKEY_DSA_SIGN].digest = EVP_sha1();
#endif
#ifndef OPENSSL_NO_RSA
        cert->pkeys[SSL_PKEY_RSA_SIGN].digest = EVP_sha1();
        cert->pkeys[SSL_PKEY_RSA_ENC].digest = EVP_sha1();
#endif
#ifndef OPENSSL_NO_ECDSA
        cert->pkeys[SSL_PKEY_ECC].digest = EVP_sha1();
#endif
        }

CERT *ssl_cert_new(void)
        {
        CERT *ret;

        ret=(CERT *)OPENSSL_malloc(sizeof(CERT));
        if (ret == NULL)
                {
                SSLerr(SSL_F_SSL_CERT_NEW,ERR_R_MALLOC_FAILURE);
                return(NULL);
                }
        memset(ret,0,sizeof(CERT));

        ret->key= &(ret->pkeys[SSL_PKEY_RSA_ENC]);
        ret->references=1;
        ssl_cert_set_default_md(ret);
        return(ret);
```



```
        }

CERT *ssl_cert_dup(CERT *cert)
        {
        CERT *ret;
        int i;

        ret = (CERT *)OPENSSL_malloc(sizeof(CERT));
        if (ret == NULL)
                {
                SSLerr(SSL_F_SSL_CERT_DUP, ERR_R_MALLOC_FAILURE);
                return(NULL);
                }

        memset(ret, 0, sizeof(CERT));

        ret->key = &ret->pkeys[cert->key - &cert->pkeys[0]];
        /* or ret->key = ret->pkeys + (cert->key - cert->pkeys),
         * if you find that more readable */

        ret->valid = cert->valid;
        ret->mask_k = cert->mask_k;
        ret->mask_a = cert->mask_a;
        ret->export_mask_k = cert->export_mask_k;
        ret->export_mask_a = cert->export_mask_a;

#ifndef OPENSSL_NO_RSA
        if (cert->rsa_tmp != NULL)
                {
                RSA_up_ref(cert->rsa_tmp);
                ret->rsa_tmp = cert->rsa_tmp;
                }
        ret->rsa_tmp_cb = cert->rsa_tmp_cb;
#endif

#ifndef OPENSSL_NO_DH
        if (cert->dh_tmp != NULL)
                {
                ret->dh_tmp = DHparams_dup(cert->dh_tmp);
                if (ret->dh_tmp == NULL)
                        {
                        SSLerr(SSL_F_SSL_CERT_DUP, ERR_R_DH_LIB);
                        goto err;
                        }
                if (cert->dh_tmp->priv_key)
                        {
                        BIGNUM *b = BN_dup(cert->dh_tmp->priv_key);
                        if (!b)
                                {
                                SSLerr(SSL_F_SSL_CERT_DUP, ERR_R_BN_LIB);
                                goto err;
                                }
                        ret->dh_tmp->priv_key = b;
                        }
                if (cert->dh_tmp->pub_key)
                        {
                        BIGNUM *b = BN_dup(cert->dh_tmp->pub_key);
                        if (!b)
                                {
                                SSLerr(SSL_F_SSL_CERT_DUP, ERR_R_BN_LIB);
                                goto err;
                                }
                        ret->dh_tmp->pub_key = b;
                        }
                }
        ret->dh_tmp_cb = cert->dh_tmp_cb;
```

```
#endif

#ifndef OPENSSL_NO_ECDH
        if (cert->ecdh_tmp)
                {
                ret->ecdh_tmp = EC_KEY_dup(cert->ecdh_tmp);
                if (ret->ecdh_tmp == NULL)
                        {
                        SSLerr(SSL_F_SSL_CERT_DUP, ERR_R_EC_LIB);
                        goto err;
                        }
                }
        ret->ecdh_tmp_cb = cert->ecdh_tmp_cb;
#endif

        for (i = 0; i < SSL_PKEY_NUM; i++)
                {
                if (cert->pkeys[i].x509 != NULL)
                        {
                        ret->pkeys[i].x509 = cert->pkeys[i].x509;
                        CRYPTO_add(&ret->pkeys[i].x509->references, 1,
                                CRYPTO_LOCK_X509);
                        }

                if (cert->pkeys[i].privatekey != NULL)
                        {
                        ret->pkeys[i].privatekey = cert->pkeys[i].privatekey;
                        CRYPTO_add(&ret->pkeys[i].privatekey->references, 1,
                                CRYPTO_LOCK_EVP_PKEY);

                        switch(i)
                                {
                                /* If there was anything special to do for
                                 * certain types of keys, we'd do it here.
                                 * (Nothing at the moment, I think.) */

                        case SSL_PKEY_RSA_ENC:
                        case SSL_PKEY_RSA_SIGN:
                                /* We have an RSA key. */
                                break;

                        case SSL_PKEY_DSA_SIGN:
                                /* We have a DSA key. */
                                break;

                        case SSL_PKEY_DH_RSA:
                        case SSL_PKEY_DH_DSA:
                                /* We have a DH key. */
                                break;

                        case SSL_PKEY_ECC:
                                /* We have an ECC key */
                                break;

                        default:
                                /* Can't happen. */
                                SSLerr(SSL_F_SSL_CERT_DUP, SSL_R_LIBRARY_BUG);
                                }
                        }
                }

        /* ret->extra_certs *should* exist, but currently the own certificate
         * chain is held inside SSL_CTX */

        ret->references=1;
        /* Set digests to defaults. NB: we don't copy existing values as they
```



```
         * will be set during handshake.
         */
        ssl_cert_set_default_md(ret);

        return(ret);

#if !defined(OPENSSL_NO_DH) || !defined(OPENSSL_NO_ECDH)
err:
#endif
#ifndef OPENSSL_NO_RSA
        if (ret->rsa_tmp != NULL)
                RSA_free(ret->rsa_tmp);
#endif
#ifndef OPENSSL_NO_DH
        if (ret->dh_tmp != NULL)
                DH_free(ret->dh_tmp);
#endif
#ifndef OPENSSL_NO_ECDH
        if (ret->ecdh_tmp != NULL)
                EC_KEY_free(ret->ecdh_tmp);
#endif

        for (i = 0; i < SSL_PKEY_NUM; i++)
                {
                if (ret->pkeys[i].x509 != NULL)
                        X509_free(ret->pkeys[i].x509);
                if (ret->pkeys[i].privatekey != NULL)
                        EVP_PKEY_free(ret->pkeys[i].privatekey);
                }

        return NULL;
        }


void ssl_cert_free(CERT *c)
        {
        int i;

        if(c == NULL)
            return;

        i=CRYPTO_add(&c->references,-1,CRYPTO_LOCK_SSL_CERT);
#ifdef REF_PRINT
        REF_PRINT("CERT",c);
#endif
        if (i > 0) return;
#ifdef REF_CHECK
        if (i < 0)
                {
                fprintf(stderr,"ssl_cert_free, bad reference count\n");
                abort(); /* ok */
                }
#endif

#ifndef OPENSSL_NO_RSA
        if (c->rsa_tmp) RSA_free(c->rsa_tmp);
#endif
#ifndef OPENSSL_NO_DH
        if (c->dh_tmp) DH_free(c->dh_tmp);
#endif
#ifndef OPENSSL_NO_ECDH
        if (c->ecdh_tmp) EC_KEY_free(c->ecdh_tmp);
#endif

        for (i=0; i<SSL_PKEY_NUM; i++)
                {
```

```
392                 if (c->pkeys[i].x509 != NULL)
393                         X509_free(c->pkeys[i].x509);
394                 if (c->pkeys[i].privatekey != NULL)
395                         EVP_PKEY_free(c->pkeys[i].privatekey);
396 #if 0
397                 if (c->pkeys[i].publickey != NULL)
398                         EVP_PKEY_free(c->pkeys[i].publickey);
399 #endif
400                 }
401         OPENSSL_free(c);
402         }

404 int ssl_cert_inst(CERT **o)
405         {
406         /* Create a CERT if there isn't already one
407          * (which cannot really happen, as it is initially created in
408          * SSL_CTX_new; but the earlier code usually allows for that one
409          * being non-existant, so we follow that behaviour, as it might
410          * turn out that there actually is a reason for it -- but I'm
411          * not sure that *all* of the existing code could cope with
412          * s->cert being NULL, otherwise we could do without the
413          * initialization in SSL_CTX_new).
414          */

416         if (o == NULL)
417                 {
418                 SSLerr(SSL_F_SSL_CERT_INST, ERR_R_PASSED_NULL_PARAMETER);
419                 return(0);
420                 }
421         if (*o == NULL)
422                 {
423                 if ((*o = ssl_cert_new()) == NULL)
424                         {
425                         SSLerr(SSL_F_SSL_CERT_INST, ERR_R_MALLOC_FAILURE);
426                         return(0);
427                         }
428                 }
429         return(1);
430         }


433 SESS_CERT *ssl_sess_cert_new(void)
434         {
435         SESS_CERT *ret;

437         ret = OPENSSL_malloc(sizeof *ret);
438         if (ret == NULL)
439                 {
440                 SSLerr(SSL_F_SSL_SESS_CERT_NEW, ERR_R_MALLOC_FAILURE);
441                 return NULL;
442                 }

444         memset(ret, 0 ,sizeof *ret);
445         ret->peer_key = &(ret->peer_pkeys[SSL_PKEY_RSA_ENC]);
446         ret->references = 1;

448         return ret;
449         }

451 void ssl_sess_cert_free(SESS_CERT *sc)
452         {
453         int i;

455         if (sc == NULL)
456                 return;
```



```
458         i = CRYPTO_add(&sc->references, -1, CRYPTO_LOCK_SSL_SESS_CERT);
459 #ifdef REF_PRINT
460         REF_PRINT("SESS_CERT", sc);
461 #endif
462         if (i > 0)
463                 return;
464 #ifdef REF_CHECK
465         if (i < 0)
466                 {
467                 fprintf(stderr,"ssl_sess_cert_free, bad reference count\n");
468                 abort(); /* ok */
469                 }
470 #endif

472         /* i == 0 */
473         if (sc->cert_chain != NULL)
474                 sk_X509_pop_free(sc->cert_chain, X509_free);
475         for (i = 0; i < SSL_PKEY_NUM; i++)
476                 {
477                 if (sc->peer_pkeys[i].x509 != NULL)
478                         X509_free(sc->peer_pkeys[i].x509);
479 #if 0 /* We don't have the peer's private key.  These lines are just
480            * here as a reminder that we're still using a not-quite-appropriate
481            * data structure. */
482                 if (sc->peer_pkeys[i].privatekey != NULL)
483                         EVP_PKEY_free(sc->peer_pkeys[i].privatekey);
484 #endif
485                 }

487 #ifndef OPENSSL_NO_RSA
488         if (sc->peer_rsa_tmp != NULL)
489                 RSA_free(sc->peer_rsa_tmp);
490 #endif
491 #ifndef OPENSSL_NO_DH
492         if (sc->peer_dh_tmp != NULL)
493                 DH_free(sc->peer_dh_tmp);
494 #endif
495 #ifndef OPENSSL_NO_ECDH
496         if (sc->peer_ecdh_tmp != NULL)
497                 EC_KEY_free(sc->peer_ecdh_tmp);
498 #endif

500         OPENSSL_free(sc);
501         }

503 int ssl_set_peer_cert_type(SESS_CERT *sc,int type)
504         {
505         sc->peer_cert_type = type;
506         return(1);
507         }

509 int ssl_verify_cert_chain(SSL *s,STACK_OF(X509) *sk)
510         {
511         X509 *x;
512         int i;
513         X509_STORE_CTX ctx;

515         if ((sk == NULL) || (sk_X509_num(sk) == 0))
516                 return(0);

518         x=sk_X509_value(sk,0);
519         if(!X509_STORE_CTX_init(&ctx,s->ctx->cert_store,x,sk))
520                 {
521                 SSLerr(SSL_F_SSL_VERIFY_CERT_CHAIN,ERR_R_X509_LIB);
522                 return(0);
523                 }
```

```
524 #if 0
525         if (SSL_get_verify_depth(s) >= 0)
526                 X509_STORE_CTX_set_depth(&ctx, SSL_get_verify_depth(s));
527 #endif
528         X509_STORE_CTX_set_ex_data(&ctx,SSL_get_ex_data_X509_STORE_CTX_idx(),s);
529
530         /* We need to inherit the verify parameters. These can be determined by
531          * the context: if its a server it will verify SSL client certificates
532          * or vice versa.
533          */
534
535         X509_STORE_CTX_set_default(&ctx,
536                                 s->server ? "ssl_client" : "ssl_server");
537         /* Anything non-default in "param" should overwrite anything in the
538          * ctx.
539          */
540         X509_VERIFY_PARAM_set1(X509_STORE_CTX_get0_param(&ctx), s->param);
541
542         if (s->verify_callback)
543                 X509_STORE_CTX_set_verify_cb(&ctx, s->verify_callback);
544
545         if (s->ctx->app_verify_callback != NULL)
546 #if 1 /* new with OpenSSL 0.9.7 */
547                 i=s->ctx->app_verify_callback(&ctx, s->ctx->app_verify_arg);
548 #else
549                 i=s->ctx->app_verify_callback(&ctx); /* should pass app_verify_a
550 #endif
551         else
552                 {
553 #ifndef OPENSSL_NO_X509_VERIFY
554                 i=X509_verify_cert(&ctx);
555 #else
556                 i=0;
557                 ctx.error=X509_V_ERR_APPLICATION_VERIFICATION;
558                 SSLerr(SSL_F_SSL_VERIFY_CERT_CHAIN,SSL_R_NO_VERIFY_CALLBACK);
559 #endif
560                 }
561
562         s->verify_result=ctx.error;
563         X509_STORE_CTX_cleanup(&ctx);
564
565         return(i);
566         }
567
568 static void set_client_CA_list(STACK_OF(X509_NAME) **ca_list,STACK_OF(X509_NAME)
569         {
570         if (*ca_list != NULL)
571                 sk_X509_NAME_pop_free(*ca_list,X509_NAME_free);
572
573         *ca_list=name_list;
574         }
575
576 STACK_OF(X509_NAME) *SSL_dup_CA_list(STACK_OF(X509_NAME) *sk)
577         {
578         int i;
579         STACK_OF(X509_NAME) *ret;
580         X509_NAME *name;
581
582         ret=sk_X509_NAME_new_null();
583         for (i=0; i<sk_X509_NAME_num(sk); i++)
584                 {
585                 name=X509_NAME_dup(sk_X509_NAME_value(sk,i));
586                 if ((name == NULL) || !sk_X509_NAME_push(ret,name))
587                         {
588                         sk_X509_NAME_pop_free(ret,X509_NAME_free);
589                         return(NULL);
```



```
590                         }
591                 }
592         return(ret);
593         }
594
595 void SSL_set_client_CA_list(SSL *s,STACK_OF(X509_NAME) *name_list)
596         {
597         set_client_CA_list(&(s->client_CA),name_list);
598         }
599
600 void SSL_CTX_set_client_CA_list(SSL_CTX *ctx,STACK_OF(X509_NAME) *name_list)
601         {
602         set_client_CA_list(&(ctx->client_CA),name_list);
603         }
604
605 STACK_OF(X509_NAME) *SSL_CTX_get_client_CA_list(const SSL_CTX *ctx)
606         {
607         return(ctx->client_CA);
608         }
609
610 STACK_OF(X509_NAME) *SSL_get_client_CA_list(const SSL *s)
611         {
612         if (s->type == SSL_ST_CONNECT)
613                 { /* we are in the client */
614                 if (((s->version>>8) == SSL3_VERSION_MAJOR) &&
615                         (s->s3 != NULL))
616                         return(s->s3->tmp.ca_names);
617                 else
618                         return(NULL);
619                 }
620         else
621                 {
622                 if (s->client_CA != NULL)
623                         return(s->client_CA);
624                 else
625                         return(s->ctx->client_CA);
626                 }
627         }
628
629 static int add_client_CA(STACK_OF(X509_NAME) **sk,X509 *x)
630         {
631         X509_NAME *name;
632
633         if (x == NULL) return(0);
634         if ((*sk == NULL) && ((*sk=sk_X509_NAME_new_null()) == NULL))
635                 return(0);
636
637         if ((name=X509_NAME_dup(X509_get_subject_name(x))) == NULL)
638                 return(0);
639
640         if (!sk_X509_NAME_push(*sk,name))
641                 {
642                 X509_NAME_free(name);
643                 return(0);
644                 }
645         return(1);
646         }
647
648 int SSL_add_client_CA(SSL *ssl,X509 *x)
649         {
650         return(add_client_CA(&(ssl->client_CA),x));
651         }
652
653 int SSL_CTX_add_client_CA(SSL_CTX *ctx,X509 *x)
654         {
655         return(add_client_CA(&(ctx->client_CA),x));
```

```
656         }

658 static int xname_cmp(const X509_NAME * const *a, const X509_NAME * const *b)
659         {
660         return(X509_NAME_cmp(*a,*b));
661         }

663 #ifndef OPENSSL_NO_STDIO
664 /*!
665  * Load CA certs from a file into a ::STACK. Note that it is somewhat misnamed;
666  * it doesn't really have anything to do with clients (except that a common use
667  * for a stack of CAs is to send it to the client). Actually, it doesn't have
668  * much to do with CAs, either, since it will load any old cert.
669  * \param file the file containing one or more certs.
670  * \return a ::STACK containing the certs.
671  */
672 STACK_OF(X509_NAME) *SSL_load_client_CA_file(const char *file)
673         {
674         BIO *in;
675         X509 *x=NULL;
676         X509_NAME *xn=NULL;
677         STACK_OF(X509_NAME) *ret = NULL,*sk;

679         sk=sk_X509_NAME_new(xname_cmp);

681         in=BIO_new(BIO_s_file_internal());

683         if ((sk == NULL) || (in == NULL))
684                 {
685                 SSLerr(SSL_F_SSL_LOAD_CLIENT_CA_FILE,ERR_R_MALLOC_FAILURE);
686                 goto err;
687                 }

689         if (!BIO_read_filename(in,file))
690                 goto err;

692         for (;;)
693                 {
694                 if (PEM_read_bio_X509(in,&x,NULL,NULL) == NULL)
695                         break;
696                 if (ret == NULL)
697                         {
698                         ret = sk_X509_NAME_new_null();
699                         if (ret == NULL)
700                                 {
701                                 SSLerr(SSL_F_SSL_LOAD_CLIENT_CA_FILE,ERR_R_MALLO
702                                 goto err;
703                                 }
704                         }
705                 if ((xn=X509_get_subject_name(x)) == NULL) goto err;
706                 /* check for duplicates */
707                 xn=X509_NAME_dup(xn);
708                 if (xn == NULL) goto err;
709                 if (sk_X509_NAME_find(sk,xn) >= 0)
710                         X509_NAME_free(xn);
711                 else
712                         {
713                         sk_X509_NAME_push(sk,xn);
714                         sk_X509_NAME_push(ret,xn);
715                         }
716                 }

718         if (0)
719                 {
720 err:
721                 if (ret != NULL) sk_X509_NAME_pop_free(ret,X509_NAME_free);
```



```
722                 ret=NULL;
723                 }
724         if (sk != NULL) sk_X509_NAME_free(sk);
725         if (in != NULL) BIO_free(in);
726         if (x != NULL) X509_free(x);
727         if (ret != NULL)
728                 ERR_clear_error();
729         return(ret);
730         }
731 #endif

733 /*!
734  * Add a file of certs to a stack.
735  * \param stack the stack to add to.
736  * \param file the file to add from. All certs in this file that are not
737  * already in the stack will be added.
738  * \return 1 for success, 0 for failure. Note that in the case of failure some
739  * certs may have been added to \c stack.
740  */

742 int SSL_add_file_cert_subjects_to_stack(STACK_OF(X509_NAME) *stack,
743                                         const char *file)
744         {
745         BIO *in;
746         X509 *x=NULL;
747         X509_NAME *xn=NULL;
748         int ret=1;
749         int (*oldcmp)(const X509_NAME * const *a, const X509_NAME * const *b);

751         oldcmp=sk_X509_NAME_set_cmp_func(stack,xname_cmp);

753         in=BIO_new(BIO_s_file_internal());

755         if (in == NULL)
756                 {
757                 SSLerr(SSL_F_SSL_ADD_FILE_CERT_SUBJECTS_TO_STACK,ERR_R_MALLOC_FA
758                 goto err;
759                 }

761         if (!BIO_read_filename(in,file))
762                 goto err;

764         for (;;)
765                 {
766                 if (PEM_read_bio_X509(in,&x,NULL,NULL) == NULL)
767                         break;
768                 if ((xn=X509_get_subject_name(x)) == NULL) goto err;
769                 xn=X509_NAME_dup(xn);
770                 if (xn == NULL) goto err;
771                 if (sk_X509_NAME_find(stack,xn) >= 0)
772                         X509_NAME_free(xn);
773                 else
774                         sk_X509_NAME_push(stack,xn);
775                 }

777         ERR_clear_error();

779         if (0)
780                 {
781 err:
782                 ret=0;
783                 }
784         if(in != NULL)
785                 BIO_free(in);
786         if(x != NULL)
787                 X509_free(x);
```

```
789         (void)sk_X509_NAME_set_cmp_func(stack,oldcmp);

791         return ret;
792         }

794 /*!
795  * Add a directory of certs to a stack.
796  * \param stack the stack to append to.
797  * \param dir the directory to append from. All files in this directory will be
798  * examined as potential certs. Any that are acceptable to
799  * SSL_add_dir_cert_subjects_to_stack() that are not already in the stack will b
800  * included.
801  * \return 1 for success, 0 for failure. Note that in the case of failure some
802  * certs may have been added to \c stack.
803  */

805 int SSL_add_dir_cert_subjects_to_stack(STACK_OF(X509_NAME) *stack,
806                                        const char *dir)
807         {
808         OPENSSL_DIR_CTX *d = NULL;
809         const char *filename;
810         int ret = 0;

812         CRYPTO_w_lock(CRYPTO_LOCK_READDIR);

814         /* Note that a side effect is that the CAs will be sorted by name */

816         while((filename = OPENSSL_DIR_read(&d, dir)))
817                 {
818                 char buf[1024];
819                 int r;

821                 if(strlen(dir)+strlen(filename)+2 > sizeof buf)
822                         {
823                         SSLerr(SSL_F_SSL_ADD_DIR_CERT_SUBJECTS_TO_STACK,SSL_R_PA
824                         goto err;
825                         }

827 #ifdef OPENSSL_SYS_VMS
828                 r = BIO_snprintf(buf,sizeof buf,"%s%s",dir,filename);
829 #else
830                 r = BIO_snprintf(buf,sizeof buf,"%s/%s",dir,filename);
831 #endif
832                 if (r <= 0 || r >= (int)sizeof(buf))
833                         goto err;
834                 if(!SSL_add_file_cert_subjects_to_stack(stack,buf))
835                         goto err;
836                 }

838         if (errno)
839                 {
840                 SYSerr(SYS_F_OPENDIR, get_last_sys_error());
841                 ERR_add_error_data(3, "OPENSSL_DIR_read(&ctx, '", dir, "')");
842                 SSLerr(SSL_F_SSL_ADD_DIR_CERT_SUBJECTS_TO_STACK, ERR_R_SYS_LIB);
843                 goto err;
844                 }

846         ret = 1;

848 err:
849         if (d) OPENSSL_DIR_end(&d);
850         CRYPTO_w_unlock(CRYPTO_LOCK_READDIR);
851         return ret;
852         }
853 #endif /* ! codereview */
```

```
**********************************************************
   52426 Wed Aug 13 19:53:40 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_ciph.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_ciph.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECC cipher suite support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
```

```
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include <openssl/objects.h>
#ifndef OPENSSL_NO_COMP
#include <openssl/comp.h>
#endif
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include "ssl_locl.h"

#define SSL_ENC_DES_IDX         0
#define SSL_ENC_3DES_IDX        1
#define SSL_ENC_RC4_IDX         2
#define SSL_ENC_RC2_IDX         3
#define SSL_ENC_IDEA_IDX        4
#define SSL_ENC_NULL_IDX        5
#define SSL_ENC_AES128_IDX      6
#define SSL_ENC_AES256_IDX      7
#define SSL_ENC_CAMELLIA128_IDX 8
#define SSL_ENC_CAMELLIA256_IDX 9
#define SSL_ENC_GOST89_IDX      10
#define SSL_ENC_SEED_IDX        11
#define SSL_ENC_AES128GCM_IDX   12
#define SSL_ENC_AES256GCM_IDX   13
#define SSL_ENC_NUM_IDX         14


static const EVP_CIPHER *ssl_cipher_methods[SSL_ENC_NUM_IDX]={
        NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL
        };

#define SSL_COMP_NULL_IDX       0
#define SSL_COMP_ZLIB_IDX       1
#define SSL_COMP_NUM_IDX        2

static STACK_OF(SSL_COMP) *ssl_comp_methods=NULL;

#define SSL_MD_MD5_IDX  0
#define SSL_MD_SHA1_IDX 1
#define SSL_MD_GOST94_IDX 2
#define SSL_MD_GOST89MAC_IDX 3
#define SSL_MD_SHA256_IDX 4
#define SSL_MD_SHA384_IDX 5
/*Constant SSL_MAX_DIGEST equal to size of digests array should be
 * defined in the
 * ssl_locl.h */
#define SSL_MD_NUM_IDX  SSL_MAX_DIGEST
static const EVP_MD *ssl_digest_methods[SSL_MD_NUM_IDX]={
        NULL,NULL,NULL,NULL,NULL,NULL
        };
/* PKEY_TYPE for GOST89MAC is known in advance, but, because
```



```
 * implementation is engine-provided, we'll fill it only if
 * corresponding EVP_PKEY_METHOD is found
 */
static int  ssl_mac_pkey_id[SSL_MD_NUM_IDX]={
        EVP_PKEY_HMAC,EVP_PKEY_HMAC,EVP_PKEY_HMAC,NID_undef,
        EVP_PKEY_HMAC,EVP_PKEY_HMAC
        };

static int ssl_mac_secret_size[SSL_MD_NUM_IDX]={
        0,0,0,0,0,0
        };

static int ssl_handshake_digest_flag[SSL_MD_NUM_IDX]={
        SSL_HANDSHAKE_MAC_MD5,SSL_HANDSHAKE_MAC_SHA,
        SSL_HANDSHAKE_MAC_GOST94, 0, SSL_HANDSHAKE_MAC_SHA256,
        SSL_HANDSHAKE_MAC_SHA384
        };

#define CIPHER_ADD      1
#define CIPHER_KILL     2
#define CIPHER_DEL      3
#define CIPHER_ORD      4
#define CIPHER_SPECIAL  5

typedef struct cipher_order_st
        {
        const SSL_CIPHER *cipher;
        int active;
        int dead;
        struct cipher_order_st *next,*prev;
        } CIPHER_ORDER;

static const SSL_CIPHER cipher_aliases[]={
        /* "ALL" doesn't include eNULL (must be specifically enabled) */
        {0,SSL_TXT_ALL,0,     0,0,~SSL_eNULL,0,0,0,0,0,0},
        /* "COMPLEMENTOFALL" */
        {0,SSL_TXT_CMPALL,0,  0,0,SSL_eNULL,0,0,0,0,0,0},

        /* "COMPLEMENTOFDEFAULT" (does *not* include ciphersuites not found in A
        {0,SSL_TXT_CMPDEF,0,  SSL_kEDH|SSL_kEECDH,SSL_aNULL,~SSL_eNULL,0,0,0,0,0

        /* key exchange aliases
         * (some of those using only a single bit here combine
         * multiple key exchange algs according to the RFCs,
         * e.g. kEDH combines DHE_DSS and DHE_RSA) */
        {0,SSL_TXT_kRSA,0,    SSL_kRSA,  0,0,0,0,0,0,0,0},

        {0,SSL_TXT_kDHr,0,    SSL_kDHr,  0,0,0,0,0,0,0,0}, /* no such ciphersuit
        {0,SSL_TXT_kDHd,0,    SSL_kDHd,  0,0,0,0,0,0,0,0}, /* no such ciphersuit
        {0,SSL_TXT_kDH,0,     SSL_kDHr|SSL_kDHd,0,0,0,0,0,0,0,0}, /* no such cip
        {0,SSL_TXT_kEDH,0,    SSL_kEDH,  0,0,0,0,0,0,0,0},
        {0,SSL_TXT_DH,0,      SSL_kDHr|SSL_kDHd|SSL_kEDH,0,0,0,0,0,0,0,0},

        {0,SSL_TXT_kKRB5,0,   SSL_kKRB5, 0,0,0,0,0,0,0,0},

        {0,SSL_TXT_kECDHr,0,  SSL_kECDHr,0,0,0,0,0,0,0,0},
        {0,SSL_TXT_kECDHe,0,  SSL_kECDHe,0,0,0,0,0,0,0,0},
        {0,SSL_TXT_kECDH,0,   SSL_kECDHr|SSL_kECDHe,0,0,0,0,0,0,0,0},
        {0,SSL_TXT_kEECDH,0,  SSL_kEECDH,0,0,0,0,0,0,0,0},
        {0,SSL_TXT_ECDH,0,    SSL_kECDHr|SSL_kECDHe|SSL_kEECDH,0,0,0,0,0,0,0,0},

        {0,SSL_TXT_kPSK,0,    SSL_kPSK,  0,0,0,0,0,0,0,0},
        {0,SSL_TXT_kSRP,0,    SSL_kSRP,  0,0,0,0,0,0,0,0},
        {0,SSL_TXT_kGOST,0, SSL_kGOST,0,0,0,0,0,0,0,0},

        /* server authentication aliases */
```

```
260         {0,SSL_TXT_aRSA,0,    0,SSL_aRSA,  0,0,0,0,0,0},
261         {0,SSL_TXT_aDSS,0,    0,SSL_aDSS,  0,0,0,0,0,0},
262         {0,SSL_TXT_DSS,0,     0,SSL_aDSS,   0,0,0,0,0,0},
263         {0,SSL_TXT_aKRB5,0,   0,SSL_aKRB5, 0,0,0,0,0,0},
264         {0,SSL_TXT_aNULL,0,   0,SSL_aNULL, 0,0,0,0,0,0},
265         {0,SSL_TXT_aDH,0,     0,SSL_aDH,   0,0,0,0,0,0}, /* no such ciphersuit
266         {0,SSL_TXT_aECDH,0,   0,SSL_aECDH, 0,0,0,0,0,0},
267         {0,SSL_TXT_aECDSA,0,  0,SSL_aECDSA,0,0,0,0,0,0},
268         {0,SSL_TXT_ECDSA,0,   0,SSL_aECDSA, 0,0,0,0,0,0},
269         {0,SSL_TXT_aPSK,0,    0,SSL_aPSK,  0,0,0,0,0,0},
270         {0,SSL_TXT_aGOST94,0,0,SSL_aGOST94,0,0,0,0,0,0},
271         {0,SSL_TXT_aGOST01,0,0,SSL_aGOST01,0,0,0,0,0,0},
272         {0,SSL_TXT_aGOST,0,0,SSL_aGOST94|SSL_aGOST01,0,0,0,0,0,0},
273         {0,SSL_TXT_aSRP,0,    0,SSL_aSRP,  0,0,0,0,0,0},

275         /* aliases combining key exchange and server authentication */
276         {0,SSL_TXT_EDH,0,     SSL_kEDH,~SSL_aNULL,0,0,0,0,0,0},
277         {0,SSL_TXT_EECDH,0,   SSL_kEECDH,~SSL_aNULL,0,0,0,0,0,0},
278         {0,SSL_TXT_NULL,0,    0,0,SSL_eNULL, 0,0,0,0,0,0},
279         {0,SSL_TXT_KRB5,0,    SSL_kKRB5,SSL_aKRB5,0,0,0,0,0,0},
280         {0,SSL_TXT_RSA,0,     SSL_kRSA,SSL_aRSA,0,0,0,0,0,0},
281         {0,SSL_TXT_ADH,0,     SSL_kEDH,SSL_aNULL,0,0,0,0,0,0},
282         {0,SSL_TXT_AECDH,0,   SSL_kEECDH,SSL_aNULL,0,0,0,0,0,0},
283         {0,SSL_TXT_PSK,0,     SSL_kPSK,SSL_aPSK,0,0,0,0,0,0},
284         {0,SSL_TXT_SRP,0,     SSL_kSRP,0,0,0,0,0,0,0},


287         /* symmetric encryption aliases */
288         {0,SSL_TXT_DES,0,     0,0,SSL_DES,   0,0,0,0,0,0},
289         {0,SSL_TXT_3DES,0,    0,0,SSL_3DES,  0,0,0,0,0,0},
290         {0,SSL_TXT_RC4,0,     0,0,SSL_RC4,   0,0,0,0,0,0},
291         {0,SSL_TXT_RC2,0,     0,0,SSL_RC2,   0,0,0,0,0,0},
292         {0,SSL_TXT_IDEA,0,    0,0,SSL_IDEA,  0,0,0,0,0,0},
293         {0,SSL_TXT_SEED,0,    0,0,SSL_SEED,  0,0,0,0,0,0},
294         {0,SSL_TXT_eNULL,0,   0,0,SSL_eNULL, 0,0,0,0,0,0},
295         {0,SSL_TXT_AES128,0,  0,0,SSL_AES128|SSL_AES128GCM,0,0,0,0,0,0},
296         {0,SSL_TXT_AES256,0,  0,0,SSL_AES256|SSL_AES256GCM,0,0,0,0,0,0},
297         {0,SSL_TXT_AES,0,     0,0,SSL_AES,0,0,0,0,0,0},
298         {0,SSL_TXT_AES_GCM,0, 0,0,SSL_AES128GCM|SSL_AES256GCM,0,0,0,0,0,0},
299         {0,SSL_TXT_CAMELLIA128,0,0,0,SSL_CAMELLIA128,0,0,0,0,0,0},
300         {0,SSL_TXT_CAMELLIA256,0,0,0,SSL_CAMELLIA256,0,0,0,0,0,0},
301         {0,SSL_TXT_CAMELLIA   ,0,0,0,SSL_CAMELLIA128|SSL_CAMELLIA256,0,0,0,0,0,0

303         /* MAC aliases */
304         {0,SSL_TXT_MD5,0,     0,0,0,SSL_MD5,   0,0,0,0,0},
305         {0,SSL_TXT_SHA1,0,    0,0,0,SSL_SHA1,  0,0,0,0,0},
306         {0,SSL_TXT_SHA,0,     0,0,0,SSL_SHA1,  0,0,0,0,0},
307         {0,SSL_TXT_GOST94,0,     0,0,0,SSL_GOST94,  0,0,0,0,0},
308         {0,SSL_TXT_GOST89MAC,0,     0,0,0,SSL_GOST89MAC,  0,0,0,0,0},
309         {0,SSL_TXT_SHA256,0,    0,0,0,SSL_SHA256,  0,0,0,0,0},
310         {0,SSL_TXT_SHA384,0,    0,0,0,SSL_SHA384,  0,0,0,0,0},

312         /* protocol version aliases */
313         {0,SSL_TXT_SSLV2,0,   0,0,0,0,SSL_SSLV2, 0,0,0,0},
314         {0,SSL_TXT_SSLV3,0,   0,0,0,0,SSL_SSLV3, 0,0,0,0},
315         {0,SSL_TXT_TLSV1,0,   0,0,0,0,SSL_TLSV1, 0,0,0,0},
316         {0,SSL_TXT_TLSV1_2,0, 0,0,0,0,SSL_TLSV1_2, 0,0,0,0},

318         /* export flag */
319         {0,SSL_TXT_EXP,0,     0,0,0,0,0,SSL_EXPORT,0,0,0},
320         {0,SSL_TXT_EXPORT,0,  0,0,0,0,0,SSL_EXPORT,0,0,0},

322         /* strength classes */
323         {0,SSL_TXT_EXP40,0,   0,0,0,0,0,SSL_EXP40, 0,0,0},
324         {0,SSL_TXT_EXP56,0,   0,0,0,0,0,SSL_EXP56, 0,0,0},
325         {0,SSL_TXT_LOW,0,     0,0,0,0,0,SSL_LOW,   0,0,0},
```



```
326         {0,SSL_TXT_MEDIUM,0,  0,0,0,0,0,SSL_MEDIUM,0,0,0},
327         {0,SSL_TXT_HIGH,0,    0,0,0,0,0,SSL_HIGH,  0,0,0},
328         /* FIPS 140-2 approved ciphersuite */
329         {0,SSL_TXT_FIPS,0,    0,0,~SSL_eNULL,0,0,SSL_FIPS,  0,0,0},
330         };
331 /* Search for public key algorithm with given name and
332  * return its pkey_id if it is available. Otherwise return 0
333  */
334 #ifdef OPENSSL_NO_ENGINE

336 static int get_optional_pkey_id(const char *pkey_name)
337         {
338         const EVP_PKEY_ASN1_METHOD *ameth;
339         int pkey_id=0;
340         ameth = EVP_PKEY_asn1_find_str(NULL,pkey_name,-1);
341         if (ameth)
342                 {
343                 EVP_PKEY_asn1_get0_info(&pkey_id, NULL,NULL,NULL,NULL,ameth);
344                 }
345         return pkey_id;
346         }

348 #else

350 static int get_optional_pkey_id(const char *pkey_name)
351         {
352         const EVP_PKEY_ASN1_METHOD *ameth;
353         ENGINE *tmpeng = NULL;
354         int pkey_id=0;
355         ameth = EVP_PKEY_asn1_find_str(&tmpeng,pkey_name,-1);
356         if (ameth)
357                 {
358                 EVP_PKEY_asn1_get0_info(&pkey_id, NULL,NULL,NULL,NULL,ameth);
359                 }
360         if (tmpeng) ENGINE_finish(tmpeng);
361         return pkey_id;
362         }

364 #endif

366 void ssl_load_ciphers(void)
367         {
368         ssl_cipher_methods[SSL_ENC_DES_IDX]=
369                 EVP_get_cipherbyname(SN_des_cbc);
370         ssl_cipher_methods[SSL_ENC_3DES_IDX]=
371                 EVP_get_cipherbyname(SN_des_ede3_cbc);
372         ssl_cipher_methods[SSL_ENC_RC4_IDX]=
373                 EVP_get_cipherbyname(SN_rc4);
374         ssl_cipher_methods[SSL_ENC_RC2_IDX]=
375                 EVP_get_cipherbyname(SN_rc2_cbc);
376 #ifndef OPENSSL_NO_IDEA
377         ssl_cipher_methods[SSL_ENC_IDEA_IDX]=
378                 EVP_get_cipherbyname(SN_idea_cbc);
379 #else
380         ssl_cipher_methods[SSL_ENC_IDEA_IDX]= NULL;
381 #endif
382         ssl_cipher_methods[SSL_ENC_AES128_IDX]=
383           EVP_get_cipherbyname(SN_aes_128_cbc);
384         ssl_cipher_methods[SSL_ENC_AES256_IDX]=
385           EVP_get_cipherbyname(SN_aes_256_cbc);
386         ssl_cipher_methods[SSL_ENC_CAMELLIA128_IDX]=
387           EVP_get_cipherbyname(SN_camellia_128_cbc);
388         ssl_cipher_methods[SSL_ENC_CAMELLIA256_IDX]=
389           EVP_get_cipherbyname(SN_camellia_256_cbc);
390         ssl_cipher_methods[SSL_ENC_GOST89_IDX]=
391           EVP_get_cipherbyname(SN_gost89_cnt);
```

```
392         ssl_cipher_methods[SSL_ENC_SEED_IDX]=
393           EVP_get_cipherbyname(SN_seed_cbc);

395         ssl_cipher_methods[SSL_ENC_AES128GCM_IDX]=
396           EVP_get_cipherbyname(SN_aes_128_gcm);
397         ssl_cipher_methods[SSL_ENC_AES256GCM_IDX]=
398           EVP_get_cipherbyname(SN_aes_256_gcm);

400         ssl_digest_methods[SSL_MD_MD5_IDX]=
401                 EVP_get_digestbyname(SN_md5);
402         ssl_mac_secret_size[SSL_MD_MD5_IDX]=
403                 EVP_MD_size(ssl_digest_methods[SSL_MD_MD5_IDX]);
404         OPENSSL_assert(ssl_mac_secret_size[SSL_MD_MD5_IDX] >= 0);
405         ssl_digest_methods[SSL_MD_SHA1_IDX]=
406                 EVP_get_digestbyname(SN_sha1);
407         ssl_mac_secret_size[SSL_MD_SHA1_IDX]=
408                 EVP_MD_size(ssl_digest_methods[SSL_MD_SHA1_IDX]);
409         OPENSSL_assert(ssl_mac_secret_size[SSL_MD_SHA1_IDX] >= 0);
410         ssl_digest_methods[SSL_MD_GOST94_IDX]=
411                 EVP_get_digestbyname(SN_id_GostR3411_94);
412         if (ssl_digest_methods[SSL_MD_GOST94_IDX])
413                 {
414                 ssl_mac_secret_size[SSL_MD_GOST94_IDX]=
415                         EVP_MD_size(ssl_digest_methods[SSL_MD_GOST94_IDX]);
416                 OPENSSL_assert(ssl_mac_secret_size[SSL_MD_GOST94_IDX] >= 0);
417                 }
418         ssl_digest_methods[SSL_MD_GOST89MAC_IDX]=
419                 EVP_get_digestbyname(SN_id_Gost28147_89_MAC);
420                 ssl_mac_pkey_id[SSL_MD_GOST89MAC_IDX] = get_optional_pkey_id("go
421                 if (ssl_mac_pkey_id[SSL_MD_GOST89MAC_IDX]) {
422                         ssl_mac_secret_size[SSL_MD_GOST89MAC_IDX]=32;
423                 }

425         ssl_digest_methods[SSL_MD_SHA256_IDX]=
426                 EVP_get_digestbyname(SN_sha256);
427         ssl_mac_secret_size[SSL_MD_SHA256_IDX]=
428                 EVP_MD_size(ssl_digest_methods[SSL_MD_SHA256_IDX]);
429         ssl_digest_methods[SSL_MD_SHA384_IDX]=
430                 EVP_get_digestbyname(SN_sha384);
431         ssl_mac_secret_size[SSL_MD_SHA384_IDX]=
432                 EVP_MD_size(ssl_digest_methods[SSL_MD_SHA384_IDX]);
433         }
434 #ifndef OPENSSL_NO_COMP

436 static int sk_comp_cmp(const SSL_COMP * const *a,
437                         const SSL_COMP * const *b)
438         {
439         return((*a)->id-(*b)->id);
440         }

442 static void load_builtin_compressions(void)
443         {
444         int got_write_lock = 0;

446         CRYPTO_r_lock(CRYPTO_LOCK_SSL);
447         if (ssl_comp_methods == NULL)
448                 {
449                 CRYPTO_r_unlock(CRYPTO_LOCK_SSL);
450                 CRYPTO_w_lock(CRYPTO_LOCK_SSL);
451                 got_write_lock = 1;

453                 if (ssl_comp_methods == NULL)
454                         {
455                         SSL_COMP *comp = NULL;

457                         MemCheck_off();
```



```
458                         ssl_comp_methods=sk_SSL_COMP_new(sk_comp_cmp);
459                         if (ssl_comp_methods != NULL)
460                                 {
461                                 comp=(SSL_COMP *)OPENSSL_malloc(sizeof(SSL_COMP)
462                                 if (comp != NULL)
463                                         {
464                                         comp->method=COMP_zlib();
465                                         if (comp->method
466                                                 && comp->method->type == NID_und
467                                                 OPENSSL_free(comp);
468                                         else
469                                                 {
470                                                 comp->id=SSL_COMP_ZLIB_IDX;
471                                                 comp->name=comp->method->name;
472                                                 sk_SSL_COMP_push(ssl_comp_method
473                                                 }
474                                         }
475                                         sk_SSL_COMP_sort(ssl_comp_methods);
476                                 }
477                         MemCheck_on();
478                         }
479                 }

481         if (got_write_lock)
482                 CRYPTO_w_unlock(CRYPTO_LOCK_SSL);
483         else
484                 CRYPTO_r_unlock(CRYPTO_LOCK_SSL);
485         }
486 #endif

488 int ssl_cipher_get_evp(const SSL_SESSION *s, const EVP_CIPHER **enc,
489                 const EVP_MD **md, int *mac_pkey_type, int *mac_secret_size,SSL_COM
490         {
491         int i;
492         const SSL_CIPHER *c;

494         c=s->cipher;
495         if (c == NULL) return(0);
496         if (comp != NULL)
497                 {
498                 SSL_COMP ctmp;
499 #ifndef OPENSSL_NO_COMP
500                 load_builtin_compressions();
501 #endif

503                 *comp=NULL;
504                 ctmp.id=s->compress_meth;
505                 if (ssl_comp_methods != NULL)
506                         {
507                         i=sk_SSL_COMP_find(ssl_comp_methods,&ctmp);
508                         if (i >= 0)
509                                 *comp=sk_SSL_COMP_value(ssl_comp_methods,i);
510                         else
511                                 *comp=NULL;
512                         }
513                 }

515         if ((enc == NULL) || (md == NULL)) return(0);

517         switch (c->algorithm_enc)
518                 {
519         case SSL_DES:
520                 i=SSL_ENC_DES_IDX;
521                 break;
522         case SSL_3DES:
523                 i=SSL_ENC_3DES_IDX;
```

```
524                 break;
525         case SSL_RC4:
526                 i=SSL_ENC_RC4_IDX;
527                 break;
528         case SSL_RC2:
529                 i=SSL_ENC_RC2_IDX;
530                 break;
531         case SSL_IDEA:
532                 i=SSL_ENC_IDEA_IDX;
533                 break;
534         case SSL_eNULL:
535                 i=SSL_ENC_NULL_IDX;
536                 break;
537         case SSL_AES128:
538                 i=SSL_ENC_AES128_IDX;
539                 break;
540         case SSL_AES256:
541                 i=SSL_ENC_AES256_IDX;
542                 break;
543         case SSL_CAMELLIA128:
544                 i=SSL_ENC_CAMELLIA128_IDX;
545                 break;
546         case SSL_CAMELLIA256:
547                 i=SSL_ENC_CAMELLIA256_IDX;
548                 break;
549         case SSL_eGOST2814789CNT:
550                 i=SSL_ENC_GOST89_IDX;
551                 break;
552         case SSL_SEED:
553                 i=SSL_ENC_SEED_IDX;
554                 break;
555         case SSL_AES128GCM:
556                 i=SSL_ENC_AES128GCM_IDX;
557                 break;
558         case SSL_AES256GCM:
559                 i=SSL_ENC_AES256GCM_IDX;
560                 break;
561         default:
562                 i= -1;
563                 break;
564                 }

566         if ((i < 0) || (i >= SSL_ENC_NUM_IDX))
567                 *enc=NULL;
568         else
569                 {
570                 if (i == SSL_ENC_NULL_IDX)
571                         *enc=EVP_enc_null();
572                 else
573                         *enc=ssl_cipher_methods[i];
574                 }

576         switch (c->algorithm_mac)
577                 {
578         case SSL_MD5:
579                 i=SSL_MD_MD5_IDX;
580                 break;
581         case SSL_SHA1:
582                 i=SSL_MD_SHA1_IDX;
583                 break;
584         case SSL_SHA256:
585                 i=SSL_MD_SHA256_IDX;
586                 break;
587         case SSL_SHA384:
588                 i=SSL_MD_SHA384_IDX;
589                 break;
```



```
590         case SSL_GOST94:
591                 i = SSL_MD_GOST94_IDX;
592                 break;
593         case SSL_GOST89MAC:
594                 i = SSL_MD_GOST89MAC_IDX;
595                 break;
596         default:
597                 i= -1;
598                 break;
599                 }
600         if ((i < 0) || (i >= SSL_MD_NUM_IDX))
601         {
602                 *md=NULL;
603                 if (mac_pkey_type!=NULL) *mac_pkey_type = NID_undef;
604                 if (mac_secret_size!=NULL) *mac_secret_size = 0;
605                 if (c->algorithm_mac == SSL_AEAD)
606                         mac_pkey_type = NULL;
607         }
608         else
609         {
610                 *md=ssl_digest_methods[i];
611                 if (mac_pkey_type!=NULL) *mac_pkey_type = ssl_mac_pkey_id[i];
612                 if (mac_secret_size!=NULL) *mac_secret_size = ssl_mac_secret_siz
613         }

615         if ((*enc != NULL) &&
616             (*md != NULL || (EVP_CIPHER_flags(*enc)&EVP_CIPH_FLAG_AEAD_CIPHER))
617             (!mac_pkey_type||*mac_pkey_type != NID_undef))
618                 {
619                 const EVP_CIPHER *evp;

621                 if (s->ssl_version>>8 != TLS1_VERSION_MAJOR ||
622                     s->ssl_version < TLS1_VERSION)
623                         return 1;

625 #ifdef OPENSSL_FIPS
626                 if (FIPS_mode())
627                         return 1;
628 #endif

630                 if      (c->algorithm_enc == SSL_RC4 &&
631                          c->algorithm_mac == SSL_MD5 &&
632                          (evp=EVP_get_cipherbyname("RC4-HMAC-MD5")))
633                         *enc = evp, *md = NULL;
634                 else if (c->algorithm_enc == SSL_AES128 &&
635                          c->algorithm_mac == SSL_SHA1 &&
636                          (evp=EVP_get_cipherbyname("AES-128-CBC-HMAC-SHA1")))
637                         *enc = evp, *md = NULL;
638                 else if (c->algorithm_enc == SSL_AES256 &&
639                          c->algorithm_mac == SSL_SHA1 &&
640                          (evp=EVP_get_cipherbyname("AES-256-CBC-HMAC-SHA1")))
641                         *enc = evp, *md = NULL;
642                 return(1);
643                 }
644         else
645                 return(0);
646         }

648 int ssl_get_handshake_digest(int idx, long *mask, const EVP_MD **md)
649 {
650         if (idx <0||idx>=SSL_MD_NUM_IDX)
651                 {
652                 return 0;
653                 }
654         *mask = ssl_handshake_digest_flag[idx];
655         if (*mask)
```

```
656                 *md = ssl_digest_methods[idx];
657         else
658                 *md = NULL;
659         return 1;
660 }

662 #define ITEM_SEP(a) \
663         (((a) == ':') || ((a) == ' ') || ((a) == ';') || ((a) == ','))

665 static void ll_append_tail(CIPHER_ORDER **head, CIPHER_ORDER *curr,
666              CIPHER_ORDER **tail)
667         {
668         if (curr == *tail) return;
669         if (curr == *head)
670                 *head=curr->next;
671         if (curr->prev != NULL)
672                 curr->prev->next=curr->next;
673         if (curr->next != NULL)
674                 curr->next->prev=curr->prev;
675         (*tail)->next=curr;
676         curr->prev= *tail;
677         curr->next=NULL;
678         *tail=curr;
679         }

681 static void ll_append_head(CIPHER_ORDER **head, CIPHER_ORDER *curr,
682              CIPHER_ORDER **tail)
683         {
684         if (curr == *head) return;
685         if (curr == *tail)
686                 *tail=curr->prev;
687         if (curr->next != NULL)
688                 curr->next->prev=curr->prev;
689         if (curr->prev != NULL)
690                 curr->prev->next=curr->next;
691         (*head)->prev=curr;
692         curr->next= *head;
693         curr->prev=NULL;
694         *head=curr;
695         }

697 static void ssl_cipher_get_disabled(unsigned long *mkey, unsigned long *auth, un
698         {
699         *mkey = 0;
700         *auth = 0;
701         *enc = 0;
702         *mac = 0;
703         *ssl = 0;

705 #ifdef OPENSSL_NO_RSA
706         *mkey |= SSL_kRSA;
707         *auth |= SSL_aRSA;
708 #endif
709 #ifdef OPENSSL_NO_DSA
710         *auth |= SSL_aDSS;
711 #endif
712         *mkey |= SSL_kDHr|SSL_kDHd; /* no such ciphersuites supported! */
713         *auth |= SSL_aDH;
714 #ifdef OPENSSL_NO_DH
715         *mkey |= SSL_kDHr|SSL_kDHd|SSL_kEDH;
716         *auth |= SSL_aDH;
717 #endif
718 #ifdef OPENSSL_NO_KRB5
719         *mkey |= SSL_kKRB5;
720         *auth |= SSL_aKRB5;
721 #endif
```



```
722 #ifdef OPENSSL_NO_ECDSA
723         *auth |= SSL_aECDSA;
724 #endif
725 #ifdef OPENSSL_NO_ECDH
726         *mkey |= SSL_kECDHe|SSL_kECDHr;
727         *auth |= SSL_aECDH;
728 #endif
729 #ifdef OPENSSL_NO_PSK
730         *mkey |= SSL_kPSK;
731         *auth |= SSL_aPSK;
732 #endif
733 #ifdef OPENSSL_NO_SRP
734         *mkey |= SSL_kSRP;
735 #endif
736         /* Check for presence of GOST 34.10 algorithms, and if they
737          * do not present, disable  appropriate auth and key exchange */
738         if (!get_optional_pkey_id("gost94")) {
739                 *auth |= SSL_aGOST94;
740         }
741         if (!get_optional_pkey_id("gost2001")) {
742                 *auth |= SSL_aGOST01;
743         }
744         /* Disable GOST key exchange if no GOST signature algs are available * *
745         if ((*auth & (SSL_aGOST94|SSL_aGOST01)) == (SSL_aGOST94|SSL_aGOST01)) {
746                 *mkey |= SSL_kGOST;
747         }
748 #ifdef SSL_FORBID_ENULL
749         *enc |= SSL_eNULL;
750 #endif



754         *enc |= (ssl_cipher_methods[SSL_ENC_DES_IDX ] == NULL) ? SSL_DES :0;
755         *enc |= (ssl_cipher_methods[SSL_ENC_3DES_IDX] == NULL) ? SSL_3DES:0;
756         *enc |= (ssl_cipher_methods[SSL_ENC_RC4_IDX ] == NULL) ? SSL_RC4 :0;
757         *enc |= (ssl_cipher_methods[SSL_ENC_RC2_IDX ] == NULL) ? SSL_RC2 :0;
758         *enc |= (ssl_cipher_methods[SSL_ENC_IDEA_IDX] == NULL) ? SSL_IDEA:0;
759         *enc |= (ssl_cipher_methods[SSL_ENC_AES128_IDX] == NULL) ? SSL_AES128:0;
760         *enc |= (ssl_cipher_methods[SSL_ENC_AES256_IDX] == NULL) ? SSL_AES256:0;
761         *enc |= (ssl_cipher_methods[SSL_ENC_AES128GCM_IDX] == NULL) ? SSL_AES128
762         *enc |= (ssl_cipher_methods[SSL_ENC_AES256GCM_IDX] == NULL) ? SSL_AES256
763         *enc |= (ssl_cipher_methods[SSL_ENC_CAMELLIA128_IDX] == NULL) ? SSL_CAME
764         *enc |= (ssl_cipher_methods[SSL_ENC_CAMELLIA256_IDX] == NULL) ? SSL_CAME
765         *enc |= (ssl_cipher_methods[SSL_ENC_GOST89_IDX] == NULL) ? SSL_eGOST2814
766         *enc |= (ssl_cipher_methods[SSL_ENC_SEED_IDX] == NULL) ? SSL_SEED:0;

768         *mac |= (ssl_digest_methods[SSL_MD_MD5_IDX ] == NULL) ? SSL_MD5 :0;
769         *mac |= (ssl_digest_methods[SSL_MD_SHA1_IDX] == NULL) ? SSL_SHA1:0;
770         *mac |= (ssl_digest_methods[SSL_MD_SHA256_IDX] == NULL) ? SSL_SHA256:0;
771         *mac |= (ssl_digest_methods[SSL_MD_SHA384_IDX] == NULL) ? SSL_SHA384:0;
772         *mac |= (ssl_digest_methods[SSL_MD_GOST94_IDX] == NULL) ? SSL_GOST94:0;
773         *mac |= (ssl_digest_methods[SSL_MD_GOST89MAC_IDX] == NULL || ssl_mac_pke

775         }

777 static void ssl_cipher_collect_ciphers(const SSL_METHOD *ssl_method,
778                 int num_of_ciphers,
779                 unsigned long disabled_mkey, unsigned long disabled_auth,
780                 unsigned long disabled_enc, unsigned long disabled_mac,
781                 unsigned long disabled_ssl,
782                 CIPHER_ORDER *co_list,
783                 CIPHER_ORDER **head_p, CIPHER_ORDER **tail_p)
784         {
785         int i, co_list_num;
786         const SSL_CIPHER *c;
```

```
788         /*
789          * We have num_of_ciphers descriptions compiled in, depending on the
790          * method selected (SSLv2 and/or SSLv3, TLSv1 etc).
791          * These will later be sorted in a linked list with at most num
792          * entries.
793          */

795         /* Get the initial list of ciphers */
796         co_list_num = 0;        /* actual count of ciphers */
797         for (i = 0; i < num_of_ciphers; i++)
798                 {
799                 c = ssl_method->get_cipher(i);
800                 /* drop those that use any of that is not available */
801                 if ((c != NULL) && c->valid &&
802 #ifdef OPENSSL_FIPS
803                     (!FIPS_mode() || (c->algo_strength & SSL_FIPS)) &&
804 #endif
805                     !(c->algorithm_mkey & disabled_mkey) &&
806                     !(c->algorithm_auth & disabled_auth) &&
807                     !(c->algorithm_enc & disabled_enc) &&
808                     !(c->algorithm_mac & disabled_mac) &&
809                     !(c->algorithm_ssl & disabled_ssl))
810                         {
811                         co_list[co_list_num].cipher = c;
812                         co_list[co_list_num].next = NULL;
813                         co_list[co_list_num].prev = NULL;
814                         co_list[co_list_num].active = 0;
815                         co_list_num++;
816 #ifdef KSSL_DEBUG
817                         printf("\t%d: %s %lx %lx %lx\n",i,c->name,c->id,c->algor
818 #endif  /* KSSL_DEBUG */
819                         /*
820                         if (!sk_push(ca_list,(char *)c)) goto err;
821                         */
822                         }
823                 }

825         /*
826          * Prepare linked list from list entries
827          */
828         if (co_list_num > 0)
829                 {
830                 co_list[0].prev = NULL;

832                 if (co_list_num > 1)
833                         {
834                         co_list[0].next = &co_list[1];

836                         for (i = 1; i < co_list_num - 1; i++)
837                                 {
838                                 co_list[i].prev = &co_list[i - 1];
839                                 co_list[i].next = &co_list[i + 1];
840                                 }

842                         co_list[co_list_num - 1].prev = &co_list[co_list_num - 2
843                         }

845                 co_list[co_list_num - 1].next = NULL;

847                 *head_p = &co_list[0];
848                 *tail_p = &co_list[co_list_num - 1];
849                 }
850         }

852 static void ssl_cipher_collect_aliases(const SSL_CIPHER **ca_list,
853                         int num_of_group_aliases,
```



```
854                         unsigned long disabled_mkey, unsigned long disabled_auth
855                         unsigned long disabled_enc, unsigned long disabled_mac,
856                         unsigned long disabled_ssl,
857                         CIPHER_ORDER *head)
858         {
859         CIPHER_ORDER *ciph_curr;
860         const SSL_CIPHER **ca_curr;
861         int i;
862         unsigned long mask_mkey = ~disabled_mkey;
863         unsigned long mask_auth = ~disabled_auth;
864         unsigned long mask_enc = ~disabled_enc;
865         unsigned long mask_mac = ~disabled_mac;
866         unsigned long mask_ssl = ~disabled_ssl;

868         /*
869          * First, add the real ciphers as already collected
870          */
871         ciph_curr = head;
872         ca_curr = ca_list;
873         while (ciph_curr != NULL)
874                 {
875                 *ca_curr = ciph_curr->cipher;
876                 ca_curr++;
877                 ciph_curr = ciph_curr->next;
878                 }

880         /*
881          * Now we add the available ones from the cipher_aliases[] table.
882          * They represent either one or more algorithms, some of which
883          * in any affected category must be supported (set in enabled_mask),
884          * or represent a cipher strength value (will be added in any case becau
885          */
886         for (i = 0; i < num_of_group_aliases; i++)
887                 {
888                 unsigned long algorithm_mkey = cipher_aliases[i].algorithm_mkey;
889                 unsigned long algorithm_auth = cipher_aliases[i].algorithm_auth;
890                 unsigned long algorithm_enc = cipher_aliases[i].algorithm_enc;
891                 unsigned long algorithm_mac = cipher_aliases[i].algorithm_mac;
892                 unsigned long algorithm_ssl = cipher_aliases[i].algorithm_ssl;

894                 if (algorithm_mkey)
895                         if ((algorithm_mkey & mask_mkey) == 0)
896                                 continue;

898                 if (algorithm_auth)
899                         if ((algorithm_auth & mask_auth) == 0)
900                                 continue;

902                 if (algorithm_enc)
903                         if ((algorithm_enc & mask_enc) == 0)
904                                 continue;

906                 if (algorithm_mac)
907                         if ((algorithm_mac & mask_mac) == 0)
908                                 continue;

910                 if (algorithm_ssl)
911                         if ((algorithm_ssl & mask_ssl) == 0)
912                                 continue;

914                 *ca_curr = (SSL_CIPHER *)(cipher_aliases + i);
915                 ca_curr++;
916                 }

918         *ca_curr = NULL;        /* end of list */
919         }
```

```
static void ssl_cipher_apply_rule(unsigned long cipher_id,
                unsigned long alg_mkey, unsigned long alg_auth,
                unsigned long alg_enc, unsigned long alg_mac,
                unsigned long alg_ssl,
                unsigned long algo_strength,
                int rule, int strength_bits,
                CIPHER_ORDER **head_p, CIPHER_ORDER **tail_p)
        {
        CIPHER_ORDER *head, *tail, *curr, *next, *last;
        const SSL_CIPHER *cp;
        int reverse = 0;

#ifdef CIPHER_DEBUG
        printf("Applying rule %d with %08lx/%08lx/%08lx/%08lx/%08lx %08lx (%d)\n
                rule, alg_mkey, alg_auth, alg_enc, alg_mac, alg_ssl, algo_streng
#endif

        if (rule == CIPHER_DEL)
                reverse = 1; /* needed to maintain sorting between currently del

        head = *head_p;
        tail = *tail_p;

        if (reverse)
                {
                next = tail;
                last = head;
                }
        else
                {
                next = head;
                last = tail;
                }

        curr = NULL;
        for (;;)
                {
                if (curr == last) break;

                curr = next;

                if (curr == NULL) break;

                next = reverse ? curr->prev : curr->next;

                cp = curr->cipher;

                /*
                 * Selection criteria is either the value of strength_bits
                 * or the algorithms used.
                 */
                if (strength_bits >= 0)
                        {
                        if (strength_bits != cp->strength_bits)
                                continue;
                        }
                else
                        {
#ifdef CIPHER_DEBUG
                        printf("\nName: %s:\nAlgo = %08lx/%08lx/%08lx/%08lx/%08l
#endif

                        if (alg_mkey && !(alg_mkey & cp->algorithm_mkey))
                                continue;
                        if (alg_auth && !(alg_auth & cp->algorithm_auth))
```



```
                                continue;
                        if (alg_enc && !(alg_enc & cp->algorithm_enc))
                                continue;
                        if (alg_mac && !(alg_mac & cp->algorithm_mac))
                                continue;
                        if (alg_ssl && !(alg_ssl & cp->algorithm_ssl))
                                continue;
                        if ((algo_strength & SSL_EXP_MASK) && !(algo_strength &
                                continue;
                        if ((algo_strength & SSL_STRONG_MASK) && !(algo_strength
                                continue;
                        }

#ifdef CIPHER_DEBUG
                printf("Action = %d\n", rule);
#endif

                /* add the cipher if it has not been added yet. */
                if (rule == CIPHER_ADD)
                        {
                        /* reverse == 0 */
                        if (!curr->active)
                                {
                                ll_append_tail(&head, curr, &tail);
                                curr->active = 1;
                                }
                        }
                /* Move the added cipher to this location */
                else if (rule == CIPHER_ORD)
                        {
                        /* reverse == 0 */
                        if (curr->active)
                                {
                                ll_append_tail(&head, curr, &tail);
                                }
                        }
                else if (rule == CIPHER_DEL)
                        {
                        /* reverse == 1 */
                        if (curr->active)
                                {
                                /* most recently deleted ciphersuites get best p
                                 * for any future CIPHER_ADD (note that the CIPH
                                 * works in reverse to maintain the order) */
                                ll_append_head(&head, curr, &tail);
                                curr->active = 0;
                                }
                        }
                else if (rule == CIPHER_KILL)
                        {
                        /* reverse == 0 */
                        if (head == curr)
                                head = curr->next;
                        else
                                curr->prev->next = curr->next;
                        if (tail == curr)
                                tail = curr->prev;
                        curr->active = 0;
                        if (curr->next != NULL)
                                curr->next->prev = curr->prev;
                        if (curr->prev != NULL)
                                curr->prev->next = curr->next;
                        curr->next = NULL;
                        curr->prev = NULL;
                        }
                }
```

```
1053         *head_p = head;
1054         *tail_p = tail;
1055         }

1057 static int ssl_cipher_strength_sort(CIPHER_ORDER **head_p,
1058                                     CIPHER_ORDER **tail_p)
1059         {
1060         int max_strength_bits, i, *number_uses;
1061         CIPHER_ORDER *curr;

1063         /*
1064          * This routine sorts the ciphers with descending strength. The sorting
1065          * must keep the pre-sorted sequence, so we apply the normal sorting
1066          * routine as '+' movement to the end of the list.
1067          */
1068         max_strength_bits = 0;
1069         curr = *head_p;
1070         while (curr != NULL)
1071                 {
1072                 if (curr->active &&
1073                     (curr->cipher->strength_bits > max_strength_bits))
1074                     max_strength_bits = curr->cipher->strength_bits;
1075                 curr = curr->next;
1076                 }

1078         number_uses = OPENSSL_malloc((max_strength_bits + 1) * sizeof(int));
1079         if (!number_uses)
1080                 {
1081                 SSLerr(SSL_F_SSL_CIPHER_STRENGTH_SORT,ERR_R_MALLOC_FAILURE);
1082                 return(0);
1083                 }
1084         memset(number_uses, 0, (max_strength_bits + 1) * sizeof(int));

1086         /*
1087          * Now find the strength_bits values actually used
1088          */
1089         curr = *head_p;
1090         while (curr != NULL)
1091                 {
1092                 if (curr->active)
1093                         number_uses[curr->cipher->strength_bits]++;
1094                 curr = curr->next;
1095                 }
1096         /*
1097          * Go through the list of used strength_bits values in descending
1098          * order.
1099          */
1100         for (i = max_strength_bits; i >= 0; i--)
1101                 if (number_uses[i] > 0)
1102                         ssl_cipher_apply_rule(0, 0, 0, 0, 0, 0, 0, CIPHER_ORD, i

1104         OPENSSL_free(number_uses);
1105         return(1);
1106         }

1108 static int ssl_cipher_process_rulestr(const char *rule_str,
1109                 CIPHER_ORDER **head_p, CIPHER_ORDER **tail_p,
1110                 const SSL_CIPHER **ca_list)
1111         {
1112         unsigned long alg_mkey, alg_auth, alg_enc, alg_mac, alg_ssl, algo_streng
1113         const char *l, *buf;
1114         int j, multi, found, rule, retval, ok, buflen;
1115         unsigned long cipher_id = 0;
1116         char ch;
```



```
1118         retval = 1;
1119         l = rule_str;
1120         for (;;)
1121                 {
1122                 ch = *l;

1124                 if (ch == '\0')
1125                         break;          /* done */
1126                 if (ch == '-')
1127                         { rule = CIPHER_DEL; l++; }
1128                 else if (ch == '+')
1129                         { rule = CIPHER_ORD; l++; }
1130                 else if (ch == '!')
1131                         { rule = CIPHER_KILL; l++; }
1132                 else if (ch == '@')
1133                         { rule = CIPHER_SPECIAL; l++; }
1134                 else
1135                         { rule = CIPHER_ADD; }

1137                 if (ITEM_SEP(ch))
1138                         {
1139                         l++;
1140                         continue;
1141                         }

1143                 alg_mkey = 0;
1144                 alg_auth = 0;
1145                 alg_enc = 0;
1146                 alg_mac = 0;
1147                 alg_ssl = 0;
1148                 algo_strength = 0;

1150                 for (;;)
1151                         {
1152                         ch = *l;
1153                         buf = l;
1154                         buflen = 0;
1155 #ifndef CHARSET_EBCDIC
1156                         while ( ((ch >= 'A') && (ch <= 'Z')) ||
1157                                 ((ch >= '0') && (ch <= '9')) ||
1158                                 ((ch >= 'a') && (ch <= 'z')) ||
1159                                  (ch == '-') || (ch == '.'))
1160 #else
1161                         while ( isalnum(ch) || (ch == '-') || (ch == '.'))
1162 #endif
1163                                  {
1164                                  ch = *(++l);
1165                                  buflen++;
1166                                  }

1168                         if (buflen == 0)
1169                                 {
1170                                 /*
1171                                  * We hit something we cannot deal with,
1172                                  * it is no command or separator nor
1173                                  * alphanumeric, so we call this an error.
1174                                  */
1175                                 SSLerr(SSL_F_SSL_CIPHER_PROCESS_RULESTR,
1176                                        SSL_R_INVALID_COMMAND);
1177                                 retval = found = 0;
1178                                 l++;
1179                                 break;
1180                                 }

1182                         if (rule == CIPHER_SPECIAL)
1183                                 {
```

```
1184                            found = 0; /* unused -- avoid compiler warning *
1185                            break;  /* special treatment */
1186                            }
1188                    /* check for multi-part specification */
1189                    if (ch == '+')
1190                            {
1191                            multi=1;
1192                            l++;
1193                            }
1194                    else
1195                            multi=0;
1197                    /*
1198                     * Now search for the cipher alias in the ca_list. Be ca
1199                     * with the strncmp, because the "buflen" limitation
1200                     * will make the rule "ADH:SOME" and the cipher
1201                     * "ADH-MY-CIPHER" look like a match for buflen=3.
1202                     * So additionally check whether the cipher name found
1203                     * has the correct length. We can save a strlen() call:
1204                     * just checking for the '\0' at the right place is
1205                     * sufficient, we have to strncmp() anyway. (We cannot
1206                     * use strcmp(), because buf is not '\0' terminated.)
1207                     */
1208                    j = found = 0;
1209                    cipher_id = 0;
1210                    while (ca_list[j])
1211                            {
1212                            if (!strncmp(buf, ca_list[j]->name, buflen) &&
1213                                (ca_list[j]->name[buflen] == '\0'))
1214                                {
1215                                found = 1;
1216                                break;
1217                                }
1218                            else
1219                                j++;
1220                            }
1222                    if (!found)
1223                            break;  /* ignore this entry */
1225                    if (ca_list[j]->algorithm_mkey)
1226                            {
1227                            if (alg_mkey)
1228                                    {
1229                                    alg_mkey &= ca_list[j]->algorithm_mkey;
1230                                    if (!alg_mkey) { found = 0; break; }
1231                                    }
1232                            else
1233                                    alg_mkey = ca_list[j]->algorithm_mkey;
1234                            }
1236                    if (ca_list[j]->algorithm_auth)
1237                            {
1238                            if (alg_auth)
1239                                    {
1240                                    alg_auth &= ca_list[j]->algorithm_auth;
1241                                    if (!alg_auth) { found = 0; break; }
1242                                    }
1243                            else
1244                                    alg_auth = ca_list[j]->algorithm_auth;
1245                            }
1247                    if (ca_list[j]->algorithm_enc)
1248                            {
1249                            if (alg_enc)
```



```
1250                                    {
1251                                    alg_enc &= ca_list[j]->algorithm_enc;
1252                                    if (!alg_enc) { found = 0; break; }
1253                                    }
1254                            else
1255                                    alg_enc = ca_list[j]->algorithm_enc;
1256                            }
1258                    if (ca_list[j]->algorithm_mac)
1259                            {
1260                            if (alg_mac)
1261                                    {
1262                                    alg_mac &= ca_list[j]->algorithm_mac;
1263                                    if (!alg_mac) { found = 0; break; }
1264                                    }
1265                            else
1266                                    alg_mac = ca_list[j]->algorithm_mac;
1267                            }
1269                    if (ca_list[j]->algo_strength & SSL_EXP_MASK)
1270                            {
1271                            if (algo_strength & SSL_EXP_MASK)
1272                                    {
1273                                    algo_strength &= (ca_list[j]->algo_stren
1274                                    if (!(algo_strength & SSL_EXP_MASK)) { f
1275                                    }
1276                            else
1277                                    algo_strength |= ca_list[j]->algo_streng
1278                            }
1280                    if (ca_list[j]->algo_strength & SSL_STRONG_MASK)
1281                            {
1282                            if (algo_strength & SSL_STRONG_MASK)
1283                                    {
1284                                    algo_strength &= (ca_list[j]->algo_stren
1285                                    if (!(algo_strength & SSL_STRONG_MASK))
1286                                    }
1287                            else
1288                                    algo_strength |= ca_list[j]->algo_streng
1289                            }
1291                    if (ca_list[j]->valid)
1292                            {
1293                            /* explicit ciphersuite found; its protocol vers
1294                             * does not become part of the search pattern!*/
1296                            cipher_id = ca_list[j]->id;
1297                            }
1298                    else
1299                            {
1300                            /* not an explicit ciphersuite; only in this cas
1301                             * protocol version is considered part of the se
1303                            if (ca_list[j]->algorithm_ssl)
1304                                    {
1305                                    if (alg_ssl)
1306                                            {
1307                                            alg_ssl &= ca_list[j]->algorithm
1308                                            if (!alg_ssl) { found = 0; break
1309                                            }
1310                                    else
1311                                            alg_ssl = ca_list[j]->algorithm_
1312                                    }
1313                            }
1315                    if (!multi) break;
```

```
1316                        }

1318                /*
1319                 * Ok, we have the rule, now apply it
1320                 */
1321                if (rule == CIPHER_SPECIAL)
1322                        {       /* special command */
1323                        ok = 0;
1324                        if ((buflen == 8) &&
1325                                !strncmp(buf, "STRENGTH", 8))
1326                                ok = ssl_cipher_strength_sort(head_p, tail_p);
1327                        else
1328                                SSLerr(SSL_F_SSL_CIPHER_PROCESS_RULESTR,
1329                                        SSL_R_INVALID_COMMAND);
1330                        if (ok == 0)
1331                                retval = 0;
1332                        /*
1333                         * We do not support any "multi" options
1334                         * together with "@", so throw away the
1335                         * rest of the command, if any left, until
1336                         * end or ':' is found.
1337                         */
1338                        while ((*l != '\0') && !ITEM_SEP(*l))
1339                                l++;
1340                        }
1341                else if (found)
1342                        {
1343                        ssl_cipher_apply_rule(cipher_id,
1344                                alg_mkey, alg_auth, alg_enc, alg_mac, alg_ssl, a
1345                                rule, -1, head_p, tail_p);
1346                        }
1347                else
1348                        {
1349                        while ((*l != '\0') && !ITEM_SEP(*l))
1350                                l++;
1351                        }
1352                if (*l == '\0') break; /* done */
1353                }

1355        return(retval);
1356        }

1358 STACK_OF(SSL_CIPHER) *ssl_create_cipher_list(const SSL_METHOD *ssl_method,
1359                STACK_OF(SSL_CIPHER) **cipher_list,
1360                STACK_OF(SSL_CIPHER) **cipher_list_by_id,
1361                const char *rule_str)
1362        {
1363        int ok, num_of_ciphers, num_of_alias_max, num_of_group_aliases;
1364        unsigned long disabled_mkey, disabled_auth, disabled_enc, disabled_mac,
1365        STACK_OF(SSL_CIPHER) *cipherstack, *tmp_cipher_list;
1366        const char *rule_p;
1367        CIPHER_ORDER *co_list = NULL, *head = NULL, *tail = NULL, *curr;
1368        const SSL_CIPHER **ca_list = NULL;

1370        /*
1371         * Return with error if nothing to do.
1372         */
1373        if (rule_str == NULL || cipher_list == NULL || cipher_list_by_id == NULL
1374                return NULL;

1376        /*
1377         * To reduce the work to do we only want to process the compiled
1378         * in algorithms, so we first get the mask of disabled ciphers.
1379         */
1380        ssl_cipher_get_disabled(&disabled_mkey, &disabled_auth, &disabled_enc, &
```



```
1382        /*
1383         * Now we have to collect the available ciphers from the compiled
1384         * in ciphers. We cannot get more than the number compiled in, so
1385         * it is used for allocation.
1386         */
1387        num_of_ciphers = ssl_method->num_ciphers();
1388 #ifdef KSSL_DEBUG
1389        printf("ssl_create_cipher_list() for %d ciphers\n", num_of_ciphers);
1390 #endif   /* KSSL_DEBUG */
1391        co_list = (CIPHER_ORDER *)OPENSSL_malloc(sizeof(CIPHER_ORDER) * num_of_c
1392        if (co_list == NULL)
1393                {
1394                SSLerr(SSL_F_SSL_CREATE_CIPHER_LIST,ERR_R_MALLOC_FAILURE);
1395                return(NULL);   /* Failure */
1396                }

1398        ssl_cipher_collect_ciphers(ssl_method, num_of_ciphers,
1399                                   disabled_mkey, disabled_auth, disabled_enc, d
1400                                   co_list, &head, &tail);


1403        /* Now arrange all ciphers by preference: */

1405        /* Everything else being equal, prefer ephemeral ECDH over other key exc
1406        ssl_cipher_apply_rule(0, SSL_kEECDH, 0, 0, 0, 0, 0, CIPHER_ADD, -1, &hea
1407        ssl_cipher_apply_rule(0, SSL_kEECDH, 0, 0, 0, 0, 0, CIPHER_DEL, -1, &hea

1409        /* AES is our preferred symmetric cipher */
1410        ssl_cipher_apply_rule(0, 0, 0, SSL_AES, 0, 0, 0, CIPHER_ADD, -1, &head,

1412        /* Temporarily enable everything else for sorting */
1413        ssl_cipher_apply_rule(0, 0, 0, 0, 0, 0, 0, CIPHER_ADD, -1, &head, &tail)

1415        /* Low priority for MD5 */
1416        ssl_cipher_apply_rule(0, 0, 0, 0, SSL_MD5, 0, 0, CIPHER_ORD, -1, &head,

1418        /* Move anonymous ciphers to the end.  Usually, these will remain disabl
1419         * (For applications that allow them, they aren't too bad, but we prefer
1420         * authenticated ciphers.) */
1421        ssl_cipher_apply_rule(0, 0, SSL_aNULL, 0, 0, 0, 0, CIPHER_ORD, -1, &head

1423        /* Move ciphers without forward secrecy to the end */
1424        ssl_cipher_apply_rule(0, 0, SSL_aECDH, 0, 0, 0, 0, CIPHER_ORD, -1, &head
1425        /* ssl_cipher_apply_rule(0, 0, SSL_aDH, 0, 0, 0, 0, CIPHER_ORD, -1, &hea
1426        ssl_cipher_apply_rule(0, SSL_kRSA, 0, 0, 0, 0, 0, CIPHER_ORD, -1, &head,
1427        ssl_cipher_apply_rule(0, SSL_kPSK, 0,0, 0, 0, 0, CIPHER_ORD, -1, &head,
1428        ssl_cipher_apply_rule(0, SSL_kKRB5, 0,0, 0, 0, 0, CIPHER_ORD, -1, &head,

1430        /* RC4 is sort-of broken -- move the the end */
1431        ssl_cipher_apply_rule(0, 0, 0, SSL_RC4, 0, 0, 0, CIPHER_ORD, -1, &head,

1433        /* Now sort by symmetric encryption strength.  The above ordering remain
1434         * in force within each class */
1435        if (!ssl_cipher_strength_sort(&head, &tail))
1436                {
1437                OPENSSL_free(co_list);
1438                return NULL;
1439                }

1441        /* Now disable everything (maintaining the ordering!) */
1442        ssl_cipher_apply_rule(0, 0, 0, 0, 0, 0, 0, CIPHER_DEL, -1, &head, &tail)


1445        /*
1446         * We also need cipher aliases for selecting based on the rule_str.
1447         * There might be two types of entries in the rule_str: 1) names
```

```
1448          * of ciphers themselves 2) aliases for groups of ciphers.
1449          * For 1) we need the available ciphers and for 2) the cipher
1450          * groups of cipher_aliases added together in one list (otherwise
1451          * we would be happy with just the cipher_aliases table).
1452          */
1453         num_of_group_aliases = sizeof(cipher_aliases) / sizeof(SSL_CIPHER);
1454         num_of_alias_max = num_of_ciphers + num_of_group_aliases + 1;
1455         ca_list = OPENSSL_malloc(sizeof(SSL_CIPHER *) * num_of_alias_max);
1456         if (ca_list == NULL)
1457                 {
1458                 OPENSSL_free(co_list);
1459                 SSLerr(SSL_F_SSL_CREATE_CIPHER_LIST,ERR_R_MALLOC_FAILURE);
1460                 return(NULL);   /* Failure */
1461                 }
1462         ssl_cipher_collect_aliases(ca_list, num_of_group_aliases,
1463                                    disabled_mkey, disabled_auth, disabled_enc,
1464                                    disabled_mac, disabled_ssl, head);
1465
1466         /*
1467          * If the rule_string begins with DEFAULT, apply the default rule
1468          * before using the (possibly available) additional rules.
1469          */
1470         ok = 1;
1471         rule_p = rule_str;
1472         if (strncmp(rule_str,"DEFAULT",7) == 0)
1473                 {
1474                 ok = ssl_cipher_process_rulestr(SSL_DEFAULT_CIPHER_LIST,
1475                         &head, &tail, ca_list);
1476                 rule_p += 7;
1477                 if (*rule_p == ':')
1478                         rule_p++;
1479                 }
1480
1481         if (ok && (strlen(rule_p) > 0))
1482                 ok = ssl_cipher_process_rulestr(rule_p, &head, &tail, ca_list);
1483
1484         OPENSSL_free((void *)ca_list);  /* Not needed anymore */
1485
1486         if (!ok)
1487                 {       /* Rule processing failure */
1488                 OPENSSL_free(co_list);
1489                 return(NULL);
1490                 }
1491
1492         /*
1493          * Allocate new "cipherstack" for the result, return with error
1494          * if we cannot get one.
1495          */
1496         if ((cipherstack = sk_SSL_CIPHER_new_null()) == NULL)
1497                 {
1498                 OPENSSL_free(co_list);
1499                 return(NULL);
1500                 }
1501
1502         /*
1503          * The cipher selection for the list is done. The ciphers are added
1504          * to the resulting precedence to the STACK_OF(SSL_CIPHER).
1505          */
1506         for (curr = head; curr != NULL; curr = curr->next)
1507                 {
1508 #ifdef OPENSSL_FIPS
1509                 if (curr->active && (!FIPS_mode() || curr->cipher->algo_strength
1510 #else
1511                 if (curr->active)
1512 #endif
1513                         {
```



```
1514                         sk_SSL_CIPHER_push(cipherstack, curr->cipher);
1515 #ifdef CIPHER_DEBUG
1516                         printf("<%s>\n",curr->cipher->name);
1517 #endif
1518                         }
1519                 }
1520         OPENSSL_free(co_list);  /* Not needed any longer */
1521
1522         tmp_cipher_list = sk_SSL_CIPHER_dup(cipherstack);
1523         if (tmp_cipher_list == NULL)
1524                 {
1525                 sk_SSL_CIPHER_free(cipherstack);
1526                 return NULL;
1527                 }
1528         if (*cipher_list != NULL)
1529                 sk_SSL_CIPHER_free(*cipher_list);
1530         *cipher_list = cipherstack;
1531         if (*cipher_list_by_id != NULL)
1532                 sk_SSL_CIPHER_free(*cipher_list_by_id);
1533         *cipher_list_by_id = tmp_cipher_list;
1534         (void)sk_SSL_CIPHER_set_cmp_func(*cipher_list_by_id,ssl_cipher_ptr_id_cm
1535
1536         sk_SSL_CIPHER_sort(*cipher_list_by_id);
1537         return(cipherstack);
1538         }
1539
1540 char *SSL_CIPHER_description(const SSL_CIPHER *cipher, char *buf, int len)
1541         {
1542         int is_export,pkl,kl;
1543         const char *ver,*exp_str;
1544         const char *kx,*au,*enc,*mac;
1545         unsigned long alg_mkey,alg_auth,alg_enc,alg_mac,alg_ssl,alg2;
1546 #ifdef KSSL_DEBUG
1547         static const char *format="%-23s %s Kx=%-8s Au=%-4s Enc=%-9s Mac=%-4s%s
1548 #else
1549         static const char *format="%-23s %s Kx=%-8s Au=%-4s Enc=%-9s Mac=%-4s%s\
1550 #endif /* KSSL_DEBUG */
1551
1552         alg_mkey = cipher->algorithm_mkey;
1553         alg_auth = cipher->algorithm_auth;
1554         alg_enc = cipher->algorithm_enc;
1555         alg_mac = cipher->algorithm_mac;
1556         alg_ssl = cipher->algorithm_ssl;
1557
1558         alg2=cipher->algorithm2;
1559
1560         is_export=SSL_C_IS_EXPORT(cipher);
1561         pkl=SSL_C_EXPORT_PKEYLENGTH(cipher);
1562         kl=SSL_C_EXPORT_KEYLENGTH(cipher);
1563         exp_str=is_export?" export":"";
1564
1565         if (alg_ssl & SSL_SSLV2)
1566                 ver="SSLv2";
1567         else if (alg_ssl & SSL_SSLV3)
1568                 ver="SSLv3";
1569         else if (alg_ssl & SSL_TLSV1_2)
1570                 ver="TLSv1.2";
1571         else
1572                 ver="unknown";
1573
1574         switch (alg_mkey)
1575                 {
1576         case SSL_kRSA:
1577                 kx=is_export?(pkl == 512 ? "RSA(512)" : "RSA(1024)"):"RSA";
1578                 break;
1579         case SSL_kDHr:
```

```
1580                kx="DH/RSA";
1581                break;
1582        case SSL_kDHd:
1583                kx="DH/DSS";
1584                break;
1585        case SSL_kKRB5:
1586                kx="KRB5";
1587                break;
1588        case SSL_kEDH:
1589                kx=is_export?(pkl == 512 ? "DH(512)" : "DH(1024)"):"DH";
1590                break;
1591        case SSL_kECDHr:
1592                kx="ECDH/RSA";
1593                break;
1594        case SSL_kECDHe:
1595                kx="ECDH/ECDSA";
1596                break;
1597        case SSL_kEECDH:
1598                kx="ECDH";
1599                break;
1600        case SSL_kPSK:
1601                kx="PSK";
1602                break;
1603        case SSL_kSRP:
1604                kx="SRP";
1605                break;
1606        case SSL_kGOST:
1607                kx="GOST";
1608                break;
1609        default:
1610                kx="unknown";
1611                }

1613        switch (alg_auth)
1614                {
1615        case SSL_aRSA:
1616                au="RSA";
1617                break;
1618        case SSL_aDSS:
1619                au="DSS";
1620                break;
1621        case SSL_aDH:
1622                au="DH";
1623                break;
1624        case SSL_aKRB5:
1625                au="KRB5";
1626                break;
1627        case SSL_aECDH:
1628                au="ECDH";
1629                break;
1630        case SSL_aNULL:
1631                au="None";
1632                break;
1633        case SSL_aECDSA:
1634                au="ECDSA";
1635                break;
1636        case SSL_aPSK:
1637                au="PSK";
1638                break;
1639        case SSL_aSRP:
1640                au="SRP";
1641                break;
1642        case SSL_aGOST94:
1643                au="GOST94";
1644                break;
1645        case SSL_aGOST01:
```



```
1646                au="GOST01";
1647                break;
1648        default:
1649                au="unknown";
1650                break;
1651                }

1653        switch (alg_enc)
1654                {
1655        case SSL_DES:
1656                enc=(is_export && kl == 5)?"DES(40)":"DES(56)";
1657                break;
1658        case SSL_3DES:
1659                enc="3DES(168)";
1660                break;
1661        case SSL_RC4:
1662                enc=is_export?(kl == 5 ? "RC4(40)" : "RC4(56)")
1663                  :((alg2&SSL2_CF_8_BYTE_ENC)?"RC4(64)":"RC4(128)");
1664                break;
1665        case SSL_RC2:
1666                enc=is_export?(kl == 5 ? "RC2(40)" : "RC2(56)"):"RC2(128)";
1667                break;
1668        case SSL_IDEA:
1669                enc="IDEA(128)";
1670                break;
1671        case SSL_eNULL:
1672                enc="None";
1673                break;
1674        case SSL_AES128:
1675                enc="AES(128)";
1676                break;
1677        case SSL_AES256:
1678                enc="AES(256)";
1679                break;
1680        case SSL_AES128GCM:
1681                enc="AESGCM(128)";
1682                break;
1683        case SSL_AES256GCM:
1684                enc="AESGCM(256)";
1685                break;
1686        case SSL_CAMELLIA128:
1687                enc="Camellia(128)";
1688                break;
1689        case SSL_CAMELLIA256:
1690                enc="Camellia(256)";
1691                break;
1692        case SSL_SEED:
1693                enc="SEED(128)";
1694                break;
1695        case SSL_eGOST2814789CNT:
1696                enc="GOST89(256)";
1697                break;
1698        default:
1699                enc="unknown";
1700                break;
1701                }

1703        switch (alg_mac)
1704                {
1705        case SSL_MD5:
1706                mac="MD5";
1707                break;
1708        case SSL_SHA1:
1709                mac="SHA1";
1710                break;
1711        case SSL_SHA256:
```

```
1712                 mac="SHA256";
1713                 break;
1714         case SSL_SHA384:
1715                 mac="SHA384";
1716                 break;
1717         case SSL_AEAD:
1718                 mac="AEAD";
1719                 break;
1720         case SSL_GOST89MAC:
1721                 mac="GOST89";
1722                 break;
1723         case SSL_GOST94:
1724                 mac="GOST94";
1725                 break;
1726         default:
1727                 mac="unknown";
1728                 break;
1729                 }

1731         if (buf == NULL)
1732                 {
1733                 len=128;
1734                 buf=OPENSSL_malloc(len);
1735                 if (buf == NULL) return("OPENSSL_malloc Error");
1736                 }
1737         else if (len < 128)
1738                 return("Buffer too small");

1740 #ifdef KSSL_DEBUG
1741         BIO_snprintf(buf,len,format,cipher->name,ver,kx,au,enc,mac,exp_str,alg_m
1742 #else
1743         BIO_snprintf(buf,len,format,cipher->name,ver,kx,au,enc,mac,exp_str);
1744 #endif /* KSSL_DEBUG */
1745         return(buf);
1746         }

1748 char *SSL_CIPHER_get_version(const SSL_CIPHER *c)
1749         {
1750         int i;

1752         if (c == NULL) return("(NONE)");
1753         i=(int)(c->id>>24L);
1754         if (i == 3)
1755                 return("TLSv1/SSLv3");
1756         else if (i == 2)
1757                 return("SSLv2");
1758         else
1759                 return("unknown");
1760         }

1762 /* return the actual cipher being used */
1763 const char *SSL_CIPHER_get_name(const SSL_CIPHER *c)
1764         {
1765         if (c != NULL)
1766                 return(c->name);
1767         return("(NONE)");
1768         }

1770 /* number of bits for symmetric cipher */
1771 int SSL_CIPHER_get_bits(const SSL_CIPHER *c, int *alg_bits)
1772         {
1773         int ret=0;

1775         if (c != NULL)
1776                 {
1777                 if (alg_bits != NULL) *alg_bits = c->alg_bits;
```



```
1778                 ret = c->strength_bits;
1779                 }
1780         return(ret);
1781         }

1783 unsigned long SSL_CIPHER_get_id(const SSL_CIPHER *c)
1784         {
1785         return c->id;
1786         }

1788 SSL_COMP *ssl3_comp_find(STACK_OF(SSL_COMP) *sk, int n)
1789         {
1790         SSL_COMP *ctmp;
1791         int i,nn;

1793         if ((n == 0) || (sk == NULL)) return(NULL);
1794         nn=sk_SSL_COMP_num(sk);
1795         for (i=0; i<nn; i++)
1796                 {
1797                 ctmp=sk_SSL_COMP_value(sk,i);
1798                 if (ctmp->id == n)
1799                         return(ctmp);
1800                 }
1801         return(NULL);
1802         }

1804 #ifdef OPENSSL_NO_COMP
1805 void *SSL_COMP_get_compression_methods(void)
1806         {
1807         return NULL;
1808         }
1809 int SSL_COMP_add_compression_method(int id, void *cm)
1810         {
1811         return 1;
1812         }

1814 const char *SSL_COMP_get_name(const void *comp)
1815         {
1816         return NULL;
1817         }
1818 #else
1819 STACK_OF(SSL_COMP) *SSL_COMP_get_compression_methods(void)
1820         {
1821         load_builtin_compressions();
1822         return(ssl_comp_methods);
1823         }

1825 int SSL_COMP_add_compression_method(int id, COMP_METHOD *cm)
1826         {
1827         SSL_COMP *comp;

1829         if (cm == NULL || cm->type == NID_undef)
1830                 return 1;

1832         /* According to draft-ietf-tls-compression-04.txt, the
1833            compression number ranges should be the following:

1835            0 to 63:    methods defined by the IETF
1836            64 to 192:  external party methods assigned by IANA
1837            193 to 255: reserved for private use */
1838         if (id < 193 || id > 255)
1839                 {
1840                 SSLerr(SSL_F_SSL_COMP_ADD_COMPRESSION_METHOD,SSL_R_COMPRESSION_I
1841                 return 0;
1842                 }
```

```
1844         MemCheck_off();
1845         comp=(SSL_COMP *)OPENSSL_malloc(sizeof(SSL_COMP));
1846         comp->id=id;
1847         comp->method=cm;
1848         load_builtin_compressions();
1849         if (ssl_comp_methods
1850                 && sk_SSL_COMP_find(ssl_comp_methods,comp) >= 0)
1851                 {
1852                 OPENSSL_free(comp);
1853                 MemCheck_on();
1854                 SSLerr(SSL_F_SSL_COMP_ADD_COMPRESSION_METHOD,SSL_R_DUPLICATE_COM
1855                 return(1);
1856                 }
1857         else if ((ssl_comp_methods == NULL)
1858                 || !sk_SSL_COMP_push(ssl_comp_methods,comp))
1859                 {
1860                 OPENSSL_free(comp);
1861                 MemCheck_on();
1862                 SSLerr(SSL_F_SSL_COMP_ADD_COMPRESSION_METHOD,ERR_R_MALLOC_FAILUR
1863                 return(1);
1864                 }
1865         else
1866                 {
1867                 MemCheck_on();
1868                 return(0);
1869                 }
1870         }

1872 const char *SSL_COMP_get_name(const COMP_METHOD *comp)
1873         {
1874         if (comp)
1875                 return comp->name;
1876         return NULL;
1877         }

1879 #endif
1880 #endif /* ! codereview */
```

```
**********************************************************
   39681 Wed Aug 13 19:53:40 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_err.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_err.c */
/* ====================================================================
 * Copyright (c) 1999-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

/* NOTE: this file was auto generated by the mkerr.pl script: any changes
 * made to it will be overwritten when the script next updates this file,
 * only reason strings will be preserved.
 */

#include <stdio.h>
```



```
#include <openssl/err.h>
#include <openssl/ssl.h>

/* BEGIN ERROR CODES */
#ifndef OPENSSL_NO_ERR

#define ERR_FUNC(func) ERR_PACK(ERR_LIB_SSL,func,0)
#define ERR_REASON(reason) ERR_PACK(ERR_LIB_SSL,0,reason)

static ERR_STRING_DATA SSL_str_functs[]=
        {
{ERR_FUNC(SSL_F_CLIENT_CERTIFICATE),    "CLIENT_CERTIFICATE"},
{ERR_FUNC(SSL_F_CLIENT_FINISHED),       "CLIENT_FINISHED"},
{ERR_FUNC(SSL_F_CLIENT_HELLO),  "CLIENT_HELLO"},
{ERR_FUNC(SSL_F_CLIENT_MASTER_KEY),     "CLIENT_MASTER_KEY"},
{ERR_FUNC(SSL_F_D2I_SSL_SESSION),       "d2i_SSL_SESSION"},
{ERR_FUNC(SSL_F_DO_DTLS1_WRITE),        "DO_DTLS1_WRITE"},
{ERR_FUNC(SSL_F_DO_SSL3_WRITE), "DO_SSL3_WRITE"},
{ERR_FUNC(SSL_F_DTLS1_ACCEPT),  "DTLS1_ACCEPT"},
{ERR_FUNC(SSL_F_DTLS1_ADD_CERT_TO_BUF), "DTLS1_ADD_CERT_TO_BUF"},
{ERR_FUNC(SSL_F_DTLS1_BUFFER_RECORD),   "DTLS1_BUFFER_RECORD"},
{ERR_FUNC(SSL_F_DTLS1_CHECK_TIMEOUT_NUM),       "DTLS1_CHECK_TIMEOUT_NUM"},
{ERR_FUNC(SSL_F_DTLS1_CLIENT_HELLO),    "DTLS1_CLIENT_HELLO"},
{ERR_FUNC(SSL_F_DTLS1_CONNECT), "DTLS1_CONNECT"},
{ERR_FUNC(SSL_F_DTLS1_ENC),     "DTLS1_ENC"},
{ERR_FUNC(SSL_F_DTLS1_GET_HELLO_VERIFY),        "DTLS1_GET_HELLO_VERIFY"},
{ERR_FUNC(SSL_F_DTLS1_GET_MESSAGE),     "DTLS1_GET_MESSAGE"},
{ERR_FUNC(SSL_F_DTLS1_GET_MESSAGE_FRAGMENT),    "DTLS1_GET_MESSAGE_FRAGMENT"},
{ERR_FUNC(SSL_F_DTLS1_GET_RECORD),      "DTLS1_GET_RECORD"},
{ERR_FUNC(SSL_F_DTLS1_HANDLE_TIMEOUT),  "DTLS1_HANDLE_TIMEOUT"},
{ERR_FUNC(SSL_F_DTLS1_HEARTBEAT),       "DTLS1_HEARTBEAT"},
{ERR_FUNC(SSL_F_DTLS1_OUTPUT_CERT_CHAIN),       "DTLS1_OUTPUT_CERT_CHAIN"},
{ERR_FUNC(SSL_F_DTLS1_PREPROCESS_FRAGMENT),     "DTLS1_PREPROCESS_FRAGMENT"},
{ERR_FUNC(SSL_F_DTLS1_PROCESS_OUT_OF_SEQ_MESSAGE),      "DTLS1_PROCESS_OUT_OF_SE
{ERR_FUNC(SSL_F_DTLS1_PROCESS_RECORD),  "DTLS1_PROCESS_RECORD"},
{ERR_FUNC(SSL_F_DTLS1_READ_BYTES),      "DTLS1_READ_BYTES"},
{ERR_FUNC(SSL_F_DTLS1_READ_FAILED),     "DTLS1_READ_FAILED"},
{ERR_FUNC(SSL_F_DTLS1_SEND_CERTIFICATE_REQUEST),        "DTLS1_SEND_CERTIFICATE_
{ERR_FUNC(SSL_F_DTLS1_SEND_CLIENT_CERTIFICATE), "DTLS1_SEND_CLIENT_CERTIFICATE"}
{ERR_FUNC(SSL_F_DTLS1_SEND_CLIENT_KEY_EXCHANGE),        "DTLS1_SEND_CLIENT_KEY_E
{ERR_FUNC(SSL_F_DTLS1_SEND_CLIENT_VERIFY),      "DTLS1_SEND_CLIENT_VERIFY"},
{ERR_FUNC(SSL_F_DTLS1_SEND_HELLO_VERIFY_REQUEST),       "DTLS1_SEND_HELLO_VERIFY
{ERR_FUNC(SSL_F_DTLS1_SEND_SERVER_CERTIFICATE), "DTLS1_SEND_SERVER_CERTIFICATE"}
{ERR_FUNC(SSL_F_DTLS1_SEND_SERVER_HELLO),       "DTLS1_SEND_SERVER_HELLO"},
{ERR_FUNC(SSL_F_DTLS1_SEND_SERVER_KEY_EXCHANGE),        "DTLS1_SEND_SERVER_KEY_E
{ERR_FUNC(SSL_F_DTLS1_WRITE_APP_DATA_BYTES),    "DTLS1_WRITE_APP_DATA_BYTES"},
{ERR_FUNC(SSL_F_GET_CLIENT_FINISHED),   "GET_CLIENT_FINISHED"},
{ERR_FUNC(SSL_F_GET_CLIENT_HELLO),      "GET_CLIENT_HELLO"},
{ERR_FUNC(SSL_F_GET_CLIENT_MASTER_KEY), "GET_CLIENT_MASTER_KEY"},
{ERR_FUNC(SSL_F_GET_SERVER_FINISHED),   "GET_SERVER_FINISHED"},
{ERR_FUNC(SSL_F_GET_SERVER_HELLO),      "GET_SERVER_HELLO"},
{ERR_FUNC(SSL_F_GET_SERVER_VERIFY),     "GET_SERVER_VERIFY"},
{ERR_FUNC(SSL_F_I2D_SSL_SESSION),       "i2d_SSL_SESSION"},
{ERR_FUNC(SSL_F_READ_N),        "READ_N"},
{ERR_FUNC(SSL_F_REQUEST_CERTIFICATE),   "REQUEST_CERTIFICATE"},
{ERR_FUNC(SSL_F_SERVER_FINISH), "SERVER_FINISH"},
{ERR_FUNC(SSL_F_SERVER_HELLO),  "SERVER_HELLO"},
{ERR_FUNC(SSL_F_SERVER_VERIFY), "SERVER_VERIFY"},
{ERR_FUNC(SSL_F_SSL23_ACCEPT),  "SSL23_ACCEPT"},
{ERR_FUNC(SSL_F_SSL23_CLIENT_HELLO),    "SSL23_CLIENT_HELLO"},
{ERR_FUNC(SSL_F_SSL23_CONNECT), "SSL23_CONNECT"},
{ERR_FUNC(SSL_F_SSL23_GET_CLIENT_HELLO),        "SSL23_GET_CLIENT_HELLO"},
{ERR_FUNC(SSL_F_SSL23_GET_SERVER_HELLO),        "SSL23_GET_SERVER_HELLO"},
{ERR_FUNC(SSL_F_SSL23_PEEK),    "SSL23_PEEK"},
{ERR_FUNC(SSL_F_SSL23_READ),    "SSL23_READ"},
{ERR_FUNC(SSL_F_SSL23_WRITE),   "SSL23_WRITE"},
```

```
{ERR_FUNC(SSL_F_SSL2_ACCEPT),   "SSL2_ACCEPT"},
{ERR_FUNC(SSL_F_SSL2_CONNECT),  "SSL2_CONNECT"},
{ERR_FUNC(SSL_F_SSL2_ENC_INIT), "SSL2_ENC_INIT"},
{ERR_FUNC(SSL_F_SSL2_GENERATE_KEY_MATERIAL),    "SSL2_GENERATE_KEY_MATERIAL"},
{ERR_FUNC(SSL_F_SSL2_PEEK),     "SSL2_PEEK"},
{ERR_FUNC(SSL_F_SSL2_READ),     "SSL2_READ"},
{ERR_FUNC(SSL_F_SSL2_READ_INTERNAL),    "SSL2_READ_INTERNAL"},
{ERR_FUNC(SSL_F_SSL2_SET_CERTIFICATE),  "SSL2_SET_CERTIFICATE"},
{ERR_FUNC(SSL_F_SSL2_WRITE),    "SSL2_WRITE"},
{ERR_FUNC(SSL_F_SSL3_ACCEPT),   "SSL3_ACCEPT"},
{ERR_FUNC(SSL_F_SSL3_ADD_CERT_TO_BUF),  "SSL3_ADD_CERT_TO_BUF"},
{ERR_FUNC(SSL_F_SSL3_CALLBACK_CTRL),    "SSL3_CALLBACK_CTRL"},
{ERR_FUNC(SSL_F_SSL3_CHANGE_CIPHER_STATE),      "SSL3_CHANGE_CIPHER_STATE"},
{ERR_FUNC(SSL_F_SSL3_CHECK_CERT_AND_ALGORITHM), "SSL3_CHECK_CERT_AND_ALGORITHM"}
{ERR_FUNC(SSL_F_SSL3_CHECK_CLIENT_HELLO),       "SSL3_CHECK_CLIENT_HELLO"},
{ERR_FUNC(SSL_F_SSL3_CLIENT_HELLO),     "SSL3_CLIENT_HELLO"},
{ERR_FUNC(SSL_F_SSL3_CONNECT),  "SSL3_CONNECT"},
{ERR_FUNC(SSL_F_SSL3_CTRL),     "SSL3_CTRL"},
{ERR_FUNC(SSL_F_SSL3_CTX_CTRL), "SSL3_CTX_CTRL"},
{ERR_FUNC(SSL_F_SSL3_DIGEST_CACHED_RECORDS),    "SSL3_DIGEST_CACHED_RECORDS"},
{ERR_FUNC(SSL_F_SSL3_DO_CHANGE_CIPHER_SPEC),    "SSL3_DO_CHANGE_CIPHER_SPEC"},
{ERR_FUNC(SSL_F_SSL3_ENC),      "SSL3_ENC"},
{ERR_FUNC(SSL_F_SSL3_GENERATE_KEY_BLOCK),       "SSL3_GENERATE_KEY_BLOCK"},
{ERR_FUNC(SSL_F_SSL3_GET_CERTIFICATE_REQUEST),  "SSL3_GET_CERTIFICATE_REQUEST"},
{ERR_FUNC(SSL_F_SSL3_GET_CERT_STATUS),  "SSL3_GET_CERT_STATUS"},
{ERR_FUNC(SSL_F_SSL3_GET_CERT_VERIFY),  "SSL3_GET_CERT_VERIFY"},
{ERR_FUNC(SSL_F_SSL3_GET_CLIENT_CERTIFICATE),   "SSL3_GET_CLIENT_CERTIFICATE"},
{ERR_FUNC(SSL_F_SSL3_GET_CLIENT_HELLO), "SSL3_GET_CLIENT_HELLO"},
{ERR_FUNC(SSL_F_SSL3_GET_CLIENT_KEY_EXCHANGE),  "SSL3_GET_CLIENT_KEY_EXCHANGE"},
{ERR_FUNC(SSL_F_SSL3_GET_FINISHED),     "SSL3_GET_FINISHED"},
{ERR_FUNC(SSL_F_SSL3_GET_KEY_EXCHANGE), "SSL3_GET_KEY_EXCHANGE"},
{ERR_FUNC(SSL_F_SSL3_GET_MESSAGE),      "SSL3_GET_MESSAGE"},
{ERR_FUNC(SSL_F_SSL3_GET_NEW_SESSION_TICKET),   "SSL3_GET_NEW_SESSION_TICKET"},
{ERR_FUNC(SSL_F_SSL3_GET_NEXT_PROTO),   "SSL3_GET_NEXT_PROTO"},
{ERR_FUNC(SSL_F_SSL3_GET_RECORD),       "SSL3_GET_RECORD"},
{ERR_FUNC(SSL_F_SSL3_GET_SERVER_CERTIFICATE),   "SSL3_GET_SERVER_CERTIFICATE"},
{ERR_FUNC(SSL_F_SSL3_GET_SERVER_DONE),  "SSL3_GET_SERVER_DONE"},
{ERR_FUNC(SSL_F_SSL3_GET_SERVER_HELLO), "SSL3_GET_SERVER_HELLO"},
{ERR_FUNC(SSL_F_SSL3_HANDSHAKE_MAC),    "ssl3_handshake_mac"},
{ERR_FUNC(SSL_F_SSL3_NEW_SESSION_TICKET),       "SSL3_NEW_SESSION_TICKET"},
{ERR_FUNC(SSL_F_SSL3_OUTPUT_CERT_CHAIN),        "SSL3_OUTPUT_CERT_CHAIN"},
{ERR_FUNC(SSL_F_SSL3_PEEK),     "SSL3_PEEK"},
{ERR_FUNC(SSL_F_SSL3_READ_BYTES),       "SSL3_READ_BYTES"},
{ERR_FUNC(SSL_F_SSL3_READ_N),   "SSL3_READ_N"},
{ERR_FUNC(SSL_F_SSL3_SEND_CERTIFICATE_REQUEST), "SSL3_SEND_CERTIFICATE_REQUEST"}
{ERR_FUNC(SSL_F_SSL3_SEND_CLIENT_CERTIFICATE),  "SSL3_SEND_CLIENT_CERTIFICATE"},
{ERR_FUNC(SSL_F_SSL3_SEND_CLIENT_KEY_EXCHANGE), "SSL3_SEND_CLIENT_KEY_EXCHANGE"}
{ERR_FUNC(SSL_F_SSL3_SEND_CLIENT_VERIFY),       "SSL3_SEND_CLIENT_VERIFY"},
{ERR_FUNC(SSL_F_SSL3_SEND_SERVER_CERTIFICATE),  "SSL3_SEND_SERVER_CERTIFICATE"},
{ERR_FUNC(SSL_F_SSL3_SEND_SERVER_HELLO),        "SSL3_SEND_SERVER_HELLO"},
{ERR_FUNC(SSL_F_SSL3_SEND_SERVER_KEY_EXCHANGE), "SSL3_SEND_SERVER_KEY_EXCHANGE"}
{ERR_FUNC(SSL_F_SSL3_SETUP_KEY_BLOCK),  "SSL3_SETUP_KEY_BLOCK"},
{ERR_FUNC(SSL_F_SSL3_SETUP_READ_BUFFER),        "SSL3_SETUP_READ_BUFFER"},
{ERR_FUNC(SSL_F_SSL3_SETUP_WRITE_BUFFER),       "SSL3_SETUP_WRITE_BUFFER"},
{ERR_FUNC(SSL_F_SSL3_WRITE_BYTES),      "SSL3_WRITE_BYTES"},
{ERR_FUNC(SSL_F_SSL3_WRITE_PENDING),    "SSL3_WRITE_PENDING"},
{ERR_FUNC(SSL_F_SSL_ADD_CLIENTHELLO_RENEGOTIATE_EXT),   "SSL_ADD_CLIENTHELLO_REN
{ERR_FUNC(SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT),    "SSL_ADD_CLIENTHELLO_TLSEXT"},
{ERR_FUNC(SSL_F_SSL_ADD_CLIENTHELLO_USE_SRTP_EXT),      "SSL_ADD_CLIENTHELLO_USE
{ERR_FUNC(SSL_F_SSL_ADD_DIR_CERT_SUBJECTS_TO_STACK),    "SSL_add_dir_cert_subjec
{ERR_FUNC(SSL_F_SSL_ADD_FILE_CERT_SUBJECTS_TO_STACK),   "SSL_add_file_cert_subje
{ERR_FUNC(SSL_F_SSL_ADD_SERVERHELLO_RENEGOTIATE_EXT),   "SSL_ADD_SERVERHELLO_REN
{ERR_FUNC(SSL_F_SSL_ADD_SERVERHELLO_TLSEXT),    "SSL_ADD_SERVERHELLO_TLSEXT"},
{ERR_FUNC(SSL_F_SSL_ADD_SERVERHELLO_USE_SRTP_EXT),      "SSL_ADD_SERVERHELLO_USE
{ERR_FUNC(SSL_F_SSL_BAD_METHOD),        "SSL_BAD_METHOD"},
{ERR_FUNC(SSL_F_SSL_BYTES_TO_CIPHER_LIST),      "SSL_BYTES_TO_CIPHER_LIST"},
```



```
{ERR_FUNC(SSL_F_SSL_CERT_DUP),  "SSL_CERT_DUP"},
{ERR_FUNC(SSL_F_SSL_CERT_INST), "SSL_CERT_INST"},
{ERR_FUNC(SSL_F_SSL_CERT_INSTANTIATE),  "SSL_CERT_INSTANTIATE"},
{ERR_FUNC(SSL_F_SSL_CERT_NEW),  "SSL_CERT_NEW"},
{ERR_FUNC(SSL_F_SSL_CHECK_PRIVATE_KEY), "SSL_check_private_key"},
{ERR_FUNC(SSL_F_SSL_CHECK_SERVERHELLO_TLSEXT),  "SSL_CHECK_SERVERHELLO_TLSEXT"},
{ERR_FUNC(SSL_F_SSL_CHECK_SRVR_ECC_CERT_AND_ALG),       "SSL_CHECK_SRVR_ECC_CERT
{ERR_FUNC(SSL_F_SSL_CIPHER_PROCESS_RULESTR),    "SSL_CIPHER_PROCESS_RULESTR"},
{ERR_FUNC(SSL_F_SSL_CIPHER_STRENGTH_SORT),      "SSL_CIPHER_STRENGTH_SORT"},
{ERR_FUNC(SSL_F_SSL_CLEAR),     "SSL_clear"},
{ERR_FUNC(SSL_F_SSL_COMP_ADD_COMPRESSION_METHOD),       "SSL_COMP_add_compressio
{ERR_FUNC(SSL_F_SSL_CREATE_CIPHER_LIST),        "SSL_CREATE_CIPHER_LIST"},
{ERR_FUNC(SSL_F_SSL_CTRL),      "SSL_ctrl"},
{ERR_FUNC(SSL_F_SSL_CTX_CHECK_PRIVATE_KEY),     "SSL_CTX_check_private_key"},
{ERR_FUNC(SSL_F_SSL_CTX_MAKE_PROFILES), "SSL_CTX_MAKE_PROFILES"},
{ERR_FUNC(SSL_F_SSL_CTX_NEW),   "SSL_CTX_new"},
{ERR_FUNC(SSL_F_SSL_CTX_SET_CIPHER_LIST),       "SSL_CTX_set_cipher_list"},
{ERR_FUNC(SSL_F_SSL_CTX_SET_CLIENT_CERT_ENGINE),        "SSL_CTX_set_client_cert
{ERR_FUNC(SSL_F_SSL_CTX_SET_PURPOSE),   "SSL_CTX_set_purpose"},
{ERR_FUNC(SSL_F_SSL_CTX_SET_SESSION_ID_CONTEXT),        "SSL_CTX_set_session_id_
{ERR_FUNC(SSL_F_SSL_CTX_SET_SSL_VERSION),       "SSL_CTX_set_ssl_version"},
{ERR_FUNC(SSL_F_SSL_CTX_SET_TRUST),     "SSL_CTX_set_trust"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_CERTIFICATE),       "SSL_CTX_use_certificate"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_CERTIFICATE_ASN1),  "SSL_CTX_use_certificate_ASN1"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_CERTIFICATE_CHAIN_FILE),    "SSL_CTX_use_certificate
{ERR_FUNC(SSL_F_SSL_CTX_USE_CERTIFICATE_FILE),  "SSL_CTX_use_certificate_file"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_PRIVATEKEY),        "SSL_CTX_use_PrivateKey"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_PRIVATEKEY_ASN1),   "SSL_CTX_use_PrivateKey_ASN1"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_PRIVATEKEY_FILE),   "SSL_CTX_use_PrivateKey_file"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_PSK_IDENTITY_HINT), "SSL_CTX_use_psk_identity_hint"}
{ERR_FUNC(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY),     "SSL_CTX_use_RSAPrivateKey"},
{ERR_FUNC(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_ASN1),        "SSL_CTX_use_RSAPrivateK
{ERR_FUNC(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_FILE),        "SSL_CTX_use_RSAPrivateK
{ERR_FUNC(SSL_F_SSL_DO_HANDSHAKE),      "SSL_do_handshake"},
{ERR_FUNC(SSL_F_SSL_GET_NEW_SESSION),   "SSL_GET_NEW_SESSION"},
{ERR_FUNC(SSL_F_SSL_GET_PREV_SESSION),  "SSL_GET_PREV_SESSION"},
{ERR_FUNC(SSL_F_SSL_GET_SERVER_SEND_CERT),      "SSL_GET_SERVER_SEND_CERT"},
{ERR_FUNC(SSL_F_SSL_GET_SERVER_SEND_PKEY),      "SSL_GET_SERVER_SEND_PKEY"},
{ERR_FUNC(SSL_F_SSL_GET_SIGN_PKEY),     "SSL_GET_SIGN_PKEY"},
{ERR_FUNC(SSL_F_SSL_INIT_WBIO_BUFFER),  "SSL_INIT_WBIO_BUFFER"},
{ERR_FUNC(SSL_F_SSL_LOAD_CLIENT_CA_FILE),       "SSL_load_client_CA_file"},
{ERR_FUNC(SSL_F_SSL_NEW),       "SSL_new"},
{ERR_FUNC(SSL_F_SSL_PARSE_CLIENTHELLO_RENEGOTIATE_EXT), "SSL_PARSE_CLIENTHELLO_R
{ERR_FUNC(SSL_F_SSL_PARSE_CLIENTHELLO_TLSEXT),  "SSL_PARSE_CLIENTHELLO_TLSEXT"},
{ERR_FUNC(SSL_F_SSL_PARSE_CLIENTHELLO_USE_SRTP_EXT),    "SSL_PARSE_CLIENTHELLO_U
{ERR_FUNC(SSL_F_SSL_PARSE_SERVERHELLO_RENEGOTIATE_EXT), "SSL_PARSE_SERVERHELLO_R
{ERR_FUNC(SSL_F_SSL_PARSE_SERVERHELLO_TLSEXT),  "SSL_PARSE_SERVERHELLO_TLSEXT"},
{ERR_FUNC(SSL_F_SSL_PARSE_SERVERHELLO_USE_SRTP_EXT),    "SSL_PARSE_SERVERHELLO_U
{ERR_FUNC(SSL_F_SSL_PEEK),      "SSL_peek"},
{ERR_FUNC(SSL_F_SSL_PREPARE_CLIENTHELLO_TLSEXT),        "SSL_PREPARE_CLIENTHELLO
{ERR_FUNC(SSL_F_SSL_PREPARE_SERVERHELLO_TLSEXT),        "SSL_PREPARE_SERVERHELLO
{ERR_FUNC(SSL_F_SSL_READ),      "SSL_read"},
{ERR_FUNC(SSL_F_SSL_RSA_PRIVATE_DECRYPT),       "SSL_RSA_PRIVATE_DECRYPT"},
{ERR_FUNC(SSL_F_SSL_RSA_PUBLIC_ENCRYPT),        "SSL_RSA_PUBLIC_ENCRYPT"},
{ERR_FUNC(SSL_F_SSL_SESSION_NEW),       "SSL_SESSION_new"},
{ERR_FUNC(SSL_F_SSL_SESSION_PRINT_FP),  "SSL_SESSION_print_fp"},
{ERR_FUNC(SSL_F_SSL_SESSION_SET1_ID_CONTEXT),   "SSL_SESSION_set1_id_context"},
{ERR_FUNC(SSL_F_SSL_SESS_CERT_NEW),     "SSL_SESS_CERT_NEW"},
{ERR_FUNC(SSL_F_SSL_SET_CERT),  "SSL_SET_CERT"},
{ERR_FUNC(SSL_F_SSL_SET_CIPHER_LIST),   "SSL_set_cipher_list"},
{ERR_FUNC(SSL_F_SSL_SET_FD),    "SSL_set_fd"},
{ERR_FUNC(SSL_F_SSL_SET_PKEY),  "SSL_SET_PKEY"},
{ERR_FUNC(SSL_F_SSL_SET_PURPOSE),       "SSL_set_purpose"},
{ERR_FUNC(SSL_F_SSL_SET_RFD),   "SSL_set_rfd"},
{ERR_FUNC(SSL_F_SSL_SET_SESSION),       "SSL_set_session"},
{ERR_FUNC(SSL_F_SSL_SET_SESSION_ID_CONTEXT),    "SSL_set_session_id_context"},
```

```
{ERR_FUNC(SSL_F_SSL_SET_SESSION_TICKET_EXT),    "SSL_set_session_ticket_ext"},
{ERR_FUNC(SSL_F_SSL_SET_TRUST), "SSL_set_trust"},
{ERR_FUNC(SSL_F_SSL_SET_WFD),   "SSL_set_wfd"},
{ERR_FUNC(SSL_F_SSL_SHUTDOWN),  "SSL_shutdown"},
{ERR_FUNC(SSL_F_SSL_SRP_CTX_INIT),      "SSL_SRP_CTX_init"},
{ERR_FUNC(SSL_F_SSL_UNDEFINED_CONST_FUNCTION),  "SSL_UNDEFINED_CONST_FUNCTION"},
{ERR_FUNC(SSL_F_SSL_UNDEFINED_FUNCTION),        "SSL_UNDEFINED_FUNCTION"},
{ERR_FUNC(SSL_F_SSL_UNDEFINED_VOID_FUNCTION),   "SSL_UNDEFINED_VOID_FUNCTION"},
{ERR_FUNC(SSL_F_SSL_USE_CERTIFICATE),   "SSL_use_certificate"},
{ERR_FUNC(SSL_F_SSL_USE_CERTIFICATE_ASN1),      "SSL_use_certificate_ASN1"},
{ERR_FUNC(SSL_F_SSL_USE_CERTIFICATE_FILE),      "SSL_use_certificate_file"},
{ERR_FUNC(SSL_F_SSL_USE_PRIVATEKEY),    "SSL_use_PrivateKey"},
{ERR_FUNC(SSL_F_SSL_USE_PRIVATEKEY_ASN1),       "SSL_use_PrivateKey_ASN1"},
{ERR_FUNC(SSL_F_SSL_USE_PRIVATEKEY_FILE),       "SSL_use_PrivateKey_file"},
{ERR_FUNC(SSL_F_SSL_USE_PSK_IDENTITY_HINT),     "SSL_use_psk_identity_hint"},
{ERR_FUNC(SSL_F_SSL_USE_RSAPRIVATEKEY), "SSL_use_RSAPrivateKey"},
{ERR_FUNC(SSL_F_SSL_USE_RSAPRIVATEKEY_ASN1),    "SSL_use_RSAPrivateKey_ASN1"},
{ERR_FUNC(SSL_F_SSL_USE_RSAPRIVATEKEY_FILE),    "SSL_use_RSAPrivateKey_file"},
{ERR_FUNC(SSL_F_SSL_VERIFY_CERT_CHAIN), "SSL_VERIFY_CERT_CHAIN"},
{ERR_FUNC(SSL_F_SSL_WRITE),     "SSL_write"},
{ERR_FUNC(SSL_F_TLS1_CERT_VERIFY_MAC),  "tls1_cert_verify_mac"},
{ERR_FUNC(SSL_F_TLS1_CHANGE_CIPHER_STATE),      "TLS1_CHANGE_CIPHER_STATE"},
{ERR_FUNC(SSL_F_TLS1_CHECK_SERVERHELLO_TLSEXT), "TLS1_CHECK_SERVERHELLO_TLSEXT"}
{ERR_FUNC(SSL_F_TLS1_ENC),      "TLS1_ENC"},
{ERR_FUNC(SSL_F_TLS1_EXPORT_KEYING_MATERIAL),   "TLS1_EXPORT_KEYING_MATERIAL"},
{ERR_FUNC(SSL_F_TLS1_HEARTBEAT),        "SSL_F_TLS1_HEARTBEAT"},
{ERR_FUNC(SSL_F_TLS1_PREPARE_CLIENTHELLO_TLSEXT),       "TLS1_PREPARE_CLIENTHELL
{ERR_FUNC(SSL_F_TLS1_PREPARE_SERVERHELLO_TLSEXT),       "TLS1_PREPARE_SERVERHELL
{ERR_FUNC(SSL_F_TLS1_PRF),      "tls1_prf"},
{ERR_FUNC(SSL_F_TLS1_SETUP_KEY_BLOCK),  "TLS1_SETUP_KEY_BLOCK"},
{ERR_FUNC(SSL_F_WRITE_PENDING), "WRITE_PENDING"},
{0,NULL}
        };

static ERR_STRING_DATA SSL_str_reasons[]=
        {
{ERR_REASON(SSL_R_APP_DATA_IN_HANDSHAKE) ,"app data in handshake"},
{ERR_REASON(SSL_R_ATTEMPT_TO_REUSE_SESSION_IN_DIFFERENT_CONTEXT),"attempt to reu
{ERR_REASON(SSL_R_BAD_ALERT_RECORD)      ,"bad alert record"},
{ERR_REASON(SSL_R_BAD_AUTHENTICATION_TYPE),"bad authentication type"},
{ERR_REASON(SSL_R_BAD_CHANGE_CIPHER_SPEC),"bad change cipher spec"},
{ERR_REASON(SSL_R_BAD_CHECKSUM)          ,"bad checksum"},
{ERR_REASON(SSL_R_BAD_DATA_RETURNED_BY_CALLBACK),"bad data returned by callback"
{ERR_REASON(SSL_R_BAD_DECOMPRESSION)     ,"bad decompression"},
{ERR_REASON(SSL_R_BAD_DH_G_LENGTH)       ,"bad dh g length"},
{ERR_REASON(SSL_R_BAD_DH_PUB_KEY_LENGTH) ,"bad dh pub key length"},
{ERR_REASON(SSL_R_BAD_DH_P_LENGTH)       ,"bad dh p length"},
{ERR_REASON(SSL_R_BAD_DIGEST_LENGTH)     ,"bad digest length"},
{ERR_REASON(SSL_R_BAD_DSA_SIGNATURE)     ,"bad dsa signature"},
{ERR_REASON(SSL_R_BAD_ECC_CERT)          ,"bad ecc cert"},
{ERR_REASON(SSL_R_BAD_ECDSA_SIGNATURE)   ,"bad ecdsa signature"},
{ERR_REASON(SSL_R_BAD_ECPOINT)           ,"bad ecpoint"},
{ERR_REASON(SSL_R_BAD_HANDSHAKE_LENGTH)  ,"bad handshake length"},
{ERR_REASON(SSL_R_BAD_HELLO_REQUEST)     ,"bad hello request"},
{ERR_REASON(SSL_R_BAD_LENGTH)            ,"bad length"},
{ERR_REASON(SSL_R_BAD_MAC_DECODE)        ,"bad mac decode"},
{ERR_REASON(SSL_R_BAD_MAC_LENGTH)        ,"bad mac length"},
{ERR_REASON(SSL_R_BAD_MESSAGE_TYPE)      ,"bad message type"},
{ERR_REASON(SSL_R_BAD_PACKET_LENGTH)     ,"bad packet length"},
{ERR_REASON(SSL_R_BAD_PROTOCOL_VERSION_NUMBER),"bad protocol version number"},
{ERR_REASON(SSL_R_BAD_PSK_IDENTITY_HINT_LENGTH),"bad psk identity hint length"},
{ERR_REASON(SSL_R_BAD_RESPONSE_ARGUMENT) ,"bad response argument"},
{ERR_REASON(SSL_R_BAD_RSA_DECRYPT)       ,"bad rsa decrypt"},
{ERR_REASON(SSL_R_BAD_RSA_ENCRYPT)       ,"bad rsa encrypt"},
{ERR_REASON(SSL_R_BAD_RSA_E_LENGTH)      ,"bad rsa e length"},
{ERR_REASON(SSL_R_BAD_RSA_MODULUS_LENGTH),"bad rsa modulus length"},
```



```
{ERR_REASON(SSL_R_BAD_RSA_SIGNATURE)     ,"bad rsa signature"},
{ERR_REASON(SSL_R_BAD_SIGNATURE)         ,"bad signature"},
{ERR_REASON(SSL_R_BAD_SRP_A_LENGTH)      ,"bad srp a length"},
{ERR_REASON(SSL_R_BAD_SRP_B_LENGTH)      ,"bad srp b length"},
{ERR_REASON(SSL_R_BAD_SRP_G_LENGTH)      ,"bad srp g length"},
{ERR_REASON(SSL_R_BAD_SRP_N_LENGTH)      ,"bad srp n length"},
{ERR_REASON(SSL_R_BAD_SRP_PARAMETERS)    ,"bad srp parameters"},
{ERR_REASON(SSL_R_BAD_SRP_S_LENGTH)      ,"bad srp s length"},
{ERR_REASON(SSL_R_BAD_SRTP_MKI_VALUE)    ,"bad srtp mki value"},
{ERR_REASON(SSL_R_BAD_SRTP_PROTECTION_PROFILE_LIST),"bad srtp protection profile
{ERR_REASON(SSL_R_BAD_SSL_FILETYPE)      ,"bad ssl filetype"},
{ERR_REASON(SSL_R_BAD_SSL_SESSION_ID_LENGTH),"bad ssl session id length"},
{ERR_REASON(SSL_R_BAD_STATE)             ,"bad state"},
{ERR_REASON(SSL_R_BAD_WRITE_RETRY)       ,"bad write retry"},
{ERR_REASON(SSL_R_BIO_NOT_SET)           ,"bio not set"},
{ERR_REASON(SSL_R_BLOCK_CIPHER_PAD_IS_WRONG),"block cipher pad is wrong"},
{ERR_REASON(SSL_R_BN_LIB)                ,"bn lib"},
{ERR_REASON(SSL_R_CA_DN_LENGTH_MISMATCH) ,"ca dn length mismatch"},
{ERR_REASON(SSL_R_CA_DN_TOO_LONG)        ,"ca dn too long"},
{ERR_REASON(SSL_R_CCS_RECEIVED_EARLY)    ,"ccs received early"},
{ERR_REASON(SSL_R_CERTIFICATE_VERIFY_FAILED),"certificate verify failed"},
{ERR_REASON(SSL_R_CERT_LENGTH_MISMATCH)  ,"cert length mismatch"},
{ERR_REASON(SSL_R_CHALLENGE_IS_DIFFERENT),"challenge is different"},
{ERR_REASON(SSL_R_CIPHER_CODE_WRONG_LENGTH),"cipher code wrong length"},
{ERR_REASON(SSL_R_CIPHER_OR_HASH_UNAVAILABLE),"cipher or hash unavailable"},
{ERR_REASON(SSL_R_CIPHER_TABLE_SRC_ERROR),"cipher table src error"},
{ERR_REASON(SSL_R_CLIENTHELLO_TLSEXT)    ,"clienthello tlsext"},
{ERR_REASON(SSL_R_COMPRESSED_LENGTH_TOO_LONG),"compressed length too long"},
{ERR_REASON(SSL_R_COMPRESSION_DISABLED)  ,"compression disabled"},
{ERR_REASON(SSL_R_COMPRESSION_FAILURE)   ,"compression failure"},
{ERR_REASON(SSL_R_COMPRESSION_ID_NOT_WITHIN_PRIVATE_RANGE),"compression id not w
{ERR_REASON(SSL_R_COMPRESSION_LIBRARY_ERROR),"compression library error"},
{ERR_REASON(SSL_R_CONNECTION_ID_IS_DIFFERENT),"connection id is different"},
{ERR_REASON(SSL_R_CONNECTION_TYPE_NOT_SET),"connection type not set"},
{ERR_REASON(SSL_R_COOKIE_MISMATCH)       ,"cookie mismatch"},
{ERR_REASON(SSL_R_DATA_BETWEEN_CCS_AND_FINISHED),"data between ccs and finished"
{ERR_REASON(SSL_R_DATA_LENGTH_TOO_LONG)  ,"data length too long"},
{ERR_REASON(SSL_R_DECRYPTION_FAILED)     ,"decryption failed"},
{ERR_REASON(SSL_R_DECRYPTION_FAILED_OR_BAD_RECORD_MAC),"decryption failed or bad
{ERR_REASON(SSL_R_DH_PUBLIC_VALUE_LENGTH_IS_WRONG),"dh public value length is wr
{ERR_REASON(SSL_R_DIGEST_CHECK_FAILED)   ,"digest check failed"},
{ERR_REASON(SSL_R_DTLS_MESSAGE_TOO_BIG)  ,"dtls message too big"},
{ERR_REASON(SSL_R_DUPLICATE_COMPRESSION_ID),"duplicate compression id"},
{ERR_REASON(SSL_R_ECC_CERT_NOT_FOR_KEY_AGREEMENT),"ecc cert not for key agreemen
{ERR_REASON(SSL_R_ECC_CERT_NOT_FOR_SIGNING),"ecc cert not for signing"},
{ERR_REASON(SSL_R_ECC_CERT_SHOULD_HAVE_RSA_SIGNATURE),"ecc cert should have rsa
{ERR_REASON(SSL_R_ECC_CERT_SHOULD_HAVE_SHA1_SIGNATURE),"ecc cert should have sha
{ERR_REASON(SSL_R_ECGROUP_TOO_LARGE_FOR_CIPHER),"ecgroup too large for cipher"},
{ERR_REASON(SSL_R_EMPTY_SRTP_PROTECTION_PROFILE_LIST),"empty srtp protection pro
{ERR_REASON(SSL_R_ENCRYPTED_LENGTH_TOO_LONG),"encrypted length too long"},
{ERR_REASON(SSL_R_ERROR_GENERATING_TMP_RSA_KEY),"error generating tmp rsa key"},
{ERR_REASON(SSL_R_ERROR_IN_RECEIVED_CIPHER_LIST),"error in received cipher list"
{ERR_REASON(SSL_R_EXCESSIVE_MESSAGE_SIZE),"excessive message size"},
{ERR_REASON(SSL_R_EXTRA_DATA_IN_MESSAGE) ,"extra data in message"},
{ERR_REASON(SSL_R_GOT_A_FIN_BEFORE_A_CCS),"got a fin before a ccs"},
{ERR_REASON(SSL_R_GOT_NEXT_PROTO_BEFORE_A_CCS),"got next proto before a ccs"},
{ERR_REASON(SSL_R_GOT_NEXT_PROTO_WITHOUT_EXTENSION),"got next proto without seei
{ERR_REASON(SSL_R_HTTPS_PROXY_REQUEST)   ,"https proxy request"},
{ERR_REASON(SSL_R_HTTP_REQUEST)          ,"http request"},
{ERR_REASON(SSL_R_ILLEGAL_PADDING)       ,"illegal padding"},
{ERR_REASON(SSL_R_INCONSISTENT_COMPRESSION),"inconsistent compression"},
{ERR_REASON(SSL_R_INVALID_CHALLENGE_LENGTH),"invalid challenge length"},
{ERR_REASON(SSL_R_INVALID_COMMAND)       ,"invalid command"},
{ERR_REASON(SSL_R_INVALID_COMPRESSION_ALGORITHM),"invalid compression algorithm"
{ERR_REASON(SSL_R_INVALID_PURPOSE)       ,"invalid purpose"},
{ERR_REASON(SSL_R_INVALID_SRP_USERNAME)  ,"invalid srp username"},
```

```
{ERR_REASON(SSL_R_INVALID_STATUS_RESPONSE),"invalid status response"},
{ERR_REASON(SSL_R_INVALID_TICKET_KEYS_LENGTH),"invalid ticket keys length"},
{ERR_REASON(SSL_R_INVALID_TRUST)         ,"invalid trust"},
{ERR_REASON(SSL_R_KEY_ARG_TOO_LONG)      ,"key arg too long"},
{ERR_REASON(SSL_R_KRB5)                  ,"krb5"},
{ERR_REASON(SSL_R_KRB5_C_CC_PRINC)       ,"krb5 client cc principal (no tkt?)"},
{ERR_REASON(SSL_R_KRB5_C_GET_CRED)       ,"krb5 client get cred"},
{ERR_REASON(SSL_R_KRB5_C_INIT)           ,"krb5 client init"},
{ERR_REASON(SSL_R_KRB5_C_MK_REQ)         ,"krb5 client mk_req (expired tkt?)"},
{ERR_REASON(SSL_R_KRB5_S_BAD_TICKET)     ,"krb5 server bad ticket"},
{ERR_REASON(SSL_R_KRB5_S_INIT)           ,"krb5 server init"},
{ERR_REASON(SSL_R_KRB5_S_RD_REQ)         ,"krb5 server rd_req (keytab perms?)"},
{ERR_REASON(SSL_R_KRB5_S_TKT_EXPIRED)    ,"krb5 server tkt expired"},
{ERR_REASON(SSL_R_KRB5_S_TKT_NYV)        ,"krb5 server tkt not yet valid"},
{ERR_REASON(SSL_R_KRB5_S_TKT_SKEW)       ,"krb5 server tkt skew"},
{ERR_REASON(SSL_R_LENGTH_MISMATCH)       ,"length mismatch"},
{ERR_REASON(SSL_R_LENGTH_TOO_SHORT)      ,"length too short"},
{ERR_REASON(SSL_R_LIBRARY_BUG)           ,"library bug"},
{ERR_REASON(SSL_R_LIBRARY_HAS_NO_CIPHERS),"library has no ciphers"},
{ERR_REASON(SSL_R_MESSAGE_TOO_LONG)      ,"message too long"},
{ERR_REASON(SSL_R_MISSING_DH_DSA_CERT)   ,"missing dh dsa cert"},
{ERR_REASON(SSL_R_MISSING_DH_KEY)        ,"missing dh key"},
{ERR_REASON(SSL_R_MISSING_DH_RSA_CERT)   ,"missing dh rsa cert"},
{ERR_REASON(SSL_R_MISSING_DSA_SIGNING_CERT),"missing dsa signing cert"},
{ERR_REASON(SSL_R_MISSING_EXPORT_TMP_DH_KEY),"missing export tmp dh key"},
{ERR_REASON(SSL_R_MISSING_EXPORT_TMP_RSA_KEY),"missing export tmp rsa key"},
{ERR_REASON(SSL_R_MISSING_RSA_CERTIFICATE),"missing rsa certificate"},
{ERR_REASON(SSL_R_MISSING_RSA_ENCRYPTING_CERT),"missing rsa encrypting cert"},
{ERR_REASON(SSL_R_MISSING_RSA_SIGNING_CERT),"missing rsa signing cert"},
{ERR_REASON(SSL_R_MISSING_SRP_PARAM)     ,"can't find SRP server param"},
{ERR_REASON(SSL_R_MISSING_TMP_DH_KEY)    ,"missing tmp dh key"},
{ERR_REASON(SSL_R_MISSING_TMP_ECDH_KEY)  ,"missing tmp ecdh key"},
{ERR_REASON(SSL_R_MISSING_TMP_RSA_KEY)   ,"missing tmp rsa key"},
{ERR_REASON(SSL_R_MISSING_TMP_RSA_PKEY)  ,"missing tmp rsa pkey"},
{ERR_REASON(SSL_R_MISSING_VERIFY_MESSAGE),"missing verify message"},
{ERR_REASON(SSL_R_MULTIPLE_SGC_RESTARTS) ,"multiple sgc restarts"},
{ERR_REASON(SSL_R_NON_SSLV2_INITIAL_PACKET),"non sslv2 initial packet"},
{ERR_REASON(SSL_R_NO_CERTIFICATES_RETURNED),"no certificates returned"},
{ERR_REASON(SSL_R_NO_CERTIFICATE_ASSIGNED),"no certificate assigned"},
{ERR_REASON(SSL_R_NO_CERTIFICATE_RETURNED),"no certificate returned"},
{ERR_REASON(SSL_R_NO_CERTIFICATE_SET)    ,"no certificate set"},
{ERR_REASON(SSL_R_NO_CERTIFICATE_SPECIFIED),"no certificate specified"},
{ERR_REASON(SSL_R_NO_CIPHERS_AVAILABLE)  ,"no ciphers available"},
{ERR_REASON(SSL_R_NO_CIPHERS_PASSED)     ,"no ciphers passed"},
{ERR_REASON(SSL_R_NO_CIPHERS_SPECIFIED)  ,"no ciphers specified"},
{ERR_REASON(SSL_R_NO_CIPHER_LIST)        ,"no cipher list"},
{ERR_REASON(SSL_R_NO_CIPHER_MATCH)       ,"no cipher match"},
{ERR_REASON(SSL_R_NO_CLIENT_CERT_METHOD) ,"no client cert method"},
{ERR_REASON(SSL_R_NO_CLIENT_CERT_RECEIVED),"no client cert received"},
{ERR_REASON(SSL_R_NO_COMPRESSION_SPECIFIED),"no compression specified"},
{ERR_REASON(SSL_R_NO_GOST_CERTIFICATE_SENT_BY_PEER),"Peer haven't sent GOST cert
{ERR_REASON(SSL_R_NO_METHOD_SPECIFIED)   ,"no method specified"},
{ERR_REASON(SSL_R_NO_PRIVATEKEY)         ,"no privatekey"},
{ERR_REASON(SSL_R_NO_PRIVATE_KEY_ASSIGNED),"no private key assigned"},
{ERR_REASON(SSL_R_NO_PROTOCOLS_AVAILABLE),"no protocols available"},
{ERR_REASON(SSL_R_NO_PUBLICKEY)          ,"no publickey"},
{ERR_REASON(SSL_R_NO_RENEGOTIATION)      ,"no renegotiation"},
{ERR_REASON(SSL_R_NO_REQUIRED_DIGEST)    ,"digest requred for handshake isn't co
{ERR_REASON(SSL_R_NO_SHARED_CIPHER)      ,"no shared cipher"},
{ERR_REASON(SSL_R_NO_SRTP_PROFILES)      ,"no srtp profiles"},
{ERR_REASON(SSL_R_NO_VERIFY_CALLBACK)    ,"no verify callback"},
{ERR_REASON(SSL_R_NULL_SSL_CTX)          ,"null ssl ctx"},
{ERR_REASON(SSL_R_NULL_SSL_METHOD_PASSED),"null ssl method passed"},
{ERR_REASON(SSL_R_OLD_SESSION_CIPHER_NOT_RETURNED),"old session cipher not retur
{ERR_REASON(SSL_R_OLD_SESSION_COMPRESSION_ALGORITHM_NOT_RETURNED),"old session c
{ERR_REASON(SSL_R_ONLY_TLS_ALLOWED_IN_FIPS_MODE),"only tls allowed in fips mode"
```



```
{ERR_REASON(SSL_R_OPAQUE_PRF_INPUT_TOO_LONG),"opaque PRF input too long"},
{ERR_REASON(SSL_R_PACKET_LENGTH_TOO_LONG),"packet length too long"},
{ERR_REASON(SSL_R_PARSE_TLSEXT)          ,"parse tlsext"},
{ERR_REASON(SSL_R_PATH_TOO_LONG)         ,"path too long"},
{ERR_REASON(SSL_R_PEER_DID_NOT_RETURN_A_CERTIFICATE),"peer did not return a cert
{ERR_REASON(SSL_R_PEER_ERROR)            ,"peer error"},
{ERR_REASON(SSL_R_PEER_ERROR_CERTIFICATE),"peer error certificate"},
{ERR_REASON(SSL_R_PEER_ERROR_NO_CERTIFICATE),"peer error no certificate"},
{ERR_REASON(SSL_R_PEER_ERROR_NO_CIPHER)  ,"peer error no cipher"},
{ERR_REASON(SSL_R_PEER_ERROR_UNSUPPORTED_CERTIFICATE_TYPE),"peer error unsupport
{ERR_REASON(SSL_R_PRE_MAC_LENGTH_TOO_LONG),"pre mac length too long"},
{ERR_REASON(SSL_R_PROBLEMS_MAPPING_CIPHER_FUNCTIONS),"problems mapping cipher fu
{ERR_REASON(SSL_R_PROTOCOL_IS_SHUTDOWN)  ,"protocol is shutdown"},
{ERR_REASON(SSL_R_PSK_IDENTITY_NOT_FOUND),"psk identity not found"},
{ERR_REASON(SSL_R_PSK_NO_CLIENT_CB)      ,"psk no client cb"},
{ERR_REASON(SSL_R_PSK_NO_SERVER_CB)      ,"psk no server cb"},
{ERR_REASON(SSL_R_PUBLIC_KEY_ENCRYPT_ERROR),"public key encrypt error"},
{ERR_REASON(SSL_R_PUBLIC_KEY_IS_NOT_RSA) ,"public key is not rsa"},
{ERR_REASON(SSL_R_PUBLIC_KEY_NOT_RSA)    ,"public key not rsa"},
{ERR_REASON(SSL_R_READ_BIO_NOT_SET)      ,"read bio not set"},
{ERR_REASON(SSL_R_READ_TIMEOUT_EXPIRED)  ,"read timeout expired"},
{ERR_REASON(SSL_R_READ_WRONG_PACKET_TYPE),"read wrong packet type"},
{ERR_REASON(SSL_R_RECORD_LENGTH_MISMATCH),"record length mismatch"},
{ERR_REASON(SSL_R_RECORD_TOO_LARGE)      ,"record too large"},
{ERR_REASON(SSL_R_RECORD_TOO_SMALL)      ,"record too small"},
{ERR_REASON(SSL_R_RENEGOTIATE_EXT_TOO_LONG),"renegotiate ext too long"},
{ERR_REASON(SSL_R_RENEGOTIATION_ENCODING_ERR),"renegotiation encoding err"},
{ERR_REASON(SSL_R_RENEGOTIATION_MISMATCH),"renegotiation mismatch"},
{ERR_REASON(SSL_R_REQUIRED_CIPHER_MISSING),"required cipher missing"},
{ERR_REASON(SSL_R_REQUIRED_COMPRESSSION_ALGORITHM_MISSING),"required compresssio
{ERR_REASON(SSL_R_REUSE_CERT_LENGTH_NOT_ZERO),"reuse cert length not zero"},
{ERR_REASON(SSL_R_REUSE_CERT_TYPE_NOT_ZERO),"reuse cert type not zero"},
{ERR_REASON(SSL_R_REUSE_CIPHER_LIST_NOT_ZERO),"reuse cipher list not zero"},
{ERR_REASON(SSL_R_SCSV_RECEIVED_WHEN_RENEGOTIATING),"scsv received when renegoti
{ERR_REASON(SSL_R_SERVERHELLO_TLSEXT)    ,"serverhello tlsext"},
{ERR_REASON(SSL_R_SESSION_ID_CONTEXT_UNINITIALIZED),"session id context uninitia
{ERR_REASON(SSL_R_SHORT_READ)            ,"short read"},
{ERR_REASON(SSL_R_SIGNATURE_ALGORITHMS_ERROR),"signature algorithms error"},
{ERR_REASON(SSL_R_SIGNATURE_FOR_NON_SIGNING_CERTIFICATE),"signature for non sign
{ERR_REASON(SSL_R_SRP_A_CALC)            ,"error with the srp params"},
{ERR_REASON(SSL_R_SRTP_COULD_NOT_ALLOCATE_PROFILES),"srtp could not allocate pro
{ERR_REASON(SSL_R_SRTP_PROTECTION_PROFILE_LIST_TOO_LONG),"srtp protection profil
{ERR_REASON(SSL_R_SRTP_UNKNOWN_PROTECTION_PROFILE),"srtp unknown protection prof
{ERR_REASON(SSL_R_SSL23_DOING_SESSION_ID_REUSE),"ssl23 doing session id reuse"},
{ERR_REASON(SSL_R_SSL2_CONNECTION_ID_TOO_LONG),"ssl2 connection id too long"},
{ERR_REASON(SSL_R_SSL3_EXT_INVALID_ECPOINTFORMAT),"ssl3 ext invalid ecpointforma
{ERR_REASON(SSL_R_SSL3_EXT_INVALID_SERVERNAME),"ssl3 ext invalid servername"},
{ERR_REASON(SSL_R_SSL3_EXT_INVALID_SERVERNAME_TYPE),"ssl3 ext invalid servername
{ERR_REASON(SSL_R_SSL3_SESSION_ID_TOO_LONG),"ssl3 session id too long"},
{ERR_REASON(SSL_R_SSL3_SESSION_ID_TOO_SHORT),"ssl3 session id too short"},
{ERR_REASON(SSL_R_SSLV3_ALERT_BAD_CERTIFICATE),"sslv3 alert bad certificate"},
{ERR_REASON(SSL_R_SSLV3_ALERT_BAD_RECORD_MAC),"sslv3 alert bad record mac"},
{ERR_REASON(SSL_R_SSLV3_ALERT_CERTIFICATE_EXPIRED),"sslv3 alert certificate expi
{ERR_REASON(SSL_R_SSLV3_ALERT_CERTIFICATE_REVOKED),"sslv3 alert certificate revo
{ERR_REASON(SSL_R_SSLV3_ALERT_CERTIFICATE_UNKNOWN),"sslv3 alert certificate unkn
{ERR_REASON(SSL_R_SSLV3_ALERT_DECOMPRESSION_FAILURE),"sslv3 alert decompression
{ERR_REASON(SSL_R_SSLV3_ALERT_HANDSHAKE_FAILURE),"sslv3 alert handshake failure"
{ERR_REASON(SSL_R_SSLV3_ALERT_ILLEGAL_PARAMETER),"sslv3 alert illegal parameter"
{ERR_REASON(SSL_R_SSLV3_ALERT_NO_CERTIFICATE),"sslv3 alert no certificate"},
{ERR_REASON(SSL_R_SSLV3_ALERT_UNEXPECTED_MESSAGE),"sslv3 alert unexpected messag
{ERR_REASON(SSL_R_SSLV3_ALERT_UNSUPPORTED_CERTIFICATE),"sslv3 alert unsupported
{ERR_REASON(SSL_R_SSL_CTX_HAS_NO_DEFAULT_SSL_VERSION),"ssl ctx has no default ss
{ERR_REASON(SSL_R_SSL_HANDSHAKE_FAILURE) ,"ssl handshake failure"},
{ERR_REASON(SSL_R_SSL_LIBRARY_HAS_NO_CIPHERS),"ssl library has no ciphers"},
{ERR_REASON(SSL_R_SSL_SESSION_ID_CALLBACK_FAILED),"ssl session id callback faile
{ERR_REASON(SSL_R_SSL_SESSION_ID_CONFLICT),"ssl session id conflict"},
```

```
{ERR_REASON(SSL_R_SSL_SESSION_ID_CONTEXT_TOO_LONG),"ssl session id context too l
{ERR_REASON(SSL_R_SSL_SESSION_ID_HAS_BAD_LENGTH),"ssl session id has bad length"
{ERR_REASON(SSL_R_SSL_SESSION_ID_IS_DIFFERENT),"ssl session id is different"},
{ERR_REASON(SSL_R_TLSV1_ALERT_ACCESS_DENIED),"tlsv1 alert access denied"},
{ERR_REASON(SSL_R_TLSV1_ALERT_DECODE_ERROR),"tlsv1 alert decode error"},
{ERR_REASON(SSL_R_TLSV1_ALERT_DECRYPTION_FAILED),"tlsv1 alert decryption failed"
{ERR_REASON(SSL_R_TLSV1_ALERT_DECRYPT_ERROR),"tlsv1 alert decrypt error"},
{ERR_REASON(SSL_R_TLSV1_ALERT_EXPORT_RESTRICTION),"tlsv1 alert export restrictio
{ERR_REASON(SSL_R_TLSV1_ALERT_INSUFFICIENT_SECURITY),"tlsv1 alert insufficient s
{ERR_REASON(SSL_R_TLSV1_ALERT_INTERNAL_ERROR),"tlsv1 alert internal error"},
{ERR_REASON(SSL_R_TLSV1_ALERT_NO_RENEGOTIATION),"tlsv1 alert no renegotiation"},
{ERR_REASON(SSL_R_TLSV1_ALERT_PROTOCOL_VERSION),"tlsv1 alert protocol version"},
{ERR_REASON(SSL_R_TLSV1_ALERT_RECORD_OVERFLOW),"tlsv1 alert record overflow"},
{ERR_REASON(SSL_R_TLSV1_ALERT_UNKNOWN_CA),"tlsv1 alert unknown ca"},
{ERR_REASON(SSL_R_TLSV1_ALERT_USER_CANCELLED),"tlsv1 alert user cancelled"},
{ERR_REASON(SSL_R_TLSV1_BAD_CERTIFICATE_HASH_VALUE),"tlsv1 bad certificate hash
{ERR_REASON(SSL_R_TLSV1_BAD_CERTIFICATE_STATUS_RESPONSE),"tlsv1 bad certificate
{ERR_REASON(SSL_R_TLSV1_CERTIFICATE_UNOBTAINABLE),"tlsv1 certificate unobtainabl
{ERR_REASON(SSL_R_TLSV1_UNRECOGNIZED_NAME),"tlsv1 unrecognized name"},
{ERR_REASON(SSL_R_TLSV1_UNSUPPORTED_EXTENSION),"tlsv1 unsupported extension"},
{ERR_REASON(SSL_R_TLS_CLIENT_CERT_REQ_WITH_ANON_CIPHER),"tls client cert req wit
{ERR_REASON(SSL_R_TLS_HEARTBEAT_PEER_DOESNT_ACCEPT),"peer does not accept heartb
{ERR_REASON(SSL_R_TLS_HEARTBEAT_PENDING) ,"heartbeat request already pending"},
{ERR_REASON(SSL_R_TLS_ILLEGAL_EXPORTER_LABEL),"tls illegal exporter label"},
{ERR_REASON(SSL_R_TLS_INVALID_ECPOINTFORMAT_LIST),"tls invalid ecpointformat lis
{ERR_REASON(SSL_R_TLS_PEER_DID_NOT_RESPOND_WITH_CERTIFICATE_LIST),"tls peer did
{ERR_REASON(SSL_R_TLS_RSA_ENCRYPTED_VALUE_LENGTH_IS_WRONG),"tls rsa encrypted va
{ERR_REASON(SSL_R_TRIED_TO_USE_UNSUPPORTED_CIPHER),"tried to use unsupported cip
{ERR_REASON(SSL_R_UNABLE_TO_DECODE_DH_CERTS),"unable to decode dh certs"},
{ERR_REASON(SSL_R_UNABLE_TO_DECODE_ECDH_CERTS),"unable to decode ecdh certs"},
{ERR_REASON(SSL_R_UNABLE_TO_EXTRACT_PUBLIC_KEY),"unable to extract public key"},
{ERR_REASON(SSL_R_UNABLE_TO_FIND_DH_PARAMETERS),"unable to find dh parameters"},
{ERR_REASON(SSL_R_UNABLE_TO_FIND_ECDH_PARAMETERS),"unable to find ecdh parameter
{ERR_REASON(SSL_R_UNABLE_TO_FIND_PUBLIC_KEY_PARAMETERS),"unable to find public k
{ERR_REASON(SSL_R_UNABLE_TO_FIND_SSL_METHOD),"unable to find ssl method"},
{ERR_REASON(SSL_R_UNABLE_TO_LOAD_SSL2_MD5_ROUTINES),"unable to load ssl2 md5 rou
{ERR_REASON(SSL_R_UNABLE_TO_LOAD_SSL3_MD5_ROUTINES),"unable to load ssl3 md5 rou
{ERR_REASON(SSL_R_UNABLE_TO_LOAD_SSL3_SHA1_ROUTINES),"unable to load ssl3 sha1 r
{ERR_REASON(SSL_R_UNEXPECTED_MESSAGE)    ,"unexpected message"},
{ERR_REASON(SSL_R_UNEXPECTED_RECORD)     ,"unexpected record"},
{ERR_REASON(SSL_R_UNINITIALIZED)         ,"uninitialized"},
{ERR_REASON(SSL_R_UNKNOWN_ALERT_TYPE)    ,"unknown alert type"},
{ERR_REASON(SSL_R_UNKNOWN_CERTIFICATE_TYPE),"unknown certificate type"},
{ERR_REASON(SSL_R_UNKNOWN_CIPHER_RETURNED),"unknown cipher returned"},
{ERR_REASON(SSL_R_UNKNOWN_CIPHER_TYPE)   ,"unknown cipher type"},
{ERR_REASON(SSL_R_UNKNOWN_DIGEST)        ,"unknown digest"},
{ERR_REASON(SSL_R_UNKNOWN_KEY_EXCHANGE_TYPE),"unknown key exchange type"},
{ERR_REASON(SSL_R_UNKNOWN_PKEY_TYPE)     ,"unknown pkey type"},
{ERR_REASON(SSL_R_UNKNOWN_PROTOCOL)      ,"unknown protocol"},
{ERR_REASON(SSL_R_UNKNOWN_REMOTE_ERROR_TYPE),"unknown remote error type"},
{ERR_REASON(SSL_R_UNKNOWN_SSL_VERSION)   ,"unknown ssl version"},
{ERR_REASON(SSL_R_UNKNOWN_STATE)         ,"unknown state"},
{ERR_REASON(SSL_R_UNSAFE_LEGACY_RENEGOTIATION_DISABLED),"unsafe legacy renegotia
{ERR_REASON(SSL_R_UNSUPPORTED_CIPHER)    ,"unsupported cipher"},
{ERR_REASON(SSL_R_UNSUPPORTED_COMPRESSION_ALGORITHM),"unsupported compression al
{ERR_REASON(SSL_R_UNSUPPORTED_DIGEST_TYPE),"unsupported digest type"},
{ERR_REASON(SSL_R_UNSUPPORTED_ELLIPTIC_CURVE),"unsupported elliptic curve"},
{ERR_REASON(SSL_R_UNSUPPORTED_PROTOCOL)  ,"unsupported protocol"},
{ERR_REASON(SSL_R_UNSUPPORTED_SSL_VERSION),"unsupported ssl version"},
{ERR_REASON(SSL_R_UNSUPPORTED_STATUS_TYPE),"unsupported status type"},
{ERR_REASON(SSL_R_USE_SRTP_NOT_NEGOTIATED),"use srtp not negotiated"},
{ERR_REASON(SSL_R_WRITE_BIO_NOT_SET)     ,"write bio not set"},
{ERR_REASON(SSL_R_WRONG_CIPHER_RETURNED) ,"wrong cipher returned"},
{ERR_REASON(SSL_R_WRONG_MESSAGE_TYPE)    ,"wrong message type"},
{ERR_REASON(SSL_R_WRONG_NUMBER_OF_KEY_BITS),"wrong number of key bits"},
{ERR_REASON(SSL_R_WRONG_SIGNATURE_LENGTH),"wrong signature length"},
```



```
{ERR_REASON(SSL_R_WRONG_SIGNATURE_SIZE)  ,"wrong signature size"},
{ERR_REASON(SSL_R_WRONG_SIGNATURE_TYPE)  ,"wrong signature type"},
{ERR_REASON(SSL_R_WRONG_SSL_VERSION)     ,"wrong ssl version"},
{ERR_REASON(SSL_R_WRONG_VERSION_NUMBER)  ,"wrong version number"},
{ERR_REASON(SSL_R_X509_LIB)              ,"x509 lib"},
{ERR_REASON(SSL_R_X509_VERIFICATION_SETUP_PROBLEMS),"x509 verification setup pro
{0,NULL}
        };

#endif

void ERR_load_SSL_strings(void)
        {
#ifndef OPENSSL_NO_ERR

        if (ERR_func_error_string(SSL_str_functs[0].error) == NULL)
                {
                ERR_load_strings(0,SSL_str_functs);
                ERR_load_strings(0,SSL_str_reasons);
                }
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
    3378 Wed Aug 13 19:53:40 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_err2.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_err2.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/err.h>
#include <openssl/ssl.h>
```



```
void SSL_load_error_strings(void)
        {
#ifndef OPENSSL_NO_ERR
        ERR_load_crypto_strings();
        ERR_load_SSL_strings();
#endif
        }
#endif /* ! codereview */
```

```
**********************************************************
   83199 Wed Aug 13 19:53:40 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/*! \file ssl/ssl_lib.c
 *  \brief Version independent SSL functions.
 */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
```



```
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2002 Sun Microsystems, Inc. ALL RIGHTS RESERVED.
 * ECC cipher suite support in OpenSSL originally developed by
 * SUN MICROSYSTEMS, INC., and contributed to the OpenSSL project.
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
```

```
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#ifdef REF_CHECK
#  include <assert.h>
#endif
#include <stdio.h>
#include "ssl_locl.h"
#include "kssl_lcl.h"
#include <openssl/objects.h>
#include <openssl/lhash.h>
#include <openssl/x509v3.h>
#include <openssl/rand.h>
#include <openssl/ocsp.h>
#ifndef OPENSSL_NO_DH
#include <openssl/dh.h>
#endif
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif

const char *SSL_version_str=OPENSSL_VERSION_TEXT;

SSL3_ENC_METHOD ssl3_undef_enc_method={
        /* evil casts, but these functions are only called if there's a library
        (int (*)(SSL *,int))ssl_undefined_function,
        (int (*)(SSL *, unsigned char *, int))ssl_undefined_function,
        ssl_undefined_function,
        (int (*)(SSL *, unsigned char *, unsigned char *, int))ssl_undefined_fun
        (int (*)(SSL*, int))ssl_undefined_function,
        (int (*)(SSL *,  const char*, int, unsigned char *))ssl_undefined_functi
        0,      /* finish_mac_length */
        (int (*)(SSL *, int, unsigned char *))ssl_undefined_function,
        NULL,   /* client_finished_label */
        0,      /* client_finished_label_len */
        NULL,   /* server_finished_label */
        0,      /* server_finished_label_len */
        (int (*)(int))ssl_undefined_function,
        (int (*)(SSL *, unsigned char *, size_t, const char *,
                 size_t, const unsigned char *, size_t,
                 int use_context)) ssl_undefined_function,
        };

int SSL_clear(SSL *s)
        {

        if (s->method == NULL)
                {
                SSLerr(SSL_F_SSL_CLEAR,SSL_R_NO_METHOD_SPECIFIED);
                return(0);
                }
```



```
        if (ssl_clear_bad_session(s))
                {
                SSL_SESSION_free(s->session);
                s->session=NULL;
                }

        s->error=0;
        s->hit=0;
        s->shutdown=0;

#if 0 /* Disabled since version 1.10 of this file (early return not
       * needed because SSL_clear is not called when doing renegotiation) */
        /* This is set if we are doing dynamic renegotiation so keep
         * the old cipher.  It is sort of a SSL_clear_lite :-) */
        if (s->renegotiate) return(1);
#else
        if (s->renegotiate)
                {
                SSLerr(SSL_F_SSL_CLEAR,ERR_R_INTERNAL_ERROR);
                return 0;
                }
#endif

        s->type=0;

        s->state=SSL_ST_BEFORE|((s->server)?SSL_ST_ACCEPT:SSL_ST_CONNECT);

        s->version=s->method->version;
        s->client_version=s->version;
        s->rwstate=SSL_NOTHING;
        s->rstate=SSL_ST_READ_HEADER;
#if 0
        s->read_ahead=s->ctx->read_ahead;
#endif

        if (s->init_buf != NULL)
                {
                BUF_MEM_free(s->init_buf);
                s->init_buf=NULL;
                }

        ssl_clear_cipher_ctx(s);
        ssl_clear_hash_ctx(&s->read_hash);
        ssl_clear_hash_ctx(&s->write_hash);

        s->first_packet=0;

#if 1
        /* Check to see if we were changed into a different method, if
         * so, revert back if we are not doing session-id reuse. */
        if (!s->in_handshake && (s->session == NULL) && (s->method != s->ctx->me
                {
                s->method->ssl_free(s);
                s->method=s->ctx->method;
                if (!s->method->ssl_new(s))
                        return(0);
                }
        else
#endif
                s->method->ssl_clear(s);
        return(1);
        }

/** Used to change an SSL_CTXs default SSL method type */
int SSL_CTX_set_ssl_version(SSL_CTX *ctx,const SSL_METHOD *meth)
        {
```

```
260         STACK_OF(SSL_CIPHER) *sk;

262         ctx->method=meth;

264         sk=ssl_create_cipher_list(ctx->method,&(ctx->cipher_list),
265                 &(ctx->cipher_list_by_id),
266                 meth->version == SSL2_VERSION ? "SSLv2" : SSL_DEFAULT_CIPHER_LIS
267         if ((sk == NULL) || (sk_SSL_CIPHER_num(sk) <= 0))
268                 {
269                 SSLerr(SSL_F_SSL_CTX_SET_SSL_VERSION,SSL_R_SSL_LIBRARY_HAS_NO_CI
270                 return(0);
271                 }
272         return(1);
273         }

275 SSL *SSL_new(SSL_CTX *ctx)
276         {
277         SSL *s;

279         if (ctx == NULL)
280                 {
281                 SSLerr(SSL_F_SSL_NEW,SSL_R_NULL_SSL_CTX);
282                 return(NULL);
283                 }
284         if (ctx->method == NULL)
285                 {
286                 SSLerr(SSL_F_SSL_NEW,SSL_R_SSL_CTX_HAS_NO_DEFAULT_SSL_VERSION);
287                 return(NULL);
288                 }

290         s=(SSL *)OPENSSL_malloc(sizeof(SSL));
291         if (s == NULL) goto err;
292         memset(s,0,sizeof(SSL));

294 #ifndef OPENSSL_NO_KRB5
295         s->kssl_ctx = kssl_ctx_new();
296 #endif  /* OPENSSL_NO_KRB5 */

298         s->options=ctx->options;
299         s->mode=ctx->mode;
300         s->max_cert_list=ctx->max_cert_list;

302         if (ctx->cert != NULL)
303                 {
304                 /* Earlier library versions used to copy the pointer to
305                  * the CERT, not its contents; only when setting new
306                  * parameters for the per-SSL copy, ssl_cert_new would be
307                  * called (and the direct reference to the per-SSL_CTX
308                  * settings would be lost, but those still were indirectly
309                  * accessed for various purposes, and for that reason they
310                  * used to be known as s->ctx->default_cert).
311                  * Now we don't look at the SSL_CTX's CERT after having
312                  * duplicated it once. */

314                 s->cert = ssl_cert_dup(ctx->cert);
315                 if (s->cert == NULL)
316                         goto err;
317                 }
318         else
319                 s->cert=NULL; /* Cannot really happen (see SSL_CTX_new) */

321         s->read_ahead=ctx->read_ahead;
322         s->msg_callback=ctx->msg_callback;
323         s->msg_callback_arg=ctx->msg_callback_arg;
324         s->verify_mode=ctx->verify_mode;
325 #if 0
```



```
326         s->verify_depth=ctx->verify_depth;
327 #endif
328         s->sid_ctx_length=ctx->sid_ctx_length;
329         OPENSSL_assert(s->sid_ctx_length <= sizeof s->sid_ctx);
330         memcpy(&s->sid_ctx,&ctx->sid_ctx,sizeof(s->sid_ctx));
331         s->verify_callback=ctx->default_verify_callback;
332         s->generate_session_id=ctx->generate_session_id;

334         s->param = X509_VERIFY_PARAM_new();
335         if (!s->param)
336                 goto err;
337         X509_VERIFY_PARAM_inherit(s->param, ctx->param);
338 #if 0
339         s->purpose = ctx->purpose;
340         s->trust = ctx->trust;
341 #endif
342         s->quiet_shutdown=ctx->quiet_shutdown;
343         s->max_send_fragment = ctx->max_send_fragment;

345         CRYPTO_add(&ctx->references,1,CRYPTO_LOCK_SSL_CTX);
346         s->ctx=ctx;
347 #ifndef OPENSSL_NO_TLSEXT
348         s->tlsext_debug_cb = 0;
349         s->tlsext_debug_arg = NULL;
350         s->tlsext_ticket_expected = 0;
351         s->tlsext_status_type = -1;
352         s->tlsext_status_expected = 0;
353         s->tlsext_ocsp_ids = NULL;
354         s->tlsext_ocsp_exts = NULL;
355         s->tlsext_ocsp_resp = NULL;
356         s->tlsext_ocsp_resplen = -1;
357         CRYPTO_add(&ctx->references,1,CRYPTO_LOCK_SSL_CTX);
358         s->initial_ctx=ctx;
359 # ifndef OPENSSL_NO_NEXTPROTONEG
360         s->next_proto_negotiated = NULL;
361 # endif
362 #endif

364         s->verify_result=X509_V_OK;

366         s->method=ctx->method;

368         if (!s->method->ssl_new(s))
369                 goto err;

371         s->references=1;
372         s->server=(ctx->method->ssl_accept == ssl_undefined_function)?0:1;

374         SSL_clear(s);

376         CRYPTO_new_ex_data(CRYPTO_EX_INDEX_SSL, s, &s->ex_data);

378 #ifndef OPENSSL_NO_PSK
379         s->psk_client_callback=ctx->psk_client_callback;
380         s->psk_server_callback=ctx->psk_server_callback;
381 #endif

383         return(s);
384 err:
385         if (s != NULL)
386                 {
387                 if (s->cert != NULL)
388                         ssl_cert_free(s->cert);
389                 if (s->ctx != NULL)
390                         SSL_CTX_free(s->ctx); /* decrement reference count */
391                 OPENSSL_free(s);
```

```
392                 }
393         SSLerr(SSL_F_SSL_NEW,ERR_R_MALLOC_FAILURE);
394         return(NULL);
395         }

397 int SSL_CTX_set_session_id_context(SSL_CTX *ctx,const unsigned char *sid_ctx,
398                                    unsigned int sid_ctx_len)
399     {
400     if(sid_ctx_len > sizeof ctx->sid_ctx)
401         {
402         SSLerr(SSL_F_SSL_CTX_SET_SESSION_ID_CONTEXT,SSL_R_SSL_SESSION_ID_CONTEXT
403         return 0;
404         }
405     ctx->sid_ctx_length=sid_ctx_len;
406     memcpy(ctx->sid_ctx,sid_ctx,sid_ctx_len);

408     return 1;
409     }

411 int SSL_set_session_id_context(SSL *ssl,const unsigned char *sid_ctx,
412                                unsigned int sid_ctx_len)
413     {
414     if(sid_ctx_len > SSL_MAX_SID_CTX_LENGTH)
415         {
416         SSLerr(SSL_F_SSL_SET_SESSION_ID_CONTEXT,SSL_R_SSL_SESSION_ID_CONTEXT_TOO
417         return 0;
418         }
419     ssl->sid_ctx_length=sid_ctx_len;
420     memcpy(ssl->sid_ctx,sid_ctx,sid_ctx_len);

422     return 1;
423     }

425 int SSL_CTX_set_generate_session_id(SSL_CTX *ctx, GEN_SESSION_CB cb)
426         {
427         CRYPTO_w_lock(CRYPTO_LOCK_SSL_CTX);
428         ctx->generate_session_id = cb;
429         CRYPTO_w_unlock(CRYPTO_LOCK_SSL_CTX);
430         return 1;
431         }

433 int SSL_set_generate_session_id(SSL *ssl, GEN_SESSION_CB cb)
434         {
435         CRYPTO_w_lock(CRYPTO_LOCK_SSL);
436         ssl->generate_session_id = cb;
437         CRYPTO_w_unlock(CRYPTO_LOCK_SSL);
438         return 1;
439         }

441 int SSL_has_matching_session_id(const SSL *ssl, const unsigned char *id,
442                                 unsigned int id_len)
443         {
444         /* A quick examination of SSL_SESSION_hash and SSL_SESSION_cmp shows how
445          * we can "construct" a session to give us the desired check - ie. to
446          * find if there's a session in the hash table that would conflict with
447          * any new session built out of this id/id_len and the ssl_version in
448          * use by this SSL. */
449         SSL_SESSION r, *p;

451         if(id_len > sizeof r.session_id)
452                 return 0;

454         r.ssl_version = ssl->version;
455         r.session_id_length = id_len;
456         memcpy(r.session_id, id, id_len);
457         /* NB: SSLv2 always uses a fixed 16-byte session ID, so even if a
```



```
458          * callback is calling us to check the uniqueness of a shorter ID, it
459          * must be compared as a padded-out ID because that is what it will be
460          * converted to when the callback has finished choosing it. */
461         if((r.ssl_version == SSL2_VERSION) &&
462                         (id_len < SSL2_SSL_SESSION_ID_LENGTH))
463                 {
464                 memset(r.session_id + id_len, 0,
465                         SSL2_SSL_SESSION_ID_LENGTH - id_len);
466                 r.session_id_length = SSL2_SSL_SESSION_ID_LENGTH;
467                 }

469         CRYPTO_r_lock(CRYPTO_LOCK_SSL_CTX);
470         p = lh_SSL_SESSION_retrieve(ssl->ctx->sessions, &r);
471         CRYPTO_r_unlock(CRYPTO_LOCK_SSL_CTX);
472         return (p != NULL);
473         }

475 int SSL_CTX_set_purpose(SSL_CTX *s, int purpose)
476         {
477         return X509_VERIFY_PARAM_set_purpose(s->param, purpose);
478         }

480 int SSL_set_purpose(SSL *s, int purpose)
481         {
482         return X509_VERIFY_PARAM_set_purpose(s->param, purpose);
483         }

485 int SSL_CTX_set_trust(SSL_CTX *s, int trust)
486         {
487         return X509_VERIFY_PARAM_set_trust(s->param, trust);
488         }

490 int SSL_set_trust(SSL *s, int trust)
491         {
492         return X509_VERIFY_PARAM_set_trust(s->param, trust);
493         }

495 int SSL_CTX_set1_param(SSL_CTX *ctx, X509_VERIFY_PARAM *vpm)
496         {
497         return X509_VERIFY_PARAM_set1(ctx->param, vpm);
498         }

500 int SSL_set1_param(SSL *ssl, X509_VERIFY_PARAM *vpm)
501         {
502         return X509_VERIFY_PARAM_set1(ssl->param, vpm);
503         }

505 void SSL_free(SSL *s)
506         {
507         int i;

509         if(s == NULL)
510             return;

512         i=CRYPTO_add(&s->references,-1,CRYPTO_LOCK_SSL);
513 #ifdef REF_PRINT
514         REF_PRINT("SSL",s);
515 #endif
516         if (i > 0) return;
517 #ifdef REF_CHECK
518         if (i < 0)
519                 {
520                 fprintf(stderr,"SSL_free, bad reference count\n");
521                 abort(); /* ok */
522                 }
523 #endif
```

```
525         if (s->param)
526                 X509_VERIFY_PARAM_free(s->param);

528         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_SSL, s, &s->ex_data);

530         if (s->bbio != NULL)
531                 {
532                 /* If the buffering BIO is in place, pop it off */
533                 if (s->bbio == s->wbio)
534                         {
535                         s->wbio=BIO_pop(s->wbio);
536                         }
537                 BIO_free(s->bbio);
538                 s->bbio=NULL;
539                 }
540         if (s->rbio != NULL)
541                 BIO_free_all(s->rbio);
542         if ((s->wbio != NULL) && (s->wbio != s->rbio))
543                 BIO_free_all(s->wbio);

545         if (s->init_buf != NULL) BUF_MEM_free(s->init_buf);

547         /* add extra stuff */
548         if (s->cipher_list != NULL) sk_SSL_CIPHER_free(s->cipher_list);
549         if (s->cipher_list_by_id != NULL) sk_SSL_CIPHER_free(s->cipher_list_by_i

551         /* Make the next call work :-) */
552         if (s->session != NULL)
553                 {
554                 ssl_clear_bad_session(s);
555                 SSL_SESSION_free(s->session);
556                 }

558         ssl_clear_cipher_ctx(s);
559         ssl_clear_hash_ctx(&s->read_hash);
560         ssl_clear_hash_ctx(&s->write_hash);

562         if (s->cert != NULL) ssl_cert_free(s->cert);
563         /* Free up if allocated */

565 #ifndef OPENSSL_NO_TLSEXT
566         if (s->tlsext_hostname)
567                 OPENSSL_free(s->tlsext_hostname);
568         if (s->initial_ctx) SSL_CTX_free(s->initial_ctx);
569 #ifndef OPENSSL_NO_EC
570         if (s->tlsext_ecpointformatlist) OPENSSL_free(s->tlsext_ecpointformatlis
571         if (s->tlsext_ellipticcurvelist) OPENSSL_free(s->tlsext_ellipticcurvelis
572 #endif /* OPENSSL_NO_EC */
573         if (s->tlsext_opaque_prf_input) OPENSSL_free(s->tlsext_opaque_prf_input)
574         if (s->tlsext_ocsp_exts)
575                 sk_X509_EXTENSION_pop_free(s->tlsext_ocsp_exts,
576                                            X509_EXTENSION_free);
577         if (s->tlsext_ocsp_ids)
578                 sk_OCSP_RESPID_pop_free(s->tlsext_ocsp_ids, OCSP_RESPID_free);
579         if (s->tlsext_ocsp_resp)
580                 OPENSSL_free(s->tlsext_ocsp_resp);
581 #endif

583         if (s->client_CA != NULL)
584                 sk_X509_NAME_pop_free(s->client_CA,X509_NAME_free);

586         if (s->method != NULL) s->method->ssl_free(s);

588         if (s->ctx) SSL_CTX_free(s->ctx);
```



```
590 #ifndef OPENSSL_NO_KRB5
591         if (s->kssl_ctx != NULL)
592                 kssl_ctx_free(s->kssl_ctx);
593 #endif  /* OPENSSL_NO_KRB5 */

595 #if !defined(OPENSSL_NO_TLSEXT) && !defined(OPENSSL_NO_NEXTPROTONEG)
596         if (s->next_proto_negotiated)
597                 OPENSSL_free(s->next_proto_negotiated);
598 #endif

600 #ifndef OPENSSL_NO_SRTP
601         if (s->srtp_profiles)
602             sk_SRTP_PROTECTION_PROFILE_free(s->srtp_profiles);
603 #endif

605         OPENSSL_free(s);
606         }

608 void SSL_set_bio(SSL *s,BIO *rbio,BIO *wbio)
609         {
610         /* If the output buffering BIO is still in place, remove it
611          */
612         if (s->bbio != NULL)
613                 {
614                 if (s->wbio == s->bbio)
615                         {
616                         s->wbio=s->wbio->next_bio;
617                         s->bbio->next_bio=NULL;
618                         }
619                 }
620         if ((s->rbio != NULL) && (s->rbio != rbio))
621                 BIO_free_all(s->rbio);
622         if ((s->wbio != NULL) && (s->wbio != wbio) && (s->rbio != s->wbio))
623                 BIO_free_all(s->wbio);
624         s->rbio=rbio;
625         s->wbio=wbio;
626         }

628 BIO *SSL_get_rbio(const SSL *s)
629         { return(s->rbio); }

631 BIO *SSL_get_wbio(const SSL *s)
632         { return(s->wbio); }

634 int SSL_get_fd(const SSL *s)
635         {
636         return(SSL_get_rfd(s));
637         }

639 int SSL_get_rfd(const SSL *s)
640         {
641         int ret= -1;
642         BIO *b,*r;

644         b=SSL_get_rbio(s);
645         r=BIO_find_type(b,BIO_TYPE_DESCRIPTOR);
646         if (r != NULL)
647                 BIO_get_fd(r,&ret);
648         return(ret);
649         }

651 int SSL_get_wfd(const SSL *s)
652         {
653         int ret= -1;
654         BIO *b,*r;
```

```
656         b=SSL_get_wbio(s);
657         r=BIO_find_type(b,BIO_TYPE_DESCRIPTOR);
658         if (r != NULL)
659                 BIO_get_fd(r,&ret);
660         return(ret);
661         }

663 #ifndef OPENSSL_NO_SOCK
664 int SSL_set_fd(SSL *s,int fd)
665         {
666         int ret=0;
667         BIO *bio=NULL;

669         bio=BIO_new(BIO_s_socket());

671         if (bio == NULL)
672                 {
673                 SSLerr(SSL_F_SSL_SET_FD,ERR_R_BUF_LIB);
674                 goto err;
675                 }
676         BIO_set_fd(bio,fd,BIO_NOCLOSE);
677         SSL_set_bio(s,bio,bio);
678         ret=1;
679 err:
680         return(ret);
681         }

683 int SSL_set_wfd(SSL *s,int fd)
684         {
685         int ret=0;
686         BIO *bio=NULL;

688         if ((s->rbio == NULL) || (BIO_method_type(s->rbio) != BIO_TYPE_SOCKET)
689                 || ((int)BIO_get_fd(s->rbio,NULL) != fd))
690                 {
691                 bio=BIO_new(BIO_s_socket());

693                 if (bio == NULL)
694                         { SSLerr(SSL_F_SSL_SET_WFD,ERR_R_BUF_LIB); goto err; }
695                 BIO_set_fd(bio,fd,BIO_NOCLOSE);
696                 SSL_set_bio(s,SSL_get_rbio(s),bio);
697                 }
698         else
699                 SSL_set_bio(s,SSL_get_rbio(s),SSL_get_rbio(s));
700         ret=1;
701 err:
702         return(ret);
703         }

705 int SSL_set_rfd(SSL *s,int fd)
706         {
707         int ret=0;
708         BIO *bio=NULL;

710         if ((s->wbio == NULL) || (BIO_method_type(s->wbio) != BIO_TYPE_SOCKET)
711                 || ((int)BIO_get_fd(s->wbio,NULL) != fd))
712                 {
713                 bio=BIO_new(BIO_s_socket());

715                 if (bio == NULL)
716                         {
717                         SSLerr(SSL_F_SSL_SET_RFD,ERR_R_BUF_LIB);
718                         goto err;
719                         }
720                 BIO_set_fd(bio,fd,BIO_NOCLOSE);
721                 SSL_set_bio(s,bio,SSL_get_wbio(s));
```



```
722                 }
723         else
724                 SSL_set_bio(s,SSL_get_wbio(s),SSL_get_wbio(s));
725         ret=1;
726 err:
727         return(ret);
728         }
729 #endif


732 /* return length of latest Finished message we sent, copy to 'buf' */
733 size_t SSL_get_finished(const SSL *s, void *buf, size_t count)
734         {
735         size_t ret = 0;

737         if (s->s3 != NULL)
738                 {
739                 ret = s->s3->tmp.finish_md_len;
740                 if (count > ret)
741                         count = ret;
742                 memcpy(buf, s->s3->tmp.finish_md, count);
743                 }
744         return ret;
745         }

747 /* return length of latest Finished message we expected, copy to 'buf' */
748 size_t SSL_get_peer_finished(const SSL *s, void *buf, size_t count)
749         {
750         size_t ret = 0;

752         if (s->s3 != NULL)
753                 {
754                 ret = s->s3->tmp.peer_finish_md_len;
755                 if (count > ret)
756                         count = ret;
757                 memcpy(buf, s->s3->tmp.peer_finish_md, count);
758                 }
759         return ret;
760         }


763 int SSL_get_verify_mode(const SSL *s)
764         {
765         return(s->verify_mode);
766         }

768 int SSL_get_verify_depth(const SSL *s)
769         {
770         return X509_VERIFY_PARAM_get_depth(s->param);
771         }

773 int (*SSL_get_verify_callback(const SSL *s))(int,X509_STORE_CTX *)
774         {
775         return(s->verify_callback);
776         }

778 int SSL_CTX_get_verify_mode(const SSL_CTX *ctx)
779         {
780         return(ctx->verify_mode);
781         }

783 int SSL_CTX_get_verify_depth(const SSL_CTX *ctx)
784         {
785         return X509_VERIFY_PARAM_get_depth(ctx->param);
786         }
```

```
int (*SSL_CTX_get_verify_callback(const SSL_CTX *ctx))(int,X509_STORE_CTX *)
        {
        return(ctx->default_verify_callback);
        }

void SSL_set_verify(SSL *s,int mode,
                    int (*callback)(int ok,X509_STORE_CTX *ctx))
        {
        s->verify_mode=mode;
        if (callback != NULL)
                s->verify_callback=callback;
        }

void SSL_set_verify_depth(SSL *s,int depth)
        {
        X509_VERIFY_PARAM_set_depth(s->param, depth);
        }

void SSL_set_read_ahead(SSL *s,int yes)
        {
        s->read_ahead=yes;
        }

int SSL_get_read_ahead(const SSL *s)
        {
        return(s->read_ahead);
        }

int SSL_pending(const SSL *s)
        {
        /* SSL_pending cannot work properly if read-ahead is enabled
         * (SSL_[CTX_]ctrl(..., SSL_CTRL_SET_READ_AHEAD, 1, NULL)),
         * and it is impossible to fix since SSL_pending cannot report
         * errors that may be observed while scanning the new data.
         * (Note that SSL_pending() is often used as a boolean value,
         * so we'd better not return -1.)
         */
        return(s->method->ssl_pending(s));
        }

X509 *SSL_get_peer_certificate(const SSL *s)
        {
        X509 *r;

        if ((s == NULL) || (s->session == NULL))
                r=NULL;
        else
                r=s->session->peer;

        if (r == NULL) return(r);

        CRYPTO_add(&r->references,1,CRYPTO_LOCK_X509);

        return(r);
        }

STACK_OF(X509) *SSL_get_peer_cert_chain(const SSL *s)
        {
        STACK_OF(X509) *r;

        if ((s == NULL) || (s->session == NULL) || (s->session->sess_cert == NUL
                r=NULL;
        else
                r=s->session->sess_cert->cert_chain;

        /* If we are a client, cert_chain includes the peer's own
```



```
         * certificate; if we are a server, it does not. */

        return(r);
        }

/* Now in theory, since the calling process own 't' it should be safe to
 * modify.  We need to be able to read f without being hassled */
void SSL_copy_session_id(SSL *t,const SSL *f)
        {
        CERT *tmp;

        /* Do we need to to SSL locking? */
        SSL_set_session(t,SSL_get_session(f));

        /* what if we are setup as SSLv2 but want to talk SSLv3 or
         * vice-versa */
        if (t->method != f->method)
                {
                t->method->ssl_free(t); /* cleanup current */
                t->method=f->method;    /* change method */
                t->method->ssl_new(t);  /* setup new */
                }

        tmp=t->cert;
        if (f->cert != NULL)
                {
                CRYPTO_add(&f->cert->references,1,CRYPTO_LOCK_SSL_CERT);
                t->cert=f->cert;
                }
        else
                t->cert=NULL;
        if (tmp != NULL) ssl_cert_free(tmp);
        SSL_set_session_id_context(t,f->sid_ctx,f->sid_ctx_length);
        }

/* Fix this so it checks all the valid key/cert options */
int SSL_CTX_check_private_key(const SSL_CTX *ctx)
        {
        if (    (ctx == NULL) ||
                (ctx->cert == NULL) ||
                (ctx->cert->key->x509 == NULL))
                {
                SSLerr(SSL_F_SSL_CTX_CHECK_PRIVATE_KEY,SSL_R_NO_CERTIFICATE_ASSI
                return(0);
                }
        if      (ctx->cert->key->privatekey == NULL)
                {
                SSLerr(SSL_F_SSL_CTX_CHECK_PRIVATE_KEY,SSL_R_NO_PRIVATE_KEY_ASSI
                return(0);
                }
        return(X509_check_private_key(ctx->cert->key->x509, ctx->cert->key->priv
        }

/* Fix this function so that it takes an optional type parameter */
int SSL_check_private_key(const SSL *ssl)
        {
        if (ssl == NULL)
                {
                SSLerr(SSL_F_SSL_CHECK_PRIVATE_KEY,ERR_R_PASSED_NULL_PARAMETER);
                return(0);
                }
        if (ssl->cert == NULL)
                {
                SSLerr(SSL_F_SSL_CHECK_PRIVATE_KEY,SSL_R_NO_CERTIFICATE_ASSIGNED
                return 0;
                }
```

```
920         if (ssl->cert->key->x509 == NULL)
921                 {
922                 SSLerr(SSL_F_SSL_CHECK_PRIVATE_KEY,SSL_R_NO_CERTIFICATE_ASSIGNED
923                 return(0);
924                 }
925         if (ssl->cert->key->privatekey == NULL)
926                 {
927                 SSLerr(SSL_F_SSL_CHECK_PRIVATE_KEY,SSL_R_NO_PRIVATE_KEY_ASSIGNED
928                 return(0);
929                 }
930         return(X509_check_private_key(ssl->cert->key->x509,
931                 ssl->cert->key->privatekey));
932         }

934 int SSL_accept(SSL *s)
935         {
936         if (s->handshake_func == 0)
937                 /* Not properly initialized yet */
938                 SSL_set_accept_state(s);

940         return(s->method->ssl_accept(s));
941         }

943 int SSL_connect(SSL *s)
944         {
945         if (s->handshake_func == 0)
946                 /* Not properly initialized yet */
947                 SSL_set_connect_state(s);

949         return(s->method->ssl_connect(s));
950         }

952 long SSL_get_default_timeout(const SSL *s)
953         {
954         return(s->method->get_timeout());
955         }

957 int SSL_read(SSL *s,void *buf,int num)
958         {
959         if (s->handshake_func == 0)
960                 {
961                 SSLerr(SSL_F_SSL_READ, SSL_R_UNINITIALIZED);
962                 return -1;
963                 }

965         if (s->shutdown & SSL_RECEIVED_SHUTDOWN)
966                 {
967                 s->rwstate=SSL_NOTHING;
968                 return(0);
969                 }
970         return(s->method->ssl_read(s,buf,num));
971         }

973 int SSL_peek(SSL *s,void *buf,int num)
974         {
975         if (s->handshake_func == 0)
976                 {
977                 SSLerr(SSL_F_SSL_PEEK, SSL_R_UNINITIALIZED);
978                 return -1;
979                 }

981         if (s->shutdown & SSL_RECEIVED_SHUTDOWN)
982                 {
983                 return(0);
984                 }
985         return(s->method->ssl_peek(s,buf,num));
```



```
986         }

988 int SSL_write(SSL *s,const void *buf,int num)
989         {
990         if (s->handshake_func == 0)
991                 {
992                 SSLerr(SSL_F_SSL_WRITE, SSL_R_UNINITIALIZED);
993                 return -1;
994                 }

996         if (s->shutdown & SSL_SENT_SHUTDOWN)
997                 {
998                 s->rwstate=SSL_NOTHING;
999                 SSLerr(SSL_F_SSL_WRITE,SSL_R_PROTOCOL_IS_SHUTDOWN);
1000                return(-1);
1001                }
1002        return(s->method->ssl_write(s,buf,num));
1003        }

1005 int SSL_shutdown(SSL *s)
1006        {
1007        /* Note that this function behaves differently from what one might
1008         * expect.  Return values are 0 for no success (yet),
1009         * 1 for success; but calling it once is usually not enough,
1010         * even if blocking I/O is used (see ssl3_shutdown).
1011         */

1013        if (s->handshake_func == 0)
1014                {
1015                SSLerr(SSL_F_SSL_SHUTDOWN, SSL_R_UNINITIALIZED);
1016                return -1;
1017                }

1019        if ((s != NULL) && !SSL_in_init(s))
1020                return(s->method->ssl_shutdown(s));
1021        else
1022                return(1);
1023        }

1025 int SSL_renegotiate(SSL *s)
1026        {
1027        if (s->renegotiate == 0)
1028                s->renegotiate=1;

1030        s->new_session=1;

1032        return(s->method->ssl_renegotiate(s));
1033        }

1035 int SSL_renegotiate_abbreviated(SSL *s)
1036        {
1037        if (s->renegotiate == 0)
1038                s->renegotiate=1;

1040        s->new_session=0;

1042        return(s->method->ssl_renegotiate(s));
1043        }

1045 int SSL_renegotiate_pending(SSL *s)
1046        {
1047        /* becomes true when negotiation is requested;
1048         * false again once a handshake has finished */
1049        return (s->renegotiate != 0);
1050        }
```

```
1052 long SSL_ctrl(SSL *s,int cmd,long larg,void *parg)
1053         {
1054         long l;

1056         switch (cmd)
1057                 {
1058         case SSL_CTRL_GET_READ_AHEAD:
1059                 return(s->read_ahead);
1060         case SSL_CTRL_SET_READ_AHEAD:
1061                 l=s->read_ahead;
1062                 s->read_ahead=larg;
1063                 return(l);

1065         case SSL_CTRL_SET_MSG_CALLBACK_ARG:
1066                 s->msg_callback_arg = parg;
1067                 return 1;

1069         case SSL_CTRL_OPTIONS:
1070                 return(s->options|=larg);
1071         case SSL_CTRL_CLEAR_OPTIONS:
1072                 return(s->options&=~larg);
1073         case SSL_CTRL_MODE:
1074                 return(s->mode|=larg);
1075         case SSL_CTRL_CLEAR_MODE:
1076                 return(s->mode &=~larg);
1077         case SSL_CTRL_GET_MAX_CERT_LIST:
1078                 return(s->max_cert_list);
1079         case SSL_CTRL_SET_MAX_CERT_LIST:
1080                 l=s->max_cert_list;
1081                 s->max_cert_list=larg;
1082                 return(l);
1083         case SSL_CTRL_SET_MTU:
1084 #ifndef OPENSSL_NO_DTLS1
1085                 if (larg < (long)dtls1_min_mtu())
1086                         return 0;
1087 #endif

1089                 if (SSL_version(s) == DTLS1_VERSION ||
1090                     SSL_version(s) == DTLS1_BAD_VER)
1091                         {
1092                         s->d1->mtu = larg;
1093                         return larg;
1094                         }
1095                 return 0;
1096         case SSL_CTRL_SET_MAX_SEND_FRAGMENT:
1097                 if (larg < 512 || larg > SSL3_RT_MAX_PLAIN_LENGTH)
1098                         return 0;
1099                 s->max_send_fragment = larg;
1100                 return 1;
1101         case SSL_CTRL_GET_RI_SUPPORT:
1102                 if (s->s3)
1103                         return s->s3->send_connection_binding;
1104                 else return 0;
1105         default:
1106                 return(s->method->ssl_ctrl(s,cmd,larg,parg));
1107                 }
1108         }

1110 long SSL_callback_ctrl(SSL *s, int cmd, void (*fp)(void))
1111         {
1112         switch(cmd)
1113                 {
1114         case SSL_CTRL_SET_MSG_CALLBACK:
1115                 s->msg_callback = (void (*)(int write_p, int version, int conten
1116                 return 1;
```



```
1118         default:
1119                 return(s->method->ssl_callback_ctrl(s,cmd,fp));
1120                 }
1121         }

1123 LHASH_OF(SSL_SESSION) *SSL_CTX_sessions(SSL_CTX *ctx)
1124         {
1125         return ctx->sessions;
1126         }

1128 long SSL_CTX_ctrl(SSL_CTX *ctx,int cmd,long larg,void *parg)
1129         {
1130         long l;

1132         switch (cmd)
1133                 {
1134         case SSL_CTRL_GET_READ_AHEAD:
1135                 return(ctx->read_ahead);
1136         case SSL_CTRL_SET_READ_AHEAD:
1137                 l=ctx->read_ahead;
1138                 ctx->read_ahead=larg;
1139                 return(l);

1141         case SSL_CTRL_SET_MSG_CALLBACK_ARG:
1142                 ctx->msg_callback_arg = parg;
1143                 return 1;

1145         case SSL_CTRL_GET_MAX_CERT_LIST:
1146                 return(ctx->max_cert_list);
1147         case SSL_CTRL_SET_MAX_CERT_LIST:
1148                 l=ctx->max_cert_list;
1149                 ctx->max_cert_list=larg;
1150                 return(l);

1152         case SSL_CTRL_SET_SESS_CACHE_SIZE:
1153                 l=ctx->session_cache_size;
1154                 ctx->session_cache_size=larg;
1155                 return(l);
1156         case SSL_CTRL_GET_SESS_CACHE_SIZE:
1157                 return(ctx->session_cache_size);
1158         case SSL_CTRL_SET_SESS_CACHE_MODE:
1159                 l=ctx->session_cache_mode;
1160                 ctx->session_cache_mode=larg;
1161                 return(l);
1162         case SSL_CTRL_GET_SESS_CACHE_MODE:
1163                 return(ctx->session_cache_mode);

1165         case SSL_CTRL_SESS_NUMBER:
1166                 return(lh_SSL_SESSION_num_items(ctx->sessions));
1167         case SSL_CTRL_SESS_CONNECT:
1168                 return(ctx->stats.sess_connect);
1169         case SSL_CTRL_SESS_CONNECT_GOOD:
1170                 return(ctx->stats.sess_connect_good);
1171         case SSL_CTRL_SESS_CONNECT_RENEGOTIATE:
1172                 return(ctx->stats.sess_connect_renegotiate);
1173         case SSL_CTRL_SESS_ACCEPT:
1174                 return(ctx->stats.sess_accept);
1175         case SSL_CTRL_SESS_ACCEPT_GOOD:
1176                 return(ctx->stats.sess_accept_good);
1177         case SSL_CTRL_SESS_ACCEPT_RENEGOTIATE:
1178                 return(ctx->stats.sess_accept_renegotiate);
1179         case SSL_CTRL_SESS_HIT:
1180                 return(ctx->stats.sess_hit);
1181         case SSL_CTRL_SESS_CB_HIT:
1182                 return(ctx->stats.sess_cb_hit);
1183         case SSL_CTRL_SESS_MISSES:
```

```
1184                 return(ctx->stats.sess_miss);
1185         case SSL_CTRL_SESS_TIMEOUTS:
1186                 return(ctx->stats.sess_timeout);
1187         case SSL_CTRL_SESS_CACHE_FULL:
1188                 return(ctx->stats.sess_cache_full);
1189         case SSL_CTRL_OPTIONS:
1190                 return(ctx->options|=larg);
1191         case SSL_CTRL_CLEAR_OPTIONS:
1192                 return(ctx->options&=~larg);
1193         case SSL_CTRL_MODE:
1194                 return(ctx->mode|=larg);
1195         case SSL_CTRL_CLEAR_MODE:
1196                 return(ctx->mode&=~larg);
1197         case SSL_CTRL_SET_MAX_SEND_FRAGMENT:
1198                 if (larg < 512 || larg > SSL3_RT_MAX_PLAIN_LENGTH)
1199                         return 0;
1200                 ctx->max_send_fragment = larg;
1201                 return 1;
1202         default:
1203                 return(ctx->method->ssl_ctx_ctrl(ctx,cmd,larg,parg));
1204                 }
1205         }

1207 long SSL_CTX_callback_ctrl(SSL_CTX *ctx, int cmd, void (*fp)(void))
1208         {
1209         switch(cmd)
1210                 {
1211         case SSL_CTRL_SET_MSG_CALLBACK:
1212                 ctx->msg_callback = (void (*)(int write_p, int version, int cont
1213                 return 1;

1215         default:
1216                 return(ctx->method->ssl_ctx_callback_ctrl(ctx,cmd,fp));
1217                 }
1218         }

1220 int ssl_cipher_id_cmp(const SSL_CIPHER *a, const SSL_CIPHER *b)
1221         {
1222         long l;

1224         l=a->id-b->id;
1225         if (l == 0L)
1226                 return(0);
1227         else
1228                 return((l > 0)?1:-1);
1229         }

1231 int ssl_cipher_ptr_id_cmp(const SSL_CIPHER * const *ap,
1232                           const SSL_CIPHER * const *bp)
1233         {
1234         long l;

1236         l=(*ap)->id-(*bp)->id;
1237         if (l == 0L)
1238                 return(0);
1239         else
1240                 return((l > 0)?1:-1);
1241         }

1243 /** return a STACK of the ciphers available for the SSL and in order of
1244  * preference */
1245 STACK_OF(SSL_CIPHER) *SSL_get_ciphers(const SSL *s)
1246         {
1247         if (s != NULL)
1248                 {
1249                 if (s->cipher_list != NULL)
```



```
1250                         {
1251                         return(s->cipher_list);
1252                         }
1253                 else if ((s->ctx != NULL) &&
1254                         (s->ctx->cipher_list != NULL))
1255                         {
1256                         return(s->ctx->cipher_list);
1257                         }
1258                 }
1259         return(NULL);
1260         }

1262 /** return a STACK of the ciphers available for the SSL and in order of
1263  * algorithm id */
1264 STACK_OF(SSL_CIPHER) *ssl_get_ciphers_by_id(SSL *s)
1265         {
1266         if (s != NULL)
1267                 {
1268                 if (s->cipher_list_by_id != NULL)
1269                         {
1270                         return(s->cipher_list_by_id);
1271                         }
1272                 else if ((s->ctx != NULL) &&
1273                         (s->ctx->cipher_list_by_id != NULL))
1274                         {
1275                         return(s->ctx->cipher_list_by_id);
1276                         }
1277                 }
1278         return(NULL);
1279         }

1281 /** The old interface to get the same thing as SSL_get_ciphers() */
1282 const char *SSL_get_cipher_list(const SSL *s,int n)
1283         {
1284         SSL_CIPHER *c;
1285         STACK_OF(SSL_CIPHER) *sk;

1287         if (s == NULL) return(NULL);
1288         sk=SSL_get_ciphers(s);
1289         if ((sk == NULL) || (sk_SSL_CIPHER_num(sk) <= n))
1290                 return(NULL);
1291         c=sk_SSL_CIPHER_value(sk,n);
1292         if (c == NULL) return(NULL);
1293         return(c->name);
1294         }

1296 /** specify the ciphers to be used by default by the SSL_CTX */
1297 int SSL_CTX_set_cipher_list(SSL_CTX *ctx, const char *str)
1298         {
1299         STACK_OF(SSL_CIPHER) *sk;

1301         sk=ssl_create_cipher_list(ctx->method,&ctx->cipher_list,
1302                 &ctx->cipher_list_by_id,str);
1303         /* ssl_create_cipher_list may return an empty stack if it
1304          * was unable to find a cipher matching the given rule string
1305          * (for example if the rule string specifies a cipher which
1306          * has been disabled). This is not an error as far as
1307          * ssl_create_cipher_list is concerned, and hence
1308          * ctx->cipher_list and ctx->cipher_list_by_id has been
1309          * updated. */
1310         if (sk == NULL)
1311                 return 0;
1312         else if (sk_SSL_CIPHER_num(sk) == 0)
1313                 {
1314                 SSLerr(SSL_F_SSL_CTX_SET_CIPHER_LIST, SSL_R_NO_CIPHER_MATCH);
1315                 return 0;
```

```
1316                }
1317        return 1;
1318        }

1320 /** specify the ciphers to be used by the SSL */
1321 int SSL_set_cipher_list(SSL *s,const char *str)
1322        {
1323        STACK_OF(SSL_CIPHER) *sk;

1325        sk=ssl_create_cipher_list(s->ctx->method,&s->cipher_list,
1326                &s->cipher_list_by_id,str);
1327        /* see comment in SSL_CTX_set_cipher_list */
1328        if (sk == NULL)
1329                return 0;
1330        else if (sk_SSL_CIPHER_num(sk) == 0)
1331                {
1332                SSLerr(SSL_F_SSL_SET_CIPHER_LIST, SSL_R_NO_CIPHER_MATCH);
1333                return 0;
1334                }
1335        return 1;
1336        }

1338 /* works well for SSLv2, not so good for SSLv3 */
1339 char *SSL_get_shared_ciphers(const SSL *s,char *buf,int len)
1340        {
1341        char *p;
1342        STACK_OF(SSL_CIPHER) *sk;
1343        SSL_CIPHER *c;
1344        int i;

1346        if ((s->session == NULL) || (s->session->ciphers == NULL) ||
1347                (len < 2))
1348                return(NULL);

1350        p=buf;
1351        sk=s->session->ciphers;

1353        if (sk_SSL_CIPHER_num(sk) == 0)
1354                return NULL;

1356        for (i=0; i<sk_SSL_CIPHER_num(sk); i++)
1357                {
1358                int n;

1360                c=sk_SSL_CIPHER_value(sk,i);
1361                n=strlen(c->name);
1362                if (n+1 > len)
1363                        {
1364                        if (p != buf)
1365                                --p;
1366                        *p='\0';
1367                        return buf;
1368                        }
1369                strcpy(p,c->name);
1370                p+=n;
1371                *(p++)=':';
1372                len-=n+1;
1373                }
1374        p[-1]='\0';
1375        return(buf);
1376        }

1378 int ssl_cipher_list_to_bytes(SSL *s,STACK_OF(SSL_CIPHER) *sk,unsigned char *p,
1379                              int (*put_cb)(const SSL_CIPHER *, unsigned char *))
1380        {
1381        int i,j=0;
```



```
1382        SSL_CIPHER *c;
1383        unsigned char *q;
1384 #ifndef OPENSSL_NO_KRB5
1385        int nokrb5 = !kssl_tgt_is_available(s->kssl_ctx);
1386 #endif /* OPENSSL_NO_KRB5 */

1388        if (sk == NULL) return(0);
1389        q=p;

1391        for (i=0; i<sk_SSL_CIPHER_num(sk); i++)
1392                {
1393                c=sk_SSL_CIPHER_value(sk,i);
1394                /* Skip TLS v1.2 only ciphersuites if lower than v1.2 */
1395                if ((c->algorithm_ssl & SSL_TLSV1_2) &&
1396                        (TLS1_get_client_version(s) < TLS1_2_VERSION))
1397                        continue;
1398 #ifndef OPENSSL_NO_KRB5
1399                if (((c->algorithm_mkey & SSL_kKRB5) || (c->algorithm_auth & SSL
1400                    nokrb5)
1401                    continue;
1402 #endif /* OPENSSL_NO_KRB5 */
1403 #ifndef OPENSSL_NO_PSK
1404                /* with PSK there must be client callback set */
1405                if (((c->algorithm_mkey & SSL_kPSK) || (c->algorithm_auth & SSL_
1406                    s->psk_client_callback == NULL)
1407                        continue;
1408 #endif /* OPENSSL_NO_PSK */
1409 #ifndef OPENSSL_NO_SRP
1410                if (((c->algorithm_mkey & SSL_kSRP) || (c->algorithm_auth & SSL_
1411                    !(s->srp_ctx.srp_Mask & SSL_kSRP))
1412                    continue;
1413 #endif /* OPENSSL_NO_SRP */
1414                j = put_cb ? put_cb(c,p) : ssl_put_cipher_by_char(s,c,p);
1415                p+=j;
1416                }
1417        /* If p == q, no ciphers and caller indicates an error. Otherwise
1418         * add SCSV if not renegotiating.
1419         */
1420        if (p != q && !s->renegotiate)
1421                {
1422                static SSL_CIPHER scsv =
1423                        {
1424                        0, NULL, SSL3_CK_SCSV, 0, 0, 0, 0, 0, 0, 0, 0, 0
1425                        };
1426                j = put_cb ? put_cb(&scsv,p) : ssl_put_cipher_by_char(s,&scsv,p)
1427                p+=j;
1428 #ifdef OPENSSL_RI_DEBUG
1429                fprintf(stderr, "SCSV sent by client\n");
1430 #endif
1431                }

1433        return(p-q);
1434        }

1436 STACK_OF(SSL_CIPHER) *ssl_bytes_to_cipher_list(SSL *s,unsigned char *p,int num,
1437                                                STACK_OF(SSL_CIPHER) **skp)
1438        {
1439        const SSL_CIPHER *c;
1440        STACK_OF(SSL_CIPHER) *sk;
1441        int i,n;
1442        if (s->s3)
1443                s->s3->send_connection_binding = 0;

1445        n=ssl_put_cipher_by_char(s,NULL,NULL);
1446        if ((num%n) != 0)
1447                {
```

```
1448                 SSLerr(SSL_F_SSL_BYTES_TO_CIPHER_LIST,SSL_R_ERROR_IN_RECEIVED_CI
1449                 return(NULL);
1450                 }
1451         if ((skp == NULL) || (*skp == NULL))
1452                 sk=sk_SSL_CIPHER_new_null(); /* change perhaps later */
1453         else
1454                 {
1455                 sk= *skp;
1456                 sk_SSL_CIPHER_zero(sk);
1457                 }

1459         for (i=0; i<num; i+=n)
1460                 {
1461                 /* Check for SCSV */
1462                 if (s->s3 && (n != 3 || !p[0]) &&
1463                         (p[n-2] == ((SSL3_CK_SCSV >> 8) & 0xff)) &&
1464                         (p[n-1] == (SSL3_CK_SCSV & 0xff)))
1465                         {
1466                         /* SCSV fatal if renegotiating */
1467                         if (s->renegotiate)
1468                                 {
1469                                 SSLerr(SSL_F_SSL_BYTES_TO_CIPHER_LIST,SSL_R_SCSV
1470                                 ssl3_send_alert(s,SSL3_AL_FATAL,SSL_AD_HANDSHAKE
1471                                 goto err;
1472                                 }
1473                         s->s3->send_connection_binding = 1;
1474                         p += n;
1475 #ifdef OPENSSL_RI_DEBUG
1476                         fprintf(stderr, "SCSV received by server\n");
1477 #endif
1478                         continue;
1479                         }

1481                 c=ssl_get_cipher_by_char(s,p);
1482                 p+=n;
1483                 if (c != NULL)
1484                         {
1485                         if (!sk_SSL_CIPHER_push(sk,c))
1486                                 {
1487                                 SSLerr(SSL_F_SSL_BYTES_TO_CIPHER_LIST,ERR_R_MALL
1488                                 goto err;
1489                                 }
1490                         }
1491                 }

1493         if (skp != NULL)
1494                 *skp=sk;
1495         return(sk);
1496 err:
1497         if ((skp == NULL) || (*skp == NULL))
1498                 sk_SSL_CIPHER_free(sk);
1499         return(NULL);
1500         }


1503 #ifndef OPENSSL_NO_TLSEXT
1504 /** return a servername extension value if provided in Client Hello, or NULL.
1505  * So far, only host_name types are defined (RFC 3546).
1506  */

1508 const char *SSL_get_servername(const SSL *s, const int type)
1509         {
1510         if (type != TLSEXT_NAMETYPE_host_name)
1511                 return NULL;

1513         return s->session && !s->tlsext_hostname ?
```



```
1514                 s->session->tlsext_hostname :
1515                 s->tlsext_hostname;
1516         }

1518 int SSL_get_servername_type(const SSL *s)
1519         {
1520         if (s->session && (!s->tlsext_hostname ? s->session->tlsext_hostname : s
1521                 return TLSEXT_NAMETYPE_host_name;
1522         return -1;
1523         }

1525 # ifndef OPENSSL_NO_NEXTPROTONEG
1526 /* SSL_select_next_proto implements the standard protocol selection. It is
1527  * expected that this function is called from the callback set by
1528  * SSL_CTX_set_next_proto_select_cb.
1529  *
1530  * The protocol data is assumed to be a vector of 8-bit, length prefixed byte
1531  * strings. The length byte itself is not included in the length. A byte
1532  * string of length 0 is invalid. No byte string may be truncated.
1533  *
1534  * The current, but experimental algorithm for selecting the protocol is:
1535  *
1536  * 1) If the server doesn't support NPN then this is indicated to the
1537  * callback. In this case, the client application has to abort the connection
1538  * or have a default application level protocol.
1539  *
1540  * 2) If the server supports NPN, but advertises an empty list then the
1541  * client selects the first protcol in its list, but indicates via the
1542  * API that this fallback case was enacted.
1543  *
1544  * 3) Otherwise, the client finds the first protocol in the server's list
1545  * that it supports and selects this protocol. This is because it's
1546  * assumed that the server has better information about which protocol
1547  * a client should use.
1548  *
1549  * 4) If the client doesn't support any of the server's advertised
1550  * protocols, then this is treated the same as case 2.
1551  *
1552  * It returns either
1553  * OPENSSL_NPN_NEGOTIATED if a common protocol was found, or
1554  * OPENSSL_NPN_NO_OVERLAP if the fallback case was reached.
1555  */
1556 int SSL_select_next_proto(unsigned char **out, unsigned char *outlen, const unsi
1557         {
1558         unsigned int i, j;
1559         const unsigned char *result;
1560         int status = OPENSSL_NPN_UNSUPPORTED;

1562         /* For each protocol in server preference order, see if we support it. *
1563         for (i = 0; i < server_len; )
1564                 {
1565                 for (j = 0; j < client_len; )
1566                         {
1567                         if (server[i] == client[j] &&
1568                             memcmp(&server[i+1], &client[j+1], server[i]) == 0)
1569                                 {
1570                                 /* We found a match */
1571                                 result = &server[i];
1572                                 status = OPENSSL_NPN_NEGOTIATED;
1573                                 goto found;
1574                                 }
1575                         j += client[j];
1576                         j++;
1577                         }
1578                 i += server[i];
1579                 i++;
```

```
1580                 }

1582         /* There's no overlap between our protocols and the server's list. */
1583         result = client;
1584         status = OPENSSL_NPN_NO_OVERLAP;

1586         found:
1587         *out = (unsigned char *) result + 1;
1588         *outlen = result[0];
1589         return status;
1590         }

1592 /* SSL_get0_next_proto_negotiated sets *data and *len to point to the client's
1593  * requested protocol for this connection and returns 0. If the client didn't
1594  * request any protocol, then *data is set to NULL.
1595  *
1596  * Note that the client can request any protocol it chooses. The value returned
1597  * from this function need not be a member of the list of supported protocols
1598  * provided by the callback.
1599  */
1600 void SSL_get0_next_proto_negotiated(const SSL *s, const unsigned char **data, un
1601         {
1602         *data = s->next_proto_negotiated;
1603         if (!*data) {
1604                 *len = 0;
1605         } else {
1606                 *len = s->next_proto_negotiated_len;
1607         }
1608 }

1610 /* SSL_CTX_set_next_protos_advertised_cb sets a callback that is called when a
1611  * TLS server needs a list of supported protocols for Next Protocol
1612  * Negotiation. The returned list must be in wire format.  The list is returned
1613  * by setting |out| to point to it and |outlen| to its length. This memory will
1614  * not be modified, but one should assume that the SSL* keeps a reference to
1615  * it.
1616  *
1617  * The callback should return SSL_TLSEXT_ERR_OK if it wishes to advertise. Other
1618  * such extension will be included in the ServerHello. */
1619 void SSL_CTX_set_next_protos_advertised_cb(SSL_CTX *ctx, int (*cb) (SSL *ssl, co
1620         {
1621         ctx->next_protos_advertised_cb = cb;
1622         ctx->next_protos_advertised_cb_arg = arg;
1623         }

1625 /* SSL_CTX_set_next_proto_select_cb sets a callback that is called when a
1626  * client needs to select a protocol from the server's provided list. |out|
1627  * must be set to point to the selected protocol (which may be within |in|).
1628  * The length of the protocol name must be written into |outlen|. The server's
1629  * advertised protocols are provided in |in| and |inlen|. The callback can
1630  * assume that |in| is syntactically valid.
1631  *
1632  * The client must select a protocol. It is fatal to the connection if this
1633  * callback returns a value other than SSL_TLSEXT_ERR_OK.
1634  */
1635 void SSL_CTX_set_next_proto_select_cb(SSL_CTX *ctx, int (*cb) (SSL *s, unsigned
1636         {
1637         ctx->next_proto_select_cb = cb;
1638         ctx->next_proto_select_cb_arg = arg;
1639         }
1640 # endif
1641 #endif

1643 int SSL_export_keying_material(SSL *s, unsigned char *out, size_t olen,
1644         const char *label, size_t llen, const unsigned char *p, size_t plen,
1645         int use_context)
```



```
1646         {
1647         if (s->version < TLS1_VERSION)
1648                 return -1;

1650         return s->method->ssl3_enc->export_keying_material(s, out, olen, label,
1651                                                            llen, p, plen,
1652                                                            use_context);
1653         }

1655 static unsigned long ssl_session_hash(const SSL_SESSION *a)
1656         {
1657         unsigned long l;

1659         l=(unsigned long)
1660                 ((unsigned int) a->session_id[0]     )|
1661                 ((unsigned int) a->session_id[1]<< 8L)|
1662                 ((unsigned long)a->session_id[2]<<16L)|
1663                 ((unsigned long)a->session_id[3]<<24L);
1664         return(l);
1665         }

1667 /* NB: If this function (or indeed the hash function which uses a sort of
1668  * coarser function than this one) is changed, ensure
1669  * SSL_CTX_has_matching_session_id() is checked accordingly. It relies on being
1670  * able to construct an SSL_SESSION that will collide with any existing session
1671  * with a matching session ID. */
1672 static int ssl_session_cmp(const SSL_SESSION *a,const SSL_SESSION *b)
1673         {
1674         if (a->ssl_version != b->ssl_version)
1675                 return(1);
1676         if (a->session_id_length != b->session_id_length)
1677                 return(1);
1678         return(memcmp(a->session_id,b->session_id,a->session_id_length));
1679         }

1681 /* These wrapper functions should remain rather than redeclaring
1682  * SSL_SESSION_hash and SSL_SESSION_cmp for void* types and casting each
1683  * variable. The reason is that the functions aren't static, they're exposed via
1684  * ssl.h. */
1685 static IMPLEMENT_LHASH_HASH_FN(ssl_session, SSL_SESSION)
1686 static IMPLEMENT_LHASH_COMP_FN(ssl_session, SSL_SESSION)

1688 SSL_CTX *SSL_CTX_new(const SSL_METHOD *meth)
1689         {
1690         SSL_CTX *ret=NULL;

1692         if (meth == NULL)
1693                 {
1694                 SSLerr(SSL_F_SSL_CTX_NEW,SSL_R_NULL_SSL_METHOD_PASSED);
1695                 return(NULL);
1696                 }

1698 #ifdef OPENSSL_FIPS
1699         if (FIPS_mode() && (meth->version < TLS1_VERSION))
1700                 {
1701                 SSLerr(SSL_F_SSL_CTX_NEW, SSL_R_ONLY_TLS_ALLOWED_IN_FIPS_MODE);
1702                 return NULL;
1703                 }
1704 #endif

1706         if (SSL_get_ex_data_X509_STORE_CTX_idx() < 0)
1707                 {
1708                 SSLerr(SSL_F_SSL_CTX_NEW,SSL_R_X509_VERIFICATION_SETUP_PROBLEMS)
1709                 goto err;
1710                 }
1711         ret=(SSL_CTX *)OPENSSL_malloc(sizeof(SSL_CTX));
```

```
1712         if (ret == NULL)
1713                 goto err;

1715         memset(ret,0,sizeof(SSL_CTX));

1717         ret->method=meth;

1719         ret->cert_store=NULL;
1720         ret->session_cache_mode=SSL_SESS_CACHE_SERVER;
1721         ret->session_cache_size=SSL_SESSION_CACHE_MAX_SIZE_DEFAULT;
1722         ret->session_cache_head=NULL;
1723         ret->session_cache_tail=NULL;

1725         /* We take the system default */
1726         ret->session_timeout=meth->get_timeout();

1728         ret->new_session_cb=0;
1729         ret->remove_session_cb=0;
1730         ret->get_session_cb=0;
1731         ret->generate_session_id=0;

1733         memset((char *)&ret->stats,0,sizeof(ret->stats));

1735         ret->references=1;
1736         ret->quiet_shutdown=0;

1738 /*      ret->cipher=NULL;*/
1739 /*      ret->s2->challenge=NULL;
1740         ret->master_key=NULL;
1741         ret->key_arg=NULL;
1742         ret->s2->conn_id=NULL; */

1744         ret->info_callback=NULL;

1746         ret->app_verify_callback=0;
1747         ret->app_verify_arg=NULL;

1749         ret->max_cert_list=SSL_MAX_CERT_LIST_DEFAULT;
1750         ret->read_ahead=0;
1751         ret->msg_callback=0;
1752         ret->msg_callback_arg=NULL;
1753         ret->verify_mode=SSL_VERIFY_NONE;
1754 #if 0
1755         ret->verify_depth=-1; /* Don't impose a limit (but x509_lu.c does) */
1756 #endif
1757         ret->sid_ctx_length=0;
1758         ret->default_verify_callback=NULL;
1759         if ((ret->cert=ssl_cert_new()) == NULL)
1760                 goto err;

1762         ret->default_passwd_callback=0;
1763         ret->default_passwd_callback_userdata=NULL;
1764         ret->client_cert_cb=0;
1765         ret->app_gen_cookie_cb=0;
1766         ret->app_verify_cookie_cb=0;

1768         ret->sessions=lh_SSL_SESSION_new();
1769         if (ret->sessions == NULL) goto err;
1770         ret->cert_store=X509_STORE_new();
1771         if (ret->cert_store == NULL) goto err;

1773         ssl_create_cipher_list(ret->method,
1774                 &ret->cipher_list,&ret->cipher_list_by_id,
1775                 meth->version == SSL2_VERSION ? "SSLv2" : SSL_DEFAULT_CIPHER_LIS
1776         if (ret->cipher_list == NULL
1777             || sk_SSL_CIPHER_num(ret->cipher_list) <= 0)
```



```
1778                 {
1779                 SSLerr(SSL_F_SSL_CTX_NEW,SSL_R_LIBRARY_HAS_NO_CIPHERS);
1780                 goto err2;
1781                 }

1783         ret->param = X509_VERIFY_PARAM_new();
1784         if (!ret->param)
1785                 goto err;

1787         if ((ret->rsa_md5=EVP_get_digestbyname("ssl2-md5")) == NULL)
1788                 {
1789                 SSLerr(SSL_F_SSL_CTX_NEW,SSL_R_UNABLE_TO_LOAD_SSL2_MD5_ROUTINES)
1790                 goto err2;
1791                 }
1792         if ((ret->md5=EVP_get_digestbyname("ssl3-md5")) == NULL)
1793                 {
1794                 SSLerr(SSL_F_SSL_CTX_NEW,SSL_R_UNABLE_TO_LOAD_SSL3_MD5_ROUTINES)
1795                 goto err2;
1796                 }
1797         if ((ret->sha1=EVP_get_digestbyname("ssl3-sha1")) == NULL)
1798                 {
1799                 SSLerr(SSL_F_SSL_CTX_NEW,SSL_R_UNABLE_TO_LOAD_SSL3_SHA1_ROUTINES
1800                 goto err2;
1801                 }

1803         if ((ret->client_CA=sk_X509_NAME_new_null()) == NULL)
1804                 goto err;

1806         CRYPTO_new_ex_data(CRYPTO_EX_INDEX_SSL_CTX, ret, &ret->ex_data);

1808         ret->extra_certs=NULL;
1809         /* No compression for DTLS */
1810         if (meth->version != DTLS1_VERSION)
1811                 ret->comp_methods=SSL_COMP_get_compression_methods();

1813         ret->max_send_fragment = SSL3_RT_MAX_PLAIN_LENGTH;

1815 #ifndef OPENSSL_NO_TLSEXT
1816         ret->tlsext_servername_callback = 0;
1817         ret->tlsext_servername_arg = NULL;
1818         /* Setup RFC4507 ticket keys */
1819         if ((RAND_pseudo_bytes(ret->tlsext_tick_key_name, 16) <= 0)
1820                 || (RAND_bytes(ret->tlsext_tick_hmac_key, 16) <= 0)
1821                 || (RAND_bytes(ret->tlsext_tick_aes_key, 16) <= 0))
1822                 ret->options |= SSL_OP_NO_TICKET;

1824         ret->tlsext_status_cb = 0;
1825         ret->tlsext_status_arg = NULL;

1827 # ifndef OPENSSL_NO_NEXTPROTONEG
1828         ret->next_protos_advertised_cb = 0;
1829         ret->next_proto_select_cb = 0;
1830 # endif
1831 #endif
1832 #ifndef OPENSSL_NO_PSK
1833         ret->psk_identity_hint=NULL;
1834         ret->psk_client_callback=NULL;
1835         ret->psk_server_callback=NULL;
1836 #endif
1837 #ifndef OPENSSL_NO_SRP
1838         SSL_CTX_SRP_CTX_init(ret);
1839 #endif
1840 #ifndef OPENSSL_NO_BUF_FREELISTS
1841         ret->freelist_max_len = SSL_MAX_BUF_FREELIST_LEN_DEFAULT;
1842         ret->rbuf_freelist = OPENSSL_malloc(sizeof(SSL3_BUF_FREELIST));
1843         if (!ret->rbuf_freelist)
```

```
1844                 goto err;
1845         ret->rbuf_freelist->chunklen = 0;
1846         ret->rbuf_freelist->len = 0;
1847         ret->rbuf_freelist->head = NULL;
1848         ret->wbuf_freelist = OPENSSL_malloc(sizeof(SSL3_BUF_FREELIST));
1849         if (!ret->wbuf_freelist)
1850                 {
1851                 OPENSSL_free(ret->rbuf_freelist);
1852                 goto err;
1853                 }
1854         ret->wbuf_freelist->chunklen = 0;
1855         ret->wbuf_freelist->len = 0;
1856         ret->wbuf_freelist->head = NULL;
1857 #endif
1858 #ifndef OPENSSL_NO_ENGINE
1859         ret->client_cert_engine = NULL;
1860 #ifdef OPENSSL_SSL_CLIENT_ENGINE_AUTO
1861 #define eng_strx(x)     #x
1862 #define eng_str(x)      eng_strx(x)
1863         /* Use specific client engine automatically... ignore errors */
1864         {
1865         ENGINE *eng;
1866         eng = ENGINE_by_id(eng_str(OPENSSL_SSL_CLIENT_ENGINE_AUTO));
1867         if (!eng)
1868                 {
1869                 ERR_clear_error();
1870                 ENGINE_load_builtin_engines();
1871                 eng = ENGINE_by_id(eng_str(OPENSSL_SSL_CLIENT_ENGINE_AUTO));
1872                 }
1873         if (!eng || !SSL_CTX_set_client_cert_engine(ret, eng))
1874                 ERR_clear_error();
1875         }
1876 #endif
1877 #endif
1878         /* Default is to connect to non-RI servers. When RI is more widely
1879          * deployed might change this.
1880          */
1881         ret->options |= SSL_OP_LEGACY_SERVER_CONNECT;

1883         return(ret);
1884 err:
1885         SSLerr(SSL_F_SSL_CTX_NEW,ERR_R_MALLOC_FAILURE);
1886 err2:
1887         if (ret != NULL) SSL_CTX_free(ret);
1888         return(NULL);
1889         }

1891 #if 0
1892 static void SSL_COMP_free(SSL_COMP *comp)
1893     { OPENSSL_free(comp); }
1894 #endif

1896 #ifndef OPENSSL_NO_BUF_FREELISTS
1897 static void
1898 ssl_buf_freelist_free(SSL3_BUF_FREELIST *list)
1899         {
1900         SSL3_BUF_FREELIST_ENTRY *ent, *next;
1901         for (ent = list->head; ent; ent = next)
1902                 {
1903                 next = ent->next;
1904                 OPENSSL_free(ent);
1905                 }
1906         OPENSSL_free(list);
1907         }
1908 #endif
```



```
1910 void SSL_CTX_free(SSL_CTX *a)
1911         {
1912         int i;

1914         if (a == NULL) return;

1916         i=CRYPTO_add(&a->references,-1,CRYPTO_LOCK_SSL_CTX);
1917 #ifdef REF_PRINT
1918         REF_PRINT("SSL_CTX",a);
1919 #endif
1920         if (i > 0) return;
1921 #ifdef REF_CHECK
1922         if (i < 0)
1923                 {
1924                 fprintf(stderr,"SSL_CTX_free, bad reference count\n");
1925                 abort(); /* ok */
1926                 }
1927 #endif

1929         if (a->param)
1930                 X509_VERIFY_PARAM_free(a->param);

1932         /*
1933          * Free internal session cache. However: the remove_cb() may reference
1934          * the ex_data of SSL_CTX, thus the ex_data store can only be removed
1935          * after the sessions were flushed.
1936          * As the ex_data handling routines might also touch the session cache,
1937          * the most secure solution seems to be: empty (flush) the cache, then
1938          * free ex_data, then finally free the cache.
1939          * (See ticket [openssl.org #212].)
1940          */
1941         if (a->sessions != NULL)
1942                 SSL_CTX_flush_sessions(a,0);

1944         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_SSL_CTX, a, &a->ex_data);

1946         if (a->sessions != NULL)
1947                 lh_SSL_SESSION_free(a->sessions);

1949         if (a->cert_store != NULL)
1950                 X509_STORE_free(a->cert_store);
1951         if (a->cipher_list != NULL)
1952                 sk_SSL_CIPHER_free(a->cipher_list);
1953         if (a->cipher_list_by_id != NULL)
1954                 sk_SSL_CIPHER_free(a->cipher_list_by_id);
1955         if (a->cert != NULL)
1956                 ssl_cert_free(a->cert);
1957         if (a->client_CA != NULL)
1958                 sk_X509_NAME_pop_free(a->client_CA,X509_NAME_free);
1959         if (a->extra_certs != NULL)
1960                 sk_X509_pop_free(a->extra_certs,X509_free);
1961 #if 0 /* This should never be done, since it removes a global database */
1962         if (a->comp_methods != NULL)
1963                 sk_SSL_COMP_pop_free(a->comp_methods,SSL_COMP_free);
1964 #else
1965         a->comp_methods = NULL;
1966 #endif

1968 #ifndef OPENSSL_NO_SRTP
1969         if (a->srtp_profiles)
1970                 sk_SRTP_PROTECTION_PROFILE_free(a->srtp_profiles);
1971 #endif

1973 #ifndef OPENSSL_NO_PSK
1974         if (a->psk_identity_hint)
1975                 OPENSSL_free(a->psk_identity_hint);
```

```
#endif
#ifndef OPENSSL_NO_SRP
        SSL_CTX_SRP_CTX_free(a);
#endif
#ifndef OPENSSL_NO_ENGINE
        if (a->client_cert_engine)
                ENGINE_finish(a->client_cert_engine);
#endif

#ifndef OPENSSL_NO_BUF_FREELISTS
        if (a->wbuf_freelist)
                ssl_buf_freelist_free(a->wbuf_freelist);
        if (a->rbuf_freelist)
                ssl_buf_freelist_free(a->rbuf_freelist);
#endif

        OPENSSL_free(a);
        }

void SSL_CTX_set_default_passwd_cb(SSL_CTX *ctx, pem_password_cb *cb)
        {
        ctx->default_passwd_callback=cb;
        }

void SSL_CTX_set_default_passwd_cb_userdata(SSL_CTX *ctx,void *u)
        {
        ctx->default_passwd_callback_userdata=u;
        }

void SSL_CTX_set_cert_verify_callback(SSL_CTX *ctx, int (*cb)(X509_STORE_CTX *,v
        {
        ctx->app_verify_callback=cb;
        ctx->app_verify_arg=arg;
        }

void SSL_CTX_set_verify(SSL_CTX *ctx,int mode,int (*cb)(int, X509_STORE_CTX *))
        {
        ctx->verify_mode=mode;
        ctx->default_verify_callback=cb;
        }

void SSL_CTX_set_verify_depth(SSL_CTX *ctx,int depth)
        {
        X509_VERIFY_PARAM_set_depth(ctx->param, depth);
        }

void ssl_set_cert_masks(CERT *c, const SSL_CIPHER *cipher)
        {
        CERT_PKEY *cpk;
        int rsa_enc,rsa_tmp,rsa_sign,dh_tmp,dh_rsa,dh_dsa,dsa_sign;
        int rsa_enc_export,dh_rsa_export,dh_dsa_export;
        int rsa_tmp_export,dh_tmp_export,kl;
        unsigned long mask_k,mask_a,emask_k,emask_a;
        int have_ecc_cert, ecdh_ok, ecdsa_ok, ecc_pkey_size;
#ifndef OPENSSL_NO_ECDH
        int have_ecdh_tmp;
#endif
        X509 *x = NULL;
        EVP_PKEY *ecc_pkey = NULL;
        int signature_nid = 0, pk_nid = 0, md_nid = 0;

        if (c == NULL) return;

        kl=SSL_C_EXPORT_PKEYLENGTH(cipher);

#ifndef OPENSSL_NO_RSA
```



```
        rsa_tmp=(c->rsa_tmp != NULL || c->rsa_tmp_cb != NULL);
        rsa_tmp_export=(c->rsa_tmp_cb != NULL ||
                (rsa_tmp && RSA_size(c->rsa_tmp)*8 <= kl));
#else
        rsa_tmp=rsa_tmp_export=0;
#endif
#ifndef OPENSSL_NO_DH
        dh_tmp=(c->dh_tmp != NULL || c->dh_tmp_cb != NULL);
        dh_tmp_export=(c->dh_tmp_cb != NULL ||
                (dh_tmp && DH_size(c->dh_tmp)*8 <= kl));
#else
        dh_tmp=dh_tmp_export=0;
#endif

#ifndef OPENSSL_NO_ECDH
        have_ecdh_tmp=(c->ecdh_tmp != NULL || c->ecdh_tmp_cb != NULL);
#endif
        cpk= &(c->pkeys[SSL_PKEY_RSA_ENC]);
        rsa_enc= (cpk->x509 != NULL && cpk->privatekey != NULL);
        rsa_enc_export=(rsa_enc && EVP_PKEY_size(cpk->privatekey)*8 <= kl);
        cpk= &(c->pkeys[SSL_PKEY_RSA_SIGN]);
        rsa_sign=(cpk->x509 != NULL && cpk->privatekey != NULL);
        cpk= &(c->pkeys[SSL_PKEY_DSA_SIGN]);
        dsa_sign=(cpk->x509 != NULL && cpk->privatekey != NULL);
        cpk= &(c->pkeys[SSL_PKEY_DH_RSA]);
        dh_rsa=  (cpk->x509 != NULL && cpk->privatekey != NULL);
        dh_rsa_export=(dh_rsa && EVP_PKEY_size(cpk->privatekey)*8 <= kl);
        cpk= &(c->pkeys[SSL_PKEY_DH_DSA]);
/* FIX THIS EAY EAY EAY */
        dh_dsa=  (cpk->x509 != NULL && cpk->privatekey != NULL);
        dh_dsa_export=(dh_dsa && EVP_PKEY_size(cpk->privatekey)*8 <= kl);
        cpk= &(c->pkeys[SSL_PKEY_ECC]);
        have_ecc_cert= (cpk->x509 != NULL && cpk->privatekey != NULL);
        mask_k=0;
        mask_a=0;
        emask_k=0;
        emask_a=0;



#ifdef CIPHER_DEBUG
        printf("rt=%d rte=%d dht=%d ecdht=%d re=%d ree=%d rs=%d ds=%d dhr=%d dhd
                rsa_tmp,rsa_tmp_export,dh_tmp,have_ecdh_tmp,
                rsa_enc,rsa_enc_export,rsa_sign,dsa_sign,dh_rsa,dh_dsa);
#endif

        cpk = &(c->pkeys[SSL_PKEY_GOST01]);
        if (cpk->x509 != NULL && cpk->privatekey !=NULL) {
                mask_k |= SSL_kGOST;
                mask_a |= SSL_aGOST01;
        }
        cpk = &(c->pkeys[SSL_PKEY_GOST94]);
        if (cpk->x509 != NULL && cpk->privatekey !=NULL) {
                mask_k |= SSL_kGOST;
                mask_a |= SSL_aGOST94;
        }

        if (rsa_enc || (rsa_tmp && rsa_sign))
                mask_k|=SSL_kRSA;
        if (rsa_enc_export || (rsa_tmp_export && (rsa_sign || rsa_enc)))
                emask_k|=SSL_kRSA;

#if 0
        /* The match needs to be both kEDH and aRSA or aDSA, so don't worry */
        if (    (dh_tmp || dh_rsa || dh_dsa) &&
                (rsa_enc || rsa_sign || dsa_sign))
```

```
2108                 mask_k|=SSL_kEDH;
2109         if ((dh_tmp_export || dh_rsa_export || dh_dsa_export) &&
2110                 (rsa_enc || rsa_sign || dsa_sign))
2111                 emask_k|=SSL_kEDH;
2112 #endif

2114         if (dh_tmp_export)
2115                 emask_k|=SSL_kEDH;

2117         if (dh_tmp)
2118                 mask_k|=SSL_kEDH;

2120         if (dh_rsa) mask_k|=SSL_kDHr;
2121         if (dh_rsa_export) emask_k|=SSL_kDHr;

2123         if (dh_dsa) mask_k|=SSL_kDHd;
2124         if (dh_dsa_export) emask_k|=SSL_kDHd;

2126         if (rsa_enc || rsa_sign)
2127                 {
2128                 mask_a|=SSL_aRSA;
2129                 emask_a|=SSL_aRSA;
2130                 }

2132         if (dsa_sign)
2133                 {
2134                 mask_a|=SSL_aDSS;
2135                 emask_a|=SSL_aDSS;
2136                 }

2138         mask_a|=SSL_aNULL;
2139         emask_a|=SSL_aNULL;

2141 #ifndef OPENSSL_NO_KRB5
2142         mask_k|=SSL_kKRB5;
2143         mask_a|=SSL_aKRB5;
2144         emask_k|=SSL_kKRB5;
2145         emask_a|=SSL_aKRB5;
2146 #endif

2148         /* An ECC certificate may be usable for ECDH and/or
2149          * ECDSA cipher suites depending on the key usage extension.
2150          */
2151         if (have_ecc_cert)
2152                 {
2153                 /* This call populates extension flags (ex_flags) */
2154                 x = (c->pkeys[SSL_PKEY_ECC]).x509;
2155                 X509_check_purpose(x, -1, 0);
2156                 ecdh_ok = (x->ex_flags & EXFLAG_KUSAGE) ?
2157                     (x->ex_kusage & X509v3_KU_KEY_AGREEMENT) : 1;
2158                 ecdsa_ok = (x->ex_flags & EXFLAG_KUSAGE) ?
2159                     (x->ex_kusage & X509v3_KU_DIGITAL_SIGNATURE) : 1;
2160                 ecc_pkey = X509_get_pubkey(x);
2161                 ecc_pkey_size = (ecc_pkey != NULL) ?
2162                     EVP_PKEY_bits(ecc_pkey) : 0;
2163                 EVP_PKEY_free(ecc_pkey);
2164                 if ((x->sig_alg) && (x->sig_alg->algorithm))
2165                         {
2166                         signature_nid = OBJ_obj2nid(x->sig_alg->algorithm);
2167                         OBJ_find_sigid_algs(signature_nid, &md_nid, &pk_nid);
2168                         }
2169 #ifndef OPENSSL_NO_ECDH
2170                 if (ecdh_ok)
2171                         {

2173                         if (pk_nid == NID_rsaEncryption || pk_nid == NID_rsa)
```



```
2174                                 {
2175                                 mask_k|=SSL_kECDHr;
2176                                 mask_a|=SSL_aECDH;
2177                                 if (ecc_pkey_size <= 163)
2178                                         {
2179                                         emask_k|=SSL_kECDHr;
2180                                         emask_a|=SSL_aECDH;
2181                                         }
2182                                 }

2184                         if (pk_nid == NID_X9_62_id_ecPublicKey)
2185                                 {
2186                                 mask_k|=SSL_kECDHe;
2187                                 mask_a|=SSL_aECDH;
2188                                 if (ecc_pkey_size <= 163)
2189                                         {
2190                                         emask_k|=SSL_kECDHe;
2191                                         emask_a|=SSL_aECDH;
2192                                         }
2193                                 }
2194                         }
2195 #endif
2196 #ifndef OPENSSL_NO_ECDSA
2197                 if (ecdsa_ok)
2198                         {
2199                         mask_a|=SSL_aECDSA;
2200                         emask_a|=SSL_aECDSA;
2201                         }
2202 #endif
2203                 }

2205 #ifndef OPENSSL_NO_ECDH
2206         if (have_ecdh_tmp)
2207                 {
2208                 mask_k|=SSL_kEECDH;
2209                 emask_k|=SSL_kEECDH;
2210                 }
2211 #endif

2213 #ifndef OPENSSL_NO_PSK
2214         mask_k |= SSL_kPSK;
2215         mask_a |= SSL_aPSK;
2216         emask_k |= SSL_kPSK;
2217         emask_a |= SSL_aPSK;
2218 #endif

2220         c->mask_k=mask_k;
2221         c->mask_a=mask_a;
2222         c->export_mask_k=emask_k;
2223         c->export_mask_a=emask_a;
2224         c->valid=1;
2225         }

2227 /* This handy macro borrowed from crypto/x509v3/v3_purp.c */
2228 #define ku_reject(x, usage) \
2229         (((x)->ex_flags & EXFLAG_KUSAGE) && !((x)->ex_kusage & (usage)))

2231 #ifndef OPENSSL_NO_EC

2233 int ssl_check_srvr_ecc_cert_and_alg(X509 *x, SSL *s)
2234         {
2235         unsigned long alg_k, alg_a;
2236         EVP_PKEY *pkey = NULL;
2237         int keysize = 0;
2238         int signature_nid = 0, md_nid = 0, pk_nid = 0;
2239         const SSL_CIPHER *cs = s->s3->tmp.new_cipher;
```

```
2241         alg_k = cs->algorithm_mkey;
2242         alg_a = cs->algorithm_auth;

2244         if (SSL_C_IS_EXPORT(cs))
2245                 {
2246                 /* ECDH key length in export ciphers must be <= 163 bits */
2247                 pkey = X509_get_pubkey(x);
2248                 if (pkey == NULL) return 0;
2249                 keysize = EVP_PKEY_bits(pkey);
2250                 EVP_PKEY_free(pkey);
2251                 if (keysize > 163) return 0;
2252                 }

2254         /* This call populates the ex_flags field correctly */
2255         X509_check_purpose(x, -1, 0);
2256         if ((x->sig_alg) && (x->sig_alg->algorithm))
2257                 {
2258                 signature_nid = OBJ_obj2nid(x->sig_alg->algorithm);
2259                 OBJ_find_sigid_algs(signature_nid, &md_nid, &pk_nid);
2260                 }
2261         if (alg_k & SSL_kECDHe || alg_k & SSL_kECDHr)
2262                 {
2263                 /* key usage, if present, must allow key agreement */
2264                 if (ku_reject(x, X509v3_KU_KEY_AGREEMENT))
2265                         {
2266                         SSLerr(SSL_F_SSL_CHECK_SRVR_ECC_CERT_AND_ALG, SSL_R_ECC_
2267                         return 0;
2268                         }
2269                 if ((alg_k & SSL_kECDHe) && TLS1_get_version(s) < TLS1_2_VERSION
2270                         {
2271                         /* signature alg must be ECDSA */
2272                         if (pk_nid != NID_X9_62_id_ecPublicKey)
2273                                 {
2274                                 SSLerr(SSL_F_SSL_CHECK_SRVR_ECC_CERT_AND_ALG, SS
2275                                 return 0;
2276                                 }
2277                         }
2278                 if ((alg_k & SSL_kECDHr) && TLS1_get_version(s) < TLS1_2_VERSION
2279                         {
2280                         /* signature alg must be RSA */

2282                         if (pk_nid != NID_rsaEncryption && pk_nid != NID_rsa)
2283                                 {
2284                                 SSLerr(SSL_F_SSL_CHECK_SRVR_ECC_CERT_AND_ALG, SS
2285                                 return 0;
2286                                 }
2287                         }
2288                 }
2289         if (alg_a & SSL_aECDSA)
2290                 {
2291                 /* key usage, if present, must allow signing */
2292                 if (ku_reject(x, X509v3_KU_DIGITAL_SIGNATURE))
2293                         {
2294                         SSLerr(SSL_F_SSL_CHECK_SRVR_ECC_CERT_AND_ALG, SSL_R_ECC_
2295                         return 0;
2296                         }
2297                 }

2299         return 1;  /* all checks are ok */
2300         }

2302 #endif

2304 /* THIS NEEDS CLEANING UP */
2305 CERT_PKEY *ssl_get_server_send_pkey(const SSL *s)
```



```
2306         {
2307         unsigned long alg_k,alg_a;
2308         CERT *c;
2309         int i;

2311         c=s->cert;
2312         ssl_set_cert_masks(c, s->s3->tmp.new_cipher);

2314         alg_k = s->s3->tmp.new_cipher->algorithm_mkey;
2315         alg_a = s->s3->tmp.new_cipher->algorithm_auth;

2317         if (alg_k & (SSL_kECDHr|SSL_kECDHe))
2318                 {
2319                 /* we don't need to look at SSL_kEECDH
2320                  * since no certificate is needed for
2321                  * anon ECDH and for authenticated
2322                  * EECDH, the check for the auth
2323                  * algorithm will set i correctly
2324                  * NOTE: For ECDH-RSA, we need an ECC
2325                  * not an RSA cert but for EECDH-RSA
2326                  * we need an RSA cert. Placing the
2327                  * checks for SSL_kECDH before RSA
2328                  * checks ensures the correct cert is chosen.
2329                  */
2330                 i=SSL_PKEY_ECC;
2331                 }
2332         else if (alg_a & SSL_aECDSA)
2333                 {
2334                 i=SSL_PKEY_ECC;
2335                 }
2336         else if (alg_k & SSL_kDHr)
2337                 i=SSL_PKEY_DH_RSA;
2338         else if (alg_k & SSL_kDHd)
2339                 i=SSL_PKEY_DH_DSA;
2340         else if (alg_a & SSL_aDSS)
2341                 i=SSL_PKEY_DSA_SIGN;
2342         else if (alg_a & SSL_aRSA)
2343                 {
2344                 if (c->pkeys[SSL_PKEY_RSA_ENC].x509 == NULL)
2345                         i=SSL_PKEY_RSA_SIGN;
2346                 else
2347                         i=SSL_PKEY_RSA_ENC;
2348                 }
2349         else if (alg_a & SSL_aKRB5)
2350                 {
2351                 /* VRS something else here? */
2352                 return(NULL);
2353                 }
2354         else if (alg_a & SSL_aGOST94)
2355                 i=SSL_PKEY_GOST94;
2356         else if (alg_a & SSL_aGOST01)
2357                 i=SSL_PKEY_GOST01;
2358         else /* if (alg_a & SSL_aNULL) */
2359                 {
2360                 SSLerr(SSL_F_SSL_GET_SERVER_SEND_PKEY,ERR_R_INTERNAL_ERROR);
2361                 return(NULL);
2362                 }

2364         return c->pkeys + i;
2365         }

2367 X509 *ssl_get_server_send_cert(const SSL *s)
2368         {
2369         CERT_PKEY *cpk;
2370         cpk = ssl_get_server_send_pkey(s);
2371         if (!cpk)
```

```
2372                return NULL;
2373        return cpk->x509;
2374        }

2376 EVP_PKEY *ssl_get_sign_pkey(SSL *s,const SSL_CIPHER *cipher, const EVP_MD **pmd)
2377        {
2378        unsigned long alg_a;
2379        CERT *c;
2380        int idx = -1;

2382        alg_a = cipher->algorithm_auth;
2383        c=s->cert;

2385        if ((alg_a & SSL_aDSS) &&
2386                (c->pkeys[SSL_PKEY_DSA_SIGN].privatekey != NULL))
2387                idx = SSL_PKEY_DSA_SIGN;
2388        else if (alg_a & SSL_aRSA)
2389                {
2390                if (c->pkeys[SSL_PKEY_RSA_SIGN].privatekey != NULL)
2391                        idx = SSL_PKEY_RSA_SIGN;
2392                else if (c->pkeys[SSL_PKEY_RSA_ENC].privatekey != NULL)
2393                        idx = SSL_PKEY_RSA_ENC;
2394                }
2395        else if ((alg_a & SSL_aECDSA) &&
2396                 (c->pkeys[SSL_PKEY_ECC].privatekey != NULL))
2397                idx = SSL_PKEY_ECC;
2398        if (idx == -1)
2399                {
2400                SSLerr(SSL_F_SSL_GET_SIGN_PKEY,ERR_R_INTERNAL_ERROR);
2401                return(NULL);
2402                }
2403        if (pmd)
2404                *pmd = c->pkeys[idx].digest;
2405        return c->pkeys[idx].privatekey;
2406        }

2408 void ssl_update_cache(SSL *s,int mode)
2409        {
2410        int i;

2412        /* If the session_id_length is 0, we are not supposed to cache it,
2413         * and it would be rather hard to do anyway :-) */
2414        if (s->session->session_id_length == 0) return;

2416        i=s->session_ctx->session_cache_mode;
2417        if ((i & mode) && (!s->hit)
2418                && ((i & SSL_SESS_CACHE_NO_INTERNAL_STORE)
2419                    || SSL_CTX_add_session(s->session_ctx,s->session))
2420                && (s->session_ctx->new_session_cb != NULL))
2421                {
2422                CRYPTO_add(&s->session->references,1,CRYPTO_LOCK_SSL_SESSION);
2423                if (!s->session_ctx->new_session_cb(s,s->session))
2424                        SSL_SESSION_free(s->session);
2425                }

2427        /* auto flush every 255 connections */
2428        if ((!(i & SSL_SESS_CACHE_NO_AUTO_CLEAR)) &&
2429                ((i & mode) == mode))
2430                {
2431                if (  (((mode & SSL_SESS_CACHE_CLIENT)
2432                        ?s->session_ctx->stats.sess_connect_good
2433                        :s->session_ctx->stats.sess_accept_good) & 0xff) == 0xff
2434                        {
2435                        SSL_CTX_flush_sessions(s->session_ctx,(unsigned long)tim
2436                        }
2437                }
```



```
2438        }

2440 const SSL_METHOD *SSL_get_ssl_method(SSL *s)
2441        {
2442        return(s->method);
2443        }

2445 int SSL_set_ssl_method(SSL *s, const SSL_METHOD *meth)
2446        {
2447        int conn= -1;
2448        int ret=1;

2450        if (s->method != meth)
2451                {
2452                if (s->handshake_func != NULL)
2453                        conn=(s->handshake_func == s->method->ssl_connect);

2455                if (s->method->version == meth->version)
2456                        s->method=meth;
2457                else
2458                        {
2459                        s->method->ssl_free(s);
2460                        s->method=meth;
2461                        ret=s->method->ssl_new(s);
2462                        }

2464                if (conn == 1)
2465                        s->handshake_func=meth->ssl_connect;
2466                else if (conn == 0)
2467                        s->handshake_func=meth->ssl_accept;
2468                }
2469        return(ret);
2470        }

2472 int SSL_get_error(const SSL *s,int i)
2473        {
2474        int reason;
2475        unsigned long l;
2476        BIO *bio;

2478        if (i > 0) return(SSL_ERROR_NONE);

2480        /* Make things return SSL_ERROR_SYSCALL when doing SSL_do_handshake
2481         * etc, where we do encode the error */
2482        if ((l=ERR_peek_error()) != 0)
2483                {
2484                if (ERR_GET_LIB(l) == ERR_LIB_SYS)
2485                        return(SSL_ERROR_SYSCALL);
2486                else
2487                        return(SSL_ERROR_SSL);
2488                }

2490        if ((i < 0) && SSL_want_read(s))
2491                {
2492                bio=SSL_get_rbio(s);
2493                if (BIO_should_read(bio))
2494                        return(SSL_ERROR_WANT_READ);
2495                else if (BIO_should_write(bio))
2496                        /* This one doesn't make too much sense ... We never try
2497                         * to write to the rbio, and an application program wher
2498                         * rbio and wbio are separate couldn't even know what it
2499                         * should wait for.
2500                         * However if we ever set s->rwstate incorrectly
2501                         * (so that we have SSL_want_read(s) instead of
2502                         * SSL_want_write(s)) and rbio and wbio *are* the same,
2503                         * this test works around that bug; so it might be safer
```

```
2504                         * to keep it. */
2505                        return(SSL_ERROR_WANT_WRITE);
2506                else if (BIO_should_io_special(bio))
2507                        {
2508                        reason=BIO_get_retry_reason(bio);
2509                        if (reason == BIO_RR_CONNECT)
2510                                return(SSL_ERROR_WANT_CONNECT);
2511                        else if (reason == BIO_RR_ACCEPT)
2512                                return(SSL_ERROR_WANT_ACCEPT);
2513                        else
2514                                return(SSL_ERROR_SYSCALL); /* unknown */
2515                        }
2516                }

2518        if ((i < 0) && SSL_want_write(s))
2519                {
2520                bio=SSL_get_wbio(s);
2521                if (BIO_should_write(bio))
2522                        return(SSL_ERROR_WANT_WRITE);
2523                else if (BIO_should_read(bio))
2524                        /* See above (SSL_want_read(s) with BIO_should_write(bio
2525                        return(SSL_ERROR_WANT_READ);
2526                else if (BIO_should_io_special(bio))
2527                        {
2528                        reason=BIO_get_retry_reason(bio);
2529                        if (reason == BIO_RR_CONNECT)
2530                                return(SSL_ERROR_WANT_CONNECT);
2531                        else if (reason == BIO_RR_ACCEPT)
2532                                return(SSL_ERROR_WANT_ACCEPT);
2533                        else
2534                                return(SSL_ERROR_SYSCALL);
2535                        }
2536                }
2537        if ((i < 0) && SSL_want_x509_lookup(s))
2538                {
2539                return(SSL_ERROR_WANT_X509_LOOKUP);
2540                }

2542        if (i == 0)
2543                {
2544                if (s->version == SSL2_VERSION)
2545                        {
2546                        /* assume it is the socket being closed */
2547                        return(SSL_ERROR_ZERO_RETURN);
2548                        }
2549                else
2550                        {
2551                        if ((s->shutdown & SSL_RECEIVED_SHUTDOWN) &&
2552                                (s->s3->warn_alert == SSL_AD_CLOSE_NOTIFY))
2553                                return(SSL_ERROR_ZERO_RETURN);
2554                        }
2555                }
2556        return(SSL_ERROR_SYSCALL);
2557        }

2559 int SSL_do_handshake(SSL *s)
2560        {
2561        int ret=1;

2563        if (s->handshake_func == NULL)
2564                {
2565                SSLerr(SSL_F_SSL_DO_HANDSHAKE,SSL_R_CONNECTION_TYPE_NOT_SET);
2566                return(-1);
2567                }

2569        s->method->ssl_renegotiate_check(s);
```



```
2571        if (SSL_in_init(s) || SSL_in_before(s))
2572                {
2573                ret=s->handshake_func(s);
2574                }
2575        return(ret);
2576        }

2578 /* For the next 2 functions, SSL_clear() sets shutdown and so
2579  * one of these calls will reset it */
2580 void SSL_set_accept_state(SSL *s)
2581        {
2582        s->server=1;
2583        s->shutdown=0;
2584        s->state=SSL_ST_ACCEPT|SSL_ST_BEFORE;
2585        s->handshake_func=s->method->ssl_accept;
2586        /* clear the current cipher */
2587        ssl_clear_cipher_ctx(s);
2588        ssl_clear_hash_ctx(&s->read_hash);
2589        ssl_clear_hash_ctx(&s->write_hash);
2590        }

2592 void SSL_set_connect_state(SSL *s)
2593        {
2594        s->server=0;
2595        s->shutdown=0;
2596        s->state=SSL_ST_CONNECT|SSL_ST_BEFORE;
2597        s->handshake_func=s->method->ssl_connect;
2598        /* clear the current cipher */
2599        ssl_clear_cipher_ctx(s);
2600        ssl_clear_hash_ctx(&s->read_hash);
2601        ssl_clear_hash_ctx(&s->write_hash);
2602        }

2604 int ssl_undefined_function(SSL *s)
2605        {
2606        SSLerr(SSL_F_SSL_UNDEFINED_FUNCTION,ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
2607        return(0);
2608        }

2610 int ssl_undefined_void_function(void)
2611        {
2612        SSLerr(SSL_F_SSL_UNDEFINED_VOID_FUNCTION,ERR_R_SHOULD_NOT_HAVE_BEEN_CALL
2613        return(0);
2614        }

2616 int ssl_undefined_const_function(const SSL *s)
2617        {
2618        SSLerr(SSL_F_SSL_UNDEFINED_CONST_FUNCTION,ERR_R_SHOULD_NOT_HAVE_BEEN_CAL
2619        return(0);
2620        }

2622 SSL_METHOD *ssl_bad_method(int ver)
2623        {
2624        SSLerr(SSL_F_SSL_BAD_METHOD,ERR_R_SHOULD_NOT_HAVE_BEEN_CALLED);
2625        return(NULL);
2626        }

2628 const char *SSL_get_version(const SSL *s)
2629        {
2630        if (s->version == TLS1_2_VERSION)
2631                return("TLSv1.2");
2632        else if (s->version == TLS1_1_VERSION)
2633                return("TLSv1.1");
2634        else if (s->version == TLS1_VERSION)
2635                return("TLSv1");
```

```
2636         else if (s->version == SSL3_VERSION)
2637                 return("SSLv3");
2638         else if (s->version == SSL2_VERSION)
2639                 return("SSLv2");
2640         else
2641                 return("unknown");
2642         }

2644 SSL *SSL_dup(SSL *s)
2645         {
2646         STACK_OF(X509_NAME) *sk;
2647         X509_NAME *xn;
2648         SSL *ret;
2649         int i;

2651         if ((ret=SSL_new(SSL_get_SSL_CTX(s))) == NULL)
2652             return(NULL);

2654         ret->version = s->version;
2655         ret->type = s->type;
2656         ret->method = s->method;

2658         if (s->session != NULL)
2659                 {
2660                 /* This copies session-id, SSL_METHOD, sid_ctx, and 'cert' */
2661                 SSL_copy_session_id(ret,s);
2662                 }
2663         else
2664                 {
2665                 /* No session has been established yet, so we have to expect
2666                  * that s->cert or ret->cert will be changed later --
2667                  * they should not both point to the same object,
2668                  * and thus we can't use SSL_copy_session_id. */

2670                 ret->method->ssl_free(ret);
2671                 ret->method = s->method;
2672                 ret->method->ssl_new(ret);

2674                 if (s->cert != NULL)
2675                         {
2676                         if (ret->cert != NULL)
2677                                 {
2678                                 ssl_cert_free(ret->cert);
2679                                 }
2680                         ret->cert = ssl_cert_dup(s->cert);
2681                         if (ret->cert == NULL)
2682                                 goto err;
2683                         }

2685                 SSL_set_session_id_context(ret,
2686                         s->sid_ctx, s->sid_ctx_length);
2687                 }

2689         ret->options=s->options;
2690         ret->mode=s->mode;
2691         SSL_set_max_cert_list(ret,SSL_get_max_cert_list(s));
2692         SSL_set_read_ahead(ret,SSL_get_read_ahead(s));
2693         ret->msg_callback = s->msg_callback;
2694         ret->msg_callback_arg = s->msg_callback_arg;
2695         SSL_set_verify(ret,SSL_get_verify_mode(s),
2696                 SSL_get_verify_callback(s));
2697         SSL_set_verify_depth(ret,SSL_get_verify_depth(s));
2698         ret->generate_session_id = s->generate_session_id;

2700         SSL_set_info_callback(ret,SSL_get_info_callback(s));
```



```
2702         ret->debug=s->debug;

2704         /* copy app data, a little dangerous perhaps */
2705         if (!CRYPTO_dup_ex_data(CRYPTO_EX_INDEX_SSL, &ret->ex_data, &s->ex_data)
2706                 goto err;

2708         /* setup rbio, and wbio */
2709         if (s->rbio != NULL)
2710                 {
2711                 if (!BIO_dup_state(s->rbio,(char *)&ret->rbio))
2712                         goto err;
2713                 }
2714         if (s->wbio != NULL)
2715                 {
2716                 if (s->wbio != s->rbio)
2717                         {
2718                         if (!BIO_dup_state(s->wbio,(char *)&ret->wbio))
2719                                 goto err;
2720                         }
2721                 else
2722                         ret->wbio=ret->rbio;
2723                 }
2724         ret->rwstate = s->rwstate;
2725         ret->in_handshake = s->in_handshake;
2726         ret->handshake_func = s->handshake_func;
2727         ret->server = s->server;
2728         ret->renegotiate = s->renegotiate;
2729         ret->new_session = s->new_session;
2730         ret->quiet_shutdown = s->quiet_shutdown;
2731         ret->shutdown=s->shutdown;
2732         ret->state=s->state; /* SSL_dup does not really work at any state, thoug
2733         ret->rstate=s->rstate;
2734         ret->init_num = 0; /* would have to copy ret->init_buf, ret->init_msg, r
2735         ret->hit=s->hit;

2737         X509_VERIFY_PARAM_inherit(ret->param, s->param);

2739         /* dup the cipher_list and cipher_list_by_id stacks */
2740         if (s->cipher_list != NULL)
2741                 {
2742                 if ((ret->cipher_list=sk_SSL_CIPHER_dup(s->cipher_list)) == NULL
2743                         goto err;
2744                 }
2745         if (s->cipher_list_by_id != NULL)
2746                 if ((ret->cipher_list_by_id=sk_SSL_CIPHER_dup(s->cipher_list_by_
2747                         == NULL)
2748                         goto err;

2750         /* Dup the client_CA list */
2751         if (s->client_CA != NULL)
2752                 {
2753                 if ((sk=sk_X509_NAME_dup(s->client_CA)) == NULL) goto err;
2754                 ret->client_CA=sk;
2755                 for (i=0; i<sk_X509_NAME_num(sk); i++)
2756                         {
2757                         xn=sk_X509_NAME_value(sk,i);
2758                         if (sk_X509_NAME_set(sk,i,X509_NAME_dup(xn)) == NULL)
2759                                 {
2760                                 X509_NAME_free(xn);
2761                                 goto err;
2762                                 }
2763                         }
2764                 }

2766         if (0)
2767                 {
```

```
2768 err:
2769                 if (ret != NULL) SSL_free(ret);
2770                 ret=NULL;
2771                 }
2772         return(ret);
2773         }

2775 void ssl_clear_cipher_ctx(SSL *s)
2776         {
2777         if (s->enc_read_ctx != NULL)
2778                 {
2779                 EVP_CIPHER_CTX_cleanup(s->enc_read_ctx);
2780                 OPENSSL_free(s->enc_read_ctx);
2781                 s->enc_read_ctx=NULL;
2782                 }
2783         if (s->enc_write_ctx != NULL)
2784                 {
2785                 EVP_CIPHER_CTX_cleanup(s->enc_write_ctx);
2786                 OPENSSL_free(s->enc_write_ctx);
2787                 s->enc_write_ctx=NULL;
2788                 }
2789 #ifndef OPENSSL_NO_COMP
2790         if (s->expand != NULL)
2791                 {
2792                 COMP_CTX_free(s->expand);
2793                 s->expand=NULL;
2794                 }
2795         if (s->compress != NULL)
2796                 {
2797                 COMP_CTX_free(s->compress);
2798                 s->compress=NULL;
2799                 }
2800 #endif
2801         }

2803 /* Fix this function so that it takes an optional type parameter */
2804 X509 *SSL_get_certificate(const SSL *s)
2805         {
2806         if (s->cert != NULL)
2807                 return(s->cert->key->x509);
2808         else
2809                 return(NULL);
2810         }

2812 /* Fix this function so that it takes an optional type parameter */
2813 EVP_PKEY *SSL_get_privatekey(SSL *s)
2814         {
2815         if (s->cert != NULL)
2816                 return(s->cert->key->privatekey);
2817         else
2818                 return(NULL);
2819         }

2821 const SSL_CIPHER *SSL_get_current_cipher(const SSL *s)
2822         {
2823         if ((s->session != NULL) && (s->session->cipher != NULL))
2824                 return(s->session->cipher);
2825         return(NULL);
2826         }
2827 #ifdef OPENSSL_NO_COMP
2828 const void *SSL_get_current_compression(SSL *s)
2829         {
2830         return NULL;
2831         }
2832 const void *SSL_get_current_expansion(SSL *s)
2833         {
```



```
2834         return NULL;
2835         }
2836 #else

2838 const COMP_METHOD *SSL_get_current_compression(SSL *s)
2839         {
2840         if (s->compress != NULL)
2841                 return(s->compress->meth);
2842         return(NULL);
2843         }

2845 const COMP_METHOD *SSL_get_current_expansion(SSL *s)
2846         {
2847         if (s->expand != NULL)
2848                 return(s->expand->meth);
2849         return(NULL);
2850         }
2851 #endif

2853 int ssl_init_wbio_buffer(SSL *s,int push)
2854         {
2855         BIO *bbio;

2857         if (s->bbio == NULL)
2858                 {
2859                 bbio=BIO_new(BIO_f_buffer());
2860                 if (bbio == NULL) return(0);
2861                 s->bbio=bbio;
2862                 }
2863         else
2864                 {
2865                 bbio=s->bbio;
2866                 if (s->bbio == s->wbio)
2867                         s->wbio=BIO_pop(s->wbio);
2868                 }
2869         (void)BIO_reset(bbio);
2870 /*      if (!BIO_set_write_buffer_size(bbio,16*1024)) */
2871         if (!BIO_set_read_buffer_size(bbio,1))
2872                 {
2873                 SSLerr(SSL_F_SSL_INIT_WBIO_BUFFER,ERR_R_BUF_LIB);
2874                 return(0);
2875                 }
2876         if (push)
2877                 {
2878                 if (s->wbio != bbio)
2879                         s->wbio=BIO_push(bbio,s->wbio);
2880                 }
2881         else
2882                 {
2883                 if (s->wbio == bbio)
2884                         s->wbio=BIO_pop(bbio);
2885                 }
2886         return(1);
2887         }

2889 void ssl_free_wbio_buffer(SSL *s)
2890         {
2891         if (s->bbio == NULL) return;

2893         if (s->bbio == s->wbio)
2894                 {
2895                 /* remove buffering */
2896                 s->wbio=BIO_pop(s->wbio);
2897 #ifdef REF_CHECK /* not the usual REF_CHECK, but this avoids adding one more pre
2898                 assert(s->wbio != NULL);
2899 #endif
```

```
2900         }
2901         BIO_free(s->bbio);
2902         s->bbio=NULL;
2903         }

2905 void SSL_CTX_set_quiet_shutdown(SSL_CTX *ctx,int mode)
2906         {
2907         ctx->quiet_shutdown=mode;
2908         }

2910 int SSL_CTX_get_quiet_shutdown(const SSL_CTX *ctx)
2911         {
2912         return(ctx->quiet_shutdown);
2913         }

2915 void SSL_set_quiet_shutdown(SSL *s,int mode)
2916         {
2917         s->quiet_shutdown=mode;
2918         }

2920 int SSL_get_quiet_shutdown(const SSL *s)
2921         {
2922         return(s->quiet_shutdown);
2923         }

2925 void SSL_set_shutdown(SSL *s,int mode)
2926         {
2927         s->shutdown=mode;
2928         }

2930 int SSL_get_shutdown(const SSL *s)
2931         {
2932         return(s->shutdown);
2933         }

2935 int SSL_version(const SSL *s)
2936         {
2937         return(s->version);
2938         }

2940 SSL_CTX *SSL_get_SSL_CTX(const SSL *ssl)
2941         {
2942         return(ssl->ctx);
2943         }

2945 SSL_CTX *SSL_set_SSL_CTX(SSL *ssl, SSL_CTX* ctx)
2946         {
2947         if (ssl->ctx == ctx)
2948                 return ssl->ctx;
2949 #ifndef OPENSSL_NO_TLSEXT
2950         if (ctx == NULL)
2951                 ctx = ssl->initial_ctx;
2952 #endif
2953         if (ssl->cert != NULL)
2954                 ssl_cert_free(ssl->cert);
2955         ssl->cert = ssl_cert_dup(ctx->cert);
2956         CRYPTO_add(&ctx->references,1,CRYPTO_LOCK_SSL_CTX);
2957         if (ssl->ctx != NULL)
2958                 SSL_CTX_free(ssl->ctx); /* decrement reference count */
2959         ssl->ctx = ctx;
2960         return(ssl->ctx);
2961         }

2963 #ifndef OPENSSL_NO_STDIO
2964 int SSL_CTX_set_default_verify_paths(SSL_CTX *ctx)
2965         {
```



```
2966         return(X509_STORE_set_default_paths(ctx->cert_store));
2967         }

2969 int SSL_CTX_load_verify_locations(SSL_CTX *ctx, const char *CAfile,
2970                 const char *CApath)
2971         {
2972         return(X509_STORE_load_locations(ctx->cert_store,CAfile,CApath));
2973         }
2974 #endif

2976 void SSL_set_info_callback(SSL *ssl,
2977         void (*cb)(const SSL *ssl,int type,int val))
2978         {
2979         ssl->info_callback=cb;
2980         }

2982 /* One compiler (Diab DCC) doesn't like argument names in returned
2983    function pointer.  */
2984 void (*SSL_get_info_callback(const SSL *ssl))(const SSL * /*ssl*/,int /*type*/,i
2985         {
2986         return ssl->info_callback;
2987         }

2989 int SSL_state(const SSL *ssl)
2990         {
2991         return(ssl->state);
2992         }

2994 void SSL_set_state(SSL *ssl, int state)
2995         {
2996         ssl->state = state;
2997         }

2999 void SSL_set_verify_result(SSL *ssl,long arg)
3000         {
3001         ssl->verify_result=arg;
3002         }

3004 long SSL_get_verify_result(const SSL *ssl)
3005         {
3006         return(ssl->verify_result);
3007         }

3009 int SSL_get_ex_new_index(long argl,void *argp,CRYPTO_EX_new *new_func,
3010                          CRYPTO_EX_dup *dup_func,CRYPTO_EX_free *free_func)
3011         {
3012         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_SSL, argl, argp,
3013                                 new_func, dup_func, free_func);
3014         }

3016 int SSL_set_ex_data(SSL *s,int idx,void *arg)
3017         {
3018         return(CRYPTO_set_ex_data(&s->ex_data,idx,arg));
3019         }

3021 void *SSL_get_ex_data(const SSL *s,int idx)
3022         {
3023         return(CRYPTO_get_ex_data(&s->ex_data,idx));
3024         }

3026 int SSL_CTX_get_ex_new_index(long argl,void *argp,CRYPTO_EX_new *new_func,
3027                              CRYPTO_EX_dup *dup_func,CRYPTO_EX_free *free_func)
3028         {
3029         return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_SSL_CTX, argl, argp,
3030                                 new_func, dup_func, free_func);
3031         }
```

```
3033 int SSL_CTX_set_ex_data(SSL_CTX *s,int idx,void *arg)
3034         {
3035         return(CRYPTO_set_ex_data(&s->ex_data,idx,arg));
3036         }

3038 void *SSL_CTX_get_ex_data(const SSL_CTX *s,int idx)
3039         {
3040         return(CRYPTO_get_ex_data(&s->ex_data,idx));
3041         }

3043 int ssl_ok(SSL *s)
3044         {
3045         return(1);
3046         }

3048 X509_STORE *SSL_CTX_get_cert_store(const SSL_CTX *ctx)
3049         {
3050         return(ctx->cert_store);
3051         }

3053 void SSL_CTX_set_cert_store(SSL_CTX *ctx,X509_STORE *store)
3054         {
3055         if (ctx->cert_store != NULL)
3056                 X509_STORE_free(ctx->cert_store);
3057         ctx->cert_store=store;
3058         }

3060 int SSL_want(const SSL *s)
3061         {
3062         return(s->rwstate);
3063         }

3065 /*!
3066  * \brief Set the callback for generating temporary RSA keys.
3067  * \param ctx the SSL context.
3068  * \param cb the callback
3069  */

3071 #ifndef OPENSSL_NO_RSA
3072 void SSL_CTX_set_tmp_rsa_callback(SSL_CTX *ctx,RSA *(*cb)(SSL *ssl,
3073                                                           int is_export,
3074                                                           int keylength))
3075     {
3076     SSL_CTX_callback_ctrl(ctx,SSL_CTRL_SET_TMP_RSA_CB,(void (*)(void))cb);
3077     }

3079 void SSL_set_tmp_rsa_callback(SSL *ssl,RSA *(*cb)(SSL *ssl,
3080                                                    int is_export,
3081                                                    int keylength))
3082     {
3083     SSL_callback_ctrl(ssl,SSL_CTRL_SET_TMP_RSA_CB,(void (*)(void))cb);
3084     }
3085 #endif

3087 #ifdef DOXYGEN
3088 /*!
3089  * \brief The RSA temporary key callback function.
3090  * \param ssl the SSL session.
3091  * \param is_export \c TRUE if the temp RSA key is for an export ciphersuite.
3092  * \param keylength if \c is_export is \c TRUE, then \c keylength is the size
3093  * of the required key in bits.
3094  * \return the temporary RSA key.
3095  * \sa SSL_CTX_set_tmp_rsa_callback, SSL_set_tmp_rsa_callback
3096  */
```



```
3098 RSA *cb(SSL *ssl,int is_export,int keylength)
3099     {}
3100 #endif

3102 /*!
3103  * \brief Set the callback for generating temporary DH keys.
3104  * \param ctx the SSL context.
3105  * \param dh the callback
3106  */

3108 #ifndef OPENSSL_NO_DH
3109 void SSL_CTX_set_tmp_dh_callback(SSL_CTX *ctx,DH *(*dh)(SSL *ssl,int is_export,
3110                                                          int keylength))
3111         {
3112         SSL_CTX_callback_ctrl(ctx,SSL_CTRL_SET_TMP_DH_CB,(void (*)(void))dh);
3113         }

3115 void SSL_set_tmp_dh_callback(SSL *ssl,DH *(*dh)(SSL *ssl,int is_export,
3116                                                  int keylength))
3117         {
3118         SSL_callback_ctrl(ssl,SSL_CTRL_SET_TMP_DH_CB,(void (*)(void))dh);
3119         }
3120 #endif

3122 #ifndef OPENSSL_NO_ECDH
3123 void SSL_CTX_set_tmp_ecdh_callback(SSL_CTX *ctx,EC_KEY *(*ecdh)(SSL *ssl,int is_
3124                                                                  int keylength))
3125         {
3126         SSL_CTX_callback_ctrl(ctx,SSL_CTRL_SET_TMP_ECDH_CB,(void (*)(void))ecdh)
3127         }

3129 void SSL_set_tmp_ecdh_callback(SSL *ssl,EC_KEY *(*ecdh)(SSL *ssl,int is_export,
3130                                                          int keylength))
3131         {
3132         SSL_callback_ctrl(ssl,SSL_CTRL_SET_TMP_ECDH_CB,(void (*)(void))ecdh);
3133         }
3134 #endif

3136 #ifndef OPENSSL_NO_PSK
3137 int SSL_CTX_use_psk_identity_hint(SSL_CTX *ctx, const char *identity_hint)
3138         {
3139         if (identity_hint != NULL && strlen(identity_hint) > PSK_MAX_IDENTITY_LE
3140                 {
3141                 SSLerr(SSL_F_SSL_CTX_USE_PSK_IDENTITY_HINT, SSL_R_DATA_LENGTH_TO
3142                 return 0;
3143                 }
3144         if (ctx->psk_identity_hint != NULL)
3145                 OPENSSL_free(ctx->psk_identity_hint);
3146         if (identity_hint != NULL)
3147                 {
3148                 ctx->psk_identity_hint = BUF_strdup(identity_hint);
3149                 if (ctx->psk_identity_hint == NULL)
3150                         return 0;
3151                 }
3152         else
3153                 ctx->psk_identity_hint = NULL;
3154         return 1;
3155         }

3157 int SSL_use_psk_identity_hint(SSL *s, const char *identity_hint)
3158         {
3159         if (s == NULL)
3160                 return 0;

3162         if (s->session == NULL)
3163                 return 1; /* session not created yet, ignored */
```

```
3165         if (identity_hint != NULL && strlen(identity_hint) > PSK_MAX_IDENTITY_LE
3166                 {
3167                 SSLerr(SSL_F_SSL_USE_PSK_IDENTITY_HINT, SSL_R_DATA_LENGTH_TOO_LO
3168                 return 0;
3169                 }
3170         if (s->session->psk_identity_hint != NULL)
3171                 OPENSSL_free(s->session->psk_identity_hint);
3172         if (identity_hint != NULL)
3173                 {
3174                 s->session->psk_identity_hint = BUF_strdup(identity_hint);
3175                 if (s->session->psk_identity_hint == NULL)
3176                         return 0;
3177                 }
3178         else
3179                 s->session->psk_identity_hint = NULL;
3180         return 1;
3181         }

3183 const char *SSL_get_psk_identity_hint(const SSL *s)
3184         {
3185         if (s == NULL || s->session == NULL)
3186                 return NULL;
3187         return(s->session->psk_identity_hint);
3188         }

3190 const char *SSL_get_psk_identity(const SSL *s)
3191         {
3192         if (s == NULL || s->session == NULL)
3193                 return NULL;
3194         return(s->session->psk_identity);
3195         }

3197 void SSL_set_psk_client_callback(SSL *s,
3198     unsigned int (*cb)(SSL *ssl, const char *hint,
3199                        char *identity, unsigned int max_identity_len, unsigned c
3200                        unsigned int max_psk_len))
3201         {
3202         s->psk_client_callback = cb;
3203         }

3205 void SSL_CTX_set_psk_client_callback(SSL_CTX *ctx,
3206     unsigned int (*cb)(SSL *ssl, const char *hint,
3207                        char *identity, unsigned int max_identity_len, unsigned c
3208                        unsigned int max_psk_len))
3209         {
3210         ctx->psk_client_callback = cb;
3211         }

3213 void SSL_set_psk_server_callback(SSL *s,
3214     unsigned int (*cb)(SSL *ssl, const char *identity,
3215                        unsigned char *psk, unsigned int max_psk_len))
3216         {
3217         s->psk_server_callback = cb;
3218         }

3220 void SSL_CTX_set_psk_server_callback(SSL_CTX *ctx,
3221     unsigned int (*cb)(SSL *ssl, const char *identity,
3222                        unsigned char *psk, unsigned int max_psk_len))
3223         {
3224         ctx->psk_server_callback = cb;
3225         }
3226 #endif

3228 void SSL_CTX_set_msg_callback(SSL_CTX *ctx, void (*cb)(int write_p, int version,
3229         {
```



```
3230         SSL_CTX_callback_ctrl(ctx, SSL_CTRL_SET_MSG_CALLBACK, (void (*)(void))cb
3231         }
3232 void SSL_set_msg_callback(SSL *ssl, void (*cb)(int write_p, int version, int con
3233         {
3234         SSL_callback_ctrl(ssl, SSL_CTRL_SET_MSG_CALLBACK, (void (*)(void))cb);
3235         }

3237 /* Allocates new EVP_MD_CTX and sets pointer to it into given pointer
3238  * vairable, freeing  EVP_MD_CTX previously stored in that variable, if
3239  * any. If EVP_MD pointer is passed, initializes ctx with this md
3240  * Returns newly allocated ctx;
3241  */

3243 EVP_MD_CTX *ssl_replace_hash(EVP_MD_CTX **hash,const EVP_MD *md)
3244 {
3245         ssl_clear_hash_ctx(hash);
3246         *hash = EVP_MD_CTX_create();
3247         if (md) EVP_DigestInit_ex(*hash,md,NULL);
3248         return *hash;
3249 }
3250 void ssl_clear_hash_ctx(EVP_MD_CTX **hash)
3251 {

3253         if (*hash) EVP_MD_CTX_destroy(*hash);
3254         *hash=NULL;
3255 }

3257 void SSL_set_debug(SSL *s, int debug)
3258         {
3259         s->debug = debug;
3260         }

3262 int SSL_cache_hit(SSL *s)
3263         {
3264         return s->hit;
3265         }

3267 #if defined(_WINDLL) && defined(OPENSSL_SYS_WIN16)
3268 #include "../crypto/bio/bss_file.c"
3269 #endif

3271 IMPLEMENT_STACK_OF(SSL_CIPHER)
3272 IMPLEMENT_STACK_OF(SSL_COMP)
3273 IMPLEMENT_OBJ_BSEARCH_GLOBAL_CMP_FN(SSL_CIPHER, SSL_CIPHER,
3274                                      ssl_cipher_id);
3275 #endif /* ! codereview */
```

```
**********************************************************
   18093 Wed Aug 13 19:53:40 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_rsa.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_rsa.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/bio.h>
```



```
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>
#include <openssl/pem.h>

static int ssl_set_cert(CERT *c, X509 *x509);
static int ssl_set_pkey(CERT *c, EVP_PKEY *pkey);
int SSL_use_certificate(SSL *ssl, X509 *x)
        {
        if (x == NULL)
                {
                SSLerr(SSL_F_SSL_USE_CERTIFICATE,ERR_R_PASSED_NULL_PARAMETER);
                return(0);
                }
        if (!ssl_cert_inst(&ssl->cert))
                {
                SSLerr(SSL_F_SSL_USE_CERTIFICATE,ERR_R_MALLOC_FAILURE);
                return(0);
                }
        return(ssl_set_cert(ssl->cert,x));
        }

#ifndef OPENSSL_NO_STDIO
int SSL_use_certificate_file(SSL *ssl, const char *file, int type)
        {
        int j;
        BIO *in;
        int ret=0;
        X509 *x=NULL;

        in=BIO_new(BIO_s_file_internal());
        if (in == NULL)
                {
                SSLerr(SSL_F_SSL_USE_CERTIFICATE_FILE,ERR_R_BUF_LIB);
                goto end;
                }

        if (BIO_read_filename(in,file) <= 0)
                {
                SSLerr(SSL_F_SSL_USE_CERTIFICATE_FILE,ERR_R_SYS_LIB);
                goto end;
                }
        if (type == SSL_FILETYPE_ASN1)
                {
                j=ERR_R_ASN1_LIB;
                x=d2i_X509_bio(in,NULL);
                }
        else if (type == SSL_FILETYPE_PEM)
                {
                j=ERR_R_PEM_LIB;
                x=PEM_read_bio_X509(in,NULL,ssl->ctx->default_passwd_callback,ss
                }
        else
                {
                SSLerr(SSL_F_SSL_USE_CERTIFICATE_FILE,SSL_R_BAD_SSL_FILETYPE);
                goto end;
                }

        if (x == NULL)
                {
                SSLerr(SSL_F_SSL_USE_CERTIFICATE_FILE,j);
                goto end;
                }

        ret=SSL_use_certificate(ssl,x);
end:
```

```
128         if (x != NULL) X509_free(x);
129         if (in != NULL) BIO_free(in);
130         return(ret);
131         }
132 #endif

134 int SSL_use_certificate_ASN1(SSL *ssl, const unsigned char *d, int len)
135         {
136         X509 *x;
137         int ret;

139         x=d2i_X509(NULL,&d,(long)len);
140         if (x == NULL)
141                 {
142                 SSLerr(SSL_F_SSL_USE_CERTIFICATE_ASN1,ERR_R_ASN1_LIB);
143                 return(0);
144                 }

146         ret=SSL_use_certificate(ssl,x);
147         X509_free(x);
148         return(ret);
149         }

151 #ifndef OPENSSL_NO_RSA
152 int SSL_use_RSAPrivateKey(SSL *ssl, RSA *rsa)
153         {
154         EVP_PKEY *pkey;
155         int ret;

157         if (rsa == NULL)
158                 {
159                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY,ERR_R_PASSED_NULL_PARAMETER);
160                 return(0);
161                 }
162         if (!ssl_cert_inst(&ssl->cert))
163                 {
164                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY,ERR_R_MALLOC_FAILURE);
165                 return(0);
166                 }
167         if ((pkey=EVP_PKEY_new()) == NULL)
168                 {
169                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY,ERR_R_EVP_LIB);
170                 return(0);
171                 }

173         RSA_up_ref(rsa);
174         EVP_PKEY_assign_RSA(pkey,rsa);

176         ret=ssl_set_pkey(ssl->cert,pkey);
177         EVP_PKEY_free(pkey);
178         return(ret);
179         }
180 #endif

182 static int ssl_set_pkey(CERT *c, EVP_PKEY *pkey)
183         {
184         int i;

186         i=ssl_cert_type(NULL,pkey);
187         if (i < 0)
188                 {
189                 SSLerr(SSL_F_SSL_SET_PKEY,SSL_R_UNKNOWN_CERTIFICATE_TYPE);
190                 return(0);
191                 }

193         if (c->pkeys[i].x509 != NULL)
```



```
194                 {
195                 EVP_PKEY *pktmp;
196                 pktmp = X509_get_pubkey(c->pkeys[i].x509);
197                 EVP_PKEY_copy_parameters(pktmp,pkey);
198                 EVP_PKEY_free(pktmp);
199                 ERR_clear_error();

201 #ifndef OPENSSL_NO_RSA
202                 /* Don't check the public/private key, this is mostly
203                  * for smart cards. */
204                 if ((pkey->type == EVP_PKEY_RSA) &&
205                         (RSA_flags(pkey->pkey.rsa) & RSA_METHOD_FLAG_NO_CHECK))
206                         ;
207                 else
208 #endif
209                 if (!X509_check_private_key(c->pkeys[i].x509,pkey))
210                         {
211                         X509_free(c->pkeys[i].x509);
212                         c->pkeys[i].x509 = NULL;
213                         return 0;
214                         }
215                 }

217         if (c->pkeys[i].privatekey != NULL)
218                 EVP_PKEY_free(c->pkeys[i].privatekey);
219         CRYPTO_add(&pkey->references,1,CRYPTO_LOCK_EVP_PKEY);
220         c->pkeys[i].privatekey=pkey;
221         c->key= &(c->pkeys[i]);

223         c->valid=0;
224         return(1);
225         }

227 #ifndef OPENSSL_NO_RSA
228 #ifndef OPENSSL_NO_STDIO
229 int SSL_use_RSAPrivateKey_file(SSL *ssl, const char *file, int type)
230         {
231         int j,ret=0;
232         BIO *in;
233         RSA *rsa=NULL;

235         in=BIO_new(BIO_s_file_internal());
236         if (in == NULL)
237                 {
238                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY_FILE,ERR_R_BUF_LIB);
239                 goto end;
240                 }

242         if (BIO_read_filename(in,file) <= 0)
243                 {
244                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY_FILE,ERR_R_SYS_LIB);
245                 goto end;
246                 }
247         if      (type == SSL_FILETYPE_ASN1)
248                 {
249                 j=ERR_R_ASN1_LIB;
250                 rsa=d2i_RSAPrivateKey_bio(in,NULL);
251                 }
252         else if (type == SSL_FILETYPE_PEM)
253                 {
254                 j=ERR_R_PEM_LIB;
255                 rsa=PEM_read_bio_RSAPrivateKey(in,NULL,
256                         ssl->ctx->default_passwd_callback,ssl->ctx->default_pass
257                 }
258         else
259                 {
```

```
260                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY_FILE,SSL_R_BAD_SSL_FILETYPE);
261                 goto end;
262                 }
263         if (rsa == NULL)
264                 {
265                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY_FILE,j);
266                 goto end;
267                 }
268         ret=SSL_use_RSAPrivateKey(ssl,rsa);
269         RSA_free(rsa);
270 end:
271         if (in != NULL) BIO_free(in);
272         return(ret);
273         }
274 #endif

276 int SSL_use_RSAPrivateKey_ASN1(SSL *ssl, unsigned char *d, long len)
277         {
278         int ret;
279         const unsigned char *p;
280         RSA *rsa;

282         p=d;
283         if ((rsa=d2i_RSAPrivateKey(NULL,&p,(long)len)) == NULL)
284                 {
285                 SSLerr(SSL_F_SSL_USE_RSAPRIVATEKEY_ASN1,ERR_R_ASN1_LIB);
286                 return(0);
287                 }

289         ret=SSL_use_RSAPrivateKey(ssl,rsa);
290         RSA_free(rsa);
291         return(ret);
292         }
293 #endif /* !OPENSSL_NO_RSA */

295 int SSL_use_PrivateKey(SSL *ssl, EVP_PKEY *pkey)
296         {
297         int ret;

299         if (pkey == NULL)
300                 {
301                 SSLerr(SSL_F_SSL_USE_PRIVATEKEY,ERR_R_PASSED_NULL_PARAMETER);
302                 return(0);
303                 }
304         if (!ssl_cert_inst(&ssl->cert))
305                 {
306                 SSLerr(SSL_F_SSL_USE_PRIVATEKEY,ERR_R_MALLOC_FAILURE);
307                 return(0);
308                 }
309         ret=ssl_set_pkey(ssl->cert,pkey);
310         return(ret);
311         }

313 #ifndef OPENSSL_NO_STDIO
314 int SSL_use_PrivateKey_file(SSL *ssl, const char *file, int type)
315         {
316         int j,ret=0;
317         BIO *in;
318         EVP_PKEY *pkey=NULL;

320         in=BIO_new(BIO_s_file_internal());
321         if (in == NULL)
322                 {
323                 SSLerr(SSL_F_SSL_USE_PRIVATEKEY_FILE,ERR_R_BUF_LIB);
324                 goto end;
325                 }
```



```
327         if (BIO_read_filename(in,file) <= 0)
328                 {
329                 SSLerr(SSL_F_SSL_USE_PRIVATEKEY_FILE,ERR_R_SYS_LIB);
330                 goto end;
331                 }
332         if (type == SSL_FILETYPE_PEM)
333                 {
334                 j=ERR_R_PEM_LIB;
335                 pkey=PEM_read_bio_PrivateKey(in,NULL,
336                         ssl->ctx->default_passwd_callback,ssl->ctx->default_pass
337                 }
338         else if (type == SSL_FILETYPE_ASN1)
339                 {
340                 j = ERR_R_ASN1_LIB;
341                 pkey = d2i_PrivateKey_bio(in,NULL);
342                 }
343         else
344                 {
345                 SSLerr(SSL_F_SSL_USE_PRIVATEKEY_FILE,SSL_R_BAD_SSL_FILETYPE);
346                 goto end;
347                 }
348         if (pkey == NULL)
349                 {
350                 SSLerr(SSL_F_SSL_USE_PRIVATEKEY_FILE,j);
351                 goto end;
352                 }
353         ret=SSL_use_PrivateKey(ssl,pkey);
354         EVP_PKEY_free(pkey);
355 end:
356         if (in != NULL) BIO_free(in);
357         return(ret);
358         }
359 #endif

361 int SSL_use_PrivateKey_ASN1(int type, SSL *ssl, const unsigned char *d, long len
362         {
363         int ret;
364         const unsigned char *p;
365         EVP_PKEY *pkey;

367         p=d;
368         if ((pkey=d2i_PrivateKey(type,NULL,&p,(long)len)) == NULL)
369                 {
370                 SSLerr(SSL_F_SSL_USE_PRIVATEKEY_ASN1,ERR_R_ASN1_LIB);
371                 return(0);
372                 }

374         ret=SSL_use_PrivateKey(ssl,pkey);
375         EVP_PKEY_free(pkey);
376         return(ret);
377         }

379 int SSL_CTX_use_certificate(SSL_CTX *ctx, X509 *x)
380         {
381         if (x == NULL)
382                 {
383                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE,ERR_R_PASSED_NULL_PARAMETER
384                 return(0);
385                 }
386         if (!ssl_cert_inst(&ctx->cert))
387                 {
388                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE,ERR_R_MALLOC_FAILURE);
389                 return(0);
390                 }
391         return(ssl_set_cert(ctx->cert, x));
```

```
392         }

394 static int ssl_set_cert(CERT *c, X509 *x)
395         {
396         EVP_PKEY *pkey;
397         int i;

399         pkey=X509_get_pubkey(x);
400         if (pkey == NULL)
401                 {
402                 SSLerr(SSL_F_SSL_SET_CERT,SSL_R_X509_LIB);
403                 return(0);
404                 }

406         i=ssl_cert_type(x,pkey);
407         if (i < 0)
408                 {
409                 SSLerr(SSL_F_SSL_SET_CERT,SSL_R_UNKNOWN_CERTIFICATE_TYPE);
410                 EVP_PKEY_free(pkey);
411                 return(0);
412                 }

414         if (c->pkeys[i].privatekey != NULL)
415                 {
416                 EVP_PKEY_copy_parameters(pkey,c->pkeys[i].privatekey);
417                 ERR_clear_error();

419 #ifndef OPENSSL_NO_RSA
420                 /* Don't check the public/private key, this is mostly
421                  * for smart cards. */
422                 if ((c->pkeys[i].privatekey->type == EVP_PKEY_RSA) &&
423                         (RSA_flags(c->pkeys[i].privatekey->pkey.rsa) &
424                          RSA_METHOD_FLAG_NO_CHECK))
425                          ;
426                 else
427 #endif /* OPENSSL_NO_RSA */
428                 if (!X509_check_private_key(x,c->pkeys[i].privatekey))
429                         {
430                         /* don't fail for a cert/key mismatch, just free
431                          * current private key (when switching to a different
432                          * cert & key, first this function should be used,
433                          * then ssl_set_pkey */
434                         EVP_PKEY_free(c->pkeys[i].privatekey);
435                         c->pkeys[i].privatekey=NULL;
436                         /* clear error queue */
437                         ERR_clear_error();
438                         }
439                 }

441         EVP_PKEY_free(pkey);

443         if (c->pkeys[i].x509 != NULL)
444                 X509_free(c->pkeys[i].x509);
445         CRYPTO_add(&x->references,1,CRYPTO_LOCK_X509);
446         c->pkeys[i].x509=x;
447         c->key= &(c->pkeys[i]);

449         c->valid=0;
450         return(1);
451         }

453 #ifndef OPENSSL_NO_STDIO
454 int SSL_CTX_use_certificate_file(SSL_CTX *ctx, const char *file, int type)
455         {
456         int j;
457         BIO *in;
```



```
458         int ret=0;
459         X509 *x=NULL;

461         in=BIO_new(BIO_s_file_internal());
462         if (in == NULL)
463                 {
464                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_FILE,ERR_R_BUF_LIB);
465                 goto end;
466                 }

468         if (BIO_read_filename(in,file) <= 0)
469                 {
470                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_FILE,ERR_R_SYS_LIB);
471                 goto end;
472                 }
473         if (type == SSL_FILETYPE_ASN1)
474                 {
475                 j=ERR_R_ASN1_LIB;
476                 x=d2i_X509_bio(in,NULL);
477                 }
478         else if (type == SSL_FILETYPE_PEM)
479                 {
480                 j=ERR_R_PEM_LIB;
481                 x=PEM_read_bio_X509(in,NULL,ctx->default_passwd_callback,ctx->de
482                 }
483         else
484                 {
485                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_FILE,SSL_R_BAD_SSL_FILETYPE
486                 goto end;
487                 }

489         if (x == NULL)
490                 {
491                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_FILE,j);
492                 goto end;
493                 }

495         ret=SSL_CTX_use_certificate(ctx,x);
496 end:
497         if (x != NULL) X509_free(x);
498         if (in != NULL) BIO_free(in);
499         return(ret);
500         }
501 #endif

503 int SSL_CTX_use_certificate_ASN1(SSL_CTX *ctx, int len, const unsigned char *d)
504         {
505         X509 *x;
506         int ret;

508         x=d2i_X509(NULL,&d,(long)len);
509         if (x == NULL)
510                 {
511                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_ASN1,ERR_R_ASN1_LIB);
512                 return(0);
513                 }

515         ret=SSL_CTX_use_certificate(ctx,x);
516         X509_free(x);
517         return(ret);
518         }

520 #ifndef OPENSSL_NO_RSA
521 int SSL_CTX_use_RSAPrivateKey(SSL_CTX *ctx, RSA *rsa)
522         {
523         int ret;
```

```
524         EVP_PKEY *pkey;

526         if (rsa == NULL)
527                 {
528                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY,ERR_R_PASSED_NULL_PARAMET
529                 return(0);
530                 }
531         if (!ssl_cert_inst(&ctx->cert))
532                 {
533                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY,ERR_R_MALLOC_FAILURE);
534                 return(0);
535                 }
536         if ((pkey=EVP_PKEY_new()) == NULL)
537                 {
538                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY,ERR_R_EVP_LIB);
539                 return(0);
540                 }

542         RSA_up_ref(rsa);
543         EVP_PKEY_assign_RSA(pkey,rsa);

545         ret=ssl_set_pkey(ctx->cert, pkey);
546         EVP_PKEY_free(pkey);
547         return(ret);
548         }

550 #ifndef OPENSSL_NO_STDIO
551 int SSL_CTX_use_RSAPrivateKey_file(SSL_CTX *ctx, const char *file, int type)
552         {
553         int j,ret=0;
554         BIO *in;
555         RSA *rsa=NULL;

557         in=BIO_new(BIO_s_file_internal());
558         if (in == NULL)
559                 {
560                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_FILE,ERR_R_BUF_LIB);
561                 goto end;
562                 }

564         if (BIO_read_filename(in,file) <= 0)
565                 {
566                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_FILE,ERR_R_SYS_LIB);
567                 goto end;
568                 }
569         if      (type == SSL_FILETYPE_ASN1)
570                 {
571                 j=ERR_R_ASN1_LIB;
572                 rsa=d2i_RSAPrivateKey_bio(in,NULL);
573                 }
574         else if (type == SSL_FILETYPE_PEM)
575                 {
576                 j=ERR_R_PEM_LIB;
577                 rsa=PEM_read_bio_RSAPrivateKey(in,NULL,
578                         ctx->default_passwd_callback,ctx->default_passwd_callbac
579                 }
580         else
581                 {
582                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_FILE,SSL_R_BAD_SSL_FILETY
583                 goto end;
584                 }
585         if (rsa == NULL)
586                 {
587                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_FILE,j);
588                 goto end;
589                 }
```



```
590         ret=SSL_CTX_use_RSAPrivateKey(ctx,rsa);
591         RSA_free(rsa);
592 end:
593         if (in != NULL) BIO_free(in);
594         return(ret);
595         }
596 #endif

598 int SSL_CTX_use_RSAPrivateKey_ASN1(SSL_CTX *ctx, const unsigned char *d, long le
599         {
600         int ret;
601         const unsigned char *p;
602         RSA *rsa;

604         p=d;
605         if ((rsa=d2i_RSAPrivateKey(NULL,&p,(long)len)) == NULL)
606                 {
607                 SSLerr(SSL_F_SSL_CTX_USE_RSAPRIVATEKEY_ASN1,ERR_R_ASN1_LIB);
608                 return(0);
609                 }

611         ret=SSL_CTX_use_RSAPrivateKey(ctx,rsa);
612         RSA_free(rsa);
613         return(ret);
614         }
615 #endif /* !OPENSSL_NO_RSA */

617 int SSL_CTX_use_PrivateKey(SSL_CTX *ctx, EVP_PKEY *pkey)
618         {
619         if (pkey == NULL)
620                 {
621                 SSLerr(SSL_F_SSL_CTX_USE_PRIVATEKEY,ERR_R_PASSED_NULL_PARAMETER)
622                 return(0);
623                 }
624         if (!ssl_cert_inst(&ctx->cert))
625                 {
626                 SSLerr(SSL_F_SSL_CTX_USE_PRIVATEKEY,ERR_R_MALLOC_FAILURE);
627                 return(0);
628                 }
629         return(ssl_set_pkey(ctx->cert,pkey));
630         }

632 #ifndef OPENSSL_NO_STDIO
633 int SSL_CTX_use_PrivateKey_file(SSL_CTX *ctx, const char *file, int type)
634         {
635         int j,ret=0;
636         BIO *in;
637         EVP_PKEY *pkey=NULL;

639         in=BIO_new(BIO_s_file_internal());
640         if (in == NULL)
641                 {
642                 SSLerr(SSL_F_SSL_CTX_USE_PRIVATEKEY_FILE,ERR_R_BUF_LIB);
643                 goto end;
644                 }

646         if (BIO_read_filename(in,file) <= 0)
647                 {
648                 SSLerr(SSL_F_SSL_CTX_USE_PRIVATEKEY_FILE,ERR_R_SYS_LIB);
649                 goto end;
650                 }
651         if (type == SSL_FILETYPE_PEM)
652                 {
653                 j=ERR_R_PEM_LIB;
654                 pkey=PEM_read_bio_PrivateKey(in,NULL,
655                         ctx->default_passwd_callback,ctx->default_passwd_callbac
```

```
656                 }
657         else if (type == SSL_FILETYPE_ASN1)
658                 {
659                 j = ERR_R_ASN1_LIB;
660                 pkey = d2i_PrivateKey_bio(in,NULL);
661                 }
662         else
663                 {
664                 SSLerr(SSL_F_SSL_CTX_USE_PRIVATEKEY_FILE,SSL_R_BAD_SSL_FILETYPE)
665                 goto end;
666                 }
667         if (pkey == NULL)
668                 {
669                 SSLerr(SSL_F_SSL_CTX_USE_PRIVATEKEY_FILE,j);
670                 goto end;
671                 }
672         ret=SSL_CTX_use_PrivateKey(ctx,pkey);
673         EVP_PKEY_free(pkey);
674 end:
675         if (in != NULL) BIO_free(in);
676         return(ret);
677         }
678 #endif

680 int SSL_CTX_use_PrivateKey_ASN1(int type, SSL_CTX *ctx, const unsigned char *d,
681              long len)
682         {
683         int ret;
684         const unsigned char *p;
685         EVP_PKEY *pkey;

687         p=d;
688         if ((pkey=d2i_PrivateKey(type,NULL,&p,(long)len)) == NULL)
689                 {
690                 SSLerr(SSL_F_SSL_CTX_USE_PRIVATEKEY_ASN1,ERR_R_ASN1_LIB);
691                 return(0);
692                 }

694         ret=SSL_CTX_use_PrivateKey(ctx,pkey);
695         EVP_PKEY_free(pkey);
696         return(ret);
697         }


700 #ifndef OPENSSL_NO_STDIO
701 /* Read a file that contains our certificate in "PEM" format,
702  * possibly followed by a sequence of CA certificates that should be
703  * sent to the peer in the Certificate message.
704  */
705 int SSL_CTX_use_certificate_chain_file(SSL_CTX *ctx, const char *file)
706         {
707         BIO *in;
708         int ret=0;
709         X509 *x=NULL;

711         ERR_clear_error(); /* clear error stack for SSL_CTX_use_certificate() */

713         in = BIO_new(BIO_s_file_internal());
714         if (in == NULL)
715                 {
716                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_CHAIN_FILE,ERR_R_BUF_LIB);
717                 goto end;
718                 }

720         if (BIO_read_filename(in,file) <= 0)
721                 {
```



```
722                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_CHAIN_FILE,ERR_R_SYS_LIB);
723                 goto end;
724                 }

726         x=PEM_read_bio_X509_AUX(in,NULL,ctx->default_passwd_callback,
727                                 ctx->default_passwd_callback_userdata);
728         if (x == NULL)
729                 {
730                 SSLerr(SSL_F_SSL_CTX_USE_CERTIFICATE_CHAIN_FILE,ERR_R_PEM_LIB);
731                 goto end;
732                 }

734         ret = SSL_CTX_use_certificate(ctx, x);

736         if (ERR_peek_error() != 0)
737                 ret = 0;  /* Key/certificate mismatch doesn't imply ret==0 ... *
738         if (ret)
739                 {
740                 /* If we could set up our certificate, now proceed to
741                  * the CA certificates.
742                  */
743                 X509 *ca;
744                 int r;
745                 unsigned long err;

747                 if (ctx->extra_certs != NULL)
748                         {
749                         sk_X509_pop_free(ctx->extra_certs, X509_free);
750                         ctx->extra_certs = NULL;
751                         }

753                 while ((ca = PEM_read_bio_X509(in, NULL,
754                                                ctx->default_passwd_callback,
755                                                ctx->default_passwd_callback_userdata))
756                         != NULL)
757                         {
758                         r = SSL_CTX_add_extra_chain_cert(ctx, ca);
759                         if (!r)
760                                 {
761                                 X509_free(ca);
762                                 ret = 0;
763                                 goto end;
764                                 }
765                         /* Note that we must not free r if it was successfully
766                          * added to the chain (while we must free the main
767                          * certificate, since its reference count is increased
768                          * by SSL_CTX_use_certificate). */
769                         }
770                 /* When the while loop ends, it's usually just EOF. */
771                 err = ERR_peek_last_error();
772                 if (ERR_GET_LIB(err) == ERR_LIB_PEM && ERR_GET_REASON(err) == PE
773                         ERR_clear_error();
774                 else
775                         ret = 0; /* some real error */
776                 }

778 end:
779         if (x != NULL) X509_free(x);
780         if (in != NULL) BIO_free(in);
781         return(ret);
782         }
783 #endif
784 #endif /* ! codereview */
```

```
**********************************************************
   34089 Wed Aug 13 19:53:41 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_sess.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/ssl_sess.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2006 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
```

```
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include <openssl/lhash.h>
#include <openssl/rand.h>
#ifndef OPENSSL_NO_ENGINE
#include <openssl/engine.h>
#endif
#include "ssl_locl.h"

static void SSL_SESSION_list_remove(SSL_CTX *ctx, SSL_SESSION *s);
static void SSL_SESSION_list_add(SSL_CTX *ctx,SSL_SESSION *s);
static int remove_session_lock(SSL_CTX *ctx, SSL_SESSION *c, int lck);

SSL_SESSION *SSL_get_session(const SSL *ssl)
/* aka SSL_get0_session; gets 0 objects, just returns a copy of the pointer */
        {
        return(ssl->session);
        }

SSL_SESSION *SSL_get1_session(SSL *ssl)
/* variant of SSL_get_session: caller really gets something */
        {
        SSL_SESSION *sess;
        /* Need to lock this all up rather than just use CRYPTO_add so that
         * somebody doesn't free ssl->session between when we check it's
         * non-null and when we up the reference count. */
        CRYPTO_w_lock(CRYPTO_LOCK_SSL_SESSION);
        sess = ssl->session;
        if(sess)
                sess->references++;
        CRYPTO_w_unlock(CRYPTO_LOCK_SSL_SESSION);
        return(sess);
        }

int SSL_SESSION_get_ex_new_index(long argl, void *argp, CRYPTO_EX_new *new_func,
             CRYPTO_EX_dup *dup_func, CRYPTO_EX_free *free_func)
        {
        return CRYPTO_get_ex_new_index(CRYPTO_EX_INDEX_SSL_SESSION, argl, argp,
                        new_func, dup_func, free_func);
        }

int SSL_SESSION_set_ex_data(SSL_SESSION *s, int idx, void *arg)
        {
        return(CRYPTO_set_ex_data(&s->ex_data,idx,arg));
        }

void *SSL_SESSION_get_ex_data(const SSL_SESSION *s, int idx)
        {
        return(CRYPTO_get_ex_data(&s->ex_data,idx));
        }

SSL_SESSION *SSL_SESSION_new(void)
        {
        SSL_SESSION *ss;

        ss=(SSL_SESSION *)OPENSSL_malloc(sizeof(SSL_SESSION));
        if (ss == NULL)
```



```
                {
                SSLerr(SSL_F_SSL_SESSION_NEW,ERR_R_MALLOC_FAILURE);
                return(0);
                }
        memset(ss,0,sizeof(SSL_SESSION));

        ss->verify_result = 1; /* avoid 0 (= X509_V_OK) just in case */
        ss->references=1;
        ss->timeout=60*5+4; /* 5 minute timeout by default */
        ss->time=(unsigned long)time(NULL);
        ss->prev=NULL;
        ss->next=NULL;
        ss->compress_meth=0;
#ifndef OPENSSL_NO_TLSEXT
        ss->tlsext_hostname = NULL;
#ifndef OPENSSL_NO_EC
        ss->tlsext_ecpointformatlist_length = 0;
        ss->tlsext_ecpointformatlist = NULL;
        ss->tlsext_ellipticcurvelist_length = 0;
        ss->tlsext_ellipticcurvelist = NULL;
#endif
#endif
        CRYPTO_new_ex_data(CRYPTO_EX_INDEX_SSL_SESSION, ss, &ss->ex_data);
#ifndef OPENSSL_NO_PSK
        ss->psk_identity_hint=NULL;
        ss->psk_identity=NULL;
#endif
#ifndef OPENSSL_NO_SRP
        ss->srp_username=NULL;
#endif
        return(ss);
        }

const unsigned char *SSL_SESSION_get_id(const SSL_SESSION *s, unsigned int *len)
        {
        if(len)
                *len = s->session_id_length;
        return s->session_id;
        }

unsigned int SSL_SESSION_get_compress_id(const SSL_SESSION *s)
        {
        return s->compress_meth;
        }

/* Even with SSLv2, we have 16 bytes (128 bits) of session ID space. SSLv3/TLSv1
 * has 32 bytes (256 bits). As such, filling the ID with random gunk repeatedly
 * until we have no conflict is going to complete in one iteration pretty much
 * "most" of the time (btw: understatement). So, if it takes us 10 iterations
 * and we still can't avoid a conflict - well that's a reasonable point to call
 * it quits. Either the RAND code is broken or someone is trying to open roughly
 * very close to 2^128 (or 2^256) SSL sessions to our server. How you might
 * store that many sessions is perhaps a more interesting question ... */

#define MAX_SESS_ID_ATTEMPTS 10
static int def_generate_session_id(const SSL *ssl, unsigned char *id,
                                   unsigned int *id_len)
{
        unsigned int retry = 0;
        do
                if (RAND_pseudo_bytes(id, *id_len) <= 0)
                        return 0;
        while(SSL_has_matching_session_id(ssl, id, *id_len) &&
                (++retry < MAX_SESS_ID_ATTEMPTS));
        if(retry < MAX_SESS_ID_ATTEMPTS)
                return 1;
```

```
260         /* else - woops a session_id match */
261         /* XXX We should also check the external cache --
262          * but the probability of a collision is negligible, and
263          * we could not prevent the concurrent creation of sessions
264          * with identical IDs since we currently don't have means
265          * to atomically check whether a session ID already exists
266          * and make a reservation for it if it does not
267          * (this problem applies to the internal cache as well).
268          */
269         return 0;
270 }

272 int ssl_get_new_session(SSL *s, int session)
273         {
274         /* This gets used by clients and servers. */

276         unsigned int tmp;
277         SSL_SESSION *ss=NULL;
278         GEN_SESSION_CB cb = def_generate_session_id;

280         if ((ss=SSL_SESSION_new()) == NULL) return(0);

282         /* If the context has a default timeout, use it */
283         if (s->session_ctx->session_timeout == 0)
284                 ss->timeout=SSL_get_default_timeout(s);
285         else
286                 ss->timeout=s->session_ctx->session_timeout;

288         if (s->session != NULL)
289                 {
290                 SSL_SESSION_free(s->session);
291                 s->session=NULL;
292                 }

294         if (session)
295                 {
296                 if (s->version == SSL2_VERSION)
297                         {
298                         ss->ssl_version=SSL2_VERSION;
299                         ss->session_id_length=SSL2_SSL_SESSION_ID_LENGTH;
300                         }
301                 else if (s->version == SSL3_VERSION)
302                         {
303                         ss->ssl_version=SSL3_VERSION;
304                         ss->session_id_length=SSL3_SSL_SESSION_ID_LENGTH;
305                         }
306                 else if (s->version == TLS1_VERSION)
307                         {
308                         ss->ssl_version=TLS1_VERSION;
309                         ss->session_id_length=SSL3_SSL_SESSION_ID_LENGTH;
310                         }
311                 else if (s->version == TLS1_1_VERSION)
312                         {
313                         ss->ssl_version=TLS1_1_VERSION;
314                         ss->session_id_length=SSL3_SSL_SESSION_ID_LENGTH;
315                         }
316                 else if (s->version == TLS1_2_VERSION)
317                         {
318                         ss->ssl_version=TLS1_2_VERSION;
319                         ss->session_id_length=SSL3_SSL_SESSION_ID_LENGTH;
320                         }
321                 else if (s->version == DTLS1_BAD_VER)
322                         {
323                         ss->ssl_version=DTLS1_BAD_VER;
324                         ss->session_id_length=SSL3_SSL_SESSION_ID_LENGTH;
325                         }
```



```
326                 else if (s->version == DTLS1_VERSION)
327                         {
328                         ss->ssl_version=DTLS1_VERSION;
329                         ss->session_id_length=SSL3_SSL_SESSION_ID_LENGTH;
330                         }
331                 else
332                         {
333                         SSLerr(SSL_F_SSL_GET_NEW_SESSION,SSL_R_UNSUPPORTED_SSL_V
334                         SSL_SESSION_free(ss);
335                         return(0);
336                         }
337 #ifndef OPENSSL_NO_TLSEXT
338                 /* If RFC4507 ticket use empty session ID */
339                 if (s->tlsext_ticket_expected)
340                         {
341                         ss->session_id_length = 0;
342                         goto sess_id_done;
343                         }
344 #endif
345                 /* Choose which callback will set the session ID */
346                 CRYPTO_r_lock(CRYPTO_LOCK_SSL_CTX);
347                 if(s->generate_session_id)
348                         cb = s->generate_session_id;
349                 else if(s->session_ctx->generate_session_id)
350                         cb = s->session_ctx->generate_session_id;
351                 CRYPTO_r_unlock(CRYPTO_LOCK_SSL_CTX);
352                 /* Choose a session ID */
353                 tmp = ss->session_id_length;
354                 if(!cb(s, ss->session_id, &tmp))
355                         {
356                         /* The callback failed */
357                         SSLerr(SSL_F_SSL_GET_NEW_SESSION,
358                                 SSL_R_SSL_SESSION_ID_CALLBACK_FAILED);
359                         SSL_SESSION_free(ss);
360                         return(0);
361                         }
362                 /* Don't allow the callback to set the session length to zero.
363                  * nor set it higher than it was. */
364                 if(!tmp || (tmp > ss->session_id_length))
365                         {
366                         /* The callback set an illegal length */
367                         SSLerr(SSL_F_SSL_GET_NEW_SESSION,
368                                 SSL_R_SSL_SESSION_ID_HAS_BAD_LENGTH);
369                         SSL_SESSION_free(ss);
370                         return(0);
371                         }
372                 /* If the session length was shrunk and we're SSLv2, pad it */
373                 if((tmp < ss->session_id_length) && (s->version == SSL2_VERSION)
374                         memset(ss->session_id + tmp, 0, ss->session_id_length -
375                 else
376                         ss->session_id_length = tmp;
377                 /* Finally, check for a conflict */
378                 if(SSL_has_matching_session_id(s, ss->session_id,
379                                                ss->session_id_length))
380                         {
381                         SSLerr(SSL_F_SSL_GET_NEW_SESSION,
382                                 SSL_R_SSL_SESSION_ID_CONFLICT);
383                         SSL_SESSION_free(ss);
384                         return(0);
385                         }
386 #ifndef OPENSSL_NO_TLSEXT
387                 sess_id_done:
388                 if (s->tlsext_hostname) {
389                         ss->tlsext_hostname = BUF_strdup(s->tlsext_hostname);
390                         if (ss->tlsext_hostname == NULL) {
391                                 SSLerr(SSL_F_SSL_GET_NEW_SESSION, ERR_R_INTERNAL
```

```
392                                 SSL_SESSION_free(ss);
393                                 return 0;
394                                 }
395                         }
396 #ifndef OPENSSL_NO_EC
397                 if (s->tlsext_ecpointformatlist)
398                         {
399                         if (ss->tlsext_ecpointformatlist != NULL) OPENSSL_free(s
400                         if ((ss->tlsext_ecpointformatlist = OPENSSL_malloc(s->tl
401                                 {
402                                 SSLerr(SSL_F_SSL_GET_NEW_SESSION, ERR_R_MALLOC_F
403                                 SSL_SESSION_free(ss);
404                                 return 0;
405                                 }
406                         ss->tlsext_ecpointformatlist_length = s->tlsext_ecpointf
407                         memcpy(ss->tlsext_ecpointformatlist, s->tlsext_ecpointfo
408                         }
409                 if (s->tlsext_ellipticcurvelist)
410                         {
411                         if (ss->tlsext_ellipticcurvelist != NULL) OPENSSL_free(s
412                         if ((ss->tlsext_ellipticcurvelist = OPENSSL_malloc(s->tl
413                                 {
414                                 SSLerr(SSL_F_SSL_GET_NEW_SESSION, ERR_R_MALLOC_F
415                                 SSL_SESSION_free(ss);
416                                 return 0;
417                                 }
418                         ss->tlsext_ellipticcurvelist_length = s->tlsext_elliptic
419                         memcpy(ss->tlsext_ellipticcurvelist, s->tlsext_ellipticc
420                         }
421 #endif
422 #endif
423                 }
424         else
425                 {
426                 ss->session_id_length=0;
427                 }

429         if (s->sid_ctx_length > sizeof ss->sid_ctx)
430                 {
431                 SSLerr(SSL_F_SSL_GET_NEW_SESSION, ERR_R_INTERNAL_ERROR);
432                 SSL_SESSION_free(ss);
433                 return 0;
434                 }
435         memcpy(ss->sid_ctx,s->sid_ctx,s->sid_ctx_length);
436         ss->sid_ctx_length=s->sid_ctx_length;
437         s->session=ss;
438         ss->ssl_version=s->version;
439         ss->verify_result = X509_V_OK;

441         return(1);
442         }

444 /* ssl_get_prev attempts to find an SSL_SESSION to be used to resume this
445  * connection. It is only called by servers.
446  *
447  *   session_id: points at the session ID in the ClientHello. This code will
448  *       read past the end of this in order to parse out the session ticket
449  *       extension, if any.
450  *   len: the length of the session ID.
451  *   limit: a pointer to the first byte after the ClientHello.
452  *
453  * Returns:
454  *   -1: error
455  *    0: a session may have been found.
456  *
457  * Side effects:
```



```
458  *   - If a session is found then s->session is pointed at it (after freeing an
459  *     existing session if need be) and s->verify_result is set from the session
460  *   - Both for new and resumed sessions, s->tlsext_ticket_expected is set to 1
461  *     if the server should issue a new session ticket (to 0 otherwise).
462  */
463 int ssl_get_prev_session(SSL *s, unsigned char *session_id, int len,
464                          const unsigned char *limit)
465         {
466         /* This is used only by servers. */

468         SSL_SESSION *ret=NULL;
469         int fatal = 0;
470         int try_session_cache = 1;
471 #ifndef OPENSSL_NO_TLSEXT
472         int r;
473 #endif

475         if (len > SSL_MAX_SSL_SESSION_ID_LENGTH)
476                 goto err;

478         if (len == 0)
479                 try_session_cache = 0;

481 #ifndef OPENSSL_NO_TLSEXT
482         r = tls1_process_ticket(s, session_id, len, limit, &ret); /* sets s->tls
483         switch (r)
484                 {
485         case -1: /* Error during processing */
486                 fatal = 1;
487                 goto err;
488         case 0: /* No ticket found */
489         case 1: /* Zero length ticket found */
490                 break; /* Ok to carry on processing session id. */
491         case 2: /* Ticket found but not decrypted. */
492         case 3: /* Ticket decrypted, *ret has been set. */
493                 try_session_cache = 0;
494                 break;
495         default:
496                 abort();
497                 }
498 #endif

500         if (try_session_cache &&
501             ret == NULL &&
502             !(s->session_ctx->session_cache_mode & SSL_SESS_CACHE_NO_INTERNAL_LO
503                 {
504                 SSL_SESSION data;
505                 data.ssl_version=s->version;
506                 data.session_id_length=len;
507                 if (len == 0)
508                         return 0;
509                 memcpy(data.session_id,session_id,len);
510                 CRYPTO_r_lock(CRYPTO_LOCK_SSL_CTX);
511                 ret=lh_SSL_SESSION_retrieve(s->session_ctx->sessions,&data);
512                 if (ret != NULL)
513                         {
514                         /* don't allow other threads to steal it: */
515                         CRYPTO_add(&ret->references,1,CRYPTO_LOCK_SSL_SESSION);
516                         }
517                 CRYPTO_r_unlock(CRYPTO_LOCK_SSL_CTX);
518                 if (ret == NULL)
519                         s->session_ctx->stats.sess_miss++;
520                 }

522         if (try_session_cache &&
523             ret == NULL &&
```

```
524             s->session_ctx->get_session_cb != NULL)
525                 {
526                 int copy=1;
527
528                 if ((ret=s->session_ctx->get_session_cb(s,session_id,len,&copy))
529                         {
530                         s->session_ctx->stats.sess_cb_hit++;
531
532                         /* Increment reference count now if the session callback
533                          * asks us to do so (note that if the session structures
534                          * returned by the callback are shared between threads,
535                          * it must handle the reference count itself [i.e. copy
536                          * or things won't be thread-safe). */
537                         if (copy)
538                                 CRYPTO_add(&ret->references,1,CRYPTO_LOCK_SSL_SE
539
540                         /* Add the externally cached session to the internal
541                          * cache as well if and only if we are supposed to. */
542                         if(!(s->session_ctx->session_cache_mode & SSL_SESS_CACHE
543                                 /* The following should not return 1, otherwise,
544                                  * things are very strange */
545                                 SSL_CTX_add_session(s->session_ctx,ret);
546                         }
547                 }
548
549         if (ret == NULL)
550                 goto err;
551
552         /* Now ret is non-NULL and we own one of its reference counts. */
553
554         if (ret->sid_ctx_length != s->sid_ctx_length
555             || memcmp(ret->sid_ctx,s->sid_ctx,ret->sid_ctx_length))
556                 {
557                 /* We have the session requested by the client, but we don't
558                  * want to use it in this context. */
559                 goto err; /* treat like cache miss */
560                 }
561
562         if((s->verify_mode & SSL_VERIFY_PEER) && s->sid_ctx_length == 0)
563                 {
564                 /* We can't be sure if this session is being used out of
565                  * context, which is especially important for SSL_VERIFY_PEER.
566                  * The application should have used SSL[_CTX]_set_session_id_con
567                  *
568                  * For this error case, we generate an error instead of treating
569                  * the event like a cache miss (otherwise it would be easy for
570                  * applications to effectively disable the session cache by
571                  * accident without anyone noticing).
572                  */
573
574                 SSLerr(SSL_F_SSL_GET_PREV_SESSION,SSL_R_SESSION_ID_CONTEXT_UNINI
575                 fatal = 1;
576                 goto err;
577                 }
578
579         if (ret->cipher == NULL)
580                 {
581                 unsigned char buf[5],*p;
582                 unsigned long l;
583
584                 p=buf;
585                 l=ret->cipher_id;
586                 l2n(l,p);
587                 if ((ret->ssl_version>>8) >= SSL3_VERSION_MAJOR)
588                         ret->cipher=ssl_get_cipher_by_char(s,&(buf[2]));
589                 else
```



```
590                         ret->cipher=ssl_get_cipher_by_char(s,&(buf[1]));
591                 if (ret->cipher == NULL)
592                         goto err;
593                 }
594
595         if (ret->timeout < (long)(time(NULL) - ret->time)) /* timeout */
596                 {
597                 s->session_ctx->stats.sess_timeout++;
598                 if (try_session_cache)
599                         {
600                         /* session was from the cache, so remove it */
601                         SSL_CTX_remove_session(s->session_ctx,ret);
602                         }
603                 goto err;
604                 }
605
606         s->session_ctx->stats.sess_hit++;
607
608         if (s->session != NULL)
609                 SSL_SESSION_free(s->session);
610         s->session=ret;
611         s->verify_result = s->session->verify_result;
612         return 1;
613
614  err:
615         if (ret != NULL)
616                 {
617                 SSL_SESSION_free(ret);
618 #ifndef OPENSSL_NO_TLSEXT
619                 if (!try_session_cache)
620                         {
621                         /* The session was from a ticket, so we should
622                          * issue a ticket for the new session */
623                         s->tlsext_ticket_expected = 1;
624                         }
625 #endif
626                 }
627         if (fatal)
628                 return -1;
629         else
630                 return 0;
631         }
632
633 int SSL_CTX_add_session(SSL_CTX *ctx, SSL_SESSION *c)
634         {
635         int ret=0;
636         SSL_SESSION *s;
637
638         /* add just 1 reference count for the SSL_CTX's session cache
639          * even though it has two ways of access: each session is in a
640          * doubly linked list and an lhash */
641         CRYPTO_add(&c->references,1,CRYPTO_LOCK_SSL_SESSION);
642         /* if session c is in already in cache, we take back the increment later
643
644         CRYPTO_w_lock(CRYPTO_LOCK_SSL_CTX);
645         s=lh_SSL_SESSION_insert(ctx->sessions,c);
646
647         /* s != NULL iff we already had a session with the given PID.
648          * In this case, s == c should hold (then we did not really modify
649          * ctx->sessions), or we're in trouble. */
650         if (s != NULL && s != c)
651                 {
652                 /* We *are* in trouble ... */
653                 SSL_SESSION_list_remove(ctx,s);
654                 SSL_SESSION_free(s);
655                 /* ... so pretend the other session did not exist in cache
```

```
656                  * (we cannot handle two SSL_SESSION structures with identical
657                  * session ID in the same cache, which could happen e.g. when
658                  * two threads concurrently obtain the same session from an exte
659                  * cache) */
660                 s = NULL;
661                 }

663         /* Put at the head of the queue unless it is already in the cache */
664         if (s == NULL)
665                 SSL_SESSION_list_add(ctx,c);

667         if (s != NULL)
668                 {
669                 /* existing cache entry -- decrement previously incremented refe
670                  * count because it already takes into account the cache */

672                 SSL_SESSION_free(s); /* s == c */
673                 ret=0;
674                 }
675         else
676                 {
677                 /* new cache entry -- remove old ones if cache has become too la

679                 ret=1;

681                 if (SSL_CTX_sess_get_cache_size(ctx) > 0)
682                         {
683                         while (SSL_CTX_sess_number(ctx) >
684                                SSL_CTX_sess_get_cache_size(ctx))
685                                 {
686                                 if (!remove_session_lock(ctx,
687                                         ctx->session_cache_tail, 0))
688                                         break;
689                                 else
690                                         ctx->stats.sess_cache_full++;
691                                 }
692                         }
693                 }
694         CRYPTO_w_unlock(CRYPTO_LOCK_SSL_CTX);
695         return(ret);
696         }

698 int SSL_CTX_remove_session(SSL_CTX *ctx, SSL_SESSION *c)
699 {
700         return remove_session_lock(ctx, c, 1);
701 }

703 static int remove_session_lock(SSL_CTX *ctx, SSL_SESSION *c, int lck)
704         {
705         SSL_SESSION *r;
706         int ret=0;

708         if ((c != NULL) && (c->session_id_length != 0))
709                 {
710                 if(lck) CRYPTO_w_lock(CRYPTO_LOCK_SSL_CTX);
711                 if ((r = lh_SSL_SESSION_retrieve(ctx->sessions,c)) == c)
712                         {
713                         ret=1;
714                         r=lh_SSL_SESSION_delete(ctx->sessions,c);
715                         SSL_SESSION_list_remove(ctx,c);
716                         }

718                 if(lck) CRYPTO_w_unlock(CRYPTO_LOCK_SSL_CTX);

720                 if (ret)
721                         {
```



```
722                         r->not_resumable=1;
723                         if (ctx->remove_session_cb != NULL)
724                                 ctx->remove_session_cb(ctx,r);
725                         SSL_SESSION_free(r);
726                         }
727                 }
728         else
729                 ret=0;
730         return(ret);
731         }

733 void SSL_SESSION_free(SSL_SESSION *ss)
734         {
735         int i;

737         if(ss == NULL)
738             return;

740         i=CRYPTO_add(&ss->references,-1,CRYPTO_LOCK_SSL_SESSION);
741 #ifdef REF_PRINT
742         REF_PRINT("SSL_SESSION",ss);
743 #endif
744         if (i > 0) return;
745 #ifdef REF_CHECK
746         if (i < 0)
747                 {
748                 fprintf(stderr,"SSL_SESSION_free, bad reference count\n");
749                 abort(); /* ok */
750                 }
751 #endif

753         CRYPTO_free_ex_data(CRYPTO_EX_INDEX_SSL_SESSION, ss, &ss->ex_data);

755         OPENSSL_cleanse(ss->key_arg,sizeof ss->key_arg);
756         OPENSSL_cleanse(ss->master_key,sizeof ss->master_key);
757         OPENSSL_cleanse(ss->session_id,sizeof ss->session_id);
758         if (ss->sess_cert != NULL) ssl_sess_cert_free(ss->sess_cert);
759         if (ss->peer != NULL) X509_free(ss->peer);
760         if (ss->ciphers != NULL) sk_SSL_CIPHER_free(ss->ciphers);
761 #ifndef OPENSSL_NO_TLSEXT
762         if (ss->tlsext_hostname != NULL) OPENSSL_free(ss->tlsext_hostname);
763         if (ss->tlsext_tick != NULL) OPENSSL_free(ss->tlsext_tick);
764 #ifndef OPENSSL_NO_EC
765         ss->tlsext_ecpointformatlist_length = 0;
766         if (ss->tlsext_ecpointformatlist != NULL) OPENSSL_free(ss->tlsext_ecpoin
767         ss->tlsext_ellipticcurvelist_length = 0;
768         if (ss->tlsext_ellipticcurvelist != NULL) OPENSSL_free(ss->tlsext_ellipt
769 #endif /* OPENSSL_NO_EC */
770 #endif
771 #ifndef OPENSSL_NO_PSK
772         if (ss->psk_identity_hint != NULL)
773                 OPENSSL_free(ss->psk_identity_hint);
774         if (ss->psk_identity != NULL)
775                 OPENSSL_free(ss->psk_identity);
776 #endif
777 #ifndef OPENSSL_NO_SRP
778         if (ss->srp_username != NULL)
779                 OPENSSL_free(ss->srp_username);
780 #endif
781         OPENSSL_cleanse(ss,sizeof(*ss));
782         OPENSSL_free(ss);
783         }

785 int SSL_set_session(SSL *s, SSL_SESSION *session)
786         {
787         int ret=0;
```

```
788         const SSL_METHOD *meth;

790         if (session != NULL)
791                 {
792                 meth=s->ctx->method->get_ssl_method(session->ssl_version);
793                 if (meth == NULL)
794                         meth=s->method->get_ssl_method(session->ssl_version);
795                 if (meth == NULL)
796                         {
797                         SSLerr(SSL_F_SSL_SET_SESSION,SSL_R_UNABLE_TO_FIND_SSL_ME
798                         return(0);
799                         }

801                 if (meth != s->method)
802                         {
803                         if (!SSL_set_ssl_method(s,meth))
804                                 return(0);
805                         }

807 #ifndef OPENSSL_NO_KRB5
808                 if (s->kssl_ctx && !s->kssl_ctx->client_princ &&
809                     session->krb5_client_princ_len > 0)
810                 {
811                     s->kssl_ctx->client_princ = (char *)OPENSSL_malloc(session->
812                     memcpy(s->kssl_ctx->client_princ,session->krb5_client_princ,
813                             session->krb5_client_princ_len);
814                     s->kssl_ctx->client_princ[session->krb5_client_princ_len] =
815                 }
816 #endif /* OPENSSL_NO_KRB5 */

818                 /* CRYPTO_w_lock(CRYPTO_LOCK_SSL);*/
819                 CRYPTO_add(&session->references,1,CRYPTO_LOCK_SSL_SESSION);
820                 if (s->session != NULL)
821                         SSL_SESSION_free(s->session);
822                 s->session=session;
823                 s->verify_result = s->session->verify_result;
824                 /* CRYPTO_w_unlock(CRYPTO_LOCK_SSL);*/
825                 ret=1;
826                 }
827         else
828                 {
829                 if (s->session != NULL)
830                         {
831                         SSL_SESSION_free(s->session);
832                         s->session=NULL;
833                         }

835                 meth=s->ctx->method;
836                 if (meth != s->method)
837                         {
838                         if (!SSL_set_ssl_method(s,meth))
839                                 return(0);
840                         }
841                 ret=1;
842                 }
843         return(ret);
844         }

846 long SSL_SESSION_set_timeout(SSL_SESSION *s, long t)
847         {
848         if (s == NULL) return(0);
849         s->timeout=t;
850         return(1);
851         }

853 long SSL_SESSION_get_timeout(const SSL_SESSION *s)
```



```
854         {
855         if (s == NULL) return(0);
856         return(s->timeout);
857         }

859 long SSL_SESSION_get_time(const SSL_SESSION *s)
860         {
861         if (s == NULL) return(0);
862         return(s->time);
863         }

865 long SSL_SESSION_set_time(SSL_SESSION *s, long t)
866         {
867         if (s == NULL) return(0);
868         s->time=t;
869         return(t);
870         }

872 X509 *SSL_SESSION_get0_peer(SSL_SESSION *s)
873         {
874         return s->peer;
875         }

877 int SSL_SESSION_set1_id_context(SSL_SESSION *s,const unsigned char *sid_ctx,
878                                 unsigned int sid_ctx_len)
879         {
880         if(sid_ctx_len > SSL_MAX_SID_CTX_LENGTH)
881                 {
882                 SSLerr(SSL_F_SSL_SESSION_SET1_ID_CONTEXT,SSL_R_SSL_SESSION_ID_CO
883                 return 0;
884                 }
885         s->sid_ctx_length=sid_ctx_len;
886         memcpy(s->sid_ctx,sid_ctx,sid_ctx_len);

888         return 1;
889         }

891 long SSL_CTX_set_timeout(SSL_CTX *s, long t)
892         {
893         long l;
894         if (s == NULL) return(0);
895         l=s->session_timeout;
896         s->session_timeout=t;
897         return(l);
898         }

900 long SSL_CTX_get_timeout(const SSL_CTX *s)
901         {
902         if (s == NULL) return(0);
903         return(s->session_timeout);
904         }

906 #ifndef OPENSSL_NO_TLSEXT
907 int SSL_set_session_secret_cb(SSL *s, int (*tls_session_secret_cb)(SSL *s, void
908         STACK_OF(SSL_CIPHER) *peer_ciphers, SSL_CIPHER **cipher, void *arg), voi
909         {
910         if (s == NULL) return(0);
911         s->tls_session_secret_cb = tls_session_secret_cb;
912         s->tls_session_secret_cb_arg = arg;
913         return(1);
914         }

916 int SSL_set_session_ticket_ext_cb(SSL *s, tls_session_ticket_ext_cb_fn cb,
917                                   void *arg)
918         {
919         if (s == NULL) return(0);
```

```
920         s->tls_session_ticket_ext_cb = cb;
921         s->tls_session_ticket_ext_cb_arg = arg;
922         return(1);
923         }

925 int SSL_set_session_ticket_ext(SSL *s, void *ext_data, int ext_len)
926         {
927         if (s->version >= TLS1_VERSION)
928                 {
929                 if (s->tlsext_session_ticket)
930                         {
931                         OPENSSL_free(s->tlsext_session_ticket);
932                         s->tlsext_session_ticket = NULL;
933                         }

935                 s->tlsext_session_ticket = OPENSSL_malloc(sizeof(TLS_SESSION_TIC
936                 if (!s->tlsext_session_ticket)
937                         {
938                         SSLerr(SSL_F_SSL_SET_SESSION_TICKET_EXT, ERR_R_MALLOC_FA
939                         return 0;
940                         }

942                 if (ext_data)
943                         {
944                         s->tlsext_session_ticket->length = ext_len;
945                         s->tlsext_session_ticket->data = s->tlsext_session_ticke
946                         memcpy(s->tlsext_session_ticket->data, ext_data, ext_len
947                         }
948                 else
949                         {
950                         s->tlsext_session_ticket->length = 0;
951                         s->tlsext_session_ticket->data = NULL;
952                         }

954                 return 1;
955                 }

957         return 0;
958         }
959 #endif /* OPENSSL_NO_TLSEXT */

961 typedef struct timeout_param_st
962         {
963         SSL_CTX *ctx;
964         long time;
965         LHASH_OF(SSL_SESSION) *cache;
966         } TIMEOUT_PARAM;

968 static void timeout_doall_arg(SSL_SESSION *s, TIMEOUT_PARAM *p)
969         {
970         if ((p->time == 0) || (p->time > (s->time+s->timeout))) /* timeout */
971                 {
972                 /* The reason we don't call SSL_CTX_remove_session() is to
973                  * save on locking overhead */
974                 (void)lh_SSL_SESSION_delete(p->cache,s);
975                 SSL_SESSION_list_remove(p->ctx,s);
976                 s->not_resumable=1;
977                 if (p->ctx->remove_session_cb != NULL)
978                         p->ctx->remove_session_cb(p->ctx,s);
979                 SSL_SESSION_free(s);
980                 }
981         }

983 static IMPLEMENT_LHASH_DOALL_ARG_FN(timeout, SSL_SESSION, TIMEOUT_PARAM)

985 void SSL_CTX_flush_sessions(SSL_CTX *s, long t)
```



```
986         {
987         unsigned long i;
988         TIMEOUT_PARAM tp;

990         tp.ctx=s;
991         tp.cache=s->sessions;
992         if (tp.cache == NULL) return;
993         tp.time=t;
994         CRYPTO_w_lock(CRYPTO_LOCK_SSL_CTX);
995         i=CHECKED_LHASH_OF(SSL_SESSION, tp.cache)->down_load;
996         CHECKED_LHASH_OF(SSL_SESSION, tp.cache)->down_load=0;
997         lh_SSL_SESSION_doall_arg(tp.cache, LHASH_DOALL_ARG_FN(timeout),
998                                  TIMEOUT_PARAM, &tp);
999         CHECKED_LHASH_OF(SSL_SESSION, tp.cache)->down_load=i;
1000        CRYPTO_w_unlock(CRYPTO_LOCK_SSL_CTX);
1001        }

1003 int ssl_clear_bad_session(SSL *s)
1004        {
1005        if (    (s->session != NULL) &&
1006                !(s->shutdown & SSL_SENT_SHUTDOWN) &&
1007                !(SSL_in_init(s) || SSL_in_before(s)))
1008                {
1009                SSL_CTX_remove_session(s->ctx,s->session);
1010                return(1);
1011                }
1012        else
1013                return(0);
1014        }

1016 /* locked by SSL_CTX in the calling function */
1017 static void SSL_SESSION_list_remove(SSL_CTX *ctx, SSL_SESSION *s)
1018        {
1019        if ((s->next == NULL) || (s->prev == NULL)) return;

1021        if (s->next == (SSL_SESSION *)&(ctx->session_cache_tail))
1022                { /* last element in list */
1023                if (s->prev == (SSL_SESSION *)&(ctx->session_cache_head))
1024                        { /* only one element in list */
1025                        ctx->session_cache_head=NULL;
1026                        ctx->session_cache_tail=NULL;
1027                        }
1028                else
1029                        {
1030                        ctx->session_cache_tail=s->prev;
1031                        s->prev->next=(SSL_SESSION *)&(ctx->session_cache_tail);
1032                        }
1033                }
1034        else
1035                {
1036                if (s->prev == (SSL_SESSION *)&(ctx->session_cache_head))
1037                        { /* first element in list */
1038                        ctx->session_cache_head=s->next;
1039                        s->next->prev=(SSL_SESSION *)&(ctx->session_cache_head);
1040                        }
1041                else
1042                        { /* middle of list */
1043                        s->next->prev=s->prev;
1044                        s->prev->next=s->next;
1045                        }
1046                }
1047        s->prev=s->next=NULL;
1048        }

1050 static void SSL_SESSION_list_add(SSL_CTX *ctx, SSL_SESSION *s)
1051        {
```

```
1052         if ((s->next != NULL) && (s->prev != NULL))
1053                 SSL_SESSION_list_remove(ctx,s);
1054
1055         if (ctx->session_cache_head == NULL)
1056                 {
1057                 ctx->session_cache_head=s;
1058                 ctx->session_cache_tail=s;
1059                 s->prev=(SSL_SESSION *)&(ctx->session_cache_head);
1060                 s->next=(SSL_SESSION *)&(ctx->session_cache_tail);
1061                 }
1062         else
1063                 {
1064                 s->next=ctx->session_cache_head;
1065                 s->next->prev=s;
1066                 s->prev=(SSL_SESSION *)&(ctx->session_cache_head);
1067                 ctx->session_cache_head=s;
1068                 }
1069         }
1070
1071 void SSL_CTX_sess_set_new_cb(SSL_CTX *ctx,
1072         int (*cb)(struct ssl_st *ssl,SSL_SESSION *sess))
1073         {
1074         ctx->new_session_cb=cb;
1075         }
1076
1077 int (*SSL_CTX_sess_get_new_cb(SSL_CTX *ctx))(SSL *ssl, SSL_SESSION *sess)
1078         {
1079         return ctx->new_session_cb;
1080         }
1081
1082 void SSL_CTX_sess_set_remove_cb(SSL_CTX *ctx,
1083         void (*cb)(SSL_CTX *ctx,SSL_SESSION *sess))
1084         {
1085         ctx->remove_session_cb=cb;
1086         }
1087
1088 void (*SSL_CTX_sess_get_remove_cb(SSL_CTX *ctx))(SSL_CTX * ctx,SSL_SESSION *sess
1089         {
1090         return ctx->remove_session_cb;
1091         }
1092
1093 void SSL_CTX_sess_set_get_cb(SSL_CTX *ctx,
1094         SSL_SESSION *(*cb)(struct ssl_st *ssl,
1095                 unsigned char *data,int len,int *copy))
1096         {
1097         ctx->get_session_cb=cb;
1098         }
1099
1100 SSL_SESSION * (*SSL_CTX_sess_get_get_cb(SSL_CTX *ctx))(SSL *ssl,
1101                 unsigned char *data,int len,int *copy)
1102         {
1103         return ctx->get_session_cb;
1104         }
1105
1106 void SSL_CTX_set_info_callback(SSL_CTX *ctx,
1107         void (*cb)(const SSL *ssl,int type,int val))
1108         {
1109         ctx->info_callback=cb;
1110         }
1111
1112 void (*SSL_CTX_get_info_callback(SSL_CTX *ctx))(const SSL *ssl,int type,int val)
1113         {
1114         return ctx->info_callback;
1115         }
1116
1117 void SSL_CTX_set_client_cert_cb(SSL_CTX *ctx,
```



```
1118         int (*cb)(SSL *ssl, X509 **x509, EVP_PKEY **pkey))
1119         {
1120         ctx->client_cert_cb=cb;
1121         }
1122
1123 int (*SSL_CTX_get_client_cert_cb(SSL_CTX *ctx))(SSL * ssl, X509 ** x509 , EVP_PK
1124         {
1125         return ctx->client_cert_cb;
1126         }
1127
1128 #ifndef OPENSSL_NO_ENGINE
1129 int SSL_CTX_set_client_cert_engine(SSL_CTX *ctx, ENGINE *e)
1130         {
1131         if (!ENGINE_init(e))
1132                 {
1133                 SSLerr(SSL_F_SSL_CTX_SET_CLIENT_CERT_ENGINE, ERR_R_ENGINE_LIB);
1134                 return 0;
1135                 }
1136         if(!ENGINE_get_ssl_client_cert_function(e))
1137                 {
1138                 SSLerr(SSL_F_SSL_CTX_SET_CLIENT_CERT_ENGINE, SSL_R_NO_CLIENT_CER
1139                 ENGINE_finish(e);
1140                 return 0;
1141                 }
1142         ctx->client_cert_engine = e;
1143         return 1;
1144         }
1145 #endif
1146
1147 void SSL_CTX_set_cookie_generate_cb(SSL_CTX *ctx,
1148         int (*cb)(SSL *ssl, unsigned char *cookie, unsigned int *cookie_len))
1149         {
1150         ctx->app_gen_cookie_cb=cb;
1151         }
1152
1153 void SSL_CTX_set_cookie_verify_cb(SSL_CTX *ctx,
1154         int (*cb)(SSL *ssl, unsigned char *cookie, unsigned int cookie_len))
1155         {
1156         ctx->app_verify_cookie_cb=cb;
1157         }
1158
1159 IMPLEMENT_PEM_rw(SSL_SESSION, SSL_SESSION, PEM_STRING_SSL_SESSION, SSL_SESSION)
1160 #endif /* ! codereview */
```

```
**********************************************************
   24446 Wed Aug 13 19:53:41 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_stat.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/ssl_stat.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
```



```
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include "ssl_locl.h"

const char *SSL_state_string_long(const SSL *s)
        {
        const char *str;

        switch (s->state)
                {
case SSL_ST_BEFORE: str="before SSL initialization"; break;
case SSL_ST_ACCEPT: str="before accept initialization"; break;
case SSL_ST_CONNECT: str="before connect initialization"; break;
case SSL_ST_OK: str="SSL negotiation finished successfully"; break;
case SSL_ST_RENEGOTIATE:        str="SSL renegotiate ciphers"; break;
case SSL_ST_BEFORE|SSL_ST_CONNECT: str="before/connect initialization"; break;
case SSL_ST_OK|SSL_ST_CONNECT: str="ok/connect SSL initialization"; break;
case SSL_ST_BEFORE|SSL_ST_ACCEPT: str="before/accept initialization"; break;
case SSL_ST_OK|SSL_ST_ACCEPT: str="ok/accept SSL initialization"; break;
#ifndef OPENSSL_NO_SSL2
case SSL2_ST_CLIENT_START_ENCRYPTION: str="SSLv2 client start encryption"; break
case SSL2_ST_SERVER_START_ENCRYPTION: str="SSLv2 server start encryption"; break
case SSL2_ST_SEND_CLIENT_HELLO_A: str="SSLv2 write client hello A"; break;
case SSL2_ST_SEND_CLIENT_HELLO_B: str="SSLv2 write client hello B"; break;
case SSL2_ST_GET_SERVER_HELLO_A: str="SSLv2 read server hello A"; break;
case SSL2_ST_GET_SERVER_HELLO_B: str="SSLv2 read server hello B"; break;
case SSL2_ST_SEND_CLIENT_MASTER_KEY_A: str="SSLv2 write client master key A"; br
case SSL2_ST_SEND_CLIENT_MASTER_KEY_B: str="SSLv2 write client master key B"; br
case SSL2_ST_SEND_CLIENT_FINISHED_A: str="SSLv2 write client finished A"; break;
case SSL2_ST_SEND_CLIENT_FINISHED_B: str="SSLv2 write client finished B"; break;
case SSL2_ST_SEND_CLIENT_CERTIFICATE_A: str="SSLv2 write client certificate A";
case SSL2_ST_SEND_CLIENT_CERTIFICATE_B: str="SSLv2 write client certificate B";
case SSL2_ST_SEND_CLIENT_CERTIFICATE_C: str="SSLv2 write client certificate C";
case SSL2_ST_SEND_CLIENT_CERTIFICATE_D: str="SSLv2 write client certificate D";
case SSL2_ST_GET_SERVER_VERIFY_A: str="SSLv2 read server verify A"; break;
case SSL2_ST_GET_SERVER_VERIFY_B: str="SSLv2 read server verify B"; break;
case SSL2_ST_GET_SERVER_FINISHED_A: str="SSLv2 read server finished A"; break;
case SSL2_ST_GET_SERVER_FINISHED_B: str="SSLv2 read server finished B"; break;
case SSL2_ST_GET_CLIENT_HELLO_A: str="SSLv2 read client hello A"; break;
case SSL2_ST_GET_CLIENT_HELLO_B: str="SSLv2 read client hello B"; break;
case SSL2_ST_GET_CLIENT_HELLO_C: str="SSLv2 read client hello C"; break;
case SSL2_ST_SEND_SERVER_HELLO_A: str="SSLv2 write server hello A"; break;
case SSL2_ST_SEND_SERVER_HELLO_B: str="SSLv2 write server hello B"; break;
case SSL2_ST_GET_CLIENT_MASTER_KEY_A: str="SSLv2 read client master key A"; brea
```

```
128 case SSL2_ST_GET_CLIENT_MASTER_KEY_B: str="SSLv2 read client master key B"; brea
129 case SSL2_ST_SEND_SERVER_VERIFY_A: str="SSLv2 write server verify A"; break;
130 case SSL2_ST_SEND_SERVER_VERIFY_B: str="SSLv2 write server verify B"; break;
131 case SSL2_ST_SEND_SERVER_VERIFY_C: str="SSLv2 write server verify C"; break;
132 case SSL2_ST_GET_CLIENT_FINISHED_A: str="SSLv2 read client finished A"; break;
133 case SSL2_ST_GET_CLIENT_FINISHED_B: str="SSLv2 read client finished B"; break;
134 case SSL2_ST_SEND_SERVER_FINISHED_A: str="SSLv2 write server finished A"; break;
135 case SSL2_ST_SEND_SERVER_FINISHED_B: str="SSLv2 write server finished B"; break;
136 case SSL2_ST_SEND_REQUEST_CERTIFICATE_A: str="SSLv2 write request certificate A"
137 case SSL2_ST_SEND_REQUEST_CERTIFICATE_B: str="SSLv2 write request certificate B"
138 case SSL2_ST_SEND_REQUEST_CERTIFICATE_C: str="SSLv2 write request certificate C"
139 case SSL2_ST_SEND_REQUEST_CERTIFICATE_D: str="SSLv2 write request certificate D"
140 case SSL2_ST_X509_GET_SERVER_CERTIFICATE: str="SSLv2 X509 read server certificat
141 case SSL2_ST_X509_GET_CLIENT_CERTIFICATE: str="SSLv2 X509 read client certificat
142 #endif

143 #ifndef OPENSSL_NO_SSL3
144 /* SSLv3 additions */
145 case SSL3_ST_CW_CLNT_HELLO_A:   str="SSLv3 write client hello A"; break;
146 case SSL3_ST_CW_CLNT_HELLO_B:   str="SSLv3 write client hello B"; break;
147 case SSL3_ST_CR_SRVR_HELLO_A:   str="SSLv3 read server hello A"; break;
148 case SSL3_ST_CR_SRVR_HELLO_B:   str="SSLv3 read server hello B"; break;
149 case SSL3_ST_CR_CERT_A:         str="SSLv3 read server certificate A"; break;
150 case SSL3_ST_CR_CERT_B:         str="SSLv3 read server certificate B"; break;
151 case SSL3_ST_CR_KEY_EXCH_A:     str="SSLv3 read server key exchange A"; break;
152 case SSL3_ST_CR_KEY_EXCH_B:     str="SSLv3 read server key exchange B"; break;
153 case SSL3_ST_CR_CERT_REQ_A:     str="SSLv3 read server certificate request A"; b
154 case SSL3_ST_CR_CERT_REQ_B:     str="SSLv3 read server certificate request B"; b
155 case SSL3_ST_CR_SESSION_TICKET_A: str="SSLv3 read server session ticket A";break
156 case SSL3_ST_CR_SESSION_TICKET_B: str="SSLv3 read server session ticket B";break
157 case SSL3_ST_CR_SRVR_DONE_A:    str="SSLv3 read server done A"; break;
158 case SSL3_ST_CR_SRVR_DONE_B:    str="SSLv3 read server done B"; break;
159 case SSL3_ST_CW_CERT_A:         str="SSLv3 write client certificate A"; break;
160 case SSL3_ST_CW_CERT_B:         str="SSLv3 write client certificate B"; break;
161 case SSL3_ST_CW_CERT_C:         str="SSLv3 write client certificate C"; break;
162 case SSL3_ST_CW_CERT_D:         str="SSLv3 write client certificate D"; break;
163 case SSL3_ST_CW_KEY_EXCH_A:     str="SSLv3 write client key exchange A"; break;
164 case SSL3_ST_CW_KEY_EXCH_B:     str="SSLv3 write client key exchange B"; break;
165 case SSL3_ST_CW_CERT_VRFY_A:    str="SSLv3 write certificate verify A"; break;
166 case SSL3_ST_CW_CERT_VRFY_B:    str="SSLv3 write certificate verify B"; break;

167 case SSL3_ST_CW_CHANGE_A:
168 case SSL3_ST_SW_CHANGE_A:       str="SSLv3 write change cipher spec A"; break;
169 case SSL3_ST_CW_CHANGE_B:
170 case SSL3_ST_SW_CHANGE_B:       str="SSLv3 write change cipher spec B"; break;
171 case SSL3_ST_CW_FINISHED_A:
172 case SSL3_ST_SW_FINISHED_A:     str="SSLv3 write finished A"; break;
173 case SSL3_ST_CW_FINISHED_B:
174 case SSL3_ST_SW_FINISHED_B:     str="SSLv3 write finished B"; break;
175 case SSL3_ST_CR_CHANGE_A:
176 case SSL3_ST_SR_CHANGE_A:       str="SSLv3 read change cipher spec A"; break;
177 case SSL3_ST_CR_CHANGE_B:
178 case SSL3_ST_SR_CHANGE_B:       str="SSLv3 read change cipher spec B"; break;
179 case SSL3_ST_CR_FINISHED_A:
180 case SSL3_ST_SR_FINISHED_A:     str="SSLv3 read finished A"; break;
181 case SSL3_ST_CR_FINISHED_B:
182 case SSL3_ST_SR_FINISHED_B:     str="SSLv3 read finished B"; break;

183 case SSL3_ST_CW_FLUSH:
184 case SSL3_ST_SW_FLUSH:          str="SSLv3 flush data"; break;

185 case SSL3_ST_SR_CLNT_HELLO_A:   str="SSLv3 read client hello A"; break;
186 case SSL3_ST_SR_CLNT_HELLO_B:   str="SSLv3 read client hello B"; break;
187 case SSL3_ST_SR_CLNT_HELLO_C:   str="SSLv3 read client hello C"; break;
188 case SSL3_ST_SW_HELLO_REQ_A:    str="SSLv3 write hello request A"; break;
189 case SSL3_ST_SW_HELLO_REQ_B:    str="SSLv3 write hello request B"; break;
```



```
190 case SSL3_ST_SW_HELLO_REQ_C:    str="SSLv3 write hello request C"; break;
191 case SSL3_ST_SW_SRVR_HELLO_A:   str="SSLv3 write server hello A"; break;
192 case SSL3_ST_SW_SRVR_HELLO_B:   str="SSLv3 write server hello B"; break;
193 case SSL3_ST_SW_CERT_A:         str="SSLv3 write certificate A"; break;
194 case SSL3_ST_SW_CERT_B:         str="SSLv3 write certificate B"; break;
195 case SSL3_ST_SW_KEY_EXCH_A:     str="SSLv3 write key exchange A"; break;
196 case SSL3_ST_SW_KEY_EXCH_B:     str="SSLv3 write key exchange B"; break;
197 case SSL3_ST_SW_CERT_REQ_A:     str="SSLv3 write certificate request A"; break;
198 case SSL3_ST_SW_CERT_REQ_B:     str="SSLv3 write certificate request B"; break;
199 case SSL3_ST_SW_SESSION_TICKET_A: str="SSLv3 write session ticket A"; break;
200 case SSL3_ST_SW_SESSION_TICKET_B: str="SSLv3 write session ticket B"; break;
201 case SSL3_ST_SW_SRVR_DONE_A:    str="SSLv3 write server done A"; break;
202 case SSL3_ST_SW_SRVR_DONE_B:    str="SSLv3 write server done B"; break;
203 case SSL3_ST_SR_CERT_A:         str="SSLv3 read client certificate A"; break;
204 case SSL3_ST_SR_CERT_B:         str="SSLv3 read client certificate B"; break;
205 case SSL3_ST_SR_KEY_EXCH_A:     str="SSLv3 read client key exchange A"; break;
206 case SSL3_ST_SR_KEY_EXCH_B:     str="SSLv3 read client key exchange B"; break;
207 case SSL3_ST_SR_CERT_VRFY_A:    str="SSLv3 read certificate verify A"; break;
208 case SSL3_ST_SR_CERT_VRFY_B:    str="SSLv3 read certificate verify B"; break;
209 #endif

210 /* SSLv2/v3 compatibility states */
211 /* client */
212 case SSL23_ST_CW_CLNT_HELLO_A:  str="SSLv2/v3 write client hello A"; break;
213 case SSL23_ST_CW_CLNT_HELLO_B:  str="SSLv2/v3 write client hello B"; break;
214 case SSL23_ST_CR_SRVR_HELLO_A:  str="SSLv2/v3 read server hello A"; break;
215 case SSL23_ST_CR_SRVR_HELLO_B:  str="SSLv2/v3 read server hello B"; break;
216 /* server */
217 case SSL23_ST_SR_CLNT_HELLO_A:  str="SSLv2/v3 read client hello A"; break;
218 case SSL23_ST_SR_CLNT_HELLO_B:  str="SSLv2/v3 read client hello B"; break;

219 /* DTLS */
220 case DTLS1_ST_CR_HELLO_VERIFY_REQUEST_A: str="DTLS1 read hello verify request A"
221 case DTLS1_ST_CR_HELLO_VERIFY_REQUEST_B: str="DTLS1 read hello verify request B"
222 case DTLS1_ST_SW_HELLO_VERIFY_REQUEST_A: str="DTLS1 write hello verify request A
223 case DTLS1_ST_SW_HELLO_VERIFY_REQUEST_B: str="DTLS1 write hello verify request B

224 default:        str="unknown state"; break;
225                 }
226         return(str);
227         }

228 const char *SSL_rstate_string_long(const SSL *s)
229         {
230         const char *str;

231         switch (s->rstate)
232                 {
233         case SSL_ST_READ_HEADER: str="read header"; break;
234         case SSL_ST_READ_BODY: str="read body"; break;
235         case SSL_ST_READ_DONE: str="read done"; break;
236         default: str="unknown"; break;
237                 }
238         return(str);
239         }

240 const char *SSL_state_string(const SSL *s)
241         {
242         const char *str;

243         switch (s->state)
244                 {
245 case SSL_ST_BEFORE:                             str="PINIT "; break;
246 case SSL_ST_ACCEPT:                             str="AINIT "; break;
247 case SSL_ST_CONNECT:                            str="CINIT "; break;
248 case SSL_ST_OK:                                 str="SSLOK "; break;
```

```
#ifndef OPENSSL_NO_SSL2
case SSL2_ST_CLIENT_START_ENCRYPTION:           str="2CSENC"; break;
case SSL2_ST_SERVER_START_ENCRYPTION:           str="2SSENC"; break;
case SSL2_ST_SEND_CLIENT_HELLO_A:               str="2SCH_A"; break;
case SSL2_ST_SEND_CLIENT_HELLO_B:               str="2SCH_B"; break;
case SSL2_ST_GET_SERVER_HELLO_A:                str="2GSH_A"; break;
case SSL2_ST_GET_SERVER_HELLO_B:                str="2GSH_B"; break;
case SSL2_ST_SEND_CLIENT_MASTER_KEY_A:          str="2SCMKA"; break;
case SSL2_ST_SEND_CLIENT_MASTER_KEY_B:          str="2SCMKB"; break;
case SSL2_ST_SEND_CLIENT_FINISHED_A:            str="2SCF_A"; break;
case SSL2_ST_SEND_CLIENT_FINISHED_B:            str="2SCF_B"; break;
case SSL2_ST_SEND_CLIENT_CERTIFICATE_A:         str="2SCC_A"; break;
case SSL2_ST_SEND_CLIENT_CERTIFICATE_B:         str="2SCC_B"; break;
case SSL2_ST_SEND_CLIENT_CERTIFICATE_C:         str="2SCC_C"; break;
case SSL2_ST_SEND_CLIENT_CERTIFICATE_D:         str="2SCC_D"; break;
case SSL2_ST_GET_SERVER_VERIFY_A:               str="2GSV_A"; break;
case SSL2_ST_GET_SERVER_VERIFY_B:               str="2GSV_B"; break;
case SSL2_ST_GET_SERVER_FINISHED_A:             str="2GSF_A"; break;
case SSL2_ST_GET_SERVER_FINISHED_B:             str="2GSF_B"; break;
case SSL2_ST_GET_CLIENT_HELLO_A:                str="2GCH_A"; break;
case SSL2_ST_GET_CLIENT_HELLO_B:                str="2GCH_B"; break;
case SSL2_ST_GET_CLIENT_HELLO_C:                str="2GCH_C"; break;
case SSL2_ST_SEND_SERVER_HELLO_A:               str="2SSH_A"; break;
case SSL2_ST_SEND_SERVER_HELLO_B:               str="2SSH_B"; break;
case SSL2_ST_GET_CLIENT_MASTER_KEY_A:           str="2GCMKA"; break;
case SSL2_ST_GET_CLIENT_MASTER_KEY_B:           str="2GCMKA"; break;
case SSL2_ST_SEND_SERVER_VERIFY_A:              str="2SSV_A"; break;
case SSL2_ST_SEND_SERVER_VERIFY_B:              str="2SSV_B"; break;
case SSL2_ST_SEND_SERVER_VERIFY_C:              str="2SSV_C"; break;
case SSL2_ST_GET_CLIENT_FINISHED_A:             str="2GCF_A"; break;
case SSL2_ST_GET_CLIENT_FINISHED_B:             str="2GCF_B"; break;
case SSL2_ST_SEND_SERVER_FINISHED_A:            str="2SSF_A"; break;
case SSL2_ST_SEND_SERVER_FINISHED_B:            str="2SSF_B"; break;
case SSL2_ST_SEND_REQUEST_CERTIFICATE_A:        str="2SRC_A"; break;
case SSL2_ST_SEND_REQUEST_CERTIFICATE_B:        str="2SRC_B"; break;
case SSL2_ST_SEND_REQUEST_CERTIFICATE_C:        str="2SRC_C"; break;
case SSL2_ST_SEND_REQUEST_CERTIFICATE_D:        str="2SRC_D"; break;
case SSL2_ST_X509_GET_SERVER_CERTIFICATE:       str="2X9GSC"; break;
case SSL2_ST_X509_GET_CLIENT_CERTIFICATE:       str="2X9GCC"; break;
#endif

#ifndef OPENSSL_NO_SSL3
/* SSLv3 additions */
case SSL3_ST_SW_FLUSH:
case SSL3_ST_CW_FLUSH:                          str="3FLUSH"; break;
case SSL3_ST_CW_CLNT_HELLO_A:                   str="3WCH_A"; break;
case SSL3_ST_CW_CLNT_HELLO_B:                   str="3WCH_B"; break;
case SSL3_ST_CR_SRVR_HELLO_A:                   str="3RSH_A"; break;
case SSL3_ST_CR_SRVR_HELLO_B:                   str="3RSH_B"; break;
case SSL3_ST_CR_CERT_A:                         str="3RSC_A"; break;
case SSL3_ST_CR_CERT_B:                         str="3RSC_B"; break;
case SSL3_ST_CR_KEY_EXCH_A:                     str="3RSKEA"; break;
case SSL3_ST_CR_KEY_EXCH_B:                     str="3RSKEB"; break;
case SSL3_ST_CR_CERT_REQ_A:                     str="3RCR_A"; break;
case SSL3_ST_CR_CERT_REQ_B:                     str="3RCR_B"; break;
case SSL3_ST_CR_SRVR_DONE_A:                    str="3RSD_A"; break;
case SSL3_ST_CR_SRVR_DONE_B:                    str="3RSD_B"; break;
case SSL3_ST_CW_CERT_A:                         str="3WCC_A"; break;
case SSL3_ST_CW_CERT_B:                         str="3WCC_B"; break;
case SSL3_ST_CW_CERT_C:                         str="3WCC_C"; break;
case SSL3_ST_CW_CERT_D:                         str="3WCC_D"; break;
case SSL3_ST_CW_KEY_EXCH_A:                     str="3WCKEA"; break;
case SSL3_ST_CW_KEY_EXCH_B:                     str="3WCKEB"; break;
case SSL3_ST_CW_CERT_VRFY_A:                    str="3WCV_A"; break;
case SSL3_ST_CW_CERT_VRFY_B:                    str="3WCV_B"; break;
```



```
case SSL3_ST_SW_CHANGE_A:
case SSL3_ST_CW_CHANGE_A:                       str="3WCCSA"; break;
case SSL3_ST_SW_CHANGE_B:
case SSL3_ST_CW_CHANGE_B:                       str="3WCCSB"; break;
case SSL3_ST_SW_FINISHED_A:
case SSL3_ST_CW_FINISHED_A:                     str="3WFINA"; break;
case SSL3_ST_SW_FINISHED_B:
case SSL3_ST_CW_FINISHED_B:                     str="3WFINB"; break;
case SSL3_ST_SR_CHANGE_A:
case SSL3_ST_CR_CHANGE_A:                       str="3RCCSA"; break;
case SSL3_ST_SR_CHANGE_B:
case SSL3_ST_CR_CHANGE_B:                       str="3RCCSB"; break;
case SSL3_ST_SR_FINISHED_A:
case SSL3_ST_CR_FINISHED_A:                     str="3RFINA"; break;
case SSL3_ST_SR_FINISHED_B:
case SSL3_ST_CR_FINISHED_B:                     str="3RFINB"; break;

case SSL3_ST_SW_HELLO_REQ_A:                    str="3WHR_A"; break;
case SSL3_ST_SW_HELLO_REQ_B:                    str="3WHR_B"; break;
case SSL3_ST_SW_HELLO_REQ_C:                    str="3WHR_C"; break;
case SSL3_ST_SR_CLNT_HELLO_A:                   str="3RCH_A"; break;
case SSL3_ST_SR_CLNT_HELLO_B:                   str="3RCH_B"; break;
case SSL3_ST_SR_CLNT_HELLO_C:                   str="3RCH_C"; break;
case SSL3_ST_SW_SRVR_HELLO_A:                   str="3WSH_A"; break;
case SSL3_ST_SW_SRVR_HELLO_B:                   str="3WSH_B"; break;
case SSL3_ST_SW_CERT_A:                         str="3WSC_A"; break;
case SSL3_ST_SW_CERT_B:                         str="3WSC_B"; break;
case SSL3_ST_SW_KEY_EXCH_A:                     str="3WSKEA"; break;
case SSL3_ST_SW_KEY_EXCH_B:                     str="3WSKEB"; break;
case SSL3_ST_SW_CERT_REQ_A:                     str="3WCR_A"; break;
case SSL3_ST_SW_CERT_REQ_B:                     str="3WCR_B"; break;
case SSL3_ST_SW_SRVR_DONE_A:                    str="3WSD_A"; break;
case SSL3_ST_SW_SRVR_DONE_B:                    str="3WSD_B"; break;
case SSL3_ST_SR_CERT_A:                         str="3RCC_A"; break;
case SSL3_ST_SR_CERT_B:                         str="3RCC_B"; break;
case SSL3_ST_SR_KEY_EXCH_A:                     str="3RCKEA"; break;
case SSL3_ST_SR_KEY_EXCH_B:                     str="3RCKEB"; break;
case SSL3_ST_SR_CERT_VRFY_A:                    str="3RCV_A"; break;
case SSL3_ST_SR_CERT_VRFY_B:                    str="3RCV_B"; break;
#endif

/* SSLv2/v3 compatibility states */
/* client */
case SSL23_ST_CW_CLNT_HELLO_A:                  str="23WCHA"; break;
case SSL23_ST_CW_CLNT_HELLO_B:                  str="23WCHB"; break;
case SSL23_ST_CR_SRVR_HELLO_A:                  str="23RSHA"; break;
case SSL23_ST_CR_SRVR_HELLO_B:                  str="23RSHA"; break;
/* server */
case SSL23_ST_SR_CLNT_HELLO_A:                  str="23RCHA"; break;
case SSL23_ST_SR_CLNT_HELLO_B:                  str="23RCHB"; break;

/* DTLS */
case DTLS1_ST_CR_HELLO_VERIFY_REQUEST_A: str="DRCHVA"; break;
case DTLS1_ST_CR_HELLO_VERIFY_REQUEST_B: str="DRCHVB"; break;
case DTLS1_ST_SW_HELLO_VERIFY_REQUEST_A: str="DWCHVA"; break;
case DTLS1_ST_SW_HELLO_VERIFY_REQUEST_B: str="DWCHVB"; break;

default:                                        str="UNKWN "; break;
                }
        return(str);
        }

const char *SSL_alert_type_string_long(int value)
        {
        value>>=8;
        if (value == SSL3_AL_WARNING)
```

```
392                 return("warning");
393         else if (value == SSL3_AL_FATAL)
394                 return("fatal");
395         else
396                 return("unknown");
397         }

399 const char *SSL_alert_type_string(int value)
400         {
401         value>>=8;
402         if (value == SSL3_AL_WARNING)
403                 return("W");
404         else if (value == SSL3_AL_FATAL)
405                 return("F");
406         else
407                 return("U");
408         }

410 const char *SSL_alert_desc_string(int value)
411         {
412         const char *str;

414         switch (value & 0xff)
415                 {
416         case SSL3_AD_CLOSE_NOTIFY:              str="CN"; break;
417         case SSL3_AD_UNEXPECTED_MESSAGE:        str="UM"; break;
418         case SSL3_AD_BAD_RECORD_MAC:            str="BM"; break;
419         case SSL3_AD_DECOMPRESSION_FAILURE:     str="DF"; break;
420         case SSL3_AD_HANDSHAKE_FAILURE:         str="HF"; break;
421         case SSL3_AD_NO_CERTIFICATE:            str="NC"; break;
422         case SSL3_AD_BAD_CERTIFICATE:           str="BC"; break;
423         case SSL3_AD_UNSUPPORTED_CERTIFICATE:   str="UC"; break;
424         case SSL3_AD_CERTIFICATE_REVOKED:       str="CR"; break;
425         case SSL3_AD_CERTIFICATE_EXPIRED:       str="CE"; break;
426         case SSL3_AD_CERTIFICATE_UNKNOWN:       str="CU"; break;
427         case SSL3_AD_ILLEGAL_PARAMETER:         str="IP"; break;
428         case TLS1_AD_DECRYPTION_FAILED:         str="DC"; break;
429         case TLS1_AD_RECORD_OVERFLOW:           str="RO"; break;
430         case TLS1_AD_UNKNOWN_CA:                str="CA"; break;
431         case TLS1_AD_ACCESS_DENIED:             str="AD"; break;
432         case TLS1_AD_DECODE_ERROR:              str="DE"; break;
433         case TLS1_AD_DECRYPT_ERROR:             str="CY"; break;
434         case TLS1_AD_EXPORT_RESTRICTION:        str="ER"; break;
435         case TLS1_AD_PROTOCOL_VERSION:          str="PV"; break;
436         case TLS1_AD_INSUFFICIENT_SECURITY:     str="IS"; break;
437         case TLS1_AD_INTERNAL_ERROR:            str="IE"; break;
438         case TLS1_AD_USER_CANCELLED:            str="US"; break;
439         case TLS1_AD_NO_RENEGOTIATION:          str="NR"; break;
440         case TLS1_AD_UNSUPPORTED_EXTENSION:     str="UE"; break;
441         case TLS1_AD_CERTIFICATE_UNOBTAINABLE:  str="CO"; break;
442         case TLS1_AD_UNRECOGNIZED_NAME:         str="UN"; break;
443         case TLS1_AD_BAD_CERTIFICATE_STATUS_RESPONSE: str="BR"; break;
444         case TLS1_AD_BAD_CERTIFICATE_HASH_VALUE: str="BH"; break;
445         case TLS1_AD_UNKNOWN_PSK_IDENTITY:      str="UP"; break;
446         default:                                str="UK"; break;
447                 }
448         return(str);
449         }

451 const char *SSL_alert_desc_string_long(int value)
452         {
453         const char *str;

455         switch (value & 0xff)
456                 {
457         case SSL3_AD_CLOSE_NOTIFY:
```



```
458                 str="close notify";
459                 break;
460         case SSL3_AD_UNEXPECTED_MESSAGE:
461                 str="unexpected_message";
462                 break;
463         case SSL3_AD_BAD_RECORD_MAC:
464                 str="bad record mac";
465                 break;
466         case SSL3_AD_DECOMPRESSION_FAILURE:
467                 str="decompression failure";
468                 break;
469         case SSL3_AD_HANDSHAKE_FAILURE:
470                 str="handshake failure";
471                 break;
472         case SSL3_AD_NO_CERTIFICATE:
473                 str="no certificate";
474                 break;
475         case SSL3_AD_BAD_CERTIFICATE:
476                 str="bad certificate";
477                 break;
478         case SSL3_AD_UNSUPPORTED_CERTIFICATE:
479                 str="unsupported certificate";
480                 break;
481         case SSL3_AD_CERTIFICATE_REVOKED:
482                 str="certificate revoked";
483                 break;
484         case SSL3_AD_CERTIFICATE_EXPIRED:
485                 str="certificate expired";
486                 break;
487         case SSL3_AD_CERTIFICATE_UNKNOWN:
488                 str="certificate unknown";
489                 break;
490         case SSL3_AD_ILLEGAL_PARAMETER:
491                 str="illegal parameter";
492                 break;
493         case TLS1_AD_DECRYPTION_FAILED:
494                 str="decryption failed";
495                 break;
496         case TLS1_AD_RECORD_OVERFLOW:
497                 str="record overflow";
498                 break;
499         case TLS1_AD_UNKNOWN_CA:
500                 str="unknown CA";
501                 break;
502         case TLS1_AD_ACCESS_DENIED:
503                 str="access denied";
504                 break;
505         case TLS1_AD_DECODE_ERROR:
506                 str="decode error";
507                 break;
508         case TLS1_AD_DECRYPT_ERROR:
509                 str="decrypt error";
510                 break;
511         case TLS1_AD_EXPORT_RESTRICTION:
512                 str="export restriction";
513                 break;
514         case TLS1_AD_PROTOCOL_VERSION:
515                 str="protocol version";
516                 break;
517         case TLS1_AD_INSUFFICIENT_SECURITY:
518                 str="insufficient security";
519                 break;
520         case TLS1_AD_INTERNAL_ERROR:
521                 str="internal error";
522                 break;
523         case TLS1_AD_USER_CANCELLED:
```

```
524                 str="user canceled";
525                 break;
526         case TLS1_AD_NO_RENEGOTIATION:
527                 str="no renegotiation";
528                 break;
529         case TLS1_AD_UNSUPPORTED_EXTENSION:
530                 str="unsupported extension";
531                 break;
532         case TLS1_AD_CERTIFICATE_UNOBTAINABLE:
533                 str="certificate unobtainable";
534                 break;
535         case TLS1_AD_UNRECOGNIZED_NAME:
536                 str="unrecognized name";
537                 break;
538         case TLS1_AD_BAD_CERTIFICATE_STATUS_RESPONSE:
539                 str="bad certificate status response";
540                 break;
541         case TLS1_AD_BAD_CERTIFICATE_HASH_VALUE:
542                 str="bad certificate hash value";
543                 break;
544         case TLS1_AD_UNKNOWN_PSK_IDENTITY:
545                 str="unknown PSK identity";
546                 break;
547         default: str="unknown"; break;
548                 }
549         return(str);
550         }

552 const char *SSL_rstate_string(const SSL *s)
553         {
554         const char *str;

556         switch (s->rstate)
557                 {
558         case SSL_ST_READ_HEADER:str="RH"; break;
559         case SSL_ST_READ_BODY:  str="RB"; break;
560         case SSL_ST_READ_DONE:  str="RD"; break;
561         default: str="unknown"; break;
562                 }
563         return(str);
564         }
565 #endif /* ! codereview */
```

```
**********************************************************
    8694 Wed Aug 13 19:53:41 2014
new/usr/src/lib/openssl/libsunw_ssl/ssl_txt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/ssl_txt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
```



```
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include <openssl/buffer.h>
#include "ssl_locl.h"

#ifndef OPENSSL_NO_FP_API
int SSL_SESSION_print_fp(FILE *fp, const SSL_SESSION *x)
        {
        BIO *b;
        int ret;

        if ((b=BIO_new(BIO_s_file_internal())) == NULL)
                {
                SSLerr(SSL_F_SSL_SESSION_PRINT_FP,ERR_R_BUF_LIB);
                return(0);
                }
        BIO_set_fp(b,fp,BIO_NOCLOSE);
        ret=SSL_SESSION_print(b,x);
        BIO_free(b);
        return(ret);
        }
#endif

int SSL_SESSION_print(BIO *bp, const SSL_SESSION *x)
        {
        unsigned int i;
        const char *s;

        if (x == NULL) goto err;
        if (BIO_puts(bp,"SSL-Session:\n") <= 0) goto err;
        if (x->ssl_version == SSL2_VERSION)
                s="SSLv2";
        else if (x->ssl_version == SSL3_VERSION)
                s="SSLv3";
        else if (x->ssl_version == TLS1_2_VERSION)
                s="TLSv1.2";
        else if (x->ssl_version == TLS1_1_VERSION)
                s="TLSv1.1";
        else if (x->ssl_version == TLS1_VERSION)
                s="TLSv1";
        else if (x->ssl_version == DTLS1_VERSION)
                s="DTLSv1";
        else if (x->ssl_version == DTLS1_BAD_VER)
                s="DTLSv1-bad";
```

```
128         else
129                 s="unknown";
130         if (BIO_printf(bp,"    Protocol  : %s\n",s) <= 0) goto err;

132         if (x->cipher == NULL)
133                 {
134                 if (((x->cipher_id) & 0xff000000) == 0x02000000)
135                         {
136                         if (BIO_printf(bp,"    Cipher    : %06lX\n",x->cipher_id
137                                 goto err;
138                         }
139                 else
140                         {
141                         if (BIO_printf(bp,"    Cipher    : %04lX\n",x->cipher_id
142                                 goto err;
143                         }
144                 }
145         else
146                 {
147                 if (BIO_printf(bp,"    Cipher    : %s\n",((x->cipher == NULL)?"u
148                         goto err;
149                 }
150         if (BIO_puts(bp,"    Session-ID: ") <= 0) goto err;
151         for (i=0; i<x->session_id_length; i++)
152                 {
153                 if (BIO_printf(bp,"%02X",x->session_id[i]) <= 0) goto err;
154                 }
155         if (BIO_puts(bp,"\n    Session-ID-ctx: ") <= 0) goto err;
156         for (i=0; i<x->sid_ctx_length; i++)
157                 {
158                 if (BIO_printf(bp,"%02X",x->sid_ctx[i]) <= 0)
159                         goto err;
160                 }
161         if (BIO_puts(bp,"\n    Master-Key: ") <= 0) goto err;
162         for (i=0; i<(unsigned int)x->master_key_length; i++)
163                 {
164                 if (BIO_printf(bp,"%02X",x->master_key[i]) <= 0) goto err;
165                 }
166         if (BIO_puts(bp,"\n    Key-Arg   : ") <= 0) goto err;
167         if (x->key_arg_length == 0)
168                 {
169                 if (BIO_puts(bp,"None") <= 0) goto err;
170                 }
171         else
172                 for (i=0; i<x->key_arg_length; i++)
173                         {
174                         if (BIO_printf(bp,"%02X",x->key_arg[i]) <= 0) goto err;
175                         }
176 #ifndef OPENSSL_NO_KRB5
177        if (BIO_puts(bp,"\n    Krb5 Principal: ") <= 0) goto err;
178            if (x->krb5_client_princ_len == 0)
179            {
180                if (BIO_puts(bp,"None") <= 0) goto err;
181                }
182         else
183                 for (i=0; i<x->krb5_client_princ_len; i++)
184                         {
185                         if (BIO_printf(bp,"%02X",x->krb5_client_princ[i]) <= 0)
186                         }
187 #endif /* OPENSSL_NO_KRB5 */
188 #ifndef OPENSSL_NO_PSK
189         if (BIO_puts(bp,"\n    PSK identity: ") <= 0) goto err;
190         if (BIO_printf(bp, "%s", x->psk_identity ? x->psk_identity : "None") <=
191         if (BIO_puts(bp,"\n    PSK identity hint: ") <= 0) goto err;
192         if (BIO_printf(bp, "%s", x->psk_identity_hint ? x->psk_identity_hint : "
193 #endif
```



```
194 #ifndef OPENSSL_NO_SRP
195         if (BIO_puts(bp,"\n    SRP username: ") <= 0) goto err;
196         if (BIO_printf(bp, "%s", x->srp_username ? x->srp_username : "None") <=
197 #endif
198 #ifndef OPENSSL_NO_TLSEXT
199         if (x->tlsext_tick_lifetime_hint)
200                 {
201                 if (BIO_printf(bp,
202                         "\n    TLS session ticket lifetime hint: %ld (seconds)",
203                         x->tlsext_tick_lifetime_hint) <=0)
204                         goto err;
205                 }
206         if (x->tlsext_tick)
207                 {
208                 if (BIO_puts(bp, "\n    TLS session ticket:\n") <= 0) goto err;
209                 if (BIO_dump_indent(bp, (char *)x->tlsext_tick, x->tlsext_tickle
210                         goto err;
211                 }
212 #endif

214 #ifndef OPENSSL_NO_COMP
215         if (x->compress_meth != 0)
216                 {
217                 SSL_COMP *comp = NULL;

219                 ssl_cipher_get_evp(x,NULL,NULL,NULL,NULL,&comp);
220                 if (comp == NULL)
221                         {
222                         if (BIO_printf(bp,"\n    Compression: %d",x->compress_me
223                         }
224                 else
225                         {
226                         if (BIO_printf(bp,"\n    Compression: %d (%s)", comp->id
227                         }
228                 }
229 #endif
230         if (x->time != 0L)
231                 {
232                 if (BIO_printf(bp, "\n    Start Time: %ld",x->time) <= 0) goto e
233                 }
234         if (x->timeout != 0L)
235                 {
236                 if (BIO_printf(bp, "\n    Timeout   : %ld (sec)",x->timeout) <=
237                 }
238         if (BIO_puts(bp,"\n") <= 0) goto err;

240         if (BIO_puts(bp, "    Verify return code: ") <= 0) goto err;
241         if (BIO_printf(bp, "%ld (%s)\n", x->verify_result,
242                 X509_verify_cert_error_string(x->verify_result)) <= 0) goto err;

244         return(1);
245 err:
246         return(0);
247         }
248 #endif /* ! codereview */
```

```
**********************************************************
    4052 Wed Aug 13 19:53:41 2014
new/usr/src/lib/openssl/libsunw_ssl/t1_clnt.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/t1_clnt.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/buffer.h>
```



```
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>

static const SSL_METHOD *tls1_get_client_method(int ver);
static const SSL_METHOD *tls1_get_client_method(int ver)
        {
        if (ver == TLS1_2_VERSION)
                return TLSv1_2_client_method();
        if (ver == TLS1_1_VERSION)
                return TLSv1_1_client_method();
        if (ver == TLS1_VERSION)
                return TLSv1_client_method();
        return NULL;
        }

IMPLEMENT_tls_meth_func(TLS1_2_VERSION, TLSv1_2_client_method,
                        ssl_undefined_function,
                        ssl3_connect,
                        tls1_get_client_method)

IMPLEMENT_tls_meth_func(TLS1_1_VERSION, TLSv1_1_client_method,
                        ssl_undefined_function,
                        ssl3_connect,
                        tls1_get_client_method)

IMPLEMENT_tls_meth_func(TLS1_VERSION, TLSv1_client_method,
                        ssl_undefined_function,
                        ssl3_connect,
                        tls1_get_client_method)
#endif /* ! codereview */
```

```
**********************************************************
   37212 Wed Aug 13 19:53:41 2014
new/usr/src/lib/openssl/libsunw_ssl/t1_enc.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/t1_enc.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
/* ====================================================================
 * Copyright 2005 Nokia. All rights reserved.
 *
 * The portions of the attached software ("Contribution") is developed by
 * Nokia Corporation and is licensed pursuant to the OpenSSL open source
 * license.
 *
 * The Contribution, originally written by Mika Kousa and Pasi Eronen of
 * Nokia Corporation, consists of the "PSK" (Pre-Shared Key) ciphersuites
 * support (see RFC 4279) to OpenSSL.
 *
 * No patent licenses or other rights except those expressly stated in
 * the OpenSSL open source license shall be deemed granted or received
 * expressly, by implication, estoppel, or otherwise.
 *
 * No assurances are provided by Nokia that the Contribution does not
 * infringe the patent or other intellectual property rights of any third
```

```
 * party or that the license provides you with all the necessary rights
 * to make use of the Contribution.
 *
 * THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IN
 * ADDITION TO THE DISCLAIMERS INCLUDED IN THE LICENSE, NOKIA
 * SPECIFICALLY DISCLAIMS ANY LIABILITY FOR CLAIMS BROUGHT BY YOU OR ANY
 * OTHER ENTITY BASED ON INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OR
 * OTHERWISE.
 */

#include <stdio.h>
#include "ssl_locl.h"
#ifndef OPENSSL_NO_COMP
#include <openssl/comp.h>
#endif
#include <openssl/evp.h>
#include <openssl/hmac.h>
#include <openssl/md5.h>
#include <openssl/rand.h>
#ifdef KSSL_DEBUG
#include <openssl/des.h>
#endif

/* seed1 through seed5 are virtually concatenated */
static int tls1_P_hash(const EVP_MD *md, const unsigned char *sec,
                       int sec_len,
                       const void *seed1, int seed1_len,
                       const void *seed2, int seed2_len,
                       const void *seed3, int seed3_len,
                       const void *seed4, int seed4_len,
                       const void *seed5, int seed5_len,
                       unsigned char *out, int olen)
        {
        int chunk;
        size_t j;
        EVP_MD_CTX ctx, ctx_tmp;
        EVP_PKEY *mac_key;
        unsigned char A1[EVP_MAX_MD_SIZE];
        size_t A1_len;
        int ret = 0;

        chunk=EVP_MD_size(md);
        OPENSSL_assert(chunk >= 0);

        EVP_MD_CTX_init(&ctx);
        EVP_MD_CTX_init(&ctx_tmp);
        EVP_MD_CTX_set_flags(&ctx, EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
        EVP_MD_CTX_set_flags(&ctx_tmp, EVP_MD_CTX_FLAG_NON_FIPS_ALLOW);
        mac_key = EVP_PKEY_new_mac_key(EVP_PKEY_HMAC, NULL, sec, sec_len);
        if (!mac_key)
                goto err;
        if (!EVP_DigestSignInit(&ctx,NULL,md, NULL, mac_key))
                goto err;
        if (!EVP_DigestSignInit(&ctx_tmp,NULL,md, NULL, mac_key))
                goto err;
        if (seed1 && !EVP_DigestSignUpdate(&ctx,seed1,seed1_len))
                goto err;
        if (seed2 && !EVP_DigestSignUpdate(&ctx,seed2,seed2_len))
                goto err;
        if (seed3 && !EVP_DigestSignUpdate(&ctx,seed3,seed3_len))
                goto err;
        if (seed4 && !EVP_DigestSignUpdate(&ctx,seed4,seed4_len))
                goto err;
        if (seed5 && !EVP_DigestSignUpdate(&ctx,seed5,seed5_len))
                goto err;
        if (!EVP_DigestSignFinal(&ctx,A1,&A1_len))
```



```
                goto err;

        for (;;)
                {
                /* Reinit mac contexts */
                if (!EVP_DigestSignInit(&ctx,NULL,md, NULL, mac_key))
                        goto err;
                if (!EVP_DigestSignInit(&ctx_tmp,NULL,md, NULL, mac_key))
                        goto err;
                if (!EVP_DigestSignUpdate(&ctx,A1,A1_len))
                        goto err;
                if (!EVP_DigestSignUpdate(&ctx_tmp,A1,A1_len))
                        goto err;
                if (seed1 && !EVP_DigestSignUpdate(&ctx,seed1,seed1_len))
                        goto err;
                if (seed2 && !EVP_DigestSignUpdate(&ctx,seed2,seed2_len))
                        goto err;
                if (seed3 && !EVP_DigestSignUpdate(&ctx,seed3,seed3_len))
                        goto err;
                if (seed4 && !EVP_DigestSignUpdate(&ctx,seed4,seed4_len))
                        goto err;
                if (seed5 && !EVP_DigestSignUpdate(&ctx,seed5,seed5_len))
                        goto err;

                if (olen > chunk)
                        {
                        if (!EVP_DigestSignFinal(&ctx,out,&j))
                                goto err;
                        out+=j;
                        olen-=j;
                        /* calc the next A1 value */
                        if (!EVP_DigestSignFinal(&ctx_tmp,A1,&A1_len))
                                goto err;
                        }
                else    /* last one */
                        {
                        if (!EVP_DigestSignFinal(&ctx,A1,&A1_len))
                                goto err;
                        memcpy(out,A1,olen);
                        break;
                        }
                }
        ret = 1;
err:
        EVP_PKEY_free(mac_key);
        EVP_MD_CTX_cleanup(&ctx);
        EVP_MD_CTX_cleanup(&ctx_tmp);
        OPENSSL_cleanse(A1,sizeof(A1));
        return ret;
        }

/* seed1 through seed5 are virtually concatenated */
static int tls1_PRF(long digest_mask,
                    const void *seed1, int seed1_len,
                    const void *seed2, int seed2_len,
                    const void *seed3, int seed3_len,
                    const void *seed4, int seed4_len,
                    const void *seed5, int seed5_len,
                    const unsigned char *sec, int slen,
                    unsigned char *out1,
                    unsigned char *out2, int olen)
        {
        int len,i,idx,count;
        const unsigned char *S1;
        long m;
        const EVP_MD *md;
```

```
260         int ret = 0;

262         /* Count number of digests and partition sec evenly */
263         count=0;
264         for (idx=0;ssl_get_handshake_digest(idx,&m,&md);idx++) {
265                 if ((m<<TLS1_PRF_DGST_SHIFT) & digest_mask) count++;
266         }
267         len=slen/count;
268         if (count == 1)
269                 slen = 0;
270         S1=sec;
271         memset(out1,0,olen);
272         for (idx=0;ssl_get_handshake_digest(idx,&m,&md);idx++) {
273                 if ((m<<TLS1_PRF_DGST_SHIFT) & digest_mask) {
274                         if (!md) {
275                                 SSLerr(SSL_F_TLS1_PRF,
276                                 SSL_R_UNSUPPORTED_DIGEST_TYPE);
277                                 goto err;
278                         }
279                         if (!tls1_P_hash(md ,S1,len+(slen&1),
280                                         seed1,seed1_len,seed2,seed2_len,seed3,se
281                                         out2,olen))
282                                 goto err;
283                         S1+=len;
284                         for (i=0; i<olen; i++)
285                         {
286                                 out1[i]^=out2[i];
287                         }
288                 }
289         }
290         ret = 1;
291 err:
292         return ret;
293 }
294 static int tls1_generate_key_block(SSL *s, unsigned char *km,
295             unsigned char *tmp, int num)
296         {
297         int ret;
298         ret = tls1_PRF(ssl_get_algorithm2(s),
299                  TLS_MD_KEY_EXPANSION_CONST,TLS_MD_KEY_EXPANSION_CONST_SIZE,
300                  s->s3->server_random,SSL3_RANDOM_SIZE,
301                  s->s3->client_random,SSL3_RANDOM_SIZE,
302                  NULL,0,NULL,0,
303                  s->session->master_key,s->session->master_key_length,
304                  km,tmp,num);
305 #ifdef KSSL_DEBUG
306         printf("tls1_generate_key_block() ==> %d byte master_key =\n\t",
307                 s->session->master_key_length);
308         {
309         int i;
310         for (i=0; i < s->session->master_key_length; i++)
311                 {
312                 printf("%02X", s->session->master_key[i]);
313                 }
314         printf("\n");  }
315 #endif    /* KSSL_DEBUG */
316         return ret;
317         }

319 int tls1_change_cipher_state(SSL *s, int which)
320         {
321         static const unsigned char empty[]="";
322         unsigned char *p,*mac_secret;
323         unsigned char *exp_label;
324         unsigned char tmp1[EVP_MAX_KEY_LENGTH];
325         unsigned char tmp2[EVP_MAX_KEY_LENGTH];
```



```
326         unsigned char iv1[EVP_MAX_IV_LENGTH*2];
327         unsigned char iv2[EVP_MAX_IV_LENGTH*2];
328         unsigned char *ms,*key,*iv;
329         int client_write;
330         EVP_CIPHER_CTX *dd;
331         const EVP_CIPHER *c;
332 #ifndef OPENSSL_NO_COMP
333         const SSL_COMP *comp;
334 #endif
335         const EVP_MD *m;
336         int mac_type;
337         int *mac_secret_size;
338         EVP_MD_CTX *mac_ctx;
339         EVP_PKEY *mac_key;
340         int is_export,n,i,j,k,exp_label_len,cl;
341         int reuse_dd = 0;

343         is_export=SSL_C_IS_EXPORT(s->s3->tmp.new_cipher);
344         c=s->s3->tmp.new_sym_enc;
345         m=s->s3->tmp.new_hash;
346         mac_type = s->s3->tmp.new_mac_pkey_type;
347 #ifndef OPENSSL_NO_COMP
348         comp=s->s3->tmp.new_compression;
349 #endif

351 #ifdef KSSL_DEBUG
352         printf("tls1_change_cipher_state(which= %d) w/\n", which);
353         printf("\talg= %ld/%ld, comp= %p\n",
354                s->s3->tmp.new_cipher->algorithm_mkey,
355                s->s3->tmp.new_cipher->algorithm_auth,
356                comp);
357         printf("\tevp_cipher == %p ==? &d_cbc_ede_cipher3\n", c);
358         printf("\tevp_cipher: nid, blksz= %d, %d, keylen=%d, ivlen=%d\n",
359                 c->nid,c->block_size,c->key_len,c->iv_len);
360         printf("\tkey_block: len= %d, data= ", s->s3->tmp.key_block_length);
361         {
362         int i;
363         for (i=0; i<s->s3->tmp.key_block_length; i++)
364                 printf("%02x", s->s3->tmp.key_block[i]);  printf("\n");
365         }
366 #endif  /* KSSL_DEBUG */

368         if (which & SSL3_CC_READ)
369                 {
370                 if (s->s3->tmp.new_cipher->algorithm2 & TLS1_STREAM_MAC)
371                         s->mac_flags |= SSL_MAC_FLAG_READ_MAC_STREAM;
372                 else
373                         s->mac_flags &= ~SSL_MAC_FLAG_READ_MAC_STREAM;

375                 if (s->enc_read_ctx != NULL)
376                         reuse_dd = 1;
377                 else if ((s->enc_read_ctx=OPENSSL_malloc(sizeof(EVP_CIPHER_CTX))
378                         goto err;
379                 else
380                         /* make sure it's intialized in case we exit later with
381                         EVP_CIPHER_CTX_init(s->enc_read_ctx);
382                 dd= s->enc_read_ctx;
383                 mac_ctx=ssl_replace_hash(&s->read_hash,NULL);
384 #ifndef OPENSSL_NO_COMP
385                 if (s->expand != NULL)
386                         {
387                         COMP_CTX_free(s->expand);
388                         s->expand=NULL;
389                         }
390                 if (comp != NULL)
391                         {
```

```
392                         s->expand=COMP_CTX_new(comp->method);
393                         if (s->expand == NULL)
394                                 {
395                                 SSLerr(SSL_F_TLS1_CHANGE_CIPHER_STATE,SSL_R_COMP
396                                 goto err2;
397                                 }
398                         if (s->s3->rrec.comp == NULL)
399                                 s->s3->rrec.comp=(unsigned char *)
400                                         OPENSSL_malloc(SSL3_RT_MAX_ENCRYPTED_LEN
401                         if (s->s3->rrec.comp == NULL)
402                                 goto err;
403                         }
404 #endif
405                 /* this is done by dtls1_reset_seq_numbers for DTLS1_VERSION */
406                 if (s->version != DTLS1_VERSION)
407                         memset(&(s->s3->read_sequence[0]),0,8);
408                 mac_secret= &(s->s3->read_mac_secret[0]);
409                 mac_secret_size=&(s->s3->read_mac_secret_size);
410                 }
411         else
412                 {
413                 if (s->s3->tmp.new_cipher->algorithm2 & TLS1_STREAM_MAC)
414                         s->mac_flags |= SSL_MAC_FLAG_WRITE_MAC_STREAM;
415                         else
416                         s->mac_flags &= ~SSL_MAC_FLAG_WRITE_MAC_STREAM;
417                 if (s->enc_write_ctx != NULL && !SSL_IS_DTLS(s))
418                         reuse_dd = 1;
419                 else if ((s->enc_write_ctx=EVP_CIPHER_CTX_new()) == NULL)
420                         goto err;
421                 dd= s->enc_write_ctx;
422                 if (SSL_IS_DTLS(s))
423                         {
424                         mac_ctx = EVP_MD_CTX_create();
425                         if (!mac_ctx)
426                                 goto err;
427                         s->write_hash = mac_ctx;
428                         }
429                 else
430                         mac_ctx = ssl_replace_hash(&s->write_hash,NULL);
431 #ifndef OPENSSL_NO_COMP
432                 if (s->compress != NULL)
433                         {
434                         COMP_CTX_free(s->compress);
435                         s->compress=NULL;
436                         }
437                 if (comp != NULL)
438                         {
439                         s->compress=COMP_CTX_new(comp->method);
440                         if (s->compress == NULL)
441                                 {
442                                 SSLerr(SSL_F_TLS1_CHANGE_CIPHER_STATE,SSL_R_COMP
443                                 goto err2;
444                                 }
445                         }
446 #endif
447                 /* this is done by dtls1_reset_seq_numbers for DTLS1_VERSION */
448                 if (s->version != DTLS1_VERSION)
449                         memset(&(s->s3->write_sequence[0]),0,8);
450                 mac_secret= &(s->s3->write_mac_secret[0]);
451                 mac_secret_size = &(s->s3->write_mac_secret_size);
452                 }

454         if (reuse_dd)
455                 EVP_CIPHER_CTX_cleanup(dd);

457         p=s->s3->tmp.key_block;
```



```
458         i=*mac_secret_size=s->s3->tmp.new_mac_secret_size;

460         cl=EVP_CIPHER_key_length(c);
461         j=is_export ? (cl < SSL_C_EXPORT_KEYLENGTH(s->s3->tmp.new_cipher) ?
462                         cl : SSL_C_EXPORT_KEYLENGTH(s->s3->tmp.new_cipher)) : cl;
463         /* Was j=(exp)?5:EVP_CIPHER_key_length(c); */
464         /* If GCM mode only part of IV comes from PRF */
465         if (EVP_CIPHER_mode(c) == EVP_CIPH_GCM_MODE)
466                 k = EVP_GCM_TLS_FIXED_IV_LEN;
467         else
468                 k=EVP_CIPHER_iv_length(c);
469         if (    (which == SSL3_CHANGE_CIPHER_CLIENT_WRITE) ||
470                 (which == SSL3_CHANGE_CIPHER_SERVER_READ))
471                 {
472                 ms=  &(p[ 0]); n=i+i;
473                 key= &(p[ n]); n+=j+j;
474                 iv=  &(p[ n]); n+=k+k;
475                 exp_label=(unsigned char *)TLS_MD_CLIENT_WRITE_KEY_CONST;
476                 exp_label_len=TLS_MD_CLIENT_WRITE_KEY_CONST_SIZE;
477                 client_write=1;
478                 }
479         else
480                 {
481                 n=i;
482                 ms=  &(p[ n]); n+=i+j;
483                 key= &(p[ n]); n+=j+k;
484                 iv=  &(p[ n]); n+=k;
485                 exp_label=(unsigned char *)TLS_MD_SERVER_WRITE_KEY_CONST;
486                 exp_label_len=TLS_MD_SERVER_WRITE_KEY_CONST_SIZE;
487                 client_write=0;
488                 }

490         if (n > s->s3->tmp.key_block_length)
491                 {
492                 SSLerr(SSL_F_TLS1_CHANGE_CIPHER_STATE,ERR_R_INTERNAL_ERROR);
493                 goto err2;
494                 }

496         memcpy(mac_secret,ms,i);

498         if (!(EVP_CIPHER_flags(c)&EVP_CIPH_FLAG_AEAD_CIPHER))
499                 {
500                 mac_key = EVP_PKEY_new_mac_key(mac_type, NULL,
501                                 mac_secret,*mac_secret_size);
502                 EVP_DigestSignInit(mac_ctx,NULL,m,NULL,mac_key);
503                 EVP_PKEY_free(mac_key);
504                 }
505 #ifdef TLS_DEBUG
506 printf("which = %04X\nmac key=",which);
507 { int z; for (z=0; z<i; z++) printf("%02X%c",ms[z],((z+1)%16)?' ':'\n'); }
508 #endif
509         if (is_export)
510                 {
511                 /* In here I set both the read and write key/iv to the
512                  * same value since only the correct one will be used :-).
513                  */
514                 if (!tls1_PRF(ssl_get_algorithm2(s),
515                                 exp_label,exp_label_len,
516                                 s->s3->client_random,SSL3_RANDOM_SIZE,
517                                 s->s3->server_random,SSL3_RANDOM_SIZE,
518                                 NULL,0,NULL,0,
519                                 key,j,tmp1,tmp2,EVP_CIPHER_key_length(c)))
520                         goto err2;
521                 key=tmp1;

523                 if (k > 0)
```

```
524                         {
525                         if (!tls1_PRF(ssl_get_algorithm2(s),
526                                       TLS_MD_IV_BLOCK_CONST,TLS_MD_IV_BLOCK_CO
527                                       s->s3->client_random,SSL3_RANDOM_SIZE,
528                                       s->s3->server_random,SSL3_RANDOM_SIZE,
529                                       NULL,0,NULL,0,
530                                       empty,0,iv1,iv2,k*2))
531                                 goto err2;
532                         if (client_write)
533                                 iv=iv1;
534                         else
535                                 iv= &(iv1[k]);
536                         }
537                 }

539         s->session->key_arg_length=0;
540 #ifdef KSSL_DEBUG
541         {
542         int i;
543         printf("EVP_CipherInit_ex(dd,c,key=,iv=,which)\n");
544         printf("\tkey= "); for (i=0; i<c->key_len; i++) printf("%02x", key[i]);
545         printf("\n");
546         printf("\t iv= "); for (i=0; i<c->iv_len; i++) printf("%02x", iv[i]);
547         printf("\n");
548         }
549 #endif  /* KSSL_DEBUG */

551         if (EVP_CIPHER_mode(c) == EVP_CIPH_GCM_MODE)
552                 {
553                 EVP_CipherInit_ex(dd,c,NULL,key,NULL,(which & SSL3_CC_WRITE));
554                 EVP_CIPHER_CTX_ctrl(dd, EVP_CTRL_GCM_SET_IV_FIXED, k, iv);
555                 }
556         else
557                 EVP_CipherInit_ex(dd,c,NULL,key,iv,(which & SSL3_CC_WRITE));

559         /* Needed for "composite" AEADs, such as RC4-HMAC-MD5 */
560         if ((EVP_CIPHER_flags(c)&EVP_CIPH_FLAG_AEAD_CIPHER) && *mac_secret_size)
561                 EVP_CIPHER_CTX_ctrl(dd,EVP_CTRL_AEAD_SET_MAC_KEY,
562                                     *mac_secret_size,mac_secret);

564 #ifdef TLS_DEBUG
565 printf("which = %04X\nkey=",which);
566 { int z; for (z=0; z<EVP_CIPHER_key_length(c); z++) printf("%02X%c",key[z],((z+1
567 printf("\niv=");
568 { int z; for (z=0; z<k; z++) printf("%02X%c",iv[z],((z+1)%16)?' ':'\n'); }
569 printf("\n");
570 #endif

572         OPENSSL_cleanse(tmp1,sizeof(tmp1));
573         OPENSSL_cleanse(tmp2,sizeof(tmp1));
574         OPENSSL_cleanse(iv1,sizeof(iv1));
575         OPENSSL_cleanse(iv2,sizeof(iv2));
576         return(1);
577 err:
578         SSLerr(SSL_F_TLS1_CHANGE_CIPHER_STATE,ERR_R_MALLOC_FAILURE);
579 err2:
580         return(0);
581         }

583 int tls1_setup_key_block(SSL *s)
584         {
585         unsigned char *p1,*p2=NULL;
586         const EVP_CIPHER *c;
587         const EVP_MD *hash;
588         int num;
589         SSL_COMP *comp;
```



```
590         int mac_type= NID_undef,mac_secret_size=0;
591         int ret=0;

593 #ifdef KSSL_DEBUG
594         printf ("tls1_setup_key_block()\n");
595 #endif  /* KSSL_DEBUG */

597         if (s->s3->tmp.key_block_length != 0)
598                 return(1);

600         if (!ssl_cipher_get_evp(s->session,&c,&hash,&mac_type,&mac_secret_size,&
601                 {
602                 SSLerr(SSL_F_TLS1_SETUP_KEY_BLOCK,SSL_R_CIPHER_OR_HASH_UNAVAILAB
603                 return(0);
604                 }

606         s->s3->tmp.new_sym_enc=c;
607         s->s3->tmp.new_hash=hash;
608         s->s3->tmp.new_mac_pkey_type = mac_type;
609         s->s3->tmp.new_mac_secret_size = mac_secret_size;
610         num=EVP_CIPHER_key_length(c)+mac_secret_size+EVP_CIPHER_iv_length(c);
611         num*=2;

613         ssl3_cleanup_key_block(s);

615         if ((p1=(unsigned char *)OPENSSL_malloc(num)) == NULL)
616                 {
617                 SSLerr(SSL_F_TLS1_SETUP_KEY_BLOCK,ERR_R_MALLOC_FAILURE);
618                 goto err;
619                 }

621         s->s3->tmp.key_block_length=num;
622         s->s3->tmp.key_block=p1;

624         if ((p2=(unsigned char *)OPENSSL_malloc(num)) == NULL)
625                 {
626                 SSLerr(SSL_F_TLS1_SETUP_KEY_BLOCK,ERR_R_MALLOC_FAILURE);
627                 goto err;
628                 }

630 #ifdef TLS_DEBUG
631 printf("client random\n");
632 { int z; for (z=0; z<SSL3_RANDOM_SIZE; z++) printf("%02X%c",s->s3->client_random
633 printf("server random\n");
634 { int z; for (z=0; z<SSL3_RANDOM_SIZE; z++) printf("%02X%c",s->s3->server_random
635 printf("pre-master\n");
636 { int z; for (z=0; z<s->session->master_key_length; z++) printf("%02X%c",s->sess
637 #endif
638         if (!tls1_generate_key_block(s,p1,p2,num))
639                 goto err;
640 #ifdef TLS_DEBUG
641 printf("\nkey block\n");
642 { int z; for (z=0; z<num; z++) printf("%02X%c",p1[z],((z+1)%16)?' ':'\n'); }
643 #endif

645         if (!(s->options & SSL_OP_DONT_INSERT_EMPTY_FRAGMENTS)
646                 && s->method->version <= TLS1_VERSION)
647                 {
648                 /* enable vulnerability countermeasure for CBC ciphers with
649                  * known-IV problem (http://www.openssl.org/~bodo/tls-cbc.txt)
650                  */
651                 s->s3->need_empty_fragments = 1;

653                 if (s->session->cipher != NULL)
654                         {
655                         if (s->session->cipher->algorithm_enc == SSL_eNULL)
```

```
656                                 s->s3->need_empty_fragments = 0;

658 #ifndef OPENSSL_NO_RC4
659                         if (s->session->cipher->algorithm_enc == SSL_RC4)
660                                 s->s3->need_empty_fragments = 0;
661 #endif
662                         }
663                 }

665         ret = 1;
666 err:
667         if (p2)
668                 {
669                 OPENSSL_cleanse(p2,num);
670                 OPENSSL_free(p2);
671                 }
672         return(ret);
673         }

675 /* tls1_enc encrypts/decrypts the record in |s->wrec| / |s->rrec|, respectively.
676  *
677  * Returns:
678  *   0: (in non-constant time) if the record is publically invalid (i.e. too
679  *       short etc).
680  *   1: if the record's padding is valid / the encryption was successful.
681  *   -1: if the record's padding/AEAD-authenticator is invalid or, if sending,
682  *       an internal error occured.
683  */
684 int tls1_enc(SSL *s, int send)
685         {
686         SSL3_RECORD *rec;
687         EVP_CIPHER_CTX *ds;
688         unsigned long l;
689         int bs,i,j,k,pad=0,ret,mac_size=0;
690         const EVP_CIPHER *enc;

692         if (send)
693                 {
694                 if (EVP_MD_CTX_md(s->write_hash))
695                         {
696                         int n=EVP_MD_CTX_size(s->write_hash);
697                         OPENSSL_assert(n >= 0);
698                         }
699                 ds=s->enc_write_ctx;
700                 rec= &(s->s3->wrec);
701                 if (s->enc_write_ctx == NULL)
702                         enc=NULL;
703                 else
704                         {
705                         int ivlen;
706                         enc=EVP_CIPHER_CTX_cipher(s->enc_write_ctx);
707                         /* For TLSv1.1 and later explicit IV */
708                         if (s->version >= TLS1_1_VERSION
709                                 && EVP_CIPHER_mode(enc) == EVP_CIPH_CBC_MODE)
710                                 ivlen = EVP_CIPHER_iv_length(enc);
711                         else
712                                 ivlen = 0;
713                         if (ivlen > 1)
714                                 {
715                                 if ( rec->data != rec->input)
716                                         /* we can't write into the input stream:
717                                          * Can this ever happen?? (steve)
718                                          */
719                                         fprintf(stderr,
720                                                 "%s:%d: rec->data != rec->input\
721                                                  __FILE__, __LINE__);
```



```
722                                 else if (RAND_bytes(rec->input, ivlen) <= 0)
723                                         return -1;
724                                 }
725                         }
726                 }
727         else
728                 {
729                 if (EVP_MD_CTX_md(s->read_hash))
730                         {
731                         int n=EVP_MD_CTX_size(s->read_hash);
732                         OPENSSL_assert(n >= 0);
733                         }
734                 ds=s->enc_read_ctx;
735                 rec= &(s->s3->rrec);
736                 if (s->enc_read_ctx == NULL)
737                         enc=NULL;
738                 else
739                         enc=EVP_CIPHER_CTX_cipher(s->enc_read_ctx);
740                 }

742 #ifdef KSSL_DEBUG
743         printf("tls1_enc(%d)\n", send);
744 #endif    /* KSSL_DEBUG */

746         if ((s->session == NULL) || (ds == NULL) || (enc == NULL))
747                 {
748                 memmove(rec->data,rec->input,rec->length);
749                 rec->input=rec->data;
750                 ret = 1;
751                 }
752         else
753                 {
754                 l=rec->length;
755                 bs=EVP_CIPHER_block_size(ds->cipher);

757                 if (EVP_CIPHER_flags(ds->cipher)&EVP_CIPH_FLAG_AEAD_CIPHER)
758                         {
759                         unsigned char buf[13],*seq;

761                         seq = send?s->s3->write_sequence:s->s3->read_sequence;

763                         if (s->version == DTLS1_VERSION || s->version == DTLS1_B
764                                 {
765                                 unsigned char dtlsseq[9],*p=dtlsseq;

767                                 s2n(send?s->d1->w_epoch:s->d1->r_epoch,p);
768                                 memcpy(p,&seq[2],6);
769                                 memcpy(buf,dtlsseq,8);
770                                 }
771                         else
772                                 {
773                                 memcpy(buf,seq,8);
774                                 for (i=7; i>=0; i--)    /* increment */
775                                         {
776                                         ++seq[i];
777                                         if (seq[i] != 0) break;
778                                         }
779                                 }

781                         buf[8]=rec->type;
782                         buf[9]=(unsigned char)(s->version>>8);
783                         buf[10]=(unsigned char)(s->version);
784                         buf[11]=rec->length>>8;
785                         buf[12]=rec->length&0xff;
786                         pad=EVP_CIPHER_CTX_ctrl(ds,EVP_CTRL_AEAD_TLS1_AAD,13,buf
787                         if (send)
```

```
788                                 {
789                                 l+=pad;
790                                 rec->length+=pad;
791                                 }
792                         }
793                 else if ((bs != 1) && send)
794                         {
795                         i=bs-((int)l%bs);

797                         /* Add weird padding of upto 256 bytes */

799                         /* we need to add 'i' padding bytes of value j */
800                         j=i-1;
801                         if (s->options & SSL_OP_TLS_BLOCK_PADDING_BUG)
802                                 {
803                                 if (s->s3->flags & TLS1_FLAGS_TLS_PADDING_BUG)
804                                         j++;
805                                 }
806                         for (k=(int)l; k<(int)(l+i); k++)
807                                 rec->input[k]=j;
808                         l+=i;
809                         rec->length+=i;
810                         }

812 #ifdef KSSL_DEBUG
813                 {
814                 unsigned long ui;
815                 printf("EVP_Cipher(ds=%p,rec->data=%p,rec->input=%p,l=%ld) ==>\n
816                         ds,rec->data,rec->input,l);
817                 printf("\tEVP_CIPHER_CTX: %d buf_len, %d key_len [%d %d], %d iv_
818                         ds->buf_len, ds->cipher->key_len,
819                         DES_KEY_SZ, DES_SCHEDULE_SZ,
820                         ds->cipher->iv_len);
821                 printf("\t\tIV: ");
822                 for (i=0; i<ds->cipher->iv_len; i++) printf("%02X", ds->iv[i]);
823                 printf("\n");
824                 printf("\trec->input=");
825                 for (ui=0; ui<l; ui++) printf(" %02x", rec->input[ui]);
826                 printf("\n");
827                 }
828 #endif  /* KSSL_DEBUG */

830                 if (!send)
831                         {
832                         if (l == 0 || l%bs != 0)
833                                 return 0;
834                         }

836                 i = EVP_Cipher(ds,rec->data,rec->input,l);
837                 if ((EVP_CIPHER_flags(ds->cipher)&EVP_CIPH_FLAG_CUSTOM_CIPHER)
838                                                 ?(i<0)
839                                                 :(i==0))
840                         return -1;      /* AEAD can fail to verify MAC */
841                 if (EVP_CIPHER_mode(enc) == EVP_CIPH_GCM_MODE && !send)
842                         {
843                         rec->data += EVP_GCM_TLS_EXPLICIT_IV_LEN;
844                         rec->input += EVP_GCM_TLS_EXPLICIT_IV_LEN;
845                         rec->length -= EVP_GCM_TLS_EXPLICIT_IV_LEN;
846                         }

848 #ifdef KSSL_DEBUG
849                 {
850                 unsigned long i;
851                 printf("\trec->data=");
852                 for (i=0; i<l; i++)
853                         printf(" %02x", rec->data[i]);  printf("\n");
```



```
854                 }
855 #endif  /* KSSL_DEBUG */

857                 ret = 1;
858                 if (EVP_MD_CTX_md(s->read_hash) != NULL)
859                         mac_size = EVP_MD_CTX_size(s->read_hash);
860                 if ((bs != 1) && !send)
861                         ret = tls1_cbc_remove_padding(s, rec, bs, mac_size);
862                 if (pad && !send)
863                         rec->length -= pad;
864                 }
865         return ret;
866         }

868 int tls1_cert_verify_mac(SSL *s, int md_nid, unsigned char *out)
869         {
870         unsigned int ret;
871         EVP_MD_CTX ctx, *d=NULL;
872         int i;

874         if (s->s3->handshake_buffer)
875                 if (!ssl3_digest_cached_records(s))
876                         return 0;

878         for (i=0;i<SSL_MAX_DIGEST;i++)
879                 {
880                   if (s->s3->handshake_dgst[i]&&EVP_MD_CTX_type(s->s3->handshake
881                         {
882                         d=s->s3->handshake_dgst[i];
883                         break;
884                         }
885                 }
886         if (!d) {
887                 SSLerr(SSL_F_TLS1_CERT_VERIFY_MAC,SSL_R_NO_REQUIRED_DIGEST);
888                 return 0;
889         }

891         EVP_MD_CTX_init(&ctx);
892         EVP_MD_CTX_copy_ex(&ctx,d);
893         EVP_DigestFinal_ex(&ctx,out,&ret);
894         EVP_MD_CTX_cleanup(&ctx);
895         return((int)ret);
896         }

898 int tls1_final_finish_mac(SSL *s,
899              const char *str, int slen, unsigned char *out)
900         {
901         unsigned int i;
902         EVP_MD_CTX ctx;
903         unsigned char buf[2*EVP_MAX_MD_SIZE];
904         unsigned char *q,buf2[12];
905         int idx;
906         long mask;
907         int err=0;
908         const EVP_MD *md;

910         q=buf;

912         if (s->s3->handshake_buffer)
913                 if (!ssl3_digest_cached_records(s))
914                         return 0;

916         EVP_MD_CTX_init(&ctx);

918         for (idx=0;ssl_get_handshake_digest(idx,&mask,&md);idx++)
919                 {
```

```
920                 if (mask & ssl_get_algorithm2(s))
921                         {
922                         int hashsize = EVP_MD_size(md);
923                         EVP_MD_CTX *hdgst = s->s3->handshake_dgst[idx];
924                         if (!hdgst || hashsize < 0 || hashsize > (int)(sizeof bu
925                                 {
926                                 /* internal error: 'buf' is too small for this c
927                                 err = 1;
928                                 }
929                         else
930                                 {
931                                 if (!EVP_MD_CTX_copy_ex(&ctx, hdgst) ||
932                                         !EVP_DigestFinal_ex(&ctx,q,&i) ||
933                                         (i != (unsigned int)hashsize))
934                                         err = 1;
935                                 q+=hashsize;
936                                 }
937                         }
938                 }
939
940         if (!tls1_PRF(ssl_get_algorithm2(s),
941                         str,slen, buf,(int)(q-buf), NULL,0, NULL,0, NULL,0,
942                         s->session->master_key,s->session->master_key_length,
943                         out,buf2,sizeof buf2))
944                 err = 1;
945         EVP_MD_CTX_cleanup(&ctx);
946
947         if (err)
948                 return 0;
949         else
950                 return sizeof buf2;
951         }
952
953 int tls1_mac(SSL *ssl, unsigned char *md, int send)
954         {
955         SSL3_RECORD *rec;
956         unsigned char *seq;
957         EVP_MD_CTX *hash;
958         size_t md_size, orig_len;
959         int i;
960         EVP_MD_CTX hmac, *mac_ctx;
961         unsigned char header[13];
962         int stream_mac = (send?(ssl->mac_flags & SSL_MAC_FLAG_WRITE_MAC_STREAM):
963         int t;
964
965         if (send)
966                 {
967                 rec= &(ssl->s3->wrec);
968                 seq= &(ssl->s3->write_sequence[0]);
969                 hash=ssl->write_hash;
970                 }
971         else
972                 {
973                 rec= &(ssl->s3->rrec);
974                 seq= &(ssl->s3->read_sequence[0]);
975                 hash=ssl->read_hash;
976                 }
977
978         t=EVP_MD_CTX_size(hash);
979         OPENSSL_assert(t >= 0);
980         md_size=t;
981
982         /* I should fix this up TLS TLS TLS TLS TLS XXXXXXXX */
983         if (stream_mac)
984                 {
985                 mac_ctx = hash;
```



```
986                 }
987                 else
988                 {
989                         if (!EVP_MD_CTX_copy(&hmac,hash))
990                                 return -1;
991                         mac_ctx = &hmac;
992                 }
993
994         if (ssl->version == DTLS1_VERSION || ssl->version == DTLS1_BAD_VER)
995                 {
996                 unsigned char dtlsseq[8],*p=dtlsseq;
997
998                 s2n(send?ssl->d1->w_epoch:ssl->d1->r_epoch, p);
999                 memcpy (p,&seq[2],6);
1000
1001                memcpy(header, dtlsseq, 8);
1002                }
1003        else
1004                memcpy(header, seq, 8);
1005
1006        /* kludge: tls1_cbc_remove_padding passes padding length in rec->type */
1007        orig_len = rec->length+md_size+((unsigned int)rec->type>>8);
1008        rec->type &= 0xff;
1009
1010        header[8]=rec->type;
1011        header[9]=(unsigned char)(ssl->version>>8);
1012        header[10]=(unsigned char)(ssl->version);
1013        header[11]=(rec->length)>>8;
1014        header[12]=(rec->length)&0xff;
1015
1016        if (!send &&
1017            EVP_CIPHER_CTX_mode(ssl->enc_read_ctx) == EVP_CIPH_CBC_MODE &&
1018            ssl3_cbc_record_digest_supported(mac_ctx))
1019                {
1020                /* This is a CBC-encrypted record. We must avoid leaking any
1021                 * timing-side channel information about how many blocks of
1022                 * data we are hashing because that gives an attacker a
1023                 * timing-oracle. */
1024                ssl3_cbc_digest_record(
1025                        mac_ctx,
1026                        md, &md_size,
1027                        header, rec->input,
1028                        rec->length + md_size, orig_len,
1029                        ssl->s3->read_mac_secret,
1030                        ssl->s3->read_mac_secret_size,
1031                        0 /* not SSLv3 */);
1032                }
1033        else
1034                {
1035                EVP_DigestSignUpdate(mac_ctx,header,sizeof(header));
1036                EVP_DigestSignUpdate(mac_ctx,rec->input,rec->length);
1037                t=EVP_DigestSignFinal(mac_ctx,md,&md_size);
1038                OPENSSL_assert(t > 0);
1039 #ifdef OPENSSL_FIPS
1040                if (!send && FIPS_mode())
1041                        tls_fips_digest_extra(
1042                                        ssl->enc_read_ctx,
1043                                        mac_ctx, rec->input,
1044                                        rec->length, orig_len);
1045 #endif
1046                }
1047
1048        if (!stream_mac)
1049                EVP_MD_CTX_cleanup(&hmac);
1050 #ifdef TLS_DEBUG
1051 printf("seq=");
```

```
{int z; for (z=0; z<8; z++) printf("%02X ",seq[z]); printf("\n"); }
printf("rec=");
{unsigned int z; for (z=0; z<rec->length; z++) printf("%02X ",rec->data[z]); pri
#endif

        if (ssl->version != DTLS1_VERSION && ssl->version != DTLS1_BAD_VER)
                {
                for (i=7; i>=0; i--)
                        {
                        ++seq[i];
                        if (seq[i] != 0) break;
                        }
                }

#ifdef TLS_DEBUG
{unsigned int z; for (z=0; z<md_size; z++) printf("%02X ",md[z]); printf("\n");
#endif
        return(md_size);
        }

int tls1_generate_master_secret(SSL *s, unsigned char *out, unsigned char *p,
             int len)
        {
        unsigned char buff[SSL_MAX_MASTER_KEY_LENGTH];
        const void *co = NULL, *so = NULL;
        int col = 0, sol = 0;


#ifdef KSSL_DEBUG
        printf ("tls1_generate_master_secret(%p,%p, %p, %d)\n", s,out, p,len);
#endif  /* KSSL_DEBUG */

#ifdef TLSEXT_TYPE_opaque_prf_input
        if (s->s3->client_opaque_prf_input != NULL && s->s3->server_opaque_prf_i
            s->s3->client_opaque_prf_input_len > 0 &&
            s->s3->client_opaque_prf_input_len == s->s3->server_opaque_prf_input
                {
                co = s->s3->client_opaque_prf_input;
                col = s->s3->server_opaque_prf_input_len;
                so = s->s3->server_opaque_prf_input;
                sol = s->s3->client_opaque_prf_input_len; /* must be same as col
                }
#endif

        tls1_PRF(ssl_get_algorithm2(s),
                TLS_MD_MASTER_SECRET_CONST,TLS_MD_MASTER_SECRET_CONST_SIZE,
                s->s3->client_random,SSL3_RANDOM_SIZE,
                co, col,
                s->s3->server_random,SSL3_RANDOM_SIZE,
                so, sol,
                p,len,
                s->session->master_key,buff,sizeof buff);
#ifdef SSL_DEBUG
        fprintf(stderr, "Premaster Secret:\n");
        BIO_dump_fp(stderr, (char *)p, len);
        fprintf(stderr, "Client Random:\n");
        BIO_dump_fp(stderr, (char *)s->s3->client_random, SSL3_RANDOM_SIZE);
        fprintf(stderr, "Server Random:\n");
        BIO_dump_fp(stderr, (char *)s->s3->server_random, SSL3_RANDOM_SIZE);
        fprintf(stderr, "Master Secret:\n");
        BIO_dump_fp(stderr, (char *)s->session->master_key, SSL3_MASTER_SECRET_S
#endif

#ifdef KSSL_DEBUG
        printf ("tls1_generate_master_secret() complete\n");
#endif  /* KSSL_DEBUG */
```



```
        return(SSL3_MASTER_SECRET_SIZE);
        }

int tls1_export_keying_material(SSL *s, unsigned char *out, size_t olen,
         const char *label, size_t llen, const unsigned char *context,
         size_t contextlen, int use_context)
        {
        unsigned char *buff;
        unsigned char *val = NULL;
        size_t vallen, currentvalpos;
        int rv;

#ifdef KSSL_DEBUG
        printf ("tls1_export_keying_material(%p,%p,%d,%s,%d,%p,%d)\n", s, out, o
#endif  /* KSSL_DEBUG */

        buff = OPENSSL_malloc(olen);
        if (buff == NULL) goto err2;

        /* construct PRF arguments
         * we construct the PRF argument ourself rather than passing separate
         * values into the TLS PRF to ensure that the concatenation of values
         * does not create a prohibited label.
         */
        vallen = llen + SSL3_RANDOM_SIZE * 2;
        if (use_context)
                {
                vallen += 2 + contextlen;
                }

        val = OPENSSL_malloc(vallen);
        if (val == NULL) goto err2;
        currentvalpos = 0;
        memcpy(val + currentvalpos, (unsigned char *) label, llen);
        currentvalpos += llen;
        memcpy(val + currentvalpos, s->s3->client_random, SSL3_RANDOM_SIZE);
        currentvalpos += SSL3_RANDOM_SIZE;
        memcpy(val + currentvalpos, s->s3->server_random, SSL3_RANDOM_SIZE);
        currentvalpos += SSL3_RANDOM_SIZE;

        if (use_context)
                {
                val[currentvalpos] = (contextlen >> 8) & 0xff;
                currentvalpos++;
                val[currentvalpos] = contextlen & 0xff;
                currentvalpos++;
                if ((contextlen > 0) || (context != NULL))
                        {
                        memcpy(val + currentvalpos, context, contextlen);
                        }
                }

        /* disallow prohibited labels
         * note that SSL3_RANDOM_SIZE > max(prohibited label len) =
         * 15, so size of val > max(prohibited label len) = 15 and the
         * comparisons won't have buffer overflow
         */
        if (memcmp(val, TLS_MD_CLIENT_FINISH_CONST,
                TLS_MD_CLIENT_FINISH_CONST_SIZE) == 0) goto err1;
        if (memcmp(val, TLS_MD_SERVER_FINISH_CONST,
                TLS_MD_SERVER_FINISH_CONST_SIZE) == 0) goto err1;
        if (memcmp(val, TLS_MD_MASTER_SECRET_CONST,
                TLS_MD_MASTER_SECRET_CONST_SIZE) == 0) goto err1;
        if (memcmp(val, TLS_MD_KEY_EXPANSION_CONST,
                TLS_MD_KEY_EXPANSION_CONST_SIZE) == 0) goto err1;
```

```
1184         rv = tls1_PRF(ssl_get_algorithm2(s),
1185                       val, vallen,
1186                       NULL, 0,
1187                       NULL, 0,
1188                       NULL, 0,
1189                       NULL, 0,
1190                       s->session->master_key,s->session->master_key_length,
1191                       out,buff,olen);

1193 #ifdef KSSL_DEBUG
1194         printf ("tls1_export_keying_material() complete\n");
1195 #endif  /* KSSL_DEBUG */
1196         goto ret;
1197 err1:
1198         SSLerr(SSL_F_TLS1_EXPORT_KEYING_MATERIAL, SSL_R_TLS_ILLEGAL_EXPORTER_LAB
1199         rv = 0;
1200         goto ret;
1201 err2:
1202         SSLerr(SSL_F_TLS1_EXPORT_KEYING_MATERIAL, ERR_R_MALLOC_FAILURE);
1203         rv = 0;
1204 ret:
1205         if (buff != NULL) OPENSSL_free(buff);
1206         if (val != NULL) OPENSSL_free(val);
1207         return(rv);
1208         }

1210 int tls1_alert_code(int code)
1211         {
1212         switch (code)
1213                 {
1214         case SSL_AD_CLOSE_NOTIFY:       return(SSL3_AD_CLOSE_NOTIFY);
1215         case SSL_AD_UNEXPECTED_MESSAGE: return(SSL3_AD_UNEXPECTED_MESSAGE);
1216         case SSL_AD_BAD_RECORD_MAC:     return(SSL3_AD_BAD_RECORD_MAC);
1217         case SSL_AD_DECRYPTION_FAILED:  return(TLS1_AD_DECRYPTION_FAILED);
1218         case SSL_AD_RECORD_OVERFLOW:    return(TLS1_AD_RECORD_OVERFLOW);
1219         case SSL_AD_DECOMPRESSION_FAILURE:return(SSL3_AD_DECOMPRESSION_FAILURE);
1220         case SSL_AD_HANDSHAKE_FAILURE:  return(SSL3_AD_HANDSHAKE_FAILURE);
1221         case SSL_AD_NO_CERTIFICATE:     return(-1);
1222         case SSL_AD_BAD_CERTIFICATE:    return(SSL3_AD_BAD_CERTIFICATE);
1223         case SSL_AD_UNSUPPORTED_CERTIFICATE:return(SSL3_AD_UNSUPPORTED_CERTIFICA
1224         case SSL_AD_CERTIFICATE_REVOKED:return(SSL3_AD_CERTIFICATE_REVOKED);
1225         case SSL_AD_CERTIFICATE_EXPIRED:return(SSL3_AD_CERTIFICATE_EXPIRED);
1226         case SSL_AD_CERTIFICATE_UNKNOWN:return(SSL3_AD_CERTIFICATE_UNKNOWN);
1227         case SSL_AD_ILLEGAL_PARAMETER:  return(SSL3_AD_ILLEGAL_PARAMETER);
1228         case SSL_AD_UNKNOWN_CA:         return(TLS1_AD_UNKNOWN_CA);
1229         case SSL_AD_ACCESS_DENIED:      return(TLS1_AD_ACCESS_DENIED);
1230         case SSL_AD_DECODE_ERROR:       return(TLS1_AD_DECODE_ERROR);
1231         case SSL_AD_DECRYPT_ERROR:      return(TLS1_AD_DECRYPT_ERROR);
1232         case SSL_AD_EXPORT_RESTRICTION: return(TLS1_AD_EXPORT_RESTRICTION);
1233         case SSL_AD_PROTOCOL_VERSION:   return(TLS1_AD_PROTOCOL_VERSION);
1234         case SSL_AD_INSUFFICIENT_SECURITY:return(TLS1_AD_INSUFFICIENT_SECURITY);
1235         case SSL_AD_INTERNAL_ERROR:     return(TLS1_AD_INTERNAL_ERROR);
1236         case SSL_AD_USER_CANCELLED:     return(TLS1_AD_USER_CANCELLED);
1237         case SSL_AD_NO_RENEGOTIATION:   return(TLS1_AD_NO_RENEGOTIATION);
1238         case SSL_AD_UNSUPPORTED_EXTENSION: return(TLS1_AD_UNSUPPORTED_EXTENSION)
1239         case SSL_AD_CERTIFICATE_UNOBTAINABLE: return(TLS1_AD_CERTIFICATE_UNOBTAI
1240         case SSL_AD_UNRECOGNIZED_NAME:  return(TLS1_AD_UNRECOGNIZED_NAME);
1241         case SSL_AD_BAD_CERTIFICATE_STATUS_RESPONSE: return(TLS1_AD_BAD_CERTIFIC
1242         case SSL_AD_BAD_CERTIFICATE_HASH_VALUE: return(TLS1_AD_BAD_CERTIFICATE_H
1243         case SSL_AD_UNKNOWN_PSK_IDENTITY:return(TLS1_AD_UNKNOWN_PSK_IDENTITY);
1244 #if 0 /* not appropriate for TLS, not used for DTLS */
1245         case DTLS1_AD_MISSING_HANDSHAKE_MESSAGE: return
1246                                         (DTLS1_AD_MISSING_HANDSHAKE_MESSAGE);
1247 #endif
1248         default:                        return(-1);
1249                 }
```



```
1250         }
1251 #endif /* ! codereview */
```

```
**********************************************************
   75311 Wed Aug 13 19:53:41 2014
new/usr/src/lib/openssl/libsunw_ssl/t1_lib.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
/* ssl/t1_lib.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2007 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */

#include <stdio.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/hmac.h>
#include <openssl/ocsp.h>
#include <openssl/rand.h>
#include "ssl_locl.h"

const char tls1_version_str[]="TLSv1" OPENSSL_VERSION_PTEXT;

#ifndef OPENSSL_NO_TLSEXT
static int tls_decrypt_ticket(SSL *s, const unsigned char *tick, int ticklen,
                              const unsigned char *sess_id, int sesslen,
                              SSL_SESSION **psess);
#endif
```

```
128 SSL3_ENC_METHOD TLSv1_enc_data={
129         tls1_enc,
130         tls1_mac,
131         tls1_setup_key_block,
132         tls1_generate_master_secret,
133         tls1_change_cipher_state,
134         tls1_final_finish_mac,
135         TLS1_FINISH_MAC_LENGTH,
136         tls1_cert_verify_mac,
137         TLS_MD_CLIENT_FINISH_CONST,TLS_MD_CLIENT_FINISH_CONST_SIZE,
138         TLS_MD_SERVER_FINISH_CONST,TLS_MD_SERVER_FINISH_CONST_SIZE,
139         tls1_alert_code,
140         tls1_export_keying_material,
141         };

143 long tls1_default_timeout(void)
144         {
145         /* 2 hours, the 24 hours mentioned in the TLSv1 spec
146          * is way too long for http, the cache would over fill */
147         return(60*60*2);
148         }

150 int tls1_new(SSL *s)
151         {
152         if (!ssl3_new(s)) return(0);
153         s->method->ssl_clear(s);
154         return(1);
155         }

157 void tls1_free(SSL *s)
158         {
159 #ifndef OPENSSL_NO_TLSEXT
160         if (s->tlsext_session_ticket)
161                 {
162                 OPENSSL_free(s->tlsext_session_ticket);
163                 }
164 #endif /* OPENSSL_NO_TLSEXT */
165         ssl3_free(s);
166         }

168 void tls1_clear(SSL *s)
169         {
170         ssl3_clear(s);
171         s->version = s->method->version;
172         }

174 #ifndef OPENSSL_NO_EC

176 static int nid_list[] =
177         {
178                 NID_sect163k1, /* sect163k1 (1) */
179                 NID_sect163r1, /* sect163r1 (2) */
180                 NID_sect163r2, /* sect163r2 (3) */
181                 NID_sect193r1, /* sect193r1 (4) */
182                 NID_sect193r2, /* sect193r2 (5) */
183                 NID_sect233k1, /* sect233k1 (6) */
184                 NID_sect233r1, /* sect233r1 (7) */
185                 NID_sect239k1, /* sect239k1 (8) */
186                 NID_sect283k1, /* sect283k1 (9) */
187                 NID_sect283r1, /* sect283r1 (10) */
188                 NID_sect409k1, /* sect409k1 (11) */
189                 NID_sect409r1, /* sect409r1 (12) */
190                 NID_sect571k1, /* sect571k1 (13) */
191                 NID_sect571r1, /* sect571r1 (14) */
192                 NID_secp160k1, /* secp160k1 (15) */
193                 NID_secp160r1, /* secp160r1 (16) */
```



```
194                 NID_secp160r2, /* secp160r2 (17) */
195                 NID_secp192k1, /* secp192k1 (18) */
196                 NID_X9_62_prime192v1, /* secp192r1 (19) */
197                 NID_secp224k1, /* secp224k1 (20) */
198                 NID_secp224r1, /* secp224r1 (21) */
199                 NID_secp256k1, /* secp256k1 (22) */
200                 NID_X9_62_prime256v1, /* secp256r1 (23) */
201                 NID_secp384r1, /* secp384r1 (24) */
202                 NID_secp521r1  /* secp521r1 (25) */
203         };

205 static int pref_list[] =
206         {
207                 NID_sect571r1, /* sect571r1 (14) */
208                 NID_sect571k1, /* sect571k1 (13) */
209                 NID_secp521r1, /* secp521r1 (25) */
210                 NID_sect409k1, /* sect409k1 (11) */
211                 NID_sect409r1, /* sect409r1 (12) */
212                 NID_secp384r1, /* secp384r1 (24) */
213                 NID_sect283k1, /* sect283k1 (9) */
214                 NID_sect283r1, /* sect283r1 (10) */
215                 NID_secp256k1, /* secp256k1 (22) */
216                 NID_X9_62_prime256v1, /* secp256r1 (23) */
217                 NID_sect239k1, /* sect239k1 (8) */
218                 NID_sect233k1, /* sect233k1 (6) */
219                 NID_sect233r1, /* sect233r1 (7) */
220                 NID_secp224k1, /* secp224k1 (20) */
221                 NID_secp224r1, /* secp224r1 (21) */
222                 NID_sect193r1, /* sect193r1 (4) */
223                 NID_sect193r2, /* sect193r2 (5) */
224                 NID_secp192k1, /* secp192k1 (18) */
225                 NID_X9_62_prime192v1, /* secp192r1 (19) */
226                 NID_sect163k1, /* sect163k1 (1) */
227                 NID_sect163r1, /* sect163r1 (2) */
228                 NID_sect163r2, /* sect163r2 (3) */
229                 NID_secp160k1, /* secp160k1 (15) */
230                 NID_secp160r1, /* secp160r1 (16) */
231                 NID_secp160r2, /* secp160r2 (17) */
232         };

234 int tls1_ec_curve_id2nid(int curve_id)
235         {
236         /* ECC curves from draft-ietf-tls-ecc-12.txt (Oct. 17, 2005) */
237         if ((curve_id < 1) || ((unsigned int)curve_id >
238                                 sizeof(nid_list)/sizeof(nid_list[0])))
239                 return 0;
240         return nid_list[curve_id-1];
241         }

243 int tls1_ec_nid2curve_id(int nid)
244         {
245         /* ECC curves from draft-ietf-tls-ecc-12.txt (Oct. 17, 2005) */
246         switch (nid)
247                 {
248         case NID_sect163k1: /* sect163k1 (1) */
249                 return 1;
250         case NID_sect163r1: /* sect163r1 (2) */
251                 return 2;
252         case NID_sect163r2: /* sect163r2 (3) */
253                 return 3;
254         case NID_sect193r1: /* sect193r1 (4) */
255                 return 4;
256         case NID_sect193r2: /* sect193r2 (5) */
257                 return 5;
258         case NID_sect233k1: /* sect233k1 (6) */
259                 return 6;
```

```
260         case NID_sect233r1: /* sect233r1 (7) */
261                 return 7;
262         case NID_sect239k1: /* sect239k1 (8) */
263                 return 8;
264         case NID_sect283k1: /* sect283k1 (9) */
265                 return 9;
266         case NID_sect283r1: /* sect283r1 (10) */
267                 return 10;
268         case NID_sect409k1: /* sect409k1 (11) */
269                 return 11;
270         case NID_sect409r1: /* sect409r1 (12) */
271                 return 12;
272         case NID_sect571k1: /* sect571k1 (13) */
273                 return 13;
274         case NID_sect571r1: /* sect571r1 (14) */
275                 return 14;
276         case NID_secp160k1: /* secp160k1 (15) */
277                 return 15;
278         case NID_secp160r1: /* secp160r1 (16) */
279                 return 16;
280         case NID_secp160r2: /* secp160r2 (17) */
281                 return 17;
282         case NID_secp192k1: /* secp192k1 (18) */
283                 return 18;
284         case NID_X9_62_prime192v1: /* secp192r1 (19) */
285                 return 19;
286         case NID_secp224k1: /* secp224k1 (20) */
287                 return 20;
288         case NID_secp224r1: /* secp224r1 (21) */
289                 return 21;
290         case NID_secp256k1: /* secp256k1 (22) */
291                 return 22;
292         case NID_X9_62_prime256v1: /* secp256r1 (23) */
293                 return 23;
294         case NID_secp384r1: /* secp384r1 (24) */
295                 return 24;
296         case NID_secp521r1:  /* secp521r1 (25) */
297                 return 25;
298         default:
299                 return 0;
300                 }
301         }
302 #endif /* OPENSSL_NO_EC */

304 #ifndef OPENSSL_NO_TLSEXT

306 /* List of supported signature algorithms and hashes. Should make this
307  * customisable at some point, for now include everything we support.
308  */

310 #ifdef OPENSSL_NO_RSA
311 #define tlsext_sigalg_rsa(md) /* */
312 #else
313 #define tlsext_sigalg_rsa(md) md, TLSEXT_signature_rsa,
314 #endif

316 #ifdef OPENSSL_NO_DSA
317 #define tlsext_sigalg_dsa(md) /* */
318 #else
319 #define tlsext_sigalg_dsa(md) md, TLSEXT_signature_dsa,
320 #endif

322 #ifdef OPENSSL_NO_ECDSA
323 #define tlsext_sigalg_ecdsa(md) /* */
324 #else
325 #define tlsext_sigalg_ecdsa(md) md, TLSEXT_signature_ecdsa,
```



```
326 #endif

328 #define tlsext_sigalg(md) \
329                 tlsext_sigalg_rsa(md) \
330                 tlsext_sigalg_dsa(md) \
331                 tlsext_sigalg_ecdsa(md)

333 static unsigned char tls12_sigalgs[] = {
334 #ifndef OPENSSL_NO_SHA512
335         tlsext_sigalg(TLSEXT_hash_sha512)
336         tlsext_sigalg(TLSEXT_hash_sha384)
337 #endif
338 #ifndef OPENSSL_NO_SHA256
339         tlsext_sigalg(TLSEXT_hash_sha256)
340         tlsext_sigalg(TLSEXT_hash_sha224)
341 #endif
342 #ifndef OPENSSL_NO_SHA
343         tlsext_sigalg(TLSEXT_hash_sha1)
344 #endif
345 };

347 int tls12_get_req_sig_algs(SSL *s, unsigned char *p)
348         {
349         size_t slen = sizeof(tls12_sigalgs);
350         if (p)
351                 memcpy(p, tls12_sigalgs, slen);
352         return (int)slen;
353         }

355 unsigned char *ssl_add_clienthello_tlsext(SSL *s, unsigned char *buf, unsigned c
356         {
357         int extdatalen=0;
358         unsigned char *orig = buf;
359         unsigned char *ret = buf;

361         /* don't add extensions for SSLv3 unless doing secure renegotiation */
362         if (s->client_version == SSL3_VERSION
363                                         && !s->s3->send_connection_binding)
364                 return orig;

366         ret+=2;

368         if (ret>=limit) return NULL; /* this really never occurs, but ... */

370         if (s->tlsext_hostname != NULL)
371                 {
372                 /* Add TLS extension servername to the Client Hello message */
373                 unsigned long size_str;
374                 long lenmax;

376                 /* check for enough space.
377                    4 for the servername type and entension length
378                    2 for servernamelist length
379                    1 for the hostname type
380                    2 for hostname length
381                    + hostname length
382                 */

384                 if ((lenmax = limit - ret - 9) < 0
385                     || (size_str = strlen(s->tlsext_hostname)) > (unsigned long)
386                         return NULL;

388                 /* extension type and length */
389                 s2n(TLSEXT_TYPE_server_name,ret);
390                 s2n(size_str+5,ret);
```

```
392                 /* length of servername list */
393                 s2n(size_str+3,ret);
395                 /* hostname type, length and hostname */
396                 *(ret++) = (unsigned char) TLSEXT_NAMETYPE_host_name;
397                 s2n(size_str,ret);
398                 memcpy(ret, s->tlsext_hostname, size_str);
399                 ret+=size_str;
400                 }
402         /* Add RI if renegotiating */
403         if (s->renegotiate)
404           {
405           int el;
407           if(!ssl_add_clienthello_renegotiate_ext(s, 0, &el, 0))
408               {
409               SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT, ERR_R_INTERNAL_ERROR);
410               return NULL;
411               }
413           if((limit - ret - 4 - el) < 0) return NULL;
415           s2n(TLSEXT_TYPE_renegotiate,ret);
416           s2n(el,ret);
418           if(!ssl_add_clienthello_renegotiate_ext(s, ret, &el, el))
419               {
420               SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT, ERR_R_INTERNAL_ERROR);
421               return NULL;
422               }
424           ret += el;
425         }
427 #ifndef OPENSSL_NO_SRP
428         /* Add SRP username if there is one */
429         if (s->srp_ctx.login != NULL)
430                 { /* Add TLS extension SRP username to the Client Hello message
432                 int login_len = strlen(s->srp_ctx.login);
433                 if (login_len > 255 || login_len == 0)
434                         {
435                         SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT, ERR_R_INTERNAL_
436                         return NULL;
437                         }
439                 /* check for enough space.
440                    4 for the srp type type and entension length
441                    1 for the srp user identity
442                    + srp user identity length
443                 */
444                 if ((limit - ret - 5 - login_len) < 0) return NULL;
446                 /* fill in the extension */
447                 s2n(TLSEXT_TYPE_srp,ret);
448                 s2n(login_len+1,ret);
449                 (*ret++) = (unsigned char) login_len;
450                 memcpy(ret, s->srp_ctx.login, login_len);
451                 ret+=login_len;
452                 }
453 #endif
455 #ifndef OPENSSL_NO_EC
456         if (s->tlsext_ecpointformatlist != NULL)
457                 {
```



```
458                 /* Add TLS extension ECPointFormats to the ClientHello message *
459                 long lenmax;
461                 if ((lenmax = limit - ret - 5) < 0) return NULL;
462                 if (s->tlsext_ecpointformatlist_length > (unsigned long)lenmax)
463                 if (s->tlsext_ecpointformatlist_length > 255)
464                         {
465                         SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT, ERR_R_INTERNAL_
466                         return NULL;
467                         }
469                 s2n(TLSEXT_TYPE_ec_point_formats,ret);
470                 s2n(s->tlsext_ecpointformatlist_length + 1,ret);
471                 *(ret++) = (unsigned char) s->tlsext_ecpointformatlist_length;
472                 memcpy(ret, s->tlsext_ecpointformatlist, s->tlsext_ecpointformat
473                 ret+=s->tlsext_ecpointformatlist_length;
474                 }
475         if (s->tlsext_ellipticcurvelist != NULL)
476                 {
477                 /* Add TLS extension EllipticCurves to the ClientHello message *
478                 long lenmax;
480                 if ((lenmax = limit - ret - 6) < 0) return NULL;
481                 if (s->tlsext_ellipticcurvelist_length > (unsigned long)lenmax)
482                 if (s->tlsext_ellipticcurvelist_length > 65532)
483                         {
484                         SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT, ERR_R_INTERNAL_
485                         return NULL;
486                         }
488                 s2n(TLSEXT_TYPE_elliptic_curves,ret);
489                 s2n(s->tlsext_ellipticcurvelist_length + 2, ret);
491                 /* NB: draft-ietf-tls-ecc-12.txt uses a one-byte prefix for
492                  * elliptic_curve_list, but the examples use two bytes.
493                  * http://www1.ietf.org/mail-archive/web/tls/current/msg00538.ht
494                  * resolves this to two bytes.
495                  */
496                 s2n(s->tlsext_ellipticcurvelist_length, ret);
497                 memcpy(ret, s->tlsext_ellipticcurvelist, s->tlsext_ellipticcurve
498                 ret+=s->tlsext_ellipticcurvelist_length;
499                 }
500 #endif /* OPENSSL_NO_EC */
502         if (!(SSL_get_options(s) & SSL_OP_NO_TICKET))
503                 {
504                 int ticklen;
505                 if (!s->new_session && s->session && s->session->tlsext_tick)
506                         ticklen = s->session->tlsext_ticklen;
507                 else if (s->session && s->tlsext_session_ticket &&
508                          s->tlsext_session_ticket->data)
509                         {
510                         ticklen = s->tlsext_session_ticket->length;
511                         s->session->tlsext_tick = OPENSSL_malloc(ticklen);
512                         if (!s->session->tlsext_tick)
513                                 return NULL;
514                         memcpy(s->session->tlsext_tick,
515                                s->tlsext_session_ticket->data,
516                                ticklen);
517                         s->session->tlsext_ticklen = ticklen;
518                         }
519                 else
520                         ticklen = 0;
521                 if (ticklen == 0 && s->tlsext_session_ticket &&
522                     s->tlsext_session_ticket->data == NULL)
523                         goto skip_ext;
```

```
524                 /* Check for enough room 2 for extension type, 2 for len
525                  * rest for ticket
526                  */
527                 if ((long)(limit - ret - 4 - ticklen) < 0) return NULL;
528                 s2n(TLSEXT_TYPE_session_ticket,ret);
529                 s2n(ticklen,ret);
530                 if (ticklen)
531                         {
532                         memcpy(ret, s->session->tlsext_tick, ticklen);
533                         ret += ticklen;
534                         }
535                 }
536                 skip_ext:

538         if (TLS1_get_client_version(s) >= TLS1_2_VERSION)
539                 {
540                 if ((size_t)(limit - ret) < sizeof(tls12_sigalgs) + 6)
541                         return NULL;
542                 s2n(TLSEXT_TYPE_signature_algorithms,ret);
543                 s2n(sizeof(tls12_sigalgs) + 2, ret);
544                 s2n(sizeof(tls12_sigalgs), ret);
545                 memcpy(ret, tls12_sigalgs, sizeof(tls12_sigalgs));
546                 ret += sizeof(tls12_sigalgs);
547                 }

549 #ifdef TLSEXT_TYPE_opaque_prf_input
550         if (s->s3->client_opaque_prf_input != NULL &&
551             s->version != DTLS1_VERSION)
552                 {
553                 size_t col = s->s3->client_opaque_prf_input_len;

555                 if ((long)(limit - ret - 6 - col < 0))
556                         return NULL;
557                 if (col > 0xFFFD) /* can't happen */
558                         return NULL;

560                 s2n(TLSEXT_TYPE_opaque_prf_input, ret);
561                 s2n(col + 2, ret);
562                 s2n(col, ret);
563                 memcpy(ret, s->s3->client_opaque_prf_input, col);
564                 ret += col;
565                 }
566 #endif

568         if (s->tlsext_status_type == TLSEXT_STATUSTYPE_ocsp &&
569             s->version != DTLS1_VERSION)
570                 {
571                 int i;
572                 long extlen, idlen, itmp;
573                 OCSP_RESPID *id;

575                 idlen = 0;
576                 for (i = 0; i < sk_OCSP_RESPID_num(s->tlsext_ocsp_ids); i++)
577                         {
578                         id = sk_OCSP_RESPID_value(s->tlsext_ocsp_ids, i);
579                         itmp = i2d_OCSP_RESPID(id, NULL);
580                         if (itmp <= 0)
581                                 return NULL;
582                         idlen += itmp + 2;
583                         }

585                 if (s->tlsext_ocsp_exts)
586                         {
587                         extlen = i2d_X509_EXTENSIONS(s->tlsext_ocsp_exts, NULL);
588                         if (extlen < 0)
589                                 return NULL;
```



```
590                         }
591                 else
592                         extlen = 0;

594                 if ((long)(limit - ret - 7 - extlen - idlen) < 0) return NULL;
595                 s2n(TLSEXT_TYPE_status_request, ret);
596                 if (extlen + idlen > 0xFFF0)
597                         return NULL;
598                 s2n(extlen + idlen + 5, ret);
599                 *(ret++) = TLSEXT_STATUSTYPE_ocsp;
600                 s2n(idlen, ret);
601                 for (i = 0; i < sk_OCSP_RESPID_num(s->tlsext_ocsp_ids); i++)
602                         {
603                         /* save position of id len */
604                         unsigned char *q = ret;
605                         id = sk_OCSP_RESPID_value(s->tlsext_ocsp_ids, i);
606                         /* skip over id len */
607                         ret += 2;
608                         itmp = i2d_OCSP_RESPID(id, &ret);
609                         /* write id len */
610                         s2n(itmp, q);
611                         }
612                 s2n(extlen, ret);
613                 if (extlen > 0)
614                         i2d_X509_EXTENSIONS(s->tlsext_ocsp_exts, &ret);
615                 }

617 #ifndef OPENSSL_NO_HEARTBEATS
618         /* Add Heartbeat extension */
619         if ((limit - ret - 4 - 1) < 0)
620                 return NULL;
621         s2n(TLSEXT_TYPE_heartbeat,ret);
622         s2n(1,ret);
623         /* Set mode:
624          * 1: peer may send requests
625          * 2: peer not allowed to send requests
626          */
627         if (s->tlsext_heartbeat & SSL_TLSEXT_HB_DONT_RECV_REQUESTS)
628                 *(ret++) = SSL_TLSEXT_HB_DONT_SEND_REQUESTS;
629         else
630                 *(ret++) = SSL_TLSEXT_HB_ENABLED;
631 #endif

633 #ifndef OPENSSL_NO_NEXTPROTONEG
634         if (s->ctx->next_proto_select_cb && !s->s3->tmp.finish_md_len)
635                 {
636                 /* The client advertises an emtpy extension to indicate its
637                  * support for Next Protocol Negotiation */
638                 if (limit - ret - 4 < 0)
639                         return NULL;
640                 s2n(TLSEXT_TYPE_next_proto_neg,ret);
641                 s2n(0,ret);
642                 }
643 #endif

645 #ifndef OPENSSL_NO_SRTP
646         if(SSL_get_srtp_profiles(s))
647                 {
648                 int el;

650                 ssl_add_clienthello_use_srtp_ext(s, 0, &el, 0);

652                 if((limit - ret - 4 - el) < 0) return NULL;

654                 s2n(TLSEXT_TYPE_use_srtp,ret);
655                 s2n(el,ret);
```

```
657                 if(ssl_add_clienthello_use_srtp_ext(s, ret, &el, el))
658                         {
659                         SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_TLSEXT, ERR_R_INTERNAL_
660                         return NULL;
661                         }
662                 ret += el;
663                 }
664 #endif
665         /* Add padding to workaround bugs in F5 terminators.
666          * See https://tools.ietf.org/html/draft-agl-tls-padding-03
667          *
668          * NB: because this code works out the length of all existing
669          * extensions it MUST always appear last.
670          */
671         if (s->options & SSL_OP_TLSEXT_PADDING)
672                 {
673                 int hlen = ret - (unsigned char *)s->init_buf->data;
674                 /* The code in s23_clnt.c to build ClientHello messages
675                  * includes the 5-byte record header in the buffer, while
676                  * the code in s3_clnt.c does not.
677                  */
678                 if (s->state == SSL23_ST_CW_CLNT_HELLO_A)
679                         hlen -= 5;
680                 if (hlen > 0xff && hlen < 0x200)
681                         {
682                         hlen = 0x200 - hlen;
683                         if (hlen >= 4)
684                                 hlen -= 4;
685                         else
686                                 hlen = 0;
687
688                         s2n(TLSEXT_TYPE_padding, ret);
689                         s2n(hlen, ret);
690                         memset(ret, 0, hlen);
691                         ret += hlen;
692                         }
693                 }
694
695         if ((extdatalen = ret-orig-2)== 0)
696                 return orig;
697
698         s2n(extdatalen, orig);
699         return ret;
700         }
701
702 unsigned char *ssl_add_serverhello_tlsext(SSL *s, unsigned char *buf, unsigned c
703         {
704         int extdatalen=0;
705         unsigned char *orig = buf;
706         unsigned char *ret = buf;
707 #ifndef OPENSSL_NO_NEXTPROTONEG
708         int next_proto_neg_seen;
709 #endif
710
711         /* don't add extensions for SSLv3, unless doing secure renegotiation */
712         if (s->version == SSL3_VERSION && !s->s3->send_connection_binding)
713                 return orig;
714
715         ret+=2;
716         if (ret>=limit) return NULL; /* this really never occurs, but ... */
717
718         if (!s->hit && s->servername_done == 1 && s->session->tlsext_hostname !=
719                 {
720                 if ((long)(limit - ret - 4) < 0) return NULL;
```



```
722                 s2n(TLSEXT_TYPE_server_name,ret);
723                 s2n(0,ret);
724                 }
725
726         if(s->s3->send_connection_binding)
727         {
728           int el;
729
730           if(!ssl_add_serverhello_renegotiate_ext(s, 0, &el, 0))
731               {
732               SSLerr(SSL_F_SSL_ADD_SERVERHELLO_TLSEXT, ERR_R_INTERNAL_ERROR);
733               return NULL;
734               }
735
736           if((limit - ret - 4 - el) < 0) return NULL;
737
738           s2n(TLSEXT_TYPE_renegotiate,ret);
739           s2n(el,ret);
740
741           if(!ssl_add_serverhello_renegotiate_ext(s, ret, &el, el))
742               {
743               SSLerr(SSL_F_SSL_ADD_SERVERHELLO_TLSEXT, ERR_R_INTERNAL_ERROR);
744               return NULL;
745               }
746
747           ret += el;
748         }
749
750 #ifndef OPENSSL_NO_EC
751         if (s->tlsext_ecpointformatlist != NULL)
752                 {
753                 /* Add TLS extension ECPointFormats to the ServerHello message *
754                 long lenmax;
755
756                 if ((lenmax = limit - ret - 5) < 0) return NULL;
757                 if (s->tlsext_ecpointformatlist_length > (unsigned long)lenmax)
758                 if (s->tlsext_ecpointformatlist_length > 255)
759                         {
760                         SSLerr(SSL_F_SSL_ADD_SERVERHELLO_TLSEXT, ERR_R_INTERNAL_
761                         return NULL;
762                         }
763
764                 s2n(TLSEXT_TYPE_ec_point_formats,ret);
765                 s2n(s->tlsext_ecpointformatlist_length + 1,ret);
766                 *(ret++) = (unsigned char) s->tlsext_ecpointformatlist_length;
767                 memcpy(ret, s->tlsext_ecpointformatlist, s->tlsext_ecpointformat
768                 ret+=s->tlsext_ecpointformatlist_length;
769
770                 }
771         /* Currently the server should not respond with a SupportedCurves extens
772 #endif /* OPENSSL_NO_EC */
773
774         if (s->tlsext_ticket_expected
775                 && !(SSL_get_options(s) & SSL_OP_NO_TICKET))
776                 {
777                 if ((long)(limit - ret - 4) < 0) return NULL;
778                 s2n(TLSEXT_TYPE_session_ticket,ret);
779                 s2n(0,ret);
780                 }
781
782         if (s->tlsext_status_expected)
783                 {
784                 if ((long)(limit - ret - 4) < 0) return NULL;
785                 s2n(TLSEXT_TYPE_status_request,ret);
786                 s2n(0,ret);
787                 }
```

```
789 #ifdef TLSEXT_TYPE_opaque_prf_input
790         if (s->s3->server_opaque_prf_input != NULL &&
791             s->version != DTLS1_VERSION)
792                 {
793                 size_t sol = s->s3->server_opaque_prf_input_len;

795                 if ((long)(limit - ret - 6 - sol) < 0)
796                         return NULL;
797                 if (sol > 0xFFFD) /* can't happen */
798                         return NULL;

800                 s2n(TLSEXT_TYPE_opaque_prf_input, ret);
801                 s2n(sol + 2, ret);
802                 s2n(sol, ret);
803                 memcpy(ret, s->s3->server_opaque_prf_input, sol);
804                 ret += sol;
805                 }
806 #endif

808 #ifndef OPENSSL_NO_SRTP
809         if(s->srtp_profile)
810                 {
811                 int el;

813                 ssl_add_serverhello_use_srtp_ext(s, 0, &el, 0);

815                 if((limit - ret - 4 - el) < 0) return NULL;

817                 s2n(TLSEXT_TYPE_use_srtp,ret);
818                 s2n(el,ret);

820                 if(ssl_add_serverhello_use_srtp_ext(s, ret, &el, el))
821                         {
822                         SSLerr(SSL_F_SSL_ADD_SERVERHELLO_TLSEXT, ERR_R_INTERNAL_
823                         return NULL;
824                         }
825                 ret+=el;
826                 }
827 #endif

829         if (((s->s3->tmp.new_cipher->id & 0xFFFF)==0x80 || (s->s3->tmp.new_ciphe
830                 && (SSL_get_options(s) & SSL_OP_CRYPTOPRO_TLSEXT_BUG))
831                 { const unsigned char cryptopro_ext[36] = {
832                         0xfd, 0xe8, /*65000*/
833                         0x00, 0x20, /*32 bytes length*/
834                         0x30, 0x1e, 0x30, 0x08, 0x06, 0x06, 0x2a, 0x85,
835                         0x03,   0x02, 0x02, 0x09, 0x30, 0x08, 0x06, 0x06,
836                         0x2a, 0x85, 0x03, 0x02, 0x02, 0x16, 0x30, 0x08,
837                         0x06, 0x06, 0x2a, 0x85, 0x03, 0x02, 0x02, 0x17};
838                         if (limit-ret<36) return NULL;
839                         memcpy(ret,cryptopro_ext,36);
840                         ret+=36;

842                 }

844 #ifndef OPENSSL_NO_HEARTBEATS
845         /* Add Heartbeat extension if we've received one */
846         if (s->tlsext_heartbeat & SSL_TLSEXT_HB_ENABLED)
847                 {
848                 if ((limit - ret - 4 - 1) < 0)
849                         return NULL;
850                 s2n(TLSEXT_TYPE_heartbeat,ret);
851                 s2n(1,ret);
852                 /* Set mode:
853                  * 1: peer may send requests
```



```
854                  * 2: peer not allowed to send requests
855                  */
856                 if (s->tlsext_heartbeat & SSL_TLSEXT_HB_DONT_RECV_REQUESTS)
857                         *(ret++) = SSL_TLSEXT_HB_DONT_SEND_REQUESTS;
858                 else
859                         *(ret++) = SSL_TLSEXT_HB_ENABLED;

861                 }
862 #endif

864 #ifndef OPENSSL_NO_NEXTPROTONEG
865         next_proto_neg_seen = s->s3->next_proto_neg_seen;
866         s->s3->next_proto_neg_seen = 0;
867         if (next_proto_neg_seen && s->ctx->next_protos_advertised_cb)
868                 {
869                 const unsigned char *npa;
870                 unsigned int npalen;
871                 int r;

873                 r = s->ctx->next_protos_advertised_cb(s, &npa, &npalen, s->ctx->
874                 if (r == SSL_TLSEXT_ERR_OK)
875                         {
876                         if ((long)(limit - ret - 4 - npalen) < 0) return NULL;
877                         s2n(TLSEXT_TYPE_next_proto_neg,ret);
878                         s2n(npalen,ret);
879                         memcpy(ret, npa, npalen);
880                         ret += npalen;
881                         s->s3->next_proto_neg_seen = 1;
882                         }
883                 }
884 #endif

886         if ((extdatalen = ret-orig-2)== 0)
887                 return orig;

889         s2n(extdatalen, orig);
890         return ret;
891         }

893 #ifndef OPENSSL_NO_EC
894 /* ssl_check_for_safari attempts to fingerprint Safari using OS X
895  * SecureTransport using the TLS extension block in |d|, of length |n|.
896  * Safari, since 10.6, sends exactly these extensions, in this order:
897  *   SNI,
898  *   elliptic_curves
899  *   ec_point_formats
900  *
901  * We wish to fingerprint Safari because they broke ECDHE-ECDSA support in 10.8,
902  * but they advertise support. So enabling ECDHE-ECDSA ciphers breaks them.
903  * Sadly we cannot differentiate 10.6, 10.7 and 10.8.4 (which work), from
904  * 10.8..10.8.3 (which don't work).
905  */
906 static void ssl_check_for_safari(SSL *s, const unsigned char *data, const unsign
907         unsigned short type, size;
908         static const unsigned char kSafariExtensionsBlock[] = {
909                 0x00, 0x0a,  /* elliptic_curves extension */
910                 0x00, 0x08,  /* 8 bytes */
911                 0x00, 0x06,  /* 6 bytes of curve ids */
912                 0x00, 0x17,  /* P-256 */
913                 0x00, 0x18,  /* P-384 */
914                 0x00, 0x19,  /* P-521 */

916                 0x00, 0x0b,  /* ec_point_formats */
917                 0x00, 0x02,  /* 2 bytes */
918                 0x01,        /* 1 point format */
919                 0x00,        /* uncompressed */
```

```
920         };

922         /* The following is only present in TLS 1.2 */
923         static const unsigned char kSafariTLS12ExtensionsBlock[] = {
924                 0x00, 0x0d,  /* signature_algorithms */
925                 0x00, 0x0c,  /* 12 bytes */
926                 0x00, 0x0a,  /* 10 bytes */
927                 0x05, 0x01,  /* SHA-384/RSA */
928                 0x04, 0x01,  /* SHA-256/RSA */
929                 0x02, 0x01,  /* SHA-1/RSA */
930                 0x04, 0x03,  /* SHA-256/ECDSA */
931                 0x02, 0x03,  /* SHA-1/ECDSA */
932         };

934         if (data >= (d+n-2))
935                 return;
936         data += 2;

938         if (data > (d+n-4))
939                 return;
940         n2s(data,type);
941         n2s(data,size);

943         if (type != TLSEXT_TYPE_server_name)
944                 return;

946         if (data+size > d+n)
947                 return;
948         data += size;

950         if (TLS1_get_client_version(s) >= TLS1_2_VERSION)
951                 {
952                 const size_t len1 = sizeof(kSafariExtensionsBlock);
953                 const size_t len2 = sizeof(kSafariTLS12ExtensionsBlock);

955                 if (data + len1 + len2 != d+n)
956                         return;
957                 if (memcmp(data, kSafariExtensionsBlock, len1) != 0)
958                         return;
959                 if (memcmp(data + len1, kSafariTLS12ExtensionsBlock, len2) != 0)
960                         return;
961                 }
962         else
963                 {
964                 const size_t len = sizeof(kSafariExtensionsBlock);

966                 if (data + len != d+n)
967                         return;
968                 if (memcmp(data, kSafariExtensionsBlock, len) != 0)
969                         return;
970                 }

972         s->s3->is_probably_safari = 1;
973 }
974 #endif /* !OPENSSL_NO_EC */

976 int ssl_parse_clienthello_tlsext(SSL *s, unsigned char **p, unsigned char *d, in
977         {
978         unsigned short type;
979         unsigned short size;
980         unsigned short len;
981         unsigned char *data = *p;
982         int renegotiate_seen = 0;
983         int sigalg_seen = 0;

985         s->servername_done = 0;
```



```
986         s->tlsext_status_type = -1;
987 #ifndef OPENSSL_NO_NEXTPROTONEG
988         s->s3->next_proto_neg_seen = 0;
989 #endif

991 #ifndef OPENSSL_NO_HEARTBEATS
992         s->tlsext_heartbeat &= ~(SSL_TLSEXT_HB_ENABLED |
993                                  SSL_TLSEXT_HB_DONT_SEND_REQUESTS);
994 #endif

996 #ifndef OPENSSL_NO_EC
997         if (s->options & SSL_OP_SAFARI_ECDHE_ECDSA_BUG)
998                 ssl_check_for_safari(s, data, d, n);
999 #endif /* !OPENSSL_NO_EC */

1001        if (data >= (d+n-2))
1002                goto ri_check;
1003        n2s(data,len);

1005        if (data > (d+n-len))
1006                goto ri_check;

1008        while (data <= (d+n-4))
1009                {
1010                n2s(data,type);
1011                n2s(data,size);

1013                if (data+size > (d+n))
1014                        goto ri_check;
1015 #if 0
1016                fprintf(stderr,"Received extension type %d size %d\n",type,size)
1017 #endif
1018                if (s->tlsext_debug_cb)
1019                        s->tlsext_debug_cb(s, 0, type, data, size,
1020                                                   s->tlsext_debug_arg);
1021 /* The servername extension is treated as follows:

1023    - Only the hostname type is supported with a maximum length of 255.
1024    - The servername is rejected if too long or if it contains zeros,
1025      in which case an fatal alert is generated.
1026    - The servername field is maintained together with the session cache.
1027    - When a session is resumed, the servername call back invoked in order
1028      to allow the application to position itself to the right context.
1029    - The servername is acknowledged if it is new for a session or when
1030      it is identical to a previously used for the same session.
1031      Applications can control the behaviour.  They can at any time
1032      set a 'desirable' servername for a new SSL object. This can be the
1033      case for example with HTTPS when a Host: header field is received and
1034      a renegotiation is requested. In this case, a possible servername
1035      presented in the new client hello is only acknowledged if it matches
1036      the value of the Host: field.
1037    - Applications must  use SSL_OP_NO_SESSION_RESUMPTION_ON_RENEGOTIATION
1038      if they provide for changing an explicit servername context for the session
1039      i.e. when the session has been established with a servername extension.
1040    - On session reconnect, the servername extension may be absent.

1042 */

1044                if (type == TLSEXT_TYPE_server_name)
1045                        {
1046                        unsigned char *sdata;
1047                        int servname_type;
1048                        int dsize;

1050                        if (size < 2)
1051                                {
```

```
1052                                    *al = SSL_AD_DECODE_ERROR;
1053                                    return 0;
1054                                    }
1055                            n2s(data,dsize);
1056                            size -= 2;
1057                            if (dsize > size  )
1058                                    {
1059                                    *al = SSL_AD_DECODE_ERROR;
1060                                    return 0;
1061                                    }

1063                            sdata = data;
1064                            while (dsize > 3)
1065                                    {
1066                                    servname_type = *(sdata++);
1067                                    n2s(sdata,len);
1068                                    dsize -= 3;

1070                                    if (len > dsize)
1071                                            {
1072                                            *al = SSL_AD_DECODE_ERROR;
1073                                            return 0;
1074                                            }
1075                                    if (s->servername_done == 0)
1076                                    switch (servname_type)
1077                                            {
1078                                    case TLSEXT_NAMETYPE_host_name:
1079                                            if (!s->hit)
1080                                                    {
1081                                                    if(s->session->tlsext_hostname)
1082                                                            {
1083                                                            *al = SSL_AD_DECODE_ERRO
1084                                                            return 0;
1085                                                            }
1086                                                    if (len > TLSEXT_MAXLEN_host_nam
1087                                                            {
1088                                                            *al = TLS1_AD_UNRECOGNIZ
1089                                                            return 0;
1090                                                            }
1091                                                    if ((s->session->tlsext_hostname
1092                                                            {
1093                                                            *al = TLS1_AD_INTERNAL_E
1094                                                            return 0;
1095                                                            }
1096                                                    memcpy(s->session->tlsext_hostna
1097                                                    s->session->tlsext_hostname[len]
1098                                                    if (strlen(s->session->tlsext_ho
1099                                                            OPENSSL_free(s->session-
1100                                                            s->session->tlsext_hostn
1101                                                            *al = TLS1_AD_UNRECOGNIZ
1102                                                            return 0;
1103                                                    }
1104                                                    s->servername_done = 1;

1106                                                    }
1107                                            else
1108                                                    s->servername_done = s->session-
1109                                                            && strlen(s->session->tl
1110                                                            && strncmp(s->session->t

1112                                            break;

1114                                    default:
1115                                            break;
1116                                            }
```



```
1118                                    dsize -= len;
1119                                    }
1120                            if (dsize != 0)
1121                                    {
1122                                    *al = SSL_AD_DECODE_ERROR;
1123                                    return 0;
1124                                    }

1126                            }
1127 #ifndef OPENSSL_NO_SRP
1128                    else if (type == TLSEXT_TYPE_srp)
1129                            {
1130                            if (size <= 0 || ((len = data[0])) != (size -1))
1131                                    {
1132                                    *al = SSL_AD_DECODE_ERROR;
1133                                    return 0;
1134                                    }
1135                            if (s->srp_ctx.login != NULL)
1136                                    {
1137                                    *al = SSL_AD_DECODE_ERROR;
1138                                    return 0;
1139                                    }
1140                            if ((s->srp_ctx.login = OPENSSL_malloc(len+1)) == NULL)
1141                                    return -1;
1142                            memcpy(s->srp_ctx.login, &data[1], len);
1143                            s->srp_ctx.login[len]='\0';

1145                            if (strlen(s->srp_ctx.login) != len)
1146                                    {
1147                                    *al = SSL_AD_DECODE_ERROR;
1148                                    return 0;
1149                                    }
1150                            }
1151 #endif

1153 #ifndef OPENSSL_NO_EC
1154                    else if (type == TLSEXT_TYPE_ec_point_formats)
1155                            {
1156                            unsigned char *sdata = data;
1157                            int ecpointformatlist_length = *(sdata++);

1159                            if (ecpointformatlist_length != size - 1)
1160                                    {
1161                                    *al = TLS1_AD_DECODE_ERROR;
1162                                    return 0;
1163                                    }
1164                            if (!s->hit)
1165                                    {
1166                                    if(s->session->tlsext_ecpointformatlist)
1167                                            {
1168                                            OPENSSL_free(s->session->tlsext_ecpointf
1169                                            s->session->tlsext_ecpointformatlist = N
1170                                            }
1171                                    s->session->tlsext_ecpointformatlist_length = 0;
1172                                    if ((s->session->tlsext_ecpointformatlist = OPEN
1173                                            {
1174                                            *al = TLS1_AD_INTERNAL_ERROR;
1175                                            return 0;
1176                                            }
1177                                    s->session->tlsext_ecpointformatlist_length = ec
1178                                    memcpy(s->session->tlsext_ecpointformatlist, sda
1179                                    }
1180 #if 0
1181                            fprintf(stderr,"ssl_parse_clienthello_tlsext s->session-
1182                            sdata = s->session->tlsext_ecpointformatlist;
1183                            for (i = 0; i < s->session->tlsext_ecpointformatlist_len
```

```
1184                                fprintf(stderr,"%i ",*(sdata++));
1185                        fprintf(stderr,"\n");
1186 #endif
1187                        }
1188                else if (type == TLSEXT_TYPE_elliptic_curves)
1189                        {
1190                        unsigned char *sdata = data;
1191                        int ellipticcurvelist_length = (*(sdata++) << 8);
1192                        ellipticcurvelist_length += (*(sdata++));

1194                        if (ellipticcurvelist_length != size - 2 ||
1195                                ellipticcurvelist_length < 1)
1196                                {
1197                                *al = TLS1_AD_DECODE_ERROR;
1198                                return 0;
1199                                }
1200                        if (!s->hit)
1201                                {
1202                                if(s->session->tlsext_ellipticcurvelist)
1203                                        {
1204                                        *al = TLS1_AD_DECODE_ERROR;
1205                                        return 0;
1206                                        }
1207                                s->session->tlsext_ellipticcurvelist_length = 0;
1208                                if ((s->session->tlsext_ellipticcurvelist = OPEN
1209                                        {
1210                                        *al = TLS1_AD_INTERNAL_ERROR;
1211                                        return 0;
1212                                        }
1213                                s->session->tlsext_ellipticcurvelist_length = el
1214                                memcpy(s->session->tlsext_ellipticcurvelist, sda
1215                                }
1216 #if 0
1217                        fprintf(stderr,"ssl_parse_clienthello_tlsext s->session-
1218                        sdata = s->session->tlsext_ellipticcurvelist;
1219                        for (i = 0; i < s->session->tlsext_ellipticcurvelist_len
1220                                fprintf(stderr,"%i ",*(sdata++));
1221                        fprintf(stderr,"\n");
1222 #endif
1223                        }
1224 #endif /* OPENSSL_NO_EC */
1225 #ifdef TLSEXT_TYPE_opaque_prf_input
1226                else if (type == TLSEXT_TYPE_opaque_prf_input &&
1227                     s->version != DTLS1_VERSION)
1228                        {
1229                        unsigned char *sdata = data;

1231                        if (size < 2)
1232                                {
1233                                *al = SSL_AD_DECODE_ERROR;
1234                                return 0;
1235                                }
1236                        n2s(sdata, s->s3->client_opaque_prf_input_len);
1237                        if (s->s3->client_opaque_prf_input_len != size - 2)
1238                                {
1239                                *al = SSL_AD_DECODE_ERROR;
1240                                return 0;
1241                                }

1243                        if (s->s3->client_opaque_prf_input != NULL) /* shouldn't
1244                                OPENSSL_free(s->s3->client_opaque_prf_input);
1245                        if (s->s3->client_opaque_prf_input_len == 0)
1246                                s->s3->client_opaque_prf_input = OPENSSL_malloc(
1247                        else
1248                                s->s3->client_opaque_prf_input = BUF_memdup(sdat
1249                        if (s->s3->client_opaque_prf_input == NULL)
```



```
1250                                {
1251                                *al = TLS1_AD_INTERNAL_ERROR;
1252                                return 0;
1253                                }
1254                        }
1255 #endif
1256                else if (type == TLSEXT_TYPE_session_ticket)
1257                        {
1258                        if (s->tls_session_ticket_ext_cb &&
1259                            !s->tls_session_ticket_ext_cb(s, data, size, s->tls_
1260                                {
1261                                *al = TLS1_AD_INTERNAL_ERROR;
1262                                return 0;
1263                                }
1264                        }
1265                else if (type == TLSEXT_TYPE_renegotiate)
1266                        {
1267                        if(!ssl_parse_clienthello_renegotiate_ext(s, data, size,
1268                                return 0;
1269                        renegotiate_seen = 1;
1270                        }
1271                else if (type == TLSEXT_TYPE_signature_algorithms)
1272                        {
1273                        int dsize;
1274                        if (sigalg_seen || size < 2)
1275                                {
1276                                *al = SSL_AD_DECODE_ERROR;
1277                                return 0;
1278                                }
1279                        sigalg_seen = 1;
1280                        n2s(data,dsize);
1281                        size -= 2;
1282                        if (dsize != size || dsize & 1)
1283                                {
1284                                *al = SSL_AD_DECODE_ERROR;
1285                                return 0;
1286                                }
1287                        if (!tls1_process_sigalgs(s, data, dsize))
1288                                {
1289                                *al = SSL_AD_DECODE_ERROR;
1290                                return 0;
1291                                }
1292                        }
1293                else if (type == TLSEXT_TYPE_status_request &&
1294                         s->version != DTLS1_VERSION)
1295                        {

1297                        if (size < 5)
1298                                {
1299                                *al = SSL_AD_DECODE_ERROR;
1300                                return 0;
1301                                }

1303                        s->tlsext_status_type = *data++;
1304                        size--;
1305                        if (s->tlsext_status_type == TLSEXT_STATUSTYPE_ocsp)
1306                                {
1307                                const unsigned char *sdata;
1308                                int dsize;
1309                                /* Read in responder_id_list */
1310                                n2s(data,dsize);
1311                                size -= 2;
1312                                if (dsize > size  )
1313                                        {
1314                                        *al = SSL_AD_DECODE_ERROR;
1315                                        return 0;
```

```
1316                                    }
1317                                while (dsize > 0)
1318                                    {
1319                                    OCSP_RESPID *id;
1320                                    int idsize;
1321                                    if (dsize < 4)
1322                                            {
1323                                            *al = SSL_AD_DECODE_ERROR;
1324                                            return 0;
1325                                            }
1326                                    n2s(data, idsize);
1327                                    dsize -= 2 + idsize;
1328                                    size -= 2 + idsize;
1329                                    if (dsize < 0)
1330                                            {
1331                                            *al = SSL_AD_DECODE_ERROR;
1332                                            return 0;
1333                                            }
1334                                    sdata = data;
1335                                    data += idsize;
1336                                    id = d2i_OCSP_RESPID(NULL,
1337                                                         &sdata, idsize);
1338                                    if (!id)
1339                                            {
1340                                            *al = SSL_AD_DECODE_ERROR;
1341                                            return 0;
1342                                            }
1343                                    if (data != sdata)
1344                                            {
1345                                            OCSP_RESPID_free(id);
1346                                            *al = SSL_AD_DECODE_ERROR;
1347                                            return 0;
1348                                            }
1349                                    if (!s->tlsext_ocsp_ids
1350                                            && !(s->tlsext_ocsp_ids =
1351                                            sk_OCSP_RESPID_new_null()))
1352                                            {
1353                                            OCSP_RESPID_free(id);
1354                                            *al = SSL_AD_INTERNAL_ERROR;
1355                                            return 0;
1356                                            }
1357                                    if (!sk_OCSP_RESPID_push(
1358                                                    s->tlsext_ocsp_ids, id))
1359                                            {
1360                                            OCSP_RESPID_free(id);
1361                                            *al = SSL_AD_INTERNAL_ERROR;
1362                                            return 0;
1363                                            }
1364                                    }

1366                                /* Read in request_extensions */
1367                                if (size < 2)
1368                                    {
1369                                    *al = SSL_AD_DECODE_ERROR;
1370                                    return 0;
1371                                    }
1372                                n2s(data,dsize);
1373                                size -= 2;
1374                                if (dsize != size)
1375                                    {
1376                                    *al = SSL_AD_DECODE_ERROR;
1377                                    return 0;
1378                                    }
1379                                sdata = data;
1380                                if (dsize > 0)
1381                                    {
```



```
1382                                        if (s->tlsext_ocsp_exts)
1383                                            {
1384                                            sk_X509_EXTENSION_pop_free(s->tl
1385                                                                       X509_
1386                                            }

1388                                        s->tlsext_ocsp_exts =
1389                                            d2i_X509_EXTENSIONS(NULL,
1390                                                &sdata, dsize);
1391                                        if (!s->tlsext_ocsp_exts
1392                                            || (data + dsize != sdata))
1393                                            {
1394                                            *al = SSL_AD_DECODE_ERROR;
1395                                            return 0;
1396                                            }
1397                                        }
1398                                    }
1399                                    /* We don't know what to do with any other type
1400                                    * so ignore it.
1401                                    */
1402                                    else
1403                                        s->tlsext_status_type = -1;
1404                                }
1405 #ifndef OPENSSL_NO_HEARTBEATS
1406                                else if (type == TLSEXT_TYPE_heartbeat)
1407                                    {
1408                                    switch(data[0])
1409                                        {
1410                                        case 0x01:      /* Client allows us to send HB r
1411                                                s->tlsext_heartbeat |= S
1412                                                break;
1413                                        case 0x02:      /* Client doesn't accept HB requ
1414                                                s->tlsext_heartbeat |= S
1415                                                s->tlsext_heartbeat |= S
1416                                                break;
1417                                        default:        *al = SSL_AD_ILLEGAL_PARAMETER;
1418                                                return 0;
1419                                        }
1420                                    }
1421 #endif
1422 #ifndef OPENSSL_NO_NEXTPROTONEG
1423                                else if (type == TLSEXT_TYPE_next_proto_neg &&
1424                                         s->s3->tmp.finish_md_len == 0)
1425                                    {
1426                                    /* We shouldn't accept this extension on a
1427                                     * renegotiation.
1428                                     *
1429                                     * s->new_session will be set on renegotiation, but we
1430                                     * probably shouldn't rely that it couldn't be set on
1431                                     * the initial renegotation too in certain cases (when
1432                                     * there's some other reason to disallow resuming an
1433                                     * earlier session -- the current code won't be doing
1434                                     * anything like that, but this might change).

1436                                     * A valid sign that there's been a previous handshake
1437                                     * in this connection is if s->s3->tmp.finish_md_len >
1438                                     * 0.  (We are talking about a check that will happen
1439                                     * in the Hello protocol round, well before a new
1440                                     * Finished message could have been computed.) */
1441                                    s->s3->next_proto_neg_seen = 1;
1442                                    }
1443 #endif

1445                                /* session ticket processed earlier */
1446 #ifndef OPENSSL_NO_SRTP
1447                                else if (type == TLSEXT_TYPE_use_srtp)
```

```
1448                         {
1449                         if(ssl_parse_clienthello_use_srtp_ext(s, data, size,
1450                                                               al))
1451                                 return 0;
1452                         }
1453 #endif

1455                 data+=size;
1456                 }

1458         *p = data;

1460         ri_check:

1462         /* Need RI if renegotiating */

1464         if (!renegotiate_seen && s->renegotiate &&
1465                 !(s->options & SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION))
1466                 {
1467                 *al = SSL_AD_HANDSHAKE_FAILURE;
1468                 SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_TLSEXT,
1469                                 SSL_R_UNSAFE_LEGACY_RENEGOTIATION_DISABLED);
1470                 return 0;
1471                 }

1473         return 1;
1474         }

1476 #ifndef OPENSSL_NO_NEXTPROTONEG
1477 /* ssl_next_proto_validate validates a Next Protocol Negotiation block. No
1478  * elements of zero length are allowed and the set of elements must exactly fill
1479  * the length of the block. */
1480 static char ssl_next_proto_validate(unsigned char *d, unsigned len)
1481         {
1482         unsigned int off = 0;

1484         while (off < len)
1485                 {
1486                 if (d[off] == 0)
1487                         return 0;
1488                 off += d[off];
1489                 off++;
1490                 }

1492         return off == len;
1493         }
1494 #endif

1496 int ssl_parse_serverhello_tlsext(SSL *s, unsigned char **p, unsigned char *d, in
1497         {
1498         unsigned short length;
1499         unsigned short type;
1500         unsigned short size;
1501         unsigned char *data = *p;
1502         int tlsext_servername = 0;
1503         int renegotiate_seen = 0;

1505 #ifndef OPENSSL_NO_NEXTPROTONEG
1506         s->s3->next_proto_neg_seen = 0;
1507 #endif

1509 #ifndef OPENSSL_NO_HEARTBEATS
1510         s->tlsext_heartbeat &= ~(SSL_TLSEXT_HB_ENABLED |
1511                                  SSL_TLSEXT_HB_DONT_SEND_REQUESTS);
1512 #endif
```



```
1514         if (data >= (d+n-2))
1515                 goto ri_check;

1517         n2s(data,length);
1518         if (data+length != d+n)
1519                 {
1520                 *al = SSL_AD_DECODE_ERROR;
1521                 return 0;
1522                 }

1524         while(data <= (d+n-4))
1525                 {
1526                 n2s(data,type);
1527                 n2s(data,size);

1529                 if (data+size > (d+n))
1530                         goto ri_check;

1532                 if (s->tlsext_debug_cb)
1533                         s->tlsext_debug_cb(s, 1, type, data, size,
1534                                                 s->tlsext_debug_arg);

1536                 if (type == TLSEXT_TYPE_server_name)
1537                         {
1538                         if (s->tlsext_hostname == NULL || size > 0)
1539                                 {
1540                                 *al = TLS1_AD_UNRECOGNIZED_NAME;
1541                                 return 0;
1542                                 }
1543                         tlsext_servername = 1;
1544                         }

1546 #ifndef OPENSSL_NO_EC
1547                 else if (type == TLSEXT_TYPE_ec_point_formats)
1548                         {
1549                         unsigned char *sdata = data;
1550                         int ecpointformatlist_length = *(sdata++);

1552                         if (ecpointformatlist_length != size - 1 ||
1553                                 ecpointformatlist_length < 1)
1554                                 {
1555                                 *al = TLS1_AD_DECODE_ERROR;
1556                                 return 0;
1557                                 }
1558                         if (!s->hit)
1559                                 {
1560                         s->session->tlsext_ecpointformatlist_length = 0;
1561                         if (s->session->tlsext_ecpointformatlist != NULL) OPENSS
1562                         if ((s->session->tlsext_ecpointformatlist = OPENSSL_mall
1563                                 {
1564                                 *al = TLS1_AD_INTERNAL_ERROR;
1565                                 return 0;
1566                                 }
1567                         s->session->tlsext_ecpointformatlist_length = ecpointfor
1568                         memcpy(s->session->tlsext_ecpointformatlist, sdata, ecpo
1569                                 }
1570 #if 0
1571                         fprintf(stderr,"ssl_parse_serverhello_tlsext s->session-
1572                         sdata = s->session->tlsext_ecpointformatlist;
1573                         for (i = 0; i < s->session->tlsext_ecpointformatlist_len
1574                                 fprintf(stderr,"%i ",*(sdata++));
1575                         fprintf(stderr,"\n");
1576 #endif
1577                         }
1578 #endif /* OPENSSL_NO_EC */
```

```
1580                 else if (type == TLSEXT_TYPE_session_ticket)
1581                         {
1582                         if (s->tls_session_ticket_ext_cb &&
1583                             !s->tls_session_ticket_ext_cb(s, data, size, s->tls_
1584                                 {
1585                                 *al = TLS1_AD_INTERNAL_ERROR;
1586                                 return 0;
1587                                 }
1588                         if ((SSL_get_options(s) & SSL_OP_NO_TICKET)
1589                                 || (size > 0))
1590                                 {
1591                                 *al = TLS1_AD_UNSUPPORTED_EXTENSION;
1592                                 return 0;
1593                                 }
1594                         s->tlsext_ticket_expected = 1;
1595                         }
1596 #ifdef TLSEXT_TYPE_opaque_prf_input
1597                 else if (type == TLSEXT_TYPE_opaque_prf_input &&
1598                      s->version != DTLS1_VERSION)
1599                         {
1600                         unsigned char *sdata = data;

1602                         if (size < 2)
1603                                 {
1604                                 *al = SSL_AD_DECODE_ERROR;
1605                                 return 0;
1606                                 }
1607                         n2s(sdata, s->s3->server_opaque_prf_input_len);
1608                         if (s->s3->server_opaque_prf_input_len != size - 2)
1609                                 {
1610                                 *al = SSL_AD_DECODE_ERROR;
1611                                 return 0;
1612                                 }

1614                         if (s->s3->server_opaque_prf_input != NULL) /* shouldn't
1615                                 OPENSSL_free(s->s3->server_opaque_prf_input);
1616                         if (s->s3->server_opaque_prf_input_len == 0)
1617                                 s->s3->server_opaque_prf_input = OPENSSL_malloc(
1618                         else
1619                                 s->s3->server_opaque_prf_input = BUF_memdup(sdat

1621                         if (s->s3->server_opaque_prf_input == NULL)
1622                                 {
1623                                 *al = TLS1_AD_INTERNAL_ERROR;
1624                                 return 0;
1625                                 }
1626                         }
1627 #endif
1628                 else if (type == TLSEXT_TYPE_status_request &&
1629                          s->version != DTLS1_VERSION)
1630                         {
1631                         /* MUST be empty and only sent if we've requested
1632                          * a status request message.
1633                          */
1634                         if ((s->tlsext_status_type == -1) || (size > 0))
1635                                 {
1636                                 *al = TLS1_AD_UNSUPPORTED_EXTENSION;
1637                                 return 0;
1638                                 }
1639                         /* Set flag to expect CertificateStatus message */
1640                         s->tlsext_status_expected = 1;
1641                         }
1642 #ifndef OPENSSL_NO_NEXTPROTONEG
1643                 else if (type == TLSEXT_TYPE_next_proto_neg &&
1644                          s->s3->tmp.finish_md_len == 0)
1645                         {
```



```
1646                         unsigned char *selected;
1647                         unsigned char selected_len;

1649                         /* We must have requested it. */
1650                         if (s->ctx->next_proto_select_cb == NULL)
1651                                 {
1652                                 *al = TLS1_AD_UNSUPPORTED_EXTENSION;
1653                                 return 0;
1654                                 }
1655                         /* The data must be valid */
1656                         if (!ssl_next_proto_validate(data, size))
1657                                 {
1658                                 *al = TLS1_AD_DECODE_ERROR;
1659                                 return 0;
1660                                 }
1661                         if (s->ctx->next_proto_select_cb(s, &selected, &selected
1662                                 {
1663                                 *al = TLS1_AD_INTERNAL_ERROR;
1664                                 return 0;
1665                                 }
1666                         s->next_proto_negotiated = OPENSSL_malloc(selected_len);
1667                         if (!s->next_proto_negotiated)
1668                                 {
1669                                 *al = TLS1_AD_INTERNAL_ERROR;
1670                                 return 0;
1671                                 }
1672                         memcpy(s->next_proto_negotiated, selected, selected_len)
1673                         s->next_proto_negotiated_len = selected_len;
1674                         s->s3->next_proto_neg_seen = 1;
1675                         }
1676 #endif
1677                 else if (type == TLSEXT_TYPE_renegotiate)
1678                         {
1679                         if(!ssl_parse_serverhello_renegotiate_ext(s, data, size,
1680                                 return 0;
1681                         renegotiate_seen = 1;
1682                         }
1683 #ifndef OPENSSL_NO_HEARTBEATS
1684                 else if (type == TLSEXT_TYPE_heartbeat)
1685                         {
1686                         switch(data[0])
1687                                 {
1688                                 case 0x01:      /* Server allows us to send HB r
1689                                         s->tlsext_heartbeat |= S
1690                                         break;
1691                                 case 0x02:      /* Server doesn't accept HB requ
1692                                         s->tlsext_heartbeat |= S
1693                                         s->tlsext_heartbeat |= S
1694                                         break;
1695                                 default:        *al = SSL_AD_ILLEGAL_PARAMETER;
1696                                         return 0;
1697                                 }
1698                         }
1699 #endif
1700 #ifndef OPENSSL_NO_SRTP
1701                 else if (type == TLSEXT_TYPE_use_srtp)
1702                         {
1703                         if(ssl_parse_serverhello_use_srtp_ext(s, data, size,
1704                                                               al))
1705                                 return 0;
1706                         }
1707 #endif

1709                 data+=size;
1710                 }
```

```
1712         if (data != d+n)
1713                 {
1714                 *al = SSL_AD_DECODE_ERROR;
1715                 return 0;
1716                 }

1718         if (!s->hit && tlsext_servername == 1)
1719                 {
1720                 if (s->tlsext_hostname)
1721                         {
1722                         if (s->session->tlsext_hostname == NULL)
1723                                 {
1724                                 s->session->tlsext_hostname = BUF_strdup(s->tlse
1725                                 if (!s->session->tlsext_hostname)
1726                                         {
1727                                         *al = SSL_AD_UNRECOGNIZED_NAME;
1728                                         return 0;
1729                                         }
1730                                 }
1731                         else
1732                                 {
1733                                 *al = SSL_AD_DECODE_ERROR;
1734                                 return 0;
1735                                 }
1736                         }
1737                 }

1739         *p = data;

1741         ri_check:

1743         /* Determine if we need to see RI. Strictly speaking if we want to
1744          * avoid an attack we should *always* see RI even on initial server
1745          * hello because the client doesn't see any renegotiation during an
1746          * attack. However this would mean we could not connect to any server
1747          * which doesn't support RI so for the immediate future tolerate RI
1748          * absence on initial connect only.
1749          */
1750         if (!renegotiate_seen
1751                 && !(s->options & SSL_OP_LEGACY_SERVER_CONNECT)
1752                 && !(s->options & SSL_OP_ALLOW_UNSAFE_LEGACY_RENEGOTIATION))
1753                 {
1754                 *al = SSL_AD_HANDSHAKE_FAILURE;
1755                 SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_TLSEXT,
1756                                 SSL_R_UNSAFE_LEGACY_RENEGOTIATION_DISABLED);
1757                 return 0;
1758                 }

1760         return 1;
1761         }


1764 int ssl_prepare_clienthello_tlsext(SSL *s)
1765         {
1766 #ifndef OPENSSL_NO_EC
1767         /* If we are client and using an elliptic curve cryptography cipher suit
1768          * and elliptic curves we support.
1769          */
1770         int using_ecc = 0;
1771         int i;
1772         unsigned char *j;
1773         unsigned long alg_k, alg_a;
1774         STACK_OF(SSL_CIPHER) *cipher_stack = SSL_get_ciphers(s);

1776         for (i = 0; i < sk_SSL_CIPHER_num(cipher_stack); i++)
1777                 {
```



```
1778                 SSL_CIPHER *c = sk_SSL_CIPHER_value(cipher_stack, i);

1780                 alg_k = c->algorithm_mkey;
1781                 alg_a = c->algorithm_auth;
1782                 if ((alg_k & (SSL_kEECDH|SSL_kECDHr|SSL_kECDHe) || (alg_a & SSL_
1783                         {
1784                         using_ecc = 1;
1785                         break;
1786                         }
1787                 }
1788         using_ecc = using_ecc && (s->version >= TLS1_VERSION);
1789         if (using_ecc)
1790                 {
1791                 if (s->tlsext_ecpointformatlist != NULL) OPENSSL_free(s->tlsext_
1792                 if ((s->tlsext_ecpointformatlist = OPENSSL_malloc(3)) == NULL)
1793                         {
1794                         SSLerr(SSL_F_SSL_PREPARE_CLIENTHELLO_TLSEXT,ERR_R_MALLOC
1795                         return -1;
1796                         }
1797                 s->tlsext_ecpointformatlist_length = 3;
1798                 s->tlsext_ecpointformatlist[0] = TLSEXT_ECPOINTFORMAT_uncompress
1799                 s->tlsext_ecpointformatlist[1] = TLSEXT_ECPOINTFORMAT_ansiX962_c
1800                 s->tlsext_ecpointformatlist[2] = TLSEXT_ECPOINTFORMAT_ansiX962_c

1802                 /* we support all named elliptic curves in draft-ietf-tls-ecc-12
1803                 if (s->tlsext_ellipticcurvelist != NULL) OPENSSL_free(s->tlsext_
1804                 s->tlsext_ellipticcurvelist_length = sizeof(pref_list)/sizeof(pr
1805                 if ((s->tlsext_ellipticcurvelist = OPENSSL_malloc(s->tlsext_elli
1806                         {
1807                         s->tlsext_ellipticcurvelist_length = 0;
1808                         SSLerr(SSL_F_SSL_PREPARE_CLIENTHELLO_TLSEXT,ERR_R_MALLOC
1809                         return -1;
1810                         }
1811                 for (i = 0, j = s->tlsext_ellipticcurvelist; (unsigned int)i <
1812                                 sizeof(pref_list)/sizeof(pref_list[0]); i++)
1813                         {
1814                         int id = tls1_ec_nid2curve_id(pref_list[i]);
1815                         s2n(id,j);
1816                         }
1817                 }
1818 #endif /* OPENSSL_NO_EC */

1820 #ifdef TLSEXT_TYPE_opaque_prf_input
1821         {
1822                 int r = 1;

1824                 if (s->ctx->tlsext_opaque_prf_input_callback != 0)
1825                         {
1826                         r = s->ctx->tlsext_opaque_prf_input_callback(s, NULL, 0,
1827                         if (!r)
1828                                 return -1;
1829                         }

1831                 if (s->tlsext_opaque_prf_input != NULL)
1832                         {
1833                         if (s->s3->client_opaque_prf_input != NULL) /* shouldn't
1834                                 OPENSSL_free(s->s3->client_opaque_prf_input);

1836                         if (s->tlsext_opaque_prf_input_len == 0)
1837                                 s->s3->client_opaque_prf_input = OPENSSL_malloc(
1838                         else
1839                                 s->s3->client_opaque_prf_input = BUF_memdup(s->t
1840                         if (s->s3->client_opaque_prf_input == NULL)
1841                                 {
1842                                 SSLerr(SSL_F_SSL_PREPARE_CLIENTHELLO_TLSEXT,ERR_
1843                                 return -1;
```

```
1844                                }
1845                        s->s3->client_opaque_prf_input_len = s->tlsext_opaque_pr
1846                        }
1847
1848                if (r == 2)
1849                        /* at callback's request, insist on receiving an appropr
1850                        s->s3->server_opaque_prf_input_len = s->tlsext_opaque_pr
1851        }
1852 #endif
1853
1854        return 1;
1855        }
1856
1857 int ssl_prepare_serverhello_tlsext(SSL *s)
1858        {
1859 #ifndef OPENSSL_NO_EC
1860        /* If we are server and using an ECC cipher suite, send the point format
1861         * if the client sent us an ECPointsFormat extension.  Note that the ser
1862         * supposed to send an EllipticCurves extension.
1863         */
1864
1865        unsigned long alg_k = s->s3->tmp.new_cipher->algorithm_mkey;
1866        unsigned long alg_a = s->s3->tmp.new_cipher->algorithm_auth;
1867        int using_ecc = (alg_k & (SSL_kEECDH|SSL_kECDHr|SSL_kECDHe)) || (alg_a &
1868        using_ecc = using_ecc && (s->session->tlsext_ecpointformatlist != NULL);
1869
1870        if (using_ecc)
1871                {
1872                if (s->tlsext_ecpointformatlist != NULL) OPENSSL_free(s->tlsext_
1873                if ((s->tlsext_ecpointformatlist = OPENSSL_malloc(3)) == NULL)
1874                        {
1875                        SSLerr(SSL_F_SSL_PREPARE_SERVERHELLO_TLSEXT,ERR_R_MALLOC
1876                        return -1;
1877                        }
1878                s->tlsext_ecpointformatlist_length = 3;
1879                s->tlsext_ecpointformatlist[0] = TLSEXT_ECPOINTFORMAT_uncompress
1880                s->tlsext_ecpointformatlist[1] = TLSEXT_ECPOINTFORMAT_ansiX962_c
1881                s->tlsext_ecpointformatlist[2] = TLSEXT_ECPOINTFORMAT_ansiX962_c
1882                }
1883 #endif /* OPENSSL_NO_EC */
1884
1885        return 1;
1886        }
1887
1888 int ssl_check_clienthello_tlsext_early(SSL *s)
1889        {
1890        int ret=SSL_TLSEXT_ERR_NOACK;
1891        int al = SSL_AD_UNRECOGNIZED_NAME;
1892
1893 #ifndef OPENSSL_NO_EC
1894        /* The handling of the ECPointFormats extension is done elsewhere, namel
1895         * ssl3_choose_cipher in s3_lib.c.
1896         */
1897        /* The handling of the EllipticCurves extension is done elsewhere, namel
1898         * ssl3_choose_cipher in s3_lib.c.
1899         */
1900 #endif
1901
1902        if (s->ctx != NULL && s->ctx->tlsext_servername_callback != 0)
1903                ret = s->ctx->tlsext_servername_callback(s, &al, s->ctx->tlsext_
1904        else if (s->initial_ctx != NULL && s->initial_ctx->tlsext_servername_cal
1905                ret = s->initial_ctx->tlsext_servername_callback(s, &al, s->init
1906
1907 #ifdef TLSEXT_TYPE_opaque_prf_input
1908        {
1909                /* This sort of belongs into ssl_prepare_serverhello_tlsext(),
```



```
1910                 * but we might be sending an alert in response to the client he
1911                 * so this has to happen here in
1912                 * ssl_check_clienthello_tlsext_early(). */
1913
1914                int r = 1;
1915
1916                if (s->ctx->tlsext_opaque_prf_input_callback != 0)
1917                        {
1918                        r = s->ctx->tlsext_opaque_prf_input_callback(s, NULL, 0,
1919                        if (!r)
1920                                {
1921                                ret = SSL_TLSEXT_ERR_ALERT_FATAL;
1922                                al = SSL_AD_INTERNAL_ERROR;
1923                                goto err;
1924                                }
1925                        }
1926
1927                if (s->s3->server_opaque_prf_input != NULL) /* shouldn't really
1928                        OPENSSL_free(s->s3->server_opaque_prf_input);
1929                s->s3->server_opaque_prf_input = NULL;
1930
1931                if (s->tlsext_opaque_prf_input != NULL)
1932                        {
1933                        if (s->s3->client_opaque_prf_input != NULL &&
1934                                s->s3->client_opaque_prf_input_len == s->tlsext_
1935                                {
1936                                /* can only use this extension if we have a serv
1937                                 * of the same length as the client opaque PRF i
1938
1939                                if (s->tlsext_opaque_prf_input_len == 0)
1940                                        s->s3->server_opaque_prf_input = OPENSSL
1941                                else
1942                                        s->s3->server_opaque_prf_input = BUF_mem
1943                                if (s->s3->server_opaque_prf_input == NULL)
1944                                        {
1945                                        ret = SSL_TLSEXT_ERR_ALERT_FATAL;
1946                                        al = SSL_AD_INTERNAL_ERROR;
1947                                        goto err;
1948                                        }
1949                                s->s3->server_opaque_prf_input_len = s->tlsext_o
1950                                }
1951                        }
1952
1953                if (r == 2 && s->s3->server_opaque_prf_input == NULL)
1954                        {
1955                        /* The callback wants to enforce use of the extension,
1956                         * but we can't do that with the client opaque PRF input
1957                         * abort the handshake.
1958                         */
1959                        ret = SSL_TLSEXT_ERR_ALERT_FATAL;
1960                        al = SSL_AD_HANDSHAKE_FAILURE;
1961                        }
1962        }
1963
1964  err:
1965 #endif
1966        switch (ret)
1967                {
1968                case SSL_TLSEXT_ERR_ALERT_FATAL:
1969                        ssl3_send_alert(s,SSL3_AL_FATAL,al);
1970                        return -1;
1971
1972                case SSL_TLSEXT_ERR_ALERT_WARNING:
1973                        ssl3_send_alert(s,SSL3_AL_WARNING,al);
1974                        return 1;
```

```
1976                 case SSL_TLSEXT_ERR_NOACK:
1977                         s->servername_done=0;
1978                         default:
1979                 return 1;
1980                 }
1981         }

1983 int ssl_check_clienthello_tlsext_late(SSL *s)
1984         {
1985         int ret = SSL_TLSEXT_ERR_OK;
1986         int al;

1988         /* If status request then ask callback what to do.
1989          * Note: this must be called after servername callbacks in case
1990          * the certificate has changed, and must be called after the cipher
1991          * has been chosen because this may influence which certificate is sent
1992          */
1993         if ((s->tlsext_status_type != -1) && s->ctx && s->ctx->tlsext_status_cb)
1994                 {
1995                 int r;
1996                 CERT_PKEY *certpkey;
1997                 certpkey = ssl_get_server_send_pkey(s);
1998                 /* If no certificate can't return certificate status */
1999                 if (certpkey == NULL)
2000                         {
2001                         s->tlsext_status_expected = 0;
2002                         return 1;
2003                         }
2004                 /* Set current certificate to one we will use so
2005                  * SSL_get_certificate et al can pick it up.
2006                  */
2007                 s->cert->key = certpkey;
2008                 r = s->ctx->tlsext_status_cb(s, s->ctx->tlsext_status_arg);
2009                 switch (r)
2010                         {
2011                         /* We don't want to send a status request response */
2012                         case SSL_TLSEXT_ERR_NOACK:
2013                                 s->tlsext_status_expected = 0;
2014                                 break;
2015                         /* status request response should be sent */
2016                         case SSL_TLSEXT_ERR_OK:
2017                                 if (s->tlsext_ocsp_resp)
2018                                         s->tlsext_status_expected = 1;
2019                                 else
2020                                         s->tlsext_status_expected = 0;
2021                                 break;
2022                         /* something bad happened */
2023                         case SSL_TLSEXT_ERR_ALERT_FATAL:
2024                                 ret = SSL_TLSEXT_ERR_ALERT_FATAL;
2025                                 al = SSL_AD_INTERNAL_ERROR;
2026                                 goto err;
2027                         }
2028                 }
2029         else
2030                 s->tlsext_status_expected = 0;

2032  err:
2033         switch (ret)
2034                 {
2035                 case SSL_TLSEXT_ERR_ALERT_FATAL:
2036                         ssl3_send_alert(s,SSL3_AL_FATAL,al);
2037                         return -1;

2039                 case SSL_TLSEXT_ERR_ALERT_WARNING:
2040                         ssl3_send_alert(s,SSL3_AL_WARNING,al);
2041                         return 1;
```



```
2043                 default:
2044                         return 1;
2045                 }
2046         }

2048 int ssl_check_serverhello_tlsext(SSL *s)
2049         {
2050         int ret=SSL_TLSEXT_ERR_NOACK;
2051         int al = SSL_AD_UNRECOGNIZED_NAME;

2053 #ifndef OPENSSL_NO_EC
2054         /* If we are client and using an elliptic curve cryptography cipher
2055          * suite, then if server returns an EC point formats lists extension
2056          * it must contain uncompressed.
2057          */
2058         unsigned long alg_k = s->s3->tmp.new_cipher->algorithm_mkey;
2059         unsigned long alg_a = s->s3->tmp.new_cipher->algorithm_auth;
2060         if ((s->tlsext_ecpointformatlist != NULL) && (s->tlsext_ecpointformatlis
2061             (s->session->tlsext_ecpointformatlist != NULL) && (s->session->tlsex
2062             ((alg_k & (SSL_kEECDH|SSL_kECDHr|SSL_kECDHe)) || (alg_a & SSL_aECDSA
2063                 {
2064                 /* we are using an ECC cipher */
2065                 size_t i;
2066                 unsigned char *list;
2067                 int found_uncompressed = 0;
2068                 list = s->session->tlsext_ecpointformatlist;
2069                 for (i = 0; i < s->session->tlsext_ecpointformatlist_length; i++
2070                         {
2071                         if (*(list++) == TLSEXT_ECPOINTFORMAT_uncompressed)
2072                                 {
2073                                 found_uncompressed = 1;
2074                                 break;
2075                                 }
2076                         }
2077                 if (!found_uncompressed)
2078                         {
2079                         SSLerr(SSL_F_SSL_CHECK_SERVERHELLO_TLSEXT,SSL_R_TLS_INVA
2080                         return -1;
2081                         }
2082                 }
2083         ret = SSL_TLSEXT_ERR_OK;
2084 #endif /* OPENSSL_NO_EC */

2086         if (s->ctx != NULL && s->ctx->tlsext_servername_callback != 0)
2087                 ret = s->ctx->tlsext_servername_callback(s, &al, s->ctx->tlsext_
2088         else if (s->initial_ctx != NULL && s->initial_ctx->tlsext_servername_cal
2089                 ret = s->initial_ctx->tlsext_servername_callback(s, &al, s->init

2091 #ifdef TLSEXT_TYPE_opaque_prf_input
2092         if (s->s3->server_opaque_prf_input_len > 0)
2093                 {
2094                 /* This case may indicate that we, as a client, want to insist o
2095                  * So first verify that we really have a value from the server t

2097                 if (s->s3->server_opaque_prf_input == NULL)
2098                         {
2099                         ret = SSL_TLSEXT_ERR_ALERT_FATAL;
2100                         al = SSL_AD_HANDSHAKE_FAILURE;
2101                         }

2103                 /* Anytime the server *has* sent an opaque PRF input, we need to
2104                  * that we have a client opaque PRF input of the same size. */
2105                 if (s->s3->client_opaque_prf_input == NULL ||
2106                     s->s3->client_opaque_prf_input_len != s->s3->server_opaque_p
2107                         {
```

```
2108                        ret = SSL_TLSEXT_ERR_ALERT_FATAL;
2109                        al = SSL_AD_ILLEGAL_PARAMETER;
2110                        }
2111                }
2112 #endif

2114        /* If we've requested certificate status and we wont get one
2115         * tell the callback
2116         */
2117        if ((s->tlsext_status_type != -1) && !(s->tlsext_status_expected)
2118                        && s->ctx && s->ctx->tlsext_status_cb)
2119                {
2120                int r;
2121                /* Set resp to NULL, resplen to -1 so callback knows
2122                 * there is no response.
2123                 */
2124                if (s->tlsext_ocsp_resp)
2125                        {
2126                        OPENSSL_free(s->tlsext_ocsp_resp);
2127                        s->tlsext_ocsp_resp = NULL;
2128                        }
2129                s->tlsext_ocsp_resplen = -1;
2130                r = s->ctx->tlsext_status_cb(s, s->ctx->tlsext_status_arg);
2131                if (r == 0)
2132                        {
2133                        al = SSL_AD_BAD_CERTIFICATE_STATUS_RESPONSE;
2134                        ret = SSL_TLSEXT_ERR_ALERT_FATAL;
2135                        }
2136                if (r < 0)
2137                        {
2138                        al = SSL_AD_INTERNAL_ERROR;
2139                        ret = SSL_TLSEXT_ERR_ALERT_FATAL;
2140                        }
2141                }

2143        switch (ret)
2144                {
2145                case SSL_TLSEXT_ERR_ALERT_FATAL:
2146                        ssl3_send_alert(s,SSL3_AL_FATAL,al);
2147                        return -1;

2149                case SSL_TLSEXT_ERR_ALERT_WARNING:
2150                        ssl3_send_alert(s,SSL3_AL_WARNING,al);
2151                        return 1;

2153                case SSL_TLSEXT_ERR_NOACK:
2154                        s->servername_done=0;
2155                        default:
2156                return 1;
2157                }
2158        }

2160 /* Since the server cache lookup is done early on in the processing of the
2161  * ClientHello, and other operations depend on the result, we need to handle
2162  * any TLS session ticket extension at the same time.
2163  *
2164  *   session_id: points at the session ID in the ClientHello. This code will
2165  *       read past the end of this in order to parse out the session ticket
2166  *       extension, if any.
2167  *   len: the length of the session ID.
2168  *   limit: a pointer to the first byte after the ClientHello.
2169  *   ret: (output) on return, if a ticket was decrypted, then this is set to
2170  *       point to the resulting session.
2171  *
2172  * If s->tls_session_secret_cb is set then we are expecting a pre-shared key
2173  * ciphersuite, in which case we have no use for session tickets and one will
```



```
2174  * never be decrypted, nor will s->tlsext_ticket_expected be set to 1.
2175  *
2176  * Returns:
2177  *   -1: fatal error, either from parsing or decrypting the ticket.
2178  *    0: no ticket was found (or was ignored, based on settings).
2179  *    1: a zero length extension was found, indicating that the client supports
2180  *       session tickets but doesn't currently have one to offer.
2181  *    2: either s->tls_session_secret_cb was set, or a ticket was offered but
2182  *       couldn't be decrypted because of a non-fatal error.
2183  *    3: a ticket was successfully decrypted and *ret was set.
2184  *
2185  * Side effects:
2186  *   Sets s->tlsext_ticket_expected to 1 if the server will have to issue
2187  *   a new session ticket to the client because the client indicated support
2188  *   (and s->tls_session_secret_cb is NULL) but the client either doesn't have
2189  *   a session ticket or we couldn't use the one it gave us, or if
2190  *   s->ctx->tlsext_ticket_key_cb asked to renew the client's ticket.
2191  *   Otherwise, s->tlsext_ticket_expected is set to 0.
2192  */
2193 int tls1_process_ticket(SSL *s, unsigned char *session_id, int len,
2194                         const unsigned char *limit, SSL_SESSION **ret)
2195        {
2196        /* Point after session ID in client hello */
2197        const unsigned char *p = session_id + len;
2198        unsigned short i;

2200        *ret = NULL;
2201        s->tlsext_ticket_expected = 0;

2203        /* If tickets disabled behave as if no ticket present
2204         * to permit stateful resumption.
2205         */
2206        if (SSL_get_options(s) & SSL_OP_NO_TICKET)
2207                return 0;
2208        if ((s->version <= SSL3_VERSION) || !limit)
2209                return 0;
2210        if (p >= limit)
2211                return -1;
2212        /* Skip past DTLS cookie */
2213        if (s->version == DTLS1_VERSION || s->version == DTLS1_BAD_VER)
2214                {
2215                i = *(p++);
2216                p+= i;
2217                if (p >= limit)
2218                        return -1;
2219                }
2220        /* Skip past cipher list */
2221        n2s(p, i);
2222        p+= i;
2223        if (p >= limit)
2224                return -1;
2225        /* Skip past compression algorithm list */
2226        i = *(p++);
2227        p += i;
2228        if (p > limit)
2229                return -1;
2230        /* Now at start of extensions */
2231        if ((p + 2) >= limit)
2232                return 0;
2233        n2s(p, i);
2234        while ((p + 4) <= limit)
2235                {
2236                unsigned short type, size;
2237                n2s(p, type);
2238                n2s(p, size);
2239                if (p + size > limit)
```

```
2240                        return 0;
2241                if (type == TLSEXT_TYPE_session_ticket)
2242                        {
2243                        int r;
2244                        if (size == 0)
2245                                {
2246                                /* The client will accept a ticket but doesn't
2247                                 * currently have one. */
2248                                s->tlsext_ticket_expected = 1;
2249                                return 1;
2250                                }
2251                        if (s->tls_session_secret_cb)
2252                                {
2253                                /* Indicate that the ticket couldn't be
2254                                 * decrypted rather than generating the session
2255                                 * from ticket now, trigger abbreviated
2256                                 * handshake based on external mechanism to
2257                                 * calculate the master secret later. */
2258                                return 2;
2259                                }
2260                        r = tls_decrypt_ticket(s, p, size, session_id, len, ret)
2261                        switch (r)
2262                                {
2263                                case 2: /* ticket couldn't be decrypted */
2264                                        s->tlsext_ticket_expected = 1;
2265                                        return 2;
2266                                case 3: /* ticket was decrypted */
2267                                        return r;
2268                                case 4: /* ticket decrypted but need to renew */
2269                                        s->tlsext_ticket_expected = 1;
2270                                        return 3;
2271                                default: /* fatal error */
2272                                        return -1;
2273                                }
2274                        }
2275                p += size;
2276                }
2277        return 0;
2278        }

2280 /* tls_decrypt_ticket attempts to decrypt a session ticket.
2281  *
2282  *   etick: points to the body of the session ticket extension.
2283  *   eticklen: the length of the session tickets extenion.
2284  *   sess_id: points at the session ID.
2285  *   sesslen: the length of the session ID.
2286  *   psess: (output) on return, if a ticket was decrypted, then this is set to
2287  *       point to the resulting session.
2288  *
2289  * Returns:
2290  *   -1: fatal error, either from parsing or decrypting the ticket.
2291  *    2: the ticket couldn't be decrypted.
2292  *    3: a ticket was successfully decrypted and *psess was set.
2293  *    4: same as 3, but the ticket needs to be renewed.
2294  */
2295 static int tls_decrypt_ticket(SSL *s, const unsigned char *etick, int eticklen,
2296                               const unsigned char *sess_id, int sesslen,
2297                               SSL_SESSION **psess)
2298        {
2299        SSL_SESSION *sess;
2300        unsigned char *sdec;
2301        const unsigned char *p;
2302        int slen, mlen, renew_ticket = 0;
2303        unsigned char tick_hmac[EVP_MAX_MD_SIZE];
2304        HMAC_CTX hctx;
2305        EVP_CIPHER_CTX ctx;
```



```
2306        SSL_CTX *tctx = s->initial_ctx;
2307        /* Need at least keyname + iv + some encrypted data */
2308        if (eticklen < 48)
2309                return 2;
2310        /* Initialize session ticket encryption and HMAC contexts */
2311        HMAC_CTX_init(&hctx);
2312        EVP_CIPHER_CTX_init(&ctx);
2313        if (tctx->tlsext_ticket_key_cb)
2314                {
2315                unsigned char *nctick = (unsigned char *)etick;
2316                int rv = tctx->tlsext_ticket_key_cb(s, nctick, nctick + 16,
2317                                                    &ctx, &hctx, 0);
2318                if (rv < 0)
2319                        return -1;
2320                if (rv == 0)
2321                        return 2;
2322                if (rv == 2)
2323                        renew_ticket = 1;
2324                }
2325        else
2326                {
2327                /* Check key name matches */
2328                if (memcmp(etick, tctx->tlsext_tick_key_name, 16))
2329                        return 2;
2330                HMAC_Init_ex(&hctx, tctx->tlsext_tick_hmac_key, 16,
2331                                        tlsext_tick_md(), NULL);
2332                EVP_DecryptInit_ex(&ctx, EVP_aes_128_cbc(), NULL,
2333                                   tctx->tlsext_tick_aes_key, etick + 16);
2334                }
2335        /* Attempt to process session ticket, first conduct sanity and
2336         * integrity checks on ticket.
2337         */
2338        mlen = HMAC_size(&hctx);
2339        if (mlen < 0)
2340                {
2341                EVP_CIPHER_CTX_cleanup(&ctx);
2342                return -1;
2343                }
2344        eticklen -= mlen;
2345        /* Check HMAC of encrypted ticket */
2346        HMAC_Update(&hctx, etick, eticklen);
2347        HMAC_Final(&hctx, tick_hmac, NULL);
2348        HMAC_CTX_cleanup(&hctx);
2349        if (CRYPTO_memcmp(tick_hmac, etick + eticklen, mlen))
2350                return 2;
2351        /* Attempt to decrypt session data */
2352        /* Move p after IV to start of encrypted ticket, update length */
2353        p = etick + 16 + EVP_CIPHER_CTX_iv_length(&ctx);
2354        eticklen -= 16 + EVP_CIPHER_CTX_iv_length(&ctx);
2355        sdec = OPENSSL_malloc(eticklen);
2356        if (!sdec)
2357                {
2358                EVP_CIPHER_CTX_cleanup(&ctx);
2359                return -1;
2360                }
2361        EVP_DecryptUpdate(&ctx, sdec, &slen, p, eticklen);
2362        if (EVP_DecryptFinal(&ctx, sdec + slen, &mlen) <= 0)
2363                {
2364                EVP_CIPHER_CTX_cleanup(&ctx);
2365                OPENSSL_free(sdec);
2366                return 2;
2367                }
2368        slen += mlen;
2369        EVP_CIPHER_CTX_cleanup(&ctx);
2370        p = sdec;
```

```
2372         sess = d2i_SSL_SESSION(NULL, &p, slen);
2373         OPENSSL_free(sdec);
2374         if (sess)
2375                 {
2376                 /* The session ID, if non-empty, is used by some clients to
2377                  * detect that the ticket has been accepted. So we copy it to
2378                  * the session structure. If it is empty set length to zero
2379                  * as required by standard.
2380                  */
2381                 if (sesslen)
2382                         memcpy(sess->session_id, sess_id, sesslen);
2383                 sess->session_id_length = sesslen;
2384                 *psess = sess;
2385                 if (renew_ticket)
2386                         return 4;
2387                 else
2388                         return 3;
2389                 }
2390         ERR_clear_error();
2391         /* For session parse failure, indicate that we need to send a new
2392          * ticket. */
2393         return 2;
2394         }

2396 /* Tables to translate from NIDs to TLS v1.2 ids */

2398 typedef struct
2399         {
2400         int nid;
2401         int id;
2402         } tls12_lookup;

2404 static tls12_lookup tls12_md[] = {
2405 #ifndef OPENSSL_NO_MD5
2406         {NID_md5, TLSEXT_hash_md5},
2407 #endif
2408 #ifndef OPENSSL_NO_SHA
2409         {NID_sha1, TLSEXT_hash_sha1},
2410 #endif
2411 #ifndef OPENSSL_NO_SHA256
2412         {NID_sha224, TLSEXT_hash_sha224},
2413         {NID_sha256, TLSEXT_hash_sha256},
2414 #endif
2415 #ifndef OPENSSL_NO_SHA512
2416         {NID_sha384, TLSEXT_hash_sha384},
2417         {NID_sha512, TLSEXT_hash_sha512}
2418 #endif
2419 };

2421 static tls12_lookup tls12_sig[] = {
2422 #ifndef OPENSSL_NO_RSA
2423         {EVP_PKEY_RSA, TLSEXT_signature_rsa},
2424 #endif
2425 #ifndef OPENSSL_NO_DSA
2426         {EVP_PKEY_DSA, TLSEXT_signature_dsa},
2427 #endif
2428 #ifndef OPENSSL_NO_ECDSA
2429         {EVP_PKEY_EC, TLSEXT_signature_ecdsa}
2430 #endif
2431 };

2433 static int tls12_find_id(int nid, tls12_lookup *table, size_t tlen)
2434         {
2435         size_t i;
2436         for (i = 0; i < tlen; i++)
2437                 {
```



```
2438                 if (table[i].nid == nid)
2439                         return table[i].id;
2440                 }
2441         return -1;
2442         }
2443 #if 0
2444 static int tls12_find_nid(int id, tls12_lookup *table, size_t tlen)
2445         {
2446         size_t i;
2447         for (i = 0; i < tlen; i++)
2448                 {
2449                 if (table[i].id == id)
2450                         return table[i].nid;
2451                 }
2452         return -1;
2453         }
2454 #endif

2456 int tls12_get_sigandhash(unsigned char *p, const EVP_PKEY *pk, const EVP_MD *md)
2457         {
2458         int sig_id, md_id;
2459         if (!md)
2460                 return 0;
2461         md_id = tls12_find_id(EVP_MD_type(md), tls12_md,
2462                                 sizeof(tls12_md)/sizeof(tls12_lookup));
2463         if (md_id == -1)
2464                 return 0;
2465         sig_id = tls12_get_sigid(pk);
2466         if (sig_id == -1)
2467                 return 0;
2468         p[0] = (unsigned char)md_id;
2469         p[1] = (unsigned char)sig_id;
2470         return 1;
2471         }

2473 int tls12_get_sigid(const EVP_PKEY *pk)
2474         {
2475         return tls12_find_id(pk->type, tls12_sig,
2476                                 sizeof(tls12_sig)/sizeof(tls12_lookup));
2477         }

2479 const EVP_MD *tls12_get_hash(unsigned char hash_alg)
2480         {
2481         switch(hash_alg)
2482                 {
2483 #ifndef OPENSSL_NO_SHA
2484                 case TLSEXT_hash_sha1:
2485                 return EVP_sha1();
2486 #endif
2487 #ifndef OPENSSL_NO_SHA256
2488                 case TLSEXT_hash_sha224:
2489                 return EVP_sha224();

2491                 case TLSEXT_hash_sha256:
2492                 return EVP_sha256();
2493 #endif
2494 #ifndef OPENSSL_NO_SHA512
2495                 case TLSEXT_hash_sha384:
2496                 return EVP_sha384();

2498                 case TLSEXT_hash_sha512:
2499                 return EVP_sha512();
2500 #endif
2501                 default:
2502                 return NULL;
```

```
2504                }
2505        }

2507 /* Set preferred digest for each key type */

2509 int tls1_process_sigalgs(SSL *s, const unsigned char *data, int dsize)
2510        {
2511        int i, idx;
2512        const EVP_MD *md;
2513        CERT *c = s->cert;
2514        /* Extension ignored for TLS versions below 1.2 */
2515        if (TLS1_get_version(s) < TLS1_2_VERSION)
2516                return 1;
2517        /* Should never happen */
2518        if (!c)
2519                return 0;

2521        c->pkeys[SSL_PKEY_DSA_SIGN].digest = NULL;
2522        c->pkeys[SSL_PKEY_RSA_SIGN].digest = NULL;
2523        c->pkeys[SSL_PKEY_RSA_ENC].digest = NULL;
2524        c->pkeys[SSL_PKEY_ECC].digest = NULL;

2526        for (i = 0; i < dsize; i += 2)
2527                {
2528                unsigned char hash_alg = data[i], sig_alg = data[i+1];

2530                switch(sig_alg)
2531                        {
2532 #ifndef OPENSSL_NO_RSA
2533                        case TLSEXT_signature_rsa:
2534                        idx = SSL_PKEY_RSA_SIGN;
2535                        break;
2536 #endif
2537 #ifndef OPENSSL_NO_DSA
2538                        case TLSEXT_signature_dsa:
2539                        idx = SSL_PKEY_DSA_SIGN;
2540                        break;
2541 #endif
2542 #ifndef OPENSSL_NO_ECDSA
2543                        case TLSEXT_signature_ecdsa:
2544                        idx = SSL_PKEY_ECC;
2545                        break;
2546 #endif
2547                        default:
2548                        continue;
2549                        }

2551                if (c->pkeys[idx].digest == NULL)
2552                        {
2553                        md = tls12_get_hash(hash_alg);
2554                        if (md)
2555                                {
2556                                c->pkeys[idx].digest = md;
2557                                if (idx == SSL_PKEY_RSA_SIGN)
2558                                        c->pkeys[SSL_PKEY_RSA_ENC].digest = md;
2559                                }
2560                        }

2562                }


2565        /* Set any remaining keys to default values. NOTE: if alg is not
2566         * supported it stays as NULL.
2567         */
2568 #ifndef OPENSSL_NO_DSA
2569        if (!c->pkeys[SSL_PKEY_DSA_SIGN].digest)
```



```
2570                c->pkeys[SSL_PKEY_DSA_SIGN].digest = EVP_sha1();
2571 #endif
2572 #ifndef OPENSSL_NO_RSA
2573        if (!c->pkeys[SSL_PKEY_RSA_SIGN].digest)
2574                {
2575                c->pkeys[SSL_PKEY_RSA_SIGN].digest = EVP_sha1();
2576                c->pkeys[SSL_PKEY_RSA_ENC].digest = EVP_sha1();
2577                }
2578 #endif
2579 #ifndef OPENSSL_NO_ECDSA
2580        if (!c->pkeys[SSL_PKEY_ECC].digest)
2581                c->pkeys[SSL_PKEY_ECC].digest = EVP_sha1();
2582 #endif
2583        return 1;
2584        }

2586 #endif

2588 #ifndef OPENSSL_NO_HEARTBEATS
2589 int
2590 tls1_process_heartbeat(SSL *s)
2591        {
2592        unsigned char *p = &s->s3->rrec.data[0], *pl;
2593        unsigned short hbtype;
2594        unsigned int payload;
2595        unsigned int padding = 16; /* Use minimum padding */

2597        if (s->msg_callback)
2598                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
2599                        &s->s3->rrec.data[0], s->s3->rrec.length,
2600                        s, s->msg_callback_arg);

2602        /* Read type and payload length first */
2603        if (1 + 2 + 16 > s->s3->rrec.length)
2604                return 0; /* silently discard */
2605        hbtype = *p++;
2606        n2s(p, payload);
2607        if (1 + 2 + payload + 16 > s->s3->rrec.length)
2608                return 0; /* silently discard per RFC 6520 sec. 4 */
2609        pl = p;

2611        if (hbtype == TLS1_HB_REQUEST)
2612                {
2613                unsigned char *buffer, *bp;
2614                int r;

2616                /* Allocate memory for the response, size is 1 bytes
2617                 * message type, plus 2 bytes payload length, plus
2618                 * payload, plus padding
2619                 */
2620                buffer = OPENSSL_malloc(1 + 2 + payload + padding);
2621                bp = buffer;

2623                /* Enter response type, length and copy payload */
2624                *bp++ = TLS1_HB_RESPONSE;
2625                s2n(payload, bp);
2626                memcpy(bp, pl, payload);
2627                bp += payload;
2628                /* Random padding */
2629                RAND_pseudo_bytes(bp, padding);

2631                r = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3 + payload +

2633                if (r >= 0 && s->msg_callback)
2634                        s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
2635                                buffer, 3 + payload + padding,
```

```
2636                                s, s->msg_callback_arg);

2638                OPENSSL_free(buffer);

2640                if (r < 0)
2641                        return r;
2642                }
2643        else if (hbtype == TLS1_HB_RESPONSE)
2644                {
2645                unsigned int seq;

2647                /* We only send sequence numbers (2 bytes unsigned int),
2648                 * and 16 random bytes, so we just try to read the
2649                 * sequence number */
2650                n2s(pl, seq);

2652                if (payload == 18 && seq == s->tlsext_hb_seq)
2653                        {
2654                        s->tlsext_hb_seq++;
2655                        s->tlsext_hb_pending = 0;
2656                        }
2657                }

2659        return 0;
2660        }

2662 int
2663 tls1_heartbeat(SSL *s)
2664        {
2665        unsigned char *buf, *p;
2666        int ret;
2667        unsigned int payload = 18; /* Sequence number + random bytes */
2668        unsigned int padding = 16; /* Use minimum padding */

2670        /* Only send if peer supports and accepts HB requests... */
2671        if (!(s->tlsext_heartbeat & SSL_TLSEXT_HB_ENABLED) ||
2672            s->tlsext_heartbeat & SSL_TLSEXT_HB_DONT_SEND_REQUESTS)
2673                {
2674                SSLerr(SSL_F_TLS1_HEARTBEAT,SSL_R_TLS_HEARTBEAT_PEER_DOESNT_ACCE
2675                return -1;
2676                }

2678        /* ...and there is none in flight yet... */
2679        if (s->tlsext_hb_pending)
2680                {
2681                SSLerr(SSL_F_TLS1_HEARTBEAT,SSL_R_TLS_HEARTBEAT_PENDING);
2682                return -1;
2683                }

2685        /* ...and no handshake in progress. */
2686        if (SSL_in_init(s) || s->in_handshake)
2687                {
2688                SSLerr(SSL_F_TLS1_HEARTBEAT,SSL_R_UNEXPECTED_MESSAGE);
2689                return -1;
2690                }

2692        /* Check if padding is too long, payload and padding
2693         * must not exceed 2^14 - 3 = 16381 bytes in total.
2694         */
2695        OPENSSL_assert(payload + padding <= 16381);

2697        /* Create HeartBeat message, we just use a sequence number
2698         * as payload to distuingish different messages and add
2699         * some random stuff.
2700         *  - Message Type, 1 byte
2701         *  - Payload Length, 2 bytes (unsigned int)
```



```
2702         *  - Payload, the sequence number (2 bytes uint)
2703         *  - Payload, random bytes (16 bytes uint)
2704         *  - Padding
2705         */
2706        buf = OPENSSL_malloc(1 + 2 + payload + padding);
2707        p = buf;
2708        /* Message Type */
2709        *p++ = TLS1_HB_REQUEST;
2710        /* Payload length (18 bytes here) */
2711        s2n(payload, p);
2712        /* Sequence number */
2713        s2n(s->tlsext_hb_seq, p);
2714        /* 16 random bytes */
2715        RAND_pseudo_bytes(p, 16);
2716        p += 16;
2717        /* Random padding */
2718        RAND_pseudo_bytes(p, padding);

2720        ret = ssl3_write_bytes(s, TLS1_RT_HEARTBEAT, buf, 3 + payload + padding)
2721        if (ret >= 0)
2722                {
2723                if (s->msg_callback)
2724                        s->msg_callback(1, s->version, TLS1_RT_HEARTBEAT,
2725                                buf, 3 + payload + padding,
2726                                s, s->msg_callback_arg);

2728                s->tlsext_hb_pending = 1;
2729                }

2731        OPENSSL_free(buf);

2733        return ret;
2734        }
2735 #endif
2736 #endif /* ! codereview */
```

```
**********************************************************
    3812 Wed Aug 13 19:53:42 2014
new/usr/src/lib/openssl/libsunw_ssl/t1_meth.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/t1_meth.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include <openssl/objects.h>
#include "ssl_locl.h"
```



```
static const SSL_METHOD *tls1_get_method(int ver)
        {
        if (ver == TLS1_2_VERSION)
                return TLSv1_2_method();
        if (ver == TLS1_1_VERSION)
                return TLSv1_1_method();
        if (ver == TLS1_VERSION)
                return TLSv1_method();
        return NULL;
        }

IMPLEMENT_tls_meth_func(TLS1_2_VERSION, TLSv1_2_method,
                        ssl3_accept,
                        ssl3_connect,
                        tls1_get_method)

IMPLEMENT_tls_meth_func(TLS1_1_VERSION, TLSv1_1_method,
                        ssl3_accept,
                        ssl3_connect,
                        tls1_get_method)

IMPLEMENT_tls_meth_func(TLS1_VERSION, TLSv1_method,
                        ssl3_accept,
                        ssl3_connect,
                        tls1_get_method)
#endif /* ! codereview */
```

```
**********************************************************
   11030 Wed Aug 13 19:53:42 2014
new/usr/src/lib/openssl/libsunw_ssl/t1_reneg.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/t1_reneg.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */
/* ====================================================================
 * Copyright (c) 1998-2009 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
```



```
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.openssl.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ''AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#include <stdio.h>
#include <openssl/objects.h>
#include "ssl_locl.h"

/* Add the client's renegotiation binding */
int ssl_add_clienthello_renegotiate_ext(SSL *s, unsigned char *p, int *len,
                                        int maxlen)
    {
    if(p)
        {
        if((s->s3->previous_client_finished_len+1) > maxlen)
            {
            SSLerr(SSL_F_SSL_ADD_CLIENTHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATE_E
            return 0;
            }

        /* Length byte */
```

```
        *p = s->s3->previous_client_finished_len;
        p++;

        memcpy(p, s->s3->previous_client_finished,
               s->s3->previous_client_finished_len);
#ifdef OPENSSL_RI_DEBUG
    fprintf(stderr, "%s RI extension sent by client\n",
                s->s3->previous_client_finished_len ? "Non-empty" : "Empty");
#endif
        }

    *len=s->s3->previous_client_finished_len + 1;


    return 1;
    }

/* Parse the client's renegotiation binding and abort if it's not
   right */
int ssl_parse_clienthello_renegotiate_ext(SSL *s, unsigned char *d, int len,
                                          int *al)
    {
    int ilen;

    /* Parse the length byte */
    if(len < 1)
        {
        SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_E
        *al=SSL_AD_ILLEGAL_PARAMETER;
        return 0;
        }
    ilen = *d;
    d++;

    /* Consistency check */
    if((ilen+1) != len)
        {
        SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_E
        *al=SSL_AD_ILLEGAL_PARAMETER;
        return 0;
        }

    /* Check that the extension matches */
    if(ilen != s->s3->previous_client_finished_len)
        {
        SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_M
        *al=SSL_AD_HANDSHAKE_FAILURE;
        return 0;
        }

    if(memcmp(d, s->s3->previous_client_finished,
              s->s3->previous_client_finished_len))
        {
        SSLerr(SSL_F_SSL_PARSE_CLIENTHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_M
        *al=SSL_AD_HANDSHAKE_FAILURE;
        return 0;
        }
#ifdef OPENSSL_RI_DEBUG
    fprintf(stderr, "%s RI extension received by server\n",
                                ilen ? "Non-empty" : "Empty");
#endif

    s->s3->send_connection_binding=1;

    return 1;
    }
```



```
/* Add the server's renegotiation binding */
int ssl_add_serverhello_renegotiate_ext(SSL *s, unsigned char *p, int *len,
                                        int maxlen)
    {
    if(p)
        {
        if((s->s3->previous_client_finished_len +
            s->s3->previous_server_finished_len + 1) > maxlen)
            {
            SSLerr(SSL_F_SSL_ADD_SERVERHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATE_E
            return 0;
            }

        /* Length byte */
        *p = s->s3->previous_client_finished_len + s->s3->previous_server_finish
        p++;

        memcpy(p, s->s3->previous_client_finished,
               s->s3->previous_client_finished_len);
        p += s->s3->previous_client_finished_len;

        memcpy(p, s->s3->previous_server_finished,
               s->s3->previous_server_finished_len);
#ifdef OPENSSL_RI_DEBUG
    fprintf(stderr, "%s RI extension sent by server\n",
                s->s3->previous_client_finished_len ? "Non-empty" : "Empty");
#endif
        }

    *len=s->s3->previous_client_finished_len
        + s->s3->previous_server_finished_len + 1;

    return 1;
    }

/* Parse the server's renegotiation binding and abort if it's not
   right */
int ssl_parse_serverhello_renegotiate_ext(SSL *s, unsigned char *d, int len,
                                          int *al)
    {
    int expected_len=s->s3->previous_client_finished_len
        + s->s3->previous_server_finished_len;
    int ilen;

    /* Check for logic errors */
    OPENSSL_assert(!expected_len || s->s3->previous_client_finished_len);
    OPENSSL_assert(!expected_len || s->s3->previous_server_finished_len);

    /* Parse the length byte */
    if(len < 1)
        {
        SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_E
        *al=SSL_AD_ILLEGAL_PARAMETER;
        return 0;
        }
    ilen = *d;
    d++;

    /* Consistency check */
    if(ilen+1 != len)
        {
        SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_E
        *al=SSL_AD_ILLEGAL_PARAMETER;
        return 0;
        }
```

```
261     /* Check that the extension matches */
262     if(ilen != expected_len)
263         {
264         SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_M
265         *al=SSL_AD_HANDSHAKE_FAILURE;
266         return 0;
267         }

269     if(memcmp(d, s->s3->previous_client_finished,
270               s->s3->previous_client_finished_len))
271         {
272         SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_M
273         *al=SSL_AD_HANDSHAKE_FAILURE;
274         return 0;
275         }
276     d += s->s3->previous_client_finished_len;

278     if(memcmp(d, s->s3->previous_server_finished,
279               s->s3->previous_server_finished_len))
280         {
281         SSLerr(SSL_F_SSL_PARSE_SERVERHELLO_RENEGOTIATE_EXT,SSL_R_RENEGOTIATION_M
282         *al=SSL_AD_ILLEGAL_PARAMETER;
283         return 0;
284         }
285 #ifdef OPENSSL_RI_DEBUG
286     fprintf(stderr, "%s RI extension received by client\n",
287                                 ilen ? "Non-empty" : "Empty");
288 #endif
289     s->s3->send_connection_binding=1;

291     return 1;
292     }
293 #endif /* ! codereview */
```

```
**********************************************************
    4075 Wed Aug 13 19:53:42 2014
new/usr/src/lib/openssl/libsunw_ssl/t1_srvr.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/t1_srvr.c */
/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
 * All rights reserved.
 *
 * This package is an SSL implementation written
 * by Eric Young (eay@cryptsoft.com).
 * The implementation was written so as to conform with Netscapes SSL.
 *
 * This library is free for commercial and non-commercial use as long as
 * the following conditions are aheared to.  The following conditions
 * apply to all code found in this distribution, be it the RC4, RSA,
 * lhash, DES, etc., code; not just the SSL code.  The SSL documentation
 * included with this distribution is covered by the same copyright terms
 * except that the holder is Tim Hudson (tjh@cryptsoft.com).
 *
 * Copyright remains Eric Young's, and as such any Copyright notices in
 * the code are not to be removed.
 * If this package is used in a product, Eric Young should be given attribution
 * as the author of the parts of the library used.
 * This can be in the form of a textual message at program startup or
 * in documentation (online or textual) provided with the package.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *    "This product includes cryptographic software written by
 *     Eric Young (eay@cryptsoft.com)"
 *    The word 'cryptographic' can be left out if the rouines from the library
 *    being used are not cryptographic related :-).
 * 4. If you include any Windows specific code (or a derivative thereof) from
 *    the apps directory (application code) you must include an acknowledgement:
 *    "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
 *
 * THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ''AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 *
 * The licence and distribution terms for any publically available version or
 * derivative of this code cannot be changed.  i.e. this code cannot simply be
 * copied and put under another distribution licence
 * [including the GNU Public Licence.]
 */

#include <stdio.h>
#include "ssl_locl.h"
#include <openssl/buffer.h>
```



```
#include <openssl/rand.h>
#include <openssl/objects.h>
#include <openssl/evp.h>
#include <openssl/x509.h>

static const SSL_METHOD *tls1_get_server_method(int ver);
static const SSL_METHOD *tls1_get_server_method(int ver)
        {
        if (ver == TLS1_2_VERSION)
                return TLSv1_2_server_method();
        if (ver == TLS1_1_VERSION)
                return TLSv1_1_server_method();
        if (ver == TLS1_VERSION)
                return TLSv1_server_method();
        return NULL;
        }

IMPLEMENT_tls_meth_func(TLS1_2_VERSION, TLSv1_2_server_method,
                        ssl3_accept,
                        ssl_undefined_function,
                        tls1_get_server_method)

IMPLEMENT_tls_meth_func(TLS1_1_VERSION, TLSv1_1_server_method,
                        ssl3_accept,
                        ssl_undefined_function,
                        tls1_get_server_method)

IMPLEMENT_tls_meth_func(TLS1_VERSION, TLSv1_server_method,
                        ssl3_accept,
                        ssl_undefined_function,
                        tls1_get_server_method)
#endif /* ! codereview */
```

```
**********************************************************
   14501 Wed Aug 13 19:53:42 2014
new/usr/src/lib/openssl/libsunw_ssl/tls_srp.c
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
/* ssl/tls_srp.c */
/* Written by Christophe Renou (christophe.renou@edelweb.fr) with
 * the precious help of Peter Sylvester (peter.sylvester@edelweb.fr)
 * for the EdelKey project and contributed to the OpenSSL project 2004.
 */
/* ====================================================================
 * Copyright (c) 2004-2011 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.OpenSSL.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    licensing@OpenSSL.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
 *    acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit (http://www.OpenSSL.org/)"
 *
 * THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
 * EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
 * ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
 * OF THE POSSIBILITY OF SUCH DAMAGE.
 * ====================================================================
 *
 * This product includes cryptographic software written by Eric Young
 * (eay@cryptsoft.com).  This product includes software written by Tim
 * Hudson (tjh@cryptsoft.com).
 *
 */
#include "ssl_locl.h"
#ifndef OPENSSL_NO_SRP
```



```
#include <openssl/rand.h>
#include <openssl/srp.h>
#include <openssl/err.h>

int SSL_CTX_SRP_CTX_free(struct ssl_ctx_st *ctx)
        {
        if (ctx == NULL)
                return 0;
        OPENSSL_free(ctx->srp_ctx.login);
        BN_free(ctx->srp_ctx.N);
        BN_free(ctx->srp_ctx.g);
        BN_free(ctx->srp_ctx.s);
        BN_free(ctx->srp_ctx.B);
        BN_free(ctx->srp_ctx.A);
        BN_free(ctx->srp_ctx.a);
        BN_free(ctx->srp_ctx.b);
        BN_free(ctx->srp_ctx.v);
        ctx->srp_ctx.TLS_ext_srp_username_callback = NULL;
        ctx->srp_ctx.SRP_cb_arg = NULL;
        ctx->srp_ctx.SRP_verify_param_callback = NULL;
        ctx->srp_ctx.SRP_give_srp_client_pwd_callback = NULL;
        ctx->srp_ctx.N = NULL;
        ctx->srp_ctx.g = NULL;
        ctx->srp_ctx.s = NULL;
        ctx->srp_ctx.B = NULL;
        ctx->srp_ctx.A = NULL;
        ctx->srp_ctx.a = NULL;
        ctx->srp_ctx.b = NULL;
        ctx->srp_ctx.v = NULL;
        ctx->srp_ctx.login = NULL;
        ctx->srp_ctx.info = NULL;
        ctx->srp_ctx.strength = SRP_MINIMAL_N;
        ctx->srp_ctx.srp_Mask = 0;
        return (1);
        }

int SSL_SRP_CTX_free(struct ssl_st *s)
        {
        if (s == NULL)
                return 0;
        OPENSSL_free(s->srp_ctx.login);
        BN_free(s->srp_ctx.N);
        BN_free(s->srp_ctx.g);
        BN_free(s->srp_ctx.s);
        BN_free(s->srp_ctx.B);
        BN_free(s->srp_ctx.A);
        BN_free(s->srp_ctx.a);
        BN_free(s->srp_ctx.b);
        BN_free(s->srp_ctx.v);
        s->srp_ctx.TLS_ext_srp_username_callback = NULL;
        s->srp_ctx.SRP_cb_arg = NULL;
        s->srp_ctx.SRP_verify_param_callback = NULL;
        s->srp_ctx.SRP_give_srp_client_pwd_callback = NULL;
        s->srp_ctx.N = NULL;
        s->srp_ctx.g = NULL;
        s->srp_ctx.s = NULL;
        s->srp_ctx.B = NULL;
        s->srp_ctx.A = NULL;
        s->srp_ctx.a = NULL;
        s->srp_ctx.b = NULL;
        s->srp_ctx.v = NULL;
        s->srp_ctx.login = NULL;
        s->srp_ctx.info = NULL;
        s->srp_ctx.strength = SRP_MINIMAL_N;
        s->srp_ctx.srp_Mask = 0;
        return (1);
```

```
128         }

130 int SSL_SRP_CTX_init(struct ssl_st *s)
131         {
132         SSL_CTX *ctx;

134         if ((s == NULL) || ((ctx = s->ctx) == NULL))
135                 return 0;
136         s->srp_ctx.SRP_cb_arg = ctx->srp_ctx.SRP_cb_arg;
137         /* set client Hello login callback */
138         s->srp_ctx.TLS_ext_srp_username_callback = ctx->srp_ctx.TLS_ext_srp_user
139         /* set SRP N/g param callback for verification */
140         s->srp_ctx.SRP_verify_param_callback = ctx->srp_ctx.SRP_verify_param_cal
141         /* set SRP client passwd callback */
142         s->srp_ctx.SRP_give_srp_client_pwd_callback = ctx->srp_ctx.SRP_give_srp_

144         s->srp_ctx.N = NULL;
145         s->srp_ctx.g = NULL;
146         s->srp_ctx.s = NULL;
147         s->srp_ctx.B = NULL;
148         s->srp_ctx.A = NULL;
149         s->srp_ctx.a = NULL;
150         s->srp_ctx.b = NULL;
151         s->srp_ctx.v = NULL;
152         s->srp_ctx.login = NULL;
153         s->srp_ctx.info = ctx->srp_ctx.info;
154         s->srp_ctx.strength = ctx->srp_ctx.strength;

156         if (((ctx->srp_ctx.N != NULL) &&
157                  ((s->srp_ctx.N = BN_dup(ctx->srp_ctx.N)) == NULL)) ||
158                 ((ctx->srp_ctx.g != NULL) &&
159                  ((s->srp_ctx.g = BN_dup(ctx->srp_ctx.g)) == NULL)) ||
160                 ((ctx->srp_ctx.s != NULL) &&
161                  ((s->srp_ctx.s = BN_dup(ctx->srp_ctx.s)) == NULL)) ||
162                 ((ctx->srp_ctx.B != NULL) &&
163                  ((s->srp_ctx.B = BN_dup(ctx->srp_ctx.B)) == NULL)) ||
164                 ((ctx->srp_ctx.A != NULL) &&
165                  ((s->srp_ctx.A = BN_dup(ctx->srp_ctx.A)) == NULL)) ||
166                 ((ctx->srp_ctx.a != NULL) &&
167                  ((s->srp_ctx.a = BN_dup(ctx->srp_ctx.a)) == NULL)) ||
168                 ((ctx->srp_ctx.v != NULL) &&
169                  ((s->srp_ctx.v = BN_dup(ctx->srp_ctx.v)) == NULL)) ||
170                 ((ctx->srp_ctx.b != NULL) &&
171                  ((s->srp_ctx.b = BN_dup(ctx->srp_ctx.b)) == NULL)))
172                 {
173                 SSLerr(SSL_F_SSL_SRP_CTX_INIT,ERR_R_BN_LIB);
174                 goto err;
175                 }
176         if ((ctx->srp_ctx.login != NULL) &&
177                 ((s->srp_ctx.login = BUF_strdup(ctx->srp_ctx.login)) == NULL))
178                 {
179                 SSLerr(SSL_F_SSL_SRP_CTX_INIT,ERR_R_INTERNAL_ERROR);
180                 goto err;
181                 }
182         s->srp_ctx.srp_Mask = ctx->srp_ctx.srp_Mask;

184         return (1);
185 err:
186         OPENSSL_free(s->srp_ctx.login);
187         BN_free(s->srp_ctx.N);
188         BN_free(s->srp_ctx.g);
189         BN_free(s->srp_ctx.s);
190         BN_free(s->srp_ctx.B);
191         BN_free(s->srp_ctx.A);
192         BN_free(s->srp_ctx.a);
193         BN_free(s->srp_ctx.b);
```



```
194         BN_free(s->srp_ctx.v);
195         return (0);
196         }

198 int SSL_CTX_SRP_CTX_init(struct ssl_ctx_st *ctx)
199         {
200         if (ctx == NULL)
201                 return 0;

203         ctx->srp_ctx.SRP_cb_arg = NULL;
204         /* set client Hello login callback */
205         ctx->srp_ctx.TLS_ext_srp_username_callback = NULL;
206         /* set SRP N/g param callback for verification */
207         ctx->srp_ctx.SRP_verify_param_callback = NULL;
208         /* set SRP client passwd callback */
209         ctx->srp_ctx.SRP_give_srp_client_pwd_callback = NULL;

211         ctx->srp_ctx.N = NULL;
212         ctx->srp_ctx.g = NULL;
213         ctx->srp_ctx.s = NULL;
214         ctx->srp_ctx.B = NULL;
215         ctx->srp_ctx.A = NULL;
216         ctx->srp_ctx.a = NULL;
217         ctx->srp_ctx.b = NULL;
218         ctx->srp_ctx.v = NULL;
219         ctx->srp_ctx.login = NULL;
220         ctx->srp_ctx.srp_Mask = 0;
221         ctx->srp_ctx.info = NULL;
222         ctx->srp_ctx.strength = SRP_MINIMAL_N;

224         return (1);
225         }

227 /* server side */
228 int SSL_srp_server_param_with_username(SSL *s, int *ad)
229         {
230         unsigned char b[SSL_MAX_MASTER_KEY_LENGTH];
231         int al;

233         *ad = SSL_AD_UNKNOWN_PSK_IDENTITY;
234         if ((s->srp_ctx.TLS_ext_srp_username_callback !=NULL) &&
235                 ((al = s->srp_ctx.TLS_ext_srp_username_callback(s, ad, s->srp_ct
236                         return al;

238         *ad = SSL_AD_INTERNAL_ERROR;
239         if ((s->srp_ctx.N == NULL) ||
240                 (s->srp_ctx.g == NULL) ||
241                 (s->srp_ctx.s == NULL) ||
242                 (s->srp_ctx.v == NULL))
243                 return SSL3_AL_FATAL;

245         if (RAND_bytes(b, sizeof(b)) <= 0)
246                 return SSL3_AL_FATAL;
247         s->srp_ctx.b = BN_bin2bn(b,sizeof(b),NULL);
248         OPENSSL_cleanse(b,sizeof(b));

250         /* Calculate:  B = (kv + g^b) % N  */

252         return ((s->srp_ctx.B = SRP_Calc_B(s->srp_ctx.b, s->srp_ctx.N, s->srp_ct
253                         SSL_ERROR_NONE:SSL3_AL_FATAL;
254         }

256 /* If the server just has the raw password, make up a verifier entry on the fly
257 int SSL_set_srp_server_param_pw(SSL *s, const char *user, const char *pass, cons
258         {
259         SRP_gN *GN = SRP_get_default_gN(grp);
```

```
260         if(GN == NULL) return -1;
261         s->srp_ctx.N = BN_dup(GN->N);
262         s->srp_ctx.g = BN_dup(GN->g);
263         if(s->srp_ctx.v != NULL)
264                 {
265                 BN_clear_free(s->srp_ctx.v);
266                 s->srp_ctx.v = NULL;
267                 }
268         if(s->srp_ctx.s != NULL)
269                 {
270                 BN_clear_free(s->srp_ctx.s);
271                 s->srp_ctx.s = NULL;
272                 }
273         if(!SRP_create_verifier_BN(user, pass, &s->srp_ctx.s, &s->srp_ctx.v, GN-

275         return 1;
276         }

278 int SSL_set_srp_server_param(SSL *s, const BIGNUM *N, const BIGNUM *g,
279                              BIGNUM *sa, BIGNUM *v, char *info)
280         {
281         if (N!= NULL)
282                 {
283                 if (s->srp_ctx.N != NULL)
284                         {
285                         if (!BN_copy(s->srp_ctx.N,N))
286                                 {
287                                 BN_free(s->srp_ctx.N);
288                                 s->srp_ctx.N = NULL;
289                                 }
290                         }
291                 else
292                         s->srp_ctx.N = BN_dup(N);
293                 }
294         if (g!= NULL)
295                 {
296                 if (s->srp_ctx.g != NULL)
297                         {
298                         if (!BN_copy(s->srp_ctx.g,g))
299                                 {
300                                 BN_free(s->srp_ctx.g);
301                                 s->srp_ctx.g = NULL;
302                                 }
303                         }
304                 else
305                         s->srp_ctx.g = BN_dup(g);
306                 }
307         if (sa!= NULL)
308                 {
309                 if (s->srp_ctx.s != NULL)
310                         {
311                         if (!BN_copy(s->srp_ctx.s,sa))
312                                 {
313                                 BN_free(s->srp_ctx.s);
314                                 s->srp_ctx.s = NULL;
315                                 }
316                         }
317                 else
318                         s->srp_ctx.s = BN_dup(sa);
319                 }
320         if (v!= NULL)
321                 {
322                 if (s->srp_ctx.v != NULL)
323                         {
324                         if (!BN_copy(s->srp_ctx.v,v))
325                                 {
```



```
326                                 BN_free(s->srp_ctx.v);
327                                 s->srp_ctx.v = NULL;
328                                 }
329                         }
330                 else
331                         s->srp_ctx.v = BN_dup(v);
332                 }
333         s->srp_ctx.info = info;

335         if (!(s->srp_ctx.N) ||
336                 !(s->srp_ctx.g) ||
337                 !(s->srp_ctx.s) ||
338                 !(s->srp_ctx.v))
339                 return -1;

341         return 1;
342         }

344 int SRP_generate_server_master_secret(SSL *s,unsigned char *master_key)
345         {
346         BIGNUM *K = NULL, *u = NULL;
347         int ret = -1, tmp_len;
348         unsigned char *tmp = NULL;

350         if (!SRP_Verify_A_mod_N(s->srp_ctx.A,s->srp_ctx.N))
351                 goto err;
352         if (!(u = SRP_Calc_u(s->srp_ctx.A,s->srp_ctx.B,s->srp_ctx.N)))
353                 goto err;
354         if (!(K = SRP_Calc_server_key(s->srp_ctx.A, s->srp_ctx.v, u, s->srp_ctx.
355                 goto err;

357         tmp_len = BN_num_bytes(K);
358         if ((tmp = OPENSSL_malloc(tmp_len)) == NULL)
359                 goto err;
360         BN_bn2bin(K, tmp);
361         ret = s->method->ssl3_enc->generate_master_secret(s,master_key,tmp,tmp_l
362 err:
363         if (tmp)
364                 {
365                 OPENSSL_cleanse(tmp,tmp_len) ;
366                 OPENSSL_free(tmp);
367                 }
368         BN_clear_free(K);
369         BN_clear_free(u);
370         return ret;
371         }

373 /* client side */
374 int SRP_generate_client_master_secret(SSL *s,unsigned char *master_key)
375         {
376         BIGNUM *x = NULL, *u = NULL, *K = NULL;
377         int ret = -1, tmp_len;
378         char *passwd = NULL;
379         unsigned char *tmp = NULL;

381         /* Checks if b % n == 0
382          */
383         if (SRP_Verify_B_mod_N(s->srp_ctx.B,s->srp_ctx.N)==0) goto err;
384         if (!(u = SRP_Calc_u(s->srp_ctx.A,s->srp_ctx.B,s->srp_ctx.N))) goto err;
385         if (s->srp_ctx.SRP_give_srp_client_pwd_callback == NULL) goto err;
386         if (!(passwd = s->srp_ctx.SRP_give_srp_client_pwd_callback(s, s->srp_ctx
387         if (!(x = SRP_Calc_x(s->srp_ctx.s,s->srp_ctx.login,passwd))) goto err;
388         if (!(K = SRP_Calc_client_key(s->srp_ctx.N, s->srp_ctx.B, s->srp_ctx.g,

390         tmp_len = BN_num_bytes(K);
391         if ((tmp = OPENSSL_malloc(tmp_len)) == NULL) goto err;
```

```
392         BN_bn2bin(K, tmp);
393         ret = s->method->ssl3_enc->generate_master_secret(s,master_key,tmp,tmp_l
394 err:
395         if (tmp)
396                 {
397                 OPENSSL_cleanse(tmp,tmp_len) ;
398                 OPENSSL_free(tmp);
399                 }
400         BN_clear_free(K);
401         BN_clear_free(x);
402         if (passwd)
403                 {
404                 OPENSSL_cleanse(passwd,strlen(passwd)) ;
405                 OPENSSL_free(passwd);
406                 }
407         BN_clear_free(u);
408         return ret;
409         }

411 int srp_verify_server_param(SSL *s, int *al)
412         {
413         SRP_CTX *srp = &s->srp_ctx;
414         /* Sanity check parameters: we can quickly check B % N == 0
415          * by checking B != 0 since B < N
416          */
417         if (BN_ucmp(srp->g, srp->N) >=0 || BN_ucmp(srp->B, srp->N) >= 0
418                 || BN_is_zero(srp->B))
419                 {
420                 *al = SSL3_AD_ILLEGAL_PARAMETER;
421                 return 0;
422                 }

424         if (BN_num_bits(srp->N) < srp->strength)
425                 {
426                 *al = TLS1_AD_INSUFFICIENT_SECURITY;
427                 return 0;
428                 }

430         if (srp->SRP_verify_param_callback)
431                 {
432                 if (srp->SRP_verify_param_callback(s, srp->SRP_cb_arg) <= 0)
433                         {
434                         *al = TLS1_AD_INSUFFICIENT_SECURITY;
435                         return 0;
436                         }
437                 }
438         else if(!SRP_check_known_gN_param(srp->g, srp->N))
439                 {
440                 *al = TLS1_AD_INSUFFICIENT_SECURITY;
441                 return 0;
442                 }

444         return 1;
445         }


448 int SRP_Calc_A_param(SSL *s)
449         {
450         unsigned char rnd[SSL_MAX_MASTER_KEY_LENGTH];

452         RAND_bytes(rnd, sizeof(rnd));
453         s->srp_ctx.a = BN_bin2bn(rnd, sizeof(rnd), s->srp_ctx.a);
454         OPENSSL_cleanse(rnd, sizeof(rnd));

456         if (!(s->srp_ctx.A = SRP_Calc_A(s->srp_ctx.a,s->srp_ctx.N,s->srp_ctx.g))
457                 return -1;
```



```
459         return 1;
460         }

462 BIGNUM *SSL_get_srp_g(SSL *s)
463         {
464         if (s->srp_ctx.g != NULL)
465                 return s->srp_ctx.g;
466         return s->ctx->srp_ctx.g;
467         }

469 BIGNUM *SSL_get_srp_N(SSL *s)
470         {
471         if (s->srp_ctx.N != NULL)
472                 return s->srp_ctx.N;
473         return s->ctx->srp_ctx.N;
474         }

476 char *SSL_get_srp_username(SSL *s)
477         {
478         if (s->srp_ctx.login != NULL)
479                 return s->srp_ctx.login;
480         return s->ctx->srp_ctx.login;
481         }

483 char *SSL_get_srp_userinfo(SSL *s)
484         {
485         if (s->srp_ctx.info != NULL)
486                 return s->srp_ctx.info;
487         return s->ctx->srp_ctx.info;
488         }

490 #define tls1_ctx_ctrl ssl3_ctx_ctrl
491 #define tls1_ctx_callback_ctrl ssl3_ctx_callback_ctrl

493 int SSL_CTX_set_srp_username(SSL_CTX *ctx,char *name)
494         {
495         return tls1_ctx_ctrl(ctx,SSL_CTRL_SET_TLS_EXT_SRP_USERNAME,0,name);
496         }

498 int SSL_CTX_set_srp_password(SSL_CTX *ctx,char *password)
499         {
500         return tls1_ctx_ctrl(ctx,SSL_CTRL_SET_TLS_EXT_SRP_PASSWORD,0,password);
501         }

503 int SSL_CTX_set_srp_strength(SSL_CTX *ctx, int strength)
504         {
505         return tls1_ctx_ctrl(ctx, SSL_CTRL_SET_TLS_EXT_SRP_STRENGTH, strength,
506                              NULL);
507         }

509 int SSL_CTX_set_srp_verify_param_callback(SSL_CTX *ctx, int (*cb)(SSL *,void *))
510         {
511         return tls1_ctx_callback_ctrl(ctx,SSL_CTRL_SET_SRP_VERIFY_PARAM_CB,
512                                       (void (*)(void))cb);
513         }

515 int SSL_CTX_set_srp_cb_arg(SSL_CTX *ctx, void *arg)
516         {
517         return tls1_ctx_ctrl(ctx,SSL_CTRL_SET_SRP_ARG,0,arg);
518         }

520 int SSL_CTX_set_srp_username_callback(SSL_CTX *ctx,
521                                       int (*cb)(SSL *,int *,void *))
522         {
523         return tls1_ctx_callback_ctrl(ctx,SSL_CTRL_SET_TLS_EXT_SRP_USERNAME_CB,
```

```
524                                     (void (*)(void))cb);
525         }

527 int SSL_CTX_set_srp_client_pwd_callback(SSL_CTX *ctx, char *(*cb)(SSL *,void *))
528         {
529         return tls1_ctx_callback_ctrl(ctx,SSL_CTRL_SET_SRP_GIVE_CLIENT_PWD_CB,
530                                     (void (*)(void))cb);
531         }

533 #endif
534 #endif /* ! codereview */
```

```
**********************************************************
    2447 Wed Aug 13 19:53:42 2014
new/usr/src/lib/pkcs11/pkcs11_tpm/Makefile.com
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
LIBRARY =       pkcs11_tpm.a
VERS =          .1

OBJECTS= api_interface.o \
        apiutil.o \
        asn1.o \
        cert.o \
        data_obj.o \
        decr_mgr.o \
        dig_mgr.o \
        encr_mgr.o \
        globals.o \
        hwf_obj.o \
        key.o \
        key_mgr.o \
        loadsave.o \
        log.o \
        mech_md5.o \
        mech_rsa.o \
        mech_sha.o \
        new_host.o \
        obj_mgr.o \
        object.o \
        sess_mgr.o \
        sign_mgr.o \
        template.o \
        tpm_specific.o \
        utility.o \
        verify_mgr.o


include $(SRC)/lib/Makefile.lib

SRCDIR= ../common

SRCS=   $(OBJECTS:%.o=$(SRCDIR)/%.c)

#       set signing mode
```



```
POST_PROCESS_SO +=      ; $(ELFSIGN_CRYPTO)

ROOTLIBDIR=$(ROOT)/usr/lib/security
ROOTLIBDIR64=$(ROOT)/usr/lib/security/$(MACH64)

LIBS=$(DYNLIB) $(DYNLIB64)

TSSROOT=$(ADJUNCT_PROTO)
TSPILIBDIR=$(TSSROOT)/usr/lib
TSPIINCDIR=$(TSSROOT)/usr/include
TSSLIB=-L$(TSPILIBDIR)
TSSLIB64=-L$(TSPILIBDIR)/$(MACH64)
TSSINC=-I$(TSPIINCDIR)

LDLIBS += $(TSSLIB) -L$(ADJUNCT_PROTO)/lib -lc -luuid -lmd -ltspi

# libsunw_crypto has no lint library, so we can only use it when
# building
$(LIBS) := LDLIBS += -lsunw_crypto
LDLIBS += $(TSSLIB) -L$(ADJUNCT_PROTO)/lib -lc -luuid -lmd -ltspi -lcrypto
CPPFLAGS += -xCC -D_POSIX_PTHREAD_SEMANTICS $(TSSINC)
CPPFLAGS64 += $(CPPFLAGS)
C99MODE=        $(C99_ENABLE)

CERRWARN +=     -_gcc=-Wno-parentheses
CERRWARN +=     -_gcc=-Wno-unused-label
CERRWARN +=     -_gcc=-Wno-uninitialized

LINTSRC= $(OBJECTS:%.o=$(SRCDIR)/%.c)

$(LINTLIB):=    SRCS    =       $(SRCDIR)/$(LINTSRC)
LINTSRC= $(SRCS)

CLOBBERFILES += C.ln

.KEEP_STATE:

all: $(LIBS)

lint: $$(LINTSRC)
        $(LINT.c) $(LINTCHECKFLAGS) $(LINTSRC) $(LDLIBS)

pics/%.o: $(SRCDIR)/%.c
        $(COMPILE.c) -o $@ $<
        $(POST_PROCESS_O)

include $(SRC)/lib/Makefile.targ
```

```
**********************************************************
   60465 Wed Aug 13 19:53:43 2014
new/usr/src/pkg/manifests/system-library.mf
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
```

```
   1 #
   2 # CDDL HEADER START
   3 #
   4 # The contents of this file are subject to the terms of the
   5 # Common Development and Distribution License (the "License").
   6 # You may not use this file except in compliance with the License.
   7 #
   8 # You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9 # or http://www.opensolaris.org/os/licensing.
  10 # See the License for the specific language governing permissions
  11 # and limitations under the License.
  12 #
  13 # When distributing Covered Code, include this CDDL HEADER in each
  14 # file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15 # If applicable, add the following below this CDDL HEADER, with the
  16 # fields enclosed by brackets "[]" replaced with your own identifying
  17 # information: Portions Copyright [yyyy] [name of copyright owner]
  18 #
  19 # CDDL HEADER END
  20 #

  22 #
  23 # Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
  24 # Copyright (c) 2010, Oracle and/or its affiliates. All rights reserved.
  25 # Copyright 2012 OmniTI Computer Consulting, Inc.  All rights reserved.
  26 # Copyright (c) 2013 Gary Mills
  27 #

  29 <include system-library.man3.inc>
  30 <include system-library.man3bsm.inc>
  31 <include system-library.man3c.inc>
  32 <include system-library.man3c_db.inc>
  33 <include system-library.man3cfgadm.inc>
  34 <include system-library.man3commputil.inc>
  35 <include system-library.man3contract.inc>
  36 <include system-library.man3curses.inc>
  37 <include system-library.man3devid.inc>
  38 <include system-library.man3devinfo.inc>
  39 <include system-library.man3dlpi.inc>
  40 <include system-library.man3elf.inc>
  41 <include system-library.man3exacct.inc>
  42 <include system-library.man3ext.inc>
  43 <include system-library.man3fstyp.inc>
  44 <include system-library.man3gen.inc>
  45 <include system-library.man3kstat.inc>
  46 <include system-library.man3kvm.inc>
  47 <include system-library.man3ldap.inc>
  48 <include system-library.man3lgrp.inc>
  49 <include system-library.man3lib.inc>
  50 <include system-library.man3mail.inc>
  51 <include system-library.man3malloc.inc>
  52 <include system-library.man3mp.inc>
  53 <include system-library.man3nsl.inc>
  54 <include system-library.man3nvpair.inc>
  55 <include system-library.man3pam.inc>
  56 <include system-library.man3scf.inc>
  57 <include system-library.man3sec.inc>
  58 <include system-library.man3secdb.inc>
  59 <include system-library.man3sip.inc>
  60 <include system-library.man3socket.inc>
  61 <include system-library.man3tsol.inc>
```



```
  62 <include system-library.man3uuid.inc>
  63 <include system-library.man3volmgt.inc>
  64 <include system-library.man3xcurses.inc>
  65 <include system-library.man3xnet.inc>
  66 <include system-library.man4.inc>
  67 <include system-library.man5.inc>
  68 <include system-library.man7p.inc>
  69 set name=pkg.fmri value=pkg:/system/library@$(PKGVERS)
  70 set name=pkg.description \
  71     value="core shared libraries for a specific instruction-set architecture"
  72 set name=pkg.summary value="Core Solaris, (Shared Libs)"
  73 set name=info.classification value=org.opensolaris.category.2008:System/Core
  74 set name=variant.arch value=$(ARCH)
  75 $(i386_ONLY)dir path=etc group=sys
  76 $(i386_ONLY)dir path=etc/flash group=sys
  77 $(i386_ONLY)dir path=etc/flash/postcreation group=sys mode=0700
  78 $(i386_ONLY)dir path=etc/flash/precreation group=sys mode=0700
  79 $(i386_ONLY)dir path=etc/flash/preexit group=sys mode=0700
  80 dir path=lib
  81 dir path=lib/$(ARCH64)
  82 dir path=lib/crypto
  83 dir path=lib/crypto/$(ARCH64)
  84 dir path=lib/mpxio
  85 dir path=lib/secure
  86 dir path=lib/secure/$(ARCH64)
  87 dir path=usr group=sys
  88 dir path=usr/bin
  89 dir path=usr/ccs
  90 dir path=usr/ccs/lib
  91 dir path=usr/ccs/lib/$(ARCH64)
  92 dir path=usr/lib
  93 dir path=usr/lib/$(ARCH64)
  94 dir path=usr/lib/cfgadm
  95 dir path=usr/lib/cfgadm/$(ARCH64)
  96 dir path=usr/lib/iconv/$(ARCH64)
  97 $(i386_ONLY)dir path=usr/lib/libc
  98 dir path=usr/lib/lwp
  99 dir path=usr/lib/lwp/$(ARCH64)
 100 dir path=usr/lib/python2.6
 101 dir path=usr/lib/python2.6/vendor-packages
 102 dir path=usr/lib/python2.6/vendor-packages/solaris
 103 dir path=usr/lib/raidcfg
 104 dir path=usr/lib/raidcfg/$(ARCH64)
 105 dir path=usr/lib/scsi
 106 dir path=usr/lib/scsi/$(ARCH64)
 107 dir path=usr/lib/scsi/plugins
 108 dir path=usr/lib/scsi/plugins/scsi
 109 dir path=usr/lib/scsi/plugins/scsi/engines
 110 dir path=usr/lib/scsi/plugins/scsi/engines/$(ARCH64)
 111 dir path=usr/lib/scsi/plugins/ses
 112 dir path=usr/lib/scsi/plugins/ses/framework
 113 dir path=usr/lib/scsi/plugins/ses/framework/$(ARCH64)
 114 dir path=usr/lib/scsi/plugins/ses/vendor
 115 $(sparc_ONLY)dir path=usr/lib/scsi/plugins/ses/vendor/$(ARCH64)
 116 dir path=usr/lib/scsi/plugins/smp
 117 dir path=usr/lib/scsi/plugins/smp/engine
 118 dir path=usr/lib/scsi/plugins/smp/engine/$(ARCH64)
 119 dir path=usr/lib/scsi/plugins/smp/framework
 120 dir path=usr/lib/scsi/plugins/smp/framework/$(ARCH64)
 121 dir path=usr/lib/security
 122 dir path=usr/lib/security/$(ARCH64)
 123 dir path=usr/share/man
 124 dir path=usr/share/man/man3
 125 dir path=usr/share/man/man3bsm
 126 dir path=usr/share/man/man3c
 127 dir path=usr/share/man/man3c_db
```

```
dir path=usr/share/man/man3cfgadm
dir path=usr/share/man/man3commputil
dir path=usr/share/man/man3contract
dir path=usr/share/man/man3curses
dir path=usr/share/man/man3devid
dir path=usr/share/man/man3devinfo
dir path=usr/share/man/man3dlpi
dir path=usr/share/man/man3elf
dir path=usr/share/man/man3exacct
dir path=usr/share/man/man3ext
dir path=usr/share/man/man3fstyp
dir path=usr/share/man/man3gen
dir path=usr/share/man/man3kstat
dir path=usr/share/man/man3kvm
dir path=usr/share/man/man3ldap
dir path=usr/share/man/man3lgrp
dir path=usr/share/man/man3lib
dir path=usr/share/man/man3mail
dir path=usr/share/man/man3malloc
dir path=usr/share/man/man3mp
dir path=usr/share/man/man3nsl
dir path=usr/share/man/man3nvpair
dir path=usr/share/man/man3pam
dir path=usr/share/man/man3pool
dir path=usr/share/man/man3scf
dir path=usr/share/man/man3sec
dir path=usr/share/man/man3secdb
dir path=usr/share/man/man3sip
dir path=usr/share/man/man3socket
dir path=usr/share/man/man3tsol
dir path=usr/share/man/man3uuid
dir path=usr/share/man/man3volmgt
dir path=usr/share/man/man3xcurses
dir path=usr/share/man/man3xnet
dir path=usr/share/man/man5
dir path=usr/share/man/man7p
dir path=usr/xpg4
dir path=usr/xpg4/lib
dir path=usr/xpg4/lib/$(ARCH64)
$(i386_ONLY)file path=etc/flash/precreation/caplib group=sys mode=0500
file path=lib/$(ARCH64)/c_synonyms.so.1
file path=lib/$(ARCH64)/ld.so.1
file path=lib/$(ARCH64)/libadm.so.1
file path=lib/$(ARCH64)/libaio.so.1
file path=lib/$(ARCH64)/libavl.so.1
file path=lib/$(ARCH64)/libbsm.so.1
file path=lib/$(ARCH64)/libc.so.1
file path=lib/$(ARCH64)/libc_db.so.1
file path=lib/$(ARCH64)/libcmdutils.so.1
file path=lib/$(ARCH64)/libcontract.so.1
file path=lib/$(ARCH64)/libcryptoutil.so.1
file path=lib/$(ARCH64)/libctf.so.1
file path=lib/$(ARCH64)/libcurses.so.1
file path=lib/$(ARCH64)/libdevice.so.1
file path=lib/$(ARCH64)/libdevid.so.1
file path=lib/$(ARCH64)/libdevinfo.so.1
file path=lib/$(ARCH64)/libdhcputil.so.1
file path=lib/$(ARCH64)/libdl.so.1
file path=lib/$(ARCH64)/libdladm.so.1
file path=lib/$(ARCH64)/libdlpi.so.1
file path=lib/$(ARCH64)/libdoor.so.1
file path=lib/$(ARCH64)/libefi.so.1
file path=lib/$(ARCH64)/libelf.so.1
$(i386_ONLY)file path=lib/$(ARCH64)/libfdisk.so.1
file path=lib/$(ARCH64)/libgen.so.1
file path=lib/$(ARCH64)/libinetutil.so.1
```



```
file path=lib/$(ARCH64)/libintl.so.1
file path=lib/$(ARCH64)/libkmf.so.1
file path=lib/$(ARCH64)/libkmfberder.so.1
file path=lib/$(ARCH64)/libkstat.so.1
file path=lib/$(ARCH64)/libld.so.4
file path=lib/$(ARCH64)/liblddbg.so.4
file path=lib/$(ARCH64)/libmd.so.1
file path=lib/$(ARCH64)/libmd5.so.1
file path=lib/$(ARCH64)/libmp.so.2
file path=lib/$(ARCH64)/libnsl.so.1
file path=lib/$(ARCH64)/libnvpair.so.1
file path=lib/$(ARCH64)/libpam.so.1
file path=lib/$(ARCH64)/libproc.so.1
file path=lib/$(ARCH64)/libpthread.so.1
file path=lib/$(ARCH64)/librcm.so.1
file path=lib/$(ARCH64)/libresolv.so.2
file path=lib/$(ARCH64)/librestart.so.1
file path=lib/$(ARCH64)/librpcsvc.so.1
file path=lib/$(ARCH64)/librt.so.1
file path=lib/$(ARCH64)/librtld.so.1
file path=lib/$(ARCH64)/librtld_db.so.1
file path=lib/$(ARCH64)/libscf.so.1
file path=lib/$(ARCH64)/libsec.so.1
file path=lib/$(ARCH64)/libsecdb.so.1
file path=lib/$(ARCH64)/libsendfile.so.1
file path=lib/$(ARCH64)/libsocket.so.1
file path=lib/$(ARCH64)/libsysevent.so.1
file path=lib/$(ARCH64)/libtermcap.so.1
file path=lib/$(ARCH64)/libthread.so.1
file path=lib/$(ARCH64)/libtsnet.so.1
file path=lib/$(ARCH64)/libtsol.so.2
file path=lib/$(ARCH64)/libumem.so.1
file path=lib/$(ARCH64)/libuuid.so.1
file path=lib/$(ARCH64)/libuutil.so.1
file path=lib/$(ARCH64)/libw.so.1
file path=lib/$(ARCH64)/libxnet.so.1
file path=lib/$(ARCH64)/nss_compat.so.1
file path=lib/$(ARCH64)/nss_dns.so.1
file path=lib/$(ARCH64)/nss_files.so.1
file path=lib/$(ARCH64)/nss_nis.so.1
file path=lib/$(ARCH64)/nss_user.so.1
file path=lib/c_synonyms.so.1
file path=lib/crypto/$(ARCH64)/kmf_mapper_cn.so.1
file path=lib/crypto/$(ARCH64)/kmf_nss.so.1
file path=lib/crypto/$(ARCH64)/kmf_openssl.so.1
file path=lib/crypto/$(ARCH64)/kmf_pkcs11.so.1
file path=lib/crypto/kmf_mapper_cn.so.1
file path=lib/crypto/kmf_nss.so.1
file path=lib/crypto/kmf_openssl.so.1
file path=lib/crypto/kmf_pkcs11.so.1
file path=lib/ld.so.1
file path=lib/libadm.so.1
file path=lib/libaio.so.1
file path=lib/libavl.so.1
file path=lib/libbsm.so.1
file path=lib/libc.so.1 reboot-needed=true
file path=lib/libc_db.so.1
file path=lib/libcmdutils.so.1
file path=lib/libcontract.so.1
file path=lib/libcryptoutil.so.1
file path=lib/libctf.so.1
file path=lib/libcurses.so.1
file path=lib/libdevice.so.1
file path=lib/libdevid.so.1
file path=lib/libdevinfo.so.1
file path=lib/libdhcpagent.so.1
```

```
260 file path=lib/libdhcputil.so.1
261 file path=lib/libdl.so.1
262 file path=lib/libdladm.so.1
263 file path=lib/libdlpi.so.1
264 file path=lib/libdoor.so.1
265 file path=lib/libefi.so.1
266 file path=lib/libelf.so.1
267 file path=lib/libelfsign.so.1
268 $(i386_ONLY)file path=lib/libfdisk.so.1
269 file path=lib/libgen.so.1
270 file path=lib/libinetutil.so.1
271 file path=lib/libintl.so.1
272 file path=lib/libipadm.so.1
273 file path=lib/libipmp.so.1
274 file path=lib/libkcfd.so.1
275 file path=lib/libkmf.so.1
276 file path=lib/libkmfberder.so.1
277 file path=lib/libkstat.so.1
278 file path=lib/libld.so.4
279 file path=lib/liblddbg.so.4
280 file path=lib/libmd.so.1
281 file path=lib/libmd5.so.1
282 file path=lib/libmp.so.1
283 file path=lib/libmp.so.2
284 file path=lib/libnsl.so.1
285 file path=lib/libnvpair.so.1
286 file path=lib/libnwam.so.1
287 file path=lib/libpam.so.1
288 file path=lib/libproc.so.1
289 file path=lib/libpthread.so.1
290 file path=lib/librcm.so.1
291 file path=lib/libresolv.so.1
292 file path=lib/libresolv.so.2
293 file path=lib/librestart.so.1
294 file path=lib/librpcsvc.so.1
295 file path=lib/librt.so.1
296 file path=lib/librtld.so.1
297 file path=lib/librtld_db.so.1
298 file path=lib/libscf.so.1
299 file path=lib/libsec.so.1
300 file path=lib/libsecdb.so.1
301 file path=lib/libsendfile.so.1
302 file path=lib/libsocket.so.1
303 file path=lib/libsysevent.so.1
304 file path=lib/libtermcap.so.1
305 file path=lib/libthread.so.1
306 file path=lib/libtsnet.so.1
307 file path=lib/libtsol.so.2
308 file path=lib/libumem.so.1
309 file path=lib/libuuid.so.1
310 file path=lib/libuutil.so.1
311 file path=lib/libw.so.1
312 file path=lib/libxnet.so.1
313 file path=lib/mpxio/stmsboot_util mode=0555
314 file path=lib/nss_compat.so.1
315 file path=lib/nss_dns.so.1
316 file path=lib/nss_files.so.1
317 file path=lib/nss_nis.so.1
318 file path=lib/nss_user.so.1
319 file path=usr/lib/$(ARCH64)/0@0.so.1
320 file path=usr/lib/$(ARCH64)/getloginx.so.1
321 file path=usr/lib/$(ARCH64)/libadutils.so.1
322 file path=usr/lib/$(ARCH64)/libast.so.1
323 file path=usr/lib/$(ARCH64)/libbsdmalloc.so.1
324 file path=usr/lib/$(ARCH64)/libcfgadm.so.1
325 file path=usr/lib/$(ARCH64)/libcmd.so.1
```



```
326 file path=usr/lib/$(ARCH64)/libcommputil.so.1
327 file path=usr/lib/$(ARCH64)/libcrle.so.1
328 file path=usr/lib/$(ARCH64)/libcrypt.so.1
329 file path=usr/lib/$(ARCH64)/libdisasm.so.1
330 file path=usr/lib/$(ARCH64)/libdll.so.1
331 file path=usr/lib/$(ARCH64)/libexacct.so.1
332 file path=usr/lib/$(ARCH64)/libform.so.1
333 file path=usr/lib/$(ARCH64)/libfstyp.so.1
334 file path=usr/lib/$(ARCH64)/libhotplug.so.1
335 file path=usr/lib/$(ARCH64)/libidmap.so.1
336 file path=usr/lib/$(ARCH64)/libike.so.1
337 file path=usr/lib/$(ARCH64)/libipmi.so.1
338 file path=usr/lib/$(ARCH64)/libipp.so.1
339 file path=usr/lib/$(ARCH64)/libipsecutil.so.1
340 file path=usr/lib/$(ARCH64)/libkvm.so.1
341 file path=usr/lib/$(ARCH64)/libl.so.1
342 file path=usr/lib/$(ARCH64)/libldap.so.5
343 file path=usr/lib/$(ARCH64)/liblgrp.so.1
344 file path=usr/lib/$(ARCH64)/liblm.so.1
345 file path=usr/lib/$(ARCH64)/libmail.so.1
346 file path=usr/lib/$(ARCH64)/libmalloc.so.1
347 file path=usr/lib/$(ARCH64)/libmapmalloc.so.1
348 file path=usr/lib/$(ARCH64)/libmenu.so.1
349 file path=usr/lib/$(ARCH64)/libmtmalloc.so.1
350 file path=usr/lib/$(ARCH64)/libnls.so.1
351 file path=usr/lib/$(ARCH64)/libpanel.so.1
352 file path=usr/lib/$(ARCH64)/libpcidb.so.1
353 file path=usr/lib/$(ARCH64)/libpkcs11.so.1
354 file path=usr/lib/$(ARCH64)/libproject.so.1
355 file path=usr/lib/$(ARCH64)/libraidcfg.so.1
356 file path=usr/lib/$(ARCH64)/libreparse.so.1
357 $(i386_ONLY)file path=usr/lib/$(ARCH64)/libsaveargs.so.1
358 file path=usr/lib/$(ARCH64)/libsched.so.1
359 file path=usr/lib/$(ARCH64)/libsctp.so.1
360 file path=usr/lib/$(ARCH64)/libshell.so.1
361 file path=usr/lib/$(ARCH64)/libsip.so.1
362 file path=usr/lib/$(ARCH64)/libsldap.so.1
363 file path=usr/lib/$(ARCH64)/libsmbios.so.1
364 file path=usr/lib/$(ARCH64)/libsoftcrypto.so.1
365 file path=usr/lib/$(ARCH64)/libsum.so.1
366 file path=usr/lib/$(ARCH64)/libsunw_crypto.so.1
367 file path=usr/lib/$(ARCH64)/libsunw_ssl.so.1
368 #endif /* ! codereview */
369 $(sparc_ONLY)file path=usr/lib/$(ARCH64)/libv12n.so.1
370 file path=usr/lib/$(ARCH64)/libvolmgt.so.1
371 file path=usr/lib/$(ARCH64)/libwrap.so.1.0
372 file path=usr/lib/$(ARCH64)/liby.so.1
373 file path=usr/lib/$(ARCH64)/libzoneinfo.so.1
374 file path=usr/lib/$(ARCH64)/nss_ad.so.1
375 file path=usr/lib/$(ARCH64)/nss_ldap.so.1
376 file path=usr/lib/$(ARCH64)/passwdutil.so.1
377 file path=usr/lib/$(ARCH64)/straddr.so.2
378 file path=usr/lib/$(ARCH64)/watchmalloc.so.1
379 file path=usr/lib/0@0.so.1
380 file path=usr/lib/cfgadm/$(ARCH64)/ib.so.1
381 file path=usr/lib/cfgadm/$(ARCH64)/pci.so.1
382 $(i386_ONLY)file path=usr/lib/cfgadm/$(ARCH64)/sata.so.1
383 file path=usr/lib/cfgadm/$(ARCH64)/scsi.so.1
384 file path=usr/lib/cfgadm/$(ARCH64)/shp.so.1
385 file path=usr/lib/cfgadm/$(ARCH64)/usb.so.1
386 file path=usr/lib/cfgadm/ib.so.1
387 file path=usr/lib/cfgadm/pci.so.1
388 $(i386_ONLY)file path=usr/lib/cfgadm/sata.so.1
389 file path=usr/lib/cfgadm/scsi.so.1
390 file path=usr/lib/cfgadm/shp.so.1
391 file path=usr/lib/cfgadm/usb.so.1
```

```
file path=usr/lib/extendedFILE.so.1
file path=usr/lib/getloginx.so.1
file path=usr/lib/lib.b mode=0444
file path=usr/lib/libadutils.so.1
file path=usr/lib/libast.so.1
file path=usr/lib/libbsdmalloc.so.1
$(i386_ONLY)file path=usr/lib/libc/libc_hwcap1.so.1 reboot-needed=true
$(i386_ONLY)file path=usr/lib/libc/libc_hwcap2.so.1 reboot-needed=true
$(i386_ONLY)file path=usr/lib/libc/libc_hwcap3.so.1 reboot-needed=true
file path=usr/lib/libcfgadm.so.1
file path=usr/lib/libcmd.so.1
file path=usr/lib/libcommputil.so.1
file path=usr/lib/libcrle.so.1
file path=usr/lib/libcrypt.so.1
file path=usr/lib/libdisasm.so.1
file path=usr/lib/libdll.so.1
file path=usr/lib/libexacct.so.1
file path=usr/lib/libform.so.1
file path=usr/lib/libfstyp.so.1
file path=usr/lib/libhotplug.so.1
file path=usr/lib/libidmap.so.1
file path=usr/lib/libike.so.1
file path=usr/lib/libinetsvc.so.1
file path=usr/lib/libipmi.so.1
file path=usr/lib/libipp.so.1
file path=usr/lib/libipsecutil.so.1
file path=usr/lib/libkvm.so.1
file path=usr/lib/libl.so.1
file path=usr/lib/libldap.so.5
file path=usr/lib/liblgrp.so.1
file path=usr/lib/liblm.so.1
file path=usr/lib/libmail.so.1
file path=usr/lib/libmalloc.so.1
file path=usr/lib/libmapmalloc.so.1
file path=usr/lib/libmenu.so.1
file path=usr/lib/libmtmalloc.so.1
file path=usr/lib/libnls.so.1
file path=usr/lib/libpanel.so.1
file path=usr/lib/libpcidb.so.1
file path=usr/lib/libpkcs11.so.1
file path=usr/lib/libproject.so.1
file path=usr/lib/libraidcfg.so.1
file path=usr/lib/libreparse.so.1
file path=usr/lib/libsched.so.1
file path=usr/lib/libsctp.so.1
file path=usr/lib/libshell.so.1
file path=usr/lib/libsip.so.1
file path=usr/lib/libsldap.so.1
file path=usr/lib/libsmbios.so.1
file path=usr/lib/libsoftcrypto.so.1
file path=usr/lib/libsum.so.1
file path=usr/lib/libsunw_crypto.so.1
file path=usr/lib/libsunw_ssl.so.1
#endif /* ! codereview */
file path=usr/lib/libsys.so.1
$(sparc_ONLY)file path=usr/lib/libv12n.so.1
file path=usr/lib/libvolmgt.so.1
file path=usr/lib/libwrap.so.1.0
file path=usr/lib/liby.so.1
file path=usr/lib/libzoneinfo.so.1
file path=usr/lib/nss_ad.so.1
file path=usr/lib/nss_ldap.so.1
file path=usr/lib/passwdutil.so.1
file path=usr/lib/python2.6/vendor-packages/solaris/__init__.py
file path=usr/lib/python2.6/vendor-packages/solaris/__init__.pyc
file path=usr/lib/python2.6/vendor-packages/solaris/misc.so
```



```
file path=usr/lib/raidcfg/$(ARCH64)/mpt.so.1
file path=usr/lib/raidcfg/mpt.so.1
file path=usr/lib/scsi/$(ARCH64)/libscsi.so.1
file path=usr/lib/scsi/$(ARCH64)/libses.so.1
file path=usr/lib/scsi/$(ARCH64)/libsmp.so.1
file path=usr/lib/scsi/libscsi.so.1
file path=usr/lib/scsi/libses.so.1
file path=usr/lib/scsi/libsmp.so.1
file path=usr/lib/scsi/plugins/scsi/engines/$(ARCH64)/uscsi.so
file path=usr/lib/scsi/plugins/scsi/engines/uscsi.so
file path=usr/lib/scsi/plugins/ses/framework/$(ARCH64)/libses.so
file path=usr/lib/scsi/plugins/ses/framework/$(ARCH64)/ses2.so
file path=usr/lib/scsi/plugins/ses/framework/libses.so
file path=usr/lib/scsi/plugins/ses/framework/ses2.so
file path=usr/lib/scsi/plugins/smp/engine/$(ARCH64)/usmp.so
file path=usr/lib/scsi/plugins/smp/engine/usmp.so
file path=usr/lib/scsi/plugins/smp/framework/$(ARCH64)/sas2.so
file path=usr/lib/scsi/plugins/smp/framework/sas2.so
file path=usr/lib/security/$(ARCH64)/crypt_bsdbf.so.1
file path=usr/lib/security/$(ARCH64)/crypt_bsdmd5.so.1
file path=usr/lib/security/$(ARCH64)/crypt_sha256.so.1
file path=usr/lib/security/$(ARCH64)/crypt_sha512.so.1
file path=usr/lib/security/$(ARCH64)/crypt_sunmd5.so.1
file path=usr/lib/security/$(ARCH64)/pam_allow.so.1
file path=usr/lib/security/$(ARCH64)/pam_authtok_check.so.1
file path=usr/lib/security/$(ARCH64)/pam_authtok_get.so.1
file path=usr/lib/security/$(ARCH64)/pam_authtok_store.so.1
file path=usr/lib/security/$(ARCH64)/pam_deny.so.1
file path=usr/lib/security/$(ARCH64)/pam_dhkeys.so.1
file path=usr/lib/security/$(ARCH64)/pam_dial_auth.so.1
file path=usr/lib/security/$(ARCH64)/pam_ldap.so.1
file path=usr/lib/security/$(ARCH64)/pam_list.so.1
file path=usr/lib/security/$(ARCH64)/pam_passwd_auth.so.1
file path=usr/lib/security/$(ARCH64)/pam_rhosts_auth.so.1
file path=usr/lib/security/$(ARCH64)/pam_roles.so.1
file path=usr/lib/security/$(ARCH64)/pam_sample.so.1
file path=usr/lib/security/$(ARCH64)/pam_tsol_account.so.1
file path=usr/lib/security/$(ARCH64)/pam_unix_account.so.1
file path=usr/lib/security/$(ARCH64)/pam_unix_auth.so.1
file path=usr/lib/security/$(ARCH64)/pam_unix_cred.so.1
file path=usr/lib/security/$(ARCH64)/pam_unix_session.so.1
file path=usr/lib/security/$(ARCH64)/pkcs11_kernel.so.1
file path=usr/lib/security/$(ARCH64)/pkcs11_softtoken.so.1
file path=usr/lib/security/$(ARCH64)/pkcs11_tpm.so.1
file path=usr/lib/security/audit_binfile.so.1
file path=usr/lib/security/audit_remote.so.1
file path=usr/lib/security/audit_syslog.so.1
file path=usr/lib/security/crypt_bsdbf.so.1
file path=usr/lib/security/crypt_bsdmd5.so.1
file path=usr/lib/security/crypt_sha256.so.1
file path=usr/lib/security/crypt_sha512.so.1
file path=usr/lib/security/crypt_sunmd5.so.1
file path=usr/lib/security/pam_allow.so.1
file path=usr/lib/security/pam_authtok_check.so.1
file path=usr/lib/security/pam_authtok_get.so.1
file path=usr/lib/security/pam_authtok_store.so.1
file path=usr/lib/security/pam_deny.so.1
file path=usr/lib/security/pam_dhkeys.so.1
file path=usr/lib/security/pam_dial_auth.so.1
file path=usr/lib/security/pam_ldap.so.1
file path=usr/lib/security/pam_list.so.1
file path=usr/lib/security/pam_passwd_auth.so.1
file path=usr/lib/security/pam_rhosts_auth.so.1
file path=usr/lib/security/pam_roles.so.1
file path=usr/lib/security/pam_sample.so.1
file path=usr/lib/security/pam_tsol_account.so.1
```

```
524 file path=usr/lib/security/pam_unix_account.so.1
525 file path=usr/lib/security/pam_unix_auth.so.1
526 file path=usr/lib/security/pam_unix_cred.so.1
527 file path=usr/lib/security/pam_unix_session.so.1
528 file path=usr/lib/security/pkcs11_kernel.so.1
529 file path=usr/lib/security/pkcs11_softtoken.so.1
530 file path=usr/lib/security/pkcs11_tpm.so.1
531 file path=usr/lib/straddr.so.2
532 file path=usr/lib/watchmalloc.so.1
533 # XXX: Obsoleted by open i18n?
534 file path=usr/xpg4/lib/$(ARCH64)/libcurses.so.1
535 file path=usr/xpg4/lib/$(ARCH64)/libcurses.so.2
536 file path=usr/xpg4/lib/libcurses.so.1
537 file path=usr/xpg4/lib/libcurses.so.2
538 legacy pkg=SUNWcsl \
539     desc="core shared libraries for a specific instruction-set architecture" \
540     name="Core Solaris, (Shared Libs)"
541 legacy pkg=SUNWcslr \
542     desc="core software for a specific instruction-set architecture" \
543     name="Core Solaris Libraries (Root)"
544 license cr_Sun license=cr_Sun
545 license lic_CDDL license=lic_CDDL
546 license lic_OSBL license=lic_OSBL
547 license lic_OSBL_preamble license=lic_OSBL_preamble
548 # libwrap is part of tcp wrappers along with tcpd
549 license usr/src/cmd/tcpd/THIRDPARTYLICENSE \
550     license=usr/src/cmd/tcpd/THIRDPARTYLICENSE
551 license usr/src/common/crypto/THIRDPARTYLICENSE.cryptogams \
552     license=usr/src/common/crypto/THIRDPARTYLICENSE.cryptogams
553 license usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.gladman \
554     license=usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.gladman
555 license usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.openssl \
556     license=usr/src/common/crypto/aes/amd64/THIRDPARTYLICENSE.openssl
557 license usr/src/common/crypto/ecc/THIRDPARTYLICENSE \
558     license=usr/src/common/crypto/ecc/THIRDPARTYLICENSE
559 license usr/src/common/crypto/md5/amd64/THIRDPARTYLICENSE \
560     license=usr/src/common/crypto/md5/amd64/THIRDPARTYLICENSE
561 license usr/src/common/mpi/THIRDPARTYLICENSE \
562     license=usr/src/common/mpi/THIRDPARTYLICENSE
563 license usr/src/lib/libast/THIRDPARTYLICENSE \
564     license=usr/src/lib/libast/THIRDPARTYLICENSE
565 license usr/src/lib/libbsdmalloc/THIRDPARTYLICENSE \
566     license=usr/src/lib/libbsdmalloc/THIRDPARTYLICENSE
567 license usr/src/lib/libc/THIRDPARTYLICENSE \
568     license=usr/src/lib/libc/THIRDPARTYLICENSE
569 license usr/src/lib/libcmd/THIRDPARTYLICENSE \
570     license=usr/src/lib/libcmd/THIRDPARTYLICENSE
571 license usr/src/lib/libdll/THIRDPARTYLICENSE \
572     license=usr/src/lib/libdll/THIRDPARTYLICENSE
573 license usr/src/lib/libinetutil/common/THIRDPARTYLICENSE \
574     license=usr/src/lib/libinetutil/common/THIRDPARTYLICENSE
575 license usr/src/lib/libkmf/THIRDPARTYLICENSE \
576     license=usr/src/lib/libkmf/THIRDPARTYLICENSE
577 license usr/src/lib/libldap5/THIRDPARTYLICENSE \
578     license=usr/src/lib/libldap5/THIRDPARTYLICENSE
579 license usr/src/lib/libmp/common/THIRDPARTYLICENSE \
580     license=usr/src/lib/libmp/common/THIRDPARTYLICENSE
581 license usr/src/lib/libresolv/THIRDPARTYLICENSE \
582     license=usr/src/lib/libresolv/THIRDPARTYLICENSE
583 license usr/src/lib/libresolv2/THIRDPARTYLICENSE \
584     license=usr/src/lib/libresolv2/THIRDPARTYLICENSE
585 license usr/src/lib/libshell/THIRDPARTYLICENSE \
586     license=usr/src/lib/libshell/THIRDPARTYLICENSE
587 license usr/src/lib/libsum/THIRDPARTYLICENSE \
588     license=usr/src/lib/libsum/THIRDPARTYLICENSE
589 license usr/src/lib/pam_modules/authtok_check/THIRDPARTYLICENSE \
```



```
590     license=usr/src/lib/pam_modules/authtok_check/THIRDPARTYLICENSE
591 license usr/src/lib/passwdutil/THIRDPARTYLICENSE \
592     license=usr/src/lib/passwdutil/THIRDPARTYLICENSE
593 license usr/src/lib/pkcs11/pkcs11_tpm/THIRDPARTYLICENSE \
594     license=usr/src/lib/pkcs11/pkcs11_tpm/THIRDPARTYLICENSE
595 license usr/src/uts/common/sys/THIRDPARTYLICENSE.unicode \
596     license=usr/src/uts/common/sys/THIRDPARTYLICENSE.unicode
597 link path=lib/$(ARCH64)/libadm.so target=libadm.so.1
598 link path=lib/$(ARCH64)/libaio.so target=libaio.so.1
599 link path=lib/$(ARCH64)/libbsm.so target=libbsm.so.1
600 link path=lib/$(ARCH64)/libc.so reboot-needed=true target=libc.so.1
601 link path=lib/$(ARCH64)/libc_db.so target=libc_db.so.1
602 link path=lib/$(ARCH64)/libcontract.so target=libcontract.so.1
603 link path=lib/$(ARCH64)/libcryptoutil.so target=libcryptoutil.so.1
604 link path=lib/$(ARCH64)/libctf.so target=libctf.so.1
605 link path=lib/$(ARCH64)/libcurses.so target=libcurses.so.1
606 link path=lib/$(ARCH64)/libdevice.so target=libdevice.so.1
607 link path=lib/$(ARCH64)/libdevid.so target=libdevid.so.1
608 link path=lib/$(ARCH64)/libdevinfo.so target=libdevinfo.so.1
609 link path=lib/$(ARCH64)/libdl.so target=libdl.so.1
610 link path=lib/$(ARCH64)/libdladm.so target=libdladm.so.1
611 link path=lib/$(ARCH64)/libdlpi.so target=libdlpi.so.1
612 link path=lib/$(ARCH64)/libdoor.so target=libdoor.so.1
613 link path=lib/$(ARCH64)/libefi.so target=libefi.so.1
614 link path=lib/$(ARCH64)/libelf.so target=libelf.so.1
615 $(i386_ONLY)link path=lib/$(ARCH64)/libfdisk.so target=libfdisk.so.1
616 link path=lib/$(ARCH64)/libgen.so target=libgen.so.1
617 link path=lib/$(ARCH64)/libintl.so target=libintl.so.1
618 link path=lib/$(ARCH64)/libkmf.so target=libkmf.so.1
619 link path=lib/$(ARCH64)/libkmfberder.so target=libkmfberder.so.1
620 link path=lib/$(ARCH64)/libkstat.so target=libkstat.so.1
621 link path=lib/$(ARCH64)/libmd.so target=libmd.so.1
622 link path=lib/$(ARCH64)/libmd5.so target=libmd5.so.1
623 link path=lib/$(ARCH64)/libmp.so target=libmp.so.2
624 link path=lib/$(ARCH64)/libnsl.so target=libnsl.so.1
625 link path=lib/$(ARCH64)/libnvpair.so target=libnvpair.so.1
626 link path=lib/$(ARCH64)/libpam.so target=libpam.so.1
627 link path=lib/$(ARCH64)/libposix4.so target=libposix4.so.1
628 link path=lib/$(ARCH64)/libposix4.so.1 target=librt.so.1
629 link path=lib/$(ARCH64)/libproc.so target=libproc.so.1
630 link path=lib/$(ARCH64)/libpthread.so target=libpthread.so.1
631 link path=lib/$(ARCH64)/librcm.so target=librcm.so.1
632 link path=lib/$(ARCH64)/libresolv.so target=libresolv.so.2
633 link path=lib/$(ARCH64)/librestart.so target=librestart.so.1
634 link path=lib/$(ARCH64)/librpcsvc.so target=librpcsvc.so.1
635 link path=lib/$(ARCH64)/librt.so target=librt.so.1
636 link path=lib/$(ARCH64)/librtld_db.so target=librtld_db.so.1
637 link path=lib/$(ARCH64)/libscf.so target=libscf.so.1
638 link path=lib/$(ARCH64)/libsec.so target=libsec.so.1
639 link path=lib/$(ARCH64)/libsecdb.so target=libsecdb.so.1
640 link path=lib/$(ARCH64)/libsendfile.so target=libsendfile.so.1
641 link path=lib/$(ARCH64)/libsocket.so target=libsocket.so.1
642 link path=lib/$(ARCH64)/libsysevent.so target=libsysevent.so.1
643 link path=lib/$(ARCH64)/libtermcap.so target=libtermcap.so.1
644 link path=lib/$(ARCH64)/libtermlib.so target=libtermlib.so.1
645 link path=lib/$(ARCH64)/libtermlib.so.1 target=libcurses.so.1
646 link path=lib/$(ARCH64)/libthread.so target=libthread.so.1
647 link path=lib/$(ARCH64)/libthread_db.so target=libc_db.so.1
648 link path=lib/$(ARCH64)/libthread_db.so.1 target=libc_db.so.1
649 link path=lib/$(ARCH64)/libtsnet.so target=libtsnet.so.1
650 link path=lib/$(ARCH64)/libtsol.so target=libtsol.so.2
651 link path=lib/$(ARCH64)/libumem.so target=libumem.so.1
652 link path=lib/$(ARCH64)/libuuid.so target=libuuid.so.1
653 link path=lib/$(ARCH64)/libuutil.so target=libuutil.so.1
654 link path=lib/$(ARCH64)/libw.so target=libw.so.1
655 link path=lib/$(ARCH64)/libxnet.so target=libxnet.so.1
```

```
656 link path=lib/32 target=.
657 link path=lib/64 target=$(ARCH64)
658 link path=lib/crypto/32 target=.
659 link path=lib/crypto/64 target=$(ARCH64)
660 link path=lib/libadm.so target=libadm.so.1
661 link path=lib/libaio.so target=libaio.so.1
662 link path=lib/libbsm.so target=libbsm.so.1
663 link path=lib/libc.so target=libc.so.1
664 link path=lib/libc_db.so target=libc_db.so.1
665 link path=lib/libcontract.so target=libcontract.so.1
666 link path=lib/libcryptoutil.so target=./libcryptoutil.so.1
667 link path=lib/libctf.so target=libctf.so.1
668 link path=lib/libcurses.so target=libcurses.so.1
669 link path=lib/libdevice.so target=libdevice.so.1
670 link path=lib/libdevid.so target=libdevid.so.1
671 link path=lib/libdevinfo.so target=libdevinfo.so.1
672 link path=lib/libdl.so target=libdl.so.1
673 link path=lib/libdladm.so target=libdladm.so.1
674 link path=lib/libdlpi.so target=libdlpi.so.1
675 link path=lib/libdoor.so target=libdoor.so.1
676 link path=lib/libefi.so target=libefi.so.1
677 link path=lib/libelf.so target=libelf.so.1
678 link path=lib/libelfsign.so target=libelfsign.so.1
679 $(i386_ONLY)link path=lib/libfdisk.so target=libfdisk.so.1
680 link path=lib/libgen.so target=libgen.so.1
681 link path=lib/libintl.so target=libintl.so.1
682 link path=lib/libipmp.so target=./libipmp.so.1
683 link path=lib/libkmf.so target=libkmf.so.1
684 link path=lib/libkmfberder.so target=libkmfberder.so.1
685 link path=lib/libkstat.so target=libkstat.so.1
686 link path=lib/libmd.so target=libmd.so.1
687 link path=lib/libmd5.so target=libmd5.so.1
688 link path=lib/libmp.so target=libmp.so.2
689 link path=lib/libnsl.so target=libnsl.so.1
690 link path=lib/libnvpair.so target=libnvpair.so.1
691 link path=lib/libnwam.so target=libnwam.so.1
692 link path=lib/libpam.so target=libpam.so.1
693 link path=lib/libposix4.so target=libposix4.so.1
694 link path=lib/libposix4.so.1 target=librt.so.1
695 link path=lib/libproc.so target=libproc.so.1
696 link path=lib/libpthread.so target=libpthread.so.1
697 link path=lib/librcm.so target=./librcm.so.1
698 link path=lib/libresolv.so target=libresolv.so.2
699 link path=lib/librpcsvc.so target=librpcsvc.so.1
700 link path=lib/librt.so target=librt.so.1
701 link path=lib/librtld_db.so target=librtld_db.so.1
702 link path=lib/libscf.so target=libscf.so.1
703 link path=lib/libsec.so target=libsec.so.1
704 link path=lib/libsecdb.so target=libsecdb.so.1
705 link path=lib/libsendfile.so target=libsendfile.so.1
706 link path=lib/libsocket.so target=libsocket.so.1
707 link path=lib/libsysevent.so target=./libsysevent.so.1
708 link path=lib/libtermcap.so target=libtermcap.so.1
709 link path=lib/libtermlib.so target=libtermlib.so.1
710 link path=lib/libtermlib.so.1 target=libcurses.so.1
711 link path=lib/libthread.so target=libthread.so.1
712 link path=lib/libthread_db.so target=libc_db.so.1
713 link path=lib/libthread_db.so.1 target=libc_db.so.1
714 link path=lib/libtsol.so target=libtsol.so.2
715 link path=lib/libumem.so target=libumem.so.1
716 link path=lib/libuuid.so target=libuuid.so.1
717 link path=lib/libw.so target=libw.so.1
718 link path=lib/libxnet.so target=libxnet.so.1
719 link path=lib/secure/32 target=.
720 link path=lib/secure/64 target=$(ARCH64)
721 link path=usr/ccs/lib/$(ARCH64)/libcurses.so \
```



```
722     target=../../../../lib/$(ARCH64)/libcurses.so.1
723 link path=usr/ccs/lib/$(ARCH64)/libform.so \
724     target=../../../lib/$(ARCH64)/libform.so.1
725 link path=usr/ccs/lib/$(ARCH64)/libgen.so \
726     target=../../../../lib/$(ARCH64)/libgen.so.1
727 link path=usr/ccs/lib/$(ARCH64)/libl.so \
728     target=../../../lib/$(ARCH64)/libl.so.1
729 link path=usr/ccs/lib/$(ARCH64)/libmalloc.so \
730     target=../../../lib/$(ARCH64)/libmalloc.so.1
731 link path=usr/ccs/lib/$(ARCH64)/libmenu.so \
732     target=../../../lib/$(ARCH64)/libmenu.so.1
733 link path=usr/ccs/lib/$(ARCH64)/libpanel.so \
734     target=../../../lib/$(ARCH64)/libpanel.so.1
735 link path=usr/ccs/lib/$(ARCH64)/libtermcap.so \
736     target=../../../../lib/$(ARCH64)/libtermcap.so.1
737 link path=usr/ccs/lib/$(ARCH64)/libtermlib.so \
738     target=../../../../lib/$(ARCH64)/libcurses.so.1
739 link path=usr/ccs/lib/$(ARCH64)/liby.so \
740     target=../../../lib/$(ARCH64)/liby.so.1
741 link path=usr/ccs/lib/libcurses.so target=../../../lib/libcurses.so.1
742 link path=usr/ccs/lib/libform.so target=../../lib/libform.so.1
743 link path=usr/ccs/lib/libgen.so target=../../../lib/libgen.so.1
744 link path=usr/ccs/lib/libl.so target=../../lib/libl.so.1
745 link path=usr/ccs/lib/libmalloc.so target=../../lib/libmalloc.so.1
746 link path=usr/ccs/lib/libmenu.so target=../../lib/libmenu.so.1
747 link path=usr/ccs/lib/libpanel.so target=../../lib/libpanel.so.1
748 link path=usr/ccs/lib/libtermcap.so target=../../../lib/libtermcap.so.1
749 link path=usr/ccs/lib/libtermlib.so target=../../../lib/libcurses.so.1
750 link path=usr/ccs/lib/liby.so target=../../lib/liby.so.1
751 link path=usr/lib/$(ARCH64)/libadm.so \
752     target=../../../lib/$(ARCH64)/libadm.so.1
753 link path=usr/lib/$(ARCH64)/libadm.so.1 \
754     target=../../../lib/$(ARCH64)/libadm.so.1
755 link path=usr/lib/$(ARCH64)/libadutils.so target=./libadutils.so.1
756 link path=usr/lib/$(ARCH64)/libaio.so \
757     target=../../../lib/$(ARCH64)/libaio.so.1
758 link path=usr/lib/$(ARCH64)/libaio.so.1 \
759     target=../../../lib/$(ARCH64)/libaio.so.1
760 link path=usr/lib/$(ARCH64)/libavl.so.1 \
761     target=../../../lib/$(ARCH64)/libavl.so.1
762 link path=usr/lib/$(ARCH64)/libbsdmalloc.so target=libbsdmalloc.so.1
763 link path=usr/lib/$(ARCH64)/libbsm.so \
764     target=../../../lib/$(ARCH64)/libbsm.so.1
765 link path=usr/lib/$(ARCH64)/libbsm.so.1 \
766     target=../../../lib/$(ARCH64)/libbsm.so.1
767 link path=usr/lib/$(ARCH64)/libc.so target=../../../lib/$(ARCH64)/libc.so.1
768 link path=usr/lib/$(ARCH64)/libc.so.1 target=../../../lib/$(ARCH64)/libc.so.1
769 link path=usr/lib/$(ARCH64)/libc_db.so \
770     target=../../../lib/$(ARCH64)/libc_db.so.1
771 link path=usr/lib/$(ARCH64)/libc_db.so.1 \
772     target=../../../lib/$(ARCH64)/libc_db.so.1
773 link path=usr/lib/$(ARCH64)/libcfgadm.so target=libcfgadm.so.1
774 link path=usr/lib/$(ARCH64)/libcmd.so target=libcmd.so.1
775 link path=usr/lib/$(ARCH64)/libcmdutils.so.1 \
776     target=../../../lib/$(ARCH64)/libcmdutils.so.1
777 link path=usr/lib/$(ARCH64)/libcommputil.so target=libcommputil.so.1
778 link path=usr/lib/$(ARCH64)/libcontract.so \
779     target=../../../lib/$(ARCH64)/libcontract.so.1
780 link path=usr/lib/$(ARCH64)/libcontract.so.1 \
781     target=../../../lib/$(ARCH64)/libcontract.so.1
782 link path=usr/lib/$(ARCH64)/libcrypt.so target=./libcrypt.so.1
783 link path=usr/lib/$(ARCH64)/libcrypt_d.so target=./libcrypt.so
784 link path=usr/lib/$(ARCH64)/libcrypt_d.so.1 target=./libcrypt.so.1
785 link path=usr/lib/$(ARCH64)/libcrypt_i.so target=./libcrypt.so
786 link path=usr/lib/$(ARCH64)/libcrypt_i.so.1 target=./libcrypt.so.1
787 link path=usr/lib/$(ARCH64)/libctf.so \
```

```
788     target=../../../lib/$(ARCH64)/libctf.so.1
789 link path=usr/lib/$(ARCH64)/libctf.so.1 \
790     target=../../../lib/$(ARCH64)/libctf.so.1
791 link path=usr/lib/$(ARCH64)/libcurses.so \
792     target=../../../lib/$(ARCH64)/libcurses.so.1
793 link path=usr/lib/$(ARCH64)/libcurses.so.1 \
794     target=../../../lib/$(ARCH64)/libcurses.so.1
795 link path=usr/lib/$(ARCH64)/libdevice.so \
796     target=../../../lib/$(ARCH64)/libdevice.so.1
797 link path=usr/lib/$(ARCH64)/libdevice.so.1 \
798     target=../../../lib/$(ARCH64)/libdevice.so.1
799 link path=usr/lib/$(ARCH64)/libdevid.so \
800     target=../../../lib/$(ARCH64)/libdevid.so.1
801 link path=usr/lib/$(ARCH64)/libdevid.so.1 \
802     target=../../../lib/$(ARCH64)/libdevid.so.1
803 link path=usr/lib/$(ARCH64)/libdevinfo.so \
804     target=../../../lib/$(ARCH64)/libdevinfo.so.1
805 link path=usr/lib/$(ARCH64)/libdevinfo.so.1 \
806     target=../../../lib/$(ARCH64)/libdevinfo.so.1
807 link path=usr/lib/$(ARCH64)/libdhcputil.so.1 \
808     target=../../../lib/$(ARCH64)/libdhcputil.so.1
809 link path=usr/lib/$(ARCH64)/libdisasm.so target=libdisasm.so.1
810 link path=usr/lib/$(ARCH64)/libdl.so target=../../../lib/$(ARCH64)/libdl.so.1
811 link path=usr/lib/$(ARCH64)/libdl.so.1 \
812     target=../../../lib/$(ARCH64)/libdl.so.1
813 link path=usr/lib/$(ARCH64)/libdlpi.so \
814     target=../../../lib/$(ARCH64)/libdlpi.so.1
815 link path=usr/lib/$(ARCH64)/libdlpi.so.1 \
816     target=../../../lib/$(ARCH64)/libdlpi.so.1
817 link path=usr/lib/$(ARCH64)/libdoor.so \
818     target=../../../lib/$(ARCH64)/libdoor.so.1
819 link path=usr/lib/$(ARCH64)/libdoor.so.1 \
820     target=../../../lib/$(ARCH64)/libdoor.so.1
821 link path=usr/lib/$(ARCH64)/libefi.so \
822     target=../../../lib/$(ARCH64)/libefi.so.1
823 link path=usr/lib/$(ARCH64)/libefi.so.1 \
824     target=../../../lib/$(ARCH64)/libefi.so.1
825 link path=usr/lib/$(ARCH64)/libelf.so \
826     target=../../../lib/$(ARCH64)/libelf.so.1
827 link path=usr/lib/$(ARCH64)/libelf.so.1 \
828     target=../../../lib/$(ARCH64)/libelf.so.1
829 link path=usr/lib/$(ARCH64)/libexacct.so target=libexacct.so.1
830 $(i386_ONLY)link path=usr/lib/$(ARCH64)/libfdisk.so \
831     target=../../../lib/$(ARCH64)/libfdisk.so.1
832 $(i386_ONLY)link path=usr/lib/$(ARCH64)/libfdisk.so.1 \
833     target=../../../lib/$(ARCH64)/libfdisk.so.1
834 link path=usr/lib/$(ARCH64)/libform.so target=libform.so.1
835 link path=usr/lib/$(ARCH64)/libfstyp.so target=./libfstyp.so.1
836 link path=usr/lib/$(ARCH64)/libgen.so \
837     target=../../../lib/$(ARCH64)/libgen.so.1
838 link path=usr/lib/$(ARCH64)/libgen.so.1 \
839     target=../../../lib/$(ARCH64)/libgen.so.1
840 link path=usr/lib/$(ARCH64)/libhotplug.so target=libhotplug.so.1
841 link path=usr/lib/$(ARCH64)/libidmap.so target=./libidmap.so.1
842 link path=usr/lib/$(ARCH64)/libinetutil.so.1 \
843     target=../../../lib/$(ARCH64)/libinetutil.so.1
844 link path=usr/lib/$(ARCH64)/libintl.so \
845     target=../../../lib/$(ARCH64)/libintl.so.1
846 link path=usr/lib/$(ARCH64)/libintl.so.1 \
847     target=../../../lib/$(ARCH64)/libintl.so.1
848 link path=usr/lib/$(ARCH64)/libipmi.so target=./libipmi.so.1
849 link path=usr/lib/$(ARCH64)/libipp.so target=libipp.so.1
850 link path=usr/lib/$(ARCH64)/libkstat.so \
851     target=../../../lib/$(ARCH64)/libkstat.so.1
852 link path=usr/lib/$(ARCH64)/libkstat.so.1 \
853     target=../../../lib/$(ARCH64)/libkstat.so.1
```



```
854 link path=usr/lib/$(ARCH64)/libkvm.so target=libkvm.so.1
855 link path=usr/lib/$(ARCH64)/libl.so target=libl.so.1
856 link path=usr/lib/$(ARCH64)/libldap.so target=libldap.so.5
857 link path=usr/lib/$(ARCH64)/liblddbg.so.4 \
858     target=../../../lib/$(ARCH64)/liblddbg.so.4
859 link path=usr/lib/$(ARCH64)/liblgrp.so target=liblgrp.so.1
860 link path=usr/lib/$(ARCH64)/liblm.so target=liblm.so.1
861 link path=usr/lib/$(ARCH64)/libmail.so target=libmail.so.1
862 link path=usr/lib/$(ARCH64)/libmalloc.so target=libmalloc.so.1
863 link path=usr/lib/$(ARCH64)/libmapmalloc.so target=libmapmalloc.so.1
864 link path=usr/lib/$(ARCH64)/libmd.so target=../../../lib/$(ARCH64)/libmd.so.1
865 link path=usr/lib/$(ARCH64)/libmd.so.1 \
866     target=../../../lib/$(ARCH64)/libmd.so.1
867 link path=usr/lib/$(ARCH64)/libmd5.so \
868     target=../../../lib/$(ARCH64)/libmd5.so.1
869 link path=usr/lib/$(ARCH64)/libmd5.so.1 \
870     target=../../../lib/$(ARCH64)/libmd5.so.1
871 link path=usr/lib/$(ARCH64)/libmenu.so target=libmenu.so.1
872 link path=usr/lib/$(ARCH64)/libmp.so target=../../../lib/$(ARCH64)/libmp.so.2
873 link path=usr/lib/$(ARCH64)/libmp.so.2 \
874     target=../../../lib/$(ARCH64)/libmp.so.2
875 link path=usr/lib/$(ARCH64)/libmtmalloc.so target=libmtmalloc.so.1
876 link path=usr/lib/$(ARCH64)/libnls.so target=libnls.so.1
877 link path=usr/lib/$(ARCH64)/libnsl.so \
878     target=../../../lib/$(ARCH64)/libnsl.so.1
879 link path=usr/lib/$(ARCH64)/libnsl.so.1 \
880     target=../../../lib/$(ARCH64)/libnsl.so.1
881 link path=usr/lib/$(ARCH64)/libnvpair.so \
882     target=../../../lib/$(ARCH64)/libnvpair.so.1
883 link path=usr/lib/$(ARCH64)/libnvpair.so.1 \
884     target=../../../lib/$(ARCH64)/libnvpair.so.1
885 link path=usr/lib/$(ARCH64)/libpam.so \
886     target=../../../lib/$(ARCH64)/libpam.so.1
887 link path=usr/lib/$(ARCH64)/libpam.so.1 \
888     target=../../../lib/$(ARCH64)/libpam.so.1
889 link path=usr/lib/$(ARCH64)/libpanel.so target=libpanel.so.1
890 link path=usr/lib/$(ARCH64)/libpkcs11.so target=libpkcs11.so.1
891 link path=usr/lib/$(ARCH64)/libposix4.so \
892     target=../../../lib/$(ARCH64)/librt.so.1
893 link path=usr/lib/$(ARCH64)/libposix4.so.1 \
894     target=../../../lib/$(ARCH64)/librt.so.1
895 link path=usr/lib/$(ARCH64)/libproc.so \
896     target=../../../lib/$(ARCH64)/libproc.so.1
897 link path=usr/lib/$(ARCH64)/libproc.so.1 \
898     target=../../../lib/$(ARCH64)/libproc.so.1
899 link path=usr/lib/$(ARCH64)/libproject.so target=libproject.so.1
900 link path=usr/lib/$(ARCH64)/libpthread.so \
901     target=../../../lib/$(ARCH64)/libpthread.so.1
902 link path=usr/lib/$(ARCH64)/libpthread.so.1 \
903     target=../../../lib/$(ARCH64)/libpthread.so.1
904 link path=usr/lib/$(ARCH64)/librcm.so \
905     target=../../../lib/$(ARCH64)/librcm.so.1
906 link path=usr/lib/$(ARCH64)/librcm.so.1 \
907     target=../../../lib/$(ARCH64)/librcm.so.1
908 link path=usr/lib/$(ARCH64)/libreparse.so target=libreparse.so.1
909 link path=usr/lib/$(ARCH64)/libresolv.so \
910     target=../../../lib/$(ARCH64)/libresolv.so.2
911 link path=usr/lib/$(ARCH64)/libresolv.so.2 \
912     target=../../../lib/$(ARCH64)/libresolv.so.2
913 $(i386_ONLY)link path=usr/lib/$(ARCH64)/librestart.so \
914     target=../../../lib/$(ARCH64)/librestart.so.1
915 link path=usr/lib/$(ARCH64)/librestart.so.1 \
916     target=../../../lib/$(ARCH64)/librestart.so.1
917 link path=usr/lib/$(ARCH64)/librpcsvc.so \
918     target=../../../lib/$(ARCH64)/librpcsvc.so.1
919 link path=usr/lib/$(ARCH64)/librpcsvc.so.1 \
```

```
920     target=../../../lib/$(ARCH64)/librpcsvc.so.1
921 link path=usr/lib/$(ARCH64)/librt.so target=../../../lib/$(ARCH64)/librt.so.1
922 link path=usr/lib/$(ARCH64)/librt.so.1 \
923     target=../../../lib/$(ARCH64)/librt.so.1
924 link path=usr/lib/$(ARCH64)/librtld.so.1 \
925     target=../../../lib/$(ARCH64)/librtld.so.1
926 link path=usr/lib/$(ARCH64)/librtld_db.so \
927     target=../../../lib/$(ARCH64)/librtld_db.so.1
928 link path=usr/lib/$(ARCH64)/librtld_db.so.1 \
929     target=../../../lib/$(ARCH64)/librtld_db.so.1
930 link path=usr/lib/$(ARCH64)/libscf.so \
931     target=../../../lib/$(ARCH64)/libscf.so.1
932 link path=usr/lib/$(ARCH64)/libscf.so.1 \
933     target=../../../lib/$(ARCH64)/libscf.so.1
934 link path=usr/lib/$(ARCH64)/libsched.so target=libsched.so.1
935 link path=usr/lib/$(ARCH64)/libsctp.so target=./libsctp.so.1
936 link path=usr/lib/$(ARCH64)/libsec.so \
937     target=../../../lib/$(ARCH64)/libsec.so.1
938 link path=usr/lib/$(ARCH64)/libsec.so.1 \
939     target=../../../lib/$(ARCH64)/libsec.so.1
940 link path=usr/lib/$(ARCH64)/libsecdb.so \
941     target=../../../lib/$(ARCH64)/libsecdb.so.1
942 link path=usr/lib/$(ARCH64)/libsecdb.so.1 \
943     target=../../../lib/$(ARCH64)/libsecdb.so.1
944 link path=usr/lib/$(ARCH64)/libsendfile.so \
945     target=../../../lib/$(ARCH64)/libsendfile.so.1
946 link path=usr/lib/$(ARCH64)/libsendfile.so.1 \
947     target=../../../lib/$(ARCH64)/libsendfile.so.1
948 link path=usr/lib/$(ARCH64)/libsip.so target=./libsip.so.1
949 link path=usr/lib/$(ARCH64)/libsldap.so target=libsldap.so.1
950 link path=usr/lib/$(ARCH64)/libsmbios.so target=libsmbios.so.1
951 link path=usr/lib/$(ARCH64)/libsocket.so \
952     target=../../../lib/$(ARCH64)/libsocket.so.1
953 link path=usr/lib/$(ARCH64)/libsocket.so.1 \
954     target=../../../lib/$(ARCH64)/libsocket.so.1
955 link path=usr/lib/$(ARCH64)/libsoftcrypto.so target=./libsoftcrypto.so.1
956 link path=usr/lib/$(ARCH64)/libsysevent.so \
957     target=../../../lib/$(ARCH64)/libsysevent.so.1
958 link path=usr/lib/$(ARCH64)/libsysevent.so.1 \
959     target=../../../lib/$(ARCH64)/libsysevent.so.1
960 link path=usr/lib/$(ARCH64)/libtermcap.so \
961     target=../../../lib/$(ARCH64)/libtermcap.so.1
962 link path=usr/lib/$(ARCH64)/libtermcap.so.1 \
963     target=../../../lib/$(ARCH64)/libtermcap.so.1
964 link path=usr/lib/$(ARCH64)/libtermlib.so \
965     target=../../../lib/$(ARCH64)/libcurses.so.1
966 link path=usr/lib/$(ARCH64)/libtermlib.so.1 \
967     target=../../../lib/$(ARCH64)/libcurses.so.1
968 link path=usr/lib/$(ARCH64)/libthread.so \
969     target=../../../lib/$(ARCH64)/libthread.so.1
970 link path=usr/lib/$(ARCH64)/libthread.so.1 \
971     target=../../../lib/$(ARCH64)/libthread.so.1
972 link path=usr/lib/$(ARCH64)/libthread_db.so \
973     target=../../../lib/$(ARCH64)/libc_db.so.1
974 link path=usr/lib/$(ARCH64)/libthread_db.so.1 \
975     target=../../../lib/$(ARCH64)/libc_db.so.1
976 link path=usr/lib/$(ARCH64)/libtsnet.so \
977     target=../../../lib/$(ARCH64)/libtsnet.so.1
978 link path=usr/lib/$(ARCH64)/libtsnet.so.1 \
979     target=../../../lib/$(ARCH64)/libtsnet.so.1
980 link path=usr/lib/$(ARCH64)/libtsol.so \
981     target=../../../lib/$(ARCH64)/libtsol.so.2
982 link path=usr/lib/$(ARCH64)/libtsol.so.2 \
983     target=../../../lib/$(ARCH64)/libtsol.so.2
984 link path=usr/lib/$(ARCH64)/libumem.so \
985     target=../../../lib/$(ARCH64)/libumem.so.1
```



```
986 link path=usr/lib/$(ARCH64)/libumem.so.1 \
987     target=../../../lib/$(ARCH64)/libumem.so.1
988 link path=usr/lib/$(ARCH64)/libuuid.so \
989     target=../../../lib/$(ARCH64)/libuuid.so.1
990 link path=usr/lib/$(ARCH64)/libuuid.so.1 \
991     target=../../../lib/$(ARCH64)/libuuid.so.1
992 $(i386_ONLY)link path=usr/lib/$(ARCH64)/libuutil.so \
993     target=../../../lib/$(ARCH64)/libuutil.so.1
994 link path=usr/lib/$(ARCH64)/libuutil.so.1 \
995     target=../../../lib/$(ARCH64)/libuutil.so.1
996 $(sparc_ONLY)link path=usr/lib/$(ARCH64)/libv12n.so target=./libv12n.so.1
997 link path=usr/lib/$(ARCH64)/libvolmgt.so target=libvolmgt.so.1
998 link path=usr/lib/$(ARCH64)/libw.so target=../../../lib/$(ARCH64)/libw.so.1
999 link path=usr/lib/$(ARCH64)/libw.so.1 target=../../../lib/$(ARCH64)/libw.so.1
1000 link path=usr/lib/$(ARCH64)/libwrap.so target=libwrap.so.1.0
1001 link path=usr/lib/$(ARCH64)/libwrap.so.1 target=libwrap.so.1.0
1002 link path=usr/lib/$(ARCH64)/libxnet.so \
1003     target=../../../lib/$(ARCH64)/libxnet.so.1
1004 link path=usr/lib/$(ARCH64)/libxnet.so.1 \
1005     target=../../../lib/$(ARCH64)/libxnet.so.1
1006 link path=usr/lib/$(ARCH64)/liby.so target=liby.so.1
1007 link path=usr/lib/$(ARCH64)/libzoneinfo.so target=./libzoneinfo.so.1
1008 link path=usr/lib/$(ARCH64)/nss_compat.so.1 \
1009     target=../../../lib/$(ARCH64)/nss_compat.so.1
1010 link path=usr/lib/$(ARCH64)/nss_dns.so.1 \
1011     target=../../../lib/$(ARCH64)/nss_dns.so.1
1012 link path=usr/lib/$(ARCH64)/nss_files.so.1 \
1013     target=../../../lib/$(ARCH64)/nss_files.so.1
1014 link path=usr/lib/$(ARCH64)/nss_nis.so.1 \
1015     target=../../../lib/$(ARCH64)/nss_nis.so.1
1016 link path=usr/lib/$(ARCH64)/nss_user.so.1 \
1017     target=../../../lib/$(ARCH64)/nss_user.so.1
1018 link path=usr/lib/$(ARCH64)/straddr.so target=straddr.so.2
1019 link path=usr/lib/32 target=.
1020 link path=usr/lib/64 target=$(ARCH64)
1021 link path=usr/lib/cfgadm/$(ARCH64)/ib.so target=./ib.so.1
1022 link path=usr/lib/cfgadm/$(ARCH64)/pci.so target=./pci.so.1
1023 $(i386_ONLY)link path=usr/lib/cfgadm/$(ARCH64)/sata.so target=./sata.so.1
1024 link path=usr/lib/cfgadm/$(ARCH64)/scsi.so target=./scsi.so.1
1025 link path=usr/lib/cfgadm/$(ARCH64)/shp.so target=./shp.so.1
1026 link path=usr/lib/cfgadm/$(ARCH64)/usb.so target=./usb.so.1
1027 link path=usr/lib/cfgadm/ib.so target=./ib.so.1
1028 link path=usr/lib/cfgadm/pci.so target=./pci.so.1
1029 $(i386_ONLY)link path=usr/lib/cfgadm/sata.so target=./sata.so.1
1030 link path=usr/lib/cfgadm/scsi.so target=./scsi.so.1
1031 link path=usr/lib/cfgadm/shp.so target=./shp.so.1
1032 link path=usr/lib/cfgadm/usb.so target=./usb.so.1
1033 link path=usr/lib/libadm.so target=../../lib/libadm.so.1
1034 link path=usr/lib/libadm.so.1 target=../../lib/libadm.so.1
1035 link path=usr/lib/libadutils.so target=./libadutils.so.1
1036 link path=usr/lib/libaio.so target=../../lib/libaio.so.1
1037 link path=usr/lib/libaio.so.1 target=../../lib/libaio.so.1
1038 link path=usr/lib/libavl.so.1 target=../../lib/libavl.so.1
1039 link path=usr/lib/libbsdmalloc.so target=./libbsdmalloc.so.1
1040 link path=usr/lib/libbsm.so target=../../lib/libbsm.so.1
1041 link path=usr/lib/libbsm.so.1 target=../../lib/libbsm.so.1
1042 link path=usr/lib/libc.so target=../../lib/libc.so.1
1043 link path=usr/lib/libc.so.1 target=../../lib/libc.so.1
1044 link path=usr/lib/libc_db.so target=../../lib/libc_db.so.1
1045 link path=usr/lib/libc_db.so.1 target=../../lib/libc_db.so.1
1046 link path=usr/lib/libcfgadm.so target=./libcfgadm.so.1
1047 link path=usr/lib/libcmd.so target=libcmd.so.1
1048 link path=usr/lib/libcmdutils.so.1 target=../../lib/libcmdutils.so.1
1049 link path=usr/lib/libcommputil.so target=./libcommputil.so.1
1050 link path=usr/lib/libcontract.so target=../../lib/libcontract.so.1
1051 link path=usr/lib/libcontract.so.1 target=../../lib/libcontract.so.1
```

```
1052 link path=usr/lib/libcrypt.so target=./libcrypt.so.1
1053 link path=usr/lib/libcrypt_d.so target=./libcrypt.so
1054 link path=usr/lib/libcrypt_d.so.1 target=./libcrypt.so.1
1055 link path=usr/lib/libcrypt_i.so target=./libcrypt.so
1056 link path=usr/lib/libcrypt_i.so.1 target=./libcrypt.so.1
1057 link path=usr/lib/libctf.so target=../../lib/libctf.so.1
1058 link path=usr/lib/libctf.so.1 target=../../lib/libctf.so.1
1059 link path=usr/lib/libcurses.so target=../../lib/libcurses.so.1
1060 link path=usr/lib/libcurses.so.1 target=../../lib/libcurses.so.1
1061 link path=usr/lib/libdevice.so target=../../lib/libdevice.so.1
1062 link path=usr/lib/libdevice.so.1 target=../../lib/libdevice.so.1
1063 link path=usr/lib/libdevid.so target=../../lib/libdevid.so.1
1064 link path=usr/lib/libdevid.so.1 target=../../lib/libdevid.so.1
1065 link path=usr/lib/libdevinfo.so target=../../lib/libdevinfo.so.1
1066 link path=usr/lib/libdevinfo.so.1 target=../../lib/libdevinfo.so.1
1067 link path=usr/lib/libdhcpagent.so.1 target=../../lib/libdhcpagent.so.1
1068 link path=usr/lib/libdhcputil.so.1 target=../../lib/libdhcputil.so.1
1069 link path=usr/lib/libdisasm.so target=./libdisasm.so.1
1070 link path=usr/lib/libdl.so target=../../lib/libdl.so.1
1071 link path=usr/lib/libdl.so.1 target=../../lib/libdl.so.1
1072 link path=usr/lib/libdlpi.so target=../../lib/libdlpi.so.1
1073 link path=usr/lib/libdlpi.so.1 target=../../lib/libdlpi.so.1
1074 link path=usr/lib/libdoor.so target=../../lib/libdoor.so.1
1075 link path=usr/lib/libdoor.so.1 target=../../lib/libdoor.so.1
1076 link path=usr/lib/libefi.so target=../../lib/libefi.so.1
1077 link path=usr/lib/libefi.so.1 target=../../lib/libefi.so.1
1078 link path=usr/lib/libelf.so target=../../lib/libelf.so.1
1079 link path=usr/lib/libelf.so.1 target=../../lib/libelf.so.1
1080 link path=usr/lib/libexacct.so target=./libexacct.so.1
1081 $(i386_ONLY)link path=usr/lib/libfdisk.so target=../../lib/libfdisk.so.1
1082 $(i386_ONLY)link path=usr/lib/libfdisk.so.1 target=../../lib/libfdisk.so.1
1083 link path=usr/lib/libform.so target=./libform.so.1
1084 link path=usr/lib/libfstyp.so target=./libfstyp.so.1
1085 link path=usr/lib/libgen.so target=../../lib/libgen.so.1
1086 link path=usr/lib/libgen.so.1 target=../../lib/libgen.so.1
1087 link path=usr/lib/libhotplug.so target=./libhotplug.so.1
1088 link path=usr/lib/libidmap.so target=./libidmap.so.1
1089 link path=usr/lib/libinetutil.so.1 target=../../lib/libinetutil.so.1
1090 link path=usr/lib/libintl.so target=../../lib/libintl.so.1
1091 link path=usr/lib/libintl.so.1 target=../../lib/libintl.so.1
1092 link path=usr/lib/libipmi.so target=./libipmi.so.1
1093 link path=usr/lib/libipp.so target=./libipp.so.1
1094 link path=usr/lib/libkstat.so target=../../lib/libkstat.so.1
1095 link path=usr/lib/libkstat.so.1 target=../../lib/libkstat.so.1
1096 link path=usr/lib/libkvm.so target=./libkvm.so.1
1097 link path=usr/lib/libl.so target=./libl.so.1
1098 link path=usr/lib/libldap.so target=libldap.so.5
1099 link path=usr/lib/liblddbg.so.4 target=../../lib/liblddbg.so.4
1100 link path=usr/lib/liblgrp.so target=./liblgrp.so.1
1101 link path=usr/lib/liblm.so target=./liblm.so.1
1102 link path=usr/lib/libmail.so target=./libmail.so.1
1103 link path=usr/lib/libmalloc.so target=./libmalloc.so.1
1104 link path=usr/lib/libmapmalloc.so target=./libmapmalloc.so.1
1105 link path=usr/lib/libmd.so target=../../lib/libmd.so.1
1106 link path=usr/lib/libmd.so.1 target=../../lib/libmd.so.1
1107 link path=usr/lib/libmd5.so target=../../lib/libmd5.so.1
1108 link path=usr/lib/libmd5.so.1 target=../../lib/libmd5.so.1
1109 link path=usr/lib/libmenu.so target=./libmenu.so.1
1110 link path=usr/lib/libmp.so target=../../lib/libmp.so.2
1111 link path=usr/lib/libmp.so.1 target=../../lib/libmp.so.1
1112 link path=usr/lib/libmp.so.2 target=../../lib/libmp.so.2
1113 link path=usr/lib/libmtmalloc.so target=./libmtmalloc.so.1
1114 link path=usr/lib/libnls.so target=./libnls.so.1
1115 link path=usr/lib/libnsl.so target=../../lib/libnsl.so.1
1116 link path=usr/lib/libnsl.so.1 target=../../lib/libnsl.so.1
1117 link path=usr/lib/libnvpair.so target=../../lib/libnvpair.so.1
```



```
1118 link path=usr/lib/libnvpair.so.1 target=../../lib/libnvpair.so.1
1119 link path=usr/lib/libpam.so target=../../lib/libpam.so.1
1120 link path=usr/lib/libpam.so.1 target=../../lib/libpam.so.1
1121 link path=usr/lib/libpanel.so target=./libpanel.so.1
1122 link path=usr/lib/libpkcs11.so target=./libpkcs11.so.1
1123 link path=usr/lib/libposix4.so target=../../lib/librt.so.1
1124 link path=usr/lib/libposix4.so.1 target=../../lib/librt.so.1
1125 link path=usr/lib/libproc.so target=../../lib/libproc.so.1
1126 link path=usr/lib/libproc.so.1 target=../../lib/libproc.so.1
1127 link path=usr/lib/libproject.so target=./libproject.so.1
1128 link path=usr/lib/libpthread.so target=../../lib/libpthread.so.1
1129 link path=usr/lib/libpthread.so.1 target=../../lib/libpthread.so.1
1130 link path=usr/lib/librcm.so target=../../lib/librcm.so.1
1131 link path=usr/lib/librcm.so.1 target=../../lib/librcm.so.1
1132 link path=usr/lib/libreparse.so target=./libreparse.so.1
1133 link path=usr/lib/libresolv.so target=../../lib/libresolv.so.2
1134 link path=usr/lib/libresolv.so.1 target=../../lib/libresolv.so.1
1135 link path=usr/lib/libresolv.so.2 target=../../lib/libresolv.so.2
1136 link path=usr/lib/librestart.so.1 target=../../lib/librestart.so.1
1137 link path=usr/lib/librpcsvc.so target=../../lib/librpcsvc.so.1
1138 link path=usr/lib/librpcsvc.so.1 target=../../lib/librpcsvc.so.1
1139 link path=usr/lib/librt.so target=../../lib/librt.so.1
1140 link path=usr/lib/librt.so.1 target=../../lib/librt.so.1
1141 link path=usr/lib/librtld.so.1 target=../../lib/librtld.so.1
1142 link path=usr/lib/librtld_db.so target=../../lib/librtld_db.so.1
1143 link path=usr/lib/librtld_db.so.1 target=../../lib/librtld_db.so.1
1144 link path=usr/lib/libscf.so target=../../lib/libscf.so.1
1145 link path=usr/lib/libscf.so.1 target=../../lib/libscf.so.1
1146 link path=usr/lib/libsched.so target=./libsched.so.1
1147 link path=usr/lib/libsctp.so target=./libsctp.so.1
1148 link path=usr/lib/libsec.so target=../../lib/libsec.so.1
1149 link path=usr/lib/libsec.so.1 target=../../lib/libsec.so.1
1150 link path=usr/lib/libsecdb.so target=../../lib/libsecdb.so.1
1151 link path=usr/lib/libsecdb.so.1 target=../../lib/libsecdb.so.1
1152 link path=usr/lib/libsendfile.so target=../../lib/libsendfile.so.1
1153 link path=usr/lib/libsendfile.so.1 target=../../lib/libsendfile.so.1
1154 link path=usr/lib/libsip.so target=./libsip.so.1
1155 link path=usr/lib/libsldap.so target=libsldap.so.1
1156 link path=usr/lib/libsmbios.so target=libsmbios.so.1
1157 link path=usr/lib/libsocket.so target=../../lib/libsocket.so.1
1158 link path=usr/lib/libsocket.so.1 target=../../lib/libsocket.so.1
1159 link path=usr/lib/libsoftcrypto.so target=./libsoftcrypto.so.1
1160 link path=usr/lib/libsys.so target=./libsys.so.1
1161 link path=usr/lib/libsysevent.so target=../../lib/libsysevent.so.1
1162 link path=usr/lib/libsysevent.so.1 target=../../lib/libsysevent.so.1
1163 link path=usr/lib/libtermcap.so target=../../lib/libtermcap.so.1
1164 link path=usr/lib/libtermcap.so.1 target=../../lib/libtermcap.so.1
1165 link path=usr/lib/libtermlib.so target=../../lib/libcurses.so.1
1166 link path=usr/lib/libtermlib.so.1 target=../../lib/libcurses.so.1
1167 link path=usr/lib/libthread.so target=../../lib/libthread.so.1
1168 link path=usr/lib/libthread.so.1 target=../../lib/libthread.so.1
1169 link path=usr/lib/libthread_db.so target=../../lib/libc_db.so.1
1170 link path=usr/lib/libthread_db.so.1 target=../../lib/libc_db.so.1
1171 link path=usr/lib/libtsnet.so target=../../lib/libtsnet.so.1
1172 link path=usr/lib/libtsnet.so.1 target=../../lib/libtsnet.so.1
1173 link path=usr/lib/libtsol.so target=../../lib/libtsol.so.2
1174 link path=usr/lib/libtsol.so.2 target=../../lib/libtsol.so.2
1175 link path=usr/lib/libumem.so target=../../lib/libumem.so.1
1176 link path=usr/lib/libumem.so.1 target=../../lib/libumem.so.1
1177 link path=usr/lib/libuuid.so target=../../lib/libuuid.so.1
1178 link path=usr/lib/libuuid.so.1 target=../../lib/libuuid.so.1
1179 link path=usr/lib/libuutil.so.1 target=../../lib/libuutil.so.1
1180 $(sparc_ONLY)link path=usr/lib/libv12n.so target=./libv12n.so.1
1181 link path=usr/lib/libvolmgt.so target=./libvolmgt.so.1
1182 link path=usr/lib/libw.so target=../../lib/libw.so.1
1183 link path=usr/lib/libw.so.1 target=../../lib/libw.so.1
```

```
link path=usr/lib/libwrap.so target=libwrap.so.1.0
link path=usr/lib/libwrap.so.1 target=libwrap.so.1.0
link path=usr/lib/libxnet.so target=../../lib/libxnet.so.1
link path=usr/lib/libxnet.so.1 target=../../lib/libxnet.so.1
link path=usr/lib/liby.so target=./liby.so.1
link path=usr/lib/libzoneinfo.so target=./libzoneinfo.so.1
link path=usr/lib/lwp/$(ARCH64)/libthread.so.1 \
    target=../../$(ARCH64)/libthread.so.1
link path=usr/lib/lwp/$(ARCH64)/libthread_db.so.1 \
    target=../../$(ARCH64)/libthread_db.so.1
link path=usr/lib/lwp/32 target=.
link path=usr/lib/lwp/64 target=$(ARCH64)
link path=usr/lib/lwp/libthread.so.1 target=../libthread.so.1
link path=usr/lib/lwp/libthread_db.so.1 target=../libthread_db.so.1
link path=usr/lib/nss_compat.so.1 target=../../lib/nss_compat.so.1
link path=usr/lib/nss_dns.so.1 target=../../lib/nss_dns.so.1
link path=usr/lib/nss_files.so.1 target=../../lib/nss_files.so.1
link path=usr/lib/nss_nis.so.1 target=../../lib/nss_nis.so.1
link path=usr/lib/nss_user.so.1 target=../../lib/nss_user.so.1
link path=usr/lib/scsi/$(ARCH64)/libscsi.so target=./libscsi.so.1
link path=usr/lib/scsi/$(ARCH64)/libses.so target=./libses.so.1
link path=usr/lib/scsi/$(ARCH64)/libsmp.so target=./libsmp.so.1
link path=usr/lib/scsi/libscsi.so target=./libscsi.so.1
link path=usr/lib/scsi/libses.so target=./libses.so.1
link path=usr/lib/scsi/libsmp.so target=./libsmp.so.1
link path=usr/lib/security/$(ARCH64)/crypt_bsdbf.so target=./crypt_bsdbf.so.1
link path=usr/lib/security/$(ARCH64)/crypt_bsdmd5.so \
    target=./crypt_bsdmd5.so.1
link path=usr/lib/security/$(ARCH64)/crypt_sha256.so \
    target=./crypt_sha256.so.1
link path=usr/lib/security/$(ARCH64)/crypt_sha512.so \
    target=./crypt_sha512.so.1
link path=usr/lib/security/$(ARCH64)/crypt_sunmd5.so \
    target=./crypt_sunmd5.so.1
link path=usr/lib/security/$(ARCH64)/pam_allow.so target=./pam_allow.so.1
link path=usr/lib/security/$(ARCH64)/pam_authtok_check.so \
    target=./pam_authtok_check.so.1
link path=usr/lib/security/$(ARCH64)/pam_authtok_get.so \
    target=./pam_authtok_get.so.1
link path=usr/lib/security/$(ARCH64)/pam_authtok_store.so \
    target=./pam_authtok_store.so.1
link path=usr/lib/security/$(ARCH64)/pam_deny.so target=./pam_deny.so.1
link path=usr/lib/security/$(ARCH64)/pam_dhkeys.so target=./pam_dhkeys.so.1
link path=usr/lib/security/$(ARCH64)/pam_dial_auth.so \
    target=./pam_dial_auth.so.1
link path=usr/lib/security/$(ARCH64)/pam_ldap.so target=./pam_ldap.so.1
link path=usr/lib/security/$(ARCH64)/pam_list.so target=./pam_list.so.1
link path=usr/lib/security/$(ARCH64)/pam_passwd_auth.so \
    target=./pam_passwd_auth.so.1
link path=usr/lib/security/$(ARCH64)/pam_rhosts_auth.so \
    target=./pam_rhosts_auth.so.1
link path=usr/lib/security/$(ARCH64)/pam_roles.so target=./pam_roles.so.1
link path=usr/lib/security/$(ARCH64)/pam_sample.so target=./pam_sample.so.1
link path=usr/lib/security/$(ARCH64)/pam_tsol_account.so \
    target=./pam_tsol_account.so.1
link path=usr/lib/security/$(ARCH64)/pam_unix_account.so \
    target=./pam_unix_account.so.1
link path=usr/lib/security/$(ARCH64)/pam_unix_auth.so \
    target=./pam_unix_auth.so.1
link path=usr/lib/security/$(ARCH64)/pam_unix_cred.so \
    target=./pam_unix_cred.so.1
link path=usr/lib/security/$(ARCH64)/pam_unix_session.so \
    target=./pam_unix_session.so.1
link path=usr/lib/security/$(ARCH64)/pkcs11_kernel.so \
    target=./pkcs11_kernel.so.1
link path=usr/lib/security/$(ARCH64)/pkcs11_softtoken.so \
```



```
    target=./pkcs11_softtoken.so.1
link path=usr/lib/security/$(ARCH64)/pkcs11_tpm.so target=./pkcs11_tpm.so.1
link path=usr/lib/security/64 target=$(ARCH64)
link path=usr/lib/security/audit_binfile.so target=./audit_binfile.so.1
link path=usr/lib/security/audit_remote.so target=./audit_remote.so.1
link path=usr/lib/security/audit_syslog.so target=./audit_syslog.so.1
link path=usr/lib/security/crypt_bsdbf.so target=./crypt_bsdbf.so.1
link path=usr/lib/security/crypt_bsdmd5.so target=./crypt_bsdmd5.so.1
link path=usr/lib/security/crypt_sha256.so target=./crypt_sha256.so.1
link path=usr/lib/security/crypt_sha512.so target=./crypt_sha512.so.1
link path=usr/lib/security/crypt_sunmd5.so target=./crypt_sunmd5.so.1
link path=usr/lib/security/pam_allow.so target=./pam_allow.so.1
link path=usr/lib/security/pam_authtok_check.so \
    target=./pam_authtok_check.so.1
link path=usr/lib/security/pam_authtok_get.so target=./pam_authtok_get.so.1
link path=usr/lib/security/pam_authtok_store.so \
    target=./pam_authtok_store.so.1
link path=usr/lib/security/pam_deny.so target=./pam_deny.so.1
link path=usr/lib/security/pam_dhkeys.so target=./pam_dhkeys.so.1
link path=usr/lib/security/pam_dial_auth.so target=./pam_dial_auth.so.1
link path=usr/lib/security/pam_ldap.so target=./pam_ldap.so.1
link path=usr/lib/security/pam_list.so target=./pam_list.so.1
link path=usr/lib/security/pam_passwd_auth.so target=./pam_passwd_auth.so.1
link path=usr/lib/security/pam_rhosts_auth.so target=./pam_rhosts_auth.so.1
link path=usr/lib/security/pam_roles.so target=./pam_roles.so.1
link path=usr/lib/security/pam_sample.so target=./pam_sample.so.1
link path=usr/lib/security/pam_tsol_account.so target=./pam_tsol_account.so.1
link path=usr/lib/security/pam_unix_account.so target=./pam_unix_account.so.1
link path=usr/lib/security/pam_unix_auth.so target=./pam_unix_auth.so.1
link path=usr/lib/security/pam_unix_cred.so target=./pam_unix_cred.so.1
link path=usr/lib/security/pam_unix_session.so target=./pam_unix_session.so.1
link path=usr/lib/security/pkcs11_kernel.so target=./pkcs11_kernel.so.1
link path=usr/lib/security/pkcs11_softtoken.so target=./pkcs11_softtoken.so.1
link path=usr/lib/security/pkcs11_tpm.so target=./pkcs11_tpm.so.1
link path=usr/lib/straddr.so target=./straddr.so.2
link path=usr/xpg4/lib/$(ARCH64)/libcurses.so target=libcurses.so.2
link path=usr/xpg4/lib/64 target=$(ARCH64)
link path=usr/xpg4/lib/libcurses.so target=./libcurses.so.2
#
# libses.so needs to dlopen(3C) plugins from usr/lib/scsi/plugins/ses/vendor/,
# a dependency which cannot be automatically derived
#
depend fmri=system/library/storage/scsi-plugins type=require
```

```
**********************************************************
    1720 Wed Aug 13 19:53:43 2014
new/usr/src/stand/lib/wanboot/Makefile
4853 illumos-gate is not lint-clean when built with openssl 1.0
**********************************************************
#
# CDDL HEADER START
#
# The contents of this file are subject to the terms of the
# Common Development and Distribution License (the "License").
# You may not use this file except in compliance with the License.
#
# You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
# or http://www.opensolaris.org/os/licensing.
# See the License for the specific language governing permissions
# and limitations under the License.
#
# When distributing Covered Code, include this CDDL HEADER in each
# file and include the License file at usr/src/OPENSOLARIS.LICENSE.
# If applicable, add the following below this CDDL HEADER, with the
# fields enclosed by brackets "[]" replaced with your own identifying
# information: Portions Copyright [yyyy] [name of copyright owner]
#
# CDDL HEADER END
#
# Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#

LIBRARY = libwanboot.a
LOCOBJS = http_aux.o bootinfo_aux.o
CMNOBJS = boot_http.o parseURL.o bootlog.o auxutil.o p12access.o \
          p12auxpars.o p12err.o p12misc.o http_errorstr.o bootconf.o bootinfo.o
OBJECTS = $(LOCOBJS) $(CMNOBJS)

include ../Makefile.com

CMNDIR  = $(CMNNETDIR)/wanboot
SRCS    = $(LOCOBJS:%.o=$(SRCDIR)/%.c) $(CMNOBJS:%.o=$(CMNDIR)/%.c)
LDLIBS += -lsunw_crypto -lsock -linet -lsunw_ssl -lnvpair
LDLIBS += -lcrypto -lsock -linet -lssl -lnvpair

CPPFLAGS += -I$(CMNNETDIR)/dhcp -I$(TOPDIR)/common/net/wanboot/crypt \
        -I../inet $(DHCPCPPFLAGS) $(SOCKCPPFLAGS)

#
# several objects need access to openssl headers, now in ../openssl
#
CPPFLAGS += -I..

CERRWARN += -_gcc=-Wno-char-subscripts
CERRWARN += -_gcc=-Wno-switch
CERRWARN += -_gcc=-Wno-parentheses
CERRWARN += -_gcc=-Wno-uninitialized
CERRWARN += -_gcc=-Wno-unused-value

include ../Makefile.targ
```